

Hanbit  
RealTime  
**134**

구글 컴퓨터 엔진 시작하기

# 빠르게 훑어보는 구글 클라우드 플랫폼

조대협, 최명근, 최유석, 윤성재, 김영균 지음



구글 컴퓨트 엔진 시작하기

# 빠르게 훑어보는 구글 클라우드 플랫폼

조대협, 최명근, 최유석, 윤성재, 김영균 지음





#### 표지 사진 이관행

이 책의 표지는 이관행님이 보내 주신 풍경사진을 담았습니다.

리얼타임은 독자의 시선을 담은 풍경사진을 책 표지로 보여주고자 합니다.

사진 보내기 [ebookwriter@hanbit.co.kr](mailto:ebookwriter@hanbit.co.kr)

## 빠르게 훑어보는 구글 클라우드 플랫폼 구글 컴퓨트 엔진 시작하기

초판발행 2016년 8월 23일

지은이 조대협 · 최명근 · 최유석 · 윤성재 · 김영균 / 펴낸이 김태현

펴낸곳 한빛미디어(주) / 주소 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부

전화 02-325-5544 / 팩스 02-336-7124

등록 1999년 9월 30일 제10-1779호

ISBN 978-89-6848-839-9 95000 / 비매품

총괄 전태호 / 책임편집 김창수 / 기획·편집 이복연

디자인 표지/내지 강은영, 초판 이경숙

마케팅 박상용, 송경석, 변지영 / 영업 김형진, 김진불, 조유미

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.

한빛미디어 홈페이지 [www.hanbit.co.kr](http://www.hanbit.co.kr) / 이메일 [ask@hanbit.co.kr](mailto:ask@hanbit.co.kr)

Published by HANBIT Media, Inc. Printed in Korea

Copyright © 2016 구글 클라우드 사용자 그룹 & HANBIT Media, Inc.

이 책의 저작권은 구글 클라우드 사용자 그룹과 한빛미디어(주)에 있습니다.

저작권법에 의해 보호를 받는 저작물이므로 무단 복제 및 무단 전재를 금합니다.

지금 하지 않으면 할 수 없는 일이 있습니다.

책으로 떠내고 싶은 아이디어나 원고를 메일([ebookwriter@hanbit.co.kr](mailto:ebookwriter@hanbit.co.kr))로 보내주세요.

한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

## 저자 소개

### 지은이\_ 조대협

구글 코리아의 클라우드 엔지니어다. 스타트업 개발자로 시작해서 BEA와 오라클에서 기술지원 엔지니어로, 마이크로소프트와 삼성전자 무선사업부에서 아키텍트로 근무하였고, 미디어 스타트업인 피카캐스트에서 CTO를 지냈다.『대용량 아키텍처와 성능 튜닝』과 『소프트웨어 개발과 테스트』(이상 프리렉, 2015)의 저자다. 기술 블로그인 조대협의 블로그 <http://bcho.tistory.com>을 운영하고 있다.

### 지은이\_ 최명근

구글 클라우드 플랫폼팀에서 아시아 지역 세일즈 엔지니어링을 책임지고 있다. 자바 개발자로 시작하여 마이크로소프트에서 기업 비즈니스 애플리케이션, 그중에서도 익스체인지 서버 전문 기술 엔지니어로 근무했다. MBA를 마친 후에는 구글에 입사해 현 역할을 맡기 전까지 기업용 구글 앱스 및 검색 서비스를 담당했다.

### 지은이\_ 최유석

나무기술(주)의 클라우드 엔지니어다.

### **지은이\_ 윤성재**

(주)락플레이스 클라우드팀의 솔루션 아키텍트다. 1990년대 하이텔, 나우누리 리눅스 동호회를 통해 초기 리눅스 활성화를 위해 노력했으며, 리눅스 벤처기업에서 기술 지원팀을 이끌었다. 통신사에서 다수의 서비스와 DB를 운영하면서 다양한 경험과 노하우를 쌓았다. 최근에는 구글 클라우드 플랫폼과 PostgreSQL의 매력에 빠져 지내고 있으며, 65세까지 리눅스 엔지니어로 일하고 싶어 하는 1인이다.

### **지은이\_ 김영균**

영우디지탈 클라우드 사업부에서 컨설팅을 담당하고 있다. 인하대학교 산업공학과 학사와 석사를 마쳤고 서울과학종합대학원대학교에서 산업보안 MBA를 마쳤다. 2001년 대상정보기술의 U-Biz 기술연구소에 입사하여 IT를 시작하게 되었다. 3년간 텔 인터내셔널의 인프라 컨설팅 서비스팀에서 솔루션 아키텍트를 담당하면서 클라우드에 입문했다.

## 저자의 말

구글에 들어오기 전에는 구글 클라우드를 볼 기회가 없었다. 막상 기술을 이해하고 나니 너무나 좋은 기능이 많은데, 한글 자료가 부족하여 초기 학습에 시간이 많이 소요되었다. 아마도 같은 경험을 한 분들이 있을 것 같아서 간단한 내용이지만 빠르게 구글 클라우드를 사용해볼 수 있도록 가이드 문서를 작성해봤다.

### 조대협

구글 클라우드 플랫폼을 처음 접하는 분에게 적합한 한글 가이드라인이 없다는 게 아쉬웠는데, 이번 기회를 통하여 조금이나마 기본적인 개념들을 이해할 수 있게 도울 수 있어 기쁘다. 앞으로 더 심화된 내용으로 꾸민 시리즈를 기대해본다.

### 최명근

클라우드란 더 이상 미래가 아닌 현재다. 거의 모든 산업에서 필수적인 요소로 자리 잡아가고 있다. 수많은 클라우드 서비스 업체가 있지만, 그 중에서 글로벌한 규모와 독보적인 기술력으로 사용자의 관점에서 서비스하는 업체는 구글이 유일하다고 생각한다. 아직 많이 미흡하지만, 구글 클라우드를 처음 사용하는 독자에게 조금이나마 도움이 되었으면 좋겠다.

### 최유석

클라우드 서비스의 춘추 전국 시대라고 해도 과언이 아닐 정도로 많은 클라우드 서비스들이 생겨나고 있다. 하지만 서비스마다 사용법이 다르고 기존 IDC에서의 아키텍처 구성과는 많은 부분이 다르기에 클라우드를 처음 접하는 이들에게는 많은 부분이

생소하고 어려울 수 있을 것이다. 내가 클라우드 서비스를 이용하기 위한 교육을 처음 받았을 때가 생각난다. 모든 문서는 영어였고 국내에는 도움을 청할 사람은 없었으며, 업체에서의 지원은 미비하여 약 3개월을 혼자 밤새워가면서 서비스 아키텍처를 설계하고 구현해야 했다.

구글이 자체적으로 사용하던 서비스를 구글 클라우드 플랫폼이라는 이름으로 서비스를 소개한 지 불과 몇 년 지나지 않은 거 같은데, Kubernetes를 기반으로 하는 Container Engine이나 자체 네트워크를 이용한 데이터센터 연동 등, 다른 클라우드 서비스에는 없는 구글만의 장점으로 클라우드 서비스를 만들어 나가고 있다. 이 모습을 보면서 지난 해부터 클라우드를 처음 접하는 사람을 위한 책을 써보고 싶었는데, 마침 이런 책을 만들 수 있는 기회가 주어져 매우 기쁜 마음으로 참여하게 되었다.

윤성재

구글 클라우드 플랫폼의 사용자로서 구글 클라우드에 입문하시고자 하는 사람들을 지면을 통해 만나게 되어 반갑다. 클라우드가 많은 관심을 받고 있지만 국내에서는 아직 낯설은 면이 많이 있다. 이 책을 통해 구글 클라우드를 시작하시는 사람들에게 조금이나마 도움이 될 수 있기를 희망한다. 구글 클라우드를 사용하는 데 어려움이 있다면 언제든지 연락 바라고, 작성된 글에 대한 피드백을 주면 반영하여 계속 보완해보도록 하겠다.

김영균

이 책은 구글 클라우드를 처음 사용하는 사람이 가상 머신(VM) 기반의 클라우드 서비스 기능들을 빠르게 사용할 수 있도록 도울 목적으로 실습 위주로 속도감 있게 구성하였다.

구글 클라우드의 가상 머신 서비스인 컴퓨터 엔진을 사용하기 위해 가입부터 VM 설정, 네트워크 설정, 오토 스케일링과 모니터링, 그리고 MySQL 매지니드 서비스인 Cloud SQL의 사용법을 소개하고, 구글 클라우드만의 장점인 빅데이터 서비스, 전 세계를 덮는 네트워크망, 합리적인 가격 정책을 소개한다.

chapter 1 구글 클라우드 소개 —— 015

1.1	빅데이터와 머신 러닝 서비스	—— 016
1.1.1	빅데이터 분석 플랫폼	—— 017
1.1.2	머신 러닝 플랫폼	—— 020
1.2	구글 전용 네트워크를 이용한 글로벌 커버리지	—— 022
1.3	저렴한 가격 모델	—— 023

chapter 2 구글 클라우드에서 VM 생성하기 —— 025

2.1	계정 가입	—— 025
2.2	프로젝트 생성	—— 026
2.3	VM 생성	—— 027
2.4	브라우저를 통한 SSH 접속	—— 030
2.5	아파치 웹 서버 설치	—— 031
2.6	VM 삭제	—— 032

chapter 3 SSH로 VM에 접속하기 —— 035

3.1	SSH 키 페어 생성	—— 036
3.1.1	맥 OS(터미널)	—— 036
3.1.2	윈도우(PuTTY)	—— 037
3.2	공개 키 등록	—— 039
3.3	SSH 접속 확인	—— 040
3.3.1	맥 OS(터미널)	—— 040
3.3.2	윈도우(PuTTY)	—— 040



## chapter 4 클라우드 로드 밸런서 달기 —— 043

4.1 주요 기능과 특징 ——	043
4.2 비관리 그룹과 관리 그룹 ——	044
4.3 VM 인스턴스 준비 ——	045
4.4 VM 인스턴스 디스크 스냅샷 추출 ——	047
4.5 인스턴스 복제 ——	047
4.6 접속 확인 ——	050
4.7 인스턴스 그룹 생성 ——	050
4.8 고정 IP 주소 할당 ——	052
4.9 로드 밸런서 생성 ——	052
4.9.1 프로토콜 선택 ——	053
4.9.2 백엔드 구성 ——	054
4.9.3 호스트 및 경로 규칙 설정 ——	058
4.9.4 프런트엔드 구성 ——	059
4.10 테스트 ——	060

## chapter 5 오토 스케일링 적용하기 —— 061

5.1 인스턴스 템플릿 정의 ——	061
5.2 오토 스케일링 그룹 생성 ——	064
5.3 로드 밸런서 설정 ——	066
5.4 테스트 ——	066
5.5 몇 가지 추가 내용 ——	068
5.5.1 오토 스케일링 조건 ——	068
5.5.2 자동 복구 ——	069

**chapter 6 서브 네트워크와 NAT 네트워크 구성하기 —— 071**

6.1 네트워크 구성요소 ——	072
6.1.1 서브 네트워크 ——	072
6.1.2 네트워크 종류 ——	072
6.1.3 NAT 게이트웨이 ——	073
6.1.4 내부 IP와 외부 IP ——	073
6.1.5 경로 ——	074
6.1.6 방화벽 규칙 ——	074
6.1.7 네트워크 부하 분산(TCP/UDP 기반) ——	075
6.1.8 지역과 영역 ——	075
6.2 네트워크 및 서버 구성 ——	076
6.2.1 프로젝트 생성 ——	076
6.2.2 네트워크 생성 ——	076
6.2.3 방화벽 규칙 생성 ——	078
6.2.4 VM 인스턴스 생성 ——	080
6.2.5 두 번째 VM 인스턴스 생성 ——	080
6.2.6 방화벽 규칙 추가 ——	081
6.2.7 네트워크 부하 분산 구성 ——	082
6.3 NAT 게이트웨이 구성 ——	086
6.3.1 외부 IP 제거 ——	086
6.3.2 NAT 게이트웨이 역할의 VM 생성 ——	087
6.3.3 경로 생성 ——	088
6.3.4 NAT 게이트웨이 설정 마무리 ——	089
6.3.5 NAT 게이트웨이 동작 테스트 ——	090
6.4 참고자료 ——	091



## chapter 7 IAM으로 사용자와 권한 관리하기 —— 093

7.1 구성원 유형 ——	093
7.2 사용자 역할 ——	094
7.3 IAM 설정 ——	094

## chapter 8 Cloud SQL에 접속하기 —— 099

8.1 1세대 Cloud SQL 연결 방식 ——	099
8.2 프록시를 이용한 2세대 Cloud SQL 연결 ——	101
8.2.1 Cloud SQL 생성 ——	101
8.2.2 사전 준비 ——	103
8.2.3 리눅스에서 접속하기 ——	107
8.2.4 맥에서 접속하기 ——	109
8.2.5 윈도우에서 접속하기 ——	111
8.2.6 애플리케이션에서 접속하기 ——	114

## chapter 9 Stackdriver로 클라우드 자원 모니터링하기 —— 117

9.1 Stackdriver 계정 생성 ——	117
9.2 모니터링 에이전트 설치 ——	119
9.2.1 윈도우에 에이전트 설치하기 ——	120
9.2.2 리눅스에 에이전트 설치하기 ——	120
9.3 Uptime Check와 경보 ——	122
9.4 대시보드와 모니터링 ——	127
9.4.1 대시보드 ——	127

9.4.2 클라우드 모니터링 ———	<b>128</b>
9.4.3 에이전트 모니터링 ———	<b>128</b>
9.5 로그 기록 ———	<b>128</b>

## chapter 10 구글 클라우드 스토리지 사용하기 ——— 131

10.1 저장소 클래스 ———	<b>131</b>
10.2 기본 개념 ———	<b>132</b>
10.3 버킷 생성 ———	<b>132</b>
10.4 버킷 간 파일 전송 ———	<b>133</b>
10.5 파일 업로드와 다운로드 ———	<b>135</b>
10.6 파일 공개 ———	<b>136</b>
10.7 서명된 URL을 통한 공유 ———	<b>137</b>
10.8 스트리밍 전송 ———	<b>139</b>
10.8.1 gsutil을 이용한 스트리밍 업로드와 다운로드 ———	<b>139</b>
10.8.2 boto를 이용한 스트리밍 업로드와 다운로드 ———	<b>139</b>
10.9 액세스 권한 관리 ———	<b>140</b>
10.10 스토리지 전송 서비스 ———	<b>142</b>



## chapter 11 구글 클라우드의 차별성 —— 143

11.1 빅쿼리, 빅데이터 저장 및 분석 플랫폼 ——	143
11.1.1 빅쿼리의 특징 ——	143
11.1.2 기존 빅데이터 플랫폼과 다른점 ——	145
11.1.3 빅쿼리 맛보기 ——	146
11.2 Cloud Vision API, 머신 러닝 서비스 ——	149
11.2.1 Cloud Vision API 맛보기 ——	149
11.2.2 다양한 이미지 분석 기능 ——	155
11.2.3 Cloud Vision API의 또 다른 의미 ——	156
11.3 합리적인 가격 정책 ——	156
11.3.1 분 단위 과금 ——	157
11.3.2 장기 계약 없이 알아서 깎아준다 ——	157
11.3.3 여러 인스턴스의 사용량을 자동으로 합산해서 깎아준다 ——	158
11.3.4 커스텀 인스턴스 ——	159
11.3.5 Preemptible VM ——	159
11.4 글로벌 광케이블을 이용한 네트워크 가속화 ——	160
11.4.1 실제 테스트 결과 ——	162

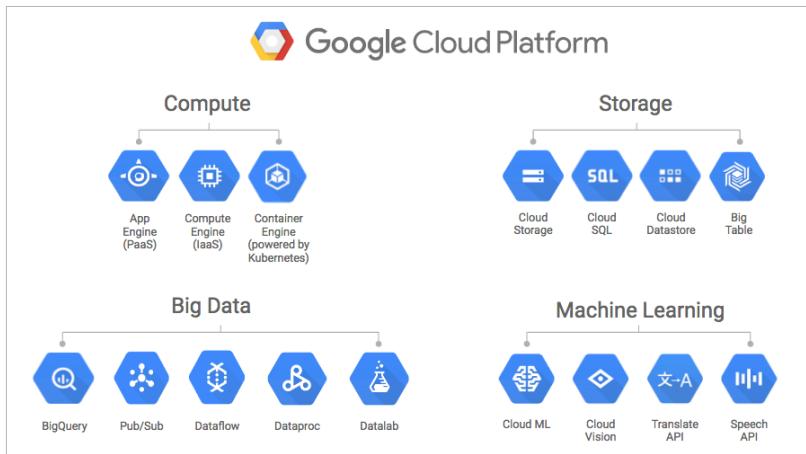


# 구글 클라우드 소개

이번 장에서는 구글 클라우드의 전반적인 모습을 간략히 소개하고 타 서비스와의 차별점을 간단하게 짚고 넘어간다.

구글 클라우드는 구글의 데이터센터 인프라를 기반으로 컴퓨터, 스토리지, 네트워킹, 빅데이터, 머신 러닝 등의 서비스를 제공하는 글로벌 클라우드다(그림 1-1). 현재 미국, 유럽, 아시아에 걸쳐서 서비스를 제공하고 있으며 계속해서 추가 데이터센터를 건립하고 있다.

그림 1-1 구글 클라우드 서비스 포트폴리오의 일부



컴퓨트 서비스는 VM 기반의 IaaS Infrastructure as a Service인 컴퓨트 엔진 Compute Engine, PaaS Platform as a Service인 앱 엔진 App Engine, 쿠버네티스 Kubernetes 기반의 Docker 런타임인

컨테이너 엔진<sup>Container Engine</sup> 등 사용자의 요구에 맞는 다양한 형태를 제공한다.

그 밖에 MySQL 서비스<sup>Google Cloud SQL</sup>, 대용량 파일을 저장하기 위한 오브젝트 스토리지 서비스<sup>Google Cloud Storage</sup> 등, 일반적인 IaaS 클라우드가 제공하는 내용은 다 있으니 별도로 소개하는 것은 큰 의미가 없을 것이다. 구글 클라우드만의 특징을 추려보자면 크게 다음의 3가지로 요약된다.

- 빅데이터와 머신 러닝 서비스
- 구글 전용 네트워크를 이용한 글로벌 커버리지
- 저렴한 가격 모델

이어지는 절들에서 각각의 특징을 자세히 알아보자.

## 1.1 빅데이터와 머신 러닝 서비스

구글은 지메일, 유튜브, 검색엔진 등 억 단위의 사용자를 보유한 대규모 서비스를 전 세계에 제공하는 회사다. 그만큼 빅데이터 처리에 관한 노하우를 많이 가지고 있고, 그 노하우를 바탕으로 한 서비스를 구글 클라우드 플랫폼을 통해서 제공하고 있다.

구글의 빅데이터 서비스는 크게 빅데이터와 머신 러닝으로 구별된다.

## 1.1.1 빅데이터 분석 플랫폼

빅데이터 플랫폼은 데이터를 분석하기 위한 데이터의 수집, 가공, 저장 기능을 제공하는 서비스로, 다음과 같은 제품들로 구성된다.

### 빅쿼리

빅쿼리BigQuery는 대규모 데이터 저장 및 분석 플랫폼으로, 일종의 데이터 웨어하우스라 생각하면 된다. 8,800개의 CPU와 3,600개의 디스크를 사용하는 대규모 인프라를 활용하여 1,000억 개의 레코드에 대한 질의를 30초 정도에 수행해주며, 가격도 저렴하다. 문법도 SQL과 유사하여 사용하기 매우 쉽다.

그림 1-2 빅쿼리 실행 화면

The screenshot shows the Google BigQuery web interface. At the top, there's a navigation bar with links for Mail, Calendar, Documents, Sites, Video, Groups, More, Settings, Help, and Sign out. Below the navigation is the BigQuery logo and a search bar labeled 'bigquery'. On the left, there's a sidebar with sections for 'COMPOSE QUERY', 'Query History', 'Job History', 'BigQuery demo' (which is expanded to show datasets like 'publicdata:samples' containing 'gsod', 'natality', 'shakespeare', 'trigrams', and 'wikipedia'), and a 'Compose Query' button. The main area is titled 'Compose Query' and contains a code editor with the following SQL query:

```
SELECT TOP(title, 20), COUNT(*) FROM publicdata:samples.wikipedia WHERE LOWER(title) CONTAINS 'google' AND wp_namespace=0;
```

Below the code editor is a red 'RUN QUERY' button. Underneath it is a 'Query Results' section showing the first 9 rows of a 20-row result set. The results are as follows:

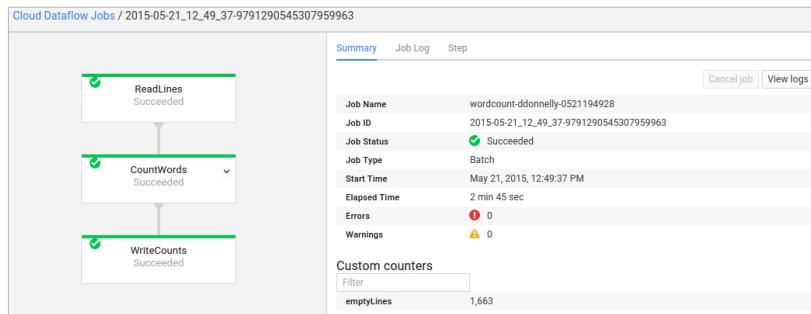
Row	f0_	f1_
1	Google	8755
2	Google search	4261
3	Google Earth	3874
4	Google Chrome	2687
5	Google Maps	2617
6	Google bomb	2345
7	Google Street View	2294
8	List of Google products	1984
9	Google's hoaxes	1258

At the bottom right of the results table are 'Download as CSV' and 'Save as Table' buttons.

### 데이터플로

데이터플로Dataflow는 아파치 스파크Spark나 플링크Flink와 같이 실시간 스트리밍 분석 및 배치 분석을 지원하는 플랫폼으로, 오픈 소스 프레임워크인 아파치 범Beam에 기초하고 있다. 수집한 데이터를 변환하거나 여러 데이터 소스와 저장소 간의 연결(라우팅)을 담당한다.

그림 1-3 데이터플로 실행 화면



## 펍/섭 큐

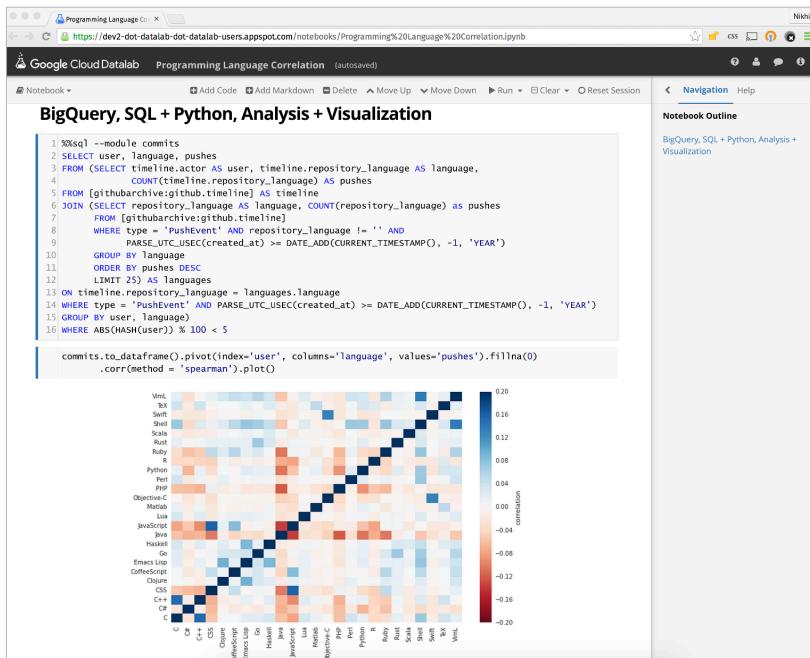
펍/섭Pub/Sub은 카프카Kafka와 같은 대규모 큐잉 시스템으로, 데이터를 대규모로 수집하는 역할을 한다.

## 클라우드 데이터랩

데이터랩Datalab은 데이터 과학자나 엔지니어가 여러 가지 데이터 소스에 접속해서 MS 워드와 같은 환경에서 작업 내용을 저장하고, 구글 빅쿼리 등과 연결해서 바로 질의를 수행할 수 있는 웹 기반의 저작 도구다. 마치 위키처럼 마크업을 이용해 데이터 작업을 할 수 있다. 오픈소스인 IPython Notebook<sup>01</sup>의 구글 클라우드 버전이라 생각하면 된다.

01 Jupyter로 이름이 변경됨. <https://ipython.org/notebook.html>

## 그림 1-4 클라우드 데이터랩 실행 화면



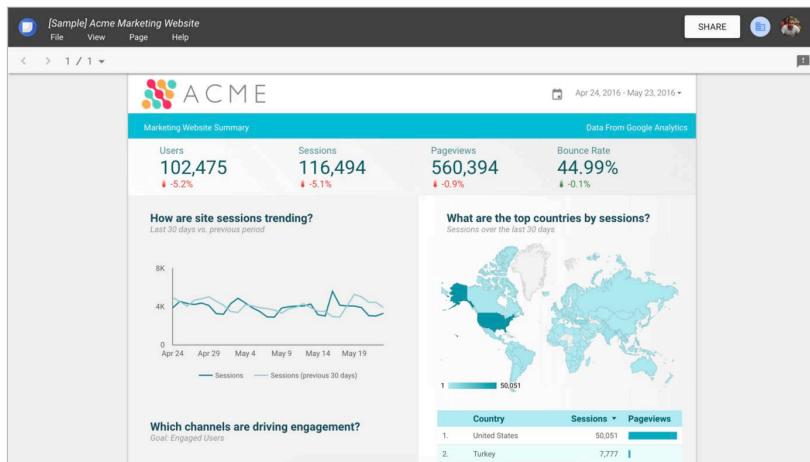
## 데이터프록

데이터프록 DataProc은 오픈소스 빅데이터 플랫폼인 하둡 Hadoop과 스파크의 매니저드 서비스로, 클러스터를 90초 안에 배포할 수 있고 분당 과금을 지원한다. 그 덕분에 빠르고 편리하게 작업하면서 사용한 만큼만 정확하게 비용을 지불하여 비용 낭비를 줄여준다.

## 데이터 스튜디오

데이터 스튜디오 Data Studio는 SQL이나 빅쿼리 등의 데이터 소스를 기반으로 시각적인 보고서를 생성해주는 제품이다. 비지오나 파워포인트처럼 간단하게 드래그-앤-드롭만으로 다음 그림과 같이 시각적인 보고서를 생성해낼 수 있다.

그림 1-5 데이터 스튜디오로 생성한 보고서



### 1.1.2 머신 러닝 플랫폼

빅데이터 분석 플랫폼과 더불어 구글 클라우드는 머신 러닝 기반의 인공 지능 기능들을 서비스로 제공한다. 크게 머신 러닝 API군과 텐서플로 기반의 머신 러닝 플랫폼을 제공한다.

#### 머신 러닝 API들

음성을 인식하여 텍스트로 변환해주는 스피치Speech API, 80여 개 나라의 언어를 읽어서 번역해주는 번역Translate API, 그리고 사진 이미지에서 사람의 표정과 사물을 인식해주는 비전Vision API 등을 제공한다. 이러한 API 서비스는 머신 러닝 개념을 모르더라도 약 20~30줄의 코드만으로 머신 러닝 기능을 바로 사용할 수 있도록 해준다.

그림 1-6 구글 비전 API로 사람들의 표정을 분석한 결과

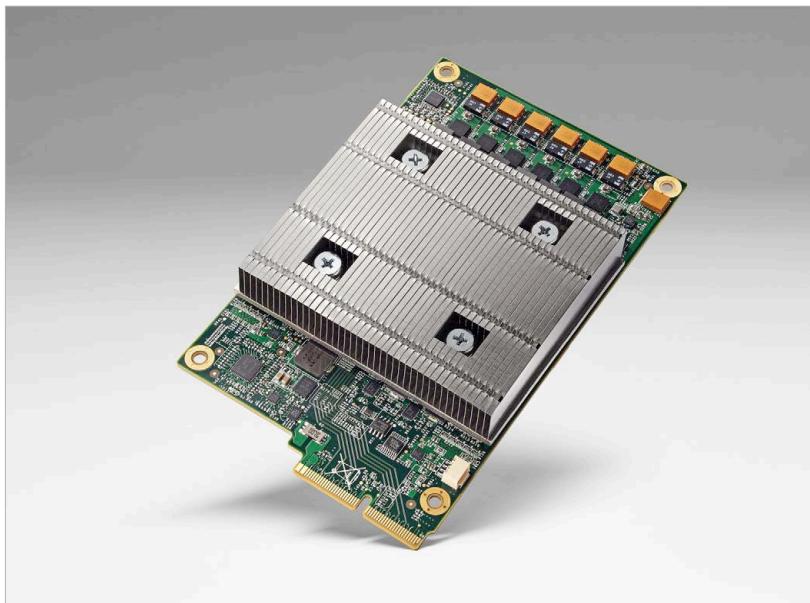


### 텐서플로(Cloud Machine Learning) 서비스

텐서플로Tensorflow는 얼마 전 이세돌 9단과 바둑 대결을 한 알파고에 탑재된 딥러닝 프레임워크다. 이 프레임워크는 오픈소스로 공개되었고, 구글 클라우드에서는 이 텐서플로를 CloudML<sup>Cloud Machine Learning</sup>이라는 이름의 클라우드 서비스로 제공한다. 이 서비스는 구글이 자체 개발한 텐서플로 전용 CPU인 TPU<sup>Tensor Processing Unit</sup>를 이용하여 실행된다.

앞서의 머신 러닝 API가 특별한 배경 지식이 없는 사람도 특정 시나리오에서 사용하기 쉽게 API화 해놓은 것이라면, CloudML은 자신만의 머신 러닝 서비스를 개발하고자 하는 사람에게 더 강력하고 대용량의 컴퓨팅 파워를 제공하는 서비스다.

그림 1-7 구글의 텐서플로 전용 CPU인 텐서 프로세싱 유닛(TPU)



## 1.2 구글 전용 네트워크를 이용한 글로벌 커버리지

구글 클라우드 기능 중에서 잘 알려지지 않은 것 하나가 전 세계에 깔린 구글 광케이블망을 이용하여 네트워크를 가속하는 기능이다.

구글은 전 세계에 70개 이상의 데이터센터를 가지고 있으며, 미국, 유럽, 아시아에 배포된 클라우드 데이터센터와 연결되어 있다.

그림 1-8 전 세계를 덮는 구글 전용 네트워크



구글 클라우드 데이터센터로 접속하려 하면 지정한 데이터센터로 바로 접속하지 않을 수도 있다. 대신 클라이언트에서 가장 가까운 센터로 접속한 후에, 데이터센터 사이를 연결하는 구글 전용망을 통해서 목표 데이터센터로 연결된다. 예를 들어 한국에서 미국의 구글 데이터센터로 연결하려 할 경우, 가까운 일본에 있는 중계용 데이터센터로 접속한 후, 일본에서부터 미국까지는 구글 전용 광케이블을 통해서 접속한다. 이러한 메커니즘 덕분에 세계 곳곳의 클라이언트들에게 다른 클라우드 서비스보다 훨씬 빠른 네트워크 접속 시간을 제공한다. 별도의 추가 비용도 없다.

### 1.3 저렴한 가격 모델

구글 클라우드는 타 클라우드보다 가격 정책이 합리적이다. 예를 들어 시간당 과금이 아니라 분당 과금을 한다든지, 연 단위 약정이 없더라도 월 단위로 사용량에 따라서 알아서 할인해주고, 이 할인 역시 개별 인스턴스 단위가 아니라 전체 인스턴스 사용량을 모아서 할인해주기 때문에, 더 큰 폭의 할인을 받을 수 있다.

아울러, 미리 정해진 인스턴스 종류를 선택할 수도 있지만, 사용자가 직접 필요한 CPU와 메모리양을 지정하여 자원 낭비 없이 딱 필요한 만큼만 사용할 수 있다.

빨리 실습으로 넘어가기 위해 우선 이 정도로 마무리하고, 각 차별화 포인트에 대한 자세한 내용은 11장에서 다시 설명하도록 하겠다.

# 구글 클라우드에서 VM 생성하기

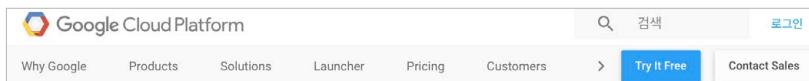
조대협

클라우드를 이용할 때 가장 기본이 되는 VM 인스턴스 생성 방법을 알아보자. 이번 장에서는 구글 클라우드에 가입하고, VM을 생성한 후, 그 VM에 웹 서버를 설치하여 잘 접속되는지까지 확인할 것이다.

## 2.1 계정 가입

구글 클라우드 플랫폼을 사용하려면 구글 계정이 있어야 한다. 기존에 Gmail 계정이 있으면 그 계정을 사용하면 된다. <http://www.google.com/cloud>로 가서 오른쪽 위의 [Try It Free] 버튼을 눌러서 구글 클라우드에 가입한다(개인 정보와 결제 정보를 입력해야 한다).

그림 2-1 구글 클라우드 플랫폼의 첫 화면



무료 평가의 경우 60일간 \$300를 사용할 수 있는 크레딧이 자동으로 등록되며, 사용자가 명시적으로 상품을 업그레이드하지 않으면 과금이 발생하지 않기 때문에 걱정하지 않아도 된다.

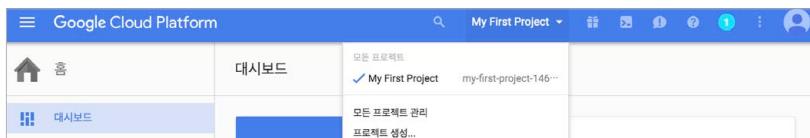
## 2.2 프로젝트 생성

계정을 생성했으면 이제 프로젝트를 생성할 차례다.

프로젝트는 VM, 네트워크 자원, SQL 등의 클라우드 자원을 묶어서 관리하는 논리적인 집합이다. 여러 사람이 하나의 클라우드를 사용할 때 이렇게 프로젝트를 만들어서 프로젝트별로 과금하거나 시스템이나 팀별로 프로젝트를 나눠서 정의하면 관리하기가 쉽다.

구글 클라우드 플랫폼 콘솔<sup>GCP</sup> 화면 오른쪽 위의 프로젝트 선택 메뉴에서 [프로젝트 생성]을 클릭해보자.

그림 2-2 GCP 콘솔의 상단 메뉴. 프로젝트 선택 메뉴를 확인할 수 있다.



그러면 프로젝트 이름을 입력하는 창이 나온다. 원하는 이름을 입력하고 [만들기] 버튼을 클릭하면 프로젝트가 만들어진다.

그림 2-3 새로운 프로젝트 생성

새 프로젝트

프로젝트 ID [?](#)  
terrycho-sandbox

프로젝트 ID는 terrycho-sandbox-1469952572199입니다. [수정](#)

고급 옵션 보기...

[만들기](#) [취소](#)

## 2.3 VM 생성

프로젝트를 만들었으면 이제 VM을 만들어보자. GCP 콘솔의 왼쪽 위의 부분을 클릭하면 구글 클라우드 플랫폼이 제공하는 제품과 서비스를 선택하는 메뉴가 나타난다.

그림 2-4 “제품 및 서비스” 선택 메뉴의 위치



어떤 제품들이 있는지 쭉 둘러보고 [Compute Engine]을 선택한다. 그러면 왼쪽 메뉴가 바뀌는데 그중 [VM 인스턴스]를 선택한 후, [인스턴스 만들기]를 클릭해서 인스턴스를 생성해보자. VM 인스턴스는 다음과 같은 조건으로 생성할 것이다.

- 인스턴스 이름 : ubuntu-instance
- 영역 : asia-east1-a
- 머신 유형 : 초소형(공유 vCPU 1개)
- 부팅 디스크 : Ubuntu 14.04 LTS, 표준 영구 디스크 10GB
- ID 및 API 액세스 > 액세스 범위 : 모든 Cloud API에 대한 전체 액세스 허용
- 방화벽 : HTTP/HTTPS 트래픽 허용
- 네트워킹 > 외부 IP : 임시

VM 생성창에서 위의 조건에 따라 아래 화면과 같이 값을 설정한 후 VM을 생성한다. 나머지 값들은 기본값으로 놔두면 된다.

## 그림 2-5 인스턴스 만들기

← 인스턴스 만들기

이름 ?  
ubuntu-instance

영역 ?  
asia-east1-a

머신 유형

초소형(공유 vCPU 1개) ▾ 0.6GB 메모리 맞춤설정  
계정을 업그레이드하면 코어가 최대 32개인 인스턴스를 만들 수 있습니다.

부팅 디스크 ?  
새로운 10GB 표준 영구 디스크 이미지 Ubuntu 14.04 LTS 변경

ID 및 API 액세스 ?  
서비스 계정 ?  
Compute Engine default service account ▾  
액세스 범위 ?  
 기본 액세스 허용  
 모든 Cloud API에 대한 전체 액세스 허용  
 각 API에 액세스 설정

방화벽 ?  
태그 및 방화벽 규칙을 추가하여 인터넷에서 특정 네트워크 트래픽을 허용합니다.  
 HTTP 트래픽 허용  
 HTTPS 트래픽 허용

관리, 디스크, 네트워킹, SSH 키

다음 옵션이 변경되었습니다.

하위 네트워크

이 인스턴스에 무료 평가판 크레딧(사용할 수 있는 경우)이 사용됩니다.

**만들기** 취소

몇 가지 주의할 점이 있다. 먼저 영역<sup>zone</sup>은 VM이 생성되는 장소라고 생각하면 된다. 지역<sup>region</sup>이 가장 큰 개념으로 대륙을 정의한다. 미국 중부, 서부, 아시아, 유럽 등이 지역에 해당한다. 각 지역에는 여러 개의 데이터센터가 있는데, 이 데이터센터를 영역으로 생각하면 된다.

[ID 및 API 액세스] 항목에서 [액세스 범위]를 “모든 Cloud API에 대한 전체 액세스 허용”으로 설정해줘야 한다. 이 부분은 구글 클라우드가 제공하는 API나 Cloud SQL, BigQuery, Vision API 등의 다른 서비스를 호출할 수 있는 권한을 이 VM에 주는 것이다. 이 설정은 VM 생성 초기에 하지 않으면 나중에 변경할 수 없으니 반드시 확인하기 바란다.

마지막으로 [네트워킹] 항목의 [외부 IP]를 “임시”로 설정한다.<sup>01</sup> 외부 IP는 지정하지 않거나(없음), 임시, 고정 주소, 이렇게 3가지를 선택할 수 있다. 지정하지 않을 경우 이 VM에는 외부 IP가 주어지지 않기 때문에 인터넷에서 이 VM에 직접 접근할 수 없다. HTTP 요청의 경우 별도의 프록시나 로드 밸런서를 앞에 놓고 접근하거나 SSH와 같은 셸 접근도 같은 내부 네트워크 대역에 있는 다른 VM을 통해서만 가능하다. 잘 알고 있겠지만, VM에 외부 IP 주소를 부여하지 않는 것은 보통 외부로부터 접근을 차단하여 보안성을 높이려는 의도다.

[만들기] 버튼을 클릭해 인스턴스를 생성하면 화면이 바뀌고, 잠시 기다리면 [그림 2-6]과 같이 생성된 인스턴스가 가동됨을 확인할 수 있다.

---

<sup>01</sup> [네트워킹] 항목은 [그림 2-5]의 맨 아래 “관리, 디스크, 네트워킹, SSH 키”를 클릭하면 나타난다.

그림 2-6 가동 중인 인스턴스의 모습

The screenshot shows the Google Cloud Platform's VM Instances page. At the top, there are navigation links: 'VM 인스턴스' (VM Instances), '인스턴스 만들기' (Create Instance), '인스턴스 그룹 만들기' (Create Instance Group), '재설정' (Reset), '시작' (Start), '중지' (Stop), and '삭제' (Delete). Below the header, there is a search bar labeled '라벨 또는 이름으로 필터링' (Filter by label or name) and a dropdown menu for time periods: '1시간' (1 hour), '6시간' (6 hours), '12시간' (12 hours), '1일' (1 day), '2일' (2 days), '4일' (4 days), '7일' (7 days), and '30일' (30 days). A '라벨' (Label) button is also present. The main content area displays a table of instances. The columns are: '이름' (Name), '영역' (Region), '머신 유형' (Machine Type), '권장사항' (Recommendations), '다음에서 사용 중' (Used in next), '내부 IP' (Internal IP), '외부 IP' (External IP), and '연결' (Connect). One instance is listed: 'ubuntu-instance' (checked), 'asia-east1-a', 'vCPU 1개, 0.6GB', '10.140.0.2', '104.199.180.200' (with a copy icon), 'SSH' (with a more options icon). A message at the bottom of the chart area says '이 차트에 데이터가 없습니다.' (No data available for this chart).

## 2.4 브라우저를 통한 SSH 접속

인스턴스가 생성되었으면 이제 SSH로 접속해보자. PuTTY와 같은 전용 SSH 터미널을 이용하는 방법은 다음 장에서 설명하도록 하겠다. 가장 간단한 방법은 해당 VM 오른쪽의 [SSH] 링크를 클릭하는 것이다([그림 2-6]의 오른쪽 아래).

그러면 별도의 SSH 터미널 없이도 [그림 2-7]과 같이 웹 브라우저상에서 바로 SSH 터미널을 열어준다.

그림 2-7 웹 브라우저가 바로 SSH 터미널이 된다.

The screenshot shows a web browser window with the URL <https://ssh.cloud.google.com/projects/terrycho-sandbox-1469952572199/zones/asia-east1-a/instances/ubuntu-instance?authuser=1>. The page content is a terminal session for an Ubuntu 14.04.4 LTS VM. The session includes system information, package updates, and a copyright notice. The terminal prompt at the bottom is `wogra_lee@ubuntu-instance:~$`.

```
wogra_lee@ubuntu-instance:~  
  ↳ https://ssh.cloud.google.com/projects/terrycho-sandbox-1469952572199/zones/asia-east1-a/instances/ubuntu-instance?authuse...  
connected, host fingerprint: ssh-rsa 2048 99:CB:7B:BD:0C:81:1A:17:DA:5B:26:8C:CD:DF:F4:05  
elcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.19.0-64-generic x86_64)  
      [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  [●]  
* Documentation: https://help.ubuntu.com/  
System information as of Sun Jul 31 10:16:48 UTC 2016  
System load: 0.08      Memory usage: 9%      Processes: 54  
Usage of /: 10.1% of 9.81GB  Swap usage: 0%  Users logged in: 0  
Graph this data and manage this system at:  
  https://landscape.canonical.com/  
Get cloud support with Ubuntu Advantage Cloud Guest:  
  http://www.ubuntu.com/business/services/cloud  
packages can be updated.  
updates are security updates.  
  
he programs included with the Ubuntu system are free software;  
he exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*copyright.  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
wogra_lee@ubuntu-instance:~$
```

## 2.5 아파치 웹 서버 설치

이제 새로 만든 VM에 아파치 웹 서버를 설치하고 가동해보자. 설치에는 우분투의 apt-get 유ти리티를 사용할 것이다.

먼저 다음 명령을 실행해 apt-get 저장소를 최신으로 업데이트한다.

---

```
$ sudo apt-get update
```

---

다음으로 apache2 웹 서버를 설치해보자.

---

```
$ sudo apt-get install apache2
```

---

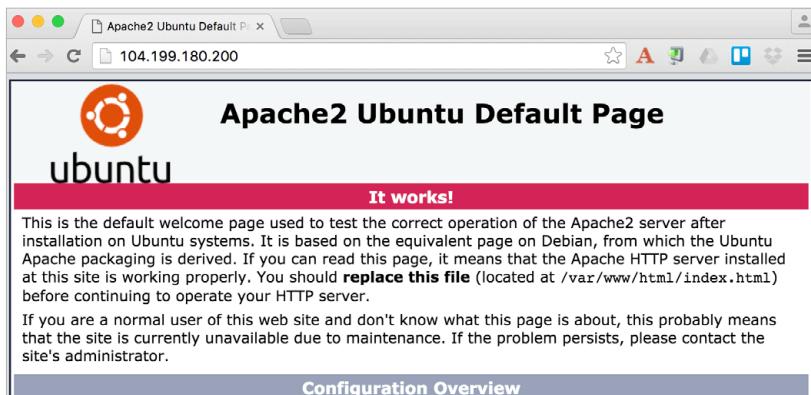
설치가 끝나면 자동으로 아파치 웹 서버가 가동된다. 가동이 되었는지를 확인하려면 브라우저에서 이 VM의 외부 IP로 접속해보자.<sup>02</sup> 접속에 성공하면 다음과 같은

---

02 외부 IP는 [그림 2-6]의 오른쪽 아래에서 확인할 수 있다.

페이지가 브라우저에 뜨는 것을 확인할 수 있다.

그림 2-8 아파치 웹 서버의 시작 화면



혹시 웹 서버가 자동으로 가동되지 않았다면 다음 명령을 내려 직접 가동하자.

---

```
$ sudo apachectl start
```

---

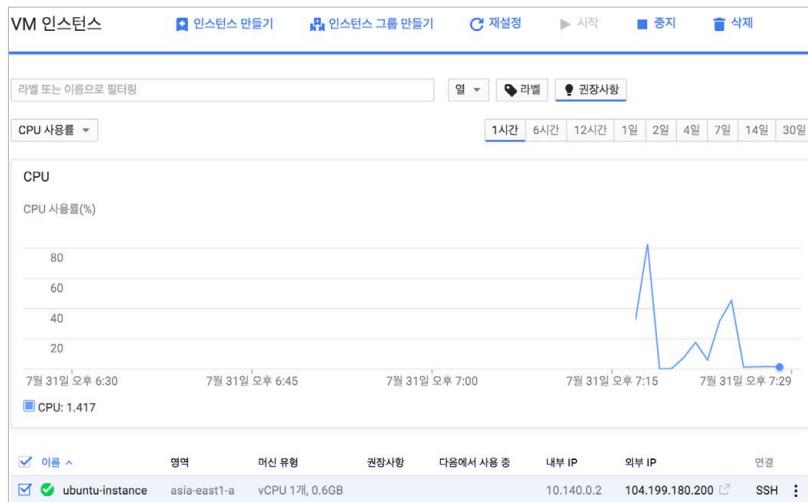
## 2.6 VM 삭제

서비스를 계속 제공하지 않을 것이면 VM을 중지하거나 삭제하면 된다. 중지와 삭제의 차이는 간단하다. 중지된 VM은 나중에 재시작할 수 있다. 단, 스토리지 비용은 계속 부과된다. 삭제는 VM을 없애버리는 것으로, 재시작할 수 없고 당연히 아무 비용도 부과되지 않는다.

VM을 중지하거나 삭제하려면 [그림 2-9]에서 보는 것처럼 아래쪽 인스턴스 목록

에서 원하는 VM을 선택한다. 그러면 오른쪽 위의 [중지]와 [삭제] 메뉴가 활성화 된다. 그런 다음 원하는 메뉴를 선택해주면 된다.

그림 2-9 인스턴스 중지 혹은 삭제하기





# SSH로 VM에 접속하기

조대협, 윤성재

클라우드에 생성한 VM에 접속할 때는 일반적으로 SSH를 이용한다. 구글 클라우드의 VM에 SSH로 접속하는 방법은 다음과 같이 크게 4가지다.

1. 브라우저 창에서 열기<sup>01</sup>
2. 맞춤 포트의 브라우저 창에서 열기
3. gcloud 명령어로 접속하기
4. 다른 전용 SSH 클라이언트를 사용하여 접속하기

이중 가장 쉬운 방법은 1번일 것이다. 웹 브라우저에서 그냥 클릭만 하면 바로 접속된다. 인스턴스 한 대에만 접속한다면 분명 가장 쉽고 편하다. 그다음으로 쉬운 방법은 3번, gcloud 명령어로 접속하는 것이다. 특히나 맥이나 리눅스 사용자에게는 무척이나 쉽고 편하게 강력한 기능들을 제공해줄 것이다. 하지만 많은 이들이 윈도우를 사용하고 있고, 윈도우에서 서버로 접속할 때는 주로 SecureCRT나 PuTTY 같은 전용 SSH 클라이언트 프로그램을 사용한다.

이번 장에서는 맥과 윈도우에서 SSH를 이용하여 클라우드의 VM에 접속하는 방법을 알아보도록 한다.

---

01 '2.4 브라우저를 통한 SSH 접속' 절에서 선보인 방법이다.

### 3.1 SSH 키 페어 생성

만약 구글 클라우드 플랫폼 프로젝트에서 승인된 공개 키가 없다면, 새로운 키 페어<sup>key-pair</sup>를 생성하고 프로젝트에 적용할 수 있다.

SSH나 SCP로 VM에 접속하기 전에 프로젝트에서 사용할 SSH 키 페어를 생성해야만 한다. 혹 gcloud 도구로 VM에 접속하고 있다면 키 페어는 이미 생성되어 있고, 다음의 위치에서 확인할 수 있다.

- 리눅스와 맥 OS
  - 공개 키 : \$HOME/.ssh/google\_compute\_engine.pub
  - 개인 키 : \$HOME/.ssh/google\_compute\_engine
- 윈도우
  - 공개 키 : C:\Users\[USER\_NAME]\.ssh\google\_compute\_engine.pub
  - 개인 키 : C:\Users\[USER\_NAME]\.ssh\google\_compute\_engine

아직 키 페어가 없다면 이어지는 하위 절의 설명을 따라 새로 생성하면 된다(gcloud를 설치했다면 앞의 위치에서 키 페어를 확인했을 것이고, 이번 절은 건너뛰어도 된다).

#### NOTE gcloud 명령줄 도구

구글 클라우드 플랫폼을 계속 사용할 계획이라면 gcloud는 반드시 필요한 도구이므로 꼭 설치하길 바란다. <https://cloud.google.com/sdk/downloads>에서 내려받아 설치할 수 있다.

#### 3.1.1 맥 OS(터미널)

먼저 터미널을 열고 다음과 같이 ssh-keygen 명령어로 RSA 키 페어를 생성한다.

---

```
$ ssh-keygen -t rsa -C "구글계정명"
```

---

이 명령을 실행하면 개인 키를 저장할 파일명과 패스프레이즈 passphrase를 묻는다. 적절히 입력하면 생성한 개인 키와 공개 키의 위치를 알려준다. 다음 그림에서는 /Users/terrycho/.ssh/terrycho-gcp-key.pub 파일에 공개 키가 담겨 있다. 이 파일의 내용은 텍스트이며 이 내용을 복사하여 3.2절에서 등록하면 된다.

그림 3-1 ssh-keygen으로 키 페어를 생성한 모습

```
terrycho-macbookpro:~ terrycho$ ssh-keygen -t rsa -C "terrycho@google.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/terrycho/.ssh/id_rsa): /Users/terrycho/.ssh/terrycho-gcp-key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/terrycho/.ssh/terrycho-gcp-key,
Your public key has been saved in /Users/terrycho/.ssh/terrycho-gcp-key.pub.
The key fingerprint is:
35:1e:c0:c4:42:d1:38:73:0d:81:4e:3b:f6:5b:c7 terrycho@google.com
The key's randomart image is:
+--[ RSA 2048]----+
| ..*oBo
| = o =
| o = o +
| = o o
| . o .S .
| . . E
| o .
|
+-----+
terrycho-macbookpro:~ terrycho$
```

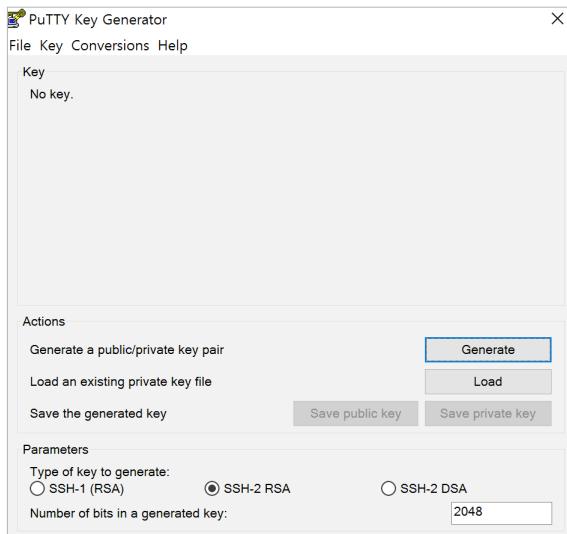
### 3.1.2 원도우(PuTTY)

먼저 다음 위치에서 puttygen.exe를 내려받는다.

- <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

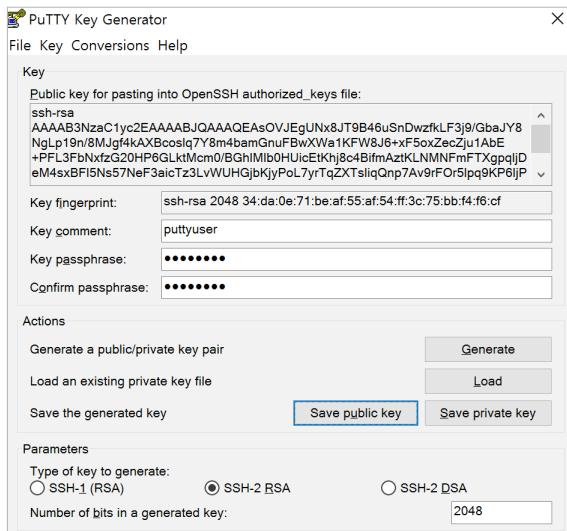
이 파일은 별다른 설치가 필요 없으니 내려받은 파일을 실행하고 [Generate] 버튼을 클릭한다.

그림 3-2 새로운 키 페어를 생성한다.



새로운 키 페어가 생성되면 [Key comment] 항목에는 구글 계정 이름을(여기서는 “puttyuser”를 사용했다), [Key passphrase]와 [Confirm passphrase]에는 암호를 입력한다(암호는 선택사항이다).

그림 3-3 키 페어가 생성되었다.



이제 [Save public key]와 [Save private key] 버튼을 클릭해 공개 키와 개인 키를 파일로 저장한다(아직은 창을 닫지 말자). 생성된 공개 키의 내용(텍스트)은 [Public key for pasting into OpenSSH authorized\_keys file:]에서도 바로 확인할 수 있다.

## 3.2 공개 키 등록

키 페어를 생성했으면, 그중 공개 키를 구글 클라우드 플랫폼에 등록해야 한다. GCP 콘솔에서 [Compute Engine] > [메타데이터]로 이동한 후 [SSH 키] 탭에서 [수정] 버튼을 눌러 앞서 생성한 공개 키의 내용(텍스트)을 [키] 입력란에 붙여 넣어 키를 등록한다.

그림 3-4 공개 키 내용을 붙여넣어 키를 등록한다.

The screenshot shows the Google Cloud Platform interface for managing metadata on a Compute Engine instance. The top navigation bar includes the GCP logo, project name 'terrycho-sandbox', and a search icon. The main menu on the left lists 'Compute Engine' (selected), 'VM 인스턴스', '인스턴스 그룹', '인스턴스 템플릿', '디스크', '스냅샷', '이미지', and '메타데이터'. The 'Compute Engine' section has a sub-menu for 'Metadata' and 'SSH 키'. A large button labeled '수정' (Edit) is prominent. Below it, a note states: '이 프로젝트의 모든 인스턴스는 이러한 SSH 키를 상속받습니다.' with a link '자세히 알아보기'. The 'SSH 키' table lists one entry: 'wegrade.lee' with the value 'ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQD1zXDCoujyzEg1JNz2wiydV...'. At the bottom of the table, there's a '동등한 REST' (Equivalent REST) link.

이 방식으로 등록한 키로는 모든 인스턴스에 접속할 수 있게 되지만, 인스턴스를 생성할 때 [SSH 키] 항목에 등록하면 그 키로는 해당 인스턴스에만 접속할 수 있다.

### 3.3 SSH 접속 확인

호스트 주소는 GCP 콘솔의 [Compute Engine] > [VM 인스턴스]에 가면 확인 할 수 있는 외부 IP를 입력하면 된다([그림 2-6]의 오른쪽 아래).

역시 맥과 윈도우에서 접속하는 방법을 차례로 알아보자.

#### 3.3.1 맥 OS(터미널)

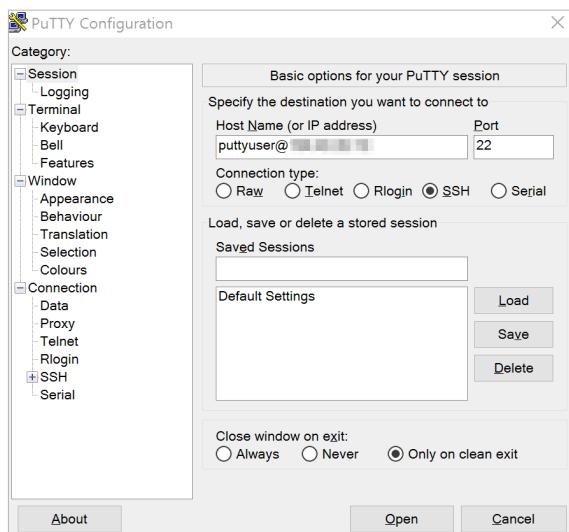
터미널을 열고 다음 명령을 실행한다.

```
$ ssh -i [개인 키 파일] 계정@호스트주소
```

#### 3.3.2 윈도우(PuTTY)

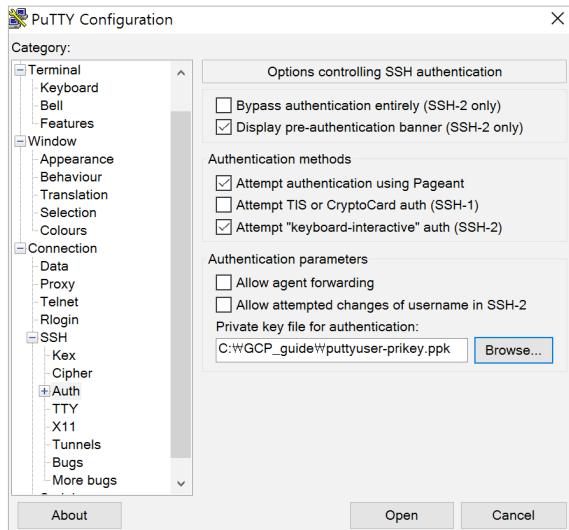
먼저 <http://www.putty.org/>에서 PuTTY를 내려받는다. PuTTY는 별도의 설치 없이 바로 실행할 수 있다. PuTTY를 실행한 후 [Session] 메뉴에서 [Host Name]에는 “puttyuser@[인스턴스\_외부\_IP\_주소]”를 입력한다.

그림 3-5 PuTTY로 VM 접속하기 1 – 호스트 이름 설정



다음으로 [Connection] > [SSH] > [Auth] 메뉴의 [Private key file for authentication]에서 [Browse…] 버튼을 클릭해서 앞서 3.1.2절에서 저장해 둔 개인 키 파일을 선택한다.

그림 3-6 PuTTY로 VM 접속하기 2 – 개인 키 파일 설정



이제 [Open]을 눌러 접속해보자. 보안 경고 화면이 나오면 “예(Y)”를 눌러주자. 다음으로, Passphrase for key “puttyuser”: 메시지가 나오면 키 페어를 생성 할 때 입력했던 암호를 넣어준다.

그림 3-7 PuTTY로 VM 접속하기 3 – 접속 성공

The screenshot shows a PuTTY terminal window with the following text output:

```
puttyuser@test-001: ~
Using username "puttyuser".
Authenticating with public key "puttyuser"
Passphrase for key "puttyuser":

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
puttyuser@test-001:~$
```

# 클라우드 로드 밸런서 달기

조대협

클라우드 VM을 생성하는 방법을 숙지하였으면, 다음으로 부하를 여러 VM에 분산해주는 **로드 밸런서**load balancer, 부하 분산기 기능을 알아보자.

## 4.1 주요 기능과 특징

다음은 구글 로드 밸런서가 제공하는 대표적인 기능이다.

- 일반적인 L4 스위치와 같이 TCP/UDP 프로토콜을 라우팅한다.
- HTTP 프로토콜에 대해서는 HTTPS Termination과 URI에 따라서 가까운 서버나 특정 서버로 라우팅하는 L7과 유사한 기능을 제공한다(자세한 내용은 <http://bcho.tistory.com/1111> 참고).
- 서버로 들어오는 트래픽을 (인터넷이 아니라) 구글이 전 세계에 배포한 에지 노드 중 가장 가까운 노드에 접속한 후, 구글의 광케이블 백본망으로 목표 서버로 전송한다. 이 기능은 네트워크 구간의 속도를 획기적으로 줄여준다(자세한 내용은 '11.4 글로벌 광케이블을 이용한 네트워크 가속화' 참고).

또한 다음과 같은 재미있는 특징을 지닌다.

- **프리웜업**pre-warm up이 필요 없다.

어떤 클라우드 로드 밸런서는 갑자기 높은 부하가 들어오면 그 부하를 받아내지 못하는 경우가 있다. 이를 방지하려면 미리 그만한 부하를 일정 시간 줘서 로드 밸런서의 크기를 키워주는 프리웜업 작업을 하거나 직접 클라우드 업체에 연락해서 용량을 증설해달라고

요청해야 한다. 그러나 이러한 방식은 예측하지 못한 부하가 들어왔을 때 대응하기 어렵다는 단점이 있고, 예측된 부하라 하더라도 프리워크이라는 개념을 모르는 사용자는 트래픽을 다 받아내지 못해서 운영상에 문제를 겪는 경우가 종종 있다. 반면 구글 클라우드의 로드 밸런서는 들어오는 부하에 맞춰서 프리워크도 없이 바로 규모를 늘려준다.

- 지역 수준의 부하 분산이 가능하다.

백엔드를 다수의 지역에 배포하였을 때, 구글의 클라우드 로드 밸런서는 단일 IP 주소로 지역 간의 로드밸런싱을 제공하다. 클라이언트에서 가까운 지역을 우선으로 라우팅하고, 만약 그 지역에 장애가 생겼다면 자동으로 다른 지역으로 라우팅한다<sup>fail-over</sup>. DNS 라운드Robin 같은 방식과 달리, 단일 정적 IP를 사용하기 때문에 별다른 설정이 필요 없고, 방화벽 규칙 같은 인프라 규칙을 설정하는 데에도 용이하다.

- 모든 트래픽을 로깅할 수 있다.

모든 부하를 로깅할 수 있다. 로드 밸런서는 보통 부하량이 많아서 로그를 남기지 않는 경우가 많은데, 외부로부터의 해킹 시도나 오류가 발생했을 때는 맨 앞단에 자리한 로드 밸런서의 액세스 로그가 중요하다. 구글 클라우드는 이 모든 기록을 로그로 남겨주고 API를 호출해 가져올 수 있어서 언제든 분석이 가능하다.

## 4.2 비관리 그룹과 관리 그룹

로드 밸런서 설정에 앞서 비관리 그룹<sup>unmanaged group</sup>과 관리 그룹<sup>managed group</sup> 개념을 이해해야 한다.

그룹은 인스턴스의 집합으로, 이 단위로 로드 밸런서에 연결할 수 있다. 그룹은 크게 비관리 그룹과 관리 그룹로 나뉜다.

- 비관리 그룹

인스턴스를 수동으로 묶어 놓은 그룹이다. 인스턴스의 크기나 종류에 상관 없이 그냥 묶어 만 놓은 것이다. 수동으로 아무 종류의 인스턴스나 추가해낼 수 있기 때문에 오토 스케일링은 불가능하다. 비관리 그룹을 이해하려면 반대로 관리 그룹 개념을 이해하면 쉽다.

- 관리 그룹

템플릿을 만들어 인스턴스 생성을 클라우드가 담당하게 한다. 인스턴스는 사용자가 직접 추가할 수 없다. 템플릿이란 인스턴스를 만들기 위한 틀과 같은 것이다. VM의 크기, OS

이미지 등의 설정을 미리 정해놓으면, 클라우드가 인스턴스를 생성할 때 해당 템플릿이 정의한 설정에 따라서 인스턴스를 추가하게 된다. 템플릿 생성 방법은 약간 내용이 길기 때문에 차후에 따로 설명하겠다.

대략적인 특징을 이해했으면, 로드 밸런서를 더한 인프라를 구성해보자. 관리 그룹은 템플릿 생성 등 절차가 더 복잡하니 나중에 따로 설명하기로 하고, 이번 장에서는 비관리 그룹을 기반으로 부하를 분산하는 구성을 갖춰보자.

## 4.3 VM 인스턴스 준비

먼저 부하를 받을 인스턴스를 구성한다. 간단한 Node.js 웹 애플리케이션을 구성하여 배포할 것이다. 이 애플리케이션은 “/”에 접속하면 해당 VM 인스턴스의 IP 정보 등을 반환하는 간단한 기능을 수행한다.

### 예제 4-1 간단한 Node.js 애플리케이션

```
var router = express.Router();
var os = require('os');
/* GET home page. */
router.get('/', function(req, res, next) {
    res.json(os.networkInterfaces());
});
module.exports = router;
```

코드가 준비되었으면 VM을 생성하자. VM 생성 조건은 다음과 같다.

- 이름 : lb-test-1
- 영역 : asia-east1-a
- 머신 유형 : 소형
- 부팅 디스크 : Ubuntu 14.04 LTS
- 액세스 범위 : 모든 Cloud API에 대한 전체 액세스 허용
- 방화벽 : HTTP/HTTPS 트래픽 허용

- 내부 IP : 자동
- 외부 IP : 임시
- 시작 스크립트 : [예제 4-2]의 셀 스크립트

이상의 조건을 ‘[2.3 VM 생성](#)’ 절을 참고해 기입한다(아직 생성하진 말자).

**NOTE**

여기서 외부 IP를 설정한 이유는 로드 밸런서가 제대로 작동함을 테스트하기 위해서다. 로드 밸런서를 거친 요청과 거치지 않은 요청을 각각 보내는 방식으로 동작을 확인할 것이다. 운영 환경에서 실제로 서비스할 때는 외부 IP 없이 내부 IP만 부여하여, 외부로부터의 직접 접근을 차단하고 로드 밸런서를 통한 HTTP 접근만 허용하는 것이 보안상 유리하다.

시작 스크립트는 VM이 가동되자마자 자동으로 수행되는 스크립트로, 입력 위치는 [관리] 탭에서 찾을 수 있다(그림 4-1).

**그림 4-1** 시작 스크립트 입력 위치



이 곳에 다음과 같은 셀 스크립트를 추가한다. 이 예제에서는 VM이 가동된 후 자동으로 Node.js 인스턴스를 구동하는 npm start를 수행하도록 하였다.

```
#! /bin/bash
export PORT=80
cd $HOME/restapi_mysql/
sudo -E npm start
```

이제 VM을 생성하고, VM에 Node.js 소스 코드를 배포한다.<sup>01</sup>

## 4.4 VM 인스턴스 디스크 스냅샷 추출

이번에는 앞 절에서 만든 인스턴스를 마스터로 새로운 인스턴스를 복제할 것이다. 이를 위해서는 디스크 내용을 복제해야 하므로 마스터 인스턴스에서 “스냅샷”을 떠놓도록 하자. [Compute Engine] > [스냅샷] 메뉴로 들어가면 디스크의 스냅샷을 뜰 수 있다.

[스냅샷 만들기]를 선택한 후 다음 값을 입력하고 [만들기] 버튼을 클릭하면 스냅샷이 만들어진다.

- 이름 : restapi-instance-snapshot
- 소스 디스크 : lb-test-1 (앞 절에서 생성한 마스터 인스턴스.)

## 4.5 인스턴스 복제

첫 번째 인스턴스를 생성하였으나, 이번에는 부하를 분산시킬 두 번째 인스턴스를 생성해보자. 두 번째 인스턴스는 첫 번째 인스턴스와 동일한 설정을 사용할 것이기 때문에 첫 번째 인스턴스를 복제해서 사용한다.

[VM 인스턴스] 화면에서 특정 인스턴스를 클릭하면 상세 정보 화면이 나타난다. 첫 번째 인스턴스의 상세 정보에 들어가 위쪽의 [복제]를 누르면 같은 설정의

<sup>01</sup> scp, ftp, 깃 저장소 등 본인 취향에 맞는 방법으로 복제하면 된다.

인스턴스를 생성할 수 있다.

그림 4-2 VM 인스턴스 복제하기



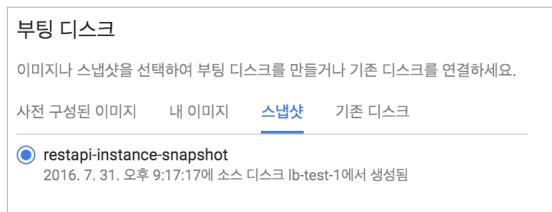
[복제]를 누르면 인스턴스 생성이 다음과 같이 동일 설정으로 자동 진행되는 것을 확인할 수 있다. 이름 뒤의 번호는 자동으로 1씩 커진다.

그림 4-3 마스터 인스턴스와 같은 설정으로 자동 진행된다.



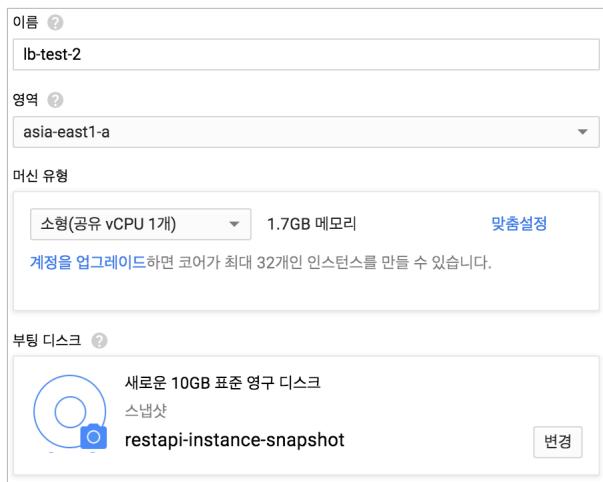
이때 부팅 디스크는 앞에서 생성한 'lb-test-1'과 동일한 이미지를 사용해야 한다. [변경] 버튼을 클릭한 후 [스냅샷] 탭으로 가면 앞서 생성한 "rest-instance-snapshot" 이미지가 보일 것이다. 이를 선택하고 화면 아래의 [선택] 버튼을 클릭한다.

**그림 4-4** 앞서 생성한 스냅샷이 보인다.



[부팅 디스크] 설정이 방금 선택한 스냅샷으로 바뀐 것을 확인할 수 있을 것이다 (그림 4-5).

**그림 4-5** [부팅 디스크]를 스냅샷으로부터 만든다.

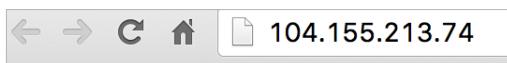


내/외부 IP가 각각 “자동”과 “임시”가 맞는지, 그리고 시작 스크립트도 같은지 확인하고(당연히 같은 것이다), [만들기] 버튼을 클릭해 두 번째 인스턴스를 생성한다.

## 4.6 접속 확인

로드 밸런서를 테스트할 lb-test-1과 lb-test-2 VM이 생성되었으면 정상적으로 웹 애플리케이션이 올라왔는지 확인해보자. 예제에서 lb-test-1과 lb-test-2에 할당된 외부/내부 IP 주소는 차례로 104.155.213.74/10.140.0.4와 107.167.186.59/10.140.0.5다. 각각의 외부 IP로 접속해보면 다음과 같이 자신의 내부 IP를 반환하는 것을 확인할 수 있다.

그림 4-6 lb-test-1의 웹 애플리케이션이 반환한 결과



```
{"lo": [{"address": "127.0.0.1", "netmask": "::1"}, {"address": "::1", "netmask": "ffff:ffff:ffff:ffff::1"}, {"address": "10.140.0.4", "netmask": "ffff:ffff:ffff:ffff::4"}, {"address": "fe80::4001:aff:fe8c:4"}]
```

그림 4-7 lb-test-2의 웹 애플리케이션이 반환한 결과



```
{"lo": [{"address": "127.0.0.1", "netmask": "::1"}, {"address": "::1", "netmask": "ffff:ffff:ffff:ffff::1"}, {"address": "10.140.0.5", "netmask": "ffff:ffff:ffff:ffff::5"}, {"address": "fe80::4001:aff:fe8c:5"}]
```

## 4.7 인스턴스 그룹 생성

로드 밸런서 테스트를 위한 VM 생성이 모두 끝났다. 이제 로드 밸런서에 이 인스턴스를 연결하기 위해서 인스턴스들을 그룹으로 묶어보자.

[Compute Engine] > [인스턴스 그룹] 메뉴에서 [인스턴스 만들기]를 선택한다. 생성 조건은 다음과 같다(명시하지 않은 조건은 기본값을 사용한다).

- 이름 : lb-test-group
- 영역 : asia-east1-a (인스턴스를 생성한 영역)
- 생성 방법 : 기존 인스턴스 선택

그리고 [VM 인스턴스]에서 앞에서 생성한 lb-test-1과 lb-test-2를 선택하여 추가한다.

**그림 4-8** 새로운 인스턴스 그룹 만들기

### ← 새 인스턴스 그룹 만들기

---

부하 분산 백엔드 서비스를 구성하거나 VM 인스턴스를 그룹화하려면 인스턴스 그룹을 사용하세요. [자세히 알아보기](#)

**이름** ?

**설명 (선택사항)**

**위치**

다중 영역 그룹은 여러 영역을 통해 제공되므로 가용성이 더 높습니다. [자세히 알아보기](#)

단일 영역  
 다중 영역

**영역** ?

asia-east1-a

**포트 이름 맵핑 지정 (선택사항)**

**생성 방법**

템플릿을 사용하여 자동으로 크기를 조절할 수 있는 동일한 인스턴스의 그룹을 만듭니다. 템플릿을 사용하지 않을 경우 각 구성원을 직접 추가하고 관리해야 합니다. [자세히 알아보기](#)

인스턴스 템플릿 사용  
 기존 인스턴스 선택

**하위 네트워크** ?

default

**VM 인스턴스**

lb-test-1 ×

lb-test-2 ×

**인스턴스 추가**

만들기
취소

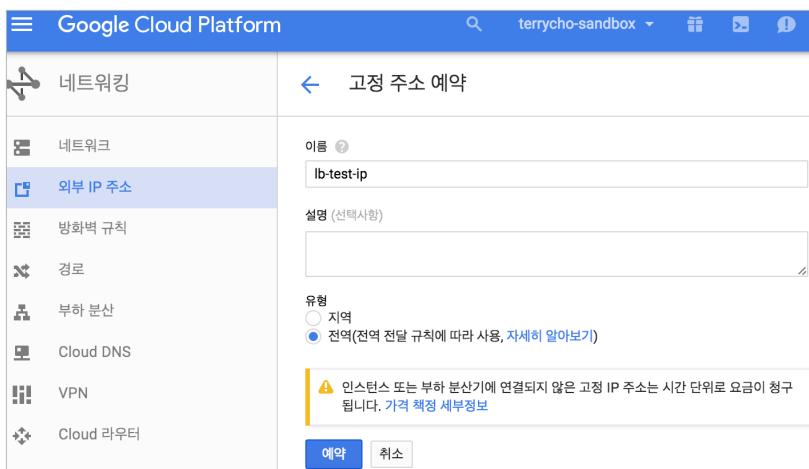
이제 로드 밸런서에 연결할 인스턴스 그룹이 생성되었다. 곧 로드 밸런서를 설정하고, 방금 생성한 그룹을 로드 밸런서에 연결해볼 것이다.

## 4.8 고정 IP 주소 할당

로드 밸런서 설정에 앞서 로드 밸런서에 할당할 고정 IP를 생성하자.

GCP 콘솔에서 [Compute Engine]이 아닌 [네트워킹] > [외부 IP 주소] 메뉴로 들어간 후, 우측 중간쯤에 [고정 주소 예약] 버튼을 클릭하자. 이름을 입력하고 [유형]은 “전역 Global”을 선택한다. 지역 Region IP는 특정 지역에만 부여할 수 있고 전역 IP는 여러 지역에 걸쳐 사용할 수 있다.

그림 4-9 고정 주소 예약하기



## 4.9 로드 밸런서 생성

이제 로드 밸런서를 설정하기 위한 모든 준비가 끝났다. 로드 밸런서를 생성하기 위해서 [네트워킹] > [부하 분산] 메뉴로 들어가자. 로드 밸런서는 크게 다음과

같은 단계로 생성하게 된다.

1. 프로토콜 선택
2. 백엔드 구성
3. 호스트 및 경로 규칙 설정
4. 프런트엔드 구성

각 단계를 자세히 알아보자.

#### 4.9.1 프로토콜 선택

로드밸런서 생성을 시작하면 제일 처음 프로토콜을 선택하게 된다. 지원하는 프로토콜은 HTTP(S), TCP, UDP 이렇게 3가지다. 이 예제에서는 HTTP(S) 프로토콜을 사용한다.

그림 4-10 로드밸런서 생성 1단계 – 프로토콜 선택하기

The screenshot shows the 'Protocol Selection' step of the load balancer creation wizard. At the top left is a back arrow labeled '부하 분산'. The main area is divided into three columns: 'HTTP(S) 부하 분산', 'TCP 부하 분산', and 'UDP 부하 분산'. Each column contains configuration options and a 'Configure Now' button.

선택	설명	선택
HTTP 및 HTTPS 애플리케이션용 Layer 7 부하 분산입니다. 다. <a href="#">자세히 알아보기</a>	TCP/SSL 프로토콜을 사용하는 애플리케이션용의 Layer 4 부하 분산 또는 프록시입니다. 자세히 알아보기	UDP 프로토콜을 사용하는 애플리케이션용의 Layer 4 부하 분산입니다. <a href="#">자세히 알아보기</a>
<b>구성</b> HTTP LB HTTPS LB	<b>구성</b> TCP LB SSL 프록시	<b>구성</b> UDP LB
<b>옵션</b> 인터넷 연결 전용 다중 지역 단일 지역	<b>옵션</b> 인터넷 연결 전용 단일 또는 다중 지역	<b>옵션</b> 인터넷 연결 전용 단일 지역

“HTTP(S) 부하 분산” 아래의 [구성 시작] 버튼을 클릭하면 [그림 4-11]과 같은 단계별 구성 화면이 나타난다.

그림 4-11 새 HTTP(S) 로드 밸런서 구성하기

새 HTTP(S) 부하 분산기	HTTP(S) 부하 분산기 만들기
<p>이름 <span>lb-test-http</span></p> <p>● 백엔드 구성 아직 백엔드를 구성하지 않았습니다.</p> <p>● 호스트 및 경로 규칙 요청이 기본 백엔드로 전달됩니다.</p> <p>● 프런트 엔드 구성 프런트 엔드를 구성했습니다.</p> <p>① 검토 및 완료 선택사항</p> <p><span>만들기</span> <span>취소</span></p>	<p>HTTP 또는 HTTPS 부하 분산기는 3가지 부분으로 구성됩니다.</p> <ol style="list-style-type: none"><li>1. 백엔드 구성 백엔드 서비스는 수신 트래픽을 인스턴스 그룹으로 전달합니다.</li><li>2. 호스트 및 경로 규칙 호스트 및 경로 규칙은 트래픽의 전달 방식을 결정합니다. 규칙을 지정 하지 않으면 트래픽이 기본 백엔드 서비스로 전달됩니다.</li><li>3. 프런트 엔드 구성 IP 주소, 프로토콜, 포트입니다. 이 IP가 클라이언트 요청이 수신되는 IP입니다.</li></ol>

## 4.9.2 백엔드 구성

이제 백엔드를 구성한다. 이때 기존의 백엔드 서비스를 사용하거나 새로 만들 수 있다. 물론 우리가 사용할 백엔드는 앞에서 생성한 인스턴스 그룹이 된다. [백엔드 구성] > [백엔드 서비스 만들기 또는 선택] > [백엔드 서비스 만들기]를 선택해보자. 이번 예제에서는 다음과 같은 조건으로 만들 것이다.

- 이름 : lb-test-backend
- 인스턴스 그룹 : lb-test-group (asia-east1-a) (앞서 생성한 인스턴스 그룹이다.)
- 포트 번호 : 80 (이 예제에서는 HTTP 트래픽을 전달할 것이다.)

그림 4-12 새로운 백엔드 서비스 생성하기



아래쪽에 보면 “상태 확인[Health check]”이라는 메뉴가 보인다. 상태 확인은 인스턴스 그룹 내의 인스턴스가 양호한지를 점검하고, 정상이 아닌 노드를 부하 분산에서 빼버리는 기능을 수행한다. 항목을 선택해서 새로운 상태 확인을 생성해보자.

[그림 4-13]에서는 이름만 “lb-test-healthcheck”로 입력하고 나머지는 그대로 두었다. [프로토콜]과 [포트]는 당연히 HTTP와 80이다. 그리고 서비스가 정상 인지를 확인하는 [요청 경로]는 /다.

그림 4-13 새로운 상태 확인 생성하기

상태 확인 생성

인스턴스 그룹 및 부하 분산 자동 복구는 상태 확인을 사용하여 인스턴스의 응답이 없을 때를 감지합니다. [자세히 알아보기](#)

이름 [?](#)

lb-test-healthcheck

설명 (선택사항)

프로토콜

HTTP

포트 [?](#)

80

요청 경로 [?](#)

/

▼ 더보기

상태 기준

상태가 어떻게 결정되는지 정의합니다. 확인 빈도, 응답 대기 시간, 성공 또는 실패 시도 횟수가 중요합니다.

간격 확인 <a href="#">?</a>	시간 초과 <a href="#">?</a>
5초	5초
정상 기준 <a href="#">?</a>	비정상 기준 <a href="#">?</a>
2연속 성공	2연속 실패

이렇게 입력하면 로드 밸런서는 백엔드 서비스 내의 인스턴스에 `http://xxx.xxx:80/`로 HTTP 요청을 보내고, HTTP 200 OK 응답이 오면 정상이라고 판단한다. 이 확인 주기를 [그림 4-13] 아래쪽의 [상태 기준] 항목에서 정의한다. [간격 확인]이 상태를 확인하는 주기고, [시간 초과]는 해당 시간 안에 응답이 없으면 장애라고 판단한다는 뜻이다.

여기서는 백엔드 서비스를 하나만 구성했지만 다음의 보충 설명에서 이야기하는 바와 같이 로드 밸런서 하나에 여러 백엔드 서비스를 구성하여 연결할 수도 있다.

#### NOTE 보충 설명

##### 인스턴스 그룹 간 부하 분산하기

로드 밸런서는 하나만 사용하면서, 서로 다른 인스턴스 그룹을 각각 다른 백엔드로 정의하여 인스턴스 그룹 간에 부하를 분산할 수도 있다.

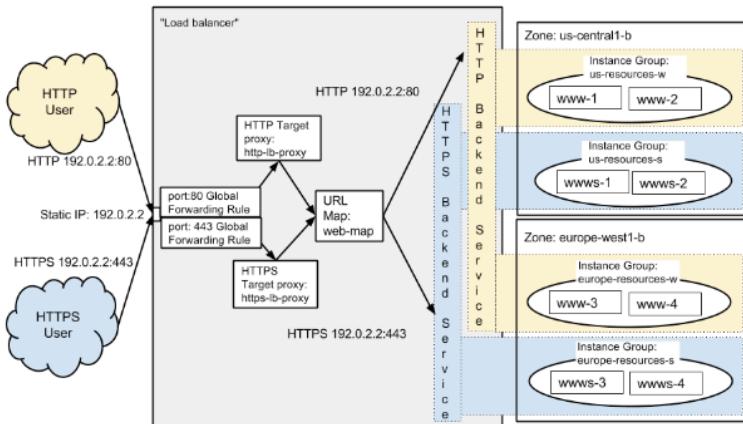
##### 둘 이상의 인스턴스 그룹을 이용한 롤링 업그레이드 구성

이 기능을 이용하면 인스턴스 그룹 A, B를 만들어서 두 그룹으로 부하를 분산하고, 서버 패치나 배포 시 A 그룹을 정지시키고 배포한 후 재가동하고, B 그룹을 정지시키고 배포한 후 재가동하는 롤링 업그레이드 방식으로 무정지 서비스 배포를 실현할 수 있다.

##### 지역 간(또는 영역 간) 라우팅 구성

또는 [그림 4-14]처럼 같은 기능의 백엔드를 서로 다른 지역에 배포할 수 있다. 이렇게 하면 지역 단위의 장애가 발생하였을 때 정상적인 지역으로 요청을 라우팅하여 지역 단위의 HA(고가용성) 구성도 가능해진다. 물론 같은 방식으로 영역 간의 라우팅을 통한 HA 구성도 가능하다.

그림 4-14 서로 다른 지역의 백엔드 간의 라우팅

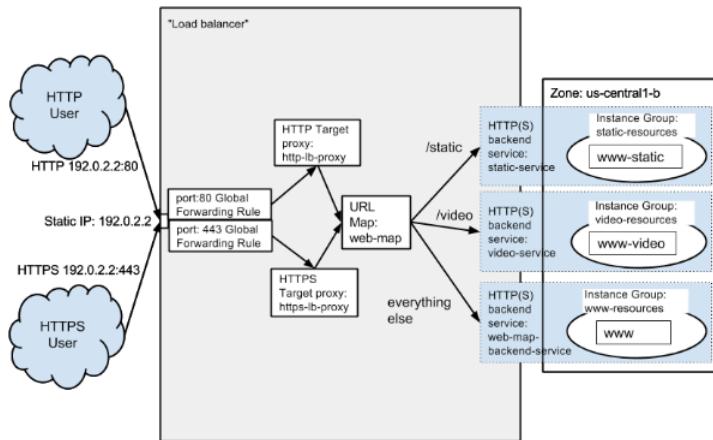


##### HTTP URI 기반의 라우팅

HTTP의 URI 또는 호스트명(서로 다른 호스트명도 하나의 로드 밸런서로 라우팅할 수 있다)을 기반으로 특정 서버 그룹으로 요청을 라우팅할 수 있다. 일반적인 네트워크 장비인 L4 등에서는 구현이 불가능한 기능인데 원리는 단순하다.

[그림 4-15]처럼 /static URL을 갖는 요청은 “static-resources”라는 서버 그룹으로 라우팅하고, /video URL을 갖는 요청은 “video-resources”라는 서버 그룹으로 라우팅하는 식이다.

그림 4-15 HTTP URI 기반의 라우팅



아주 간단한 기능이지만 활용도가 매우 높다. 웹 서버, 스트리밍 서버 등으로 콘텐츠 유형에 따라 서버를 나눌 수도 있다. 하지만 마이크로서비스 아키텍처(MSA)로 되어 있는 경우, 각 서비스 컴포넌트가 다른 URL을 가지기 때문에 앞단에 API 게이트웨이와 같이 URL에 따라 라우팅해주는 프록시 계층이 필요하다. 다행히 구글의 로드 밸런서는 이 기능을 지원하기 때문에 백엔드 서비스가 MSA로 구성되어 있을 경우 유용하게 사용할 수 있다.

### 4.9.3 호스트 및 경로 규칙 설정

호스트 및 경로는 HTTP 요청의 **호스트명**(photo.example.com, api.example.com)에 따라 백엔드 서비스로 라우팅하는 규칙을 설정하는 것과 HTTP URI를 기반(앞의 보충 설명 참고)으로 라우팅 규칙을 설정하는 단계다. 여기서는 모든 트래픽을 한 백엔드 서비스로 라우팅할 것이기 때문에 **호스트**나 URI 부분에 별도의 규칙을 명기하지 않고 비워놓은 상태에서 앞서 생성한 lb-test-backend 백엔드 서비스를 지정한다.

그림 4-16 호스트 및 경로 규칙

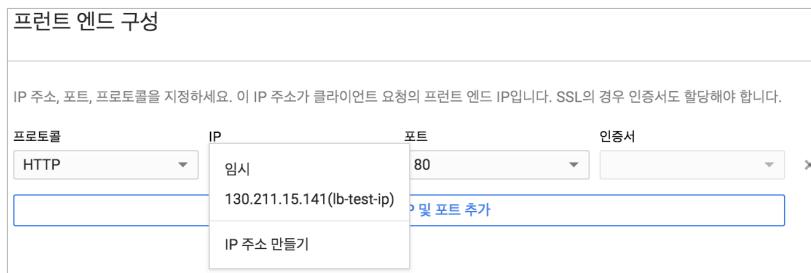


#### 4.9.4 프런트엔드 구성

마지막으로 프런트엔드 서비스를 설정하는데, 여기서는 어떤 프로토콜을 어떤 IP와 포트로 받을지를 지정하고, HTTPS의 경우에는 SSL 인증서<sup>SSL Certificate</sup>를 설정한다.

[그림 4-17]와 같이 HTTP 프로토콜과 80 포트로 설정하고, IP는 4.8절에서 할당한 정적 IP인 lb-test-ip를 할당한다.

그림 4-17 프런트엔드 구성



드디어 모든 설정이 완료되었다. 이제 [그림 4-11] 아래쪽의 [만들기]가 활성화될 것이다. 이 버튼을 클릭하면 마침내 로드 밸런서가 만들어진다(그림 4-18).

그림 4-18 새로 만들어진 로드 밸런서

The screenshot shows a web-based interface for managing a load balancer. At the top, there are tabs for '부하 분산기' (Load Balancer) and '+ 부하 분산기 만들기' (Create Load Balancer). Below this, there are sections for '부하 분산기' (Load Balancer) and '프로토콜' (Protocol). A table lists a single item: 'lb-test-http' under '부하 분산기' and 'HTTP(S)' under '프로토콜'. To the right of the table are icons for trash ('삭제') and edit ('수정'). A note at the bottom states: '전달 규칙 및 대상 프록시와 같은 부하 분산 리소스를 수정하려면 고급 메뉴로 이동하세요.' (Move to the advanced menu to modify delivery rules and target proxies).

## 4.10 테스트

모든 구성이 완료되었으니 테스트를 진행해보자. 웹 브라우저를 열고 로드 밸런서에 할당한 정적 IP로 반복해서 접속하며 백엔드를 구성하는 두 VM이 번갈아 응답하고 있음을 확인할 수 있다(결과 텍스트에서 address 항목을 확인하면 내부 IP가 번갈아 바뀌고 있을 것이다).

**NOTE**

로드 밸런서가 초기 셋업되기까지 약 30초 정도 걸리기 때문에 테스트를 바로 진행하면 서버 에러가 반환될 수도 있다.

# 오토 스케일링 적용하기

조대협

클라우드의 가장 큰 장점 하나는 들어오는 부하에 따라서 서버를 늘리고 줄일 수 있는 유연성에 있다. 그중에서도 부하량에 따라서 서버를 자동으로 늘리고 줄여주는 오토 스케일링auto scaling, 자동 확장 기능은 거의 필수라고 할 수 있다.

이번 장에서는 구글 클라우드에서 오토 스케일링을 설정하는 방법을 알아보도록 한다.

오토 스케일링을 설정하는 절차는 다음과 같다.

1. 인스턴스 템플릿 정의
2. 오토 스케일링 그룹 생성
3. 로드 밸런서 연결

각 단계를 자세히 알아보자.

## 5.1 인스턴스 템플릿 정의

인스턴스 템플릿이란 인스턴스를 생성할 때의 설정값(디스크 이미지, 인스턴스 크기, 네트워크 설정)을 미리 정해놓고, 템플릿에 정의한 대로 인스턴스를 생성하게 해주는 설정값의 모음이다.

인스턴스를 템플릿으로 생성할 때는 디스크 이미지를 선택하도록 되어 있다. 이때

OS와 각종 서버 프로그램 등이 미리 설치되어 있는 이미지를 생성한 후, 이 이미지로 템플릿을 생성해놓고, 오토 스케일링 시에 이 템플릿으로 인스턴스를 생성하게 한다. 또는 OS만 설치되어 있는 이미지를 사용하여 템플릿을 만들고, 서버가 시작될 때마다 자동으로 필요한 프로그램을 설치하는 스크립트를 실행하도록 할 수도 있다. 하지만 후자는 서버가 시작할 때마다 설치에 시간이 들기 때문에 그다지 권장하지 않는다.

그럼 이제부터 템플릿을 생성하는 방법을 알아보자.

### 1단계 : 인스턴스 생성 및 소프트웨어 설치

먼저 인스턴스를 하나 생성해서, 그 인스턴스에 필요한 소프트웨어를 모두 설치한다.

### 2단계 : 스냅샷 생성

[Compute Engine] > [스냅샷] > [스냅샷 만들기] 메뉴로 들어와 다음의 조건으로 새로운 스냅샷을 생성한다.

- 이름 : as-snapshot
- 소스 디스크 : (1단계에서 생성한 인스턴스)

스냅샷을 생성하는 자세한 방법은 4.4절에서도 설명했으니 참고하기 바란다.

### 3단계 : 디스크 생성

[Compute Engine] > [디스크] > [디스크 만들기] 메뉴로 들어와 다음의 조건으로 새로운 디스크를 생성한다.

- 이름 : as-image-disk
- 영역 : asia-east1-a
- 소스 유형 : 스냅샷
- 소스 스냅샷 : as-snapshot (2단계에서 생성한 스냅샷)

## 4단계 : 이미지 생성

[Compute Engine] > [이미지] > [이미지 만들기] 메뉴로 들어와 다음의 조건으로 새로운 이미지를 생성한다.

- 이름 : as-image
- 소스 : 디스크
- 소스 디스크 : as-image-disk (3단계에서 생성한 이미지)

## 5단계 : 템플릿 생성

[Compute Engine] > [인스턴스 템플릿] > [인스턴스 템플릿 만들기] 메뉴를 선택한다. 템플릿에 들어가는 내용은 인스턴스를 생성할 때의 내용과 동일하다. 단, 템플릿을 만들 때 [부팅 디스크]를 4단계에서 생성한 “as-image”를 선택해준다.

그림 5-1 직접 만든 이미지를 템플릿의 부팅 디스크로 사용하기



다음으로, 이 템플릿으로 생성된 인스턴스가 시작될 때 자동으로 애플리케이션을 시작하도록 해보자. 이를 위해 [관리] 탭에서 [시작 스크립트] 부분에 [예제 5-1]의 코드를 기입한다. 이 스크립트는 [예제 4-2]와 동일한 것으로, 이미지에 설치되어 있는 Node.js 애플리케이션을 자동으로 실행해준다.

### 예제 5-1 Node.js 애플리케이션을 구동하는 셀 스크립트

```
#! /bin/bash
export PORT=80
cd $HOME/restapi_mysql/
sudo -E npm start
```

## 5.2 오토 스케일링 그룹 생성

이제 이 템플릿을 이용해서 오토 스케일링을 지원하는 그룹을 생성해보자.  
[Compute Engine] > [인스턴스 그룹] > [인스턴스 그룹 만들기] 메뉴를 선택하자. 생성 조건은 다음과 같다.

- 이름 : as-group
- 영역 : asia-east1-a
- 생성 방법 : 인스턴스 템플릿 사용
- 인스턴스 템플릿 : as-instance-template (앞 절에서 생성한 템플릿이다.)
- 자동 확장 : 켜기
- 자동 확장 : CPU 사용량
- 타겟 CPU 사용량 : 50

## 그림 5-2 오토 스케일링을 위한 새 인스턴스 그룹 만들기

← 새 인스턴스 그룹 만들기

이름 ?  
as-group

설명 (선택사항)

위치  
디중 영역 그룹은 여러 영역을 통해 제공되므로 가용성이 더 높습니다. [자세히 알아보기](#)

단일 영역  
 디중 영역

영역 ?  
asia-east1-a

포트 이름 매핑 지정 (선택사항)

생성 방법  
템플릿을 사용하여 자동으로 크기를 조절할 수 있는 동일한 인스턴스의 그룹을 만듭니다. 템플릿을 사용하지 않을 경우 각 구성원을 직접 추가하고 관리해야 합니다. [자세히 알아보기](#)

인스턴스 템플릿 사용  
 기존 인스턴스 선택

인스턴스 템플릿 ?  
as-instance-template

자동 확장 ?  
켜기

자동 확장 기반: ?  
최상의 결과를 얻으려면 [자동 크기 조절 인스턴스 그룹을 구성하는 방법](#)을 참조하세요.

CPU 사용량

타겟 CPU 사용량 ?  
크기 조절 기능은 그룹 대상에 맞게 동적으로 VM을 만들거나 삭제합니다. [자세히 알아보기](#)

50 %

인스턴스 최소 개수 ?  
1

인스턴스의 최대 개수 ?  
10

대기 기간 ?  
60 초

## 5.3 로드 밸런서 설정

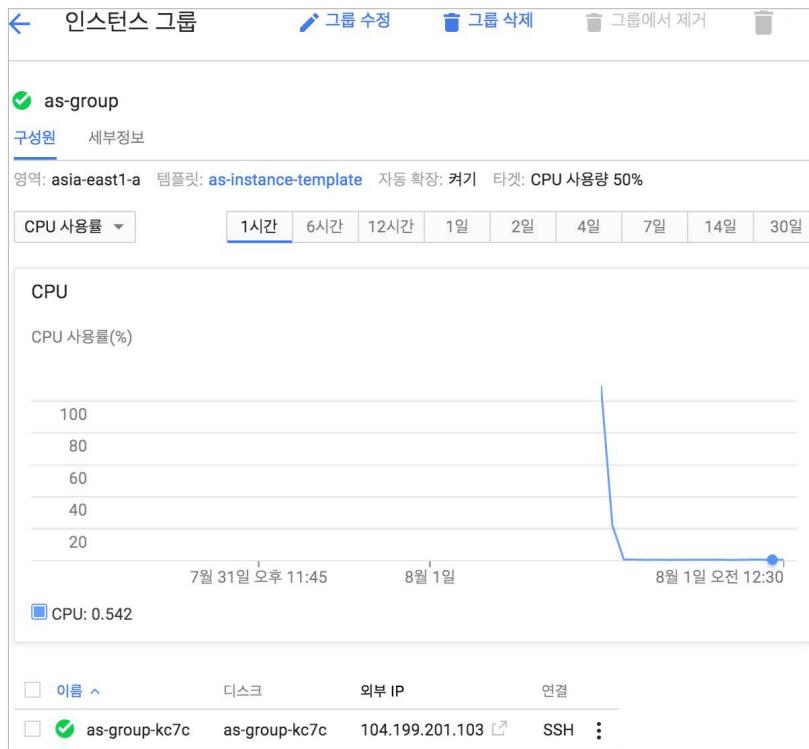
그룹 생성이 끝났으면 이 그룹을 로드 밸런서에 연결해줘야 한다. 로드 밸런서 설정 및 연결은 4장을 참고하기 바란다.

## 5.4 테스트

이제 모든 설정이 끝났으니 제대로 동작하는지 테스트해보자.

오토 스케일링을 적용한 그룹을 모니터링해보면 처음에는 다음과 같이 앞에서 설정한 대로 인스턴스 하나만으로 구동되고 있음을 확인할 수 있다.

그림 5-3 인스턴스 1개만 구동 중이다.



이제 아파치 ab 부하 테스트 툴로 부하를 넣어보자. 다음의 명령으로 클라이언트 100개가 각각 부하 1,000개씩을 주도록 한다.

```
%ab -n 1000 -c 100 http://130.211.11.38/
```

그림 5-4 아파치 ab로 부하를 늘려보자.

```
[terrycho-macbookpro:~ terrycho$ ab -n 1000 -c 100 http://130.211.11.38/
This is ApacheBench, Version 2.3 <Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 130.211.11.38 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:
Server Hostname: 130.211.11.38
Server Port: 80

Document Path: /
Document Length: 514 bytes

Concurrency Level: 100
Time taken for tests: 1.450 seconds
Complete requests: 1000
Failed requests: 0
Total transferred: 716000 bytes
HTML transferred: 514000 bytes
Requests per second: 689.47 [/sec] (mean)
Time per request: 145.039 [ms] (mean)
Time per request: 1.450 [ms] (mean, across all concurrent requests)
Transfer rate: 482.09 [Kbytes/sec] received

Connection Times (ms)
              min  mean [+/-sd] median   max
Connect:      38    54   10.8     51    91
Processing:   54    82   25.2     72   167
Waiting:      53    82   25.2     72   166
Total:        93   137   31.6    128   255

Percentage of the requests served within a certain time (ms)
 50%    128
 66%    146
 75%    155
 80%    163
 90%    182
 95%    203
 98%    219
 99%    229
```

한편으로 부하를 주는 동안 GCP 콘솔에서 오토 스케일링을 적용한 그룹을 살펴 보면 부하량에 따라 인스턴스 수가 자동으로 늘거나 줄 것을 확인할 수 있을 것이다.

## 5.5 몇 가지 추가 내용

### 5.5.1 오토 스케일링 조건

앞의 예제에서는 간단하게 CPU 기반의 오토 스케일링 규칙을 정의했는데, 이를 포함하여 오토 스케일링에 사용할 수 있는 조건은 다음과 같다.

- CPU 사용량
- HTTP 부하 분산(로드 밸런서 사용량)
- 측정항목 모니터링
- 위에 3가지를 조합한 값

CPU는 이미 살펴보았고, 로드 밸런서의 사용량이 많아져도 확장할 수 있다. 다음으로, 흥미로운 조건인 측정항목<sup>custom metric</sup>인데, 인스턴스의 모니터링 지표를 사용자가 직접 정의할 수 있다. CPU뿐 아니라 IO 또는 애플리케이션 서버(Tomcat, Django 등)의 응답 시간이나 유휴 스레드<sup>Idle Thread</sup>의 수 등을 모니터링 지표로 정의한 후, 이 지표가 일정 값이 넘어가면 인스턴스를 늘릴 수 있다. 자세한 내용은 <https://goo.gl/O8VgWG> 문서를 참고하기 바란다.

마지막으로 이상의 세 지표를 조합해서 조건을 정의할 수 있다. 예를 들어서 CPU 사용량이 60%이거나 로드 밸런서의 타겟 인스턴스의 사용량이 50%이거나, 타겟 인스턴스의 측정항목 값이 1,000일 때 확장하도록 할 수 있다.

오토 스케일링을 결정하는 상세한 조건에 대해서는 <https://goo.gl/cei3HZ> 문서를 참고하기 바란다.

## 5.5.2 자동 복구

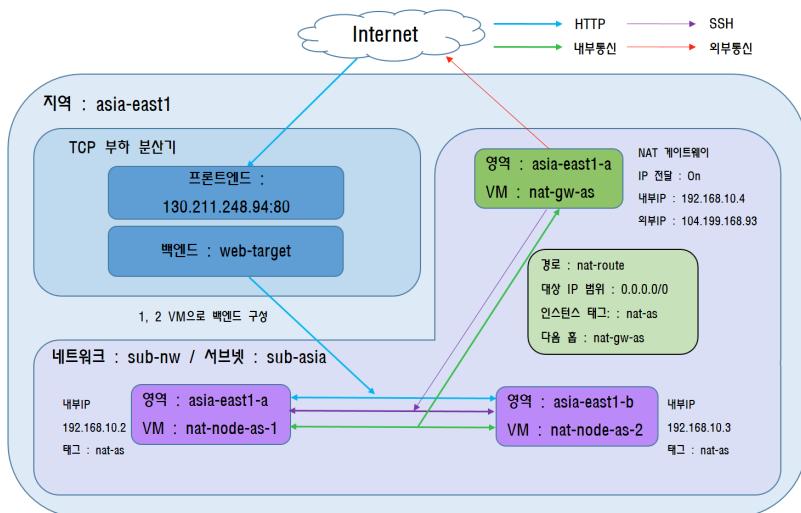
오토 스케일링 그룹은 자동 복구 Auto Healing라는 기능을 제공한다. HTTP 상태 확인을 이용하는 경우, 인스턴스가 일정 시간 응답이 없으면 문제가 있는 인스턴스라고 생각하고 자동으로 재시작해준다(이 책의 집필 시점에는 아직 베타 버전이다).



# 서브 네트워크와 NAT 네트워크 구성하기

최유석

이번 장에서는 구글 클라우드의 VM 인스턴스와 네트워크를 활용하여 서브 네트워크를 구성해보고, 여기에 NAT 게이트웨이를 추가해보도록 한다. 다음은 이 장에서 실습해볼 네트워크의 구성도다.



본격적인 구성에 앞서 다양한 네트워크 구성요소들의 개념을 간략히 정리해보도록 하겠다.

## 6.1 네트워크 구성요소

### 6.1.1 서브 네트워크

서브 네트워크<sup>subnet</sup>, 하위 네트워크란 하나의 공공 네트워크(공인 IP)를 논리적으로 분할하여 여러 개의 지역 네트워크(사설 IP)로 사용할 수 있게 해주는 방법이다. 이는 한정적인 네트워크 자원인 공인 IP 주소를 효율적으로 관리하고 사용하도록 해준다. 또한 트래픽을 분산하여 통신 속도를 높이는 효과가 있으며, 네트워크를 독립적인 여러 네트워크로 분할하여 보안 측면에도 유리하다.

구글 클라우드에서 서브 네트워크(사설 IP 사용)는 해당 지역의 미리 지정된 범위에서 생성되는 자동 모드와 사용자가 직접 IP 범위를 지정하는 맞춤설정 모드가 있다.

### 6.1.2 네트워크 종류

#### 자동 모드

자동<sup>auto</sup> 모드에서는 지역당 하나의 서브 네트워크를 가질 수 있으며, 다음과 같이 미리 정의된 범위에서 IP가 자동 할당된다.

표 6-1 지역별 IP 범위와 기본 게이트웨이

지역	IP 범위	기본 게이트웨이
us-west1	10.138.0.0/20	10.138.0.1
us-central1	10.128.0.0/20	10.128.0.1
us-east1	10.142.0.0/20	10.142.0.1
europe-west1	10.132.0.0/20	10.132.0.1
asia-east1	10.140.0.0/20	10.140.0.1

#### 맞춤설정 모드

맞춤설정<sup>custom</sup> 모드는 사용자가 직접 IP 범위를 지정하여 서브 네트워크를 구성 할 수 있다.

## 레거시 모드

레거시<sup>non-subnetwork</sup> 모드는 구글 클라우드 안에서 서브 네트워크 없이 단일 IP 범위와 단일 게이트웨이로 구성되는 네트워크로, 모든 지역 범위에 걸쳐서 사용할 수 있다. IP 주소는 전체 범위로 생성되기 때문에 지역이나 존 범위에서 연속적이지 않을 수 있다.<sup>01</sup>

### NOTE 네트워크 참고 사항

- 새로운 프로젝트의 기본 네트워크는 자동 모드 서브 네트워크다.
- 프로젝트당 최대 5개의 네트워크를 구성할 수 있다.
- 프로젝트당 최대 100개의 서브 네트워크를 생성할 수 있다.
- 각 네트워크는 상호 변경이 불가능하며, VM 인스턴스를 처음 생성할 때에 지정한 내부 네트워크는 변경할 수 없다.

## 6.1.3 NAT 게이트웨이

NAT<sup>Network Address Translation</sup> 게이트웨이는 통신망에서 사설 IP를 공인 IP로 변환해 주는 변환기로, 다수의 사설 IP를 하나의 공인 IP로 변환하여 한정된 자원인 공인 IP를 절약하기 위해 사용한다. 또한, 사설 IP를 외부에 노출하지 않고 방화벽을 설치하여 외부 공격으로부터 사용자의 통신망을 보호하는 수단으로 활용할 수 있다.

## 6.1.4 내부 IP와 외부 IP

IP 주소는 크게 내부 IP와 외부 IP로 구분할 수 있다.

### 내부(사설) IP

모든 VM에는 생성 시 내부 IP가 할당된다. 앞서 설명한 서브 네트워크 또는 레거시로 지정한 범위 안의 IP 주소가 할당되고, 같은 네트워크에서는 VM 인스턴스

<sup>01</sup> 레거시 모드는 클라우드 콘솔(웹 UI)로는 생성할 수 없고, 대신 명령줄 도구(Cloud SDK, SSH 등)에서 gcloud 명령어로 생성해야 한다.

의 이름을 기반으로 내부 VM들과 통신한다. 외부 인터넷과의 통신이 제한된다.

### 외부(공인) IP

외부 네트워크(인터넷)와 통신할 수 있으며, 임시<sup>ephemeral</sup>, 고정<sup>static</sup>, 없음<sup>none</sup>으로 생성할 수 있다. 외부 IP 주소를 가진 VM 인스턴스끼리는 다른 네트워크에 있더라도 서로 통신할 수 있다. 또한 외부 IP로 통신하더라도 구글 내부망을 이용하기 때문에 속도, 보안, 비용 측면에서의 이점은 그대로 유지된다.

### 6.1.5 경로

복잡한 네트워크에서 통신을 효율적으로 하려면 수많은 네트워크 중 최적의 길을 결정하는 역할이 필요하다. 이때 필요한 것이 바로 경로<sup>route</sup>다.

구글 클라우드에서의 경로란 일종의 라우팅 테이블<sup>routing table</sup>이며, 전체 글로벌 범위에서 사용할 수 있다. 네트워크를 생성하면 인터넷 게이트웨이와 내부 네트워크에 대한 경로는 기본으로 생성되며, 추가 경로 설정을 통해 다대일<sup>many-to-one</sup> NAT 게이트웨이, VPN<sup>Virtual Private Network</sup> 게이트웨이 구성 등에 사용할 수 있다.

### 6.1.6 방화벽 규칙

구글 클라우드 네트워크로 들어오는 모든 인바운드<sup>inbound</sup> 트래픽은 기본적으로 막혀 있으며, 이를 허용하는 규칙만 생성할 수 있다.

기본 네트워크에서는 TCP 22번 포트(SSH), 3389번 포트(RDP), ICMP, 내부(VM 간의 통신)를 제외하고 모두 막혀 있다. 반면, 직접 서브 네트워크나 레거시로 구성하는 경우에는 특별한 방화벽 규칙이 없다.

CIDR 표기법과 VM 인스턴스 태그를 사용한 범위로 트래픽을 제한할 수 있다.

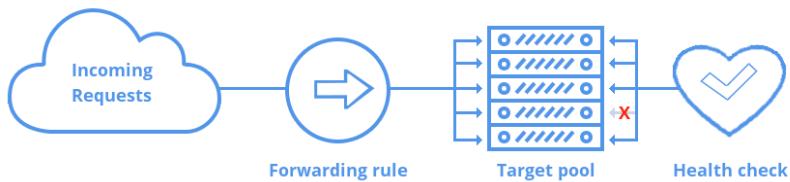
### 6.1.7 네트워크 부하 분산(TCP/UDP 기반)

구글 클라우드는 4장에서 설명한 HTTP 기반의 부하 분산 기능뿐 아니라, TCP/UDP 기반의 부하 분산도 지원한다.

TCP 로드 밸런서는 HTTP 로드 밸런서와는 다르게 지역 단위로만 사용할 수 있다(지역 간의 분산은 HTTP 로드 밸런서만 가능하다). 지역 범위에서 서버의 부하를 분산하고 장애를 인지하여 분산하는 L4 스위치의 기능과 비슷하다.

또한 프리웨일이 필요 없어서 구성 후 바로 사용할 수 있다.

그림 6-1 기본적인 TCP/UDP 네트워크 부하 분산의 흐름(출처 : <https://goo.gl/7ArorK>)



프로토콜 전달 protocol forwarding을 사용하기 위해서 전달 규칙forwarding rule에서 지정한 IP를 통해 들어오는 요청을 지정한 인스턴스 또는 인스턴스 풀에 전달하여 처리한다. 추가 옵션으로 상태 확인health check을 구성하여 주기적으로 인스턴스의 상태를 확인하고 비정상적인 인스턴스를 제외하는 기능을 한다.

### 6.1.8 지역과 영역

지역region은 지리적인 위치인 대륙 범위로 생각하면 된다. 현재 미국 동부/중앙/서부, 유럽 서부, 아시아 동부 지역이 있다. 반면, 영역zone은 각 지역에 실제로 위치한 데이터센터다.

## 6.2 네트워크 및 서버 구성

앞 절에서 네트워크를 구성하기 위한 기본 요소와 용어를 간단히 소개하였으니, 이제 서브 네트워크와 TCP 부하 분산을 이용해 간단한 웹 애플리케이션을 구성하고, 추가로 NAT 게이트웨이를 구성해보도록 하자.

### 6.2.1 프로젝트 생성

GCP 콘솔의 프로젝트 선택 메뉴에서 [프로젝트 생성] 메뉴를 선택해 실습용 프로젝트를 새로 생성해보자.

그림 6-2 실습용 프로젝트 생성



### 6.2.2 네트워크 생성

[네트워크] > [네트워크] 메뉴로 들어가면 위쪽에 [네트워크 만들기] 버튼이 보일 것이다. 이를 클릭하고 다음의 조건으로 새로운 네트워크를 생성한다.

- 이름 : sub-nw
- 하위 네트워크 : 맞춤설정
  - 이름 : sub-asia
  - 지역 : asia-east1
  - IP 주소 범위 : 192.168.10.0/24

### 그림 6-3 새로운 네트워크 만들기

← 네트워크 만들기

이름 [?](#)  
sub-nw

설명 (선택사항)

하위 네트워크  
하위 네트워크를 사용하면 Google Cloud 내에 자체 비공개 클라우드 토플로지를 만들 수 있습니다. 각 지역에서 하위 네트워크를 만들려면 '자동'을 클릭하고 하위 네트워크를 직접 정의하려면 '맞춤설정'을 클릭하세요. [자세히 알아보기](#)

맞춤설정 자동

이름 [?](#)  
sub-asia

설명 추가

지역 [?](#)  
asia-east1

IP 주소 범위 [?](#)  
192.168.10.0/24

+ 하위 네트워크 추가

만들기 취소

네트워크가 만들어지면 [그림 6-4]에서 보이는 것처럼 기본 네트워크(default) 외에 방금 생성한 sub-nw 네트워크도 보일 것이다. 이 화면에서 특정 네트워크의 이름을 클릭하면 자세한 정보를 살펴볼 수 있다.

그림 6-4 새로 생성한 네트워크(sub-nw)가 보인다.

네트워크		+ 네트워크 만들기		
이름 ^	지역	하위 네트워크	IP 주소 범위	게이트웨이
default		5		4
	us-central1	default	10.128.0.0/20	10.128.0.1
	europe-west1	default	10.132.0.0/20	10.132.0.1
	us-west1	default	10.138.0.0/20	10.138.0.1
	asia-east1	default	10.140.0.0/20	10.140.0.1
	us-east1	default	10.142.0.0/20	10.142.0.1
sub-nw		1		0
	asia-east1	sub-asia	192.168.10.0/24	192.168.10.1

### 6.2.3 방화벽 규칙 생성

[네트워킹] > [방화벽 규칙] 메뉴를 선택하면 방화벽 규칙 정보를 확인하고 관리 할 수 있다. 앞서 설명한 대로 default 네트워크를 제외하고는 내부 통신을 위한 TCP, UDP, ICMP까지 모두 수동으로 생성해야 한다.

GCP 콘솔 위쪽의 [방화벽 규칙 만들기]를 클릭하여 필요한 규칙들을 생성해보자. 먼저 같은 네트워크에 속한 VM 사이의 통신을 위해 tcp, udp, icmp를 허용하는 규칙을 만들 것이다. 조건은 다음과 같이 설정한다.

- 이름 : asia-sub-internal
- 네트워크: sub-nw
- 소스 필터 : 하위 네트워크 → sub-asia 192.168.10.0/24
- 허용된 프로토콜 및 포트 : tcp;udp;icmp

## 그림 6-5 방화벽 규칙 만들기

방화벽 규칙 만들기

기본적으로 네트워크 외부에서 수신되는 트래픽이 차단됩니다. 수신 트래픽을 허용하려면 방화벽 규칙을 설정하세요. 방화벽 규칙은 인스턴스로 수신되는 트래픽만 조정합니다. 인스턴스 내에 연결을 구성하면 이 연결을 통해 양방향으로 트래픽이 허용됩니다. [자세히 알아보기](#)

이름 ?

설명 (선택사항)

네트워크 ?

소스 필터 ?

하위 네트워크 ?

허용된 프로토콜 및 포트 ?

대상 태그 (선택사항) ?

만들기 취소

참고로 [소스 필터]의 경우 앞서 설명한 CIDR 범위나 인스턴스 태그를 지정하여 허용하게 할 수 있다. [대상 태그]는 인스턴스 태그만 지정할 수 있으며, 아무것도 지정하지 않으면 해당 네트워크의 모든 인스턴스에 적용된다.

같은 방식으로 외부에서 SSH로 접속할 수 있도록 TCP 22번 포트를 허용하는 방화벽 규칙도 생성한다. 조건은 다음과 같다.

- 이름 : asia-sub-ssh
- 네트워크: sub-nw
- 소스 필터 : 모든 소스의 트래픽 허용(0.0.0.0/0)
- 허용된 프로토콜 및 포트 : tcp:22

이 예제에서는 [소스 필터]를 모든 범위를 허용하도록 했지만, 여러분은 프로젝트 환경에 맞게 지정하길 바란다.

#### 6.2.4 VM 인스턴스 생성

네트워크를 생성하였으니, 이제 이 네트워크 환경을 사용할 VM을 생성해보자.

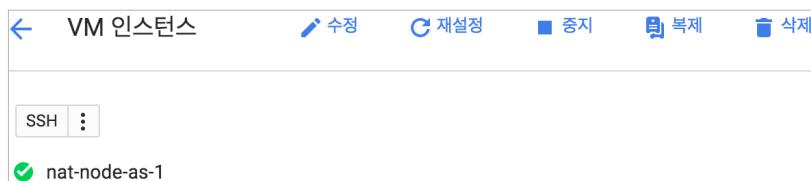
GCP 콘솔에서 [Compute Engine] > [VM 인스턴스] > [인스턴스 만들기]를 선택한다. 이 예제에서 사용할 인스턴스 생성 조건은 다음과 같다.

- 이름 : nat-node-as-1
- 영역 : asia-east1-a
- 머신 유형 : vCPU 1개, 3.75GB 메모리
- 부팅 디스크 : Ubuntu 14.04 LTS / SSD 10GB
- ID 및 API 액세스 : 모든 Cloud API에 대한 전체 액세스 허용 (기본 액세스 허용으로도 충분 하다.)
- 관리
  - 태그 : nat-as
- 네트워킹
  - 네트워크 : sub-nw
  - 하위 네트워크 : sub-asia
  - 내부 IP : 자동 (sub-asia 서브 네트워크의 범위인 192.168.10.0\24 안에서 자동 생성된다.)
  - 외부 IP : 임시 (이후 NAT 구성 시 “없음”으로 변경할 것이다.)

#### 6.2.5 두 번째 VM 인스턴스 생성

두 번째 인스턴스는 간단히 앞 절에서 생성한 nat-node-as-1 인스턴스를 복제해 만들 것이다. nat-node-as-1 인스턴스가 생성되면 이름을 클릭하여 상세페이지로 들어간다. 위쪽에 있는 [복제] 메뉴를 찾아 클릭하자.

그림 6-6 VM 인스턴스 상세페이지의 메뉴. 오른쪽에서 두 번째에 [복제] 메뉴가 보인다.



두 번째 인스턴스는 이름을 nat-node-as-2로, 영역을 asia-east1-b로 선택한 후, [만들기] 버튼을 클릭해 생성한다.

## 6.2.6 방화벽 규칙 추가

웹 브라우저로 접근할 수 있도록 HTTP가 사용하는 TCP 80 포트 방화벽 규칙을 추가해보자. 다시 [네트워킹] > [방화벽 규칙] > [방화벽 규칙 만들기] 메뉴로 들어간다. 이번에 생성할 규칙의 조건은 다음과 같다.

- 이름 : asia-sub-http
- 네트워크: sub-nw
- 소스 필터 : 모든 소스의 트래픽 허용(0.0.0.0/0)
- 허용된 프로토콜 및 포트 : tcp:80

다음으로 [Compute Engine] > [VM 인스턴스] 메뉴로 가서 앞서 생성한 VM의 내부/외부 IP 정보를 확인하자.

그림 6-7 두 VM 인스턴스의 요약 정보

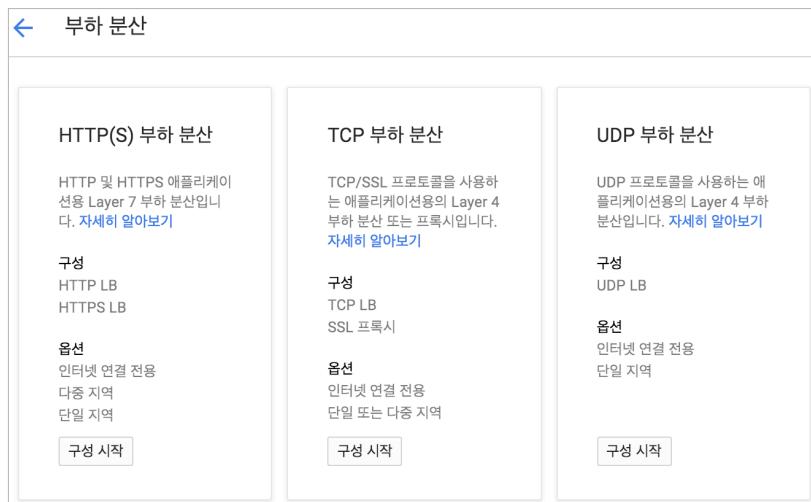
이름	영역	마신 유형	권장사양	다음에서 사용 중	내부 IP	외부 IP	연결
<input type="checkbox"/>	nat-node-as-1	asia-east1-a	vCPU 1개, 3.75GB		192.168.10.2	104.199.180.252	SSH
<input checked="" type="checkbox"/>	nat-node-as-2	asia-east1-b	vCPU 1개, 3.75GB		192.168.10.3	104.199.168.93	SSH

## 6.2.7 네트워크 부하 분산 구성

이제 서로 다른 IP로 구성된 VM 인스턴스들에 네트워크 부하 분산을 적용하여 하나의 IP로 접근할 수 있도록 하겠다. [네트워킹] > [부하 분산] 메뉴에서 [부하 분산 만들기] 버튼을 클릭해보자.

4장에서와 달리, 이번에는 “TCP 부하 분산” 방식으로 구성해볼 것이다.

그림 6-8 부하 분산 구성 방식. 이번에는 “TCP 부하 분산”으로 구성한다.



[구성 시작] 버튼을 클릭하면, 먼저 SSL 처리를 로드 밸런서로 이전할지를 묻는다. 기본값인 “아니요(TCP)”로 두고 계속한다.

그림 6-9 SSL 처리 이전 여부 선택



이제 [그림 6-10]과 같은 단계별 구성 화면이 나타난다.

그림 6-10 새 TCP 로드 밸런서 생성하기

새 TCP 부하 분산기	TCP 부하 분산기 만들기
<p>이름 <span>?</span></p> <input type="text" value="web-target"/> <p>● 백엔드 구성 아직 백엔드를 구성하지 않았습니다.</p> <p>● 프런트 엔드 구성 아직 프런트 엔드를 구성하지 않았습니다.</p> <p>① 검토 및 완료 <small>선택사항</small></p> <p><span>만들기</span> <span>취소</span></p>	<p>A TCP load balancer has 2 parts:</p> <ol style="list-style-type: none"><li>1. 백엔드 구성 백엔드 서비스는 연결된 백엔드 중 하나로 사용자 요청을 전달합니다. 백엔드는 인스턴스 그룹 및 용량 정보로 구성됩니다. 용량은 CPU 또는 RPS(초당 요청)로 표시됩니다.</li><li>2. 프런트 엔드 구성 백엔드 서비스로 트래픽을 전달할 프로토콜, 포트, IP 주소를 지정하세요.</li></ol>

이름을 “web-target”으로 하고, 백엔드와 프런트엔드를 차례로 구성해보자. 백엔드 구성 조건은 다음과 같다.

- 이름 : web-target (로드 밸런서 이름을 자동으로 가져온다.)
- 지역 : asia-east1
- 백엔드 : 기존 인스턴스 선택
  - nat-node-as-1 추가
  - nat-node-as-2 추가

그림 6-11 새 TCP 로드 밸런서의 백엔드 구성

백엔드 구성

이름 ?  
web-target

지역 ?  
asia-east1

백엔드 ?  
기존 인스턴스 그룹 선택 기존 인스턴스 선택

nat-node-as-1 (asia-east1-a) ×  
nat-node-as-2 (asia-east1-b) ×

사용 가능한 인스턴스 없음

백업 풀 ? (선택사항)  
없음

장애 조치율 ?  
10 %

상태 확인 ?  
상태 확인 없음

세션 연관성 ?  
없음

여기서 [백업 풀]은 사전에 백업 대상 풀을 준비해두고, 기본 풀을 구성하는 인스턴스들의 가동률이 명시한 [장애 조치율] 이하가 되면 백업 풀에서 요청을 처리하는 장애조치 옵션이다. [상태 확인]을 설정하면 백엔드를 구성하는 인스턴스들의 상태를 점검하여 비정상 인스턴스는 부하 분산에서 제외한다. [세션 연관성]은 클라이언트의 IP 주소나 프로토콜에 따라 일련의 연관된 요청들을 같은 백엔드 인스턴스로 전달하도록 설정할 수 있는 기능이다.

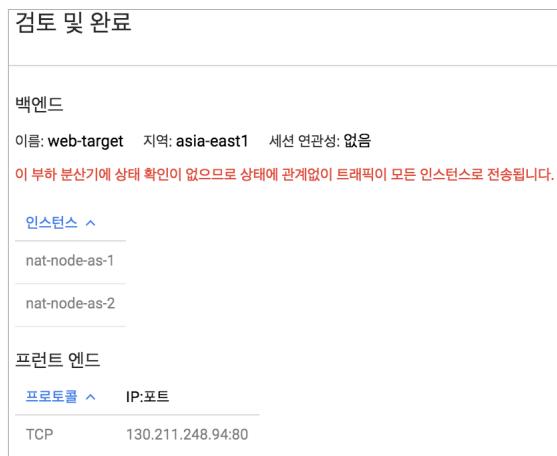
이어서 프런트엔드를 구성해보자. TCP 로드 밸런서의 프로토콜은 TCP로 고정되어 있다. IP 주소는 고정 IP를 만들어 할당하고, 포트는 HTTP용인 80으로 한다.

그림 6-12 새 TCP 로드 밸런서의 프런트엔드 구성



이것으로 구성이 완료되었다. [만들기] 버튼을 누르기 전에 마지막으로 [검토 및 완료] 부분을 확인해보자.

그림 6-13 새 TCP 로드 밸런서 구성의 상세 정보



상태 확인을 구성하지 않았다는 경고가 나온다. 이 기능은 나중에 추가할 수 있고, 이번 장에서 설명하는 주요 내용이 아니기 때문에 이 예제에서는 생략했다.

로드 밸런서를 만들고 VM 인스턴스들의 정보를 보면 새로 생성한 로드 밸런서 (web-target)에서 사용하고 있음을 확인할 수 있다(그림 6-14).

그림 6-14 로드 밸런서 구성 후의 두 VM 인스턴스의 요약 정보

이름	영역	머신 유형	권장사항	다음에서 사용 중	내부 IP	외부 IP	연결
<input type="checkbox"/> nat-node-as-1	asia-east1-a	vCPU 1개, 3.75GB		web-target	192.168.10.2	104.199.180.252	SSH
<input checked="" type="checkbox"/> nat-node-as-2	asia-east1-b	vCPU 1개, 3.75GB		web-target	192.168.10.3	104.199.168.93	SSH

이제 각 인스턴스의 외부 IP뿐 아니라 로드 밸런서의 IP로도 이들 인스턴스에 접근할 수 있다. 물론 로드 밸런서가 두 VM 인스턴스에 요청을 자동으로 배분해주게 된다.

## 6.3 NAT 게이트웨이 구성

이제 로드 밸런서를 통해 고정 IP로 접속할 수 있으니 각 VM들의 외부 IP를 제거해도 서비스를 이용할 수 있다. 즉, 외부 인터넷과의 직접 연결을 차단하여 보안성을 향상할 수 있다.

단, 내부 IP만으로는 인터넷에 접근할 수 없으니, 이후 새로운 소프트웨어 설치 등 인터넷에 연결해야 할 경우에 제한이 생긴다. 따라서 현재의 보안성을 유지하고 인터넷에 접근할 수 있도록 해주는 NAT 게이트웨이를 구성해보자.

### 6.3.1 외부 IP 제거

각 VM 인스턴스의 이름을 클릭하여 상세페이지로 이동하여 위쪽의 [수정]을 클릭한다. [외부 IP] 항목을 “없음”으로 바꾸고 화면 맨 아래의 [저장]을 클릭하면 된다. 두 인스턴스를 모두 수정했으면 [그림 6-15]처럼 될 것이다.

그림 6-15 외부 IP를 제거한 모습

이름	영역	머신 유형	권장사항	다음에서 사용 중	내부 IP	외부 IP	연결
<input type="checkbox"/> nat-node-as-1	asia-east1-a	vCPU 1개, 3.75GB		web-target	192.168.10.2	없음	SSH
<input checked="" type="checkbox"/> nat-node-as-2	asia-east1-b	vCPU 1개, 3.75GB		web-target	192.168.10.3	없음	SSH

### 6.3.2 NAT 게이트웨이 역할의 VM 생성

이제 NAT 게이트웨이가 될 VM 인스턴스를 생성한다. 구성 정보는 전체적으로 6.2.4 절에서 생성한 기존 VM과 같고, [이름]과 [IP 전달] 부분만 다르다(태그는 생략해도 된다).

- 이름 : nat-gw-as
- 영역 : asia-east1-a
- 머신 유형 : vCPU 1개, 3.75GB 메모리
- 부팅 디스크 : Ubuntu 14.04 LTS / SSD 10GB
- ID 및 API 액세스 : 모든 Cloud API에 대한 전체 액세스 허용
- 관리
  - 태그 : nat-as
- 네트워킹
  - 네트워크 : sub-nw
  - 하위 네트워크 : sub-asia
  - 내부 IP : 자동
  - 외부 IP : 임시
  - IP 전달 : 사용

그림 6-16 [IP 전달]을 “사용”으로 설정



**NOTE**

- IP 전달은 VM을 최초 생성할 때만 설정할 수 있으니 주의하자.
- 혼업에서는 일관된 기능을 위해 외부 IP를 고정으로 생성하는 것을 권장한다. 이 예제에서 임시로 설정한 이유는 무료 평가 계정에서는 고정 IP를 지역당 하나만 생성할 수 있기 때문이다.

### 6.3.3 경로 생성

6.1.4절에서 설명한 것처럼, 경로란 네트워크에서 통신하기 위해 거쳐 가는 길이다. 이러한 경로에 여러 옵션을 적용하여 외부 인터넷과 연결할 수 있게 해주는 NAT 게이트웨이에 필요한 설정을 할 수 있다. 여기에서는 애플리케이션 역할을 하는 nat-node-as-1과 nat-node-as-2에서 외부(인터넷)로 향하는 트래픽을 내부 통신을 통해서 NAT 게이트웨이 역할의 nat-gw-as로 전달하고, 전달받은 트래픽 요청을 nat-gw-as의 외부 IP로 인터넷에 접근하는 방식을 적용한다.

[네트워킹] > [경로] 메뉴로 이동하여 [경로 만들기] 버튼을 클릭하자. 이번 예제에서 사용할 경로의 조건은 다음과 같다.

- 이름 : nat-route
- 네트워크 : sub-nw
- 대상 IP 범위 : 0.0.0.0/0 (제한 없음)
- 우선순위 : 800 (기본값은 1,000이다.)
- 인스턴스 태그: nat-as (백엔드를 구성하는 인스턴스들과 같은 태그)
- 다음 흡 : 인스턴스 지정
- 다음 흡 인스턴스 : nat-gw-as (NAT 게이트웨이 역할의 VM 인스턴스)

그림 6-17 경로 만들기

경로 만들기

이름 ?  
nat-route

설명 (선택사항)

네트워크 ?  
sub-nw

대상 IP 범위 ?  
0.0.0.0/0

우선순위 ?  
800

인스턴스 태그 (선택사항) ?  
nat-as ×

다음 흡 ?  
인스턴스 지정

다음 흡 인스턴스 ?  
nat-gw-as

[만들기] [취소]

### 6.3.4 NAT 게이트웨이 설정 마무리

이제 NAT 게이트웨이 역할의 인스턴스에 접속하여 IP 전송과 마스커레이드  
MASQUERADE를 활성화하여 사설 IP를 외부 인터넷과 연결해보자.

[Compute Engine] > [VM 인스턴스]로 이동해 nat-gw-as의 오른쪽 [SSH] 버튼을 클릭해 SSH로 접속한다. 다음의 두 명령어는 차례로 IP 전송과 마스커레이드를 활성화해줄 것이다.

---

```
$ echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

---

이제 모든 구성이 완료되었다.

### 6.3.5 NAT 게이트웨이 동작 테스트

이제 지금까지 구성한 NAT 게이트웨이가 제대로 동작하는지 확인해보자. 이를 위해 우리는 게이트웨이 안쪽의 두 인스턴스(nat-node-as-1과 nat-node-as-2)에서 다음의 두 가지를 테스트해볼 것이다.

1. 인터넷(외부 네트워크)에 접속할 수 있는가?
2. 트래픽이 NAT 게이트웨이(nat-gw-as)를 거쳐 이동하는가?

먼저 각 인스턴스에 SSH로 접속하려고 보니, 외부 IP가 없어 바로는 불가능하다 (그림 6-18).

그림 6-18 외부 IP가 없어서 오른쪽의 [SSH] 버튼이 비활성화되어 있다.

이름	영역	머신 유형	권장사항	다음에서 사용 중	내부 IP	외부 IP	연결
<input type="checkbox"/> nat-gw-as	asia-east1-a	vCPU 1개, 3.75GB			192.168.10.4	104.199.168.93	SSH
<input checked="" type="checkbox"/> nat-node-as-1	asia-east1-a	vCPU 1개, 3.75GB		web-target	192.168.10.2	없음	SSH
<input checked="" type="checkbox"/> nat-node-as-2	asia-east1-b	vCPU 1개, 3.75GB		web-target	192.168.10.3	없음	SSH

이럴 때는 외부 IP가 있는 nat-gw-as로 먼저 접속한 후, 같은 서브 네트워크의 nat-node-as-1로 우회 접근하는 방법이 있다(nat-node-as-2도 마찬가지다). nat-gw-as의 오른쪽 [SSH] 버튼을 클릭해 터미널을 연후 다음 명령을 실행해 nat-node-as-1에 접속해보자.

```
$ ssh nat-node-as-1
```

**NOTE**

이때 권한이 부족하다고 나오면 다음 두 단계의 명령어를 실행해서 접속을 시도해본다.

```
$ eval `ssh-agent`  
$ ssh-add ~/.ssh/google_compute_engine
```

두 번째 명령어 실행 시 해당 파일과 디렉터리를 찾지 못한다면 다음 명령어로 SSH 키를 생성하고 다시 시도한다.

```
$ gcloud compute ssh nat-gw-as --ssh-flag= "-A" --zone asia-east1-a
```

이제 앞서의 두 조건을 확인해보자. 각각을 따로 확인할 수도 있지만 traceroute 명령으로 한 번에 확인할 수도 있다. 다음과 같이 traceroute로 www.google.com에 접속해보자. [그림 6-19]를 보면 NAT 게이트웨이인 nat-gw-as(내부 IP는 192.168.10.4)를 통해 갔음을 확인할 수 있다.

그림 6-19 traceroute로 NAT 게이트웨이 동작 확인하기

```
vegra_lee@nat-node-as-2:~$ traceroute www.google.com  
traceroute to www.google.com (74.125.204.104), 30 hops max, 60 byte packets  
1  nat-gw-as.c.net-test-1469982320762.internal (192.168.10.4)  0.953 ms  0.945 ms  0.907 ms  
2  ti-in-f104.1e100.net (74.125.204.104)  1.334 ms  1.324 ms  1.319 ms  
vegra_lee@nat-node-as-2:~$
```

## 6.4 참고자료

- 네트워크와 방화벽 사용하기 : <https://cloud.google.com/compute/docs/networking>
- 서브 네트워크 사용하기 : <https://cloud.google.com/compute/docs/subnetworks>

- 프로토콜 포워딩 : <https://cloud.google.com/compute/docs/protocol-forwarding/>
- 전송 규칙 : <https://cloud.google.com/compute/docs/load-balancing/network/forwarding-rules>
- 대상 풀 : <https://cloud.google.com/compute/docs/load-balancing/network/target-pools>
- 네트워크 로드 밸런싱 설정하기 : <https://cloud.google.com/compute/docs/load-balancing/network/>
- 네트워크 구성 실습 #1 : <https://goo.gl/9Zxdjl>
- 네트워크 구성 실습 #2 : <https://goo.gl/tvphFE>
- 네트워크 구성 실습 #3 : <https://goo.gl/QDi6Ch>

# IAM으로 사용자와 권한 관리하기

최유석

여러 사용자가 자원을 공유해야 하는 기업 프로젝트 환경에서는 자원에 대한 액세스 권한 관리는 선택이 아니라 필수다. 구글의 클라우드가 제공하는 IAM<sup>Identity and Access Management</sup>은 구글 클라우드 서비스들로의 액세스를 편리하고 일관된 방식으로 제어하도록 해준다.

## 7.1 구성원 유형

IAM에서 권한을 부여할 수 있는 구성원의 유형은 다음과 같다.

- Google 계정 : 명시한 특정 사용자에게 권한을 부여한다.  
예) user@gmail.com
- Google 그룹 : 해당 그룹의 구성원 모두에게 권한을 부여한다.  
예) admins@googlegroups.com
- 서비스 계정 : 개별 사용자 대신 구글 클라우드에서 호스팅되는 사용자 애플리케이션에 부여하는 ID다. 하나의 애플리케이션이라도 기능별로 서로 다른 서비스 ID를 부여할 수 있다.  
예) server@example.gserviceaccount.com
- Google Apps 도메인 : 해당 도메인의 구성원 모두에게 권한을 부여한다.  
예) example.com

그 외 다음과 같이 모든 사용자에게 권한을 부여할 수도 있다.

- 모든 구성원 : 'allUsers'를 입력하면 모든 사용자에게 액세스 권한이 부여된다.
- 모든 Google 계정 : 'allAuthenticatedUsers'를 입력하면 Google 계정으로 로그인 한 모든 사용자에게 액세스 권한이 부여된다.

## 7.2 사용자 역할

구글 클라우드 IAM은 단순하며 유연한 역할을 제공한다. 기본적인 역할은 다음과 같다.

- 뷰어<sup>Viewer</sup> : 구글 클라우드 자원을 보기만 할 수 있다.
- 편집자<sup>Editor</sup> : 뷰어 권한에 자원을 생성/삭제/변경할 수 있는 권한이 추가된다.
- 소유자<sup>Owner</sup> : 편집자 권한에 프로젝트 자체와 그 구성원을 관리하는 권한이 추가된다. 프로젝트를 처음 만든 계정에는 자동으로 이 권한이 부여된다.

기본적인 역할 외에 프로젝트 레벨에는 서비스 계정 행위자<sup>Service Account Actor</sup> 역할이 있는데, 서비스별로 세분화한 역할을 지정하여 사용 시에 서비스 계정의 정보를 획득하고 인증 시에 사용하여 보안성을 높인다. (세부적인 역할 사용 시 동시 사용)

또한, 결제 관련 사항은 결제 계정 관리자<sup>Billing Administrator</sup>를 통해 관리한다.

## 7.3 IAM 설정

IAM을 이용하면 매우 쉽게 권한을 설정할 수 있다. 6장에서 생성한 net-test 프로젝트에 구성원을 추가하여 IAM이 동작하는 모습을 간략히 살펴보자.

GCP 콘솔에서 [IAM 및 관리자] > [IAM] 메뉴로 이동해 위쪽의 [추가]를 클릭한다. 구성원 추가창이 뜨면 다른 구글 계정을 입력하고 역할은 [프로젝트] > [뷰어]

를 선택한 후 [추가] 버튼을 클릭한다.

그림 7-1 새로운 구성원을 뷰어 역할로 추가한다.

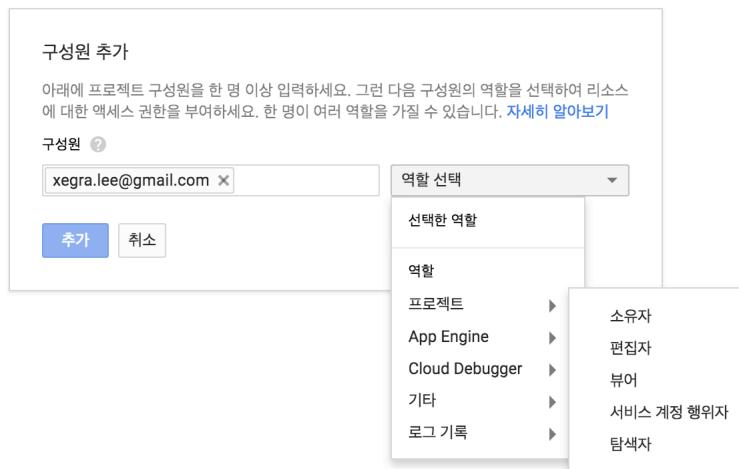
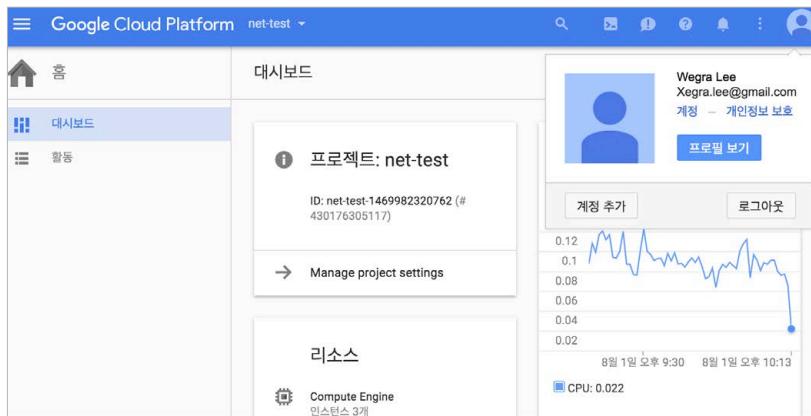


그림 7-2 뷰어 권한을 가진 구성원이 추가된 모습

IAM		
		+ 추가
프로젝트 'net-test' 권한		
이러한 권한은 전체 'net-test' 프로젝트 및 모든 관련 리소스에 영향을 미칩니다. 역할에 따른 액세스 권한을 부여하려면 사용자, 도메인, 그룹 또는 서비스 계정을 프로젝트에 추가하세요.		
일부 역할은 베타 개발 버전이며 향후 변경되거나 지원 중단될 수도 있습니다.		
<a href="#">자세히 알아보기</a>		
<input type="text"/> 이름 또는 역할로 필터링		
□ 유형	구성원 ^	역할
<input type="checkbox"/>	Compute Engine default service account 430176305117-compute@developer.gserviceaccount.com	편집자 ▾ 刪除
<input type="checkbox"/>	Google API 서비스 계정 ? 430176305117@cloudservices.gserviceaccount.com	편집자 ▾ 刪除
<input type="checkbox"/>	App Engine default service account net-test-1469982320762@appspot.gserviceaccount.com	편집자 ▾ 刪除
<input type="checkbox"/>	Wegra Lee wegra.lee@gmail.com	소유자 ▾ 刪除
<input type="checkbox"/>	xegra.lee@gmail.com	뷰어 ▾ 刪除

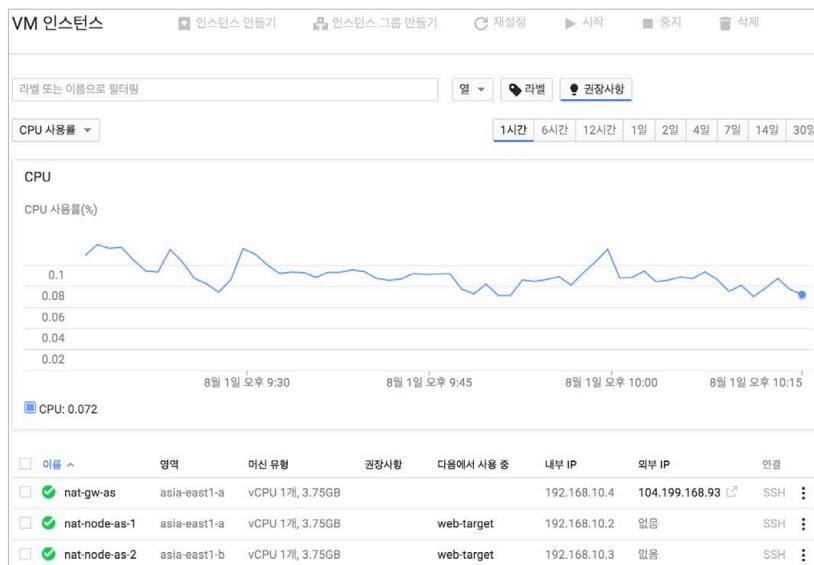
새로 추가한 구성원의 구글 계정으로 GCP에 로그인해보면 net-test 프로젝트의 자원을 확인할 수 있다.

그림 7-3 새 구성원의 계정으로 접속한 모습



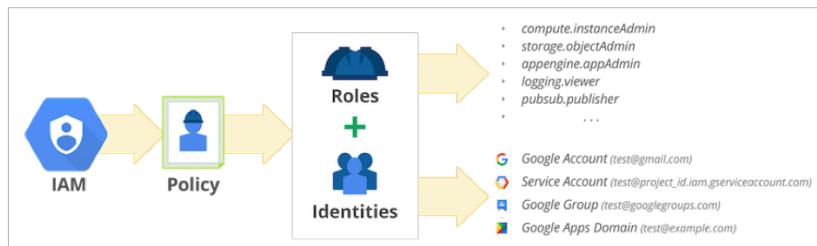
프로젝트의 곳곳을 둘러보면 많은 메뉴와 버튼들이 비활성화되어 있음을 확인할 수 있다. 예를 들어 [그림 7-4]에서는 [인스턴스 만들기]나 [SSH] 등의 버튼이 모두 비활성화되었다.

그림 7-4 뷰어 역할의 사용자로 접속한 모습(많은 기능이 비활성화되어 있다.)



지금까지 살펴본 기본 역할인 소유자, 편집자, 뷰어는 프로젝트 전체 자원에 대해 적용된다. 이와 달리 구글 클라우드에서는 아래와 같이 서비스별로 세분화된 제어도 제공한다.

그림 7-5 IAM의 구성원과 역할 모델(출처 : <https://cloud.google.com/iam/docs/overview>)



서비스별 특화 역할은 서비스에 맞는 여러 정책을 복합적으로 적용하여 서비스 각각에 따로 지정할 수 있어 전체 자원 관리에 대한 부담을 줄일 수 있다. 더 자세한 내용은 <https://cloud.google.com/iam/docs/> 문서를 참고하기 바란다.



# Cloud SQL에 접속하기

윤성재, 조대협

구글 클라우드에서는 Cloud SQL이라는 이름으로 MySQL의 매니지드 서비스를 제공한다. 이번 절에서는 서버에서 Cloud SQL에 접근하는 방법과 일반적인 MySQL 클라이언트로 접근하는 방법을 설명하고자 한다.

Cloud SQL은 아마존의 RDS와 같다고 보면 되는데, 현재는 1세대를 서비스하고 있고, 곧 2세대가 서비스될 예정이다. 1세대는 용량을 500GB까지만 지원하지만 2세대는 10TB까지 지원한다. 현재 지원되는 MySQL 버전은 5.5와 5.6이고, 내부 엔진으로는 InnoDB만을 제공한다.

2세대에 기대되는 기능으로는 On Prem(호스팅 센터에 있는) MySQL과 복제 구성이 있다. On Prem에 있는 서버를 마스터로 할 수도 있고, 반대로 마스터를 Cloud SQL로 사용하고 읽기 노드를 On Prem에 두는 등 다양한 하이브리드 구성이 가능하다.

자동 백업, 확장을 위한 읽기 전용 노드<sup>read replica</sup> 등 필수적인 기능을 제공하고 있다.

## 8.1 1세대 Cloud SQL 연결 방식

Cloud SQL은 RDS와는 다르게 아직 공인 IP만을 지원한다. 그래서 서버에 접

근하려면 이 공인 IP를 이용하면 된다. 보안을 위한 접근 통제 방법으로 특정 IP 주소에서 들어오는 트래픽만을 받도록 설정할 수 있다. 또는 PaaS 서비스인 구글 앱 앤진을 사용하는 경우에는 구글 앱 앤진의 인스턴스나 그룹 단위로 접근을 통제할 수 있다.

MySQL 클라이언트를 이용한다면 IP 등을 직접 입력하여 접속해도 되지만, 이렇게 하려면 Cloud SQL에서 IP를 허용해줘야 한다. 따라서 개발 중이나 운영 환경에서 IP 주소가 바뀌면 매번 설정을 수정해줘야 해서 불편할 수 있다.

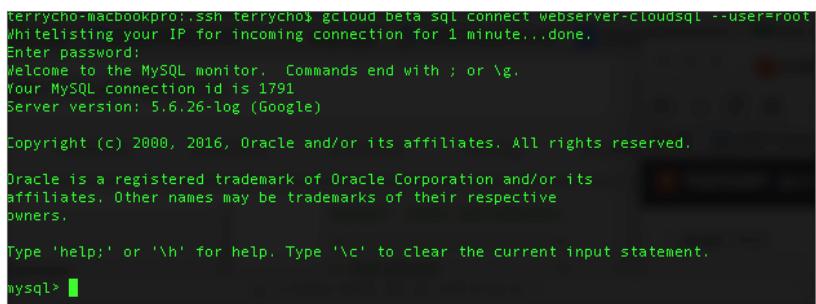
반면, 구글에서 제공하는 gcloud 도구를 이용하면 별도로 접속 IP를 열지 않아도 되어 더욱 편리하다. 먼저 gcloud를 설치한 후 다음 명령을 실행하면 MySQL 클라이언트와 동일한 툴로 접속된다.

---

```
$ gcloud config set project <프로젝트_이름>
$ gcloud beta sql connect <Cloud_SQL_인스턴스_이름> --user=root
```

---

그림 8-1



The screenshot shows a terminal window with the following text:

```
terrycho-macbookpro:~ ssh terrycho$ gcloud beta sql connect webserver-cloudsql --user=root
Whitelisting your IP for incoming connection for 1 minute...done.
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1791
Server version: 5.6.26-log (Google)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

--user=root 옵션에서 root 사용자는 MySQL 인스턴스 내의 사용자를 뜻한다.

gcloud 도구를 이용한 자세한 접속 방법은 <https://cloud.google.com/sql/docs/mysql-client#connect-ssl>에 잘 나와 있다. 참고로 <https://cloud.google.com/sql/docs/admin-tools#squirrel>에서는 Toad, MySQL

Workbench, SQuirrel SQL로 접속하는 방법을 자세히 설명하고 있다.

## 8.2 프록시를 이용한 2세대 Cloud SQL 연결

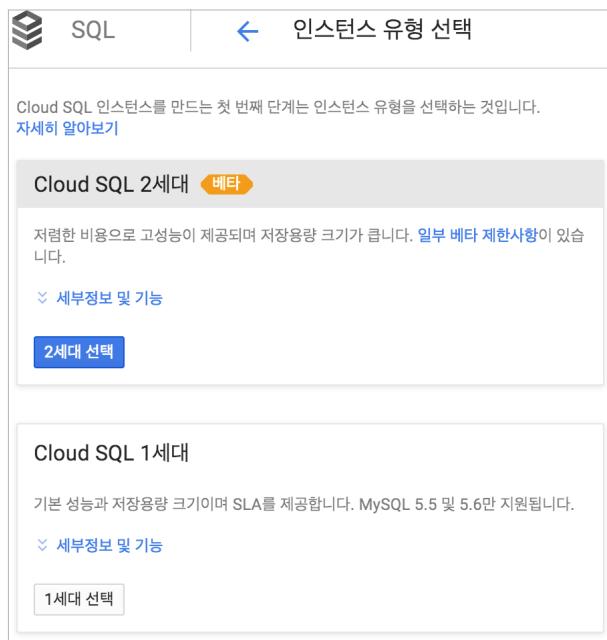
1세대 Cloud SQL은 공인 IP를 통해서만 접속할 수 있다는 것이 단점이다. 이에 2세대에서는 클라이언트에 프록시를 설치하여, 공인 IP를 이용한 방식보다 보안상 안전한 방식을 제공한다.

이번 절에서는 2세대 Cloud SQL을 구성하고 연결하는 일을 함께 해보려 한다.

### 8.2.1 Cloud SQL 생성

당연하게도, 가장 먼저 할 일은 Cloud SQL을 생성하는 일이다. GCP 콘솔에서 [SQL] 메뉴로 가 [인스턴스 생성] 버튼을 클릭해 새로운 DB를 만들어보자.

그림 8-2 Cloud SQL 유형(세대) 선택



**NOTE**

[그림 8-2]에서 Cloud SQL 2세대 항목의 “세부정보 및 기능”을 클릭하면 1세대보다 향상된 기능과 베타로서의 제약사항은 무엇이 있는지 볼 수 있다. 관련 정보는 <https://goo.gl/cO13bP>에서도 확인할 수 있다.

2세대를 선택하면 인스턴스를 만들 수 있는 화면으로 바뀐다. 우리가 생성할 조건은 다음과 같다.

- 인스턴스 ID : testdb-001
- 지역 : asia-east1
- 영역 : 자동 선택
- 머신 유형 : db-n1-standard-1
- 저장소 유형 : HDD
- 저장용량 크기 : 50GB
- 고가용성 : 장애조치용 복제본 만들기
- 자동 백업 사용 설정 : 오전 2:00 ~ 오전 6:00
- 승인된 네트워크 : [네트워크 추가]를 클릭해 집이나 사무실 등 직접 접속을 할 네트워크를 추가한다.
- 고급 옵션
  - 데이터베이스 버전 : MySQL 5.6
  - MySQL 플래그 : [항목 추가]를 클릭해 다음 플래그들을 추가한다.
- general\_log : On
- long\_query\_time : 5
- slow\_query\_log : On

여기까지 입력하고 마지막의 [생성]을 클릭하면 인스턴스가 만들어진다.

그림 8-3 새로 생성된 2세대 Cloud SQL

SQL	인스턴스	+ 인스턴스 생성
	인스턴스 ID ⓘ	유형
<span>✓</span> testdb-001	2세대	IP 주소 104.199.170.226
<span>✓</span> testdb-001-failover	장애 조치용 복제본	IP 주소 104.199.188.243
	사용한 저장용량 ⓘ	저장소 유형
	—	HDD
	장애 조치	사용 설정됨
		asia-east1
		⋮
		asia-east1
		⋮

## 8.2.2 사전 준비

SQL 인스턴스가 생성되었으면 이제부터 프록시를 이용하여 접속하는 방법을 보여줄 것이다(IP를 통한 접속 방법은 1세대와 같으니 8.1절을 참고하자). 가장 먼저 할 일은 프록시 설치다.

### 1단계 : Cloud SQL API 활성화

우선 Google Cloud SQL API를 사용할 수 있도록 해줘야 한다. GCP 콘솔에서 [API 관리자] > [라이브러리] 메뉴에서 “Google Cloud SQL API”를 찾아 클릭하자. 그러면 [그림 8-4]와 같이 API 정보 화면이 나타날 것이다.

그림 8-4 검색창에 검색어를 입력해 “Google Cloud SQL API”를 찾는 모습

The screenshot shows the Google Cloud API library interface. In the search bar at the top, the text "Google Cloud SQL API" is entered. Below the search bar, there is a list of results. The first result is "API 정보" (API Information), which is selected. This section contains a brief description: "API for Cloud SQL database instance management." To the right of the description, there are links to "문서" (Documentation) and "APIs Explorer에서 API 사용해 보기" (View API usage in APIs Explorer). Below this, there is a diagram illustrating the authentication flow. It shows an "App" (with a "GITHUB" logo and a "2" icon) connected to a "User Agent" (laptop and smartphone icons with checkmarks) and a "User Data" (profile icon). The "User Data" is also connected to a "Service" (server icons) and a "Login" (login screen icon). At the bottom of the diagram, there is a note: "이 API를 사용하여 웹 애플리케이션과 Google 서비스 간의 상호작용과 같은 서버 간 상호작용을 수행할 수 있습니다. 앱 수준 인증을 사용 설정하는 서비스 개정이 필요하며, API의 Google 헬스를 승인하는 데 사용하는 서비스 계정 키도 필요합니다. 자세히 알아보기".

[그림 8-4] 화면에서 [사용 설정]을 클릭해주면 잠시 후에 활성화된다. 여기에 “사용 중지”라고 나온다면 이미 사용 설정이 되어 있는 상태다.

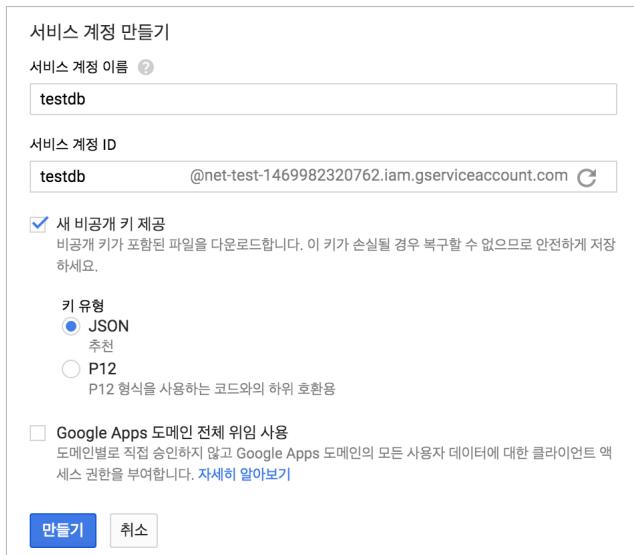
## 2단계 : 서비스 계정과 비공개 키 생성

클라이언트에서 프록시를 통해서 Cloud SQL 2세대에 접속할 때는 클라이언트에 프록시를 설치해야 한다. 리눅스, 맥, 윈도우를 지원하는데 각각의 설정 방법은 곧 설명할 것이다. OS별 설정 전에 공통적으로 접속 키 설정 등의 사전 작업이 필요하다.

먼저 프록시가 사용할 서비스 개정과 비공개 키를 생성한다. GCP 콘솔에서 [IAM 및 관리자] > [서비스 계정] 메뉴로 가보자. [서비스 계정 만들기]를 클릭하고 다음과 같이 입력한 뒤 [만들기]를 클릭한다.

- 서비스 계정 이름 : testdb
- 새 비공개 키 제공 : 체크
  - 키 유형 : JSON

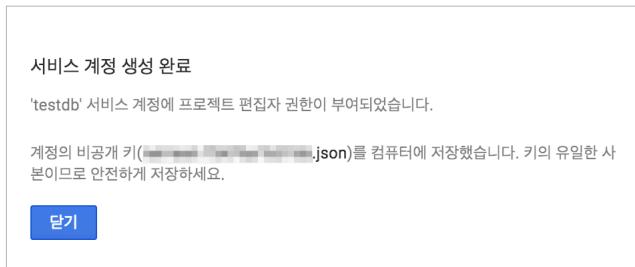
그림 8-5 새로운 서비스 계정 만들기



[그림 8-6]과 같은 메시지가 보인다면 서비스 계정 생성이 완료된 것이다. 다운로드 폴더에 가보면 비공개 키 파일이 JSON 파일로 저장되어 있을 것이다. 이 키파

일은 DB에 접속할 때 사용하는 매우 중요한 파일이므로 적절한 곳에 잘 보관하도록 한다.

그림 8-6 서비스 계정 생성 완료



### 3단계 : 사용자 비밀번호 변경

이제 MySQL을 생성한 후 root의 비밀번호를 변경한다. 이 예제에서는 클라이언트가 root 사용자로 볼도록 하였는데, 필요하다면 새로운 사용자를 생성해서 그 사용자를 이용하는 것을 권장한다.

GCP 콘솔에서 [SQL] 메뉴로 가서 “testdb-001” 인스턴스를 선택하면 세부정보 화면이 나타난다. [액세스 제어] > [사용자] 탭을 선택한다.

그림 8-7 SQL 인스턴스의 루트 사용자 비밀번호 변경

The screenshot shows the 'Instances' page for the 'testdb-001' instance. The 'Access' tab is selected. At the bottom, there is a button labeled 'Root Password Change'.

[그림 8-7] 아래에 보이는 “루트 비밀번호 변경” 버튼을 클릭해 비밀번호를 변경하면 클라이언트에서 Cloud SQL에 프록시를 통해 접속하기 위한 준비가 끝난다.

#### 4단계 : 방화벽 설정

구글 클라우드 플랫폼에는 기본으로 방화벽이 설정되어 있어 외부로부터의 모든 접근이 차단된다. 그래서 외부에서 접속하려면 해당 IP를 Cloud SQL에 등록해 줘야 한다.

인스턴스 세부정보 화면에서 [액세스 제어] > [승인] 탭을 열고 [+ 네트워크 추가]를 클릭해서 현재 사용 중인 외부 IP를 등록해주면 된다.<sup>01</sup> IP는 CIDR 형식에 맞게 입력해주면 되는데, 잘 모르겠다면 확인된 IP 뒤에 “/32”를 붙여서 등록한다. 이는 본인 IP 하나만 접속을 허용하겠다는 의미다.

그림 8-8 방화벽 규칙에 접근 허용 네트워크 추가하기

승인된 네트워크	저장되지 않음	수정
myhome (61.72.250.98/32)	저장되지 않음	수정

+ 네트워크 추가

01 IP를 모르겠다면 <http://whatismyipaddress.com>에서 쉽게 확인할 수 있다.

### 8.2.3 리눅스에서 접속하기

리눅스에서 접속하는 방법을 살펴보기 위해서 편의상 구글 클라우드에 VM을 생성해놓고, 이 VM으로부터 프록시를 이용하여 접속해보도록 하겠다. 이 예제는 데비안 리눅스를 기준으로 작성하였다.

#### 1단계 : 인스턴스 준비

GCP 콘솔에서 [Compute Engine] > [VM 인스턴스] > [인스턴스 만들기]를 선택하여 다음의 조건으로 생성해준다(자세한 생성 방법은 2장을 참고하기 바란다).

- 이름 : testapp-001
- 영역 : asia-east1-a
- 머신 유형 : vCPU 1개
- 부팅 디스크 : Debian GNU/Linux 8 (jessie)
- ID 및 API 액세스 : 기본 액세스 허용
- 방화벽 : HTTP 트래픽 허용

인스턴스 생성이 완료되면 해당 인스턴스 오른쪽 끝의 [SSH] 링크를 눌러 서버에 접속한다.

#### 2단계 : 프록시 설치

로그인에 성공했으면 다음 명령어를 입력하여 프록시를 내려받는다.

---

```
$ wget https://dl.google.com/cloudsql/cloud_sql_proxy.linux.amd64
```

---

이제 편의상 이 파일의 이름을 보기 좋게 바꾸고 실행 권한을 부여한다.

---

```
$ mv cloud_sql_proxy.linux.amd64 cloud_sql_proxy  
$ chmod +x cloud_sql_proxy
```

---

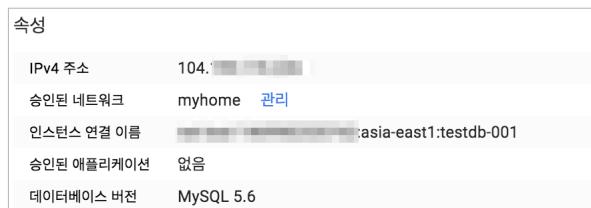
### 3단계 : 프록시 시작

인스턴스의 콘솔로 가서 프록시를 시작해보자.

```
$ sudo mkdir /cloudsql  
$ sudo chmod 777 /cloudsql  
$ sudo ./cloud_sql_proxy -dir=/cloudsql -instances=<인스턴스_연결_이름>  
-credential_file=<키_파일_경로> &
```

<인스턴스\_연결\_이름>은 Cloud SQL의 인스턴스 세부정보에서 확인할 수 있다.  
<키\_파일\_경로>는 앞서 8.2.2절 2단계에서 생성한 비공개 키 JSON 파일을 서버에 복사한 후 이 파일의 위치를 적어주면 된다.

그림 8-9 Cloud SQL 인스턴스 세부정보의 “속성” 영역. “인스턴스 연결 이름”을 확인할 수 있다.



이제 다음 명령에서 <프로젝트\_이름>을 현재 프로젝트 이름으로 바꿔 실행하자. “Ready for new connections”란 메시지가 보이면 프록시가 정상적으로 실행된 것이다.

```
$ sudo ./cloud_sql_proxy -dir=/cloudsql -instances=<프로젝트_이름>:asia-east1:testdb-001 -credential_file=/home/sjyun/.keys/78a8c4166b03.json &
```

### 4단계 : MySQL 클라이언트 설치 및 접속

먼저 다음 명령으로 MySQL 클라이언트를 설치하자.

```
% sudo apt-get update  
% sudo apt-get install mysql-client-5.6
```

MySQL 클라이언트가 설치되었으면 직접 연결해볼 차례다. 다음 명령어를 이용하면 접속할 수 있다.

---

```
% mysql -u root -p -S /cloudsql/<인스턴스_연결_이름>
```

---

“Enter password:” 메시지가 나오면 8.2.2절 3단계에서 설정한 루트 암호를 입력하면 접속된다.

### 8.2.4 맥에서 접속하기

맥 사용자는 Cloud SQL에 접속하기 위해 MySQL Workbench와 터미널을 모두 이용할 수 있다. Workbench 방식은 다음 절에서 설명하도록 하고, 여기에서는 많은 이가 선호하는 터미널 방식을 알아보도록 하겠다.

#### 1단계 : MySQL 설치

자신의 맥에 MySQL이 설치되어 있다면 바로 접속하면 되는데, 없다면 설치해주도록 하자. <http://dev.mysql.com/downloads/mysql/>에서 운영체제 버전에 맞는 파일을 내려받아 설치하면 된다.

Download를 클릭하면 로그인이나 가입을 하라고 나오는데, 그냥 무시하고 조금 아래의 “No thanks, just start my download.”를 클릭하면 바로 다운로드를 시작한다.

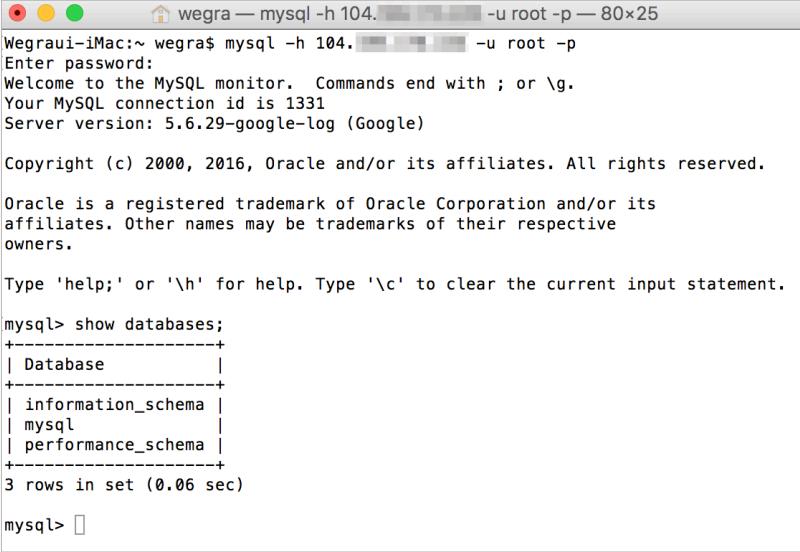
설치 방법은 다른 맥용 앱 설치와 다르지 않으니 자세한 설명은 생략하겠다.

#### 2단계 : 터미널로 접속

이제 터미널을 열어서 mysql 명령으로 Cloud SQL에 연결해보자. 접속을 위한 외부 IP는 Cloud SQL의 처음 화면에서 확인할 수 있다. 다음과 같이 입력하고 root 암호를 넣어주면 접속된다.

```
$ mysql -h <Cloud_SQL_외부_IP> -u root -p
```

그림 8-10 터미널에서 MySQL로 Cloud SQL 인스턴스에 접속한 모습



The screenshot shows a terminal window titled 'wogra — mysql -h 104.154.14.121 -u root -p — 80x25'. The session starts with the MySQL monitor prompt: 'Welcome to the MySQL monitor. Commands end with ; or \g.' It displays the server version: '5.6.29-google-log (Google)'. Copyright information from Oracle is shown. A note about trademarks follows. The user then runs the command 'show databases;' which lists three databases: 'information\_schema', 'mysql', and 'performance\_schema'. Finally, the user types 'exit' to leave the MySQL monitor.

```
wogra — mysql -h 104.154.14.121 -u root -p — 80x25
Wegraui-iMac:~ wegra$ mysql -h 104.154.14.121 -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1331
Server version: 5.6.29-google-log (Google)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.06 sec)

mysql> exit
```

#### 주의

MySQL을 처음 설치했다면 터미널에서 mysql 명령어를 찾을 수 없어 오류가 날 것이다. 분명 설치 했는데 왜 명령어를 못 찾을까? 맥용 MySQL이 설치된 기본 위치는 /usr/local/mysql인데, 인스톨러가 PATH 환경변수에 추가해주지 않아서 발생하는 오류다. 텍스트 편집기로 \$HOME 폴더 안의 .bash\_profile 파일을 열어 다음과 같이 PATH를 추가해주자.

```
export PATH="$PATH:/usr/local/mysql/bin"
```

이제 터미널을 닫았다가 다시 열거나 다음 명령을 실행해주면 변경한 내용이 적용된다.

```
$ . .bash_profile
```

## 8.2.5 윈도우에서 접속하기

이번엔 윈도우에서 MySQL Workbench를 이용하여 접속하는 방법을 알아보자.

MySQL Workbench를 이용하면 PC에 MySQL 서버나 클라이언트를 설치하지 않고도 서버에 접속할 수 있어 편리하다. 또한, Workbench는 윈도우뿐 아니라 우분투, 데드햇, 페도라, 맥 OS 등 다양한 플랫폼을 지원한다.

### 1단계 : MySQL Workbench 설치

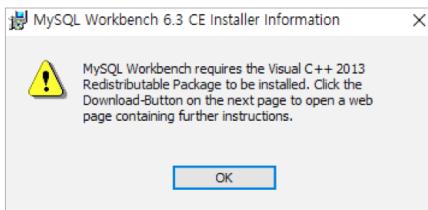
<http://dev.mysql.com/downloads/workbench/>에 방문해 자신의 윈도우 버전에 맞는 MySQL Workbench 설치 파일을 내려받아 설치해보자.

MySQL 설치 때와 같이 로그인이나 가입을 권유하는데, 무시하고 아래쪽의 “No thanks, just start my download.”를 클릭하면 바로 내려받는다.

설치 방법은 다른 프로그램들과 다르지 않다. 내용을 보면서 “Next”를 눌러 주면 설치가 진행된다. 설치가 끝나고 마지막 화면에서 “Finish”를 클릭하면 Workbench가 자동으로 실행된다.

#### 주의

설치를 시작하면 다음과 같은 화면이 나타날 수 있다.



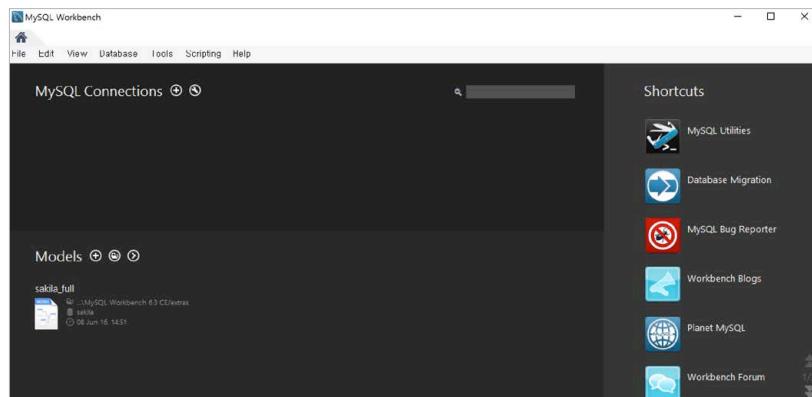
이는 Workbench가 필요로 하는 두 프로그램이 설치되어 있지 않을 때 나타난다. [OK]를 클릭하면 다음 그림이 나타나는데, 원쪽 아래에 보면 “Download Prerequisites” 버튼이 보인다. 버튼을 클릭하여 요구하는 프로그램들을 설치한 후 Workbench 설치를 다시 실행하면 정상적으로 진행될 것이다.



## 2단계 : Workbench로 접속하기

[그림 8-11]은 MySQL Workbench가 실행된 화면이다.

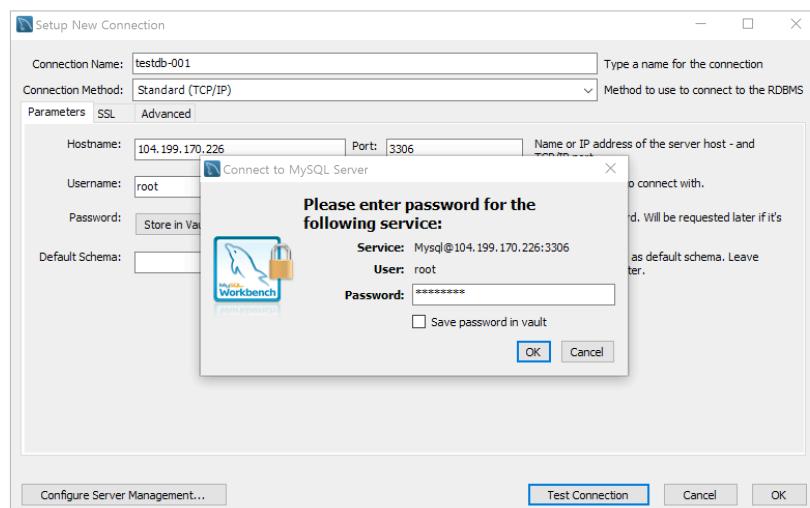
그림 8-11 MySQL Workbench 실행 화면



"MySQL Connections" 옆의 "+" 기호를 클릭하면 새로운 연결을 설정하는 창이 나타난다. 다음의 정보를 입력하고 아래쪽의 [Test Connection] 버튼을 클릭해 이 연결이 올바른지 확인할 수 있다. 사용자 암호를 입력하고 테스트를 수행하면 성공 여부를 알려줄 것이다.

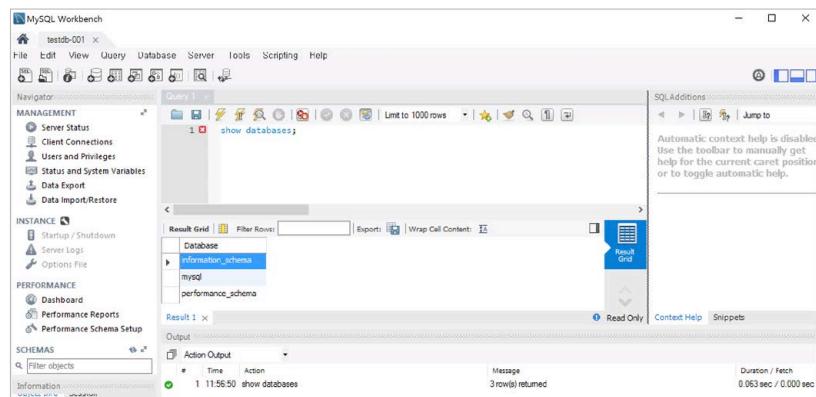
- Connection Name : testdb-001
- Hostname : [Cloud SQL 인스턴스의 외부 IP]

그림 8-12 새로운 연결 설정과 테스트



연결 테스트도 성공했다면 [OK]를 누르자. 초기 화면(그림 8-11)에 “testdb-001” 연결이 추가되었을 것이고, 이를 클릭하면 Cloud SQL 인스턴스에 접속하게 된다. 접속 후의 화면은 [그림 8-13]과 같으며, 가운데 “Query 1” 탭에 SQL문을 입력하고 실행하면 바로 아래에서 결과를 볼 수 있다.

그림 8-13 MySQL Workbench로 Cloud SQL 인스턴스 접속한 모습



이제 MySQL Workbench의 모든 기능을 사용할 수 있다.

## 8.2.6 애플리케이션에서 접속하기

이번에는 애플리케이션에서 프록시를 통하여 Cloud SQL에 접속하는 방법을 알아보자. 여기서는 VM 인스턴스에 설치한 아파치 웹 서버와 PHP를 이용하여 MySQL에 연결하는 예를 보여줄 것이다.

### 1단계 : 아파치 웹 서버와 PHP 설치하기

먼저 8.2.3절에서 만들어둔 VM 인스턴스(testapp-001)의 터미널을 열어 Apache 2 서버와 PHP5를 설치하자.

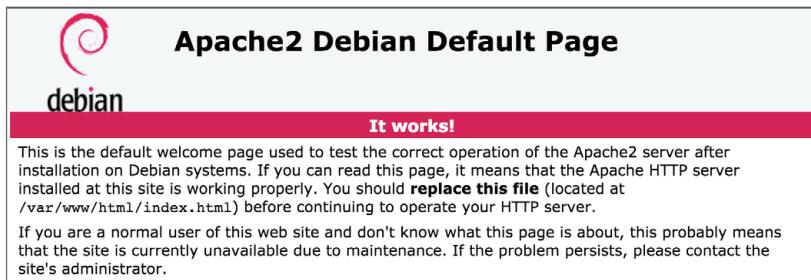
---

```
$ sudo apt-get install apache2 php5
```

---

이제 웹 브라우저에서 VM 인스턴스의 외부 IP로 접속하면 Apache2 Debian Default Page 화면을 볼 수 있을 것이다.

그림 8-14 데비안용 아파치2 웹 서버의 기본 화면



만약 아파치 시작 화면이 보이지 않는다면 apache2가 실행되고 있는지 확인하고 시작해준다.

---

```
$ sudo service apache2 status  
$ sudo service apache2 start
```

---

다음으로 PHP5 용 mysql 모듈을 설치한다.

---

```
$ sudo apt-get install php5-mysql  
$ sudo service apache2 stop  
$ sudo service apache2 start
```

---

이제 다음 코드에서 <프로젝트\_이름> 부분을 현재 프로젝트 이름으로 바꿔주고, /var/www/html 폴더에 mysql\_connect.php라는 이름으로 저장한다.

---

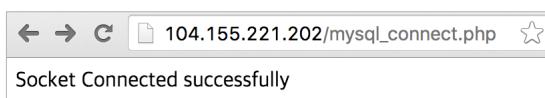
```
<?php  
// we connect to localhost and socket  
$link = mysql_connect(':/cloudsql/<프로젝트_이름>:asia-east1:testdb-001', 'root',  
'testdb');  
if (!$link) {  
    die('Could not connect: ' . mysql_error());  
}  
echo 'Socket Connected successfully';  
mysql_close($link);  
?>
```

---

## 2단계 : 브라우저로 접속하기

이제 브라우저에서 접속해보자. [그림 8-15]와 같은 결과를 볼 수 있을 것이다.

그림 8-15 아파치 + PHP로 Cloud SQL 인스턴스에 접속한 모습



지금까지 최근 업데이트된 Google Cloud SQL 2세대의 생성부터 접속 방법까지 알아보았다. Cloud SQL을 처음 사용하려는 이들에게 도움이 되었길 바란다.



# Stackdriver로 클라우드 자원 모니터링하기

김영균

Stackdriver로는 구글 클라우드뿐만 아니라 아마존 클라우드 양쪽을 모니터링 할 수 있다. 단순한 인프라 모니터링에서부터, 모든 시스템과 애플리케이션의 로그 수집 및 분석, 애플리케이션 성능 추적과 에러 보고 기능을 제공한다. 또한 슬랙Slack이나 페이저더티Pagerduty와 같은 외부 서비스와도 쉽게 연동할 수 있다.

이번 장에서는 Stackdriver를 사용하기 위하여 계정 생성, 모니터링 에이전트 설치, 측정항목<sup>metric</sup> 설정 순서로 설명할 것이다.

## 9.1 Stackdriver 계정 생성

Stackdriver를 사용하려면 구글 클라우드 플랫폼 계정을 연동하고 몇 가지 설정을 추가해주어야 한다.

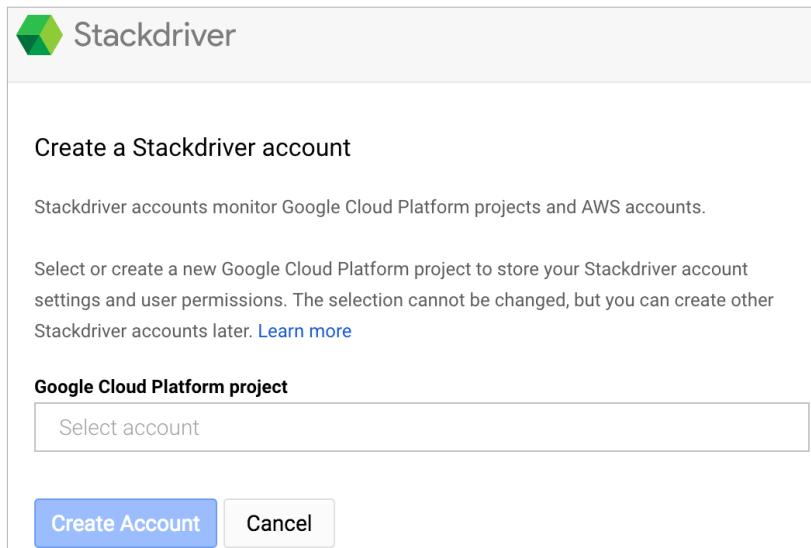
### 1단계 : Stackdriver 계정 생성

GCP 콘솔에서 [모니터링] 메뉴로 들어가 GCP 계정으로 로그인하면 [그림 9-1]과 같은 화면이 나타날 것이다.<sup>01</sup>

---

01 집필 시점에 Stackdriver는 아직 한글화가 되어 있지 않다.

그림 9-1 Stackdriver 계정 생성 화면



우선 모니터링할 프로젝트를 선택해야 하는데, 기존 프로젝트 중에서 고르거나 새로 생성할 수 있다. 프로젝트 이름을 입력한 후 [Create Account] 버튼을 클릭한다.

## 2단계 : GCP 프로젝트 추가 (선택 사항)

이번 단계에서는 함께 모니터링할 다른 프로젝트를 선택할 수 있다. 이 예제에서는 6장에서 만든 net-test만 선택하고 화면 아래의 [Continue]를 클릭해 다음 단계로 건너뛰도록 한다.

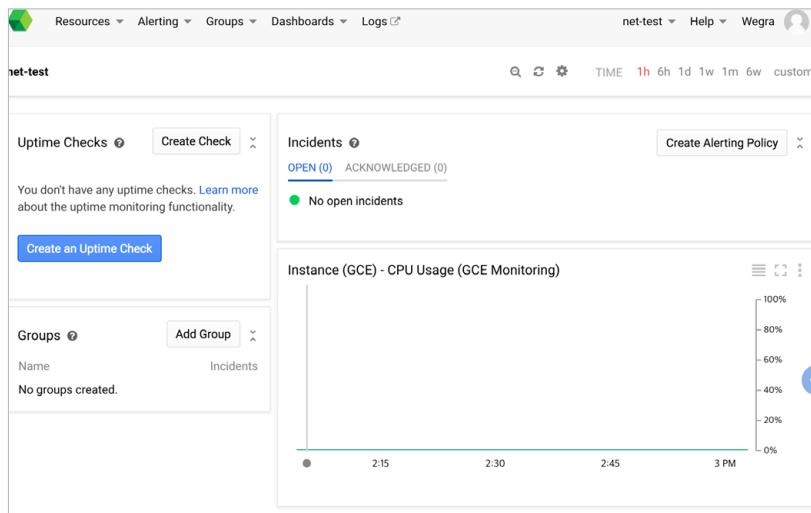
## 3단계 : AWS 계정 추가 (선택 사항)

앞서 Stackdriver는 GCP뿐 아니라 AWS도 모니터링할 수 있다고 하였다. 이번 단계의 화면에는 AWS를 Stackdriver에 인증하는 방법에 대한 설명과 AWS 계정 정보를 입력하는 곳이 보일 것이다. 하지만 역시 이번 예제에서는 화면 가장 아래의 [Done] 버튼을 클릭해 건너뛰도록 한다.

## 4단계 : 모니터링 시작

이제 초기 수집이 완료되었다는 메시지와 함께 [Launch monitoring] 버튼이 보일 것이다. 이 버튼을 클릭하면 마지막으로 이메일 보고서 수신 주기(매일, 매주, 받지 않음)를 묻고, 드디어 모니터링 화면 대시보드가 나타난다(그림 9-2).

그림 9-2 Stackdriver 대시보드



## 9.2 모니터링 에이전트 설치

Stackdriver 에이전트는 VM에서 시스템과 애플리케이션의 CPU 사용률, 디스크 트래픽, 네트워크 트래픽, 가동 시간과 같은 측정항목을 수집하는 데몬이다. 구글 컴퓨터 엔진과 앱 엔진은 물론 아마존 EC2 등의 다양한 VM과 운영체제를 지원한다. 자세한 내용은 <https://cloud.google.com/monitoring/agent/>를 참고하자.

## 9.2.1 윈도우에 에이전트 설치하기

윈도우 에이전트는 CPU와 메모리 사용률, 페이지 파일 및 볼륨 사용을 기록한다. IIS나 SQL 서버를 실행하고 있다면 이들 서버 관련 정보도 자동으로 수집한다. 윈도우 에이전트는 서비스 형태로 실행되며, 설치하는 즉시 측정항목을 수집하기 시작한다.

### 에이전트 설치

다음의 URL에서 설치 파일을 내려받아 실행하면 에이전트가 설치된다(별다른 설정은 요구하지 않는다).

- <https://repo.stackdriver.com/windows/StackdriverInstaller-GCM-17.exe>

### HTTP 프록시 사용

HTTP 프록시를 사용한다면 에이전트가 아웃바운드 HTTPS를 사용하여 Stackdriver로 데이터를 보낼 수 있도록 HTTP\_PROXY 환경변수를 설정한다.

---

```
$ setx http_proxy http://<프록시_주소> /m
```

---

### 에이전트 버전 결정, 업데이트, 삭제

- 윈도우에 설치할 에이전트의 버전을 결정할 방법이 현재 없다.
- 윈도우에서 에이전트를 업데이트하려면 에이전트를 다시 설치해야 한다.
- 에이전트를 삭제하려면 제어판에서 서비스를 비활성화한다.

## 9.2.2 리눅스에 에이전트 설치하기

### 에이전트 설치

리눅스용 Stackdriver 모니터링 에이전트와 로깅 에이전트를 설치하려면 다음

과 같이 설치 스크립트를 내려받아 실행한다.

```
// 모니터링 에이전트 설치  
$ curl -O https://repo.stackdriver.com/stack-install.sh  
$ sudo bash stack-install.sh --write-gcm  
  
// 로깅 에이전트 설치  
$ curl -sS0 https://dl.google.com/cloudagents/install-logging-agent.sh  
$ sudo bash install-logging-agent.sh
```

**주의**

- 에이전트는 root 사용자나 sudo 명령어를 사용하여 설치한다.
- 설치 스크립트 실행 시 --write-gcm 옵션을 포함했는지 확인한다.
- HTTP 프록시를 가지고 있는지 확인한다.
- VM에 private-key credentials를 설치했다면 VM에 credential을 저장했는지 확인한다.

## HTTP 프록시 사용

HTTP 프록시를 사용한다면 에이전트가 아웃바운드 HTTPS를 사용하여 모니터링 데이터를 전송할 수 있도록 구성해야 한다. 설치한 에이전트의 구성 파일에서 여러분의 HTTP 프록시를 가리키도록 PROXY\_URL을 변경하면 된다. 구성 파일의 위치와 이름은 리눅스 배포판에 따라 다르니 주의하자.

- 데비안, 우분투 : /etc/default/stackdriver-agent
- 아마존 리눅스, 레드햇, 센트OS : /etc/sysconfig/stackdriver

이제 에이전트를 다시 실행한다.

```
$ sudo service stackdriver-agent restart
```

## 에이전트 버전 결정, 업데이트, 삭제

이들 명령 역시 리눅스 배포판에 따라 조금씩 다르니 주의하라.

### 데비안, 우분투

```
// 버전 결정  
$ dpkg -l stackdriver-agent  
// 업데이트  
$ sudo apt-get update  
$ sudo apt-get install stackdriver-agent  
// 삭제  
$ sudo apt-get remove stackdriver-agent
```

---

### 아마존 리눅스, 레드햇, 센트OS

```
// 버전 결정  
$ sudo yum list stackdriver-agent  
// 업데이트  
$ sudo yum update stackdriver-agent  
// 삭제  
$ sudo yum remove stackdriver-agent
```

---

## 9.3 Uptime Check와 경보

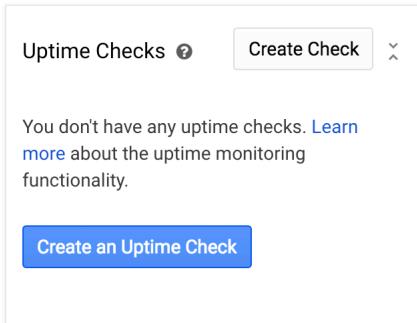
Uptime Check(가동 상태 점검)는 인터넷에 공개된 특정 서비스가 구동 중인지 (가용성)를 세계 각지로부터 확인해주는 기능이다.<sup>02</sup> 또한, 감시 대상에 이상이 생기면 경보Alert를 보낼 수 있다.

---

<sup>02</sup> 외부 IP가 없는 자원의 상태를 확인하려면 Uptime Check 서비스에서 들어오는 트래픽을 허용하도록 방화벽을 설정해야 한다.

Uptime Check는 Stackdriver 화면 왼쪽 위에 자리하며, 처음에는 [그림 9-3]과 같은 모습일 것이다.

그림 9-3 Stackdriver의 가동시간 확인 위젯(아직 확인 대상이 없다.)



그럼 먼저 Uptime Check를 생성하고, 특정 조건에서 경보를 주도록 설정해 보자.

### Uptime Check 생성

[그림 9-3] 위쪽의 [Create Check] 버튼을 클릭하면 [그림 9-4]와 같이 외부 서비스를 모니터링하기 위한 정보를 요구한다. 다음의 조건으로 8장에서 만든 testapp-001 인스턴스의 가용 여부를 알려주는 Uptime Check를 만들어보자.

- Title : testapp-001
- Check Type : HTTP
- Resource Type : Instance
- Applies To
  - Single
  - testapp-001

그림 9-4 새로운 Uptime Check 생성 화면

## New Uptime Check

Title  
testapp-001

Check Type  
HTTP

Resource Type  
Instance

Applies To  
Single testapp-001

Path  
/index.html  
Select protocol and enter the path component of the URL you want to check.

Check every  
1 minute

Advanced Options

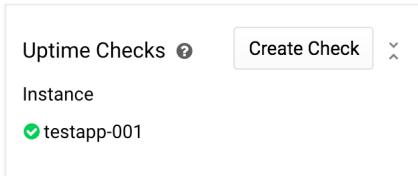
Save Test Cancel

여기서 [Check Type]은 HTTP, HTTPS, TCP와 같은 프로토콜 종류를 뜻하며, [Resource Type]은 검사할 대상의 유형으로, 그 종류와 의미는 다음과 같다.

- Instance: 컴퓨터 엔진 혹은 AWS EC2 인스턴스
- App Engine: 구글 앱 엔진 애플리케이션(모듈)
- Elastic Load Balancer
- URL : 특정 웹 호스트 이름 혹은 경로

설정이 잘 완료되고 대상 서비스가 가동 중이라면 잠시 후 녹색 표시가 들어온다. [그림 9-5]를 보면 Instance 유형의 testapp-001을 감시하고 있고, 현재 정상 동작 중임을 알 수 있다.

그림 9-5 검사 대상을 등록한 후의 Uptime Check 모습



[그림 9-5]에서 제목인 “Uptime Checks”를 클릭하면 상세정보가 나온다. [그림 9-6]을 보면 감시 대상인 testapp-001을 세계 각지의 총 6곳에서 확인하였고, 모두 성공했음을 알 수 있다.

그림 9-6 Uptime Check 상세정보

Alerting / Uptime Checks							Add Uptime Check	⋮
CHECKS	VIRGINIA	OREGON	IOWA	BELGIUM	SINGAPORE	SAO PAULO	POLICIES	ACTIONS
testapp-001	✓	✓	✓	✓	✓	✓	✓	<span>Edit</span> ⋮
Showing 1-1 of 1 item								

### 경보 정책 설정

이제 감시 대상에 문제가 있을 때 경보를 주도록 설정해보자. 앞서 [그림 9-6]에서 오른쪽 POLICES 열의 종 모양 아이콘이 회색인데, 경보 정책 Alerting Policy을 설정하지 않았기 때문이다. 이 아이콘을 클릭하면 [그림 9-7]과 같은 설정 화면이 나타난다. 여기서 정책 이름, 경보 발생 조건, 경보 방법(이메일, SMS, 슬랙 채널 등), 경보 메시지(조치 사항) 등을 설정하고 [Save Policy] 버튼을 클릭하면 정책이 만들어진다.

### 그림 9-7 경보 정책 설정 화면

Policy Name

testapp-001 Policy

Conditions ⓘ

**HTTP check on instance testapp-001** Edit : Delete  
violates when: Uptime Check Health on Instance testapp-001 fails

+ Add Condition 1 of 6 conditions added

Notifications ⓘ

METHOD

Email

Add Notification

EMAIL  remove

You will be notified of violations for this policy within the application.  
If you wish to also receive notifications by other methods, please add them here.

Documentation ⓘ

Optionally add a Markdown document here which describes your policy or offers instructions if an alert is triggered. These instructions will be included in alert emails.

Cancel Save Policy

경보가 정책에 맞게 잘 작동하는지 확인하려면 지정한 조건에 해당하는 상황을 만들면 된다. 예를 들어 서비스가 응답하지 않을 때 이메일을 보내도록 설정했다면, 감시 대상 VM을 잠시 중지시키고 경보 이메일이 날아오는지 확인해보자.

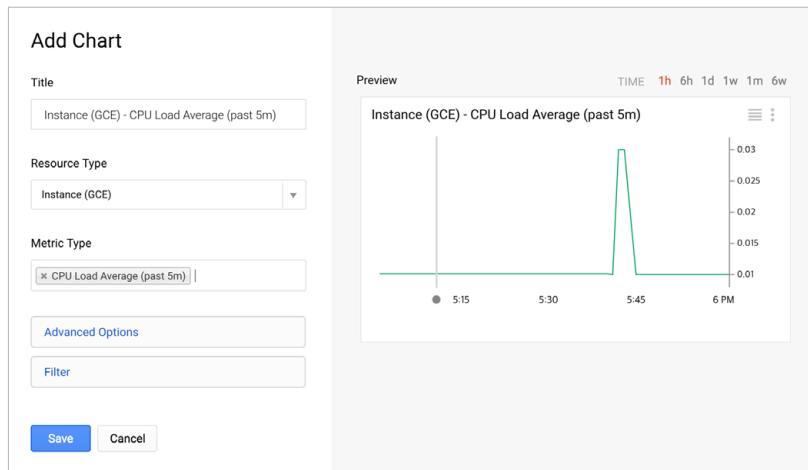
## 9.4 대시보드와 모니터링

### 9.4.1 대시보드

대시보드는 Stackdriver 에이전트가 수집한 측정항목들을 다양한 방식으로 표시한다. 원하는 자원이나 서비스별로 분류하여 대시보드의 모습을 입맛에 맞게 꾸밀 수 있다.

Stackdriver 모니터링 콘솔에서 [Dashboards] > [create..] 메뉴로 들어가 [Add Chart] 버튼을 클릭해보면 [그림 9-8]과 같은 차트 생성 화면이 나타난다. 차트 하나에 복수의 측정항목을 넣을 수 있으며, 오른쪽 Preview 창에서 만들어 질 차트 모양을 미리 확인해볼 수 있다.

그림 9-8 대시보드의 차트 추가 화면



[Resource Type]은 구글 클라우드 플랫폼이 제공하는 다양한 자원들과 AWS EC2, 로그 등의 측정항목을 지원한다. [Metric Type]은 크게 세 범주로 나뉜다. 먼저 모든 대상에 기본으로 제공되는 “클라우드 모니터링” 범주의 측정항목들이 있고, 다음으로 에이전트가 설치된 대상에서 가져올 수 있는 “에이전트”와 “윈도

우 에이전트” 범주가 있다. [Advanced Options]에서는 Threshold Line과 Aggregate Resources를 설정할 수 있고, 마지막으로 [Filter]를 이용해 원하는 자원과 관련된 데이터만 받아볼 수 있다.

### 9.4.2 클라우드 모니터링

Stackdriver는 별도 모니터링 에이전트를 설치하지 않아도 기본적인 모니터링을 수행할 수 있으며, 이를 클라우드 모니터링이라 한다. 클라우드 모니터링으로는 CPU Usage, Disk Read Write I/O, Network Inbound/Outbound 등 의 정보를 얻어올 수 있다.

### 9.4.3 에이전트 모니터링

에이전트를 설치하면 OS 자원과 특정 서비스의 정보를 수집해준다. 전용 에이전트를 설치한 만큼 클라우드 모니터링보다 세분화된 정보를 제공한다. 예를 들어 CPU의 경우 Load Average(1m, 5m, 15m), Steal, idle, interrupt, system 등 측정항목의 종류가 12개로 늘어난다.

## 9.5 로그 기록

로그는 미묘한 오류나 해커의 공격을 분석할 때 큰 도움을 주곤 한다. 구글 클라우드 플랫폼의 로깅은 Stackdriver 모니터링과 밀접하게 통합되어 있다.

로그를 보려면 GCP 콘솔에서 [로그 기록] > [로그] 메뉴로 가거나 Stackdriver 콘솔 위쪽 메뉴 중 [Logs]를 클릭하면 된다. 로그 기록은 갖가지 로그를 모두 모아 보여주기 때문에 훌륭한 필터링 기능을 내장하고 있다. 예를 들어 VM 인스턴스인 “testapp-001”가 기록한 로그를 보고 싶다면 [Compute Engine] > [instance] > [testapp-001]을 선택하면 된다(그림 9-9).

### 그림 9-9 구글 클라우드 플랫폼의 강력한 로깅 기능

The screenshot shows the Google Cloud Logging interface. At the top, there are dropdown menus for '라벨 또는 텍스트 검색 기준 필터링' (Label or text search filter), '모든 로그' (All logs), '모든 로그 수준' (All log levels), '이동할 날짜' (Move date), and a button with three dots. Below these are two main sections: '최근에 선택한 리소스' (Recently selected resources) and 'Compute Engine, 모든 리소스 유형, 모든 리소스 ID...'. The 'Compute Engine' section is expanded, showing categories like App Engine, Cloud Resource Manager, Cloud SQL, Compute Engine, IAM, and several log entries. To the right, a detailed view of the 'Compute Engine' category is shown with a search bar '프리픽스로 검색...' (Search by prefix...) and a list of resource IDs: nat-gw-as, nat-node-as-1, nat-node-as-2, testapp-001, followed by network, project, reservedAddress, route, and subnetnetwork.

그 외에도 유형별(syslog, activity\_log 등)과 수준별(중요, 오류, 주의, 정보, 디버그 등) 혹은 날짜별로 필터링해 볼 수 있고, 필터 오른쪽의 플레이 버튼(▶)을 누르면 이벤트를 실시간 스트리밍으로 받아볼 수 있다(그림 9-10).

### 그림 9-10 모든 Compute Engine 인스턴스의 '정보' 로그를 실시간 스트리밍으로 확인

The screenshot shows the Google Cloud Logging interface with a specific filter applied: 'Compute Engine, instance, 모든 리소스 ID'. The results show a series of log entries from 2016-08-02, all of which are of level '정보' (Info). The log entries are: 09:22:47.999 compute.instances.insert {"event\_timestamp\_us": "1470097367999208", "event\_..."}, 09:23:16.788 compute.instances.insert {"event\_timestamp\_us": "1470097396788967", "event\_..."}, 09:25:03.722 compute.instances.stop {"event\_timestamp\_us": "1470097503722555", "event\_ty..."}, 09:25:18.301 compute.instances.stop {"event\_timestamp\_us": "1470097518301446", "event\_ty..."}, 09:25:24.459 compute.instances.stop {"event\_timestamp\_us": "1470097524459840", "event\_ty..."}, 09:25:35.284 compute.instances.stop {"event\_timestamp\_us": "1470097535284327", "event\_ty..."}, 09:25:42.810 compute.instances.stop {"event\_timestamp\_us": "1470097542810551", "event\_ty..."}, and 09:25:45.546 compute.instances.stop {"event\_timestamp\_us": "1470097545546224", "event\_ty..."}.



# 구글 클라우드 스토리지 사용하기

최명근

구글 클라우드 플랫폼에서는 다양한 방식으로 데이터를 저장할 수 있다. 목적에 따라 관계형과 비관계형(NoSQL) 데이터베이스, 혹은 바이너리 파일을 저장하는 오브젝트 스토리지를 이용할 수 있다. 이번 장에서는 그중에서도 오브젝트 스토리지인 구글 클라우드 스토리지 Google Cloud Storage, GCS를 소개한다.

## 10.1 저장소 클래스

GCS는 속도와 가용성에 따라 크게 세 가지 저장소 클래스 storage class를 제공된다. 모든 클래스는 동일한 API를 사용하며 AWS의 API와도 호환된다. 각 버전의 특장점은 다음과 같다.

- **Standard** : 저장된 오브젝트에 빠르게 혹은 자주 액세스해야 할 때 적합하다. 예를 들어 웹사이트의 콘텐츠나 모바일 혹은 게임 애플리케이션을 위한 데이터를 저장하는 데 적합하다. 데이터 가용성 data availability이 가장 높다.
- **Durable Reduced Availability** : Standard 버전보다 저렴하며 데이터 가용성은 조금 낮다. 데이터 백업과 일괄 작업 batch job에 적합하다.
- **Nearline** : 가장 저렴하며 데이터 아카이빙 archiving이나 온라인 백업과 재해복구용으로 적합하다. AWS의 glacier와 비슷하다고 볼 수 있으나, 더 저렴하고 특히나 저장된 데이터를 가져오는 데 수 초 정도밖에 걸리지 않는다.

## 10.2 기본 개념

GCS를 사용하기 위해 기본적으로 알아야 할 개념들을 소개한다.

- **버킷**<sup>bucket</sup> : 데이터를 저장하기 위한 컨테이너다. GCS에 저장되는 모든 오브젝트는 반드시 버킷에 저장된다. 버킷을 통해 데이터를 관리하고 또 데이터로의 액세스도 제어한다. 버킷을 생성할 시에는 앞 절에서 언급한 저장소 종류와 위치<sup>location</sup>를 지정해야 하는데, 이를 설정은 버킷이 한 번 생성되면 변경할 수 없다.
- **오브젝트**<sup>object</sup> : GCS에 저장되는 각 데이터를 의미하며, 한 오브젝트의 최대 크기는 5TB다. 오브젝트는 object data와 object metadata로 구성된다. data는 주로 GCS에 저장하고자 하는 “파일”이 되며, metadata는 다양한 오브젝트를 설명하는 이름-값 쌍의 모음이다. 버킷 안에 생성할 수 있는 오브젝트 개수에는 제한이 없다.
- **오브젝트 불변성**<sup>object immutability</sup> : 업로드된 오브젝트는 스토리지의 생애<sup>lifetime</sup> 동안 변경될 수 없다. 생애라 함은 오브젝트가 생성되어 삭제될 때까지의 기간을 의미한다. 이는 곧 한 번 업로드된 오브젝트를 변경하기는 어렵다는 뜻이다. 하지만 덮어쓰기는 가능하다. 각 오브젝트는 초당 한 번의 업데이트 혹은 덮어쓰기가 가능하다. 같은 오브젝트를 초당 한 번 이상의 속도로 업데이트하려 하면 503 Service Unavailable 에러가 발생할 수 있다.
- **계층구조**<sup>hierarchy</sup> : GCS는 오브젝트를 저장하기 위해 기본적으로 flat 네임스페이스를 이용한다. 하지만 GCP 콘솔이나 gsutil을 이용하면 가상의 계층구조<sup>hierarchy</sup>로 저장할 수 있다.
- **네임스페이스**<sup>namespace</sup> : GCS에는 오직 하나의 네임스페이스가 있다. 즉, 모든 버킷은 전체 GCS 네임스페이스에 걸쳐서 유일한 이름을 가져야만 하고, 오브젝트 이름은 주어진 버킷 내에서 유일해야 한다.

## 10.3 버킷 생성

GCP 콘솔의 [Storage] > [브라우저] 메뉴로 들어가 화면 위쪽의 [버킷 생성]을 클릭하면 “버킷 만들기” 창이 나타난다(그림 10-1). 다음 값들을 입력하고 [만들기] 버튼을 클릭하자.

- 이름 : in\_my\_bucket
- 저장소 클래스 : Standard
- 위치 : asia-east1

그림 10-1 버킷 만들기 화면

The screenshot shows the 'Bucket creation' dialog. It includes fields for 'Name' (in\_my\_bucket), 'Storage class' (Standard), and 'Location' (asia-east1). A note about sensitive information in the name is displayed below the location field. At the bottom are 'Create' and 'Cancel' buttons.

그림 10-2 버킷 목록. 방금 만든 버킷을 확인할 수 있다.

The screenshot shows the 'Buckets' list page. It displays a single bucket named 'in\_my\_bucket' with details: Storage Class 'Standard', Location 'ASIA-EAST1', and a three-dot menu icon. Navigation links for 'Browser' (current), 'Create bucket', 'Search', and 'Delete' are at the top.

## 10.4 버킷 간 파일 전송

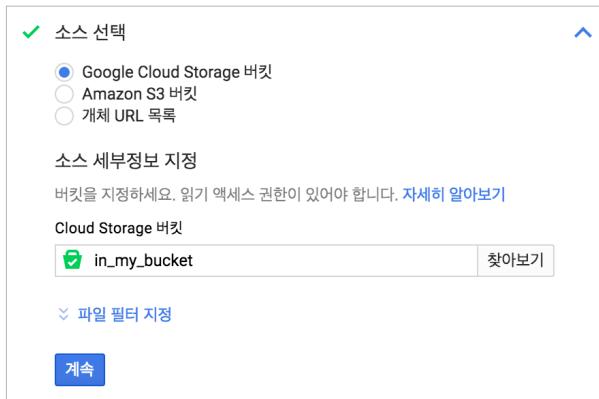
때에 따라 한 버킷의 오브젝트들을 다른 버킷으로 복사해야 할 필요가 있는데, 구글 클라우드 스토리지는 이럴 때를 위해 아주 편리한 “전송”이라는 기능을 제공한다. 전송은 “소스 선택”, “대상 선택”, “전송 구성”, 이렇게 세 단계로 진행한다.

10.3.1 절을 참고해 두 번째 버킷을 만들고, GCP 콘솔의 [Storage] > [전송] 메뉴로 가서 [전송 만들기]를 클릭한다.

### 1단계 : 소스 선택

소스란 파일을 가져올 버킷이다. 구글 클라우드 스토리지 외에 아마존 S3 버킷과 URL 목록도 소스로 사용할 수 있다. 파일 이름이나 마지막 수정 시간을 기준으로 특정 조건을 만족하는 파일만 옮기고 싶다면 추가로 파일 필터를 구성하면 된다.

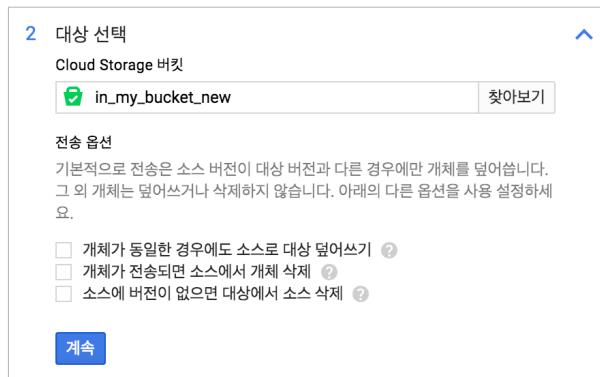
그림 10-3 전송 만들기 1/3 : 소스 선택



### 2단계 : 대상 선택

대상 버킷은 구글 클라우드 스토리지 버킷만 지원한다. [전송 옵션]으로 덮어쓰기나 전송 후 원본 삭제 등을 선택할 수 있다.

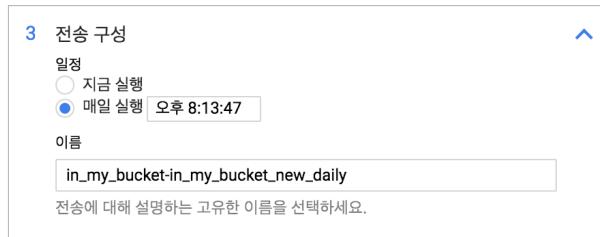
그림 10-4 전송 만들기 2/3 : 대상 선택



### 3단계 : 전송 구성

마지막으로 전송 일정과 전송의 이름을 정하고 [완료]를 클릭하면 된다. [일정]은 지금(1회)과 매일(특정 시간에 반복) 중 선택할 수 있다.

그림 10-5 전송 만들기 3/3 : 전송 구성



## 10.5 파일 업로드와 다운로드

GCP 콘솔의 웹 UI로도 버킷에 파일이나 폴더를 업로드할 수는 있지만, gsutil 을 이용하면 훨씬 편하다. 다음은 gsutil cp 명령어를 이용하여 로컬 컴퓨터의 cloud-storage-logo.png 파일을 my-awesome-bucket으로 복사하는 방법을 보여준다.

---

```
gsutil cp Desktop/cloud-storage-logo.png gs://my-awesome-bucket
```

---

성공하면 다음과 같은 메시지가 출력된다.

---

```
Copying file://Desktop/cloud-storage-logo.png [Content-Type=image/png]...
Uploading   gs://my-awesome-bucket/cloud-storage-logo.png:      0 B/2.58 KiB
Uploading   gs://my-awesome-bucket/cloud-storage-logo.png:  2.58 KiB/2.58 KiB
```

---

파일 업로드와 마찬가지로 `gsutil cp` 명령으로 특정 버킷에 있는 파일을 다운로드할 수 있다.

---

```
gsutil cp gs://my-awesome-bucket/cloud-storage-logo.png Desktop
```

---

성공하면 다음과 같은 메시지가 출력된다.

---

```
Copying gs://my-awesome-bucket/cloud-storage-logo.png...
Downloading file://Desktop/cloud-storage-logo.png:      0 B/2.58 KiB
Downloading file://Desktop/cloud-storage-logo.png:  2.58 KiB/2.58 KiB
```

---

## 10.6 파일 공개

버킷에 저장된 파일에 별도의 인증 없이 누구나 액세스할 수 있도록 설정할 수 있다(하지만 이 방식은 뜻하지 않게 외부에 노출될 우려가 있으니, 인증 없이도 액세스할 수 있도록 구성해야 할 때는 다음 절의 서명된 URL을 추천한다).

GCP 콘솔의 버킷 브라우저에서 해당 버킷에 들어가 공개할 파일의 “공개적으로 공유하기” 옆의 체크박스에 체크하면 된다. 그러면 [그림 10-6]에서 보는 바와 같이 “공개 링크”가 만들어지며, 이를 클릭하면 해당 링크로 이동한다.

### 그림 10-6 버킷 안의 파일 공개

버킷 / in_my_bucket					프리미엄으로 필터링...
□ 이름	크기	유형	최종 수정 시간	공개적으로 공유하기	⋮
<input type="checkbox"/> 9-9.png	160.49KB	image/png	16. 8. 2. 오후 8:23	<input checked="" type="checkbox"/> 공개 링크	

또한, gsutil acl ch 명령어로 설정할 수도 있다. 예를 들어 다음 명령은 누구든지 해당 버킷의 파일을 읽을 수 있도록 한다.

```
gsutil acl ch -u AllUsers:R gs://example-bucket/example-object
```

다음 명령은 누구든 쓰기가 가능한 버킷으로 설정한다.

```
gsutil acl ch -u AllUsers:W gs://example-bucket
```

## 10.7 서명된 URL을 통한 공유

구글 계정이 없는 사용자에게 클라우드 스토리지에 액세스하도록 하면서, 동시에 액세스 제어를 원하는 경우가 있다. 서명된 URL<sup>Signed URL</sup>은 버킷과 파일을 위한 쿼리 스트링 인증 메커니즘으로, 이러한 상황에서 사용할 수 있는 유일한 방법이다. 하나의 서명된 URL은 하나의 버킷 혹은 파일과 연결되며, 명시한 시간 동안 그 자원을 읽거나 쓸 수 있다. 적합한 액세스 권한을 갖는 URL을 가진 사람은 (구글 계정 소유 여부와 관계없이) 누구든 액세스할 수가 있다.

서명된 URL은 gsutil signurl 명령을 이용하면 쉽게 생성할 수 있다. 다음과 같이 gsutil signurl 명령을 이용하여 개인 키와 원하는 버킷, 혹은 파일의 URL을 명시한다.

```
gsutil signurl -d <지속시간> <개인_키_파일_경로> <버킷_혹은_파일의_URL>
```

예를 들어 다음 명령은 Desktop 폴더 안의 키를 사용하여 cat.jpeg 파일을 10분 동안 볼 수 있는 서명된 URL을 생성한다.

```
gsutil signurl -d 10m Desktop/private-key.json gs://example-bucket/cat.jpeg
```

성공하면 다음과 같은 메시지가 출력될 것이다.

URL	HTTP Method	Expiration	Signed URL
gs://example-bucket/cat.jpeg	GET	2016-03-17 11:17:10	<a href="https://storage.googleapis.com/example-bucket/cat.jpeg?GoogleAccessId=example@example-project.iam.gserviceaccount.com&amp;Expires=1458238630&amp;Signature=VVUgfqvIDCov%2B%2BKnmV0kwBR2o1SbId51kSibuQeiH8ucGFy0fAVbH5J%2B5V0gDYIioO2dDGH9Fsj6YdwxWv65HE71VE0EsVPUs8CVb%2BVeeIzmEe8z7X7o1d%2BcWbPEo4exILQbj3ROM3T20rkNBU9sbHq0mLbDMhiQZ3xCaICQdsrMEdYVvAFggPuPq%2FEQyQZmyJK3ty%2Bmr7kAFW16I9pD11jfbsD1XXjKTJzgd%2FMGSde4Va4J1RtHoX7r5i7YR7Mvf%2Fb17z1AuG1zVUf%2FzmhLPqtfKinVrcqdlmamMcmlow8eLG%2B1yYW%2F7t1s2hvqSfcw8eMUUjiHiSWgZLEVIG4Lw%3D%3D">https://storage.googleapis.com/example-bucket/cat.jpeg?GoogleAccessId=example@example-project.iam.gserviceaccount.com&amp;Expires=1458238630&amp;Signature=VVUgfqvIDCov%2B%2BKnmV0kwBR2o1SbId51kSibuQeiH8ucGFy0fAVbH5J%2B5V0gDYIioO2dDGH9Fsj6YdwxWv65HE71VE0EsVPUs8CVb%2BVeeIzmEe8z7X7o1d%2BcWbPEo4exILQbj3ROM3T20rkNBU9sbHq0mLbDMhiQZ3xCaICQdsrMEdYVvAFggPuPq%2FEQyQZmyJK3ty%2Bmr7kAFW16I9pD11jfbsD1XXjKTJzgd%2FMGSde4Va4J1RtHoX7r5i7YR7Mvf%2Fb17z1AuG1zVUf%2FzmhLPqtfKinVrcqdlmamMcmlow8eLG%2B1yYW%2F7t1s2hvqSfcw8eMUUjiHiSWgZLEVIG4Lw%3D%3D</a>

**NOTE** GCP 콘솔에서 개인 키 생성하기

1. GCP 콘솔에서 [API 관리자] > [사용자 인증 정보] 메뉴로 간다.
2. [사용자 인증 정보] 탭의 [사용자 인증 정보 만들기]를 클릭한다.
3. “서비스 계정 키”를 선택한다.  
“사용자 계정 키 만들기” 화면이 나타날 것이다.
4. [서비스 계정] 드롭다운 버튼을 클릭해 “새 서비스 계정”을 선택한다.
5. 서비스 계정의 이름을 입력한다.
6. “키 유형”을 선택한다. (JSON 혹은 P12)
7. [생성] 버튼을 클릭한다.

생성된 개인 키 파일이 자동으로 다운로드될 것이다.

## 10.8 스트리밍 전송

구글 클라우드 스토리지는 gsutil 도구나 boto 라이브러리를 통해 HTTP 전송 인코딩을 기반으로 한 스트리밍 전송을 지원한다. 이를 통해 로컬에 따로 버퍼링 하지 않고도 업로드할 수 있으며, 특정 계산이 필요한 파이프라인 시나리오에서 생성된 데이터를 바로바로 클라우드 스토리지로 보내려 할 때 유용하다.

### 10.8.1 gsutil을 이용한 스트리밍 업로드와 다운로드

gsutil을 이용하여 스트리밍 업로드를 수행하고자 한다면 데이터 파이프라인 을 gsutil cp 명령으로 연결하고 복사할 파일 대상을 “-”로 대체하면 된다. 다음 예는 collect\_measurements라는 프로세스의 출력을 클라우드 스토리지의 data\_measurements라는 파일로 스트리밍 전송하는 경우다.

---

```
collect_measurements | gsutil cp - gs://my_app_bucket/data_measurements
```

---

이와 비슷하게 다운로드는 다음과 같이 적용할 수 있다.

---

```
gsutil cp gs://bucket/object - | <process data>
```

---

### 10.8.2 boto를 이용한 스트리밍 업로드와 다운로드

boto 라이브러리를 사용해서도 스트리밍 업로드와 다운로드를 할 수 있다. 다음은 스트리밍 업로드를 수행하는 코드다.

---

```
dst_uri = boto.storage_uri(<bucket> + '/' + <object>, 'gs')
dst_uri.new_key().set_contents_from_stream(<stream object>)
```

---

예를 들어 다음 코드는 이름이 data\_file인 파일을 my\_app\_bucket 안의 같은 이름의 파일로 스트리밍 업로드한다.

---

```
filename = 'data_file'
MY_BUCKET = 'my_app_bucket'
my_data = open(filename, 'rb')
dst_uri = boto.storage_uri(MY_BUCKET + '/' + filename, 'gs')
dst_uri.new_key().set_contents_from_stream(my_data)
```

---

스트리밍 다운로드의 경우에는 다음 코드를 사용한다.

---

```
import sys
src_uri = boto.storage_uri(<bucket> + '/' + <object>, 'gs')
src_uri.get_key().get_file(sys.stdout)
```

---

예들 들어 다음 코드는 이름이 data\_file인 파일을 스트리밍 다운로드한다.

---

```
downloaded_file = 'saved_data_file'
MY_BUCKET = 'my_app_bucket'
object_name = 'data_file'
src_uri = boto.storage_uri(MY_BUCKET + '/' + object_name, 'gs')
src_uri.get_key().get_file(sys.stdout)
```

---

## 10.9 액세스 권한 관리

ACL (Access Control List)은 다른 사용자가 개별 버킷이나 오브젝트(파일)를 읽거나 쓸 수 있게 하는 메커니즘이다. ACL은 개별 객체 혹은 버킷별로 하나씩 만들어지며 각 ACL은 하나 이상의 엔트리로 구성된다. 엔트리는 특정 동작을 수행할 수 있는 능력을 특정 사용자 혹은 그룹에 부여한다. 각 엔트리는 다음의 정보로 구성된다.

- 항목 : 아래의 “이름”이 가리키는 대상의 유형을 정의한다. 유형은 사용자, 그룹, 도메인, 프로젝트가 있다.
- 이름 : 누가 수행할 수 있는지를 정의한다. 항목에서 정의한 유형에 따라 구글 계정 이름, 그룹 이름, 프로젝트 이름 등이 될 수 있다.

- 액세스 : 어떤 액션을 수행할 수 있는지를 정의한다. 크게 다음의 세 가지로 구분한다.

- 리더 : 읽기
- 작성자 : 쓰기(생성, 삭제)
- 소유자 : 권한 변경

예를 들어 특정 버킷의 권한을 수정하고 싶다면 GCP 콘솔에서 [Storage] > [브라우저] 메뉴에서 원하는 버킷을 찾아 오른쪽 추가 메뉴를 열고 “버킷 권한 수정”을 클릭한다(그림 10-7). 그러면 [그림 10-8]과 같은 수정 화면이 나타난다. 예를 들어 [그림 10-8]의 마지막 엔트리(사용자, allAuthenticatedUsers, 리더)는 인증된 모든 사용자가 이 버킷의 내용을 읽을 수 있도록 해준다.

**그림 10-7** 버킷 권한 수정 메뉴의 위치

버킷			프리픽스로 필터링...
	이름	저장소 클래스	위치
<input type="checkbox"/>	in_my_bucket	Standard	ASIA-EAST1
<input type="checkbox"/>	in_my_bucket_new	Standard	ASIA-E
<input type="checkbox"/>	net-test-1469982320762.appspot.com	Standard	US

⋮

버킷 권한 수정  
개체 기본 권한 수정

**그림 10-8** 버킷 권한 수정 화면

in\_my\_bucket 권한

이 권한은 버킷 자체에만 영향을 미치며 버킷의 기존 개체에는 적용되지 않습니다.

이 URL로 버킷 공유  
[https://console.cloud.google.com/storage/browser/in\\_my\\_bucket](https://console.cloud.google.com/storage/browser/in_my_bucket)

항목	이름	액세스
프로젝트	owners-430176305117	소유자
프로젝트	editors-430176305117	소유자
프로젝트	viewers-430176305117	리더
사용자	allAuthenticatedUsers	리더

**+ 항목 추가**

**저장** **취소**

하나의 버킷에 대해 생성할 수 있는 ACL 엔트리 수는 최대 100개다. 하나의 그룹은 그룹에 포함된 사용자 수와 상관없이 엔트리 하나로 취급된다. 권한이 없는 사용자가 특정 작업을 요청하면 “403 Forbidden” 에러가 발생한다.

## 10.10 스토리지 전송 서비스

10.4절에서 살펴본 스토리지 전송 서비스를 이용하면 데이터 소스로부터 구글 클라우드 스토리지로 데이터를 빠르게 이전할 수 있다. 이때 데이터 소스로는 아마존 S3 버킷과 HTTP/HTTPS 소스, 혹은 또 다른 구글 클라우드 스토리지 버킷이 될 수 있다. 따라서 다른 스토리지 제공자의 데이터를 구글 클라우드 스토리지로 백업하는 경우나, Standard 스토리지 버킷에서 Nearline 스토리지 버킷으로 이동하려 할 때 유용하게 사용할 수 있다.

크게 보면, 데이터 이전 방법은 gsutil과 스토리지 전송 서비스 두 가지로 볼 수 있는데, 보통은 다음의 기준에 따라 적절한 방법을 선택하면 된다.

- 데이터 전송량이 1TB 미만일 때 : gsutil
- 데이터 전송량이 10TB 이상일 때 : 스토리지 전송 서비스
- 데이터 전송량이 1TB에서 10TB 사이일 때 : gsutil 혹은 스토리지 전송 서비스

# 구글 클라우드의 차별성

조대협

이번 장에서는 1장에서 간략히 언급하고 넘어간 구글 클라우드의 차별화된 기능 몇 가지를 구체적으로 설명하여 더 자세히 이해할 수 있도록 돋고자 한다.

## 11.1 빅쿼리, 빅데이터 저장 및 분석 플랫폼

구글의 빅데이터 플랫폼은 Dataflow, Datalab 등 여러 가지가 있는데, 그중 가장 강력한 서비스라 할 수 있는 빅쿼리를 알아보자. 빅쿼리는 페타바이트급의 데이터 저장 및 분석용 클라우드 서비스다. 요즘은 페타바이트급의 데이터 웨어하우스로 부르는데, 쉽게 말해서 페타바이트 단위의 데이터를 저장해놓고, 쿼리를 통해서 조회나 통계 작업 등을 할 수 있는 데이터베이스다(사실 데이터베이스라고 보기에는 약간 애매하지만 말이다).

### 11.1.1 빅쿼리의 특징

대략적인 특징을 살펴보면 다음과 같다.

#### NoOps, 설치/운영이 필요 없다

어디에 설치해서 사용하는 서비스가 아니라 클라우드 서비스로 제공되는 빅데이터 저장 분석 서비스다. 클릭 몇 번으로 사용할 수 있고, 별도의 설정이나 운영이 필요 없다.

## SQL 언어 사용

RDBMS에서 사용하는 SQL을 그대로 지원하기 때문에 사용하기 매우 쉽다.

### 클라우드 규모의 인프라를 통한 대용량 지원과 빠른 성능

빅쿼리의 성능과 스케일은 <https://goo.gl/fDN8YE>의 예를 보면 짐작할 수 있다.

그림 11-1 빅쿼리 분석 예 (출처 : <https://goo.gl/fDN8YE>)

The screenshot shows the Google BigQuery Query Editor interface. The title bar says "100B Benchmark with 3 wildcards". The main area contains the following SQL code:

```
1 SELECT language, SUM(views) as views
2 FROM `bigquery-samples:wikipedia_benchmark.Wiki100B`
3 WHERE REGEXP_MATCH(title, "G.*o.*o.*g")
4 GROUP BY language
5 ORDER BY views desc;
```

Below the code, there's a message: "No Cached Results". At the bottom are several buttons: "RUN QUERY" (highlighted in red), "Save Query", "Save View", "Format Query", and "Show Options". A status message at the bottom says "Query complete (24.7s elapsed, 4.06 TB processed)". A green checkmark icon is in the bottom right corner.

이 예는 위키피디아에서 1,000억 개의 레코드를 스캔해서 정규표현식으로 “G.\*o.\*o.\*g” 문자열을 찾아내고, 그 문서의 뷰 수를 세고 있다. 대략 4TB 용량의 데이터가 처리되고, 약 30초가 소요된다. 30초 동안, 약 3,300개의 CPU와 330개의 하드디스크와 330Gb의 네트워크가 사용된다. (출처 : <https://goo.gl/ptCDhC>)

이 쿼리를 수행하는 데 소요되는 비용은 딱 \$20다. 일반적인 인프라에서 빅데이터 연산을 하는데 3,300개의 CPU를 동시에 사용하기란 말처럼 쉽지 않은 일이고, 이런 대용량 연산을 단돈 \$20에 처리해주는 것은 대용량 인프라를 공유하는 클라우드 서비스이기에 가능하다.

### 데이터 복제를 통한 안정성

데이터는 3개로 복제하여 서로 다른 3개의 데이터센터에 분산 저장하기 때문에 유실 위험이 적다.

## 배치와 스트리밍 모두 지원

한꺼번에 데이터를 로딩하는 일괄 작업<sup>batch</sup> 외에도 REST API 등을 통해서 실시간으로 데이터를 입력할 수 있는 스트리밍 기능을 제공한다. 스트리밍 시에는 데이터를 초당 100,000 행<sup>row</sup>씩 입력할 수 있다.

## 비용 정책

비용 정책 역시 클라우드 서비스답게 딱 저장되는 데이터 크기와 쿼리 시에 발생하는 트렌잭션 비용만큼만 과금된다. 데이터 저장 요금은 GB당 \$0.02이고, 90 일이 지나서 사용하지 않는 데이터는 자동으로 \$0.01로 떨어진다. 일반적으로 가격이 싸다고 알려진 오브젝트 스토리지보다 싸다(구글 클라우드 스토리지는 GB 당 \$0.026다). 트렌잭션 비용은 쿼리 수행 시 스캔되는 데이터를 기준으로 TB 당 \$5다(월 1TB는 무료). 나중에 자세하게 설명하겠지만, 스캔되는 컬럼당 비용이 나오기 때문에 사실상 비용을 계산해보면 그리 높지 않다. 자세한 가격 정책은 <https://cloud.google.com/bigquery/pricing>을 참조하자.

### 11.1.2 기존 빅데이터 플랫폼과 다른점

그렇다면 빅쿼리가 기존의 빅데이터 분석 플랫폼인 하둡이나 스파크 등과는 어떻게 다를까? 앞의 장점을 기반으로 그 차이점을 정리하자면 크게 다음의 3가지를 들 수 있다.

- 쉽다. 하둡이나 스파크로 분석할 때는 맵리듀스 로직을 사용하거나 Spark SQL을 사용하는데, 이 방식은 일정 수준 이상의 전문성이 필요하다. 특히 맵리듀스 로직의 경우 전문성 있는 개발자가 분석 로직을 개발해야 하기 때문에 시간이 상대적으로 오래 걸린다. 반면, 빅쿼리는 로그인하고 SQL만 수행하면 되니 상대적으로 빅데이터 분석이 쉽다.
- 운영이 필요 없다. 하둡이나 스파크와 같은 빅데이터 솔루션은 설치와 설정, 그리고 클러스터를 유지보수하기가 보통 일이 아니다. 그래서 별도의 운영 조직이 필요하고 여기에 많은 자원이 소모된다. 하지만 빅쿼리는 클라우드 서비스기 때문에 이러한 일들은 잊고 개발과 분석에만 집중하면 된다.

- 인프라 투자 없이 막강한 컴퓨팅 자원을 활용한다. 앞의 예에서 본 것처럼, 빅쿼리를 이용하면 수천 개의 CPU와 수백/수천 개의 컴퓨팅 자원을 사용할 수 있다. 물론 기존 빅데이터 플랫폼도 클라우드 환경에 올리면 가능한 일이지만, 그 설정 노력과 비용 측면에서 차이가 크다.

### 11.1.3 빅쿼리 맛보기

그러면 이제부터 직접 빅쿼리를 사용해보자. 빅쿼리 버전 HelloWorld라고 생각하면 된다.

#### 1단계 : 가입하기

빅쿼리를 사용하려면 물론 구글 클라우드 서비스에 가입하고 로그인해야 한다. 관련 방법은 2장을 참고하자.

#### 2단계 : 빅쿼리 콘솔로 이동하기

GCP 콘솔에서 [BigQuery] 메뉴로 들어가면 [그림 11-2]와 비슷한 작업 창이 나오다.

그림 11-2 GCP 콘솔의 빅쿼리 작업 화면

The screenshot shows the Google BigQuery web interface. On the left, there's a sidebar with navigation links like 'COMPOSE QUERY', 'Query History', and 'Job History'. Below that, it says 'net-test' and '프로젝트명'. Under 'Public Datasets', there are several entries: 'bigquery-public-data:hacker\_news', 'bigquery-public-data:noaa\_gsod', 'bigquery-public-data:samples', 'github\_nested', 'github\_timeline', 'gsod', 'natality', 'shakespeare', 'trigrams', and 'wikipedia'. The main area has a 'New Query' tab open, showing a code editor with the following SQL query:

```
1 SELECT FROM [bigquery-public-data:samples.wikipedia] LIMIT 1000
```

The code editor is labeled '쿼리 입력창' (Query Input Box). Below the editor are buttons for 'RUN QUERY', 'Save Query', 'Save View', 'Format Query', and 'Show Options'. A red box highlights the 'RUN QUERY' button. At the bottom, there's a section titled 'Table Details: wikipedia' with tabs for 'Schema', 'Details', and 'Preview'. The 'Schema' tab is selected, showing the table structure:

	Schema	Details	Preview	테이블 정보와 쿼리 결과
<b>title</b>	STRING	REQUIRED		The title of the page, as displayed on the page (not in the URL). Always starts with a capital letter and may begin with a namespace (e.g., "Talk:", "User:", "User Talk:", ...)
<b>id</b>	INTEGER	NULLABLE		A unique ID for the article that was revised. These correspond to the order in which articles were created, except for the first several thousand IDs, which are issued in alphabetical order.
<b>language</b>	STRING	REQUIRED		Empty in the current dataset.

왼쪽은 프로젝트와 프로젝트에 속한 데이터셋(dataset)과 테이블 목록이 나온다. 데이터셋은 RDBMS의 db와 비슷한 개념으로, 테이블의 집합이라고 보면 되고 그 안에 개별 테이블들이 들어가 있다(데이터 모델은 나중에 다시 설명하겠다). 오른쪽 위의 쿼리 입력창에는 왼쪽의 [COMPOSE QUERY] 버튼을 클릭하면 나타나고, 이 곳에 SQL을 입력해서 실행하면 쿼리 결과가 그 아래 영역에 나타난다.

### 3단계 : 쿼리 실행

빅쿼리에서는 테스트를 위해서 데이터셋 몇 개를 공개해놓았는데, [그림 11-2] 왼쪽 아래의 “Public Datasets”가 바로 그것이다. 이 데이터셋을 이용해 실제로 간단한 쿼리를 수행해보자.

bigquery-samples:wikipedia\_benchmark라는 데이터셋에서 레코드가 1,000억 개에 이르는 Wiki100B 테이블에서 제목(title)이 “Seoul” 또는 “seoul”인 페이지들을 뽑아 뷰 수(<sup>views</sup>)를 기준으로 정렬해볼 것이다. 이를 위한 쿼리문은 다음과 같다.

---

```
select title,sum/views) as views
from [bigquery-samples:wikipedia_benchmark.Wiki100B]
where regexp_match(title,'[Ss]eoul')
group by title
order by views desc;
```

---

이 코드를 쿼리 입력창에 입력하고 잠시 기다리면 쿼리 입력창이 [그림 11-3]과 같이 될 것이다. 중간의 녹색 테두리 상자에 “Valid: This query will process 3.64 TB when run.”이란 메시지를 볼 수 있다. 이처럼 빅쿼리는 입력한 쿼리문이 문법적으로 올바른지(잘못되었다면 관련 오류 메시지를 보여준다), 스캔할 데이터의 용량은 얼마나 되는지를 알려준다. 이를 통해 쿼리 수행 비용을 예측해볼 수 있다.

그림 11-3 쿼리문을 입력한 모습

New Query ? Query Editor UDF Editor X

```
1 select title, sum/views) as views
2 from [bigquery-public-data:samples.wikipedia]
3 where regexp_match(title, '[Ss]eoul')
4 group by title
5 order by views desc;
```

Valid: This query will process 6.79 GB when run.

RUN QUERY ▼ Save Query Save View Format Query Show Options Query complete ✓

[Run Query] 버튼을 눌러서 쿼리를 수행하면 다음과 같은 결과를 얻을 수 있다.

그림 11-4 쿼리 수행 결과

New Query ? Query Editor UDF Editor X

```
1 select title,sum/views) as views
2 from [bigquery-public-data:samples.wikipedia]
3 where regexp_match(title, '[Ss]eoul')
4 group by title
5 order by views desc;
6
```

RUN QUERY ▼ Save Query Save View Format Query Show Options Query complete (38.9s elapsed, 3.64 TB processed) ✓

Results Explanation Download as CSV Download as JSON Save as Table Save to Google Sheets

We can notify you when long-running queries complete. [Enable notifications](#) [Don't ask again](#) X

Row	title	views
1	Seoul	11258720
2	Seoul_National_University	894040
3	FC_Seoul	802570
4	File:Seoul_landmark_picture_4.jpg	729090
5	Seoul_Metropolitan_Subway	716090
6	Seoul_Broadcasting_System	694380
7	File:Starcraft_II_Commercial_on_Korean_Air_-Seoul_Incheon_Airport.JPG	526000
8	Seoul_World_Cup_Stadium	497660
9	Seoul_National_Capital_Area	425730
10	File:Korea-Seoul-Cheonggyecheon-2008-01.jpg	417030
11	K-1_World_Grand_Prix_2010_in_Seoul_Final_16	390790

Table JSON First < Prev Rows 1 - 11 of 36981 Next > Last

[RUN QUERY] 버튼 오른쪽 끝을 보면, 총 3.64TB를 처리했고 38.9초가 걸렸음을 확인할 수 있다. 그리고 아래쪽에 쿼리 결과가 나타난다. 제목이 “Seoul”

인 페이지가 11,258,720회 조회되었고, “Seoul\_National\_University”가 894,040회, “FC\_Seoul”이 802,570회 조회된 것을 확인할 수 있다.

이상으로 아주 간단하게나마 빅쿼리에 대해서 알아보았다. 더 자세한 내용은 구글 문서를 참고하기 바란다.

## 11.2 Cloud Vision API, 머신 러닝 서비스

이미지 인식 기능을 제공하는 Cloud Vision API는 음성 인식, 번역 등과 함께 구글이 제공하는 강력한 인공 지능 서비스다. 이번 절에서는 Node.js를 이용하여 간단한 이미지를 분석해보며 이 API의 사용법을 알아보고자 한다.

### 11.2.1 Cloud Vision API 맛보기

#### 1단계 : API 활성화

API 호출 전에 이 기능을 사용할 수 있도록 활성화해보자. GCP 콘솔의 [API 관리자] > [라이브러리] 메뉴로 들어가 API 검색 창에 “vision”이라고 입력하면 Cloud Vision API를 찾아줄 것이다(그림 11-5).

그림 11-5 API 관리자에서 Vision API 찾기

API	API 관리자	라이브러리				
	대시보드	<a href="#">Google API</a>				
	라이브러리	<input type="text" value="vision"/> <a href="#">인기 API로 돌아가기</a>				
	사용자 인증 정보	<table><tr><td>이름</td><td>설명</td></tr><tr><td><a href="#">Google Cloud Vision API</a></td><td>Integrates Google <b>Vision</b> features, including image labeling, face, logo, and landmark detection, optical character recognition (OCR), and detection of explicit content, into applications.</td></tr></table>	이름	설명	<a href="#">Google Cloud Vision API</a>	Integrates Google <b>Vision</b> features, including image labeling, face, logo, and landmark detection, optical character recognition (OCR), and detection of explicit content, into applications.
이름	설명					
<a href="#">Google Cloud Vision API</a>	Integrates Google <b>Vision</b> features, including image labeling, face, logo, and landmark detection, optical character recognition (OCR), and detection of explicit content, into applications.					

찾은 API 이름을 클릭하면 API 개요 페이지가 나타나는데, 가장 위에 [▶ 사용 설정] 버튼이 보일 것이다. 이를 클릭하면 Cloud Vision API가 활성화된다.

## 그림 11-6 Google Cloud Vision API 개요 화면

The screenshot shows the Google Cloud Vision API overview page. At the top, there are navigation links: 'Google Cloud Vision API' (with a back arrow), '사용 설정' (with a forward arrow), and a '문서' link. Below this is a section titled 'API 정보' with the sub-section '이 API를 통해 사용자 인증 정보 사용'. It explains OAuth 2.0 authentication and includes a diagram illustrating the process:

**OAuth 2.0으로 사용자 데이터 액세스**

이 API를 사용자 데이터에 액세스할 수 있습니다. 사용자 인증 정보 페이에서 OAuth 2.0 클라이언트 ID는 앱에서 사용자 데이터에 액세스할 수 있도록 사용자 동의를 요청합니다. Google로 API 호출 시 클라이언트 ID를 포함하세요. [자세히 알아보기](#)

The diagram illustrates the OAuth 2.0 flow:

- 앱**: A purple circle containing a white '2' labeled 'AUTH'.
- 사용자 등의**: A laptop and a smartphone, both showing a green checkmark icon.
- 사용자 데이터**: A blue folder icon with a person silhouette.
- 서버 간 상호작용**: A note explaining that the API can be used with mobile applications and Google services.
- 사용자의 서비스**: Two server icons.
- 승인**: A teal rectangular box with a yellow circular icon containing a magnifying glass.
- Google 서비스**: Two server icons.

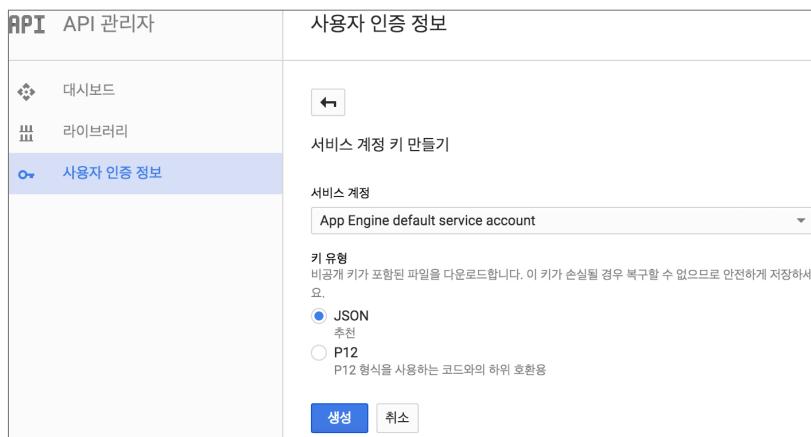
Arrows indicate the flow from the app to the user, from the user to the user data, and from the user data back to the app. Arrows also show the interaction between the app and the user's service, and between the user's service and Google's service.

## 2단계 : API 키 생성

다음으로 API를 외부에서 호출하기 위한 API 키를 발급받자. 구글 클라우드 플랫폼 API에서는 API에 접근할 수 있는 다양한 방식을 제공한다. OAuth 방식, 서버를 위한 API 키 방식 등을 지원하는데, 이 예제에서는 서비스 계정 키를 JSON 형태로 내려받아 사용하도록 한다.

계정 키는 GCP 콘솔의 [API 관리자] > [사용자 인증 정보] > [사용자 인증 정보 만들기] > [서비스 계정 키] 메뉴를 통해서 생성하여 JSON 파일 형태로 내려받을 수 있다.

그림 11-7 API 호출용 서비스 계정 키 생성 화면



얼굴 검출 튜토리얼(<https://cloud.google.com/vision/docs/face-tutorial>)을 보면 Cloud Vision API로 얼굴을 찾는 방법이 자세하게 설명되어 있다. 제공하는 기능에 비해서 API 사용법은 매우 간단한데, 요청을 JSON으로 만들어 REST 방식으로 API를 호출하면 분석 결과를 다시 JSON으로 반환한다. 어렵지 않으니 문서를 한 번 쭉 보면서 흐름을 따라가 보기 좋다.

### 3단계 : Node.js 모듈 준비하기

이제 Node.js에서 이미지를 분석하는 예제를 만들어보자. Node.js에서 Cloud Vision API를 사용하려면 gcloud 모듈을 설치해야 한다. 다음과 같이 설치해 보자.

---

```
$ npm install gcloud
```

---

이제 Cloud Vision API를 호출하기 위한 준비가 끝났다.

## 4단계 : Node.js로 Cloud Vision API 호출하기

이제 gcloud 모듈을 이용하여 API를 호출해보자.

---

```
var gcloud = require('gcloud');
var vision = gcloud.vision();
var fs = require('fs');
var types = [
  'face',
  'label'
]

// 요청 페이로드를 생성한다.
console.log("processing "+process.argv[2]);
inputFile = process.argv[2];

vision.detect(inputFile,types,function(err,result){
  if(err){
    console.log("ERROR:");
    console.log(err);
    return;
  }
  console.log("PASS");
  console.log("write to file");
  fs.writeFile("./"+inputFile+".vision",JSON.stringify(result),function(err){
    if (err){
      console.log("file write error "+err);
    }
  });
});
```

---

이 코드는 이미지 파일의 이름을 입력받아, 해당 이미지를 분석하여 그 결과를 \*.vision이라는 파일로 저장해주는 예제다. types 변수에 인식하고자 하는 정보 유형(얼굴, 라벨, 랜드마크, 로고, 유해 사진, 텍스트)을 정의한다. 이 예제는 얼굴(face)과 라벨(label) 검출만을 지정하였다.

이제 다음 명령을 실행해보자.

---

```
$ node visionAPI.js <분석할_이미지_파일명>
```

---

실행 결과는 <분석할\_이미지\_파일명>.vision이라는 파일에 JSON 형태로 저장해준다. 저자가 테스트에 사용한 이미지와 결과는 다음과 같다.

그림 11-8 테스트에서 사용한 이미지



---

```
{
  "responses": [
    {
      "faceAnnotations": [
        {
          "boundingPoly": {
            "vertices": [
              {
                "x": 122,
                "y": 52
              },
              {
                "x": 674,
                "y": 52
              },
              {
                "x": 674,
                "y": 693
              },
              {
                "x": 122,
                "y": 693
              }
            ]
          }
        }
      ]
    }
  ]
}
```

---

```
        "x":122,
        "y":693
    }
]
},
"fdBoundingPoly":{
    "vertices":[
        {
            "x":176,
            "y":208
        },
        ...
    ]
},
"landmarks":[
    {
        "type":"LEFT_EYE",
        "position":{
            "x":282.99844,
            "y":351.67017,
            "z":-0.0033840234
        }
    },
    {
        "type":"RIGHT_EYE",
        "position":{
            "x":443.8624,
            "y":336.31445,
            "z":-35.029751
        }
    },
    ...
],
"rollAngle":-3.8402841,
"panAngle":-12.196975,
"tiltAngle":-0.68598062,
"detectionConfidence":0.8096019,
"landmarkingConfidence":0.64295566,
"joyLikelihood":"LIKELY",
"sorrowLikelihood":"VERY_UNLIKELY",
"angerLikelihood":"VERY_UNLIKELY",
```

```
        "surpriseLikelihood":"VERY_UNLIKELY",
        "underExposedLikelihood":"VERY_UNLIKELY",
        "blurredLikelihood":"VERY_UNLIKELY",
        "headwearLikelihood":"VERY_UNLIKELY"
    }
],
"labelAnnotations": [
{
    "mid":"/m/068jd",
    "description":"photograph",
    "score":0.92346138
},
{
    "mid":"/m/09jwl",
    "description":"musician",
    "score":0.86925673
}
]
}
}
```

---

결과를 살펴보면 눈, 코, 입의 위치와 감정 상태 등을 상세하게 알려줌을 알 수 있다. joyLikeihood는 기쁨, sorrowLikeihood는 슬픔을 나타낸다.

Cloud Vision API는 여러 특성<sup>feature</sup>을 동시에 분석해줄 수 있다. 예를 들어 얼굴 인식과 로고 인식을 같이 활용하여 사람들이 특정 브랜드를 보고 있을 때의 표정을 분석하여 브랜드에 대한 대략적인 반응을 인식한다든지, 랜드마크와 표정 분석을 통해서 그 장소가 재미있는 곳인지 슬픈 곳인지 등을 알아내는 식으로 활용 할 수 있다.

### 11.2.2 다양한 이미지 분석 기능

Cloud Vision API는 여러 형태의 이미지 분석을 제공하는데, 대략적으로 훑어 보면 다음과 같다.

- Label Detection : 사진에서 사물을 찾아준다. (가구, 동물, 음식 등)
- Logo Detection : 사진에서 특정 로고를 찾아준다. (회사 로고 등)
- Landmark Detection : 사진에서 유명한 랜드마크를 찾아준다. (남산 타워, 경복궁 등과 같은 건축물이나 자연 경관 이름 등)
- Face Detection : 사진에서 사람 얼굴을 찾아준다. 이게 좀 재미있는데 눈, 코, 입의 위치 등은 물론 표정을 분석하여 감정 상태를 분석하여 반환한다. (화남, 기쁨, 슬픔 등)
- Explicit Content Detection : 사진 속 콘텐츠의 위험도(또는 건전성)를 검출한다. (성인 콘텐츠, 의학 콘텐츠, 폭력 콘텐츠 등의 정도)
- Optical Character Recognition : 문자 인식

더 자세한 내용은 <https://cloud.google.com/vision/> 문서를 참고하기 바란다.

### 11.2.3 Cloud Vision API의 또 다른 의미

Cloud Vision API는 그 자체로만으로도 대단히 흥미롭지만, 또 다른 의미를 가지고 있다고 본다. 한때 머신 러닝, 빅데이터, 인공 지능은 대규모 하드웨어 자원을 갖춘 업체와 특정 데이터 과학자의 전유물이었지만, 이제는 다양한 빅데이터 기술이 클라우드를 기반으로 서비스 형태로 제공되어 누구나 쉽게 활용할 수 있는 시대가 되어가고 있다. 이미 구글이나 마이크로소프트는 머신 러닝 알고리즘을 클라우드 API로 제공하기 시작했고, 대규모 데이터 분석 쪽도 구글의 웹로그 분석 Analytics이나 애후!의 Flurry Analytics 등을 통해서 거의 무료로 사용할 수 있다. 코드 몇 줄이면 앱에 추가할 수도 있다.

## 11.3 합리적인 가격 정책

구글 클라우드는 대단히 합리적인 가격 정책을 제공하여 클라우드 서비스를 상대적으로 저렴하게 이용할 수 있도록 해준다. 이번 절에서는 이 가격 정책에 대해서 자세하게 알아보자.

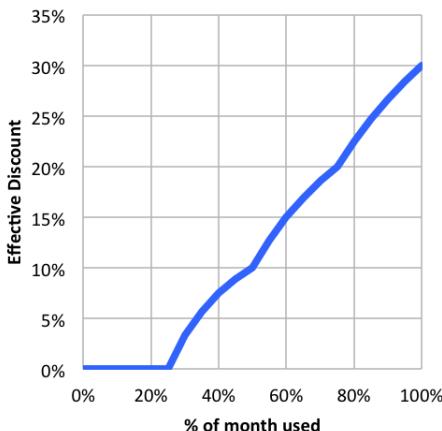
### 11.3.1 분 단위 과금

구글 클라우드의 인스턴스 사용료는 분 단위로 과금된다. 최초 10분은 항시 과금이 되고, 그 후의 사용량에 대해서는 1분 단위로 과금된다. 다른 회사의 클라우드 서비스는 시간 단위 과금이 많은데, 그렇다 보니 1시간 1분을 사용하더라도 고스란히 2시간 치 비용이 청구된다. 특히 하둡이나 배치 작업은 일정 시간에만 구동되고 작업을 완료한 후에는 아무 일도 하지 않기 때문에 사용 시간을 조밀하게 나눠야 사용자에게 유리하다. 그래서 분 단위로 계산되는 구글 클라우드는 짧게 사용하는 배치나 분산 연산 작업에 효율적이다.

### 11.3.2 장기 계약 없이 알아서 깎아준다

재미있는 정책 중 하나로, 인스턴스를 일정 시간 사용하면 자동으로 할인해준다. 인스턴스 가동률이 한 달 중 25%를 넘어서기 시작하면 단계적으로 할인율이 높아져, 한 달 내내 사용하면 최대 30%까지 할인된다. 자동으로!!

그림 11-9 사용량에 따른 할인률



이게 왜 중요할까? 보통 다른 클라우드에서는 사용할 인스턴스의 유형과 기간을 미리 명시해야 가격을 할인해주는데, 이런 방식은 몇 가지 문제가 있다.

첫 번째, 장기 계약 시 시간이 지나도 구세대 인스턴스를 계속 사용해야 한다. 예컨대 3년 계약으로 할인을 받으면 3년 동안 계속 구세대 인스턴스를 사용한다는 이야기다. 클라우드 서비스의 인스턴스 성능이 나날이 발전하기 때문에, 계약 당시 싸다고 느껴졌던 비용도 (시간이 지날수록 상대적으로 저성능 인스턴스를 사용하는 형태라) 과연 정말로 이득을 보는 것인지 고민해봐야 한다.

두 번째, 장기로 계약하면 무조건 그 인스턴스를 계속해서 사용해야 한다. 유동적인 비즈니스 상황에 맞춰 다른 인스턴스로 바꾸거나 인스턴스 수를 줄일 수 없다.

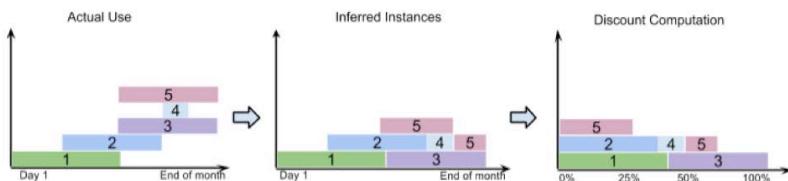
구글 클라우드를 사용한다면 이런 고민 없이 최신이든 구형이든 인스턴스를 그냥 쓰면 사용한 양에 따라서 알아서 할인된다.

### 11.3.3 여러 인스턴스의 사용량을 자동으로 합산해서 깎아준다

이것도 재미있는 할인 정책이다. 앞 절의 설명에 따르면 30% 할인을 받으려면 한 달 내내 써야 하는데, 만약 인스턴스 A를 30%, 인스턴스 B를 30%, 인스턴스 C를 20%, 인스턴스 D를 20% 기간만 쓴다면 어떻게 될까? 100%가 아니니 안 될 거 같지만, 정답은 “된다”다.

구글 클라우드는 총 사용량을 모아서 할인해준다. 즉 A, B, C, D의 총 사용량이 월 100%가 되기 때문에, 네 인스턴스 금액 전체의 30%를 할인해준다.

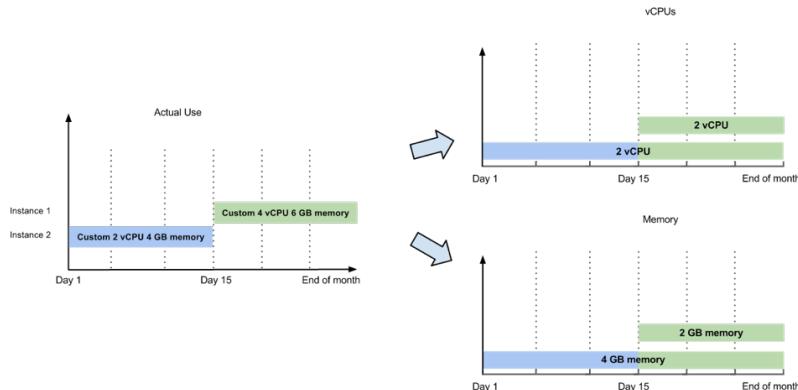
그림 11-10 다수 인스턴스의 사용량 합산 방식



그렇다면 인스턴스 유형이 달라도 가능할까? 답은 역시 “된다”다. 원리는 간단하다. 작은 인스턴스 스펙으로 잘라서 이 사용량을 뷰은 후 할인하는 게 핵심이다.

다음 그림처럼 CPU 2개에 메모리 4GB의 인스턴스를 50%, CPU 4개에 메모리 6GB인 인스턴스를 50% 사용한다고 해보자. 그러면 전체 사용량을 둘 중 더 작은 2CPU/4GB 인스턴스에 맞춰서 합산해 이를 100%로 계산하고, 남은 사용량은 2CPU/2GB 인스턴스를 50% 사용한 것으로 별도 과금하는 방식이다.

그림 11-11 유형이 다른 인스턴스들의 사용량 합산 방식



### 11.3.4 커스텀 인스턴스

미리 정해진 인스턴스 유형뿐 아니라 필요한 인스턴스의 스펙을 직접 정해서 사용 할 수 있다. 필요한 만큼의 CPU와 메모리를 지정할 수 있으니 특정 자원만 많이 필요함에도 불필요하게 큰 인스턴스를 사용할 일이 없어진다. 딱 필요한 크기로 적용하고 그에 해당하는 금액만 지불하면 된다.

### 11.3.5 Preemptible VM

Preemptible 유형 인스턴스는 일반 가격의 20%만으로 사용할 수 있는 VM으로, 최대 24시간 연속 사용이 가능하고 그 후에는 자동으로 셧다운된다(물론 다시 생성할 수 있다). 구글이 클라우드 자원 중 남는 부분을 저가로 제공하는 서비스로, 여러분이 사용 중이라도 구글이 요청하면 자동으로 셧다운될 수 있다. 그래

서 preemptible(선점할 수 있는)이다. 이러한 특성 때문에 이 VM에서 구동하는 서비스는 언제든 재시작이 가능한 구조로 구현하는 것이 중요하다. 대신 배치 연산과 대규모 데이터 분석 작업 등을 저렴하게 수행할 수 있다는 장점이 있다.

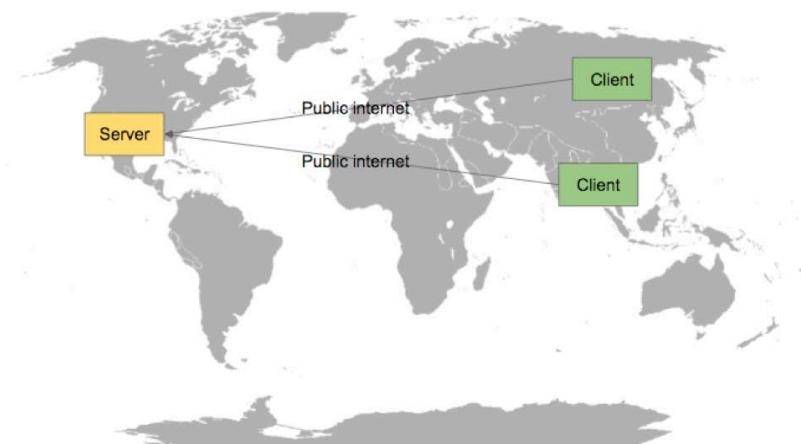
정리하자면, 구글 클라우드는 돈을 뜯어내는 게 아니라 되도록이면 할인을 해주려 노력한다. 이거저거 생각할 필요없이 그냥 쓰면 최적화된 과금 방법 찾아서 알아서 할인해준다. 과금에 대한 자세한 내용은 [https://cloud.google.com/compute/pricing#sustained\\_use](https://cloud.google.com/compute/pricing#sustained_use) 문서를 참고하기 바란다.

## 11.4 글로벌 광케이블을 이용한 네트워크 가속화

구글은 전 세계적으로 수많은 네트워크 망을 가진 것으로 유명하고, 특히나 구글의 데이터센터들은 광케이블 기반의 백본 네트워크로 묶여 있다.

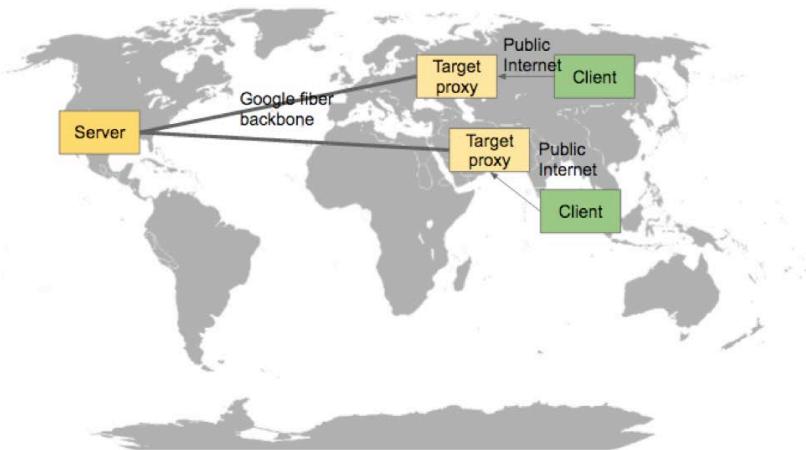
일반적인 글로벌 서비스에서의 클라이언트와 서버는 [그림 11-12]처럼 일반 인터넷망을 통해서 서로 통신한다. 따라서 글로벌하게 배포되어 있는 서비스는 거리가 멀어지면 반응 속도도 느려진다.

그림 11-12 일반적인 인터넷망을 통해 통신하는 서버와 클라이언트



구글은 데이터센터 외에 여러 개의 거점에 Target Proxy라는 일종의 중계 서버를 배치해놨고, 이 Target Proxy에서 각 데이터센터까지는 광케이블로 연결해 두었다. 구글 클라우드의 글로벌 로드 밸런서를 사용하면 글로벌 IP라는 것을 사용할 수 있다. 글로벌 IP는 Anycast(<http://goo.gl/cRdhyL>) 기반의 IP로, 클라이언트가 이 IP로 접근하면 해당 IP와 가장 가까운 Target Proxy 서버로 프로토콜을 라우팅해준다. 그후 Target Proxy에서부터 데이터센터까지는 광케이블 네트워크를 타기 때문에 일반 인터넷망으로 접근하는 것보다 매우 빠른 응답 속도를 보장할 수 있다.

그림 11-13



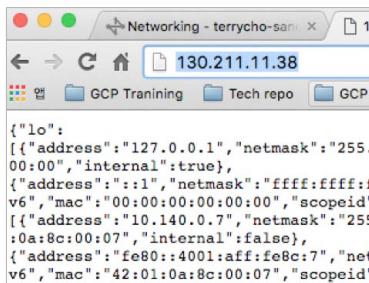
콘텐츠 전송 네트워크 CDN처럼 정적 콘텐츠를 서비스할 때뿐만 아니라, 일반적인 웹 호출이나 REST API도 이 구조를 사용할 수 있어서 서비스를 글로벌로 제공한다면 일반적인 인터넷망보다 빠른 속도를 보장받게 된다. 자세한 내용은 <https://cloud.google.com/compute/docs/load-balancing/http/#overview> 문서를 참고하기 바란다.

## 11.4.1 실제 테스트 결과

이 가속 기능이 어느 정도의 효과를 보여주는지 실제로 테스트해보았다.

서버를 미국 중부에 배포한 후, 첫 번째는 그 서버 IP로 직접 부하를 줘보고, 두 번째는 그 서버 앞에 구글 클라우드 로드 밸런서를 배치한 후 로드 밸런서를 통해서 호출하게 하였다. Node.js 서버로 간단하게 서버의 네트워크 정보를 출력하는 프로그램을 만들어서 호출하였다. 이 서버는 [그림 11-14]와 같이 응답하며, 응답 값의 총 크기는 514바이트다.

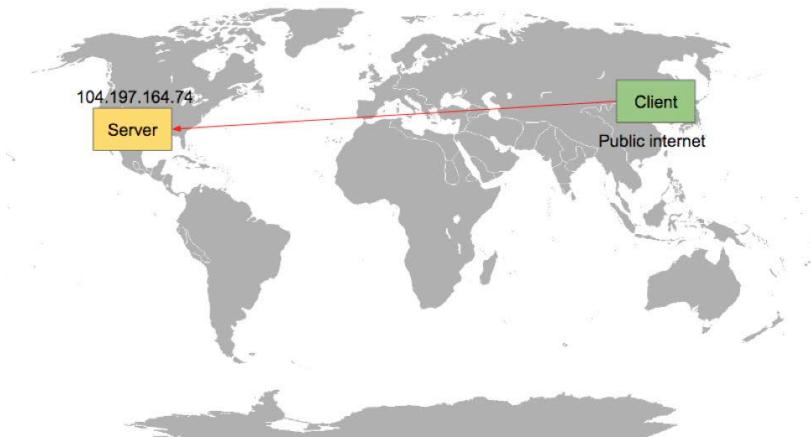
그림 11-14 테스트 서버의 응답



```
{"lo": [{"address": "127.0.0.1", "netmask": "255.00:00", "internal": true}, {"address": "::1", "netmask": "ffff:ffff:f6", "mac": "00:00:00:00:00:00", "scopeid": 1}, {"address": "10.140.0.7", "netmask": "255:0a:8c:00:07", "internal": false}, {"address": "fe80::4001:aff:fe8c:7", "netmask": "ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff", "mac": "42:01:0a:8c:00:07", "scopeid": 2}], "eth0": [{"linklayer": "10.140.0.7", "ip": "10.140.0.7", "netmask": "255.0a:8c:00:07"}]}
```

첫 번째 시나리오는 서버로 직접 호출한 경우다. 서버의 IP는 104.197.165.74를 사용하였다.

그림 11-15 일반 인터넷망으로 직접 호출



아파치 ab 벤치마크 도구를 사용해서 스레드 20개로 3,000번을 호출하였다. 결과는 다음과 같다.

---

Server Software:

Server Hostname: 104.197.164.74

Server Port: 80

Document Path: /

Document Length: 514 bytes

Concurrency Level: 20

Time taken for tests: 58.945 seconds

Complete requests: 3000

Failed requests: 0

Total transferred: 2154000 bytes

HTML transferred: 1542000 bytes

Requests per second: 50.89 [#/sec] (mean)

Time per request: 392.968 [ms] (mean)

Time per request: 19.648 [ms] (mean, across all concurrent requests)

Transfer rate: 35.69 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
--	-----	-------------	--------	-----

Connect:	191	195	4.0	193	219
----------	-----	-----	-----	-----	-----

Processing:	192	196	4.1	194	223
-------------	-----	-----	-----	-----	-----

Waiting:	192	196	4.1	194	223
----------	-----	-----	-----	-----	-----

Total:	383	392	8.0	387	427
--------	-----	-----	-----	-----	-----

Percentage of the requests served within a certain time (ms)

50%	387
-----	-----

66%	393
-----	-----

75%	402
-----	-----

80%	402
-----	-----

90%	403
-----	-----

95%	404
-----	-----

98%	405
-----	-----

99%	407
-----	-----

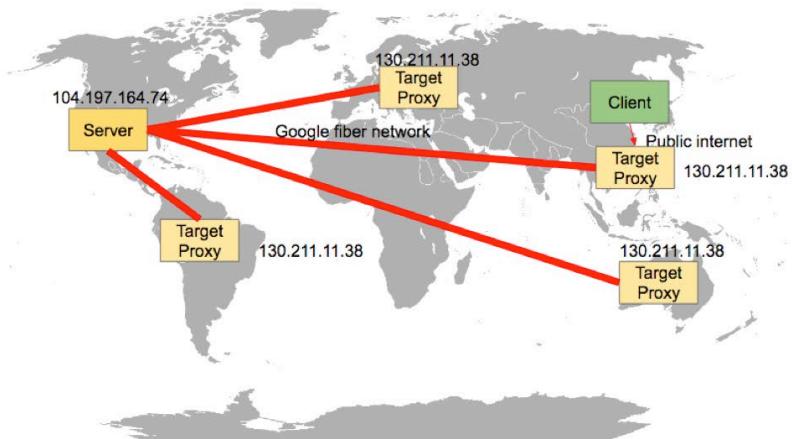
100%	427 (longest request)
------	-----------------------

---

총 58.945초가 소요되었고, 평균 처리 시간이 392ms로 나왔다. 그중에서 연결 시간이 195ms가 나온다.

이번에는 로드 밸런서를 사용해서 트래픽이 구글 백본망을 통해 들어오도록 테스트하였다.

그림 11-16 구글 백본망을 통해 호출



Target Proxy는 전 세계에 배포되어 있고 IP Anycast 로직에 따라서 가장 가까운 곳을 찾아서 라우팅된다. 테스트 조건은 동일하고 결과는 다음과 같다.

---

Server Software:

Server Hostname: 130.211.11.38

Server Port: 80

Document Path: /

Document Length: 514 bytes

Concurrency Level: 20

Time taken for tests: 47.295 seconds

Complete requests: 3000

Failed requests: 0

Total transferred: 2148000 bytes

HTML transferred: 1542000 bytes

Requests per second: 63.43 [#/sec] (mean)  
 Time per request: 315.298 [ms] (mean)  
 Time per request: 15.765 [ms] (mean, across all concurrent requests)  
 Transfer rate: 44.35 [Kbytes/sec] received

#### Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	36	41	1.7	41
Processing:	192	273	78.3	211
Waiting:	192	273	78.3	210
Total:	229	314	78.3	252
				623

#### Percentage of the requests served within a certain time (ms)

50%	252
66%	387
75%	389
80%	395
90%	404
95%	407
98%	420
99%	422
100%	623 (longest request)

terrycho-macbookpro:~ terrycho\$

전체 소요 시간은 47.295초, 호출당 평균 소요 시간은 314ms, 연결 시간은 평균 41ms다. 서버에서 요청을 처리하는 시간은 두 경우 모두 같으니, 결국 연결 시간이 중요할 수밖에 없다. 그리고 그 결과는 4~5배 정도 빨라지는 것을 볼 수 있다. 실제로 파일 크기가 커질수록 응답 시간의 차이도 벌어진다. 이상의 결과를 비교해 [표 11-1]로 정리해보았다.

표 11-1 직접 호출과 구글 백본망을 통한 호출 비교

라벨	직접 호출	로드 밸런서(구글 백본)를 통해 호출
총 소요 시간	58.945초 (124%)	47.295초
호출당 평균 소요 시간	392ms (124%)	314ms
연결 시간	195ms (470%)	41ms

2016년 7월에 일본과 미국 사이에 16TB 대역폭의 구글 클라우드 전용 광케이블 망이 증설되었다. 이상의 테스트는 2016년 6월에 수행한 것으로, 현재는 훨씬 빨라졌다.

## 한빛 리얼타임

한빛 리얼타임은 IT 개발자를 위한 전자책입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾기도 쉽지 않습니다. 또한, 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

### 1 eBook First –

#### 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다

500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 좀 더 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한, 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

### 2 무료로 업데이트되는 전자책 전용 서비스입니다

종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

### 3 독자의 편의를 위해 DRM-Free로 제공합니다

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

### 4 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 어려운 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 총고에 귀 기울이며 지속해서 발전시켜 나가겠습니다.

지금 보시는 전자책에 소유 권한을 표시한 문구가 없거나 타인의 소유권한을 표시한 문구가 있다면 위법하게 사용하고 있을 가능성이 큽니다. 이 경우 저작권법에 따라 불이익을 받으실 수 있습니다.

다양한 기기에 사용할 수 있습니다. 또한, 한빛미디어 사이트에서 구매하신 후에는 횟수와 관계없이 내려받으실 수 있습니다.

한빛미디어 전자책은 인쇄, 검색, 복사하여 불이기가 가능합니다.

전자책은 오픈자 교정이나 내용의 수정·보완이 이뤄지면 업데이트 관련 공지를 이메일로 알려 드리며, 구매하신 전자책의 수정본은 무료로 내려받으실 수 있습니다.

이런 특별한 권한은 한빛미디어 사이트에서 구매하신 독자에게만 제공되며, 다른 사람에게 양도나 이전은 허락되지 않습니다.