

# 构建消息消费服务（MQC）

MQC：订阅消息队列的topic，收到消息后执行本地服务，支持的类型有：

名称	说明
redis	推荐，基于list实现
mqtt	推荐，物联网消息队列
stomp	支持activeMQ，rabbitMQ

特点：

- 支持主备，对等，分片
- 消息执行线程可配置，默认为10个协程
- 可动态订阅，取消订阅消息

## 1. 创建服务器

main.go

```
package main

import "github.com/micro-plat/hydra/hydra"

type flowserver struct {
    *hydra.MicroApp
}

func main() {
    app := &flowserver{
        hydra.NewApp(
            hydra.WithPlatName("mall"),
            hydra.WithSystemName("flowserver"),
            hydra.WithServerTypes("mqc")),
    }
    app.init()
    app.Start()
}
```

## 2. 设置集群模式

主配置中设置变量 sharding 的值， =1 为主备模式， >1 为分片模式， =0 或不设置为对等模式

```
flow.Conf.API.SetMainConf(`{"sharding":1}
```

### 3. 服务器配置

conf.dev.go server 配置消息队列服务器，必须配置

```
// +build !prod

package main

func (flow *flowserver) config() {
    flow.IsDebug = true

    flow.Conf.MQC.SetSubConf("server", `
        {
            "proto":"redis",
            "addrs":[
                "192.168.0.111:6379",
                "192.168.0.112:6379",
                "192.168.0.113:6379",
                "192.168.0.114:6379"
            ],
            "db":1,
            "dial_timeout":10,
            "read_timeout":10,
            "write_timeout":10,
            "pool_size":10
        }
    `)
}
```

将redis作为消息队列服务器，订阅消息

### 4. 静态订阅

订阅信息保存到注册中心

```

flow.Conf.MQC.SetSubConf("queue", `{
    "queues":[
        {
            "queue":"coupon:base:coupon_produce",
            "service":"/coupon/produce"
        },
        {
            "queue":"coupon:base:down_payment",
            "service":"/order/pay",
            "concurrency":1000
        }
    ]
}`)

```

使用动态队列则无需设置

通过 concurrency 设置并行处理协程数，未设置为 10

## 5. 动态订阅

订阅信息保存到本机内存

```

package main

import "github.com/micro-plat/hydra/conf"

func (flow *flowserver) init() {

    ch := flow.GetDynamicQueue()

    //订阅消息
    ch <- &conf.Queue{Queue: "mall:flow:order_pay", Service:
        "/order/pay",Concurrency:100}

    //取消订阅
    // ch<- &cf.Queue{Queue: "mall:flow:order_pay", Disable:true}

    flow.MQC("/order", order.NewOrderHandler)
}

```

订阅消息 Queue ， Service 为必须参数。取消订阅 Queue ， Disable 为必须参数

服务注册使用 flow.MQC （仅注册到MQC服务）或 flow.Flow （注册到MQC服务和CRON服务）。

通过 concurrency 设置并行处理协程数，未设置为 10

订阅成功后日志会显示: 订阅(mall:flow:order\_pay)消息：

```
~/work/bin$ flowserver01 run -r zk://192.168.0.109 -c yl
[2019/07/01 16:59:17.485857][i][c79ab8206]Connected to 192.168.0.109:2181
[2019/07/01 16:59:17.489212][i][c79ab8206]Authenticated: id=246395503264334087, timeout
[2019/07/01 16:59:17.489216][i][c79ab8206]Re-submitting `0` credentials after reconnect
[2019/07/01 16:59:17.534987][i][c79ab8206]初始化 /mall_debug/flowserver/mqc/yl
[2019/07/01 16:59:17.544972][i][c4317fd95]开始启动[MQC]服务...
[2019/07/01 16:59:17.545481][d][c4317fd95][未启用 metric设置]
[2019/07/01 16:59:17.545881][d][c4317fd95][订阅(mall:flow:order_pay)消息]
[2019/07/01 16:59:18.61300][i][c4317fd95]master mqc server
[2019/07/01 16:59:18.66093][i][c4317fd95]服务启动成功(MQC, mqc://192.168.4.121, 0)
```

## 6. 服务编写

与其它服务器的服务层代码相同

```
package order

import (
    "github.com/micro-plat/hydra/component"
    "github.com/micro-plat/hydra/context"
)

type OrderHandler struct {
    container component.IContainer
}

func NewOrderHandler(container component.IContainer) (u *OrderHandler, err error) {
    return &OrderHandler{container: container}, nil
}

//Handle 处理订单支付
func (u *OrderHandler) PayHandle(ctx *context.Context) (r interface{}) {
    //业务逻辑
    return "success"
}
```