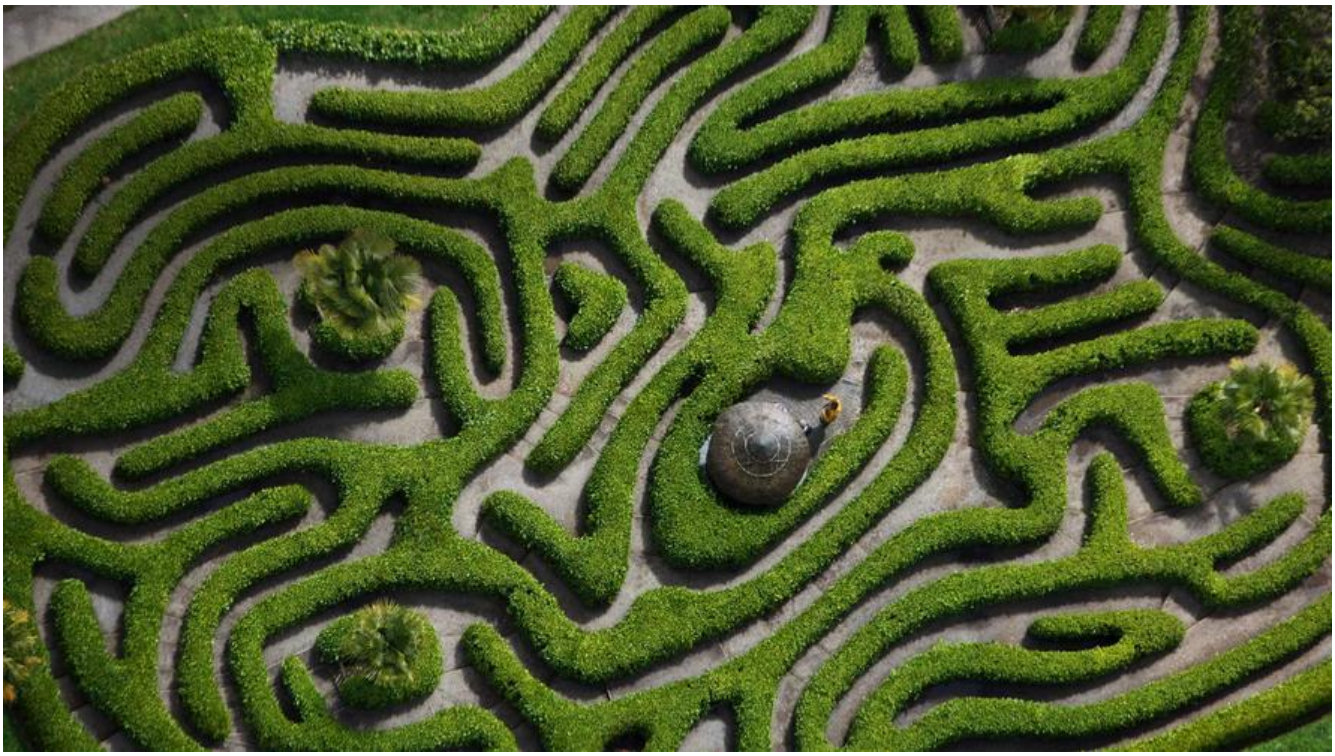
链滴

【GO-Micro】hystrix 熔断及 dashboard 展示

作者：Allenxuxu

原文链接：https://ld246.com/article/1561168555917

来源网站：链滴



github完整代码地址

hystrix-go

hystrix是Netflix开源的一个JAVA项目，不过GitHub也有golang的实现版本hystrix-go

hystrix-dashboard

hystrix并没有自带一个仪表盘，无法直观的查看接口的健康状况。所以，我们采用GitHub的一个开实现hystrix-dashboard。

```
docker run --name hystrix-dashboard -d -p 8081:9002 mlabouardy/hystrix-dashboard:latest
```

micro API网关插件

关于hystrix的工作原理，可以查阅相关资料，这里只讲解如何封装插件在micro API网关中使用。

```
err := hystrix.Do("my_command", func() error {
    // talk to other services
    return nil
}, nil)
```

使用hystrix.Do() 同步API，第一个参数是command, 应该是与当前请求一一对应的一个名称，如入GET-/test”。第二个参数传入一个函数，函数包含我我们自己的错误逻辑，当请求失败时应该返回error。hystrix会根据我们的失败率执行熔断策略。

封装Handler

```
// BreakerWrapper hystrix breaker
```

```
func BreakerWrapper(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        name := r.Method + "-" + r.RequestURI
        log.Println(name)
        err := hystrix.Do(name, func() error {
            sct := &status_code.StatusCodeTracker{ResponseWriter: w, Status: http.StatusOK}
            h.ServeHTTP(sct.WrappedResponseWriter(), r)

            if sct.Status >= http.StatusBadRequest {
                str := fmt.Sprintf("status code %d", sct.Status)
                log.Println(str)
                return errors.New(str)
            }
            return nil
        }, nil)
        if err != nil {
            log.Println("hystrix breaker err: ", err)
            return
        }
    })
}
...
// 注册插件
plugin.Register(plugin.NewPlugin(
        plugin.WithName("breaker"),
        plugin.WithHandler(
            hystrix.BreakerWrapper,
        ),
    ))
...
```

在 hystrix.Do 中，首先执行 h.ServeHTTP，该函数返回后，即请求执行完成。我们判断HTTP状态码如果大于StatusBadRequest，则认为这次请求失败，返回一个错误，hystrix会收集错误，如果错误达到某个阀值，就会触发断路器。

在做实验时，可以直接在main函数里设置hystrix的几个默认配置参数，方便看效果

```
// hystrix-go/hystrix/settings.go

    // DefaultTimeout is how long to wait for command to complete, in milliseconds
    DefaultTimeout = 1000
    // DefaultMaxConcurrent is how many commands of the same type can run at the same ti
e
    DefaultMaxConcurrent = 10
    // DefaultVolumeThreshold is the minimum number of requests needed before a circuit can
be tripped due to health
    DefaultVolumeThreshold = 20
    // DefaultSleepWindow is how long, in milliseconds, to wait after a circuit opens before test
ng for recovery
    DefaultSleepWindow = 5000
    // DefaultErrorPercentThreshold causes circuits to open once the rolling measure of errors
xceeds this percent of requests
    DefaultErrorPercentThreshold = 50
```

hystrix-go库还提供为每个commond动态设置配置的接口，我们可以通过这个接口结合配置中心，态调节服务。

```
hystrix.ConfigureCommand("my_command", hystrix.CommandConfig{
    Timeout:                1000,
    MaxConcurrentRequests: 100,
    ErrorPercentThreshold: 25,
})
```

接入hystrix-dashboard

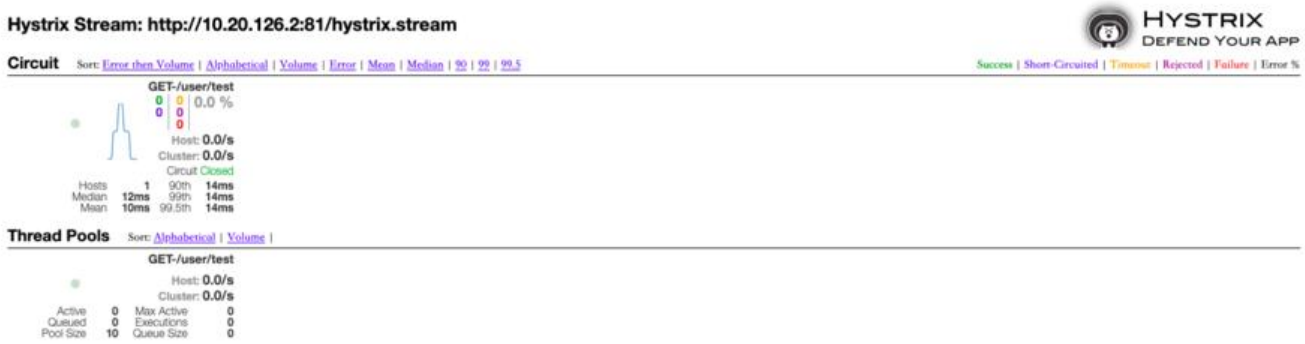
```
docker run --name hystrix-dashboard -d -p 8081:9002 mlabouardy/hystrix-dashboard:latest
```

打开 http://localhost:8081/hystrix , 输入 http://{ip}:81/hystrix.stream , 此处ip为本机ip，因为hystrix-dashboard是容器启动的，无法直接访问本机127.0.0.1。

Enable dashboard metrics

In your main.go, register the event stream HTTP handler on a port and launch it in a goroutine Once you configure turbine for your Hystrix Dashboard to start streaming events, your commands will automatically begin appearing.

```
hystrixStreamHandler := hystrix.NewStreamHandler()
hystrixStreamHandler.Start()
go http.ListenAndServe(net.JoinHostPort("", "81"), hystrixStreamHandler)
```



原文链接：【GO-Micro】hystrix 熔断及 dashboard 展示