

**课程读书报告**

**论文题目：亚马逊简单队列服务的研究与应用**

**专 业：软件工程**

**班 级：1621815**

**姓 名：雷舒静**

**学 号：201620181514**

**教 师：邹云康**

目录

摘要....................................................................3

1. 绪论.............................................................4

1.1 Amazon SQS的基础概念................................................4

1. Amazon SQS详细介绍..............................................4

2.1 Amazon SQS的功能...................................................4

2.2主要组成部分........................................................5

2.3 使用前提............................................................6

1. Amazon SQS的应用................................................7

3.1 下载与安装.........................................................7

3.2 使用注意...........................................................8

3.3.1 区域.............................................................8

3.3.2 可见性超时值.....................................................8

3.3.3 轮询间隔.........................................................8

3.3.4 队列的前缀.......................................................9

3.3.5 结果.............................................................9

3.4 具体应用...........................................................9

3.4.1 如何创建和配置队列...............................................9

3.4.2 创建一个对象来连接到Amazon SQS..................................14

3.4.3 使用Amazon SQS实现与AutoCAD远程交互............................17

1. 结论..........................................................27

4.1 优缺点............................................................27

4.1.1 优势............................................................27

4.1.2 限制............................................................27

4.2 未来发展..........................................................28

参考文献...............................................................28

摘 要

Amazon Simple Queue service(SQS)是一种用于分布式应用的组件之间数据传递的消息队列服务，这些组件可能分布在不同的计算机上，甚至是不同的网络申。利用SQS能够将分布式应用的各个组件以松耦合的方式结合起来，从而创建可靠的大规模的分布式系统。组合的组件之间相对独立性强，系统中任何一个组件的失效都不会影响整个系统的运行。

消息和队列是SQS实现的核心。消息是可以存储到SQS队列中的文本数据，可以由应用通过SQS的公共访问接口执行添加、读取、删除操作。队列是消息的容器，提供了消息传递及访问控制的配置选项。SQS是一种支持并发访问的消息队列服务，它支持多个组件并发的操作队列，如向同一个队列发送或者读取消息。消息一且被某个组件处理，则该消息将被锁定，并且被隐藏，其他组件不能访问和操作此消息，此时队列中的其他消息仍然可以被各个组件访问。

SQS采用分布式构架实现，每一条消息都可能保存在不同的机器中，甚至保存衣不同的数据中心里。这种分布式存储策略保证了系统的可靠性，同时也体现出其与中央管理队列的差异，这些差异需要分布式系统设计者和SQS使用者充分理解。首先，SQS并不严格保证消息的顺序，先送人队列的消息也可能晚些时候才会可见;其次，分布式队列中有些已经被处理的消息，在一定时间内还存在于其他队列中，因此同一个消息可能会被处理多次;再次，获取消息时不能确保得到所有的消息，可能只得到部分服务器中队列理的消息;最后，消息的传递可能有延迟，不能期望发出的消息马上被其他组件看到。

借助 SQS，能消除管理开销，可以有效地分离应用程序组件，以让其独立运行，在发生故障时不影响其他组件，从而提高系统的总体容错能力。能够加密每个消息正文，以在应用程序之间交换敏感数据。由此看来，亚马逊简单队列服务在技术上成给了我们很大的便利。

关键词（3-5）

消息； 队列； 数据传递； 云服务

第1章 绪论

1.1 Amazon SQS的基础概念

Amazon Simple Queue Service (SQS) 是一种完全托管的消息队列服务，可让您分离和扩展微服务、分布式系统和无服务器应用程序。SQS 消除了与管理和运营消息型中间件相关的复杂性和开销，并使开发人员能够专注于重要工作。借助 SQS，您可以在软件组件之间发送、存储和接收任何规模的消息，而不会丢失消息，并且无需其他服务即可保持可用。使用 AWS 控制台、命令行界面或您选择的 SDK 和三个简单的命令，在几分钟内即可开始使用 SQS。

SQS 提供两种消息队列类型。标准队列提供最高吞吐量、最大努力排序和至少一次传送。SQS FIFO 队列旨在确保按照消息的发送顺序对消息进行严格一次处理。简单说，它是独立的消息队列服务。它有以下几个特点：

1. 在任何区域中创建无限数量的 Amazon SQS 队列，每个队列负责各自消息，互不干扰，并且消息数量不受限制。

2. 纯消息堆栈，针对同一个设备/接口进行请求时，对堆栈内消息进行轮循返回，每次只返回其中一条。

3. 每条数据每次轮循时都会标记轮循数量+1。

4. 可以对轮循信息进行各种操作，例如：读取、删除等。

第2章 Amazon SQS详细介绍

2.1 Amazon SQS的功能

要想构建一个灵活且可扩展的系统，低耦合度是很有必要的。因为只有系统各个组件之间的关联度尽可能低，才可以根据系统需要随时从系统中增加或者删除某些组件。但松散的耦合度也带来了组件之间的通信问题，如何实现安全、高效地通信是设计一个低耦合度的分布式系统所必须考虑的问题。简单队列服务（Simple Queue Service, SQS)是Amazon为了解决其云计算平台之间不同组件的通信而专门设计开发的。它有以下功能功能：

1. 无限的队列和消息

在任何区域中创建无限数量的 Amazon SQS 队列，并且消息数量不受限制。

1. 负载大小

消息负载可包含最多 256 KB 的任意格式的文本。每 64 KB“区块”的已发布数据以 1 次请求计费。例如，1 次负载为 256 KB 的 API 调用将以 4 次请求计费。要发送大小超过 256KB 的消息，您可以使用[适用于 Java的 Amazon SQS 扩展客户端库](https://github.com/awslabs/amazon-sqs-java-extended-client-lib)，它采用 Amazon S3 来存储消息负载。可使用 SQS 发送消息负载的引用。

1. 批处理

批量发送、接收或删除消息，每批最多 10 条消息或 256 KB。批消息与单条消息的消耗量相同，这意味着对于使用批消息的客户而言，SQS 更具成本效益。

1. [长轮询](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-long-polling.html)在尽可能快地接收新消息的时候减少外部轮询，从而实现最低成本。在您的队列为空时，长轮询请求会为下一条消息等待至多 20 秒。长轮询请求和普通请求的消耗量相同。
2. 可将消息在队列中最多保留 14 天。同时发送和读取消息。
3. 消息锁定

收到消息后，在处理期间它会变为“锁定”状态。这可防止其他计算机同时处理该消息。如果消息处理失败，锁定会过期，而消息也重新变为可用。

1. 队列共享

匿名或使用特定 AWS 账户安全共享 Amazon SQS 队列。队列共享还可按 IP 地址和一天中的时刻进行限制。

1. 服务器端加密 (SSE)

使用由 AWS Key Management Service (AWS KMS) 托管的密钥来保护 Amazon SQS 队列中的消息内容。只要 Amazon SQS 收到消息，SSE 就会对其进行加密。这些消息以加密的形式进行存储，且仅当它们发送到授权的客户时，Amazon SQS 才会对其进行解密。

1. 死信队列 (DLQ)

处理具有死信队列的用户未成功处理的消息。当超出消息的最大接收数时，消息将会移动到与原队列相关的 DLQ 中。为 DLQ 设置单独的用户进程，以便帮助分析和理解消息卡住的原因。DLQ 的类型必须与来源队列 (标准队列或 FIFO 队列) 相同。

2.2 主要组成部分

SQS由三个基本部分组成：系统组件 (Component)、队列(Queue)和消息(Message)。系统组件是SQS的服务对象，而SQS则是组件之间沟通的桥梁。组件在这里有双重角色，它既可以是消息的发送者，也可以是消息的接收者。组件、 队列和消息可以形象地比喻为储户、银行和储户账户中的资金。储户随时可以向银行中自己的账户存钱；同时，储户述可以接受别人给他的汇款或给别人汇款；当有需要时，用户可以从银行中取出自己账内中的钱；不需要时，账户中的资金会很安全壤焊存在银行中。SQS也是如此，组件既发送消息也接收消息，不接收时消息会被安全地存放在队列中。  
 消息和队列是SQS中最重要的两个概念。消息是发送者创建的具有一定格式的文本数据，接收对象可以是一个或多个组件。消息的大小是有限制的，目前Amazon规定每条消息不得超过8KB,但是消息的数量并未做限制。队列是存放消息的容器，类似于S3中的桶，队列的数目也是任意的，创建队列时用户必须给其指定一个在SQS账户内唯一的名称。当需要定位某个队列时采用URL的方式进行访问，URL是系统自动给创建的队列分配的。队列在发送消息时尽最大努力保证“先进先出”；并非绝对地保证先进的数据一定会最先被投递给指定的接收者，这是它和普通的队列最大不同之处。不过SQS允许用户在消 息中添加有关的序列数据，对于数据发送顺序要求比较髙的用户可以在发送消息之前向其中加入相关信息。和队列相比，消息涉及的内容更多，需要考虑的问题更复杂,下面就消息的内容进行分析。

2.3 使用前提

1. 需要创建队列

2. 指定队列权限，可以让其他程序、服务推送信息到队列中进行轮循

开发人员最初使用 Amazon SQS 时只需用到五个 API：CreateQueue、SendMessage、ReceiveMessage、ChangeMessageVisibility 和 DeleteMessage。

Amazon SQS 会尽量保持消息顺序，但是由于队列的分布式特性，SQS无法保证发送消息的先后顺序。每个 Amazon SQS 队列都具有可配置的可见性超时(Default Visibility Timeout)。在从队列中读取消息后的指定时间内，该消息对其他读取者保持不可见。只要消息的处理时间短于可见性超时，每条消息都会得以处理并删除。如果处理消息的组件出现失败或不可用，可见性超时结束后该消息即对读取该队列的任何组件可见。这允许您让多个组件同时从同一队列中读取消息，每个组件负责处理不同的消息。

在 Amazon SQS 返回消息后，该消息会保存在队列中，称之为In Flight状态。删除请求可确认已处理了该消息。如果不删除消息，Amazon SQS 将在另一个接收请求中递送该消息。由于分布式 Amazon SQS 系统中的某台服务器在执行删除时不可用，DeleteMessage 操作未能删除消息的所有副本。该消息副本可能会被再次递送。所以应该在应用中进行相应的设计，以便在再次收到已删除的消息时不会出现错误或不一致。

如果连续 30 天以上时间没有针对某个队列签发以下任何请求，SQS有可能删除该队列：SendMessage、ReceiveMessage、DeleteMessage、GetQueueAttributes 和 SetQueueAttributes。

短轮询和长轮询(区别是是否采样)

如果使用短轮询，则您从队列中检索消息时，Amazon SQS 会对服务器的一个子集（基于加权随机分布）进行采样，并且仅从这些服务器返回消息。这意味着，特定接收请求可能不会返回您的所有消息。或者，如果您的队列中有少量消息（少于 1 000 条），则意味着，特定请求可能不会返回您的任何消息，而后续请求则会返回您的任何消息。如果您继续从您的队列中检索消息，则 Amazon SQS 会对所有服务器进行采样，您会收到您的所有消息。

使用 Amazon SQS 进行长轮询的一个好处是：在没有消息可返回以答复发送到 Amazon SQS 队列的 ReceiveMessage 请求时，可以减少空响应数量。在发送响应之前，长轮询允许 Amazon SQS 产品等到队列中的消息可用为止。因此，如果连接没有超时，则对 ReceiveMessage 请求的响应将至少包含一条可用的消息（如果有），最多可包含 ReceiveMessage 调用中请求的最大消息数。

Receive Message Wait Time就是设置短轮询还是长轮询的,0代表短轮询,最大到20。

Amazon SQS 使用您需要熟悉的以下三个标识符：队列 URL，消息 ID，接收句柄。每个队列飞行消息(Messages in Flight)的数量限制为 120,000。消息被队列接收后会处于飞行状态，但尚未从队列中删除。如果达到 120,000 的限制，将会收到一条来自 Amazon SQS 的“OverLimit”错误消息。为避免达到限制，应该在处理消息后将其从队列删除。

延迟队列允许将队列中新消息的传递操作推迟特定的秒数(Delivery Delay)。如果您创建延迟队列，则发送到该队列的任何消息在延迟期间对使用者都不可见。延迟队列类似于可见性超时，因为这两种功能都使得使用者在特定的时间段内无法获得消息。延迟队列和可见性超时之间的区别在于：对于延迟队列，消息在首次添加到队列时是隐藏的；而对于可见性超时，消息只有在从队列取回后才是隐藏的。

默认拒绝和显式拒绝的区别很重要，因为默认拒绝可以被允许覆盖，但是显式拒绝就不能。显示拒绝是并的关系,默认拒绝是或的关系。

第3章 Amazon SQS的应用

3.1 下载与安装

第一步：安装  
对于Amazon SQS支持，您必须安装额外的依赖项。您可以使用Celery[sqs]包一次性安装Celery和这些依赖项:

$ pip install celery[sqs]

第二步：配置  
您必须在代理URL中指定SQS:

broker\_url = 'sqs://ABCDEFGHIJKLMNOPQRST:ZYXK7NiynGlTogH8Nj+P9nlE73sq3@'

其中URL格式为:

sqs://aws\_access\_key\_id:aws\_secret\_access\_key@

3.2 使用注意

请注意，您必须记住在结尾处包含@符号并对密码进行编码，以便始终能够正确地解析它。例如:

from kombu.utils.url import quote

aws\_access\_key=quote("ABCDEFGHIJKLMNOPQRST")aws\_secret\_key=quote("ZYXK7NiynGlTogH8Nj+P9nlE73sq3")

broker\_url="sqs://{aws\_access\_key}:{aws\_secret\_key}@".format(aws\_access\_key=aws\_access\_key, aws\_secret\_key=aws\_secret\_key,)

还可以使用环境变量AWS\_ACCESS\_KEY\_ID和AWS\_SECRET\_ACCESS\_KEY设置登录凭据，在这种情况下，代理URL只能是sqs://。  
 如果在实例上使用IAM角色，可以将BROKER\_URL设置为:sqs://，并且kombu将尝试从实例元数据中检索访问令牌。

3.3 选项

3.3.1 区域

默认区域是us-east-1，但是您可以通过配置broker\_transport\_options设置来选择另一个区域:broker\_transport\_options = {'region': 'eu-west-1'}。另请参阅Amazon Web Services区域概览可在这里找到:

<http://aws.amazon.com/about-aws/globalinfrastructure/>。

3.3.2 可见性超时值

可见性超时定义了在消息重新传递给另一个工作人员之前，等待工作人员确认任务的秒数。例如，这个选项是通过broker\_transport\_options设置设置的:broker\_transport\_options = {'visibility\_timeout': 3600} # 1 hour。默认的可见性超时是30分钟。

3.3.3 轮询间隔

轮询间隔决定在不成功的轮询之间休眠的秒数。这个值可以是int，也可以是float。默认值为1秒:这意味着当没有更多的消息需要读取时，工作人员将休眠1秒。您必须注意，更频繁的轮询也更昂贵，因此增加轮询间隔可以节省资金。可以通过broker\_transport\_options设置轮询间隔:broker\_transport\_options = {'polling\_interval': 0.3}。

非常频繁的轮询间隔可能导致繁忙的循环，导致工作人员使用大量CPU时间。如果需要亚毫秒精度，则应该考虑使用另一种传输方式，如RabbitMQ <broker-amqp>或Redis <broker-redis>。

3.3.4 队列的前缀

默认情况下，Celery不会为队列名称分配任何前缀，如果您有其他使用SQS的服务，您可以使用broker\_transport\_options设置来配置它:broker\_transport\_options = {'queue\_name\_prefix': 'celery-'}。

警告：

如果任务没有在visibility\_timeout中得到确认，则任务将重新交付给另一个工作人员并执行。这会导致ETA/倒计时/重试任务出现问题，其中执行的时间超过了可见性超时;事实上，如果发生这种情况，它将被执行一次又一次的循环。因此，您必须增加可见性超时，以匹配您计划使用的最长ETA时间。

请注意，Celery会在工作人员关机时重新发送消息，所以长时间的可见性超时只会在断电或强制终止工作人员时延迟“丢失”任务的重新发送。周期性任务不会受到可见性超时的影响，因为它是一个独立于ETA/倒计时的概念。在撰写本文时，AWS支持的最大可见性超时为12小时(43200秒):broker\_transport\_options = {'visibility\_timeout': 43200} SQS还不支持工作人员远程控制命令。SQS还不支持事件，因此不能与Celery事件、celerymon或Django管理监视器一起使用。

3.3.5 结果

Amazon Web Services家族中的多个产品可能是存储或发布结果的理想选择，但是目前还没有包含这样的结果后端。

警告：不要将amqp结果后端与SQS一起使用。它将为每个任务创建一个队列，并且不会收集队列。这可能会花费您的钱，您最好将AWS结果存储后端贡献给Celery。

3.4 具体应用

3.4.1 如何创建和配置队列

最首要最常见的 Amazon SQS 任务是创建队列：如何创建和配置队列。

步骤 1：创建队列

登录Amazon SQS控制台。

选择新建队列。

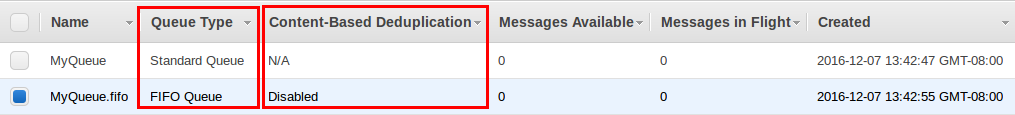
在新建队列页面上，确保您位于正确的区域，然后键入队列名称。

注意：FIFO 队列的名称必须以 .fifo 后缀结尾。

标准 默认处于选中状态。选择 FIFO。

要使用默认参数创建队列，请选择快速创建队列。您的新队列随即将会创建并在队列列表中处于选中状态。

注意：创建队列时，队列可能需要一小段时间在整个 Amazon SQS 中传播。Queue Type 列将帮助您一眼区分出标准队列与 FIFO 队列。对于 FIFO 队列，Content-Based Deduplication 列将显示您是否已启用[一次性处理](https://docs.amazonaws.cn/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html" \l "FIFO-queues-exactly-once-processing)。



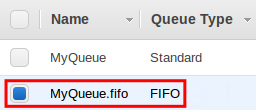
您的队列的名称、URL 和 ARN 将显示在详细信息选项卡上。



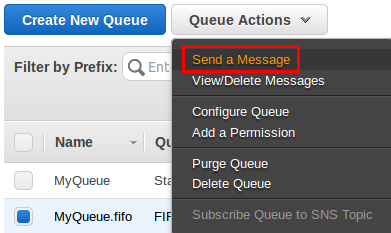
步骤 2：发送消息

在创建队列之后，可向其发送消息。以下示例说明如何向现有队列发送消息。

从队列列表中，选择您已创建的队列。

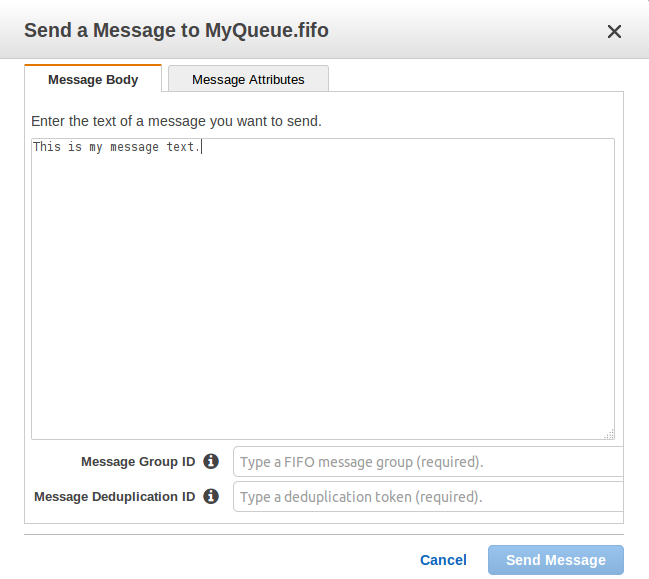


在队列操作中，选择发送消息。



将显示 Send a Message to QueueName 对话框。

以下示例显示了特定于 FIFO 队列（已禁用 [content-based deduplication](https://docs.amazonaws.cn/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html" \l "FIFO-queues-exactly-once-processing)）的消息组 ID 和消息重复数据删除 ID 参数。

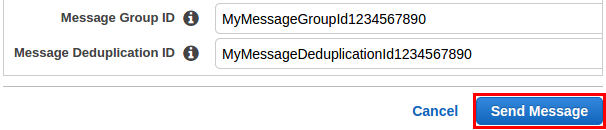


3. 要将消息发送到 FIFO 队列，请键入 Message Body、Message Group

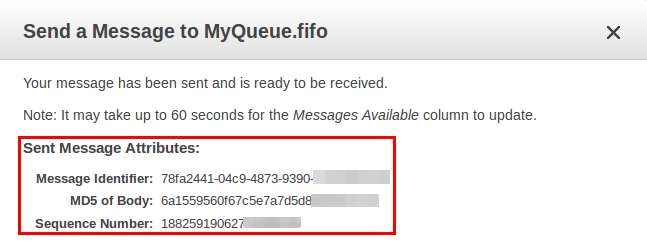
IDMyMessageGroupId1234567890 和 Message Deduplication

IDMyMessageDeduplicationId1234567890，然后选择 Send Message。有关更多信息，请参阅 [FIFO 传送逻辑](https://docs.amazonaws.cn/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html" \l "FIFO-queues-understanding-logic)。

注意：始终需要 message group ID。但是，如果启用 content-based deduplication，则 message deduplication ID 是可选的。



您的消息将会发送，并显示 Send a Message to QueueName 对话框，其中将显示已发送消息的属性。以下示例显示了特定于 FIFO 队列的 Sequence Number属性。

4. 选择 Close。

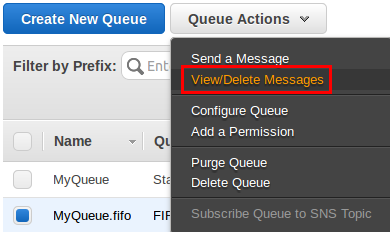
步骤 3：接收和删除消息

将消息发送到队列中之后，您可使用它（从队列中检索消息）。向队列请求消息时，您无法指定要获取的消息。而应指定要获取的最大消息数量（最多 10 条）。

在本教程中，您将学习如何接收和删除消息。

从队列列表中，选择您已创建的队列。

从 Queue Actions (队列操作) 中，选择 View/Delete Messages (查看/删除消息)。



将显示 View/Delete Messages in QueueName (在 <队列名称> 中查看/删除消息) 对话框。

注意：当您第一次执行此操作时，将会显示一个信息屏幕。要隐藏屏幕，请选中 Don't show this again (不再显示)复选框。

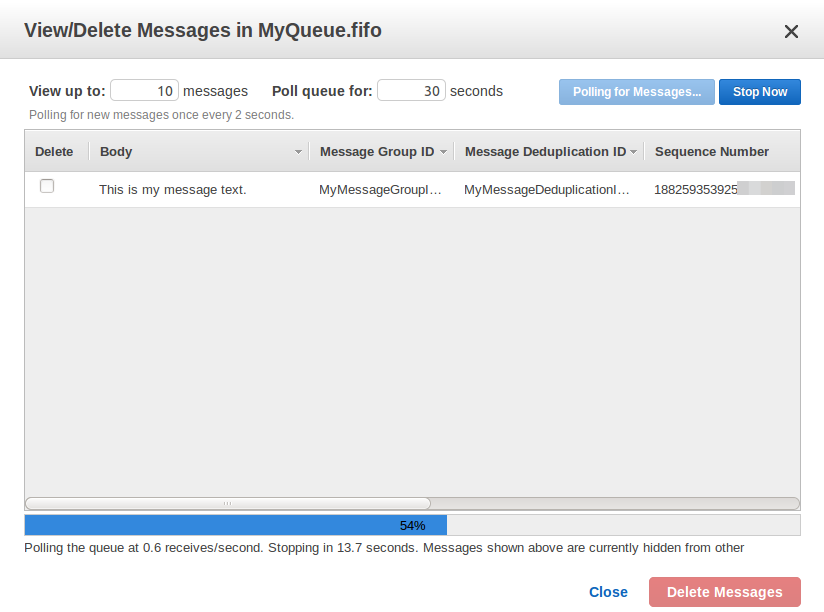
选择 Start Polling for messages (开始轮询消息)



Amazon SQS 开始轮询队列中的消息。对话框将显示来自队列的消息。对话框底部的进度条显示消息可见性超时的状态。

以下示例显示特定于 FIFO 队列的 Message Group ID (消息组 ID)、Message Deduplication

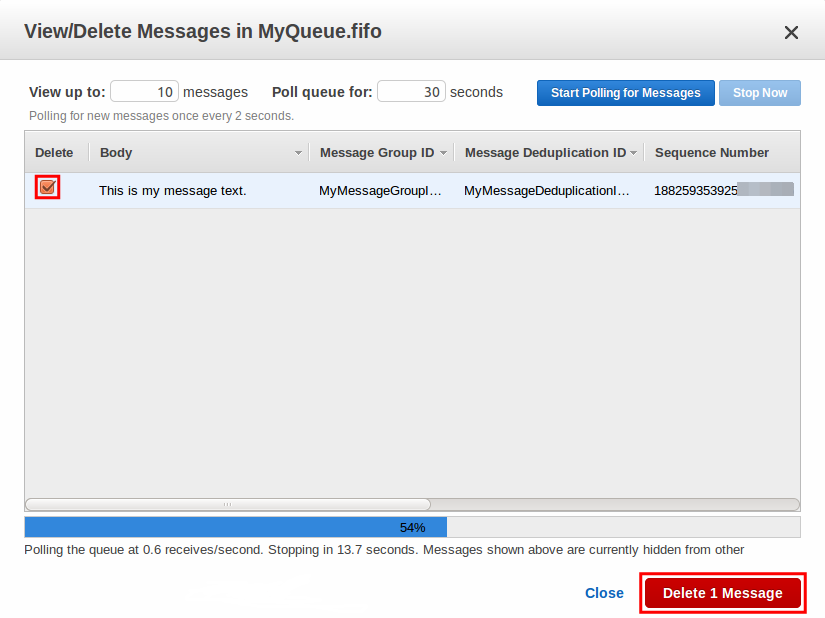
ID (消息重复数据删除 ID)和 Sequence Number</guilabel> (序列号) 列。



注意：当进度栏填满时，[可见性超时](https://docs.amazonaws.cn/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html)将过期，消息将变得对使用者可见.

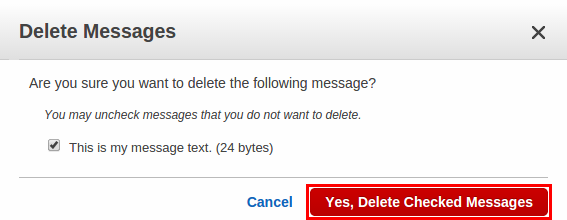
在可见性超时过期之前，选择您要删除的消息，然后选择 Delete 1 Message (删

除消息)。



在 Delete Messages (删除消息) 对话框中，确认选中要删除的消息，然后选择 Yes, Delete

Checked Messages (是，删除所选消息)



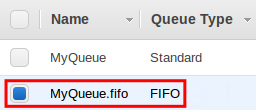
选定的消息已删除。

选择 Close (关闭)。

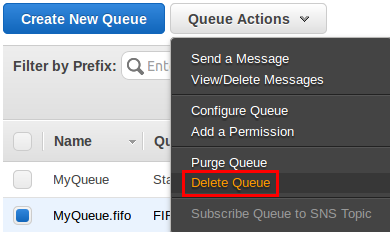
步骤 4：删除队列

如果您不使用 Amazon SQS 队列 (并且预期近期不会使用)，最好是将其从 Amazon SQS 中删除。在本教程中，您将学习如何删除队列。

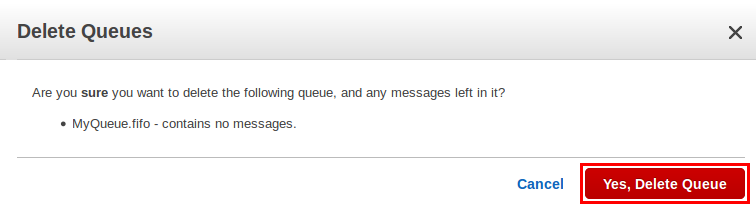
从队列列表中，选择您已创建的队列。



从队列操作中，选择删除队列。



此时将显示 Delete Queues 对话框。



选择 Yes, Delete Queue。

队列将被删除。

3.4.2 创建一个对象来连接到Amazon SQS

AmazonSQSClient objClient = new AmazonSQSClient("AssaaIAxxxxx", "lssssX1axxxxxxx");

使用通过key来new 一个 AmzonSQSClient对象。这个key是你创建亚马逊云服务账号，再创建SQS队列，系统生成的。我这边是云服务发邮件过来的。具体key就不公布出去。

2） 展示云端队列。

　　展示云端队列，一个是调用objClient的某个方法获取队列列表。二是需要一个对象来存储这个列表。这二者都可以在SDK中找到。

ListQueuesResponse objqueuesResponseList = new ListQueuesResponse();

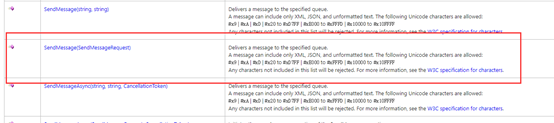
objqueuesResponseList = objClient.ListQueues(new ListQueuesRequest());

ListQueuesResult Result = objqueuesResponseList.ListQueuesResult;

主要是基于Request 和 Response的。 因此需要 ListQueuesRequest对象，和ListQueuesResponse对象。

3） 发送消息。

　　调用objClient.SendMessage方法。



从SDK中可以看到。SendMessage中需要一个MessageRequest对象。因此我们将我们需要发送的消息。封装成一个MessageRequest对象即可。以下是一个MessageRequest对象的实例。

       new SendMessageRequest() {MessageBody = xxxxx ,

QueueUrl = xxxx } ;

其中QueueUrl可以从2）中的result中获取，在此就不赘述。

4） 接受消息

a)new一个 ReceiveMessageResponse 对象。

b)调用objClient 的 ReceiveMessage方法

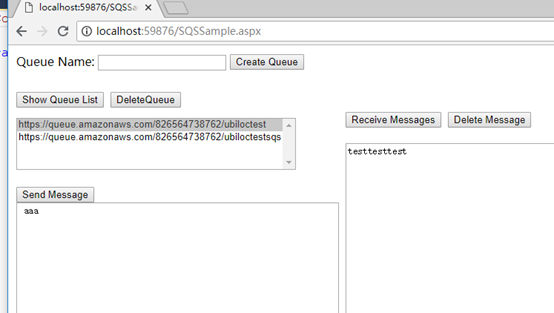
ReceiveMessageResponse queueReceiveMessageResponse

= new ReceiveMessageResponse();

queueReceiveMessageResponse=objClient.ReceiveMessage(new ReceiveMessageRequest() { QueueUrl = xxx, MaxNumberOfMessages = 10 });

c)new一个 ReceiveMessageResult对象存储 Response的结果

d)使用一个List<Message>列表存储Message列表。 ReceiveMessageResult.Message



以下是部分源代码：

private void DisplayQueues()  
{  
ListQueuesResponse objqueuesResponseList = new ListQueuesResponse();

objqueuesResponseList = objClient.ListQueues(new ListQueuesRequest());

ListQueuesResult Result = objqueuesResponseList.ListQueuesResult;  
  
this.QueuesList.Visible = true;  
this.QueuesList.DataSource = Result.QueueUrl;  
this.QueuesList.DataBind();  
}

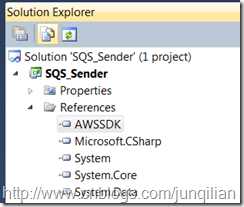
protected void btnSendMessage\_Click(object sender, EventArgs e)  
{  
if (QueuesList.SelectedIndex > -1)  
{  
if (!string.IsNullOrEmpty(txtMessage.Text))  
{  
string selectedQueue = this.QueuesList.SelectedItem.Value.ToString();

objClient.SendMessage(new SendMessageRequest() { MessageBody = this.txtMessage.Text, QueueUrl = selectedQueue });

Response.Write("<script>alert('Message sent successfully to Queue.');</script>");  
}  
else  
{  
Response.Write("<script>alert('Please enter Message Text.');</script>");  
}  
}  
else  
{  
Response.Write("<script>alert('Please select Queue from list.');</script>");  
}  
}

3.4.3 使用Amazon SQS实现与AutoCAD远程交互

首先建立一个普通的windows form应用程序作为发送端。添加到AWSSDK.dll的引用，这个dll位于C:\Program Files (x86)\AWS SDK for .NET\bin目录下。

[](http://images.cnblogs.com/cnblogs_com/junqilian/201203/201203251553384673.png)

需要创建一个对象来连接到Amazon SQS。连接到SQS需要你的Access Key和Secret Key。这两个key可以从[这里](https://aws-portal.amazon.com/gp/aws/securityCredentials)找到。可以通过下面的方式把AccessKey和Secret Key以写在代码中。

AmazonSQS sqs = AWSClientFactory.CreateAmazonSQSClient(

<Access Key placeholder>,

<Secret Key placeholder>);

更好的方式是写在配置文件App.Config中，如下：

<?xml version="1.0" encoding="utf-8" ?><configuration>

<appSettings>

<add key="AWSAccessKey" value="\*\*\*\*\*"/>

<add key="AWSSecretKey" value="\*\*\*\*\*"/>

</appSettings></configuration>

这样在创建AmazonSQS对象时直接用如下语句即可。程序会从App.Config中查找相关的Key信息。

//Create AzazomSQS client using information of appconfig.xml

AmazonSQS sqsClient = AWSClientFactory.CreateAmazonSQSClient();

这个发送端程序要完成的工作就是向远程的AutoCAD发送工作序列，其中有三个任务：

"Say: Hello AutoCAD" : 在AutoCAD命令行中输出这个信息；

"Action: drawline" ： 在AutoCAD中画一条线；

"Action: drawcircle" ： 在AutoCAD中画一圆。

下面是发送端的完整代码：

using System;using System.Collections.Generic;

using System.ComponentModel;

using System.Data;using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;using Amazon;using Amazon.SQS;

using Amazon.SQS.Model;

namespace SQS\_Sender

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

//Create AzazomSQS client using information of appconfig.xml

AmazonSQS sqsClient = AWSClientFactory.CreateAmazonSQSClient();

//AmazonSQS sqs =

// AWSClientFactory.CreateAmazonSQSClient(

// <Access Key placeholder>,

// <Secret Key placeholder>

// );

try

{

//Create a queue if it does not exits

CreateQueueRequest sqsRequest = new CreateQueueRequest();

sqsRequest.QueueName = "AutoCADJobQueue";

CreateQueueResponse sqsResponse = sqsClient.CreateQueue(sqsRequest);

string myQueueUrl = string.Empty; ;

if (sqsResponse.IsSetCreateQueueResult())

{

myQueueUrl = sqsResponse.CreateQueueResult.QueueUrl;

}

//post a message to the queue

SendMessage(sqsClient, myQueueUrl, "Say: Hello AutoCAD");

SendMessage(sqsClient, myQueueUrl, "Action: drawline");

SendMessage(sqsClient, myQueueUrl,"Action: drawcircle");

}

catch (AmazonSQSException aEx)

{

MessageBox.Show(aEx.Message);

}

}

private static void SendMessage(AmazonSQS sqsClient,

string myQueueUrl,

string messageBody)

{

SendMessageRequest sendMsgRequest = new SendMessageRequest();

sendMsgRequest.QueueUrl = myQueueUrl;

sendMsgRequest.MessageBody = messageBody;

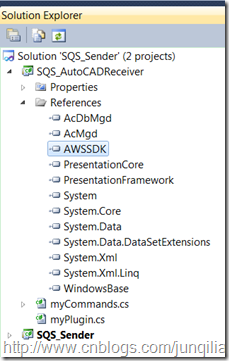
sqsClient.SendMessage(sendMsgRequest);

}

}

}

下面完成接收端。接收端一个AutoCAD插件，我们可以利用[AutoCAD 2010-2012 dotNet Wizards.zip](http://images.autodesk.com/adsk/files/AutoCAD_2010-2012_dotNET_Wizards.zip)来创建一个插件模版，然后添加到AWSSDK.dll的引用。

[](http://images.cnblogs.com/cnblogs_com/junqilian/201203/201203251553453946.png)

直接贴出完整代码如下：

Using System;

using Autodesk.AutoCAD.ApplicationServices;using

Autodesk.AutoCAD.DatabaseServices;

Using Autodesk.AutoCAD.EditorInput;

using Autodesk.AutoCAD.Geometry;

using Autodesk.AutoCAD.Runtime;using System.Collections.Generic;

using Amazon;using Amazon.SQS;using Amazon.SQS.Model;

// This line is not mandatory, but improves loading performances

[assembly: CommandClass(typeof(SQS\_AutoCADReceiver.MyCommands))]

namespace SQS\_AutoCADReceiver

{

// This class is instantiated by AutoCAD for each document when

// a command is called by the user the first time in the context

// of a given document. In other words, non static data in this class

// is implicitly per-document!

public class MyCommands

{

[CommandMethod("DoJobFromCloud",CommandFlags.Modal)]

public void DoJobFromCloud() //this method can be any name

{

Document doc = Application.DocumentManager.MdiActiveDocument;

Database db = doc.Database;

Editor ed = doc.Editor;

//get the queue client

AmazonSQS sqsClient = AWSClientFactory.CreateAmazonSQSClient(

"<your access key>",

"<your secret key>"

);

try

{

string myQueueUrl = string.Empty;

//checking whether the queue exits or not

ListQueuesRequest lstQueueRequest = new ListQueuesRequest();

ListQueuesResponse lstQueueResponse = sqsClient.ListQueues(lstQueueRequest);

if (lstQueueResponse.IsSetListQueuesResult())

{

ListQueuesResult listQueueResult = lstQueueResponse.ListQueuesResult;

foreach (string queueUrl in listQueueResult.QueueUrl)

{

ed.WriteMessage(queueUrl+ "\n");

if (queueUrl.Contains("AutoCADJobQueue"))

{

myQueueUrl = queueUrl;

}

}

}

if (myQueueUrl.Length > 0) //The queue exits

{

//Get the message number in the queue

int numMessage = GetMessageNumber(sqsClient, myQueueUrl);

if (numMessage > 0)

{

do

{

//receive a message

ReceiveMessageRequest receiveReq = new ReceiveMessageRequest();

receiveReq.QueueUrl = myQueueUrl;

ReceiveMessageResponse receiveRes

= sqsClient.ReceiveMessage(receiveReq);

if (receiveRes.IsSetReceiveMessageResult())

{

foreach (Message msg in

receiveRes.ReceiveMessageResult.Message)

{

if (msg.IsSetBody())

{

if (msg.Body.StartsWith("Say"))

{

//print the message in AutoCAD

ed.WriteMessage(msg.Body + "\n");

}

if (msg.Body.StartsWith("Action"))

{

if (msg.Body == "Action: drawline")

{

//draw a line in AutoCAD

DrawLine();

ed.WriteMessage(msg.Body

+ " executed!\n");

}

if (msg.Body == "Action: drawcircle")

{

//draw a circle in AutoCAD

DrawCircle();

ed.WriteMessage(msg.Body

+ " executed!\n");

}

}

}

//delete the message from queue,

//since we have aleady worked on it

DeleteMessageRequest delMsgRequest

= new DeleteMessageRequest();

delMsgRequest.QueueUrl = myQueueUrl;

delMsgRequest.ReceiptHandle= msg.ReceiptHandle;

sqsClient.DeleteMessage(delMsgRequest);

}

}

numMessage = GetMessageNumber(sqsClient, myQueueUrl);

} while (numMessage > 0);

}

}

}

catch (AmazonSQSException aEx)

{

ed.WriteMessage("\n Error: " + aEx.Message);

}

}

private static int GetMessageNumber(AmazonSQS sqsClient, string myQueueUrl)

{

int numMessage = 0;

GetQueueAttributesRequest req = new GetQueueAttributesRequest();

req.AttributeName.Add("All");

req.QueueUrl = myQueueUrl;

GetQueueAttributesResponse res = sqsClient.GetQueueAttributes(req);

if (res.IsSetGetQueueAttributesResult())

{

GetQueueAttributesResult attResult = res.GetQueueAttributesResult;

numMessage = attResult.ApproximateNumberOfMessages;

}

return numMessage;

}

public void DrawCircle()

{

Document doc = Application.DocumentManager.MdiActiveDocument;

Database db = doc.Database;

using (Transaction trans = db.TransactionManager.StartTransaction())

{

BlockTable bt = trans.GetObject(db.BlockTableId, OpenMode.ForWrite) as BlockTable;

BlockTableRecord modelSpace

= trans.GetObject(bt[BlockTableRecord.ModelSpace], OpenMode.ForWrite)

as BlockTableRecord;

using (Circle c = new Circle())

{

c.Center = new Point3d(0, 0, 0);

c.Radius = 10;

modelSpace.AppendEntity(c);

trans.AddNewlyCreatedDBObject(c, true);

}

trans.Commit();

}

}

public void DrawLine()

{

Document doc = Application.DocumentManager.MdiActiveDocument;

Database db = doc.Database;

using(Transactiontrans= db.TransactionManager.StartTransaction())

{

BlockTablebt=trans.GetObject(db.BlockTableId, OpenMode.ForWrite) as BlockTable;

BlockTableRecord modelSpace

= trans.GetObject(bt[BlockTableRecord.ModelSpace],

OpenMode.ForWrite)

as BlockTableRecord;

using (Line ln = new Line())

{ln.StartPoint = new Point3d(0,0,0);

ln.EndPoint = new Point3d(10, 10, 0);

modelSpace.AppendEntity(ln);

trans.AddNewlyCreatedDBObject(ln, true);

}trans.Commit();

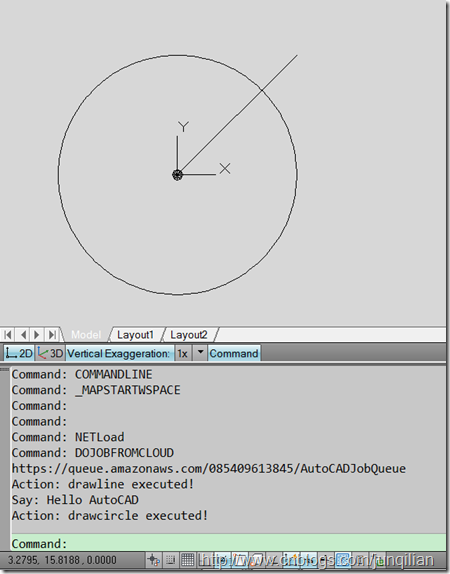
}

}

}

}

下面是在AutoCAD中的运行结果：

[](http://images.cnblogs.com/cnblogs_com/junqilian/201203/201203251553497944.png)

第4章 结论

4.1.1 优势

1. 消除管理开销

AWS 负责管理所有正在进行的操作和底层基础设施，以提供高度可用且可扩展的消息队列服务。SQS 无需前期成本，无需购买、安装和配置消息收发软件，无需耗时地构建和维护配套基础设施。SQS 队列会以动态方式自动创建和扩展，从而使您可以快速而高效地构建和扩展应用程序。

1. 可靠传送消息

使用 Amazon SQS 可以在任意吞吐量级别传输任何规模的数据，而不会丢失消息，并且无需其他服务即可保持可用。借助 SQS，您可以分离应用程序组件，以让其独立运行，在发生故障时不影响其他组件，从而提高系统的总体容错能力。每个消息有多个副本以冗余的方式存储在多个可用区中，以确保它们在需要时随时可用。

1. 保证敏感数据安全

借助 Amazon SQS，您可以使用服务器端加密 (SSE) 功能加密每个消息正文，以在应用程序之间交换敏感数据。Amazon SQS SSE 与 AWS Key Management Service (KMS) 集成，使您能够集中管理保护 SQS 消息的密钥以及保护其他 AWS 资源的密钥。AWS KMS 会将加密密钥的每次使用情况记录到 AWS CloudTrail，以帮助满足您的监管与4. 合规性需求

以成本高效的方式进行弹性扩展

Amazon SQS 利用 AWS 云按需进行动态扩展。SQS 可以根据您的应用情况进行弹性扩展，因此，您无需担心容量规划和预配置。每个队列的消息数量不限，而且标准队列能提供几乎无限的吞吐量。相对于自行管理的消息收发中间件采用的“不中断”模式，按使用量付费的模式可以为您节约大量成本。

4.1.2 限制

1. [与队列相关的限制](http://docs.amazonaws.cn/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-limits.html" \l "limits-queues)

延迟队列。对于大多数标准队列（取决于队列流量和消息积压），最多可能有 120000 个传输中的消息（使用者已从队列接收，但尚未从队列中删除）。如果您达到此限制，Amazon SQS 将返回 OverLimit 错误消息。为避免达到此限制，您应该在处理消息后将其从队列中删除。您还可以增加用来处理消息的队列的数量。要请求提高上限，请[提交支持请求](https://console.amazonaws.cn/support/home" \l "/case/create?issueType=service-limit-increase&limitType=service-code-sqs" \t "http://docs.amazonaws.cn/AWSSimpleQueueService/latest/SQSDeveloperGuide/_blank)。列出的队列：每个 [ListQueues](https://docs.amazonaws.cn/AWSSimpleQueueService/latest/APIReference/API_ListQueues.html) 请求 1,000 个队列。FIFO 队列的名称必须以 .fifo 后缀结尾。建议向队列添加的标签不要超过 50 个。

1. [与策略相关的限制](http://docs.amazonaws.cn/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-limits.html" \l "limits-policies)

批处理消息 ID最多可包含 80 个字符。一条消息可以包含最多 10 个

数据属性。一个消息批请求中最多可包含 10 条消息。消息可以仅包含 XML、JSON 和非格式化的文本。默认情况下，消息将保留 4 天。在不使用批处理的情况下，FIFO 队列的每个操作每秒最多支持 300 条消息。消息的默认（最小）不可见期间为 0 秒，可见性超时为 30 秒。。最大延迟为 15 分钟。要发送大于 256 KB 的消息，需使用[适用于 Java 的 Amazon SQS 扩展客户端库](https://github.com/awslabs/amazon-sqs-java-extended-client-lib" \t "http://docs.amazonaws.cn/AWSSimpleQueueService/latest/SQSDeveloperGuide/_blank)。

4.2 未来发展

Amazon SQS 是一个由Amazon hosted的,可信赖以及扩展性强的一个分布式在线消息服务。  
 利用Amazon SQS, 开发者可以更简易地开发出分布式的消息应用， 而不需要要求分布式节点的完全可用以及稳定， 因为消息都是由Amazon SQS 存储和分发。

2001年互联网泡沫使业务量锐减,系统资源大量闲置。在这种背景下，亚马逊提出一个创新的想法,即将硬件设施等基础资源封装成服务供用户使用,即通过虚拟化技术提供可动态调度的弹性服务,之后经过不断完善。亚马逊云开发由此出现，简单队列服务(SQS)亚马逊为了处理数量庞大的并发访问和交易购置了大量服务器。提供一组广泛的全球计算,存储,数据库,分析,应用程序和部署服务,可帮助组织更快地迁移,降低,成本和扩展应用程序。

之所以Amazon 的SQS 能结合进云计算环境中，主要是因为它很简单。对需要使用消息队列的大多数系统而言，所要做的不过是通过一个简单的API 来提交和取得消息，完全不用操心消息在队列中的完整性。在这样简单的平台上搞开发和进行维护确实很乏味，但是，比起许多复杂、昂贵的消息队列商业软件来，还是简单一点好。

综上所述，云计算确实给我的生活带来了极大的便利，推动了整个信息产业的发展与进步。但同时云计算技术仍然存在一些尚未解决的技术性问题，以及云安全问题。 我们相信随着科学信息技术的进一步发展，关于云计算的技术性问题可以逐步解决，云计算的应用将会越来越广泛。

参考文献

1. 杨帆.中国新通信[J].财政研究，2004,21(1)：56-58.中国新通信2015年17期
2. 王毅.《亚马逊AWS云基础与实战》[M].厦门：厦门大学出版社，
3. Thomas Erl等.《云计算：概念、技术与架构》（龚奕利等）.北京：机械工业出版社．2017-09-17
4. 林伟伟 刘波.《分布式计算、云计算与大数据》.北京：机械工业出版社.2017-04
5. Kai Hwang等.《云计算与分布式系统：从并行处理到物联网》（武永卫等）.北京：机械工业出版社.2013-01
6. 刘鹏.《云计算》.北京：电子工业出版社。2010-03
7. 杨传辉.《大规模分布式存储系统》.北京：机械工业出版社.2013-09-01
8. 林允溥.《AWS云端企业实战圣经:亚马逊如何构造云端计算》.2012-08
9. 博格兰德等.《云服务测试》.北京：人民邮电出版社.2014-07-01
10. 张为民等.《云计算:深刻改变未来》.北京：科学出版社.2009-12