aws

开发人员指南

AWS Panorama

AWS Panorama: 开发人员指南

Table of Contents

终止支持通知：2026 年 5 月 31 日， AWS 将终止对的支持。 AWS Panorama 2026 年 5 月 31 日之后，您将无法再访问 AWS Panorama 控制台或 AWS Panorama 资源。有关更多信息，请参阅AWS Panorama 终止支持。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。

什么是 AWS Panorama？

AWS Panorama 是一项将计算机视觉引入您的本地摄像机网络的服务。您在数据中心安装 AWS Panorama 设备或其他兼容设备，将其注册到云端 AWS Panorama，然后从云端部署计算机视觉应用程序。 AWS Panorama 可与您现有的实时流媒体协议 (RTSP) 网络摄像机配合使用。该设备运行AWS合作伙伴提供的安全计算机视觉应用程序，或您使用应用程序 SDK 自行构建的 AWS Panorama 应用程序。

该 AWS Panorama 设备是一款紧凑型边缘设备，它使用针对机器学习工作负载进行了优化的强大 system-on-module (SOM)。该设备可以针对多个视频流并行运行多个计算机视觉模型，并实时输出结果。它专为在商业和工业环境中使用而设计，具有防尘和液体防护等级 (IP-62)。

该 AWS Panorama 设备使您能够在边缘运行独立的计算机视觉应用程序，而无需将图像发送到 AWS 云。通过使用 AWS 开发工具包，您可以与其他 AWS 服务集成，并使用它们来跟踪应用程序中随时间推移的数据。通过与其他 AWS 服务集成 AWS Panorama ，您可以使用执行以下操作：

- 分析流量模式 – 使用 AWS 开发工具包记录数据以在 Amazon DynamoDB 中进行零售分析。使用无服务器应用程序分析一段时间内收集的数据，检测数据中的异常情况，并预测未来的行为。
- 接收现场安全警报 – 监控工业现场的禁区。当您的应用程序检测到潜在的不安全情况时，请将图像上传到 Amazon Simple Storage Service (Amazon S3) 并向 Amazon Simple Notification Service (Amazon SNS) 主题发送通知，以便收件人可以采取纠正措施。
- 改进质量控制 – 监控装配线的输出，以识别不符合要求的零件。用文本和边界框突出显示不合格零件的图像，并将其显示在监视器上，供质量控制团队查看。
- 收集训练和测试数据 – 上传计算机视觉模型无法识别的对象的图像，或者模型对其猜测的置信度处于临界值的对象的图像。使用无服务器应用程序创建需要标记的图像队列。标记图像并使用它们在 Amazon A SageMaker I 中重新训练模型。

AWS Panorama 使用其他 AWS 服务来管理 AWS Panorama 设备、访问模型和代码以及部署应用程序。 AWS Panorama 在不要求您与其他服务进行交互的情况下尽可能做到这一点，但是了解以下服务可以帮助您了解其 AWS Panorama 工作原理。

- SageMaker AI — 您可以使用 SageMaker AI 从摄像头或传感器收集训练数据，构建机器学习模型，然后对其进行计算机视觉训练。 AWS Panorama 使用 SageMaker AI Neo 来优化模型以在 AWS Panorama 设备上运行。

- Amazon S 3 — 您可以使用 Amazon S3 接入点暂存应用程序代码、模型和配置文件，以便部署到 AWS Panorama 设备。
- AWS IoT— AWS Panorama 使用 AWS IoT 服务监控 AWS Panorama 设备状态、管理软件更新和部署应用程序。您无需 AWS IoT 直接使用。

要开始使用 AWS Panorama 设备并了解有关该服务的更多信息，请继续入门 AWS Panorama。

AWS Panorama 终止支持

经过深思熟虑，我们决定从 2026 年 5 月 31 日起终止对 AWS Panorama 的支持。从 2025 年 5 月 20 日起，AWS Panorama 将不再接受新客户。作为账户在 2025 年 5 月 20 日之前注册该服务的现有客户，您可以继续使用 AWS Panorama 功能。2026 年 5 月 31 日之后，您将无法再使用 AWS Panorama。

AWS Panorama 的替代品

如果您对 AWS Panorama 的替代方案感兴趣 AWS ，那么买家和建筑商都可以选择。

对于 out-of-the-box解决方案，AWS 合作伙伴网络提供来自多个合作伙伴的解决方案。您可以在 AWS 解决方案库中浏览我们许多合作伙伴提供的解决方案。这些合作伙伴解决方案包括硬件、软件、软件即服务 (SaaS) 应用程序、托管解决方案或基于您需求的定制实施选项。这种方法提供的解决方案可以满足您的用例，而无需您具备计算机视觉、人工智能或应用程序开发方面的专业知识。通过利用 AWS 合作伙伴的专业知识，这通常可以更快地实现价值。

如果您更喜欢构建自己的解决方案，可以 AWS 提供 AI 工具和服务，以帮助您开发基于 AI 的计算机视觉应用程序并管理边缘的应用程序和设备。Amazon SageMaker 提供了一组工具，可通过完全托管的基础架构、工具和工作流程为您的用例构建、训练和部署机器学习模型。除了允许您构建自己的模型外，Amazon 还 SageMaker JumpStart提供内置的计算机视觉算法，这些算法可以根据您的特定用例进行微调。

为了管理边缘设备和应用程序，AWS IoT Gre engrass 是一款久经考验的安全解决方案，用于为物联网设备部署和更新应用程序。对于基于服务器的实施，AWS Systems Manager 提供了一套用于管理服务器的工具，Amazon E KS An ywhere 或 ECS Anywh ere 可以管理边缘服务器上的应用程序容器。Amazon 在《使用 A WS 保护物联网 (IoT)》白皮书第 4 节中提供了一些管理边缘设备的指南，以及其他资源。这种构建器方法为您提供了加快人工智能和设备管理开发的工具，同时提供了完全的灵活性，可以构建满足您的确切要求并与现有硬件和软件基础架构集成的解决方案。这通常可以降低解决方案的运营成本。

从 AWS Panorama 迁移

要将现有应用程序从 AWS Panorama 迁移到替代方案，您需要更换现有硬件设备，从 AWS Panorama 服务迁移应用程序，并为新解决方案实施边缘管理和安全。下文将详细探讨其中的每一个领域：

硬件更换

现有的 AWS Panorama 设备基于 Nvidia Jetson Xavier 平台。硬件可以替换为符合您要求的基于当前一代Nvidia Jetson平台的类似off-the-shelf 设备或边缘服务器。虽然大多数 AWS Panorama 部署可以用类似的设备取代，但我们已经看到一些在单个位置使用大量摄像头的客户发现，服务器是更好的选择。

应用程序迁移

需要重写 AWS Panorama 应用程序，以取消使用任何特定于 AWS Panorama 的 API 调用。AWS Panorama 应用程序仅支持使用 H.264 通过实时流媒体协议 (RTSP) 源输入视频，这些视频输入是使用 AWS Panorama 设备软件开发工具包中的摄像机节点提供的。

要移植现有应用程序，您需要实现一个类似于 AWS Panorama 的应用程序类，这样现有代码就可以大部分重复使用。banner-code.zip 文件中提供了示例代码，该文件显示了同时使用 PyAV 和 OpenCV 实现的示例。

这是一种简单的方法，只需最少的代码更改，但就支持的视频流类型而言，它有许多与当前基于 AWS Panorama 的实现相同的限制。

另一种选择是重新设计应用程序，以更好地利用系统资源并支持新的应用程序功能。对于此选项，您可以使用GStreamer或实现从媒体源DeepStream到推理结果和业务逻辑的媒体管道，或者使用功能更丰富、性能更好的机器学习 (ML) 运行时实现，例如 Nvidia Triton 推理服务器。这种方法需要对更多的视频处理管道进行更改，但既效率更高，又具有更大的灵活性，可以支持更广泛的编解码器、摄像机类型和其他传感器。

边缘管理和安全

无论媒体管道如何，您还必须实现凭据的安全存储，例如 RTSP 流用户名和密码。 AWS 为应用程序提供了不同的安全存储参数的方法：

- AWS IoT Device Shadow 服务用于存储传递给应用程序的参数，以及跟踪边缘设备上应用程序的状态。

- AWS Secrets Manager 用于存储此类证书，以更好地保护访问媒体流的证书。
- 如果您使用 Amazon EK S 或 Amazon EC S，则还可以使用安全的 AWS System Manager 参数存储来存储证书和其他应用程序参数。

选择取决于应用程序的安全要求，以及您计划使用哪些其他 AWS 产品来实现应用程序。

在将 AWS Panorama 设备替换为通用边缘设备时，还必须实现应用程序所需的安全功能，并配置设备以符合您的安全要求。 AWS 在 AWS Well-Architected Framework 的安全支柱中提供了这方面的指导。虽然该框架主要侧重于云应用程序，但大多数原则也适用于边缘设备。此外，您还应使用所选解决方案的硬件安全功能，例如 AWS IoT AWS Greengrass V2 硬件安全集成，并使用所选操作系统和/或设备提供的安全功能，例如全盘加密。

摘要

尽管AWS Panorama计划于2026年5月31日关闭，但它以亚马逊 SageMaker 工具的形式 AWS 提供了一套强大的人工智能/机器学习服务和解决方案，用于构建计算机视觉模型和设备管理服务，例如AWS IoT Greengrass、Amazon EKS和Amazon ECS Anywhere，以及支持类似解决方案开发的 AWS Sy st em Manager。 AWS 如果您更愿意购买而不是构建解决方案，还可以从 AWS 合作伙伴网络中的合作伙伴那里获得一系列产品。如果您愿意，我们提供了示例代码和实施指南，以帮助您迁移到替代解决方案。您应该探索这些选项，以确定哪种方法最适合您的特定需求。

有关更多详细信息，请参阅以下资源：

- Amazon SageMaker 开发者指南 — 有关如何构建模型或使用中提供的内置计算机视觉算法的详细文档SageMaker JumpStart。
- AWS IoT Core 开发人员指南-有关如何连接和管理物联网设备的详细文档。
- AWS IoT Greengrass V2 开发人员指南 — 有关如何在您的设备上构建、部署和管理物联网应用程序的详细文档。
- ECS Anywhere 开发者指南-有关在边缘运行 ECS 的详细文档。
- EKS Anywhere 最佳实践指南-有关在边缘运行 EKS 的详细文档。
- AWS 解决方案库 — 一系列提供商提供的合作伙伴产品，提供预建或定制的计算机视觉解决方案。
- Panoram a FAQs-其他全景图信息。

常见问题

Panorama 停产的时间是什么时候？

该公告是在2025年5月20日发布的。在此日期之后，未使用该服务的客户将无法再访问Panorama。活跃客户将能够继续正常使用该服务，直至2026年5月31日。在此之前，客户必须将其应用程序转移到替代解决方案并迁移Panorama的应用程序。2026 年 5 月 31 日之后，任何尝试访问 Panorama 服务的应用程序都将无法运行，Panorama 设备也将无法运行。

现有客户将受到怎样的影响？

现有客户可以继续正常使用该服务，直到 2026 年 5 月 31 日。之后，尝试访问Panorama的应用程序将不再运行。在此日期之后，Panorama 设备也将无法再使用。

新客户是否被接受？

不是。 从2025年5月20日起，只有Panorama的活跃用户才能使用该服务。如果客户需要访问之前使用过的服务中的应用程序，则他们可以向客户支持部门创建案例，请求访问其账户。如果客户之前没有使用过该服务，则不会被授予访问权限。

客户可以探索哪些替代方案？

AWS 提供了一系列可以取代 Panorama 功能的服务。我们建议客户使用 off-the-shelf硬件，通过组合使用符合其要求的 AWS IoT Core、AWS IoT Greengrass、Amazon AKS Anywhere、Amazon ECS Anywhere 和/或 AWS 系统管理器来管理设备和应用程序。AWS 合作伙伴网络还提供多种解决方案，这些解决方案由具有特定计算机视觉专业知识的合作伙伴提供，可供客户考虑。

客户如何才能从 Panorama 迁移？

需要修改Panorama应用程序，以消除对Panorama特定的依赖关系 APIs，这些依赖主要与相机连接和直播有关。 AWS 提供了示例代码来演示如何进行这些更改。删除这些依赖关系后，可以将应用程序移至备用硬件平台。

如果我在 2025 年 5 月 20 日当天或之后遇到问题，将提供哪些支持？

AWS 将继续为 Panorama 提供支持，直到停产通知期结束（2026 年 5 月 31 日）。对于任何支持需求，客户都应通过其正常的支持渠道提交支持案例。 AWS 将提供安全更新、错误修复和可用性增强。

我无法在 2026 年 5 月 31 日之前进行迁移。可以延长日期吗？

我们相信，Panorama提供的替代方案使客户能够在2026年5月31日之前迁移到替代解决方案，并且我们没有计划将该服务的可用性延长到该日期之后。

服务结束后，我的边缘应用程序会继续运行吗？

不是。 Panorama 设备和应用程序依赖于 Panorama 云服务的连接。该服务于2026年5月31日停止后，Panorama应用程序和Panorama设备都将无法继续运行。

入门 AWS Panorama

首先 AWS Panorama，请先了解该服务的概念和本指南中使用的术语。然后，您可以使用 AWS Panorama 控制台注册您的 AWS Panorama 设备并创建应用程序。只需大约一小时，您就可以配置设备、更新软件并部署示例应用程序。要完成本节中的教程，您需要使用 AWS Panorama 设备和通过本地网络传输视频的摄像头。

> **Note**
>
> 要购买 AWS Panorama 设备，请访问AWS Panorama 控制台。

AWS Panorama 示例应用程序演示了 AWS Panorama 功能的使用。它包括一个使用 SageMaker AI 训练过的模型和使用 AWS Panorama 应用程序 SDK 运行推理和输出视频的示例代码。该示例应用程序包括一个 CloudFormation 模板和脚本，这些脚本展示了如何通过命令行自动执行开发和部署工作流程。

本章的最后两个主题详细介绍了模型和摄像机的要求，以及AWS Panorama 设备的硬件规格。如果您还未购买设备和摄像机，或计划开发自己的计算机视觉模型，请先查看这些主题的内容，以了解更多信息。

主题

- AWS Panorama 概念
- 设置 AWS Panorama Appliance
- 部署 AWS Panorama 示例应用程序
- 开发 AWS Panorama 应用程序
- 支持的计算机视觉模型和摄像头
- AWS Panorama Appliance 规范
- 服务限额

AWS Panorama 概念

在 AWS Panorama 中，您可以创建计算机视觉应用程序并将其部署到 AWS Panorama Appliance 或兼容设备上，以分析来自网络摄像机的视频流。您可以用 Python 编写应用程序代码，并使用 Docker 构建应用程序容器。您可以使用 AWS Panorama Application CLI 在本地或从 Amazon Simple Storage Service (Amazon S3) 导入机器学习模型。应用程序使用 AWS Panorama 应用程序 SDK 接收来自摄像机的视频输入，并与模型进行交互。

概念

- AWS Panorama Appliance
- 兼容设备
- 应用程序
- Nodes
- 模型

AWS Panorama Appliance

AWS Panorama Appliance 是运行应用程序的硬件。您可使用 AWS Panorama 控制台注册设备、更新其软件以及向其部署应用程序。AWS Panorama Appliance 上的软件可连接摄像机视频流，向您的应用程序发送视频帧，并在连接的显示器上显示视频输出。

AWS Panorama Appliance 是一款由 Nvidia Jetson AGX Xavier 提供支持的边缘设备。它不会将图像发送到 AWS 云端进行处理，而是在经过优化的硬件上本地运行应用程序。这使您能够实时分析视频并在本地处理结果。设备需要互联网连接以报告其状态、上传日志以及执行软件更新和部署。

有关更多信息，请参阅 管理 AWS Panorama 设备。

兼容设备

除了 AWS Panorama 设备外，AWS Panorama 还支持 AWS 合作伙伴提供的兼容设备。兼容设备支持与 AWS Panorama Appliance 相同的功能。您可以使用 AWS Panorama 控制台和 API 注册并管理兼容设备，并以相同的方式构建和部署应用程序。

- 联想 ThinkEdge® SE7 0 — 由 Nvidia Jetson Xavier NX 提供支持

本指南中的内容和示例应用程序是使用 AWS Panorama Appliance 开发的。有关设备特定硬件和软件功能的详细信息，请参阅制造商的文档。

应用程序

应用程序在 AWS Panorama Appliance 上运行，用于对视频流执行计算机视觉任务。您可以组合 Python 代码和机器学习模型，以构建计算机视觉应用程序，然后通过互联网将其部署到 AWS Panorama Appliance 上。应用程序可将视频发送到显示器，也可以使用 AWS SDK 将结果发送到 AWS 服务。

要构建和部署应用程序，您需要使用 AWS Panorama 应用程序 CLI。AWS Panorama 应用程序 CLI 是一个命令行工具，用于生成默认应用程序文件夹和配置文件、使用 Docker 构建容器并上传资产。您可以在一台设备上运行多个应用程序。

有关更多信息，请参阅 管理 AWS Panorama 应用程序。

Nodes

应用程序包含多个称为节点的组件，代表输入、输出、模型和代码。节点可以仅是配置（输入和输出），也可以包含构件（模型和代码）。应用程序的代码节点捆绑在您上传到 Amazon S3 接入点的节点包中，AWS Panorama Appliance 可以访问这些节点。应用程序清单是定义节点之间连接的配置文件。

有关更多信息，请参阅 应用程序节点。

模型

计算机视觉模型是一种机器学习网络，经过训练可以处理图像。计算机视觉模型可以执行各种任务，如分类、检测、分割和跟踪。计算机视觉模型将图像作为输入，并输出有关图像或图像中对象的信息。

AWS Panorama 支持使用 PyTorch、Apache MXNet 和 TensorFlow构建的模型。您可以使用 Amazon A SageMaker I 或在您的开发环境中构建模型。有关更多信息，请参阅 ???。

设置 AWS Panorama Appliance

要开始使用您的 AWS Panorama Appliance 或兼容设备，请在 AWS Panorama 控制台中注册并更新软件。在设置过程中，您可以在 AWS Panorama 中创建代表物理设备的设备资源，然后使用 USB 驱动器将文件复制到该设备。设备使用这些证书和配置文件连接 AWS Panorama 服务。然后，您可以使用 AWS Panorama 控制台更新设备软件并注册摄像头。

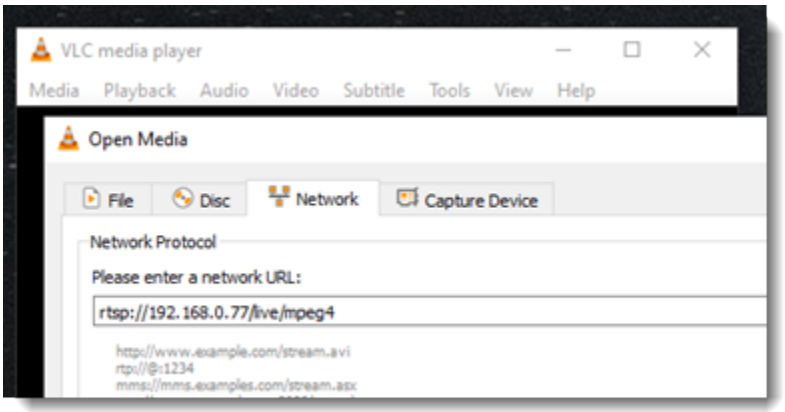
Sections

先决条件

要学习本教程，您需要一台 AWS Panorama Appliance 或兼容设备以及以下硬件：

- 显示器 - 带有 HDMI 输入的显示器，用于查看示例应用程序输出。
- U 盘（AWS Panorama Appliance 随附）— 一种 FAT32格式化的 USB 3.0 闪存盘，存储空间至少为 1 GB，用于将包含配置文件和证书的档案传输到 AWS Panorama 设备。
- 摄像机 - 输出 RTSP 视频流的 IP 摄像机。

使用摄像机制造商提供的工具和说明确定摄像机的 IP 地址和数据流路径。您可以使用视频播放器（如 VLC）将其作为网络媒体源打开，以验证流 URL：



AWS Panorama 控制台使用其他 AWS 服务组装应用程序组件、管理权限和验证设置。要注册设备并部署示例应用程序，您需要以下权限：

- AWSPanoramaFullAccess— 提供对 AWS Panorama、亚马逊 S3 中的 AWS Panorama 接入点 AWS Secrets Manager、中的设备证书和亚马逊中的设备日志的完全访问权限 CloudWatch。包括为 AWS Panorama 创建服务相关角色的权限。
- AWS Identity and Access Management (IAM) — 首次运行时，用于创建 AWS Panorama 服务和 AWS Panorama 设备使用的角色。

如果您没有在 IAM 中创建角色的权限，请让管理员打开 AWS Panorama 控制台并接受创建服务角色的提示。

注册并配置 AWS Panorama Appliance

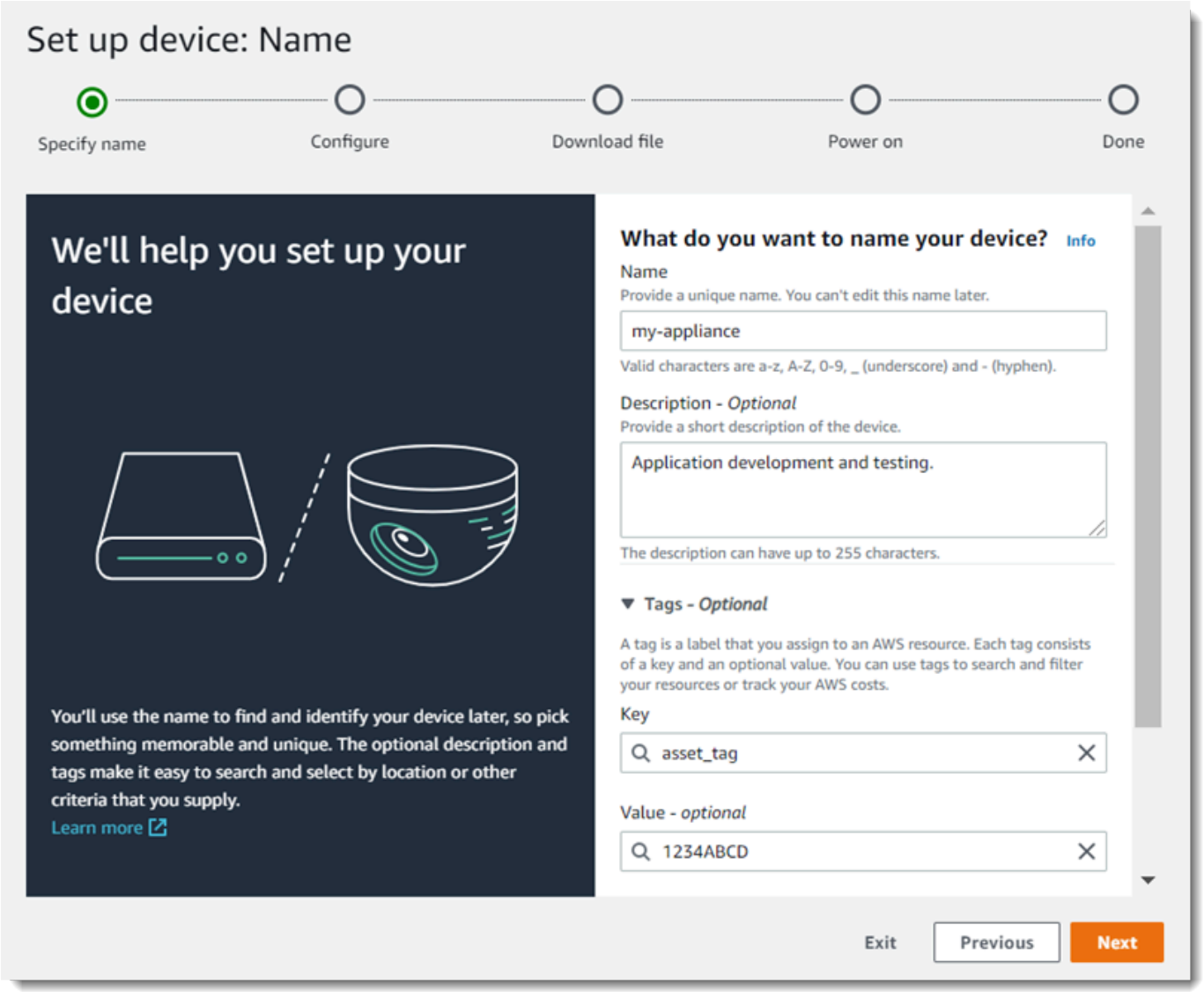
AWS Panorama Appliance 是一种硬件设备，可通过本地网络连接到支持网络的摄像机。该设备使用基于 Linux 的操作系统，包括 AWS Panorama 应用程序 SDK 和用于运行计算机视觉应用程序的支持软件。

为了 AWS 进行设备管理和应用程序部署，设备使用设备证书。您可以使用 AWS Panorama 控制台生成预置证书。设备使用此临时证书完成初始设置并下载永久设备证书。

> **Important**
>
> 您在此过程中生成的预置证书仅在 5 分钟内有效。如果您未在该时间内完成注册流程，则必须重新开始。

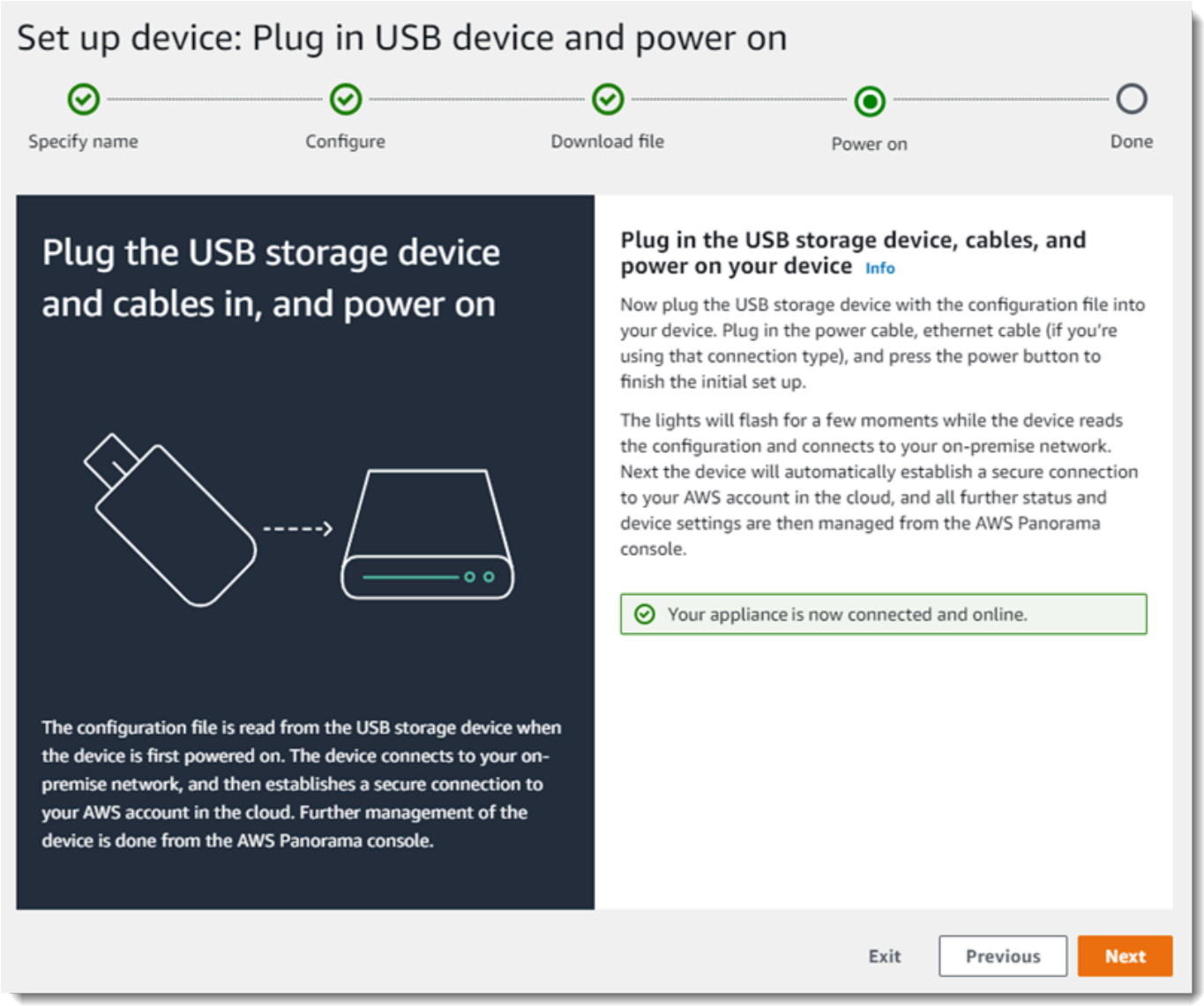
注册设备

1. 将 USB 驱动器连接至您的计算机。连接网络和电源线，以准备设备。设备启动并等待连接 USB 驱动器。
2. 打开 AWS Panorama 控制台的“开始”页面。
3. 选择添加设备。
4. 选择开始设置。
5. 输入 AWS Panorama 中代表设备的设备资源的名称和描述。选择下一步



6. 如果您需要手动分配 IP 地址、NTP 服务器或 DNS 设置，请选择高级网络设置。否则，请选择下一步。
7. 选择 下载存档。选择下一步。
8. 将配置存档复制到 USB 驱动器的根目录。
9. 将 USB 驱动器连接到设备前端的 HDMI 端口旁边的 USB 3.0 端口。

   当您连接 USB 驱动器时，设备会将配置存档和网络配置文件复制到自己并连接到 AWS 云端。设备完成连接后，状态指示灯从绿色变为蓝色，然后变回绿色。
10. 要继续，请选择下一步。



11. 选择完成。

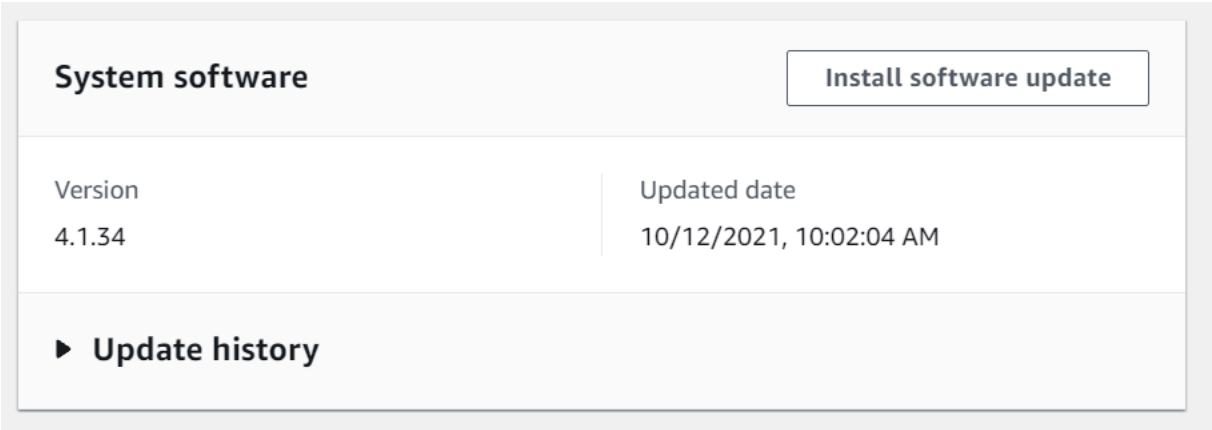
升级设备软件

AWS Panorama Appliance 有多个软件组件，包括 Linux 操作系统、AWS Panorama 应用程序 SDK 以及支持的计算机视觉库和框架。为确保您可以在设备上使用最新的功能和应用程序，请在设置完成后且有可用更新时升级其软件。

更新设备软件

1. 打开 AWS Panorama 控制台的设备页面。
2. 选择设备。

3. 选择设置。
4. 在系统软件下，选择安装软件更新。



5. 选择新版本，然后选择安装。

> **Important**
>
> 在继续操作之前，请从设备中取出 USB 驱动器并格式化以删除其中的内容。配置存档包含敏感数据，不会自动删除。

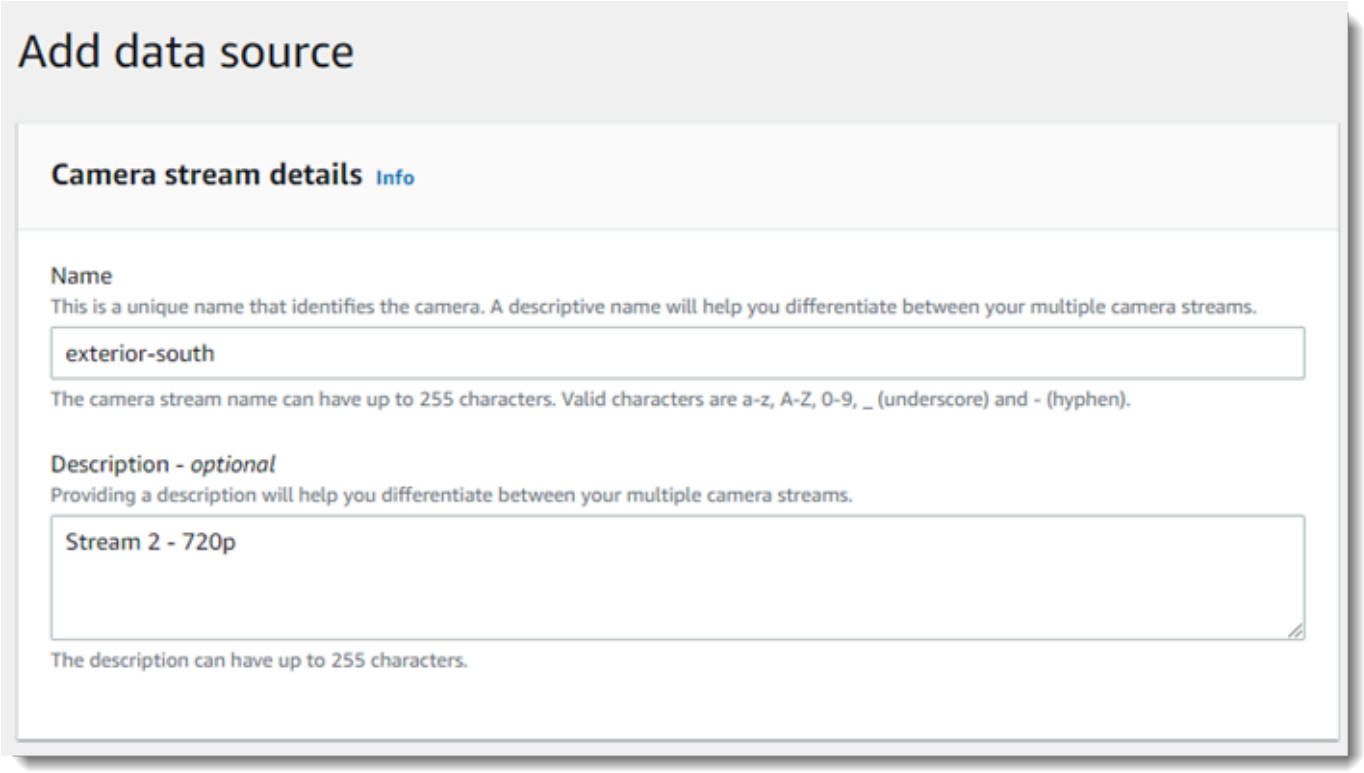
升级过程可能需要 30 分钟或者更长时间。您可以在 AWS Panorama 控制台或连接的显示器上监控其进度。该过程完成后，设备将重新启动。

添加摄像机视频流

接下来，在 AWS Panorama 控制台上注册摄像机视频流。

注册摄像机视频流

1. 打开 AWS Panorama 控制台的“数据来源”页面。
2. 选择添加数据来源。



3. 配置以下设置。

   - 名称 - 摄像机视频流的名称。
   - 描述 - 对摄像机、其位置或其他详细信息的简短描述。
   - RTSP URL - 指定摄像机 IP 地址和流路径的 URL。例如，`rtsp://192.168.0.77/live/mpeg4/`
   - 凭证 - 如果使用密码保护摄像机视频流，请指定用户名和密码。

4. 选择保存。

AWS Panorama 将您的相机凭证安全地存储在中 AWS Secrets Manager。多个应用程序可以同时处理同一个摄像机视频流。

后续步骤

如果设置过程中遇到错误，请参阅 故障排除。

要部署示例应用程序，请继续阅读下一个主题。

部署 AWS Panorama 示例应用程序

在设置 AWS Panorama 设备或兼容设备并升级其软件后，部署示例应用程序。在以下部分中，您将使用 AWS Panorama Application CLI 导入示例应用程序，并使用 AWS Panorama 控制台进行部署。

示例应用程序使用机器学习模型对来自网络摄像头的视频帧中的对象进行分类。它使用 AWS Panorama 应用程序开发工具包加载模型、获取图像并运行模型。然后，应用程序将结果叠加在原始视频之上，并将其输出到连接的显示器。

在零售环境中，通过分析客流量模式，您可以预测客流量水平。通过将分析与其他数据相结合，您可以计划节假日和其他活动前后增加的人员需求，衡量广告和促销活动的有效性，或优化展示位置和库存管理。

Sections

先决条件

为了遵循本教程中的步骤，您需要一个命令行终端或 Shell，以便运行命令。在代码列表中，命令前为提示符 ($) 和当前目录的名称（如适用）。

```
~/panorama-project$ this is a command
this is output
```

对于长命令，我们使用转义字符 (\) 将命令拆分为多行。

在 Linux 和 macOS 中，可使用您首选的外壳程序和程序包管理器。在 Windows 10 中，您可以安装 Windows Subsystem for Linux，获取 Ubuntu 和 Bash 与 Windows 集成的版本。有关在 Windows 中设置开发环境的帮助，请参阅 在 Windows 中设置开发环境。

您可以使用 Python 开发 AWS Panorama 应用程序，并使用 Python 的软件包管理器 pip 安装工具。如果您还没有 Python，请安装最新版本。如果您有 Python 3 但没有 pip，请使用操作系统的包管理器安装 pip，或安装随 pip 一起提供的新版本的 Python。

在本教程中，你将使用 Docker 生成运行应用程序代码的容器。从 Docker 网站安装 Docker：获取 Docker

本教程使用 AWS Panorama Application CLI 导入示例应用程序、构建程序包和上传构件。AWS Panorama 应用程序 CLI 使用 AWS Command Line Interface (AWS CLI) 调用服务 API 操作。如果您已经有 AWS CLI，请将其升级到最新版本。要安装 AWS Panorama 应用程序 CLI 和 AWS CLI，请使用`pip`。

```
$ pip3 install --upgrade awscli panoramacli
```

下载示例应用程序，并将其提取到工作区中。

- 示例应用程序 — aws-panorama-sample.zip

导入示例应用程序

要导入示例应用程序以在您的账户中使用，请使用 AWS Panorama Application CLI。应用程序的文件夹和清单包含对占位符帐号的引用。若要使用您的帐号更新这些内容，请运行 `panorama-cli import-application` 命令。

```
aws-panorama-sample$ panorama-cli import-application
```

`packages` 目录中的 `SAMPLE_CODE` 包包含应用程序的代码和配置，包括使用应用程序基础映像 `panorama-application` 的 Dockerfile。若要生成在设备上运行的应用程序容器，请使用 `panorama-cli build-container` 命令。

```
aws-panorama-sample$ ACCOUNT_ID=$(aws sts get-caller-identity --output text --query
 'Account')
aws-panorama-sample$ panorama-cli build-container --container-asset-name code_asset --
package-path packages/${ACCOUNT_ID}-SAMPLE_CODE-1.0
```

AWS Panorama 应用程序 CLI 的最后一步是注册应用程序的代码和模型节点，并将资产上传到该服务提供的 Amazon S3 接入点。这些资产包括代码的容器映像、模型和每个资产的描述符文件。要注册节点并上传资产，请运行 `panorama-cli package-application` 命令。

```
aws-panorama-sample$ panorama-cli package-application
Uploading package model
Registered model with patch version
 bc9c58bd6f83743f26aa347dc86bfc3dd2451b18f964a6de2cc4570cb6f891f9
Uploading package code
Registered code with patch version
 11fd7001cb31ea63df6aaed297d600a5ecf641a987044a0c273c78ceb3d5d806
```

部署应用程序

使用 AWS Panorama 控制台将应用程序部署到您的设备。

要部署应用程序

1. 打开 AWS Panorama 控制台已部署的应用程序页面。
2. 选择部署应用程序。
3. 将应用程序清单的内容 `graphs/aws-panorama-sample/graph.json` 粘贴到文本编辑器中。选择下一步。
4. 对于 应用程序名称 ，输入 `aws-panorama-sample`。
5. 选择继续部署。
6. 选择开始部署。
7. 选择下一步，而不选择角色。
8. 选择选择设备，然后选择您的设备。选择下一步。
9. 在选择数据源步骤中，选择查看输入，然后将摄像头流添加为数据源。选择下一步。
10. 在配置步骤中，选择下一步。
11. 选择部署，然后选择完成。
12. 在已部署的应用程序列表中，选择aws-panorama-sample。

刷新此页面以获取更新，或使用以下脚本从命令行监视部署。

Example monitor-deployment.sh

```
while true; do
  aws panorama list-application-instances --query 'ApplicationInstances[?Name==`aws-
panorama-sample`]'
  sleep 10
```

```
done
```

```
[
    {
        "Name": "aws-panorama-sample",
        "ApplicationInstanceId": "applicationInstance-x264exmpl33gq5pchc2ekoi6uu",
        "DefaultRuntimeContextDeviceName": "my-appliance",
        "Status": "DEPLOYMENT_PENDING",
        "HealthStatus": "NOT_AVAILABLE",
        "StatusDescription": "Deployment Workflow has been scheduled.",
        "CreatedTime": 1630010747.443,
        "Arn": "arn:aws:panorama:us-west-2:123456789012:applicationInstance/
applicationInstance-x264exmpl33gq5pchc2ekoi6uu",
        "Tags": {}
    }
]
[
    {
        "Name": "aws-panorama-sample",
        "ApplicationInstanceId": "applicationInstance-x264exmpl33gq5pchc2ekoi6uu",
        "DefaultRuntimeContextDeviceName": "my-appliance",
        "Status": "DEPLOYMENT_PENDING",
        "HealthStatus": "NOT_AVAILABLE",
        "StatusDescription": "Deployment Workflow has completed data validation.",
        "CreatedTime": 1630010747.443,
        "Arn": "arn:aws:panorama:us-west-2:123456789012:applicationInstance/
applicationInstance-x264exmpl33gq5pchc2ekoi6uu",
        "Tags": {}
    }
]
...
```

如果应用程序无法开始运行，请在 Amazon Logs 中查看应用程序和设备 CloudWatch 日志。

查看输出

部署完成后，应用程序开始处理视频流并将日志发送到 CloudWatch。

在 “日志” 中查看 CloudWatch 日志

1. 打开日志控制台的 CloudWatch 日志组页面。
2. 在以下组中查找 AWS Panorama 应用程序和设备日志：

- 设备日志 - /aws/panorama/devices/*device-id*
- 应用程序日志 – /aws/panorama/devices/*device-id*/applications/*instance-id*

```
2022-08-26 17:43:39 INFO     INITIALIZING APPLICATION
2022-08-26 17:43:39 INFO     ## ENVIRONMENT VARIABLES
{'PATH': '/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin', 'TERM':
 'xterm', 'container': 'podman'...}
2022-08-26 17:43:39 INFO     Configuring parameters.
2022-08-26 17:43:39 INFO     Configuring AWS SDK for Python.
2022-08-26 17:43:39 INFO     Initialization complete.
2022-08-26 17:43:39 INFO     PROCESSING STREAMS
2022-08-26 17:46:19 INFO     epoch length: 160.183 s (0.936 FPS)
2022-08-26 17:46:19 INFO     avg inference time: 805.597 ms
2022-08-26 17:46:19 INFO     max inference time: 120023.984 ms
2022-08-26 17:46:19 INFO     avg frame processing time: 1065.129 ms
2022-08-26 17:46:19 INFO     max frame processing time: 149813.972 ms
2022-08-26 17:46:29 INFO     epoch length: 10.562 s (14.202 FPS)
2022-08-26 17:46:29 INFO     avg inference time: 7.185 ms
2022-08-26 17:46:29 INFO     max inference time: 15.693 ms
2022-08-26 17:46:29 INFO     avg frame processing time: 66.561 ms
2022-08-26 17:46:29 INFO     max frame processing time: 123.774 ms
```

要查看应用程序的视频输出，请使用 HDMI 电缆将设备连接到显示器。默认情况下，应用程序会显示置信度超过 20% 的任何分类结果。

Example squeezenet_classes.json

```
["tench", "goldfish", "great white shark", "tiger shark",
"hammerhead", "electric ray", "stingray", "cock", "hen", "ostrich",
"brambling", "goldfinch", "house finch", "junco", "indigo bunting",
"robin", "bulbul", "jay", "magpie", "chickadee", "water ouzel",
"kite", "bald eagle", "vulture", "great grey owl",
"European fire salamander", "common newt", "eft",
"spotted salamander", "axolotl", "bullfrog", "tree frog",
...
```

示例模型有 1000 个类，包括许多动物、食物和常见物体。尝试将摄像头对准键盘或咖啡杯。

为简单起见，示例应用程序使用轻量级分类模型。该模型输出一个数组，其中包含其每个类的概率。现实世界的应用程序更频繁地使用具有多维输出的对象检测模型。有关具有更复杂模型的示例应用程序，请参阅 示例应用程序、脚本和模板。

为 Python 启用 SDK

示例应用程序使用向 Amazon 发送指标 CloudWatch。 AWS SDK for Python (Boto) 要启用此功能，请创建一个角色，该角色授予应用程序发送指标的权限，然后重新部署附加了该角色的应用程序。

示例应用程序包含一个 CloudFormation 模板，该模板用于创建具有所需权限的角色。要创建角色，请使用 `aws cloudformation deploy` 命令。

```
$ aws cloudformation deploy --template-file aws-panorama-sample.yml --stack-name aws-panorama-sample-runtime --capabilities CAPABILITY_NAMED_IAM
```

重新部署应用程序

1. 打开 AWS Panorama 控制台已部署的应用程序页面。
2. 选择应用程序。
3. 选择替换。
4. 完成以下步骤以部署应用程序。在指定 IAM 角色中，选择您创建的角色。它的名称以 `aws-panorama-sample-runtime` 开头。
5. 部署完成后，打开CloudWatch控制台并查看AWSPanoramaApplication命名空间中的指标。每 150 帧，应用程序记录并上传帧处理和推理时间的指标。

清理

如果您已使用完示例应用程序，则可以使用 AWS Panorama 控制台将其从设备中删除。

要从设备中删除应用程序

1. 打开 AWS Panorama 控制台已部署的应用程序页面。
2. 选择应用程序。
3. 选择从设备中删除。

后续步骤

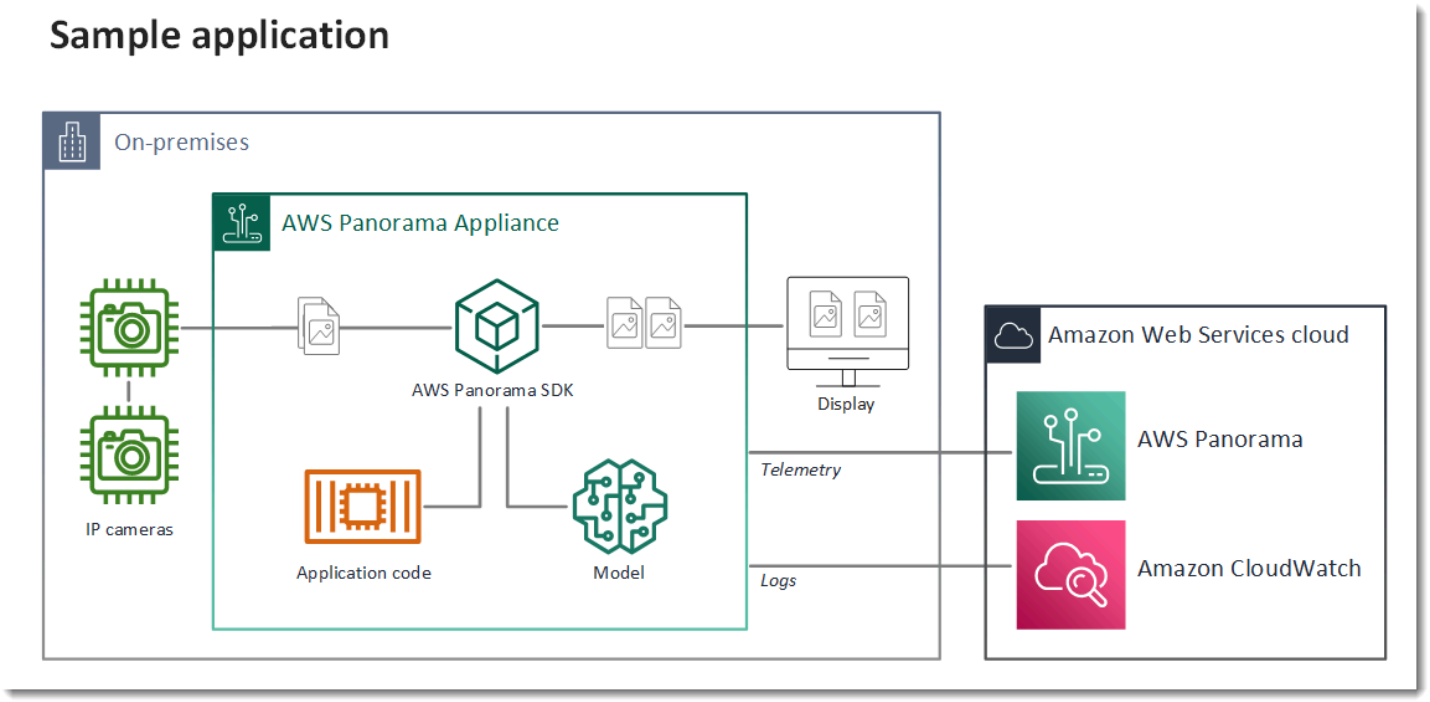
如果在部署或运行示例应用程序时遇到错误，请参阅 故障排除。

若要详细了解示例应用程序的功能和实现，请继续阅读下一主题。

开发 AWS Panorama 应用程序

您可以使用示例应用程序了解 AWS Panorama 应用程序结构，并作为您自己的应用程序的起点。

下图显示了在 AWS Panorama Appliance 上运行的应用程序的主要组件。应用程序代码使用 AWS Panorama 应用程序 SDK 获取图像并与模型交互，但无法直接访问模型。该应用程序会将视频输出到连接的显示器，但不会将图像数据发送到本地网络之外。



在此示例中，应用程序使用 AWS Panorama 应用程序 SDK 从摄像机获取视频帧，对视频数据进行预处理，然后将数据发送到可检测对象的计算机视觉模型。应用程序在连接到设备的 HDMI 显示器上显示结果。

Sections

应用程序清单

应用程序清单是 graphs 文件夹中名为 graph.json 的文件。清单定义了应用程序的组件，即程序包、节点和边缘。

软件包包括应用程序代码、模型、摄像头和显示器的代码、配置和二进制文件。此示例应用程序使用 4 个程序包：

Example **graphs/aws-panorama-sample/graph.json** - 程序包

```
    "packages": [
        {
            "name": "123456789012::SAMPLE_CODE",
            "version": "1.0"
        },
        {
            "name": "123456789012::SQUEEZENET_PYTORCH_V1",
            "version": "1.0"
        },
        {
            "name": "panorama::abstract_rtsp_media_source",
            "version": "1.0"
        },
        {
            "name": "panorama::hdmi_data_sink",
            "version": "1.0"
        }
    ],
```

前两个程序包的定义见应用程序中的 packages 目录。包含此应用程序特有的代码和模型。后两个套餐是 AWS Panorama 服务提供的通用相机和显示器程序包。abstract_rtsp_media_source 程序包是摄像机的占位符，您可以在部署期间将其覆盖。hdmi_data_sink 程序包代表设备上的 HDMI 输出连接器。

节点是程序包的接口，也可以是非软件包参数的接口，这些参数有默认值，您可以在部署时将其覆盖。代码和模型程序包在指定输入和输出的 package.json 文件中定义接口，这些文件可以是视频流或基本数据类型，例如浮点、布尔或字符串类型。

例如，code_node 节点指的是 SAMPLE_CODE 程序包中的接口。

```
    "nodes": [
        {
```

```
                "name": "code_node",
                "interface": "123456789012::SAMPLE_CODE.interface",
                "overridable": false,
                "launch": "onAppStart"
            },
```

此接口是在程序包配置文件 package.json 中定义的。该接口指定程序包为业务逻辑，并将名为 video_in 的视频流和一个名为 threshold 的浮点数作为输入。该接口还规定，代码需要一个名为 video_out 的视频流缓冲才能将视频输出到显示器

Example **packages/123456789012-SAMPLE_CODE-1.0/package.json**

```
{
    "nodePackage": {
        "envelopeVersion": "2021-01-01",
        "name": "SAMPLE_CODE",
        "version": "1.0",
        "description": "Computer vision application code.",
        "assets": [],
        "interfaces": [
            {
                "name": "interface",
                "category": "business_logic",
                "asset": "code_asset",
                "inputs": [
                    {
                        "name": "video_in",
                        "type": "media"
                    },
                    {
                        "name": "threshold",
                        "type": "float32"
                    }
                ],
                "outputs": [
                    {
                        "description": "Video stream output",
                        "name": "video_out",
                        "type": "media"
                    }
                ]
            }
        ]
```

```
    }
}
```

在应用程序清单中，camera_node 节点代表来自摄像机的视频流。该节点包括一个装饰器，在您部署应用程序时会出现在控制台中，提示您选择摄像机视频流。

Example **graphs/aws-panorama-sample/graph.json** - 摄像机节点

```
            {
                "name": "camera_node",
                "interface": "panorama::abstract_rtsp_media_source.rtsp_v1_interface",
                "overridable": true,
                "launch": "onAppStart",
                "decorator": {
                    "title": "Camera",
                    "description": "Choose a camera stream."
                }
            },
```

参数节点 threshold_param 定义应用程序代码使用的置信度阈值参数。默认值为 60，可在部署过程中覆盖。

Example **graphs/aws-panorama-sample/graph.json** - 参数节点

```
            {
                "name": "threshold_param",
                "interface": "float32",
                "value": 60.0,
                "overridable": true,
                "decorator": {
                    "title": "Confidence threshold",
                    "description": "The minimum confidence for a classification to be
 recorded."
                }
            }
```

应用程序清单的最后一部分 edges，是在节点之间建立连接。摄像机视频流和阈值参数连接到代码节点的输入端，代码节点的视频输出连接到显示器。

Example **graphs/aws-panorama-sample/graph.json** - 边缘

```
        "edges": [
```

```
        {
            "producer": "camera_node.video_out",
            "consumer": "code_node.video_in"
        },
        {
            "producer": "code_node.video_out",
            "consumer": "output_node.video_in"
        },
        {
            "producer": "threshold_param",
            "consumer": "code_node.threshold"
        }
    ]
```

使用示例应用程序构建

您可以将示例应用程序作为自己的应用程序的起点。

在您的账户中，每个程序包的名称必须是唯一的。如果您和您账户中的另一位用户都使用通用程序包名称（例如 code 或 model），则部署时可能会得到错误的程序包版本。将代码包的名称改为能代表您的应用程序的名称。

重命名代码包

1. 重命名程序包文件夹：packages/123456789012-*SAMPLE_CODE*-1.0/。
2. 更新以下位置的程序包名称。

   - 应用程序清单 - graphs/aws-panorama-sample/graph.json
   - 程序包配置 - packages/123456789012-SAMPLE_CODE-1.0/package.json
   - 生成脚本 - 3-build-container.sh

更新应用程序代码

1. 在 packages/123456789012-SAMPLE_CODE-1.0/src/application.py 中修改应用程序代码。
2. 要构建容器，需运行 3-build-container.sh。

```
aws-panorama-sample$ ./3-build-container.sh
TMPDIR=$(pwd) docker build -t code_asset packages/123456789012-SAMPLE_CODE-1.0
```

```
Sending build context to Docker daemon  61.44kB
Step 1/2 : FROM public.ecr.aws/panorama/panorama-application
 ---> 9b197f256b48
Step 2/2 : COPY src /panorama
 ---> 55c35755e9d2
Successfully built 55c35755e9d2
Successfully tagged code_asset:latest
docker export --output=code_asset.tar $(docker create code_asset:latest)
gzip -9 code_asset.tar
Updating an existing asset with the same name
{
    "name": "code_asset",
    "implementations": [
        {
            "type": "container",
            "assetUri":
 "98aaxmpl1c1ef64cde5ac13bd3be5394e5d17064beccee963b4095d83083c343.tar.gz",
            "descriptorUri":
 "1872xmpl129481ed053c52e66d6af8b030f9eb69b1168a29012f01c7034d7a8f.json"
        }
    ]
}
Container asset for the package has been succesfully built at  ~/aws-panorama-
sample-dev/
assets/98aaxmpl1c1ef64cde5ac13bd3be5394e5d17064beccee963b4095d83083c343.tar.gz
```

CLI 会自动从 `assets` 文件夹中删除旧容器资产并更新程序包配置。

3. 要上传程序包，请运行 `4-package-application.py`。
4. 打开 AWS Panorama 控制台已部署的应用程序页面。
5. 选择应用程序。
6. 选择替换。
7. 完成以下步骤以部署应用程序。如果需要，您可以更改应用程序清单、摄像机视频流或参数。

更改计算机视觉模型

示例应用程序包括计算机视觉模型。要使用自己的模型，请修改模型节点的配置，然后使用 AWS Panorama 应用程序 CLI 将其作为资产导入。

以下示例使用 MXNet SSD ResNet 50 型号，您可以从本指南的 GitHub 存储库中下载该型号：ssd_512_resnet50_v1_voc.tar.gz

更改示例应用程序的模型

1. 重命名程序包文件夹，以匹配您的模型。例如，前往 packages/*123456789012*-*SSD_512_RESNET50_V1_VOC*-1.0/。
2. 更新以下位置的程序包名称。

   - 应用程序清单 - graphs/aws-panorama-sample/graph.json
   - 程序包配置 - packages/*123456789012*-*SSD_512_RESNET50_V1_VOC*-1.0/package.json
3. 在程序包配置文件 (package.json) 中。将 assets 值改为空白数组。

```
{
    "nodePackage": {
        "envelopeVersion": "2021-01-01",
        "name": "SSD_512_RESNET50_V1_VOC",
        "version": "1.0",
        "description": "Compact classification model",
        "assets": [],
```

4. 打开程序包描述符文件 (descriptor.json)。更新 framework 和 shape 值，以匹配您的模型。

```
{
    "mlModelDescriptor": {
        "envelopeVersion": "2021-01-01",
        "framework": "MXNET",
        "inputs": [
            {
                "name": "data",
                "shape": [ 1, 3, 512, 512 ]
            }
        ]
    }
}
```

   shape 的值 1,3,512,512 表示模型输入的图像数量 (1)、每张图像中的通道数（3 - 红色、绿色和蓝色）以及图像的尺寸（512 x 512）。数组的值和顺序因模型而异。

5. 使用 AWS Panorama 应用程序 CLI 导入模型。AWS Panorama 应用程序 CLI 将模型和描述符文件复制到具有唯一名称的 assets 文件夹中，并更新程序包配置。

```
aws-panorama-sample$ panorama-cli add-raw-model --model-asset-name model-asset \
--model-local-path ssd_512_resnet50_v1_voc.tar.gz \
--descriptor-path packages/123456789012-SSD_512_RESNET50_V1_VOC-1.0/descriptor.json
 \
--packages-path packages/123456789012-SSD_512_RESNET50_V1_VOC-1.0
{
    "name": "model-asset",
    "implementations": [
        {
            "type": "model",
            "assetUri":
 "b1a1589afe449b346ff47375c284a1998c3e1522b418a7be8910414911784ce1.tar.gz",
            "descriptorUri":
 "a6a9508953f393f182f05f8beaa86b83325f4a535a5928580273e7fe26f79e78.json"
        }
    ]
}
```

6. 要上传模型，请运行 panorama-cli package-application。

```
$ panorama-cli package-application
Uploading package SAMPLE_CODE
Patch Version 1844d5a59150d33f6054b04bac527a1771fd2365e05f990ccd8444a5ab775809
 already registered, ignoring upload
Uploading package SSD_512_RESNET50_V1_VOC
Patch version for the package
 244a63c74d01e082ad012ebf21e67eef5d81ce0de4d6ad1ae2b69d0bc498c8fd
upload: assets/
b1a1589afe449b346ff47375c284a1998c3e1522b418a7be8910414911784ce1.tar.gz to
 s3://arn:aws:s3:us-west-2:454554846382:accesspoint/panorama-123456789012-
wc66m5eishf4si4sz5jefhx
63a/123456789012/nodePackages/SSD_512_RESNET50_V1_VOC/binaries/
b1a1589afe449b346ff47375c284a1998c3e1522b418a7be8910414911784ce1.tar.gz
upload: assets/
a6a9508953f393f182f05f8beaa86b83325f4a535a5928580273e7fe26f79e78.json to
 s3://arn:aws:s3:us-west-2:454554846382:accesspoint/panorama-123456789012-
wc66m5eishf4si4sz5jefhx63
a/123456789012/nodePackages/SSD_512_RESNET50_V1_VOC/binaries/
a6a9508953f393f182f05f8beaa86b83325f4a535a5928580273e7fe26f79e78.json
{
    "ETag": "\"2381dabba34f4bc0100c478e67e9ab5e\"",
    "ServerSideEncryption": "AES256",
```

```
    "VersionId": "KbY5fpESdpYamjWZ0YyGqHo3.LQQWUC2"
}
Registered SSD_512_RESNET50_V1_VOC with patch version
 244a63c74d01e082ad012ebf21e67eef5d81ce0de4d6ad1ae2b69d0bc498c8fd
Uploading package SQUEEZENET_PYTORCH_V1
Patch Version 568138c430e0345061bb36f05a04a1458ac834cd6f93bf18fdacdffb62685530
 already registered, ignoring upload
```

7. 更新应用程序代码。大部分代码可以重复使用。模型响应的特定代码包含在 `process_results` 方法中。

```
    def process_results(self, inference_results, stream):
        """Processes output tensors from a computer vision model and annotates a
 video frame."""
        for class_tuple in inference_results:
            indexes = self.topk(class_tuple[0])
        for j in range(2):
            label = 'Class [%s], with probability %.3f.'%
 (self.classes[indexes[j]], class_tuple[0][indexes[j]])
            stream.add_label(label, 0.1, 0.25 + 0.1*j)
```

根据您的模型，您可能还需要更新 `preprocess` 方法。

预处理图像

在应用程序将图像发送到模型之前，它会调整图像大小并将颜色数据规范化，为推理做好准备。应用程序使用的模型需要具有三个颜色通道的 224 x 224 像素图像，以匹配其第一层的输入数。应用程序将每个颜色值转换为 0 到 1 之间的数字，减去该颜色的平均值，再除以标准差，以调整每个颜色值。最后，它合并颜色通道并将其转换为模型可以处理的 NumPy 数组。

Example application.py - 预处理

```
def preprocess(self, img, width):
    resized = cv2.resize(img, (width, width))
    mean = [0.485, 0.456, 0.406]
    std = [0.229, 0.224, 0.225]
    img = resized.astype(np.float32) / 255.
    img_a = img[:, :, 0]
    img_b = img[:, :, 1]
    img_c = img[:, :, 2]
    # Normalize data in each channel
```

```
        img_a = (img_a - mean[0]) / std[0]
        img_b = (img_b - mean[1]) / std[1]
        img_c = (img_c - mean[2]) / std[2]
        # Put the channels back together
        x1 = [[[], [], []]]
        x1[0][0] = img_a
        x1[0][1] = img_b
        x1[0][2] = img_c
        return np.asarray(x1)
```

此过程在以 0 为中心的可预测范围内给出模型值。它与应用于训练数据集中图像的预处理相匹配，这是一种标准方法，但每个模型可能会有所不同。

使用 SDK for Python 上传指标

该示例应用程序使用适用于 Python 的软件开发工具包将指标上传到亚马逊 CloudWatch。

Example application.py - SDK for Python

```
    def process_streams(self):
        """Processes one frame of video from one or more video streams."""
        ...
            logger.info('epoch length: {:.3f} s ({:.3f} FPS)'.format(epoch_time,
epoch_fps))
            logger.info('avg inference time: {:.3f} ms'.format(avg_inference_time))
            logger.info('max inference time: {:.3f} ms'.format(max_inference_time))
            logger.info('avg frame processing time: {:.3f}
ms'.format(avg_frame_processing_time))
            logger.info('max frame processing time: {:.3f}
ms'.format(max_frame_processing_time))
            self.inference_time_ms = 0
            self.inference_time_max = 0
            self.frame_time_ms = 0
            self.frame_time_max = 0
            self.epoch_start = time.time()
            self.put_metric_data('AverageInferenceTime', avg_inference_time)
            self.put_metric_data('AverageFrameProcessingTime',
avg_frame_processing_time)

    def put_metric_data(self, metric_name, metric_value):
        """Sends a performance metric to CloudWatch."""
        namespace = 'AWSPanoramaApplication'
        dimension_name = 'Application Name'
```

```
        dimension_value = 'aws-panorama-sample'
        try:
            metric = self.cloudwatch.Metric(namespace, metric_name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{
                    'MetricName': metric_name,
                    'Value': metric_value,
                    'Unit': 'Milliseconds',
                    'Dimensions': [
                        {
                            'Name': dimension_name,
                            'Value': dimension_value
                        },
                        {
                            'Name': 'Device ID',
                            'Value': self.device_id
                        }
                    ]
                }]
            )
            logger.info("Put data for metric %s.%s", namespace, metric_name)
        except ClientError:
            logger.warning("Couldn't put data for metric %s.%s", namespace,
metric_name)
        except AttributeError:
            logger.warning("CloudWatch client is not available.")
```

其权限来自您在部署过程中分配的运行时角色。角色在`aws-panorama-sample.yml` CloudFormation 模板中定义。

Example aws-panorama-sample.yml

```
Resources:
  runtimeRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          -
            Effect: Allow
            Principal:
```

```
            Service:
              - panorama.amazonaws.com
          Action:
            - sts:AssumeRole
      Policies:
        - PolicyName: cloudwatch-putmetrics
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action: 'cloudwatch:PutMetricData'
                Resource: '*'
      Path: /service-role/
```

示例应用程序使用 pip 安装 SDK for Python 和其他依赖项。构建应用程序容器时，`Dockerfile` 会运行命令在基础映像的基础上安装库。

Example Dockerfile

```
FROM public.ecr.aws/panorama/panorama-application
WORKDIR /panorama
COPY . .
RUN pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir -r requirements.txt
```

要在应用程序代码中使用 AWS SDK，请先修改模板以添加应用程序使用的所有 API 操作的权限。`1-create-role.sh`每次进行更改时都要运行来更新 CloudFormation 堆栈。然后，将更改部署到应用程序代码中。

对于修改或使用现有资源的操作，最佳做法是在单独的语句中为目标 `Resource` 指定名称或模式，以尽量缩小此策略的范围。有关每项服务支持的操作和资源的详细信息，请参阅服务授权参考中的操作、资源和条件键

后续步骤

有关使用 AWS Panorama 应用程序 CLI 从头开始构建应用程序和创建程序包的说明，请参阅 CLI 的 README 文件。

- github com/aws/aws-panorama-cli

如需了解更多示例代码和测试工具，以便在部署前验证应用程序代码，请访问 AWS Panorama 示例存储库。

- github com/aws-samples/aws-全景图样本

支持的计算机视觉模型和摄像头

AWS Panorama 支持使用 PyTorch、Apache MXNet 和 TensorFlow构建的模型。当您部署应用程序时，AWS Panorama 会在 A SageMaker I Neo 中编译您的模型。只要您使用与 A SageMaker I Neo 兼容的层，您就可以在 Amazon SageMaker AI 或开发环境中构建模型。

为了处理视频并获取要发送到模型的图像，AWS Panorama 设备使用 RTSP 协议连接到 H.264 编码的视频流。AWS Panorama 测试了各种常见摄像头的兼容性。

Sections

支持的型号

当您为 AWS Panorama 构建应用程序时，您需要提供应用程序用于计算机视觉的机器学习模型。您可以使用模型框架提供的预生成和预训练模型、示例模型或您自己生成和训练的模型。

> **Note**
>
> 有关已使用 AWS Panorama 测试的预构建模型的列表，请参阅模型兼容性。

部署应用程序时，AWS Panorama 使用 A SageMaker I Neo 编译器来编译您的计算机视觉模型。SageMaker AI Neo 是一种编译器，可优化模型以在目标平台上高效运行，目标平台可以是亚马逊弹性计算云 (Amazon EC2) 中的实例，也可以是边缘设备，例如AWS Panorama设备。

AWS Panorama 支持 Apache MXNet 和 TensorFlow A SageMaker I Neo 支持的边缘设备版本。 PyTorch构建自己的模型时，可以使用 SageMaker AI Neo 发行说明中列出的框架版本。在 SageMaker AI 中，您可以使用内置的图像分类算法。

有关在 AWS Panorama 中使用模型的更多信息，请参阅 计算机视觉模型。

支持的摄像头

AWS Panorama 设备支持来自通过本地网络输出 RTSP 的摄像头的 H.264 视频流。对于大于 200 万像素的摄像头流式传输，设备会将图像缩小到 1920x1080 像素或保留流纵横比的等效大小。

以下摄像头型号已经过测试，可与 AWS Panorama 设备兼容：

- 轴 – M3057-PLVE、M3058-PLVE、P1448-LE、P3225-LV Mk II
- LaView— LV PB3 -040W
- Vivotek — 0-H IB936
- Amcrest — M-841B IP2
- Anpviz – IPC-B850W-S-3X、IPC-D250W-S
- WGCC – Dome PoE 4MP ONVIF

有关设备的硬件规格，请参阅 AWS Panorama Appliance 规范。

AWS Panorama Appliance 规范

AWS Panorama Appliance 具有以下硬件规范。对于其他兼容设备，请参阅制造商的文档。

组件	规范
处理器和 GPU	拥有 32GB RAM 的 Nvidia Jetson AGX Xavier
以太网=	2x 1000 Base-T（千兆字节）
USB	1x USB 2.0 和 1x USB 3.0 Type-A 母口
HDMI 输出	2.0a
尺寸	7.75 英寸 x 9.6 英寸 x 1.6 英寸（197毫米 x 243 毫米 x 40 毫米）
重量	3.7 磅（1.7 千克）
电源	交流电 100V-240V 50-60Hz 65W
电源输入	IEC 60320 C6（3 孔）插座
防尘防液保护	IP-62
EMI/EMC 监管合规	FCC 第 15 部分（美国）
热敏触控限制	IEC-62368
工作温度	-20°C 至 60°C
工作湿度	0% 到 95% 相对湿度
存储温度	-20°C 至 85°C
存储湿度	在低温下不受控制。高温下相对湿度为 90%
冷却	强制空气热量提取（风扇）
安装选项	机架安装或独立安装

组件	规范
电源线	6 英尺（1.8 米）
电源控制	按钮
重启	瞬时开关
状态和网络 LEDs	可编程 3 色 RGB LED 指示灯

设备上有 Wi-Fi、蓝牙和 SD 卡存储功能，但不可用。

AWS Panorama Appliance 包括两颗用于安装在服务器机架上的螺丝。您可以在 19 英寸的机架 side-by-side上安装两台设备。

服务限额

AWS Panorama 会对您在账户中创建的资源和部署的应用程序限额。如果您在多个 AWS 区域使用 AWS Panorama，则配额将分别适用于每个区域。AWS Panorama 限额不可调整。

AWS Panorama 中的资源包括设备、应用程序节点软件包和应用程序实例。

- 设备 - 每个区域最多 50 台注册设备。
- 节点软件包 - 每个区域 50 个软件包，每个包最多 20 个版本。
- 应用程序实例 - 每台设备最多 10 个应用程序。每个应用程序最多可以监控 8 个摄像机视频流。每台设备每天的部署量限制为 200。

当您将 AWS Panorama 应用程序 CLI 或 AWS 软件开发工具包与 AWS Panorama 服务配合使用时，配额适用于您进行的 API 调用次数。 AWS Command Line Interface您每秒最多可以发出 5 个请求。创建或修改资源的部分 API 操作会额外应用每秒 1 个请求的限制。

有关限额的完整列表，请访问服务限额控制台，或在 Amazon Web Services 一般参考中查看 AWS Panorama 端点和限额。

AWS Panorama 权限

您可以使用 AWS Identity and Access Management (IAM) 来管理对 AWS Panorama 服务和资源（如设备和应用程序）的访问权限。对于您账户中使用的用户 AWS Panorama，您可以在权限策略中管理权限，该策略可以应用于 IAM 角色。若要管理应用程序的权限，请创建一个角色并将其分配给应用程序。

要管理您账户中用户的权限，请使用 AWS Panorama 提供的托管策略，或自行编写。您需要访问其他 AWS 服务的权限才能获取应用程序和设备日志、查看指标以及为应用程序分配角色。

AWS Panorama 设备还具有授予其访问 AWS 服务和资源的权限的角色。设备的角色是该服务用来代表您访问其他服务的服务角色之一。 AWS Panorama

应用程序角色是您为应用程序创建的单独服务角色，用于授予应用程序使用 AWS 服务的权限 AWS SDK for Python (Boto)。要创建应用程序角色，您需要管理权限或管理员的帮助。

可以按操作所影响的资源，以及在某些情况下借助额外条件来限制用户权限。例如，您可以为应用程序的 Amazon 资源名称 (ARN) 指定模式，要求用户将其用户名包含在其创建的应用程序的名称中。有关各操作支持的资源和条件，请参阅服务授权参考中的AWS Panorama的操作、资源和条件键。

有关更多信息，请参阅 IAM 用户指南中的什么是 IAM？

主题

- AWS Panorama 的基于身份的 IAM 策略
- AWS Panorama 服务角色和跨服务资源
- 向应用程序授予权限

AWS Panorama 的基于身份的 IAM 策略

要授予您账户中的用户访问 AWS Panorama 的权限，您可以在 AWS Identity and Access Management (IAM) 中使用基于身份的策略。将基于身份的策略应用于与用户关联的 IAM 角色。您也可以授予另一个账户中的用户在您的账户中代入角色和访问您的 AWS Panorama 资源的权限。

AWS Panorama 提供托管策略，授予对 AWS Panorama API 操作的访问权限，在某些情况下还可以访问其他服务，用于开发和管理 AWS Panorama 资源。AWS Panorama 根据需要更新托管策略，确保您的用户有权在新功能发布时进行访问。

- AWSPanoramaFullAccess— 提供对 AWS Panorama、亚马逊 S3 中的 AWS Panorama 接入点 AWS Secrets Manager、中的设备证书和亚马逊中的设备日志的完全访问权限 CloudWatch。包括为 AWS Panorama 创建服务相关角色的权限。查看策略

`AWSPanoramaFullAccess` 策略允许您标记 AWS Panorama 资源，但不具有 AWS Panorama 控制台使用的所有与标签相关的权限。要授予这些权限，请添加以下策略。

- ResourceGroupsandTagEditorFullAccess— 查看政策

`AWSPanoramaFullAccess` 策略不包括从 AWS Panorama 控制台购买设备的权限。要授予这些权限，请添加以下策略。

- ElementalAppliancesSoftwareFullAccess— 查看政策

托管策略授予 API 操作的权限，而不限制用户可以修改的资源。要进行更精细的控制，您可以创建自己的策略来限制用户权限的范围。使用完全访问策略作为策略的起点。

> **创建服务角色**
>
> 首次使用 AWS Panorama 控制台时，您需要获得创建 AWS Panorama 设备使用的服务角色的权限。服务角色授予服务管理资源或与其他服务交互的权限。在授予用户访问权限之前，请先创建此角色。

有关可用于限制 AWS Panorama 中用户权限范围的资源和条件的详细信息，请参阅服务授权参考中的 AWS Pnorama 的操作、资源和条件键。

AWS Panorama 服务角色和跨服务资源

AWS Panorama 使用其他 AWS 服务来管理 AWS Panorama 设备、存储数据和导入应用程序资源。服务角色授予服务管理资源或与其他服务交互的权限。当您首次登录 AWS Panorama 控制台时，您将创建以下服务角色：

- AWSServiceRoleForAWSPanorama— 允许 AWS Panorama 管理 AWS 物联网、AWS Secrets Manager 和 AWS Panorama 中的资源。

  托管策略：AWSPanoramaServiceLinkedRolePolicy
- AWSPanoramaApplianceServiceRole— 允许 AWS Panorama 设备将日志上传到 AWS Panorama 创建的 Amazon S3 接入点并从中获取对象。 CloudWatch

  托管策略：AWSPanoramaApplianceServiceRolePolicy

要查看附加到每个角色的权限，请使用 IAM 控制台。在可能的情况下，角色的权限仅限于与 AWS Panorama 使用的命名模式匹配的资源。例如，仅`AWSServiceRoleForAWSPanorama`授予服务访问其名称`panorama`中包含的 AWS IoT 资源的权限。

Sections

保护设备角色

AWS Panorama 设备使用 `AWSPanoramaApplianceServiceRole` 角色访问您账户中的资源。该设备有权将日志上传到日 CloudWatch 志 AWS Secrets Manager、从中读取摄像机直播凭证以及访问 AWS Panorama 创建的亚马逊简单存储服务 (Amazon S3) 接入点中的应用程序项目。

> **Note**
>
> 应用程序不使用设备的权限。要授予您的应用程序使用 AWS 服务的权限，请创建应用程序角色。

AWS Panorama 对您帐户中的所有设备使用相同的服务角色，并且不会跨帐户使用角色。为了增加安全层，您可以修改设备角色的信任策略以明确强制执行此操作，当您使用角色授予服务访问帐户中资源的权限时，这是最佳实践。

要更新设备角色信任策略

1. 在 IAM 控制台中打开设备角色：AWSPanoramaApplianceServiceRole
2. 选择编辑信任关系。
3. 更新策略内容，然后选择更新信任策略。

以下信任策略包含一个条件，该条件确保当 AWS Panorama 担任设备角色时，它会对您账户中的设备执行此操作。`aws:SourceAccount` 条件将 AWS Panorama 指定的账户 ID 与您包含在策略中的账户 ID 进行比较。

Example信任策略 – 特定帐户

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "panorama.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

如果您想进一步限制 AWS Panorama，并允许其仅承担特定设备的角色，您可以通过 ARN 指定设备。`aws:SourceArn` 条件将 AWS Panorama 指定的设备的 ARN 与您包含在策略中的 ARN 进行比较。

Example信任策略 – 单个设备

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "panorama.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:panorama:us-east-1:123456789012:device/
device-lk7exmplpvcr3heqwjmesw76ky"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

如果重置并重新置备设备，则必须暂时删除源 ARN 条件，然后使用新的设备 ID 再次添加。

有关这些条件的更多信息，以及服务使用角色访问账户资源时的安全最佳实践，请参阅 IAM 用户指南中的混淆代理问题。

使用其他服务

AWS Panorama 在以下服务中创建或访问资源：

- AWS IoT – AWS Panorama 设备的事物、策略、证书和作业
- Amazon S3 — 用于暂存应用程序模型、代码和配置的接入点。
- Secrets Manager – AWS Panorama 设备的短期凭证。

有关每项服务的 Amazon 资源名称（ARN） 格式或权限范围的信息，请参阅此列表中链接到的IAM 用户指南中的主题。

向应用程序授予权限

您可以为应用程序创建一个角色以授予其调用 AWS 服务的权限。默认情况下，应用程序没有任何权限。您可以在 IAM 中创建应用程序角色，并在部署期间将其分配给应用程序。要仅向应用程序授予其所需的权限，请为其创建一个具有特定 API 操作权限的角色。

示例应用程序包括用于创建应用程序角色的 CloudFormation 模板和脚本。这是 AWS Panorama 可以担任的服务角色。此角色授予应用程序调用上传指标 CloudWatch 的权限。

Example aws-panorama-sample.yml — 应用程序角色

```
Resources:
  runtimeRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - panorama.amazonaws.com
            Action:
              - sts:AssumeRole
      Policies:
        - PolicyName: cloudwatch-putmetrics
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action: 'cloudwatch:PutMetricData'
                Resource: '*'
      Path: /service-role/
```

您可以通过为 `Action` 的值指定 API 操作或模式列表来扩展此脚本以向其他服务授予权限。

有关 AWS Panorama 中权限的更多信息，请参阅 AWS Panorama 权限。

管理 AWS Panorama 设备

AWS Panorama 设备是运行应用程序的硬件。您可以使用 AWS Panorama 控制台注册设备、更新其软件并向其部署应用程序。 AWS Panorama 设备上的软件连接到摄像机流，向您的应用程序发送视频帧，并在连接的显示器上显示视频输出。

设置好设备或其他兼容设备后，您可以注册摄像机以便与应用程序一起使用。您可以在 AWS Panorama 控制台中管理摄像机直播。部署应用程序时，您可以选择设备向其发送哪些摄像机视频流以供处理。

有关使用示例应用程序介绍 AWS Panorama 设备的教程，请参阅入门 AWS Panorama。

主题

管理 AWS Panorama Appliance

您可以使用 AWS Panorama 控制台配置、升级或注销 AWS Panorama Appliance 和其他兼容设备。

要设置设备，请按照入门教程中的说明操作。设置过程在 AWS Panorama 中创建资源，用于跟踪您的设备并协调更新和部署。

要使用 AWS Panorama API 注册设备，请参阅 自动执行设备注册。

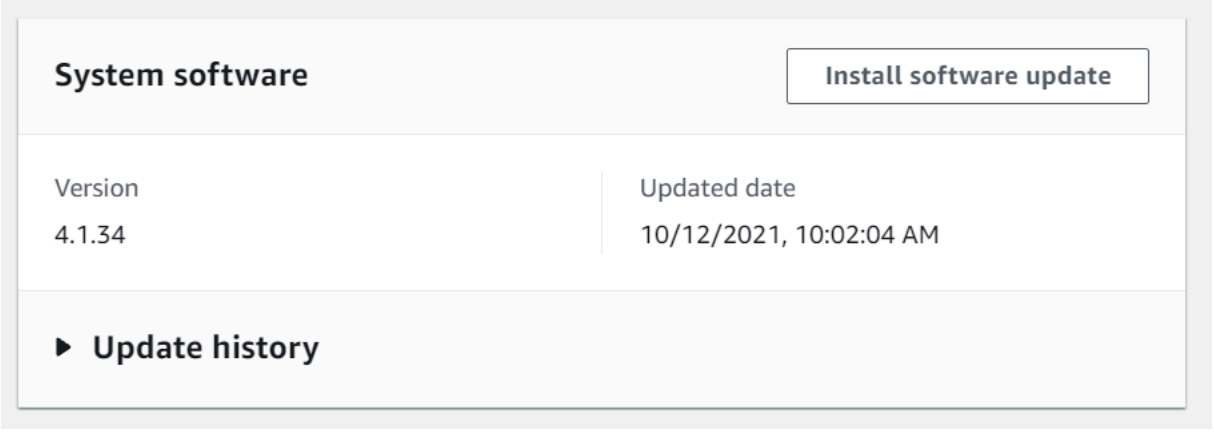
Sections

更新设备软件

您可以在 AWS Panorama 控制台中查看和部署设备的软件更新。更新可以是必需的，也可以是可选的。当必需的更新可用时，控制台会提示您进行应用。您可以在设备设置页面上应用可选更新。

更新设备软件

1. 打开 AWS Panorama 控制台的设备页面。
2. 选择设备。
3. 选择设置。
4. 在系统软件下，选择安装软件更新。



5. 选择新版本，然后选择安装。

注销设备

如果您已使用完设备，则可以使用 AWS Panorama 控制台将其注销并删除相关 AWS IoT 资源。

删除设备

1. 打开 AWS Panorama 控制台的设备页面。
2. 选择设备名称。
3. 选择删除。
4. 输入设备名称并选择删除。

当您从 AWS Panorama 服务中删除设备时，设备上的数据不会自动删除。已取消注册的设备无法连接到 AWS 服务，并且在重置之前无法重新注册。

重启设备

您可以远程重启设备。

重启设备

1. 打开 AWS Panorama 控制台的设备页面。
2. 选择设备名称。
3. 选择重启。

控制台会向设备发送一条消息，要求重启设备。要接收信号，设备必须能够连接 AWS IoT。要使用 AWS Panorama API 重启设备，请参阅 重新启动设备。

重置设备

要在不同区域或通过不同账户使用设备，必须重置设备并用新的证书重新预置。重置设备会应用所需的最新软件版本并删除所有账户数据。

要开始重置操作，必须接通设备电源并关机。同时按住电源键和重置键 5 秒钟。松开按钮后，状态指示灯会闪烁橙色。等到状态指示灯闪烁绿色后再预置设备或断开设备连接。

您也可以重置设备软件，而不删除设备上的证书。有关更多信息，请参阅 电源和重置按钮。

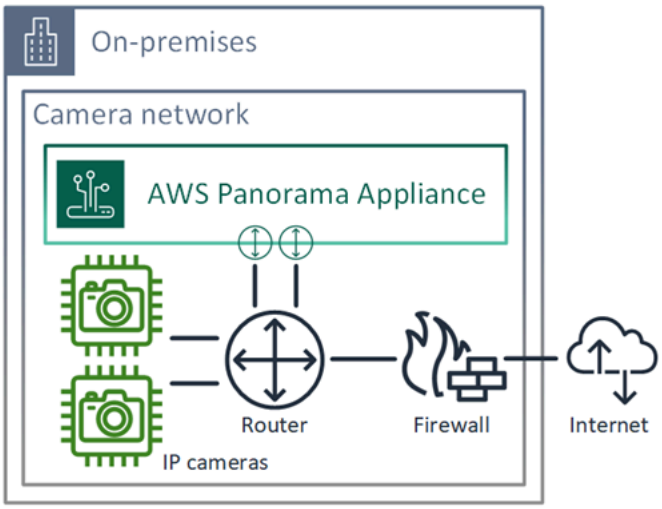
将 AWS Panorama Appliance 连接到您的网络

AWS Panorama 设备需要同时连接到 AWS 云端和您的本地 IP 摄像机网络。您可以将设备连接到允许同时访问两个网络的单个防火墙，或将设备的两个网络接口分别连接到不同的子网。无论哪种情况，您都必须保护设备的网络连接安全性，以防止未经授权访问您的摄像机视频流。

Sections

单个网络配置

设备有两个以太网端口。如果您通过单个路由器连接进出设备的所有流量，则可以使用第二个端口实现冗余，以防与第一个端口的物理连接中断。配置您的路由器，使设备只能连接到摄像机视频流和互联网，并阻止摄像机视频流以其他方式离开内部网络。

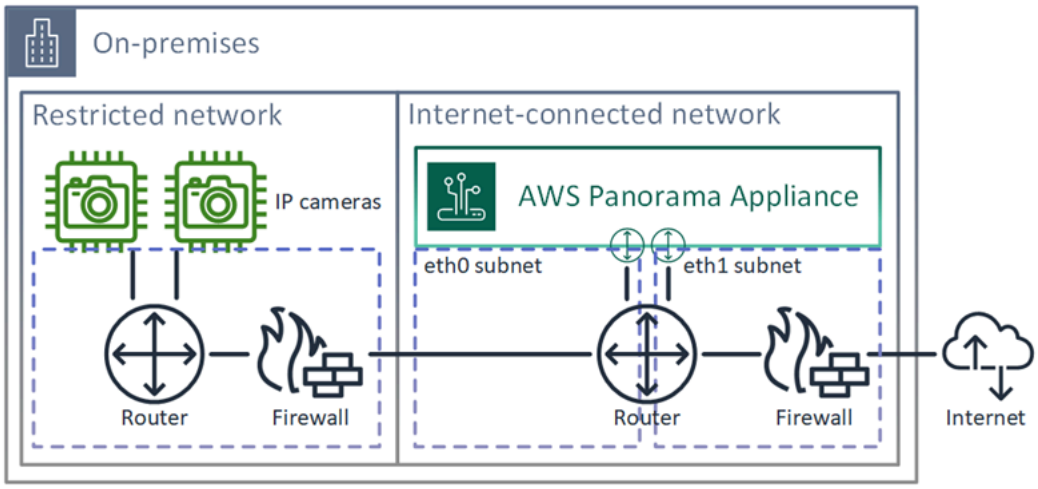


有关设备需要访问的端口和端点的详细信息，请参阅 配置服务访问权限 和 配置本地网络访问权限。

双重网络配置

为提高安全性，您可以将设备置于与摄像机网络不同的互联网连接网络中。受限摄像机网络和设备网络之间的防火墙仅允许设备访问视频流。如果出于安全考虑，您的摄像机网络之前采用了气隙系统，您可能更喜欢这种方法，而不是将摄像机网络连接到同时允许访问互联网的路由器上。

以下示例显示了设备在每个端口上连接到不同的子网。路由器将 `eth0` 接口置于连接到摄像机网络的子网上，并将 `eth1` 接口置于连接到互联网的子网上。



您可以在 AWS Panorama 控制台中确认每个端口的 IP 地址和 MAC 地址。

配置服务访问权限

在预置期间，您可以将设备配置为请求特定的 IP 地址。提前选择 IP 地址以简化防火墙配置，并确保设备在长时间离线时地址不会变化。

该设备使用 AWS 服务来协调软件更新和部署。配置防火墙以允许设备连接到这些端点。

互联网访问

- AWS IoT（HTTPS 和 MQTT，端口 443、8443 和 8883）以及设备管理端点 AWS IoT Core 。有关详细信息，请参阅 Amazon Web Services 一般参考中的 AWS IoT Device Management 端点和限额。
- AWS IoT 凭证（HTTPS，端口 443）`credentials.iot.<region>.amazonaws.com`和子域名。
- Amazon Elastic 容器注册表（HTTPS、端口 443）- `api.ecr.<region>.amazonaws.com`、`dkr.ecr.<region>.amazonaws.com` 和子域。

- 亚马逊 CloudWatch （HTTPS，端口 443）—monitoring.<region>.amazonaws.com.
- 亚马逊 CloudWatch 日志（HTTPS，端口 443）— logs.<region>.amazonaws.com。
- Amazon Simple Storage Service（HTTPS、端口 443）- s3.<region>.amazonaws.com、s3-accesspoint.<region>.amazonaws.com 和子域。

如果您的应用程序调用其他 AWS 服务，则设备还需要访问这些服务的终端节点。有关更多信息，请参阅服务端点和限额。

配置本地网络访问权限

设备需要在本地访问 RTSP 视频流，但不能通过互联网访问。配置防火墙，允许设备在内部通过 554 端口访问 RTSP 视频流，且不允许视频流向互联网或从互联网传入。

本地访问

- 实时流媒体协议（RTSP、端口 554）- 用于读取摄像机视频流。
- 网络时间协议（NTP、端口 123）- 用于设备的时钟保持同步。如果您没有在网络上运行 NTP 服务器，则设备也可以通过互联网连接到公共 NTP 服务器。

私有连接

如果您将 AWS Panorama 设备部署在具有 VPN 连接的私有 VPC 子网中，则无需访问互联网 AWS。您可以使用 Site-to-Site VPN 或 Direct Connect 在本地路由器和之间创建 VPN 连接 AWS。在您的私有 VPC 子网中，您可以创建终端节点，让设备连接到 Amazon 简单存储服务和其他服务。 AWS IoT 有关更多信息，请参阅 将设备连接到私有子网。

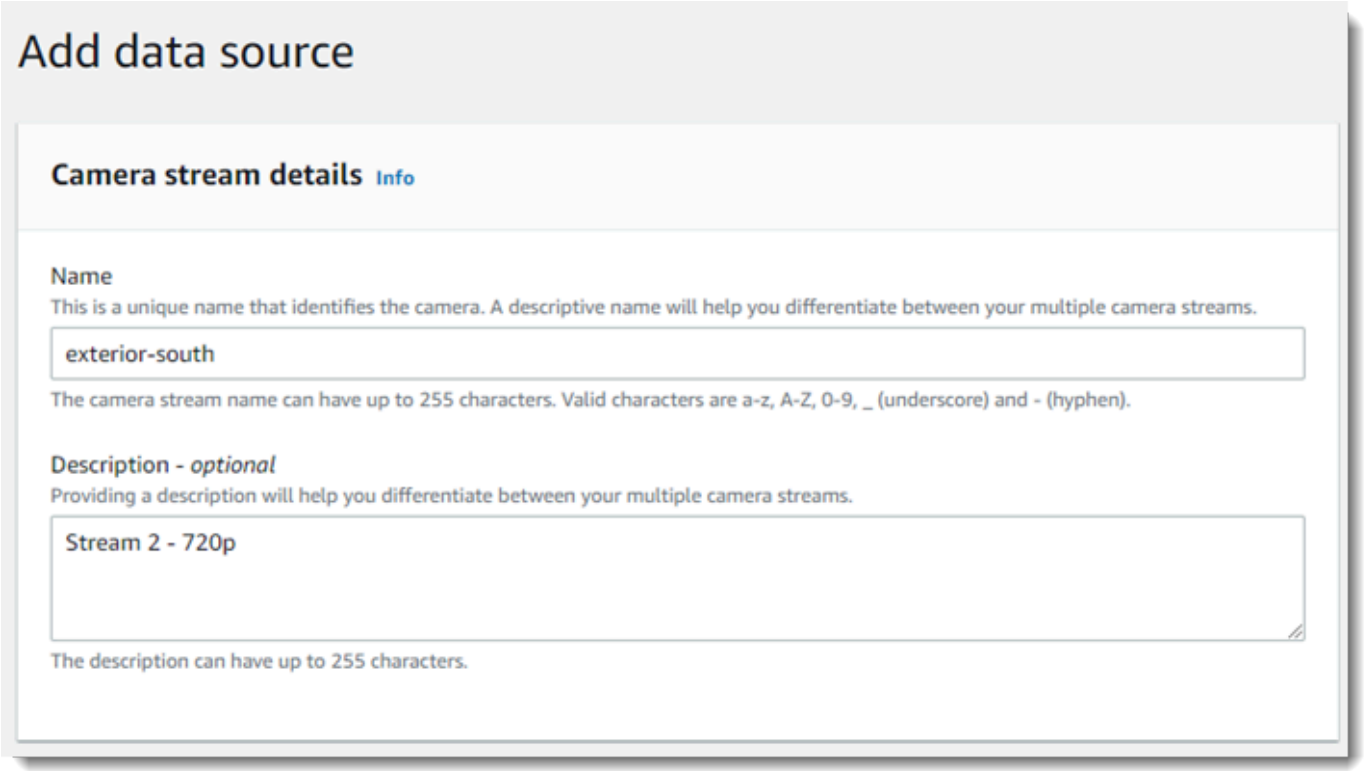
在 AWS Panorama 中管理摄像机视频流

要将视频流注册为应用程序的数据来源，请使用 AWS Panorama 控制台。一个应用程序可以同时处理多个流，多个设备可以连接到同一个流。

> **Important**
>
> 应用程序可以连接到任何可从其连接的本地网络路由的摄像机视频流。为确保视频流安全，请将您的网络配置为仅允许本地 RTSP 流量。有关更多信息，请参阅 AWS Panorama 中的安全性。

注册摄像机视频流

1. 打开 AWS Panorama 控制台的“数据来源”页面。
2. 选择添加数据来源。



3. 配置以下设置。

   - 名称 - 摄像机视频流的名称。
   - 描述 - 对摄像机、其位置或其他详细信息的简短描述。

- RTSP URL - 指定摄像机 IP 地址和流路径的 URL。例如，`rtsp://192.168.0.77/live/mpeg4/`
- 凭证 - 如果使用密码保护摄像机视频流，请指定用户名和密码。

4. 选择保存。

要使用 AWS Panorama API 注册摄像机视频流，请参阅 自动执行设备注册。

有关与 AWS Panorama Appliance 兼容的摄像机列表，请参阅 支持的计算机视觉模型和摄像头。

移除流

您可以在 AWS Panorama 控制台中删除摄像机视频流。

移除摄像机视频流

1. 打开 AWS Panorama 控制台的“数据来源”页面。
2. 选择摄像机视频流。
3. 选择删除数据来源。

从服务中移除摄像机视频流不会使应用程序停止运行，也不会从 Secrets Manager 中删除摄像机凭证。要删除密钥，请使用 Secrets Manager 控制台。

在 AWS Panorama Appliance 上管理应用程序

应用程序是代码、模型和配置的组合。在 AWS Panorama 控制台的设备页面，您可以管理设备上的应用程序。

在 AWS Panorama Appliance 上管理应用程序

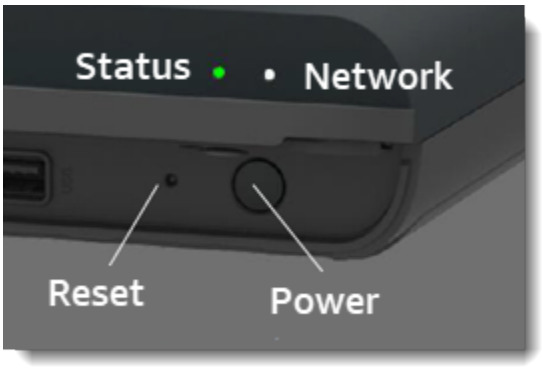
1. 打开 AWS Panorama 控制台的设备页面。
2. 选择设备。

已部署的应用程序页面显示已部署到设备的应用程序。

使用此页面上的选项从设备中移除已部署的应用程序，或者用新版本替换正在运行的应用程序。您也可以克隆应用程序（正在运行的或已删除的应用程序），以部署其新副本。

AWS Panorama 设备按钮和指示灯

AWS Panorama 设备的电源按钮上方有两个 LED 灯，用于指示设备状态和网络连接。



状态指示灯

LEDs 更改颜色并闪烁以指示状态。缓慢闪烁是每三秒一次。快速闪烁是每秒一次。

状态 LED 状态

- 绿灯快速闪烁 – 设备正在启动。
- 绿灯长亮 – 设备运行正常。
- 慢速闪烁蓝色-设备正在复制配置文件并尝试向 AWS IoT注册。
- 蓝灯快速闪烁 – 设备正在将日志映像复制到 USB 驱动器。
- 红灯快速闪烁 – 设备在启动过程中遇到错误或过热。
- 橙灯缓慢闪烁 – 设备正在恢复最新的软件版本。
- 橙灯快速闪烁 – 设备正在恢复最低软件版本。

网络灯

网络 LED 具有以下状态：

网络 LED 状态

- 绿灯长亮 – 以太网电缆已连接。
- 绿色闪烁 – 设备正在通过网络进行通信。
- 红灯长亮 – 未连接以太网电缆。

电源和重置按钮

电源和重置按钮位于设备正面的保护盖下方。重置按钮较小且凹陷。使用小螺丝刀或回形针按压。

重置设备

1. 设备必须插入电源并关闭电源。要关闭设备电源，请按住电源按钮 1 秒钟，然后等待关机序列完成。关机序列大约需要 10 秒。
2. 要重置设备，请使用以下按钮组合。短按一次为 1 秒。长按一次为 5 秒。对于需要多个按钮的操作，请同时按住两个按钮。

   - 完全重置 – 长按电源并重置。

     恢复最低软件版本并删除所有配置文件和应用程序。
   - 恢复最新软件版本 – 短按重置。

     将最新的软件更新重新应用于设备。
   - 恢复最低软件版本 – 长按重置。

     将所需的最新软件更新重新应用于设备。
3. 松开两个按钮。设备通电，状态指示灯呈橙色闪烁几分钟。
4. 设备准备就绪后，状态指示灯呈绿色闪烁。

重置设备不会将其从 AWS Panorama 服务中删除。有关更多信息，请参阅 注销设备。

管理 AWS Panorama 应用程序

应用程序在 AWS Panorama 设备上运行，用于对视频流执行计算机视觉任务。您可以通过组合 Python 代码和机器学习模型来构建计算机视觉应用程序，然后通过互联网将它们部署到 AWS Panorama 设备上。应用程序可将视频发送到显示器，也可以使用 AWS SDK 将结果发送到 AWS 服务。

主题

部署应用程序

要部署应用程序，您可以使用 AWS Panorama 应用程序 CLI 将其导入您的账户，构建容器，上传和注册资产，然后创建应用程序实例。本主题详细介绍了其中的每个步骤，并介绍后台操作。

如果您尚未部署应用程序，请参阅 入门 AWS Panorama 以获取演练。

有关定制和扩展示例应用程序的详细信息，请参阅 构建 AWS Panorama 应用程序。

Sections

- 安装 AWS Panorama 应用程序 CLI
- 导入应用程序
- 构建容器映像
- 导入模型
- 上传应用程序资产
- 使用 AWS Panorama 控制台部署应用程序
- 自动化应用程序部署

安装 AWS Panorama 应用程序 CLI

要安装 AWS Panorama 应用程序 CLI 和 AWS CLI，请使用 pip。

```
$ pip3 install --upgrade awscli panoramacli
```

要使用 AWS Panorama 应用程序 CLI 构建应用程序映像，您需要使用 Docker。在 Linux 上，还需要 qemu 和相关系统库。有关安装和配置 AWS Panorama 应用程序 CLI 的更多信息，请参阅项目 GitHub 存储库中的自述文件。

- gitub。 com/aws/aws-panorama-cli

有关使用在 Windows 中设置构建环境的说明 WSL2，请参阅在 Windows 中设置开发环境。

导入应用程序

如果您正在使用示例应用程序或第三方提供的应用程序，请使用 AWS Panorama 应用程序 CLI 导入该应用程序。

```
my-app$ panorama-cli import-application
```

此命令使用您的账户 ID 重命名应用程序软件包。软件包名称以其部署账户的账户 ID 开头。将应用程序部署到多个账户时，必须为每个账户分别导入和打包应用程序。

例如，本指南的示例应用程序是一个代码包和一个模型包，每个包都以占位符账户 ID 命名。该`import-application`命令将它们重命名为使用 CLI 从您的工作空间凭据中推断出的账户 ID。AWS

```
/aws-panorama-sample
### assets
### graphs
#   ### my-app
#       ### graph.json
### packages
    ### 123456789012-SAMPLE_CODE-1.0
    #   ### Dockerfile
    #   ### application.py
    #   ### descriptor.json
    #   ### package.json
    #   ### requirements.txt
    #   ### squeezenet_classes.json
    ### 123456789012-SQUEEZENET_PYTORCH-1.0
        ### descriptor.json
        ### package.json
```

在软件包目录名称和指代这些包的应用程序清单 (`graph.json`) 中，123456789012 将被替换为您的账户 ID。您可以致电`aws sts get-caller-identity`确认您的账户 ID AWS CLI。

```
$ aws sts get-caller-identity
{
    "UserId": "AIDAXMPL7W66UC3GFXMPL",
    "Account": "210987654321",
    "Arn": "arn:aws:iam::210987654321:user/devenv"
}
```

构建容器映像

您的应用程序代码打包在 Docker 容器映像中，其中包括您在 Dockerfile 中安装的应用程序代码和库。使用 AWS Panorama 应用程序 CLI `build-container` 命令构建 Docker 映像并导出文件系统映像。

```
my-app$ panorama-cli build-container --container-asset-name code_asset --package-path
 packages/210987654321-SAMPLE_CODE-1.0
{
    "name": "code_asset",
    "implementations": [
        {
            "type": "container",
            "assetUri":
 "5fa5xmplbc8c16bf8182a5cb97d626767868d3f4d9958a4e49830e1551d227c5.tar.gz",
            "descriptorUri":
 "1872xmpl129481ed053c52e66d6af8b030f9eb69b1168a29012f01c7034d7a8f.json"
        }
    ]
}
Container asset for the package has been succesfully built at
 assets/5fa5xmplbc8c16bf8182a5cb97d626767868d3f4d9958a4e49830e1551d227c5.tar.gz
```

此命令创建名为 `code_asset` 的 Docker 映像，并将文件系统导出到 `assets` 文件夹中的 `.tar.gz` 存档中。CLI 会根据应用程序 Dockerfile 所述，从 Amazon Elastic 容器注册表 (Amazon ECR) 中提取应用程序基础映像。

除容器存档外，CLI 还会为程序包描述符 (`descriptor.json`) 创建资产。两个文件都使用反映了原始文件哈希值的唯一标识符重命名。AWS Panorama 应用程序 CLI 还在程序包配置中添加了一个块，用于记录这两种资产的名称。这些名称会由设备在部署过程中使用。

Example packages/123456789012-SAMPLE_CODE-1.0/package.json – 带资产块

```
{
    "nodePackage": {
        "envelopeVersion": "2021-01-01",
        "name": "SAMPLE_CODE",
        "version": "1.0",
        "description": "Computer vision application code.",
        "assets": [
            {
                "name": "code_asset",
                "implementations": [
                    {
                        "type": "container",
                        "assetUri":
 "5fa5xmplbc8c16bf8182a5cb97d626767868d3f4d9958a4e49830e1551d227c5.tar.gz",
```

```
                    "descriptorUri":
"1872xmpl129481ed053c52e66d6af8b030f9eb69b1168a29012f01c7034d7a8f.json"
                }
            ]
        }
    ],
    "interfaces": [
        {
            "name": "interface",
            "category": "business_logic",
            "asset": "code_asset",
            "inputs": [
                {
                    "name": "video_in",
                    "type": "media"
                },
```

build-container 命令中指定的代码资产名称必须与程序包配置中 asset 字段的值相匹配。在前面的示例中，两个值都是 code_asset。

导入模型

您的应用程序的资产文件夹中可能有模型存档，或者您可以单独下载。如果您有新模型、更新的模型或更新的模型描述符文件，请使用 add-raw-model 命令将其导入。

```
my-app$ panorama-cli add-raw-model --model-asset-name model_asset \
      --model-local-path my-model.tar.gz \
      --descriptor-path packages/210987654321-SQUEEZENET_PYTORCH-1.0/descriptor.json \
      --packages-path packages/210987654321-SQUEEZENET_PYTORCH-1.0
```

如果您只需要更新描述符文件，则可以复用资产目录中的现有模型。您可能需要更新描述符文件，以配置诸如浮点精度模式等功能。例如，以下脚本展示了如何使用示例应用程序执行该操作。

Example util-scripts/ .sh update-model-config

```
#!/bin/bash
set -eo pipefail
MODEL_ASSET=fd1axmplacc3350a5c2673adacffab06af54c3f14da6fe4a8be24cac687a386e
MODEL_PACKAGE=SQUEEZENET_PYTORCH
ACCOUNT_ID=$(ls packages | grep -Eo '[0-9]{12}' | head -1)
```

```
panorama-cli add-raw-model --model-asset-name model_asset --model-local-path assets/
${MODEL_ASSET}.tar.gz --descriptor-path packages/${ACCOUNT_ID}-${MODEL_PACKAGE}-1.0/
descriptor.json --packages-path packages/${ACCOUNT_ID}-${MODEL_PACKAGE}-1.0
cp packages/${ACCOUNT_ID}-${MODEL_PACKAGE}-1.0/package.json packages/${ACCOUNT_ID}-
${MODEL_PACKAGE}-1.0/package.json.bup
```

在您使用 CLI 重新导入模型程序包目录中的描述符文件之前，不会应用对描述符文件所做的更改。CLI 使用新的资产名称更新模型程序包配置，类似于在重建容器时更新应用程序代码包配置的方式。

上传应用程序资产

要上传和注册应用程序资产，包括模型存档、容器文件系统存档及其描述符文件，请使用 package-application 命令。

```
my-app$ panorama-cli package-application
Uploading package SQUEEZENET_PYTORCH
Patch version for the package
 5d3cxmplb7113faa1d130f97f619655d8ca12787c751851a0e155e50eb5e3e96
Deregistering previous patch version
 e845xmpl8ea0361eb345c313a8dded30294b3a46b486dc8e7c174ee7aab29362
Asset fd1axmplacc3350a5c2673adacffab06af54c3f14da6fe4a8be24cac687a386e.tar.gz already
 exists, ignoring upload
upload: assets/87fbxmpl6f18aeae4d1e3ff8bbc6147390feaf47d85b5da34f8374974ecc4aaf.json
 to s3://arn:aws:s3:us-east-2:212345678901:accesspoint/
panorama-210987654321-6k75xmpl2jypelgzst7uux62ye/210987654321/nodePackages/
SQUEEZENET_PYTORCH/
binaries/87fbxmpl6f18aeae4d1e3ff8bbc6147390feaf47d85b5da34f8374974ecc4aaf.json
Called register package version for SQUEEZENET_PYTORCH with patch version
 5d3cxmplb7113faa1d130f97f619655d8ca12787c751851a0e155e50eb5e3e96
...
```

如果资产文件或程序包配置没有更改，CLI 会跳过。

```
Uploading package SAMPLE_CODE
Patch Version ca91xmplca526fe3f07821fb0c514f70ed0c444f34cb9bd3a20e153730b35d70 already
 registered, ignoring upload
Register patch version complete for SQUEEZENET_PYTORCH with patch version
 5d3cxmplb7113faa1d130f97f619655d8ca12787c751851a0e155e50eb5e3e96
Register patch version complete for SAMPLE_CODE with patch version
 ca91xmplca526fe3f07821fb0c514f70ed0c444f34cb9bd3a20e153730b35d70
All packages uploaded and registered successfully
```

CLI 会将每个程序包的资产上传到您账户特定的 Amazon S3 接入点。AWS Panorama 为您管理接入点，并通过 DescribePackageAPI 提供有关接入点的信息。CLI 将每个程序包的资产上传到为其提供的位置，并按照程序包配置描述的设置将其注册到 AWS Panorama 服务。

使用 AWS Panorama 控制台部署应用程序

您可以使用 AWS Panorama 控制台部署应用程序。在部署过程中，您可以选择将哪些摄像机视频流传递给应用程序代码，并配置应用程序开发人员提供的选项。

部署应用程序

1. 打开 AWS Panorama 控制台已部署的应用程序页面。
2. 选择部署应用程序。
3. 将应用程序清单的内容 `graph.json` 粘贴到文本编辑器中。选择下一步。
4. 输入名称和描述。
5. 选择继续部署。
6. 选择开始部署。
7. 如果您的应用程序使用角色，请从下拉菜单中选择。选择下一步。
8. 选择选择设备，然后选择您的设备。选择下一步。
9. 在选择数据源步骤中，选择查看输入，然后将摄像头流添加为数据源。选择下一步。
10. 在配置步骤中，配置开发人员定义的所有特定于应用程序的设置。选择下一步。
11. 选择部署，然后选择完成。
12. 在已部署的应用程序列表中，选择要监控其状态的应用程序。

部署过程需要 15 到 20 分钟。当应用程序启动时，设备的输出可能长时间处于空白状态。如果您遇到错误消息，请参阅 故障排除。

自动化应用程序部署

您可以使用 CreateApplicationInstanceAPI 自动执行应用程序部署过程。API 采用两个配置文件作为输入。应用程序清单说明了使用的程序包及其关系。第二个文件是覆盖文件，用于指定部署时对应用程序清单中值的覆盖。使用覆盖文件可以让您使用相同的应用程序清单，以不同的摄像机数据视频流部署应用程序，并配置其他应用程序的特定设置。

有关更多信息以及本主题中每个步骤的示例脚本，请参阅 自动化应用程序部署。

在 AWS Panorama 控制台中管理应用程序

使用 AWS Panorama 控制台管理已部署的应用程序。

Sections

更新或复制应用程序

要更新应用程序，请使用替换选项。替换应用程序时，您可以更新其代码或模型。

要更新应用程序

1. 打开 AWS Panorama 控制台已部署的应用程序页面。
2. 选择应用程序。
3. 选择替换。
4. 按照说明创建新版本或应用程序。

还有一个克隆选项，其作用类似于替换，但不会删除旧版本的应用程序。您可以使用此选项在不停止正在运行的版本的情况下测试对应用程序的更改，或重新部署已删除的版本。

删除版本和应用程序

要清理未使用的应用程序版本，请将其从设备中删除。

删除 应用程序

1. 打开 AWS Panorama 控制台已部署的应用程序页面。
2. 选择应用程序。
3. 选择从设备中删除。

程序包配置

当您使用 AWS Panorama 应用 CLI 命令 `panorama-cli package-application` 时，CLI 会将应用程序的资产上传到 Amazon S3 并将其注册到 AWS Panorama。资产包括二进制文件（容器映像和模型）和描述符文件，AWS Panorama 设备在部署期间下载这些文件。若要注册包的资产，请提供一个单独的包配置文件，用于定义包、其资产和接口。

以下示例显示了具有一个输入和一个输出的代码节点的包配置。视频输入提供对来自摄像头流式传输的图像数据的访问。输出节点将处理后的图像发送到显示器。

Example packages/1234567890-SAMPLE_CODE-1.0/package.json

```
{
    "nodePackage": {
        "envelopeVersion": "2021-01-01",
        "name": "SAMPLE_CODE",
        "version": "1.0",
        "description": "Computer vision application code.",
        "assets": [
            {
                "name": "code_asset",
                "implementations": [
                    {
                        "type": "container",
                        "assetUri":
 "3d9bxmplbdb67a3c9730abb19e48d78780b507f3340ec3871201903d8805328a.tar.gz",
                        "descriptorUri":
 "1872xmpl129481ed053c52e66d6af8b030f9eb69b1168a29012f01c7034d7a8f.json"
                    }
                ]
            }
        ],
        "interfaces": [
            {
                "name": "interface",
                "category": "business_logic",
                "asset": "code_asset",
                "inputs": [
                    {
                        "name": "video_in",
                        "type": "media"
                    }
                ],
```

```
                "outputs": [
                    {
                        "description": "Video stream output",
                        "name": "video_out",
                        "type": "media"
                    }
                ]
            }
        ]
    }
}
```

assets 部分指定 AWS Panorama 应用 CLI 上传到 Amazon S3 的构件的名称。如果您导入示例应用程序或来自其他用户的应用程序，则此部分可以为空，也可以引用不在您的帐户中的资产。当您运行 panorama-cli package-application 时，AWS Panorama 应用 CLI 会使用正确的值填充此部分。

AWS Panorama 应用程序清单

部署应用程序时，您需要提供一个名为应用程序清单的配置文件。此文件将应用程序定义为具有节点和边缘的图形。应用程序清单是应用程序源代码的一部分，存储在 graphs 目录中。

Example graphs/aws-panorama-sample/graph.json

```
{
    "nodeGraph": {
        "envelopeVersion": "2021-01-01",
        "packages": [
            {
                "name": "123456789012::SAMPLE_CODE",
                "version": "1.0"
            },
            {
                "name": "123456789012::SQUEEZENET_PYTORCH_V1",
                "version": "1.0"
            },
            {
                "name": "panorama::abstract_rtsp_media_source",
                "version": "1.0"
            },
            {
                "name": "panorama::hdmi_data_sink",
                "version": "1.0"
            }
        ],
        "nodes": [
            {
                "name": "code_node",
                "interface": "123456789012::SAMPLE_CODE.interface"
            }
            {
                "name": "model_node",
                "interface": "123456789012::SQUEEZENET_PYTORCH_V1.interface"
            },
            {
                "name": "camera_node",
                "interface": "panorama::abstract_rtsp_media_source.rtsp_v1_interface",
                "overridable": true,
                "overrideMandatory": true,
                "decorator": {
```

```
                    "title": "IP camera",
                    "description": "Choose a camera stream."
                }
            },
            {
                "name": "output_node",
                "interface": "panorama::hdmi_data_sink.hdmi0"
            },
            {
                "name": "log_level",
                "interface": "string",
                "value": "INFO",
                "overridable": true,
                "decorator": {
                    "title": "Logging level",
                    "description": "DEBUG, INFO, WARNING, ERROR, or CRITICAL."
                }
            }
            ...
        ],
        "edges": [
            {
                "producer": "camera_node.video_out",
                "consumer": "code_node.video_in"
            },
            {
                "producer": "code_node.video_out",
                "consumer": "output_node.video_in"
            },
            {
                "producer": "log_level",
                "consumer": "code_node.log_level"
            }
        ]
    }
}
```

节点通过边缘连接，边缘指定了节点输入和输出之间的映射。一个节点的输出连接到另一个节点的输入，形成一个图形。

JSON 架构

应用程序清单和覆盖文档的格式在 JSON 架构中定义。您可以在部署之前使用 JSON 架构来验证配置文档。JSON 架构可在本指南的 GitHub 存储库中找到。

- JSON 架构 — aws-panorama-developer-guide/资源

应用程序节点

节点包括模型、代码、相机流、输出和参数。节点有一个接口，用于定义其输入和输出。该接口可以在您账户中的程序包、AWS Panorama 提供的程序包或内置类型中定义。

在以下示例中，code_node 和 model_node 引用示例应用程序附带的示例代码和模型包。camera_node 使用 AWS Panorama 提供的软件包为您在部署期间指定的相机流创建占位符。

Example graph.json - 节点

```
        "nodes": [
            {
                "name": "code_node",
                "interface": "123456789012::SAMPLE_CODE.interface"
            },
            {
                "name": "model_node",
                "interface": "123456789012::SQUEEZENET_PYTORCH_V1.interface"
            },
            {
                "name": "camera_node",
                "interface": "panorama::abstract_rtsp_media_source.rtsp_v1_interface",
                "overridable": true,
                "overrideMandatory": true,
                "decorator": {
                    "title": "IP camera",
                    "description": "Choose a camera stream."
                }
            }
    ]
```

Edges

边缘将一个节点的输出映射到另一个节点的输入。在以下示例中，第一个边缘将相机流节点的输出映射到应用程序代码节点的输入。名称 video_in 和 video_out 在节点包的接口中定义。

Example graph.json – 边缘

```
        "edges": [
            {
                "producer": "camera_node.video_out",
                "consumer": "code_node.video_in"
```

```
        },
        {
            "producer": "code_node.video_out",
            "consumer": "output_node.video_in"
        },
```

在应用程序代码中，使用 `inputs` 和 `outputs` 属性从输入流中获取图像，并将图像发送到输出流。

Example application.py – 视频输入和输出

```
    def process_streams(self):
        """Processes one frame of video from one or more video streams."""
        frame_start = time.time()
        self.frame_num += 1
        logger.debug(self.frame_num)
        # Loop through attached video streams
        streams = self.inputs.video_in.get()
        for stream in streams:
            self.process_media(stream)
        ...
        self.outputs.video_out.put(streams)
```

抽象节点

在应用程序清单中，抽象节点是指由 AWS Panorama 定义的程序包，您可以将其用作应用程序清单中的占位符。AWS Panorama 提供两种类型的抽象节点。

- 相机流 – 选择应用程序在部署期间使用的相机流。

  软件包名称 – `panorama::abstract_rtsp_media_source`

  接口名称 – `rtsp_v1_interface`
- HDMI 输出 – 指示应用程序输出视频。

  软件包名称 – `panorama::hdmi_data_sink`

  接口名称 – `hdmi0`

以下示例显示了一组基本的包、节点和边缘，用于处理相机流并将视频输出到显示器的应用程序。相机节点使用 AWS Panorama 中 `abstract_rtsp_media_source` 包的接口，可以接受多个相机流作为

输入。引用 hdmi_data_sink 的输出节点允许应用程序代码访问从设备的 HDMI 端口输出的视频缓冲区。

Example graph.json – 抽象节点

```
{
    "nodeGraph": {
        "envelopeVersion": "2021-01-01",
        "packages": [
            {
                "name": "123456789012::SAMPLE_CODE",
                "version": "1.0"
            },
            {
                "name": "123456789012::SQUEEZENET_PYTORCH_V1",
                "version": "1.0"
            },
            {
                "name": "panorama::abstract_rtsp_media_source",
                "version": "1.0"
            },
            {
                "name": "panorama::hdmi_data_sink",
                "version": "1.0"
            }
        ],
        "nodes": [
            {
                "name": "camera_node",
                "interface": "panorama::abstract_rtsp_media_source.rtsp_v1_interface",
                "overridable": true,
                "decorator": {
                    "title": "IP camera",
                    "description": "Choose a camera stream."
                }
            },
            {
                "name": "output_node",
                "interface": "panorama::hdmi_data_sink.hdmi0"
            }
        ],
        "edges": [
            {
                "producer": "camera_node.video_out",
```

```
                "consumer": "code_node.video_in"
            },
            {
                "producer": "code_node.video_out",
                "consumer": "output_node.video_in"
            }
        ]
    }
}
```

应用程序参数

参数是具有基本类型的节点，可以在部署期间覆盖。参数可以具有默认值和修饰器，用于指示应用程序用户如何配置它。

参数类型

- `string` – 字符串。例如，`DEBUG`。
- `int32` – 整数。例如，`20`
- `float32` – 浮点数。例如，`47.5`
- `boolean` – `true` 或 `false`。

下面的示例演示两个参数，一个字符串和一个数字，它们作为输入发送到代码节点。

Example graph.json – 参数

```
       "nodes": [
           {
               "name": "detection_threshold",
               "interface": "float32",
               "value": 20.0,
               "overridable": true,
               "decorator": {
                   "title": "Threshold",
                   "description": "The minimum confidence percentage for a positive
classification."
               }
           },
           {
               "name": "log_level",
               "interface": "string",
               "value": "INFO",
               "overridable": true,
               "decorator": {
                   "title": "Logging level",
                   "description": "DEBUG, INFO, WARNING, ERROR, or CRITICAL."
               }
           }
           ...
       ],
```

```
        "edges": [
            {
                "producer": "detection_threshold",
                "consumer": "code_node.threshold"
            },
            {
                "producer": "log_level",
                "consumer": "code_node.log_level"
            }
            ...
        ]
    }
```

可以直接在应用程序清单中修改参数，也可以在部署时提供具有替代的新值。有关更多信息，请参阅带覆盖功能的部署时配置。

带覆盖功能的部署时配置

您可以在部署期间配置参数和抽象节点。如果您使用 AWS Panorama 控制台进行部署，则可以为每个参数指定一个值，并选择相机流式传输作为输入。如果您使用 AWS Panorama API 部署应用程序，则可以使用覆盖文档指定这些设置。

覆盖文档在结构上与应用程序清单类似。对于具有基本类型的参数，您可以定义一个节点。对于相机流式传输，您可以定义映射到已注册相机流式传输的节点和包。然后，为每个节点定义一个覆盖，该覆盖指定它所替换的应用程序清单中的节点。

Example overrides.json

```
{
    "nodeGraphOverrides": {
        "nodes": [
            {
                "name": "my_camera",
                "interface": "123456789012::exterior-south.exterior-south"
            },
            {
                "name": "my_region",
                "interface": "string",
                "value": "us-east-1"
            }
        ],
        "packages": [
            {
                "name": "123456789012::exterior-south",
                "version": "1.0"
            }
        ],
        "nodeOverrides": [
            {
                "replace": "camera_node",
                "with": [
                    {
                        "name": "my_camera"
                    }
                ]
            },
            {
                "replace": "region",
                "with": [
```

```
                    {
                        "name": "my_region"
                    }
                ]
            }
        ],
        "envelopeVersion": "2021-01-01"
    }
}
```

在前面的示例中，该文档定义了一个字符串参数和一个抽象相机节点的覆盖。nodeOverrides 告诉 AWS Panorama 本文档中的哪些节点会覆盖应用程序清单中的哪些节点。

构建 AWS Panorama 应用程序

应用程序在 AWS Panorama 设备上运行，用于对视频流执行计算机视觉任务。您可以通过组合 Python 代码和机器学习模型来构建计算机视觉应用程序，然后通过互联网将它们部署到 AWS Panorama 设备上。应用程序可将视频发送到显示器，也可以使用 AWS SDK 将结果发送到 AWS 服务。

模型分析图像以检测人、车辆和其他对象。根据训练过程中看到的图像，模型会告诉您它认为某样东西是什么，以及它对自己的猜测有多大信心。您可以使用自己的图像数据训练模型，也可以从使用样本数据开始。

应用程序的代码处理来自摄像机视频流的静态图像，将其发送到模型，然后处理结果。一个模型可能会检测到多个对象，并反馈其形状和位置。该代码可以使用这些信息向视频添加文本或图形，或将结果发送到 AWS 服务存储或进一步处理。

要从直播中获取图像、与模型交互并输出视频，应用程序代码使用AWS Panorama 应用程序 SDK。应用程序 SDK 是一个 Python 库，它支持使用 PyTorch MXNet、Apache 和 TensorFlow生成的模型。

主题

计算机视觉模型

计算机视觉模型是一种经过训练的软件程序，用于检测图像中的对象。模型首先通过训练分析这些对象的图像，从而学习识别一组对象。计算机视觉模型将图像作为输入并输出有关其检测到的对象的信息，例如对象的类型及其位置。AWS Panorama 支持使用 PyTorch、Apache MXNet 和 TensorFlow构建的计算机视觉模型。

> **Note**
>
> 有关已使用 AWS Panorama 测试的预构建模型的列表，请参阅模型兼容性。

Sections

在代码中使用模型

模型返回一个或多个结果，其中可能包括检测到的类别的概率、位置信息和其他数据。以下示例演示了如何对视频流中的图像运行推理，并将模型的输出发送到处理函数。

Example application.py – 推理

```
    def process_media(self, stream):
        """Runs inference on a frame of video."""
        image_data = preprocess(stream.image,self.MODEL_DIM)
        logger.debug('Image data: {}'.format(image_data))
        # Run inference
        inference_start = time.time()
        inference_results = self.call({"data":image_data}, self.MODEL_NODE)
         # Log metrics
        inference_time = (time.time() - inference_start) * 1000
        if inference_time > self.inference_time_max:
            self.inference_time_max = inference_time
        self.inference_time_ms += inference_time
        # Process results (classification)
```

```
        self.process_results(inference_results, stream)
```

以下示例演示了一个处理基本分类模型结果的函数。示例模型返回一个概率数组，该数组是结果数组中的第一个也是唯一的值。

Example application.py – 处理结果

```
    def process_results(self, inference_results, stream):
        """Processes output tensors from a computer vision model and annotates a video
frame."""
        if inference_results is None:
            logger.warning("Inference results are None.")
            return
        max_results = 5
        logger.debug('Inference results: {}'.format(inference_results))
        class_tuple = inference_results[0]
        enum_vals = [(i, val) for i, val in enumerate(class_tuple[0])]
        sorted_vals = sorted(enum_vals, key=lambda tup: tup[1])
        top_k = sorted_vals[::-1][:max_results]
        indexes =  [tup[0] for tup in top_k]

        for j in range(max_results):
            label = 'Class [%s], with probability %.3f.'% (self.classes[indexes[j]],
class_tuple[0][indexes[j]])
            stream.add_label(label, 0.1, 0.1 + 0.1*j)
```

应用程序代码会查找概率最高的值，并将其映射到初始化期间加载的资源文件中的标签。

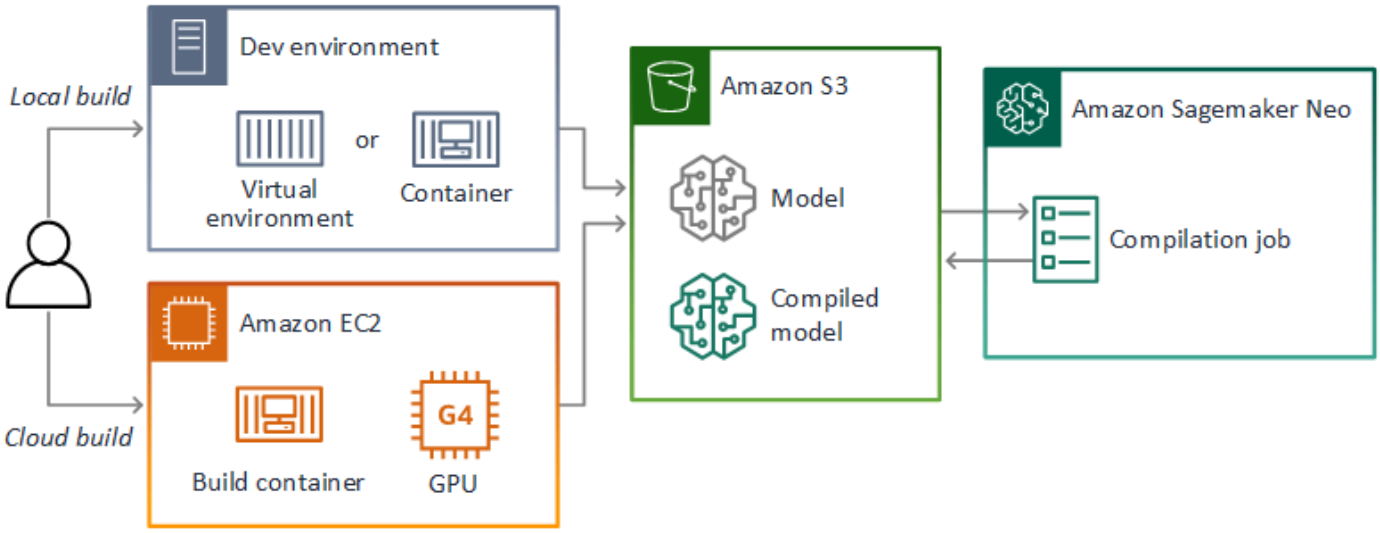
构建自定义模型

您可以在 Apache MXNet 和 AWS Panorama 应用程序 TensorFlow 中使用自己构建的模型。 PyTorch 作为在 SageMaker AI 中构建和训练模型的替代方案，您可以使用经过训练的模型或使用支持的框架构建和训练自己的模型，然后将其导出到本地环境或 Amazon 中 EC2。

> **Note**
>
> 有关 SageMaker AI Neo 支持的框架版本和文件格式的详细信息，请参阅 Amazon A SageMaker I 开发者指南中的支持的框架。

本指南的存储库提供了一个示例应用程序，以 TensorFlow SavedModel格式演示 Keras 模型的此工作流程。它使用 TensorFlow 2，可以在虚拟环境或 Docker 容器中本地运行。示例应用程序还包括用于在 Amazon EC2 实例上构建模型的模板和脚本。

- 自定义模型示例应用程序



AWS Panorama 使用 A SageMaker I Neo 编译模型以在 AWS Panorama 设备上使用。对于每个框架，使用 SageMaker AI Neo 支持的格式，并将模型打包到.tar.gz存档中。

有关更多信息，请参阅 Amazon A SageMaker I 开发者指南中的使用 Neo 编译和部署模型。

打包模型

模型包由描述符、包配置和模型存档组成。与应用程序映像包一样，程序包配置会告知 AWS Panorama 服务模型和描述符在 Amazon S3 中的存储位置。

Example packages/123456789012-SQUEEZENET_PYTORCH-1.0/descriptor.json

```
{
    "mlModelDescriptor": {
        "envelopeVersion": "2021-01-01",
        "framework": "PYTORCH",
        "frameworkVersion": "1.8",
        "precisionMode": "FP16",
        "inputs": [
```

```
            {
                "name": "data",
                "shape": [
                    1,
                    3,
                    224,
                    224
                ]
            }
        ]
    }
}
```

> **Note**
>
> 仅指定框架版本的主要版本和次要版本。有关支持的框架 PyTorch、Apache MXNet TensorFlow 版本和版本列表，请参阅支持的框架。

要导入模型，请使用 AWS Panorama 应用程序 CLI `import-raw-model` 命令。如果对模型或其描述符进行任何更改，则必须重新运行此命令以更新应用程序的资产。有关更多信息，请参阅 更改计算机视觉模型。

有关描述符文件的 JSON 架构，请参阅 assetDescriptor.schema.json。

训练模型

训练模型时，请使用目标环境或与目标环境非常相似的测试环境中的图像。请考虑以下可能影响模型性能的因素：

- 照明 – 拍摄对象反射的光量决定了模型必须分析的细节量。使用光线充足的拍摄对象图像训练的模型在弱光或背光环境中可能无法正常工作。
- 分辨率 – 模型的输入大小通常固定在 224 到 512 像素宽之间的分辨率，呈方形纵横比。在将一帧视频传递给模型之前，可以将其缩小或裁剪以适应所需的尺寸。
- 图像失真 – 相机的焦距和镜头形状会导致图像在远离画面中心的地方出现失真。相机的位置也决定了拍摄对象的哪些特征清晰可见。例如，带有广角镜头的高架相机在拍摄对象位于画面中心时会显示拍摄对象的顶部，而当拍摄对象远离中心时会显示拍摄对象侧面的倾斜视图。

要解决这些问题，可以在将图像发送给模型前对其进行预处理，并在反映现实环境中差异的更广泛的图像上训练模型。如果一个模型需要在照明条件下使用各种相机进行操作，则需要更多数据进行训练。除了收集更多图像之外，您还可以通过创建倾斜或具有不同光照的现有图像的变体来获取更多训练数据。

构建应用程序映像

AWS Panorama Appliance 以从您构建的映像中导出的容器文件系统的形式运行应用程序。您可以在 Dockerfile 中指定应用程序的依赖项和资源，该 Dockerfile 使用 AWS Panorama 应用程序基础映像作为起点。

要构建应用程序映像，您可以使用 Docker 和 AWS Panorama 应用程序 CLI。本指南示例应用程序中的以下示例演示了这些用例。

Example packages/123456789012-SAMPLE_CODE-1.0/Dockerfile

```
FROM public.ecr.aws/panorama/panorama-application
WORKDIR /panorama
COPY . .
RUN pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir -r requirements.txt
```

使用以下 Dockerfile 指令。

- FROM - 加载应用程序基础映像 (public.ecr.aws/panorama/panorama-application)。
- WORKDIR - 在映像上设置工作目录。/panorama 用于应用程序代码和相关文件。此设置仅在构建期间保留，不会影响应用程序运行时的工作目录 (/)。
- COPY - 将文件从本地路径复制到映象中的路径。COPY . .将当前目录（程序包目录）中的文件复制到映象上的工作目录中。例如，应用程序代码是从 packages/123456789012-SAMPLE_CODE-1.0/application.py 复制到 /panorama/application.py 的。
- RUN- 在构建期间对映像运行 shell 命令。通过在命令之间使用 &&，单个 RUN 操作可以按顺序运行多个命令。此示例更新 pip 程序包管理器，然后安装 requirements.txt 中列出的库。

您可以使用在构建时非常有用的其他指令，例如 ADD 和 ARG。向容器添加运行时系统信息的说明（例如 ENV）不适用于 AWS Panorama。AWS Panorama 不会从映像中运行容器。仅使用映像导出文件系统，然后将其传输到设备。

指定依赖项

requirements.txt 是一个 Python 要求文件，用于指定应用程序使用的库。示例应用程序使用 Open CV 和 AWS SDK for Python (Boto3)。

Example packages/123456789012-SAMPLE_CODE-1.0/requirements.txt

```
boto3==1.24.*
opencv-python==4.6.*
```

Dockerfile 中的 `pip install` 命令会将这些库安装到 `/usr/local/lib` 下的 Python `dist-packages` 目录，以便您的应用程序代码可以将其导入。

本地存储

AWS Panorama 保留了用于存储应用程序的 `/opt/aws/panorama/storage` 目录。您的应用程序可以在此路径上创建和修改文件。在存储目录中创建的文件在重新启动后仍会保留。其他临时文件位置会在启动时清除。

构建映像资产

使用 AWS Panorama 应用程序 CLI 为应用程序包构建映像时，CLI 将在程序包目录中运行 `docker build`。这将生成一个包含应用程序代码的应用程序映像。然后，CLI 会创建一个容器，导出其文件系统，对其进行压缩，然后将其存储在 `assets` 文件夹中。

```
$ panorama-cli build-container --container-asset-name code_asset --package-path
 packages/123456789012-SAMPLE_CODE-1.0
docker build -t code_asset packages/123456789012-SAMPLE_CODE-1.0 --pull
docker export --output=code_asset.tar $(docker create code_asset:latest)
gzip -1 code_asset.tar
{
    "name": "code_asset",
    "implementations": [
        {
            "type": "container",
            "assetUri":
 "6f67xmpl32743ed0e60c151a02f2f0da1bf70a4ab9d83fe236fa32a6f9b9f808.tar.gz",
            "descriptorUri":
 "1872xmpl129481ed053c52e66d6af8b030f9eb69b1168a29012f01c7034d7a8f.json"
        }
    ]
}
Container asset for the package has been succesfully built at  /home/
user/aws-panorama-developer-guide/sample-apps/aws-panorama-sample/
assets/6f67xmpl32743ed0e60c151a02f2f0da1bf70a4ab9d83fe236fa32a6f9b9f808.tar.gz
```

输出中的 JSON 数据块是资产定义，CLI 将其添加到程序包配置 (`package.json`) 中，并注册 AWS Panorama 服务。CLI 还会复制描述符文件，该文件指定了应用程序脚本的路径（应用程序的入口点）。

Example [packages/123456789012-SAMPLE_CODE-1.0/descriptor.json]

```
{
    "runtimeDescriptor":
    {
        "envelopeVersion": "2021-01-01",
        "entry":
        {
            "path": "python3",
            "name": "/panorama/application.py"
        }
    }
}
```

在资产文件夹中，描述符和应用程序映像以其 SHA-256 校验码命名。当在 Amazon S3 存储资产时，此名称用作该资产的唯一标识符。

从您的应用程序代码调用 AWS 服务

您可以使用通过应用程序代码 AWS SDK for Python (Boto) 调用 AWS 服务。例如，如果您的模型检测到异常情况，您可以向亚马逊发布指标，使用亚马逊 SNS 发送通知 CloudWatch，将图像保存到 Amazon S3，或者调用 Lambda 函数进行进一步处理。大多数 AWS 服务都有公共 API，您可以将其与 AWS SDK 配合使用。

默认情况下，设备没有访问任何 AWS 服务的权限。要向其授予权限，请为应用程序创建一个角色，并在部署期间将其分配给应用程序实例。

Sections

- 使用 Amazon S3
- 使用 AWS IoT MQTT 主题

使用 Amazon S3

您可以使用 Amazon S3 存储处理结果和其他应用程序数据。

```
import boto3
s3_client=boto3.client("s3")
s3_clients3.upload_file(data_file,
                    s3_bucket_name,
                    os.path.basename(data_file))
```

使用 AWS IoT MQTT 主题

您可以使用适用于 Python 的 SDK (Boto3) 向 AWS IoT中的 MQTT 主题发送消息。在以下示例中，应用程序会发布到以设备事物名称命名的主题中，您可以在AWS IoT 控制台中找到该主题。

```
import boto3
iot_client=boto3.client('iot-data')
topic = "panorama/panorama_my-appliance_Thing_a01e373b"
iot_client.publish(topic=topic, payload="my message")
```

选择一个能显示您选择的设备 ID 或其他标识符的名称。要发布消息，应用程序需要获得调用 `iot:Publish` 的权限。

监控 MQTT 队列

1. 打开AWS IoT 控制台的“测试”页面。
2. 如需订阅主题，输入主题名称。例如，`panorama/panorama_my-appliance_Thing_a01e373b`。
3. 选择订阅主题。

AWS Panorama 应用程序 SDK

AWS Panorama 应用程序 SDK 是一个用于开发 AWS Panorama 应用程序的 Python 库。在您的应用程序代码中，您可以使用 AWS Panorama 应用程序 SDK 加载计算机视觉模型、运行推理并将视频输出到显示器。

> **Note**
>
> 为确保您可以使用 AWS Panorama 应用程序 SDK 的最新功能，请升级设备软件。

有关应用程序 SDK 定义的类及其方法的详细信息，请参阅应用程序 SDK 参考。

Sections

在输出视频中添加文本和方框

使用 AWS Panorama SDK，您可以将视频流输出到显示器。视频可以包含显示模型输出、应用程序当前状态或其他数据的文本和方框。

`video_in` 数组中的每个对象都是来自连接到设备的摄像机视频流的图像。此对象的类型是 `panoramasdk.media`。它具有向图像添加文本和矩形框的方法，然后您可以将其分配给 `video_out` 数组。

在以下示例中，示例应用程序为每个结果添加了一个标签。每个结果都位于相同的左侧位置，但高度不同。

```
        for j in range(max_results):
            label = 'Class [%s], with probability %.3f.'% (self.classes[indexes[j]],
class_tuple[0][indexes[j]])
            stream.add_label(label, 0.1, 0.1 + 0.1*j)
```

要在输出图像中添加方框，请使用 `add_rect`。此方法采用 4 个 0 到 1 之间的值，表示方框左上角和右下角的位置。

```
        w,h,c = stream.image.shape
        stream.add_rect(x1/w, y1/h, x2/w, y2/h)
```

运行多个线程

您可以运行应用程序逻辑处理线程，也可以使用其他线程处理其他后台进程。例如，您可以创建一个为 HTTP 流量提供调试的线程，或者创建一个监控推理结果并将数据发送到 AWS的线程。

要运行多个线程，可以使用 Python 标准库中的线程模块为每个进程创建一个线程。以下示例显示了调试服务器示例应用程序的主循环，该循环创建了一个应用程序对象并使用它来运行三个线程。

Example packages/123456789012-DEBUG_SERVER-1.0/application.py - 主循环

```
def main():
    panorama = panoramasdk.node()
    while True:
        try:
            # Instantiate application
            logger.info('INITIALIZING APPLICATION')
            app = Application(panorama)
            # Create threads for stream processing, debugger, and client
            app.run_thread = threading.Thread(target=app.run_cv)
            app.server_thread = threading.Thread(target=app.run_debugger)
            app.client_thread = threading.Thread(target=app.run_client)
            # Start threads
            logger.info('RUNNING APPLICATION')
            app.run_thread.start()
            logger.info('RUNNING SERVER')
            app.server_thread.start()
            logger.info('RUNNING CLIENT')
            app.client_thread.start()
            # Wait for threads to exit
            app.run_thread.join()
            app.server_thread.join()
            app.client_thread.join()
            logger.info('RESTARTING APPLICATION')
        except:
            logger.exception('Exception during processing loop.')
```

当所有线程都退出时，应用程序会自行重新启动。run_cv 循环处理来自摄像机视频流的图像。如果它收到停止信号，就会关闭调试程序进程，而调试程序进程在 HTTP 服务器上运行，无法自行关闭。每个线程必须处理各自的错误。如果未捕获并记录错误，则线程将以静默方式退出。

Example packages/123456789012-DEBUG_SERVER-1.0/application.py - 处理循环

```
    # Processing loop
    def run_cv(self):
        """Run computer vision workflow in a loop."""
        logger.info("PROCESSING STREAMS")
        while not self.terminate:
            try:
                self.process_streams()
                # turn off debug logging after 15 loops
                if logger.getEffectiveLevel() == logging.DEBUG and self.frame_num ==
15:
                    logger.setLevel(logging.INFO)
            except:
                logger.exception('Exception on processing thread.')
        # Stop signal received
        logger.info("SHUTTING DOWN SERVER")
        self.server.shutdown()
        self.server.server_close()
        logger.info("EXITING RUN THREAD")
```

线程通过应用程序的 `self` 对象进行通信。要重新启动应用程序处理循环，调试程序线程会调用 `stop` 方法。此方法设置了一个 `terminate` 属性，向其他线程发出关闭信号。

Example packages/123456789012-DEBUG_SERVER-1.0/application.py - 停止方法

```
    # Interrupt processing loop
    def stop(self):
        """Signal application to stop processing."""
        logger.info("STOPPING APPLICATION")
        # Signal processes to stop
        self.terminate = True
    # HTTP debug server
    def run_debugger(self):
        """Process debug commands from local network."""
        class ServerHandler(SimpleHTTPRequestHandler):
            # Store reference to application
            application = self
            # Get status
            def do_GET(self):
                """Process GET requests."""
                logger.info('Get request to {}'.format(self.path))
                if self.path == "/status":
```

```
                self.send_200('OK')
            else:
                self.send_error(400)
        # Restart application
        def do_POST(self):
            """Process POST requests."""
            logger.info('Post request to {}'.format(self.path))
            if self.path == '/restart':
                self.send_200('OK')
                ServerHandler.application.stop()
            else:
                self.send_error(400)
```

提供入站流量

您可以在运行应用程序代码的同时运行 HTTP 服务器，从而在本地监控或调试应用程序。要为外部流量提供服务，您可以将 AWS Panorama 设备上的端口映射到应用程序容器上的端口。

> **Important**
>
> 默认情况下，AWS Panorama 设备不接受任何端口的传入流量。打开设备上的端口存在隐含的安全风险。使用此功能时，必须采取其他步骤来保护设备免受外部流量的影响，并确保授权客户端与设备之间的通信安全。
> 本指南中包含的示例代码仅用于演示目的，不实现身份验证、授权或加密。

您可以在设备上打开 8000–9000 范围内的端口。这些端口在打开时可接收来自任何可路由客户端的流量。部署应用程序时，您需要指定要打开的端口，并将设备上的端口映射到应用程序容器上的端口。设备软件将流量转发到容器，并将响应发送回请求者。在指定的设备端口上接收请求，在随机的临时端口上发出响应。

配置入站端口

您可以在应用程序配置中的三个位置指定端口映射。代码包的 `package.json`，您可以指定代码节点在 `network` 块中侦听的端口。下面的示例声明节点通过 80 端口侦听。

Example packages/123456789012-DEBUG_SERVER-1.0/package.json

```
            "outputs": [
                {
                    "description": "Video stream output",
                    "name": "video_out",
                    "type": "media"
                }
            ],
            "network": {
                "inboundPorts": [
                    {
                        "port": 80,
                        "description": "http"
                    }
                ]
            }
```

在应用程序清单中，您声明了一条路由规则，该规则将设备上的端口映射到应用程序代码容器上的端口。以下示例添加了一条规则，将设备上的端口 8080 映射到 code_node 容器上的端口 80。

Example graphs/my-app/graph.json

```
            {
                "producer": "model_input_width",
                "consumer": "code_node.model_input_width"
            },
            {
                "producer": "model_input_order",
                "consumer": "code_node.model_input_order"
            }
        ],
        "networkRoutingRules": [
            {
                "node": "code_node",
                "containerPort": 80,
                "hostPort": 8080,
                "decorator": {
                    "title": "Listener port 8080",
                    "description": "Container monitoring and debug."
                }
            }
        ]
```

部署应用程序时，您可以在 AWS Panorama 控制台中指定相同的规则，或者使用传递给 CreateApplicationInstanceAPI 的替代文档。您必须在部署时提供此配置，以确认要在设备上打开端口。

Example graphs/my-app/override.json

```
            {
                "replace": "camera_node",
                "with": [
                    {
                        "name": "exterior-north"
                    }
                ]
            }
        ],
        "networkRoutingRules":[
```

```
            {
                "node": "code_node",
                "containerPort": 80,
                "hostPort": 8080
            }
        ],
        "envelopeVersion": "2021-01-01"
    }
}
```

如果应用程序清单中指定的设备端口正由其他应用程序使用，则可以使用覆盖文档选择其他端口。

提供流量

在容器上打开端口后，可以打开套接字或运行服务器来处理传入请求。debug-server 示例演示了与计算机视觉应用程序代码一起运行的 HTTP 服务器的基本实现。

> **Important**
>
> 该示例实现对于生产用途来说并不安全。为避免设备易遭受攻击，您必须在代码和网络配置中实施适当的安全控制。

Example packages/123456789012-DEBUG_SERVER-1.0/application.py – HTTP 服务器

```
    # HTTP debug server
    def run_debugger(self):
        """Process debug commands from local network."""
        class ServerHandler(SimpleHTTPRequestHandler):
            # Store reference to application
            application = self
            # Get status
            def do_GET(self):
                """Process GET requests."""
                logger.info('Get request to {}'.format(self.path))
                if self.path == '/status':
                    self.send_200('OK')
                else:
                    self.send_error(400)
            # Restart application
            def do_POST(self):
                """Process POST requests."""
```

```
                logger.info('Post request to {}'.format(self.path))
                if self.path == '/restart':
                    self.send_200('OK')
                    ServerHandler.application.stop()
                else:
                    self.send_error(400)
            # Send response
            def send_200(self, msg):
                """Send 200 (success) response with message."""
                self.send_response(200)
                self.send_header('Content-Type', 'text/plain')
                self.end_headers()
                self.wfile.write(msg.encode('utf-8'))
        try:
            # Run HTTP server
            self.server = HTTPServer(("", self.CONTAINER_PORT), ServerHandler)
            self.server.serve_forever(1)
            # Server shut down by run_cv loop
            logger.info("EXITING SERVER THREAD")
        except:
            logger.exception('Exception on server thread.')
```

服务器接受 `/status` 路径处的 GET 请求，以检索有关应用程序的某些信息。它还接受对 `/restart` 的 POST 请求以重新启动应用程序。

为了演示此功能，示例应用程序在单独的线程上运行 HTTP 客户端。客户端在启动后不久通过本地网络调用 `/status` 路径，并在几分钟后重新启动应用程序。

Example [packages/123456789012-DEBUG_SERVER-1.0/application.py](#) – HTTP 客户端

```
    # HTTP test client
    def run_client(self):
        """Send HTTP requests to device port to demnostrate debug server functions."""
        def client_get():
            """Get container status"""
            r = requests.get('http://{}:{}/status'.format(self.device_ip,
self.DEVICE_PORT))
            logger.info('Response: {}'.format(r.text))
            return
        def client_post():
            """Restart application"""
            r = requests.post('http://{}:{}/restart'.format(self.device_ip,
self.DEVICE_PORT))
```

```
            logger.info('Response: {}'.format(r.text))
            return
        # Call debug server
        while not self.terminate:
            try:
                time.sleep(30)
                client_get()
                time.sleep(300)
                client_post()
            except:
                logger.exception('Exception on client thread.')
        # stop signal received
        logger.info("EXITING CLIENT THREAD")
```

主循环管理线程，并在线程退出时重新启动应用程序。

Example packages/123456789012-DEBUG_SERVER-1.0/application.py - 主循环

```
def main():
    panorama = panoramasdk.node()
    while True:
        try:
            # Instantiate application
            logger.info('INITIALIZING APPLICATION')
            app = Application(panorama)
            # Create threads for stream processing, debugger, and client
            app.run_thread = threading.Thread(target=app.run_cv)
            app.server_thread = threading.Thread(target=app.run_debugger)
            app.client_thread = threading.Thread(target=app.run_client)
            # Start threads
            logger.info('RUNNING APPLICATION')
            app.run_thread.start()
            logger.info('RUNNING SERVER')
            app.server_thread.start()
            logger.info('RUNNING CLIENT')
            app.client_thread.start()
            # Wait for threads to exit
            app.run_thread.join()
            app.server_thread.join()
            app.client_thread.join()
            logger.info('RESTARTING APPLICATION')
        except:
            logger.exception('Exception during processing loop.')
```

要部署示例应用程序，请参阅本指南 GitHub 存储库中的说明。

使用 GPU

您可以访问 AWS Panorama Appliance 上的图形处理器 (GPU)，以使用 GPU 加速库，或在应用程序代码中运行机器学习模型。要启用 GPU 访问权限，请在构建应用程序代码容器后将 GPU 访问权限作为一项需求添加至程序包配置中。

> **Important**
>
> 如果您启用 GPU 访问权限，则无法在设备上的任何应用程序中运行模型节点。出于安全考虑，当设备运行使用 SageMaker AI Neo 编译的模型时，GPU 访问会受到限制。有了 GPU 访问权限，您必须在应用程序代码节点中运行模型，且设备上的所有应用程序都共享 GPU 的访问权限。

要为您的应用程序启用 GPU 访问权限，请在使用 AWS Panorama 应用程序 CLI 构建程序包后更新程序包配置。以下示例显示了向应用程序代码节点添加 GPU 访问权限的 `requirements` 数据块。

Example含要求数据块的 package.json

```
{
    "nodePackage": {
        "envelopeVersion": "2021-01-01",
        "name": "SAMPLE_CODE",
        "version": "1.0",
        "description": "Computer vision application code.",
        "assets": [
            {
                "name": "code_asset",
                "implementations": [
                    {
                        "type": "container",
                        "assetUri":
 "eba3xmpl71aa387e8f89be9a8c396416cdb80a717bb32103c957a8bf41440b12.tar.gz",
                        "descriptorUri":
 "4abdxmpl5a6f047d2b3047adde44704759d13f0126c00ed9b4309726f6bb43400ba9.json",
                        "requirements": [
                            {
                                "type": "hardware_access",
                                "inferenceAccelerators": [
                                    {
                                        "deviceType": "nvhost_gpu",
```

```
                                    "sharedResourcePolicy": {
                                        "policy" : "allow_all"
                                    }
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ],
    "interfaces": [
    ...
```

在开发工作流程的构建和打包步骤之间更新程序包配置。

部署有 GPU 访问权限的应用程序

1. 若要生成应用程序容器，请使用 `build-container` 命令。

```
$ panorama-cli build-container --container-asset-name code_asset --package-path
 packages/123456789012-SAMPLE_CODE-1.0
```

2. 将该 `requirements` 数据块添加到程序包配置中。
3. 要上传容器资产和程序包配置，请使用 `package-application` 命令。

```
$ panorama-cli package-application
```

4. 部署 应用程序。

有关使用 GPU 访问权限的示例应用程序，请访问aws-panorama-samples GitHub 存储库。

在 Windows 中设置开发环境

要构建 AWS Panorama 应用程序，您需要使用 Docker、命令行工具和 Python。在 Windows 中，您可以使用与 Windows Subsystem for Linux 和 Ubuntu 兼容的 Docker Desktop 来设置开发环境。本教程将指导您了解使用 AWS Panorama 工具和示例应用程序测试过的开发环境的设置过程。

Sections

先决条件

要学习本教程，您需要一个与 Windows Subsystem for Linux 2 (WSL 2) 的 Windows 版本。

- Windows 10 版本 1903 及更高版本（Build 18362 及更高版本）或 Windows 11
- Windows 功能
  - Windows Subsystem for Linux
  - Hyper-V
  - 虚拟机平台

本教程使用以下软件版本开发。

- Ubuntu 20.04
- Python 3.8.5
- Docker 20.10.8

安装 WSL 2 与 Ubuntu

如果你有 Windows 10 版本 2004 及更高版本（Build 19041 及更高版本），则可以使用以下命令安装 WSL 2 和 Ubuntu 20.04。 PowerShell

```
> wsl --install -d Ubuntu-20.04
```

对于较旧的 Windows 版本，请按照 WSL 2 文档中的说明进行操作：旧版本手动安装步骤

安装 Docker

要安装 Docker Desktop，请从 hub.docker.com 下载安装程序包并运行。如遇到问题，请按照 Docker 网站上的说明进行操作：Docker Desktop WSL 2 后端。

运行 Docker Desktop 并按照首次运行的教程构建示例容器。

> **Note**
>
> Docker Desktop 仅在默认发行版中启用 Docker。如果您在运行本教程之前安装了其他 Linux 发行版本，请在新安装的 Ubuntu 发行版中，在资源、WSL 集成下的 Docker Desktop 设置菜单中启用 Docker。

配置 Ubuntu

您现在可以在 Ubuntu 虚拟机中运行 Docker 命令。要打开命令行终端，请从开始菜单运行发行版本。首次运行时，需要设置用户名和密码，用于运行管理员命令。

要完成开发环境的配置，请更新虚拟机的软件并安装工具。

配置虚拟机

1. 更新 Ubuntu 自带的软件。

   ```
   $ sudo apt update && sudo apt upgrade -y && sudo apt autoremove
   ```

2. 使用 apt 安装开发工具。

   ```
   $ sudo apt install unzip python3-pip
   ```

3. 使用 pip 安装 Python 库。

   ```
   $ pip3 install awscli panoramacli
   ```

4. 打开一个新终端，然后运行 `aws configure` 以配置 AWS CLI。

```
$ aws configure
```

如果没有访问密钥，您可以在 IAM 控制台中生成。

最后，下载并导入示例应用程序。

获取示例应用程序

1. 下载并解压示例应用程序。

```
$ wget https://github.com/awsdocs/aws-panorama-developer-guide/releases/download/
v1.0-ga/aws-panorama-sample.zip
$ unzip aws-panorama-sample.zip
$ cd aws-panorama-sample
```

2. 运行随附的脚本以测试编译、构建应用程序容器并将程序包上传到 AWS Panorama。

```
aws-panorama-sample$ ./0-test-compile.sh
aws-panorama-sample$ ./1-create-role.sh
aws-panorama-sample$ ./2-import-app.sh
aws-panorama-sample$ ./3-build-container.sh
aws-panorama-sample$ ./4-package-app.sh
```

AWS Panorama 应用程序 CLI 会上传程序包并将其注册到 AWS Panorama 服务。现在，您可以使用 AWS Panorama 控制台部署示例应用程序。

后续步骤

要浏览和编辑项目文件，您可以使用文件资源管理器或支持 WSL 的集成式开发环境 (IDE)。

要访问虚拟机的文件系统，请打开文件资源管理器并在导航栏中输入 `\\wsl$`。此目录包含链接至虚拟机的文件系统 (`Ubuntu-20.04`) 和 Docker 数据的文件系统的链接。在 `Ubuntu-20.04` 下，您的用户目录位于 `home\`*`username`*。

> **Note**
>
> 要从 Ubuntu 中访问 Windows 安装中的文件，请导航到 `/mnt/c` 目录。例如，您可以通过运行 `ls /mnt/c/Users/`*`windows-username`*`/Downloads` 来列出下载目录中的文件。

您可以使用 Visual Studio Code 在开发环境中编辑应用程序代码，并通过集成终端运行命令。要安装 Visual Studio Code，请访问 code.visualstudio.com。完成安装后，添加远程 WSL 扩展。

您一直在使用标准 Ubuntu 终端运行命令，而 Windows 终端是其替代方案。它支持多个选项卡 PowerShell，可以在你安装的任何其他类型的 Linux 上运行、命令提示符和终端。它支持使用和进行复制Ctrl+C和粘贴Ctrl+V URLs、可点击以及其他有用的改进。要安装 Windows 终端，请访问 microsoft.com。

AWS Panorama API

您可以使用 AWS Panorama 服务的公共 API 来自动执行设备和应用程序管理工作流程。使用 AWS Command Line Interface 或 AWS SDK，您可以开发用于管理资源和部署的脚本或应用程序。本指南的 GitHub 存储库包含脚本，您可以将其用作自己代码的起点。

- aws-panorama-developer-guide/util-script

Sections

- 自动执行设备注册
- 使用 AWS Panorama API 管理设备
- 自动化应用程序部署
- 使用 AWS Panorama API 管理应用程序
- 使用 VPC 端点

自动执行设备注册

要配置设备，请使用 ProvisionDeviceAPI。响应包括一个 ZIP 文件，其中包含设备的配置和临时凭证。对文件进行解码，并将其保存在前缀为 `certificates-omni_` 的存档中。

Example provision-device.sh

```
if [[ $# -eq 1 ]] ; then
    DEVICE_NAME=$1
else
    echo "Usage: ./provision-device.sh <device-name>"
    exit 1
fi
CERTIFICATE_BUNDLE=certificates-omni_${DEVICE_NAME}.zip
aws panorama provision-device --name ${DEVICE_NAME} --output text --query Certificates
 | base64 --decode > ${CERTIFICATE_BUNDLE}
echo "Created certificate bundle ${CERTIFICATE_BUNDLE}"
```

配置存档中的凭证将在 5 分钟后过期。使用随附的 USB 驱动器将存档传输到您的设备。

要注册摄像头，请使用 CreateNodeFromTemplateJobAPI。此 API 采用摄像头用户名、密码和 URL 的模板参数映射。可以使用 Bash 字符串操作将此映射格式化为 JSON 文档。

Example register-camera.sh

```
if [[ $# -eq 3 ]] ; then
    NAME=$1
    USERNAME=$2
    URL=$3
else
    echo "Usage: ./register-camera.sh <stream-name> <username> <rtsp-url>"
    exit 1
fi
echo "Enter camera stream password: "
read PASSWORD
TEMPLATE='{"Username":"MY_USERNAME","Password":"MY_PASSWORD","StreamUrl": "MY_URL"}'
TEMPLATE=${TEMPLATE/MY_USERNAME/$USERNAME}
TEMPLATE=${TEMPLATE/MY_PASSWORD/$PASSWORD}
TEMPLATE=${TEMPLATE/MY_URL/$URL}
echo ${TEMPLATE}
```

```
JOB_ID=$(aws panorama create-node-from-template-job --template-type RTSP_CAMERA_STREAM
 --output-package-name ${NAME} --output-package-version "1.0" --node-name ${NAME} --
template-parameters "${TEMPLATE}" --output text)
```

或者，您可以从文件加载 JSON 配置。

```
--template-parameters file://camera-template.json
```

使用 AWS Panorama API 管理设备

您可以使用 AWS Panorama API 自动执行设备管理任务。

查看设备

要获取带有设备的设备列表 IDs，请使用 ListDevicesAPI。

```
$ aws panorama list-devices
    "Devices": [
        {
            "DeviceId": "device-4tafxmplhtmzabv5lsacba4ere",
            "Name": "my-appliance",
            "CreatedTime": 1652409973.613,
            "ProvisioningStatus": "SUCCEEDED",
            "LastUpdatedTime": 1652410973.052,
            "LeaseExpirationTime": 1652842940.0
        }
    ]
}
```

要获取有关设备的更多详细信息，请使用 DescribeDeviceAPI。

```
$ aws panorama describe-device --device-id device-4tafxmplhtmzabv5lsacba4ere
{
    "DeviceId": "device-4tafxmplhtmzabv5lsacba4ere",
    "Name": "my-appliance",
    "Arn": "arn:aws:panorama:us-west-2:123456789012:device/
device-4tafxmplhtmzabv5lsacba4ere",
    "Type": "PANORAMA_APPLIANCE",
    "DeviceConnectionStatus": "ONLINE",
    "CreatedTime": 1648232043.421,
    "ProvisioningStatus": "SUCCEEDED",
    "LatestSoftware": "4.3.55",
    "CurrentSoftware": "4.3.45",
    "SerialNumber": "GFXMPL0013023708",
    "Tags": {},
    "CurrentNetworkingStatus": {
        "Ethernet0Status": {
            "IpAddress": "192.168.0.1/24",
            "ConnectionStatus": "CONNECTED",
            "HwAddress": "8C:XM:PL:60:C5:88"
```

```
        },
        "Ethernet1Status": {
            "IpAddress": "--",
            "ConnectionStatus": "NOT_CONNECTED",
            "HwAddress": "8C:XM:PL:60:C5:89"
        }
    },
    "LeaseExpirationTime": 1652746098.0
}
```

升级设备软件

如果 `LatestSoftware` 版本比 `CurrentSoftware` 版本新，您可以升级设备。使用 CreateJobForDevicesAPI 创建 over-the-air (OTA) 更新任务。

```
$ aws panorama create-job-for-devices --device-ids device-4tafxmplhtmzabv5lsacba4ere \
  --device-job-config '{"OTAJobConfig": {"ImageVersion": "4.3.55"}}' --job-type OTA
{
    "Jobs": [
        {
            "JobId": "device-4tafxmplhtmzabv5lsacba4ere-0",
            "DeviceId": "device-4tafxmplhtmzabv5lsacba4ere"
        }
    ]
}
```

在脚本中，可以使用 Bash 字符串操作填充作业配置文件中的映像版本字段。

Example check-updates.sh

```
apply_update() {
    DEVICE_ID=$1
    NEW_VERSION=$2
    CONFIG='{"OTAJobConfig": {"ImageVersion": "NEW_VERSION"}}'
    CONFIG=${CONFIG/NEW_VERSION/$NEW_VERSION}
    aws panorama create-job-for-devices --device-ids ${DEVICE_ID} --device-job-config
 "${CONFIG}" --job-type OTA
}
```

设备将下载指定的软件版本并自行更新。使用 DescribeDeviceJobAPI 查看更新进度。

```
$ aws panorama describe-device-job --job-id device-4tafxmplhtmzabv5lsacba4ere-0
```

```
{
    "JobId": "device-4tafxmplhtmzabv5lsacba4ere-0",
    "DeviceId": "device-4tafxmplhtmzabv5lsacba4ere",
    "DeviceArn": "arn:aws:panorama:us-west-2:559823168634:device/
device-4tafxmplhtmzabv5lsacba4ere",
    "DeviceName": "my-appliance",
    "DeviceType": "PANORAMA_APPLIANCE",
    "ImageVersion": "4.3.55",
    "Status": "REBOOTING",
    "CreatedTime": 1652410232.465
}
```

要获取所有正在运行的作业的列表，请使用ListDevicesJobs。

```
$ aws panorama list-devices-jobs
{
    "DeviceJobs": [
        {
            "DeviceName": "my-appliance",
            "DeviceId": "device-4tafxmplhtmzabv5lsacba4ere",
            "JobId": "device-4tafxmplhtmzabv5lsacba4ere-0",
            "CreatedTime": 1652410232.465
        }
    ]
}
```

有关检查和应用更新的示例脚本，请参阅本指南 GitHub 存储库中的 check-updates.sh。

重新启动设备

要重启设备，请使用 CreateJobForDevicesAPI。

```
$ aws panorama create-job-for-devices --device-ids device-4tafxmplhtmzabv5lsacba4ere --
job-type REBOOT
{
    "Jobs": [
        {
            "JobId": "device-4tafxmplhtmzabv5lsacba4ere-0",
            "DeviceId": "device-4tafxmplhtmzabv5lsacba4ere"
        }
    ]
}
```

在脚本中，您可以获取设备列表，并选择一个设备以交互方式重新启动。

Example reboot-device.sh – 用法

```
$ ./reboot-device.sh
Getting devices...
0: device-53amxmplyn3gmj72epzanacniy     my-se70-1
1: device-6talxmpl5mmik6qh5moba6jium     my-manh-24
Choose a device
1
Reboot device device-6talxmpl5mmik6qh5moba6jium? (y/n)y
{
    "Jobs": [
        {
            "DeviceId": "device-6talxmpl5mmik6qh5moba6jium",
            "JobId": "device-6talxmpl5mmik6qh5moba6jium-8"
        }
    ]
}
```

自动化应用程序部署

要部署应用程序，您可以同时使用 AWS Panorama 应用程序 CLI 和 AWS Command Line Interface。构建应用程序容器后，将其和其他资产上传到 Amazon S3 接入点。然后，您可以使用 CreateApplicationInstanceAPI 部署应用程序。

有关使用所示脚本的更多上下文和说明，请按照示例应用程序自述文件中的说明进行操作。

Sections

构建容器

若要生成应用程序容器，请使用 `build-container` 命令。此命令会构建一个 Docker 容器，并将其作为压缩文件系统保存在 `assets` 文件夹中。

Example 3-build-container.sh

```
CODE_PACKAGE=SAMPLE_CODE
ACCOUNT_ID=$(aws sts get-caller-identity --output text --query 'Account')
panorama-cli build-container --container-asset-name code_asset --package-path packages/
${ACCOUNT_ID}-${CODE_PACKAGE}-1.0
```

还可以使用命令行完成来填充路径参数，方法是键入部分路径，然后按 TAB。

```
$ panorama-cli build-container --package-path packages/TAB
```

上传容器并注册节点

要上传应用程序，请使用 `package-application` 命令。此命令会将资产从 `assets` 文件夹上传到 AWS Panorama 管理的 Amazon S3 接入点。

Example 4-package-app.sh

```
panorama-cli package-application
```

AWS Panorama 应用程序 CLI 会上传每个软件包中软件包配置 (`package.json`) 引用的容器和描述符资产，并将软件包注册为 AWS Panorama 中的节点。然后，在应用程序清单 (`graph.json`) 中引用这些节点来部署应用程序。

部署应用程序

要部署应用程序，您可以使用 CreateApplicationInstanceAPI。除其他外，此操作采用以下参数。

- `ManifestPayload` – 定义应用程序节点、包、边缘和参数的应用程序清单(`graph.json`)。
- `ManifestOverridesPayload` – 第二个清单，覆盖第一个清单中的参数。应用程序清单可被视为应用程序源中的静态资源，其中覆盖清单提供了用于自定义部署的部署时设置。
- `DefaultRuntimeContextDevice` – 目标设备。
- `RuntimeRoleArn` – 应用程序用于访问 AWS 服务和资源的 IAM 角色的 ARN。
- `ApplicationInstanceIdToReplace` – 要从设备中移除的现有应用程序实例的 ID。

清单和覆盖负载是 JSON 文档，必须作为嵌套在另一个文档中的字符串值提供。为此，脚本以字符串形式从文件中加载清单，并使用jq 工具构造嵌套文档。

Example 5-deploy.sh – 撰写清单

```
GRAPH_PATH="graphs/my-app/graph.json"
OVERRIDE_PATH="graphs/my-app/override.json"
# application manifest
GRAPH=$(cat ${GRAPH_PATH} | tr -d '\n' | tr -d '[:blank:]')
MANIFEST="$(jq --arg value "${GRAPH}" '.PayloadData="\($value)"' <<< {})"
# manifest override
OVERRIDE=$(cat ${OVERRIDE_PATH} | tr -d '\n' | tr -d '[:blank:]')
MANIFEST_OVERRIDE="$(jq --arg value "${OVERRIDE}" '.PayloadData="\($value)"' <<< {})"
```

部署脚本使用 ListDevicesAPI 获取当前区域中已注册设备的列表，并将用户的选择保存到本地文件中以供后续部署。

Example 5-deploy.sh – 查找设备

```
    echo "Getting devices..."
    DEVICES=$(aws panorama list-devices)
```

```
    DEVICE_NAMES=($((echo ${DEVICES} | jq -r '.Devices |=sort_by(.LastUpdatedTime) |
 [.Devices[].Name] | @sh') | tr -d \'\"))
    DEVICE_IDS=($((echo ${DEVICES} | jq -r '.Devices |=sort_by(.LastUpdatedTime) |
 [.Devices[].DeviceId] | @sh') | tr -d \'\"))
    for (( c=0; c<${#DEVICE_NAMES[@]}; c++ ))
    do
        echo "${c}: ${DEVICE_IDS[${c}]}     ${DEVICE_NAMES[${c}]}"
    done
    echo "Choose a device"
    read D_INDEX
    echo "Deploying to device ${DEVICE_IDS[${D_INDEX}]}"
    echo -n ${DEVICE_IDS[${D_INDEX}]} > device-id.txt
    DEVICE_ID=$(cat device-id.txt)
```

应用程序角色由另一个脚本 (1-create-role.sh) 创建。部署脚本从中获取此角色的 ARN。 AWS CloudFormation如果应用程序已部署到设备上，则脚本会从本地文件中获取该应用程序实例的 ID。

Example 5-deploy.sh – 角色 ARN 和替换参数

```
# application role
STACK_NAME=panorama-${NAME}
ROLE_ARN=$(aws cloudformation describe-stacks --stack-name panorama-${PWD##*/} --query
 'Stacks[0].Outputs[?OutputKey==`roleArn`].OutputValue' --output text)
ROLE_ARG="--runtime-role-arn=${ROLE_ARN}"

# existing application instance id
if [ -f "application-id.txt" ]; then
    EXISTING_APPLICATION=$(cat application-id.txt)
    REPLACE_ARG="--application-instance-id-to-replace=${EXISTING_APPLICATION}"
    echo "Replacing application instance ${EXISTING_APPLICATION}"
fi
```

最后，该脚本会将所有部分组合在一起，以创建应用程序实例并将应用程序部署到设备上。该服务以实例 ID 作为响应，脚本会存储该实例 ID 以供日后使用。

Example 5-deploy.sh – 部署应用程序

```
APPLICATION_ID=$(aws panorama create-application-instance ${REPLACE_ARG} --manifest-
payload="${MANIFEST}" --default-runtime-context-device=${DEVICE_ID} --name=${NAME}
 --description="command-line deploy" --tags client=sample --manifest-overrides-
payload="${MANIFEST_OVERRIDE}" ${ROLE_ARG} --output text)
echo "New application instance ${APPLICATION_ID}"
```

```
echo -n $APPLICATION_ID > application-id.txt
```

监控部署

要监控部署，请使用 ListApplicationInstancesAPI。监控脚本从应用程序目录中的文件获取设备 ID 和应用程序实例 ID，并使用它们构造 CLI 命令。然后它会循环调用。

Example 6-monitor-deployment.sh

```
APPLICATION_ID=$(cat application-id.txt)
DEVICE_ID=$(cat device-id.txt)
QUERY="ApplicationInstances[?ApplicationInstanceId==\`APPLICATION_ID\`]"
QUERY=${QUERY/APPLICATION_ID/$APPLICATION_ID}
MONITOR_CMD="aws panorama list-application-instances --device-id ${DEVICE_ID} --query
 ${QUERY}"
MONITOR_CMD=${MONITOR_CMD/QUERY/$QUERY}
while true; do
    $MONITOR_CMD
    sleep 60
done
```

部署完成后，您可以通过调用 Amazon 日志 API 来查看 CloudWatch 日志。查看日志脚本使用日 CloudWatch 志 `GetLogEvents` API。

Example view-logs.sh

```
GROUP="/aws/panorama/devices/MY_DEVICE_ID/applications/MY_APPLICATION_ID"
GROUP=${GROUP/MY_DEVICE_ID/$DEVICE_ID}
GROUP=${GROUP/MY_APPLICATION_ID/$APPLICATION_ID}
echo "Getting logs for group ${GROUP}."
#set -x
while true
do
    LOGS=$(aws logs get-log-events --log-group-name ${GROUP} --log-stream-name
 code_node --limit 150)
    readarray -t ENTRIES < <(echo $LOGS | jq -c '.events[].message')
    for ENTRY in "${ENTRIES[@]}"; do
        echo "$ENTRY" | tr -d \"
    done
    sleep 20
done
```

使用 AWS Panorama API 管理应用程序

您可以使用 AWS Panorama API 监控和管理应用程序。

查看应用程序

要获取设备上运行的应用程序列表，请使用 ListApplicationInstancesAPI。

```
$ aws panorama list-application-instances
    "ApplicationInstances": [
        {
            "Name": "aws-panorama-sample",
            "ApplicationInstanceId": "applicationInstance-ddaxxmpl2z7bg74ywutd7byxuq",
            "DefaultRuntimeContextDevice": "device-4tafxmplhtmzabv5lsacba4ere",
            "DefaultRuntimeContextDeviceName": "my-appliance",
            "Description": "command-line deploy",
            "Status": "DEPLOYMENT_SUCCEEDED",
            "HealthStatus": "RUNNING",
            "StatusDescription": "Application deployed successfully.",
            "CreatedTime": 1661902051.925,
            "Arn": "arn:aws:panorama:us-east-2:123456789012:applicationInstance/
applicationInstance-ddaxxmpl2z7bg74ywutd7byxuq",
            "Tags": {
                "client": "sample"
            }
        },
    ]
}
```

要获取有关应用程序实例节点的更多详细信息，请使用 ListApplicationInstanceNodeInstancesAPI。

```
$ aws panorama list-application-instance-node-instances --application-instance-id
 applicationInstance-ddaxxmpl2z7bg74ywutd7byxuq
{
    "NodeInstances": [
        {
            "NodeInstanceId": "code_node",
            "NodeId": "SAMPLE_CODE-1.0-fd3dxmpl-interface",
            "PackageName": "SAMPLE_CODE",
            "PackageVersion": "1.0",
            "PackagePatchVersion":
 "fd3dxmpl2bdfa41e6fe1be290a79dd2c29cf014eadf7416d861ce7715ad5e8a8",
            "NodeName": "interface",
```

```
            "CurrentStatus": "RUNNING"
        },
        {
            "NodeInstanceId": "camera_node_override",
            "NodeId": "warehouse-floor-1.0-9eabxmpl-warehouse-floor",
            "PackageName": "warehouse-floor",
            "PackageVersion": "1.0",
            "PackagePatchVersion":
 "9eabxmple89f0f8b2f2852cca2a6e7971aa38f1629a210d069045e83697e42a7",
            "NodeName": "warehouse-floor",
            "CurrentStatus": "RUNNING"
        },
        {
            "NodeInstanceId": "output_node",
            "NodeId": "hdmi_data_sink-1.0-9c23xmpl-hdmi0",
            "PackageName": "hdmi_data_sink",
            "PackageVersion": "1.0",
            "PackagePatchVersion":
 "9c23xmplc4c98b92baea4af676c8b16063d17945a3f6bd8f83f4ff5aa0d0b394",
            "NodeName": "hdmi0",
            "CurrentStatus": "RUNNING"
        },
        {
            "NodeInstanceId": "model_node",
            "NodeId": "SQUEEZENET_PYTORCH-1.0-5d3cabda-interface",
            "PackageName": "SQUEEZENET_PYTORCH",
            "PackageVersion": "1.0",
            "PackagePatchVersion":
 "5d3cxmplb7113faa1d130f97f619655d8ca12787c751851a0e155e50eb5e3e96",
            "NodeName": "interface",
            "CurrentStatus": "RUNNING"
        }
    ]
}
```

管理摄像头流式传输

您可以使用 SignalApplicationInstanceNodeInstancesAPI 暂停和恢复摄像机直播节点。

```
$ aws panorama signal-application-instance-node-instances --application-instance-id
 applicationInstance-ddaxxmpl2z7bg74ywutd7byxuq \
       --node-signals '[{"NodeInstanceId": "camera_node_override", "Signal":
 "PAUSE"}]'
```

```
{
    "ApplicationInstanceId": "applicationInstance-ddaxxmpl2z7bg74ywutd7byxuq"
}
```

在脚本中，您可以获取节点列表，并选择一个节点以交互方式暂停或恢复。

Example pause-camera.sh – 使用

```
my-app$ ./pause-camera.sh

Getting nodes...
0: SAMPLE_CODE              RUNNING
1: warehouse-floor          RUNNING
2: hdmi_data_sink           RUNNING
3: entrance-north           PAUSED
4: SQUEEZENET_PYTORCH       RUNNING
Choose a node
1
Signalling node warehouse-floor
+ aws panorama signal-application-instance-node-instances --application-instance-id
 applicationInstance-r3a7xmplcbmpjqeds7vj4b6pjy --node-signals '[{"NodeInstanceId":
 "warehouse-floor", "Signal": "PAUSE"}]'
{
    "ApplicationInstanceId": "applicationInstance-r3a7xmplcbmpjqeds7vj4b6pjy"
}
```

通过暂停和恢复摄像头节点，您可以循环处理比同时处理数量更多的摄像头流式传输。为此，请将多个摄像头流式传输映射到覆盖清单中的同一输入节点。

在以下示例中，覆盖清单将两个摄像头流式传输 warehouse-floor 和 entrance-north 映射到同一输入节点 (camera_node)。当应用程序启动并且 entrance-north 节点等待信号开启时，warehouse-floor 流式传输处于活动状态。

Example override-multicam.json

```
    "nodeGraphOverrides": {
        "nodes": [
            {
                "name": "warehouse-floor",
                "interface": "123456789012::warehouse-floor.warehouse-floor",
                "launch": "onAppStart"
            },
```

```
            {
                "name": "entrance-north",
                "interface": "123456789012::entrance-north.entrance-north",
                "launch": "onSignal"
            },
        ...
        "packages": [
            {
                "name": "123456789012::warehouse-floor",
                "version": "1.0"
            },
            {
                "name": "123456789012::entrance-north",
                "version": "1.0"
            }
        ],
        "nodeOverrides": [
            {
                "replace": "camera_node",
                "with": [
                    {
                        "name": "warehouse-floor"
                    },
                    {
                        "name": "entrance-north"
                    }
                ]
            }
```

有关使用 API 进行部署的详细信息，请参阅 自动化应用程序部署。

使用 VPC 端点

如果您在无法访问互联网的 VPC 中工作，则可以创建一个 VPC 端点以与 AWS Panorama 一起使用。VPC 端点允许在私有子网中运行的客户端无需互联网连接即可连接到 AWS 服务。

有关 AWS Panorama 设备使用的端口和端点的详细信息，请参阅 ???。

Sections

- 创建 VPC 端点
- 将设备连接到私有子网
- 示例 AWS CloudFormation 模板

创建 VPC 端点

要在 VPC 和 AWS Panorama 之间建立私有连接，请创建 VPC 端点。使用 AWS Panorama 不需要 VPC 端点。只有在无法访问互联网的 VPC 中工作时，才需要创建 VPC 端点。当 AWS CLI 或软件开发工具包尝试连接到 AWS Panorama 时，流量将通过 VPC 端点进行路由。

使用以下设置为 AWS Panorama 创建 VPC 端点：

- 服务名称 – **com.amazonaws.*us-west-2*.panorama**
- 类型 – 接口

VPC 端点可使用服务的 DNS 名称从 AWS 软件开发工具包客户端获取流量，而无需任何额外配置。有关使用 VPC 端点的更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 端点。

将设备连接到私有子网

AWS Panorama 设备可以通过私有 VPN 连接与 AWS Site-to-Site VPN 或进行连接 AWS Direct Connect。 AWS 借助这些服务，您可以创建一个延伸至数据中心的私有子网。设备连接到私有子网并通过 VPC 终端节点访问 AWS 服务。

Site-to-Site VPN 以及 Direct Connect 用于将您的数据中心安全地连接到 Amazon VPC 的服务。使用 Site-to-Site VPN，您可以使用市售的网络设备进行连接。 Direct Connect 使用 AWS 设备进行连接。

- Site-to-Site VPN — 什么是 AWS Site-to-Site VPN？
- Direct Connect – 什么是 AWS Direct Connect？

将本地网络连接到 VPC 中的私有子网后，请为以下服务创建 VPC 端点。

- Amazon Simple Storage Service – 适用于 Amazon S3 的AWS PrivateLink
- AWS IoT Core – 将 AWS IoT Core 与接口 VPC 端点配合使用（数据面板和凭证提供程序）
- Amazon Elastic 容器注册表 – 亚马逊弹性容器注册表接口 VPC 端点
- 亚马逊 CloudWatch — 使用 CloudWatch 接口 VPC 终端节点
- Amazon CloudWatch 日志 — 使用带有接口 VPC 终端节点的 CloudWatch 日志

该设备无需连接 AWS Panorama 服务。它通过中的消息渠道与 AWS Panorama 通信。 AWS IoT

除了 VPC 终端节点外，Amazon S3 和还 AWS IoT 需要使用 Amazon Route 53 私有托管区域。私有托管区域将来自子域（包括 Amazon S3 接入点和 MQTT 主题的子域）的流量路由到正确的 VPC 端点。有关私有托管区域的信息，请参阅 Amazon Route 53 开发者指南中的使用私有托管区域。

有关包含 VPC 端点和私有托管区域的 VPC 配置示例，请参阅 示例 AWS CloudFormation 模板。

示例 AWS CloudFormation 模板

本指南的 GitHub 存储库提供了 AWS CloudFormation 模板，您可以使用这些模板来创建用于 AWS Panorama 的资源。这些模板创建了一个包含两个私有子网、一个公有子网和一个 VPC 端点的 VPC。您可以使用 VPC 中的私有子网来托管与互联网隔离的资源。公有子网中的资源可以与私有资源通信，但无法从互联网访问私有资源。

Example vpc-endpoint.yml – 私有子网

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 172.31.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
      Tags:
        - Key: Name
          Value: !Ref AWS::StackName
  privateSubnetA:
    Type: AWS::EC2::Subnet
    Properties:
```

```
      VpcId: !Ref vpc
      AvailabilityZone:
        Fn::Select:
         - 0
         - Fn::GetAZs: ""
      CidrBlock: 172.31.3.0/24
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub  ${AWS::StackName}-subnet-a
...
```

vpc-endpoint.yml 模板展示了如何为 AWS Panorama 创建 VPC 端点。您可以使用此终端节点通过 AWS 软件开发工具包管理 AWS Panorama 资源或 AWS CLI。

Example vpc-endpoint.yml – VPC 端点

```
panoramaEndpoint:
  Type: AWS::EC2::VPCEndpoint
  Properties:
    ServiceName: !Sub com.amazonaws.${AWS::Region}.panorama
    VpcId: !Ref vpc
    VpcEndpointType: Interface
    SecurityGroupIds:
    - !GetAtt vpc.DefaultSecurityGroup
    PrivateDnsEnabled: true
    SubnetIds:
    - !Ref privateSubnetA
    - !Ref privateSubnetB
    PolicyDocument:
      Version: 2012-10-17
      Statement:
      - Effect: Allow
        Principal: "*"
        Action:
          - "panorama:*"
        Resource:
          - "*"
```

PolicyDocument 是一种基于资源的权限策略，用于定义可使用端点进行的 API 调用。您可以修改策略以限制可通过端点访问的操作和资源。有关更多信息，请参阅《Amazon VPC User Guide》中的 Controlling access to services with VPC endpoints。

vpc-appliance.yml 模板展示了如何为 AWS Panorama 设备使用的服务创建 VPC 端点和私有托管区域。

Example vpc-appliance.yml – 带有私有托管区域的 Amazon S3 接入点端点

```
  s3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub com.amazonaws.${AWS::Region}.s3
      VpcId: !Ref vpc
      VpcEndpointType: Interface
      SecurityGroupIds:
      - !GetAtt vpc.DefaultSecurityGroup
      PrivateDnsEnabled: false
      SubnetIds:
      - !Ref privateSubnetA
      - !Ref privateSubnetB
...
  s3apHostedZone:
    Type: AWS::Route53::HostedZone
    Properties:
      Name: !Sub s3-accesspoint.${AWS::Region}.amazonaws.com
      VPCs:
        - VPCId: !Ref vpc
          VPCRegion: !Ref AWS::Region
  s3apRecords:
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref s3apHostedZone
      Name: !Sub "*.s3-accesspoint.${AWS::Region}.amazonaws.com"
      Type: CNAME
      TTL: 600
      # first DNS entry, split on :, second value
      ResourceRecords:
      - !Select [1, !Split [":", !Select [0, !GetAtt s3Endpoint.DnsEntries ] ] ]
```

示例模板演示了如何使用示例 VPC 创建 Amazon VPC 和 Route 53 资源。您可以通过删除 VPC 资源并将对子网、安全组和 VPC 的引用替换为您的资源来调整这些内容，以适应您的 IDs 用例。 IDs

示例应用程序、脚本和模板

本指南的 GitHub 存储库为 AWS Panorama 设备提供了示例应用程序、脚本和模板。使用这些示例来学习最佳实践并自动执行开发工作流。

Sections

示例应用程序

示例应用程序演示了 AWS Panorama 功能的使用和常见的计算机视觉任务。这些示例应用程序包括用于自动执行设置和部署的脚本和模板。只需进行最少的配置，即可从命令行部署和更新应用程序。

- aws-panorama-sample— 带有分类模型的基本计算机视觉。使用将指标上传 AWS SDK for Python (Boto) 到 CloudWatch、仪器预处理和推理方法以及配置日志记录。
- debug-server – 在设备上打开入站端口，并将流量转发到应用程序代码容器。使用多线程同时运行应用程序代码、HTTP 服务器和 HTTP 客户端。
- 自定义模型 — 从代码中导出模型并使用 SageMaker AI Neo 进行编译，以测试与 AWS Panorama 设备的兼容性。在 Python 开发中、在 Docker 容器中或亚马逊 EC2 实例中进行本地构建。在 Keras 中针对特定版本 TensorFlow 或 Python 版本导出和编译所有内置应用程序模型。

如需更多示例应用程序，也请访问aws-panorama-samples存储库。

实用程序脚本

`util-scripts`目录中的脚本可以管理 AWS Panorama 资源或自动执行开发工作流程。

- provision-device.sh – 预置设备。
- check-updates.sh – 检查并应用设备软件更新。

- reboot-device.sh – 重启设备。
- register-camera.sh – 注册一台相机。
- deregister-camera.sh – 删除相机节点。
- view-logs.sh – 查看应用程序实例的日志。
- pause-camera.sh – 暂停或恢复相机流式传输。
- push.sh – 构建、上传和部署应用程序。
- rename-package.sh – 重命名节点包。更新目录名称、配置文件和应用程序清单。
- samplify.sh – 将您的帐户 ID 替换为示例帐户 ID，然后恢复备份配置以删除本地配置。
- update-model-config.sh-更新描述符文件后，将模型重新添加到应用程序中。
- cleanup-patches.sh – 取消注册旧补丁版本并从 Amazon S3 中删除其清单。

有关用法的详细信息，请参阅自述文件。

CloudFormation 模板

使用`cloudformation-templates`目录中的 CloudFormation 模板为 AWS Panorama 应用程序创建资源。

- alarm-application.yml – 创建用于监控应用程序错误的警报。如果应用程序实例出现错误或停止运行 5 分钟，警报会发送一封电子邮件通知。
- alarm-device.yml – 创建用于监控设备连接的警报。如果设备停止发送指标 5 分钟，警报会发送一封电子邮件通知。
- application-role.yml – 创建应用程序角色。该角色包括向其发送指标的权限 CloudWatch。在策略声明中为应用程序使用的其他 API 操作添加权限。
- vpc-appliance.yml — 为设备创建具有私有子网服务访问权限的 VPC。 AWS Panorama 要将设备连接到 VPC，请使用 AWS Direct Connect 或 AWS Site-to-Site VPN。
- vpc-endpoint.yml — 创建一个允许私有子网服务访问该服务的 VPC。 AWS Panorama VPC 内部的资源无需连接互联网即可连接， AWS Panorama 以监控和管理 AWS Panorama 资源。

此目录中的`create-stack.sh`脚本创建 CloudFormation 堆栈。它需要可变数量的参数。第一个参数是模板的名称，其余参数将覆盖模板中的参数。

例如，以下命令可创建应用程序角色。

```
$ ./create-stack.sh application-role
```

更多示例和工具

aws-panorama-samples存储库中有更多的示例应用程序和有用的工具。

- 应用程序 – 适用于各种模型架构和用例的示例应用程序。
- 相机流式传输验证 -验证相机流式传输。
- PanoJupyter— 在 AWS Panorama 设备 JupyterLab 上运行。
- Sideloading – 无需构建或部署应用程序容器即可更新应用程序代码。

该 AWS 社区还开发了以下方面的工具和指南： AWS Panorama. 请查看以下开源项目 GitHub。

- cookiecutter-panorama — 应用程序的 Cookiecutt er 模板。 AWS Panorama
- backpack – 用于访问运行时环境详细信息、分析和其他视频输出选项的 Python 模块。

监控 AWS Panorama 资源和应用程序

您可以在 AWS Panorama 控制台和 Amazon 上监控 AWS Panorama 资源 CloudWatch。 AWS Panorama 设备通过互联网连接到 AWS 云端，以报告其状态和已连接摄像机的状态。开启时，设备还会实时向 Lo CloudWatch gs 发送日志。

该设备从您在首次使用 AWS Panorama 控制台时创建的服务角色获得使用 AWS IoT、 CloudWatch 日志和其他 AWS 服务的权限。有关更多信息，请参阅 AWS Panorama 服务角色和跨服务资源。

有关排除特定错误的帮助，请参阅 故障排除。

主题

- 在 AWS Panorama 控制台中进行监控
- 查看 AWS Panorama 日志
- 使用 Amazon 监控设备和应用程序 CloudWatch

在 AWS Panorama 控制台中进行监控

您可以使用 AWS Panorama 控制台监控您的 AWS Panorama Appliance 和摄像机。控制台 AWS IoT 用于监控设备的状态。

在 AWS Panorama 控制台中监控您的设备

1. 打开 AWS Panorama 控制台。
2. 打开 AWS Panorama 控制台的设备页面。
3. 选择设备。
4. 要查看应用程序实例的状态，请从列表中选择。
5. 要查看设备网络接口的状态，请选择设置。

页面顶部会显示设备的总体状态。如果状态为 “在线”，则表示设备已连接 AWS 并定期发送状态更新。

查看 AWS Panorama 日志

AWS Panorama 向亚马逊 CloudWatch 日志报告应用程序和系统事件。遇到问题时，您可以使用事件日志来帮助调试 AWS Panorama 应用程序或排除应用程序配置的故障。

在 “日志” 中查看 CloudWatch 日志

1. 打开日志控制台的 CloudWatch 日志组页面。
2. 在以下组中查找 AWS Panorama 应用程序和设备日志：

   - 设备日志 - `/aws/panorama/devices/device-id`
   - 应用程序日志 - `/aws/panorama/devices/device-id/applications/instance-id`

更新系统软件后重新配置设备时，您还可以查看预置 USB 驱动器上的日志。

Sections

查看设备日志

AWS Panorama Appliance 会为设备创建一个日志组，并为您部署的每个应用程序实例创建一个组。设备日志包含有关应用程序状态、软件升级和系统配置的信息。

设备日志 - **`/aws/panorama/devices/device-id`**

- `occ_log` - 控制器进程的输出。该流程负责协调应用程序的部署，并报告每个应用程序实例节点的状态。
- `ota_log`— 协调 over-the-air (OTA) 软件升级过程的输出。
- `syslog` - 设备的 syslog 进程的输出，用于捕获进程之间发送的消息。
- `kern_log` - 来自设备 Linux 内核的事件。
- `logging_setup_logs`— 配置 CloudWatch 日志代理的过程的输出。

- `cloudwatch_agent_logs`— 来自 CloudWatch 日志代理的输出。
- `shadow_log` - 来自AWS IoT 设备影子的输出。

查看应用程序日志

应用程序实例的日志组包含以节点命名的每个节点的日志流。

应用程序日志 - **/aws/panorama/devices/*device-id*/applications/*instance-id***

- 代码 - 应用程序代码和 AWS Panorama 应用程序 SDK 的输出。聚合来自 `/opt/aws/panorama/logs` 的应用程序日志。
- 模型 - 使用模型协调推理请求的过程的输出。
- 流 - 解码摄像机视频流过程中的输出。
- 显示 - 从渲染 HDMI 端口视频输出过程中的输出。
- `mds` - 来自设备元数据服务器的日志。
- `console_output` - 捕获代码容器中的标准输出和错误流。

如果您在日志中看不到日 CloudWatch 志，请确认您位于正确的 AWS 区域。如果是，则设备与 AWS 的连接或设备 AWS Identity and Access Management (IAM) 角色的权限可能存在问题。

配置应用程序日志

配置 Python 记录器以将日志文件写入 `/opt/aws/panorama/logs`。设备会将日志从此位置流式传输到 CloudWatch 日志。为避免占用过多磁盘空间，请使用最大为 10 MiB 的文件，备份次数为 1。以下示例展示了创建记录器的方法。

Example application.py - 记录器配置

```
def get_logger(name=__name__,level=logging.INFO):
    logger = logging.getLogger(name)
    logger.setLevel(level)
    LOG_PATH = '/opt/aws/panorama/logs'
    handler = RotatingFileHandler("{}/app.log".format(LOG_PATH), maxBytes=10000000,
 backupCount=1)
    formatter = logging.Formatter(fmt='%(asctime)s %(levelname)-8s %(message)s',
                                  datefmt='%Y-%m-%d %H:%M:%S')
    handler.setFormatter(formatter)
```

```
    logger.addHandler(handler)
    return logger
```

在全局范围内初始化记录器，并在整个应用程序代码中使用。

Example application.py - 初始化记录器

```
def main():
    try:
        logger.info("INITIALIZING APPLICATION")
        app = Application()
        logger.info("PROCESSING STREAMS")
        while True:
            app.process_streams()
            # turn off debug logging after 150 loops
            if logger.getEffectiveLevel() == logging.DEBUG and app.frame_num == 150:
                logger.setLevel(logging.INFO)
    except:
        logger.exception('Exception during processing loop.')

logger = get_logger(level=logging.INFO)
main()
```

查看预置日志

在预置期间，AWS Panorama Appliance 会将日志复制到用于将配置存档传输到设备的 USB 驱动器。使用这些日志可排除采用最新软件版本的设备上的预置问题故障。

> **Important**
>
> 预置日志适用于更新至软件版本 4.3.23 或更高版本的设备。

应用程序日志

- `/panorama/occ.log` - AWS Panorama 控制器软件日志。
- `/panorama/ota_agent.log`— AWS Panorama over-the-air 更新代理日志。
- `/panorama/syslog.log` - Linux 系统日志。
- `/panorama/kern.log` - Linux 内核日志。

从设备导出日志

如果您的设备和应用程序日志未显示在 “ CloudWatch 日志” 中，则可以使用 USB 驱动器从设备上获取加密的日志图像。AWS Panorama 服务团队可以代表您解密日志并协助您进行调试。

先决条件

要按照步骤操作，您将需要以下硬件：

- U 盘 — 一种 FAT32格式化的 USB 闪存盘，存储空间至少为 1 GB，用于从 AWS Panorama 设备传输日志文件。

从设备导出日志

1. 准备一个 USB 驱动器，在 `panorama` 文件夹里放入 `managed_logs` 文件夹。

   ```
   /
    ### panorama
        ### managed_logs
   ```

2. 将 USB 驱动器连接到设备。
3. 关闭 AWS Panorama Appliance。
4. 打开 AWS Panorama Appliance。
5. 该设备将日志复制到设备。在执行此操作时，状态 LED 指示灯会闪烁蓝色。
6. 然后可以在 `managed_logs` 目录中找到日志文件，格式为 `panorama_device_log_`*`v1_dd_hh_mm`*`.img`

您无法自行解密日志图像。请与客户支持、AWS Panorama 的技术客户经理或解决方案架构师合作，与服务团队协调。

使用 Amazon 监控设备和应用程序 CloudWatch

当设备处于联机状态时，AWS Panorama 会向亚马逊发送指标 CloudWatch。您可以在 CloudWatch 控制台中使用这些指标构建图表和仪表板，以监控设备活动，并设置警报，以便在设备离线或应用程序遇到错误时通知您。

在 CloudWatch 控制台中查看指标

1. 打开 AWS Panorama 控制台的指标页面（`PanoramaDeviceMetrics` 命名空间）。
2. 选择维度架构。
3. 选择指标以将其添加到图表。
4. 要选择其他统计数据并自定义图表，请使用图表化指标选项卡上的选项。默认情况下，图表将使用所有指标的 `Average` 统计数据。

> **定价**
>
> CloudWatch 有 “永远免费” 等级。超出免费套餐阈值时，会对指标、仪表板、警报、日志和见解 CloudWatch 收费。有关详细信息，请参阅 CloudWatch 定价。

有关的更多信息 CloudWatch，请参阅 Amazon CloudWatch 用户指南。

Sections

使用设备指标

当设备处于联机状态时，它会向 Amazon 发送指标 CloudWatch。您可以使用这些指标来监控设备活动，并在设备离线时触发警报。

- `DeviceActive` - 设备处于活动状态时定期发送。

  维度 - `DeviceId` 和 `DeviceName`。

查看 DeviceActive 指标与 Average 统计数据。

使用应用程序指标

当应用程序遇到错误时，它会向 Amazon 发送指标 CloudWatch。应用程序停止运行时，您可以使用这些指标触发警报。

- ApplicationErrors - 记录的应用程序错误数。

  维度 - ApplicationInstanceName 和 ApplicationInstanceId。

使用 Sum 统计数据查看应用程序指标。

配置警报

要在指标超过阈值时收到通知，请创建警报。例如，您可以创建一个警报，当 ApplicationErrors 指标的总和连续 20 分钟保持在 1 时发送通知。

创建警报

1. 打开 Amazon CloudWatch 控制台的 “警报” 页面。
2. 选择创建警报。
3. 选择选择指标，然后找到适合您设备的指标，例如 applicationInstance-gk75xmplqbqtenlnmz4ehiu7xa 与 my-application 的 ApplicationErrors。
4. 按照说明为警报配置条件、操作和名称。

有关详细说明，请参阅 Amazon CloudWatch 用户指南中的创建 CloudWatch 警报。

故障排除

以下主题针对您在使用 AWS Panorama 控制台、设备或 SDK 时可能遇到的错误和问题提供了故障排除建议。如果您发现某个问题未在此处列出，请使用此页面上的提供反馈按钮进行报告。

您可以在 Amazon Logs 控制台中查找设备的 CloudWatch 日志。设备在生成应用程序代码、设备软件和 AWS IoT 进程时上传日志。有关更多信息，请参阅 查看 AWS Panorama 日志。

预置

问题：(macOS) 我的计算机无法识别带有 USB-C 适配器的 USB 驱动器。

如果将 USB 驱动器插入已连接到计算机的 USB-C 适配器，则可能会发生这种情况。尝试断开适配器，然后将其与已连接的 USB 驱动器重新连接。

问题：当我使用自己的 USB 驱动器时，预置失败。

问题：使用设备的 USB 2.0 端口时，预置失败。

本 AWS Panorama 设备与 1 到 32 GB 的 USB 闪存设备兼容，但并非所有设备都兼容。使用 USB 2.0 端口进行预置时，已观察到一些问题。为了获得一致的结果，请使用附带的 USB 驱动器和 USB 3.0 端口（靠近 HDMI 端口）。

对于联想 ThinkEdge® SE7 0，设备中不包括USB驱动器。使用至少具有 1 GB 存储空间的 USB 3.0 驱动器。

设备配置

问题：设备在启动过程中显示空白屏幕。

完成初始启动序列（大约需要一分钟）后，设备会在加载模型并启动应用程序时显示一分钟或更长时间的空白屏幕。此外，如果在显示器打开后连接显示器，设备不会输出视频。

问题：当我按住电源按钮将其关闭时，设备没有响应。

设备最多需要 10 秒才能安全关闭。您只需按住电源按钮 1 秒钟即可开始关机序列。有关按钮操作的完整列表，请参阅 AWS Panorama 设备按钮和指示灯。

问题：我需要生成新的配置存档来更改设置或替换丢失的证书。

AWS Panorama 下载设备证书或网络配置后不会存储它们，并且您无法重复使用配置存档。使用 AWS Panorama 控制台删除设备，然后使用新的配置存档创建一个新设备。

应用程序配置

问题：当我运行多个应用程序时，我无法控制哪个应用程序使用 HDMI 输出。

部署具有输出节点的多个应用程序时，最近启动的应用程序将使用 HDMI 输出。如果此应用程序停止运行，则另一个应用程序可以使用该输出。要仅授予一个应用程序访问输出的权限，请从另一个应用程序的应用程序清单中删除输出节点和相应的边缘，然后重新部署。

问题：应用程序输出未显示在日志中

配置 Python 记录器以将日志文件写入 `/opt/aws/panorama/logs`。这些内容在代码容器节点的日志流中捕获。标准输出和错误流在名为 `console-output` 的单独日志流中捕获。如果使用 `print`，请使用 `flush=True` 选项来防止消息卡在输出缓冲区中。

错误：You've reached the maximum number of versions for package SAMPLE_CODE. Deregister unused package versions and try again.

来源： AWS Panorama 服务

每次对应用程序进行更改时，都会注册一个补丁版本，该补丁版本表示它使用的每个包的包配置和资产文件。使用清理补丁脚本取消注册未使用的补丁版本。

相机流式传输

错误：liveMedia0: Failed to get SDP description: Connection to server failed: Connection timed out (-115)

错误：liveMedia0: Failed to get SDP description: 404 Not Found; with the result code: 404

错误：liveMedia0: Failed to get SDP description: DESCRIBE send() failed: Broken pipe; with the result code: -32

来源：相机节点日志

设备无法连接到应用程序的相机流式传输。发生这种情况时，当应用程序等待来自应用程序 SDK 的视频帧时，视频输出为空白或在最后处理的帧上冻结。 AWS Panorama 设备软件尝试连接到相机流并在

摄像机节点日志中记录超时错误。验证您的摄像头流 URL 是否正确，以及 RTSP 流量是否可在网络内的摄像头和设备之间路由。有关更多信息，请参阅 将 AWS Panorama Appliance 连接到您的网络。

错误：ERROR finalizeInterface(35) Camera credential fetching for port [username] failed

来源：OCC 日志

找不到摄像机直播凭据的 AWS Secrets Manager 秘密。删除相机流并重新创建它。

错误：Camera did not provide an H264 encoded stream

来源：相机节点日志

相机流具有 H.264 以外的编码，例如 H.265。使用 H.264 相机流重新部署应用程序。有关支持的相机的详细信息，请参阅 支持的摄像头。

AWS Panorama 中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方 AWS 的共同责任。责任共担模式将其描述为云的 安全性和云中 的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为 AWS 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用于 AWS Panorama 的合规性计划，请参阅合规性计划范围内的AWS 服务。
- 云中的安全性：您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 AWS Panorama 时应用责任共担模型 以下主题说明如何配置 AWS Panorama 以实现您的安全性和合规性目标。您还将了解如何使用其他亚马逊云科技服务来帮助您监控和保护 AWS Panorama 资源。

主题

- AWS Panorama Appliance 安全功能
- AWS Panorama 设备安全最佳实践
- AWS Panorama 中的数据保护
- 适用于 AWS Panorama 的身份和访问管理
- AWS Panorama 的合规性验证
- AWS Panorama 中的基础设施安全性
- AWS Panorama 中的运行时环境软件

AWS Panorama Appliance 安全功能

为保护您的应用程序、模型和硬件免受恶意代码和其他漏洞的攻击，AWS Panorama Appliance 实施了一套全面的安全功能。这些功能包括但不限于以下内容：

- 全盘加密 — 设备实现 Linux 统一密钥设置 (LUKS2) 全盘加密。所有系统软件和应用程序数据均使用设备专用的密钥加密。即使对设备进行物理访问，攻击者也无法检查存储中的内容。
- 内存布局随机化 - 为防范针对加载到内存中的可执行代码的攻击，AWS Panorama Appliance 使用了地址空间布局随机化 (ASLR)。当操作系统代码加载到内存中时，ASLR 会随机化其位置。该操作可以防止有人利用漏洞，通过预测代码在运行时的存储位置尝试覆盖或运行特定部分的代码。
- 可信执行环境-该设备使用基于 ARM 的可信执行环境 (TEE) TrustZone，具有隔离的存储、内存和处理资源。存储在信任区域中的密钥和其他敏感数据只能由可信应用程序访问，该应用程序在 TEE 内的独立操作系统中运行。AWS Panorama Appliance 软件与应用程序代码一起在不受信任的 Linux 环境中运行。它只能通过向安全应用程序发出请求来访问加密操作。
- 安全预置 - 当您预置设备时，传输到设备的凭证（密钥、证书和其他加密材料）仅在短时间内有效。设备使用短期凭证进行连接， AWS IoT 并为其自身请求有效期更长的证书。AWS Panorama 服务会生成凭证，并使用设备上硬编码的密钥对其进行加密。只有申请证书的设备才能对其进行解密并与 AWS Panorama 通信。
- 安全启动 - 设备启动时，每个软件组件在运行之前都要经过验证。引导 ROM 是处理器中硬编码的软件，不可修改，它使用硬编码加密密钥来解密启动加载程序，从而验证可信的执行环境内核等。
- 已签名的内核 - 使用非对称加密密钥对内核模块进行签名。操作系统内核使用公有密钥解密签名，并在将模块加载到内存之前验证签名是否与模块的签名相匹配。
- dm-verity - 与验证内核模块的方式类似，设备在装载设备软件映像前使用 Linux Device Mapper 的 `dm-verity` 功能来验证其完整性。如果设备软件被修改，则无法运行。
- 回滚保护 - 更新设备软件时，设备会熔断 SoC（系统级芯片）上的电子保险丝。每个软件版本预计会有越来越多的保险丝被熔断，如果更多保险丝被熔断，软件就无法运行。

AWS Panorama 设备安全最佳实践

使用 AWS Panorama 设备时，请牢记以下最佳实践。

- 对设备进行物理保护 – 将设备安装在封闭的服务器机架或安全房间中。仅限授权人员实际访问设备。
- 保护设备的网络连接 – 将设备连接到限制内部和外部资源访问的路由器。该设备需要连接到位于安全内部网络上的摄像头。它还需要连接到 AWS。将第二个以太网端口仅用于物理冗余，并将路由器配置为仅允许所需流量。

  使用推荐的网络配置之一来规划网络布局。有关更多信息，请参阅 将 AWS Panorama Appliance 连接到您的网络。
- 格式化 USB 驱动器 – 预置设备后，移除 USB 驱动器并将其格式化。设备在 AWS Panorama 服务中注册后不会使用 USB 驱动器。格式化驱动器以删除临时凭证、配置文件和预置日志。
- 使设备保持最新状态 – 及时应用设备软件更新。当您在 AWS Panorama 控制台中查看设备时，控制台会通知您是否有可用的软件更新。有关更多信息，请参阅 管理 AWS Panorama Appliance。

  DescribeDevice通过 API 操作，您可以通过比较`LatestSoftware`和`CurrentSoftware`字段来自动检查更新。当最新的软件版本与当前版本不同时，请使用控制台或使用CreateJobForDevices操作来应用更新。
- 如果停止使用设备，请重置设备 – 在将设备移出安全数据中心之前，请将其完全重置。关闭设备电源并接通电源后，同时按住电源和重置按钮 5 秒钟。这将从设备中删除帐户凭证、应用程序和日志。

  有关更多信息，请参阅 AWS Panorama 设备按钮和指示灯。
- 限制对 AWS Panorama 和其他 AWS 服务的访问 — 允许访问所有 AWS Panorama API 操作，并在必要时访问其他服务。AWSPanoramaFullAccess在可能的情况下，该策略会根据命名约定限制对资源的访问。例如，它提供对名称以开头的 AWS Secrets Manager 机密的访问权限`panorama`。对于需要只读访问权限或访问更具体的资源集的用户，请使用托管策略作为最低权限策略的起点。

  有关更多信息，请参阅 AWS Panorama 的基于身份的 IAM 策略。

AWS Panorama 中的数据保护

AWS 分担责任模型分适用于 AWS Panorama 中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与资源通信。 AWS 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API 或 AWS 服务 使用其他方式使用 AWS Panorama 或其他网站的情况 AWS SDKs。 AWS CLI在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

小节目录

传输中加密

AWS Panorama API 端点仅支持基于 HTTPS 的安全连接。使用 AWS Management Console、AWS 开发工具包或 AWS Panorama API 管理 AWS Panorama 资源时，所有通信都使用传输层安全性协议（TLS）进行加密。AWS Panorama 设备和 AWS 之间的通信也使用 TLS 加密。AWS Panorama 设备与摄像头之间通过 RTSP 进行的通信未加密。

有关 API 端点的完整列表，请参阅 AWS 一般参考 中的 AWS 区域和端点。

AWS Panorama 设备

AWS Panorama 设备具有用于以太网、HDMI 视频和 USB 存储的物理端口。SD 卡插槽、Wi-Fi 和蓝牙不可用。USB 端口仅在配置期间用于将配置存档传输到设备。

配置存档的内容（包括设备的置备证书和网络配置）未加密。AWS Panorama 不存储这些文件;只有在注册设备时才能检索它们。将配置存档传输到设备后，将其从计算机和 USB 存储设备中删除。

设备的整个文件系统均已加密。此外，该设备还应用了多种系统级保护，包括所需软件更新的回滚保护、签名内核和引导加载程序以及软件完整性验证。

停止使用设备时，请执行完全重置以删除应用程序数据并重置设备软件。

应用程序

您可以控制部署到设备的代码。在部署所有应用程序代码之前，无论其来源如何，都要验证其是否存在安全问题。如果在应用程序中使用第三方库，请仔细考虑这些库的许可和支持策略。

应用程序 CPU、内存和磁盘使用率不受设备软件的限制。使用过多资源的应用程序可能会对其他应用程序和设备的运行产生负面影响。在合并或部署到生产环境之前，单独测试应用程序。

应用程序资产（代码和模型）不会与帐户、设备或构建环境中的访问隔离。AWS Panorama 应用 CLI 生成的容器映像和模型存档未加密。对生产工作负载使用单独的帐户，并且仅允许根据需要进行访问。

其他服务

为了将您的模型和应用程序容器安全地存储在 Amazon S3 中，AWS Panorama 使用具有 Amazon S3 管理的密钥的服务器端加密。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的通过加密保护数据。

摄像机直播凭据在静态状态下被加密 AWS Secrets Manager。设备的 IAM 角色授予其检索密钥的权限，以便访问流的用户名和密码。

AWS Panorama 设备将日志数据发送到亚马逊 CloudWatch 日志。 CloudWatch 默认情况下，日志会对这些数据进行加密，并且可以将其配置为使用客户托管密钥。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南 AWS KMS中的使用加密 CloudWatch 日志中的日志数据。

适用于 AWS Panorama 的身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制可以通过身份验证（登录）和授权（拥有权限）使用 AWS Panorama 资源的人员。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅对 AWS Panorama 身份和访问权限进行故障排除）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅 AWS Panorama 如何与 IAM 一起使用）
- IAM 管理员：编写用于管理访问权限的策略（请参阅 AWS Panorama 基于身份的策略示例）

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center （例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 User Guide》中的 How to sign in to your AWS 账户。

对于编程访问， AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。强烈建议您不要使用根用户执行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

IAM 用户和群组

IAM 用户是对单个人员或应用程序具有特定权限的身份。我们建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户的使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon 上运行的应用程序非常有用。 EC2有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以对什么资源以及在什么条件下执行操作，来指定谁有权访问什么内容。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以代入这些角色。IAM 策略定义权限，而不考虑您使用哪种方法来执行操作。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可在什么条件下对哪些资源执行什么操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管式策略（附加到多个身份的独立策略）。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。您必须在基于资源的策略中指定主体。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持的服务示例 ACLs。 AWS WAF要了解更多信息 ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界：设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅 AWS Organizations 用户指南中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略：在为角色或联合用户创建临时会话时，作为参数传递的高级策略。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

AWS Panorama 如何与 IAM 一起使用

在使用 IAM 管理 AWS Panorama 的访问权限之前，您应了解哪些 IAM 功能可用于 AWS Panorama。要全面了解 AWS Panorama 和其他 AWS 服务如何与 IAM 配合使用，请参阅 IAM 用户指南中与 IAM 配合使用的AWS 服务。

有关 AWS Panorama 使用的权限、策略和角色的概述，请参阅 AWS Panorama 权限。

AWS Panorama 基于身份的策略示例

默认情况下，IAM 用户和角色没有创建或修改 AWS Panorama 资源的权限。他们也无法使用 AWS Management Console AWS CLI、或 AWS API 执行任务。IAM 管理员必须创建 IAM 策略，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅《IAM 用户指南》中的在 JSON 选项卡上创建策略。

主题

- 策略最佳实践
- 使用 AWS Panorama 控制台
- 允许用户查看他们自己的权限

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 AWS Panorama 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略或工作职能的AWS 托管式策略。

- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 AWS Panorama 控制台

要访问 AWS Panorama 控制台，您必须拥有最低权限。这些权限必须允许您列出和查看有关您 AWS 账户中 AWS Panorama 资源的详细信息。如果您创建的基于身份的策略比所需的最低权限更严格，则无法为具有该策略的实体（IAM 用户或角色）正常运行控制台。

有关更多信息，请参阅AWS Panorama 的基于身份的 IAM 策略

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
```

```
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS AWS Panorama 的托管策略

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略。

AWS Panorama 提供以下托管策略。有关每个策略的完整内容和更改历史记录，请参阅 IAM 控制台中的链接页面。

- AWSPanoramaFullAccess— 提供对 AWS Panorama、亚马逊 S3 中的 AWS Panorama 接入点 AWS Secrets Manager、中的设备证书和亚马逊中的设备日志的完全访问权限 CloudWatch。包括为 AWS Panorama 创建服务相关角色的权限。
- AWSPanoramaServiceLinkedRolePolicy— 允许 AWS Panorama 管理 AWS 物联网、AWS Secrets Manager 和 AWS Panorama 中的资源。
- AWSPanoramaApplianceServiceRolePolicy— 允许 AWS Panorama 设备将日志上传到 AWS Panorama 创建的 Amazon S3 接入点并从中获取对象。 CloudWatch

AWS Panorama 更新 AWS 了托管策略

下表介绍了对 AWS Panorama 托管策略的更新。

更改	描述	日期
AWSPanoramaApplianceServiceRolePolicy — 更新现有政策	将 StringLike 条件替换 ArnLike 为用于写入 ARNs。	2024-12-10
AWSPanoramaFullAccess — 更新现有政策	将 StringLike 条件替换 ArnLike 为用于写入 ARNs。	2024-12-10
AWSPanoramaFullAccess — 更新现有政策	向用户策略添加了权限，允许用户在日志控制台中查看 CloudWatch 日志组。	2022-01-13
AWSPanoramaFullAccess — 更新现有政策	向用户策略添加了权限，允许用户管理 AWS Panorama 服务相关角色以及访问其他服务（包括 IAM、Amazon S3 和 Secrets Manager）中的 AWS Panorama 资源。 CloudWatch	2021-10-20
AWSPanoramaApplianceServiceRolePolicy — 新政策	AWS Panorama Appliance 服务角色的新策略	2021-10-20
AWSPanoramaServiceLinkedRolePolicy — 新政策	AWS Panorama 服务相关角色的新策略。	2021-10-20

更改	描述	日期
AWS Panorama 开始跟踪更改	AWS Panorama 开始跟踪其 AWS 托管策略的变更。	2021-10-20

将服务相关角色用于 AWS Panorama

AWS Panorama 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种与之直接关联的 IAM 角色的独特类型。 AWS Panorama服务相关角色由服务预定义 AWS Panorama ，包括该服务代表您调用其他 AWS 服务所需的所有权限。

服务相关角色使设置变得 AWS Panorama 更加容易，因为您不必手动添加必要的权限。 AWS Panorama 定义其服务相关角色的权限，除非另有定义，否则 AWS Panorama 只能担任其角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务相关角色。这将保护您的 AWS Panorama 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的AWS 服务，并查找 Service-linked role（服务相关角色）列中显示为 Yes（是）的服务。请选择是与查看该服务的服务相关角色文档的链接。

Sections

- 的服务相关角色权限 AWS Panorama
- 为创建服务相关角色 AWS Panorama
- 编辑的服务相关角色 AWS Panorama
- 删除的服务相关角色 AWS Panorama
- AWS Panorama 服务相关角色支持的区域

的服务相关角色权限 AWS Panorama

AWS Panorama 使用名为 AWSServiceRoleForAWSPanorama— 允许 AWS Panorama 管理 AWS IoT、AWS Secrets Manager 和 AWS Panorama 中的资源...

AWSServiceRoleForAWSPanorama 服务相关角色信任以下服务来代入该角色：

- `panorama.amazonaws.com`

角色权限策略 AWS Panorama 允许完成以下操作：

- 监控 AWS Panorama 资源
- 管理 AWS Panorama 设备的 AWS IoT 资源
- 访问 AWS Secrets Manager 密钥以获取摄像机凭证

有关权限的完整列表，请在 IAM 控制台中查看 AWSPanoramaServiceLinkedRolePolicy 策略。

您必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

为创建服务相关角色 AWS Panorama

您无需手动创建服务相关角色。当您在 AWS Management Console、或 AWS API 中注册设备时，AWS Panorama 会为您创建服务相关角色。 AWS CLI

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。注册设备时， AWS Panorama 会再次为您创建服务相关角色。

编辑的服务相关角色 AWS Panorama

AWS Panorama 不允许您编辑 AWSServiceRoleForAWSPanorama 服务相关角色。创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

删除的服务相关角色 AWS Panorama

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，必须先清除服务相关角色的资源，然后才能手动删除它。

要删除使用的 AWS Panorama 资源 AWSService RoleForAWSPanorama，请使用本指南以下各节中的步骤。

- 删除版本和应用程序
- 注销设备

> **Note**
>
> 如果您尝试删除资源时 AWS Panorama 服务正在使用该角色，则删除可能会失败。如果发生这种情况，请等待几分钟后重试。

要删除 AWSServiceRoleForAWSPanorama 服务相关角色，请使用 IAM 控制台 AWS CLI、或 AWS API。有关更多信息，请参阅《IAM 用户指南》中的删除服务相关角色。

AWS Panorama 服务相关角色支持的区域

AWS Panorama 支持在提供服务的所有地区使用服务相关角色。有关更多信息，请参阅 AWS 区域和端点。

防止跨服务混淆座席

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 AWS，跨服务模仿可能会导致混乱的副手问题。一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况，我们 AWS 提供了一些工具，帮助您保护所有服务的数据，这些服务委托人已被授予访问您账户中资源的权限。

我们建议在资源策略中使用`aws:SourceArn`和`aws:SourceAccount`全局条件上下文密钥来限制为资源 AWS Panorama 提供其他服务的权限。如果使用两个全局条件上下文键，在同一策略语句中使用时，`aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户必须使用相同的账户 ID。

的值`aws:SourceArn`必须是设备的 ARN。 AWS Panorama

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符 (*) 的 `aws:SourceArn` 全局上下文条件键。例如，`arn:aws:`*`servicename`*`::`*`123456789012`*`:*`。

有关保护 AWS Panorama 用于向 AWS Panorama 设备授予权限的服务角色的说明，请参阅保护设备角色。

对 AWS Panorama 身份和访问权限进行故障排除

使用以下信息可帮助您诊断和修复在使用 AWS Panorama 和 IAM 时可能遇到的常见问题。

主题

我无权在 AWS Panorama 中执行操作

如果 AWS Management Console 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。管理员是指提供用户名和密码的人员。

当 `mateojackson` IAM 用户尝试使用控制台查看设备的详细信息，但不具有 `panorama:DescribeAppliance` 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 panorama:DescribeAppliance on resource: my-appliance
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `panorama:DescribeAppliance` 操作访问 `my-appliance` 资源。

我无权执行 iam：PassRole

如果您收到“无权执行 `iam:PassRole` 操作”的错误信息，则必须更新策略以允许您将向 AWS Panorama 传递角色。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 AWS Panorama 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想要允许我的 AWS 账户之外的用户访问我的 AWS Panorama 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 AWS Panorama 是否支持这些特征，请参阅 AWS Panorama 如何与 IAM 一起使用。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

AWS Panorama 的合规性验证

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

有他人在场时的其他注意事项

以下是在可能有他人在场的场景中使用 AWS Panorama 时要考虑的一些最佳实践：

- 确保您了解并遵守适用于您的用例的所有适用法律和法规。这可能包括与相机的位置和视野相关的法律、放置和使用相机时的通知和标识要求，以及视频中可能出现的人员的权利（包括其隐私权）。
- 请考虑您的相机对他人及其隐私的影响。除了法律要求外，还要考虑在相机所在区域放置告示是否合适，以及相机是否应放置在视线范围内且没有任何遮挡物，这样人们就不会对自己可能出现在镜头前感到惊讶。
- 制定适当的政策和程序来操作相机并审查从相机获取的数据。
- 对于从相机获取的数据，请考虑适当的访问控制、使用限制和保留期。

AWS Panorama 中的基础设施安全性

作为一项托管服务，AWS Panorama 受到 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 AWS Panorama。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

在您的数据中心部署 AWS Panorama Appliance

AWS Panorama 设备需要互联网访问才能与 AWS 服务通信。还需要访问您的内部摄像机网络。请务必仔细检查您的网络配置，只为每台设备提供所需的访问权限。如果您的配置允许 AWS Panorama Appliance 充当通往敏感 IP 摄像机网络的桥梁，请务必小心。

您负责执行以下操作：

- AWS Panorama Appliance 的物理和逻辑网络安全。
- 使用 AWS Panorama Appliance 时，安全地操作网络连接的摄像头。
- 更新 AWS Panorama Appliance 和相机软件。
- 遵守与您从生产环境中收集的视频和图像内容相关的任何适用法律或法规，包括与隐私相关的法律或法规。

AWS Panorama Appliance 使用未加密的 RTSP 摄像机视频流。有关将 AWS Panorama Appliance 连接到网络的详细信息，请参阅 将 AWS Panorama Appliance 连接到您的网络。有关加密的详细信息，请参阅 AWS Panorama 中的数据保护。

AWS Panorama 中的运行时环境软件

AWS Panorama 提供的软件可在 AWS Panorama 设备上基于 Ubuntu Linux 的环境中运行应用程序代码。AWS Panorama 负责使设备映像中的软件保持最新状态。AWS Panorama 会定期发布软件更新，您可以使用 AWS Panorama 控制台应用这些更新。

您可以通过在应用程序的 `Dockerfile` 中安装库来在应用程序代码中使用这些库。若要确保应用程序在生成过程中的稳定性，请选择每个库的特定版本。定期更新依赖项以解决安全问题。

发行版

下表显示了 AWS Panorama 服务、软件和文档的功能和软件更新的发布时间。为确保您可以访问所有功能，请将 AWS Panorama 设备更新到最新的软件版本。有关版本的更多信息，请参阅链接的主题。

变更	说明	日期
终止支持通知	终止支持通知：2026 年 5 月 31 日， AWS 将终止对的支持。 AWS Panorama 2026 年 5 月 31 日之后，您将无法再访问 AWS Panorama 控制台或 AWS Panorama 资源。有关更多信息，请参阅AWS Panorama 终止支持。	2025 年 5 月 20 日
更新了托管式策略	AWS Identity and Access Management 的托管策略 AWS Panorama 已更新。有关详细信息，请参阅 AWS 托管式策略。	2024 年 12 月 10 日
设备软件更新	版本 7.0.13 是一个主要版本更新，它改变了设备管理软件更新的方式。如果您限制设备出站的网络通信，或将其连接到私有 VPC 子网，则必须在应用更新前允许访问其他端点和端口。有关更多信息，请参阅更改日志。	2023 年 12 月 28 日
设备软件更新	版本 6.2.1 包括错误修复。有关更多信息，请参阅更改日志。	2023 年 9 月 6 日

设备软件更新	版本 6.0.8 包括错误修复和安全改进。有关更多信息，请参阅更改日志。	2023 年 7 月 6 日
设备软件更新	版本 5.1.7 包括错误修复和错误处理改进。有关更多信息，请参阅更改日志。	2023 年 3 月 31 日
控制台更新	现在，您可以从管理控制台购买该 AWS Panorama 设备。要授予用户购买设备的权限，请参阅适用于 AWS Panorama 的基于身份的 IAM 策略。	2023 年 2 月 2 日
设备软件更新	版本 5.0.74 包括错误修复和错误处理改进。有关更多信息，请参阅更改日志。	2023 年 1 月 23 日
API 更新	向 `OTAJobConfig` 添加了 `AllowMajorVersionUpdate` 选项，以使设备软件主要版本更新选择加入。有关更多信息，请参阅 CreateJobForDevices。	2023 年 1 月 19 日
面向开发人员的新工具	AWS Panorama 示例 GitHub 存储库中提供了一个名为 “侧载” 的新工具。您可以使用此工具更新应用程序代码，而无需构建和部署容器。有关更多信息，请参阅自述文件。	2022 年 11 月 16 日
应用程序基础映像更新	版本 1.2.0 为 `video_in.get()` 添加了超时选项，设置了 `AWS_REGION` 环境变量，并改进了错误处理。有关更多信息，请参阅更改日志。	2022 年 11 月 16 日

设备软件更新	版本 5.0.42 包括错误修复和安全更新。有关更多信息，请参阅更改日志。	2022 年 11 月 16 日
设备软件更新	版本 5.0.7 增加了对远程重启设备和远程暂停摄像头流式传输的支持。有关更多信息，请参阅更改日志。	2022 年 10 月 13 日
设备软件更新	版本 4.3.93 增加了对从离线设备检索日志的支持。有关更多信息，请参阅更改日志。	2022 年 8 月 24 日
设备软件更新	版本 4.3.72 包括错误修复和安全更新。有关更多信息，请参阅更改日志。	2022 年 6 月 23 日
AWS PrivateLink 支持	AWS Panorama 支持用于管理私有子网 AWS Panorama 资源的 VPC 终端节点。有关更多信息，请参阅使用 VPC 端点。	2022 年 6 月 2 日
设备软件更新	4.3.55 版本提高了 `console_output` 日志的存储利用率。有关更多信息，请参阅更改日志。	2022 年 5 月 5 日
联想 ThinkEdge® SE7 0	联想推出了一 AWS Panorama 款新的设备。由 Nvidia Jetson Xavier NX 提供支持的联想 ThinkEdge® SE7 0 支持与该设备相同的功能。 AWS Panorama 有关更多信息，请参阅兼容设备。	2022 年 4 月 6 日

应用程序基础映像更新	版本 1.1.0 提高了运行后台线程时的性能，并为媒体对象添加了一个标志 (is_cached)，用于指示映像是否新鲜。有关更多信息，请参阅 gallery.ecr.aws。	2022 年 3 月 29 日
设备软件更新	版本 4.3.45 增加了对GPU 访问和入站端口的支持。有关更多信息，请参阅更改日志。	2022 年 3 月 24 日
设备软件更新	版本 4.3.35 提高了安全性和性能。有关更多信息，请参阅更改日志。	2022 年 2 月 22 日
更新了托管式策略	AWS Identity and Access Management 的托管策略 AWS Panorama 已更新。有关详细信息，请参阅 AWS 托管式策略。	2022 年 1 月 13 日
配置日志	使用设备软件 4.3.23 时，设备会在配置期间将日志写入 USB 驱动器。有关更多信息，请参阅日志。	2022 年 1 月 13 日
NTP 服务器配置	现在，您可以将 AWS Panorama 设备配置为使用特定的 NTP 服务器进行时钟同步。在设备设置期间使用其他网络设置配置 NTP 设置。有关更多信息，请参阅设置。	2022 年 1 月 13 日
其他区域	AWS Panorama 现已在亚太地区（新加坡）和亚太地区（悉尼）地区推出。	2022 年 1 月 13 日

设备软件更新	版本 4.3.4 增加了对模型 `precisionMode` 设置的支持，并更新了日志记录行为。有关更多信息，请参阅更改日志。	2021 年 11 月 8 日
更新了托管式策略	AWS Identity and Access Management 的托管策略 AWS Panorama 已更新。有关详细信息，请参阅 AWS 托管式策略。	2021 年 10 月 20 日
通用版	AWS Panorama 现已向美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）和加拿大（中部）地区的所有客户开放。要购买 AWS Panorama 设备，请访问AWS Panorama。	2021 年 10 月 20 日
预览	AWS Panorama 应邀在美国东部（弗吉尼亚北部）和美国西部（俄勒冈）地区提供。	2020 年 12 月 1 日