链滴

高性能消息队列 NSQ---GO--demo

作者：InkDP

原文链接：https://ld246.com/article/1590046690321

来源网站：链滴

前言

  关于NSQ是什么，NSQ是做什么的，怎么启动，网上的资料太多太多，详请移步分布式实时消息平台NSQ，里面讲的很详细，还附带demo。

客户端/生产者(producer)

  NSQ发送消息非常简单，分两步完成：

- 创建Producer实例
- 调用 Publish发送一个新的消息到指定的topic中

具体实现如下所示

```
func main(){
    cfg := nsq.NewConfig()
    nsqd := "127.0.0.1:4150"
    producer, err := nsq.NewProducer(nsqd, cfg)
    if err != nil {
        log.Fatal(err)
    }
    if err := producer.Publish("test", []byte("Hello NSQ")); err != nil {
        log.Fatal("publish error:" + err.Error())
    }
}
```

服务端/消费者(consumer)

  消费者用于接收指定 topic中的消息，实现需分为3步：

- 调用 NewConsumer为指定的主题/渠道创建消费者的新实例
- 调用 AddHandler为此使用者接收的消息设置处理程序
- 调用 ConnectToNSQD使用nsqd地址直接连接，有多个地址时使用ConnectToNSQDs，这里官文档推荐使用ConnectToNSQLookupd

具体实现如下：

```
func main(){
    cfg := nsq.NewConfig()
    c, err := nsq.NewConsumer(topic, channel, cfg)
    if err != nil {
        panic(err)
    }
    c.AddHandler(&ConsumerT{})

    if err := c.ConnectToNSQD(address); err != nil {
        panic(err)
    }
}
func (*ConsumerT) HandleMessage(msg *nsq.Message) error {
    fmt.Println("receive", msg.NSQDAddress, "message:", string(msg.Body))
```

```
    return nil
}
```

消费者生产者搭配使用

  我上面所写的demo虽实现了最基本的NSQ的功能，但是对于一个demo来，整体不够直观。理想中的状态应该是：服务端一直处于执行状态，客户端发送消息时，服务端接受处理。

  改造后服务端：

```
package main

import (
    "fmt"

    "github.com/nsqio/go-nsq"
)

type ConsumerT struct{}

func main() {
    InitConsumer("test", "ch1", "127.0.0.1:4150")
    select {}
}

func (*ConsumerT) HandleMessage(msg *nsq.Message) error {
    fmt.Println("receive", msg.NSQDAddress, "message:", string(msg.Body))
    return nil
}

func InitConsumer(topic string, channel string, address string) {
    cfg := nsq.NewConfig()
    c, err := nsq.NewConsumer(topic, channel, cfg)
    if err != nil {
        panic(err)
    }
    c.AddHandler(&ConsumerT{})

    if err := c.ConnectToNSQD(address); err != nil {
        panic(err)
    }
}
```

  客户端这边呢，倒是没什么具体要更改的，可是每次都需要重复去运行才能送消息也确实麻烦，所以做了点小更改，让客户端也一直处于运行状态，通过命令行的输入来发送消，具体如下：

```
package main

import (
    "bufio"
    "fmt"
```

```
    "github.com/nsqio/go-nsq"
    "log"
    "os"
    "strings"
)

func main() {
    cfg := nsq.NewConfig()
    nsqd := "127.0.0.1:4150"
    producer, err := nsq.NewProducer(nsqd, cfg)
    if err != nil {
        log.Fatal(err)
    }

    reader := bufio.NewReader(os.Stdin)
    fmt.Println("Simple Shell")
    fmt.Println("---------------------")

    for {
        fmt.Print("-> topic: ")
        topic, _ := reader.ReadString('\n')
        topic = strings.Replace(topic, "\n", "", -1)
        fmt.Print("-> message: ")
        message, _ := reader.ReadString('\n')
        message = strings.Replace(message, "\n", "", -1)
        fmt.Println("消息发送中\n")

        if err := producer.Publish(topic, []byte(message)); err != nil {
            log.Fatal("publish error:" + err.Error())
        }

    }
}
```

源码请访问本人GitHub下载：https://github.com/InkDP/nsq-demo