看下面的代码调用了server.Init()之后就支持命令行读取配置了

package main

import (

"github.com/gin-gonic/gin"

"github.com/micro/go-micro/registry"

"github.com/micro/go-micro/web"

"github.com/micro/go-plugins/registry/consul"

)

func main() {

//新建一个consul注册的地址，也就是我们consul服务启动的机器ip+端口

consulReg := consul.NewRegistry(

registry.Addrs("localhost:8500"),

)

ginRouter := gin.Default()

ginRouter.Handle("GET", "/user", func(context \*gin.Context) {

context.String(200, "user api")

})

ginRouter.Handle("GET", "/news", func(context \*gin.Context) {

context.String(200, "news api")

})

// go-micro很灵性的实现了注册和反注册，我们启动后直接ctrl+c退出这个server，它会自动帮我们实现反注册

server := web.NewService(

web.Name("prodservice"), //注册进consul服务中的service名字

//web.Address(":8001"), //注册进consul服务中的端口

web.Handler(ginRouter), //web.Handler()返回一个Option，我们直接把ginRouter穿进去，就可以和gin完美的结合

web.Registry(consulReg), //注册到哪个服务器伤的consul中

)

server.Init() //加了这句就可以使用命令行的形式去设置我们一些启动的配置

server.Run()

}

**Init支持以下参数 这是源代码**

func (s \*service) Init(opts ...Option) error {

for \_, o := range opts {

o(&s.opts)

}

serviceOpts := []micro.Option{}

if len(s.opts.Flags) > 0 {

serviceOpts = append(serviceOpts, micro.Flags(s.opts.Flags...))

}

if s.opts.Registry != nil {

serviceOpts = append(serviceOpts, micro.Registry(s.opts.Registry))

}

serviceOpts = append(serviceOpts, micro.Action(func(ctx \*cli.Context) {

if ttl := ctx.Int("register\_ttl"); ttl > 0 {

s.opts.RegisterTTL = time.Duration(ttl) \* time.Second

}

if interval := ctx.Int("register\_interval"); interval > 0 {

s.opts.RegisterInterval = time.Duration(interval) \* time.Second

}

if name := ctx.String("server\_name"); len(name) > 0 {

s.opts.Name = name

}

if ver := ctx.String("server\_version"); len(ver) > 0 {

s.opts.Version = ver

}

if id := ctx.String("server\_id"); len(id) > 0 {

s.opts.Id = id

}

if addr := ctx.String("server\_address"); len(addr) > 0 {

s.opts.Address = addr

}

if adv := ctx.String("server\_advertise"); len(adv) > 0 {

s.opts.Advertise = adv

}

if s.opts.Action != nil {

s.opts.Action(ctx)

}

}))

s.opts.Service.Init(serviceOpts...)

srv := s.genSrv()

srv.Endpoints = s.srv.Endpoints

s.srv = srv

return nil

}

一般我们不推荐用这种方式去做服务注册，因为以后维护的人不知道你传的什么参数去注册的服务，导致维护困难，一般还是用读取配置文件的方式去注册服务。