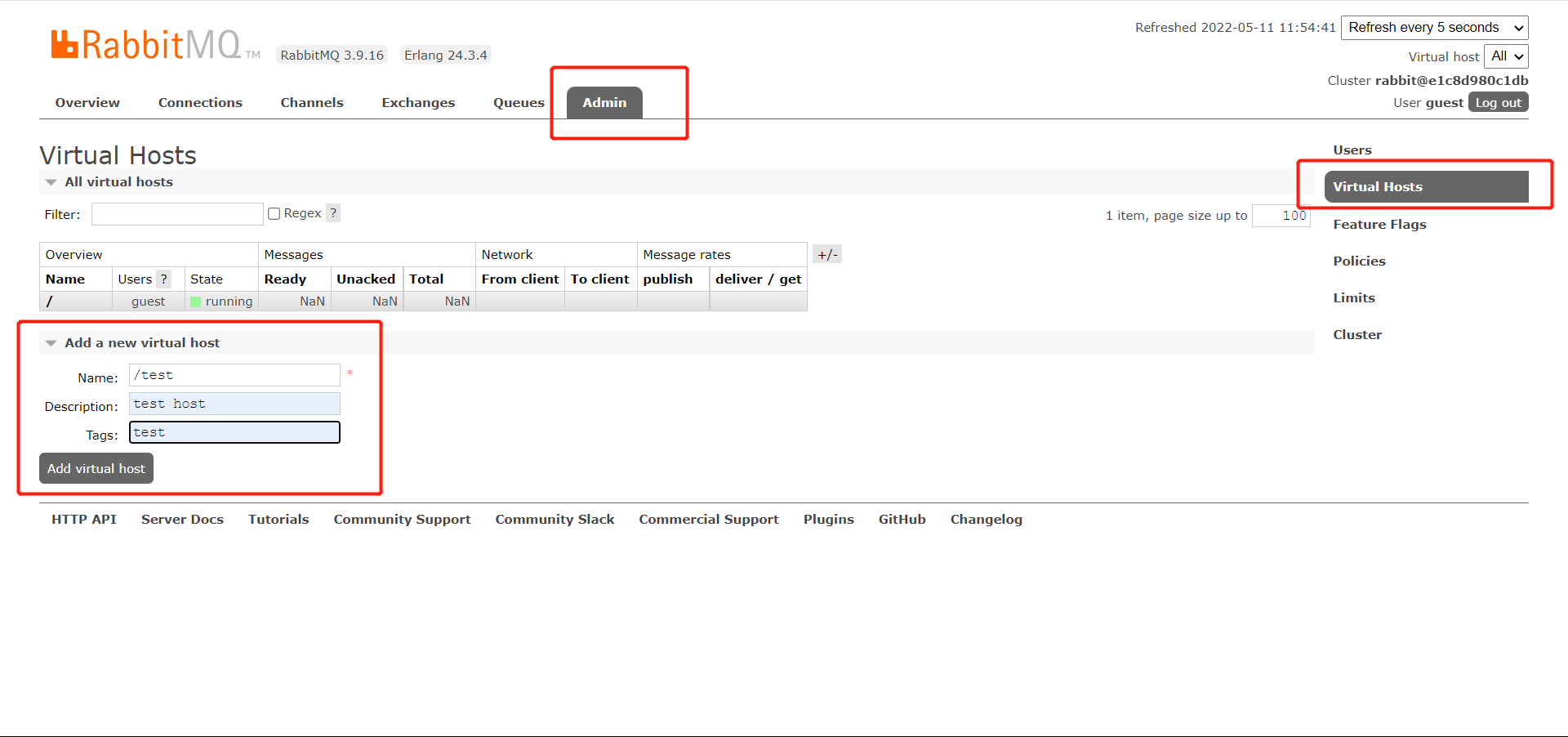
官方文档：<https://www.rabbitmq.com/getstarted.html>

**准备工作**

在开始前我们需要创建一个Virtual Host

创建Virtual Host



注：用户我们使用默认的用户guest

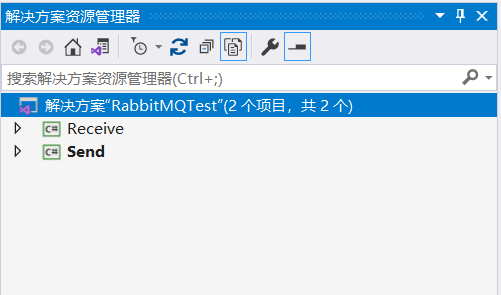
**简单的示例（一个发布者，一个消费者）**

来看下这种模式，发布者将消息发送到队列，消费者从队列接收消息，并给RabbitMQ回复ack，RabbitMQ收到ack后会将消息删除

IMG_256

1. 准备工作

我们创建2个控制台项目Receive和Send，并且在这2个项目安装RabbitMQ.Client包



1. Send代码

using System;

using RabbitMQ.Client;

using System.Text;

class Send

{

public static void Main()

{

// 指定 Virtual Host 和用户

var factory = new ConnectionFactory() { HostName = "192.168.102.130", UserName = "guest", Password = "guest", VirtualHost = "/test" };

// 创建一个连接

using (var connection = factory.CreateConnection())

// 创建一个管道

using (var channel = connection.CreateModel())

{

// 定义一个队列，如果不存在则会创建

channel.QueueDeclare(queue: "hello", durable: false, exclusive: false, autoDelete: false, arguments: null);

string message = "Hello World!";

var body = Encoding.UTF8.GetBytes(message);

// 将消息发送到默认交换器

channel.BasicPublish(exchange: "", routingKey: "hello", basicProperties: null, body: body);

Console.WriteLine(" [x] Sent {0}", message);

}

Console.WriteLine(" Press [enter] to exit.");

Console.ReadLine();

}

}

1. Receive代码

using RabbitMQ.Client;

using RabbitMQ.Client.Events;

using System;

using System.Text;

class Receive

{

public static void Main()

{

// 指定 Virtual Host 和用户

var factory = new ConnectionFactory() { HostName = "192.168.102.130", UserName = "guest", Password = "guest", VirtualHost = "/test" };

// 创建一个连接

using (var connection = factory.CreateConnection())

// 创建一个管道

using (var channel = connection.CreateModel())

{

// 定义一个队列

channel.QueueDeclare(queue: "hello", durable: false, exclusive: false, autoDelete: false, arguments: null);

Console.WriteLine(" [\*] Waiting for messages.");

// 实例一个消费者

var consumer = new EventingBasicConsumer(channel);

// 消息处理器

consumer.Received += (model, ea) =>

{

var body = ea.Body.ToArray();

var message = Encoding.UTF8.GetString(body);

Console.WriteLine(" [x] Received {0}", message);

};

// 启动消费者，在接收消息后自动回复ack

channel.BasicConsume(queue: "hello", autoAck: true, consumer: consumer);

Console.WriteLine(" Press [enter] to exit.");

Console.ReadLine();

}

}

}

1. 知识点

- 定义队列

我们使用如下代码定义队列，当队列不存在时会创建，该代码定义队列hello

// 定义一个队列，如果不存在则会创建

channel.QueueDeclare(queue: "hello", durable: false, exclusive: false, autoDelete: false, arguments: null);

- 发送消息

我们使用如下代码发送消息，这会将消息发送到默认交换器

// 将消息发送到默认交换器

channel.BasicPublish(exchange: "", routingKey: "hello", basicProperties: null, body: body);

- 启动消费者

该代码将消费者连接到hello队列，消费者接收到消息后会回复ack

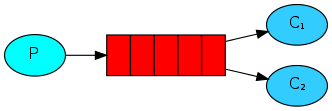
// 启动消费者，在接收消息后自动回复ack

channel.BasicConsume(queue: "hello", autoAck: true, consumer: consumer);

注：消费者可以不回复ack吗？不行，消费者必须回复ack，自动或者手动回复

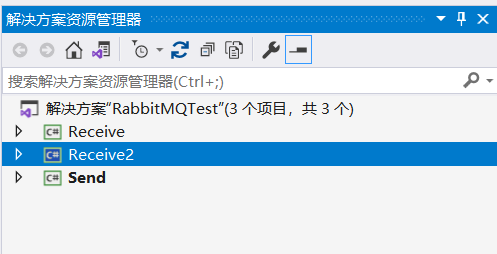
**Work queues（工作队列）**

如果我们有一个发布者，多个消费者，消费者在处理消息时可能会失败，并且RabbitMQ可能会挂掉，那如何处理这些问题呢？



1. 准备工作

我们再增加一个Receive2，其代码跟Receive一样



1. Send代码

using System;

using RabbitMQ.Client;

using System.Text;

using System.Threading;

class Send

{

public static void Main()

{

// 指定 Virtual Host 和用户

var factory = new ConnectionFactory() { HostName = "192.168.102.130", UserName = "guest", Password = "guest", VirtualHost = "/test" };

// 创建一个连接

using (var connection = factory.CreateConnection())

// 创建一个管道

using (var channel = connection.CreateModel())

{

// 定义一个队列，如果不存在则会创建

channel.QueueDeclare(queue: "task\_queue",

durable: true, // 持久化队列

exclusive: false,

autoDelete: false,

arguments: null);

// 设置消息持久化

var properties = channel.CreateBasicProperties();

properties.Persistent = true;

var n = 0;

while (true) {

n++;

string message = "Hello World! " + n;

var body = Encoding.UTF8.GetBytes(message);

// 将消息发送到默认交换器

channel.BasicPublish(exchange: "", routingKey: "task\_queue", basicProperties: properties, body: body);

Console.WriteLine(" [x] Sent {0}", message);

Thread.Sleep(1000);

}

}

Console.WriteLine(" Press [enter] to exit.");

Console.ReadLine();

}

}

1. Receive代码

using RabbitMQ.Client;

using RabbitMQ.Client.Events;

using System;

using System.Text;

using System.Threading;

class Receive

{

public static void Main()

{

// 指定 Virtual Host 和用户

var factory = new ConnectionFactory() { HostName = "192.168.102.130", UserName = "guest", Password = "guest", VirtualHost = "/test" };

// 创建一个连接

using (var connection = factory.CreateConnection())

// 创建一个管道

using (var channel = connection.CreateModel())

{

// 定义队列

channel.QueueDeclare(queue: "task\_queue",

durable: true, // 持久化队列

exclusive: false,

autoDelete: false,

arguments: null);

Console.WriteLine(" [\*] Waiting for messages.");

// 实例消费者

var consumer = new EventingBasicConsumer(channel);

consumer.Received += (sender, ea) =>

{

var body = ea.Body.ToArray();

var message = Encoding.UTF8.GetString(body);

Console.WriteLine(" [x] Received {0}", message);

// 假装我们正在执行一个长时间任务

Thread.Sleep(3000);

Console.WriteLine(" [x] Done");

// 手动确认

channel.BasicAck(deliveryTag: ea.DeliveryTag, multiple: false);

};

// 启动消费者,autoAck: false表示ack由接收器回复

channel.BasicConsume(queue: "task\_queue", autoAck: false, consumer: consumer);

Console.WriteLine(" Press [enter] to exit.");

Console.ReadLine();

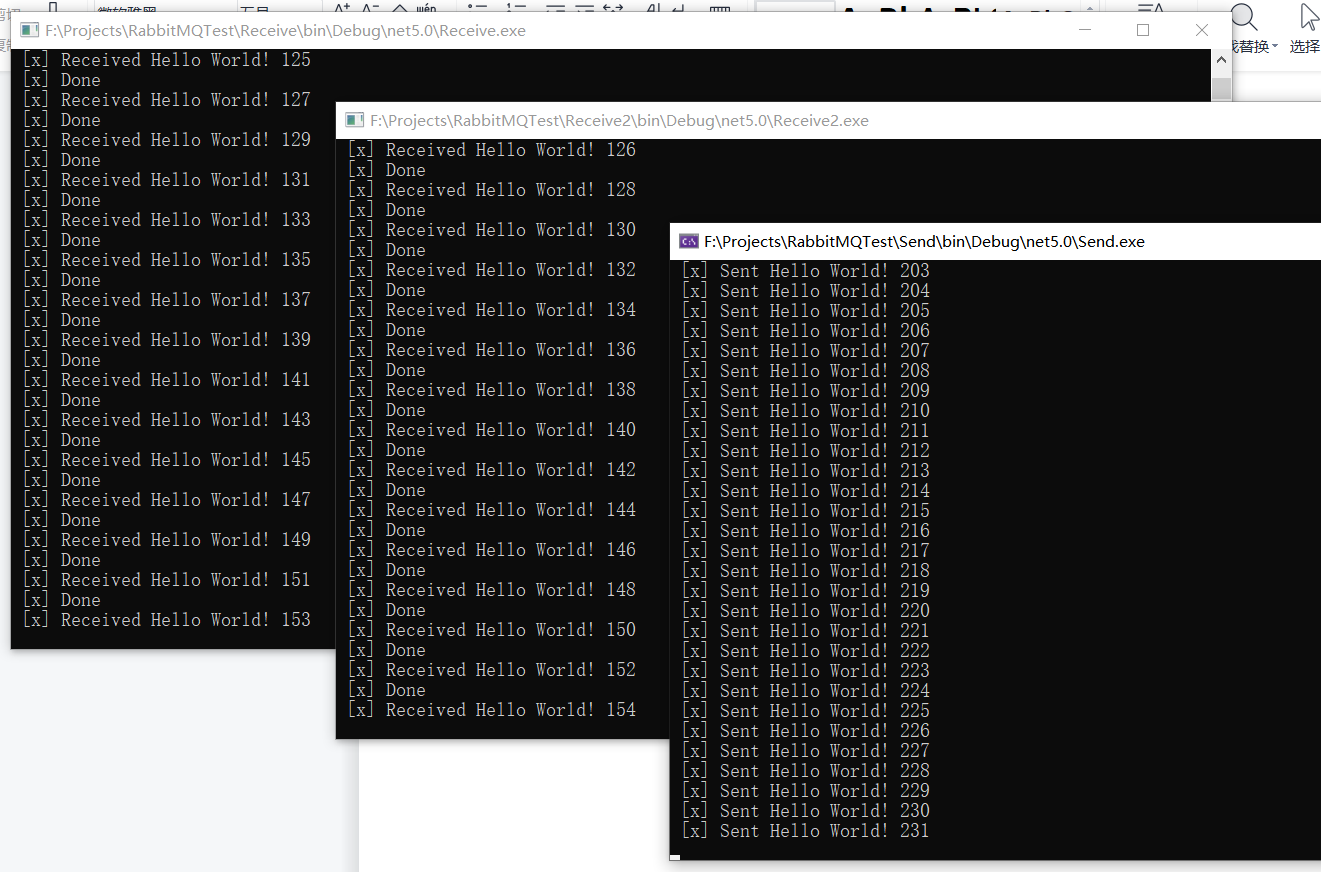
}

}

}

1. 运行效果

可以看下面的图, 2个消费者的处理速度小于发布者的速度



1. 知识点

- 消息持久化

消息持久化首先要定义队列可持久化

// 定义一个队列，如果不存在则会创建

channel.QueueDeclare(queue: "task\_queue",

durable: true, // 持久化队列

exclusive: false,

autoDelete: false,

arguments: null);

然后指定消息持久化

// 设置消息持久化

var properties = channel.CreateBasicProperties();

properties.Persistent = true;

....

// 将消息发送到队列

channel.BasicPublish(exchange: "", routingKey: "task\_queue", basicProperties: properties, body: body);

- 控制的ack回复

要自己回复ack首先要再启动消费者时指定autoAck: false

// 启动消费者,autoAck: false表示ack由接收器回复

channel.BasicConsume(queue: "task\_queue", autoAck: false, consumer: consumer);

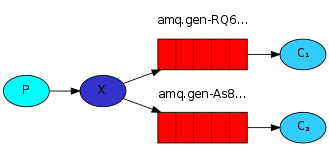
然后处理完任务后手动回复

// 手动确认

channel.BasicAck(deliveryTag: ea.DeliveryTag, multiple: false);

**发布订阅模式**

发布订阅模式下，发布者不关心是否存在消费者，如果有消费者，则消息被发送到消费者，否则消息会被丢弃



1. Send代码

using System;

using RabbitMQ.Client;

using System.Text;

using System.Threading;

class Send

{

public static void Main()

{

// 指定 Virtual Host 和用户

var factory = new ConnectionFactory() { HostName = "192.168.102.130", UserName = "guest", Password = "guest", VirtualHost = "/test" };

// 创建一个连接

using (var connection = factory.CreateConnection())

// 创建一个管道

using (var channel = connection.CreateModel())

{

// 定义一个交换器，交换器类型问 Fanout，如果不存在则创建

channel.ExchangeDeclare(exchange: "logs", type: ExchangeType.Fanout);

string message = "Hello World! ";

var body = Encoding.UTF8.GetBytes(message);

// 将消息发送到logs交换器

channel.BasicPublish(exchange: "logs",

routingKey: "",

basicProperties: null,

body: body);

Console.WriteLine(" [x] Sent {0}", message);

}

Console.WriteLine(" Press [enter] to exit.");

Console.ReadLine();

}

}

1. Receive代码

using RabbitMQ.Client;

using RabbitMQ.Client.Events;

using System;

using System.Text;

using System.Threading;

class Receive

{

public static void Main()

{

// 指定 Virtual Host 和用户

var factory = new ConnectionFactory() { HostName = "192.168.102.130", UserName = "guest", Password = "guest", VirtualHost = "/test" };

// 创建一个连接

using (var connection = factory.CreateConnection())

// 创建一个管道

using (var channel = connection.CreateModel())

{

// 定义一个交换器，交换器类型问 Fanout，如果不存在则创建

channel.ExchangeDeclare(exchange: "logs", type: ExchangeType.Fanout);

// 生成一个随机队列

var queueName = channel.QueueDeclare().QueueName;

// 将队列绑定到交换器logs

channel.QueueBind(queue: queueName,

exchange: "logs",

routingKey: "");

Console.WriteLine(" [\*] Waiting for logs.");

// 实例消费者

var consumer = new EventingBasicConsumer(channel);

consumer.Received += (model, ea) =>

{

var body = ea.Body.ToArray();

var message = Encoding.UTF8.GetString(body);

Console.WriteLine(" [x] {0}", message);

};

// 启动消费者

channel.BasicConsume(queue: queueName,

autoAck: true,

consumer: consumer);

Console.WriteLine(" Press [enter] to exit.");

Console.ReadLine();

}

}

}

1. 知识点

- 定义交换器

如下代码定义了一个交换器

// 定义一个交换器，交换器类型为 Fanout，如果不存在则创建

channel.ExchangeDeclare(exchange: "logs", type: ExchangeType.Fanout);

- 生成随机队列

如下代码生成一个随机队列，当消费者断开连接时，该队列会被删除

// 生成一个随机队列

var queueName = channel.QueueDeclare().QueueName;

- 队列绑定到交换器

// 将队列绑定到交换器logs

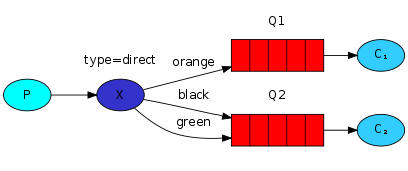
channel.QueueBind(queue: queueName,

exchange: "logs",

routingKey: "");

**消息路由**

我们将队列绑定到路由上，这样我们就可以通过指定路由来指定消息发送到哪个队列上了



1. Send代码

using System;

using RabbitMQ.Client;

using System.Text;

using System.Threading;

class Send

{

public static void Main()

{

// 指定 Virtual Host 和用户

var factory = new ConnectionFactory() { HostName = "192.168.102.130", UserName = "guest", Password = "guest", VirtualHost = "/test" };

// 创建一个连接

using (var connection = factory.CreateConnection())

// 创建一个管道

using (var channel = connection.CreateModel())

{

// 定义一个交换器，交换器类型为 Direct，如果不存在则创建

channel.ExchangeDeclare(exchange: "direct\_logs",

type: ExchangeType.Direct);

string message = "Hello World! ";

var body = Encoding.UTF8.GetBytes(message);

// 将消息发送到交换器，routingKey 为 myroutekey

channel.BasicPublish(exchange: "direct\_logs",

routingKey: "myroutekey",

basicProperties: null,

body: body);

Console.WriteLine(" [x] Sent {0}", message);

}

Console.WriteLine(" Press [enter] to exit.");

Console.ReadLine();

}

}

1. Receive代码

using RabbitMQ.Client;

using RabbitMQ.Client.Events;

using System;

using System.Text;

using System.Threading;

class Receive

{

public static void Main()

{

// 指定 Virtual Host 和用户

var factory = new ConnectionFactory() { HostName = "192.168.102.130", UserName = "guest", Password = "guest", VirtualHost = "/test" };

// 创建一个连接

using (var connection = factory.CreateConnection())

// 创建一个管道

using (var channel = connection.CreateModel())

{

// 定义一个交换器，交换器类型为 Direct，如果不存在则创建

channel.ExchangeDeclare(exchange: "direct\_logs",

type: "direct");

// 创建一个随机队列

var queueName = channel.QueueDeclare().QueueName;

// 将队列绑定到direct\_logs，绑定的 routingKey 为 myroutekey

channel.QueueBind(queue: queueName,

exchange: "direct\_logs",

routingKey: "myroutekey");

// 将队列绑定到direct\_logs，绑定的 routingKey 为 myroutekey2

channel.QueueBind(queue: queueName,

exchange: "direct\_logs",

routingKey: "myroutekey2");

Console.WriteLine(" [\*] Waiting for messages.");

// 实例一个消费者

var consumer = new EventingBasicConsumer(channel);

consumer.Received += (model, ea) =>

{

var body = ea.Body.ToArray();

var message = Encoding.UTF8.GetString(body);

var routingKey = ea.RoutingKey;

Console.WriteLine(" [x] Received '{0}':'{1}'",

routingKey, message);

};

// 启动消费者

channel.BasicConsume(queue: queueName,

autoAck: true,

consumer: consumer);

Console.WriteLine(" Press [enter] to exit.");

Console.ReadLine();

}

}

}

1. 知识点

- 定义交换器

// 定义一个交换器，交换器类型为 Direct，如果不存在则创建

channel.ExchangeDeclare(exchange: "direct\_logs",

type: ExchangeType.Direct);

- 绑定队列

channel.QueueBind(queue: queueName,

exchange: "direct\_logs",

routingKey: "myroutekey");