中国移动
China Mobile

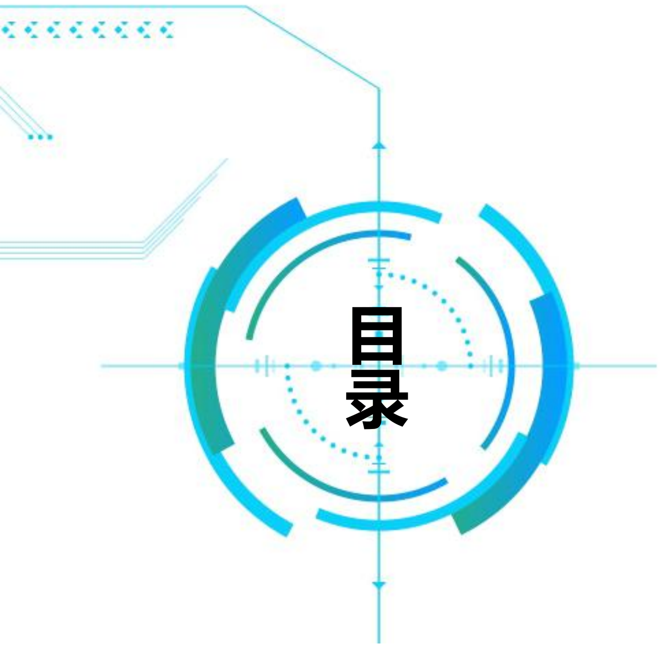
基于Pulsar架构的移动云AMQP消息队列设计与应用

中国移动云能力中心

张浩

www.10086.cn

目录



1 相关背景

2 AMQP 0-9-1

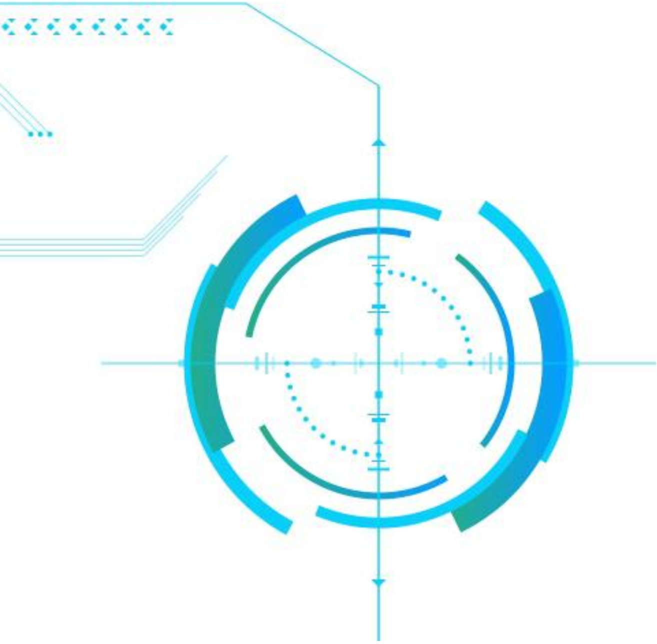
3 Protocol Handler

4 我们怎么实现AoP

5 怎么使用AoP

6 怎么使用移动云上的AMQP

7 后续计划



相关背景

相关背景

- 我们为什么要自己做AMQP消息队列?
  - OpenStack中默认使用Rabbitmq作为RPC通信组件
    - 线上环境经常出问题，运维比较难
  - 移动云客户的需求
    - 移动云我们已上线消息队列RocketMQ以及MQTT服务，但是有不少客户对AMQP消息队列需求强烈
- 我们为什么选择基于Pulsar来实现AMQP协议?
  - 与StreamNative的合作开发
    - 社区支持，共同开发AMQP on Pulsar.

AMQP 0-9-1

什么是 AMQP 0-9-1?

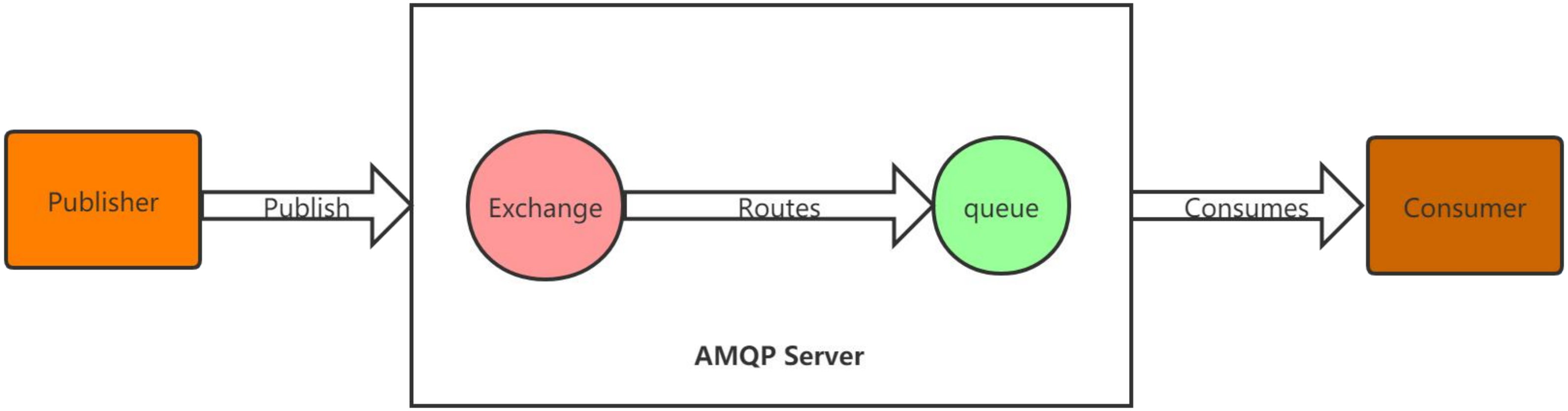
AMQP 0-9-1 (Advanced Message Queuing Protocol) is a messaging protocol that enables conforming client applications to communicate with conforming messaging middleware brokers.

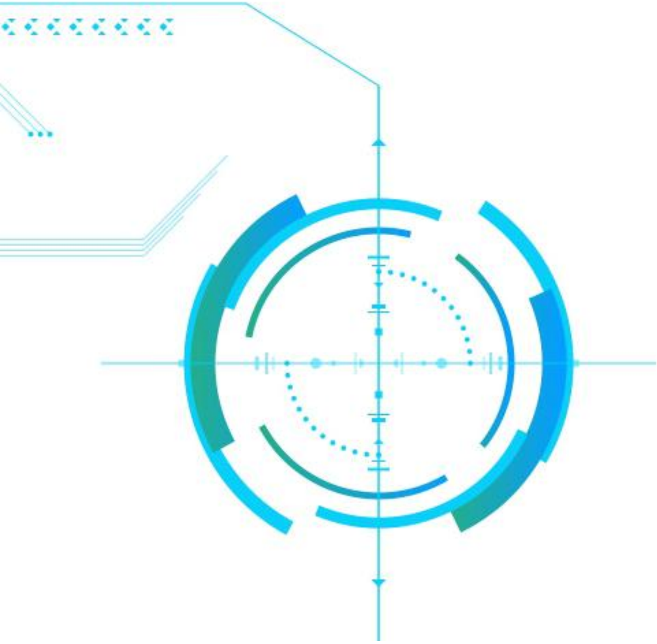
AMQP-0-9-1

- VirtualHost
  - 资源的逻辑分组以及隔离
- Exchange
  - 消息路由
- Queue
  - 消息存储
- Binding
  - 路由规则

AMQP-0-9-1

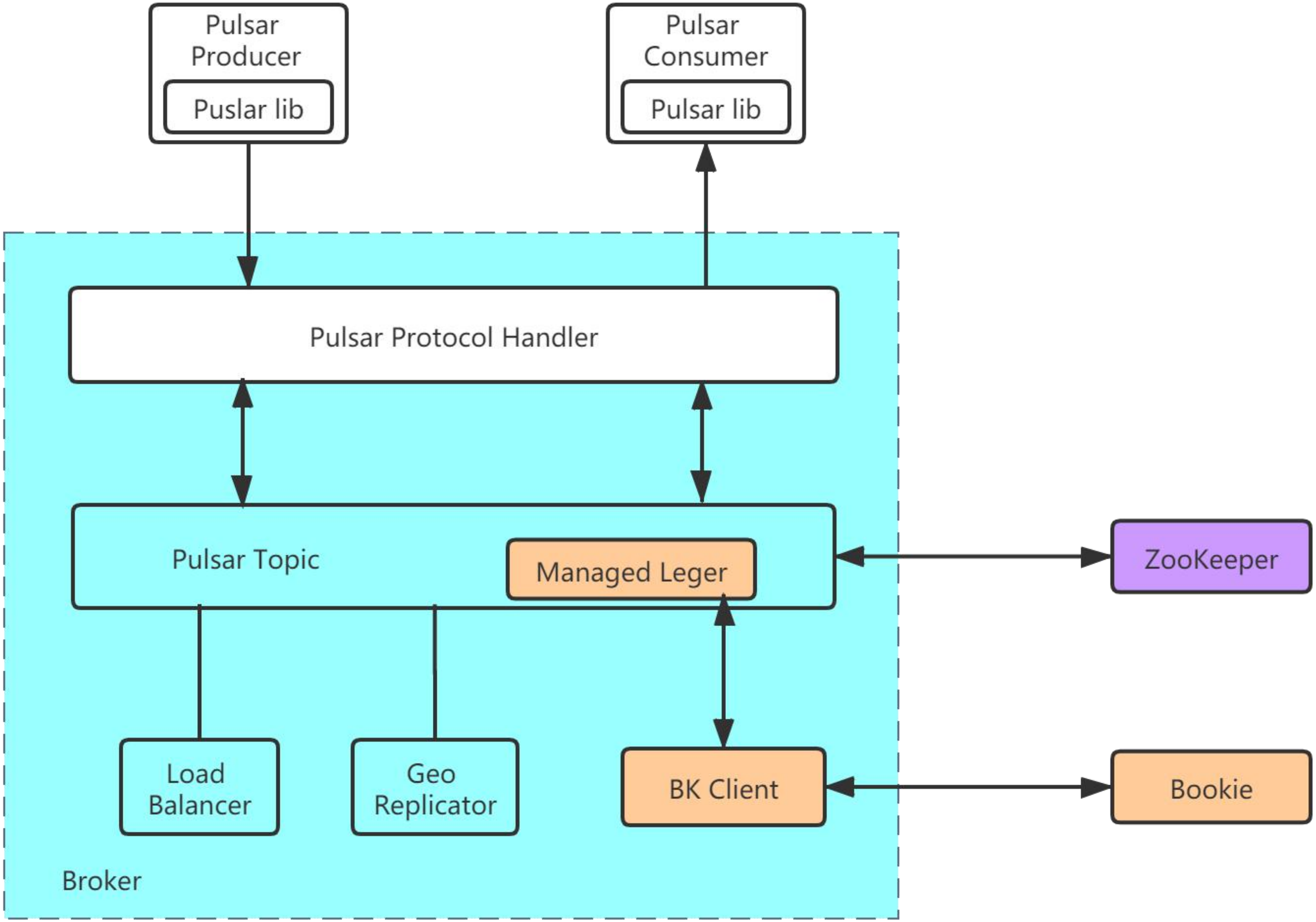
“Hello, world” example routing



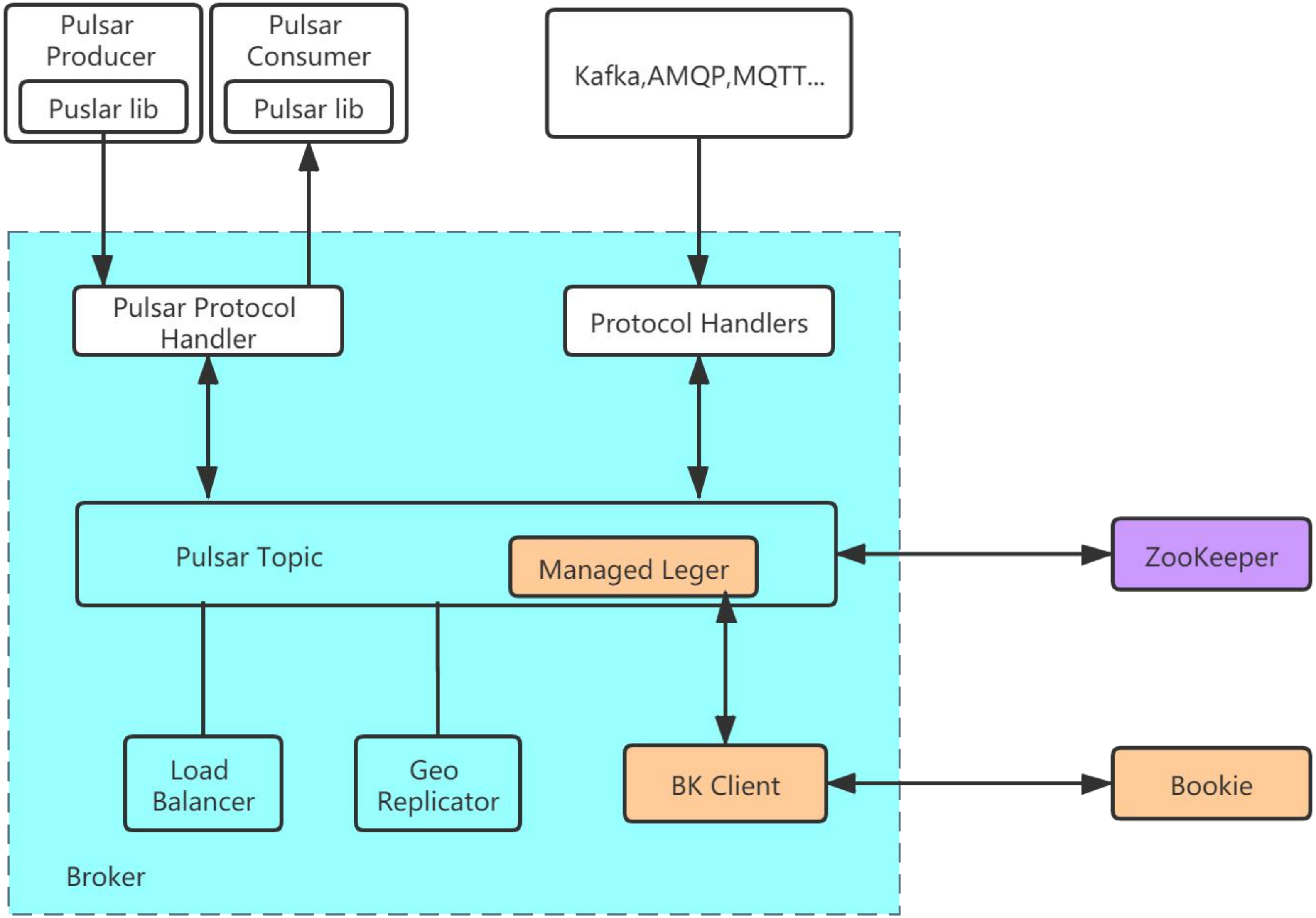


Protocol Handler

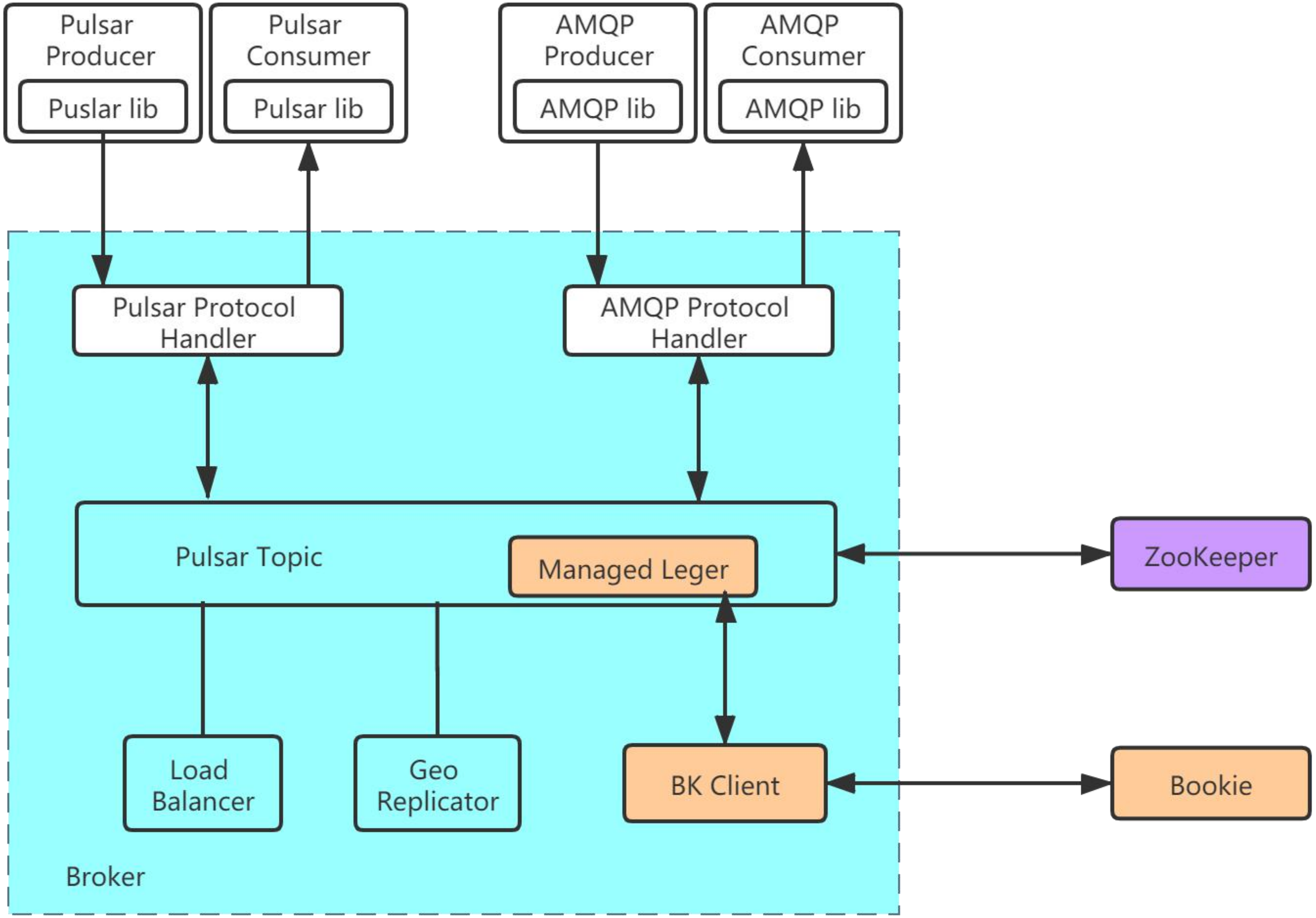
Pulsar Protocol Handler



Protocol Handlers



AMQP-on-Pulsar Protocol Handler



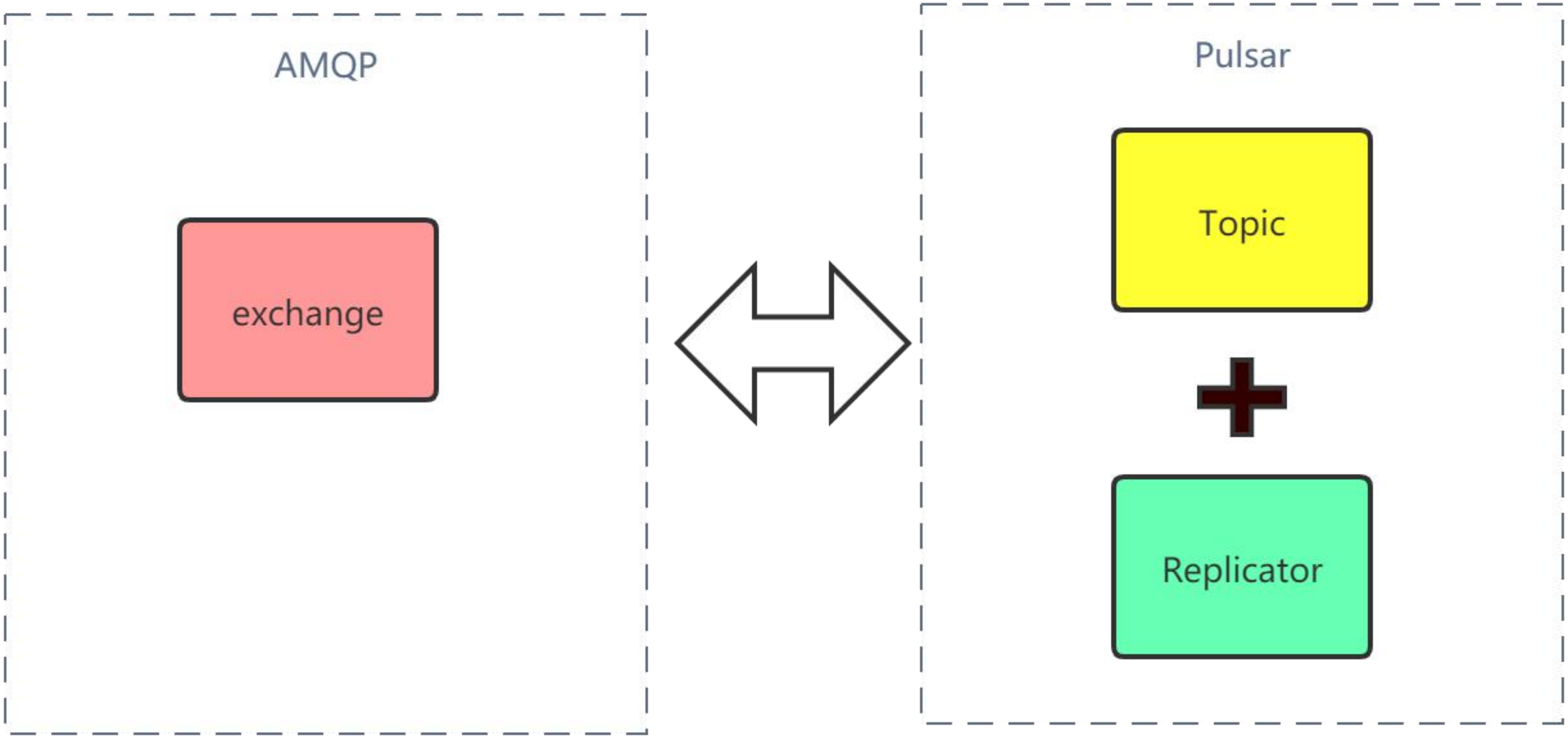
我们是怎么实现AoP?

实现

- Pulsar to AMQP model
- Message Publish
- Message Consume
- Proxy

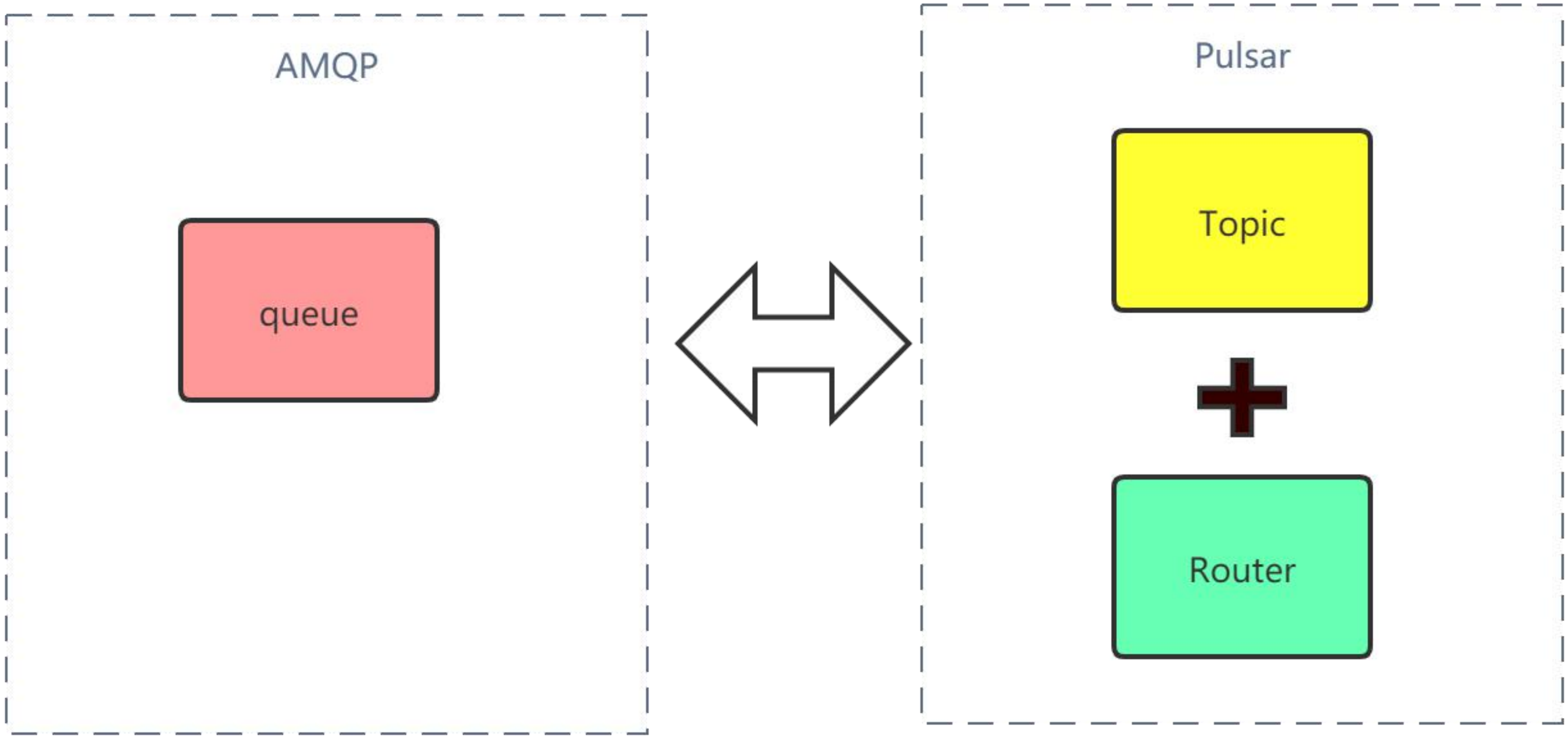
Pulsar to AMQP model

AMQP Exchange-> Pulsar topic + Replicator



Pulsar to AMQP model

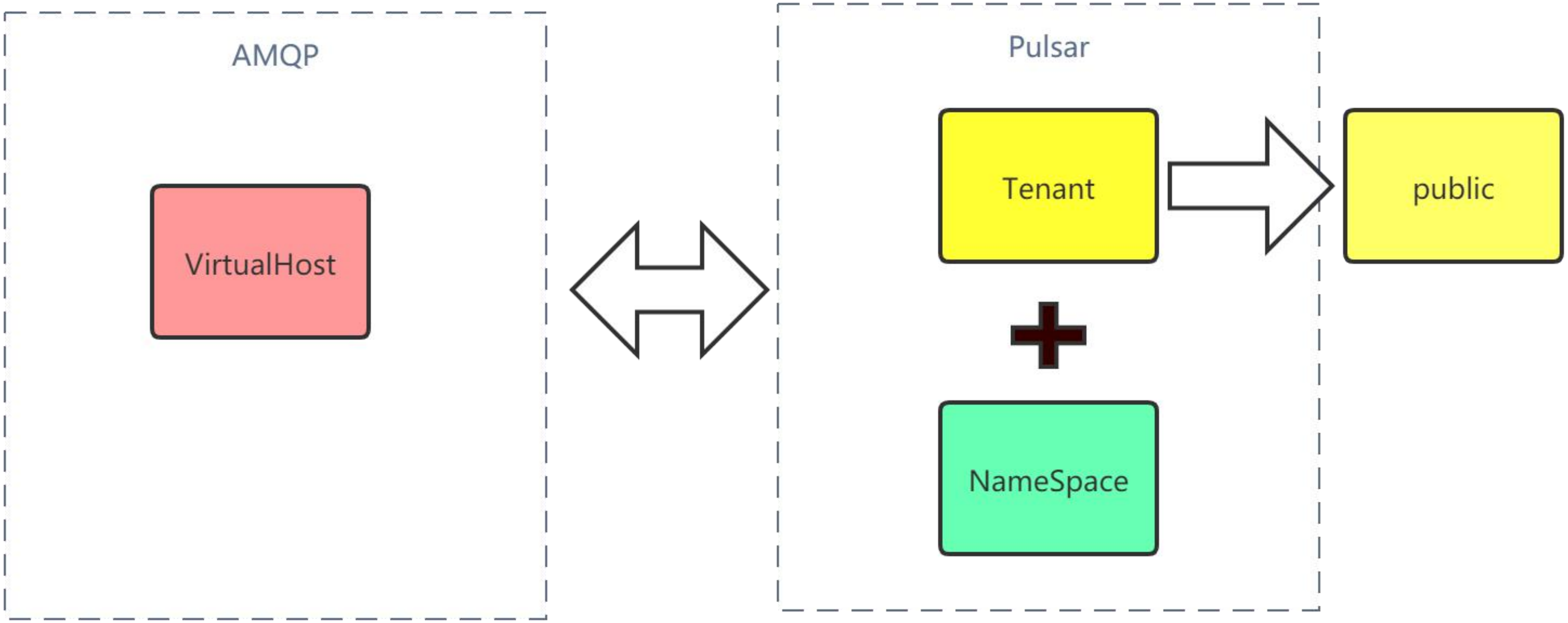
AMQP Queue-> Pulsar topic + Router



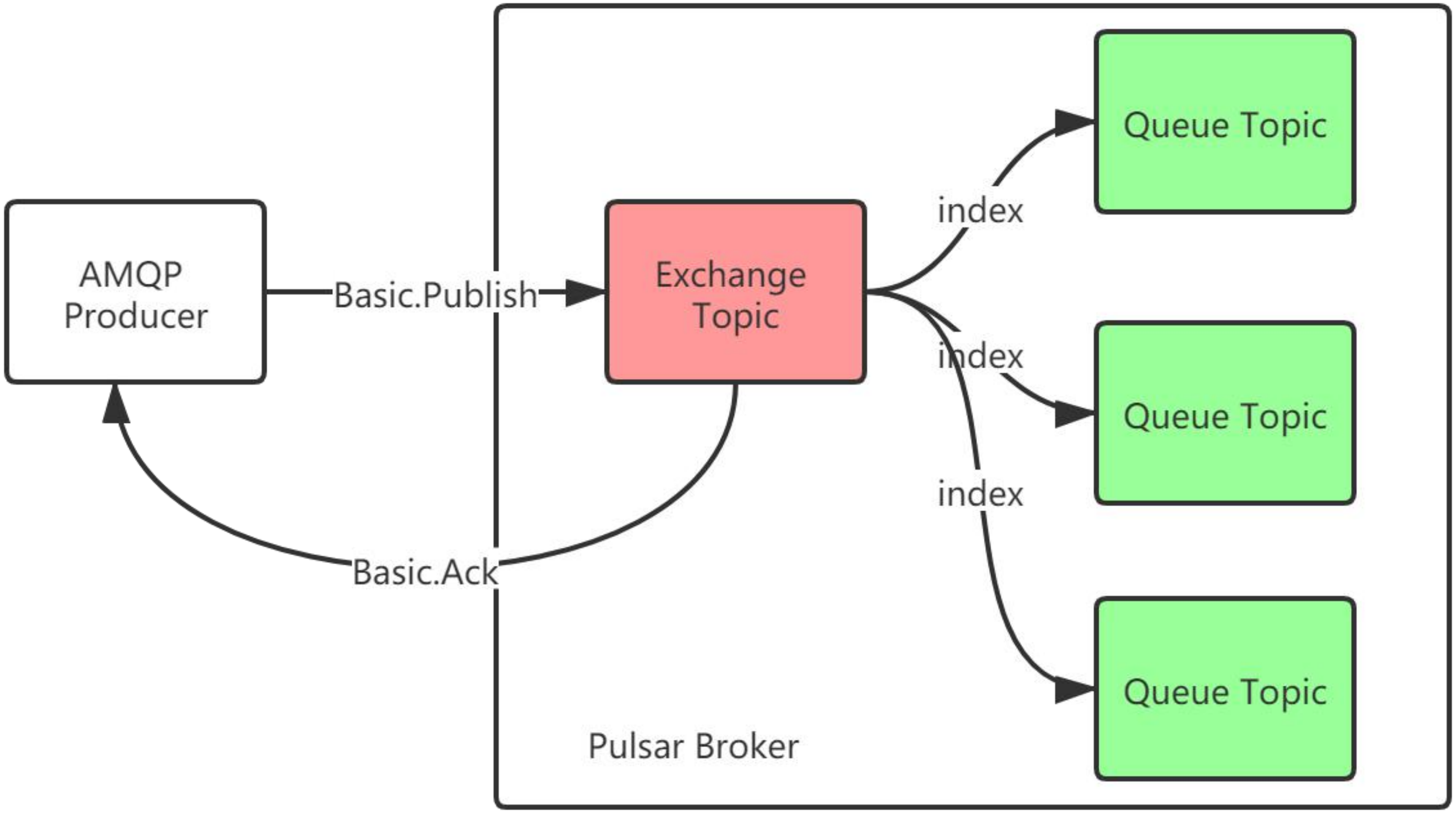
Pulsar to AMQP model

AMQP VirtualHost -> Pulsar tenant/namespace

persistent	://	my-tenant	/	my-namespace	/	my-topic
Topic Domain		Tenant		Namespace		Topic



Message Publish



Message Publish

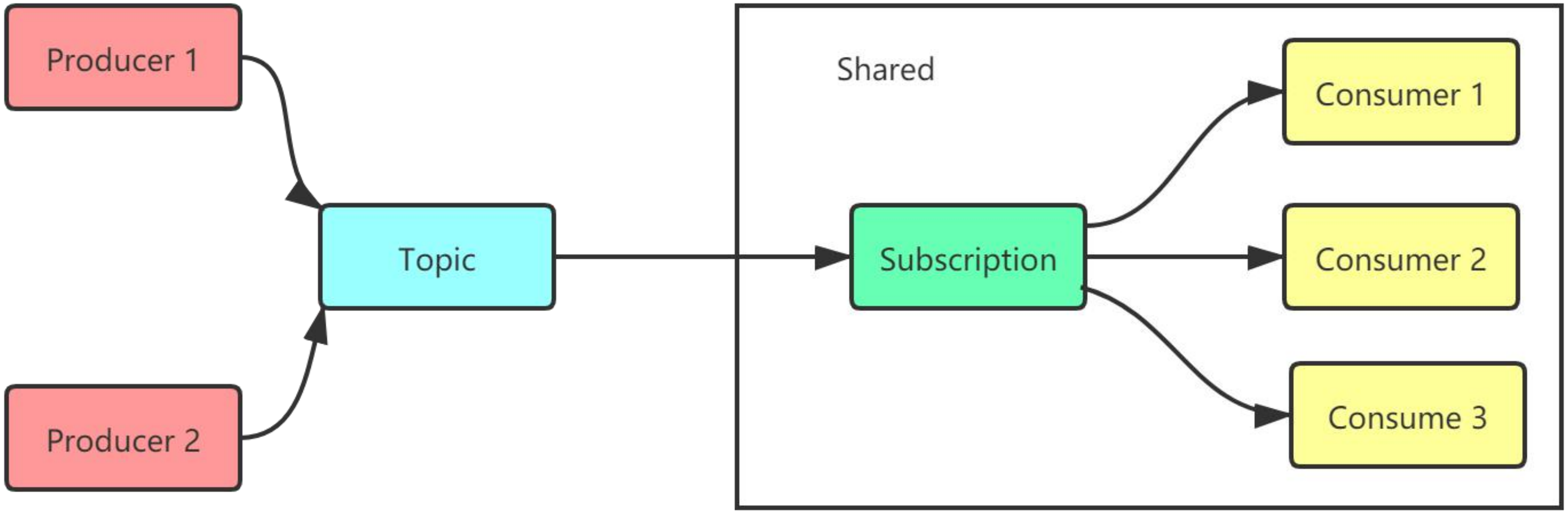
- AMQP message -> Pulsar message
- topic.publishMessage
- Exchange topic replicator -> Queue topic

Message Publish

Exchange topic中为什么要添加replicator?

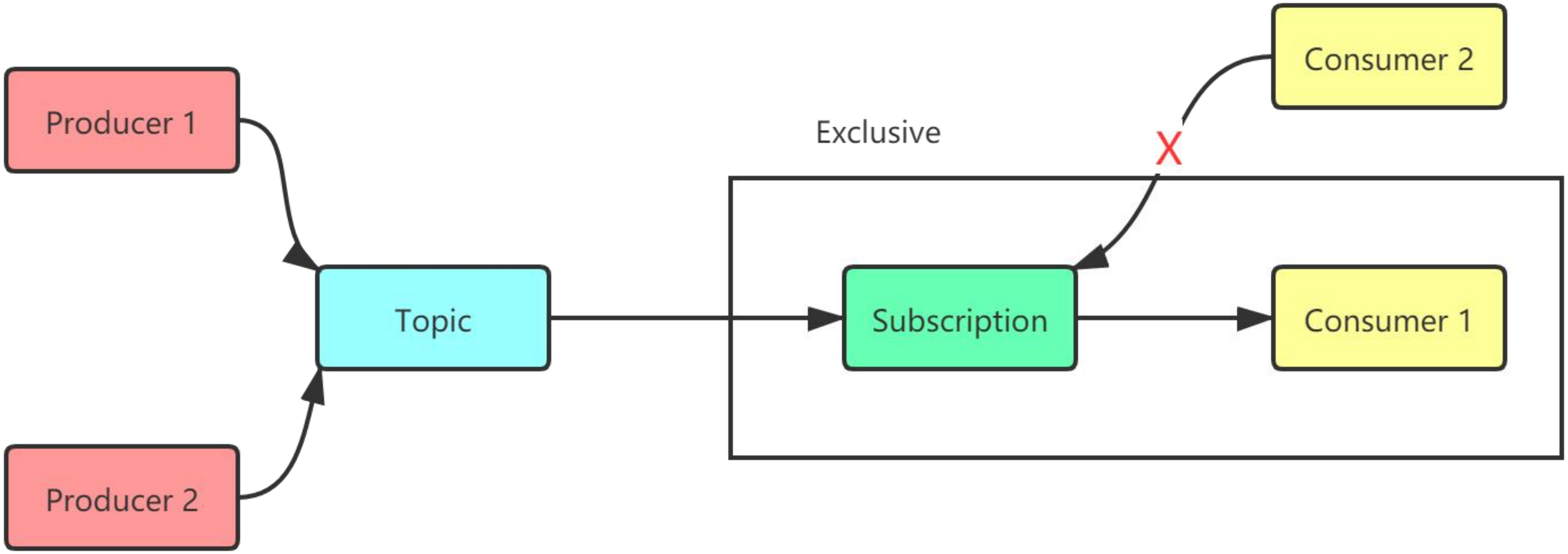
Message Consume

Pulsar shared subscription-> AMQP default



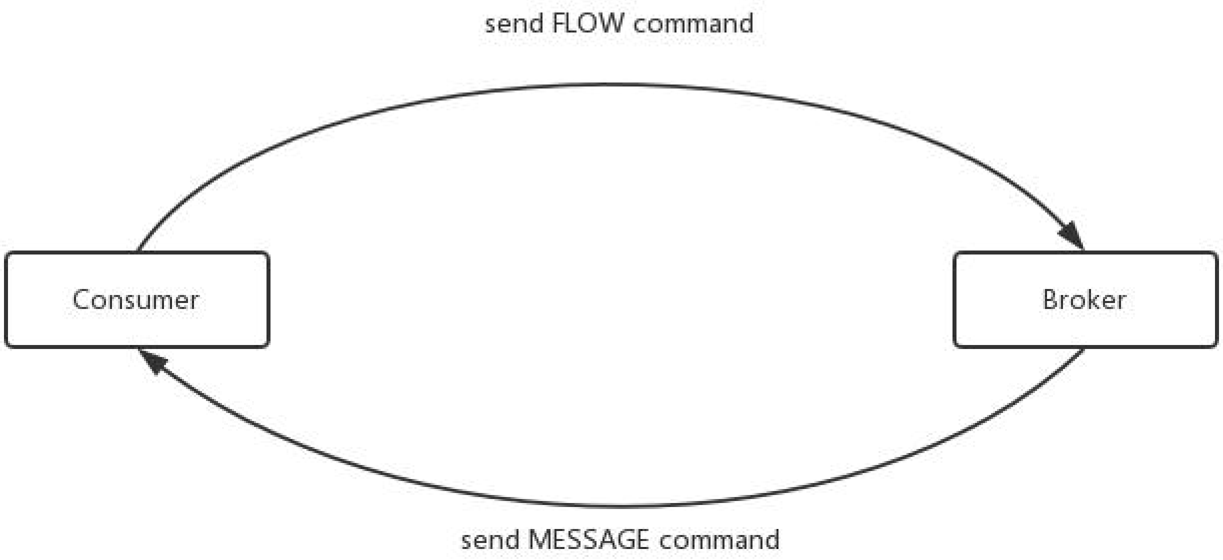
Message Consume

Pulsar exclusive subscription -> AMQP exclusive queue

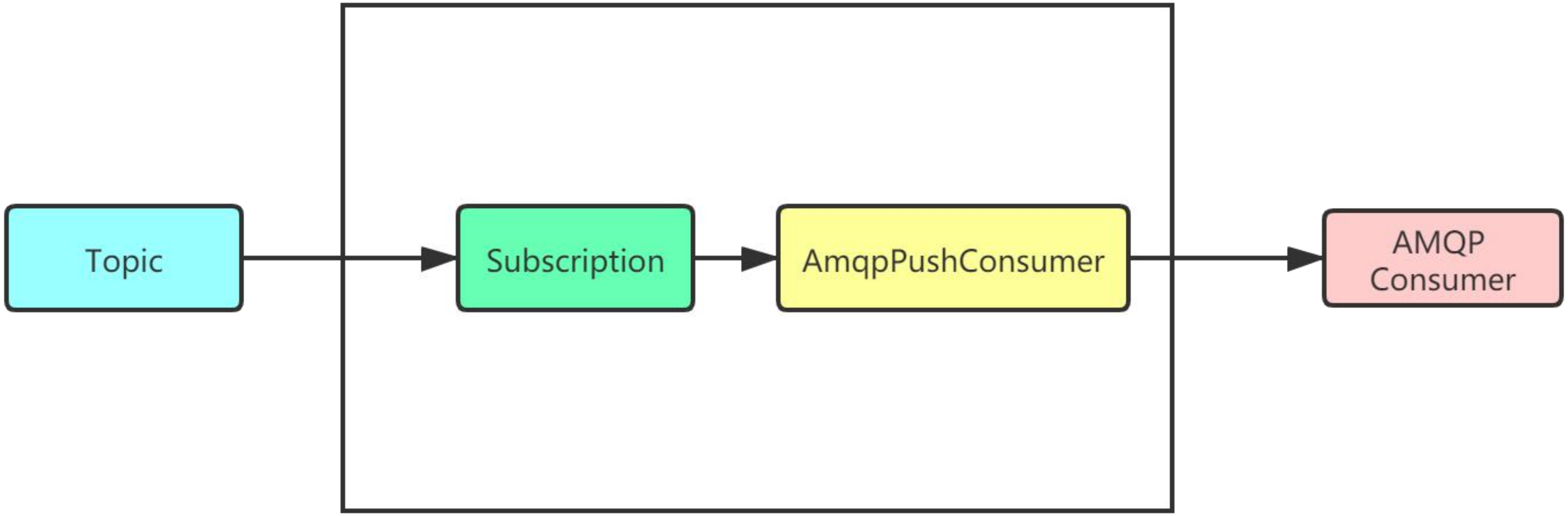


Message Consume

Pulsar consume modle



Message Consume->AMQP Basic.consume



Message Consume->AMQP Basic.consume

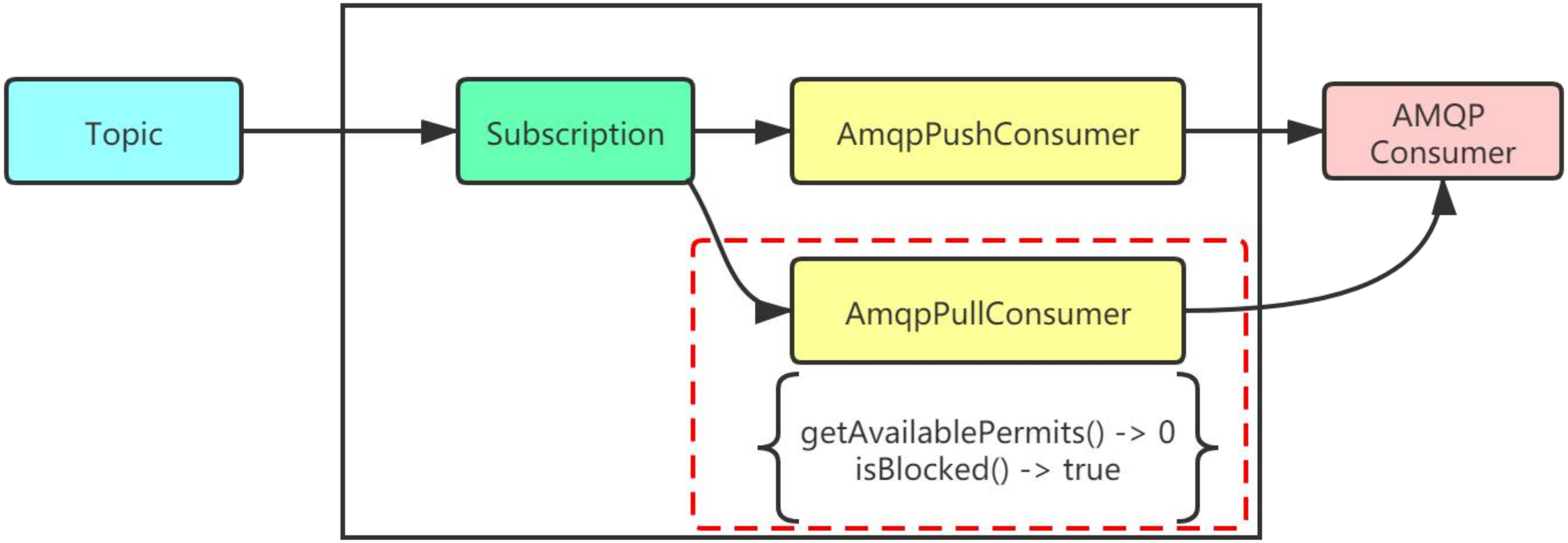
Push实现

- AMQPConsumer <--> Pulsar Consumer
  - 重写sendMessages()方法
    - 消息转换：pulsar message -> AMQP message
    - 将获取的消息推送到AMQP的客户端

- subscription.addConsumer

- consumer.handleFlow

Message Consume->AMQP Basic.get

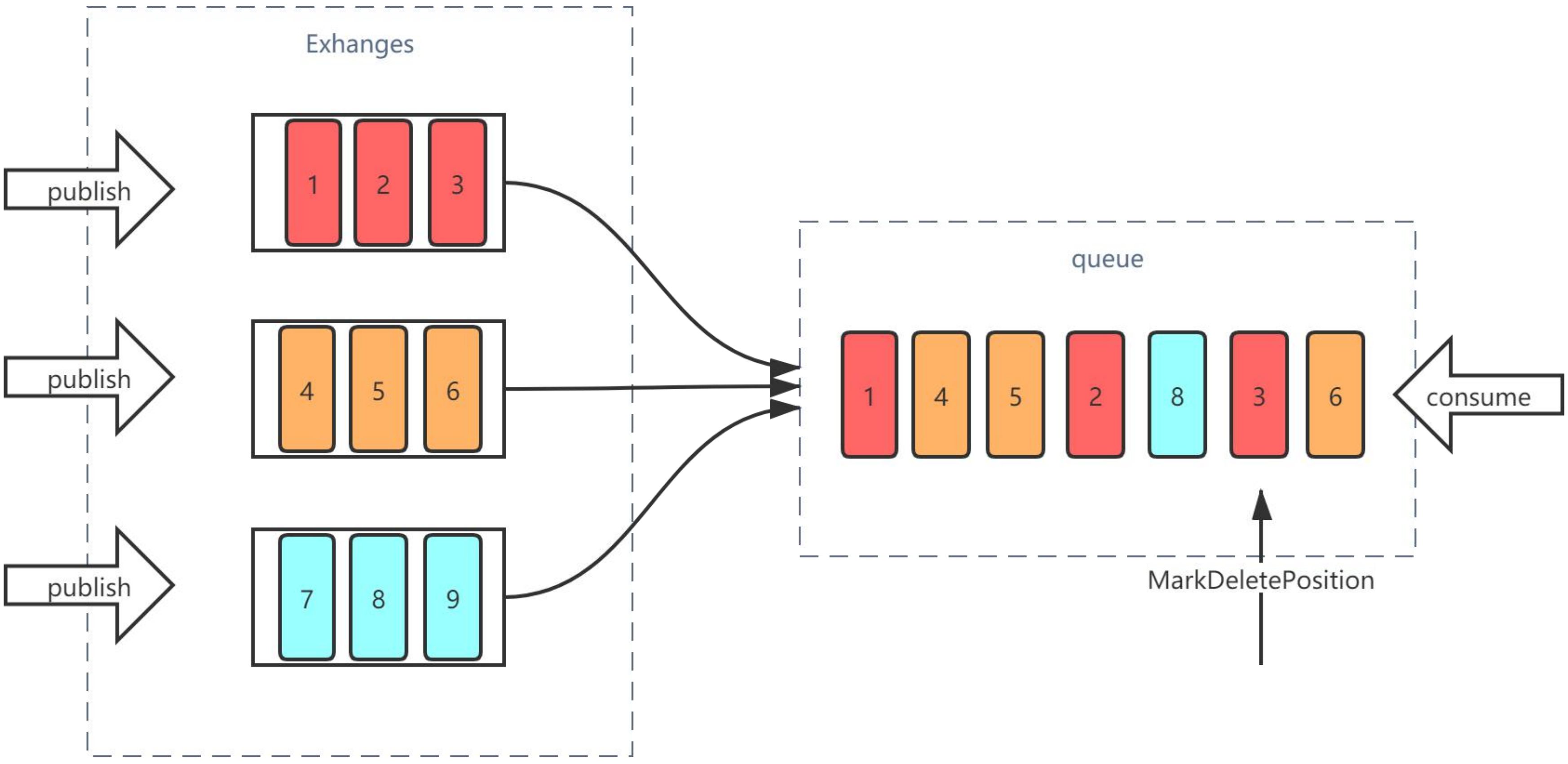


Message Consume->AMQP Basic.get

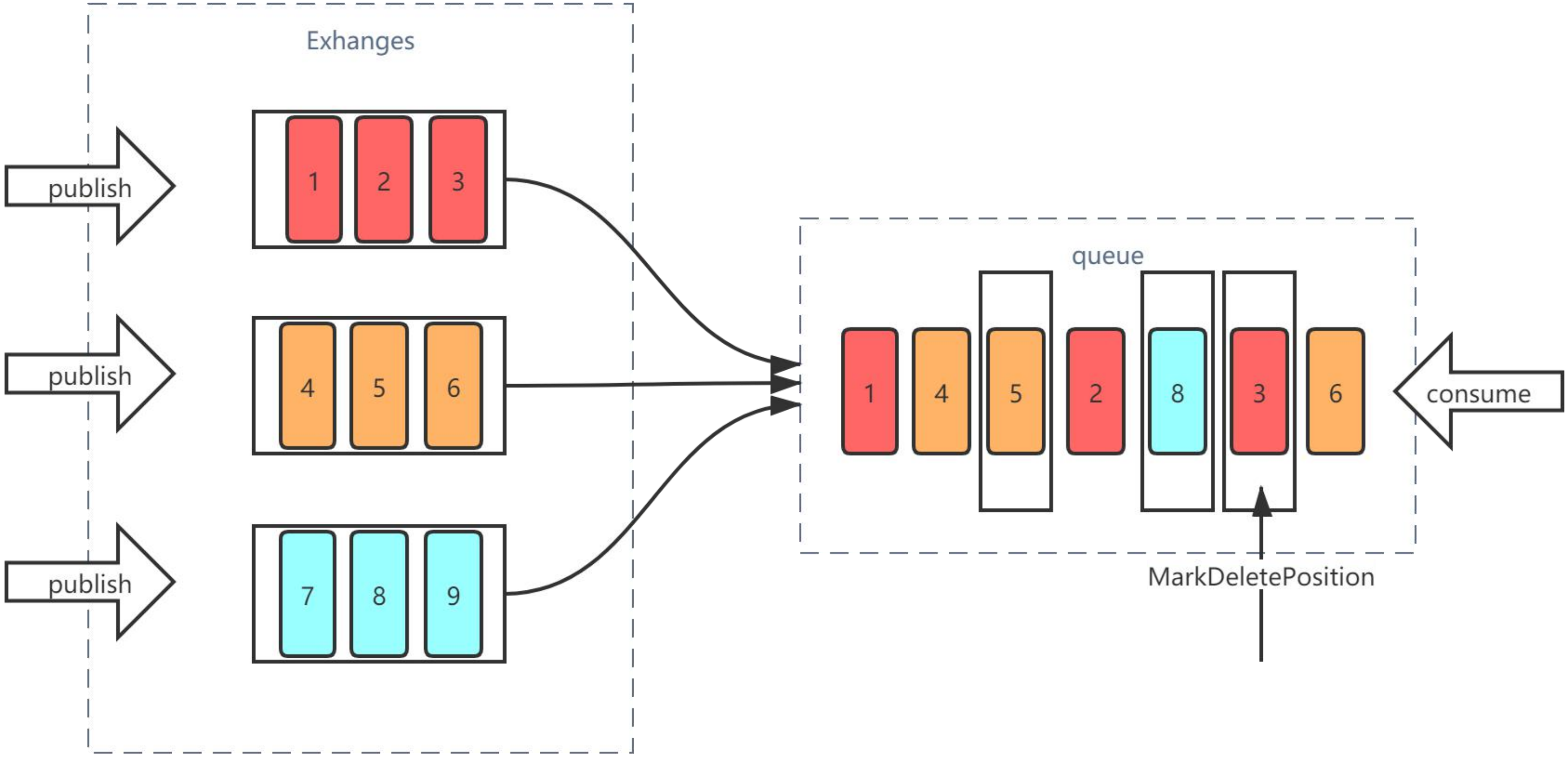
Pull实现

- Pull 消费流程
  - 主动从cursor中读取消息，然后和Push的consumer共享read position
  - 消息签收的方式和Push方式一致

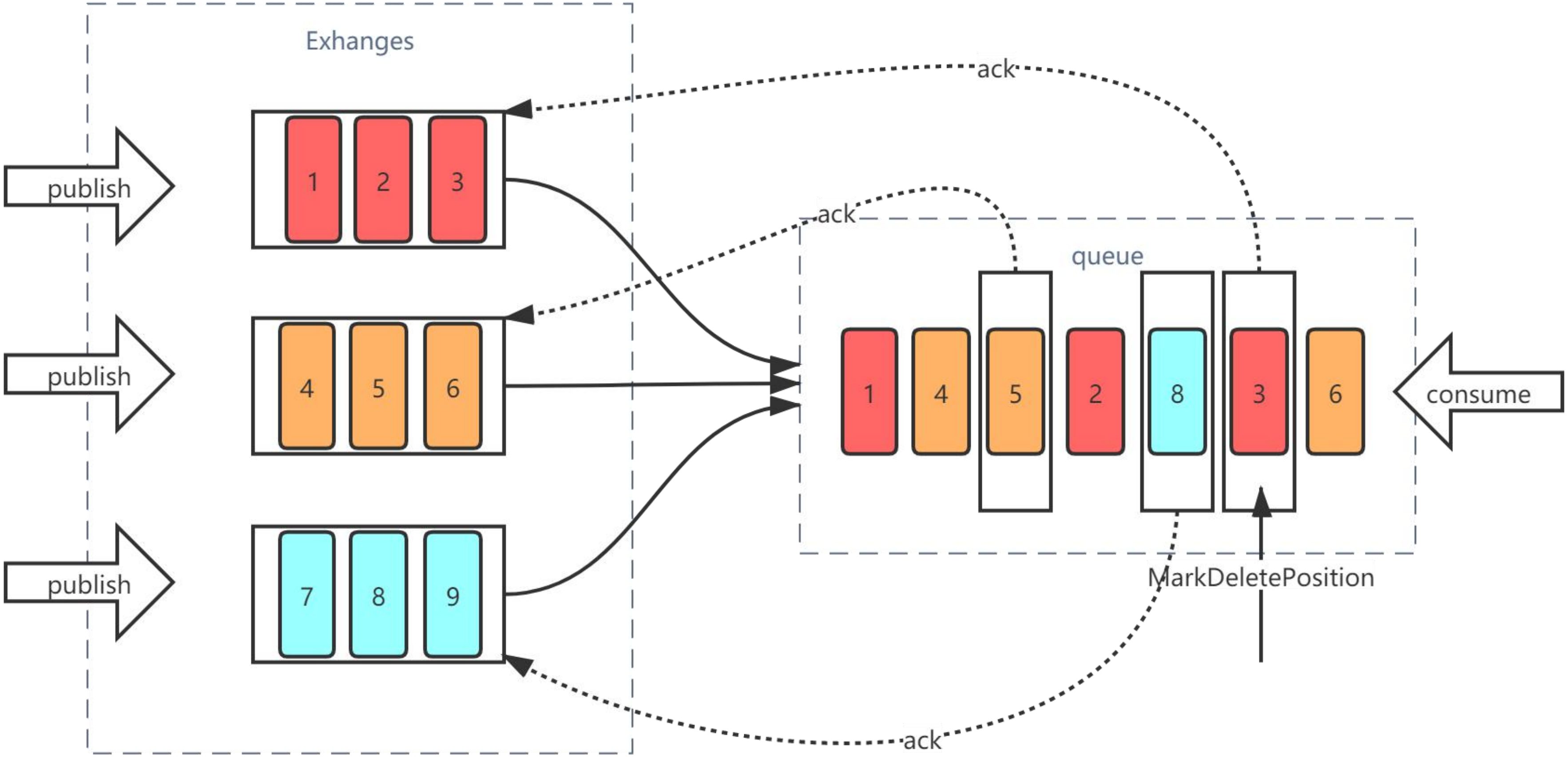
Message ACK


Exhanges
publish
publish
publish
1
2
3
4
5
6
7
8
9
queue
1
4
5
2
8
3
6
consume
MarkDeletePosition

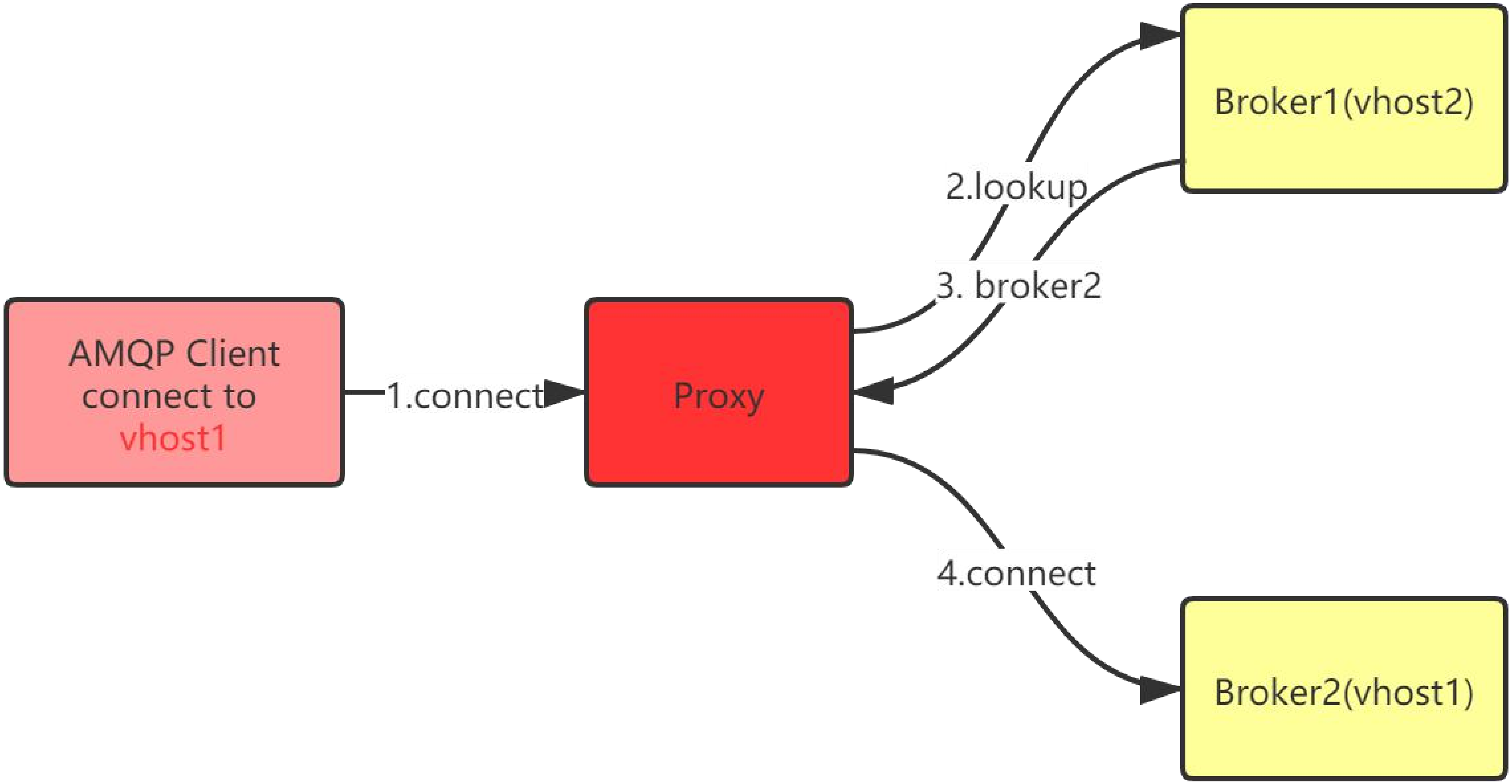
Message ACK

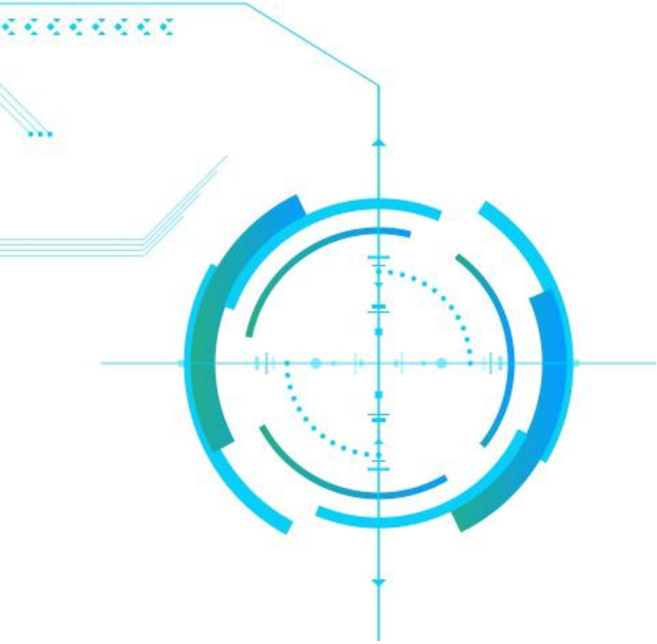

Exhanges
publish
publish
publish
1
2
3
4
5
6
7
8
9
queue
1
4
5
2
8
3
6
consume
MarkDeletePosition

Message ACK


Exhanges
ack
publish
1
2
3
ack
queue
publish
4
5
6
1
4
5
2
8
3
6
consume
publish
7
8
9
MarkDeletePosition
ack

AoP proxy


AMQP Client connect to vhost1
1.connect
Proxy
2.lookup
3. broker2
Broker1(vhost2)
4.connect
Broker2(vhost1)

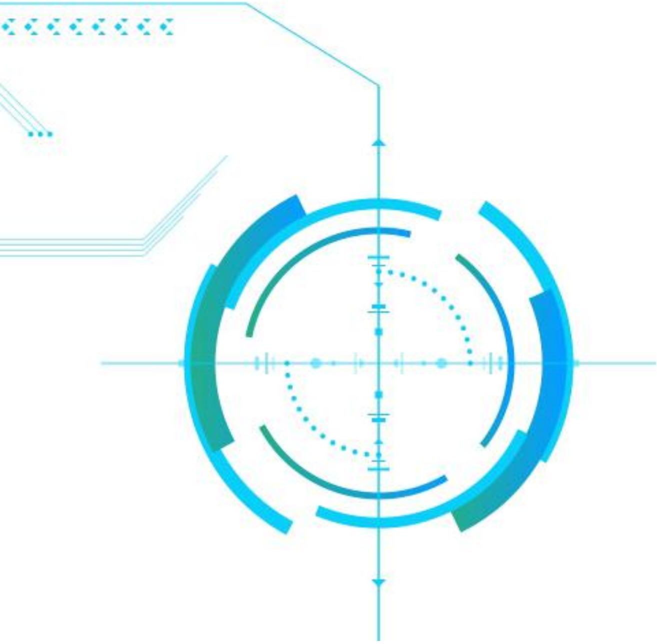


怎么使用AoP?

怎么使用AoP -- Standalone

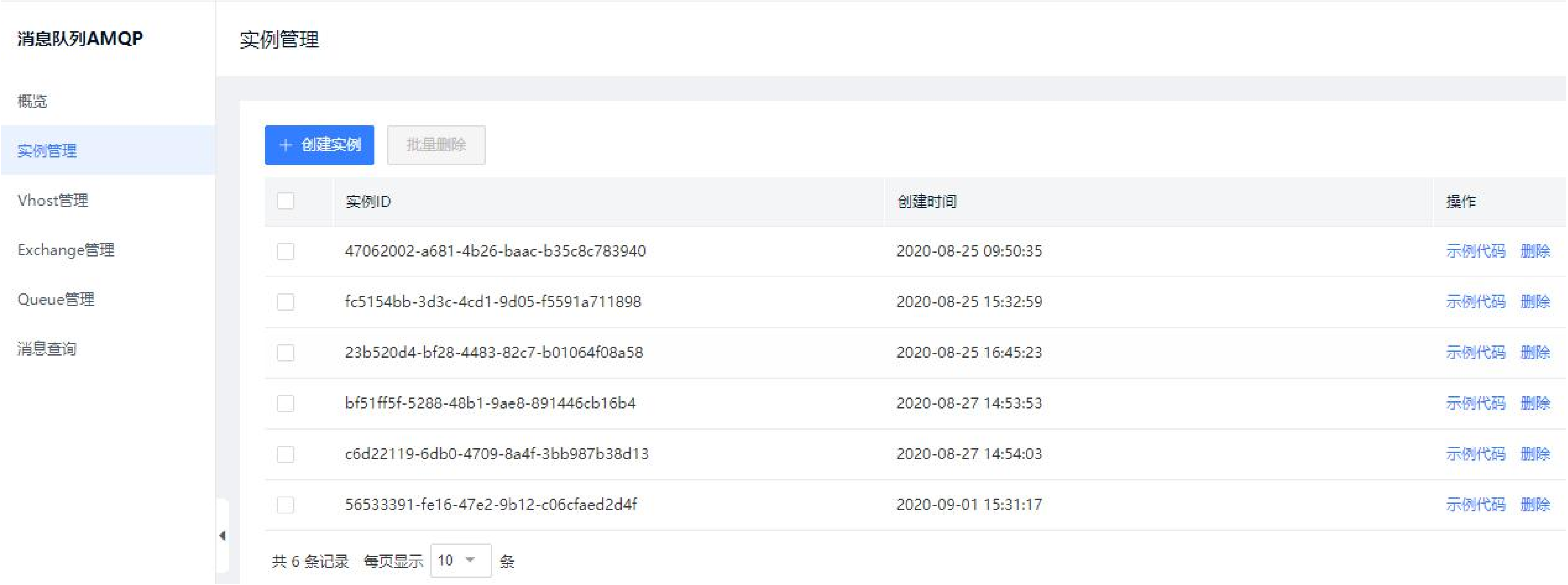
https://github.com/streamnative/aop

- Download or build pulsar-protocol-handler-amqp-${version}.nar and copy to $PULSAR_HOME/protocols directory
- Modify pulsar standalone conf
  - messagingProtocols=amqp
  - protocolHandlerDirectory=./protocols
  - amqpListeners=amqp://127.0.0.1:5672
  - advertisedAddress=127.0.0.1
- Start pulsar use standalone mode
- Add namespace for vhost



怎么使用移动云AMQP

Demo of public cloud

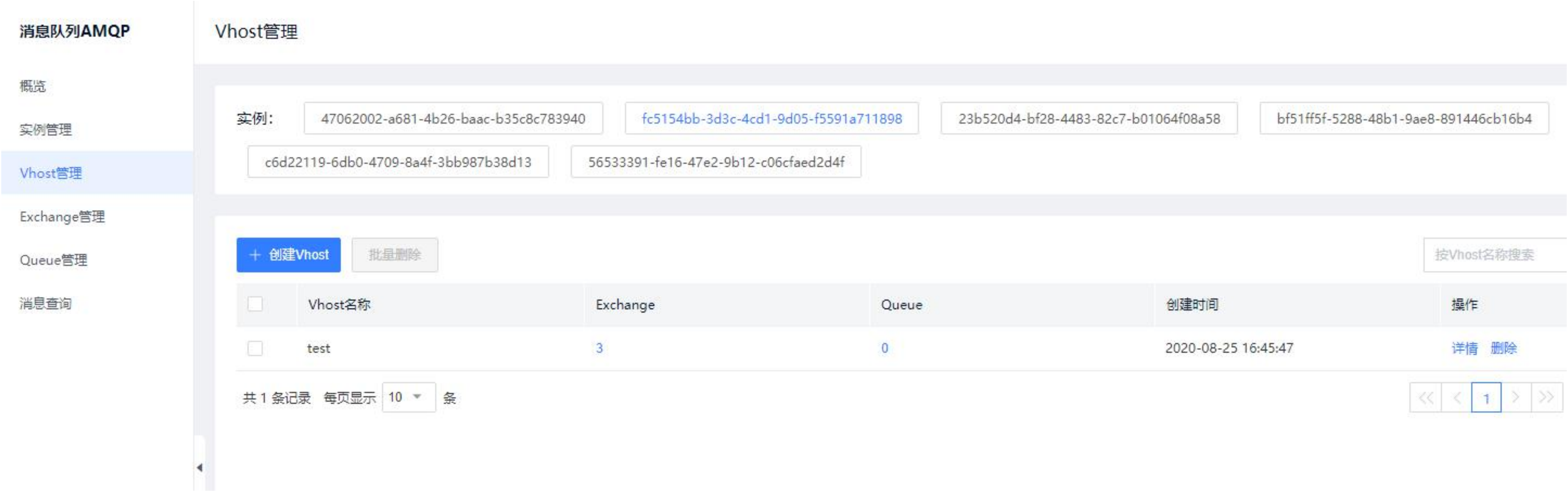


实例管理

□	实例ID	创建时间	操作
□	47062002-a681-4b26-baac-b35c8c783940	2020-08-25 09:50:35	示例代码 删除
□	fc5154bb-3d3c-4cd1-9d05-f5591a711898	2020-08-25 15:32:59	示例代码 删除
□	23b520d4-bf28-4483-82c7-b01064f08a58	2020-08-25 16:45:23	示例代码 删除
□	bf51ff5f-5288-48b1-9ae8-891446cb16b4	2020-08-27 14:53:53	示例代码 删除
□	c6d22119-6db0-4709-8a4f-3bb987b38d13	2020-08-27 14:54:03	示例代码 删除
□	56533391-fe16-47e2-9b12-c06cfaed2d4f	2020-09-01 15:31:17	示例代码 删除

共 6 条记录 每页显示 10 条

Demo of public cloud



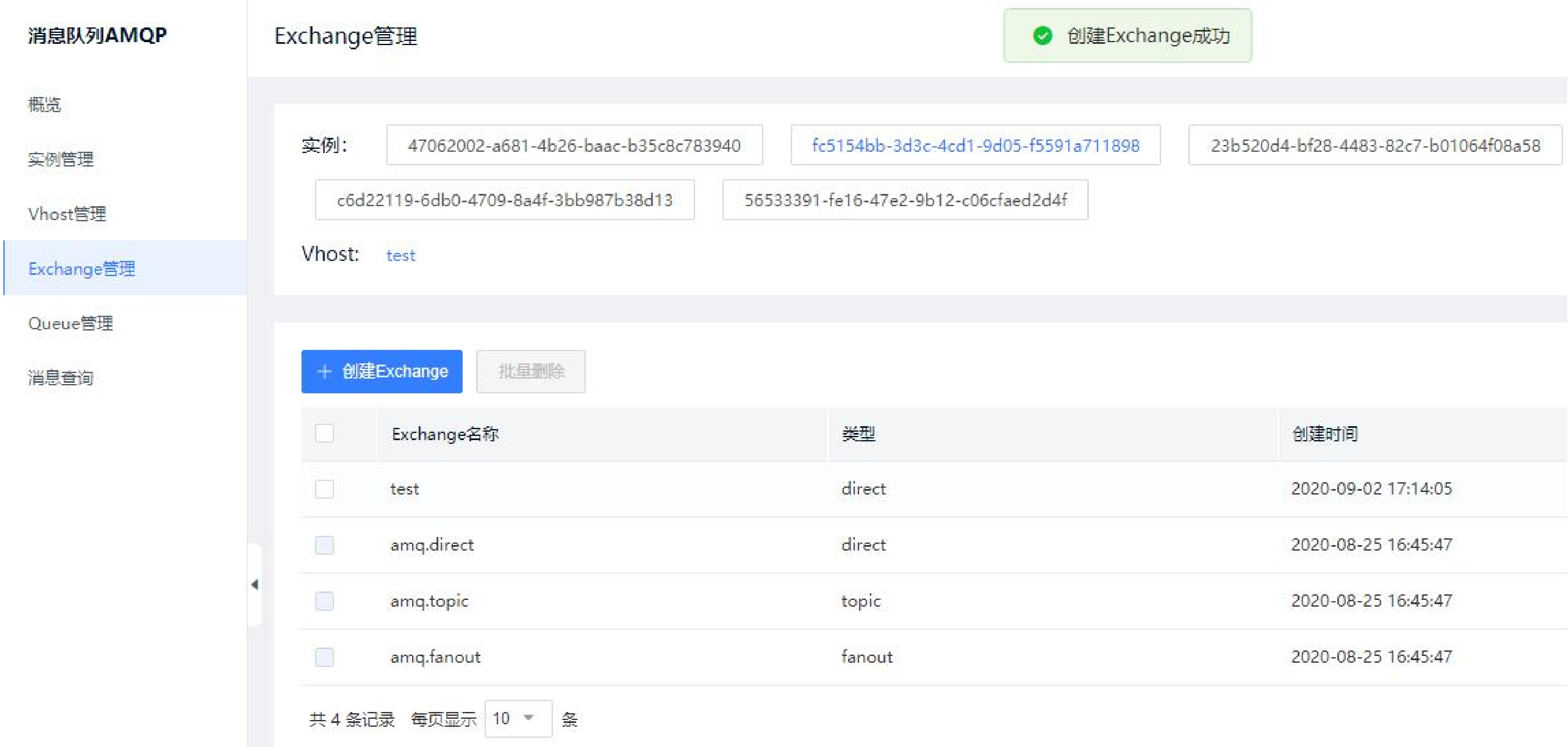
Vhost管理

实例：47062002-a681-4b26-baac-b35c8c783940 fc5154bb-3d3c-4cd1-9d05-f5591a711898 23b520d4-bf28-4483-82c7-b01064f08a58 bf51ff5f-5288-48b1-9ae8-891446cb16b4 c6d22119-6db0-4709-8a4f-3bb987b38d13 56533391-fe16-47e2-9b12-c06cfaed2d4f

□	Vhost名称	Exchange	Queue	创建时间	操作
□	test	3	0	2020-08-25 16:45:47	详情 删除

共 1 条记录 每页显示 10 条

Demo of public cloud



消息队列AMQP

Exchange管理

✓ 创建Exchange成功

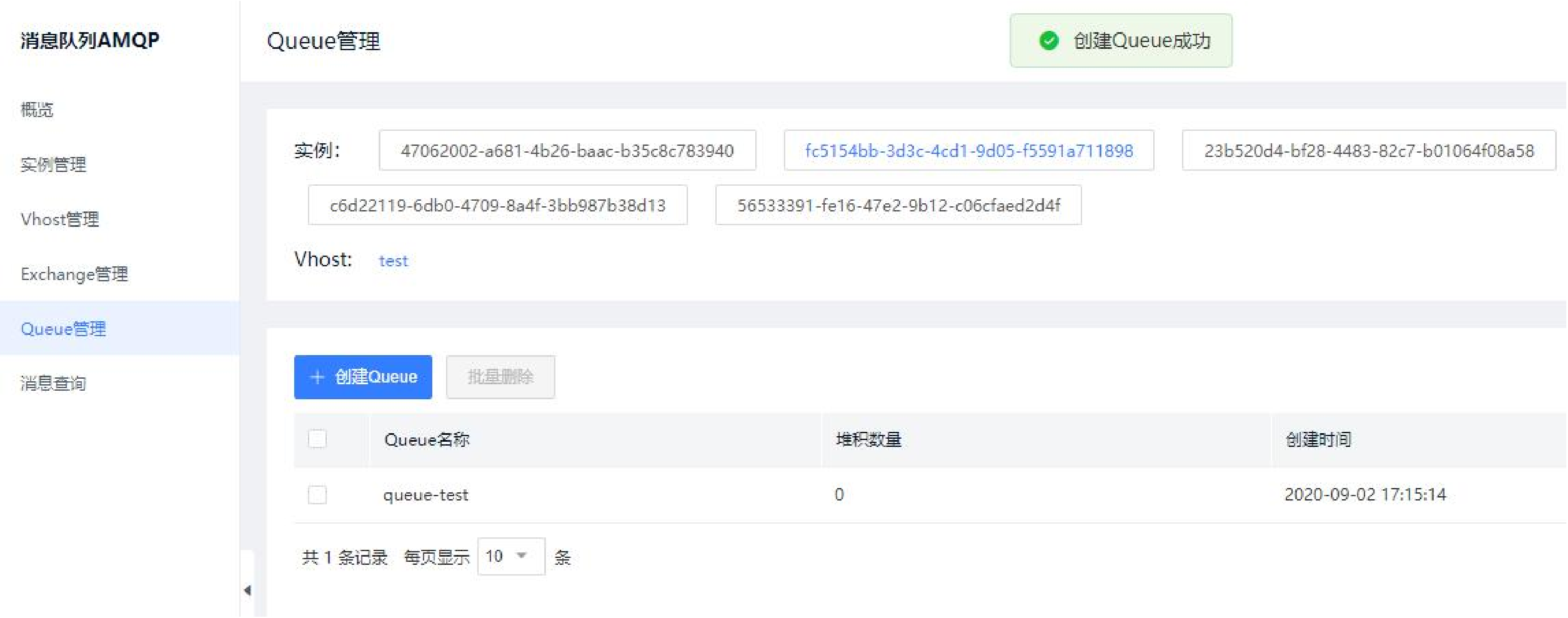
实例： 47062002-a681-4b26-baac-b35c8c783940 | fc5154bb-3d3c-4cd1-9d05-f5591a711898 | 23b520d4-bf28-4483-82c7-b01064f08a58 | c6d22119-6db0-4709-8a4f-3bb987b38d13 | 56533391-fe16-47e2-9b12-c06cfaed2d4f

Vhost: test

□	Exchange名称	类型	创建时间
□	test	direct	2020-09-02 17:14:05
□	amq.direct	direct	2020-08-25 16:45:47
□	amq.topic	topic	2020-08-25 16:45:47
□	amq.fanout	fanout	2020-08-25 16:45:47

共 4 条记录 每页显示 10 条

Demo of public cloud



✓ 创建Queue成功

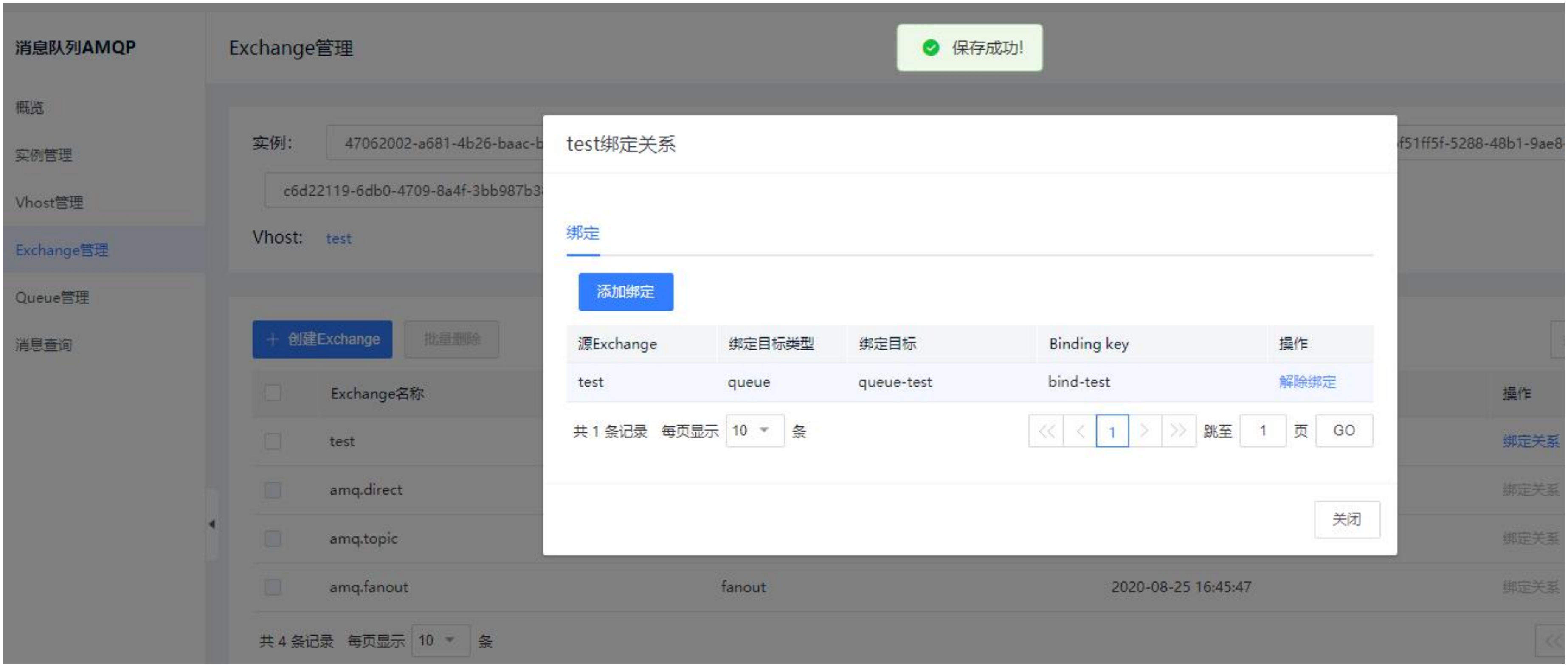
实例：47062002-a681-4b26-baac-b35c8c783940　fc5154bb-3d3c-4cd1-9d05-f5591a711898　23b520d4-bf28-4483-82c7-b01064f08a58　c6d22119-6db0-4709-8a4f-3bb987b38d13　56533391-fe16-47e2-9b12-c06cfaed2d4f

Vhost: test

□	Queue名称	堆积数量	创建时间
□	queue-test	0	2020-09-02 17:15:14

共 1 条记录　每页显示 10 ▾ 条

Demo of public cloud



消息队列AMQP

- 概览
- 实例管理
- Vhost管理
- Exchange管理
- Queue管理
- 消息查询

Exchange管理

保存成功!

实例: 47062002-a681-4b26-baac-b f51ff5f-5288-48b1-9ae8

c6d22119-6db0-4709-8a4f-3bb987b3

Vhost: test

+ 创建Exchange　批量删除

	Exchange名称			操作
□	test			绑定关系
□	amq.direct			绑定关系
□	amq.topic			绑定关系
□	amq.fanout	fanout	2020-08-25 16:45:47	绑定关系

共 4 条记录　每页显示 10 条

test绑定关系

绑定

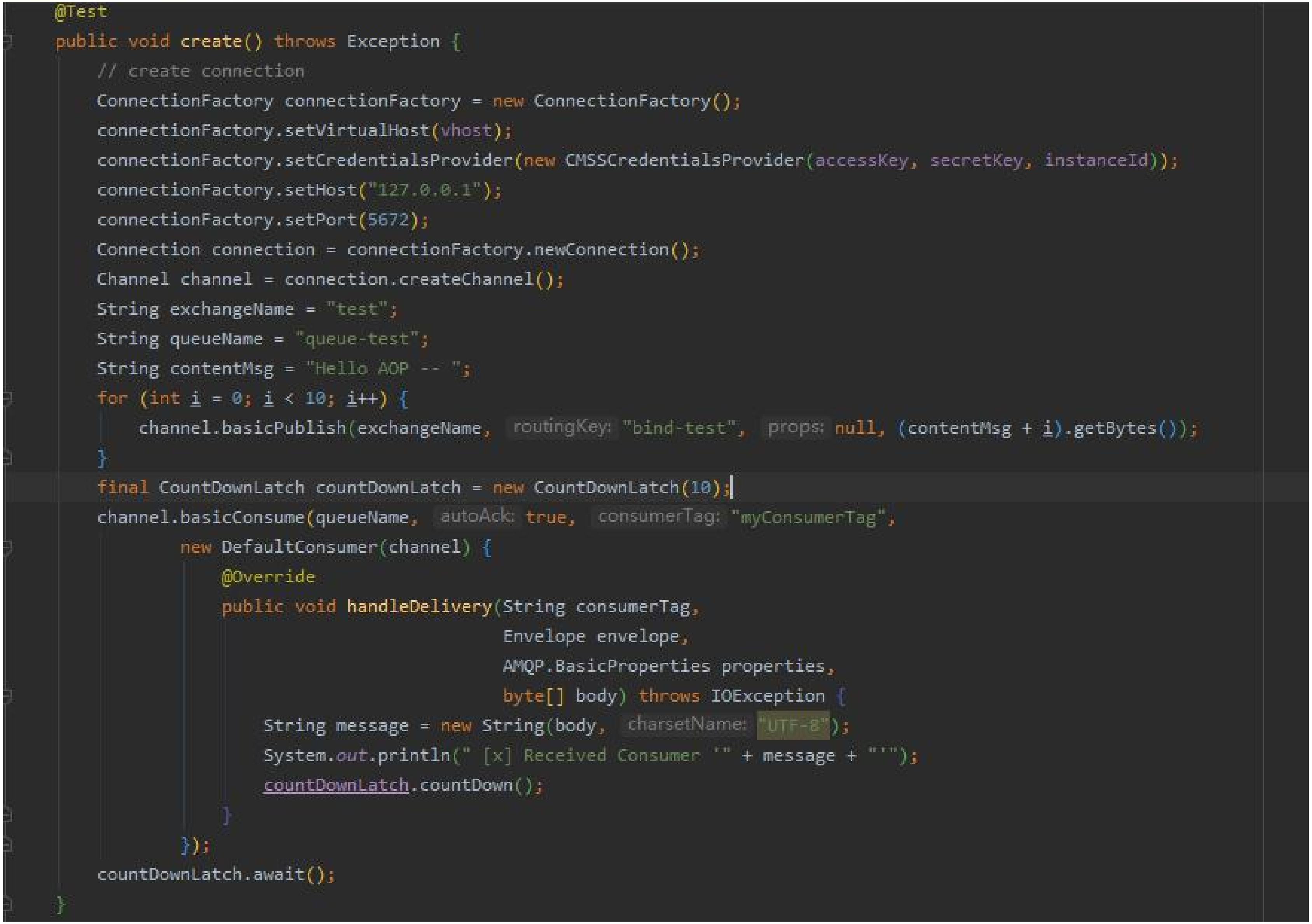
添加绑定

源Exchange	绑定目标类型	绑定目标	Binding key	操作
test	queue	queue-test	bind-test	解除绑定

共 1 条记录　每页显示 10 条　1　跳至 1 页 GO

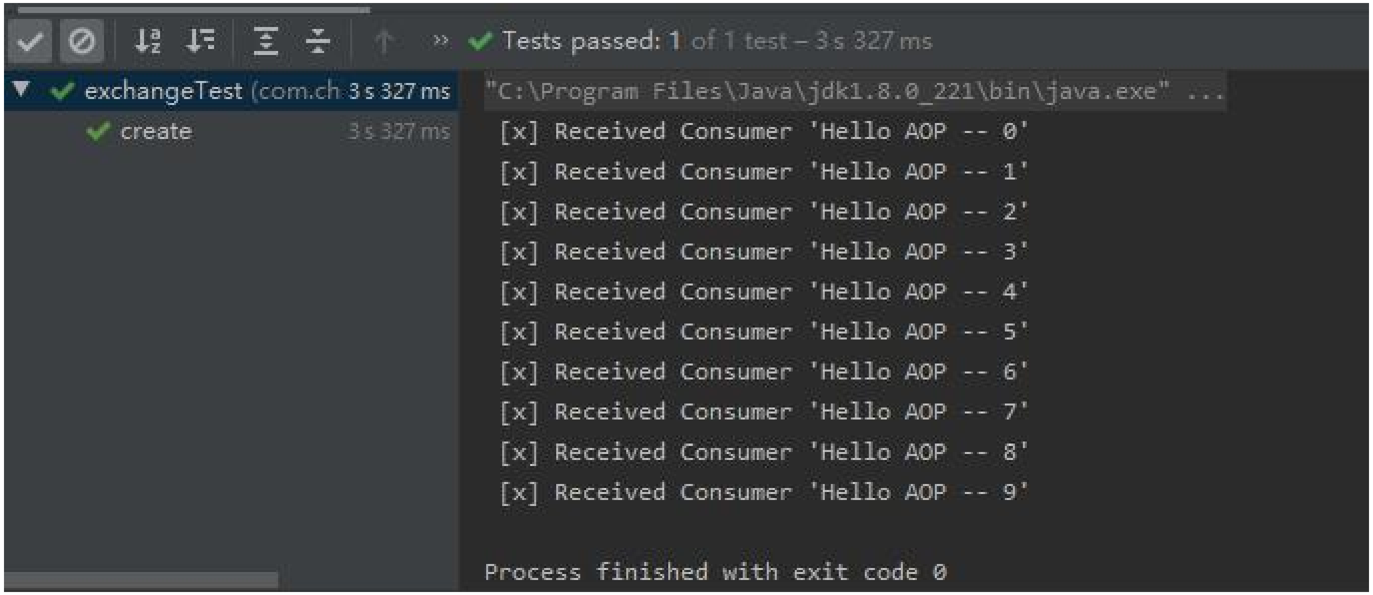
关闭

Demo of public cloud



```
@Test
public void create() throws Exception {
    // create connection
    ConnectionFactory connectionFactory = new ConnectionFactory();
    connectionFactory.setVirtualHost(vhost);
    connectionFactory.setCredentialsProvider(new CMSSCredentialsProvider(accessKey, secretKey, instanceId));
    connectionFactory.setHost("127.0.0.1");
    connectionFactory.setPort(5672);
    Connection connection = connectionFactory.newConnection();
    Channel channel = connection.createChannel();
    String exchangeName = "test";
    String queueName = "queue-test";
    String contentMsg = "Hello AOP -- ";
    for (int i = 0; i < 10; i++) {
        channel.basicPublish(exchangeName, routingKey: "bind-test", props: null, (contentMsg + i).getBytes());
    }
    final CountDownLatch countDownLatch = new CountDownLatch(10);
    channel.basicConsume(queueName, autoAck: true, consumerTag: "myConsumerTag",
            new DefaultConsumer(channel) {
                @Override
                public void handleDelivery(String consumerTag,
                                           Envelope envelope,
                                           AMQP.BasicProperties properties,
                                           byte[] body) throws IOException {
                    String message = new String(body, charsetName: "UTF-8");
                    System.out.println(" [x] Received Consumer '" + message + "'");
                    countDownLatch.countDown();
                }
            });
    countDownLatch.await();
}
```

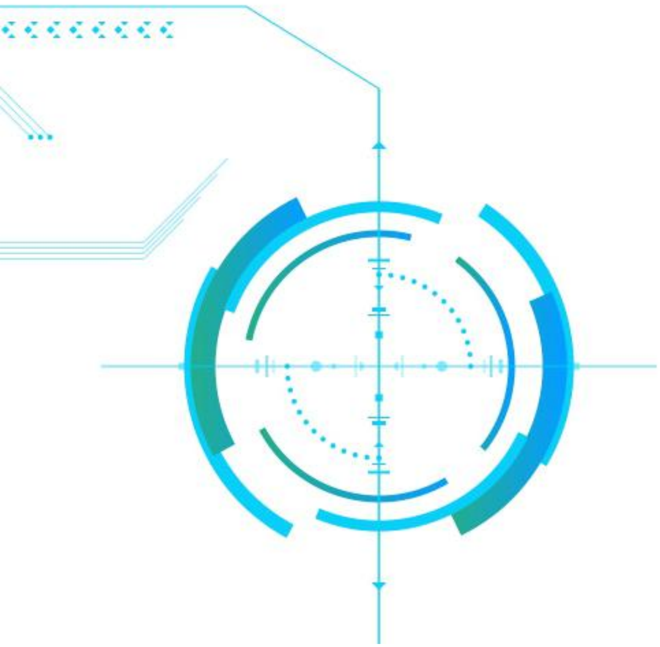
Demo of public cloud



Tests passed: 1 of 1 test – 3 s 327 ms
exchangeTest (com.ch 3 s 327 ms
create 3 s 327 ms

```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
 [x] Received Consumer 'Hello AOP -- 0'
 [x] Received Consumer 'Hello AOP -- 1'
 [x] Received Consumer 'Hello AOP -- 2'
 [x] Received Consumer 'Hello AOP -- 3'
 [x] Received Consumer 'Hello AOP -- 4'
 [x] Received Consumer 'Hello AOP -- 5'
 [x] Received Consumer 'Hello AOP -- 6'
 [x] Received Consumer 'Hello AOP -- 7'
 [x] Received Consumer 'Hello AOP -- 8'
 [x] Received Consumer 'Hello AOP -- 9'

Process finished with exit code 0
```



后续计划

后续计划

- 支持AMQP 1.0协议
- AMQP元数据信息集中化管理
- 丰富AMQP相关监控项
- 支持namespace的多bundle

中国移动
China Mobile

Q&A