

GDG Cloud Team : Google Cloud Associate

3rd Study

2025/01/04

Minseo Kim



Section 6 : Getting Started with Instance Groups in Google Cloud

인스턴스 그룹은 유사한 가상 머신(VM)을 그룹화하여 단일 엔터티로 관리하는 방식.

이를 통해 일관된 구성을 유지하고 관리 효율성을 극대화할 수 있음

주요 이점

- 유사한 VM의 수명 주기를 통합 관리 가능.
- 인스턴스 템플릿을 사용하여 일관된 구성을 효율적으로 유지.



Types of Instance Groups

Managed Instance Group (MIG)

- 인스턴스 템플릿 기반으로 동일한 VM 생성.
- 자동 배율, 자동 치유, 로드 분산 등의 기능 제공.
- 지역적 가용성을 지원하며 여러 존(zone)에 VM을 분산 가능.

Unmanaged Instance Group

- 다양한 구성의 VM을 허용.
- 자동화 기능 미지원.
- 단순 그룹화 및 사용자 지정 구성 관리에 적합.

Key Features of MIG



Autoscaling

사용자 수나 시스템 부하에 따라 VM 인스턴스 수 자동 조정



Autoscaling

상태 확인 실패 시
비정상 인스턴스 자동 대체



Rolling Updates

기존 VM의 템플릿 → 새로운 버전으로 점진적 전환해 서비스 중단 방지



Canary Deployment

새 템플릿을 일부 VM에 먼저 테스트 후 전체 적용 방식



Setting Up a Managed Instance Group

Instance Template

동일한 VM 구성을 생성하기 위한 기본 설정, 필수 구성 요소

Automation Settings

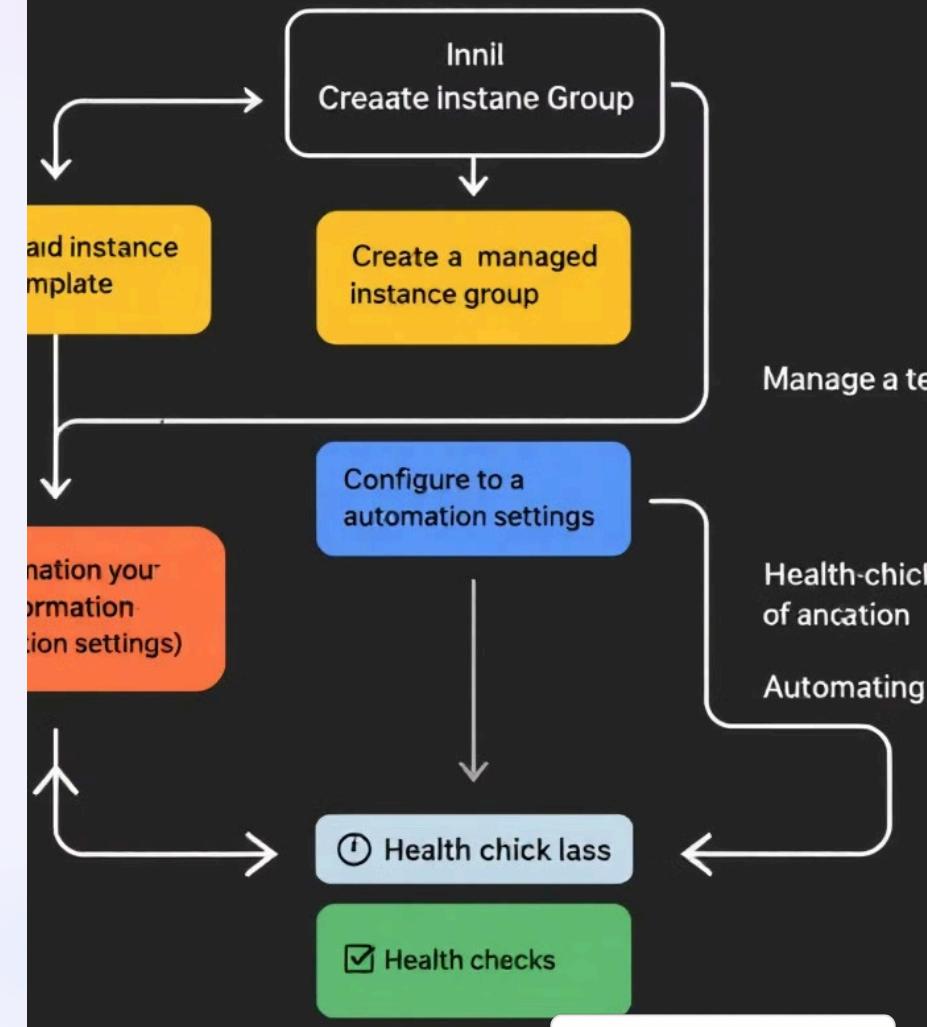
최소 및 최대 인스턴스 수를 설정하여 확장성과 비용을 제어

CPU 사용률, 부하 분산 지표 등 스케일링 조건을 구성

Health Checks

VM 상태를 점검하기 위해 프로토콜, 타임아웃, 초기 지연값 등 설정

Managed Instance Group





Policy Updates and Auto-healing

Auto-healing Policy

VM 상태 확인을 통해 비정상 인스턴스를 자동으로 대체,

초기 지연 시간을 설정해 새로 생성된 VM의 안정성을 확보

Policy Update Commands

gcloud 명령어를 사용하여 설정을 업데이트

건강 확인 추가

```
gcloud compute instance-groups managed update -  
-health-check
```

초기 지연값 설정

```
gcloud compute instance-groups managed update -  
-initial-delay
```

Rolling Actions

Rolling Updates

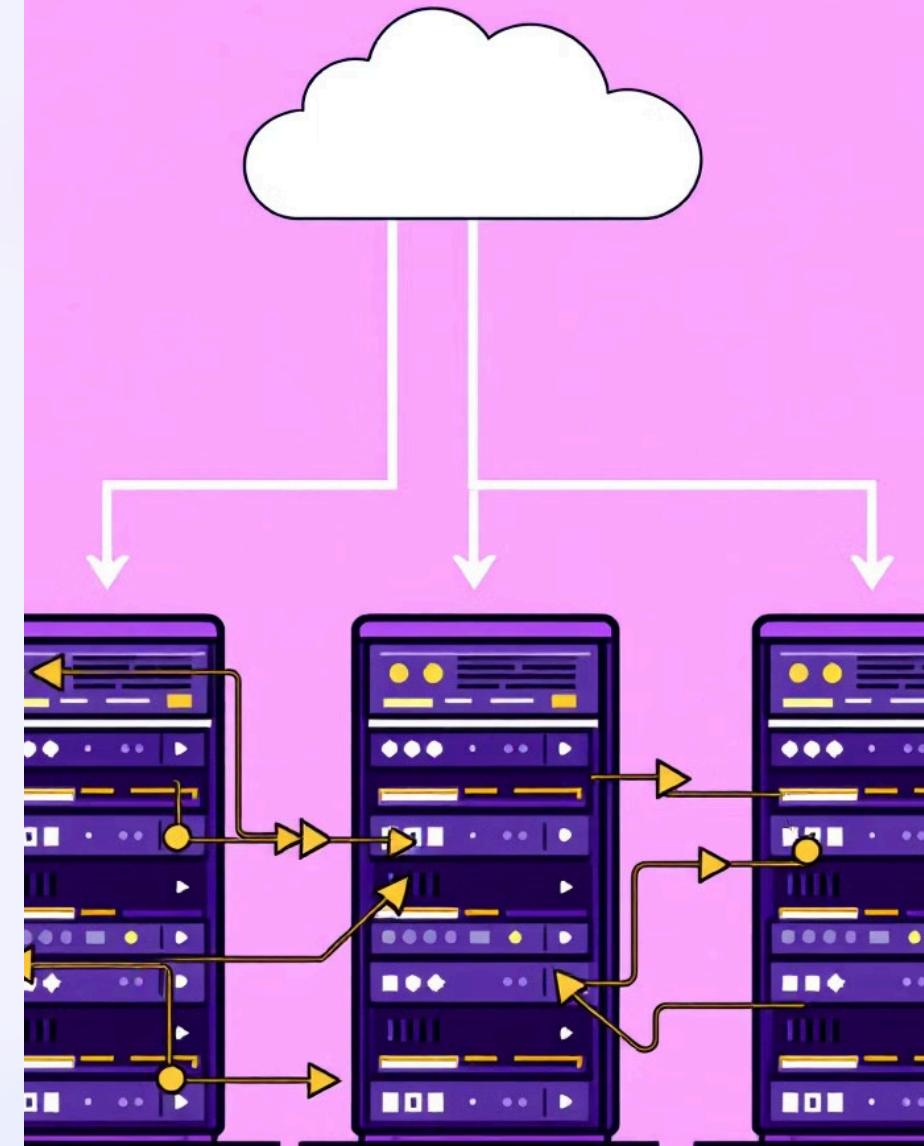
새 템플릿을 사용하여 점진적으로 인스턴스를 업데이트, 특정 인스턴스 수를 유지하면서도 새로운 버전 테스트 가능.

Rolling Restart

기존 템플릿을 유지한 채 VM을 다시 시작하여 구성 변경 적용

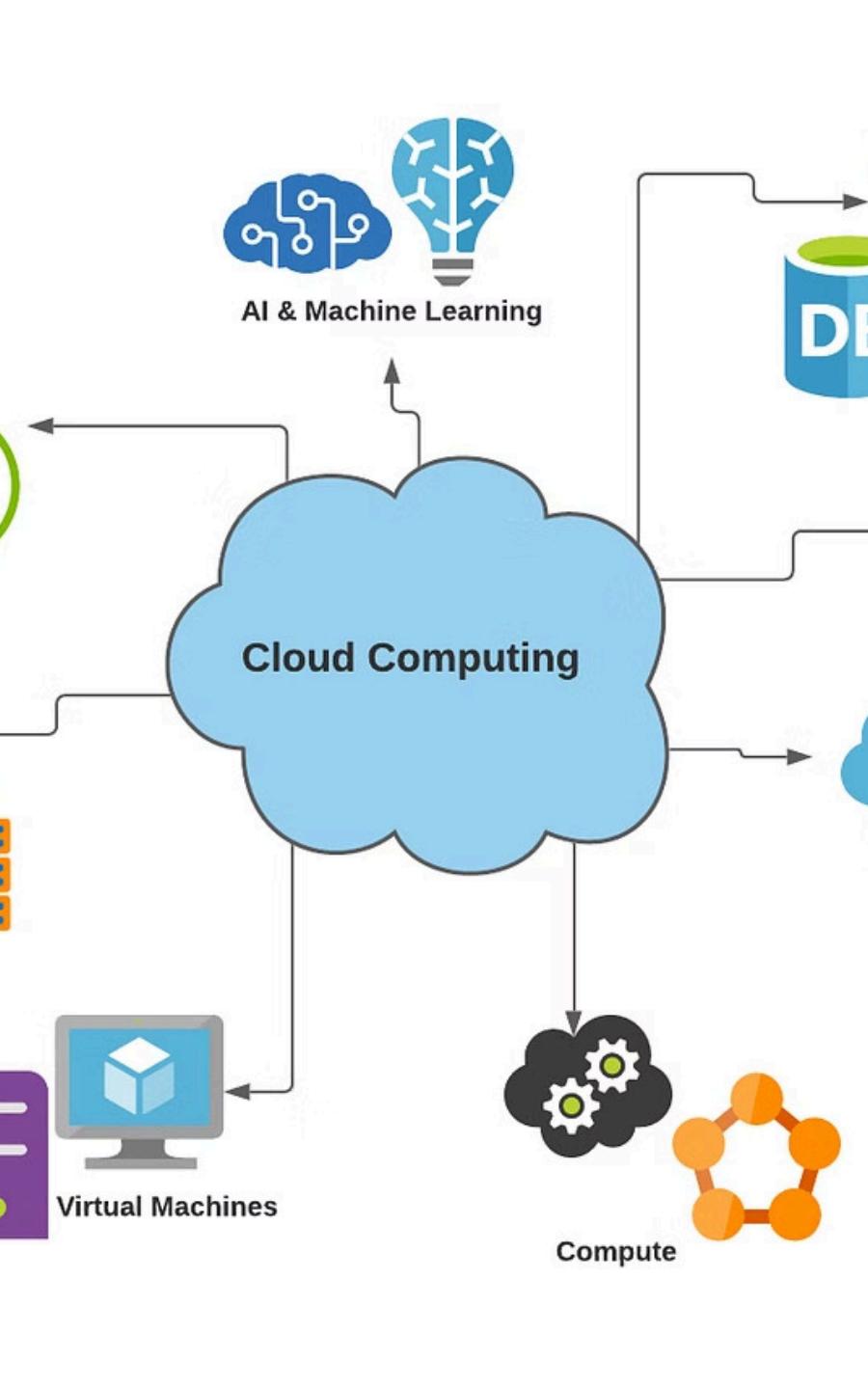
Replace

기존 VM을 삭제하고 동일한 템플릿으로 새로 생성





Scenarios for Instance Groups



1

Regional MIG

여러 존에 인스턴스를 분산 배치하여 구역별 장애를 극복, 가용성을 향상

2

Stateful MIG

VM의 상태(메타데이터, 디스크 등)를 유지하며 데이터베이스와 같은 상태 의존적인 워크로드에 적합

3

Diverse Configurations

관리되지 않은 인스턴스 그룹을 활용하여 다양한 VM 구성의 요구를 충족



Recommendations

When to Use MIG

동일한 VM이 필요한 경우, 자동화 기능(자동 배율, 자가 치유 등)이 요구되는 경우, 또는 높은 가용성이 필요한 경우 사용을 권장

When to Use Unmanaged Instance Groups

VM 구성 다양성이 필요하고 자동화 기능이 필수가 아닌 경우에 적합



Conclusion

○ 요약

MIG는 고가용성과 자동화를 제공하며 대부분의 워크로드에 적합

○ 초기 설정의 중요성

초기 설정을 철저히 구성하면 안정적이고 효율적인 서비스 운영이 가능



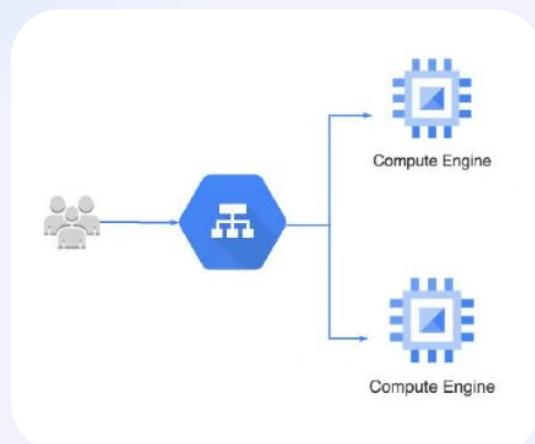
Section 7 - Cloud Load Balancing



What is Cloud Load Balancing?

○ Definition

사용자 트래픽을 단일 지역 또는 여러 지역의 애플리케이션 인스턴스에 분산하는 관리형 서비스



○ Key Features

완전 분산된 소프트웨어 정의 서비스.

Health Check: 정상 인스턴스로 트래픽 라우팅.

Auto Scaling: 트래픽 변화에 따라 인스턴스 자동 조정.

Global Load Balancing: 단일 Anycast IP 사용.

내부 및 외부 로드 밸런싱 지원.



Protocols Overview

1

Network Layer

IP를 사용해 데이터를 전송하지만 신뢰성이 낮음.

2

Transport Layer

TCP: 신뢰성 우선, 성능 희생.

TLS: TCP 보안 버전.

UDP: 성능 우선, 신뢰성 낮음.

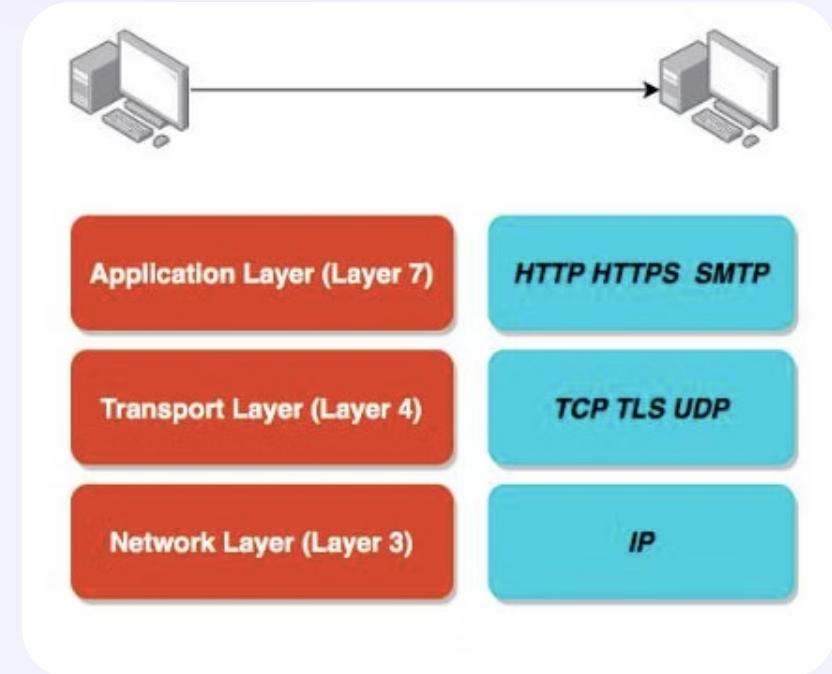
3

Application Layer

HTTP: 무상태 요청-응답.

HTTPS: HTTP의 보안 버전.

SMTP: 이메일 전송 프로토콜.





Application and Transport Layers

Application Layer Communication

웹, 이메일, 파일 전송 등 대부분의 애플리케이션은 이 레이어에서 작동.

Transport Layer for High Performance

게임 및 실시간 스트리밍은 UDP를 사용해 이 레이어에서 직접 통신.

Key Terminology



Backend

로드 밸런서가 트래픽을 전달하는 엔드포인트 그룹.



Frontend

IP 주소, 포트, 프로토콜로 요청을 처리. SSL 사용 시 인증서 필요.

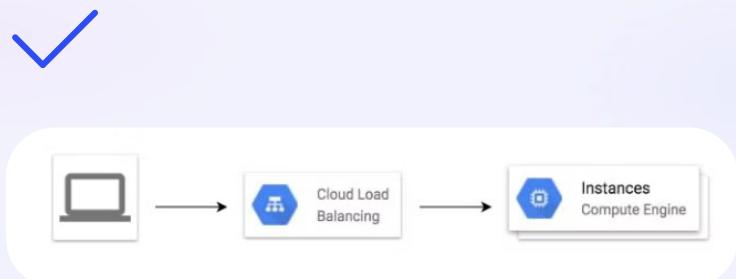


Host and Path Rules

경로 기반: /a, /b로 트래픽 리다이렉트.

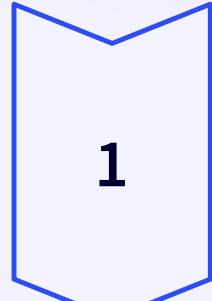
호스트 기반: a.com, b.com으로 분배.

HTTP 메소드 및 헤더 기반으로 설정 가능.



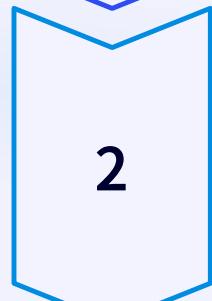


SSL/TLS Termination



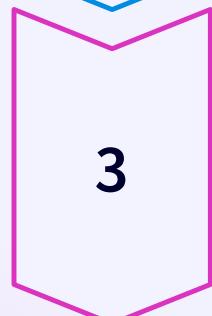
Client Load Balancer

HTTPS 권장.



Load Balancer VM Instance

HTTP 가능하지만 HTTPS 권장.

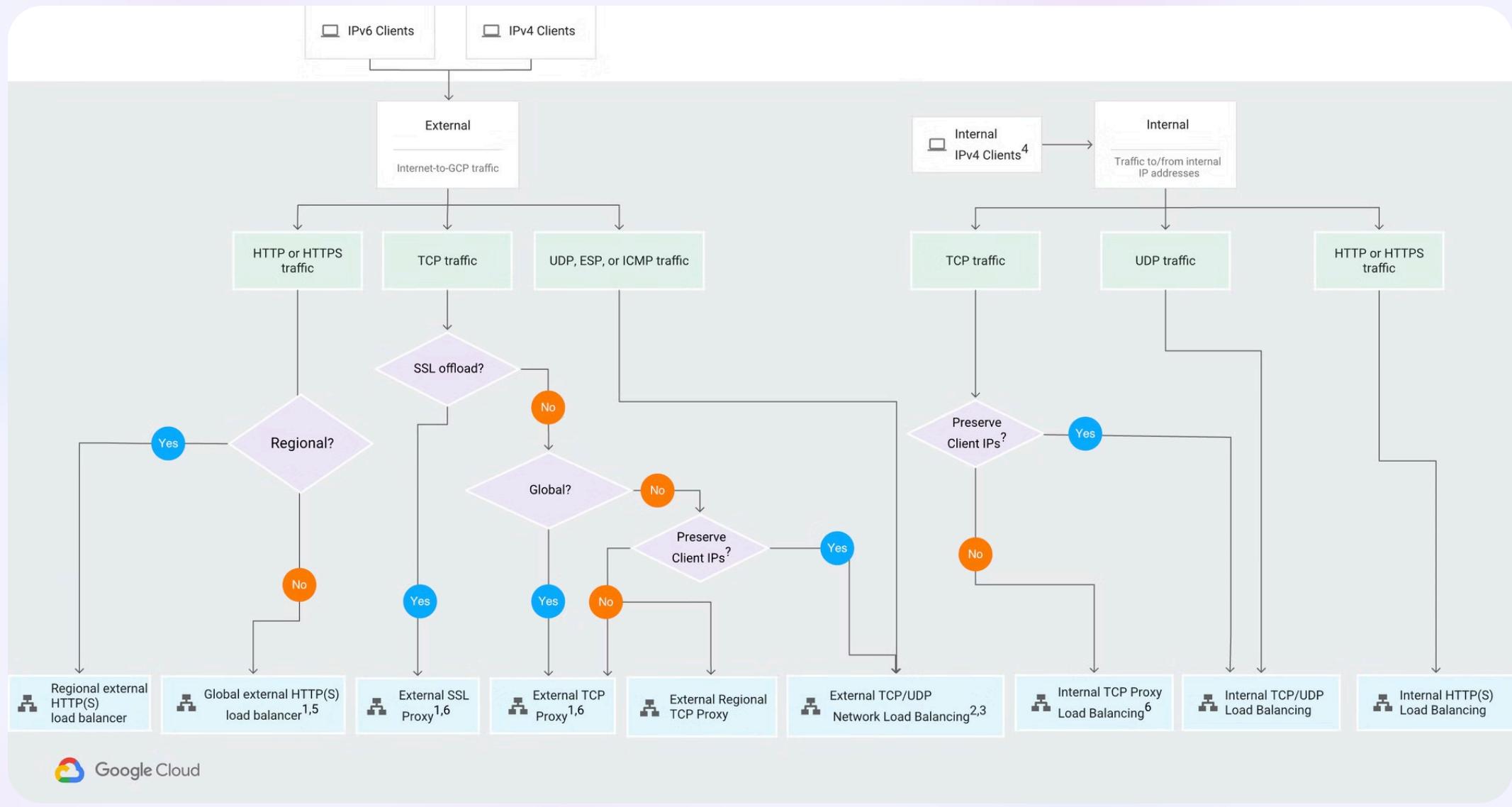


Termination

클라이언트와 로드 밸런서는 HTTPS로 통신, 로드 밸런서와 VM은 HTTP 사용 가능.



Choosing Load Balancer





Load Balancer Types

Type	Traffic	Proxy/Pass-through	Ports
External HTTP(S)	글로벌, 외부 HTTP/HTTPS	Proxy	HTTP: 80, 8080 / HTTPS: 443
Internal HTTP(S)	지역, 내부 HTTP/HTTPS	Proxy	HTTP: 80, 8080 / HTTPS: 443
SSL Proxy	글로벌, 외부 TCP/SSL	Proxy	다양한 포트 지원
TCP Proxy	글로벌, 외부 TCP	Proxy	다양한 포트 지원
External Network TCP/UDP	지역, 외부 TCP/UDP	Pass-through	모든 포트 지원



Scenarios and Solutions

Healthy Instance Traffic

Health Check 구성.

High Availability

여러 지역에 MIG 생성 후 로드 밸런서 사용.

Multiple Microservices

각 마이크로서비스에 대해 MIG 생성, Host 및 Path Rules로 분배.

Global HTTPS Traffic

External HTTP(S) Load Balancer 선택.

SSL Termination

SSL Proxy Load Balancer 사용.



Section 8 : Managed Services



1. What are Managed Services?

○ Definition

클라우드에서 애플리케이션을 실행하거나 기존 데이터 센터 방식과 달리 다양한 클라우드 제공 방식을 활용하는 서비스.

○ Key Terms

IaaS (Infrastructure as a Service): 인프라 사용.

PaaS (Platform as a Service): 플랫폼 사용.

FaaS (Function as a Service): 함수 기반 서비스.

CaaS (Container as a Service): 컨테이너 관리 서비스.

Serverless: 서버 관리 없이 코드를 실행.



2. IaaS (Infrastructure as a Service)

Overview

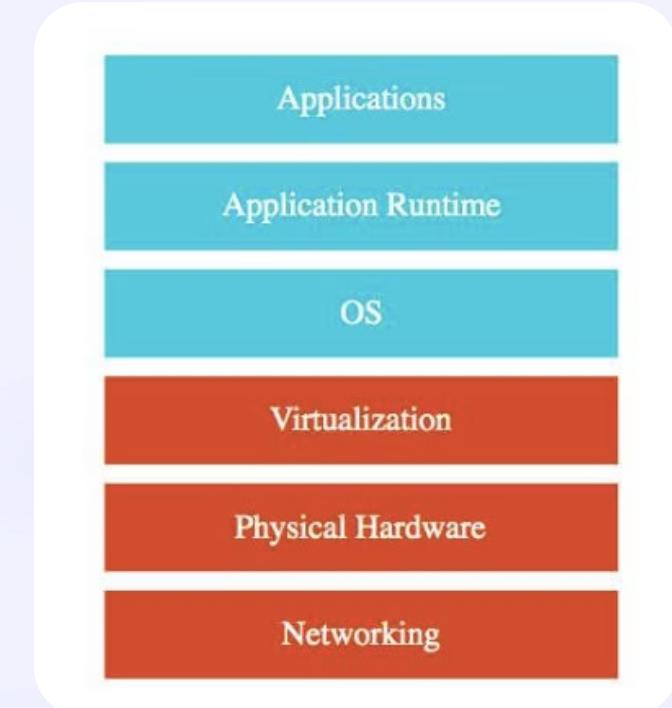
클라우드 제공자로부터 인프라만 사용.

Responsibilities

- 애플리케이션 코드, 런타임 관리.
- 로드 밸런싱 구성, 자동 스케일링 설정.
- OS 업그레이드 및 패치.
- 가용성 관리.

Example

- VM을 사용해 애플리케이션 또는 데이터베이스 배포.





3. PaaS (Platform as a Service)

Overview

클라우드 제공 플랫폼을 활용.

Cloud Provider Responsibilities

- OS 관리 (업데이트 및 패치 포함).
- 애플리케이션 런타임, 자동 스케일링, 로드 밸런싱 등 관리.



Varieties

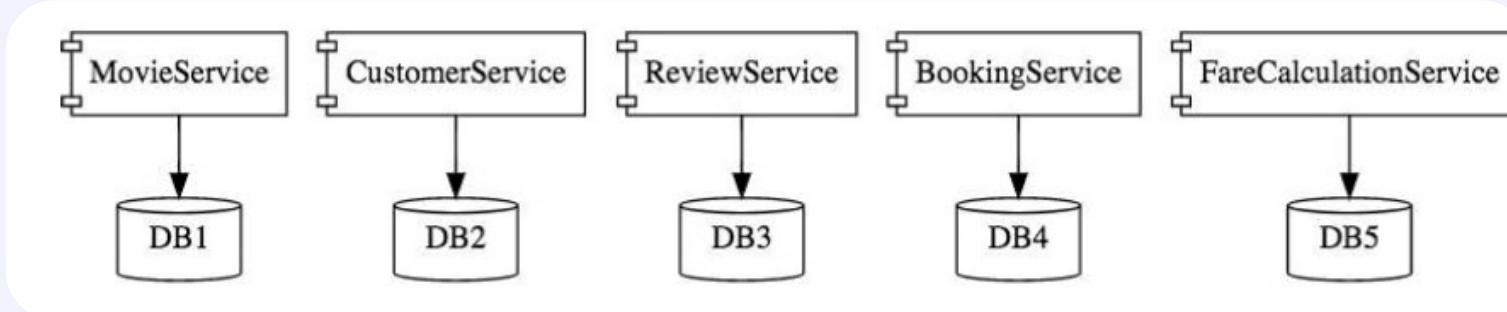
- **CaaS:** 컨테이너를 사용해 관리.
- **FaaS:** 함수 단위로 관리.
- 데이터베이스 (관계형 및 NoSQL), 큐, AI/ML 등 포함.

User Responsibilities

- 애플리케이션 및 서비스 구성.
- 필요 시 애플리케이션 코드 작성.



4. Microservices



1 Overview

작은 단위로 독립적 마이크로서비스를 구축.

2 Benefits

다양한 프로그래밍 언어(Go, Java, Python, JavaScript 등)로 개발 가능.

3 Challenges

복잡한 배포 환경.

4 Solution

컨테이너 사용.



5. Containers with Docker



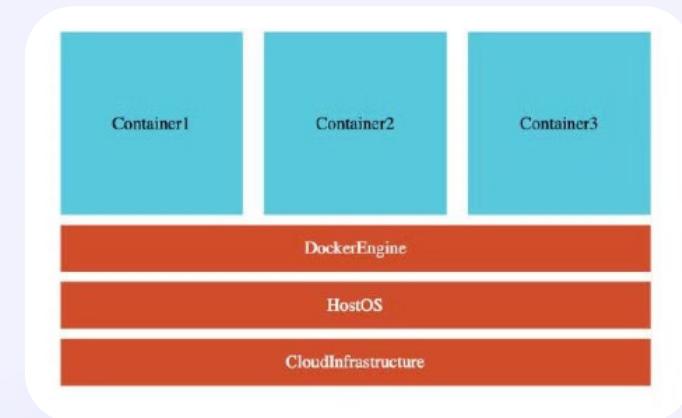
What is Docker?

각 마이크로서비스에 필요한 런타임, 코드, 종속성을 포함한 도커 이미지 생성.



Advantages

가벼움 (Guest OS가 없음).



Cloud Neutral

인프라와 무관하게 동일한 방식으로 실행.



Isolation

클라우드 중립적이며, 격리 환경 제공.



컨테이너 오케스트레이션

1 자동 확장

수요에 따라 컨테이너를 확장합니다.

2 서비스 검색

마이크로서비스가 서로를 찾을 수 있도록 지원합니다.

3 부하 분산

마이크로서비스의 여러 인스턴스에 부하를 분산합니다.

4 자가 치유

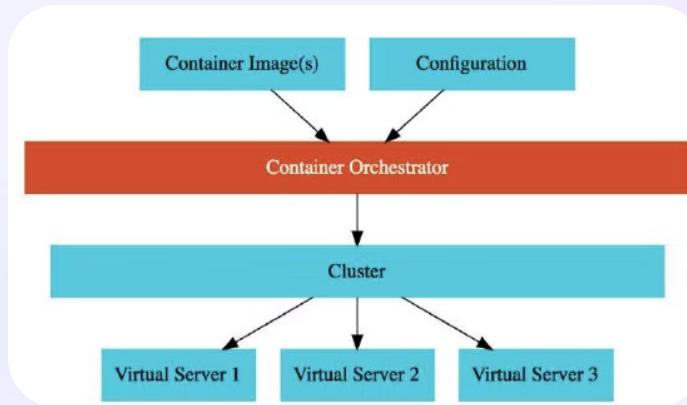
건강 검사를 수행하고 오류가 발생한 인스턴스를 대체합니다.

5 제로 다운타임 배포

다운타임 없이 새 버전을 출시합니다.



6. Container Orchestration



○ Requirements

여러 컨테이너를 효율적으로 관리하고 배포.

○ Feature

Auto Scaling

수요에 따른 컨테이너 자동 확장.

Service Discovery

マイ크로서비스 간 연결.

Load Balancer

트래픽 분산.

Self Healing & Zero Downtime

장애 컨테이너 자동 교체 및 다운타임 없는 배포.



7. Serverless

Definition

인프라를 신경 쓰지 않고 코드를 실행하는 방식.

Key Features

- 인프라 관리 불필요.
- 유연한 스케일링 및 높은 가용성.
- 사용량에 따라 비용 지불 (요청 없으면 비용 0).



8. Serverless Levels

1

Level 1

Google App Engine, AWS Fargate

2

Level 1 Features

로드가 없을 때 인스턴스 0으로 축소 가능

3

Level 2

Google Functions, AWS Lambda, Azure Functions

4

Level 2 Features

요청에 따라 비용 지불

Level 1: 인스턴스 수에 따라 비용 지불.



9. GCP Managed Services for Compute



Compute Engine

고성능 VM 제공 (IaaS).



Google Kubernetes Engine

컨테이너화된 마이크로서비스를 관리 및 조정 (CaaS).



App Engine

완전 관리형 플랫폼에서 애플리케이션 빌드 및 실행 (PaaS).



Cloud Functions

단일 함수로 이벤트 중심 애플리케이션 개발 (FaaS).



Cloud Run

클러스터 없이 컨테이너화된 애플리케이션 실행 (CaaS).



Section 9 : App Engine



1. App Engine Overview

○ Definition

GCP에서 애플리케이션을 배포하고 확장하는
가장 간단한 방법.

○ Features

- 애플리케이션 관리 자동화 제공.
- 사전 구성된 런타임(GO, Java, Python 등) 지원.
- 맞춤 런타임으로 언어에 관계없이 코드 작성 가능.
- Google Cloud 저장소 제품(Cloud SQL 등)과 연동.
- 사용량에 따른 비용 지불.
- 자동 로드 밸런싱, 자동 스케일링, 애플리케이션 모니터링,
트래픽 분할 등 지원.



2. Compute Engine vs App Engine

Compute Engine (IaaS)

더 높은 유연성.

하드웨어, 소프트웨어, 인증서, 방화벽 등 세부 구성 가능.

OS 관리 포함.

App Engine (PaaS)

서버리스 환경.

더 적은 책임.

제한된 유연성.



3. App Engine Environments

Standard Environment

언어별 샌드박스에서 실행.

V1: 제한된 네트워크 접근.

V2: 네트워크 접근 제한 없음.

Flexible Environment

Docker 컨테이너에서 애플리케이션 실행.

Compute Engine VM 활용.

사용자 정의 런타임 지원.



4. Application Component Hierarchy

1 Application

프로젝트당 1개의 앱만 생성 가능.

2 Service(s)

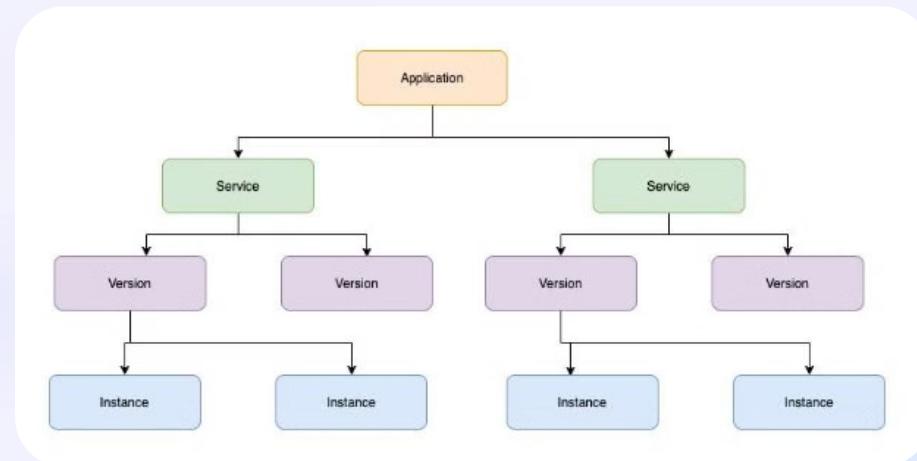
단일 애플리케이션에 다중 서비스 가능.

각 서비스는 서로 다른 설정 가짐.

3 Version(s)

코드 및 설정 버전 관리.

여러 버전 공존 가능하며 롤백 및 트래픽 분할 지원.





5. App Engine Scaling



Automatic Scaling

로드에 따라 인스턴스 자동 확장.

CPU 사용률, 최대 동시 요청 등 기준 설정 가능.



Basic Scaling

요청 발생 시 인스턴스 생성, 요청 없으면 종료.

비용 효율적.

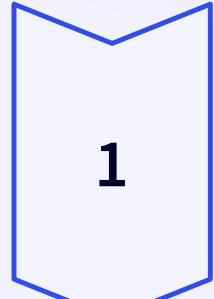


Manual Scaling

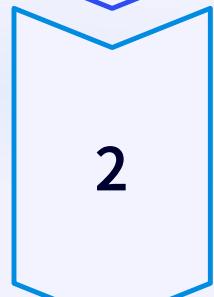
실행할 인스턴스 수를 수동으로 설정.



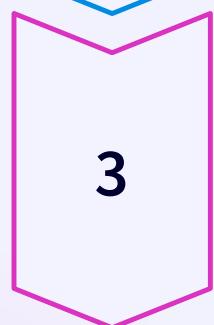
6. Request Routing



URL 기반 라우팅



Dispatch 파일(dispatch.yaml)로 라우팅



Cloud Load Balancer 사용



7. Deploying New Versions

1 Zero Downtime Deployment

V1에서 V2로 이동 시 트래픽 분할로 다운타임 방지.

2 No-Promote Deployment

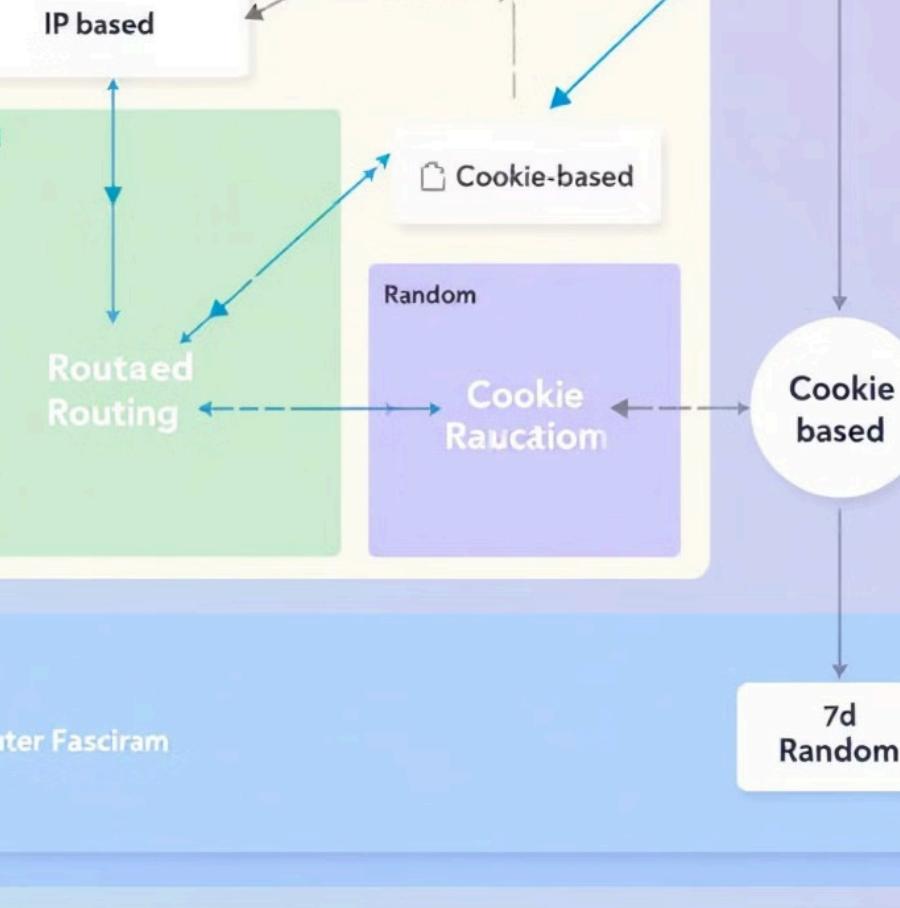
--no-promote로 트래픽 이동 없이 V2 배포.

3 Traffic Splitting

트래픽 분할: 점진적 이동 또는 A/B 테스트.



on



8. Traffic Splitting Options

○ Splitting Methods

IP 기반: IP 주소로 트래픽 분할.

Cookie 기반: 애플리케이션 쿠키로 트래픽 분할.

랜덤: 무작위로 트래픽 분할.

ter Fasiram



9. YAML Configuration

app.yaml

런타임, 스케일링 옵션, 환경 변수 등 정의.

cron.yaml

주기적인 작업을 설정 (예: 캐시 갱신).

dispatch.yaml

라우팅 규칙 재정의.

app.yaml Reference

```
runtime: python28 #The name of the runtime environment that is used by your app
api_version: 1 #RECOMMENDED - Specify here - gcloud app deploy -v [YOUR_VERSION_ID]
instance_class: F1
service: service-name
#env: flex

inbound_services:
- warmup

env_variables:
ENV_VARIABLE: "value"

handlers:
- url: /
  script: home.app

automatic_scaling:
target_cpu_utilization: 0.65
min_instances: 5
max_instances: 100
max_concurrent_requests: 50
#basic_scaling:
#  max_instances: 11
#  idle_timeout: 10m
#manual_scaling:
#  instances: 5
```

In
28
Minutes

101



10. Key Scenarios

Creating Multiple Apps in One Project

프로젝트당 하나의 App Engine 앱만 가능.
다중 서비스와 다중 버전은 지원됨.

Moving App to a Different Region

앱은 특정 지역에 고정되며 이동할 수 없음.
새로운 프로젝트와 앱을 생성해야 함

Canary Deployment

새로운 버전(V2) 배포 후 점진적으로 트래픽을 이동.



11. App Engine 유연성

Resident 인스턴스

지속적으로 실행되는 인스턴스.

지속적인 가용성을 보장

Dynamic 인스턴스

로드에 따라 동적으로 추가 및 제거됨.

비용 효율적



Cloud Load Balancing

Question: You need to create an autoscaling managed instance group for an HTTPS web application. You want to ensure that unhealthy VMs are recreated. What should you do?

- A. Create a health check on port 443 and use that when creating the Managed Instance Group.
- B. Select Multi-Zone instead of Single-Zone when creating the Managed Instance Group.
- C. In the Instance Template, add the label ‘health-check’.
- D. In the Instance Template, add a startup script that sends a heartbeat to the metadata server.



Managed Services

Question: Your company has an App Engine application that needs to store stateful data in a proper storage service. Your data is non-relational, and you do not expect the database size to grow beyond 10 GB. You need to have the ability to scale down to zero to avoid unnecessary costs. Which storage service should you use?

Options:

- A. Cloud SQL
- B. Cloud Bigtable
- C. Cloud Datastore
- D. Cloud Dataproc



App Engine

Question: You deployed an App Engine application using gcloud app deploy, but it did not deploy to the intended project. You want to find out why this happened and where the application deployed. What should you do?

Options:

- A. Check the app.yaml file for your application and check project settings.
- B. Check the web-application.xml file for your application and check project settings.
- C. Go to Deployment Manager and review settings for deployment of applications.
- D. Go to Cloud Shell and run gcloud config list to review the Google Cloud configuration used for deployment.



Additional GCP Services

Question: Your company is migrating all applications from the on-premises data center to Google Cloud, and one of the applications is dependent on WebSockets protocol and session affinity. You want to ensure this application can be migrated to Google Cloud Platform and continue serving requests without issues. What should you do?

Options:

- A. Modify application code to not depend on session affinity.
- B. Review the design with the security team.
- C. Modify application code to use HTTP streaming.
- D. Discuss load balancer options with the relevant teams.