链滴

golang 开源的消息队列 nsq 安装使用介绍

作者：450370050

原文链接：https://ld246.com/article/1524558577932

来源网站：链滴

NSQ 是实时的分布式消息处理平台，其设计的目的是用来大规模地处理每天数以十亿计级别的消息
它具有分布式和去中心化拓扑结构，该结构具有无单点故障、故障容错、高可用性以及能够保证消息
可靠传递的特征。

安装

本机测试时使用的是windows环境就独自编译了nsq的各模块

```
go get github.com/nsqio/nsq
cd apps nsqd
go build nsqd.go
```

nsq可以搭建mq集群，通过nsqlookupd发现管理nsqd实例，nsqadmin以web的方式管理nsqd

1.运行nsqlookupd

```
D:\go\gopath\src\github.com\nsqio\nsq\apps\nsqlookupd>nsqlookupd.exe
[nsqlookupd] 2017/11/07 17:52:46.063484 INFO: nsqlookupd v1.0.0-compat (built w/
go1.8rc2)
[nsqlookupd] 2017/11/07 17:52:46.084485 INFO: TCP: listening on [::]:4160
[nsqlookupd] 2017/11/07 17:52:46.088485 INFO: HTTP: listening on [::]:4161
```

2.运行nsqld

```
D:\go\gopath\src\github.com\nsqio\nsq\apps\nsqd>nsqd --lookupd-tcp-address=127.0.0.1:
160                                                 [nsqd] 2017/11/07 17:55:17.983173 INFO: nsq
 v1.0.0-compat (built w/go1.8rc2)   [nsqd] 2017/11/07 17:55:18.010175 INFO: ID: 710
            [nsqd] 2017/11/07 17:55:18.011175 INFO: TOPIC(test): created                [nsqd] 2
17/11/07 17:55:18.012175
```

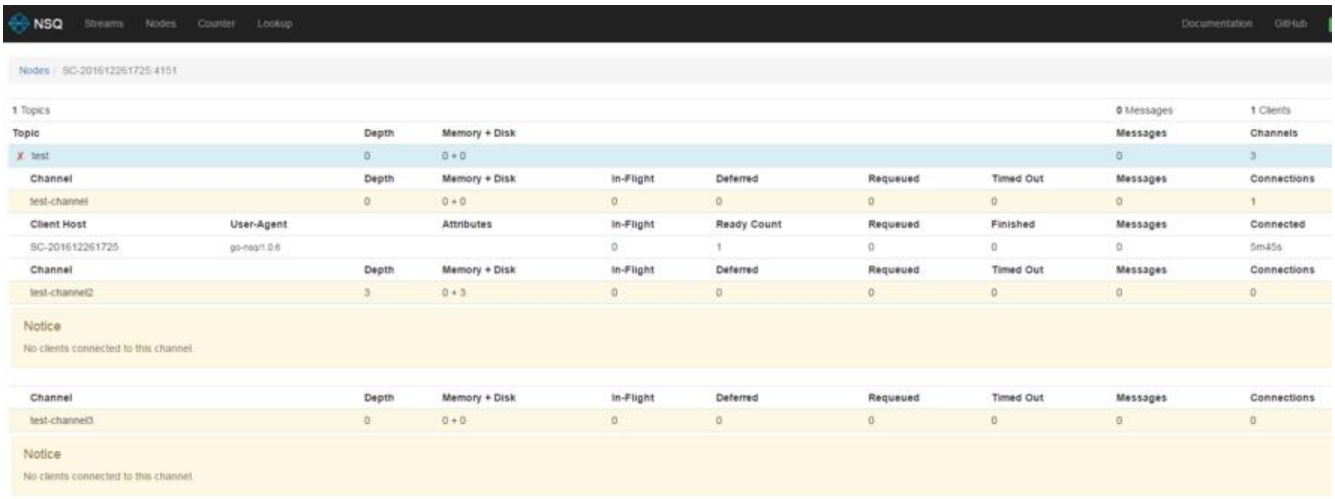
3.运行nsqadmin

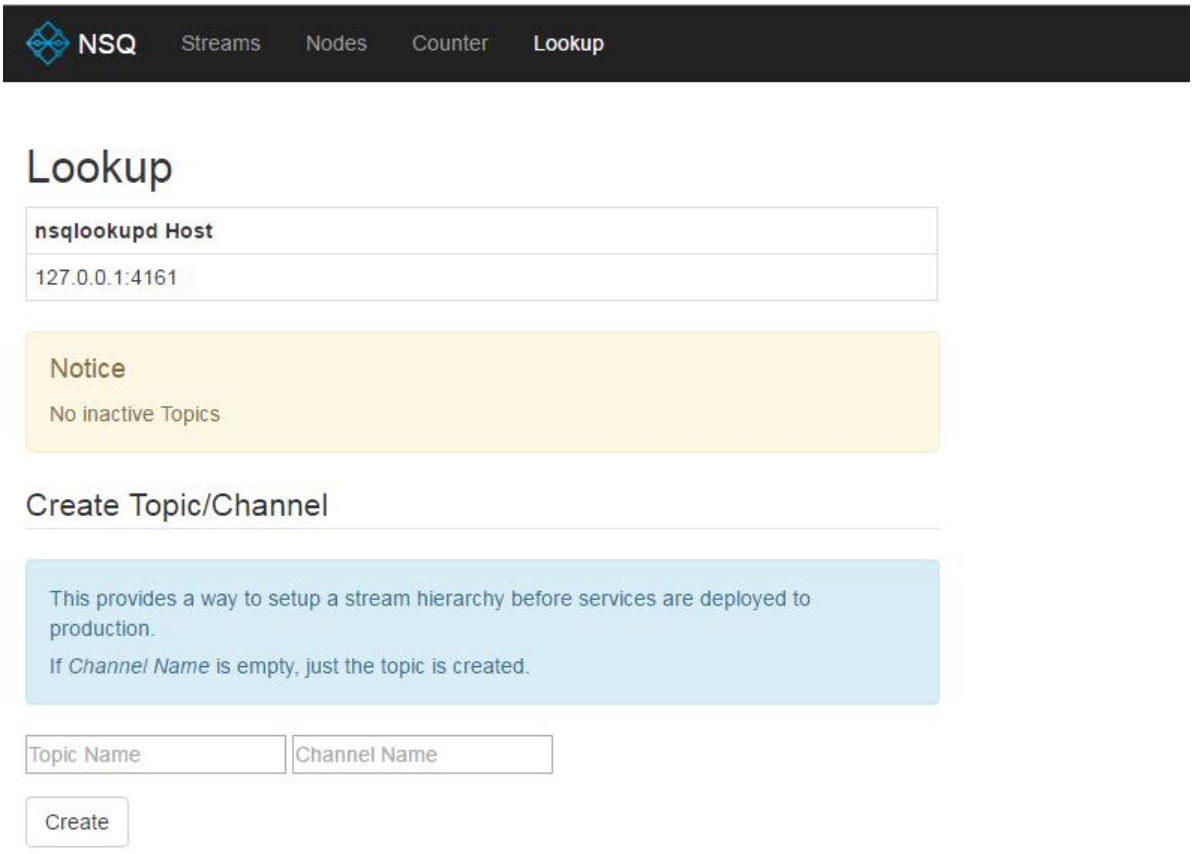
```
D:\go\gopath\src\github.com\nsqio\nsq\apps\nsqadmin>nsqadmin --lookupd-http-address
127.0.0.1:4161                                      [nsqadmin] 2017/11/07 17:58:30.405179
NFO: nsqadmin v1.0.0-compat (built w/go1.8rc2)
 [nsqadmin] 2017/11/07 17:58:30.426180 INFO: HTTP: listening on [::]:4171
```

使用

1.管理

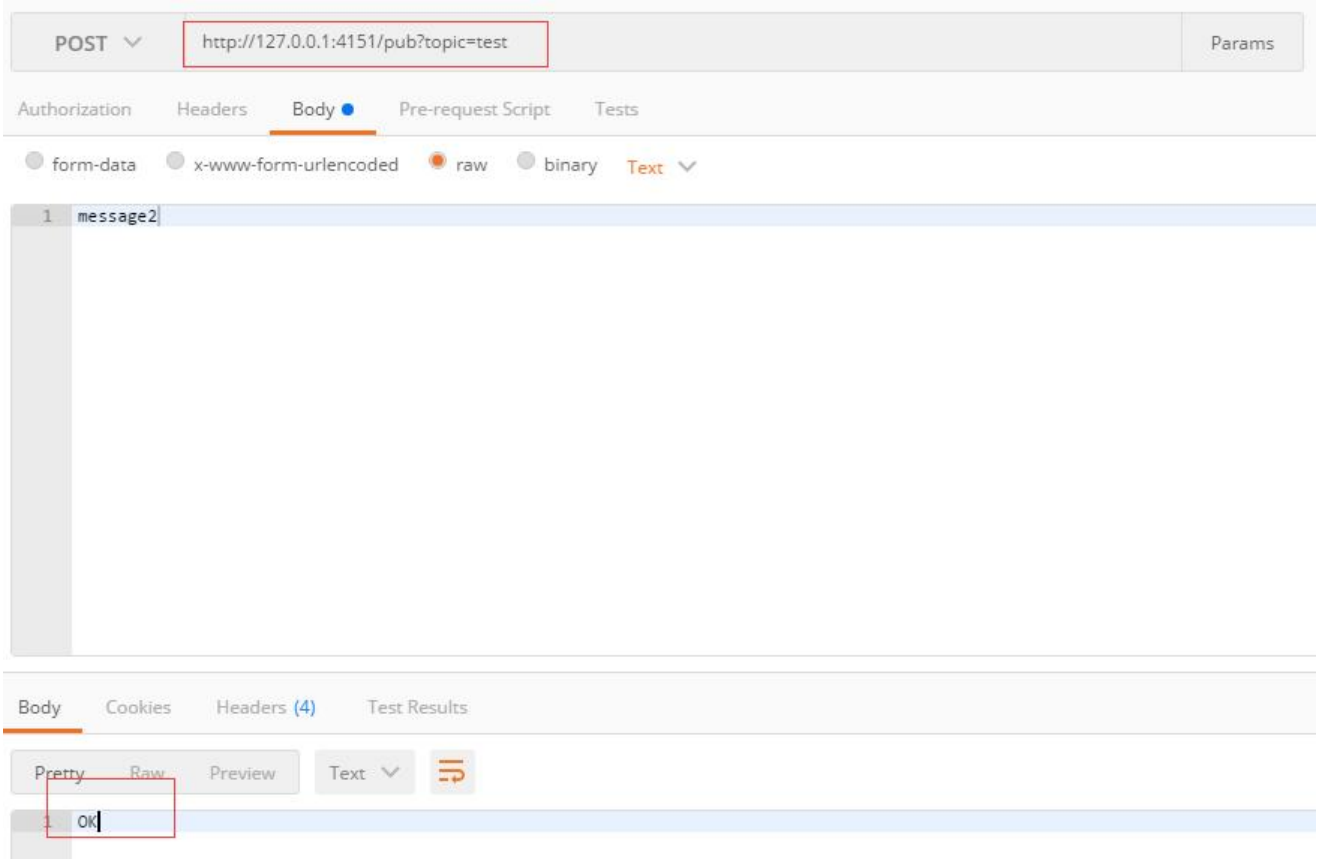
我们可以访问http://127.0.0.1/ 来管理我们的nsq





2.创建消息

除了客户端连接创建消息之外我们还可以通过http 提交消息



3.消费消息

nsq的topic可以创建多个消费channel，一条消息可以多个通道消费使用：

```
package main

import (
    "fmt"
    "time"
    "github.com/nsqio/go-nsq"
)

// 消费者
type ConsumerT struct{}

// 主函数
func main() {
    InitConsumer("test", "test-channel", "127.0.0.1:4161")
    for {
        time.Sleep(time.Second * 10)
    }
}

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

```
//初始化消费者
func InitConsumer(topic string, channel string, address string) {
    cfg := nsq.NewConfig()
    cfg.LookupdPollInterval = time.Second          //设置重连时间
    c, err := nsq.NewConsumer(topic, channel, cfg) // 新建一个消费者
    if err != nil {
        panic(err)
    }
    c.SetLogger(nil, 0)        //屏蔽系统日志
    c.AddHandler(&ConsumerT{}) // 添加消费者接口

    //建立NSQLookupd连接
    if err := c.ConnectToNSQLookupd(address); err != nil {
        panic(err)
    }
}
```

运行返回：

```
receive SC-201612261725:4150 message: test
receive SC-201612261725:4150 message: test
receive SC-201612261725:4150 message: test
receive SC-201612261725:4150 message: message1
receive SC-201612261725:4150 message: message2
receive SC-201612261725:4150 message: message2
```