개발자 안내서

AWS Panorama

AWS Panorama: 개발자 안내서

Table of Contents

AWS Panorama란 무엇인가요?

AWS Panorama는 컴퓨터 비전을 온프레미스 카메라 네트워크에 제공하는 서비스입니다. AWS Panorama 어플라이언스 또는 기타 호환 디바이스를 데이터 센터에 설치하고, AWS Panorama를 사용하여 등록하고, 클라우드에서 컴퓨터 비전 애플리케이션을 배포합니다. AWS Panorama은 기존 실시간 스트리밍 프로토콜(RTSP) 네트워크 카메라와 함께 작동합니다. 어플라이언스는 AWS 파트너의 보안 컴퓨터 비전 애플리케이션 또는 AWS Panorama Application SDK로 직접 구축한 애플리케이션을 실행합니다.

AWS Panorama 어플라이언스는 기계 학습 워크로드에 최적화된 강력한 시스템 온 모듈(SOM)을 사용하는 소형 엣지 어플라이언스입니다. 어플라이언스는 여러 비디오 스트림에 대해 여러 컴퓨터 비전 모델을 병렬로 실행하고 결과를 실시간으로 출력할 수 있습니다. 상업 및 산업 환경에서 사용하도록 설계되었으며 방진 및 방수 등급(IP-62)을 받았습니다.

AWS Panorama 어플라이언스를 사용하면 이미지를 AWS 클라우드로 전송하지 않고도 엣지에서 독립형 컴퓨터 비전 애플리케이션을 실행할 수 있습니다. AWS SDK를 사용하면 다른 AWS 서비스와 통합하고 이를 사용하여 시간 경과에 따른 애플리케이션 데이터를 추적할 수 있습니다. 다른 AWS 서비스와 통합하여 AWS Panorama을 사용하면 다음과 같은 작업을 수행할 수 있습니다:

- 트래픽 패턴 분석 – AWS SDK를 사용하여 Amazon DynamoDB에 소매 분석을 위한 데이터를 기록할 수 있습니다. 서버리스 애플리케이션을 사용하여 시간 경과에 따라 수집된 데이터를 분석하고, 데이터에서 이상을 감지하고, 향후 행동을 예측합니다.
- 현장 안전 알림 수신 – 산업 현장의 출입 금지 구역을 모니터링합니다. 애플리케이션에서 위험 가능성이 있는 상황을 감지하면 Amazon Simple Storage Service(S3)에 이미지를 업로드하고 Amazon Simple Notification Service(SNS) 토픽에 알림을 보내 수신자가 시정 조치를 할 수 있도록 합니다.
- 품질 관리 개선 – 조립 라인의 출력을 모니터링하여 요구 사항을 준수하지 않는 부품을 파악합니다. 텍스트와 경계 상자로 부적합 부품의 이미지를 강조 표시하고 품질 관리 팀이 검토할 수 있도록 모니터에 표시합니다.
- 교육 및 테스트 데이터 수집 – 컴퓨터 비전 모델이 식별할 수 없거나 모델의 추측에 대한 신뢰도가 경계선에 있는 물체의 이미지를 업로드합니다. 서버리스 애플리케이션을 사용하여 태그를 지정해야 하는 이미지 대기열을 만들 수 있습니다. 이미지에 태그를 지정하고 이를 사용하여 Amazon SageMaker에서 모델을 다시 학습시킵니다.

AWS Panorama은 다른 AWS 서비스를 사용하여 AWS Panorama 어플라이언스를 관리하고, 모델과 코드에 액세스하고, 애플리케이션을 배포합니다. AWS Panorama은 다른 서비스와 상호 작용할 필요

없이 최대한 많은 작업을 수행하지만, 다음 서비스에 대한 지식이 있으면 AWS Panorama의 작동 방식을 이해하는 데 도움이 될 수 있습니다.

- SageMaker – SageMaker를 사용하여 카메라 또는 센서에서 훈련 데이터를 수집하고, 기계 학습 모델을 구축하고, 컴퓨터 비전에 맞게 훈련할 수 있습니다. AWS Panorama은 SageMaker Neo를 사용하여 AWS Panorama 어플라이언스에서 실행되도록 모델을 최적화합니다.
- Amazon S3 – Amazon S3 액세스 포인트를 사용하여 AWS Panorama 어플라이언스에 배포할 애플리케이션 코드, 모델 및 구성 파일을 스테이징합니다.
- AWS IoT – AWS Panorama은 AWS IoT 서비스를 사용하여 AWS Panorama 어플라이언스의 상태를 모니터링하고, 소프트웨어 업데이트를 관리하고, 애플리케이션을 배포합니다. 직접 AWS IoT를 사용할 필요가 없습니다.

AWS Panorama 어플라이언스를 시작하고 서비스에 대해 자세히 알아보려면 AWS Panorama 시작하기로 진행하십시오.

AWS Panorama 시작하기

AWS Panorama을 시작하려면 먼저 서비스의 개념과 이 설명서에서 사용되는 용어에 대해 알아보십시오. 그런 다음 AWS Panorama 콘솔을 사용하여 AWS Panorama 어플라이언스를 등록하고 애플리케이션을 생성할 수 있습니다. 약 1시간 내에 디바이스를 구성하고, 소프트웨어를 업데이트하고, 샘플 애플리케이션을 배포할 수 있습니다. 이 단원의 자습서를 완료하려면 AWS Panorama 어플라이언스와 로컬 네트워크를 통해 비디오를 스트리밍하는 카메라를 사용합니다.

> **Note**
>
> AWS Panorama 어플라이언스를 구매하려면 AWS Panorama 콘솔을 방문하십시오.

AWS Panorama 샘플 애플리케이션은 AWS Panorama 기능의 사용법을 보여줍니다. 여기에는 SageMaker로 학습된 모델과 AWS Panorama Application SDK를 사용하여 추론을 실행하고 비디오를 출력하는 샘플 코드가 포함되어 있습니다. 샘플 애플리케이션에는 명령줄에서 개발 및 배포 워크플로를 자동화하는 방법을 보여주는 AWS CloudFormation 템플릿과 스크립트가 포함되어 있습니다.

이 장의 마지막 두 주제에서는 모델 및 카메라에 대한 요구 사항과 AWS Panorama 어플라이언스의 하드웨어 사양에 대해 자세히 설명합니다. 아직 어플라이언스와 카메라를 구입하지 않았거나 자체 컴퓨터 비전 모델을 개발할 계획이 없다면 먼저 이 주제에서 자세한 내용을 참조하십시오.

주제

- AWS Panorama 개념
- AWS Panorama 어플라이언스 설정
- AWS Panorama 샘플 애플리케이션 배포
- AWS Panorama 애플리케이션 개발
- 지원되는 컴퓨터 비전 모델 및 카메라
- AWS Panorama 어플라이언스 사양
- Service Quotas

AWS Panorama 개념

AWS Panorama에서는 컴퓨터 비전 애플리케이션을 생성하고 이를 AWS Panorama 어플라이언스 또는 호환 디바이스에 배포하여 네트워크 카메라의 비디오 스트림을 분석합니다. Python으로 애플리케이션 코드를 작성하고 Docker를 사용하여 애플리케이션 컨테이너를 빌드합니다. AWS Panorama Application CLI를 사용하여 기계 학습 모델을 로컬에서 가져오거나 Amazon Simple Storage Service(S3) 에서 가져올 수 있습니다. 애플리케이션은 AWS Panorama Application SDK를 사용하여 카메라로부터 비디오 입력을 수신하고 모델과 상호 작용합니다.

개념

- AWS Panorama 어플라이언스
- 호환되는 디바이스
- 애플리케이션
- 노드
- 모델

AWS Panorama 어플라이언스

AWS Panorama 어플라이언스는 애플리케이션을 실행하는 하드웨어입니다. AWS Panorama 콘솔은 어플라이언스를 등록하고, 소프트웨어를 업데이트하며, 어플라이언스를 배포하는 데 사용합니다. AWS Panorama 어플라이언스의 소프트웨어는 카메라 스트림에 연결하여 애플리케이션으로 비디오 프레임을 전송하고 연결된 디스플레이에 비디오 출력을 표시합니다.

AWS Panorama 어플라이언스는 Nvidia Jetson AGX Xavier로 구동되는 엣지 디바이스입니다. 처리를 위해 이미지를 AWS 클라우드로 보내는 대신 최적화된 하드웨어에서 로컬로 애플리케이션을 실행합니다. 이를 통해 실시간으로 비디오를 분석하고 결과를 로컬에서 처리할 수 있습니다. 어플라이언스의 상태를 보고하고, 로그를 업로드하고, 소프트웨어 업데이트 및 배포를 수행하려면 인터넷 연결이 필요합니다.

자세한 내용은 AWS Panorama 어플라이언스 관리 단원을 참조하세요.

호환되는 디바이스

AWS Panorama 어플라이언스 외에도 AWS Panorama는 AWS 파트너의 호환 가능한 디바이스를 지원합니다. 호환 디바이스는 AWS Panorama 어플라이언스와 동일한 기능을 지원합니다. AWS Panorama 콘솔 및 API를 사용하여 호환 디바이스를 등록 및 관리하고 동일한 방식으로 애플리케이션을 구축 및 배포합니다.

- Lenovo ThinkEdge® SE70 — Nvidia Jetson Xavier NX 기반

이 설명서의 콘텐츠 및 샘플 애플리케이션은 AWS Panorama 어플라이언스를 사용하여 개발되었습니다. 디바이스의 특정 하드웨어 및 소프트웨어 기능에 대한 자세한 내용은 제조업체의 설명서를 참조하십시오.

애플리케이션

애플리케이션은 AWS Panorama 어플라이언스에서 실행되어 비디오 스트림에서 컴퓨터 비전 작업을 수행합니다. Python 코드와 기계 학습 모델을 결합하여 컴퓨터 비전 애플리케이션을 구축하고 인터넷을 통해 AWS Panorama 어플라이언스에 배포할 수 있습니다. 애플리케이션은 비디오를 디스플레이로 전송하거나 AWS SDK를 사용하여 결과를 AWS 서비스에 전송할 수 있습니다.

애플리케이션을 구축하고 배포하려면 AWS Panorama Application CLI를 사용합니다. AWS Panorama Application CLI는 기본 애플리케이션 폴더 및 구성 파일을 생성하고, Docker로 컨테이너를 구축하고, 자산을 업로드하는 명령줄 도구입니다. 디바이스 하나에서 여러 애플리케이션을 실행할 수 있습니다.

자세한 내용은 AWS Panorama 애플리케이션 관리 단원을 참조하세요.

노드

애플리케이션은 입력, 출력, 모델 및 코드를 나타내는 노드라는 여러 구성 요소로 구성됩니다. 노드는 구성(입력 및 출력)만 될 수도 있고 아티팩트(모델 및 코드)를 포함할 수도 있습니다. 애플리케이션의 코드 노드는 사용자가 Amazon S3 액세스 포인트에 업로드하는 노드 패키지에 번들로 포함되며, AWS Panorama 어플라이언스가 해당 액세스 포인트에 액세스할 수 있습니다. 애플리케이션 매니페스트는 노드 간 연결을 정의하는 구성 파일입니다.

자세한 내용은 애플리케이션 노드 단원을 참조하세요.

모델

컴퓨터 비전 모델은 이미지를 처리하도록 훈련된 기계 학습 네트워크입니다. 컴퓨터 비전 모델은 분류, 감지, 분할 및 추적 등 다양한 작업을 수행할 수 있습니다. 컴퓨터 비전 모델은 이미지를 입력으로 받아 이미지 또는 이미지 내 물체에 대한 정보를 출력합니다.

AWS Panorama는 PyTorch, Apache MXNet, TensorFlow로 구축된 모델을 지원합니다. Amazon SageMaker를 사용하거나 개발 환경에서 모델을 구축할 수 있습니다. 자세한 내용은 ??? 단원을 참조하세요.

AWS Panorama 어플라이언스 설정

AWS Panorama 어플라이언스 또는 호환 디바이스 사용을 시작하려면 AWS Panorama 콘솔에 등록하고 소프트웨어를 업데이트하세요. 설정 프로세스 중에 물리적 어플라이언스를 나타내는 어플라이언스 리소스를 AWS Panorama에서 생성하고 USB 드라이브로 어플라이언스에 파일을 복사합니다. 어플라이언스는 이러한 인증서와 구성 파일을 사용하여 AWS Panorama 서비스에 연결합니다. 그런 다음 AWS Panorama 콘솔을 사용하여 어플라이언스의 소프트웨어를 업데이트하고 카메라를 등록합니다.

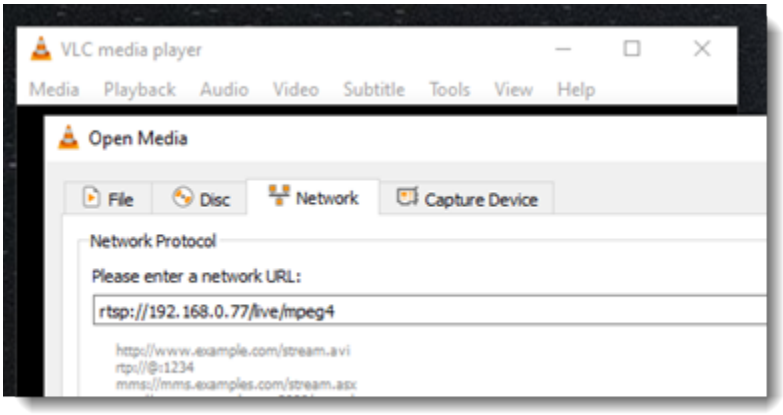
단원

필수 조건

이 자습서를 시작하려면 AWS Panorama 어플라이언스 또는 호환 디바이스와 다음 하드웨어가 필요합니다.

- 디스플레이 – 샘플 애플리케이션 출력을 볼 수 있는 HDMI 입력을 사용하는 디스플레이입니다.
- USB 드라이브(AWS Panorama 어플라이언스에 포함) — 구성 파일 및 인증서가 포함된 아카이브를 AWS Panorama 어플라이언스로 전송하기 위한 최소 1GB의 스토리지가 있는 FAT32 형식의 USB 3.0 플래시 메모리 드라이브입니다.
- 카메라 – RTSP 비디오 스트림을 출력하는 IP 카메라입니다.

카메라 제조업체에서 제공하는 도구와 지침을 사용하여 카메라의 IP 주소와 스트림 경로를 식별하십시오. VLC와 같은 비디오 플레이어를 사용하여 네트워크 미디어 소스로 열어 스트림 URL을 확인할 수 있습니다.



AWS Panorama 콘솔은 다른 AWS 서비스를 사용하여 애플리케이션 구성 요소를 조합하고, 권한을 관리하고, 설정을 확인합니다. 어플라이언스를 등록하고 샘플 애플리케이션을 배포하려면 다음 권한이 필요합니다.

- AWSPanoramaFullAccess – AWS Panorama, Amazon S3의AWS Panorama 액세스 포인트, AWS Secrets Manager의 어플라이언스 보안 인증 정보, Amazon CloudWatch의 어플라이언스 로그에 대한 전체 액세스를 제공합니다. AWS Panorama의 서비스 연결 역할을 생성할 수 있는 권한이 포함되어 있습니다.
- AWS Identity and Access Management(IAM) – 처음 실행할 때 AWS Panorama 서비스와 AWS Panorama 어플라이언스에서 사용하는 역할을 생성합니다.

IAM에서 역할을 생성할 권한이 없는 경우, 관리자가 AWS Panorama 콘솔을 열고 서비스 역할을 생성하라는 메시지를 수락하도록 요청하십시오.

AWS Panorama 어플라이언스 등록 및 구성

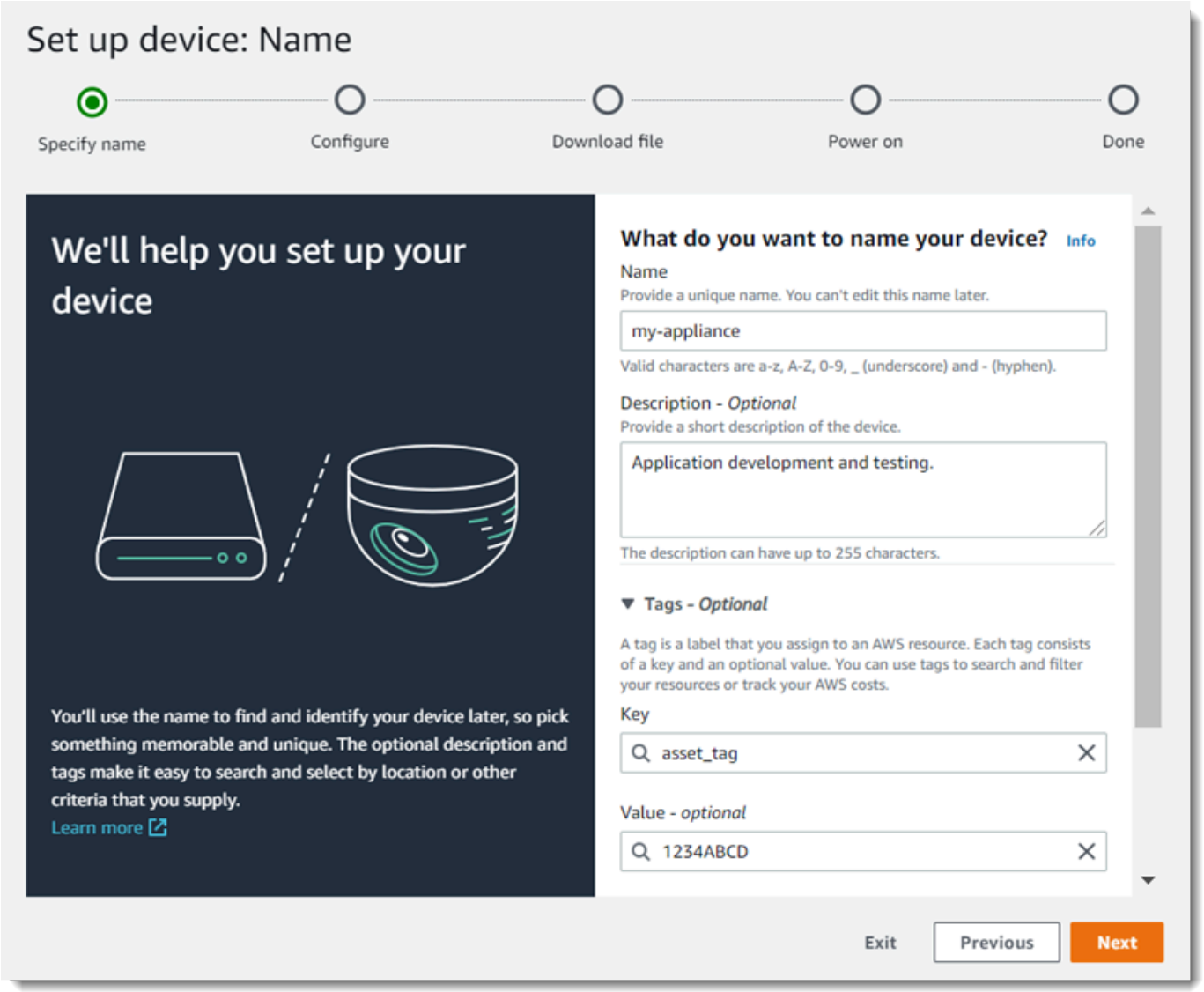
AWS Panorama 어플라이언스는 로컬 네트워크 연결을 통해 네트워크 지원 카메라에 연결하는 하드웨어 디바이스입니다. AWS Panorama Application SDK가 포함된 Linux 기반 운영 체제와 컴퓨터 비전 애플리케이션을 실행하기 위한 지원 소프트웨어를 사용합니다.

어플라이언스 관리 및 애플리케이션 배포를 위해 AWS에 연결하기 위해 어플라이언스는 디바이스 인증서를 사용합니다. AWS Panorama 콘솔을 사용하여 프로비저닝 인증서를 생성합니다. 어플라이언스는 이 임시 인증서를 사용하여 초기 설정을 완료하고 영구 디바이스 인증서를 다운로드합니다.

> **Important**
>
> 이 절차에서 생성한 프로비저닝 인증서는 5분 동안만 유효합니다. 이 시간 내에 등록 프로세스를 완료하지 않으면 다시 시작해야 합니다.

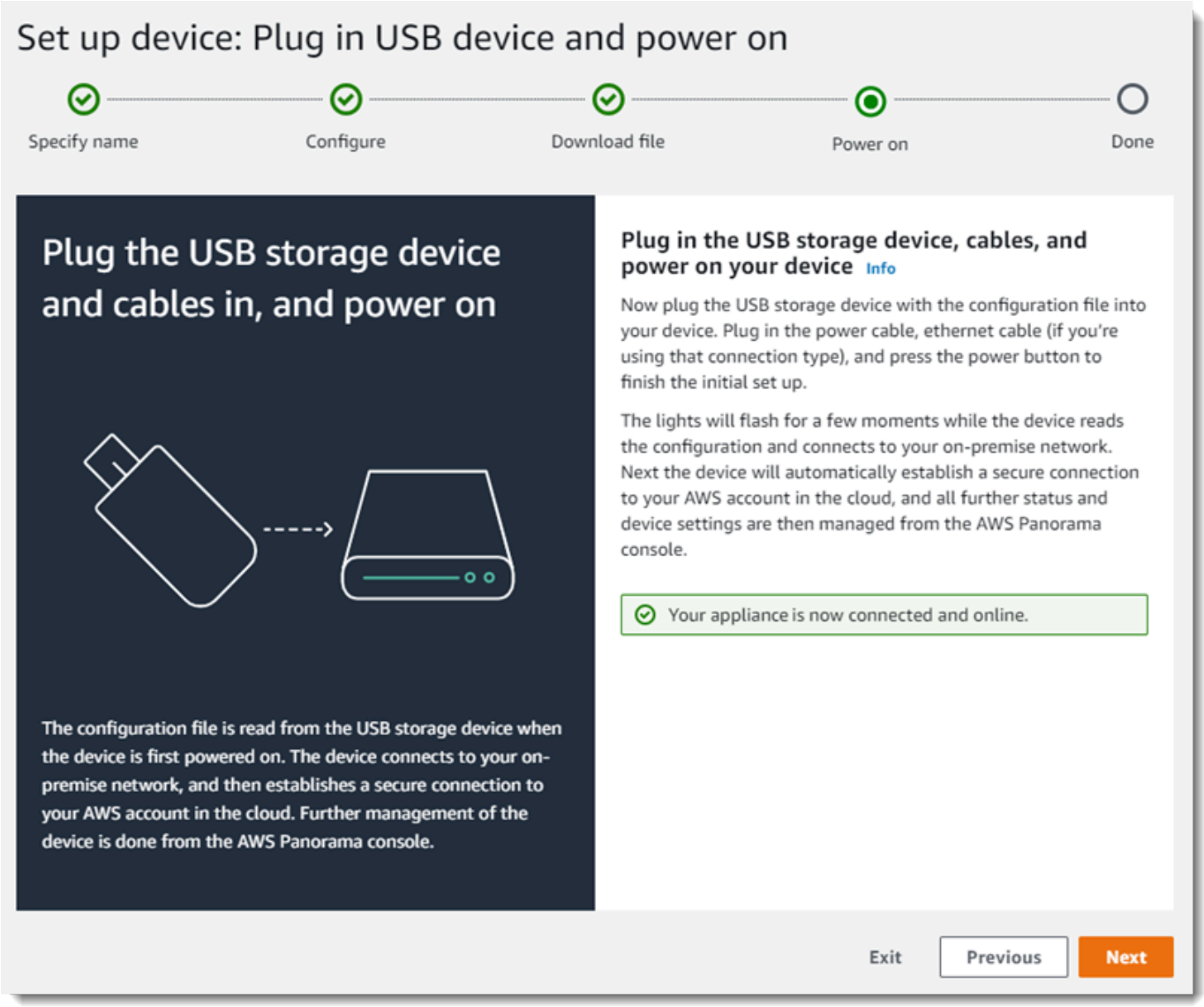
어플라이언스를 등록하려면

1. 컴퓨터에 USB 드라이브를 연결합니다. 네트워크 및 전원 케이블을 연결하여 어플라이언스를 준비합니다. 어플라이언스는 전원이 켜지고 USB 드라이브가 연결될 때까지 대기합니다.
2. AWS Panorama 콘솔 시작하기 페이지를 엽니다.
3. 디바이스 추가를 선택합니다.
4. 설정 시작을 선택합니다.
5. AWS Panorama의 어플라이언스를 나타내는 디바이스 리소스에 대한 이름과 설명을 입력합니다. Next(다음)를 선택합니다.



6. IP 주소, NTP 서버 또는 DNS 설정을 수동으로 할당해야 하는 경우 고급 네트워크 설정을 선택합니다. 그렇지 않은 경우 [Next]를 선택합니다.
7. 아카이브 다운로드를 선택합니다. 다음(Next)을 선택합니다.
8. 구성 아카이브를 USB 드라이브의 루트 디렉토리에 복사합니다.
9. USB 드라이브를 어플라이언스 전면의 HDMI 포트 옆에 있는 USB 3.0 포트에 연결합니다.

   USB 드라이브를 연결하면 어플라이언스가 구성 아카이브와 네트워크 구성 파일을 자체적으로 복사하고 AWS 클라우드에 연결합니다. 연결이 완료되면 어플라이언스의 상태 표시등이 녹색에서 파란색으로 바뀌고 다시 녹색으로 바뀝니다.
10. 계속하려면 다음을 선택합니다.



11. 완료(Done)를 선택합니다.

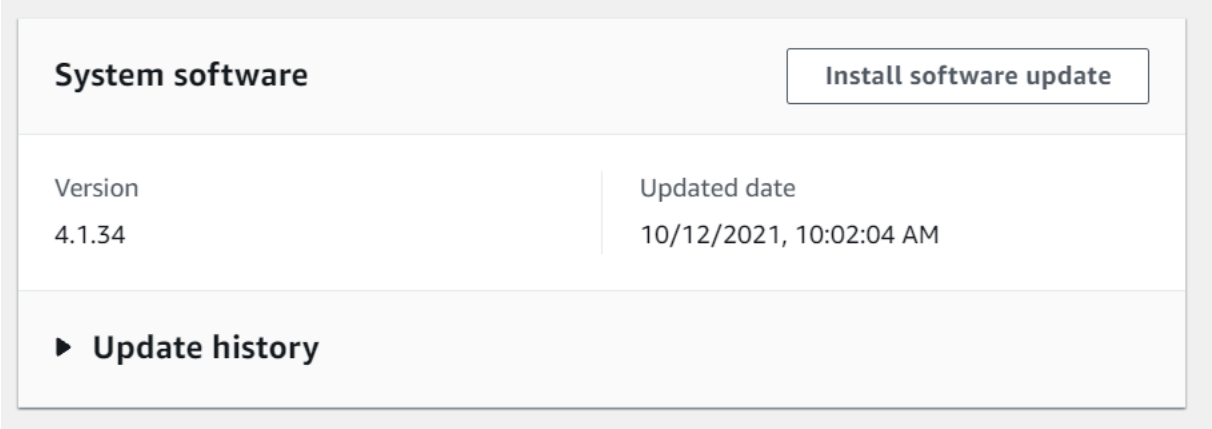
어플라이언스 소프트웨어 업그레이드

AWS Panorama 어플라이언스에는 Linux 운영 체제, AWS Panorama Application SDK, 지원되는 컴퓨터 비전 라이브러리 및 프레임워크를 비롯한 여러 소프트웨어 구성 요소가 있습니다. 어플라이언스에서 최신 기능과 애플리케이션을 사용할 수 있도록 하려면 설치 후 업데이트가 제공될 때마다 소프트웨어를 업그레이드하십시오.

어플라이언스 소프트웨어를 업데이트하려면

1. AWS Panorama 콘솔 디바이스 페이지를 엽니다.

2. 어플라이언스를 선택합니다.
3. 설정을 선택합니다.
4. 시스템 소프트웨어에서 소프트웨어 업데이트 설치를 선택합니다.



5. 새 버전을 선택한 다음 설치를 선택합니다.

> **Important**
> 계속하기 전에 어플라이언스에서 USB 드라이브를 제거하고 포맷하여 내용을 삭제하십시오. 구성 아카이브에는 민감한 데이터가 포함되며 자동으로 삭제되지 않습니다.

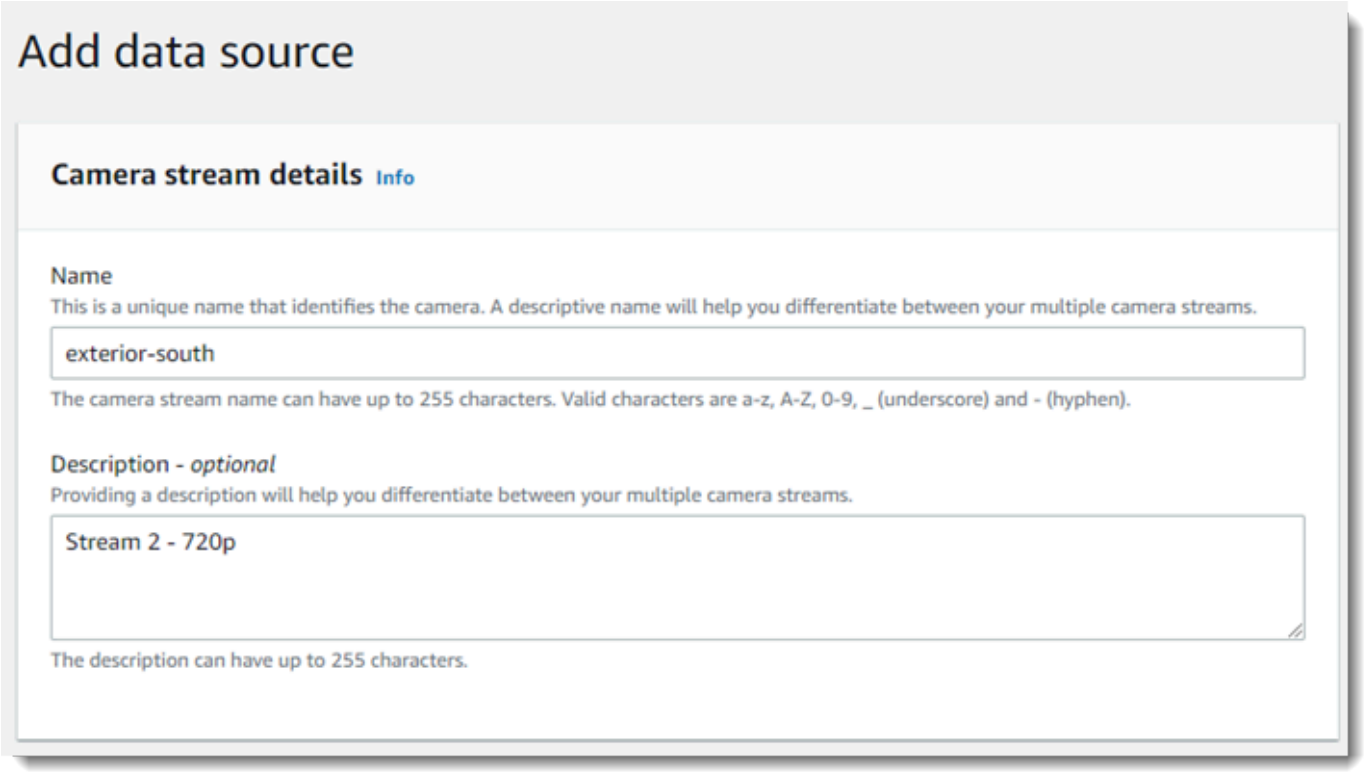
업그레이드 프로세스는 30분 이상 걸릴 수 있습니다. AWS Panorama 콘솔 또는 연결된 모니터에서 진행 상황을 모니터링할 수 있습니다. 프로세스가 완료되면 어플라이언스가 재부팅됩니다.

카메라 스트림 추가

다음으로, AWS Panorama 콘솔에 카메라 스트림을 등록하세요.

카메라 스트림을 등록하려면

1. AWS Panorama 콘솔 데이터 소스 페이지를 엽니다.
2. 데이터 소스 추가를 선택합니다.



3. 다음 설정을 구성합니다.

   - 이름 – 카메라 스트림의 이름입니다.
   - 설명 – 카메라, 위치 또는 기타 세부 정보에 대한 간략한 설명입니다.
   - RTSP URL – 카메라의 IP 주소와 스트림 경로를 지정하는 URL입니다. 예: `rtsp://192.168.0.77/live/mpeg4/`
   - 보안 인증 정보 - 카메라 스트림을 암호로 보호하는 경우 사용자 이름과 암호를 지정합니다.
4. 저장을 선택합니다.

AWS Panorama는 카메라의 보안 인증 정보를 AWS Secrets Manager에 안전하게 저장합니다. 여러 애플리케이션이 동일한 카메라 스트림을 동시에 처리할 수 있습니다.

다음 단계

설정 중에 오류가 발생한 경우 문제 해결을 참조하십시오.

샘플 애플리케이션을 배포하려면 다음 주제로 계속 진행하십시오.

AWS Panorama 샘플 애플리케이션 배포

AWS Panorama 어플라이언스 또는 호환 디바이스를 설정하고 소프트웨어를 업그레이드한 후 샘플 애플리케이션을 배포하십시오. 다음 섹션에서는 애플리케이션을 AWS Panorama 애플리케이션 CLI로 샘플 가져와서 AWS Panorama 콘솔에 배포합니다.

샘플 애플리케이션은 기계 학습 모델을 사용하여 네트워크 카메라의 비디오 프레임 단위로 개체를 분류합니다. AWS Panorama Application SDK를 사용하여 모델을 로드하고, 이미지를 가져오고, 모델을 실행합니다. 그런 다음 애플리케이션은 결과를 원본 비디오 위에 오버레이하고 연결된 디스플레이에 출력합니다.

소매 환경에서는 유동인구 패턴을 분석하여 트래픽 수준을 예측할 수 있습니다. 분석 내용을 다른 데이터와 결합하여 휴일 및 기타 행사에 필요한 인력 충원에 대한 계획을 세우고, 광고 및 판촉의 효과를 측정하거나, 디스플레이 배치 및 재고 관리를 최적화할 수 있습니다.

단원

필수 조건

이 자습서의 절차를 따르려면 명령을 실행할 쉘 또는 명령줄 터미널이 필요합니다. 코드 목록에서 명령어 앞에는 프롬프트 기호($)와 현재 디렉토리 이름(해당하는 경우)이 표시됩니다.

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

긴 명령의 경우 이스케이프 문자(\)를 사용하여 명령을 여러 줄로 분할합니다.

Linux 및 macOS는 선호 쉘과 패키지 관리자를 사용합니다. Windows 10에서 Linux용 Windows Subsystem을 설치하여 Ubuntu와 Bash의 Windows 통합 버전을 가져옵니다. Windows에서 개발 환경을 설정하는 데 도움이 필요하면 Windows에서 개발 환경 설정를 참조하십시오.

Python의 패키지 관리자인 pip를 사용하여 AWS Panorama 애플리케이션을 개발하고 도구를 설치할 수 있습니다. 아직 Python 사용 중이라면 최신 버전을 설치하세요. Python 3을 사용하고 pip는 사용하지 않는 경우 운영 체제의 패키지 관리자를 사용하여 pip를 설치하거나 pip와 함께 제공되는 새 버전의 Python을 설치하십시오.

이 자습서에서는 Docker를 사용하여 애플리케이션 코드를 실행하는 컨테이너를 빌드합니다. Docker 웹 사이트에서 Docker 설치: Docker 다운로드

이 자습서에서는 AWS Panorama Application CLI를 사용하여 샘플 애플리케이션을 가져오고, 패키지를 빌드하고, 아티팩트를 업로드합니다. AWS Panorama 애플리케이션 CLI는 AWS Command Line Interface(AWS CLI)를 사용하여 서비스 API 작업을 호출합니다. 이미 AWS CLI를 사용 중이라면 최신 버전으로 업그레이드하세요. AWS Panorama 애플리케이션 CLI와 AWS CLI을 설치하려면 `pip`를 사용합니다.

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

샘플 애플리케이션을 다운로드하여 작업 영역에 추출하십시오.

- 샘플 애플리케이션 — aws-panorama-sample.zip

샘플 애플리케이션 가져오기

계정에서 사용할 샘플 애플리케이션을 가져오려면 AWS Panorama Application CLI를 사용하십시오. 애플리케이션의 폴더와 매니페스트에는 자리 표시자 계정 번호에 대한 참조가 포함되어 있습니다. 계정 번호로 업데이트하려면 `panorama-cli import-application` 명령을 실행하세요.

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

`packages` 디렉토리의 `SAMPLE_CODE` 패키지에는 애플리케이션 기반 이미지 `panorama-application`를 사용하는 Dockerfile을 비롯한 애플리케이션의 코드와 구성이 들어 있습니다. 어플라이언스에서 실행되는 애플리케이션 컨테이너를 빌드하려면 `panorama-cli build-container` 명령을 사용하세요.

```
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 Application CLI의 마지막 단계는 애플리케이션의 코드와 모델 노드를 등록하고 서비스에서 제공하는 Amazon S3 액세스 포인트에 자산을 업로드하는 것입니다. 자산에는 코드의 컨테이너 이미지, 모델 및 각각에 대한 설명자 파일이 포함됩니다. 노드를 등록하고 에셋을 업로드하려면 `panorama-cli package-application` 명령을 실행합니다.

```
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
```

애플리케이션 배포

AWS Panorama 콘솔을 사용하여 애플리케이션을 어플라이언스에 배포하십시오.

애플리케이션을 배포하려면

1. AWS Panorama 콘솔 배포 애플리케이션 페이지를 엽니다.
2. 애플리케이션 배포를 선택합니다.
3. 애플리케이션 매니페스트의 콘텐츠 `graphs/aws-panorama-sample/graph.json`를 텍스트 편집기에 붙여넣습니다. 다음(Next)을 선택합니다.
4. 애플리케이션 이름에 `aws-panorama-sample`을 입력합니다.
5. 배포로 진행을 선택합니다.
6. 배포 시작을 선택합니다.
7. 역할을 선택하지 않고 다음을 선택합니다.
8. 디바이스 선택을 선택한 다음 어플라이언스를 선택합니다. 다음(Next)을 선택합니다.
9. 데이터 소스 선택 단계에서 입력 보기를 선택하고 카메라 스트림을 데이터 소스로 추가합니다. 다음(Next)을 선택합니다.
10. 구성 단계에서 다음을 선택합니다.
11. 배포를 선택한 다음 완료를 선택합니다.
12. 배포된 애플리케이션 목록에서 aws-panorama-sample을 선택합니다.

업데이트를 위해 이 페이지를 새로 고치거나 다음 스크립트를 사용하여 명령줄에서 배포를 모니터링하세요.

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": {}
    }
]
...
```


애플리케이션이 실행되지 않는 경우 Amazon CloudWatch Logs에서 애플리케이션 및 디바이스 로그를 확인하십시오.

출력 검토

배포가 완료되면 애플리케이션이 비디오 스트림 처리를 시작하고 로그를 CloudWatch로 보냅니다.

CloudWatch Logs에서 로그를 보려면

1. CloudWatch 로그 콘솔의 로그 그룹 페이지를 엽니다.
2. 다음 그룹에서 AWS Panorama 애플리케이션 및 어플라이언스 로그를 확인할 수 있습니다.

   - 디바이스 로그 – `/aws/panorama/devices/device-id`
   - 애플리케이션 로그 – `/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
```

애플리케이션의 비디오 출력을 보려면 HDMI 케이블을 사용하여 어플라이언스를 모니터에 연결합니다. 기본적으로 애플리케이션은 신뢰도가 20% 이상인 모든 분류 결과를 표시합니다.

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",
...
```

샘플 모델에는 많은 동물, 음식, 일반 개체를 포함한 1,000개의 클래스가 있습니다. 카메라로 키보드나 커피잔을 가리켜 보세요.

단순화를 위해 샘플 애플리케이션은 간단한 분류 모델을 사용합니다. 모델은 각 클래스에 대한 확률을 통해 단일 배열을 출력합니다. 실제 애플리케이션에서는 다차원 출력이 있는 개체 감지 모델을 더 자주 사용합니다. 더 복잡한 모델을 사용하는 샘플 애플리케이션에 대해서는 샘플 애플리케이션, 스크립트, 템플릿를 참조하십시오.

SDK for Python 사용

샘플 애플리케이션은 AWS SDK for Python (Boto)을 사용하여 Amazon CloudWatch에 지표를 전송합니다. 이 기능을 활성화하려면 애플리케이션에 지표 전송 권한을 부여하는 역할을 생성하고 역할이 연결된 애플리케이션을 재배포하십시오.

샘플 애플리케이션에는 필요한 권한을 가진 역할을 생성하는 AWS CloudFormation 템플릿이 포함되어 있습니다. 역할을 만들려면 `aws cloudformation deploy` 명령을 사용하세요.

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

애플리케이션을 재배포하려면

1. AWS Panorama 콘솔 배포 애플리케이션 페이지를 엽니다.
2. 애플리케이션을 선택합니다.
3. 바꾸기를 선택합니다.
4. 애플리케이션 배포 단계를 완료합니다. IAM 역할 지정에서 사용자가 생성한 역할을 선택합니다. 이름이 `aws-panorama-sample-runtime`로 시작하는 함수입니다.
5. 배포가 완료되면 CloudWatch 콘솔을 열고 `AWSPanoramaApplication` 네임스페이스에서 지표를 확인합니다. 150프레임마다 애플리케이션은 프레임 처리 및 추론 시간에 대한 지표를 기록하고 업로드합니다.

정리

샘플 애플리케이션 작업을 완료한 경우 AWS Panorama 콘솔을 사용하여 어플라이언스에서 해당 애플리케이션을 제거할 수 있습니다.

어플라이언스에서 애플리케이션을 제거하려면

1. AWS Panorama 콘솔 배포 애플리케이션 페이지를 엽니다.
2. 애플리케이션을 선택합니다.
3. 디바이스에서 삭제를 선택합니다.

다음 단계

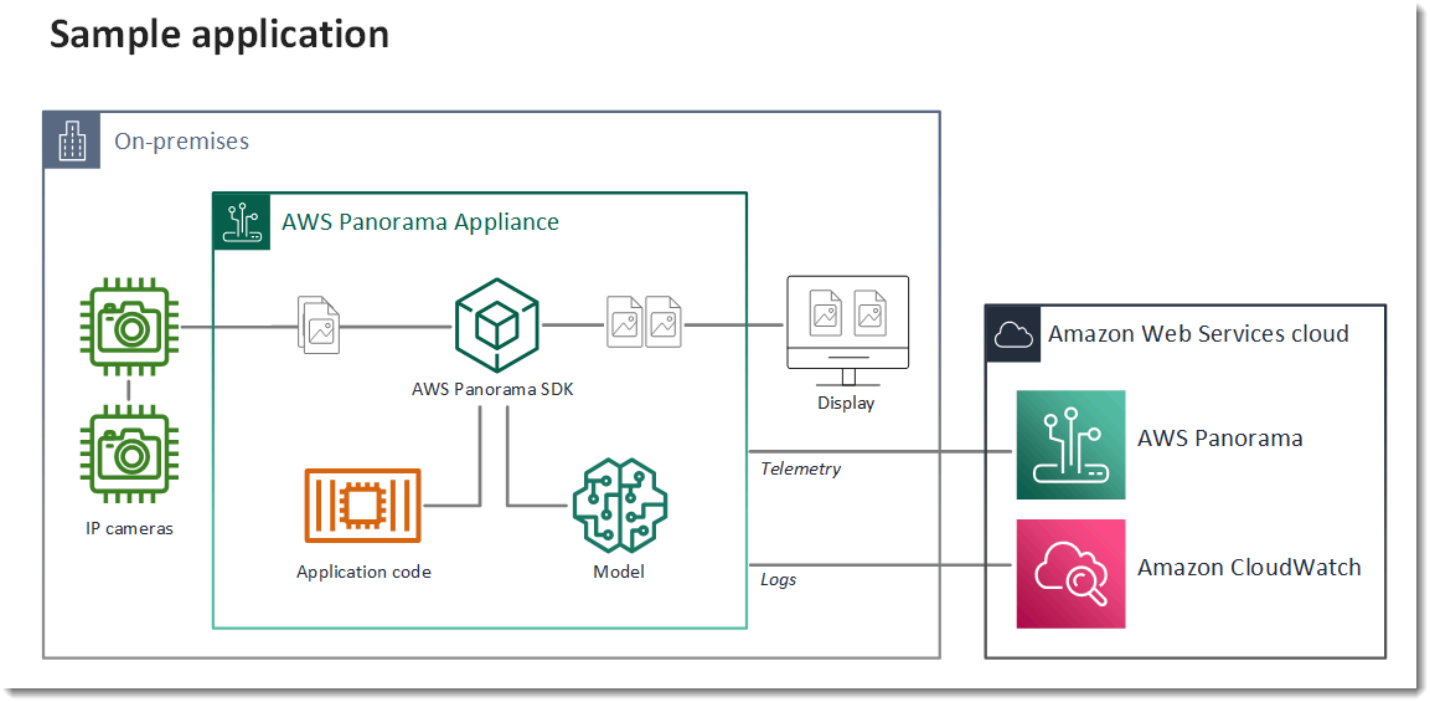
샘플 애플리케이션을 배포하거나 실행하는 동안 오류가 발생한 경우 문제 해결을 참조하십시오.

샘플 애플리케이션의 기능 및 구현에 대해 자세히 알아보려면 다음 주제를 계속 진행하십시오.

AWS Panorama 애플리케이션 개발

샘플 애플리케이션은 AWS Panorama 애플리케이션 구조에 대해 배우는 데 사용할 수 있으며 애플리케이션을 자체 애플리케이션의 시작점으로 사용할 수 있습니다.

다음 다이어그램은 AWS Panorama 어플라이언스에서 실행되는 애플리케이션의 주요 구성 요소를 보여줍니다. 애플리케이션 코드는 AWS Panorama Application SDK를 사용하여 이미지를 가져오고 모델과 상호 작용하지만, 모델에 직접 액세스할 수는 없습니다. 애플리케이션은 연결된 디스플레이에 비디오를 출력하지만 로컬 네트워크 외부로 이미지 데이터를 전송하지는 않습니다.



이 예제에서 애플리케이션은 AWS Panorama Application SDK를 사용하여 카메라로부터 비디오 프레임을 가져오고, 비디오 데이터를 전처리하고, 객체를 감지하는 컴퓨터 비전 모델로 데이터를 전송합니다. 애플리케이션은 어플라이언스에 연결된 HDMI 디스플레이에 결과를 표시합니다.

단원

- 애플리케이션 매니페스트
- 샘플 애플리케이션을 사용한 빌드
- 컴퓨터 비전 모델 변경
- 이미지 사전 처리
- SDK for Python을 통한 지표 업로드
- 다음 단계

애플리케이션 매니페스트

애플리케이션 매니페스트는 graphs 폴더의 graph.json이라는 이름의 파일입니다. 매니페스트는 패키지, 노드, 엣지와 같은 애플리케이션의 구성 요소를 정의합니다.

패키지는 애플리케이션 코드, 모델, 카메라 및 디스플레이를 위한 코드, 구성 및 바이너리 파일입니다. 샘플 애플리케이션은 4개의 패키지를 사용합니다.

Example **graphs/aws-panorama-sample/graph.json** – 패키지

```
        "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"
            }
        ],
```

처음 두 패키지는 애플리케이션 내의 packages 디렉토리에서 정의됩니다. 여기에는 이 애플리케이션과 관련된 코드와 모델이 들어 있습니다. 다음 두 패키지는 AWS Panorama 서비스에서 제공하는 일반 카메라 및 디스플레이 패키지입니다. abstract_rtsp_media_source 패키지는 배포 중에 재정의할 수 있는 카메라의 자리 표시자입니다. hdmi_data_sink 패키지는 장치의 HDMI 출력 커넥터를 나타냅니다.

노드는 패키지에 대한 인터페이스일 뿐만 아니라 배포 시 재정의하는 기본값을 가질 수 있는 비패키지 파라미터입니다. 코드 및 모델 패키지는 입력과 출력을 지정하는 package.json 파일의 인터페이스를 정의합니다. 인터페이스는 비디오 스트림이거나 실수, 부울, 문자열과 같은 기본 데이터 유형일 수 있습니다.

예를 들어, code_node 노드는 SAMPLE_CODE 패키지의 인터페이스를 참조합니다.

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

이 인터페이스는 패키지 구성 파일 package.json에 정의되어 있습니다. 이 인터페이스는 패키지가 비즈니스 로직이며 video_in이라는 비디오 스트림과 threshold라는 부동 소수점 숫자를 입력 값으로 받는다고 지정합니다. 또한 이 인터페이스는 디스플레이에 비디오를 출력하기 위해 코드에 video_out이라는 이름의 비디오 스트림 버퍼가 필요함을 지정합니다.

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"
```

```
                    }
                ]
            }
        ]
    }
}
```

애플리케이션 매니페스트에서 camera_node 노드는 카메라의 비디오 스트림을 나타냅니다. 여기에는 애플리케이션을 배포할 때 콘솔에 나타나는 데코레이터가 포함되어 있어 카메라 스트림을 선택하라는 메시지가 표시됩니다.

Example **graphs/aws-panorama-sample/graph.json** – 카메라 노드

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

파라미터 노드 threshold_param는 애플리케이션 코드에서 사용하는 신뢰도 임계값 파라미터를 정의합니다. 기본값은 60이며 배포 중에 재정의할 수 있습니다.

Example **graphs/aws-panorama-sample/graph.json**— 파라미터 노드

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

애플리케이션 매니페스트의 마지막 섹션인 edges는 노드 간 연결을 만듭니다. 카메라의 비디오 스트림과 임계값 파라미터는 코드 노드의 입력에 연결되고 코드 노드의 비디오 출력은 디스플레이에 연결됩니다.

Example **graphs/aws-panorama-sample/graph.json** – 엣지

```
    "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"
        }
    ]
```

샘플 애플리케이션을 사용한 빌드

샘플 애플리케이션을 자체 애플리케이션의 시작점으로 사용할 수 있습니다.

각 패키지의 이름은 계정에서 고유해야 합니다. 계정 내 다른 사용자와 함께 code 또는 model 같은 일반 패키지 이름을 사용하는 경우 배포할 때 패키지 버전이 잘못될 수 있습니다. 코드 패키지의 이름을 애플리케이션을 나타내는 이름으로 변경하십시오.

코드 패키지의 이름을 변경하려면

1. 패키지 폴더 이름 변경: packages/123456789012-*SAMPLE_CODE*-1.0/.
2. 다음 위치에서 패키지 이름을 업데이트하십시오.

   - 애플리케이션 매니페스트 – graphs/aws-panorama-sample/graph.json
   - 패키지 구성 – packages/123456789012-SAMPLE_CODE-1.0/package.json
   - 빌드 스크립트 – 3-build-container.sh

애플리케이션의 코드를 업데이트하려면

1. packages/123456789012-SAMPLE_CODE-1.0/src/application.py에서 애플리케이션 코드를 수정합니다.
2. 컨테이너를 빌드하려면 3-build-container.sh를 실행하세요.

```
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
```

CLI는 assets 폴더에서 이전 컨테이너 자산을 자동으로 삭제하고 패키지 구성을 업데이트합니다.

3. 패키지를 업로드하려면 4-package-application.py를 실행하세요.
4. AWS Panorama 콘솔 배포 애플리케이션 페이지를 엽니다.
5. 애플리케이션을 선택합니다.
6. 바꾸기를 선택합니다.

7. 애플리케이션 배포 단계를 완료합니다. 필요한 경우 애플리케이션 매니페스트, 카메라 스트림 또는 파라미터를 변경할 수 있습니다.

컴퓨터 비전 모델 변경

샘플 애플리케이션에는 컴퓨터 비전 모델이 포함되어 있습니다. 자체 모델을 사용하려면 모델 노드의 구성을 수정하고 AWS Panorama Application CLI를 사용하여 모델을 자산으로 가져오십시오.

다음 예시는 MXnet SSD ResNet50 모델을 사용하며, 이 모델은 이 설명서의 GitHub 리포지토리 ssd_512_resnet50_v1_voc.tar.gz에서 다운로드할 수 있습니다.

샘플 애플리케이션의 모델을 변경하려면

1. 모델에 맞게 패키지 폴더의 이름을 변경합니다. 예를 들어 `packages/123456789012-SSD_512_RESNET50_V1_VOC-1.0/`로 변경합니다.
2. 다음 위치에서 패키지 이름을 업데이트하십시오.

   - 애플리케이션 매니페스트 – `graphs/aws-panorama-sample/graph.json`
   - 패키지 구성 – `packages/123456789012-SSD_512_RESNET50_V1_VOC-1.0/package.json`
3. 패키지 구성 파일(`package.json`)에서. `assets` 값을 빈 배열로 변경합니다.

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

4. 패키지 설명자 파일(`descriptor.json`)을 엽니다. `framework` 및 `shape` 값을 모델에 맞게 업데이트하십시오.

```
{
    "mlModelDescriptor": {
        "envelopeVersion": "2021-01-01",
        "framework": "MXNET",
        "inputs": [
            {
```

```
                "name": "data",
                "shape": [ 1, 3, 512, 512 ]
            }
        ]
    }
}
```

모양 값 `1,3,512,512`는 모델이 입력으로 받는 이미지 수(1), 각 이미지의 채널 수(3--빨간색, 녹색, 파란색), 이미지 크기(512 x 512)를 나타냅니다. 배열의 값과 순서는 모델마다 다릅니다.

5. AWS Panorama Application CLI를 사용하여 모델을 가져옵니다. AWS Panorama Application CLI는 모델 및 설명자 파일을 고유한 이름이 있는 `assets` 폴더에 복사하고 패키지 구성을 업데이트합니다.

```
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. 모델을 업로드하려면 `panorama-cli package-application`을 실행하세요.

```
$ 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. 애플리케이션 코드를 업데이트합니다. 대부분의 코드는 재사용할 수 있습니다. 모델 응답과 관련된 코드는 `process_results` 메서드에 있습니다.

```
    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)
```

모델에 따라 `preprocess` 메서드를 업데이트해야 할 수도 있습니다.

이미지 사전 처리

애플리케이션은 이미지를 모델로 보내기 전에 이미지의 크기를 조정하고 색상 데이터를 정규화하여 추론을 준비합니다. 애플리케이션이 사용하는 모델에는 첫 번째 레이어의 입력 개수와 일치하도록 세 개의 색상 채널이 있는 224 x 224 픽셀 이미지가 필요합니다. 애플리케이션은 각 색상 값을 0에서 1 사

이의 숫자로 변환하고, 해당 색상의 평균값을 빼고, 표준 편차로 나누어 조정합니다. 마지막으로 색상 채널을 결합하여 모델이 처리할 수 있는 NumPy 배열로 변환합니다.

Example application.py – 사전 처리

```
    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)
```

이 프로세스는 0을 중심으로 한 예측 가능한 범위의 모델 값을 제공합니다. 이는 표준 접근 방식이지만 모델마다 다를 수 있는 교육 데이터 세트의 이미지에 적용되는 전처리와 일치합니다.

SDK for Python을 통한 지표 업로드

샘플 애플리케이션은 SDK for Python을 사용하여 Amazon CloudWatch에 지표를 업로드합니다.

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.")
```

배포 중에 할당한 런타임 역할에서 권한을 얻습니다. 역할은 `aws-panorama-sample.yml` AWS CloudFormation 템플릿에 정의되어 있습니다.

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/
```

샘플 애플리케이션은 pip를 사용하여 SDK for Python 및 기타 종속 항목을 설치합니다. 애플리케이션 컨테이너를 빌드하는 경우 `Dockerfile`은 명령을 실행하여 기본 이미지와 함께 제공되는 항목 위에 라이브러리를 설치합니다.

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
```

애플리케이션 코드에서 AWS SDK를 사용하려면 먼저 템플릿을 수정하여 애플리케이션이 사용하는 모든 API 작업에 대한 권한을 추가해야 합니다. 변경할 때마다 `1-create-role.sh`를 실행하여 AWS CloudFormation 스택을 업데이트합니다. 그런 다음 애플리케이션 코드에 변경 내용을 배포하십시오.

기존 리소스를 수정하거나 사용하는 작업의 경우 별도의 문장에 대상 `Resource`의 이름이나 패턴을 지정하여 이 정책의 적용 범위를 최소화하는 것이 가장 좋습니다. 각 서비스에서 지원하는 작업 및 리소스에 대한 자세한 내용은 서비스 승인 참조의 작업, 리소스 및 조건 키를 참조하세요.

다음 단계

AWS Panorama 애플리케이션 CLI를 사용하여 애플리케이션을 구축하고 처음부터 패키지를 생성하는 방법에 대한 지침은 CLI의 README를 참조하십시오.

- github.com/aws/aws-panorama-cli

배포 전에 애플리케이션 코드를 검증하는 데 사용할 수 있는 샘플 코드와 테스트 유틸리티에 대한 자세한 내용은 AWS Panorama 샘플 리포지토리를 참조하십시오.

- github.com/aws-samples/aws-panorama-samples

지원되는 컴퓨터 비전 모델 및 카메라

AWS Panorama는 PyTorch, Apache MXNet, TensorFlow로 구축된 모델을 지원합니다. 애플리케이션을 배포할 때 AWS Panorama는 SageMaker Neo에서 모델을 컴파일합니다. SageMaker Neo와 호환되는 레이어를 사용하는 경우, Amazon SageMaker 또는 개발 환경에서 모델을 구축할 수 있습니다.

비디오를 처리하고 모델로 전송할 이미지를 가져오기 위해 AWS Panorama 어플라이언스는 RTSP 프로토콜을 사용하여 H.264로 인코딩된 비디오 스트림에 연결합니다. AWS Panorama는 다양한 일반 카메라의 호환성을 테스트합니다.

단원

- 지원되는 모델
- 지원되는 카메라

지원되는 모델

AWS Panorama용 애플리케이션을 빌드할 경우, 사용자는 애플리케이션이 컴퓨터 비전에 사용하는 기계 학습 모델을 제공합니다. 모델 프레임워크에서 제공하는 사전 빌드 및 사전 학습된 모델, 샘플 모델 또는 직접 구축하고 학습한 모델을 사용할 수 있습니다.

> **Note**
>
> AWS Panorama로 테스트한 사전 빌드 모델 목록은 모델 호환성을 참조하십시오.

애플리케이션을 배포할 때 AWS Panorama는 SageMaker Neo 컴파일러를 사용하여 컴퓨터 비전 모델을 컴파일합니다. SageMaker Neo는 대상 플랫폼에서 효율적으로 실행되도록 모델을 최적화하는 컴파일러로, Amazon Elastic Compute Cloud(Amazon EC2)의 인스턴스 또는 AWS Panorama 어플라이언스와 같은 엣지 디바이스가 될 수 있습니다.

AWS Panorama는 SageMaker Neo에서 엣지 디바이스용으로 지원하는 PyTorch, Apache MXnet 및 TensorFlow 버전을 지원합니다. 자체 모델을 구축하는 경우, SageMaker Neo 출시 정보에 나열된 프레임워크 버전을 사용할 수 있습니다. SageMaker에서는 내장된 이미지 분류 알고리즘을 사용할 수 있습니다.

AWS Panorama에서 모델을 사용하는 방법은 컴퓨터 비전 모델 단원을 참조하세요.

지원되는 카메라

AWS Panorama 어플라이언스는 로컬 네트워크를 통해 RTSP를 출력하는 카메라의 H.264 비디오 스트림을 지원합니다. 2메가픽셀을 초과하는 카메라 스트림의 경우 어플라이언스는 이미지를 1920x1080 픽셀 또는 스트림의 가로 세로 비율을 유지하는 동등한 크기로 축소합니다.

다음 카메라 모델은 AWS Panorama 어플라이언스와의 호환성 테스트를 완료했습니다.

- Axis – M3057-PLVE, M3058-PLVE, P1448-LE, P3225-LV Mk II
- LaView – LV-PB3040W
- Vivotek – IB9360-H
- Amcrest – IP2M-841B
- Anpviz – IPC-B850W-S-3X, IPC-D250W-S
- WGCC – Dome PoE 4MP ONVIF

어플라이언스의 하드웨어 사양은 AWS Panorama 어플라이언스 사양를 참조하십시오.

AWS Panorama 어플라이언스 사양

AWS Panorama 어플라이언스의 하드웨어 사양은 다음과 같습니다. 다른 호환 디바이스에 대해서는 제조업체 설명서를 참조하십시오.

구성 요소	사양
프로세서 및 GPU	Nvidia Jetson AGX Xavier(32GB RAM 포함)
이더넷	2x 1000 Base-T(기가바이트)
USB	USB 2.0 1개 및 USB 3.0 타입-A 1개(암형)
HDMI 출력	2.0a
치수	7.75” x 9.6” x 1.6”(197mm x 243mm x 40mm)
무게	3.7파운드(1.7kg)
전원 공급	100V-240V 50-60Hz AC 65W
전원 입력	IEC 60320 C6(3핀) 리셉터클
방진 및 방수	IP-62
EMI/EMC 규제 준수	FCC Part-15(미국)
열 접촉 제한	IEC-62368
작동 온도	-20°C~60°C
작동 습도	0%~95% RH
보관 온도	-20°C~85°C
보관 습도	저온에서는 제어되지 않습니다. 고온에서 90% RH
냉각	강제 공기 열 추출(팬)
장착 옵션	랙마운트 또는 독립형

구성 요소	사양
전원 코드	6피트(1.8미터)
전원 제어	푸시 버튼
Reset	모멘터리 스위치
상태 및 네트워크 LED	프로그래밍형 3색 RGB LED

어플라이언스에 Wi-Fi, Bluetooth 및 SD 카드 스토리지가 있지만 사용할 수 없습니다.

AWS Panorama 어플라이언스에 서버 랙에 장착하기 위한 나사 2개가 포함되어 있습니다. 19인치 랙에 어플라이언스 두 개를 나란히 장착할 수 있습니다.

Service Quotas

AWS Panorama는 계정에서 생성한 리소스와 배포하는 애플리케이션에 할당량을 적용합니다. 여러 AWS 리전에서 AWS Panorama를 사용하는 경우, 각 리전에 할당량이 별도로 적용됩니다. AWS Panorama 할당량은 조절할 수 없습니다.

AWS Panorama의 리소스에는 디바이스, 애플리케이션 노드 패키지 및 애플리케이션 인스턴스가 포함됩니다.

- 디바이스 – 지역당 등록된 어플라이언스 최대 50개.
- 노드 패키지 – 지역당 50개 패키지, 패키지당 최대 20개 버전.
- 애플리케이션 인스턴스 – 디바이스당 최대 10개 애플리케이션. 각 애플리케이션은 최대 8개의 카메라 스트림을 모니터링할 수 있습니다. 배포는 각 디바이스에 대해 하루 200개로 제한됩니다.

AWS Panorama 서비스와 함께 AWS Panorama Application CLI, AWS Command Line Interface 또는 AWS SDK를 사용하는 경우, API 호출 수에 할당량이 적용됩니다. 초당 총 5개까지 요청할 수 있습니다. 리소스를 만들거나 수정하는 API 작업의 하위 집합에는 초당 요청 1개라는 추가 제한이 적용됩니다.

전체 할당량 목록을 보려면 Service Quotas 콘솔을 방문하거나 Amazon Web Services 일반 참조에서 AWS Panorama 엔드포인트 및 할당량을 참조하십시오.

AWS Panorama 권한

AWS Identity and Access Management(IAM)을 사용하여 AWS Panorama 서비스 및 리소스(어플라이언스 및 애플리케이션 등)에 대한 액세스를 관리할 수 있습니다. 계정에서 AWS Panorama를 사용하는 사용자의 경우 IAM 역할에 적용할 수 있는 권한 정책에서 권한을 관리합니다. 애플리케이션에 대한 권한을 관리하려면 역할을 생성하여 애플리케이션에 할당합니다.

계정의 사용자에 대한 권한을 관리하려면 AWS Panorama에서 제공하는 관리형 정책을 사용하거나 직접 작성하십시오. 애플리케이션 및 어플라이언스 로그를 가져오고, 메트릭을 보고, 애플리케이션에 역할을 할당하려면 다른 AWS 서비스에 대한 권한이 필요합니다.

AWS Panorama 어플라이언스에는 AWS 서비스 및 리소스에 대한 액세스 권한을 부여하는 역할도 있습니다. 어플라이언스의 역할은 AWS Panorama 서비스가 사용자를 대신하여 다른 서비스에 액세스하는 데 사용하는 서비스 역할 중 하나입니다.

애플리케이션 역할은 애플리케이션에 대해 생성하는 별도의 서비스 역할로, AWS SDK for Python (Boto)을 통해 AWS 서비스를 사용할 수 있는 권한을 부여합니다. 애플리케이션 역할을 생성하려면 관리자 권한이나 관리자의 도움이 필요합니다.

작업이 영향을 주는 리소스별로 또는 경우에 따라 추가 조건을 적용하여 사용자 권한을 제한할 수 있습니다. 예를 들어 애플리케이션의 Amazon 리소스 이름(ARN) 패턴을 지정하여 생성하는 애플리케이션 이름에 사용자가 사용자 이름을 포함시키도록 할 수 있습니다. 각 작업에서 지원되는 리소스 및 조건에 대해서는 서비스 승인 참조에서 AWS Panorama의 작업, 리소스 및 조건 키를 참조하십시오.

자세한 내용은 IAM 사용 설명서의 IAM이란?을 참조하십시오.

주제

- AWS Panorama에 대한 보안 인증 기반 IAM 정책
- AWS Panorama 서비스 역할 및 교차 서비스 리소스
- 애플리케이션에 권한 부여

AWS Panorama에 대한 보안 인증 기반 IAM 정책

계정의 사용자에게 AWS Panorama에 대한 액세스 권한을 부여하려면 AWS Identity and Access Management(IAM)에서 보안 인증 기반 정책을 사용하세요. 보안 인증 기반 정책은 사용자와 연결된 IAM 역할에 적용합니다. 다른 계정의 사용자에게 내 계정의 역할을 수행할 수 있는 권한 및 AWS Panorama 리소스에 대한 액세스 권한을 부여할 수도 있습니다.

AWS Panorama는 AWS Panorama API 작업에 대해 액세스 권한을 부여하는 관리형 정책을 제공합니다. 경우에 따라 이 정책은 AWS Panorama 리소스를 개발하고 관리하는 데 사용되는 다른 서비스에 대한 액세스 권한도 부여합니다. AWS Panorama는 필요에 따라 관리형 정책을 업데이트하여 정책 릴리스 시 사용자가 새 기능에 대한 액세스 권한을 가질 수 있도록 보장합니다.

- AWSPanoramaFullAccess – AWS Panorama, Amazon S3의AWS Panorama 액세스 포인트, AWS Secrets Manager의 어플라이언스 보안 인증 정보, Amazon CloudWatch의 어플라이언스 로그에 대한 전체 액세스를 제공합니다. AWS Panorama의 서비스 연결 역할을 생성할 수 있는 권한이 포함되어 있습니다. 정책 보기

`AWSPanoramaFullAccess` 정책을 통해 AWS Panorama 리소스에 태그를 지정할 수 있지만, AWS Panorama 콘솔에서 사용하는 모든 태그 관련 권한을 갖고 있지는 않습니다. 이러한 권한을 부여하려면 다음 정책을 추가하세요.

- ResourceGroupsandTagEditorFullAccess – 정책 보기

이 `AWSPanoramaFullAccess` 정책에는 AWS Panorama 콘솔에서 디바이스를 구매할 수 있는 권한이 포함되어 있지 않습니다. 이러한 권한을 부여하려면 다음 정책을 추가하세요.

- ElementalAppliancesSoftwareFullAccess – 정책 보기

관리형 정책은 사용자가 수정할 수 있는 리소스를 제한하지 않고 API 작업에 대한 권한을 부여합니다. 보다 세부적으로 제어하기 위해 사용자의 권한을 제한하는 정책을 직접 만들 수 있습니다. 전체 액세스 정책을 정책의 출발점으로 활용하십시오.

> **서비스 역할 생성**
>
> AWS Panorama 콘솔을 처음 사용하는 경우, AWS Panorama 어플라이언스에서 사용하는 서비스 역할을 생성할 수 있는 권한이 필요합니다. 서비스 역할은 리소스를 관리하거나 다른 서

비스와 상호 작용할 수 있는 권한을 서비스에 부여합니다. 사용자에게 액세스 권한을 부여하기 전에 먼저 이 역할을 생성하세요.

AWS Panorama에서 사용자 권한 범위를 제한하는 데 사용할 수 있는 리소스 및 조건에 대한 자세한 내용은 서비스 인증 참조의 AWS Panorama의 작업, 리소스 및 조건 키를 참조하십시오.

AWS Panorama 서비스 역할 및 교차 서비스 리소스

AWS Panorama는 다른 AWS 서비스를 사용하여 AWS Panorama 어플라이언스를 관리하고, 데이터를 저장하고, 애플리케이션 리소스를 가져옵니다. 서비스 역할은 리소스를 관리하거나 다른 서비스와 상호 작용할 수 있는 권한을 서비스에 부여합니다. AWS Panorama 콘솔에 처음으로 로그인하는 경우 다음 서비스 역할을 생성합니다.

- AWSServiceRoleForAWSPanorama – AWS Panorama에서 AWS IoT, AWS Secrets Manager, AWS Panorama 등에서 리소스를 관리할 수 있습니다.

  관리형 정책: AWSPanoramaServiceLinkedRolePolicy
- AWSPanoramaApplianceServiceRole – AWS Panorama 어플라이언스가 CloudWatch에 로그를 업로드하고, AWS Panorama에서 생성한 Amazon S3 액세스 포인트에서 개체를 가져올 수 있습니다.

  관리형 정책: AWSPanoramaApplianceServiceRolePolicy

각 역할에 연결된 권한을 보려면 IAM 콘솔을 사용하십시오. 가능한 경우 역할의 권한은 AWS Panorama가 사용하는 이름 지정 패턴과 일치하는 리소스로 제한됩니다. 예를 들어 `AWSServiceRoleForAWSPanorama`는 이름에 `panorama`가 있는 AWS IoT 리소스에 액세스할 수 있는 권한만 서비스에게 부여합니다.

섹션

- 어플라이언스 역할 보안
- 기타 서비스 사용

어플라이언스 역할 보안

AWS Panorama 어플라이언스는 `AWSPanoramaApplianceServiceRole` 역할을 사용하여 계정의 리소스에 액세스합니다. 어플라이언스는 로그를 CloudWatch Logs에 업로드하고, AWS Secrets Manager에서 카메라 스트림 보안 인증 정보을 읽고, AWS Panorama가 생성하는 Amazon Simple Storage Service(S3) 액세스 포인트의 애플리케이션 아티팩트에 액세스할 수 있는 권한을 가집니다.

> **Note**
>
> 애플리케이션은 어플라이언스의 권한을 사용하지 않습니다. 애플리케이션에 AWS 서비스 사용 권한을 부여하려면 애플리케이션 역할을 생성하십시오.

AWS Panorama는 계정의 모든 어플라이언스에서 동일한 서비스 역할을 사용하며 계정 간에 역할을 사용하지 않습니다. 보안을 강화하기 위해 어플라이언스 역할의 신뢰 정책을 수정하여 이를 명시적으로 적용할 수 있습니다. 역할을 사용하여 서비스에 계정의 리소스에 액세스할 수 있는 권한을 부여하는 것이 가장 좋습니다.

어플라이언스 역할 신뢰 정책을 업데이트하려면

1. IAM 콘솔에서 어플라이언스 역할: AWSPanoramaApplianceServiceRole을 엽니다.
2. 신뢰 관계 편집(Edit trust relationship)을 선택합니다.
3. 정책 내용을 업데이트한 다음 신뢰 정책 업데이트를 선택합니다.

다음 신뢰 정책에는 AWS Panorama가 어플라이언스 역할을 맡을 때 계정의 어플라이언스에 대해 해당 역할을 수행하도록 보장하는 조건이 포함되어 있습니다. `aws:SourceAccount` 조건은 AWS Panorama에서 지정한 계정 ID를 정책에 포함된 계정 ID와 비교합니다.

Example 신뢰 정책 – 특정 계정

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

AWS Panorama를 더 제한하여 특정 디바이스에서만 역할을 수행하도록 허용하려면 ARN으로 디바이스를 지정하면 됩니다. `aws:SourceArn` 조건은 AWS Panorama에서 지정한 어플라이언스 ARN을 정책에 포함된 어플라이언스 ARN과 비교합니다.

Example 신뢰 정책 – 단일 어플라이언스

```
{
  "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"
        }
      }
    }
  ]
}
```

어플라이언스를 재설정하고 다시 프로비저닝하는 경우 소스 ARN 조건을 일시적으로 제거한 다음 새 디바이스 ID로 다시 추가해야 합니다.

이러한 조건에 대한 자세한 내용과 서비스가 역할을 사용하여 계정의 리소스에 액세스할 때의 보안 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 혼란스러운 대리인 문제를 참조하십시오.

기타 서비스 사용

AWS Panorama는 다음 서비스에서 리소스를 생성하거나 액세스합니다.

- AWS IoT – AWS Panorama 어플라이언스를 위한 사물, 정책, 인증서 및 작업입니다.
- Amazon S3 – 애플리케이션 모델, 코드 및 구성을 스테이징하기 위한 액세스 포인트입니다.
- Secrets Manager – AWS Panorama 어플라이언스의 단기 보안 인증 정보입니다.

Amazon 리소스 이름(ARN) 형식 또는 각 서비스의 권한 범위에 대한 자세한 내용은 이 목록에 링크된 IAM 사용 설명서의 주제를 참조하십시오.

애플리케이션에 권한 부여

애플리케이션에 AWS 서비스를 호출할 권한을 부여하는 역할을 생성할 수 있습니다. 기본적으로 애플리케이션에는 권한이 없습니다. IAM에서 애플리케이션 역할을 생성하고 배포 중에 애플리케이션에 할당합니다. 애플리케이션에 필요한 권한만 부여하려면 특정 API 작업에 대한 권한이 있는 역할을 생성하십시오.

샘플 애플리케이션에는 애플리케이션 역할을 생성하는 AWS CloudFormation 템플릿과 스크립트가 포함되어 있습니다. AWS Panorama가 맡을 수 있는 서비스 역할입니다. 이 역할은 애플리케이션이 CloudWatch를 호출하여 지표를 업로드할 수 있는 권한을 부여합니다.

Example aws-panorama-sample.yml – 애플리케이션 역할

```
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/
```

`Action`의 값에 대한 API 작업 또는 패턴 목록을 지정하고 이 스크립트를 확장하여 다른 서비스에 권한을 부여할 수 있습니다.

AWS Panorama에서 권한을 부여하는 방법에 대한 자세한 내용은 AWS Panorama 권한 단원을 참조하십시오.

AWS Panorama 어플라이언스 관리

AWS Panorama 어플라이언스는 애플리케이션을 실행하는 하드웨어입니다. AWS Panorama 콘솔은 어플라이언스를 등록하고, 소프트웨어를 업데이트하며, 어플라이언스를 배포하는 데 사용합니다. AWS Panorama 어플라이언스의 소프트웨어는 카메라 스트림에 연결하여 애플리케이션으로 비디오 프레임을 전송하고 연결된 디스플레이에 비디오 출력을 표시합니다.

어플라이언스 또는 다른 호환 디바이스를 설정한 후 애플리케이션에서 사용할 카메라를 등록합니다. AWS Panorama 콘솔에서 카메라 스트림을 관리합니다. 애플리케이션을 배포할 때 어플라이언스가 처리를 위해 전송할 카메라 스트림을 선택합니다.

샘플 애플리케이션과 함께 AWS Panorama 어플라이언스를 소개하는 자습서는 AWS Panorama 시작하기 단원을 참조하십시오.

주제

AWS Panorama 어플라이언스 관리

AWS Panorama 콘솔을 사용하여 AWS Panorama 어플라이언스 및 기타 호환 디바이스를 구성, 업그레이드 또는 등록 취소할 수 있습니다.

어플라이언스를 설정하려면 자습서 시작하기의 지침을 따르십시오. 설정 프로세스는 AWS Panorama에 리소스를 생성하여 어플라이언스를 추적하고 업데이트 및 배포를 조정합니다.

AWS Panorama API에 어플라이언스를 등록하려면 디바이스 등록 자동화을 참조하십시오.

단원

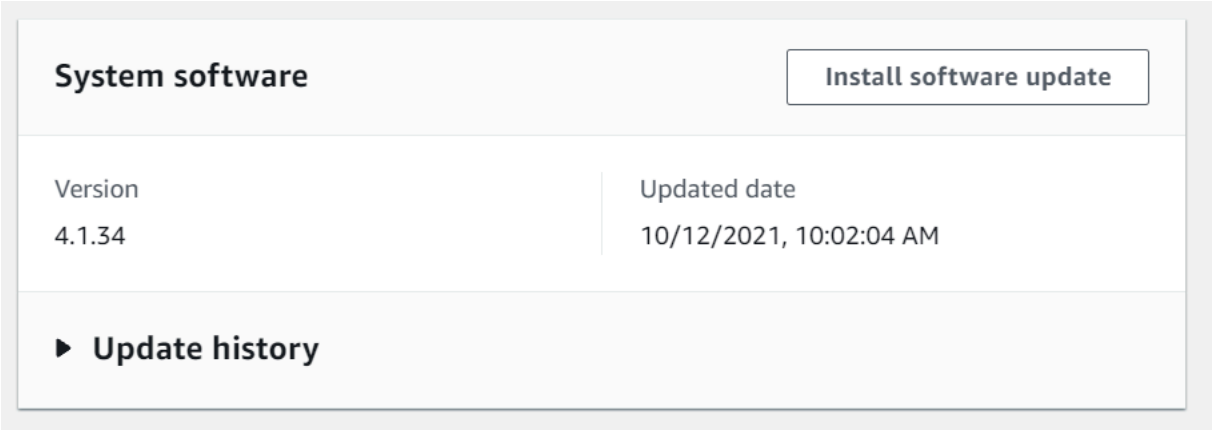
- 어플라이언스 소프트웨어 업데이트
- 어플라이언스 등록 취소
- 어플라이언스 재부팅
- 어플라이언스 재설정

어플라이언스 소프트웨어 업데이트

AWS Panorama 콘솔에서 어플라이언스에 대한 소프트웨어 업데이트를 보고 배포할 수 있습니다. 업데이트는 필수이거나 선택 사항일 수 있습니다. 필요한 업데이트가 제공되면 콘솔에 업데이트를 적용하라는 메시지가 표시됩니다. 어플라이언스 설정 페이지에서 선택적으로 업데이트를 적용할 수 있습니다.

어플라이언스 소프트웨어를 업데이트하려면

1. AWS Panorama 콘솔 디바이스 페이지를 엽니다.
2. 어플라이언스를 선택합니다.
3. 설정을 선택합니다.
4. 시스템 소프트웨어에서 소프트웨어 업데이트 설치를 선택합니다.



5. 새 버전을 선택한 다음 설치를 선택합니다.

어플라이언스 등록 취소

어플라이언스 작업을 완료한 경우, AWS Panorama 콘솔을 사용하여 어플라이언스의 등록을 취소하고 관련 AWS IoT 리소스를 삭제할 수 있습니다.

어플라이언스를 삭제하려면

1. AWS Panorama 콘솔 디바이스 페이지를 엽니다.
2. 어플라이언스 이름을 선택합니다.
3. 삭제를 선택합니다.
4. 어플라이언스 이름을 입력하고 삭제를 선택합니다.

AWS Panorama 서비스에서 어플라이언스를 삭제해도 어플라이언스의 데이터는 자동으로 삭제되지 않습니다. 등록이 취소된 어플라이언스는 AWS 서비스에 연결할 수 없으며 재설정될 때까지 다시 등록할 수 없습니다.

어플라이언스 재부팅

어플라이언스를 원격으로 재부팅할 수 있습니다.

어플라이언스를 재부팅하려면

1. AWS Panorama 콘솔 디바이스 페이지를 엽니다.
2. 어플라이언스 이름을 선택합니다.
3. 재부팅을 선택합니다.

콘솔은 어플라이언스에 메시지를 전송하여 어플라이언스를 재부팅합니다. 신호를 수신하려면 어플라이언스를 AWS IoT에 연결할 수 있어야 합니다. AWS Panorama API를 사용하여 어플라이언스를 재부팅하려면 어플라이언스 재부팅를 참조하십시오.

어플라이언스 재설정

다른 리전 또는 다른 계정의 어플라이언스를 사용하려면 어플라이언스를 재설정하고 새 인증서로 어플라이언스를 다시 프로비저닝해야 합니다. 디바이스를 재설정하면 필요한 최신 소프트웨어 버전이 적용되고 모든 계정 데이터가 삭제됩니다.

재설정 작업을 시작하려면 어플라이언스의 플러그를 꽂고 전원을 꺼야 합니다. 전원 및 재설정 버튼을 함께 5초 동안 길게 누릅니다. 버튼에서 손을 떼면 상태 표시등이 주황색으로 깜박입니다. 어플라이언스를 프로비저닝하거나 연결을 끊기 전에 상태 표시등이 녹색으로 깜박일 때까지 기다리십시오.

또한 디바이스에서 인증서를 삭제하지 않고도 어플라이언스 소프트웨어를 재설정할 수 있습니다. 자세한 내용은 전원 및 재설정 버튼 단원을 참조하세요.

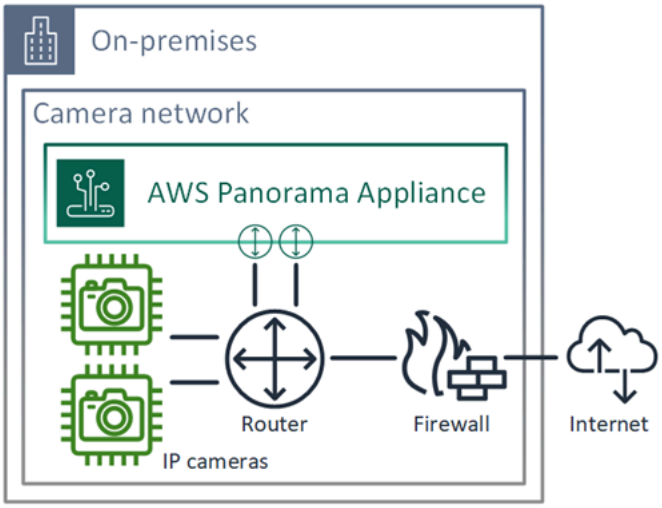
AWS Panorama 어플라이언스를 네트워크에 연결하기

AWS Panorama 어플라이언스를 사용하려면 AWS 클라우드와 온프레미스 IP 카메라 네트워크 모두에 연결해야 합니다. 어플라이언스를 단일 방화벽에 연결하여 둘 다에 대한 액세스를 허용하거나 디바이스의 두 네트워크 인터페이스를 각각 다른 서브넷에 연결할 수 있습니다. 어느 경우든 카메라 스트림에 대한 무단 액세스를 방지하려면 어플라이언스의 네트워크 연결을 보호해야 합니다.

단원

단일 네트워크 구성

어플라이언스에는 이더넷 포트가 두 개 있습니다. 장치로 들어오고 나가는 모든 트래픽을 단일 라우터를 통해 라우팅하는 경우, 첫 번째 포트와의 물리적 연결이 끊어지는 상황에 대비하여 두 번째 포트를 이중화에 사용할 수 있습니다. 어플라이언스가 카메라 스트림과 인터넷에만 연결되도록 하고 카메라 스트림이 내부 네트워크를 벗어나는 것을 차단하도록 라우터를 구성하십시오.

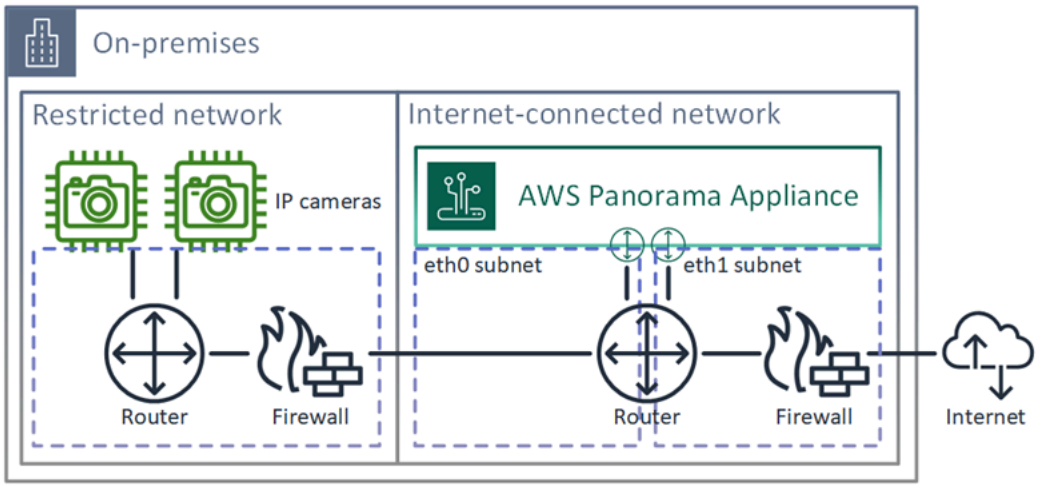


어플라이언스가 액세스해야 하는 포트 및 엔드포인트에 대한 자세한 내용은 서비스 액세스 구성 및 로컬 네트워크 액세스 구성을 참조하십시오.

이중 네트워크 구성

보안을 강화하기 위해 카메라 네트워크와 분리된 인터넷 연결 네트워크에 어플라이언스를 배치할 수 있습니다. 제한된 카메라 네트워크와 어플라이언스 네트워크 사이의 방화벽은 어플라이언스가 비디오 스트림에만 액세스하도록 허용합니다. 이전에 보안을 위해 카메라 네트워크를 에어갭 연결한 적이 있는 경우, 인터넷 액세스를 허용하는 라우터에 카메라 네트워크를 연결하는 것보다 이 방법이 더 좋을 수 있습니다.

다음 예는 각 포트의 다른 서브넷에 연결하는 어플라이언스를 보여줍니다. 라우터는 카메라 네트워크로 라우팅되는 서브넷에 `eth0` 인터페이스를, 인터넷으로 라우팅되는 서브넷에 `eth1` 인터페이스를 배치합니다.



AWS Panorama 콘솔에서 각 포트의 IP 주소 및 MAC 주소를 확인할 수 있습니다.

서비스 액세스 구성

프로비저닝 중에 특정 IP 주소를 요청하도록 어플라이언스를 구성할 수 있습니다. 미리 IP 주소를 선택하여 방화벽 구성을 단순화하고 어플라이언스가 장기간 오프라인 상태인 경우 어플라이언스의 주소가 변경되지 않도록 하십시오.

어플라이언스는 AWS 서비스를 사용하여 소프트웨어 업데이트 및 배포를 조정합니다. 어플라이언스가 이러한 엔드포인트에 연결할 수 있도록 방화벽을 구성하십시오.

인터넷 액세스

- AWS IoT(HTTPS 및 MQTT, 포트 443, 8443 및 8883) – AWS IoT Core 및 디바이스 관리 엔드포인트. 자세한 내용은 Amazon Web Services 일반 참조의 AWS IoT Device Management 엔드포인트 및 할당량을 참조하십시오.

- AWS IoT 보안 인증 정보(HTTPS, 포트 443) – `credentials.iot.<region>.amazonaws.com` 및 하위 도메인.
- Amazon Elastic Container Registry(HTTPS, 포트 443) – `api.ecr.<region>.amazonaws.com`, `dkr.ecr.<region>.amazonaws.com` 및 하위 도메인.
- Amazon CloudWatch(HTTPS, 포트 443) – `monitoring.<region>.amazonaws.com`.
- Amazon CloudWatch Logs(HTTPS, 포트 443) – `logs.<region>.amazonaws.com`.
- Amazon Simple Storage Service(HTTPS, 포트 443) – `s3.<region>.amazonaws.com`, `s3-accesspoint.<region>.amazonaws.com` 및 하위 도메인.

애플리케이션이 다른 AWS 서비스를 호출하는 경우 어플라이언스는 해당 서비스의 엔드포인트에도 액세스해야 합니다. 자세한 내용은 서비스 엔드포인트 및 할당량을 참조하세요.

로컬 네트워크 액세스 구성

어플라이언스는 로컬에서 RTSP 비디오 스트림에 액세스해야 하지만 인터넷을 통해서는 액세스할 수 없습니다. 어플라이언스가 포트 554의 RTSP 스트림에 내부적으로 액세스할 수 있도록 허용하고 스트림이 인터넷에서 송수신되지 않도록 방화벽을 구성하십시오.

로컬 액세스

- 실시간 스트리밍 프로토콜(RTSP, 포트 554) - 카메라 스트림을 읽습니다.
- 네트워크 시간 프로토콜(NTP, 포트 123) - 어플라이언스의 시계를 동기화된 상태로 유지합니다. 네트워크에서 NTP 서버를 실행하지 않는 경우 어플라이언스는 인터넷을 통해 공용 NTP 서버에도 연결할 수 있습니다.

프라이빗 연결

AWS Panorama 어플라이언스를 AWS에 대한 VPN 연결이 있는 사설 VPC 서브넷에 배포하는 경우 인터넷 액세스가 필요하지 않습니다. 사이트 간 VPN 또는 AWS Direct Connect를 사용하여 온프레미스 라우터와 AWS 간에 VPN 연결을 만들 수 있습니다. 프라이빗 VPC 서브넷 내에서 어플라이언스가 Amazon Simple Storage Service, AWS IoT 및 기타 서비스에 연결할 수 있는 엔드포인트를 생성합니다. 자세한 내용은 어플라이언스를 프라이빗 서브넷에 연결 단원을 참조하세요.

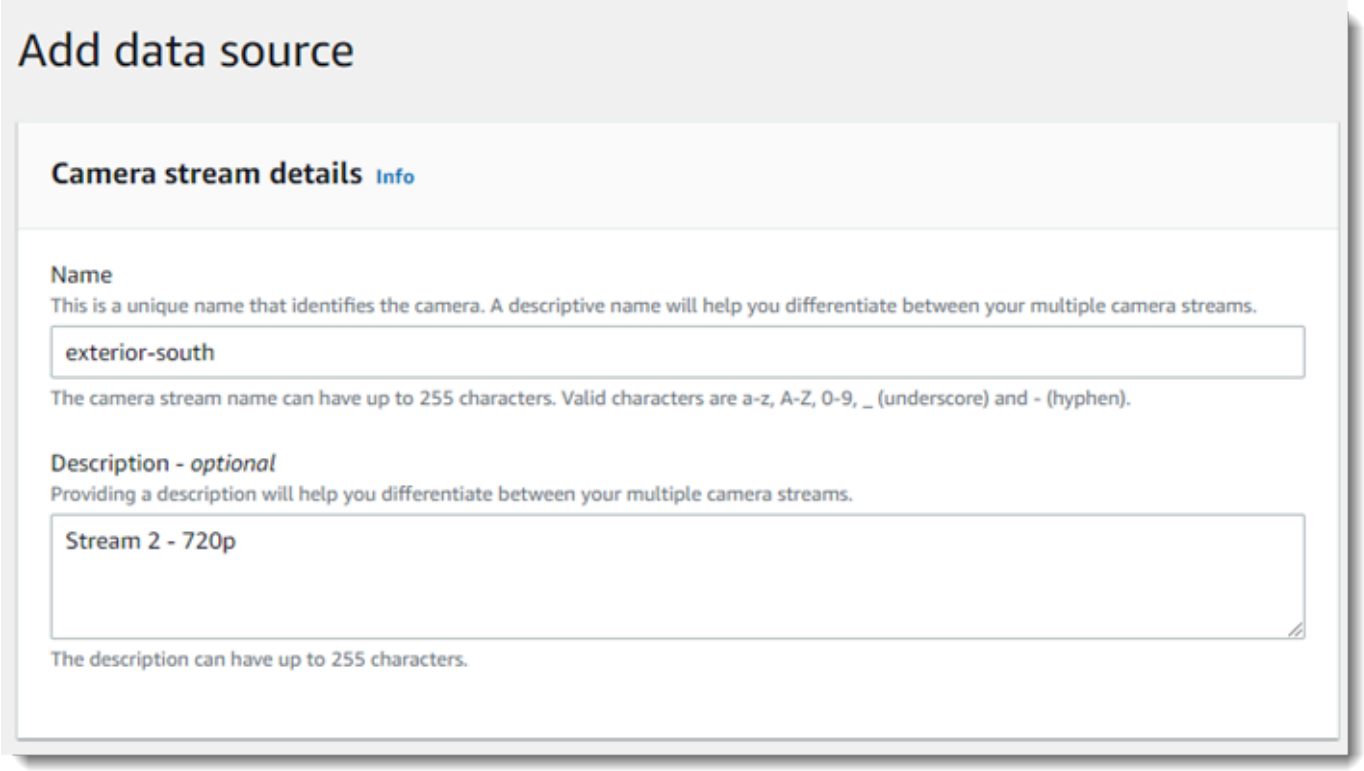
AWS Panorama에서의 카메라 스트림 관리

비디오 스트림을 애플리케이션의 데이터 소스로 등록하려면 AWS Panorama 콘솔을 사용하세요. 애플리케이션은 여러 스트림을 동시에 처리할 수 있으며 여러 어플라이언스를 동일한 스트림에 연결할 수 있습니다.

> **Important**
>
> 애플리케이션은 연결되는 로컬 네트워크에서 라우팅할 수 있는 모든 카메라 스트림에 연결할 수 있습니다. 비디오 스트림을 보호하려면 로컬에서 RTSP 트래픽만 허용하도록 네트워크를 구성하세요. 자세한 내용은 AWS Panorama의 보안 단원을 참조하세요.

카메라 스트림을 등록하려면

1. AWS Panorama 콘솔 데이터 소스 페이지를 엽니다.
2. 데이터 소스 추가를 선택합니다.



3. 다음 설정을 구성합니다.

   - 이름 – 카메라 스트림의 이름입니다.

- 설명 – 카메라, 위치 또는 기타 세부 정보에 대한 간략한 설명입니다.
- RTSP URL – 카메라의 IP 주소와 스트림 경로를 지정하는 URL입니다. 예: `rtsp://192.168.0.77/live/mpeg4/`
- 보안 인증 정보 - 카메라 스트림을 암호로 보호하는 경우 사용자 이름과 암호를 지정합니다.

4. 저장을 선택합니다.

AWS Panorama API에 카메라 스트림을 등록하려면 디바이스 등록 자동화을 참조하십시오.

AWS Panorama 어플라이언스와 호환되는 카메라 목록은 지원되는 컴퓨터 비전 모델 및 카메라를 참조하십시오.

스트림 제거

AWS Panorama 콘솔에서 카메라 스트림을 삭제할 수 있습니다.

카메라 스트림을 제거하려면

1. AWS Panorama 콘솔 데이터 소스 페이지를 엽니다.
2. 카메라 스트림을 선택합니다.
3. 데이터 소스 삭제를 선택합니다.

서비스에서 카메라 스트림을 제거해도 애플리케이션 실행이 중단되거나 Secrets Manager에서 카메라 보안 인증 정보가 삭제되지 않습니다. 보안 암호를 삭제하려면 Secrets Manager 콘솔을 사용하세요.

AWS Panorama 어플라이언스의 애플리케이션 관리

애플리케이션에는 코드, 모델, 구성이 조합되어 있습니다. AWS Panorama 콘솔의 디바이스 페이지에서 어플라이언스의 애플리케이션을 관리할 수 있습니다.

AWS Panorama 어플라이언스의 애플리케이션을 관리하려면

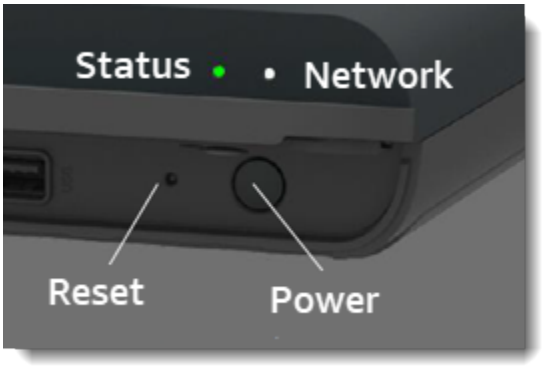
1. AWS Panorama 콘솔 디바이스 페이지를 엽니다.
2. 어플라이언스를 선택합니다.

배포된 애플리케이션 페이지에는 어플라이언스에 배포된 애플리케이션이 표시됩니다.

이 페이지의 옵션을 사용하여 배포된 애플리케이션을 장치에서 제거하거나 실행 중인 애플리케이션을 새 버전으로 교체하십시오. 또한 실행 중인 애플리케이션 또는 삭제된 애플리케이션을 복제하여 새 복사본을 배포할 수 있습니다.

AWS Panorama 어플라이언스 버튼 및 표시등

AWS Panorama 어플라이언스는 전원 버튼 위에 디바이스 상태와 네트워크 연결을 나타내는 LED 표시등 두 개가 있습니다.



상태 표시등

LED 색상이 바뀌고 깜박여 상태를 나타냅니다. 느리게 깜박이는 것은 3초에 한 번입니다. 빠르게 깜박이는 것은 1초에 한 번입니다.

상태 LED의 상태

- 녹색으로 빠르게 깜박임 - 어플라이언스가 부팅 중입니다.
- 초록색 점등 – 어플라이언스가 정상적으로 작동하고 있습니다.
- 파란색으로 느리게 깜박임 - 어플라이언스가 구성 파일을 복사하고 AWS IoT에 등록을 시도하고 있습니다.
- 파란색으로 빠르게 깜박임 - 어플라이언스가 USB 드라이브에 로그 이미지를 복사하고 있습니다.
- 빨간색으로 빠르게 깜박임 - 어플라이언스 시작 중에 오류가 발생했거나 과열되었습니다.
- 주황색으로 느리게 깜박임 - 어플라이언스가 최신 소프트웨어 버전을 복원하는 중입니다.
- 주황색으로 빠르게 깜박임 - 어플라이언스가 최소 소프트웨어 버전을 복원하는 중입니다.

네트워크 표시등

네트워크 LED의 상태는 다음과 같습니다.

네트워크 LED 상태

- 녹색 점등 – 이더넷 케이블이 연결되었습니다.

- 녹색으로 깜박임 - 어플라이언스가 네트워크를 통해 통신하고 있습니다.
- 빨간색 점등 - 이더넷 케이블이 연결되지 않았습니다.

전원 및 재설정 버튼

전원 및 재설정 버튼은 장치 전면의 보호 커버 아래에 있습니다. 재설정 버튼은 더 작고 움푹 들어가 있습니다. 작은 드라이버나 클립을 사용하여 누르세요.

어플라이언스를 재설정하는 방법

1. 어플라이언스의 플러그를 꽂고 전원을 꺼야 합니다. 어플라이언스의 전원을 끄려면 전원 버튼을 1초 동안 누르고 종료 시퀀스가 완료될 때까지 기다립니다. 종료 시퀀스는 약 10초 정도 걸립니다.
2. 어플라이언스를 재설정하려면 다음 버튼 조합을 사용하세요. 짧게 누르는 것은 1초입니다. 길게 누르는 시간은 5초입니다. 여러 개의 버튼이 필요한 작업의 경우 두 버튼을 동시에 길게 누르세요.

   - 전체 재설정 – 전원과 재설정을 길게 누르세요.

     최소 소프트웨어 버전을 복원하고 모든 구성 파일 및 애플리케이션을 삭제합니다.
   - 최신 소프트웨어 버전 복원 – 재설정을 짧게 누르세요.

     최신 소프트웨어 업데이트를 어플라이언스에 다시 적용합니다.
   - 최소 소프트웨어 버전 복원 – 재설정을 길게 누르세요.

     최신 필수 소프트웨어 업데이트를 어플라이언스에 다시 적용합니다.
3. 두 버튼을 모두 놓습니다. 어플라이언스가 켜지고 상태 표시등이 몇 분 동안 주황색으로 깜박입니다.
4. 어플라이언스가 준비되면 상태 표시등이 녹색으로 깜박입니다.

어플라이언스를 재설정해도 AWS Panorama 서비스에서 삭제되지는 않습니다. 자세한 내용은 어플라이언스 등록 취소 단원을 참조하세요.

AWS Panorama 애플리케이션 관리

애플리케이션은 AWS Panorama 어플라이언스에서 실행되어 비디오 스트림에서 컴퓨터 비전 작업을 수행합니다. Python 코드와 기계 학습 모델을 결합하여 컴퓨터 비전 애플리케이션을 구축하고 인터넷을 통해 AWS Panorama 어플라이언스에 배포할 수 있습니다. 애플리케이션은 비디오를 디스플레이로 전송하거나 AWS SDK를 사용하여 결과를 AWS 서비스에 전송할 수 있습니다.

주제

애플리케이션 배포

애플리케이션을 배포하려면 AWS Panorama Application CLI를 사용하여 계정으로 애플리케이션을 가져오고, 컨테이너를 빌드하고, 자산을 업로드 및 등록하고, 애플리케이션 인스턴스를 생성합니다. 이 주제에서는 각 단계를 자세히 살펴보고 백그라운드에서 어떤 작업이 수행되는지 설명합니다.

아직 애플리케이션을 배포하지 않은 경우 자세한 내용은 AWS Panorama 시작하기 단원을 참조하십시오.

샘플 애플리케이션을 사용자 지정하고 확장하는 방법에 대한 자세한 내용은 AWS Panorama 애플리케이션 빌드 단원을 참조하십시오.

섹션

- AWS Panorama Application CLI 설치
- 애플리케이션 가져오기
- 컨테이너 이미지 빌드
- 모델 가져오기
- 애플리케이션 자산 업로드
- AWS Panorama 콘솔을 사용하여 애플리케이션 배포
- 애플리케이션 배포 자동화

AWS Panorama Application CLI 설치

AWS Panorama Application CLI와 AWS CLI를 설치하려면 pip를 사용합니다.

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

AWS Panorama Application CLI로 애플리케이션 이미지를 빌드하려면 Docker가 필요합니다. Linux에서는 qemu와 관련 시스템 라이브러리도 필요합니다. AWS Panorama Application CLI의 설치 및 구성에 대한 자세한 내용은 프로젝트의 GitHub 리포지토리에 있는 README 파일을 참조하십시오.

- github.com/aws/aws-panorama-cli

WSL2를 사용하여 Windows에서 빌드 환경을 설정하기 위한 지침은 Windows에서 개발 환경 설정 단원을 참조하십시오.

애플리케이션 가져오기

샘플 애플리케이션 또는 타사에서 제공한 애플리케이션으로 작업하는 경우, AWS Panorama Application CLI를 사용하여 애플리케이션을 가져오십시오.

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

이 명령은 애플리케이션 패키지의 이름을 계정 ID로 바꿉니다. 패키지 이름은 패키지가 배포되는 계정의 계정 ID로 시작합니다. 애플리케이션을 여러 계정에 배포하는 경우 각 계정별로 애플리케이션을 가져와 패키징해야 합니다.

예를 들어, 이 설명서의 샘플 애플리케이션에는 코드 패키지와 모델 패키지가 있으며, 각각 플레이스홀더 계정 ID로 이름이 지정되어 있습니다. 이 `import-application` 명령은 CLI가 작업 영역의 AWS 보안 인증 정보에서 추론하는 계정 ID를 사용하도록 이름을 바꿉니다.

```
/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
```

123456789012는 패키지 디렉토리 이름과 이를 참조하는 애플리케이션 매니페스트(`graph.json`)에서 계정 ID로 대체됩니다. AWS CLI를 사용하여 `aws sts get-caller-identity`를 호출하면 계정 ID를 확인할 수 있습니다.

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

```
}
```

컨테이너 이미지 빌드

애플리케이션 코드는 Dockerfile에 설치한 애플리케이션 코드와 라이브러리가 포함된 Docker 컨테이너 이미지에 패키징됩니다. AWS Panorama Application CLI `build-container` 명령을 사용하여 도커 이미지를 구축하고 파일 시스템 이미지를 내보냅니다.

```
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
```

이 명령은 `code_asset`라는 이름의 도커 이미지를 생성하고 파일 시스템을 `assets` 폴더의 `.tar.gz` 아카이브로 내보냅니다. CLI는 애플리케이션의 Dockerfile에 지정된 대로 Amazon Elastic Container Registry(Amazon ECR)에서 애플리케이션 기본 이미지를 가져옵니다.

CLI는 컨테이너 아카이브 외에도 패키지 설명자(`descriptor.json`)에 대한 자산을 생성합니다. 두 파일 모두 원본 파일의 해시를 반영하는 고유 식별자로 이름이 바뀝니다. 또한 AWS Panorama Application CLI는 두 자산의 이름을 기록하는 블록을 패키지 구성에 추가합니다. 이러한 이름은 배포 프로세스 중에 어플라이언스에서 사용됩니다.

Example packages/123456789012-SAMPLE_CODE-1.0/package.json – 자산 블록 포함

```
{
    "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 명령에 지정된 코드 자산의 이름은 패키지 구성의 asset 필드 값과 일치해야 합니다. 앞의 예에서는 두 값이 모두 code_asset입니다.

모델 가져오기

애플리케이션의 자산 폴더에 모델 아카이브가 있거나 별도로 다운로드한 모델 아카이브가 있을 수 있습니다. 새 모델, 업데이트된 모델 또는 업데이트된 모델 설명자 파일이 있는 경우 add-raw-model 명령을 사용하여 해당 파일을 가져오십시오.

```
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
```

설명자 파일만 업데이트해야 하는 경우 자산 디렉토리에 있는 기존 모델을 재사용할 수 있습니다. 부동 소수점 정밀도 모드와 같은 기능을 구성하려면 설명자 파일을 업데이트해야 할 수 있습니다. 예를 들어, 다음 스크립트는 샘플 앱으로 이 작업을 수행하는 방법을 보여줍니다.

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

```
#!/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
```

모델 패키지 디렉토리의 디스크립터 파일에 대한 변경 사항은 CLI로 다시 가져올 때까지 적용되지 않습니다. CLI는 컨테이너를 다시 빌드할 때 애플리케이션 코드 패키지의 구성을 업데이트하는 방식과 유사하게 새 자산 이름을 사용하여 모델 패키지 구성을 업데이트합니다.

애플리케이션 자산 업로드

모델 아카이브, 컨테이너 파일 시스템 아카이브 및 설명자 파일을 포함하는 애플리케이션 자산을 업로드하고 등록하려면 `package-application` 명령을 사용하십시오.

```
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
...
```

자산 파일 또는 패키지 구성에 변경 사항이 없는 경우 CLI는 이를 건너뜁니다.

```
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
```

CLI는 각 패키지의 자산을 사용자 계정별 Amazon S3 액세스 포인트에 업로드합니다. AWS Panorama는 사용자를 대신하여 액세스 포인트를 관리하고 DescribePackage API를 통해 정보를 제공합니다. CLI는 각 패키지의 자산을 해당 패키지에 제공된 위치에 업로드하고 패키지 구성에 설명된 설정을 사용하여 AWS Panorama 서비스에 등록합니다.

AWS Panorama 콘솔을 사용하여 애플리케이션 배포

AWS Panorama 콘솔을 사용하여 애플리케이션을 배포할 수 있습니다. 배포 프로세스 중에 애플리케이션 코드로 전달할 카메라 스트림을 선택하고 애플리케이션 개발자가 제공하는 옵션을 구성합니다.

애플리케이션을 배포하려면

1. AWS Panorama 콘솔 배포 애플리케이션 페이지를 엽니다.
2. 애플리케이션 배포를 선택합니다.
3. 애플리케이션 매니페스트의 콘텐츠 `graph.json`를 텍스트 편집기에 붙여넣습니다. 다음을 선택합니다.
4. 이름 및 설명을 입력합니다.
5. 배포로 진행을 선택합니다.
6. 배포 시작을 선택합니다.
7. 애플리케이션에서 역할을 사용하는 경우 드롭다운 메뉴에서 선택합니다. 다음을 선택합니다.
8. 디바이스 선택을 선택한 다음 어플라이언스를 선택합니다. 다음을 선택합니다.
9. 데이터 소스 선택 단계에서 입력 보기를 선택하고 카메라 스트림을 데이터 소스로 추가합니다. 다음을 선택합니다.
10. 구성 단계에서 개발자가 정의한 애플리케이션별 설정을 구성합니다. 다음을 선택합니다.
11. 배포를 선택한 다음 완료를 선택합니다.
12. 배포된 애플리케이션 목록에서 상태를 모니터링할 애플리케이션을 선택합니다.

배포 프로세스는 15~20분이 걸립니다. 애플리케이션이 시작되는 동안 어플라이언스의 출력이 오랫동안 비어 있을 수 있습니다. 오류가 발생하는 경우 문제 해결 단원을 참조하십시오.

애플리케이션 배포 자동화

CreateApplicationInstance API를 사용하여 애플리케이션 배포 프로세스를 자동화할 수 있습니다. API는 두 개의 구성 파일을 입력으로 받습니다. 애플리케이션 매니페스트는 사용된 패키지와 그 관계를 지정합니다. 두 번째 파일은 애플리케이션 매니페스트 값의 배포 시 재정의를 지정하는 오버라이드 파일입니다. 오버라이드 파일을 사용하면 동일한 애플리케이션 매니페스트를 사용하여 다양한 카메라 스트림과 함께 애플리케이션을 배포하고 기타 애플리케이션별 설정을 구성할 수 있습니다.

이 주제의 각 단계에 대한 자세한 내용 및 예제 스크립트는 애플리케이션 배포 자동화 단원을 참조하십시오.

AWS Panorama 콘솔에서 애플리케이션 관리

AWS Panorama 콘솔을 사용하여 배포된 애플리케이션을 관리합니다.

섹션

애플리케이션 업데이트 또는 복사

애플리케이션을 업데이트하려면 바꾸기 옵션을 사용하십시오. 애플리케이션을 교체할 때 해당 코드나 모델을 업데이트할 수 있습니다.

애플리케이션을 업데이트하려면

1. AWS Panorama 콘솔 배포 애플리케이션 페이지를 엽니다.
2. 애플리케이션을 선택합니다.
3. 바꾸기를 선택합니다.
4. 지침에 따라 새 버전 또는 애플리케이션을 만듭니다.

바꾸기와 비슷하게 작동하지만 이전 버전의 애플리케이션을 제거하지는 않는 복제 옵션도 있습니다. 이 옵션을 사용하면 실행 중인 버전을 중지하지 않고도 애플리케이션의 변경 내용을 테스트하거나 이미 삭제한 버전을 재배포할 수 있습니다.

버전 및 애플리케이션 삭제

사용하지 않는 애플리케이션 버전을 정리하려면 어플라이언스에서 삭제하십시오.

애플리케이션을 삭제하려면

1. AWS Panorama 콘솔 배포 애플리케이션 페이지를 엽니다.
2. 애플리케이션을 선택합니다.
3. 디바이스에서 삭제를 선택합니다.

패키지 구성

AWS Panorama Application CLI 명령 `panorama-cli package-application`을 사용하면 CLI가 애플리케이션의 자산을 Amazon S3에 업로드하고 이를 AWS Panorama에 등록합니다. 자산에는 AWS Panorama 어플라이언스가 배포 중에 다운로드하는 바이너리 파일(컨테이너 이미지 및 모델) 및 설명자 파일이 포함됩니다. 패키지 자산을 등록하려면 패키지, 자산 및 인터페이스를 정의하는 별도의 패키지 구성 파일을 제공합니다.

다음 예제는 입력과 출력이 하나씩 있는 코드 노드의 패키지 구성을 보여줍니다. 비디오 입력을 통해 카메라 스트림의 이미지 데이터에 액세스할 수 있습니다. 출력 노드는 처리된 이미지를 디스플레이로 전송합니다.

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"
                    }
                ]
            }
        ]
    }
}
```

이 `assets` 섹션은 AWS Panorama Application CLI가 Amazon S3에 업로드한 아티팩트의 이름을 지정합니다. 샘플 애플리케이션 또는 다른 사용자로부터 애플리케이션을 가져오는 경우 이 섹션이 비어 있거나 계정에 없는 자산을 참조할 수 있습니다. `panorama-cli package-application`을 실행하면 AWS Panorama Application CLI가 이 섹션을 올바른 값으로 채웁니다.

AWS Panorama 애플리케이션 매니페스트

애플리케이션을 배포할 때 애플리케이션 매니페스트라는 구성 파일을 제공하세요. 이 파일은 애플리케이션을 노드와 엣지가 있는 그래프로 정의합니다. 애플리케이션 매니페스트는 애플리케이션 소스 코드의 일부이며 `graphs` 디렉토리에 저장됩니다.

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"
            }
        ]
    }
}
```

노드는 엣지에 의해 연결되며, 엣지는 노드의 입력과 출력 간의 매핑을 지정합니다. 한 노드의 출력이 다른 노드의 입력에 연결되어 그래프를 만듭니다.

JSON 스키마

애플리케이션 매니페스트 및 재정의 문서의 형식은 JSON 스키마에 정의되어 있습니다. 배포하기 전에 JSON 스키마를 사용하여 구성 문서의 유효성을 검사할 수 있습니다. JSON 스키마는 이 설명서의 GitHub 리포지토리에서 사용할 수 있습니다.

- JSON 스키마 – aws-panorama-developer-guide/resources

애플리케이션 노드

노드는 모델, 코드, 카메라 스트림, 출력, 파라미터입니다. 노드에는 입력과 출력을 정의하는 인터페이스가 있습니다. 인터페이스는 계정의 패키지, AWS Panorama에서 제공하는 패키지 또는 내장형으로 정의할 수 있습니다.

다음 예시에서 `code_node`와 `model_node`는 샘플 애플리케이션에 포함된 샘플 코드와 모델 패키지를 참조합니다. `camera_node`는 AWS Panorama 서 제공하는 패키지를 사용하여 배포 중에 지정하는 카메라 스트림의 자리 표시자를 생성합니다.

Example graph.json — 노드

```
        "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."
                }
            }
    ]
```

엣지

엣지는 한 노드의 출력을 다른 노드의 입력에 매핑합니다. 다음 예제에서 첫 번째 엣지는 카메라 스트림 노드의 출력을 애플리케이션 코드 노드의 입력에 매핑합니다. 이름 `video_in` 및 `video_out`는 노드 패키지의 인터페이스에 정의되어 있습니다.

Example graph.json – 엣지

```
        "edges": [
```

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

애플리케이션 코드에서는 `inputs` 및 `outputs` 속성을 사용하여 입력 스트림에서 이미지를 가져오고 출력 스트림으로 이미지를 보냅니다.

Example application.py – 비디오 입력 및 출력

```
    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)
```

추상 노드

애플리케이션 매니페스트에서 추상 노드는 AWS Panorama에서 정의한 패키지를 의미하며, 애플리케이션 매니페스트에서 자리 표시자로 사용할 수 있습니다. AWS Panorama는 두 가지 유형의 추상 노드를 제공합니다.

- 카메라 스트림 – 애플리케이션이 배포 중에 사용하는 카메라 스트림을 선택합니다.

  패키지 이름 – `panorama::abstract_rtsp_media_source`

  인터페이스 이름 – `rtsp_v1_interface`
- HDMI 출력 - 애플리케이션이 비디오를 출력함을 나타냅니다.

  패키지 이름 – `panorama::hdmi_data_sink`

인터페이스 이름 – `hdmi0`

다음 예시는 카메라 스트림을 처리하고 디스플레이에 비디오를 출력하는 애플리케이션의 기본 패키지, 노드 및 엣지 세트를 보여줍니다. AWS Panorama에서 `abstract_rtsp_media_source` 패키지의 인터페이스를 사용하는 카메라 노드는 여러 카메라 스트림을 입력값으로 받을 수 있습니다. `hdmi_data_sink`를 참조하는 출력 노드는 어플라이언스의 HDMI 포트에서 출력되는 비디오 버퍼에 대한 애플리케이션 코드 액세스 권한을 부여합니다.

Example graph.json – 추상 노드

```
{
    "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"
            }
        ]
    }
}
```

애플리케이션 파라미터

파라미터는 기본 유형을 가지며 배포 중에 재정의할 수 있는 노드입니다. 파라미터에는 기본값과 애플리케이션 사용자에게 구성 방법을 지시하는 데코레이터가 있을 수 있습니다.

파라미터 유형

- `string` – 문자열. 예: `DEBUG`.
- `int32` – 정수. 예: `20`
- `float32` – 부동 소수점 수. 예: `47.5`
- `boolean` – `true` 또는 `false`입니다.

다음 예시는 코드 노드에 입력으로 전송되는 문자열과 숫자라는 두 파라미터를 보여줍니다.

Example graph.json – 파라미터

```
       "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"
            }
            ...
        ]
    }
```

애플리케이션 매니페스트에서 직접 파라미터를 수정하거나 배포 시 재정의를 통해 새 값을 제공할 수 있습니다. 자세한 내용은 재정의를 사용한 배포 시 구성 단원을 참조하십시오.

재정의를 사용한 배포 시 구성

배포 중에 파라미터와 추상 노드를 구성합니다. AWS Panorama 콘솔을 사용하여 배포하는 경우, 각 파라미터의 값을 지정하고 카메라 스트림을 입력으로 선택할 수 있습니다. AWS Panorama API를 사용하여 애플리케이션을 배포하는 경우, 오버라이드 문서로 이러한 설정을 지정합니다.

오버라이드 문서는 애플리케이션 매니페스트와 구조가 비슷합니다. 기본 유형이 있는 파라미터의 경우 노드를 정의합니다. 카메라 스트림의 경우 등록된 카메라 스트림에 매핑되는 노드와 패키지를 정의합니다. 그런 다음 애플리케이션 매니페스트에서 대체할 노드를 지정하는 각 노드에 대한 오버라이드를 정의합니다.

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"
    }
}
```

앞의 예시에서는 하나의 문자열 파라미터와 추상 카메라 노드에 대한 오버라이드를 정의합니다. `nodeOverrides`는 이 문서 오버라이드에서 어떤 노드가 애플리케이션 매니페스트에 있는지를 AWS Panorama에 알려줍니다.

AWS Panorama 애플리케이션 빌드

애플리케이션은 AWS Panorama 어플라이언스에서 실행되어 비디오 스트림에서 컴퓨터 비전 작업을 수행합니다. Python 코드와 기계 학습 모델을 결합하여 컴퓨터 비전 애플리케이션을 구축하고 인터넷을 통해 AWS Panorama 어플라이언스에 배포할 수 있습니다. 애플리케이션은 비디오를 디스플레이로 전송하거나 AWS SDK를 사용하여 결과를 AWS 서비스에 전송할 수 있습니다.

모델은 이미지를 분석하여 사람, 차량 및 기타 개체를 감지합니다. 모델은 학습 중에 본 이미지를 기반으로 사물이 무엇이라고 생각하는지, 그 추측을 얼마나 확신하는지 알려줍니다. 자체 이미지 데이터로 모델을 학습시키거나 샘플로 시작할 수 있습니다.

애플리케이션의 코드는 카메라 스트림의 스틸 이미지를 처리하여 모델로 전송하고 결과를 처리합니다. 모델은 여러 개체를 감지하여 모양과 위치를 반환할 수 있습니다. 코드는 이 정보를 사용하여 비디오에 텍스트 또는 그래픽을 추가하거나 저장 또는 추가 처리를 위해 결과를 AWS 서비스에 보낼 수 있습니다.

스트림에서 이미지를 가져오고, 모델과 상호 작용하고, 비디오를 출력하기 위해 애플리케이션 코드는 AWS Panorama 애플리케이션 SDK를 사용합니다. 애플리케이션 SDK는 PyTorch, Apache MXNet, TensorFlow로 생성된 모델을 지원하는 Python 라이브러리입니다.

주제

- 컴퓨터 비전 모델
- 애플리케이션 이미지 구축
- 애플리케이션 코드에서 AWS 서비스 호출
- AWS Panorama Application SDK
- 여러 스레드 실행
- 인바운드 트래픽 지원
- GPU 사용하기
- Windows에서 개발 환경 설정

컴퓨터 비전 모델

컴퓨터 비전 모델은 이미지에서 개체를 감지하도록 학습된 소프트웨어 프로그램입니다. 모델은 먼저 학습을 통해 해당 개체의 이미지를 분석하여 일련의 물체를 인식하는 방법을 학습합니다. 컴퓨터 비전 모델은 이미지를 입력으로 받아 감지한 개체에 대한 정보(예: 개체 유형 및 위치)를 출력합니다. AWS Panorama는 PyTorch, Apache MXNet, TensorFlow로 구축된 컴퓨터 비전 모델을 지원합니다.

> **Note**
>
> AWS Panorama로 테스트한 사전 빌드 모델 목록은 모델 호환성을 참조하십시오.

섹션

코드에서 모델 사용

모델은 감지된 클래스의 확률, 위치 정보 및 기타 데이터를 포함할 수 있는 하나 이상의 결과를 반환합니다. 다음 예시는 비디오 스트림의 이미지에 대한 추론을 실행하고 모델의 출력을 처리 함수로 보내는 방법을 보여줍니다.

Example application.py – 추론

```
    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)
```

다음 예시에서는 기본 분류 모델의 결과를 처리하는 함수를 보여줍니다. 샘플 모델은 결과 배열의 첫 번째이자 유일한 값인 확률 배열을 반환합니다.

Example application.py – 프로세스 결과

```
    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)
```

애플리케이션 코드는 확률이 가장 높은 값을 찾아 초기화 중에 로드되는 리소스 파일의 레이블에 매핑합니다.

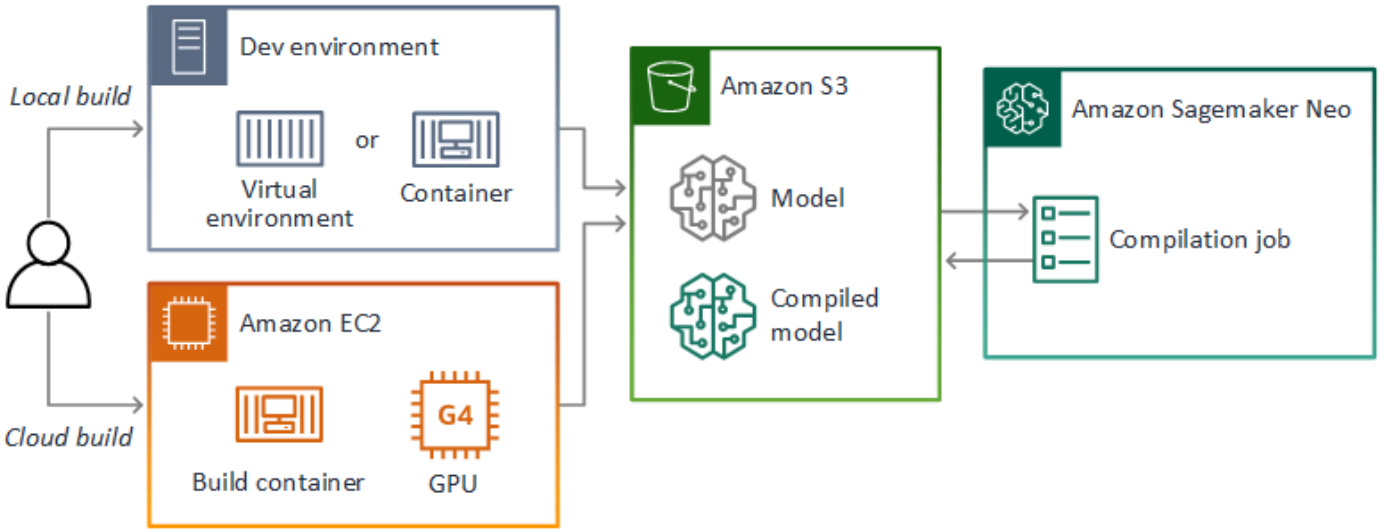
사용자 지정 모델 빌드

PyTorch, Apache MXnet 및 TensorFlow에서 빌드한 모델을 AWS Panorama 애플리케이션에서 사용할 수 있습니다. SageMaker에서 모델을 빌드하고 교육하는 대신 학습된 모델을 사용하거나 지원되는 프레임워크로 자체 모델을 빌드 및 교육한 다음 로컬 환경 또는 Amazon EC2로 내보낼 수 있습니다.

> **Note**
>
> SageMaker Neo에서 지원하는 프레임워크 버전 및 파일 형식에 대한 자세한 내용은 Amazon SageMaker 개발자 안내서의 지원되는 프레임워크를 참조하십시오.

이 설명서의 리포지토리는 TensorFlow SavedModel 형식의 Keras 모델에 대한 이 워크플로를 보여주는 샘플 애플리케이션을 제공합니다. TensorFlow 2를 사용하며 가상 환경 또는 도크 컨테이너에서 로컬로 실행할 수 있습니다. 샘플 앱에는 Amazon EC2 인스턴스에서 모델을 빌드하기 위한 템플릿과 스크립트도 포함되어 있습니다.

- 사용자 지정 모델 샘플 애플리케이션



AWS Panorama는 SageMaker Neo를 사용하여 AWS Panorama 어플라이언스에서 사용할 모델을 컴파일합니다. 각 프레임워크에 대해 SageMaker Neo에서 지원하는 형식을 사용하고 모델을 `.tar.gz` 아카이브로 패키징하십시오.

자세한 내용은 Amazon SageMaker 개발자 안내서의 Neo를 통한 모델 컴파일 및 배포 단원을 참조하십시오.

모델 패키징

모델 패키지는 설명자, 패키지 구성, 모델 아카이브로 구성됩니다. 애플리케이션 이미지 패키지에서와 마찬가지로 패키지 구성은 AWS Panorama 서비스에 Amazon S3의 모델 및 설명자가 저장된 위치를 알려줍니다.

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**
>
> 프레임워크 버전의 메이저 버전과 마이너 버전만 지정하십시오. 지원되는 PyTorch, Apache MXNet, TensorFlow 버전 목록은 지원되는 프레임워크를 참조하십시오.

모델을 가져오려면 AWS Panorama 애플리케이션 CLI `import-raw-model` 명령을 사용하십시오. 모델 또는 설명자를 변경하는 경우 이 명령을 다시 실행하여 애플리케이션 자산을 업데이트해야 합니다. 자세한 내용은 컴퓨터 비전 모델 변경 단원을 참조하십시오.

설명자 파일의 JSON 스키마는 assetDescriptor.schema.json을 참조하십시오.

모델 학습

모델을 학습시킬 때는 대상 환경이나 대상 환경과 매우 유사한 테스트 환경의 이미지를 사용하십시오. 모델 성능에 영향을 줄 수 있는 다음 요인을 고려하십시오.

- 조명 – 피사체에 반사되는 빛의 양에 따라 모델이 분석해야 하는 세부 정보의 양이 결정됩니다. 조명이 밝은 피사체의 이미지로 학습된 모델은 저조도 또는 역광 환경에서는 잘 작동하지 않을 수 있습니다.
- 해상도 – 모델의 입력 크기는 일반적으로 정사각형 종횡비에서 폭 224~512픽셀 사이의 해상도로 고정됩니다. 비디오 프레임을 모델로 전달하기 전에 필요한 크기에 맞게 축소하거나 자를 수 있습니다.

- 이미지 왜곡 – 카메라의 초점 거리와 렌즈 모양으로 인해 프레임 중앙에서 멀어지면 이미지가 왜곡될 수 있습니다. 카메라의 위치에 따라 피사체의 어떤 특징이 보이는지도 결정됩니다. 예를 들어 광각 렌즈가 장착된 오버헤드 카메라는 피사체가 프레임 중앙에 있을 때는 피사체의 위쪽을, 중심에서 멀어지면 피사체의 측면을 비뚤어진 모습으로 보여줍니다.

이러한 문제를 해결하려면 이미지를 모델로 보내기 전에 전처리하고 실제 환경의 변화를 반영하는 더 다양한 이미지에 대해 모델을 학습시킬 수 있습니다. 조명 상황에서 다양한 카메라로 모델을 작동시켜야 하는 경우 학습에 더 많은 데이터가 필요합니다. 더 많은 이미지를 수집하는 것 외에도 왜곡되거나 조명이 다른 기존 이미지의 변형을 만들어 더 많은 학습 데이터를 얻을 수 있습니다.

애플리케이션 이미지 구축

AWS Panorama 어플라이언스는 구축한 이미지에서 내보낸 컨테이너 파일 시스템으로 애플리케이션을 실행합니다. AWS Panorama 애플리케이션 기본 이미지를 시작점으로 사용하는 Dockerfile에서 애플리케이션의 종속 파일과 리소스를 지정합니다.

애플리케이션 이미지를 빌드하려면 Docker와 AWS Panorama Application CLI를 사용하십시오. 이 설명서의 샘플 애플리케이션에 있는 다음 예시는 이러한 사용 사례를 보여줍니다.

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
```

다음 Dockerfile 지침이 사용됩니다.

- FROM – 애플리케이션 기본 이미지(public.ecr.aws/panorama/panorama-application)를 로드합니다.
- WORKDIR – 이미지에 작업 디렉토리를 설정합니다. /panorama는 애플리케이션 코드 및 관련 파일에 사용됩니다. 이 설정은 빌드 중에만 유지되며 런타임 시 애플리케이션의 작업 디렉토리에는 영향을 주지 않습니다(/).
- COPY – 로컬 경로에서 이미지의 경로로 파일을 복사합니다. COPY . .는 현재 디렉토리(패키지 디렉토리)의 파일을 이미지의 작업 디렉토리로 복사합니다. 예를 들어, 애플리케이션 코드는 packages/123456789012-SAMPLE_CODE-1.0/application.py에서 /panorama/application.py로 복사됩니다.
- RUN – 빌드 중에 이미지에서 셸 명령을 실행합니다. 단일 RUN 작업은 명령 사이에 &&을 사용하여 여러 명령을 순차적으로 실행할 수 있습니다. 이 예시는 pip 패키지 관리자를 업데이트한 다음 requirements.txt에 나열된 라이브러리를 설치합니다.

빌드 시 유용한 기타 지침(예: ADD 및ARG)을 사용할 수 있습니다. 컨테이너에 런타임 정보를 추가하는 명령(예: ENV)은 AWS Panorama와 함께 사용할 수 없습니다. AWS Panorama는 해당 이미지에서 컨테이너를 실행하지 않습니다. 이미지를 사용하여 파일 시스템을 내보내고, 파일 시스템은 어플라이언스로 전송됩니다.

종속 파일 지정

`requirements.txt`는 애플리케이션에서 사용하는 라이브러리를 지정하는 Python 요구 사항 파일입니다. 샘플 애플리케이션은 Open CV와 AWS SDK for Python (Boto3)를 사용합니다.

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

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

Dockerfile의 `pip install` 명령은 이러한 라이브러리를 애플리케이션 코드에서 가져올 수 있도록 `/usr/local/lib` 아래의 Python `dist-packages` 디렉토리에 설치합니다.

로컬 스토리지

AWS Panorama는 애플리케이션 스토리지용 `/opt/aws/panorama/storage` 디렉토리를 예약합니다. 애플리케이션은 이 경로에서 파일을 생성하고 수정할 수 있습니다. 스토리지 디렉토리에 생성된 파일은 재부팅 후에도 유지됩니다. 다른 임시 파일 위치는 부팅 시 지워집니다.

이미지 자산 구축

AWS Panorama Application CLI로 애플리케이션 패키지의 이미지를 구축할 때, CLI는 패키지 디렉터리에서 `docker build`를 실행합니다. 그러면 애플리케이션 코드가 포함된 애플리케이션 이미지가 구축됩니다. 그런 다음 CLI는 컨테이너를 생성하고, 해당 파일 시스템을 내보내고, 압축하여 `assets` 폴더에 저장합니다.

```
$ 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
```

출력의 JSON 블록은 CLI가 패키지 구성(`package.json`)에 추가하고 AWS Panorama 서비스에 등록하는 자산 정의입니다. 또한 CLI는 애플리케이션 스크립트(애플리케이션의 진입점) 경로를 지정하는 설명자 파일을 복사합니다.

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

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

자산 폴더에서 설명자와 애플리케이션 이미지는 SHA-256 체크섬의 이름을 따서 명명됩니다. 이 이름은 자산이 Amazon S3에 저장될 때 해당 자산의 고유 식별자로 사용됩니다.

애플리케이션 코드에서 AWS 서비스 호출

AWS SDK for Python (Boto)을 사용하여 애플리케이션 코드에서 AWS 서비스를 호출할 수 있습니다. 예를 들어, 모델에서 이상을 감지하면 Amazon CloudWatch에 지표를 게시하거나, Amazon SNS를 통해 알림을 보내거나, Amazon S3에 이미지를 저장하거나, 추가 처리를 위해 Lambda 함수를 간접적으로 호출할 수 있습니다. 대부분의 AWS 서비스에는 AWS SDK와 함께 사용할 수 있는 공개 API가 있습니다.

어플라이언스에는 기본적으로 AWS 서비스에 액세스할 권한이 없습니다. 권한을 부여하려면 애플리케이션에 대한 역할을 생성하고 배포 중에 애플리케이션 인스턴스에 할당하십시오.

단원

- Amazon S3 사용
- AWS IoT MQTT 주제 사용

Amazon S3 사용

Amazon S3를 사용하여 처리 결과 및 기타 애플리케이션 데이터를 저장할 수 있습니다.

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

AWS IoT MQTT 주제 사용

SDK for Python(Boto3)을 사용하여 AWS IoT에서 MQTT 주제로 메시지를 보낼 수 있습니다. 다음 예에서는 AWS IoT 콘솔에서 확인할 수 있는 어플라이언스의 사물 이름을 따서 이름을 지정한 주제에 애플리케이션이 게시합니다.

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

디바이스 ID 또는 선택한 기타 식별자를 나타내는 이름을 선택합니다. 메시지를 게시하려면 애플리케이션에 `iot:Publish`를 호출할 권한이 있어야 합니다.

MQTT 대기열을 모니터링하려면

1. AWS IoT 콘솔 테스트 페이지를 엽니다.
2. 구독 주제에 주제의 이름을 입력합니다. 예: `panorama/panorama_my-appliance_Thing_a01e373b`.
3. 주제 구독을 선택합니다.

AWS Panorama Application SDK

AWS Panorama Application SDK는 AWS 파노라마 애플리케이션 개발을 위한 Python 라이브러리입니다. 애플리케이션 코드에서 AWS Panorama Application SDK를 사용하여 컴퓨터 비전 모델을 로드하고, 추론을 실행하고, 모니터에 비디오를 출력합니다.

> **Note**
>
> AWS Panorama 애플리케이션 SDK의 최신 기능에 액세스하려면 어플라이언스 소프트웨어를 업그레이드하십시오.

애플리케이션 SDK가 정의하는 클래스와 메서드에 대한 자세한 내용은 Application SDK 참조를 참조하십시오.

단원

- 출력 비디오에 텍스트 및 상자 추가

출력 비디오에 텍스트 및 상자 추가

AWS Panorama SDK를 사용하면 디스플레이에 비디오 스트림을 출력할 수 있습니다. 비디오에는 모델의 출력, 애플리케이션의 현재 상태 또는 기타 데이터를 보여주는 텍스트와 상자가 포함될 수 있습니다.

`video_in` 배열의 각 개체는 기기에 연결된 카메라 스트림의 이미지입니다. 이 개체의 유형은 `panoramasdk.media`입니다. 이미지에 텍스트와 사각형 상자를 추가한 다음 `video_out` 배열에 할당할 수 있는 메서드가 있습니다.

다음 예제에서 샘플 애플리케이션은 각 결과에 레이블을 추가합니다. 각 결과는 왼쪽 위치는 같지만 높이는 서로 다릅니다.

```
        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)
```

출력 이미지에 상자를 추가하려면 `add_rect`를 사용하세요. 이 메서드는 0과 1 사이의 4개 값을 사용하여 상자의 왼쪽 상단과 오른쪽 하단 모서리의 위치를 나타냅니다.

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

여러 스레드 실행

처리 스레드에서 애플리케이션 로직을 실행하고 다른 백그라운드 프로세스에는 다른 스레드를 사용할 수 있습니다. 예를 들어 디버깅을 위해 HTTP 트래픽을 제공하는 스레드나 추론 결과를 모니터링하고 데이터를 AWS로 전송하는 스레드를 만들 수 있습니다.

여러 개의 스레드를 실행하려면 Python 표준 라이브러리의 스레딩 모듈을 사용하여 각 프로세스에 대한 스레드를 만듭니다. 다음 예시는 애플리케이션 개체를 만들고 이를 사용하여 세 개의 스레드를 실행하는 디버그 서버 샘플 애플리케이션의 메인 루프를 보여줍니다.

Example packages/123456789012-DEBUG_SERVER-1.0/application.py – 메인 루프

```
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.')
```

모든 스레드가 종료되면 애플리케이션이 자동으로 다시 시작됩니다. `run_cv` 루프는 카메라 스트림의 이미지를 처리합니다. 중지 신호를 받으면 디버거 프로세스를 종료합니다. 디버거 프로세스는 HTTP 서버를 실행하지만 스스로 종료할 수는 없습니다. 각 스레드는 자체 오류를 처리해야 합니다. 오류가 발견되어 로그되지 않으면 스레드가 자동으로 종료됩니다.

Example packages/123456789012-DEBUG_SERVER-1.0/application.py – 처리 루프

```
    # 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")
```

스레드는 애플리케이션의 `self` 개체를 통해 통신합니다. 디버거 스레드는 애플리케이션 처리 루프를 다시 시작하기 위해 `stop` 메서드를 호출합니다. 이 메서드는 다른 스레드에 종료 신호를 보내는 `terminate` 속성을 설정합니다.

Example packages/123456789012-DEBUG_SERVER-1.0/application.py – 중지 메서드

```
    # 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)
```

인바운드 트래픽 지원

애플리케이션 코드와 함께 HTTP 서버를 실행하여 로컬에서 애플리케이션을 모니터링하거나 디버깅할 수 있습니다. 외부 트래픽을 처리하려면 AWS Panorama 어플라이언스의 포트를 애플리케이션 컨테이너의 포트에 매핑합니다.

> **Important**
>
> 기본적으로 AWS Panorama 어플라이언스는 어떤 포트에서도 들어오는 트래픽을 수락하지 않습니다. 어플라이언스에서 포트를 열면 보안 위험의 가능성이 발생합니다. 이 기능을 사용할 때는 외부 트래픽으로부터 어플라이언스를 보호하고 인증된 클라이언트와 어플라이언스 간의 통신을 보호하기 위한 추가 조치를 취해야 합니다.
> 이 가이드에 포함된 샘플 코드는 데모용이며 인증, 권한 부여 또는 암호화를 구현하지 않습니다.

어플라이언스에서 8000~9000 범위의 포트를 열 수 있습니다. 이러한 포트를 열면 모든 라우팅 가능한 클라이언트로부터 트래픽을 수신할 수 있습니다. 애플리케이션을 배포할 때 열려는 포트를 지정하고 어플라이언스의 포트를 애플리케이션 컨테이너의 포트에 매핑합니다. 어플라이언스 소프트웨어는 트래픽을 컨테이너로 전달하고 다시 요청자에게 응답을 보냅니다. 지정한 어플라이언스 포트에서 요청이 수신되고 응답은 임의의 임시 포트로 전송됩니다.

인바운드 포트 구성

애플리케이션 구성의 세 위치에서 포트 매핑을 지정합니다. 코드 패키지가 `package.json`이면 `network` 블록에서 코드 노드가 수신 대기하는 포트를 지정합니다. 다음 예시에서는 노드가 포트 80에서 수신한다고 선언합니다.

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"
                }
            ]
        }
```

애플리케이션 매니페스트에서 어플라이언스의 포트를 애플리케이션 코드 컨테이너의 포트에 매핑하는 라우팅 규칙을 선언합니다. 다음 예시에서는 디바이스의 포트 8080을 code_node 컨테이너의 포트 80에 매핑하는 규칙을 추가합니다.

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."
            }
        }
    ]
```

애플리케이션을 배포할 때 AWS Panorama 콘솔에서 또는 CreateApplicationInstance API에 전달된 오버라이드 문서를 사용하여 동일한 규칙을 지정합니다. 배포 시 이 구성을 제공하여 어플라이언스의 포트를 열 것인지 확인해야 합니다.

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"
    }
}
```

애플리케이션 매니페스트에 지정된 디바이스 포트를 다른 애플리케이션에서 사용 중인 경우 오버라이드 문서를 사용하여 다른 포트를 선택할 수 있습니다.

트래픽 처리

컨테이너의 포트가 열려 있으면 소켓을 열거나 서버를 실행하여 들어오는 요청을 처리할 수 있습니다. 이 `debug-server` 샘플은 컴퓨터 비전 애플리케이션 코드와 함께 실행되는 HTTP 서버의 기본 구현을 보여줍니다.

> **Important**
>
> 샘플 구현은 프로덕션 용도로 사용하기에 안전하지 않습니다. 어플라이언스를 공격에 취약하지 않게 만들려면 코드 및 네트워크 구성에 적절한 보안 제어를 구현해야 합니다.

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

```
    # 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.')
```

서버는 /status 경로에서 GET 요청을 수락하여 애플리케이션에 대한 일부 정보를 검색합니다. 또한 애플리케이션을 다시 시작하기 위해 /restart에 대한 POST 요청을 수락합니다.

이 기능을 시연하기 위해 샘플 애플리케이션은 별도의 스레드에서 HTTP 클라이언트를 실행합니다. 클라이언트는 시작 직후 로컬 네트워크를 통해 /status 경로를 호출하고 몇 분 후에 애플리케이션을 다시 시작합니다.

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

```
    # 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")
```

메인 루프는 스레드를 관리하고 스레드가 종료되면 애플리케이션을 다시 시작합니다.

Example packages/123456789012-DEBUG_SERVER-1.0/application.py – 메인 루프

```
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.')
```

샘플 애플리케이션을 배포하려면 이 설명서의 GitHub 리포지토리에서 지침을 참조하십시오.

GPU 사용하기

AWS Panorama 어플라이언스의 그래픽 프로세서(GPU)에 액세스하여 GPU 가속 라이브러리를 사용하거나 애플리케이션 코드에서 기계 학습 모델을 실행할 수 있습니다. GPU 액세스를 활성화하려면 애플리케이션 코드 컨테이너를 빌드한 후 패키지 구성에 GPU 액세스를 요구 사항으로 추가합니다.

> **Important**
>
> GPU 액세스를 활성화하면 어플라이언스의 어떤 애플리케이션에서도 모델 노드를 실행할 수 없습니다. 보안을 위해 어플라이언스가 SageMaker Neo로 컴파일된 모델을 실행할 때는 GPU 액세스가 제한됩니다. GPU 액세스를 사용하면 애플리케이션 코드 노드에서 모델을 실행해야 하며 디바이스의 모든 애플리케이션이 GPU에 대한 액세스 권한을 공유해야 합니다.

애플리케이션에 대한 GPU 액세스를 활성화하려면 AWS Panorama Application CLI로 패키지를 빌드한 후 패키지 구성을 업데이트하십시오. 다음 예시는 애플리케이션 코드 노드에 GPU 액세스를 추가하는 `requirements` 블록을 보여줍니다.

Example 요구 사항 블록이 포함된 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": [
    ...
```

개발 워크플로의 빌드와 패키징 단계 사이의 패키지 구성을 업데이트하십시오.

GPU 액세스를 사용하여 애플리케이션을 배포하려면

1. 애플리케이션 컨테이너를 빌드하려면 `build-container` 명령을 사용하십시오.

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

2. `requirements` 블록을 패키지 구성에 추가합니다.
3. 컨테이너 자산 및 패키지 구성을 업로드하려면 `package-application` 명령을 사용하십시오.

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

4. 애플리케이션을 배포합니다.

GPU 액세스를 사용하는 샘플 애플리케이션을 보려면 aws-panorama-samples GitHub 리포지토리를 방문하십시오.

Windows에서 개발 환경 설정

AWS Panorama 애플리케이션을 빌드하려면 Docker, 명령줄 도구, Python을 사용합니다. Windows에서는 Linux 및 Ubuntu용 Windows Subsystem과 함께 Docker Desktop을 사용하여 개발 환경을 설정할 수 있습니다. 이 자습서에서는 AWS Panorama 도구 및 샘플 애플리케이션으로 테스트된 개발 환경의 설정 프로세스를 안내합니다.

섹션

필수 조건

이 자습서를 따르려면 Windows Subsystem for Linux 2(WSL 2)를 지원하는 Windows 버전이 필요합니다.

- Windows 10 버전 1903 이상(빌드 18362 이상) 또는 Windows 11
- Windows 기능
  - Windows Subsystem for Linux
  - Hyper-V
  - 가상 머신 플랫폼

이 자습서는 다음 소프트웨어 버전으로 개발되었습니다.

- Ubuntu 20.04
- Python 3.8.5
- Docker 20.10.8

WSL 2 및 Ubuntu 설치

Windows 10 버전 2004 이상(빌드 19041 이상)을 사용하는 경우 다음 PowerShell 명령을 사용하여 WSL 2와 Ubuntu 20.04를 설치할 수 있습니다.

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

이전 Windows 버전의 경우 WSL 2 설명서의 이전 버전을 위한 수동 설치 단계 지침을 따르십시오.

Docker 설치

Docker Desktop을 설치하려면 hub.docker.com에서 설치 프로그램 패키지를 다운로드하여 실행하십시오. 문제가 발생하는 경우 Docker 웹 사이트 Docker Desktop WSL 2 backend의 지침을 따르십시오.

Docker Desktop을 실행하고 첫 실행 자습서에 따라 예시 컨테이너를 빌드하십시오.

> **Note**
>
> Docker Desktop은 기본 배포에서만 Docker를 활성화합니다. 이 자습서를 실행하기 전에 다른 Linux 배포판을 설치한 경우 리소스, WSL 통합의 Docker Desktop 설정 메뉴에서 새로 설치한 Ubuntu 배포판에서 Docker를 활성화하십시오.

Ubuntu 구성

이제 Ubuntu 가상 머신에서 Docker 명령을 실행할 수 있습니다. 명령줄 터미널을 열려면 시작 메뉴에서 배포를 실행하십시오. 처음 실행할 때는 관리자 명령을 실행하는 데 사용할 수 있는 사용자 이름과 암호를 구성합니다.

개발 환경 구성을 완료하려면 가상 컴퓨터의 소프트웨어를 업데이트하고 도구를 설치하십시오.

가상 컴퓨터를 구성하려면

1. Ubuntu와 함께 제공되는 소프트웨어를 업데이트하십시오.

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

2. apt를 사용하여 개발 도구를 설치합니다.

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

3. pip를 사용하여 Python 라이브러리를 설치합니다.

```
$ pip3 install awscli panoramacli
```

4. 새 터미널을 열고 `aws configure`를 실행하여 AWS CLI를 구성합니다.

```
$ aws configure
```

액세스 키가 없는 경우 IAM 콘솔에서 생성할 수 있습니다.

마지막으로 샘플 애플리케이션을 다운로드하여 가져옵니다.

샘플 애플리케이션을 가져오려면

1. 샘플 애플리케이션을 다운로드하여 압축을 풉니다.

```
$ 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. 포함된 스크립트를 실행하여 컴파일을 테스트하고, 애플리케이션 컨테이너를 구축하고, 패키지를 AWS Panorama에 업로드합니다.

```
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 Application CLI는 패키지를 업로드하고 AWS Panorama 서비스에 등록합니다. 이제 AWS Panorama 콘솔로 샘플 앱을 배포할 수 있습니다.

다음 단계

프로젝트 파일을 탐색하고 편집하려면 파일 탐색기 또는 WSL을 지원하는 통합 개발 환경(IDE)을 사용할 수 있습니다.

가상 머신의 파일 시스템에 액세스하려면 파일 탐색기를 열고 탐색 표시줄에 `\\wsl$`를 입력합니다. 이 디렉토리에는 가상 머신의 파일 시스템(`Ubuntu-20.04`) 및 Docker 데이터의 파일 시스템에 대한 링크가 포함되어 있습니다. `Ubuntu-20.04` 아래의 home*username*에 사용자 디렉토리가 있습니다.

> **Note**
>
> Ubuntu 내에서 Windows 설치에 있는 파일에 액세스하려면 해당 `/mnt/c` 디렉토리로 이동하십시오. 예를 들어 `ls /mnt/c/Users/`*windows-username*`/Downloads`를 실행하여 다운로드 디렉토리의 파일을 나열할 수 있습니다.

Visual Studio Code를 사용하면 개발 환경에서 애플리케이션 코드를 편집하고 통합 터미널에서 명령을 실행할 수 있습니다. Visual Studio Code를 설치하려면 code.visualstudio.com을 방문하십시오. 설치 후 원격 WSL 확장 프로그램을 추가합니다.

Windows 터미널은 명령을 실행하던 표준 Ubuntu 터미널의 대안입니다. 여러 탭을 지원하며 설치하는 다른 다양한 Linux에 대해 PowerShell, 명령 프롬프트 및 터미널을 실행할 수 있습니다. Ctrl+C 및 Ctrl+V를 사용한 복사 및 붙여넣기, 클릭 가능한 URL 및 기타 유용한 개선 사항을 지원합니다. Windows 터미널을 설치하려면 microsoft.com을 방문하십시오.

AWS Panorama API

AWS Panorama 서비스의 퍼블릭 API를 사용하여 디바이스 및 애플리케이션 관리 워크플로를 자동화할 수 있습니다. AWS Command Line Interface 또는 AWS SDK를 사용하여 리소스 및 배포를 관리하는 스크립트 또는 애플리케이션을 개발할 수 있습니다. 이 설명서의 GitHub 리포지토리에는 자체 코드의 시작점으로 사용할 수 있는 스크립트가 들어 있습니다.

- aws-panorama-developer-guide/util-scripts

단원

- 디바이스 등록 자동화
- AWS Panorama API로 어플라이언스 관리
- 애플리케이션 배포 자동화
- AWS Panorama API로 어플라이언스 관리
- VPC 엔드포인트 사용

디바이스 등록 자동화

어플라이언스를 프로비저닝하려면 ProvisionDevice API를 사용하세요. 응답에는 장치의 구성 및 임시 보안 인증 정보이 포함된 ZIP 파일이 포함됩니다. 파일을 디코딩하고 접두사 `certificates-omni_`를 사용하여 아카이브에 저장합니다.

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}"
```

구성 아카이브의 보안 인증 정보은 5분 후에 만료됩니다. 포함된 USB 드라이브를 사용하여 아카이브를 어플라이언스로 전송하십시오.

카메라를 등록하려면 CreateNodeFromTemplateJob API를 사용하십시오. 이 API는 카메라의 사용자 이름, 암호, URL에 대한 템플릿 파라미터 맵을 사용합니다. Bash 문자열 조작을 사용하여 이 맵을 JSON 문서로 포맷할 수 있습니다.

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)
```

또는 파일에서 JSON 구성을 로드할 수 있습니다.

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

AWS Panorama API로 어플라이언스 관리

AWS Panorama API를 사용하여 어플라이언스 관리 작업을 자동화할 수 있습니다.

디바이스 보기

디바이스 ID를 사용하여 디바이스 목록을 가져오려면 ListDevices API를 사용하세요.

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

어플라이언스에 대한 자세한 내용을 보려면 DescribeDevice API를 사용하세요.

```
$ 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
}
```

어플라이언스 소프트웨어 업그레이드

`LatestSoftware` 버전이 `CurrentSoftware`보다 최신인 경우 장치를 업그레이드할 수 있습니다. CreateJobForDevices API를 사용하여 무선 업데이트(OTA) 업데이트 작업을 만들 수 있습니다.

```
$ 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"
        }
    ]
}
```

스크립트에서 Bash 문자열 조작으로 작업 구성 파일의 이미지 버전 필드를 채울 수 있습니다.

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
}
```

어플라이언스가 지정된 소프트웨어 버전을 다운로드하고 자체적으로 업데이트합니다. DescribeDeviceJob API를 사용하여 업데이트 진행 상황을 확인하십시오.

```
$ 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
}
```

실행 중인 모든 작업의 목록을 가져오려면 ListDevicesJobs를 사용하십시오.

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

업데이트를 확인하고 적용하는 샘플 스크립트는 이 설명서의 GitHub 리포지토리에서 check-updates.sh를 참조하십시오.

어플라이언스 재부팅

어플라이언스를 재부팅하려면 CreateJobForDevices API를 사용하세요.

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

```
    ]
}
```

스크립트에서 장치 목록을 가져오고 대화형 방식으로 재부팅할 장치를 선택할 수 있습니다.

Example reboot-device.sh – 사용법

```
$ ./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"
        }
    ]
}
```

애플리케이션 배포 자동화

애플리케이션을 배포하려면 AWS Panorama 애플리케이션 CLI와 AWS Command Line Interface을 모두 사용합니다. 애플리케이션 컨테이너를 구축한 후 Amazon S3 액세스 포인트에 애플리케이션 및 기타 자산을 업로드합니다. 그러면 CreateApplicationInstance API를 사용하여 애플리케이션 배포할 수 있습니다.

표시된 스크립트 사용에 대한 자세한 컨텍스트 및 지침은 샘플 애플리케이션 README의 지침을 따르십시오.

섹션

- 컨테이너 빌드
- 컨테이너 업로드 및 노드 등록
- 애플리케이션 배포
- 배포 모니터링

컨테이너 빌드

애플리케이션 컨테이너를 빌드하려면 `build-container` 명령을 사용하십시오. 이 명령은 Docker 컨테이너를 빌드하고 `assets` 폴더에 압축된 파일 시스템으로 저장합니다.

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
```

명령줄 완성 기능을 사용하여 경로의 일부를 입력한 다음 TAB을 눌러 경로 인수를 채울 수도 있습니다.

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

컨테이너 업로드 및 노드 등록

애플리케이션을 업로드하려면 `package-application` 명령을 사용하십시오. 이 명령은 `assets` 폴더의 자산을 AWS Panorama가 관리하는 Amazon S3 액세스 포인트에 업로드합니다.

Example 4-package-app.sh

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

AWS Panorama Application CLI는 각 패키지의 패키지 구성(package.json)에서 참조하는 컨테이너 및 설명자 자산을 업로드하고 패키지를 AWS Panorama의 노드로 등록합니다. 그런 다음 애플리케이션 매니페스트(graph.json)에서 이러한 노드를 참조하여 애플리케이션을 배포합니다.

애플리케이션 배포

애플리케이션을 배포하려면 CreateApplicationInstance API를 사용하십시오. 이 작업에는 특히 다음과 같은 파라미터가 사용됩니다.

- ManifestPayload – 애플리케이션의 노드, 패키지, 엣지 및 파라미터를 정의하는 애플리케이션 매니페스트(graph.json)입니다.
- ManifestOverridesPayload – 첫 번째 매니페스트의 파라미터를 재정의하는 두 번째 매니페스트입니다. 애플리케이션 매니페스트는 애플리케이션 소스의 정적 리소스로 간주될 수 있으며, 오버라이드 매니페스트는 배포를 사용자 지정하는 배포 시간 설정을 제공합니다.
- DefaultRuntimeContextDevice – 대상 디바이스입니다.
- RuntimeRoleArn – AWS 서비스 및 리소스에 액세스하기 위해 애플리케이션이 사용하는 IAM 역할의 ARN입니다.
- ApplicationInstanceIdToReplace – 디바이스에서 제거할 기존 애플리케이션 인스턴스의 ID입니다.

매니페스트 및 오버라이드 페이로드는 다른 문서 내에 중첩된 문자열 값으로 제공되어야 하는 JSON 문서입니다. 이를 위해 스크립트는 파일에서 매니페스트를 문자열로 로드하고 jq 도구를 사용하여 중첩 문서를 생성합니다.

Example 5-deploy.sh – 매니페스트 작성

```
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)"' <<< {})"
```

배포 스크립트는 ListDevices API를 사용하여 현재 리전에 등록된 디바이스 목록을 가져오고, 이후 배포를 위해 사용자 선택 항목을 로컬 파일에 저장합니다.

Example 5-deploy.sh – 디바이스 찾기

```
    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)
```

애플리케이션 역할은 다른 스크립트(1-create-role.sh)에 의해 생성됩니다. 배포 스크립트는 이 역할의 ARN을 AWS CloudFormation에서 가져옵니다. 애플리케이션이 이미 디바이스에 배포된 경우 스크립트는 로컬 파일에서 해당 애플리케이션 인스턴스의 ID를 가져옵니다.

Example 5-deploy.sh – 역할 ARN 및 대체 인수

```
# 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
```

마지막으로 스크립트는 모든 요소를 조합하여 애플리케이션 인스턴스를 만들고 애플리케이션을 디바이스에 배포합니다. 서비스는 스크립트가 나중에 사용할 수 있도록 저장하는 인스턴스 ID로 응답합니다.

Example 5-deploy.sh – 애플리케이션 배포

```
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
```

배포 모니터링

배포를 모니터링하려면 ListApplicationInstances API를 사용하십시오. 모니터 스크립트는 애플리케이션 디렉토리의 파일에서 디바이스 ID 및 애플리케이션 인스턴스 ID를 가져오고 이를 사용하여 CLI 명령을 생성합니다. 그런 다음 루프에서 호출합니다.

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
```

배포가 완료되면 Amazon CloudWatch Logs API를 호출하여 로그를 볼 수 있습니다. 로그 보기 스크립트는 CloudWatch Logs `GetLogEvents` API를 사용합니다.

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
```

AWS Panorama API로 어플라이언스 관리

AWS Panorama API를 사용하여 애플리케이션을 모니터링하고 관리할 수 있습니다.

애플리케이션 보기

어플라이언스에서 실행 중인 애플리케이션 목록을 가져오려면 ListApplicationInstances API를 사용하십시오.

```
$ 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"
            }
        },
    ]
}
```

애플리케이션 인스턴스의 노드에 대한 자세한 내용을 보려면 ListApplicationInstanceNodeInstances API를 사용하십시오.

```
$ 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"
        }
    ]
}
```

카메라 스트림 관리

SignalApplicationInstanceNodeInstances API를 사용하여 카메라 스트림 노드를 일시 중지하고 재개할 수 있습니다.

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

스크립트에서 노드 목록을 가져와서 대화형 방식으로 일시 중지하거나 재개할 노드를 선택하면 됩니다.

Example pause-camera.sh – 사용법

```
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"
}
```

카메라 노드를 일시 중지했다가 다시 시작하면 동시에 처리할 수 있는 것보다 더 많은 수의 카메라 스트림을 순환할 수 있습니다. 이렇게 하려면 여러 카메라 스트림을 오버라이드 매니페스트의 동일한 입력 노드에 매핑하십시오.

다음 예시에서는 오버라이드 매니페스트가 `warehouse-floor`와 `entrance-north`라는 두 개의 카메라 스트림을 동일한 입력 노드(`camera_node`)에 매핑합니다. `warehouse-floor` 스트림은 애플리케이션이 시작되고 `entrance-north` 노드가 신호가 켜질 때까지 기다리면 활성화됩니다.

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"
                    }
                ]
            }
```

API를 사용한 배포에 대한 자세한 내용은 애플리케이션 배포 자동화 단원을 참조하십시오.

VPC 엔드포인트 사용

인터넷에 액세스할 수 없는 VPC에서 작업하는 경우, AWS Panorama와 함께 사용할 VPC 엔드포인트를 생성할 수 있습니다. VPC 엔드포인트를 사용하면 프라이빗 서브넷에서 실행되는 클라이언트가 인터넷 연결 없이 AWS 서비스에 연결할 수 있습니다.

AWS Panorama 어플라이언스에서 사용하는 포트 및 엔드포인트에 대한 자세한 내용은 ??? 단원을 참조하십시오.

섹션

- VPC 엔드포인트 생성
- 어플라이언스를 프라이빗 서브넷에 연결
- 샘플 AWS CloudFormation 템플릿

VPC 엔드포인트 생성

VPC와 AWS Panorama 간에 비공개 연결을 설정하려면 VPC 엔드포인트를 생성하십시오. AWS Panorama를 사용하는 데 VPC 엔드포인트는 필요하지 않습니다. 인터넷에 액세스할 수 없는 VPC에서 작업하는 경우에만 VPC 엔드포인트를 생성하면 됩니다. AWS CLI 또는 SDK가 AWS Panorama에 연결을 시도하면 트래픽이 VPC 엔드포인트를 통해 라우팅됩니다.

다음 설정을 사용하여 AWS Panorama용 VPC 엔드포인트를 생성하십시오.

- 서비스 이름 – **`com.amazonaws.us-west-2.panorama`**
- 유형 – 인터페이스

VPC 엔드포인트는 추가 구성 없이 서비스의 DNS 이름을 사용하여 AWS SDK 클라이언트로부터 트래픽을 가져옵니다. VPC 엔드포인트 사용에 대한 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 참조하십시오.

어플라이언스를 프라이빗 서브넷에 연결

AWS Panorama 어플라이언스는 AWS Site-to-Site VPN 또는 AWS Direct Connect와의 프라이빗 VPN 연결을 통해 AWS에 연결할 수 있습니다. 이러한 서비스를 사용하여 데이터 센터까지 확장되는 프라이빗 서브넷을 생성할 수 있습니다. 어플라이언스는 프라이빗 서브넷에 연결하고 VPC 엔드포인트를 통해 AWS 서비스에 액세스합니다.

Site-to-Site VPN과 AWS Direct Connect는 데이터 센터를 Amazon VPC에 안전하게 연결하기 위한 서비스입니다. Site-to-Site VPN을 사용하면 상용 네트워크 장치를 사용하여 연결할 수 있습니다. AWS Direct Connect는 AWS 디바이스를 사용하여 연결합니다.

- Site-to-Site VPN – AWS Site-to-Site VPN란?
- AWS Direct Connect – AWS Direct Connect란?

로컬 네트워크를 VPC의 프라이빗 서브넷에 연결한 후 다음 서비스를 위한 VPC 엔드포인트를 생성합니다.

- Amazon Simple Storage Service – Amazon S3용 AWS PrivateLink
- AWS IoT Core – 인터페이스 VPC 엔드포인트와 함께 AWS IoT Core 사용(데이터 영역 및 보안 인증 정보 공급자)
- Amazon Elastic 컨테이너 레지스트리 – Amazon Elastic 컨테이너 레지스트리 인터페이스 VPC 엔드포인트
- Amazon CloudWatch – 인터페이스 VPC 엔드포인트와 함께 CloudWatch 사용
- Amazon CloudWatch Logs – 인터페이스 VPC 엔드포인트와 함께 CloudWatch 로그 사용

어플라이언스는 AWS Panorama 서비스에 연결할 필요가 없습니다. AWS IoT의 메시징 채널을 통해 AWS Panorama와 통신합니다.

VPC 엔드포인트 외에도 Amazon S3 및 AWS IoT를 사용하려면 Amazon Route 53 프라이빗 호스팅 영역을 사용해야 합니다. 프라이빗 호스팅 영역은 Amazon S3 액세스 포인트의 하위 도메인 및 MQTT 항목을 비롯한 하위 도메인의 트래픽을 올바른 VPC 엔드포인트로 라우팅합니다. 프라이빗 호스팅 영역에 대한 자세한 내용은 Amazon Route 53 개발자 가이드에서 프라이빗 호스팅 영역으로 작업하기를 참조하십시오.

VPC 엔드포인트와 프라이빗 호스팅 영역을 포함하는 샘플 VPC 구성은 샘플 AWS CloudFormation 템플릿 단원을 참조하십시오.

샘플 AWS CloudFormation 템플릿

이 설명서의 GitHub 리포지토리에서는 AWS Panorama에 사용할 리소스를 만드는 데 사용할 수 있는 AWS CloudFormation 템플릿을 제공합니다. 템플릿은 두 개의 프라이빗 서브넷, 퍼블릭 서브넷, VPC 엔드포인트가 있는 VPC를 생성합니다. VPC의 프라이빗 서브넷을 사용하여 인터넷으로부터 격리된

리소스를 호스팅할 수 있습니다. 퍼블릭 서브넷의 리소스는 프라이빗 리소스와 통신할 수 있지만, 프라이빗 리소스는 인터넷에서 액세스할 수 없습니다.

Example vpc-endpoint.yml – 프라이빗 서브넷

```
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` 템플릿에서는 AWS Panorama에 대한 VPC 엔드포인트를 생성하는 방법을 보여줍니다. 이 엔드포인트를 사용하여 AWS SDK 또는 AWS CLI로 AWS Panorama 리소스를 관리할 수 있습니다.

Example vpc-endpoint.yml – VPC 엔드포인트

```
  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:
            - "*"
```

PolicyDocument는 엔드포인트에서 수행할 수 있는 API 호출을 정의하는 리소스 기반 권한 정책입니다. 정책을 수정하여 엔드포인트를 통해 액세스할 수 있는 작업 및 리소스를 제한할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하십시오.

vpc-appliance.yml 템플릿은 AWS Panorama 어플라이언스에서 사용하는 서비스를 위한 VPC 엔드포인트와 프라이빗 호스팅 영역을 생성하는 방법을 보여줍니다.

Example vpc-appliance.yml – 프라이빗 호스팅 영역이 있는 Amazon S3 액세스 포인트 엔드포인트

```
  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 ] ] ]
```

샘플 템플릿에서는 샘플 VPC를 사용하여 Amazon VPC 및 Route 53 리소스의 생성을 보여줍니다. VPC 리소스를 제거하고 서브넷, 보안 그룹 및 VPC ID에 대한 참조를 리소스의 ID로 대체하여 사용 사례에 맞게 조정할 수 있습니다.

샘플 애플리케이션, 스크립트, 템플릿

이 설명서의 GitHub 리포지토리는 AWS Panorama 디바이스용 샘플 애플리케이션, 스크립트 및 템플릿을 제공합니다. 이러한 샘플을 사용하여 모범 사례를 알아보고 개발 워크플로우를 자동화하십시오.

섹션

샘플 애플리케이션

샘플 애플리케이션은 AWS Panorama 기능 사용법 및 일반적인 컴퓨터 비전 작업을 보여줍니다. 이러한 샘플 애플리케이션에는 설정 및 배포를 자동화하는 스크립트와 템플릿이 포함되어 있습니다. 최소 구성으로 명령줄에서 애플리케이션을 배포하고 업데이트할 수 있습니다.

- aws-panorama-sample – 분류 모델을 사용한 기본 컴퓨터 비전입니다. AWS SDK for Python (Boto)를 사용하여 CloudWatch에 지표를 업로드하고, 계측기 사전 처리 및 추론 방법을 사용하고, 로깅을 구성합니다.
- 디버그 서버 - 디바이스의 인바운드 포트를 열고 트래픽을 애플리케이션 코드 컨테이너로 전달합니다. 멀티스레딩을 사용하여 애플리케이션 코드, HTTP 서버 및 HTTP 클라이언트를 동시에 실행할 수 있습니다.
- 사용자 지정 모델 – 코드에서 모델을 내보내고 SageMaker Neo로 컴파일하여 AWS Panorama 어플라이언스와의 호환성을 테스트할 수 있습니다. Python 개발, Docker 컨테이너 또는 Amazon EC2 인스턴스에서 로컬로 빌드합니다. 특정 TensorFlow 또는 Python 버전용 Keras의 내장된 모든 애플리케이션 모델을 내보내고 컴파일합니다.

더 많은 샘플 애플리케이션을 보려면 aws-panorama-samples 리포지토리를 방문하십시오.

유틸리티 스크립트

`util-scripts` 디렉토리의 스크립트는 AWS Panorama 리소스를 관리하거나 개발 워크플로우를 자동화합니다.

- provision-device.sh – 디바이스를 프로비저닝합니다.
- check-updates.sh – 어플라이언스 소프트웨어 업데이트를 확인하고 적용합니다.
- reboot-device.sh – 디바이스를 재부팅합니다.
- register-camera.sh – 카메라를 등록합니다.
- deregister-camera.sh – 카메라 노드를 삭제합니다.
- view-logs.sh – 애플리케이션 인스턴스의 로그를 볼 수 있습니다.
- pause-camera.sh – 카메라 스트림을 일시 중지하거나 재개합니다.
- push.sh – 애플리케이션을 빌드, 업로드, 배포합니다.
- rename-package.sh – 노드 패키지 이름을 바꿉니다. 디렉토리 이름, 구성 파일 및 애플리케이션 매니페스트를 업데이트합니다.
- samplify.sh – 계정 ID를 예제 계정 ID로 바꾸고 백업 구성을 복원하여 로컬 구성을 제거합니다.
- update-model-config.sh – 설명자 파일을 업데이트한 후 모델을 애플리케이션에 다시 추가합니다.
- cleanup-patches.sh – 이전 패치 버전의 등록을 취소하고 Amazon S3에서 해당 매니페스트를 삭제합니다.

사용 세부 정보는 README를 참조하십시오.

AWS CloudFormation 템플릿

`cloudformation-templates` 디렉토리의 AWS CloudFormation 템플릿을 사용하여 AWS Panorama 애플리케이션용 리소스를 생성하십시오.

- alarm-application.yml – 애플리케이션의 오류를 모니터링하는 경보를 생성합니다. 애플리케이션 인스턴스에서 오류가 발생하거나 5분 동안 실행이 중지되면 경보가 알림 이메일을 보냅니다.
- alarm-device.yml – 디바이스의 연결을 모니터링하는 경보를 생성합니다. 디바이스가 5분 동안 메트릭 전송을 중단하면 경보가 알림 이메일을 보냅니다.
- application-role.yml – 애플리케이션 역할을 생성합니다. 이 역할에는 CloudWatch로 지표를 전송할 수 있는 권한이 포함됩니다. 애플리케이션에서 사용하는 다른 API 작업에 대한 권한을 정책 설명에 추가하십시오.
- vpc-appliance.yml – AWS Panorama 어플라이언스에 대한 프라이빗 서브넷 서비스 액세스 권한이 있는 VPC를 생성합니다. 어플라이언스를 VPC에 연결하려면 AWS Direct Connect 또는 AWS Site-to-Site VPN를 사용하십시오

- vpc-endpoint.yml – AWS Panorama 서비스에 대한 프라이빗 서브넷 서비스 액세스 권한이 있는 VPC를 생성합니다. VPC 내부의 리소스를 AWS Panorama에 연결하여 인터넷에 연결하지 않고도 AWS Panorama 리소스를 모니터링하고 관리할 수 있습니다.

이 디렉토리의 `create-stack.sh` 스크립트는 AWS CloudFormation 스택을 생성합니다. 다양한 수의 인수를 사용합니다. 첫 번째 인수는 템플릿 이름이고 나머지 인수는 템플릿의 파라미터에 대한 오버라이드입니다.

예를 들어 다음 명령은 애플리케이션 역할을 생성합니다.

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

더 많은 샘플과 도구

aws-panorama-samples 리포지토리에는 더 많은 샘플 애플리케이션과 유용한 도구가 있습니다.

- 애플리케이션 – 다양한 모델 아키텍처 및 사용 사례를 위한 샘플 애플리케이션입니다.
- 카메라 스트림 검증 – 카메라 스트림을 검증합니다.
- PanoJupyter – AWS Panorama 어플라이언스에서 upyterLab을 실행합니다.
- Sideloading – 애플리케이션 컨테이너를 구축하거나 배포하지 않고도 애플리케이션 코드를 업데이트할 수 있습니다.

AWS 커뮤니티에서는 AWS Panorama을 위한 도구와 지침도 개발했습니다. GitHub에서 다음 오픈소스 프로젝트를 확인해 보십시오.

- cookiecutter-panorama – AWS Panorama 애플리케이션을 위한 Cookiecutter 템플릿입니다.
- 백팩 — 런타임 환경 세부 정보, 프로파일링 및 추가 비디오 출력 옵션에 액세스하기 위한 Python 모듈입니다.

AWS Panorama 리소스 및 애플리케이션 모니터링

AWS Panorama 콘솔과 Amazon CloudWatch에서 AWS Panorama 리소스를 모니터링할 수 있습니다. AWS Panorama 어플라이언스는 인터넷을 통해 AWS 클라우드에 연결하여 상태 및 연결된 카메라의 상태를 보고합니다. 어플라이언스는 켜져 있는 동안 실시간으로 로그를 CloudWatch Logs로 전송하기도 합니다.

어플라이언스는 AWS Panorama 콘솔을 처음 사용할 때 생성하는 서비스 역할에서 AWS IoT, CloudWatch 로그 및 기타 AWS 서비스를 사용할 수 있는 권한을 얻습니다. 자세한 내용은 AWS Panorama 서비스 역할 및 교차 서비스 리소스 단원을 참조하십시오.

특정 오류를 해결하는 데 도움이 필요하면 문제 해결 단원을 참조하십시오.

주제

- AWS Panorama 콘솔에서의 모니터링
- AWS Panorama 로그 보기
- Amazon CloudWatch를 사용한 어플라이언스 및 애플리케이션 모니터링

AWS Panorama 콘솔에서의 모니터링

AWS Panorama 콘솔을 사용하여 AWS Panorama 어플라이언스 및 카메라를 모니터링할 수 있습니다. 콘솔은 AWS IoT를 사용하여 어플라이언스의 상태를 모니터링합니다.

AWS Panorama 콘솔에서 어플라이언스를 모니터링하려면

1. AWS Panorama 콘솔을 엽니다.
2. AWS Panorama 콘솔 디바이스 페이지를 엽니다.
3. 어플라이언스를 선택합니다.
4. 애플리케이션 인스턴스의 상태를 보려면 목록에서 선택합니다.
5. 어플라이언스의 네트워크 인터페이스 상태를 보려면 설정을 선택합니다.

페이지 상단에 어플라이언스의 전체 상태가 표시됩니다. 상태가 온라인이면 어플라이언스가 AWS에 연결되어 정기적인 상태 업데이트를 전송하고 있는 것입니다.

AWS Panorama 로그 보기

AWS Panorama는 Amazon CloudWatch Logs에 애플리케이션 및 시스템 이벤트를 보고합니다. 문제가 발생하면 이벤트 로그를 사용하여 AWS Panorama 애플리케이션을 디버깅하거나 애플리케이션의 구성 문제를 해결할 수 있습니다.

CloudWatch Logs에서 로그를 보려면

1. CloudWatch 로그 콘솔의 로그 그룹 페이지를 엽니다.
2. 다음 그룹에서 AWS Panorama 애플리케이션 및 어플라이언스 로그를 확인할 수 있습니다.

   - 디바이스 로그 – `/aws/panorama/devices/device-id`
   - 애플리케이션 로그 – `/aws/panorama/devices/device-id/applications/instance-id`

시스템 소프트웨어를 업데이트한 후 어플라이언스를 다시 프로비전하면 프로비저닝 USB 드라이브의 로그도 확인할 수 있습니다.

섹션

디바이스 로그 보기

AWS Panorama 어플라이언스는 디바이스에 대한 로그 그룹과 배포하는 각 애플리케이션 인스턴스에 대한 그룹을 생성합니다. 디바이스 로그에는 애플리케이션 상태, 소프트웨어 업그레이드 및 시스템 구성에 대한 정보가 포함됩니다.

디바이스 로그 – **`/aws/panorama/devices/device-id`**

- `occ_log` – 컨트롤러 프로세스의 출력입니다. 이 프로세스는 애플리케이션 배포를 조정하고 각 애플리케이션 인스턴스의 노드 상태를 보고합니다.
- `ota_log` – 무선 업데이트(OTA) 소프트웨어 업그레이드를 조정하는 프로세스의 결과입니다.

- `syslog` – 프로세스 간에 전송되는 메시지를 캡처하는 디바이스의 syslog 프로세스의 출력입니다.
- `kern_log` – 디바이스의 Linux 커널에서 발생한 이벤트입니다.
- `logging_setup_logs` – CloudWatch Logs 에이전트를 구성하는 프로세스의 출력입니다.
- `cloudwatch_agent_logs` – CloudWatch Logs 에이전트의 출력입니다.
- `shadow_log` – AWS IoT 디바이스 섀도우의 출력입니다.

애플리케이션 로그 보기

애플리케이션 인스턴스의 로그 그룹에는 노드의 이름을 딴 각 노드에 대한 로그 스트림이 포함됩니다.

애플리케이션 로그 – **/aws/panorama/devices/*device-id*/applications/*instance-id***

- 코드 – 애플리케이션 코드와 AWS Panorama Application SDK의 출력입니다. `/opt/aws/panorama/logs`에서 애플리케이션 로그를 집계합니다.
- 모델 - 모델에 따라 추론 요청을 조정하는 프로세스의 출력입니다.
- 스트림 - 카메라 스트림의 비디오를 디코딩하는 프로세스의 출력입니다.
- 디스플레이 - HDMI 포트의 비디오 출력을 렌더링하는 프로세스의 출력입니다.
- `mds` – 어플라이언스 메타데이터 서버의 로그입니다.
- `console_output` – 코드 컨테이너에서 표준 출력 및 오류 스트림을 캡처합니다.

CloudWatch Logs에 로그가 표시되지 않는 경우 올바른 AWS 리전에 있는지 확인하십시오. 그렇다면 어플라이언스와 AWS의 연결 또는 어플라이언스 AWS Identity and Access Management(IAM) 역할에 대한 권한에 문제가 있을 수 있습니다.

애플리케이션 로그 구성

Python 로거를 구성하여 `/opt/aws/panorama/logs`에 로그 파일을 기록하십시오. 어플라이언스는 이 위치의 로그를 CloudWatch Logs로 스트리밍합니다. 디스크 공간을 너무 많이 사용하지 않으려면 최대 파일 크기 10MB, 백업 개수 1을 사용하십시오. 다음 예에서는 로거를 만드는 메서드를 보여줍니다.

Example application.py – 로거 구성

```
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
```

글로벌 범위에서 로거를 초기화하고 애플리케이션 코드 전체에서 사용하십시오.

Example application.py – 로거 초기화

```
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()
```

프로비저닝 로그 보기

프로비저닝 중에 AWS Panorama 어플라이언스는 구성 아카이브를 어플라이언스로 전송하는 데 사용하는 USB 드라이브에 로그를 복사합니다. 이러한 로그를 사용하여 최신 소프트웨어 버전을 사용하는 어플라이언스의 프로비저닝 문제를 해결할 수 있습니다.

> **Important**
>
> 프로비저닝 로그는 소프트웨어 버전 4.3.23 이상으로 업데이트된 어플라이언스에 사용할 수 있습니다.

애플리케이션 로그

- `/panorama/occ.log` – AWS Panorama 컨트롤러 소프트웨어 로그입니다.
- `/panorama/ota_agent.log` – AWS Panorama 무선 업데이트 에이전트 로그입니다.
- `/panorama/syslog.log` – Linux 시스템 로그입니다.
- `/panorama/kern.log` – Linux 커널 로그입니다.

디바이스에서 로그 전송

디바이스 및 애플리케이션 로그가 CloudWatch Logs에 표시되지 않는 경우 USB 드라이브를 사용하여 디바이스에서 암호화된 로그 이미지를 가져올 수 있습니다. AWS Panorama 서비스 팀이 사용자를 대신하여 로그를 해독하고 디버깅을 지원할 수 있습니다.

필수 조건

절차를 따르려면 다음 하드웨어가 필요합니다.

- USB 드라이브 – 최소 1GB의 저장 공간이 있는 FAT32 형식의 USB 플래시 메모리 드라이브로, AWS Panorama 어플라이언스에서 로그 파일을 전송할 때 사용합니다.

디바이스에서 로그를 전송하려면

1. `panorama` 폴더 안에 `managed_logs` 폴더가 있는 USB 드라이브를 준비합니다.

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

2. USB 드라이브를 디바이스에 연결합니다.
3. AWS Panorama 어플라이언스의 전원을 끕니다.
4. AWS Panorama 어플라이언스의 전원을 켭니다.
5. 디바이스가 로그를 디바이스에 복사합니다. 진행 중에는 상태 LED가 파란색으로 깜박입니다.
6. 그러면 `managed_logs` 디렉토리에서 `panorama_device_log_`*`v1_dd_hh_mm`*`.img` 형식의 로그 파일을 찾을 수 있습니다.

로그 이미지를 직접 해독할 수는 없습니다. 고객 지원팀, AWS Panorama의 기술 계정 관리자 또는 솔루션 아키텍트와 협력하여 서비스 팀과 조율하십시오.

Amazon CloudWatch를 사용한 어플라이언스 및 애플리케이션 모니터링

어플라이언스가 온라인 상태인 경우, AWS Panorama는 측정치를 Amazon CloudWatch로 전송합니다. CloudWatch 콘솔에서 이러한 지표를 통해 그래프와 대시보드를 작성하여 어플라이언스 활동을 모니터링하고, 디바이스가 오프라인이 되거나 애플리케이션에 오류가 발생할 경우 알려주는 경보를 설정할 수 있습니다.

CloudWatch 콘솔에서 지표를 보려면

1. AWS Panorama 콘솔의 지표 페이지(`PanoramaDeviceMetrics` 네임스페이스)를 엽니다.
2. 차원 스키마를 선택합니다.
3. 지표를 선택하여 그래프에 추가합니다.
4. 다른 통계를 선택하고 그래프를 사용자 지정하려면 그래프로 표시된 지표 탭의 옵션을 사용합니다. 기본적으로 그래프는 모든 지표에 대해 `Average` 통계를 사용합니다.

> **요금**
>
> CloudWatch에는 상시 무료 티어가 있습니다. 프리 티어 임계값 외에도 지표, 대시보드, 경보, 로그 및 인사이트에 대한 CloudWatch 요금이 부과됩니다. 자세한 내용은 CloudWatch 요금을 참조하십시오.

CloudWatch에 대한 자세한 정보는 Amazon CloudWatch 사용 설명서를 참조하십시오.

섹션

- 디바이스 지표 사용
- 애플리케이션 지표 사용
- 경보 구성

디바이스 지표 사용

어플라이언스가 온라인 상태인 경우, 측정치를 Amazon CloudWatch로 전송합니다. 이러한 지표를 사용하여 디바이스 활동을 모니터링하고 디바이스가 오프라인 상태가 되면 경보를 트리거할 수 있습니다.

- `DeviceActive` – 디바이스가 활성 상태일 때 주기적으로 전송됩니다.

  차원 – `DeviceId` 및`DeviceName`.

`Average` 통계와 함께 `DeviceActive` 지표를 확인합니다.

애플리케이션 지표 사용

애플리케이션에서 오류가 발생하면 지표를 Amazon CloudWatch로 전송합니다. 이러한 지표를 활용하여 애플리케이션 실행이 중지되는 경우 경보를 트리거할 수 있습니다.

- `ApplicationErrors` – 기록된 애플리케이션 오류 수입니다.

  차원 – `ApplicationInstanceName` 및`ApplicationInstanceId`.

`Sum` 통계와 함께 애플리케이션 지표를 확인합니다.

경보 구성

지표가 임계값을 초과할 때 알림을 받으려면 경보를 만드십시오. 예를 들어, `ApplicationErrors` 지표의 합계가 20분 동안 1로 유지될 때 알림을 보내는 경보를 만들 수 있습니다.

경보를 만들려면

1. Amazon CloudWatch 콘솔의 경보 페이지를 엽니다.
2. 경보 생성을 선택합니다.
3. 지표 선택을 선택하고 디바이스에 대한 지표(예: `applicationInstance-gk75xmplqbqtenlnmz4ehiu7xa,my-application`용 `ApplicationErrors`)를 찾습니다.
4. 지침에 따라 경보의 조건, 조치 및 이름을 구성하십시오.

구체적인 지침은 Amazon CloudWatch 사용 설명서의 CloudWatch 경보 생성을 참조하십시오.

문제 해결

다음 항목에서는 AWS Panorama 콘솔, 어플라이언스 또는 SDK를 사용할 때 발생할 수 있는 오류 및 문제에 대한 문제 해결 조언을 제공합니다. 여기에 나열되지 않은 문제를 발견하는 경우 이 페이지의 피드백 제공 버튼을 사용하여 해당 문제를 보고하세요.

Amazon CloudWatch Logs 콘솔에서 사용자의 어플라이언스에 대한 로그를 확인할 수 있습니다. 어플라이언스는 애플리케이션 코드, 어플라이언스 소프트웨어, AWS IoT 프로세스에서 생성되는 로그를 업로드합니다. 자세한 내용은 AWS Panorama 로그 보기 단원을 참조하세요.

프로비저닝

문제: (macOS) 컴퓨터가 USB-C 어댑터와 함께 제공된 USB 드라이브를 인식하지 못합니다.

컴퓨터에 이미 연결되어 있는 USB-C 어댑터에 USB 드라이브를 연결하면 이 문제가 발생할 수 있습니다. 어댑터를 분리했다가 이미 연결된 USB 드라이브에 다시 연결해 보십시오.

문제: 자체 USB 드라이브를 사용할 때 프로비저닝이 실패합니다.

문제: 어플라이언스의 USB 2.0 포트를 사용할 때 프로비저닝이 실패합니다.

AWS Panorama 어플라이언스는 1GB~32GB 사이의 USB 플래시 메모리 장치와 호환되지만 모든 장치가 호환되는 것은 아닙니다. 프로비저닝에 USB 2.0 포트를 사용할 때 몇 가지 문제가 확인되었습니다. 일관된 결과를 얻으려면 함께 제공된 USB 드라이브를 USB 3.0 포트(HDMI 포트 옆)에 사용하십시오.

Lenovo ThinkEdge® SE70의 경우 USB 드라이브는 어플라이언스에 포함되어 있지 않습니다. 최소 1GB의 저장 공간이 있는 USB 3.0 드라이브를 사용하십시오.

어플라이언스 구성

문제: 부팅 중에 어플라이언스에 빈 화면이 표시됩니다.

초기 부팅 시퀀스(약 1분 소요)를 완료한 후, 어플라이언스는 모델을 로드하고 애플리케이션을 시작하는 동안 1분 이상 빈 화면을 표시합니다. 또한 어플라이언스가 켜진 후 디스플레이를 연결하면 어플라이언스에서 비디오가 출력되지 않습니다.

문제: 전원 버튼을 눌러 전원을 끄면 어플라이언스가 반응하지 않습니다.

어플라이언스가 안전하게 종료되는 데는 최대 10초가 걸립니다. 종료 시퀀스를 시작하려면 전원 버튼을 1초만 누르고 있으면 됩니다. 버튼 작업의 전체 목록은 AWS Panorama 어플라이언스 버튼 및 표시등 단원을 참조하십시오.

문제: 설정을 변경하거나 분실된 인증서를 교체하려면 새 구성 아카이브를 생성해야 합니다.

AWS Panorama는 장치 인증서 또는 네트워크 구성을 다운로드한 후에는 이를 저장하지 않으며, 구성 아카이브를 재사용할 수 없습니다. AWS Panorama 콘솔을 사용하여 어플라이언스를 삭제한 후 새로운 구성 아카이브로 어플라이언스를 새로 생성합니다.

애플리케이션 구성

문제: 여러 애플리케이션을 실행할 때 HDMI 출력을 사용하는 애플리케이션을 제어할 수 없습니다.

출력 노드가 있는 여러 애플리케이션을 배포하는 경우 가장 최근에 시작된 애플리케이션이 HDMI 출력을 사용합니다. 이 애플리케이션의 실행이 중지되면 다른 애플리케이션이 출력을 사용할 수 있습니다. 한 애플리케이션에만 출력에 대한 액세스 권한을 부여하려면 다른 애플리케이션의 애플리케이션 매니페스트에서 출력 노드와 해당 엣지를 제거하고 재배포하십시오.

문제: 애플리케이션 출력이 로그에 표시되지 않습니다.

Python 로거를 구성하여 `/opt/aws/panorama/logs`에 로그 파일을 기록하세요. 코드 컨테이너 노드의 로그 스트림에 캡처됩니다. 표준 출력 및 오류 스트림은 `console-output`이라는 별도의 로그 스트림에 캡처됩니다. `print`를 사용하는 경우 `flush=True` 옵션을 사용하여 메시지가 출력 버퍼에 걸리지 않도록 하십시오.

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

출처: AWS Panorama 서비스

애플리케이션에 변경 내용을 배포할 때마다 사용하는 각 패키지의 패키지 구성과 자산 파일을 나타내는 패치 버전을 등록합니다. 클린업 패치 스크립트를 사용하여 사용하지 않는 패치 버전을 등록 취소하십시오.

카메라 스트림

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

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

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

출처: 카메라 노드 로그

어플라이언스가 애플리케이션의 카메라 스트림에 연결할 수 없습니다. 이 경우 애플리케이션이 AWS Panorama Application SDK에서 비디오 프레임을 기다리는 동안 비디오 출력이 비어 있거나 마지막으로 처리된 프레임에서 멈춥니다. 어플라이언스 소프트웨어가 카메라 스트림에 연결을 시도하고 카메라 노드 로그에 타임아웃 오류를 기록합니다. 카메라 스트림 URL이 올바른지, 네트워크 내의 카메라와 어플라이언스 간에 RTSP 트래픽을 라우팅할 수 있는지 확인하십시오. 자세한 내용은 AWS Panorama 어플라이언스를 네트워크에 연결하기 단원을 참조하세요.

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

출처: OCC 로그

카메라 스트림의 보안 인증 정보가 포함된 AWS Secrets Manager 보안 암호를 찾을 수 없습니다. 카메라 스트림을 삭제하고 다시 생성하세요.

오류: Camera did not provide an H264 encoded stream

출처: 카메라 노드 로그

카메라 스트림에 H.264 이외의 인코딩이 있습니다(예: H.265). H.264 카메라 스트림을 사용하여 애플리케이션을 재배포하십시오. 지원되는 카메라에 대한 자세한 내용은 지원되는 카메라 단원을 참조하십시오.

AWS Panorama의 보안

AWS에서 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객은 보안에 매우 민감한 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와 귀하의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드의 보안 - AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호합니다. AWS는 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사원은 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. AWS Panorama에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하십시오.
- 클라우드 내 보안 – 귀하의 책임은 귀하가 사용하는 AWS 서비스에 의해 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 AWS Panorama 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 AWS Panorama를 구성하는 방법을 보여줍니다. 또한 AWS Panorama 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 배우게 됩니다.

주제

AWS Panorama 어플라이언스 보안 기능

악성 코드 및 기타 악용으로부터 애플리케이션, 모델 및 하드웨어를 보호하기 위해 AWS Panorama 어플라이언스는 다양한 보안 기능 세트를 구현합니다. 여기에는 다음이 포함되지만 이에 국한되지는 않습니다.

- 전체 디스크 암호화 – 어플라이언스는 Linux 통합 키 설정(LUKS2) 전체 디스크 암호화를 구현합니다. 모든 시스템 소프트웨어 및 애플리케이션 데이터는 디바이스 전용 키로 암호화됩니다. 공격자는 장치에 물리적으로 액세스할 수 있지만 저장소의 콘텐츠를 검사할 수 없습니다.
- 메모리 레이아웃 무작위화 – 메모리에 로드된 실행 코드를 표적으로 하는 공격으로부터 보호하기 위해 AWS Panorama Appliance는 주소 공간 레이아웃 무작위화(ASLR)를 사용합니다. ASLR은 메모리에 로드되는 운영 체제 코드의 위치를 무작위로 지정합니다. 이렇게 하면 런타임 시 코드가 저장되는 위치를 예측하여 코드의 특정 섹션을 덮어쓰거나 악용 시도를 방지할 수 있습니다.
- 신뢰할 수 있는 실행 환경 – 어플라이언스는 격리된 스토리지, 메모리 및 처리 리소스가 있는 ARM TrustZone 기반의 신뢰할 수 있는 실행 환경(TEE)을 사용합니다. 보안 영역에 저장된 키 및 기타 민감한 데이터는 TEE 내의 별도 운영 체제에서 실행되는 신뢰할 수 있는 애플리케이션에서만 액세스할 수 있습니다. AWS Panorama 어플라이언스 소프트웨어는 애플리케이션 코드와 함께 신뢰할 수 없는 Linux 환경에서 실행됩니다. 보안 애플리케이션에 요청해야만 암호화 작업에 액세스할 수 있습니다.
- 보안 프로비저닝 – 어플라이언스를 프로비저닝할 때 디바이스로 전송하는 보안 인증 정보(키, 인증서 및 기타 암호화 구성 요소)는 짧은 기간 동안만 유효합니다. 어플라이언스는 수명이 짧은 보안 인증 정보를 사용하여 AWS IoT에 연결하고 유효 기간이 더 긴 인증서를 요청합니다. AWS Panorama 서비스는 보안 인증 정보를 생성하고 디바이스에 하드코딩된 키를 사용하여 보안 인증 정보를 암호화합니다. 인증서를 요청한 디바이스만 인증서를 해독하고 AWS Panorama와 통신할 수 있습니다.
- 보안 부팅 – 디바이스가 시작되면 각 소프트웨어 구성 요소가 실행되기 전에 인증됩니다. 프로세서에 하드코딩되어 수정할 수 없는 소프트웨어인 부팅 ROM은 하드코딩된 암호화 키를 사용하여 부트로더를 해독하고 신뢰할 수 있는 실행 환경 커널 등을 검증합니다.
- 서명된 커널 – 커널 모듈은 비대칭 암호화 키로 서명됩니다. 운영 체제 커널은 공개 키로 서명을 해독하고 모듈을 메모리에 로드하기 전에 해당 서명이 모듈의 서명과 일치하는지 확인합니다.
- dm-verity – 커널 모듈을 검증하는 방법과 마찬가지로, 어플라이언스는 Linux Device Mapper의 `dm-verity` 기능을 사용하여 어플라이언스 소프트웨어 이미지를 마운트하기 전에 무결성을 확인합니다. 어플라이언스 소프트웨어가 수정되면 실행되지 않습니다.
- 롤백 방지 – 어플라이언스 소프트웨어를 업데이트하면 어플라이언스가 SoC(시스템 온 칩)의 전자 퓨즈를 끊습니다. 각 소프트웨어 버전에서는 점점 더 많은 퓨즈가 끊어질 것으로 예상되며, 더 많이 끊어지면 작동할 수 없습니다.

AWS Panorama 어플라이언스 보안 모범 사례

AWS Panorama 어플라이언스를 사용할 때는 다음 모범 사례를 고려하십시오.

- 어플라이언스를 물리적으로 보호 – 밀폐된 서버 랙 또는 보안실에 어플라이언스를 설치합니다. 승인된 직원만 디바이스에 물리적으로 접근할 수 있도록 제한하십시오.
- 어플라이언스의 네트워크 연결 보호 - 내부 및 외부 리소스에 대한 액세스를 제한하는 라우터에 어플라이언스를 연결합니다. 어플라이언스는 안전한 내부 네트워크를 사용할 수 있는 카메라에 연결해야 합니다. 또한 AWS에 연결해야 합니다. 두 번째 이더넷 포트는 물리적 이중화에만 사용하고 필요한 트래픽만 허용하도록 라우터를 구성하십시오.

  권장 네트워크 구성 중 하나를 사용하여 네트워크 레이아웃을 계획하십시오. 자세한 내용은 AWS Panorama 어플라이언스를 네트워크에 연결하기 단원을 참조하십시오.
- USB 드라이브 포맷 - 어플라이언스를 프로비저닝한 후 USB 드라이브를 제거하고 포맷합니다. AWS Panorama 서비스에 등록되면 어플라이언스는 USB 드라이브를 사용하지 않습니다. 드라이브를 포맷하여 임시 보안 인증 정보, 구성 파일 및 프로비저닝 로그를 제거합니다.
- 어플라이언스를 최신 상태로 유지 – 어플라이언스 소프트웨어 업데이트를 적시에 적용합니다. AWS Panorama 콘솔에서 어플라이언스를 보면 콘솔에서 소프트웨어 업데이트가 있는지 알려줍니다. 자세한 내용은 AWS Panorama 어플라이언스 관리 단원을 참조하십시오.

  DescribeDevice API 작업을 사용하면 `LatestSoftware`와 `CurrentSoftware` 필드를 비교하여 업데이트 확인을 자동화할 수 있습니다. 최신 소프트웨어 버전이 현재 버전과 다를 경우 콘솔을 사용하거나 CreateJobForDevices 작업을 사용하여 업데이트를 적용하십시오.
- 어플라이언스 사용을 중지한 경우, 어플라이언스 재설정 - 어플라이언스를 보안 데이터 센터 외부로 이동하기 전에 완전히 재설정합니다. 어플라이언스의 전원을 끄고 플러그를 꽂은 상태에서 전원 및 재설정 버튼을 동시에 5초 동안 누릅니다. 이렇게 하면 어플라이언스에서 계정 보안 인증 정보, 애플리케이션 및 로그가 삭제됩니다.

  자세한 내용은 AWS Panorama 어플라이언스 버튼 및 표시등 단원을 참조하십시오.
- AWS Panorama 및 기타 AWS 서비스에 대한 액세스 제한 – AWSPanoramaFullAccess는 모든 AWS Panorama API 작업에 대한 액세스와 필요한 경우 다른 서비스에 대한 액세스를 제공합니다. 가능한 경우 정책은 명명 규칙에 따라 리소스에 대한 액세스를 제한합니다. 예를 들어, `panorama`로 시작하는 이름을 가진 AWS Secrets Manager 보안 암호에 대한 액세스를 제공합니다. 읽기 전용 액세스 또는 보다 구체적인 리소스 세트에 대한 액세스가 필요한 사용자의 경우 관리형 정책을 최소 권한 정책의 시작점으로 사용하십시오.

  자세한 내용은 AWS Panorama에 대한 보안 인증 기반 IAM 정책 단원을 참조하십시오.

AWS Panorama의 데이터 보호

AWS 공동 책임 모델은 AWS Panorama의 데이터 보호에 적용됩니다. 이 모델이 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하십시오. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터를 보호하려면 AWS 계정 보안 인증 정보를 보호하고 AWS IAM Identity Center 또는 AWS Identity and Access Management(IAM)을 통해 개별 사용자 계정을 설정하는 것이 좋습니다. 이 방식을 사용하면 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2가 필수이며 TLS 1.3을 권장합니다.
- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정합니다.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용합니다.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2 섹션을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 AWS Panorama 또는 기타 AWS 서비스에서 콘솔, API, AWS CLI 또는 AWS SDK를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

단원

- 전송 중 암호화
- AWS Panorama 어플라이언스
- 애플리케이션

- 기타 서비스

전송 중 암호화

AWS Panorama API 엔드포인트는 HTTPS를 통한 보안 연결만을 지원합니다. AWS Management Console, AWS SDK 또는 AWS Panorama API를 사용하여 AWS Panorama 리소스를 관리하면 모든 통신이 TLS(전송 계층 보안)로 암호화됩니다. AWS Panorama 어플라이언스와 AWS 간의 통신도 TLS로 암호화됩니다. RTSP를 통한 AWS Panorama 어플라이언스와 카메라 간의 통신은 암호화되지 않습니다.

API 엔드포인트 전체 목록은 AWS 일반 참조의 AWS 리전 및 엔드포인트를 참조하십시오.

AWS Panorama 어플라이언스

AWS Panorama 어플라이언스에는 이더넷, HDMI 비디오 및 USB 스토리지를 위한 물리적 포트가 있습니다. SD 카드 슬롯, Wi-Fi 및 블루투스는 사용할 수 없습니다. USB 포트는 구성 아카이브를 어플라이언스로 전송하기 위해 프로비저닝 중에만 사용됩니다.

어플라이언스의 프로비전 인증서 및 네트워크 구성을 포함하는 구성 아카이브의 내용은 암호화되지 않습니다. AWS Panorama는 이러한 파일을 저장하지 않으며, 어플라이언스를 등록할 때만 검색할 수 있습니다. 구성 아카이브를 어플라이언스로 전송한 후에는 컴퓨터와 USB 스토리지 디바이스에서 삭제하십시오.

어플라이언스의 전체 파일 시스템이 암호화됩니다. 또한 어플라이언스는 필수 소프트웨어 업데이트에 대한 롤백 보호, 서명된 커널 및 부트로더, 소프트웨어 무결성 검증을 비롯한 여러 시스템 수준 보호를 적용합니다.

어플라이언스 사용을 중지하면 전체 재설정을 수행하여 애플리케이션 데이터를 삭제하고 어플라이언스 소프트웨어를 재설정하십시오.

애플리케이션

어플라이언스에 배포하는 코드를 제어할 수 있습니다. 소스에 관계없이 배포하기 전에 모든 애플리케이션 코드의 보안 문제를 검증하십시오. 애플리케이션에서 타사 라이브러리를 사용하는 경우 해당 라이브러리의 라이선스 및 지원 정책을 신중하게 고려하십시오.

애플리케이션 CPU, 메모리 및 디스크 사용량은 어플라이언스 소프트웨어의 제약을 받지 않습니다. 너무 많은 리소스를 사용하는 애플리케이션은 다른 애플리케이션과 디바이스 작동에 부정적인 영향을

미칠 수 있습니다. 프로덕션 환경에 결합하거나 배포하기 전에 애플리케이션을 개별적으로 테스트하십시오.

애플리케이션 자산(코드 및 모델)은 계정, 어플라이언스 또는 빌드 환경 내의 액세스와 분리되지 않습니다. AWS Panorama Application CLI에서 생성된 컨테이너 이미지와 모델 아카이브는 암호화되지 않습니다. 프로덕션 워크로드에는 별도의 계정을 사용하고 특정 주기 없이 필요에 따라 액세스를 허용합니다.

기타 서비스

모델 및 애플리케이션 컨테이너를 Amazon S3에 안전하게 저장하기 위해 AWS Panorama는 Amazon S3에서 관리하는 키를 사용한 서버 측 암호화를 사용합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 암호화를 사용하여 데이터 보호를 참조하세요.

카메라 스트림 보안 인증 정보는 미사용 시 AWS Secrets Manager에 암호화됩니다. 어플라이언스의 IAM 역할은 스트림의 사용자 이름과 암호에 액세스하기 위해 보안 암호를 검색할 권한을 부여합니다.

AWS Panorama 어플라이언스는 Amazon CloudWatch Logs에 로그 데이터를 전송합니다. CloudWatch 로그는 기본적으로 이 데이터를 암호화하며 고객 관리 키를 사용하도록 구성할 수 있습니다. 자세한 내용은 Amazon Logs 사용 설명서의 로그 데이터 암호화를 사용하여 CloudWatch AWS KMS 로그의 CloudWatch 로그 데이터 암호화를 참조하십시오.

AWS Panorama에 대한 ID 및 액세스 관리

AWS Identity and Access Management(IAM)은 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 누가 AWS Panorama 리소스를 사용하도록 인증되고(로그인됨) 권한이 부여되는지(권한 있음)를 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

주제

고객

AWS Identity and Access Management(IAM)을 사용하는 방법은 AWS Panorama에서 수행하는 작업에 따라 달라집니다.

서비스 사용자 – AWS Panorama 서비스를 사용하여 작업을 수행하는 경우 필요한 보안 인증과 권한을 관리자가 제공합니다. 더 많은 AWS Panorama 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. AWS Panorama의 기능에 액세스할 수 없는 경우 AWS Panorama 보안 인증 및 액세스 문제 해결 단원을 참조하세요.

서비스 관리자 - 회사에서 AWS Panorama 리소스를 책임지고 있는 경우 AWS Panorama에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 AWS Panorama 기능과 리소스를 결정합니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사가 AWS Panorama에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 AWS Panorama가 IAM과 작동하는 방식 단원을 참조하세요.

IAM 관리자 – IAM 관리자라면 AWS Panorama에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 AWS Panorama 보안 인증 기반 정책 예제를 보려면 AWS Panorama 보안 인증 기반 정책 예제을 참조하세요.

보안 인증을 통한 인증

인증은 ID 보안 인증 정보를 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자이나 IAM 사용자로 또는 IAM 역할을 수임하여 인증(AWS에 로그인)되어야 합니다.

보안 인증 정보 소스를 통해 제공된 보안 인증 정보를 사용하여 페더레이션형 ID로 AWS에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증, Google 또는 Facebook 보안 인증 정보가 페더레이션형 ID의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다.

사용자 유형에 따라 AWS Management Console 또는AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법을 참조하십시오.

AWS에 프로그래밍 방식으로 액세스하는 경우, AWS에서는 보안 인증 정보를 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK) 및 명령줄 인터페이스(CLI)를 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청에 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS는 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하십시오.

AWS 계정 루트 사용자

AWS 계정를 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 단일 로그인 ID로 시작합니다. 이 자격 증명은 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에는 루트 사용자를 가급적 사용하지 않는 것이 좋습니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하십시오.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 보안 인증이 있는 IAM 사용자를 생성하는 대신 임시 보안 인증을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하십시오.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 귀하는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins(이)라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 보안 인증을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 정보는 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하십시오.

IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정 계정 내 ID입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할 전환하여 AWS Management Console에서 IAM 역할을 임시로 수임할 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할 사용을 참조하십시오.

임시 보안 인증 정보가 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 자격 증명에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 Creating a role for a third-party Identity Provider(서드 파티 자격 증명 공급자의 역할 만들기) 부분을 참조하십시오. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 아이덴티티가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연결합니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하십시오.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스: IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입

니다. 그러나 일부 AWS 서비스를 사용하면 역할을(프록시로 사용하는 대신) 리소스에 정책을 직접 연결할 수 있습니다. 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.

- 교차 서비스 액세스 - 일부 AWS 서비스는 다른 AWS 서비스의 기능을 사용합니다. 예를 들어 서비스에서 직접적으로 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하십시오.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하십시오.
  - 서비스 연결 역할 - 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.
- Amazon EC2에서 실행 중인 애플리케이션 – IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 보안 인증을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하십시오.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하십시오.

정책을 사용한 액세스 관리

정책을 생성하고 AWS 자격 증명 또는 리소스에 연결하여 AWS 내 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS는 보안 주체(사용

자, 루트 사용자 또는 역할 세션)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되는지 또는 거부되는지를 결정합니다. 대부분의 정책은 AWS에 JSON 설명서로서 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하십시오.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는 지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console, AWS CLI또는 AWSAPI에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

자격 증명 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 AWS 계정에 속한 다수의 사용자, 그룹 및 역할에 독립적으로 추가할 수 있는 정책입니다. 관리형 정책에는 AWS관리형 정책과 고객 관리형 정책이 포함되어 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 AWS 서비스가 포함될 수 있습니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3, AWS WAF 및 Amazon VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 액세스 제어 목록(ACL) 개요를 참조하십시오.

기타 정책 유형

AWS은(는) 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 유형은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 엔터티의 ID 기반 정책 및 해당 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계를 참조하십시오.
- 서비스 제어 정책(SCP) – SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations은 기업이 소유하는 여러 개의 AWS 계정를 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자을 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하십시오.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책 및 세션 정책의 교집합입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하십시오.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지를 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

AWS Panorama가 IAM과 작동하는 방식

IAM을 사용하여 AWS Panorama에 대한 액세스를 관리하려면 먼저 어떤 IAM 기능을 AWS Panorama에 사용할 수 있는지를 이해해야 합니다. AWS Panorama 및 기타 AWS 서비스에서 IAM을 사용하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스를 참조하세요.

AWS Panorama에서 사용 중인 역할, 권한과 정책에 대한 개요는 AWS Panorama 권한 단원을 참조하세요.

AWS Panorama 보안 인증 기반 정책 예제

기본적으로 IAM 사용자 및 역할은 AWS Panorama 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS CLI또는 AWSAPI를 사용해 태스크를 수행할 수 없습니다. IAM 관리자는 지정된 리소스에서 특정 API 태스크를 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 정책을 연결해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 JSON 탭에서 정책 생성을 참조하세요.

주제

- 정책 모범 사례
- AWS Panorama 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 AWS Panorama 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아가기 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 관리형 정책은 AWS 계정에서 사용할 수 있습니다. 사용 사례에 고유한 AWS고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS managed policies(관리형 정책) 또는 AWS managed policies for job functions(직무에 대한 관리형 정책)를 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는

작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 Policies and permissions in IAM(IAM의 정책 및 권한)을 참조하세요.

- IAM 정책의 조건을 사용하여 액세스 추가 제한: 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 특정 AWS 서비스(예: AWS CloudFormation)을(를) 통해 사용되는 경우에만 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 IAM 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 권장 사항을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하십시오.
- 다중 인증(MFA) 필요: AWS 계정 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 설정합니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 Configuring MFA-protected API access(MFA 보호 API 액세스 구성)를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례 섹션을 참조하십시오.

AWS Panorama 콘솔 사용

AWS Panorama 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은 AWS 계정에서 AWS Panorama 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다. 최소 필수 권한보다 더 제한적인 보안 인증 기반 정책을 만들면 콘솔이 해당 정책에 연결된 개체(IAM 사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

자세한 내용은 AWS Panorama에 대한 보안 인증 기반 IAM 정책 단원을 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예시는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI나 AWS API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "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 Panorama의 AWS 관리형 정책

AWS 관리형 정책은 AWS에 의해 생성되고 관리되는 독립 실행형 정책입니다. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에서 정의한 권한은 변경할 수 없습니다. AWS에서 AWS 관리형 정책에 정의된 권한을 업데이트할 경우 정책이 연결되어 있는 모든 보안 주체 엔터티(사용자, 그룹 및 역할)에도 업데

이트가 적용됩니다. 새로운 AWS 서비스를 시작하거나 새로운 API 작업을 기존 서비스에 이용하는 경우 AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하십시오.

AWS Panorama는 다음과 같은 관리형 정책을 제공합니다. 각 정책의 전체 내용과 변경 내역은 IAM 콘솔의 링크 페이지를 참조하십시오.

- AWSPanoramaFullAccess – AWS Panorama, Amazon S3의AWS Panorama 액세스 포인트, AWS Secrets Manager의 어플라이언스 보안 인증 정보, Amazon CloudWatch의 어플라이언스 로그에 대한 전체 액세스를 제공합니다. AWS Panorama의 서비스 연결 역할을 생성할 수 있는 권한이 포함되어 있습니다.
- AWSPanoramaServiceLinkedRolePolicy – AWS Panorama에서 AWS IoT, AWS Secrets Manager, AWS Panorama 등에서 리소스를 관리할 수 있습니다.
- AWSPanoramaApplianceServiceRolePolicy – AWS Panorama 어플라이언스가 CloudWatch에 로그를 업로드하고, AWS Panorama에서 생성한 Amazon S3 액세스 포인트에서 개체를 가져올 수 있습니다.

AWS 관리형 정책에 대한 AWS Panorama 업데이트

다음 표에서는 AWS Panorama의 관리형 정책 업데이트를 설명합니다.

변경 사항	설명	날짜
AWSPanoramaFullAccess – 기존 정책 업데이트	사용자가 CloudWatch Logs 콘솔에서 로그 그룹을 볼 수 있도록 사용자 정책에 권한을 추가했습니다.	2022년 1월 13일
AWSPanoramaFullAccess – 기존 정책 업데이트	사용자가 AWS Panorama 서비스 연결 역할을 관리하고 IAM, Amazon S3, CloudWatch 및 Secrets Manager를 비롯한 다른 서비스에서 AWS Panorama 리소스에 액세스할 수 있도록 하는 권한을 사용자 정책에 추가했습니다.	2021년 10월 20일

변경 사항	설명	날짜
AWSPanoramaApplianceServiceRolePolicy – 새 정책	AWS Panorama 어플라이언스 서비스 역할에 대한 새 정책입니다.	2021년 10월 20일
AWSPanoramaServiceLinkedRolePolicy — 새 정책	AWS Panorama 서비스 역할에 대한 새 정책입니다.	2021년 10월 20일
AWS Panorama에서 변경 내용 추적 시작	AWS Panorama에서 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 10월 20일

AWS Panorama에 서비스 연결 역할 사용

AWS Panorama은 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 AWS Panorama에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 AWS Panorama에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 통해 AWS Panorama 설정이 쉬워지는데 필요한 권한을 수동으로 추가할 필요가 없기 때문입니다. AWS Panorama에서 서비스 연결 역할 권한을 정의하므로, 달리 정의되지 않은 한 AWS Panorama에서만 해당 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며, 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 AWS Panorama 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-linked roles) 열에 예(Yes)가 있는 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

단원

- AWS Panorama에 대한 서비스 연결 역할 권한
- AWS Panorama에 대한 서비스 연결 역할 생성
- AWS Panorama에 대한 서비스 연결 역할 편집
- AWS Panorama에 대한 서비스 연결 역할 삭제

- AWS Panorama 서비스 연결 역할이 지원되는 리전

AWS Panorama에 대한 서비스 연결 역할 권한

AWS Panorama는 AWSServiceRoleForAWSPanorama라는 서비스 연결 역할을 사용합니다. — AWS Panorama에서 AWS IoT, AWS Secrets Manager, AWS Panorama 등에서 리소스를 관리할 수 있습니다.

AWSServiceRoleForAWSPanorama 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `panorama.amazonaws.com`

역할 권한 정책을 통해 AWS Panorama는 다음 작업을 완료할 수 있습니다.

- AWS Panorama 리소스 모니터링
- AWS Panorama 어플라이언스의 AWS IoT 리소스 관리
- AWS Secrets Manager 보안 암호에 액세스하여 카메라 보안 인증 정보 받기

전체 권한 목록은 IAM 콘솔의 AWSPanoramaServiceLinkedRolePolicy 정책을 참조하세요.

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 연결 역할을 작성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

AWS Panorama에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI 또는 AWS API에 어플라이언스를 등록하면 AWS Panorama에서 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제한 다음 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 어플라이언스를 등록하면 AWS Panorama에서 서비스 연결 역할을 다시 생성합니다.

AWS Panorama에 대한 서비스 연결 역할 편집

AWS Panorama은 AWSServiceRoleForAWSPanorama 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 연결 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

AWS Panorama에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 개체가 없도록 합니다. 단, 서비스 연결 역할에 대한 리소스를 먼저 정리해야 수동으로 삭제할 수 있습니다.

AWS 파노라마용 AWS 서비스 역할에서 사용하는 AWS Panorama 리소스를 삭제하려면 이 설명서의 다음 섹션에 있는 절차를 사용하십시오.

- 버전 및 애플리케이션 삭제
- 어플라이언스 등록 취소

> **Note**
>
> 리소스를 삭제하려 할 때 AWS Panorama 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

AWSServiceRoleForAWSPanorama 서비스 연결 역할을 삭제하려면 IAM 콘솔, AWS CLI 또는 AWS API를 사용하세요. 자세한 내용은 IAM 사용 설명서의 서비스에 연결 역할 삭제를 참조하세요.

AWS Panorama 서비스 연결 역할이 지원되는 리전

AWS Panorama는 서비스가 제공되는 모든 리전에서 서비스 연결 역할을 사용하도록 지원합니다. 자세한 내용은 AWS 리전 및 엔드포인트 단원을 참조하세요.

교차 서비스 혼동된 대리자 예방

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. AWS에서는 교차 서비스 가장으로 인해 혼동된 대리자 문제가 발생할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 호출할 때 발생할 수 있습니다. 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

AWS Panorama가 리소스에 다른 서비스를 제공하는 권한을 제한하려면 리소스 정책에서 `aws:SourceArn` 및 `aws:SourceAccount` 글로벌 조건 컨텍스트 키를 사용하는 것이 좋습니다. 두

글로벌 조건 컨텍스트 키를 모두 사용하는 경우 `aws:SourceAccount` 값과 `aws:SourceArn` 값의 계정은 동일한 정책 문에서 사용할 경우 동일한 계정 ID를 사용해야 합니다.

`aws:SourceArn`의 값은 AWS Panorama 디바이스의 ARN이어야 합니다.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모를 경우 또는 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드(*)를 포함한 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용합니다. 예: `arn:aws:servicename::123456789012:*`.

AWS Panorama이 AWS Panorama 어플라이언스에 대한 권한을 부여하는 데 사용하는 서비스 역할 보안에 대한 지침은 어플라이언스 역할 보안을 참조하십시오.

AWS Panorama 보안 인증 및 액세스 문제 해결

다음 정보를 사용하여 AWS Panorama 및 IAM에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- AWS Panorama에서 작업을 수행할 권한이 없음
- IAM을 수행할 권한이 없습니다. PassRole
- 내 AWS 계정 외부의 사람이 내 AWS Panorama 리소스에 액세스하도록 허용하려고 함

AWS Panorama에서 작업을 수행할 권한이 없음

AWS Management Console에서 작업을 수행할 권한이 없다는 메시지가 나타나는 경우 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 사용자 이름과 비밀번호를 제공한 사람입니다.

다음 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 어플라이언스에 대한 세부 정보를 보려고 하지만 `panorama:DescribeAppliance` 권한이 없는 경우에 발생합니다.

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

이 경우 Mateo는 `my-appliance` 작업을 사용하여 `panorama:DescribeAppliance` 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

IAM을 수행할 권한이 없습니다. PassRole

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 AWS Panorama에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예시 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 AWS Panorama에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

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

이 경우 Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하십시오. 관리자는 로그인 보안 인증을 제공한 사용자입니다.

내 AWS 계정 외부의 사람이 내 AWS Panorama 리소스에 액세스하도록 허용하려고 함

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하십시오.

- AWS Panorama에서 이러한 기능을 지원하는지 여부를 알아보려면 AWS Panorama가 IAM과 작동하는 방식 단원을 참조하세요.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 자신이 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 권한 제공을 참조하십시오.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 서드 파티가 소유한 AWS 계정에 대한 액세스 제공을 참조하십시오.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하십시오.

- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.

AWS Panorama의 규정 준수 검증

AWS 서비스가 특정 규정 준수 프로그램의 범위에 포함되는지 알아보려면 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하고 관심 있는 규정 준수 프로그램을 선택하십시오. 일반적인 정보는 AWS 규정 준수 프로그램을 참조하십시오.

AWS Artifact를 사용하여 서드 파티 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하십시오.

AWS 서비스 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다.AWS에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – 이 배포 안내서에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수에 중점을 둔 기본 AWS환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Services에서 HIPAA 보안 및 규정 준수 기술 백서 설계 - 이 백서는 기업에서 AWS를 사용하여 HIPAA를 준수하는 애플리케이션을 만드는 방법을 설명합니다.

> **Note**
>
> 모든 AWS 서비스에 HIPAA 자격이 있는 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하십시오.

- AWS 규정 준수 리소스 - 고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 컬렉션입니다.
- AWS 고객 규정 준수 가이드 - 규정 준수의 관점에서 공동 책임 모델을 이해합니다. 이 가이드에서는 AWS 서비스를 보호하기 위한 모범 사례를 요약하고 여러 프레임워크(미국 표준 기술 연구소(NIST), 결제 카드 산업 보안 표준 위원회(PCI), 국제 표준화기구(ISO) 등)에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 가이드의 규칙을 사용하여 리소스 평가 – AWS Config 서비스는 내부 사례, 산업 지침 및 규제에 대한 리소스 구성의 준수 상태를 평가합니다.
- AWS Security Hub – 이 AWS 서비스는 AWS내의 보안 상태에 대한 포괄적인 보기를 제공합니다. Security Hub는 보안 제어를 사용하여 AWS리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하십시오.
- AWS Audit Manager - 이 AWS 서비스는 AWS사용을 지속해서 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 간소화할 수 있도록 지원합니다.

사람이 있는 경우에 대한 추가 고려 사항

다음은 사람이 있을 수 있는 시나리오에 AWS Panorama를 사용할 때 고려해야 할 몇 가지 모범 사례입니다.

- 사용 사례에 적용되는 모든 법률 및 규정을 숙지하고 준수해야 합니다. 여기에는 카메라의 위치 및 시야각과 관련된 법률, 카메라 배치 및 사용 시 안내문 및 표지판 요구 사항, 동영상에 등장할 수 있는 사람의 권리(개인정보 보호 권리 포함)가 포함될 수 있습니다.
- 카메라가 사람과 개인 정보 보호에 미치는 영향을 고려하세요. 법적 요건 외에도 카메라가 있는 구역에 안내문을 부착하는 것이 적절한지, 사람들이 카메라에 찍힌다는 사실에 놀라지 않도록 카메라가 잘 보이도록 가려지지 않게 배치해야 하는지 등을 고려하세요.
- 카메라 작동 및 카메라에서 얻은 데이터 검토에 대한 적절한 정책과 절차를 마련하세요.
- 카메라에서 얻은 데이터에 대한 적절한 액세스 제어, 사용 제한 및 보존 기간을 고려하세요.

AWS Panorama의 인프라 보안

관리형 서비스로서 AWS Panorama는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스와 AWS의 인프라 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하십시오. 인프라 보안에 대한 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하십시오.

AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 AWS Panorama에 액세스합니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 보안 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

데이터 센터에 AWS Panorama 어플라이언스 배포

AWS Panorama 어플라이언스는 AWS 서비스와의 통신을 위해 인터넷 액세스가 필요합니다. 또한 내부 카메라 네트워크에도 액세스할 수 있어야 합니다. 네트워크 구성을 신중하게 고려하고 각 디바이스에 필요한 액세스 권한만 제공하는 것이 중요합니다. 구성에서 AWS Panorama 어플라이언스가 민감한 IP 카메라 네트워크에 대한 브리지 역할을 하도록 허용하는 경우 주의해야 합니다.

사용자는 다음을 수행할 책임이 있습니다.

- AWS Panorama 어플라이언스의 물리적 및 논리적 네트워크 보안.
- AWS Panorama 어플라이언스를 사용할 때 네트워크 연결 카메라를 안전하게 운영.
- AWS Panorama 어플라이언스 및 카메라 소프트웨어를 최신 상태로 유지.
- 개인 정보 보호 관련 사항을 포함하여 프로덕션 환경에서 수집한 비디오 및 이미지의 콘텐츠와 관련된 모든 관련 법률 또는 규정 준수.

AWS Panorama 어플라이언스는 암호화되지 않은 RTSP 카메라 스트림을 사용합니다. AWS Panorama 어플라이언스를 네트워크에 연결하는 방법에 대한 자세한 내용은 AWS Panorama 어플라

이언스를 네트워크에 연결하기 단원을 참조하십시오. 암호화에 대한 자세한 내용은 AWS Panorama의 데이터 보호 단원을 참조하십시오.

AWS Panorama의 런타임 환경 소프트웨어

AWS Panorama는 AWS Panorama 어플라이언스의 Ubuntu Linux 기반 환경에서 애플리케이션 코드를 실행하는 소프트웨어를 제공합니다. AWS Panorama는 어플라이언스 이미지의 소프트웨어를 최신 상태로 유지하는 역할을 합니다. AWS Panorama는 정기적으로 소프트웨어 업데이트를 릴리스하며, AWS Panorama 콘솔을 사용하여 적용할 수 있습니다.

애플리케이션 코드에 라이브러리를 설치하면 애플리케이션의 `Dockerfile`에 라이브러리를 사용할 수 있습니다. 빌드 전반에서 애플리케이션 안정성을 보장하려면 각 라이브러리의 특정 버전을 선택하세요. 종속 항목을 정기적으로 업데이트하여 보안 문제를 해결하세요.

출시

다음 표에는 AWS Panorama 서비스, 소프트웨어 및 설명서에 대한 기능 및 소프트웨어 업데이트가 출시된 시기가 나와 있습니다. 모든 기능에 액세스하려면 AWS Panorama 어플라이언스를 최신 소프트웨어 버전으로 업데이트하십시오. 릴리스에 대한 자세한 내용은 링크된 주제를 참조하십시오.

변경 사항	설명	날짜
어플라이언스 소프트웨어 업데이트	버전 7.0.13은 어플라이언스가 소프트웨어 업데이트를 관리하는 방식을 변경하는 메이저 버전 업데이트입니다. 어플라이언스로부터의 네트워크 통신 아웃바운드를 제한하거나 프라이빗 VPC 서브넷에 연결하는 경우 업데이트를 적용하기 전에 추가 엔드포인트 및 포트에 대한 액세스를 허용해야 합니다. 자세한 내용은 변경 로그를 참조하십시오.	2023년 12월 28일
어플라이언스 소프트웨어 업데이트	버전 6.2.1에는 버그 수정이 포함되어 있습니다. 자세한 내용은 변경 로그를 참조하십시오.	2023년 9월 6일
어플라이언스 소프트웨어 업데이트	버전 6.0.8에는 버그 수정 및 보안 개선 사항이 포함되어 있습니다. 자세한 내용은 변경 로그를 참조하십시오.	2023년 7월 6일
어플라이언스 소프트웨어 업데이트	버전 5.1.7에는 버그 수정 및 오류 처리 개선 사항이 포함되어 있습니다. 자세한 내용은 변경 로그를 참조하십시오.	2023년 3월 31일
콘솔 업데이트	이제 관리 콘솔에서 AWS Panorama 어플라이언스를 구	2023년 2월 2일

	입할 수 있습니다. 사용자에게 디바이스 구매 권한을 부여하려면 AWS Panorama에 대한 자격 증명 기반 IAM 정책을 참조하십시오.	
어플라이언스 소프트웨어 업데이트	버전 5.0.74에는 버그 수정 및 오류 처리 개선 사항이 포함되어 있습니다. 자세한 내용은 변경 로그를 참조하십시오.	2023년 1월 23일
API 업데이트	어플라이언스 소프트웨어 주요 버전 업데이트를 옵트인할 수 있는 `AllowMajorVersionUpdate` 옵션을 `OTAJobConfig` 에 추가했습니다. 자세한 내용은 을 참조하십시오. CreateJobForDevices	2023년 1월 19일
개발자를 위한 새로운 도구	AWS Panorama샘플 GitHub 저장소에서 새 도구인 “사이드 로딩”을 사용할 수 있습니다. 이 도구를 사용하면 컨테이너를 빌드하고 배포하지 않고도 애플리케이션 코드를 업데이트할 수 있습니다. 자세한 내용은 README를 참조하십시오.	2022년 11월 16일
애플리케이션 기반 이미지 업데이트	버전 1.2.0은 `video_in.get()` 에 시간 제한 옵션을 추가하고 `AWS_REGION` 환경 변수를 설정하며 오류 처리를 개선합니다. 자세한 내용은 변경 로그를 참조하십시오.	2022년 11월 16일

어플라이언스 소프트웨어 업데이트	버전 5.0.42에는 버그 수정 및 보안 업데이트가 포함되어 있습니다. 자세한 내용은 변경 로그를 참조하십시오.	2022년 11월 16일
어플라이언스 소프트웨어 업데이트	버전 5.0.7에는 원격으로 어플라이언스 재부팅 및 원격으로 카메라 스트림 일시정지 기능이 추가되었습니다. 자세한 내용은 변경 로그를 참조하십시오.	2022년 10월 13일
어플라이언스 소프트웨어 업데이트	버전 4.3.93에는 오프라인 디바이스에서 로그를 검색하는 기능이 추가되었습니다. 자세한 내용은 변경 로그를 참조하십시오.	2022년 8월 24일
어플라이언스 소프트웨어 업데이트	버전 4.3.72에는 버그 수정 및 보안 업데이트가 포함되어 있습니다. 자세한 내용은 변경 로그를 참조하십시오.	2022년 6월 23일
AWS PrivateLink 지원	AWS Panorama는 프라이빗 서브넷에서 AWS Panorama 리소스를 관리하기 위한 VPC 엔드포인트를 지원합니다. 자세한 내용을 알아보려면 VPC 엔드포인트 사용을 참조하십시오.	2022년 6월 2일
어플라이언스 소프트웨어 업데이트	버전 4.3.55는 `console_output` 로그의 스토리지 사용률을 개선합니다. 자세한 내용은 변경 로그를 참조하십시오.	2022년 5월 5일

레노버® SE70 ThinkEdge	AWS Panorama을 위한 새로운 어플라이언스는 Lenovo에서 구입할 수 있습니다. 엔비디아 젯슨 자비에르 NX로 구동되는 레노버 ThinkEdge® SE70은 어플라이언스와 동일한 기능을 지원합니다. AWS Panorama 자세한 내용은 호환 가능한 디바이스를 참조하십시오.	2022년 4월 6일
애플리케이션 기반 이미지 업데이트	버전 1.1.0은 백그라운드 스레드를 실행할 때 성능을 개선하고 이미지가 최신인지 여부를 나타내는 플래그(is_cached)를 미디어 개체에 추가합니다. 자세한 내용은 gallery.ecr.aws를 참조하십시오.	2022년 3월 29일
어플라이언스 소프트웨어 업데이트	버전 4.3.45에는 GPU 액세스 및 인바운드 포트에 대한 지원이 추가되었습니다. 자세한 내용은 변경 로그를 참조하십시오.	2022년 3월 24일
어플라이언스 소프트웨어 업데이트	버전 4.3.35는 보안 및 성능을 개선합니다. 자세한 내용은 변경 로그를 참조하십시오.	2022년 2월 22일
업데이트된 관리형 정책	AWS Panorama에 대한 AWS Identity and Access Management 관리형 정책이 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책을 참조하십시오.	2022년 1월 13일

프로비저닝 로그	어플라이언스 소프트웨어 4.3.23에서는 어플라이언스가 프로비저닝 중에 USB 드라이브에 로그를 기록합니다. 자세한 내용을 알아보려면 로그를 참조하십시오.	2022년 1월 13일
NTP 서버 구성	이제 시계 동기화에 특정 NTP 서버를 사용하도록 AWS Panorama 어플라이언스를 구성할 수 있습니다. 어플라이언스를 설치하는 동안 다른 네트워킹 설정을 사용하여 NTP 설정을 구성합니다. 자세한 내용은 설정을 참조하십시오.	2022년 1월 13일
추가 리전	AWS Panorama은 이제 아시아 아시아 태평양(싱가포르) 및 아시아 태평양(시드니) 리전에서 사용할 수 있습니다.	2022년 1월 13일
어플라이언스 소프트웨어 업데이트	버전 4.3.4에서는 모델의 `precisionMode` 설정에 대한 지원이 추가되고 로깅 동작이 업데이트됩니다. 자세한 내용은 변경 로그를 참조하십시오.	2021년 11월 8일
업데이트된 관리형 정책	AWS Panorama에 대한 AWS Identity and Access Management 관리형 정책이 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책을 참조하십시오.	2021년 10월 20일

정식 출시	AWS Panorama는 이제 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드) 및 캐나다(중부) 리전의 모든 고객이 사용할 수 있습니다. AWS Panorama 어플라이언스를 구매하려면 AWS Panorama을 방문하십시오.	2021년 10월 20일
평가판	AWS Panorama는 미국 동부(버지니아 북부) 및 미국 서부(오레곤) 리전에서 초대를 통해 사용할 수 있습니다.	2020년 12월 1일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.