

## 前言

之前有文章 [《从0到1学习Flink》](#) —— Flink 写入数据到 Kafka 写过 Flink 将处理后的数据后发到 Kafka 消息队列中去，当然我们常用的消息队列可不止这一种，还有 RocketMQ、RabbitMQ 等，刚好 Flink 也支持将数据写入到 RabbitMQ，所以今天我们就来写篇文章讲讲如何将 Flink 处理后的数据写入到 RabbitMQ。

## 前提准备

### 安装 RabbitMQ

这里我直接用 docker 命令安装吧，先把 docker 在 mac 上启动起来。

在命令行中执行下面的命令：

```
docker run -d -p 15672:15672 -p 5672:5672 -e
RABBITMQ_DEFAULT_USER=admin -e RABBITMQ_DEFAULT_PASS=admin --name
rabbitmq rabbitmq:3-management
```

```
[2017-08-08 21:30:09.220] INFO [ExpirationController @ RabbitMQ]: Shutdown completed (KafkaServ
x zhisheng@zhisheng ➤ docker run -d -p 15672:15672 -p 5672:5672 -e RABBITMQ_DEFAULT_USER=admin -e RABBITMQ_DEFAULT_PASS=admin --name r
abbitmq rabbitmq:3-management

Unable to find image 'rabbitmq:3-management' locally
3-management: Pulling from library/rabbitmq
6cf436f81810: Pull complete
987088a85b96: Pull complete
b4624b3efe06: Pull complete
d42beb8ded59: Pull complete
938f9893acce: Pull complete
244ab2317e05: Pull complete
4fe5c77cc496: Pull complete
f0de0d0d1acb: Pull complete
61a34ab1b592: Pull complete
31055da823d0: Pull complete
a803756475cf: Pull complete
707112081429: Pull complete
Digest: sha256:3b00e078b648b03d2ffa82d7e0ca040265291b1060280e718baffa662ea535cc
Status: Downloaded newer image for rabbitmq:3-management
cbf0d018082b45fac55a5c733f0dcae7a357f467e2d95c3ab278e76692351a40
zhisheng@zhisheng ➤ docker ps
CONTAINER ID        IMAGE               COMMAND             NAMES             CREATED             STATUS             PORTS
cbf0d018082b        rabbitmq:3-manage  "docker-entrypoint.s..."  rabbitmq          5 hours ago         Up 5 hours         4369/tcp, 5671/tcp, 0.0.0.0:5672->
5672/tcp, 15671/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp
zhisheng@zhisheng ➤
```

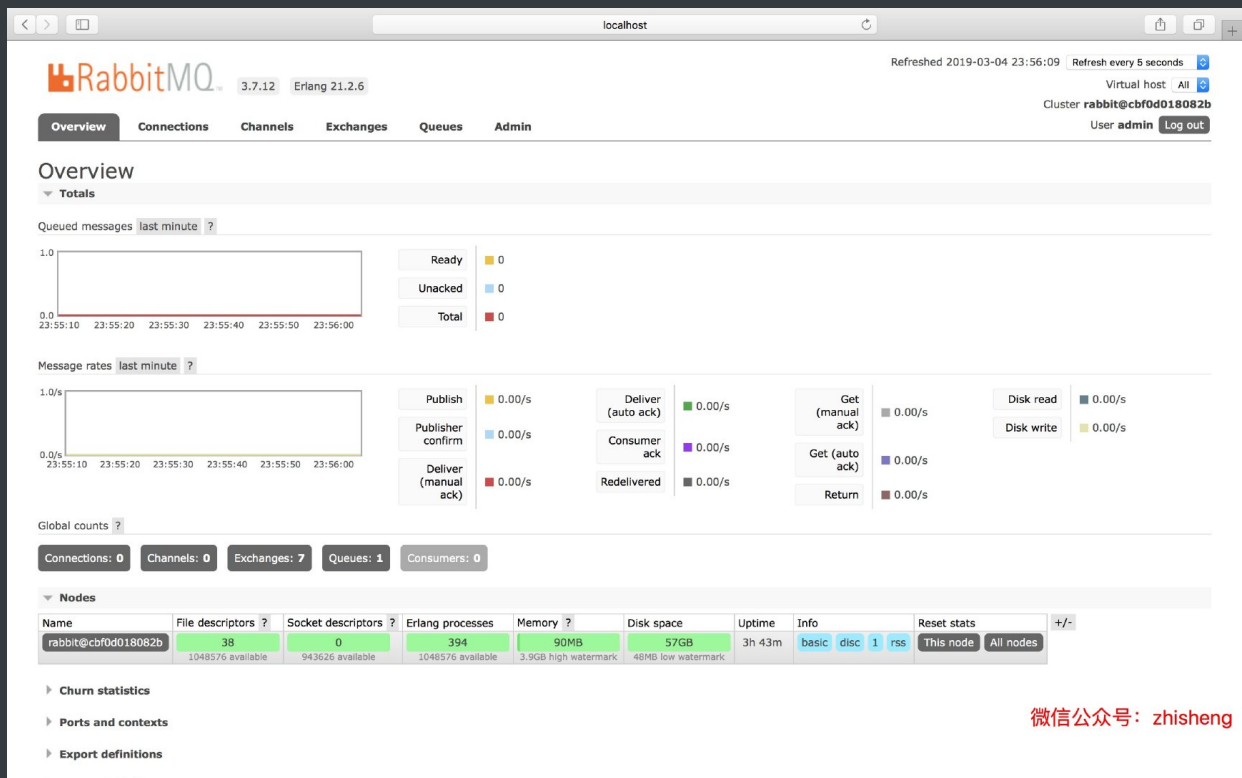
执行这个命令

微信公众号: zhisheng

对这个命令不懂的童鞋可以看看我以前的文

章: <http://www.54tianzhisheng.cn/2018/01/26/SpringBoot-RabbitMQ/>

登录用户名和密码分别是: admin / admin , 登录进去是这个样子就代表安装成功了:



## 依赖

pom.xml 中添加 Flink connector rabbitmq 的依赖如下:

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-rabbitmq_${scala.binary.version}</artifactId>
  <version>${flink.version}</version>
</dependency>
```

## 生产者

这里我们依旧自己写一个工具类一直的往 RabbitMQ 中的某个 queue 中发数据, 然后由 Flink 去消费这些数据。

注意按照我的步骤来一步步操作, 否则可能会出现一些错误!

RabbitMQProducerUtil.java

```
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;

public class RabbitMQProducerUtil {
    public final static String QUEUE_NAME = "zhisheng";
```

```

public static void main(String[] args) throws Exception {
    //创建连接工厂
    ConnectionFactory factory = new ConnectionFactory();

    //设置RabbitMQ相关信息
    factory.setHost("localhost");
    factory.setUsername("admin");
    factory.setPassword("admin");
    factory.setPort(5672);

    //创建一个新的连接
    Connection connection = factory.newConnection();

    //创建一个通道
    Channel channel = connection.createChannel();

    // 声明一个队列
    //      channel.queueDeclare(QUEUE_NAME, false, false, false, null);

    //发送消息到队列中
    String message = "Hello zhisheng";

    //我们这里演示发送一千条数据
    for (int i = 0; i < 1000; i++) {
        channel.basicPublish("", QUEUE_NAME, null, (message +
i).getBytes("UTF-8"));
        System.out.println("Producer Send +" + message + i);
    }

    //关闭通道和连接
    channel.close();
    connection.close();
}
}

```

## Flink 主程序

```

import com.zhisheng.common.utils.ExecutionEnvUtil;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.api.java.utils.ParameterTool;
import org.apache.flink.streaming.api.datastream.DataStreamSource;
import
org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;

```

```

import org.apache.flink.streaming.connectors.rabbitmq.RMQSource;
import
org.apache.flink.streaming.connectors.rabbitmq.common.RMQConnectionCon
fig;

/**
 * 从 rabbitmq 读取数据
 */
public class Main {
    public static void main(String[] args) throws Exception {
        final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        ParameterTool parameterTool = ExecutionEnvUtil.PARAMETER_TOOL;

        //这些配置建议可以放在配置文件中，然后通过 parameterTool 来获取对应的参
数值
        final RMQConnectionConfig connectionConfig = new
RMQConnectionConfig
            .Builder().setHost("localhost").setVirtualHost("/")

.setPort(5672).setUserName("admin").setPassword("admin")
            .build();

        DataStreamSource<String> zhisheng = env.addSource(new
RMQSource<>(connectionConfig,
            "zhisheng",
            true,
            new SimpleStringSchema()))
            .setParallelism(1);
        zhisheng.print();

        //如果想保证 exactly-once 或 at-least-once 需要把 checkpoint 开启
//
env.enableCheckpointing(10000);
        env.execute("flink learning connectors rabbitmq");
    }
}

```

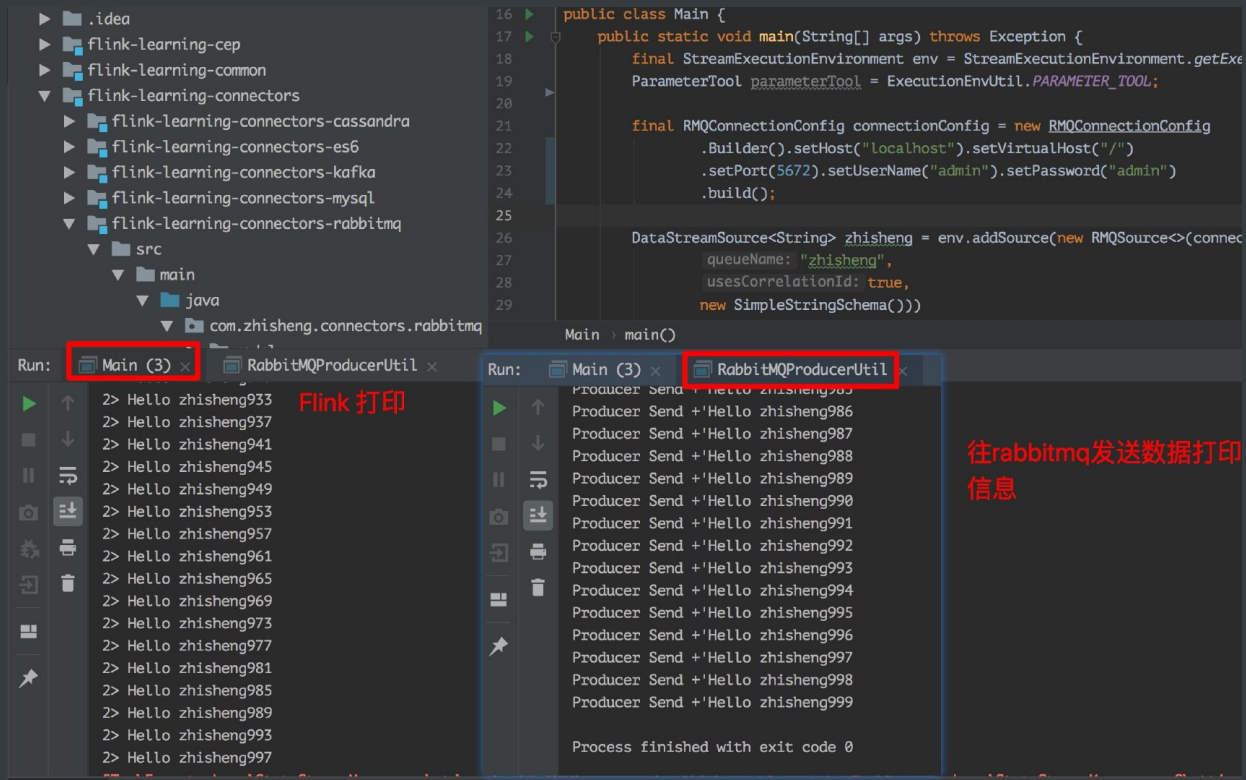
运行 RabbitMQProducerUtil 类，再运行 Main 类！

注意⚠️：

1、RMQConnectionConfig 中设置的用户名和密码要设置成 admin/admin，如果你换成是 guest/guest，其实是在 RabbitMQ 里面是没有这个用户名和密码的，所以就会报这个错误：

```
nested exception is
com.rabbitmq.client.AuthenticationFailureException: ACCESS_REFUSED -
Login was refused using authentication mechanism PLAIN. For details
see the broker logfile.
```

不出意外的话应该你运行 RabbitMQProducerUtil 类后，立马两个运行的结果都会出来，速度还是很快的。



2、如果你在 RabbitMQProducerUtil 工具类中把注释的那行代码打开的话：

```
// 声明一个队列
// channel.queueDeclare(QueueName, false, false, false, null);
```

就会出现这种错误：

```
Caused by: com.rabbitmq.client.ShutdownSignalException: channel error;
protocol method: #method<channel.close>(reply-code=406, reply-
text=PRECONDITION_FAILED - inequivalent arg 'durable' for queue
'zhisheng' in vhost '/': received 'true' but current is 'false',
class-id=50, method-id=10)
```

这是因为你打开那个注释的话，一旦你运行了该类就会创建一个叫做 zhisheng 的 Queue，当你再运行 Main 类中的时候，它又会创建这样一个叫 zhisheng 的 Queue，然后因为已经有同名的 Queue 了，所以就有了冲突，解决方法就是把那行代码注释就好了。

3、该 connector（连接器）中提供了 RMQSource 类去消费 RabbitMQ queue 中的消息和确认 checkpoints 上的消息，它提供了三种不一样的保证：

- Exactly-once(只消费一次): 前提条件有，1 是要开启 checkpoint，因为只有在 checkpoint 完成后，才会返回确认消息给 RabbitMQ（这时，消息才会在 RabbitMQ 队列中删除）；2 是要使用 Correlation ID，在将消息发往 RabbitMQ 时，必须在消息属性中设置 Correlation ID。数据源根据 Correlation ID 把从 checkpoint 恢复的数据进行去重；3 是数据源不能并行，这种限制主要是由于 RabbitMQ 将消息从单个队列分派给多个消费者。
- At-least-once(至少消费一次): 开启了 checkpoint，但未使用相 Correlation ID 或 数据源是并行的时候，那么就只能保证数据至少消费一次了
- No guarantees(无法保证): Flink 接收到数据就返回确认消息给 RabbitMQ

## Sink 数据到 RabbitMQ

RabbitMQ 除了可以作为数据源，也可以当作下游，Flink 消费数据做了一些处理之后也能把数据发往 RabbitMQ，下面演示下 Flink 消费 Kafka 数据后写入到 RabbitMQ。

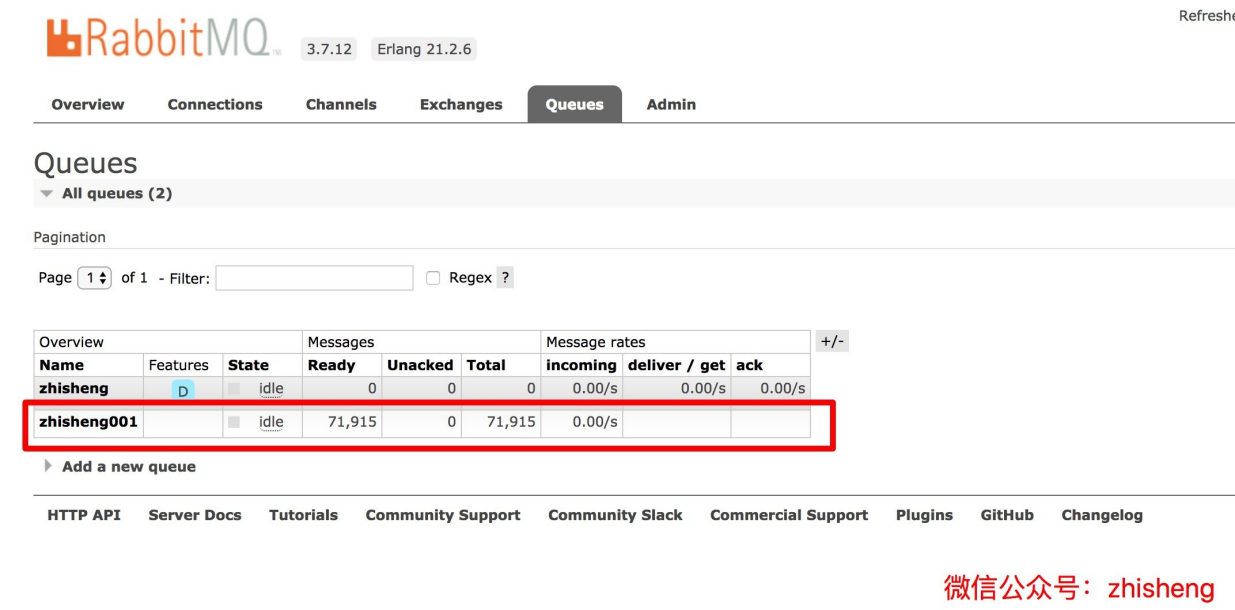
```
public class Main1 {
    public static void main(String[] args) throws Exception {
        final ParameterTool parameterTool =
            ExecutionEnvUtil.createParameterTool(args);
        StreamExecutionEnvironment env =
            ExecutionEnvUtil.prepare(parameterTool);
        DataStreamSource<Metrics> data =
            KafkaConfigUtil.buildSource(env);

        final RMQConnectionConfig connectionConfig = new
            RMQConnectionConfig
                .Builder().setHost("localhost").setVirtualHost("/")
                .setPort(5672).setUserName("admin").setPassword("admin")
                .build();

        //注意，换一个新的 queue，否则也会报错
        data.addSink(new RMQSink<>(connectionConfig, "zhisheng001",
            new MetricSchema()));
        env.execute("flink learning connectors rabbitmq");
    }
}
```

是不是很简单？但是需要注意的是，要换一个之前不存在的 queue，否则会报错的。

不出意外的话，你可以看到 RabbitMQ 的监控页面会出现新的一个 queue 出来，如下图：



The screenshot shows the RabbitMQ Management interface. At the top, there's a navigation bar with tabs: Overview, Connections, Channels, Exchanges, Queues (selected), and Admin. Below the tabs, the 'Queues' section is active, showing 'All queues (2)'. A pagination bar indicates 'Page 1 of 1' and a filter input. A table lists the queues. The first queue is 'zhisheng' with state 'idle' and 0 messages. The second queue, 'zhisheng001', is highlighted with a red box; it has state 'idle' and 71,915 messages. Below the table is a link to 'Add a new queue'. At the bottom, there's a footer with various links like HTTP API, Server Docs, Tutorials, etc. A red text overlay in the bottom right corner reads '微信公众号: zhisheng'.

Overview			Messages			Message rates			
Name	Features	State	Ready	Unacked	Total	Incoming	deliver / get	ack	+/-
zhisheng	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	
zhisheng001		idle	71,915	0	71,915	0.00/s			

## 总结

本文先把 RabbitMQ 作为数据源，写了个 Flink 消费 RabbitMQ 队列里面的数据进行打印出来，然后又写了个 Flink 消费 Kafka 数据后写入到 RabbitMQ 的例子！

本文原创地址是: <http://www.54tianzhisheng.cn/2019/01/20/Flink-RabbitMQ-sink/>，未经允许禁止转载。

## 关注我

微信公众号: zhisheng


另外我自己整理了些 Flink 的学习资料，目前已经全部放到微信公众号了。你可以加我的微信: zhisheng\_tian，然后回复关键字: **Flink** 即可无条件获取到。



更多私密资料请加入知识星球！



**Flink 精进学习**  
星主: zhisheng

 知识星球  
微信扫码预览星球详情



## Github 代码仓库

<https://github.com/zhisheng17/flink-learning/>

以后这个项目的所有代码都将放在这个仓库里，包含了学习 flink 的一些 demo 和博客。



本文的项目代码在 <https://github.com/zhisheng17/flink-learning/tree/master/flink-learning-connectors/flink-learning-connectors-rabbitmq>

## 相关文章

- 1、[《从0到1学习Flink》—— Apache Flink 介绍](#)
- 2、[《从0到1学习Flink》—— Mac 上搭建 Flink 1.6.0 环境并构建运行简单程序入门](#)
- 3、[《从0到1学习Flink》—— Flink 配置文件详解](#)
- 4、[《从0到1学习Flink》—— Data Source 介绍](#)
- 5、[《从0到1学习Flink》—— 如何自定义 Data Source ?](#)
- 6、[《从0到1学习Flink》—— Data Sink 介绍](#)
- 7、[《从0到1学习Flink》—— 如何自定义 Data Sink ?](#)
- 8、[《从0到1学习Flink》—— Flink Data transformation\(转换\)](#)
- 9、[《从0到1学习Flink》—— 介绍Flink中的Stream Windows](#)
- 10、[《从0到1学习Flink》—— Flink 中的几种 Time 详解](#)
- 11、[《从0到1学习Flink》—— Flink 写入数据到 ElasticSearch](#)
- 12、[《从0到1学习Flink》—— Flink 项目如何运行?](#)
- 13、[《从0到1学习Flink》—— Flink 写入数据到 Kafka](#)
- 14、[《从0到1学习Flink》—— Flink JobManager 高可用性配置](#)
- 15、[《从0到1学习Flink》—— Flink parallelism 和 Slot 介绍](#)
- 16、[《从0到1学习Flink》—— Flink 读取 Kafka 数据批量写入到 MySQL](#)
- 17、[《从0到1学习Flink》—— Flink 读取 Kafka 数据写入到 RabbitMQ](#)