

Easy Note

CloudWeGo 生态入门

李宜茗 justlorain

2023/03/08



- 李宜茗
(<https://github.com/justlorain>)
- 在校大学生
- CloudWeGo – Hertz Reviewer
- 喜欢看动漫和睡觉



如何入门 CloudWeGo 生态

Hello, CloudWeGo!



Hertz, Kitex, GORM 介绍

HTTP, RPC, ORM



Easy Note 解读

biz-demo



01

如何入门 CloudWeGo 生态

Hello, CloudWeGo!

Tip 01: 官方文档

遇事不决，先查文档

<https://www.cloudwego.io/zh/> <https://github.com/cloudwego/cloudwego.github.io>

- CloudWeGo 具有非常完善的中文（英文）文档，文档的维护和更新都非常及时；
- 如果你想快速入门一款 CloudWeGo 产品，那么就去查看对应文档的 Quick Start 吧！
- 在 GitHub 上也有官网对应的仓库，可以随时提交文档问题，PR；

The screenshot shows the CloudWeGo official website documentation page. The header includes the CloudWeGo logo and navigation links: 文档, 关于, 博客, 社区, 用户案例, 安全, 中文. The left sidebar lists the documentation structure: 文档, Hertz (Overview, Quick Start, Guide, Examples, Basic Features, Governance, Framework Extension, hz Command Line Tools, Common Questions, References), and References (Configuration, Version, JSON Marshal, Migration, Hertz Support QUIC & HTTP/3). The main content area is titled '快速开始' (Quick Start) and '准备 Golang 开发环境' (Prepare Golang Development Environment). It lists three steps: 1. Check if Golang is installed, 2. Recommend the latest version of Golang, and 3. Ensure go mod support is enabled. It also mentions that Hertz supports Linux, macOS, and Windows systems.

The screenshot shows the CloudWeGo GitHub repository page. The header includes the repository name 'cloudwego / cloudwego.github.io' and navigation links: Code, Issues, Pull requests, Actions, Security, Insights. The left sidebar shows the repository structure: .github, assets, content, i18n, layouts, static, .editorconfig, .gitignore, .hugo_build.lock, .nojekyll, .prettierrc, .prettierrc.json, CONTRIBUTING.md. The main content area shows the commit history, including a recent commit 'Duslia docs: update url in cwgo layout zh docs (#551)' and a list of contributors.

Tip 02: Examples

e.g. <https://github.com/cloudwego/hertz-examples>

<https://github.com/cloudwego/kitex-examples>

- 如果查看文档后仍然觉得没有完全了解一些特性，那么可以查看产品对应的 Examples 仓库；
- Examples 仓库一般会包含各个主要特性的更加细致的使用方法，为你提供一些使用示例参考；

Hertz Examples

English | 中文

How to run

You can enter the example for information about "How to run"

Bizdemo

- [bizdemo/hertz_gorm](#): Example of using gorm in hertz server
- [bizdemo/hertz_gorm_gen](#): Example of using gorm/gen & proto IDL in hertz server
- [bizdemo/hertz_jwt](#): Example of using jwt in hertz server
- [bizdemo/hertz_session](#): Example of using distributed session and csrf in hertz server

Server

- [hello](#): Example of launching a hertz "hello world" application
- [config](#): Example of configuring hertz server
- [protocol](#): Example of using http1, tls and other protocols of hertz
- [middleware](#): Example of using middleware of hertz
 - [basicauth](#): Example of using BasicAuth middleware
 - [cors](#): Example of using CORS middleware
 - [csrf](#): Example of using csrf middleware
 - [custom](#): Example of using custom middleware
 - [pprof](#): Example of using pprof middleware
 - [requestid](#): Example of using RequestID middleware
 - [gzip](#): Example of using Gzip middleware
 - [bindings](#): Example of parameter binding and validation

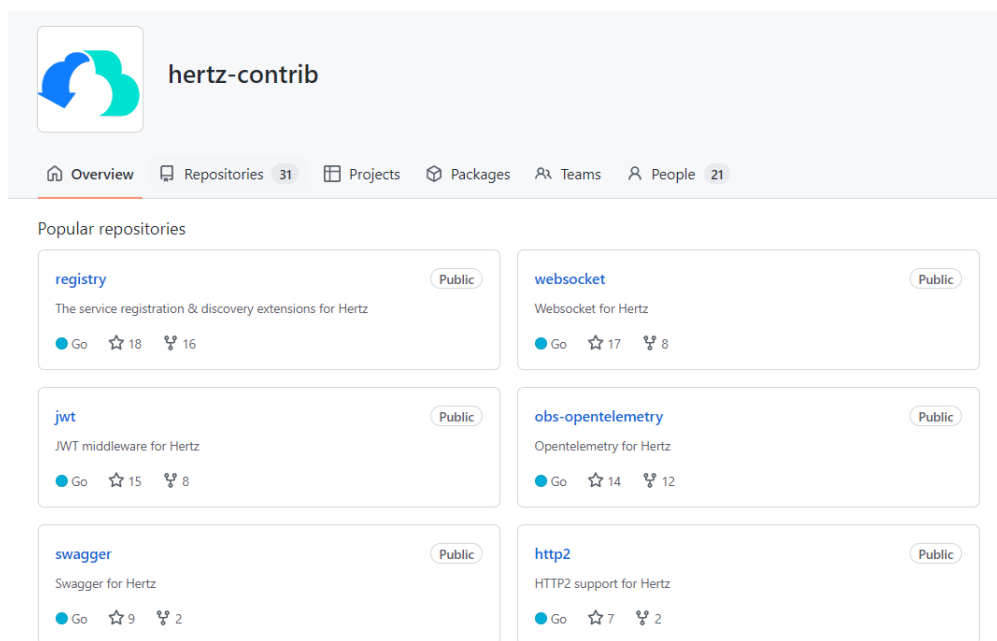
The screenshot shows the GitHub repository page for `cloudwego/kitex-examples`. The repository is public and has 12 watchers, 182 forks, and 292 stars. The main branch is `main`, with 2 branches and 2 tags. The repository contains a list of example directories, each with a description and a commit hash. The directories are: `.github`, `async_call`, `basic`, `bizdemo/easy_note`, `codec`, `discovery`, `generic`, `governance`, `grpcproxy`, `hello`, `kitex_gen`, `licenses`, and `loadbalancer`. The repository also has a README, Apache-2.0 license, Code of conduct, 292 stars, 12 watching, and 182 forks. The latest release is `release v0.11` on Dec 21, 2022. There are 2 packages published and 32 uses.

Directory	Description	Commit Hash
<code>.github</code>	chore: fix CODEOWNERS team name	2 months ago
<code>async_call</code>	feat: add async call example (#32)	8 months ago
<code>basic</code>	feat: use info level to print log (#24)	last year
<code>bizdemo/easy_note</code>	chore: upgrade dependency version for k8s.io/client-go and others (#63)	2 weeks ago
<code>codec</code>	feat: use info level to print log (#24)	last year
<code>discovery</code>	feat: use info level to print log (#24)	last year
<code>generic</code>	feat: use info level to print log (#24)	last year
<code>governance</code>	feat: use info level to print log (#24)	last year
<code>grpcproxy</code>	chore: add grpc proxy example (#37)	8 months ago
<code>hello</code>	use kitex v0.1.4 to generate code.	last year
<code>kitex_gen</code>	use kitex v0.1.4 to generate code.	last year
<code>licenses</code>	feat: add easy demo (#22)	last year
<code>loadbalancer</code>	feat: use info level to print log (#24)	last year

Tip 03: 扩展仓库

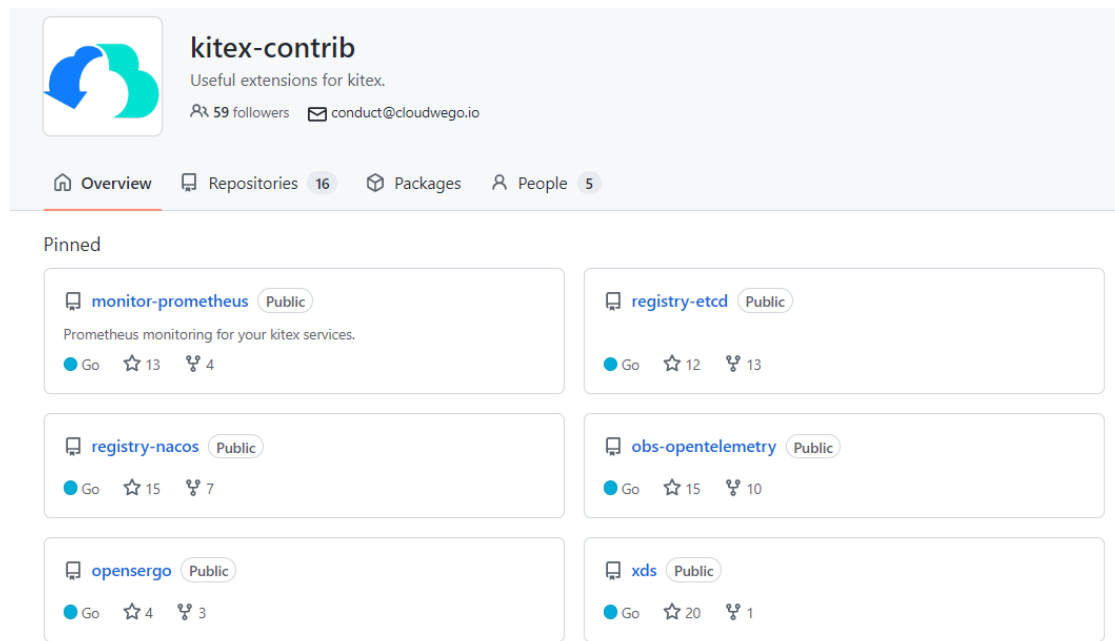
e.g. <https://github.com/hertz-contrib> <https://github.com/kitex-contrib>

- 产品的扩展一般会单独分离出仓库维护，例如 Hertz 的扩展都维护在 hertz-contrib 下；
- 一般产品都会实现一些常用的扩展而不需要用户自己进行开发，例如 Kitex 和 Hertz 对常见的服务发现与注册中心进行了支持（nacos,etcd,consul...），Hertz 对常用的 Web 中间件进行了支持（jwt,gzip,sessions...）；
- 并且大多数扩展仓库都会附带对应的使用示例，方便用户上手；
- 在使用常用的功能扩展时不妨先看看这些扩展仓库：)



The screenshot shows the GitHub profile for hertz-contrib. The header includes the repository name and a navigation bar with links to Overview, Repositories (31), Projects, Packages, Teams, and People (21). Below the header, the 'Popular repositories' section displays six repositories in a 3x2 grid:

Repository Name	Description	Go	Stars	Forks
registry	The service registration & discovery extensions for Hertz	Go	18	16
websocket	Websocket for Hertz	Go	17	8
jwt	JWT middleware for Hertz	Go	15	8
obs-opentelemetry	Opentelemetry for Hertz	Go	14	12
swagger	Swagger for Hertz	Go	9	2
http2	HTTP2 support for Hertz	Go	7	2



The screenshot shows the GitHub profile for kitex-contrib. The header includes the repository name, a description 'Useful extensions for kitex.', 59 followers, and an email address. The navigation bar shows Overview, Repositories (16), Packages, and People (5). Below the header, the 'Pinned' section displays six repositories in a 3x2 grid:

Repository Name	Description	Go	Stars	Forks
monitor-prometheus	Prometheus monitoring for your kitex services.	Go	13	4
registry-etcd		Go	12	13
registry-nacos		Go	15	7
obs-opentelemetry		Go	15	10
opensergo		Go	4	3
xds		Go	20	1

Tip 04: 关于提问

- 查询引擎，各大技术社区，官网，GitHub 历史 issue，ChatGPT...
- 大多数的常见问题都能通过以上方式查询到解决方案，如果都解决不了那么欢迎提 issue 这样一方面可以对问题进行记录，方便之后遇到相同问题的同学，一方面也帮助产品本身进一步完善成长；

Google

 stack overflow


GitHub

 OpenAI

 掘金

Bai  百度



02

Hertz, Kitex, GORM 介绍

HTTP, RPC, ORM

框架概述



Hertz

Hertz 是字节开源的 HTTP 框架，参考了其他开源框架的优势，结合字节跳动内部的需求，具有高易用，高性能，高扩展性的特点。



Kitex

Kitex 是字节开源的 Golang 微服务 RPC 框架，具有高性能，强可扩展的主要特点，支持多协议并且拥有丰富的开源扩展。

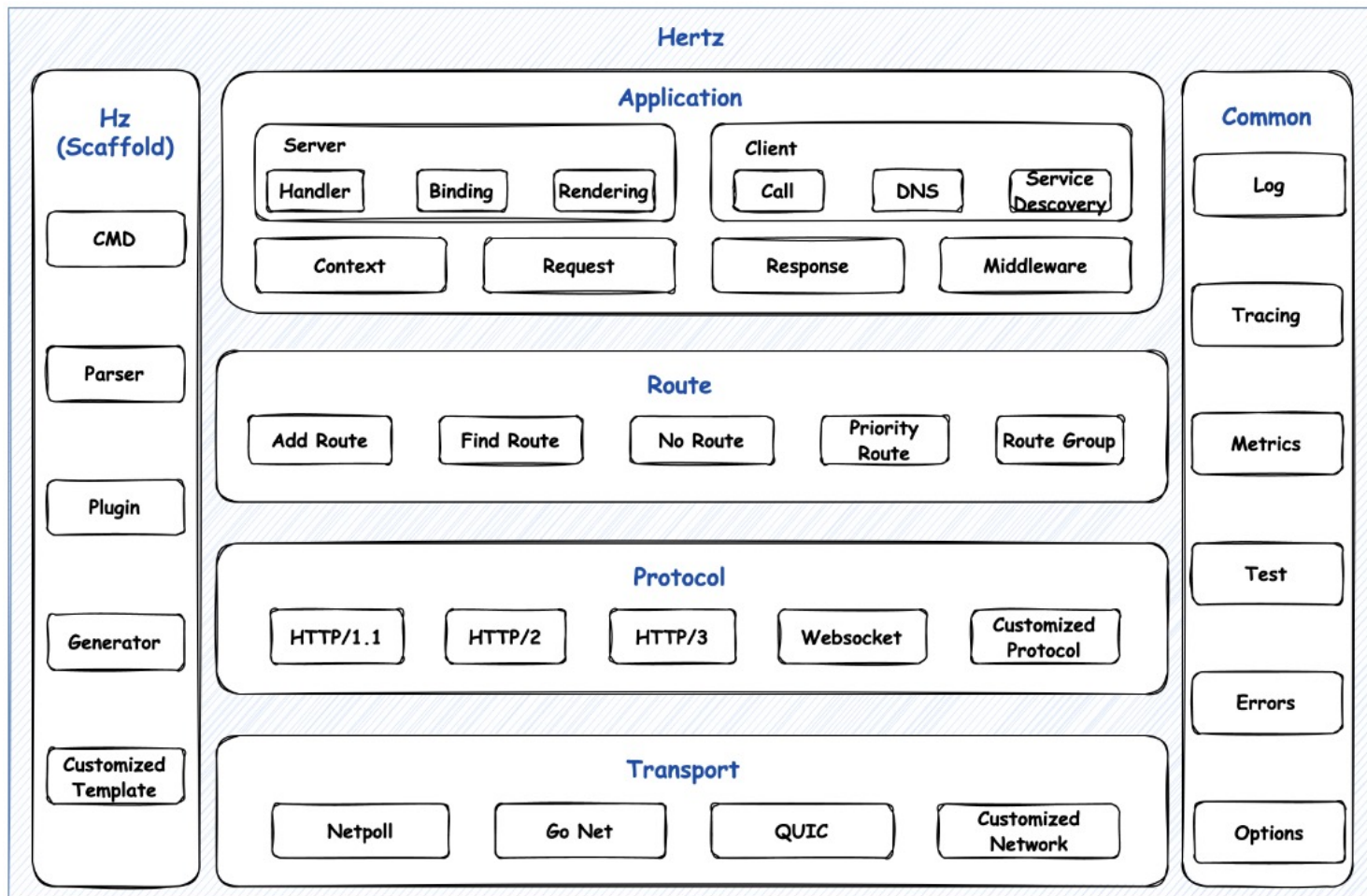
**GORM**

GORM

GORM 是一个已经迭代了10年+的功能强大的 ORM 框架，在字节内部被广泛使用并且拥有非常丰富的开源扩展。

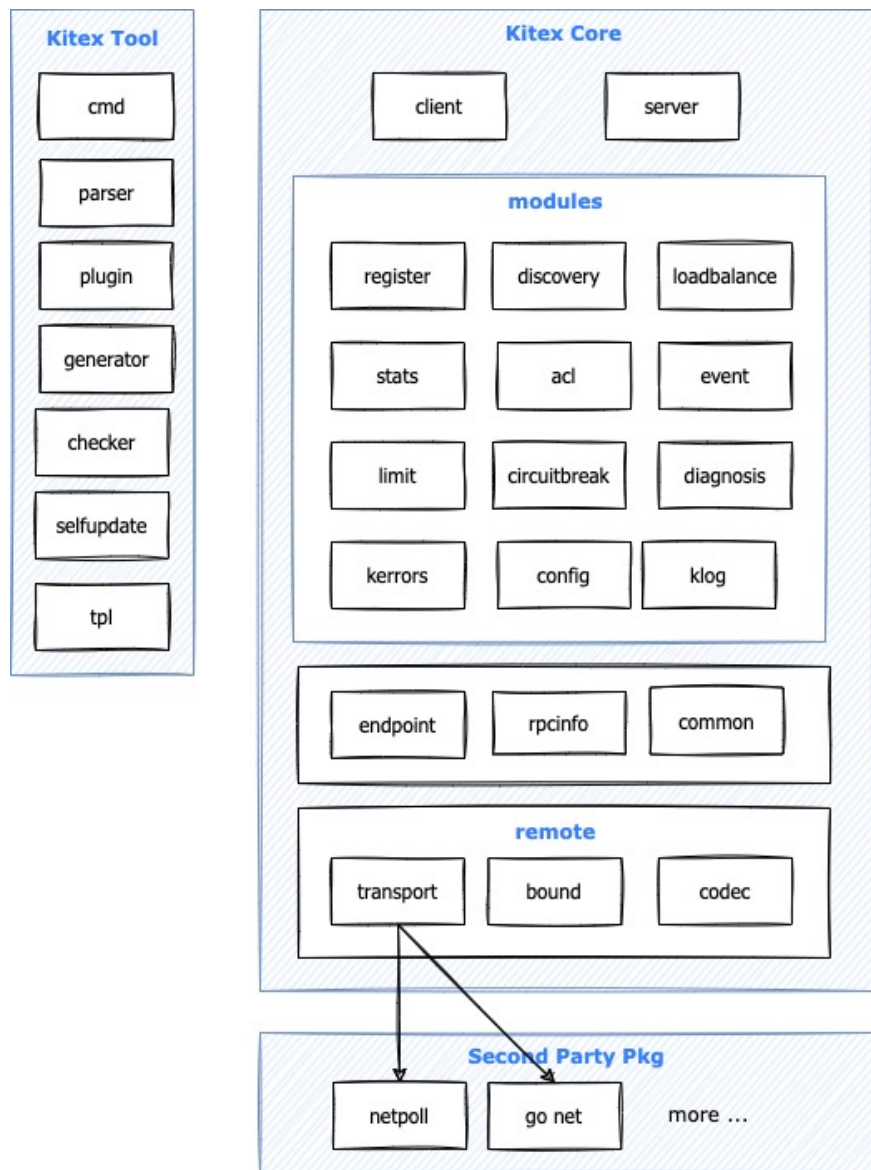
Hertz - 架构设计

<https://www.cloudwego.io/zh/docs/hertz/overview/>



- 高易用性
- 高性能
- 高扩展性
- 多协议支持
- 网络层切换能力

Kitex



<https://www.cloudwego.io/zh/docs/kitex/overview/>

- 高性能
- 扩展性
- 多消息协议
- 多传输协议
- 多种消息类型
- 服务治理
- 代码生成

GORM

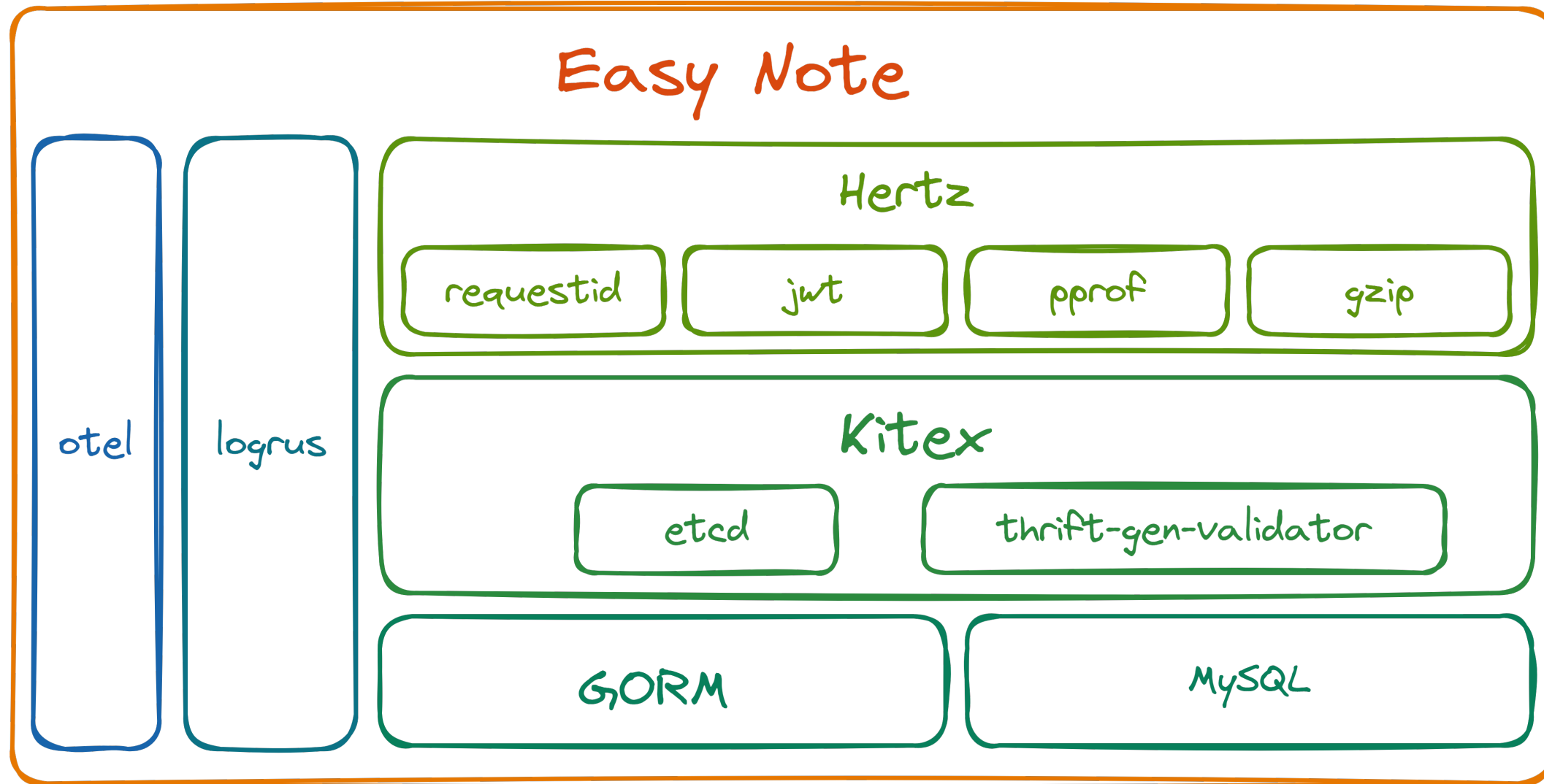
- 全功能 ORM
- 关联 (has one, has many, belongs to, many to many, 多态, 单表继承)
- Create, Save, Update, Delete, Find 中钩子方法
- 支持 Preload、Joins 的预加载
- 事务, 嵌套事务, Save Point, Rollback To to Saved Point
- Context、预编译模式、DryRun 模式
- 批量插入, FindInBatches, Find/Create with Map, 使用 SQL 表达式、Context Valuer 进行
- SQL 构建器, Upsert, 锁, Optimizer/Index/Comment Hint, 命名参数, 子查询
- 复合主键, 索引, 约束
- Auto Migration
- 自定义 Logger
- 灵活的可扩展插件 API: Database Resolver (多数据库, 读写分离)、Prometheus...
- 每个特性都经过了测试的重重考验
- 开发者友好



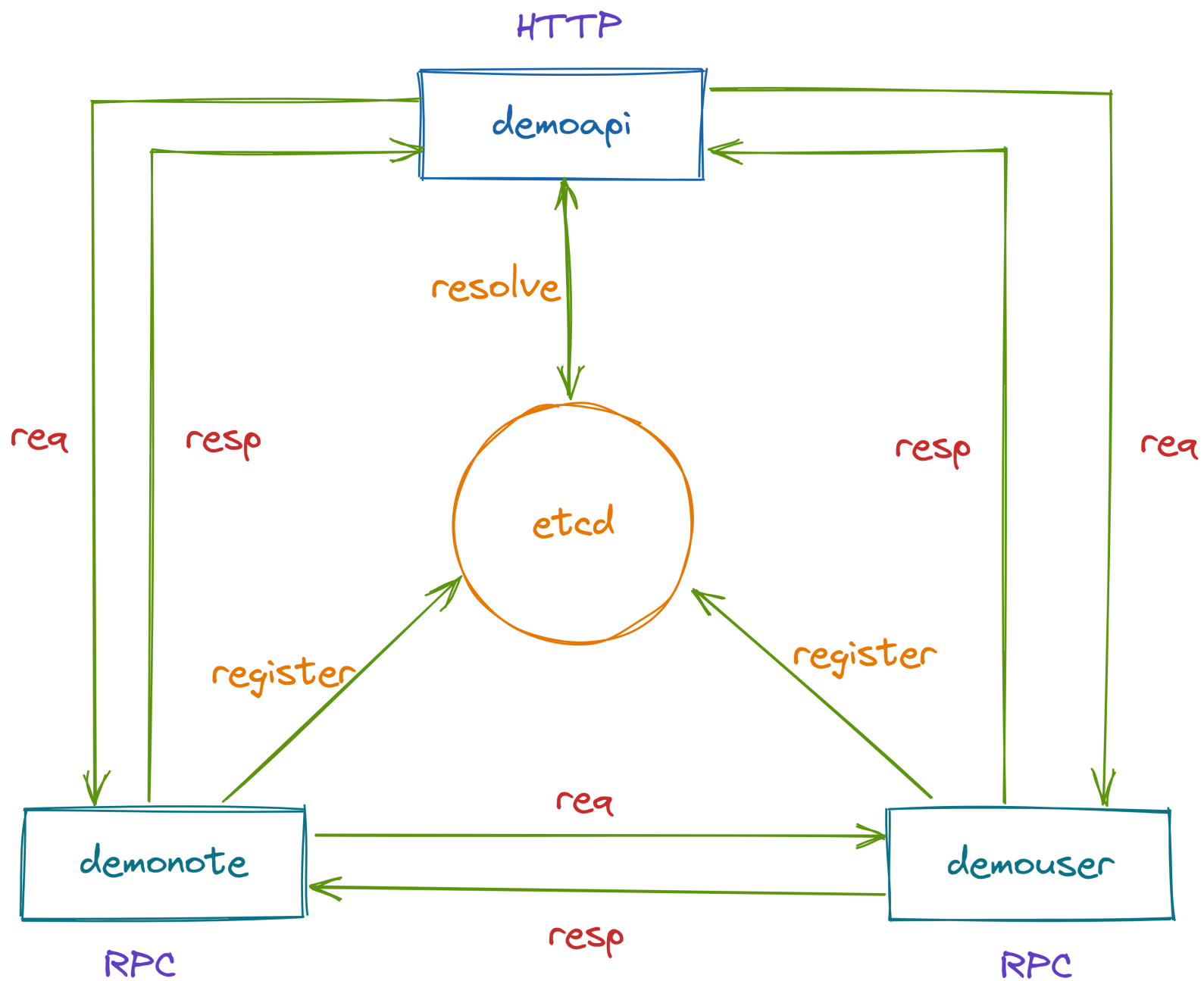
03

Easy Note 解读

biz-demo



调用关系



目录介绍

目录	介绍
handler	HTTP handlers
service	主要业务逻辑
rpc	RPC 调用逻辑
dal	数据库操作
pack	数据包装
pkg/mw	RPC 中间件
pkg/consts	常量
pkg/errno	自定义错误码
pkg/configs	SQL, Tracing 配置

cmd/api: demoapi

使用 Hertz 搭建的 HTTP 服务，为用户直接提供服务，通过 RPC 调用 demouser 和 demonote 服务；

cmd/user: demouser

使用 Kitex 和 GROM 搭建的 RPC 服务，负责用户的创建，查询，校验等；

cmd/note: demonote

使用 Kitex 和 GORM 搭建的 RPC 服务，负责笔记的增删改查等；

开发流程 – 定义 IDL

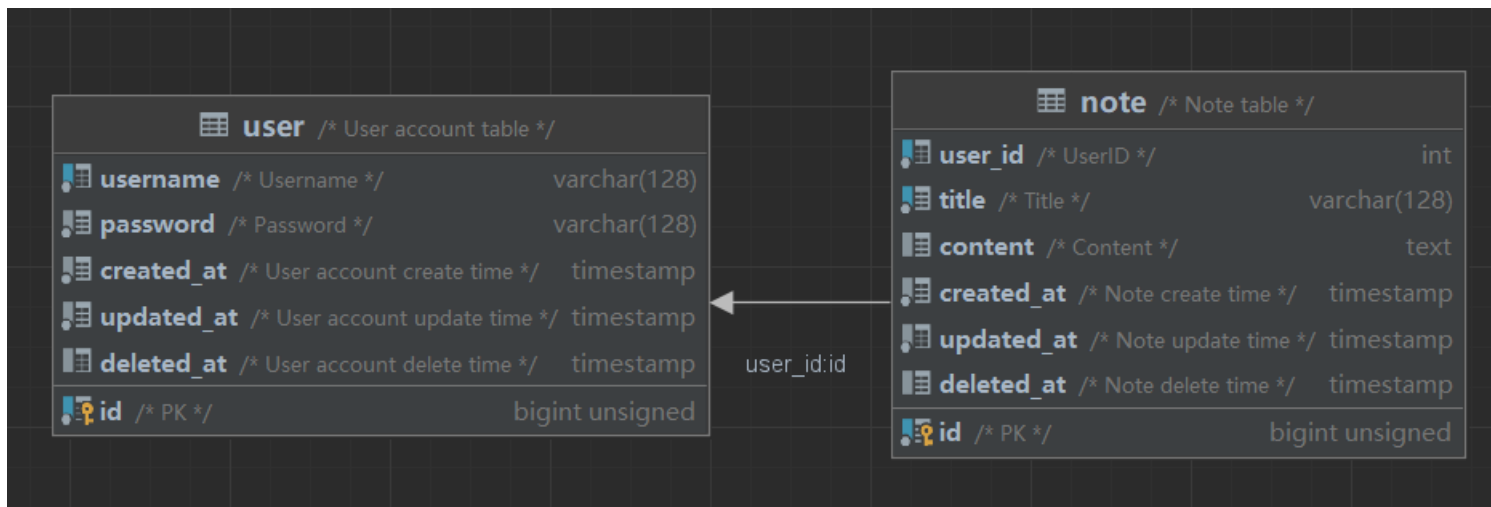
```
struct QueryNoteRequest {  
    1: i64 user_id  
    2: optional string search_key (api.query="search_key", api.vd="len($) > 0")  
    3: i64 offset (api.query="offset", api.vd="len($) >= 0")  
    4: i64 limit (api.query="limit", api.vd="len($) >= 0")  
}
```

```
service ApiService {  
    CreateUserResponse CreateUser(1: CreateUserRequest req) (api.post="/v2/user/register")  
    CheckUserResponse CheckUser(1: CheckUserRequest req) (api.post="/v2/user/login")  
    CreateNoteResponse CreateNote(1: CreateNoteRequest req) (api.post="/v2/note")  
    QueryNoteResponse QueryNote(1: QueryNoteRequest req) (api.get="/v2/note/query")  
    UpdateNoteResponse UpdateNote(1: UpdateNoteRequest req) (api.put="/v2/note/:note_id")  
    DeleteNoteResponse DeleteNote(1: DeleteNoteRequest req) (api.delete="/v2/note/:note_id")  
}
```

```
struct CreateNoteRequest {  
    1: string title (vt.min_size = "1")  
    2: string content (vt.min_size = "1")  
    3: i64 user_id (vt.gt = "0")  
}
```

- 定义注解，使用 Hertz 参数绑定与验证功能
- 定义请求方式与路由，使用 hz 生成脚手架代码
- 定义注解使用 thrift-gen-validator 插件进行结构体校验

开发流程 – 建立数据库表



```
// gorm.Model 的定义
type Model struct {
    ID          uint           `gorm:"primaryKey"`
    CreatedAt   time.Time
    UpdatedAt   time.Time
    DeletedAt   gorm.DeletedAt `gorm:"index"`
}
```

- 根据 gorm.Model 设计数据库表，编写 SQL
- 使用 docker-compose volumes 属性将 SQL 文件挂载到 MySQL 容器中，
- MySQL 官方镜像会在容器启动时自动执行 /docker-entrypoint-initdb.d 文件夹下的 SQL 脚本从而完成数据库初始化

开发流程 – 代码生成

<https://github.com/cloudwego/cwgo>

cwgo

cwgo 是 CloudWeGo All in one 代码生成工具，整合了各个组件的优势，提高开发者提体验。cwgo 工具可以方便生成工程化模版，其主要功能特点如下：

工具特点

- 用户友好生成方式

cwgo 工具同时提供了交互式命令行和静态命令行两种方式。交互式命令行可以低成本生成代码，不用再去关心传递什么参数，也不用执行多次命令，适合大部分用户；而对高级功能有需求的用户，仍可使用常规的静态命令行构造生成命令。

- 支持生成工程化模板

cwgo 工具支持生成 MVC 项目 Layout，用户只需要根据不同目录的功能，在相应的位置完成自己的业务代码即可，聚焦业务逻辑。

- 支持生成 Server、Client 代码

cwgo 工具支持生成 Kitex、Hertz 的 Server 和 Client 代码，提供了对 Client 的封装。用户可以开箱即用的调用下游，免去封装 Client 的繁琐步骤

- 支持生成数据库代码

cwgo 工具支持生成数据库 CURD 代码。用户无需再自行封装繁琐的 CURD 代码，提高用户的工作效率。

- 使用 hz 和 kitex 工具生成脚手架代码，简化开发流程；
- Tips: 最新的 cwgo 工具集成了 hz，kitex，gorm-gen 工具并提供了更加丰富的模板和代码生成功能。

开发流程 – demouser, demonote

demouser

- 利用 GORM 完成对用户的创建与查询
- 完成用户创建，检查，查询的具体业务逻辑
- 将服务注册进 etcd 以供其他服务调用

demonote

- 利用 GORM 完成对笔记的增删改查
- 通过 RPC 调用 demouser 服务获取用户信息
- 完成笔记增删改查具体的业务逻辑
- 将服务注册进 etcd 以供其他服务调用

开发流程 – demoapi

- 使用 jwt, requestid, gzip, pprof 中间件
- 完成用户注册登录功能
- 通过 jwt 认证授权后的用户才能对笔记进行一系列操作
- 通过 RPC 调用 demouser 和 demonote 服务完成业务逻辑
- 返回响应数据给前端

代码细节 – 链路追踪

- Hertz, Kitex 和 GROM 的 OpenTelemetry 扩展提供了 tracing、metrics、logging 的支持；
- 通过 Jaeger 进行链路追踪，通过 Grafana 面板进行观测；
- 利用 traceid 替换 requestid 的默认生成，方便根据 requestid 直接查询链路；

```
// use requestid mw
requestid.New(
    requestid.WithGenerator(func(ctx context.Context, c *app.RequestContext) string {
        traceID := trace.SpanFromContext(ctx).SpanContext().TraceID().String()
        return traceID
    }),
),
```

代码细节 – Kitex 中间件

- 定义 server, client 以及 common 中间件并以选项的模式嵌入 Kitex 对 RPC 调用信息进行日志输出

```
var _ endpoint.Middleware = CommonMiddleware

// CommonMiddleware common mw print some rpc info, real request and real response
func CommonMiddleware(next endpoint.Endpoint) endpoint.Endpoint {
    return func(ctx context.Context, req, resp interface{}) (err error) {
        ri := rpcinfo.GetRPCInfo(ctx)
        // get real request
        klog.Infof(format: "real request: %+v\n", req)
        // get remote service information
        klog.Infof(format: "remote service name: %s, remote method: %s\n", ri.To().ServiceName(), ri.To().Method())
        if err = next(ctx, req, resp); err != nil { err }
        // get real response
        klog.Infof(format: "real response: %+v\n", resp)
        return err: nil
    }
}
```


代码细节 – 错误码封装

- 预先在 IDL 中定义错误码
- 自定义错误类型与函数对业务异常进行转换，一致化错误处理

```
enum ErrCode {  
    SuccessCode           = 0  
    ServiceErrCode        = 10001  
    ParamErrCode          = 10002  
    UserAlreadyExistErrCode = 10003  
    AuthorizationFailedErrCode = 10004  
}
```

```
// ConvertErr convert error to Errno  
func ConvertErr(err error) ErrNo {  
    Err := ErrNo{}  
    if errors.As(err, &Err) {  
        return Err  
    }  
    s := ServiceErr  
    s.ErrMsg = err.Error()  
    return s  
}
```

```
var (  
    Success          = NewErrNo(int64(demouser.ErrCode_SuccessCode), msg: "Success")  
    ServiceErr       = NewErrNo(int64(demouser.ErrCode_ServiceErrCode), msg: "Service is unable to start successfully")  
    ParamErr         = NewErrNo(int64(demouser.ErrCode_ParamErrCode), msg: "Wrong Parameter has been given")  
    UserAlreadyExistErr = NewErrNo(int64(demouser.ErrCode_UserAlreadyExistErrCode), msg: "User already exists")  
    AuthorizationFailedErr = NewErrNo(int64(demouser.ErrCode_AuthorizationFailedErrCode), msg: "Authorization failed")  
)
```

总结回顾

- CloudWeGo 生态入门（文档，扩展，如何提问）
- Hertz, Kitex 和 GORM
- Easy Note（架构，代码生成，开发流程，代码细节）

欢迎大家为 Hertz, Kitex 和 GORM 贡献代码！

- <https://github.com/cloudwego/hertz>
- <https://github.com/cloudwego/kitex>
- <https://github.com/go-gorm/gorm>



THANKS