

# Golang with Firebase

Go 배우기 망설이는 그대에게

이준호 / INGRADIENT



# Speaker

**이준호**

INGRADIENT


- GDG Jeju Organizer
- INGRADIENT CEO



# Index

- Python과 GO의 비교
  - 문법 비교 및 코드 가독성
- Go의 특징
  - 동시성(Concurrency) 및 병렬 처리
  - Compiler vs Interpreter
  - 배포 및 운영
- Benchmark
- 왜 회사에서 GO를 찾는지
- Firebase with Go Demo : Chat





# Python과 Go의 비교



# Python vs Go

## Python

- **동적 타입 언어:** 런타임 시 타입이 결정되며, 이는 빠른 개발과 코드 유연성을 제공하지만, 때로는 성능 저하를 초래
- **가독성 높은 문법:** 코드의 가독성과 간결성을 강조하여 초보자도 쉽게 배울 수 있음
- **광범위한 표준 라이브러리:** 다양한 기능을 제공하는 풍부한 표준 라이브러리를 포함하고 있어, 여러 작업을 쉽게 수행

# Python vs Go

## Python

- **인터프리터 언어:** 코드가 한 줄씩 실행되므로 빠른 테스트와 디버깅 가능.
- **크로스 플랫폼:** 다양한 운영 체제에서 실행 가능하며, 플랫폼 간 호환성이 뛰어남
- **다양한 프레임워크 및 도구:** 웹 개발, 데이터 과학, 인공지능, 자동화 등 다양한 분야에서 활용할 수 있는 프레임워크와 도구가 풍부

# Python vs Go

Go

- **정적 타입 언어:** 컴파일 시 타입이 결정되며, 이는 코드의 안정성과 성능을 높이는 데 기여
- **간결한 문법:** 코드의 가독성을 높이고 작성하기 쉽게 설계
- **고루틴(Goroutine):** 경량 스레드를 지원하여 동시성과 병렬 처리를 효율적으로 관리
- **컴파일러 언어:** Go는 컴파일러 언어로, 코드를 기계어로 변환하여 빠른 실행 속도



# 문법 비교 및 코드 가독성





# Python 예제 코드

```
def main():  
    x = 10  
    y = 20  
    print("Sum:", x + y)
```

```
if __name__ == "__main__":  
    main()
```

# Go 예제 코드

```
package main
```

```
import "fmt"
```

```
func main() {  
    var x int = 10  
    y := 20  
    fmt.Println("Sum:", x + y)  
}
```

# Python vs Go

## GO : fmt 라이브러리 소개

**fmt** 패키지는 Golang 표준 라이브러리 중 하나로, 포맷된 I/O를 제공

### 출력 함수

1. **Print, Printf, Println**:  
콘솔에 데이터 출력
2. **Fprint, Fprintf, Fprintln**:  
지정된 **io.Writer**에 데이터  
출력
3. **Sprint, Sprintf, Sprintln**:  
포맷된 문자열 반환

### 입력 함수

1. **Scan, Scanf, Scanln**: 표준  
입력에서 데이터 읽음
2. **Fscan, Fscanf, Fscanln**: 지정된  
**io.Reader**에서 데이터 읽음
3. **Sscan, Sscanf, Sscanln**:  
문자열에서 데이터 읽음

# Python vs Go

## GO 소개

Go는 명확하고 엄격한 코드 구조를 따르도록 강제

코드 스타일 가이드가 있으며, 이를 위한 **gofmt** 도구가 제공

```
# Python 오류 처리 예제
```

```
try:
```

```
    with open("example.txt", "r")
```

```
as file:
```

```
    print("File opened  
successfully")
```

```
except FileNotFoundError as e:
```

```
    print("Error:", e)
```

```
// Golang 오류 처리 예제
```

```
package main
```

```
import (
```

```
    "fmt"
```

```
    "os"
```

```
)
```

```
func main() {
```

```
    file, err := os.Open("example.txt")
```

```
    if err != nil {
```

```
        fmt.Println("Error:", err)
```

```
        return
```

```
    }
```

```
    defer file.Close()
```

```
    fmt.Println("File opened successfully")
```

```
}
```



# 동시성 (Concurrency) 및 병렬 처리

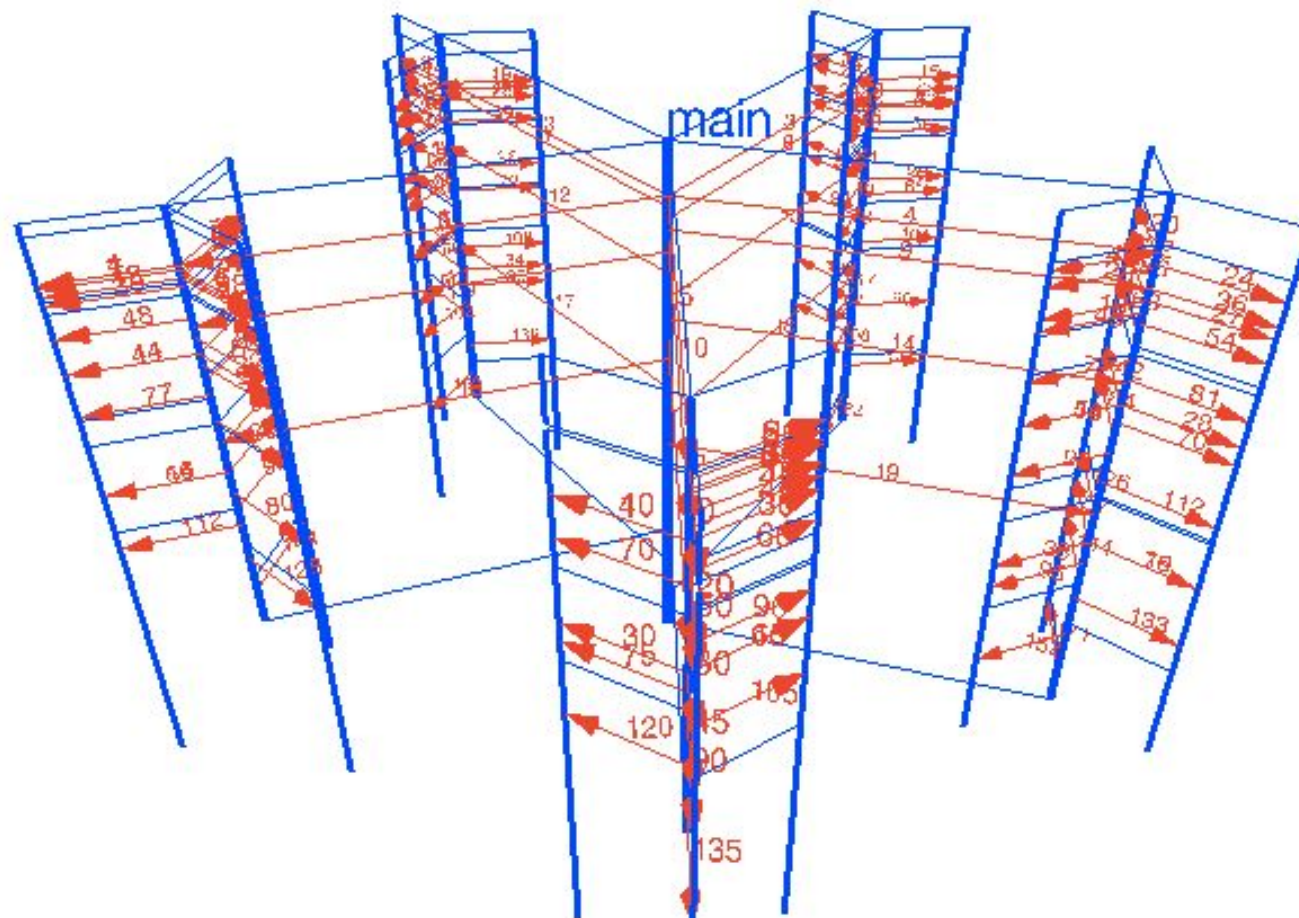


# 동시성 및 병렬 처리

## Goroutine 이란

고루틴은 Golang에서 동시성 처리를 위한 경량 스레드. 수천 개를 동시에 실행할 수 있으며, 메모리 사용이 적고 Go 런타임에 의해 효율적으로 관리

1. 경량 스레드: 고루틴은 운영체제의 스레드보다 훨씬 가볍고, 작은 메모리 공간을 사용
2. 병렬 처리: 수천 개의 고루틴을 병렬로 실행할 수 있어, 고성능 애플리케이션을 개발 가능
3. 자동 스케줄링: Go 런타임이 고루틴을 스케줄링하고 관리하여, 개발자가 직접 스레드를 관리 불필요
4. 채널(Channels): 고루틴 간의 통신을 위해 채널을 사용하여, 데이터의 안전한 교환이 가능



[https://divan.dev/posts/go\\_concurrency\\_visualize/](https://divan.dev/posts/go_concurrency_visualize/)


```
package main

import (
    "fmt"
    "time"
)

func say(s string) {
    for i := 0; i < 5; i++ {
        time.Sleep(100 * time.Millisecond)
        fmt.Println(s)
    }
}

func main() {
    go say("world") // 고루틴 실행
    say("hello")    // 메인 고루틴 실행
}
```





# Compiler vs Interpreter



# Interpreter vs Compiler

## Interpreter

- **실행 과정:** 소스 코드를 한 줄씩 해석하고 실행.
- **실행 속도:** 소스 코드를 실행 시마다 해석하므로 상대적으로 느림.
- **오류 검출:** 실행 중에 오류가 발생하면 해당 시점에서 중단됨.
- **플랫폼 독립성:** 동일한 인터프리터가 설치된 모든 플랫폼에서 소스 코드 그대로 실행 가능.

# Interpreter vs Compiler

## Compiler

- **컴파일 과정:** 소스 코드를 한 번에 기계어로 변환하여 실행 파일을 생성.
- **실행 속도:** 컴파일된 프로그램은 기계어로 실행되므로 빠름.
- **오류 검출:** 컴파일 시 모든 구문 및 타입 오류를 검사하여, 실행 전에 모든 오류를 해결해야 함.
- **플랫폼 종속성:** 컴파일된 바이너리는 특정 플랫폼(운영체제 및 하드웨어)에 종속적임.



# 배포 및 운영



# 배포 및 운영

## Golang Application

- **컨테이너화:** Docker와 같은 컨테이너 기술과 잘 통합되어 있으며, 작은 크기의 이미지를 생성할 수 있습니다. 이는 배포와 운영을 단순화하고, 자원 효율성을 높임
- **빠른 시작 시간:** Go 바이너리는 경량이며 빠르게 시작할 수 있어, 마이크로서비스 아키텍처와 같은 환경에서 유리

# 배포 및 운영

## Golang Application

- **단일 바이너리 파일:** Go는 모든 종속성을 포함한 단일 실행 파일을 생성하므로, 추가적인 라이브러리 설치나 종속성 관리가 필요 없음
- **크로스 컴파일:** Go는 다양한 운영 체제와 아키텍처에 대한 크로스 컴파일을 지원합니다. 예를 들어, Linux에서 Windows용 바이너리를 쉽게 생성

# 예시: Linux용 바이너리 생성

```
GOOS=linux GOARCH=amd64 go build -o myapp main.go
```

# 배포 및 운영

## Python


- Python 애플리케이션 배포는 유연하지만, 종속성 관리와 환경 설정이 복잡할 수 있습니다. Python은 인터프리터 언어로, 소스 코드와 종속성을 함께 배포 (유연하지만 종속성 관리와 환경 설정이 복잡할 수 있으며, 가상 환경과 컨테이너화)





# Benchmark





# 100x

Python Average: 106.756

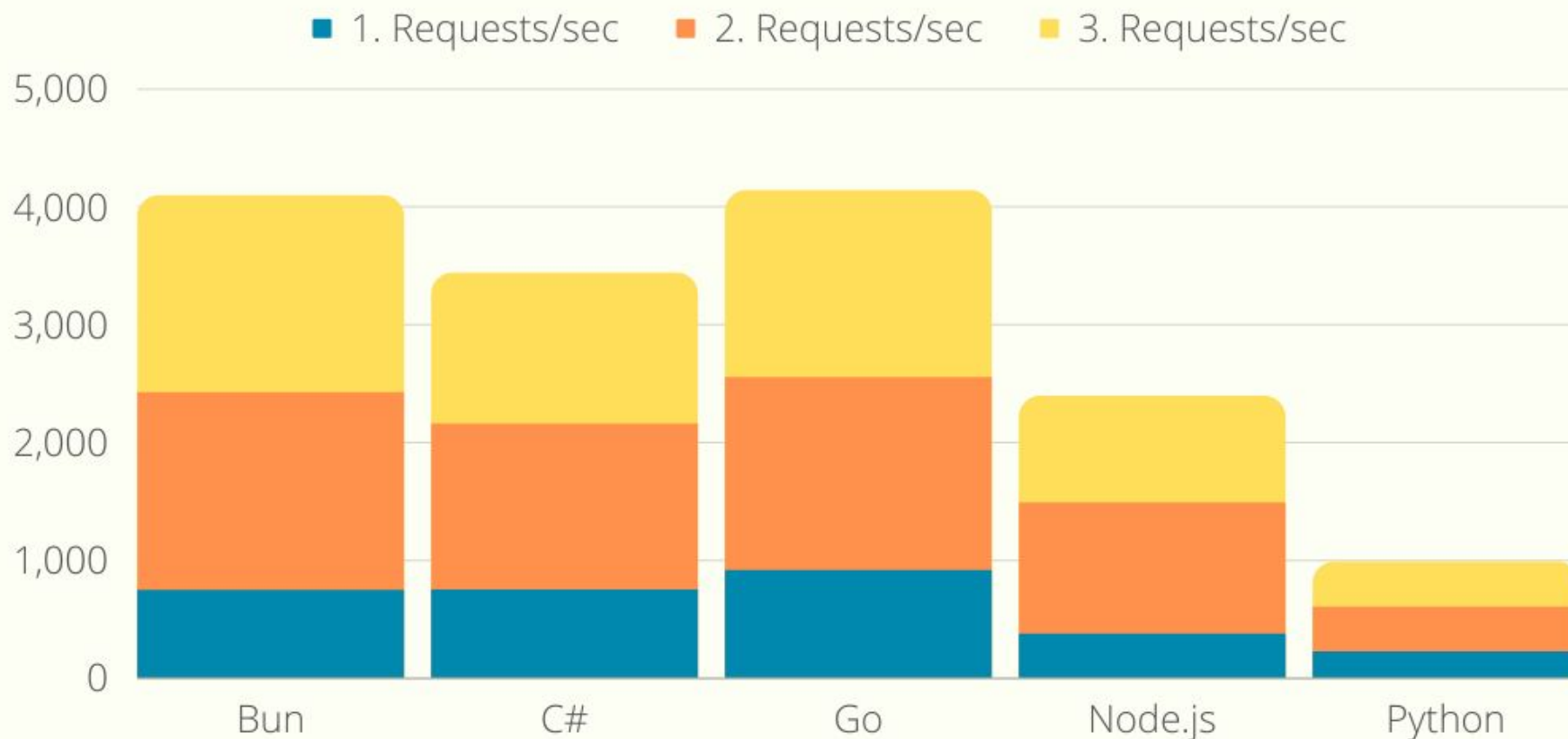
Java Average: 9.0565

Go Average: 8.98625

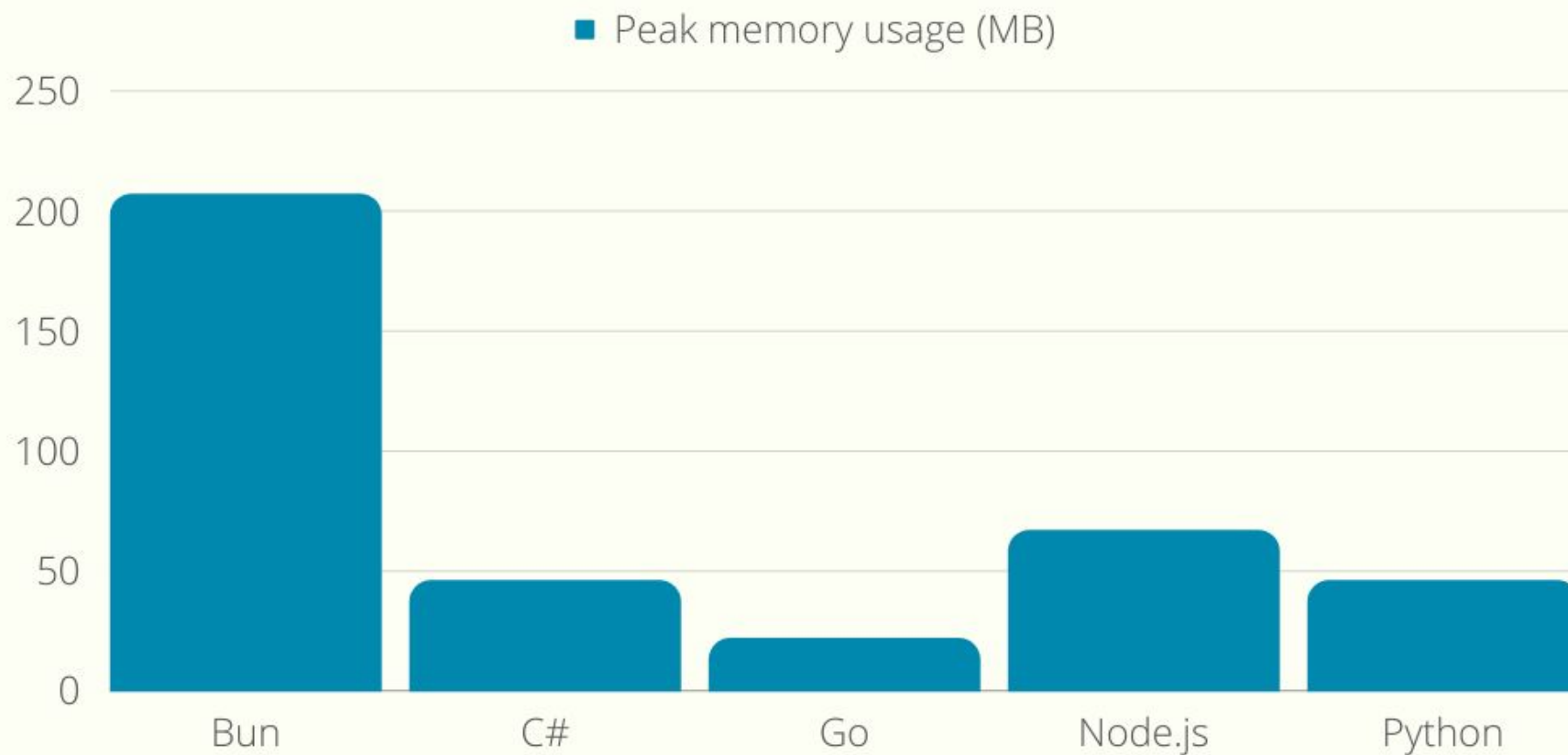
Go Average: 8.98625

## BENCHMARK RESULTS

1. **/food/321358**  
Get JSON data of food by fdclid. Involves a hash table lookup. The JSON that's returned is over 40 KB.
2. **/search/nutrients/1005:0.1-0.1**  
Return all fdclid's from foods that contain exactly 0.1 carbohydrates.
3. **/search/nutrients/1005:0.5-200.0**  
Return all fdclid's from foods that contain between 0.5 and 200 carbohydrates.



## BENCHMARK RESULTS





# 왜 회사에서 GO를 원하는지



# Go의 장점

높은 수요와 낮은 공급

- **경쟁이 적음:** 많은 개발자들이 **Python, JavaScript** 등 인기 언어를 선호해 **Go**를 숙달한 개발자는 경쟁이 적음.
- **높은 수요:** 클라우드 컴퓨팅, 컨테이너화, 분산 시스템 등 특정 산업에서 **Go**의 효율성과 성능을 중시.

# Go의 장점

## 성과와 효율성

- **고성능:** Go는 컴파일 언어로, 빠른 실행 속도와 낮은 메모리 사용을 자랑.
- **효율적 동시성 처리:** 고루틴(goroutine)과 채널(channel)을 통한 효율적인 동시성 처리

# Go의 장점

## 경량화 및 병렬 처리의 중요성

- **경량화된 환경:** 인공지능의 발전으로 더 경량화되고 빠른 환경이 필요해짐. **Go**는 경량화와 병렬 처리가 잘 되어 있어 이러한 요구에 적합
- **효율성:** 빠른 실행 속도와 효율적인 자원 관리를 통해 고성능 환경에서의 경쟁력을 제공



# Go의 장점


## 성장하는 커뮤니티와 생태계


- **성장 잠재력:** Google의 후원과 Docker, Kubernetes 같은 프로젝트 덕분에 Go는 지속적으로 성장.
- **기여 기회:** 오픈 소스 프로젝트에 기여하거나 커뮤니티에서 중요한 역할을 맡을 수 있는 기회.



## Python

Repositories related to the Python Programming language

 **21.2k** followers

 <https://www.python.org/>


Verified



## Go

The Go Programming Language

 **7.8k** followers

 <https://go.dev>

# Go의 장점

## 기업의 신뢰와 지원

- **기업의 채택:** Google, Dropbox, Uber 등 많은 기업이 Go를 사용, Go 개발자에 대한 신뢰와 수요가 높아짐.
- **지속적인 언어 발전:** 대형 기업의 지원을 받아 Go는 지속적으로 발전, 언어의 안정성과 장기적인 커리어 전망을 보장.



StackOverflow 질문 수: 73320 Github Stars : ★ 120634

공식 사이트 바로가기

Github 바로가기

## 사용 기업

인공지능 이커머스 소셜/컨텐츠 부동산/인테리어 기타 푸드테크 헬스케어 금융/보험 패션 모빌리티 직장 여행 종합 블록체인 교육

슈퍼브에이아이	식스샵	드림어스컴퍼니	스푼	당근	버킷플레이스
버즈빌	마켓컬리	에이비일팔공	왓차	버드뷰	채널코퍼레이션
카카오페이	카카오택스타일	42dot	카카오엔터테인먼트...	의식주컴퍼니	카카오엔터프라이즈
위대한상상	두나무	쿠팡	야놀자	하이퍼커넥트	카카오모빌리티
뱅크샐러드	카카오	우아한형제들	카카오뱅크	라인	네이버
소카	번개장터	미소	더스윙	라포랩스	미스터블루
에이아이트릭스	페이하이어	엔라이트	숨고	딜리셔스	차이코퍼레이션
					아임웹
					오누이
					폴라리스웨어테크

라이너	매서어답션	코드브릭
메타몰프	무신사	에이블리
마이지박스	커리어데이	SK플래닛
아트웍스코리아	모두싸인	매스프레스
핏펫	지바이크	스테이폴리오
SK텔레콤	오퍼스엠	루닛
외식인	우아한형제들Tech	카카오엔터테인먼
엔라이즈	스위트코리아	종고나라
퀴타랩	비바리퍼블리카	업라이즈
피플랜드컴퍼니	데브시스템즈	디셈버엔컴퍼니
파운트	레몬베이스	큐피스트
투싼월드	에듀윌	오토피디아
아임웹	오누이	폴라리스웨어테크

출처 : <https://www.codenary.co.kr/techstack/detail/go>

Golang Korea



# Firestore with Go



프로젝트 개요

설정

생성형 AI

Build with Gemini

신규

프로젝트 바로가기

Hosting

Realtime Database

Firestore Database

Messaging

Functions

Storage

Authentication

제품 카테고리

빌드

실행

애널리틱스

모든 제품

Spark

무료 \$0/월

업그레이드

realtime-chat

Realtime Database

데이터

규칙

백업

사용량

Extensions

https://realtime-chat-b57a1-default-rtdb.asia-southeast1.firebaseio.com

-01VrKGqxcvCgjMIEhZ00

-01VrMBW\_JnSpduIX2yA

-01Wamy0jwKVjIzG1ye4

-01WanOeL3UG6emE9Sro

-01Wansp12Mwk1eDyR9G

-01WbNWWzpTMS1cqSgCk

-01WbSiNWJdzMqRweFRc

-01WcAdnHD6-C2Ck3CSK

-01WcBkXjhWbWusrOfG\_

-01Wdz2BIcJrnbRWx1Q

-01Wdz5q77qC5YWmr-1Z

content: "정답"

timestamp: 1720699056

user: "Announcer"

-01WeFCzddfJVhFlgc1N

content: "gemini"

timestamp: 1720699127068

user: "JUNE"

데이터베이스 위치: 싱가포르(asia-southeast1)

프로젝트 개요

설정

생성형 AI

Build with Gemini

프로젝트 바로가기

Hosting

Realtime Database

Firestore Database

Messaging

Functions

Storage

Authentication

제품 카테고리

빌드

실행

애널리틱스

모든 제품

Spark

무료 \$0/월

업그레이드

realtime-chat

Hosting

사이트 관리

대시보드

사용량

현재 버전

이전 출시 버전

도메인

★ june@ingradient.ai

24. 7. 8. 오후 8:49

ff8f4f

★ june@ingradient.ai

24. 7. 8. 오후 8:49

ff8f4f

⬆ june@ingradient.ai

24. 7. 8. 오후 8:01

2f6bf4

⬆ june@ingradient.ai

24. 7. 8. 오후 8:00

174da2

⬆ june@ingradient.ai

24. 7. 8. 오후 7:58

2c4dac

출시 스토리지 설정

전체 기록 보기

realtime-chat-b57a1.web.app

기본값

realtime-chat-b57a1.firebaseio.com

기본값

커스텀 도메인 추가

모든 도메인 2개 보기

채널 미리보기

베타

Golang Korea



```
(base) june@ingradientubuntu:~/workspace/tutorial/realtime-chat/realtime-chat$ firebase init
(node:2226760) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
```

```
#####  #####  #####  #####  #####  #####  #####  #####
##      ##  ##      ##  ##      ##      ##  ##  ##      ##
#####   ##  #####  #####  #####  #####  #####  #####
##       ##  ##      ##  ##      ##      ##  ##      ##  ##
##       #####  ##  #####  #####  ##      ##  #####  #####
```

You're about to initialize a Firebase project in this directory:

```
/home/june/workspace/tutorial/realtime-chat/realtime-chat
```

Before we get started, keep in mind:

- \* You are initializing within an existing Firebase project directory

? Which Firebase features do you want to set up for this directory? Press Space to select features, then Enter to confirm your choices. (Press <space> to select, <a> to toggle all, <i> to invert selection, and <enter> to proceed)

X) Realtime Database: Configure a security rules file for Realtime Database and (optionally) provision default instance

- () Firestore: Configure security rules and indexes files for Firestore
- () Functions: Configure a Cloud Functions directory and its files
- () Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys
- () Hosting: Set up GitHub Action deploys
- () Storage: Configure a security rules file for Cloud Storage

(Move up and down to reveal more choices)



```
● (base) june@ingradientubuntu:~/workspace/tutorial/realtime-chat/realtime-chat$ firebase deploy --only hosting
(node:2225891) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)

=== Deploying to 'realtime-chat-b57a1'...

i  deploying hosting
i  hosting[realtime-chat-b57a1]: beginning deploy...
i  hosting[realtime-chat-b57a1]: found 15 files in build
✓  hosting[realtime-chat-b57a1]: file upload complete
i  hosting[realtime-chat-b57a1]: finalizing version...
✓  hosting[realtime-chat-b57a1]: version finalized
i  hosting[realtime-chat-b57a1]: releasing new version...
✓  hosting[realtime-chat-b57a1]: release complete

✓  Deploy complete!

Project Console: https://console.firebase.google.com/project/realtime-chat-b57a1/overview
Hosting URL: https://realtime-chat-b57a1.web.app
```

```
❖ (base) june@ingradientubuntu:~/workspace/tutorial/realtime-chat/go-server$ go run main.go
Listening for messages...
Messages:
[04:15:49] JUNE: sdf
[14:21:55] JUNE: test message
[20:37:53] june: 바보
[19:44:04] JUNE: gemini
[22:30:03] june: test image
[05:01:29] JUNE: test stest
[07:39:58] june: 바보
[11:29:04] tester: asdf
[01:05:31] 이준호: 바보
[18:20:05] june: test
[11:34:00] june: test
[13:17:18] hello2: 요훗
```





FrontEnd : n명

BackEnd : n명

weniv0107@gmail.com

# Q&A



**Thank you!**