# 分布式消息服务 RabbitMQ 版

# 开发指南

**文档版本** 01

**发布日期** 2023-11-15

**华为云计算技术有限公司**

## 商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

# 目 录



  

# 1 概述

本指南主要介绍RabbitMQ实例连接信息的收集，如获取RabbitMQ实例连接地址与端口、访问实例的用户名和密码，然后提供Python语言和Spring Boot的连接示例。

RabbitMQ实例完全兼容开源RabbitMQ协议，Python以外的语言，请参考RabbitMQ官网提供的不同语言的连接和使用向导：**https://www.rabbitmq.com/getstarted.html**。

## 开源 SDK 列表

分布式消息服务RabbitMQ版支持所有开源版本的SDK，常见的开源SDK如**表1-1**所示。

**表 1-1** 开源 SDK 列表

| 编程语言 | SDK |
| --- | --- |
| Java | **rabbitmq-java-client** |
| Spring Framework | **SpringAMQP** |
| .Net | **rabbitmq-dotnet-client** |
| Python | **pika** |
| PHP | **php-amqplib** |
| C | **rabbitmq-c** |
| Go | **amqp091-go** |

**说明**

推荐使用最新Release版本的SDK。

## 客户端网络环境说明

客户端可以通过以下方式访问RabbitMQ实例：

1. VPC内子网地址访问

   如果客户端是云上ECS，与RabbitMQ实例处于同region同VPC，则可以直接访问RabbitMQ实例提供的VPC内子网地址。

2. VPC对等连接方式访问

   如果客户端是云上ECS，与RabbitMQ实例处于相同region但不同VPC，则可以通过建立VPC对等连接后，访问RabbitMQ实例提供的VPC内子网地址。

   关于创建和使用VPC对等连接，可参考**VPC对等连接说明**。

3. 公网访问

   客户端在其他网络环境，或者与RabbitMQ实例处于不同region，则访问实例的公网地址。

**说明**

不同网络环境，对于客户端配置来说，只是连接地址的差异，其他都一样。因此，本手册以同一VPC内子网地址的方式，介绍客户端开发环境搭建。

遇到连接超时或失败时，请注意确认网络是否连通。可使用telnet方式，检测实例连接地址与端口。

# 2 收集连接信息

- 实例连接地址与端口

  实例创建后，从实例的“基本信息”页签的“连接信息”中获取。

  **图 2-1** 查看 RabbitMQ 实例连接地址与端口

  
  

- 访问实例的用户名和密码

  实例创建后，从实例的“基本信息”页签的“连接信息”中获取用户名。如果忘记了密码，单击“重置密码”，重新设置密码。

  

# 3 Python 客户端使用说明

本文以Linux CentOS环境为例，介绍Python版本的RabbitMQ客户端连接指导，包括RabbitMQ客户端安装，以及生产、消费消息。

使用前请参考**收集连接信息**收集RabbitMQ所需的连接信息。

## 准备环境

- Python

  一般系统预装了Python。在命令行输入**python**，得到如下回显，说明Python已安装。

  ```
  [root@ecs-test python]# python3
  Python 3.7.1 (default, Jul  5 2020, 14:37:24)
  [GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux
  Type "help", "copyright", "credits" or "license" for more information.
  >>>
  ```

  如果未安装Python，请使用以下命令安装：

  **yum install python**

- Python版的RabbitMQ客户端，本文使用pika作为连接RabbitMQ的客户端

  执行以下命令，安装推荐版本的pika：

  **pip install pika**

  如果无法使用pip命令安装pika，建议改用pip3命令安装pika：

  **pip3 install pika**

## 生产消息

**说明**

以下加粗内容需要替换为实例自有信息，请根据实际情况替换。

- SSL认证方式

  ```
  import pika
  import ssl

  # 连接信息
  conf = {
      'host': 'ip',
      'port': 5671,
      'queue_name': 'queue-test',
      'username': 'root',
      'password': 'password'
  ```

```
}

context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
credentials = pika.PlainCredentials(conf['username'], conf['password'])
parameters = pika.ConnectionParameters(conf['host'],
                                       conf['port'],
                                       '/',
                                       credentials,
                                       ssl_options=pika.SSLOptions(context))

connection = pika.BlockingConnection(parameters)
channel = connection.channel()
channel.queue_declare(conf['queue_name'])
data = bytes('Hello World!', encoding='utf-8')
channel.basic_publish(exchange='',
                      routing_key=conf['queue_name'],
                      body=data)

print(" [x] Sent 'Hello World!'")

connection.close()
```

- 非SSL认证方式

```
import pika

# 连接信息
conf = {
    'host': 'ip',
    'port': 5672,
    'queue_name': 'queue-test',
    'username': 'root',
    'password': 'password'
}

credentials = pika.PlainCredentials(conf['username'], conf['password'])
parameters = pika.ConnectionParameters(conf['host'],
                                       conf['port'],
                                       '/',
                                       credentials)

connection = pika.BlockingConnection(parameters)
channel = connection.channel()

channel.queue_declare(conf['queue_name'])

data = bytes("Hello World!", encoding="utf-8")

channel.basic_publish(exchange='',
                      routing_key=conf['queue_name'],
                      body=data)

print(" [x] Sent 'Hello World!'")

connection.close()
```

## 消费消息

**说明**

以下加粗内容需要替换为实例自有信息，请根据实际情况替换。

- SSL认证方式

```
import pika
import ssl

# 连接信息
conf = {
    'host': 'ip',
    'port': 5671,
```

```
    'queue_name': 'queue-test',
    'username': 'root',
    'password': 'password'
}

context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
credentials = pika.PlainCredentials(conf['username'], conf['password'])
parameters = pika.ConnectionParameters(conf['host'],
                                       conf['port'],
                                       '/',
                                       credentials,
                                       ssl_options=pika.SSLOptions(context))

connection = pika.BlockingConnection(parameters)
channel = connection.channel()
channel.queue_declare(conf['queue_name'])


def callback(ch, method, properties, body):
    print(" [x] Received %r" % body.decode('utf-8'))


channel.basic_consume(queue=conf['queue_name'], on_message_callback=callback, auto_ack=True)

print(' [*] Waiting for messages. To exit press CTRL+C')
channel.start_consuming()
```

- 非SSL认证方式

```
import pika

# 连接信息
conf = {
    'host': 'ip',
    'port': 5672,
    'queue_name': 'queue-test',
    'username': 'root',
    'password': 'password'
}

credentials = pika.PlainCredentials(conf['username'], conf['password'])
parameters = pika.ConnectionParameters(conf['host'],
                                       conf['port'],
                                       '/',
                                       credentials)

connection = pika.BlockingConnection(parameters)
channel = connection.channel()
channel.queue_declare(conf['queue_name'])


def callback(ch, method, properties, body):
    print(" [x] Received %r" % body.decode('utf-8'))


channel.basic_consume(queue=conf['queue_name'], on_message_callback=callback, auto_ack=True)

print(' [*] Waiting for messages. To exit press CTRL+C')
channel.start_consuming()
```

# 4 使用 Spring Boot 连接 RabbitMQ 实例

本文介绍如何使用Spring Boot连接RabbitMQ实例进行消息的生产和消费。

使用前请参考**收集连接信息**收集RabbitMQ所需的连接信息。

## 在 pom.xml 文件中引入 spring-boot-starter-amqp 依赖

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-amqp</artifactId>
  <version>2.3.1.RELEASE</version>
</dependency>
```

## （可选）在 application.properties 文件中填写配置

如果RabbitMQ实例已开启SSL，在“application.properties”文件中填写如下配置。

```
#开启SSL认证
spring.rabbitmq.ssl.enabled=true
#SSL使用的算法
spring.rabbitmq.ssl.algorithm=TLSv1.2
#是否启用主机验证
spring.rabbitmq.ssl.verify-hostname=false
#是否启用服务端证书验证
spring.rabbitmq.ssl.validate-server-certificate=false
```

## 生产消息

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ProducerController {
    private static final Logger LOG = LoggerFactory.getLogger(ProducerController.class);

    @Autowired
    private RabbitTemplate;

    @GetMapping(value = "/send")
    public boolean send(String msg, Long delayTime) {
        rabbitTemplate.convertAndSend("ex-sour", "abc", msg,
            message -> {
                /**
```

```
                * 设置延迟时间
                */
               message.getMessageProperties().setHeader("x-delay", delayTime);
               return message;
           });
       LOG.info("发送延迟消息：{}，延时：{}ms", msg, delayTime);
       return true;
   }
}
```

## 消费消息

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.stereotype.Component;
import java.text.SimpleDateFormat;
import java.util.Date;


@Component
public class ReceiveMsgService {
   Logger LOG =  LoggerFactory.getLogger(ReceiveMsgService.class);

   @RabbitListener(queues = "test")
   public void receive(String message) {
       SimpleDateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
       LOG.info("receive message: {}", message + " 接收时间：" + simpleDateFormat.format(new Date()));
   }
}
```