널포유 10201 구본무

O R M

#gin #ServerLessDB

ORM을 쓰는 이유

개발자

```
INSERT INTO users (username, password)
VALUES (admin,123456);
```

ORM을 쓰는 이유 컬럼추가

개발자

```
INSERT INTO users (username, password)
VALUES (admin,123456);
```

users 테이블에 email 컬럼 추가됐습니다!

ORM을 쓰는 이유 컬럼추가

코드 계속 수정해야함

개발자

```
INSERT INTO users (username, email, password)
VALUES (admin,myemail@com ,123456);
```

users 테이블에 email 컬럼 추가됐습니다!

ORM을 쓰는 이유 노가다

컬럼이 너무 많아(노가다)

개발자

```
INSERT INTO users (username, email, password, age, phone, address)
VALUES (admin,myemail@com ,123456, 67, 01082714229,eunpyung);
```

ORM을 쓰는 이유 DBMS 변경

개발자

SELECT CONCAT(first_name, last_name) FROM users;

저희 DBMS mysql에서 포스트그레스로 바꿉니다

ORM을 쓰는 이유 DBMS 변경

문법 갈아엎기 이런게 100개 더 있으면…..

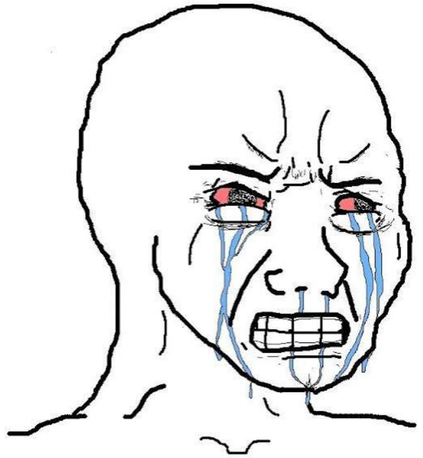
개발자

```
SELECT first_name || last_name FROM users;
```

저희 DBMS mysql에서 포스트그레스로 바꿉니다

ORM을 쓰는 이유 DBMS 변경

문법 갈아엎기 이런게 100개 더 있으면…..



개발자

SELECT first_name || last_name FROM users;

저희 DBMS mysql에서 포스트그레스로 바꿉니다

ORM으로 해결할수있다!

Object-**R**elational **M**apping

각 언어별로 유명한 ORM들

JPA

Prisma

Django

각 언어별로 유명한 ORM들

JPA

스프링부트전용 ORM

Prisma

Django

각 언어별로 유명한 ORM들

JPA

스프링부트전용 ORM

Prisma

Django

Nodejs 계열 웹프레임워크 사용 가능

각 언어별로 유명한 ORM들

JPA

스프링부트전용 ORM

Prisma

Django

프레임워크 내에 내장되어있음

Nodejs 계열 웹프레임워크 사용 가능

각 언어별로 유명한 ORM들

Go 전용 ORM GORM

진 프레임워크란?

Go 언어 전용 웹 프레임워크

진 프레임워크란?

VS

스프링부트에 비해 내용이 얕고 문법이 간단한편

진 프레임워크란?

VS

스프링부트에 비해 내용이 얕고 문법이 간단한편

Gin (Go)	**약 10MB ~ 25MB**
Spring Boot (Java)	**약 300MB ~ 500MB**

진 프레임워크란?

스레드

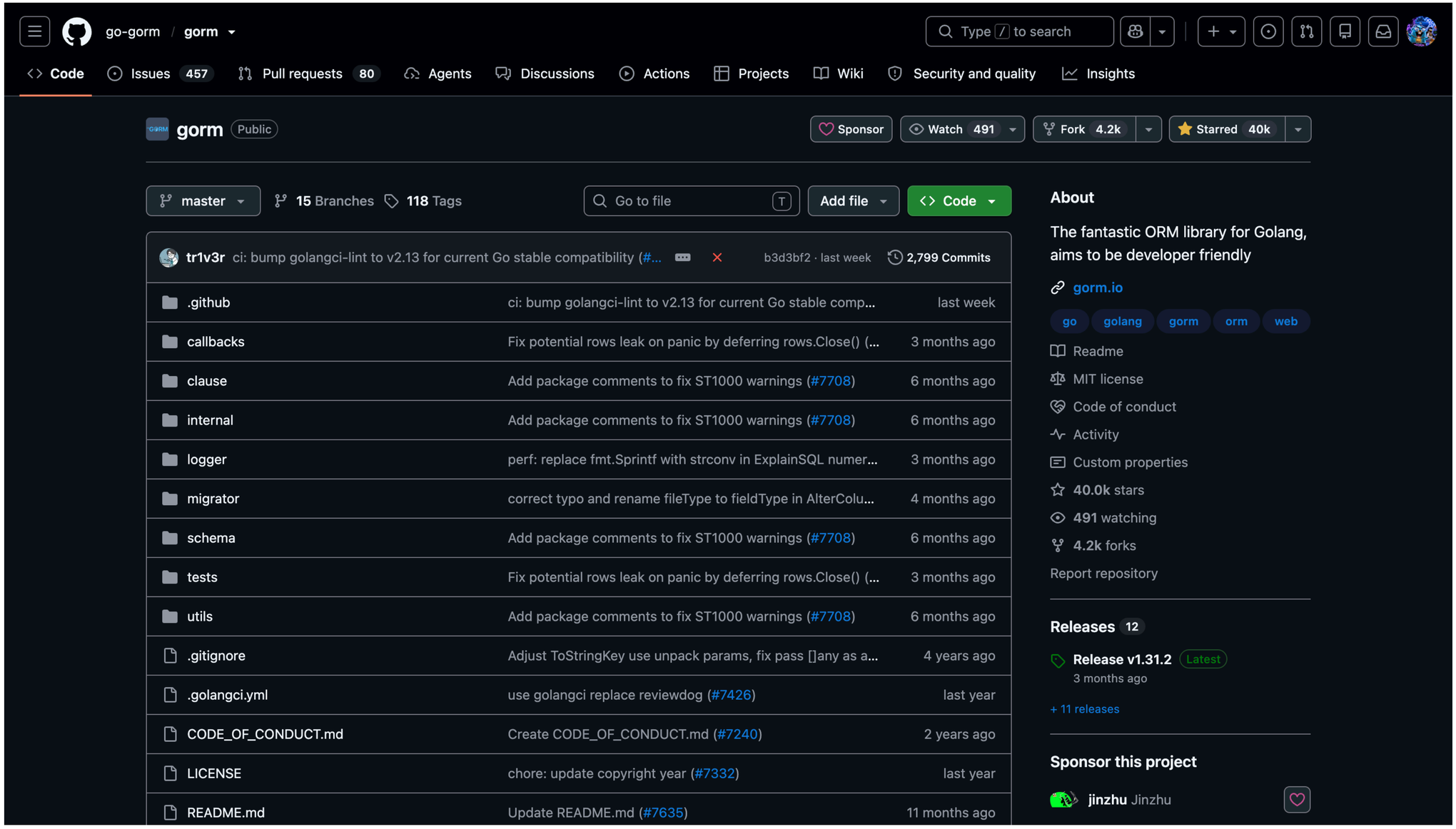
VS

고루틴

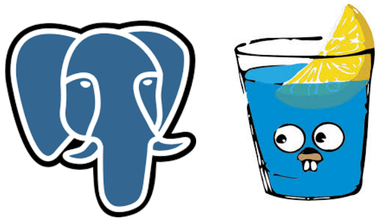
JVM

GORM사용해보기

github.com/go-gorm/gorm

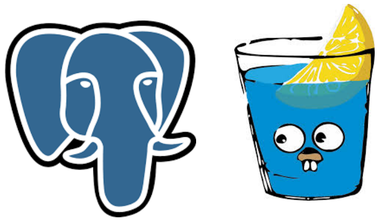


GORM사용해보기



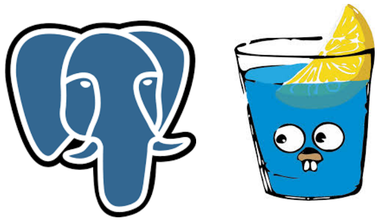
```
gin_study=> SELECT * FROM users;
 id | username |        email         |            create_at
----+----------+----------------------+-------------------------------
  1 | 구본무    | kubonmo519@gmail.com | 2026-09-14 20:06:21.893633+09
```

GORM사용해보기



```
gin_study=> SELECT * FROM users;
 id | username |        email         |            create_at
----+----------+----------------------+-------------------------------
  1 | 구본무    | kubonmo519@gmail.com | 2026-09-14 20:06:21.893633+09
```

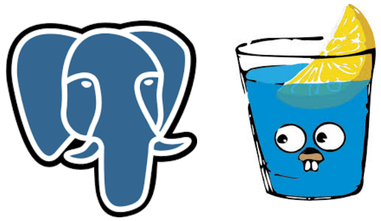
GORM사용해보기



구조체 태그 지정

```go
type User struct {
    ID       int64     `gorm:"primaryKey" json:"id"`
    Username string    `gorm:"column:user_name" json:"username"`
    Email    string    `json:"email"`
    CreateAt time.Time `json:"create_at"`
}
```

GORM사용해보기

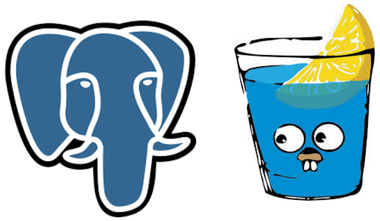


구조체 태그 지정

```go
type User struct {
    ID       int64     `gorm:"primaryKey" json:"id"`
    Username string    `gorm:"column:user_name" json:"username"`
    Email    string    `json:"email"`
    CreateAt time.Time `json:"create_at"`
}
```

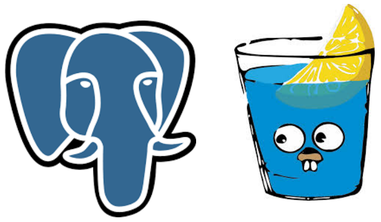
GIN에서 구조체는 많은 역할을 함,

GORM사용해보기



```go
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {
    rows, err := s.db.Query(ctx, `SELECT id, username, email, create_at FROM users ORDER BY id`)
    if err != nil {
        return nil, err
    }
    defer rows.Close()
    users := make([]model.User, 0)
    for rows.Next() {
        var u model.User
        if err := rows.Scan(&u.ID, &u.Username, &u.Email, &u.CreateAt); err != nil {
            return nil, err
        }
        users = append(users, u)
    }
    return users, rows.Err()
}
```

GORM사용해보기

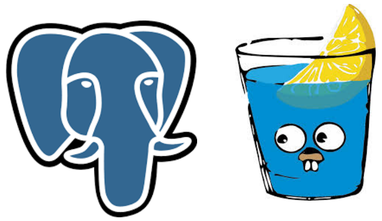


```go
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {
    rows, err := s.db.Query(ctx, `SELECT id, username, email, create_at FROM users ORDER BY id`)
    if err != nil {
        return nil, err
    }
    defer rows.Close()
    users := make([]model.User, 0)
    for rows.Next() {
        var u model.User
        if err := rows.Scan(&u.ID, &u.Username, &u.Email, &u.CreateAt); err != nil {
            return nil, err
        }
        users = append(users, u)
    }
    return users, rows.Err()
}
```

너무 길고 복잡해요!

```go
func (h *UserHandler) ListUsers(c *gin.Context) {
    users, err := h.service.ListUsers(c.Request.Context())
    if err != nil {
        c.JSON(500, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, users)
}
```

GORM사용해보기

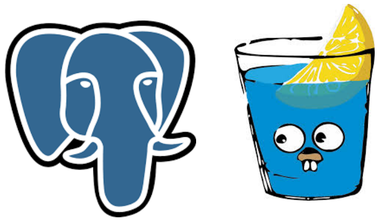


```go
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {
    users := make([]model.User, 0)
    err := s.db.WithContext(ctx).Order("id").Find(&users).Error
    return users, err
}
```

엄청 짧아짐

```go
func (h *UserHandler) ListUsers(c *gin.Context) {
    users, err := h.service.ListUsers(c.Request.Context())
    if err != nil {
        c.JSON(500, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, users)
}
```

GORM사용해보기(차이점)

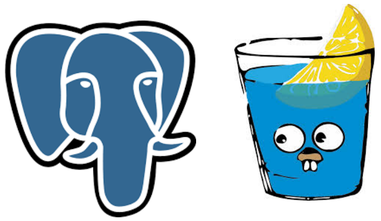


```go
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {
    users := make([]model.User, 0)
    err := s.db.WithContext(ctx).Order("id").Find(&users).Error
    return users, err
}
```

쿼리문이 어디갔지???

```go
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {
    rows, err := s.db.Query(ctx, `SELECT id, username, email, create_at FROM users ORDER BY id`)
    if err != nil {
        return nil, err
    }
    defer rows.Close()
    users := make([]model.User, 0)
    for rows.Next() {
        var u model.User
        if err := rows.Scan(&u.ID, &u.Username, &u.Email, &u.CreateAt); err != nil {
            return nil, err
        }
        users = append(users, u)
    }
    return users, rows.Err()
}
```

GORM사용해보기(차이점)

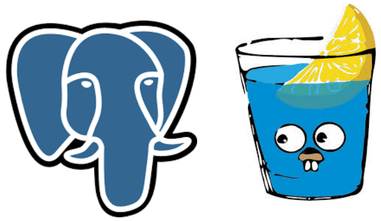


```go
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {
    users := make([]model.User, 0)
    err := s.db.WithContext(ctx).Order("id").Find(&users).Error
    return users, err
}
```

SELECT * FROM users ORDER BY id;

```go
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {
    rows, err := s.db.Query(ctx, `SELECT id, username, email, create_at FROM users ORDER BY id`)
    if err != nil {
        return nil, err
    }
    defer rows.Close()
    users := make([]model.User, 0)
    for rows.Next() {
        var u model.User
        if err := rows.Scan(&u.ID, &u.Username, &u.Email, &u.CreateAt); err != nil {
            return nil, err
        }
        users = append(users, u)
    }
    return users, rows.Err()
}
```

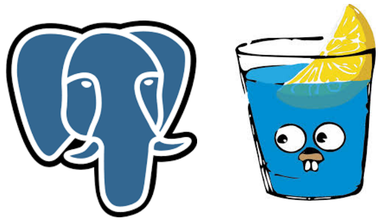
GORM사용해보기(차이점)



```go
type User struct {
    ID       int64     `gorm:"primaryKey" json:"id"`
    Username string    `gorm:"column:user_name" json:"username"`
    Email    string    `json:"email"`
    CreateAt time.Time `json:"create_at"`
}
```

User 모델의 여러 행을 조회해서 이 슬라이스에 넣어야 하는구나
ORM이 판단 가능

GORM사용해보기



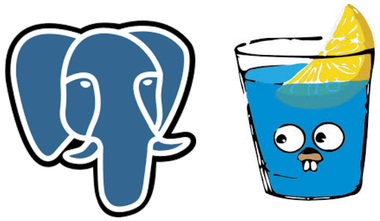
ORM 패키지 함수들 총집합

https://gorm.io/ko_KR/docs/query.html

GORM사용해보기 - 메서드들

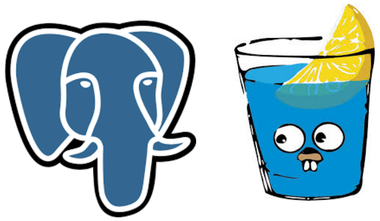
users는 슬라이스(배열/리스트) 변수

db.Find(&users)
테이블의 여러 행 조회



GORM사용해보기 - 메서드들

users는 슬라이스(배열/리스트) 변수

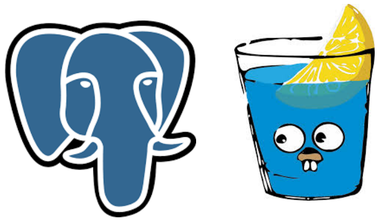


db.Find(&users)
테이블의 여러 행 조회

db.First(&user)
첫번째 행 조회

GORM사용해보기 - 메서드들

users는 슬라이스(배열/리스트) 변수

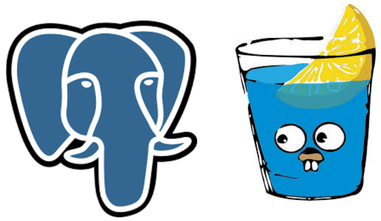


db.Find(&users)
테이블의 여러 행 조회

db.First(&user)
첫번째 행 조회

db.First(&user, id)
ID로 한 행 조회

GORM사용해보기 - 메서드들



users는 슬라이스(배열/리스트) 변수

db.Find(&users)
테이블의 여러 행 조회

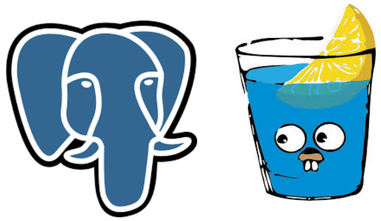
db.First(&user)
첫번째 행 조회

db.First(&user, id)
ID로 한 행 조회

db.Where("email = ?", email).First(&user)
조건에 맞는 첫 행 조회

GORM사용해보기 - 메서드들

users는 슬라이스(배열/리스트) 변수



db.Find(&users)
테이블의 여러 행 조회

db.First(&user)
첫번째 행 조회

db.First(&user, id)
ID로 한 행 조회

db.Where("email = ?", email).First(&user)
조건에 맞는 첫 행 조회

db.Order("id DESC").Find(&users)
정렬해서 조회

사용하다 느낀점

시퀄을 좀더 배워야겠다
sqld를 따야하나?…

서버리스 NEONDB

서버리스란?

자기가 직접 관리 안해도 된다!
과금 문제 그런거 신경 안써도됨 AWS에서 치면 람다

서버리스 NEONDB

서버리스란?

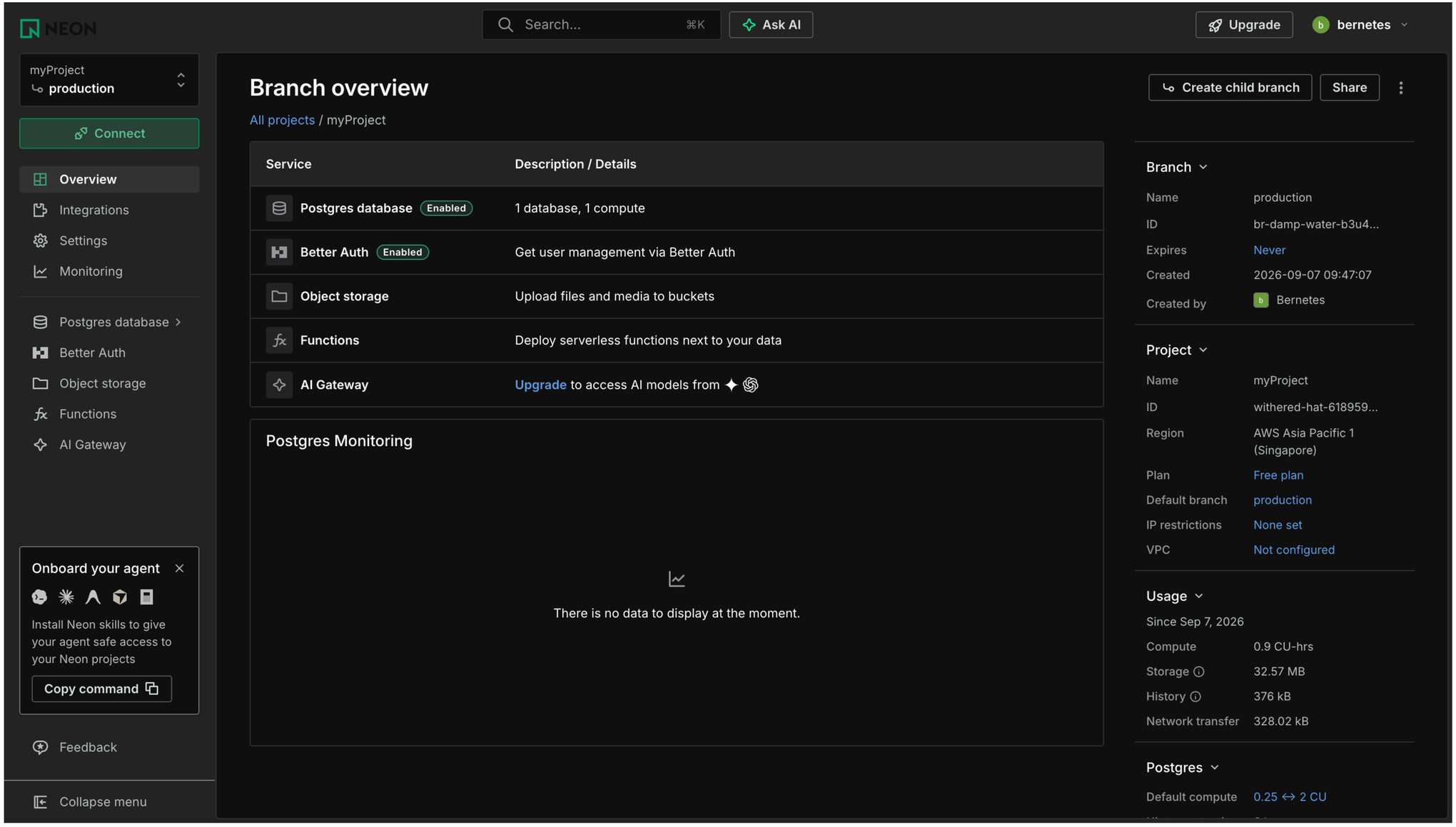
자기가 직접 관리 안해도 된다!
과금 문제 그런거 신경 안써도됨 AWS에서 치면 람다
관리 안해도 되는 DB가 있다면?

서버리스 NEONDB

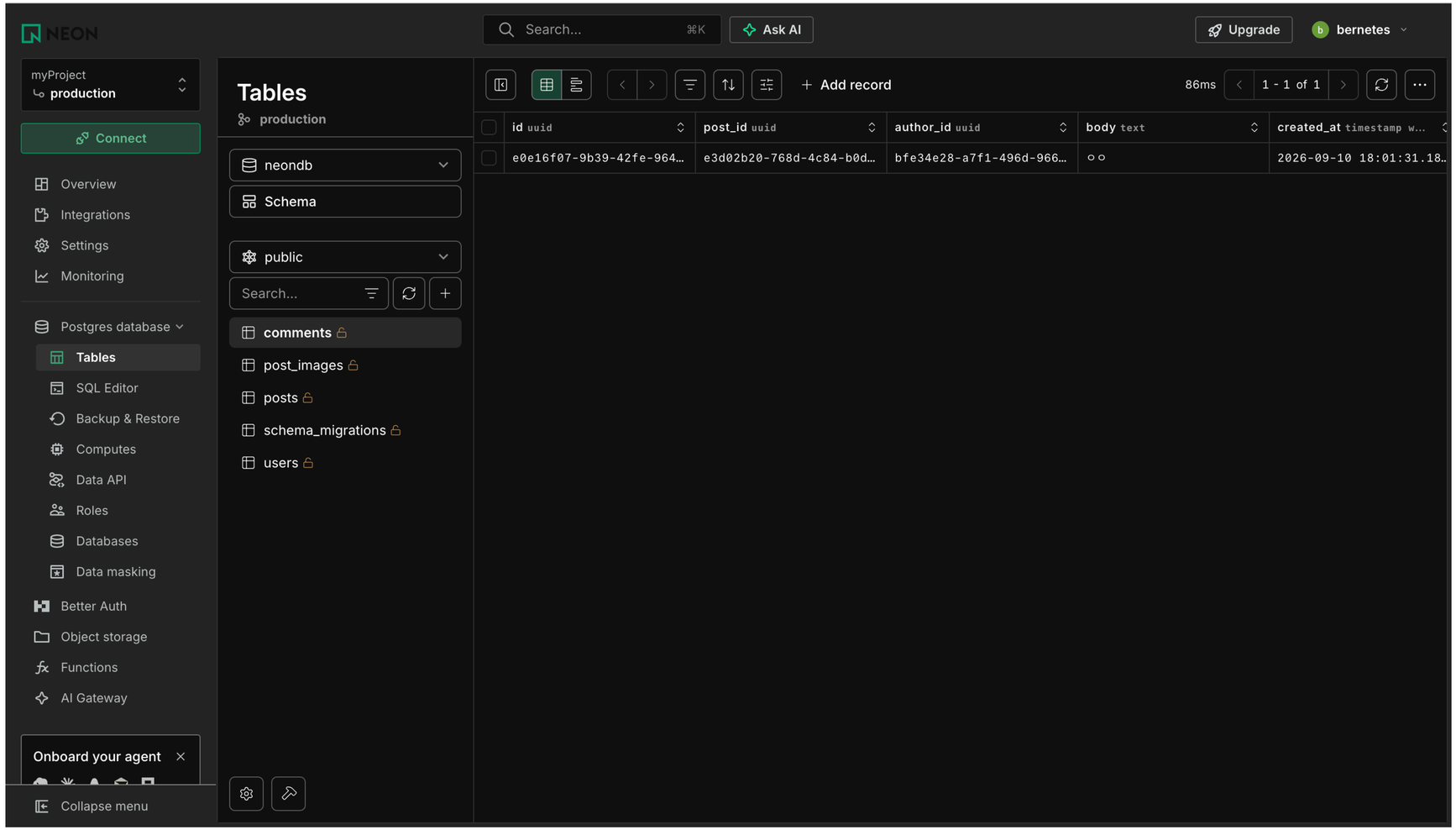
서버리스란?

자기가 직접 관리 안해도 된다!
과금 문제 그런거 신경 안써도됨 AWS에서 치면 람다
관리 안해도 되는 DB가 있다면?

서버리스 NEONDB

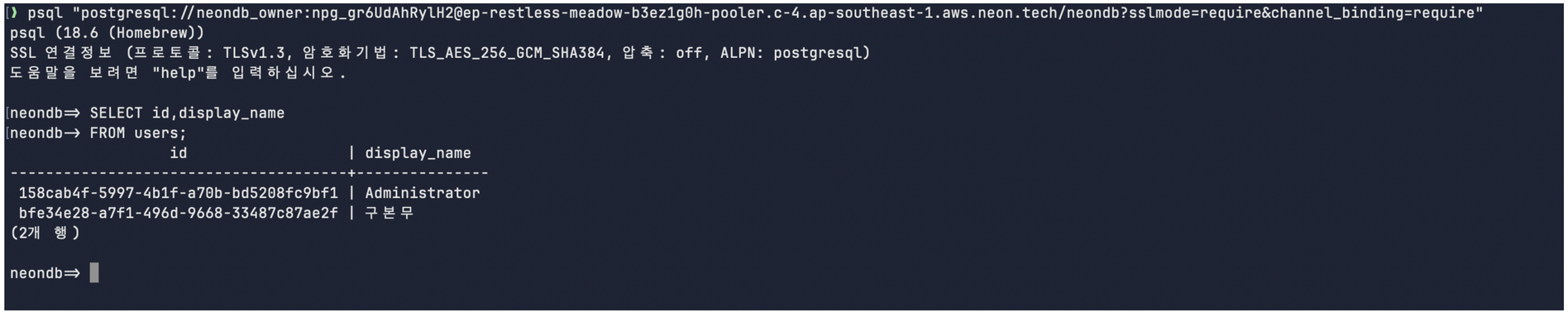

NEON
Search... ⌘K
Ask AI
Upgrade
bernetes
myProject
production
Connect
Overview
Integrations
Settings
Monitoring
Postgres database
Better Auth
Object storage
Functions
AI Gateway
Onboard your agent
Install Neon skills to give your agent safe access to your Neon projects
Copy command
Feedback
Collapse menu
Branch overview
All projects / myProject
Create child branch
Share
Service
Description / Details
Postgres database Enabled
1 database, 1 compute
Better Auth Enabled
Get user management via Better Auth
Object storage
Upload files and media to buckets
Functions
Deploy serverless functions next to your data
AI Gateway
Upgrade to access AI models from
Postgres Monitoring
There is no data to display at the moment.
Branch
Name production
ID br-damp-water-b3u4...
Expires Never
Created 2026-09-07 09:47:07
Created by Bernetes
Project
Name myProject
ID withered-hat-618959...
Region AWS Asia Pacific 1 (Singapore)
Plan Free plan
Default branch production
IP restrictions None set
VPC Not configured
Usage
Since Sep 7, 2026
Compute 0.9 CU-hrs
Storage 32.57 MB
History 376 kB
Network transfer 328.02 kB
Postgres
Default compute 0.25 ↔ 2 CU

서버리스 NEONDB



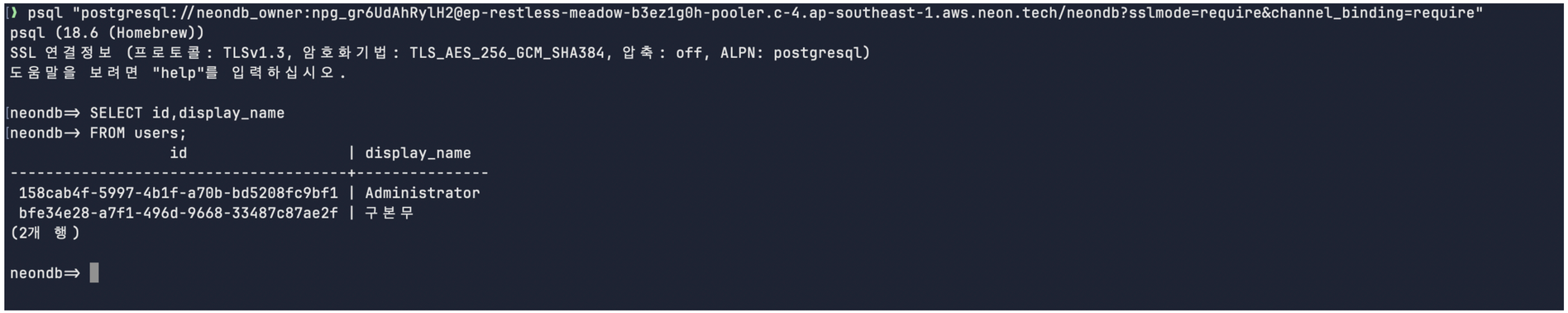
영알못이 아닌 이상 엄청나게 쉽게 사용 가능 (스키마 화면)

서버리스 NEONDB



터미널에서도 당연히 접근 가능하다.

서버리스 NEONDB



터미널에서도 당연히 접근 가능하다.

서버리스 NEONDB

Provider and region availability

Neon runs on AWS. Lakebase Postgres on Databricks inherits the cloud reach of the Databricks platform, with availability that varies by provider. For the full, current region lists, follow the links below.

Cloud provider	Neon	Databricks
AWS	Yes (Neon regions)	Yes, generally available (AWS regions ↗)
Azure	No, Azure support is being deprecated (Neon regions)	Yes, in beta (Azure regions ↗)
GCP	Not available	Yes, in beta (GCP regions ↗)

AWS환경에서도 있다.

널포유 10201 구본무

O R M

#gin #ServerLessDB