AWS IoT 详解

AWS 架构完善的框架

2019 年 12 月

声明

客户有责任对本文档中的信息进行独立评估。本文档：(a) 仅供参考，(b) 代表 AWS 当前的产品服务和实践，如有变更，恕不另行通知，以及 (c) 不构成 AWS 及其附属公司、供应商或授权商的任何承诺或保证。AWS 产品或服务均“按原样”提供，没有任何明示或暗示的担保、声明或条件。AWS 对其客户的责任和义务由 AWS 协议决定，本文档与 AWS 和客户之间签订的任何协议无关，亦不影响任何此类协议。

目录

摘要

本白皮书介绍针对 AWS 架构完善的框架的 **AWS IoT 详解**。客户可以借助本“详解”来审查并改进其基于云的架构，并更好地了解设计决策对业务的影响。本文档针对架构完善的框架，讲解五大支柱的一般设计原则以及具体最佳实践与指南。

简介

AWS 架构完善的框架能够帮助您认识到您在 AWS 上构建系统时所做决策的优缺点。通过使用此框架，您将了解在云中设计和运行可靠、安全、高效且具成本效益的系统的架构最佳实践。该框架提供了一种方法，使您能够根据最佳实践持续衡量架构，并确定需要改进的方面。我们相信，拥有架构完善的系统能够大大提高业务成功的可能性。

在这篇“详解”中，我们重点介绍如何在 AWS 云中设计、部署 IoT（物联网）工作负载，以及进行架构设计。要实施架构完善的 IoT 应用程序，从采购互联的物理资产（事物）开始，到用安全、可靠和自动化的方式停用这些资产，您必须自始至终遵循架构完善原则。除了 AWS 云最佳实践之外，本文档还阐述了将物理资产连接到互联网的影响、注意事项和建议。

本文档仅涵盖架构完善的框架中特定于 IoT 的工作负载详细信息。建议您阅读 AWS 架构完善的框架白皮书，综合考虑其他“详解”中的最佳实践和问题。

本文档面向技术人员，例如首席技术官 (CTO)、架构师、开发人员、嵌入式工程师和运维团队成员。阅读本文档后，您将了解适用于 IoT 应用程序的 AWS 最佳实践和策略。

定义

AWS 架构完善的框架基于五大支柱建立，即“卓越运营”、“安全性”、“可靠性”、“性能效率”和“成本优化”。在设计技术解决方案时，您必须基于您的业务环境在各个支柱之间做出合理取舍。AWS 提供了适用于 IoT 工作负载的多种服务，可助您为应用程序设计出可靠的架构。物联网 (IoT) 应用程序由许多设备（或事物）组成，这些设备安全地连接互为补充的基于边缘的组件和基于云的组件并与之交互，以提供业务价值。IoT 应用程序收集、处理和分析互联设备生成的数据并根据数据执行操作。本节概述了整个文档中用于构建 IoT 工作负载的 AWS 组件。构建物联网工作负载时，需要考虑七个不同的逻辑层：

- 设计和制造层
- 边缘层
- 预置层

- 通信层
- 提取层
- 分析层
- 应用程序层

设计和制造层

设计和制造层包括产品概念化、业务和技术需求收集、原型设计、模块和产品布局与设计、组件采购和制造。每个阶段做出的决定都会影响 IoT 工作负载的下一个逻辑层，如下文所述。例如，某些 IoT 设备创建者更喜欢由签约制造商刻录和测试通用固件映像。这项决定将在一定程度上决定在预置层中需要执行哪些步骤。

您可以更进一步，在制造过程中为每个设备刻录唯一证书和私钥。这项决定可能会影响通信层，因为凭据的类型可能会影响网络协议的后续选择。如果该凭证永不过期，则可以简化通信层和预置层，但可能会因证书颁发机构的权限而增加数据丢失风险。

边缘层

IoT 工作负载的边缘层包括设备的物理硬件、管理设备上进程的嵌入式操作系统，以及设备固件（在 IoT 设备上编程的软件和说明）。边缘负责感测其他外围设备并在这些外围设备上执行操作。常见使用案例包括读取连接到边缘设备的传感器，或者根据用户动作更改外围设备的状态（例如，在运动传感器激活时开灯）。

AWS IoT 设备开发工具包通过针对您的编程语言或平台量身定制的 API，让您可以更轻松地将 AWS IoT Core 与您的设备和应用程序结合使用。

Amazon FreeRTOS 是用于微控制器的实时操作系统，使您可以对小型低功耗边缘设备进行编程，同时利用内存效率高、安全的嵌入式库。

AWS IoT Greengrass 是扩展您的 IoT 设备的 Linux 操作系统的软件组件。AWS IoT Greengrass 允许您在设备、数据缓存、AWS IoT Shadow 同步、本地 AWS Lambda 函数和机器学习算法之间运行 MQTT 本地路由。

预置层

IoT 工作负载的预置层由用于创建唯一设备标识的公钥基础设施 (PKI) 和向设备提供配置数据的应用程序工作流程组成。预置层还涉及设备的持续维护和最终停用。IoT 应用程序需要强大且自动化的预置层，以便 IoT 应用程序可以轻松地添加和管理设备。预置 IoT 设备时，您必须在这些设备上安装唯一的加密凭证。

通过使用 **X.509 证书**，您可以实施一个预置层，该层可为您的设备安全地创建可信身份，用于对通信层进行身份验证和授权。X.509 证书则由一家名为证书颁发机构 (CA) 的可信实体颁发。虽然由于内存和处理需要，X.509 证书确实要耗用受限设备上的资源，但由于它们的操作具有可伸缩性并享受标准网络协议的广泛支持，因此属于理想的身份识别机制。

AWS Certificate Manager 私有 CA 帮助您使用 API 自动管理 IoT 设备私有证书生命周期。私有证书（例如 X.509 证书）提供了一种安全的方法来为设备赋予长期身份。该身份可以在预置期间创建，并用于识别和授予针对 IoT 应用程序的设备权限。

AWS IoT Just In Time Registration (JITR) 使您能够以编程方式注册设备，以用于托管 IoT 平台（如 AWS IoT Core）。使用 Just-In-Time-Registration，在设备首次连接到您的 AWS IoT Core 终端节点时，您可以自动触发工作流程，来确定证书身份的有效性并确定应授予的权限。

通信层

通信层用于处理远程设备之间的连接、消息路由，以及设备与云之间的路由。通过通信层，您可以确定设备如何发送和接收 IoT 消息，以及设备如何在云端表示和存储其物理状态。

AWS IoT Core 通过提供托管消息代理（支持使用 MQTT 协议在设备之间发布和订阅 IoT 消息）来帮助您构建 IoT 应用程序。

AWS IoT 设备注册表可帮助您管理和操作事物。事物是特定设备或逻辑实体在云端的表现形式。事物还可以具有自定义的静态属性，这些属性可以帮助您在部署后识别、分类和搜索资产。

借助 **AWS IoT Device Shadow 服务**，您可以创建一个数据存储，将特定设备的当前状态包含在内。Device Shadow 服务能够以单独设备影子的形式为您连接到 AWS IoT 的各设备建立虚拟表示形式。每个设备的影子通过相应事物的名称唯一标识。

通过 **Amazon API Gateway**，您的 IoT 应用程序可以发出 HTTP 请求，以控制您的 IoT 设备。IoT 应用程序需要用于内部系统（例如，远程技术人员的控制面板）和外部系统（例如，家用消费类移动应用程序）的 API 接口。通过 Amazon API Gateway，您可以创建公用 API 接口，而无需预置和管理底层基础设施。

提取层

IoT 的关键业务驱动因素是能够聚合设备创建的所有不同数据流，并以安全可靠的方式将数据传输到物联网应用程序。提取层发挥着关键作用，在收集设备数据的同时将数据流与设备之间的通信分离开来。

借助 **AWS IoT 规则引擎**，您可以构建 IoT 应用程序，以使您的设备能够与 AWS 服务交互。它能根据接收消息的 MQTT 主题流，分析 AWS IoT 规则并执行操作。

Amazon Kinesis 是一项用于流式传输数据的托管服务，使您能够及时了解情况并快速响应 IoT 设备发来的新信息。Amazon Kinesis 直接与 AWS IoT 规则引擎集成，创建了一种无缝桥接方式，可以将使用 MQTT 的设备的轻量级设备协议与使用其他协议的内部 IoT 应用程序进行桥接。

与 Kinesis 类似，应在 IoT 应用程序中使用 **Amazon Simple Queue Service (Amazon SQS)**，将通信层与应用程序层分离。当您的应用程序需要在无需消息顺序的情况下处理 IoT 应用程序时，Amazon SQS 会启用事件驱动的可扩展提取队列。

分析层

实施 IoT 解决方案的好处之一是能够深入了解本地/边缘环境中发生的情况并获取相关数据。实现有上下文的深刻见解的主要方法是实施可处理和执行 IoT 数据分析的解决方案。

存储服务

IoT 工作负载通常设计用于生成大量数据。确保在持久存储的同时安全地传输、处理和使用这些离散数据。

Amazon S3 是基于对象的存储，设计用于在互联网上的任何位置存储和检索任意数量的数据。借助 Amazon S3，您可以构建可存储大量数据的 IoT 应用程序，以支持各种用途：监管、业务发展、指标、纵向研究、分析、机器学习和组织支持。Amazon S3 在管理数据的方式上为您提供了超高灵活性，不仅涉及到成本优化和延迟优化，还涉及到访问控制和合规性。

分析和机器学习服务

IoT 数据到达中央存储位置后，您可以通过对设备行为实施分析和机器学习来开始释放 IoT 的全部价值。借助分析系统，您可以根据分析做出由数据驱动的决策，从而开始在设备固件以及边缘和云逻辑中实施改进。借助分析和机器学习，IoT 系统可以实施主动策略（如预测维护或异常检测），从而提高系统效率。

AWS IoT Analytics 支持轻松在卷上对 IoT 数据执行复杂分析。在您使用自己的分析查询或 Jupyter 笔记本构建数据的不同物化视图时，AWS IoT Analytics 会管理基础 IoT 数据存储。

Amazon Athena 是一种交互式查询服务，可使用此服务通过标准 SQL 在 Amazon S3 中轻松分析数据。Athena 为无服务器服务，因此无需管理基础设施，客户只需为运行的查询付费。

Amazon SageMaker 是一种完全托管的平台，使您能够在云端和边缘层中快速构建、训练和部署机器学习模型。借助 Amazon SageMaker，IoT 架构可以开发历史设备遥测模型，以推断未来的行为。

应用程序层

AWS IoT 提供了多种方法来简化云原生应用程序使用 IoT 设备生成的数据的方式。这些互联功能包括无服务器计算的功能，用于创建 IoT 数据的物化视图的关系数据库，以及用于操作、检查、保护和管理 IoT 操作的管理应用程序。

管理应用程序

管理应用程序的目的是，在现场部署设备之后，创建可扩展的方式来操作您的设备。在启动之前，必须执行一些常见的操作任务，例如检查设备的连接状态、确保正确配置设备凭证以及根据设备的当前状态查询设备，以保证您的系统具有对应用程序进行故障排除所需的可见性。

AWS IoT Device Defender 是一项完全托管的服务，可审核您的设备队列、检测异常设备行为、向您发出安全问题警报，并帮助您调查和缓解常见的 IoT 安全问题。

AWS IoT Device Management 简化了大规模 IoT 设备的组织、监控和管理。在规模较大的情况下，客户需要管理多个物理位置的设备队列。AWS IoT Device Management 使您能够对设备进行分组，从而简化管理。您还可以利用设备管理队列索引，启用设备当前状态的实时搜索索引。在确定必须更新哪些目标设备时，设备组和队列索引可以与无线更新 (OTA) 结合使用。

用户应用程序

除了托管应用程序之外，其他内部和外部系统需利用 IoT 数据的不同部分来构建不同的应用程序。为了支持最终用户视图、业务操作控制面板以及您随着时间的推移构建的其他全新应用程序，您还需要用到其他几种技术，来从连接层和提取层接收所需的信息，并将其格式化以供其他系统使用。

数据库服务 – NoSQL 和 SQL

尽管数据湖可以充当您所有未格式化的 IoT 生成数据的着陆区，但要支持基于 IoT 数据的所有格式化视图，您需要用结构化和半结构化数据存储来补充数据湖。为此，您应该同时使用 NoSQL 和 SQL 数据库。这些类型的数据库使您可以为应用程序的不同最终用户创建 IoT 数据的不同视图。

Amazon DynamoDB 是一项用于 IoT 数据的快速灵活的 NoSQL 数据库服务。对于 IoT 应用程序，客户通常需要具有可靠性能和吞吐容量自动扩展能力的灵活数据模型。

借助 **Amazon Aurora**，您的 IoT 架构可以将结构化数据存储在性能出众且具成本效益的开源数据库中。当其他 IoT 应用程序需要访问您的数据以进行预定义的 SQL 查询时，关系数据库为您提供了另一种机制，用于将提取层的设备流与最终的业务应用程序（需要对数据的不同部分执行操作）分离。

计算服务

通常，在生成、提取或使用/实现数据时，IoT 工作负载需要执行应用程序代码。只要需要执行计算代码，无服务器计算就是一种极具成本效益的选择。无论是从边缘到核心，还是从核心到应用程序和分析，您都可以利用无服务器计算。

AWS Lambda 允许您运行代码，而无需预置或管理服务器。由于 IoT 工作负载的数据提取规模庞大，AWS Lambda 非常适合在托管平台上运行无状态、事件驱动的 IoT 应用程序。

一般设计原则

架构完善的框架定义了如下一系列设计原则，以促进良好的云端 IoT 设计：

- **将提取从处理中分离出来**：在 IoT 应用程序中，提取层必须是一个高度可扩展的平台，必须能够处理高速率的流式设备数据。通过使用队列、缓冲区和消息传递服务，将高速率的提取从应用程序的处理部分中分离出来，您的 IoT 应用程序即可制定多项决策而不会影响设备，例如其处理数据的频率或关注的数据类型。
- **专为离线行为设计**：由于出现连接问题或配置错误等情况，设备离线的时间可能比预期的要长得多。您的嵌入式软件应该设计为可应对更长的离线连接时间，并在云端创建指标，以跟踪未在常规时间段内通信的设备。
- **在边缘设计精益数据并在云中充实这些数据**：鉴于 IoT 设备的局限性，初始设备架构将针对物理设备上的存储以及从设备到 IoT 应用程序的有效传输进行优化。因此，未格式化的设备数据通常不会使用可通过云端推断出的静态应用程序信息来充实。出于这些原因，在将数据提取到您的应用程序中时，您应该优先使用人类可读的属性充实数据，反序列化或扩展设备已序列化的任何字段，然后针对为支持您的应用程序读取要求而优化的数据存储中的数据进行格式化。

- **处理个性化**：通过 Wi-Fi 连接到边缘或云的设备必须接收接入点名称和网络密码，并将此作为设备设置的第一步。由于此类数据较为敏感且特定于站点，或者需要从云端获取但设备尚未连接，因此在制造过程中通常无法将其写入设备。这些因素经常使个性化数据不同于概念上处于上游的设备客户端证书和私钥，以及概念上处于下游的云端提供的固件和配置更新。支持个性化可能会影响设计和制造，因为这可能意味着设备本身需要用于直接数据输入的用户界面，或者需要提供智能手机应用程序以将设备连接至本地网络。
- **确保设备定期发送状态检查**：即使设备长时间处于离线状态，也应确保设备固件包含应用程序逻辑，并且逻辑中设置了按固定时间间隔将设备状态信息发送到 IoT 应用程序。设备必须是积极的参与者，以确保您的应用程序具有适当的可见性。发送此类定期发生的 IoT 消息可确保您的 IoT 应用程序能获得设备整体状态的更新视图，并且可以创建适用于设备其预期无法通信的时间段的流程。

场景

本节介绍与 IoT 应用程序相关的常见场景，重点介绍每种场景如何影响 IoT 工作负载的架构。这些示例并不详尽，但包含了 IoT 的一些常见模式。我们提供了每种场景的背景信息、系统设计的一般注意事项以及各种场景应如何实施的参考架构。

设备预置

在 IoT 领域，设备预置涉及到多个连续的步骤。最重要的方面是，必须为每个设备赋予唯一身份，然后由 IoT 应用程序使用该身份进行身份验证。

因此，预置设备的第一步是安装设备身份。您在设备设计和制造过程中做出的决定将确定设备在送到客户手中时是否具有可用于生产的固件映像和/或唯一的客户凭证。您的决定将确定在安装生产设备身份之前是否还必须执行其他预置时步骤。

在 IoT 中为您的应用程序使用 X.509 客户端证书 – 与静态密码相比，这些证书要更加安全，更易于大规模管理。在 AWS IoT Core 中，设备使用其证书以及一个唯一的事物标识符进行注册。随后，已注册的设备将与一项 IoT 策略相关联。IoT 策略使您能够为每个设备创建精细权限。精细权限可确保只有一台设备具有与其自己的 MQTT 主题和消息进行交互的权限。

该注册过程可确保将设备识别为 IoT 资产，并且可以通过 AWS IoT 及 AWS 生态系统的其余部分使用其生成的数据。要预置设备，您必须启用自动注册并将预置模板或 AWS Lambda 函数与初始设备预置事件相关联。

此注册机制依赖于设备在预置期间（可能在制造期间或制造之后发生）接收唯一证书，该证书用于向 IoT 应用程序（在本例中为 AWS IoT）进行身份验证。这种方法的一个优点是，该设备可以转移到另一个实体，然后重新预置，从而允许使用新所有者的 AWS IoT 账户详细信息重复注册过程。

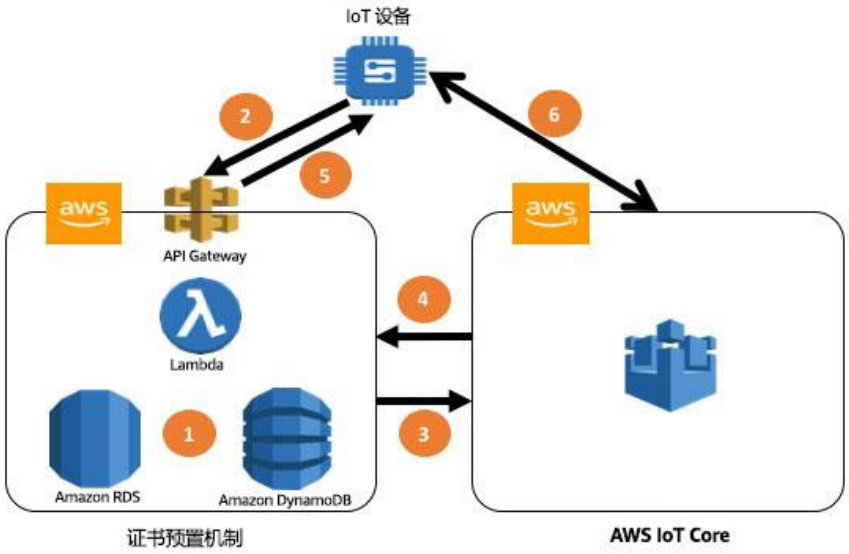

图 1：注册流程

1. 在数据库中设置制造设备标识符。
2. 设备连接到 API 网关并从 CPM 请求注册。请求经过验证。
3. Lambda 向您的私人证书颁发机构 (CA) 请求 X.509 证书。
4. 您的预置系统向 AWS IoT Core 注册您的 CA。
5. API 网关将设备证书传递到设备。
6. 设备启动向 AWS IoT Core 注册的工作流程。

设备遥测

在许多使用案例（例如工业物联网）中，IoT 的价值在于收集有关机器性能的遥测数据。举例来说，这些数据可用于启用预测性维护，从而防止代价高昂且不可预见的设备故障。遥测数据必须从机器中收集并上传到 IoT 应用程序中。发送遥测数据的另一个好处是您的云端应用程序可以使用这些数据执行分析，并解释您随着时间的推移可能对固件做出的优化。

所收集到的遥测数据是只读的，并且会传输到 IoT 应用程序。由于遥测数据是被动的，因此请确保遥测消息的 MQTT 主题与任何 IoT 命令相关主题无重叠。例如，遥测主题可以是 data/device/sensortype，此时任何以“data”开头的 MQTT 主题都会被视为遥测主题。

从逻辑角度来看，我们定义了几种捕获设备数据遥测并与之交互的场景。

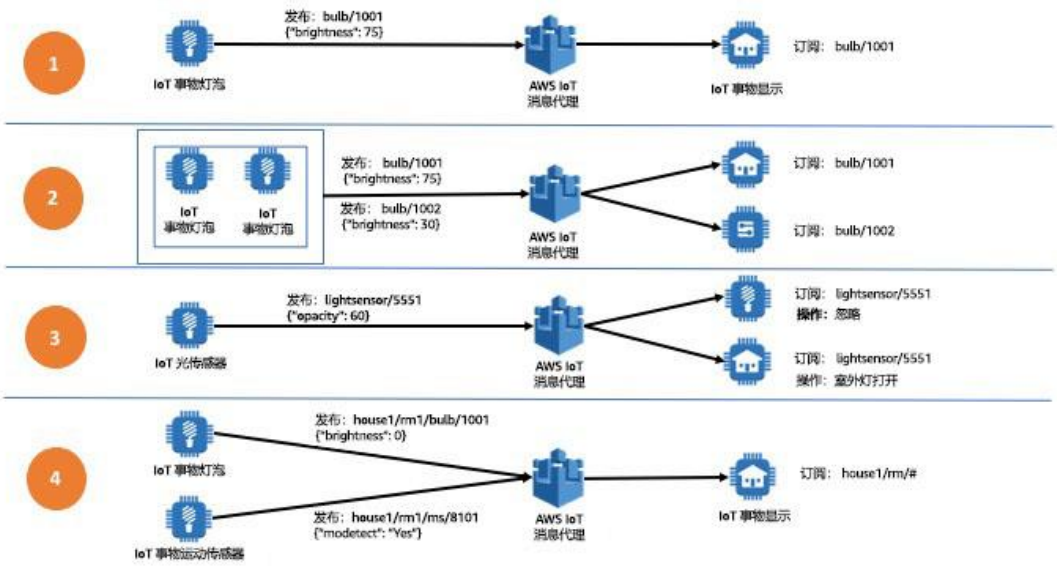

图 2：捕获遥测数据的选项

1. 一个发布主题和一个订阅者。例如，某个智能灯泡将其亮度水平发布到只有一个应用程序可以订阅的单个主题。
2. 一个带有多个变量的发布主题和一个订阅者。例如，一系列智能灯泡在类似但不同的主题上发布其亮度。每个订阅者都可以侦听唯一的发布消息。

3. 单一发布主题和多个订阅者。在这种情况下，光线传感器将其值发布到房屋中所有灯泡均订阅的一个主题。
4. 多个发布主题和单个订阅者。例如，一系列灯泡与运动传感器。智能家居系统订阅所有灯泡主题，包括运动传感器，并创建亮度和运动传感器数据的复合视图。

设备命令

构建 IoT 应用程序时，您需要能够通过命令与设备进行远程交互。工业领域的一个示例是使用远程命令向一台设备请求特定数据。智能家居行业中的一个示例应用是使用远程命令远程调度警报系统。

借助 AWS IoT Core，您可以使用 MQTT 主题或 AWS IoT Device Shadow 来实施命令，以将命令发送到设备并在设备执行命令时接收确认。使用 MQTT 主题上的 Device Shadow 实施命令。在跟踪请求来源、用于管理冲突解决的版本号以及将命令存储到云端（防止设备离线状态并且无法接收到其发出的命令）时，与使用标准 MQTT 主题（例如 clientToken）相比，Device Shadow 有多方面的优势。如果即使设备当前未联网，也需要在云端持久存储命令，通常会使用设备影子。当设备重新联网后，设备会请求最新的影子信息并执行命令。

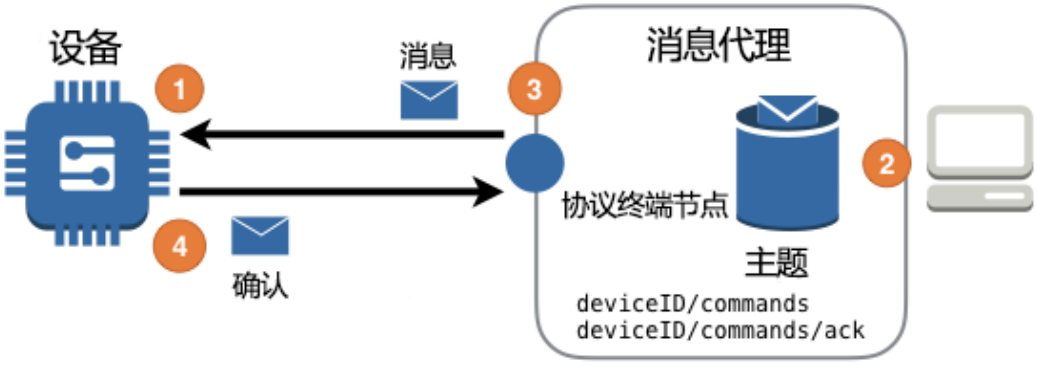

图 3：使用消息代理将命令发送到设备。

AWS IoT Device Shadow 服务

在 AWS IoT Core 中使用 Device Shadow 服务的 IoT 解决方案，以可靠、可扩展且直接的方式管理命令请求。Device Shadow 服务遵循一种规范性方法来管理与设备相关的状态以及传达状态更

改的方式。这种方法描述了 Device Shadow 服务如何使用 JSON 文档存储设备的当前状态、理想的将来状态以及当前状态与理想状态之间的差异。

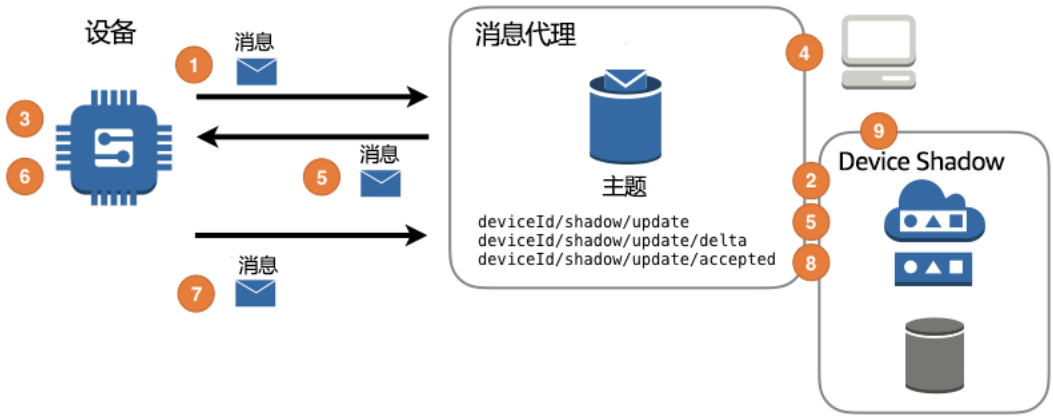

图 4：将 Device Shadow 服务与设备结合使用。

1. 设备通过将该状态作为消息发布到更新主题 `deviceID/shadow/update` 来报告初始设备状态。
2. Device Shadow 从主题中读取消息，并将设备状态记录在持久数据存储中。
3. 设备会订阅设备相关状态更改消息将送达的增量消息传递主题 `deviceId/shadow/update/delta`。
4. 解决方案的一个组件向主题 `deviceID/shadow/update` 发布理想状态消息，跟踪此设备的 Device Shadow 将理想设备状态记录在持久数据存储中。
5. Device Shadow 将增量消息发布到主题 `deviceId/shadow/update/delta`，消息代理将消息发送到设备。
6. 设备接收增量消息并执行所需的状态更改。
7. 设备会向更新主题 `deviceID/shadow/update` 发布反映新状态的确认消息，跟踪此设备的 Device Shadow 会将新状态记录在持久数据存储中。
8. Device Shadow 将消息发布到主题 `deviceId/shadow/update/accepted`。
9. 解决方案的组件现在可以从 Device Shadow 请求更新状态。

固件更新

所有 IoT 解决方案都必须允许设备固件更新。支持无需人为干预的固件升级对于安全性、可扩展性和新功能的交付至关重要。

AWS IoT Device Management 为您提供一种安全简便的方法来管理 IoT 部署，包括执行固件更新及跟踪固件更新的状态。AWS IoT Device Management 将 MQTT 协议与 AWS IoT 消息代理和 AWS IoT 作业结合使用，以将固件更新命令发送至设备，并不断接收这些固件更新的状态。

IoT 解决方案必须使用下图所示的 AWS IoT 作业实施固件更新，才能提供此功能。

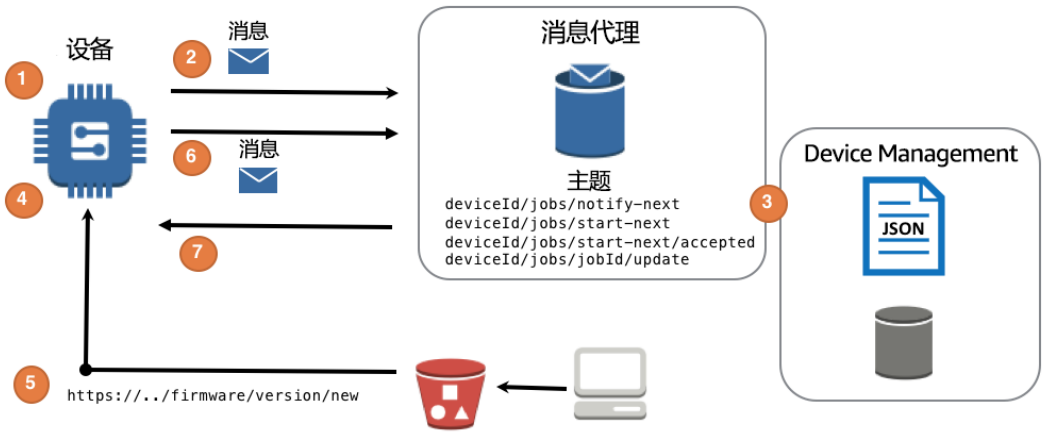

图 5：更新设备的固件。

1. 设备订阅 IoT 作业通知消息将送达的 IoT 作业通知主题 `deviceId/jobs/notify-next`。
2. 设备将消息发布到 `deviceId/jobs/start-next` 以开始下一个作业，并获取下一个作业、其作业文档以及其他详细信息，包括 statusDetails 中保存的任何状态。
3. AWS IoT 作业服务检索特定设备的下一个作业文档，并将此文档发送到所订阅的主题 `deviceId/jobs/start-next/accepted`。
4. 设备使用 `deviceId/jobs/jobId/update MQTT` 主题执行作业文档指定的操作，以报告作业进度。

5. 在升级过程中，设备会使用适用于 Amazon S3 的预签名 URL 下载固件。固件上传到 Amazon S3 时，会使用代码签名进行签名。通过对固件进行代码签名，终端设备可以在安装固件之前验证固件的真实性。Amazon FreeRTOS 设备可以直接通过 MQTT 下载固件映像，因此不再需要单独的 HTTPS 连接。
6. 设备将更新状态消息发布到作业主题 `deviceId/jobs/jobId/update` 以报告成功或失败。
7. 由于此作业的执行状态已更改为最终状态，因此下一个可用于执行的 IoT 作业（如果有）将发生更改。

架构完善的框架的支柱

本节将介绍每个支柱，包括定义、最佳实践、问题、注意事项和为 AWS IoT 构建解决方案时涉及到的基本 AWS 服务。

卓越运营支柱

卓越运营支柱包括用于管理生产工作负载的运营实践和程序。卓越运营包括计划变更的执行方式以及对意外运营事件的响应。变更执行和响应应该实现自动化。卓越运营的所有流程和程序都必须形成文档、执行测试并定期进行审查。

设计原则

除了架构完善的框架的总体卓越运营设计原则之外，还有五项针对云端 IoT 的卓越运营设计原则：

- **规划设备预置**：设计设备预置过程，以在安全位置创建初始设备身份。实施一个公钥基础设施 (PKI)，负责向 IoT 设备分发唯一证书。如前文所述，选择使用具有预生成的私有密钥和证书的加密硬件能够消除运行 PKI 产生的运营成本。否则，可以在制造过程中或在设备引导过程中使用硬件安全模块 (HSM) 离线完成 PKI。请使用可以在云中管理证书颁发机构 (CA) 和 HSM 的技术。

- **实施设备引导**：支持技术人员（在工业领域）或用户（在消费者领域）进行个性化设置的设备也支持预置。例如，智能手机应用程序可以通过蓝牙 LE 与设备交互，并通过 Wi-Fi 与云交互。您必须为设备设计相应功能，使设备能够使用全局分布式引导启动 API 以编程方式更新配置信息。引导设计确保您能够以编程方式通过云发送设备的新配置设置。这些更改应包括设置，例如与哪个 IoT 终端节点进行通信、发送设备总体状态的频率以及更新后的安全设置（例如服务器证书）。引导过程超出了最初的预置范围，并且提供一种以编程方式通过云更新设备配置的方式，在设备运营中发挥了重要作用。
- **记录设备通信模式**：在 IoT 应用程序中，设备行为在硬件级别手动记录。在云中，运营团队必须制定出将单个设备部署到设备队列之后，设备行为的扩展方式。云工程师应检查设备通信模式，推断设备数据的预期总入站和出站流量，确定云中支持整个设备队列所需的预期基础设施情况。在运营规划期间，应使用设备和云端指标来衡量这些模式，以确保满足系统中的预期使用模式。
- **实施无线更新 (OTA)**：为了从硬件的长期投资中受益，您必须能够持续更新设备上的固件以使用新功能。在云中，您可以应用可靠的固件更新过程，该过程支持针对特定设备进行固件更新、随时间应用更改、跟踪更新的成功和失败状态以及回滚或停止基于 KPI 的固件更改。
- **对实体资产实施功能测试**：IoT 硬件和固件必须经过严格的测试才能进行现场部署。验收和功能测试对于生产准备至关重要。功能测试的目标是通过将硬件组件、嵌入式固件和设备应用程序软件放在苛刻的测试场景（例如间歇性连接、不良连接或外围传感器故障）中运行，并对这些场景中的硬件性能进行性能分析。这些测试可确保您的 IoT 设备在部署时能够按预期运行。

定义

在云中实现卓越运营有三个方面的最佳实践：

1. 准备
2. 运营
3. 演进

除了架构完善的框架所涵盖的流程、运行手册和实际试用相关内容之外，您还应该了解一些特定领域，以在 IoT 应用程序中推动实现卓越运营。

最佳实践

准备

对于 IoT 应用程序，需要在各种环境中采购、预置、测试和部署硬件，这意味着必须扩大卓越运营的准备范围，以涵盖主要在物理设备上运行（而不会在云中运行）的部署。必须定义运营指标以便衡量和改进业务成果，然后确定设备是否应生成任何指标并将其发送到您的 IoT 应用程序。您还必须通过创建简化的功能测试流程来规划卓越运营，使您能够模拟设备在各种环境中的行为。

请您务必询问这些重要事宜：如何确保 IoT 工作负载能够灵活应对故障；设备如何能够在没有人工干预的情况下从问题中自行恢复；基于云的 IoT 应用程序将如何扩展以满足互联硬件不断增长的负载需求。

使用 IoT 平台时，您可以使用其他组件/工具来处理 IoT 运营。您可以使用这些工具包含的服务来监控和检查设备行为、捕获连接性指标、使用唯一身份预置设备以及对设备数据进行长期分析。

IOTOPS 1.哪些因素决定了运营重点?

IOTOPS 2.如何确保为支持 IoT 工作负载设备运营做好准备?

IOTOPS 3.如何确保新预置的设备满足运营所需的先决条件?

IoT 和数据中心具有相似的逻辑安全性，因为两者都主要涉及机器对机器身份验证。但是，它们的不同之处在于，IoT 设备经常会部署到无法假定为从物理层面来讲安全的环境中。IoT 应用程序通常还需要敏感数据才能遍历互联网。基于这些考虑，拥有符合以下条件的架构对您而言至关重要：能够确定设备安全获得身份的方式，能够持续证明设备身份，已植入适当级别的元数据，已经过整理和分类可进行监控，并已使用正确的权限集启用。

对于成功且可扩展的 IoT 应用程序，管理流程应实现自动化、以数据为依据并基于先前、当前和预期的设备行为。IoT 应用程序必须支持增量部署和回滚策略。将此纳入您的运营效率计划后，您就做好了启动支持容错且高效 IoT 应用程序的硬件准备。

在 AWS IoT 中，有多种功能可让您将由 CA 签名的单个设备身份预置到云。该过程包括为设备预置身份，然后使用即时预置 (JITP)、即时注册 (JITR) 或自带证书 (BYOC) 安全地将设备证书注册到云中。借助 Route 53、Amazon API Gateway、Lambda 和 DynamoDB 等 AWS 服务，创建一个简单 API 接口，以通过设备引导扩展预置过程。

运营

在 IoT 中，运营状况不仅包括云应用程序的运营状况，还扩展到了对设备（这些设备是您的应用程序的组成部分，但被远程部署在可能难以或无法进行本地故障排除的位置）进行衡量、监控、故障排除和修复的能力。在设计和实施时请务必考虑对远程运营的要求，确保您能够对这些远程设备发送的指标进行检查、分析并作出响应。

在 IoT 中，您必须为设备建立适当的行为基线指标，做到能够聚合并推断设备间发生的问题，同时具备可靠的修复计划（该计划既要在云中执行，也要纳入设备固件运维）。您必须实施各种设备模拟 canary，它们会持续针对您的生产系统直接测试常见设备交互。设备 canary 有助于缩小潜在区域的范围，以便在出现不满足运营指标的情况时进行调查。当 canary 指标低于预期的 SLA 时，您可以使用设备 canary 发出抢先式警报。

在 AWS 中，您可以在 AWS IoT Core 的设备注册表中为每个物理设备创建一个 AWS IoT 事物。通过在注册表中创建事物，您可以将元数据关联到设备、对设备进行分组以及为设备配置安全权限。您应该使用 AWS IoT 事物在事物注册表中存储静态数据，在事物的相关设备影子中存储动态设备数据。设备的影子是一个 JSON 文档，用于存储和检索设备的状态信息。

在运营过程中，除了在设备注册表中创建设备的虚拟表示之外，您还必须创建事物类型，这些事物类型会封装定义 IoT 设备的类似静态属性。事物类型类似于设备的产品分类。事物、事物类型和设备影子的组合可以作为您存储未来用于 IoT 运营的重要元数据的第一个入口点。

在 AWS IoT 中，您可以使用事物组来按类别管理设备。组中可以包含其他组，这意味着您可以构建层次结构。 在 IoT 应用程序中建立组织结构后，您可以按设备组快速识别相关设备并采取相应措施。利用云，您可以实现基于业务逻辑和设备生命周期自动向组中添加或从中删除设备。

在 IoT 中，您的设备会创建遥测或诊断消息，这些消息不会存储在注册表或设备的影子中。相反，这些消息会传递到使用多个 MQTT 主题的 AWS IoT。为使这些数据具有行动价值，请使用 AWS IoT 规则引擎将错误消息路由到您的自动修复流程，并将诊断信息添加到 IoT 消息中。下面的示例展示了如何将包含错误状态代码的消息路由到自定义工作流程。规则引擎会检查消息的状

态，如果遇到错误消息，规则引擎将启动 Step Function 工作流程以基于错误消息详情负载来修复设备。

```
{
    "sql": "SELECT * FROM 'command/iot/response WHERE code =
'eror'",
    "ruleDisabled": false,
    "description": "Error Handling Workflow",
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
        "stepFunctions": {
            "executionNamePrefix": "errorExecution",
            "stateMachineName": "errorStateMachine",
            "roleArn":
"arn:aws:iam::123456789012:role/aws_iot_step_functions"
        }
    }]
}
```

为了获取云应用程序方面的运营见解，请为从 AWS IoT Core 的设备代理收集的所有指标生成控制面板。您可以通过 CloudWatch 指标获得这些指标。此外，CloudWatch Logs 包含出入站消息总数、连接成功和错误等信息。

为了扩大您的生产设备部署，请在 Amazon Elastic Compute Cloud (Amazon EC2) 上实施 IoT 模拟，将其作为跨多个 AWS 区域的设备 canary。这些设备 canary 负责为您的多个业务使用案例建立镜像，使用案例包括模拟错误条件（例如长时间运行的事务）、发送遥测和实施控制操作等。设备模拟框架必须输出多种指标（包括但不限于成功、错误、延迟和设备排序），然后将所有指标传输到您的运营系统。

除自定义控制面板外，AWS IoT 还在 Thing Registry 和 Device Shadow 服务的支持下，通过搜索功能（例如 AWS IoT 队列索引）提供了队列级和设备级见解。无论这些问题发生在设备级还是队列级，您都可以在整个队列中进行搜索，从而减轻了诊断 IoT 问题产生的运营开销。

演进

IOTOPS 4.如何在对下游 IoT 设备影响最小的情况下实现 IoT 应用程序演进?

IoT 解决方案通常会面临设备功耗低、位置偏远、带宽较低和网络连接不稳定等问题。这些因素中的任何一项都会为通信带来挑战（包括升级固件）。因此，为了最大程度降低对下游设备和运营的影响，整合并实施 IoT 更新过程非常重要。除了减少对下游的影响之外，设备还必须能够灵活应对当地环境中存在的常见挑战，例如网络连接不稳定和断电。请将分组后的 IoT 设备组合在一起使用，以在一段时间内进行部署和重大固件升级。监控设备现场更新时的行为，只有当一定比例的设备成功升级后才能继续。

请使用 AWS IoT Device Management 来创建设备的部署组并将无线更新 (OTA) 传送到特定的设备组。在升级期间，继续收集所有 CloudWatch Logs、遥测和 IoT 设备作业消息，并将这些信息与 KPI（用于衡量整体应用程序运行状况和长期运行的 canary 的性能）结合在一起。

在固件更新之前和之后，请与跨业务部门的参与者进行运营指标的回顾分析，以确定改进的机会和方法。AWS IoT Analytics 和 AWS IoT Device Defender 等服务用于跟踪设备整体行为的异常情况以及衡量性能差异（性能差异可能表明更新的固件存在问题）。

关键 AWS 服务

您可以使用多种服务来推动实现 IoT 应用程序的卓越运营。AWS 设备认证计划能够帮助您选择针对 AWS IoT 互操作性进行了设计和测试的硬件组件。使用合格的硬件有助于缩短上市时间并减少运营摩擦。AWS IoT Core 提供了用于管理设备初始注册的功能。AWS IoT Device Management 可减少执行队列范围操作（例如设备分组和搜索）产生的运营开销。此外，Amazon CloudWatch 还用于监控 IoT 指标、收集日志、生成警报和触发响应。其他支持卓越运营三个领域的服务和功能如下：

- **准备**：**AWS IoT Core** 支持现场预置设备和设备注册，包括使用即时预置、即时注册或自带证书来注册设备身份。然后，可以使用 Thing Registry 和 Device Shadow 将设备与其元数据和设备状态相关联。
- **运营**：**AWS IoT 事物组和队列索引**让您可以快速为设备开发组织结构，并搜索设备的当前元数据以执行重复性设备运营。借助 Amazon CloudWatch，您可以监控设备和应用程序的运行状况。

- **响应**：通过 **AWS IoT 作业**，您可以主动将更新（例如固件更新或设备配置）推送到一个或多个设备。通过 AWS IoT 规则引擎，您可以检查 AWS IoT Core 收到的 IoT 消息，并立即以最精细的方式响应数据。借助 AWS IoT Analytics 和 AWS IoT Device Defender，您可以基于 AWS IoT Analytics 的实时分析以及 Device Defender 的实时安全性和数据阈值来主动触发通知或修复。

安全性支柱

安全性支柱包括在提供业务价值的同时保护信息、系统和资产的能力。

设计原则

除了架构完善的框架的总体安全设计原则外，另有针对 IoT 安全性的特定设计原则：

- **从整体出发管理设备安全性生命周期**：数据安全性要从设计阶段开始，直到硬件和数据弃用和销毁为止。为了保持竞争优势并维持客户信任，对 IoT 解决方案的安全性生命周期采取端到端方法至关重要。
- **确保最低特权权限**：设备都应具有精细访问权限，用于限制设备可用于通信的主题。通过限制访问，可以有效降低遭到受影响的设备对其他任何设备造成影响的概率。
- **静态保护设备凭证**：设备应通过适当的机制（例如专用加密元素或安全闪存）实现凭证信息的静态安全存储。
- **实施设备身份生命周期管理**：设备从创建时就具有设备身份，这一身份在设备的整个生命周期中始终存在。设计完善的身份系统会跟踪设备身份和身份有效期，并随时间推移主动扩展或撤销 IoT 权限。
- **从全局视角关注数据安全性**：在 IoT 部署中，大量远程部署的设备容易成为不法分子的攻击面，从而导致数据失窃或私密性受到威胁。请使用开放可信技术供应商标准等模型，系统性地审查供应链和解决方案设计中的风险，然后采取适当的缓解措施。

定义

在云中实现安全性包括五个方面的最佳实践：

1. Identity and Access Management (IAM)
2. 检测性控制
3. 基础设施保护
4. 数据保护
5. 事件响应

基础设施和数据保护包括 IoT 设备硬件以及端到端解决方案。IoT 实施要求扩展您的安全模型，以确保设备实施硬件安全最佳实践，同时确保您的 IoT 应用程序遵循各个方面（例如适当范围的设备权限和检测性控制）的安全性最佳实践。

安全性支柱侧重于保护信息和系统。关键主题包括数据的机密性和完整性，识别和管理哪些人员可以通过权限管理进行哪些操作，保护系统以及建立控制措施来检测安全事件。

最佳实践

Identity and Access Management (IAM)

IoT 设备容易成为攻击目标，因为这些设备通过可信身份预置，可存储或访问战略客户或业务数据（例如固件本身），可以通过互联网远程访问，并且容易遭受直接的物理篡改。为了防止未经授权的访问，您需要始终从一开始就在设备一级采取安全措施。从硬件角度考虑，您可以采用多种机制来减少攻击面以防设备上的敏感信息被篡改，例如：

- 硬件加密模块
- 软件支持的解决方案，包括安全闪存
- 无法克隆的物理功能模块
- 采用最新的加密库和标准，包括 PKCS #11 和 TLS 1.2

为了保护设备硬件，您需要通过实施解决方案，确保私钥和敏感身份对于设备具有唯一性，并且仅存储在设备中的安全硬件位置。请实施基于硬件或软件的模块，这些模块可以安全地存储和管理对用于与 AWS IoT 进行通信的私钥的访问权限。除了硬件安全性之外，您还必须为 IoT 设备赋予有效身份，用于 IoT 应用程序中的身份验证和授权。

在设备的生命周期内，您需要能够管理证书的续订和撤销。要处理设备上证书信息的任何更改，您必须首先具有现场更新设备的能力。能够对硬件执行固件更新是实现架构完善的 IoT 应用程序的重要基础。通过 OTA 更新，您可以在证书到期之前安全地轮换设备证书（包括证书颁发机构）。

IOTSEC 1.如何安全地存储设备证书和私钥?

IOTSEC 2.如何将 AWS IoT 身份关联到设备?

例如，使用 AWS IoT，您首先要预置 X.509 证书，然后分别创建用于连接到 IoT、发布和订阅消息以及接收更新的 IoT 权限。身份和权限的分离让您可以灵活地管理设备安全性。在权限配置期间，您可以通过创建 IoT 策略（限制每个设备对 MQTT 操作的访问），来确保所有设备都具有相应的身份级别和访问控制级别。

确保每个设备在 AWS IoT 中都有唯一的 X.509 证书，设备不得共享证书（一个证书对应一个设备规则）。除了做到每台设备使用一个证书之外，在使用 AWS IoT 时，每个设备在 IoT 注册表中还必须对应唯一的事物，事物名称作为 MQTT 连接的 MQTT ClientID 的基础。

通过在 AWS IoT Core 中创建这种关联（单个证书与其对应的证书进行配对），您可以确保无法通过遭到攻击的证书轻易担任其他设备的身份。MQTT ClientID 和事物名称匹配还可以降低故障排除和修复的工作难度，因为您可以将任何 ClientID 日志消息关联到与该特定通信相关联的事物。

为了支持设备身份更新，请使用 AWS IoT 作业，这是一个托管平台，用于将 OTA 通信和二进制文件分发到您的设备。AWS IoT 作业可用于定义一组远程操作，这些操作被发送到一个或多个连接到 AWS IoT 的设备并在这些设备上执行。默认情况下，AWS IoT 作业集成了几种最佳实践，包括双向身份验证和授权、设备跟踪更新进度以及适用于给定更新的队列级指标。

请启用 AWS IoT Device Defender 审计以跟踪设备配置、设备策略，并自动检查证书是否过期。例如，Device Defender 可以按计划运行审计，还可以在证书过期时发出通知。配合接收已撤销证书或临近到期证书的相关通知，您可以设置自动执行 OTA 计划以主动轮换证书。

IOTSEC 3.如何对用户进行身份验证和授权他们访问您的 IoT 应用程序?

尽管许多应用程序都侧重于 IoT 中的事物，但几乎在所有 IoT 行业中，依然存在需要能够与设备进行通信并从设备接收通知的人为要素。例如，消费类 IoT 通常要求用户通过将设备关联到在线账户来完成设备注册。工业 IoT 通常需要能够近乎实时地分析硬件遥测。无论哪种情况，都请务必确定您的应用程序对需要与特定设备进行交互的用户进行识别、身份验证和授权的方式。

从身份入手，控制用户对 IoT 资产的访问权限。您的 IoT 应用程序必须拥有存储（通常是数据库），用于跟踪用户的身份以及用户如何使用该身份进行身份验证。身份存储可以包括可在授权时使用的其他用户属性（例如用户组成员身份）。

IoT 设备遥测数据是一种可保护资产。通过这样处理，您可以控制每个用户的访问权限并审计单个用户的交互。

使用 AWS 来验证用户身份并为其授予 IoT 应用程序权限时，您可以通过多种方法实施身份存储以及该存储对用户属性的维护方式。对于您自己的应用程序，请使用 Amazon Cognito 作为身份存储。Amazon Cognito 提供了一种用于表达身份和验证用户身份的标准机制，您的应用程序和其他 AWS 服务可以直接使用这种方式来做出授权决策。使用 AWS IoT 时，您可以从多种身份和授权服务（包括 Amazon Cognito 身份池、AWS IoT 策略和 AWS IoT 自定义授权方）中进行选择。

为了向您的用户提供遥测的分离视图，请使用移动服务，例如 AWS AppSync 或 Amazon API Gateway。借助这两项 AWS 服务，您可以创建抽象层，将 IoT 数据流与用户的设备数据通知流分离。例如，您可以在中介数据存储中为外部用户创建单独的数据视图。通过 Amazon DynamoDB 或 Amazon ElasticSearch Service，您可以使用 AWS AppSync 接收特定于用户的通知（仅基于您的中介存储内允许的数据）。除了通过 AWS AppSync 使用外部数据存储之外，您还可以定义特定于用户的通知主题，这些主题可用于将特定 IoT 数据视图推送给外部用户。

如果外部用户需要直接与 AWS IoT 终端节点进行通信，请确保用户身份符合以下两种情况：关联至已授权 Amazon Cognito 角色和精细 IoT 策略的已授权 Amazon Cognito 联合身份，或者使用授权由您自己的授权服务管理的 AWS IoT 自定义授权方。无论采用哪种方法，都请将精细策略与每个用户相关联，以限制用户可以连接、发布、订阅和接收来自有关 MQTT 通信的消息的内容。

IOTSEC 4.如何确保将最低权限应用于与您的 IoT 应用程序进行通信的主体?

在注册设备并为其建立身份之后，您可能需要植入监控、指标、遥测或命令和控制所需的其他设备信息。您需要为每个资源分配各自的访问控制规则。通过减少设备或用户可以对您的应用程序执行的操作，以及确保对每个资源进行了单独保护，可以限制无意间使用任何单个身份或资源可能造成的影响。

在 AWS IoT 中，通过使用 IoT 注册表中的一组固定命名约定来创建精细访问权限。第一种约定是使用设备的唯一标识符作为设备的 MQTT ClientID 和 AWS IoT 事物名称。通过在所有这些位置使用相同的唯一标识符，您可以轻松创建一组初始的 IoT 权限，并可以使用 AWS IoT 事物策略变量将这些权限应用于所有设备。第二种命名约定是将设备的唯一标识符嵌入设备证书中。继续使用这种方法，将唯一标识符作为 CommonName 存储在证书的主题名称中，以便使用证书策略变量将 IoT 权限绑定到每个唯一设备证书。

通过使用策略变量，您可以创建一些 IoT 策略，这些策略可以在保持最低权限的同时应用于所有设备证书。例如，以下 IoT 策略会对所有设备进行限制，使其仅使用设备的唯一标识符（存储在通用名称中）作为其 MQTT ClientID 进行连接，并且仅在证书已附加到设备时才可以连接。此策略还会限制设备仅能在一个影子上发布：

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["arn:aws:iot:us-east-
1:123456789012:client/${iot:Certificate.Subject.CommonName}"],
        "Condition":{
            "Bool":{
                "iot:Connection.Thing.IsAttached":["true"]
            }
        }
    },
    {
        "Effect":"Allow",
        "Action":["iot:Publish"],
        "Resource":["arn:aws:iot:us-east-
1:123456789012:topic/$aws/things/${iot:Connection.Thing.ThingName}/
shadow/update"]
    }
    ]
```

```
}
```

使用 AttachThingPrincipal 将设备身份（证书或 Amazon Cognito 联合身份）附加到 AWS IoT 注册表中的事物。

尽管这些场景适用于单个设备与其主题集和设备影子集进行通信，但在某些情况下，单个设备需要根据其他设备的状态或主题作出响应。例如，在工业环境中运营边缘设备，创建家庭网关来管理家庭中的协调自动化，或者允许用户根据其特定角色访问其他组的设备。对于这些使用案例，您可以利用已知实体（例如组标识符或边缘网关的身份）作为与网关通信的所有设备的前缀。通过让所有终端节点设备使用相同的前缀，您可以在 IoT 策略中使用通配符“*”。这种方法实现了 MQTT 主题安全性和可管理性的平衡。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect":"Allow",
        "Action":["iot:Publish"],
        "Resource":["arn:aws:iot:us-east-
1:123456789012:topic/$aws/things/edgegateway123-*/shadow/update"]
    }
    ]
}
```

在前面的示例中，IoT 运营商会将策略关联到具有标识符 edgegateway123 的边缘网关。然后，此策略中的权限会允许边缘设备发布到由边缘网关管理的其他设备影子。这可以通过将所有连接到网关的设备强制设置为使用以网关标识符为前缀的事物名称来实现。例如，下游运动传感器使用标识符 edgegateway123-motionsensor1，从而在限制权限的同时由边缘网关管理该传感器。

检测性控制

由于 IoT 应用程序中数据、指标和日志的规模，使得聚合和监控成为架构完善的 IoT 应用程序的重要组成部分。未经授权的用户会探查 IoT 应用程序中的错误，并试图利用单个设备来进一步访问其他设备、应用程序和云资源。为了运营整套 IoT 解决方案，您不仅需要管理单个设备的检测

性控制，还需要管理应用程序中整个设备队列的检测性控制。您需要启用多个级别的日志记录、监控和警报，以在设备级别以及整个队列级别检测问题。

在架构完善的 IoT 应用程序中，IoT 应用程序的每一层都会生成指标和日志。您的架构至少应具有与以下事项相关的指标和日志：物理设备、设备连接行为、每台设备的消息输入和输出速率、预置活动、授权尝试以及设备数据从一个应用程序到另一个应用程序的内部路由事件。

IOTSEC 5：如何分析跨云和设备的应用程序日志和指标?

在 AWS IoT 中，您可以使用 AWS IoT Device Defender、CloudWatch Logs 和 CloudWatch 指标实施检测性控制。AWS IoT Device Defender 处理与设备行为和设备连接行为有关的日志和指标。借助 AWS IoT Device Defender，您可以持续监控设备和 AWS IoT Core 的安全性指标，了解这些指标与您为每台设备定义的理想行为之间的差异。

当设备行为或连接行为偏离正常活动时，设置一组默认阈值。

通过 Amazon CloudWatch 指标、由 AWS IoT Core 生成的 Amazon CloudWatch Logs 和 Amazon GuardDuty 来增强 Device Defender 指标。这些服务级别日志不仅可以提供有关与 AWS IoT 平台服务以及 AWS IoT Core 协议使用相关的活动的重要见解，还可以提供有关 AWS 中运行的下游应用程序（端到端 IoT 应用程序的关键组件）的见解。建议您集中分析所有 Amazon CloudWatch Logs，以关联来自所有来源的日志信息。

IOTSEC 6：如何管理 IoT 应用程序中的无效身份?

对 IoT 应用程序而言，安全身份是设备信任和授权的重点。因此，能够集中管理无效身份（例如证书）至关重要。无效证书包括已撤销、已过期或已禁用的证书。作为架构完善的应用程序的一部分，您必须具有用于捕获所有无效证书的过程以及基于证书触发器状态的自动响应。除了能够获取无效证书事件外，您的设备还应具有与您的 IoT 平台建立安全通信的辅助手段。如上所述启用引导模式（在设备中使用两种身份）后，您可以创建可靠的回退机制来检测无效证书，并为设备或管理员提供一种机制来建立可信的安全通信以进行补救。

架构完善的 IoT 应用程序会建立证书撤销列表 (CRL)，用于跟踪所有已撤销的设备证书或证书颁发机构 (CA)。使用您自己的可信 CA 注册设备，并定期将您的 CRL 与 IoT 应用程序同步。您的 IoT 应用程序必须拒绝来自失效身份的连接。

借助 AWS，您不需要管理整个本地 PKI。使用 AWS Certificate Manager (ACM) 私有证书颁发机构在云中托管您的 CA。或者，您可以与 APN 合作伙伴合作，将预先配置好的安全元素添加到您的 IoT 设备硬件规范中。ACM 能够将撤销的证书导出为 S3 存储桶中的文件。该文件可用于针对 AWS IoT Core 以编程方式撤销证书。

证书的另一个状态是即将到期，但仍有效。客户端证书必须至少在设备的服务生命周期内有效。您的 IoT 应用程序可跟踪设备即将到期的情况，并执行 OTA 流程，将证书更新为稍后到期的新证书，同时记录有关为何需要进行证书轮换以进行审核的信息。

启用与证书和 CA 到期相关的 AWS IoT Device Defender 审核。Device Defender 会生成证书的审核日志，并将其设置为在 30 天内到期。使用此列表可在证书失效之前以编程方式更新设备。您还可以选择构建自己的到期存储以管理证书到期日期，并以编程方式查询、识别和触发 OTA 流程以更换或续订设备证书。

基础设施保护

设计时间是在设备和解决方案的整个生命周期内，考虑基础设施保护方面安全要求的理想阶段。将您的设备视为基础设施扩展后，您可以考虑整个设备生命周期对基础设施保护设计的影响。从成本的角度来看，在设计阶段进行更改比以后做出更改成本更低。从有效性的角度来看，在设计阶段实施的数据丢失缓解措施可能比后续改进的缓解措施更全面。因此，在设计阶段规划设备和解决方案安全生命周期可降低业务风险，并为在启动前执行前期基础设施安全分析提供了机会。

实现设备生命周期安全性的一种方法是通过供应链分析。例如，即使是规模得当的 IoT 设备制造商或解决方案集成商，其供应链也包含大量直接或间接供应商。要最大限度地延长解决方案的使用寿命并提升可靠性，请确保您收到的是原装组件。

软件也是供应链的一部分。设备的生产固件映像包括来自多种来源的驱动程序和库，包括硅树脂合作伙伴、开源聚合站点（如 GitHub 和 SourceForge）、以前的第一方产品以及内部工程开发的新代码。

要了解第一方固件和软件方面的下游维护和支持，您必须分析供应链中的每个软件提供商，以确定其是否提供支持以及如何提供补丁。这种分析对于互联设备特别重要：软件错误不可避免，并

会给您的客户带来风险，因为容易受到攻击的设备可能会遭到远程利用。您的 IoT 设备制造商或解决方案工程团队必须及时了解并修补错误，以降低这些风险。

IOTSEC 7.您如何审查供应商、签约制造商以及其他外包关系?

IOTSEC 8.您如何规划 IoT 设备的安全生命周期?

IOTSEC 9.您如何确保及时就第三方固件和软件组件中的安全错误发出通知?

虽然使用 AWS IoT 服务不需要管理云基础设施，但存在 AWS IoT Core 代表您与其他 AWS 服务交互的集成点。例如，AWS IoT 规则引擎由接受分析的规则组成，这些规则可以根据 MQTT 主题流触发其他 AWS 服务的下游操作。由于 AWS IoT 会与您的其他 AWS 资源通信，因此您必须确保为应用程序配置相应的服务角色权限。

数据保护

在构建 IoT 应用程序之前，必须设计和记录数据分类、监管和控制措施，以反映数据如何在云中永久存储，以及数据应如何在设备上或设备与云之间加密。与传统云应用程序不同的是，数据敏感度和监管扩展到了部署在网络边界以外远程位置的 IoT 设备。这些技术非常重要，因为它们支持为从设备传输的个人身份数据提供保护并遵守法规义务。

在设计过程中，确定在设备寿命终止时如何处理硬件、固件和数据。将存在时间较长的历史数据存储在云中。将当前的部分传感器数据存储在设备本地，即仅限执行本地操作所需的数据。仅在设备上存储所需的最少数据，以限制意外访问的风险。

除了在本地减少数据存储外，还必须在设备使用寿命终止时采取其他缓解措施。首先，设备应提供重置选项，以便将硬件和固件重置为默认出厂设置。其次，您的 IoT 应用程序可以对每个设备的最后一次登录时间运行定期扫描。可以撤销离线时间过长的设备，或者与不活跃客户账户关联的设备。第三，使用特定设备独有的密钥加密必须永久保留在设备上的敏感数据。

IOTSEC 10：您如何分类、管理并保护传输中和静态数据?

必须使用传输层安全性 (TLS) 协议对进出 AWS IoT 的所有流量进行加密。在 AWS IoT 中，当数据在 AWS IoT 和其他设备或 AWS 服务之间移动时，安全机制可保护数据。除 AWS IoT 之外，您还

必须采取设备级安全措施，从而不仅可以保护设备的私有密钥，还可以保护在设备上收集和处理的数据。

针对嵌入式开发，AWS 提供了多种服务，它们可抽象应用程序层组件，同时默认在边缘整合 AWS 安全最佳实践。对于微控制器，AWS 建议使用 Amazon FreeRTOS。Amazon FreeRTOS 通过蓝牙 LE、TCP/IP 和其他协议的库扩展 FreeRTOS 内核。此外，Amazon FreeRTOS 还包含一组安全 API，允许您创建与 AWS IoT 安全通信的嵌入式应用程序。

对于基于 Linux 的边缘网关，AWS IoT Greengrass 可用于将云功能扩展到网络边缘。AWS IoT Greengrass 实施多种安全功能，包括与互联设备相互进行基于 X.509 证书的身份验证，用于管理 AWS IoT Greengrass 和云应用程序之间通信权限的 AWS IAM 策略和角色，以及用于确定是否可以在互联设备和 Greengrass 核心之间路由数据及路由方法的订阅。

事件响应

准备在 IoT 中进行事件响应需要规划如何处理 IoT 工作负载中的两种类型的事件。第一个事件是攻击单个 IoT 设备，试图降低设备性能或影响设备的行为。第二个事件是规模更大的 IoT 事件，例如网络中断和 DDoS 攻击。在这两种情况下，IoT 应用程序的架构在以下方面发挥了重要作用：确定诊断事件的速度、关联事件中的数据，然后以可靠的自动化方式将运行手册应用到受影响的设备。

对于 IoT 应用程序，请遵循以下事件响应最佳实践：

- IoT 设备按照设备属性（如位置和硬件版本）划分为不同的组。
- 用户可按照动态属性（如连接状态、固件版本、应用程序状态和设备运行状况）搜索 IoT 设备。
- OTA 更新可为设备暂存一段时间以便进行部署。部署过程受到监控，如果设备未能保持相应的 KPI，则可以自动中止部署。
- 任何更新过程都可以灵活应对错误，设备可以从失败的软件更新中恢复和回滚。
- 提供详细的日志记录、指标和设备遥测数据，包含有关设备当前执行情况和一段时间内执行情况的背景信息。
- 队列级指标会监控队列的整体运行状况，并在一段时间内未达到运营 KPI 时发出警报。

- 可隔离、检测和分析任何偏离预期行为的设备，以确定固件和应用程序是否可能受损。

IOTSEC 11：您准备如何应对影响单台设备或设备队列的事件?

实施一项策略，让您的信息安全团队可以快速识别需要修复的设备。确保信息安全小组有考虑设备更新的固件版本控制和修补的运行手册。创建自动化流程，在容易受到攻击的设备联网时主动为其应用安全补丁。

至少，您的安全团队应该能够根据设备日志和当前设备行为检测特定设备的事件。确定事件后，下一阶段是隔离应用程序。要通过 AWS IoT 服务实现这一目标，您可以使用具有更严格的 IoT 策略的 AWS IoT 事物组，并为这些设备启用自定义组日志记录。这样一来，您只需启用与故障排除相关的功能，并收集更多数据即可确定根本原因和补救措施。最后，事件解决后，您必须能够将固件更新部署到设备，以使其返回已知状态。

关键 AWS 服务

IoT 方面的重要 AWS 安全服务包括 AWS IoT 注册表、AWS IoT Device Defender、AWS Identity and Access Management (IAM) 和 Amazon Cognito。结合使用这些服务，您可以安全地控制用户对 IoT 设备、AWS 服务和资源的访问。以下服务和功能支持五大安全领域：

设计：AWS 设备认证计划提供已预先通过 AWS IoT 互操作性测试的 IoT 终端节点和边缘硬件。测试包括双向身份验证和远程补丁的 OTA 支持。

AWS Identity and Access Management (IAM)：设备凭证（X.509 证书、IAM、Amazon Cognito 身份池和 Amazon Cognito 用户池或自定义授权令牌）使您能够安全地控制设备和外部用户对 AWS 资源的访问。AWS IoT 策略增加了对 IoT 设备实施精细访问控制的能力。ACM 私有 CA 提供了基于云的方法来创建和管理设备证书。使用 AWS IoT 事物组在组一级管理 IoT 权限，而不是单独管理每台设备。

检测性控制：AWS IoT Device Defender 会记录来自 AWS IoT Core 的设备通信和云端指标。AWS IoT Device Defender 可通过 Amazon Simple Notification Service (Amazon SNS) 向内部系统或管理员发送通知，从而自动执行安全响应。AWS CloudTrail 会记录您的 IoT 应用程序的管理操作。Amazon CloudWatch 是一种与 AWS IoT Core 集成的监控服务，可触发 CloudWatch 事件

以自动执行安全响应。CloudWatch 可捕获与 IoT 边缘组件和云服务之间的连接和安全事件相关的详细日志。

基础设施保护：AWS IoT Core 是一种云服务，使互联设备可以轻松安全地与云应用程序及其他设备交互。AWS IoT Core 中的 AWS IoT 规则引擎使用 IAM 权限与其他下游 AWS 服务进行通信。

数据保护：AWS IoT 包含基于 TLS 的设备加密功能，可保护传输中的数据。AWS IoT 可直接与支持静态加密的服务（如 Amazon S3 和 Amazon DynamoDB）集成。此外，AWS Key Management Service (AWS KMS) 可让您创建和控制用于加密的密钥。在设备上，您可以使用 AWS 边缘产品（如 Amazon FreeRTOS、AWS IoT Greengrass 或 AWS IoT 嵌入式 C 开发工具包）来支持安全通信。

事件响应：AWS IoT Device Defender 允许您创建安全配置文件，这些配置文件可用于检测与正常设备行为的偏差，并触发自动响应，包括 AWS Lambda。AWS IoT Device Management 应用于对需要修复的设备进行分组，然后使用 AWS IoT 作业将修复部署到设备。

资源

请参阅以下资源，详细了解安全方面的最佳实践：

文档和博客

- IoT 安全性和身份
- AWS IoT Device Defender
- IoT 身份验证模型
- 端口 443 上的 MQTT
- 使用 Device Defender 检测异常

白皮书

- MQTT 主题设计

可靠性支柱

可靠性支柱侧重于预防故障和快速从故障中恢复以便满足业务和客户需求的能力。关键主题包括设置相关的基本要素、跨项目要求、恢复计划以及变更管理。

设计原则

除了架构完善的框架的总体设计原则外，另有针对云中 IoT 可靠性的三大设计原则：

- **以生产规模模拟设备行为**：创建可密切反映生产部署的生产规模测试环境。利用多步骤模拟计划，您可以在上线前以更大的负载测试应用程序。在开发过程中，在一段时间内提升模拟测试，从一次测试总流量的 10% 开始，并随着时间的推移而增加（即 25%、50%，然后是第一天设备流量的 100%）。在模拟测试期间，监控性能并查看日志，以确保整个解决方案的行为符合预期。
- **通过流或队列从 IoT 规则引擎缓存消息传送**：利用托管服务实现高吞吐量遥测。通过在高吞吐量主题后面注入排队层，IoT 应用程序可以管理故障、聚合消息和扩展其他下游服务。
- **应对故障和实现弹性的设计**：规划设备本身的弹性非常重要。根据您的使用案例，弹性恢复功能可能需要强大的重试逻辑来应对间歇性连接，能够回滚固件更新，故障转移到不同的网络协议或进行本地通信以实现关键消息传送，运行冗余传感器或边缘网关以应对硬件故障，以及执行出厂重置。

定义

在云中有三个领域的可靠性最佳实践：

1. 基础
2. 变更管理
3. 故障管理

为实现可靠性，系统必须具有经过周密规划的基础和适当的监控功能，同时能够根据需求、具体要求处理各项变更，或防御可能出现的未经授权的拒绝服务攻击。系统的设计应具有故障检测和自动修复功能。

最佳实践

基础

如果遇到网络或云错误，IoT 设备必须继续以某种容量运行。设计设备固件以处理间歇性连接或连接中断，所采用的方法对内存和电力限制较敏感。IoT 云应用程序还必须设计为处理经常在在线和离线之间转换的远程设备，以保持数据一致性并随着时间的推移水平扩展。监控 IoT 的整体利用率，并创建自动增加容量的机制，以确保您的应用程序能够管理 IoT 峰值流量。

要防止设备产生不必要的峰值流量，必须实施设备固件，以防止整个设备组同时尝试相同的操作。例如，如果 IoT 应用程序包含报警系统，并且所有报警系统都在当地时间上午 9 点发送激活事件，则 IoT 应用程序将立即迎来整个队列的峰值。您应将随机化因素纳入这些计划活动中，例如定时事件和指数回退，允许 IoT 设备在一段时间内更均匀地分布其峰值流量。

以下问题主要针对可靠性的准备阶段。

IOTREL 1.您如何针对 IoT 应用程序中的峰值处理 AWS 服务限制?

AWS IoT 针对不同使用方式提供了一系列软硬限制。AWS IoT 在 IoT 限制页面上概述了所有数据层面限制。数据层面操作（例如，MQTT 连接、MQTT 发布和 MQTT 订阅）是设备连接的主要驱动程序。因此，务必要查看 IoT 限制，确保您的应用程序遵守与数据层面相关的任何软限制，同时不超过数据层面施加的任何硬限制。

IoT 扩展方法的最重要部分是确保您围绕任何硬性限制设计架构，因为超过不可调整的限制会导致应用程序错误，例如限制和客户端错误。硬限制与单条 IoT 连接的吞吐量有关。如果您发现您的应用程序超过了硬限制，我们建议您重新设计您的应用程序，以避免出现这些情况。这可以通过多种方式完成，例如重组 MQTT 主题或实施云端逻辑以在将消息发送到相关设备之前聚合或筛选消息。

AWS IoT 中的软限制通常与独立于单个设备的账户级限制相关联。对于所有账户级限制，您均应计算单个设备的 IoT 使用量，然后将该使用量乘以设备数量，以确定您的应用程序在产品初次发布时所需的基本 IoT 限制。AWS 建议您配置一个上升期，在这段时间内，您的限制提额将与您当前的生产峰值使用量保持一致，并提供额外的缓冲。为确保 IoT 应用程序预置充足，请采取以下措施：

- 查阅发布的 AWS IoT CloudWatch 指标，了解所有限制。
- 在 AWS IoT Core 中监控 CloudWatch 指标。
- 如果您需要提高限额，有关 CloudWatch 限制指标的警报会发出信号。
- 为 IoT 中的所有阈值设置警报，包括 MQTT 连接、发布、订阅、接收和规则引擎操作。
- 确保您在达到 100% 容量之前及时请求提高限额。

除数据层面限制外，AWS IoT 服务还具有管理 API 的控制层面。控制层面管理创建和存储 IoT 策略和主体、在注册表中创建事物以及关联 IoT 主体（包括证书和 Amazon Cognito 联合身份）的流程。由于引导和设备注册对整个过程至关重要，因此规划控制层面操作和限制非常重要。控制层面 API 调用基于以每秒请求数为单位衡量的吞吐量。控制层面调用通常按每秒数十个请求的大小顺序进行。对您来说，请务必从预期的峰值注册使用量反向推导，确定是否需要提高控制层面操作限额。规划注册设备的持续提升期，以便 IoT 限额提升与日常数据层面使用情况保持一致。

要防止控制层面请求突增，您的架构应将对这些 API 的访问限制为仅向用户或内部应用程序授权。实施回退和重试逻辑，并将入站请求排队以控制这些 API 的数据速率。

IOTREL 2.您管理 IoT 数据到其他应用程序的提取和处理吞吐量的策略是什么？

虽然 IoT 应用程序仅可在其他设备之间路由通信，但有些消息仍会在您的应用程序中处理和存储。在这些情况下，其余的 IoT 应用程序必须准备好响应传入数据。依赖该数据的所有内部服务都需要一种无缝扩展数据提取和处理的方法。在架构完善的 IoT 应用程序中，内部系统通过提取层与 IoT 平台的连接层分离。提取层由队列和流组成，这些队列和流支持持久的短期存储，同时允许计算资源不以提取速率处理数据。

为了优化吞吐量，请先使用 AWS IoT 规则将入站设备数据路由到 Amazon Kinesis Data Streams、Amazon Kinesis Data Firehose 或 Amazon Simple Queue Service 等服务，然后再执行任何计算操作。确保所有中间流点都已预置为可处理峰值容量。此方法会创建上游应用程序以弹性方式处理数据所需的排队层。

IOTREL 3.在与云通信时，您如何处理设备可靠性?

IoT 解决方案可靠性还必须涵盖设备本身。设备部署在远程位置，并且要处理由于各种超出 IoT 应用程序控制范围的外部因素导致的间歇性连接或连接中断。例如，如果 ISP 中断数小时，设备将如何处理并响应这些可能存在时间较长的潜在网络中断？ 在设备上实施一组最少的嵌入式操作，使其更具弹性，以应对管理与 AWS IoT Core 的连接和通信之间的细微差别。

您的 IoT 设备必须能够在没有互联网连接的情况下运行。您必须使提供以下功能的固件实现稳定运行：

- 可长期离线存储重要消息，并在重新连接后将这些消息发送到 AWS IoT Core。
- 在连接尝试失败时实施指数重试和回退逻辑。
- 如有必要，设置单独的故障转移网络通道，以向 AWS IoT 提供关键信息。这可能包括从 Wi-Fi 故障转移到备用蜂窝网络，或故障转移到无线个人区域网络协议（如蓝牙 LE），以便将消息发送到已连接的设备或网关。
- 使用 NTP 客户端或低漂移实时时钟设置当前时间的方法。设备应等待实现时间同步，然后再尝试与 AWS IoT Core 建立连接。否则，系统将为用户提供一种设置设备时间的方法，以便后续连接能够成功。
- 向 AWS IoT Core 发送错误代码和整体诊断消息。

变更管理

IOTREL 4.如何将更改部署到您的 IoT 应用程序和回滚?

请务必实施恢复到设备固件或云应用程序早期版本的功能，以应对部署失败的情况。如果您的应用程序架构完善，您将从设备中获取指标，以及 AWS IoT Core 和 AWS IoT Device Defender 生

成的指标。当您的设备 canary 在任何云端更改后偏离预期行为时，您也会收到警报。根据运行指标中的任何偏差，您需要能够：

- 使用 Amazon S3 控制所有设备固件的版本。
- 对设备固件的清单或执行步骤进行版本控制。
- 为设备实施已知安全的默认固件版本，以便在出现错误时回退。
- 使用加密代码签名实施更新策略、版本检查和多个非易失性存储分区，用于部署软件映像和回滚。
- 在 CloudFormation 中对所有 IoT 规则引擎配置进行版本控制。
- 使用 CloudFormation 对所有下游 AWS 云资源进行版本控制。
- 使用 CloudFormation 和其他基础设施即代码工具，实施恢复云端更改的回滚策略。

处理 AWS 上的基础设施即代码，以便自动监控和管理 IoT 应用程序更改。对所有设备固件构件进行版本控制，确保在必要时可以验证、安装或回滚更新。

故障管理

IOTREL 5.您的 IoT 应用程序如何应对故障?

由于 IoT 是一种事件驱动型工作负载，因此您的应用程序代码必须能够灵活处理已知和未知错误，这些错误可能会在事件渗透到您的应用程序中时发生。架构完善的 IoT 应用程序能够记录和重试数据处理中的错误。IoT 应用程序将以原始格式存档所有数据。通过存档所有有效和无效的数据，架构可以更准确地将数据恢复到给定时间点。

借助 IoT 规则引擎，应用程序可以实现 IoT 错误操作。如果调用操作时出现问题，规则引擎将调用错误操作。这使您可以捕获、监控、发出警报并最终重试可能无法传送到主要 IoT 操作的消息。我们建议使用与主要操作不同的 AWS 服务配置 IoT 错误操作。请将 Amazon SQS 或 Amazon Kinesis 等持久存储用于错误操作。

应用程序逻辑应从规则引擎开始，首先处理队列中的消息，并验证消息的架构是否正确。您的应用程序逻辑应捕获并记录任何已知错误，并有选择地将这些消息移至自己的 DLQ 以进行进一步分析。拥有全面的 IoT 规则，该规则使用 Amazon Kinesis Data Firehose 和 AWS IoT Analytics 通

道将所有原始、未格式化的消息传输到 Amazon S3 中的长期存储、AWS IoT Analytics 数据存储和用于数据仓库的 Amazon Redshift。

IOTREL 6.如何验证物理资产的不同级别的硬件故障模式?

IoT 实施必须允许在设备级别出现多种类型的故障。故障可能是硬件、软件、连接问题或意外的不利条件。规划事物故障的一种方法是在可能的情况下成对部署设备，或者在同一覆盖区域（网格化）部署的设备队列中部署双传感器。

无论设备故障的根本原因是什么，如果设备可以与您的云应用程序通信，则应使用诊断主题将有关硬件故障的诊断信息发送至 AWS IoT Core。如果设备由于硬件故障而断开连接，请使用处于连接状态的队列索引来跟踪连接状态变化。如果设备长时间处于离线状态，则触发设备可能需要修复的警报。

关键 AWS 服务

使用 Amazon CloudWatch 监控运行时指标并确保可靠性。支持三个可靠性领域的其他服务和功能如下所示：

基础：AWS IoT Core 使您无需管理底层基础设施即可扩展您的 IoT 应用程序。您可以通过请求增加账户级限额来扩展 AWS IoT Core。

变更管理：AWS IoT Device Management 使您能够就地更新设备，同时使用 Amazon S3 对设备的所有固件、软件和更新清单进行版本控制。AWS CloudFormation 允许您记录 IoT 基础设施即代码，并使用 CloudFormation 模板预置云资源。

故障管理：Amazon S3 允许您从设备对遥测数据进行持久存档。AWS IoT 规则引擎错误操作使您能够在主要 AWS 服务返回错误时回退到其他 AWS 服务。

资源

请参阅以下资源，了解有关可靠性方面最佳实践的更多信息：

文档和博客

- 使用设备时间验证 AWS IoT 服务器证书
- AWS IoT Core 限制
- IoT 错误操作
- 队列索引
- IoT Atlas

性能效率支柱

性能效率支柱侧重于有效使用计算资源。关键主题包括根据工作负载要求选择正确的资源类型和规模、监控性能，以及根据业务和技术需求的变化做出明智的决策来保持效率。性能效率支柱专注于高效利用计算资源来满足需求，以及在需求发生变化和技术不断演进的情况下保持这种效率。

设计原则

除了架构完善的框架性能效率的总体设计原则外，另有针对云中 IoT 性能效率的三大设计原则：

- **使用托管服务**：AWS 跨数据库、计算和存储提供多种托管服务，可帮助您的架构提高整体可靠性和性能。
- **批量处理数据**：将 IoT 应用程序的连接部分与 IoT 中的提取和处理部分分离。通过分离提取层，您的 IoT 应用程序可以处理聚合数据，并通过一次处理多条 IoT 消息来实现更无缝的扩展。
- **使用事件驱动的架构**：IoT 系统通过设备发布事件，并将这些事件渗透到 IoT 应用程序中的其他子系统。设计符合事件驱动型架构的机制，例如利用队列、消息处理、幂等性、死信队列和状态机。

定义

在云中实现性能效率包括四个方面的最佳实践：

1. 选择
2. 审核
3. 监控
4. 权衡

选择高性能架构时使用数据驱动型方法。收集架构各方面的数据，从总体设计到资源类型的选择与配置都包括在内。通过周期性审视您的选择，您可以确保自身充分发挥 AWS 平台持续发展所带来的优势。监控可以确保您随时发现与预期性能的偏差，并允许您采取针对性措施。您可以对您的架构作出权衡以便提高性能，例如使用压缩或缓存，或放宽一致性要求。

最佳实践

选择

架构完善的 IoT 解决方案由多个系统和组件组成，如设备、连接、数据库、数据处理和分析。在 AWS 中，有多种 IoT 服务、数据库产品和分析解决方案，可帮助您快速构建架构完善的解决方案，同时专注于业务目标。AWS 建议您利用最适合您的工作负载的托管 AWS 服务组合。以下问题主要针对性能效率方面的注意事项。

> **IOTPERF 1.如何选择性能最佳的 IoT 架构?**

当您选择架构实施方式时，可着眼于长远运营目标，采用数据驱动型方法。IoT 应用程序与事件驱动型架构可实现自然融合。您的架构将结合使用与事件驱动型模式集成的服务，如通知、发布和订阅数据、流式处理和事件驱动型计算。在以下章节中，我们将介绍您应考虑的五种主要 IoT 资源类型（设备、连接、数据库、计算和分析）。

设备

适合特定系统的最佳嵌入式软件因设备硬件覆盖范围而异。例如，网络安全协议虽然对于保护数据隐私和完整性必不可少，但可能会占用相对较大的 RAM。对于内联网和互联网连接，请使用 TLS，这样不仅可实现高强度密码套件，而且占用的空间极少。对于使用 TLS 连接到 AWS IoT 的设备，AWS IoT 支持椭圆曲线加密 (ECC) 算法。在设备的选择标准中，应优先考虑设备上的安全

软件和硬件平台。AWS 也有许多 IoT 合作伙伴，他们提供了能够安全集成到 AWS IoT 的硬件解决方案。

除了选择适当的硬件合作伙伴之外，您还可以选择使用许多软件组件在设备上运行应用程序逻辑，包括 Amazon FreeRTOS 和 AWS IoT Greengrass。

IOTPERF 2.如何为 IoT 设备选择硬件和操作系统?

IoT 连接

在开发用来与云通信的固件之前，请先实施一个安全、可扩展的连接平台，以支持设备的长期增长。根据预期的设备数量，IoT 平台必须能够扩展设备和云之间的通信工作流程，不管该工作流程是简单的提取遥测数据还是设备之间的命令和响应通信。

您可以使用 EC2 等 AWS 服务构建 IoT 应用程序，但您需要承担千篇一律的繁重工作来为 IoT 产品打造独特价值。因此，AWS 建议为 IoT 平台使用 AWS IoT Core。

AWS IoT Core 支持 HTTP、WebSockets 和 MQTT，MQTT 是一种轻量级通信协议，这种协议采用专门设计，能够容忍间歇性连接，最大限度地减少设备上的代码量，并降低网络带宽需求。

IOTPERF 3.如何选择主要的 IoT 平台?

数据库

IoT 应用程序将包含多个数据库，每个数据库都精心选择以实现一定的属性，例如向数据库写入数据的频率、从数据库读取数据的频率，以及数据的结构化和查询方式。在选择数据库产品时还需要考虑其他条件：

- 数据量和保留期。
- 内在数据组织和结构。
- 使用数据（原始数据或处理后的数据）的用户和应用程序及其地理位置/分布。
- 高级分析需求，如机器学习或实时可视化。
- 其他团队、组织和业务部门之间同步数据的情况。

- 行、表和数据库级别的数据安全性。
- 与其他相关数据驱动型事件（如企业应用程序、钻取控制面板或交互系统）的交互。

AWS 拥有多种支持 IoT 解决方案的数据库产品。对于结构化数据，您应使用 Amazon Aurora，这是一个高度可扩展的组织数据关系接口。对于查询需要低延迟且将由多个使用者使用的半结构化数据，可以使用 DynamoDB，这是一个完全托管、多区域、多主机数据库，提供一致的个位数毫秒延迟，并内置安全性、备份和还原以及内存中缓存。

若要存储原始、未格式化的事件数据，请使用 AWS IoT Analytics。AWS IoT Analytics 在将 IoT 数据存储到时间序列数据存储库中执行分析之前，会先对其进行筛选、转换和充实。 通过使用 Greengrass 机器学习推理等 AWS IoT 服务，可以使用 Amazon SageMaker，基于您的 IoT 数据在云中和边缘构建、训练和部署机器学习模型。请考虑将原始格式的时间序列数据存储在 Amazon Redshift 等数据仓库解决方案中。未格式化的数据可以通过 Amazon S3 和 Amazon Kinesis Data Firehose 导入到 Amazon Redshift。在可扩展的托管型数据存储解决方案中存档未格式化的数据后，便可以开始获取业务见解，探索数据，并识别随时间变化的趋势和模式。

除了存储和利用 IoT 数据的历史趋势外，您还必须有一个系统来存储设备的当前状态，并能够查询所有设备的当前状态。这可支持内部分析以及面向客户的 IoT 数据视图。

AWS IoT Shadow 服务是一项在云中存储设备虚拟表示的有效机制。AWS IoT Device Shadow 最适合用于管理每台设备的当前状态。此外，对于需要查询影子系统以满足运营需求的内部团队，可以利用队列索引的托管功能，该服务提供了一个包含 IoT 注册表和影子元数据的可搜索索引。例如，对于某个消费类应用程序，如果需要为大量外部用户提供基于索引的搜索和筛选功能，可结合使用 IoT 规则引擎、Kinesis Data Firehose 和 Amazon ElasticSearch Service 动态存档影子状态，从而以允许外部用户精细查询访问的格式存储数据。

IOTPERF 4.如何选择用于存储 IoT 设备状态的数据库?

计算

IoT 应用程序有助于实现需要在消息流上进行连续处理的大量提取操作。因此，架构必须选择以下计算服务：支持在数据存储期间和之前稳定丰富流处理和执行业务应用程序。

在 IoT 中最常用的计算服务是 AWS Lambda，该服务允许在遥测数据到达 AWS IoT Core 或 AWS IoT Greengrass 时调用操作。AWS Lambda 可以在整个 IoT 的不同地点使用。选择使用 AWS Lambda 触发业务逻辑的位置受希望处理特定数据事件的时间的影响。

Amazon EC2 实例还可用于各种 IoT 使用案例。它们可用于托管关系数据库系统和各种应用程序，如 Web、报告或托管现有本地解决方案。

IOTPERF 5.如何选择处理 AWS IoT 事件的计算解决方案?

分析

实施 IoT 解决方案的主要业务是更快地响应设备的性能和在现场的使用情况。通过直接对传入的遥测数据进行处理，企业可以做出更明智的决策，确定哪些新产品或功能需要优先处理，或者如何更有效地运行组织内的工作流程。必须采用以下方式选择分析服务：能够根据所执行的分析类型提供不同的数据视图。AWS 提供多种服务，可适合不同的分析工作流程，包括时间序列分析、实时指标，以及存档和数据湖使用案例。

利用 IoT 数据，您的应用程序可以基于流式数据消息生成时间序列分析。您可以计算不同时间的指标，然后将值流式传输到其他 AWS 服务。

此外，使用 AWS IoT Analytics 的 IoT 应用程序可以实施托管 AWS Data Pipeline，从而能够在将数据存储到时间序列数据存储库中之前执行数据转换、充实和筛选。此外，通过 AWS IoT Analytics，可以使用 QuickSight 和 Jupyter Notebooks 原生执行可视化和分析。

审核

IOTPERF 6.如何根据 IoT 应用程序的历史分析改进架构?

构建复杂的 IoT 解决方案时，可能会在对业务成果无直接影响的工作上投入大量时间。例如，管理 IoT 协议、保护设备身份以及在设备和云之间传输遥测数据。虽然 IoT 的这些方面很重要，但并不能直接带来差异化价值。加快 IoT 创新步伐也可能是一项挑战。

AWS 会根据 IoT 的常见挑战定期发布新功能和新服务。定期查看您的数据，了解新的 AWS IoT 服务是否能解决您架构中当前的 IoT 缺口，或者是否能取代您架构中的非核心业务组件。请利用专门构建的服务来聚合 IoT 数据、存储数据，然后将数据可视化以执行历史分析。您可以发送

IoT 设备中的时间戳信息，然后利用 AWS IoT Analytics 和基于时间的索引等服务存档具有相关时间戳信息的数据。AWS IoT Analytics 中的数据可以与您设备的其他 IT 或 OT 运营和效率日志一起存储在您自己的 Amazon S3 存储桶中。通过将这种存档状态的 IoT 数据与可视化工具相结合，您可以做出以数据为依据的决策，了解新的 AWS 服务如何提供额外价值，并衡量服务如何提高整个队列的效率。

监控

IOTPERF 7.如何对 IoT 应用程序运行端到端模拟测试?

可以使用设置为测试设备（具有特定的测试 MQTT 命名空间）的生产设备模拟 IoT 应用程序，也可以使用模拟设备进行模拟。使用 IoT 规则引擎捕获的所有传入数据使用生产所用的相同工作流程进行处理。

端到端模拟的频率取决于特定的发布周期或设备采用率。您应该测试失败路径（仅在失败期间执行的代码），以确保解决方案可以从容应对错误。您还应该针对生产和生产前账户持续运行设备 canary。在模拟测试中，设备 canary 是系统性能的关键指标。测试结果应形成文档，并应起草补救计划。应该执行用户验收测试。

IOTPERF 8.如何在 IoT 实施中使用性能监控?

有几种与 IoT 部署相关的关键性能监控类型，包括设备、云性能和存储/分析。使用从日志收集的数据以及遥测和命令数据创建适当的性能指标。首先跟踪基本性能，然后随着业务核心竞争力的壮大，基于指标进行构建。

利用 CloudWatch Logs 指标筛选条件通过正则表达式模式匹配将您的 IoT 应用程序标准输出转换为自定义指标。根据应用程序的自定义指标创建 CloudWatch 警报，以便快速深入了解 IoT 应用程序的行为。

设置精细日志来跟踪特定事物组。在 IoT 解决方案开发过程中，启用调试日志记录，以便清晰了解每个 IoT 消息从您的设备通过消息代理和规则引擎时的事件进展。在生产环境中，将日志记录更改为错误和警告。

除了云检测之外，还必须在部署之前在设备上运行检测，以确保设备充分使用本地资源，并且固件代码不会导致出现意外情况，如内存泄漏。部署针对受限设备进行高度优化的代码，并使用从嵌入式应用程序发布到 AWS IoT 的设备诊断消息监控设备的运行状况。

权衡

IoT 解决方案在广大的重要企业职能领域（如运营、客户服务、财务、销售和营销）实现丰富的分析功能。同时，还可以作为边缘网关的高效出口点。务必仔细考虑构建高效的 IoT 实施方案，即数据和分析由设备推送到云，机器学习算法从云提取到设备网关。

单个设备将受到给定网络支持的吞吐量限制。数据交换的频率必须与传输层和设备可以有选择地存储、聚合数据，然后将数据发送到云的能力相平衡。按照与后端应用程序处理数据和对数据采取措施所需的时间一致的时间间隔将数据从设备发送到云。例如，如果需要以一秒为增量查看数据，则设备必须以比一秒更频繁的时间间隔发送数据。相反，如果应用程序只按小时读取数据，则可以通过在边缘聚合数据点并每半小时发送一次数据来权衡性能。

IOTPERF 9.如何确保来自 IoT 设备的数据可随时供业务和运营系统使用?

IOTPERF 10.多长时间将数据从设备传输到 IoT 应用程序一次?

企业应用程序、业务和运营查看 IoT 遥测数据的速度决定了处理 IoT 数据的最高效点。在网络受限而硬件不受限制的环境中，可以使用 AWS IoT Greengrass 等边缘解决方案从云离线操作和处理数据。在网络和硬件都受到限制的情况下，需通过使用二进制格式并将类似的消息组合为单个请求，想方设法压缩消息负载。

对于可视化，可使用 Amazon Kinesis Data Analytics 快速编写 SQL 代码，从而以近乎实时的方式持续读取、处理和存储数据。对流式数据使用标准的 SQL 查询可构建转换数据并提供数据洞察的应用程序。借助 Kinesis Data Analytics，可以将 IoT 数据用于流式分析。

关键 AWS 服务

用于实现性能效率的关键 AWS 服务是 Amazon CloudWatch，它可集成多项 IoT 服务，包括 AWS IoT Core、AWS IoT Device Defender、AWS IoT Device Management、AWS Lambda 和

DynamoDB。通过 Amazon CloudWatch，可以了解应用程序的整体性能和运行状况。以下服务也有利于提高性能效率：

选择

设备：AWS 硬件合作伙伴提供生产就绪型 IoT 设备，可用作 IoT 应用程序的一部分。Amazon FreeRTOS 是一款适用于微控制器的操作系统，并附带软件库。AWS IoT Greengrass 支持在边缘运行本地计算、消息收发、数据缓存、同步和 ML。

连接：AWS IoT Core 是一款托管的 IoT 平台，支持 MQTT，后者是一种适用于设备通信的轻量级发布和订阅协议。

数据库：Amazon DynamoDB 是一种完全托管的 NoSQL 数据存储，支持个位数毫秒延迟请求，可支持快速检索不同的 IoT 数据视图。

计算：AWS Lambda 是一种事件驱动型计算服务，无需预置服务器，即可运行应用程序代码。Lambda 与从 AWS IoT Core 或上游服务（如 Amazon Kinesis 和 Amazon SQS）触发的 IoT 事件原生集成。

分析：AWS IoT Analytics 是一种托管服务，可执行设备级分析，同时为 IoT 遥测数据提供时间序列数据存储。

审核：可通过 AWS 网站上的 AWS IoT 博客部分了解作为 AWS IoT 一部分新推出的内容。

监控：Amazon CloudWatch 指标和 Amazon CloudWatch Logs 提供指标、日志、筛选条件、警报和通知，您可以将它们与现有的监控解决方案集成。可以通过设备遥测数据增强这些指标以监控您的应用程序。

权衡：AWS IoT Greengrass 和 Amazon Kinesis 服务支持在 IoT 应用程序的不同位置聚合和批量处理数据，从而提供更高效的计算性能。

资源

请参阅以下资源，详细了解与性能效率有关的最佳实践：

文档和博客

- AWS Lambda 入门

- DynamoDB 入门
- AWS IoT Analytics 用户指南
- Amazon FreeRTOS 入门
- AWS IoT Greengrass 入门
- AWS IoT 博客

成本优化支柱

成本优化支柱包括系统在整个生命周期中不断完善和改进的过程。从最开始的概念验证的初始设计到生产工作负载的持续运营，您都可以采用本文中的实践来构建和运营具有成本意识的系统，从而在实现业务成果的同时尽可能降低成本，使您的企业能够最大限度地提高投资回报率。

设计原则

除了架构完善的框架的总体成本优化设计原则外，另有一条针对云中 IoT 实现成本优化的设计原则：

- **管理制造成本权衡**：业务合作标准、硬件组件选择、固件复杂性和分销要求都会影响制造成本。最大限度地减少成本有助于确定一个产品是否可以通过多代产品的形式成功推向市场。但是，在选择组件和制造商时走捷径可能会增加下游成本。例如，与信誉良好的制造商合作有助于最大程度地减少下游硬件故障和客户支持成本。选择专用加密组件可能会增加物料清单 (BOM) 成本，但会降低下游制造和预置的复杂性，因为该部件可能已经附带私有密钥和证书。

定义

在云中实现成本优化包括四个方面的最佳实践：

1. 资源成本效益
2. 供需匹配
3. 支出认知
4. 持续优化

需要权衡各种因素。例如，您想优化上市速度还是优化成本？ 在某些情况下，最好优化上市速度以便快速上市、交付新功能或按时完成任务，而不是优化预付成本。由于人们总是倾向于过度补偿，而不是花时间进行基准测试，逐渐找出成本最优的部署，导致设计决策有时会过于仓促，缺乏对经验数据的参考。这会导致过度预置和优化不足的部署。以下各节介绍了一些技巧和战略性指导，以帮助您的部署实现初始和持续成本优化。

最佳实践

资源成本效益

鉴于 IoT 应用程序可以生成设备和数据的规模，您的系统使用适当的 AWS 服务是节约成本的关键。除了 IoT 解决方案的总体成本外，IoT 架构师还通过剖析 BOM 成本，了解连接性。对于 BOM 计算，您必须预测和监控在设备的整个生命周期内管理与 IoT 应用程序的连接有关的长期成本。利用 AWS 服务将帮助您计算初始 BOM 成本，利用经济高效的事件驱动型服务，并更新架构以继续降低连接的总体生命周期成本。

提高资源成本效益的最直接方式是将 IoT 事件分组，集中处理数据。通过按组处理事件，可以缩短每个单独消息的总体计算时间。聚合可帮助您节省计算资源，并在压缩和存档数据后进行保存时启用解决方案。此策略将减少总体存储占用空间，而不会丢失数据或影响数据查询。

COST 1.如何选择批量处理、充实和聚合从 IoT 平台传送到其他服务的数据的方法?

AWS IoT 非常适合用于处理直接使用或历史分析方面的流式数据。对从 AWS IoT Core 传送到其他 AWS 服务的数据进行批量处理有多种方式，而它们的不同之处在于是批量处理原始数据（按原样），还是充实数据后再进行批量处理。要在提取期间（或提取后立即）充实、转换和筛选 IoT 遥测数据，最好创建 AWS IoT 规则，该规则可将数据发送到 Amazon Kinesis Streams、Amazon Kinesis Firehose、AWS IoT Analytics 或 Amazon SQS。这些服务支持一次处理多个数据事件。

在处理来自这一批量管道的原始设备数据时，您可以使用 AWS IoT Analytics 和 Amazon Kinesis Data Firehose 将数据传输到 S3 存储桶和 Amazon Redshift。要降低 Amazon S3 中的存储成本，应用程序可以利用生命周期策略，将数据存档到成本较低的存储，如 Amazon S3 Glacier。

供需匹配

实现最佳供需匹配能够尽可能降低系统成本。但是，考虑到 IoT 工作负载对数据突增的敏感性，解决方案必须动态可扩展，并在预置资源时考虑峰值容量。对于事件驱动型数据流，您可以选择自动预置 AWS 资源以满足峰值容量，然后在已知的低流量期间扩展和缩减。

以下问题主要针对成本优化的准备阶段：

COST 2.如何将资源供应与设备需求匹配?

无服务器技术（如 AWS Lambda 和 API Gateway）可以帮助您创建更具可扩展性和弹性的架构，并且您只需要在应用程序使用这些服务时付费。AWS IoT Core、AWS IoT Device Management、AWS IoT Device Defender、AWS IoT Greengrass 和 AWS IoT Analytics 也是托管服务，按使用量收费，不收取闲置计算容量费用。托管服务的好处是 AWS 管理资源的自动预置。如果您利用托管服务，您将负责监控和设置 AWS 服务的限额提高警报。

当设计架构以实现供需匹配时，需主动规划长时间的预期使用量以及最有可能超出的限制。需要在您未来的计划中考虑提高限额。

持续优化

通过评估 AWS 的新功能，可以通过分析设备性能来优化成本，并更改设备与 IoT 的通信方式。

要通过更改设备固件来优化解决方案成本，您应查看 AWS 服务（如 AWS IoT）的定价模式，确定哪些部分低于给定服务的计费计量阈值，然后在成本与性能之间做出取舍。

COST 3.如何优化设备与 IoT 平台之间的负载大小?

IoT 应用程序必须在终端设备可实现的网络吞吐量与 IoT 应用程序处理数据的最有效方法之间实现平衡。我们建议 IoT 部署最初根据设备限制优化数据传输。首先，将离散的数据事件从设备发送到云，尽量避免在一条消息中批量处理多个事件。稍后，如果需要，您可以使用序列化框架压缩消息，然后再将消息发送到 IoT 平台。

从成本角度来看，MQTT 负载大小是 AWS IoT Core 的一个关键成本优化要素。IoT 消息以 5KB 的增量计费，最高为 128KB。因此，每个 MQTT 负载应尽可能接近任意 5KB。例如，当前大小

为 6KB 的负载的成本效益不如 10KB 的负载，因为尽管一条消息比另一条消息大，但发布该消息的总成本是相同的。

为了充分利用负载大小，需想方设法压缩数据或将数据聚合到消息中：

- 应缩短值，同时保持清晰易读。如果 5 位数精度足以满足要求，则不应在负载中使用 12 位数。
- 如果不需要 IoT 规则引擎负载检查，可以使用序列化框架将负载压缩为更小的大小。
- 您可以减少发送数据的频率，并在可计费的增量内将消息聚合在一起。例如，每秒发送一条 2-KB 消息，可以更低的 IoT 消息成本实现，即每隔一秒发送两条 2-KB 消息。

在实施此方法之前应考虑一些权衡因素。在设备中增加复杂性或延迟可能会意外地增加处理成本。只有当您的解决方案投入生产，并且您可以使用以数据为依据的方法确定更改数据发送到 AWS IoT Core 的方式所带来的成本影响之后，才能对 IoT 负载进行成本优化。

COST 4.如何优化存储 IoT 设备当前状态的成本?

架构完善的 IoT 应用程序在云中具有设备的虚拟表示。此虚拟表示由托管数据存储或专用 IoT 应用程序数据存储组成。在这两种情况下，您的终端设备必须以一种将设备状态更改高效传输到 IoT 应用程序的方式进行编程。例如，如果固件逻辑指示完整设备状态可能不同步，并且最好通过发送所有当前设置来进行协调，则设备应仅发送完整设备状态。当单个状态发生更改时，设备应优化将这些更改传输到云的频率。

在 AWS IoT 中，设备影子和注册表操作以 1KB 为增量进行计量，以每百万次访问/修改操作进行计费。影子存储每台设备的所需状态或实际状态，并使用注册表命名和管理设备。

设备影子和注册表的成本优化过程侧重于管理执行了多少操作以及每个操作的大小。如果您的操作对影子和注册表操作成本敏感，则应想方设法优化影子操作。例如，对于影子，可以将多个报告的字段聚合到一个影子消息更新，而不是单独发送每个报告的更改。将影子更新组合在一起后，将合并对服务的更新，从而降低影子的总体成本。

关键 AWS 服务

支持成本优化的关键 AWS 功能是成本分配标签，该功能可帮助您了解系统的成本。以下服务和功能在成本优化的三个领域非常重要：

- **资源成本效益**：AWS 服务 Amazon Kinesis、AWS IoT Analytics 和 Amazon S3 可帮助您在单次请求中处理多条 IoT 消息，从而提高计算资源的成本效益。
- **供需匹配**：AWS IoT Core 是一款托管型 IoT 平台，用于管理连接、云中的设备安全性、消息路由和设备状态。
- **持续优化**：可通过 AWS 网站上的 AWS IoT 博客部分了解作为 AWS IoT 一部分新推出的内容。

资源

请参阅以下资源，详细了解有关成本优化的 AWS 最佳实践。

文档和博客

- AWS IoT 博客

总结

AWS 架构完善的框架涵盖了五大支柱，强调了在云中为 IoT 应用程序设计和运行可靠、安全、高效且经济实惠的系统的架构最佳实践。此框架提供了一系列问题，您可以利用这些问题审核现有或将要实施的 IoT 架构，此外还针对每个支柱提供了一系列 AWS 最佳实践。在架构中应用该框架可帮助您打造稳定且高效的系统，从而使您能够将主要精力集中在功能需求上。

贡献者

以下是对本文档做出贡献的个人和组织：

- Olawale Oladehin，IoT 解决方案架构师专家

- Dan Griffin，IoT 软件开发工程师
- Catalin Vieru，IoT 解决方案架构师专家
- Brett Francis，IoT 产品解决方案架构师
- Craig Williams，IoT 合作伙伴解决方案架构师
- Philip Fitzsimons，Amazon Web Services Well-Architected 高级经理

文档修订

日期	描述
2019 年 12 月	更新包含有关 IoT 开发工具包使用、引导、设备生命周期管理和 IoT 的其他指导
2018 年 11 月	首次发布