ALIBABA CLOUD

阿里云

消息队列 MQ

最佳实践

文档版本：20200924

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	**单击设置> 网络> 设置网络类型。**
粗体	表示按键、菜单、页面名称等UI元素。	**在结果确认页面，单击确定。**
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.Topic 与 Tag 最佳实践 ---- 05
2.消费幂等 ---- 07
3.订阅关系一致 ---- 09
4.在线迁移 RocketMQ ---- 14
5.在线教育视频直播 ---- 16

1.Topic 与 Tag 最佳实践

在

消息队列 RocketMQ 版

中，Topic 与 Tag 都是业务上用来归类的标识，区分在于 Topic 是一级分类，而 Tag 可以理解为是二级分类。您可通过本文了解如何搭配使用 Topic 和 Tag 来实现消息过滤。

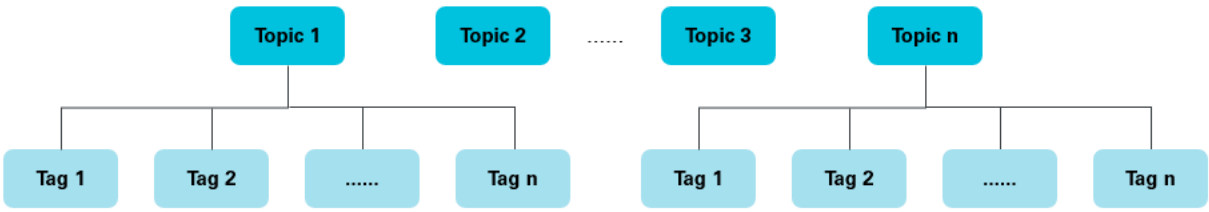
背景信息

Topic 和 Tag 的定义如下：

Topic
消息主题，通过 Topic 对不同的业务消息进行分类。

Tag
消息标签，用来进一步区分某个 Topic 下的消息分类，消息从生产者发出即带上的属性。

Topic 和 Tag 的关系如下图所示。



适用场景

您可能会有这样的疑问：到底什么时候该用 Topic，什么时候该用 Tag?

建议您从以下几个方面进行判断：

- 消息类型是否一致：如普通消息、事务消息、定时（延时）消息、顺序消息，不同的消息类型使用不同的 Topic，无法通过 Tag 进行区分。
- 业务是否相关联：没有直接关联的消息，如淘宝交易消息，京东物流消息使用不同的 Topic 进行区分；而同样是天猫交易消息，电器类订单、女装类订单、化妆品类订单的消息可以用 Tag 进行区分。
- 消息优先级是否一致：如同样是物流消息，盒马必须小时内送达，天猫超市 24 小时内送达，淘宝物流则相对会慢一些，不同优先级的消息用不同的 Topic 进行区分。
- 消息量级是否相当：有些业务消息虽然量小但是实时性要求高，如果跟某些万亿量级的消息使用同一个 Topic，则有可能会因为过长的等待时间而“饿死”，此时需要将不同量级的消息进行拆分，使用不同的 Topic。

总的来说，针对消息分类，您可以选择创建多个 Topic，或者在同一个 Topic 下创建多个 Tag。但通常情况下，不同的 Topic 之间的消息没有必然的联系，而 Tag 则用来区分同一个 Topic 下相互关联的消息，例如全集和子集的关系、流程先后的关系。

场景示例

以天猫交易平台为例，订单消息和支付消息属于不同业务类型的消息，分别创建 Topic_Order 和 Topic_Pay，其中订单消息根据商品品类以不同的 Tag 再进行细分，列如电器类、男装类、女装类、化妆品类等被各个不同的系统所接收。

通过合理的使用 Topic 和 Tag，可以让业务结构清晰，更可以提高效率。

如何通过 Tag 实现消息过滤，请参见消息过滤。

更多信息

订阅关系一致。

2.消费幂等

为了防止消息重复消费导致业务处理异常，

消息队列 RocketMQ 版

的消费者在接收到消息后，有必要根据业务上的唯一 Key 对消息做幂等处理。本文介绍消息幂等的概念、适用场景以及处理方法。

什么是消息幂等

当出现消费者对某条消息重复消费的情况时，重复消费的结果与消费一次的结果是相同的，并且多次消费并未对业务系统产生任何负面影响，那么这整个过程就可实现消息幂等。

例如，在支付场景下，消费者消费扣款消息，对一笔订单执行扣款操作，扣款金额为 100 元。如果因网络不稳定等原因导致扣款消息重复投递，消费者重复消费了该扣款消息，但最终的业务结果是只扣款一次，扣费 100 元，且用户的扣款记录中对应的订单只有一条扣款流水，不会多次扣除费用。那么这次扣款操作是符合要求的，整个消费过程实现了消费幂等。

适用场景

在互联网应用中，尤其在网络不稳定的情况下，

消息队列 RocketMQ 版

的消息有可能会出现重复。如果消息重复会影响您的业务处理，请对消息做幂等处理。

消息重复的场景如下：

- 发送时消息重复

  当一条消息已被成功发送到服务端并完成持久化，此时出现了网络闪断或者客户端宕机，导致服务端对客户端应答失败。 如果此时生产者意识到消息发送失败并尝试再次发送消息，消费者后续会收到两条内容相同并且 Message ID 也相同的消息。

- 投递时消息重复

  消息消费的场景下，消息已投递到消费者并完成业务处理，当客户端给服务端反馈应答的时候网络闪断。为了保证消息至少被消费一次，

  消息队列 RocketMQ 版

  的服务端将在网络恢复后再次尝试投递之前已被处理过的消息，消费者后续会收到两条内容相同并且 Message ID 也相同的消息。

- 负载均衡时消息重复（包括但不限于网络抖动、Broker 重启以及消费者应用重启）

  当

  消息队列 RocketMQ 版

  的 Broker 或客户端重启、扩容或缩容时，会触发 Rebalance，此时消费者可能会收到重复消息。

处理方法

因为 Message ID 有可能出现冲突（重复）的情况，所以真正安全的幂等处理，不建议以 Message ID 作为处理依据。最好的方式是以业务唯一标识作为幂等处理的关键依据，而业务的唯一标识可以通过消息 Key 设置。

以支付场景为例，可以将消息的 Key 设置为订单号，作为幂等处理的依据。具体代码示例如下：

```
Message message = new Message();
message.setKey("ORDERID_100");
SendResult sendResult = producer.send(message);
```

消费者收到消息时可以根据消息的 Key，即订单号来实现消息幂等：

```
consumer.subscribe("ons_test", "*", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        String key = message.getKey()
        // 根据业务唯一标识的 Key 做幂等处理
    }
});
```

3.订阅关系一致

订阅关系一致指的是同一个消费者 Group ID 下所有 Consumer 实例所订阅的 Topic、Group ID、Tag 必须完全一致。一旦订阅关系不一致，消息消费的逻辑就会混乱，甚至导致消息丢失。本文提供订阅关系一致的正确示例代码以及订阅关系不一致的错误示例代码，帮助您顺畅地订阅消息。

背景信息

消息队列 RocketMQ 版

里的一个消费者 Group ID 代表一个 Consumer 实例群组。对于大多数分布式应用来说，一个消费者 Group ID 下通常会挂载多个 Consumer 实例。

由于

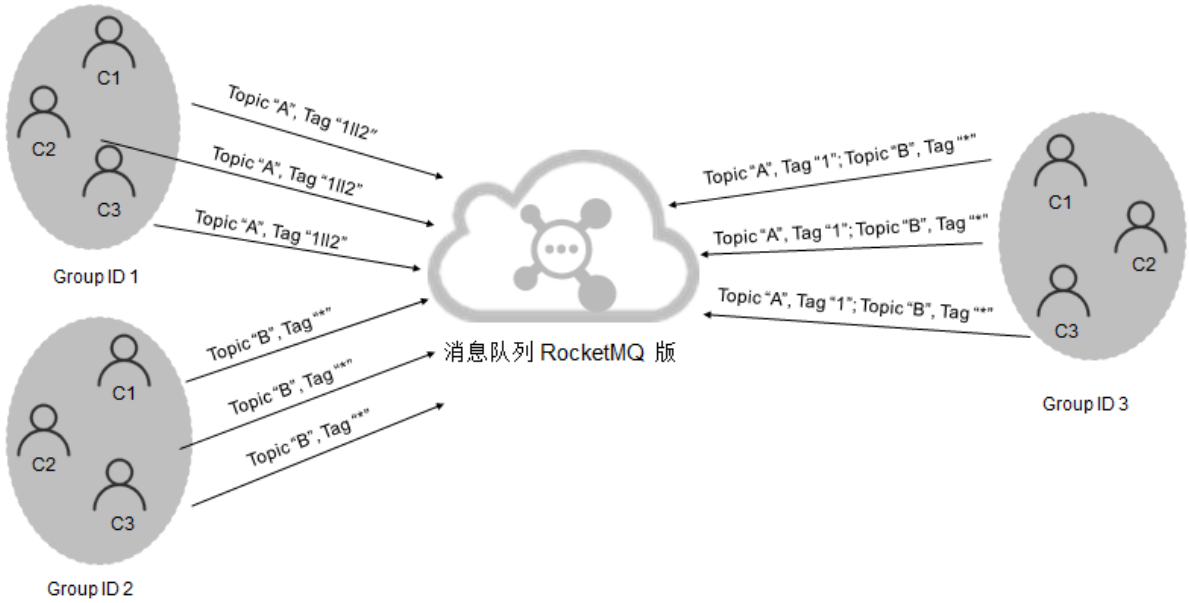
消息队列 RocketMQ 版

的订阅关系主要由 Topic + Tag 共同组成，因此，保持订阅关系一致意味着同一个消费者 Group ID 下所有的实例需在以下两方面均保持一致：

- 订阅的 Topic 必须一致
- 订阅的 Topic 中的 Tag 必须一致

正确订阅关系图片示例

多个 Group ID 订阅了多个 Topic，并且每个 Group ID 里的多个消费者实例的订阅关系保持了一致。



正确订阅关系代码示例

以下例子中，同一个 Group ID 下的实例订阅相同的 Topic 和 Tag。

- Consumer 实例 1-1：

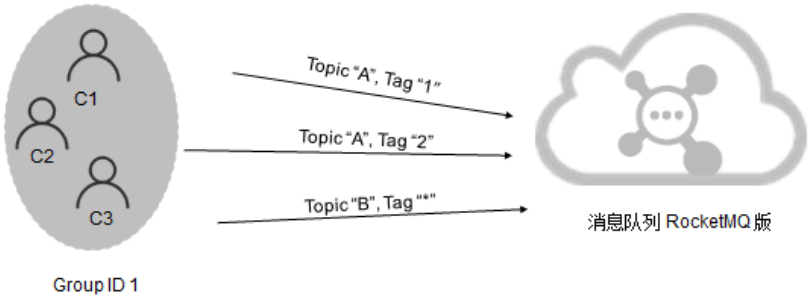
```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID_jodie_test_1");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("jodie_test_A", "Tag1||2", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println(message.getMsgID());
        return Action.CommitMessage;
    }
});
```

- Consumer 实例 1-2:

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID_jodie_test_1");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("jodie_test_A", "Tag1||2", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println(message.getMsgID());
        return Action.CommitMessage;
    }
});
consumer.subscribe("jodie_test_A", "Tag1||2", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println(message.getMsgID());
        return Action.CommitMessage;
    }
});
```

错误订阅关系图片示例

单个 Group ID 订阅了多个 Topic，但是该 Group ID 里的多个消费者实例的订阅关系并没有保持一致。



错误订阅关系代码示例一

以下例子中，同一个 Group ID 下的两个实例订阅的 Topic 不一致。

- Consumer 实例 1-1：

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID_jodie_test_1");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("jodie_test_A", "*", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println(message.getMsgID());
        return Action.CommitMessage;
    }
});
```

- Consumer 实例 1-2：

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID_jodie_test_1");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("jodie_test_B ", "*", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println(message.getMsgID());
        return Action.CommitMessage;
    }
});
```

错误订阅关系代码示例二

以下例子中，同一个 Group ID 下订阅 Topic 的 Tag 不一致。Consumer 实例 2-1 订阅了 TagA，而 Consumer 实例 2-2 未指定 Tag。

- Consumer 实例 2-1：

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID_jodie_test_2");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("jodie_test_A", "TagA", new MessageListener() {
  public Action consume(Message message, ConsumeContext context) {
    System.out.println(message.getMsgID());
    return Action.CommitMessage;
  }
});
```

- Consumer 实例 2-2：

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID_jodie_test_2");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("jodie_test_A", "*", new MessageListener() {
  public Action consume(Message message, ConsumeContext context) {
    System.out.println(message.getMsgID());
    return Action.CommitMessage;
  }
});
```

错误订阅关系代码示例三

此例中，同一个 Group ID 下订阅 Topic 个数不一致，且订阅的 Topic 的 Tag 不一致。

- Consumer 实例 3-1：

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID_jodie_test_3");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("jodie_test_A", "TagA", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println(message.getMsgID());
        return Action.CommitMessage;
    }
});
consumer.subscribe("jodie_test_B", "TagB", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println(message.getMsgID());
        return Action.CommitMessage;
    }
});
```

- Consumer 实例 3-2:

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID_jodie_test_3");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("jodie_test_A", "TagB", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println(message.getMsgID());
        return Action.CommitMessage;
    }
});
```

4.在线迁移 RocketMQ

您可将自建 RocketMQ 应用迁移到阿里云消息队列 RocketMQ 版。本文提供迁移方案的最佳实践的引导。

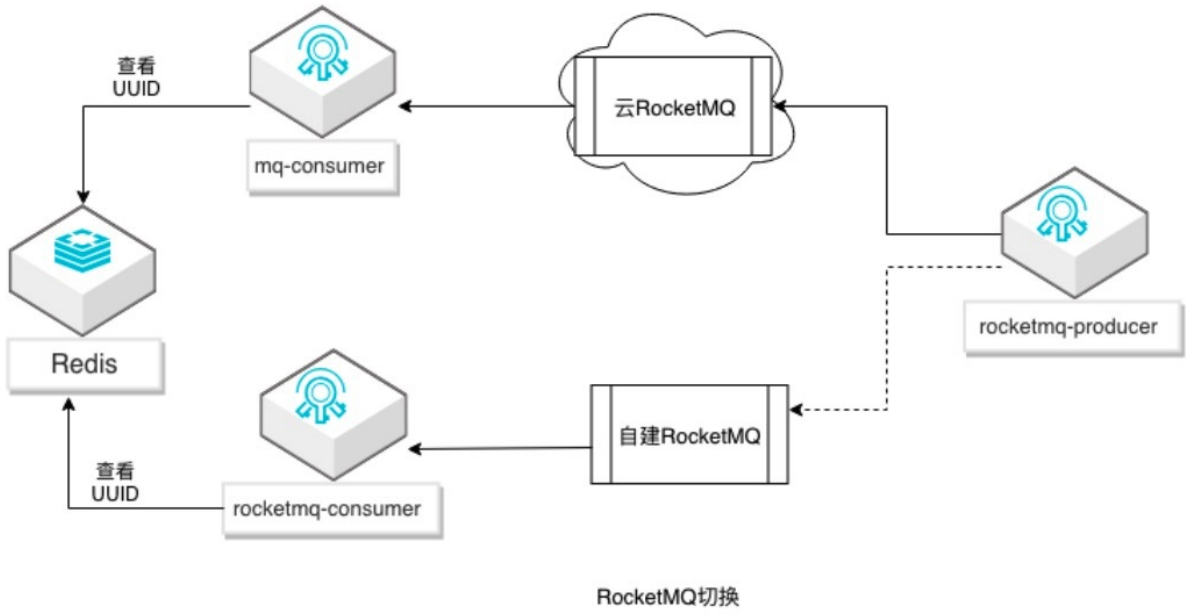
场景描述

适用于企业自建 RocketMQ 应用迁移到阿里云消息队列 RocketMQ 版，迁移后将能利用云 RocketMQ 高可用、高弹性、低延时的能力的场景。

解决的问题

- 自建 RocketMQ 消息集群迁移到云 RocketMQ。
- 迁移中实现应用不停机的切换过程。

部署架构图



RocketMQ切换

选用的产品

- **消息队列 RocketMQ 版**

  消息队列 RocketMQ 版是由阿里巴巴自研并捐赠给 Apache 基金会，并与开源社区共建的消息中间件，该产品服务于阿里巴巴集团已超过13 年，经过交易核心链路反复打磨与历年双十一高并发场景的严苛考验，是一个真正具备低延迟、高并发、高可用、高可靠，可支撑万亿级数据洪峰的分布式消息中间件。

  更多关于消息队列 RocketMQ 版的介绍，参见消息队列 RocketMQ 版产品详情页。

- **云数据库 Redis 版（可选）**

  阿里云数据库Redis版是兼容开源Redis协议标准、提供内存加硬盘混合存储的数据库服务，基于高可靠双机热备架构及可平滑扩展的集群架构，可充分满足高吞吐、低延迟及弹性变配的业务需求。

  更多关于云数据库 Redis 版的介绍，参见云数据库 Redis 版产品详情页。

详细信息

详情请参见自建 RocketMQ 迁移云 RocketMQ 最佳实践。

更多最佳实践

如需查看更多阿里云产品的最佳实践，请参见阿里云最佳实践。

5.在线教育视频直播

本文提供在线教育视频直播场景的最佳实践引导。

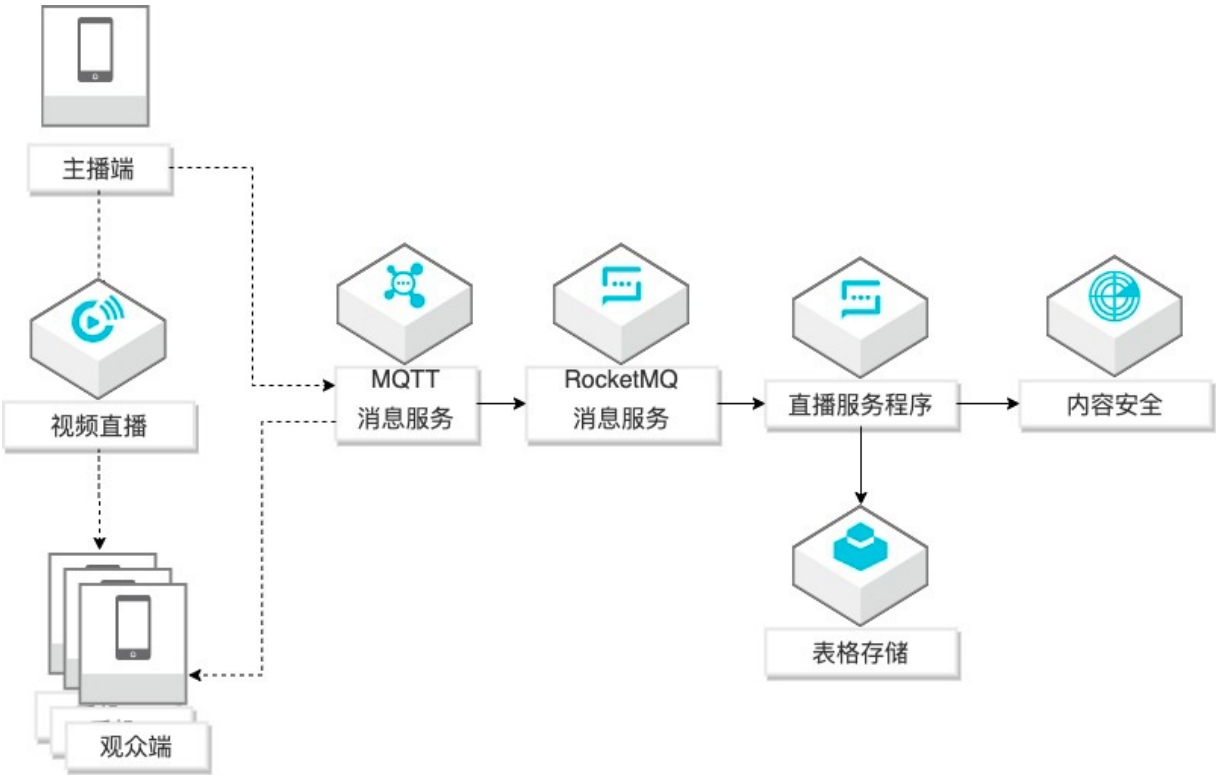
场景描述

适用于在线教育的视频直播场景，通过阿里云视频直播服务实现移动端的推拉流能力，同时移动端利用微消息队列 MQTT 版实现发送和接收消息，在消息转发时利用内容安全服务对消息内容进行审核。发送的消息可以存储在表格存储中。

解决的问题

- 解决视频直播中手机端的推拉流问题。
- 解决直播中手机端互发消息和消息内容的安全审核问题，同时低成本保存消息。

部署架构图



选用的产品

- **消息队列 RocketMQ 版**

  消息队列 RocketMQ 版是由阿里巴巴自研并捐赠给 Apache 基金会，并与开源社区共建的消息中间件，该产品服务于阿里巴巴集团已超过13 年，经过交易核心链路反复打磨与历年双十一高并发场景的严苛考验，是一个真正具备低延迟、高并发、高可用、高可靠，可支撑万亿级数据洪峰的分布式消息中间件。

  更多关于消息队列 RocketMQ 版的介绍，参见消息队列 RocketMQ 版产品详情页。

- **微消息队列 MQTT 版**

微消息队列 MQTT 版是专为移动互联网、物联网领域设计的消息产品，覆盖互动直播、金融支付、智能餐饮、即时聊天、移动 Apps、智能设备、车联网等多种应用场景；通过对 MQTT、WebSocket 等协议的全面支持，连接端和云之间的双向通信，实现 C2C、C2B、B2C 等业务场景之间的消息通信，可支撑千万级设备与消息并发，真正做到万物互联。

更多关于微消息队列 MQTT 版的介绍，参见微消息队列 MQTT 版产品详情页。

- **内容安全**

  内容安全基于深度学习技术， 提供图片、视频、语音、文字等多媒体的内容风险智能识别服务，不仅能帮助用户降低色情、暴恐、涉政等违规风险，而且能大幅度降低人工审核成本。

- **表格存储（Tablestore）**

  表格存储（Tablestore）是阿里云自研的面向海量结构化数据存储的 Serverless NoSQL 多模型数据库，被广泛用于社交、物联网、人工智能、元数据和大数据等业务场景。提供兼容 HBase 的 WideColumn 模型、消息模型 Timeline 以及时空模型 Timestream，可提供PB级存储、千万 TPS 以及毫秒级延迟的服务能力。

  更多关于表格存储的介绍，参见表格存储产品详情页。

- **视频直播**

  视频直播（ApsaraVideo Live）是基于领先的内容接入与分发网络和大规模分布式实时视频处理技术（含窄带高清 TM）打造的音视频直播平台，提供易接入、低延迟、高并发、高清流畅的音视频直播服务。

  更多关于视频直播的介绍，参见视频直播产品详情页。

详细信息

详情请参见在线教育视频直播最佳实践。

更多最佳实践

如需查看更多阿里云产品的最佳实践，请参见阿里云最佳实践。