aws

Developer Guide

AWS Panorama

AWS Panorama: Developer Guide

Table of Contents

End of support notice: On May 31, 2026, AWS will end support for AWS Panorama. After May 31, 2026, you will no longer be able to access the AWS Panorama console or AWS Panorama resources. For more information, see AWS Panorama end of support.

What is AWS Panorama?

AWS Panorama is a service that brings computer vision to your on-premises camera network. You install the AWS Panorama Appliance or another compatible device in your datacenter, register it with AWS Panorama, and deploy computer vision applications from the cloud. AWS Panorama works with your existing real time streaming protocol (RTSP) network cameras. The appliance runs secure computer vision applications from AWS Partners, or applications that you build yourself with the AWS Panorama Application SDK.

The AWS Panorama Appliance is a compact edge appliance that uses a powerful system-on-module (SOM) that is optimized for machine learning workloads. The appliance can run multiple computer vision models against multiple video streams in parallel and output the results in real time. It is designed for use in commercial and industrial settings and is rated for dust and liquid protection (IP-62).

The AWS Panorama Appliance enables you to run self-contained computer vision applications at the edge, without sending images to the AWS Cloud. By using the AWS SDK, you can integrate with other AWS services and use them to track data from the application over time. By integrating with other AWS services, you can use AWS Panorama to do the following:

- **Analyze traffic patterns** – Use the AWS SDK to record data for retail analytics in Amazon DynamoDB. Use a serverless application to analyze the collected data over time, detect anomalies in the data, and predict future behavior.
- **Receive site safety alerts** – Monitor off-limits areas at an industrial site. When your application detects a potentially unsafe situation, upload an image to Amazon Simple Storage Service (Amazon S3) and send a notification to an Amazon Simple Notification Service (Amazon SNS) topic so recipients can take corrective action.
- **Improve quality control** – Monitor an assembly line's output to identify parts that don't conform to requirements. Highlight images of nonconformant parts with text and a bounding box and display them on a monitor for review by your quality control team.
- **Collect training and test data** – Upload images of objects that your computer vision model couldn't identify, or where the model's confidence in its guess was borderline. Use a serverless application to create a queue of images that need to be tagged. Tag the images and use them to retrain the model in Amazon SageMaker AI.

AWS Panorama uses other AWS services to manage the AWS Panorama Appliance, access models and code, and deploy applications. AWS Panorama does as much as possible without requiring you to interact with other services, but a knowledge of the following services can help you understand how AWS Panorama works.

- SageMaker AI – You can use SageMaker AI to collect training data from cameras or sensors, build a machine learning model, and train it for computer vision. AWS Panorama uses SageMaker AI Neo to optimize models to run on the AWS Panorama Appliance.
- Amazon S3 – You use Amazon S3 access points to stage application code, models, and configuration files for deployment to an AWS Panorama Appliance.
- AWS IoT – AWS Panorama uses AWS IoT services to monitor the state of the AWS Panorama Appliance, manage software updates, and deploy applications. You don't need to use AWS IoT directly.

To get started with the AWS Panorama Appliance and learn more about the service, continue to Getting started with AWS Panorama.

AWS Panorama end of support

After careful consideration, we decided to end support for the AWS Panorama, effective May 31, 2026. AWS Panorama will no longer accept new customers beginning May 20, 2025. As an existing customer with an account signed up for the service before May 20, 2025, you can continue to use AWS Panorama features. After May 31, 2026, you will no longer be able to use AWS Panorama.

Alternatives to AWS Panorama

If you're interested in an alternative to AWS Panorama, AWS has options for both buyers and builders.

For an out-of-the-box solution, the AWS Partner Network offers solutions from multiple partners. You can browse solutions on the AWS Solutions Library from many of our partners. These partner solutions include options for hardware, software, software as a service (SaaS) applications, managed solutions, or custom implementations based on your needs. This approach provides a solution that addresses your use case without requiring you to have expertise in computer vision, AI, or application development. This typically provides faster time to value by taking advantage of the specialized expertise of the AWS Partners.

If you prefer to build your own solution, AWS offers AI tools and services to help you develop an AI-based computer vision application and manage the applications and devices at the edge. Amazon SageMaker provides a set of tools to build, train, and deploy ML models for your use case with fully managed infrastructure, tools, and workflows. In addition to enabling you to build your own models, Amazon SageMaker JumpStart offers built-in computer vision algorithms that can be fine-tuned to your specific use case.

For managing devices and applications at the edge, AWS IoT Greengrass is a proven and secure solution to deploy and update applications for IoT devices. For a server-based implementation, AWS Systems Manager provides a suite of tools for managing servers and Amazon EKS Anywhere or ECS Anywhere can manage application containers on edge servers. Amazon provides some guidelines for managing edge devices, along with additional resources in Section 4 of the Securing Internet of Things (IoT) with AWS whitepaper. This builder approach provides you the tools to accelerate your AI and device management development while providing complete flexibility to build a solution that meets your exact requirements and integrates with your existing hardware and software infrastructure. This typically provides lower operating costs for a solution.

Migrating from AWS Panorama

To move an existing application from AWS Panorama to an alternative implementation, you will need to replace the existing hardware device, migrate the application from the AWS Panorama service, and implement edge management and security for the new solution. Each of those areas will be explored in detail below:

Hardware Replacement

The existing AWS Panorama appliance is based on the Nvidia Jetson Xavier platform. The hardware can be replaced with a similar off-the-shelf device based on the current generation Nvidia Jetson platform that meets your requirements, or an edge server. While most AWS Panorama deployments can be replaced with a similar device, we have seen some customers who utilize a large number of cameras in a single location find that a server is a better alternative.

Application Migration

AWS Panorama applications need to be rewritten to eliminate the use of any AWS Panorama-specific API calls. AWS Panorama applications only support video input through Real-Time Streaming Protocol (RTSP) feeds using H.264 and those video inputs are provided using camera nodes in the AWS Panorama device SDK.

To port an existing application, you will need to implement an application class similar to AWS Panorama so that the existing code can be mostly re-used. Sample code is available in the banner-code.zip file that shows an example of this implementation using both PyAV and OpenCV.

This is a simple approach with a minimal amount of code changes, but has many of the same limitations as the current AWS Panorama based implementation in terms of the types of video streams supported.

Another option would be to re-architect the application to make better use of system resources and to support new application capabilities. For this option, you use GStreamer or DeepStream to implement the media pipeline from media source to inference results and business logic, or use a more feature rich and better performing machine learning (ML) runtime implementation, like the Nvidia Triton inference server. This approach requires changes to more of the video processing pipeline, but is both more efficient and allows more flexibility to support a wider range of codecs, camera types, and other sensors.

Edge Management and Security

Regardless of the media pipeline, you will also have to implement a secure store for credentials, e.g. RTSP stream username and password. AWS provides different ways of securely storing parameters for applications:

- AWS IoT Device Shadow service is used to store parameters that are passed to applications, as well as to track the status of the applications on the edge device.
- AWS Secrets Manager is used to store such credentials to better protect the credentials to access media streams.
- If you use Amazon EKS or Amazon ECS, then you can also use the secure AWS System Manager Parameter store for credentials and other application parameters.

The choice depends on the security requirements of the application, as well as which other AWS products you plan to use to implement your application.

When you replace the AWS Panorama appliance with a generic edge device, you must also implement required security features for your applications and configure the devices to comply with your security requirements. AWS provides guidance on this in the Security Pillar of the AWS Well-Architected Framework. While the framework primarily focuses on cloud applications, most principles also apply to edge devices. In addition, you should use hardware security features of the

chosen solution, such as AWS IoT Greengrass V2 hardware security integration, and use security features provided by the chosen OS and/or device, such as full disk encryption.

Summary

Although AWS Panorama is planning to shut down on May 31, 2026, AWS offers a powerful set of AI/ML services and solutions in the form of Amazon SageMaker tools to build computer vision models and device management services, such as AWS IoT Greengrass, Amazon EKS and Amazon ECS Anywhere and AWS System Manager to support the development of similar solutions. AWS also has a range of offerings from partners in the AWS Partner Network if you prefer to buy instead of build a solution. Example code and implementation guidance is provided to help to migrate to an alternate solution if you so choose. You should explore these options to determine what works best for your specific needs.

For more details, refer to the following resources:

- Amazon SageMaker Developer Guide – Detailed documentation on how to build a model or work with built-in computer vision algorithms available in SageMaker JumpStart.
- AWS IoT Core Developer Guide - Detailed documentation on how to connect and manage IoT Devices.
- AWS IoT Greengrass V2 Developer Guide – Detailed documentation on how to build, deploy and manage IoT applications on your devices.
- ECS Anywhere Developer Guide - Detailed documentation on running ECS at the edge.
- EKS Anywhere Best Practices Guide - Detailed documentation on running EKS at the edge.
- AWS Solutions Library – Partner offerings from a range of providers offering pre-built or customized computer vision solutions.
- Panorama FAQs - Additional Panorama Information.

Frequently Asked Questions

What is the timing for the Panorama discontinuation?

The announcement was made on May 20, 2025. After this date, customers who are not active on the service will no longer have access to Panorama. Active customers will be able to continue to use the service normally until May 31, 2026. Customers have until that time to move their application to an alternative solution and migrate applications of Panorama. After May 31, 2026,

any application that tries to access the Panorama service will no longer work and Panorama devices will no longer function.

How will existing customers be impacted?

Existing customers can continue to use the service normally until May 31, 2026. After that, applications that try to access Panorama will no longer work. Panorama devices will also no longer work after that date.

Are new customers being accepted?

No. As of May 20, 2025, only customers who are active users of Panorama will have access to the service. If a customer has applications in the service from prior usage that they need to access, they can create a case with customer support to request access for their account. If a customer does not have prior usage of the service, they will not be granted access.

What are alternatives customers can explore?

AWS offers a range of services which can replace the Panorama capabilities. We recommend customers utilize off-the-shelf hardware and manage the device and application via the combination of AWS IoT Core, AWS IoT Greengrass, Amazon AKS Anywhere, Amazon ECS Anywhere, and / or AWS System Manager that meets their requirements. The AWS Partner Network also has several solutions available from partners with specific Computer Visions expertise that customers can consider.

How can customers migrate off Panorama?

Panorama applications need to be modified to remove any dependencies on Panorama-specific APIs, which primarily relate to camera connection and streaming. AWS has provided sample code to show how to make these changes. Once those dependencies are removed, the application can be moved to an alternative hardware platform.

If I am having issues on or after May 20, 2025, what support will be available?

AWS will continue to provide support for Panorama up until the end of the discontinuation notification period (May 31, 2026). For any support requirements, customers should enter a support case through their normal support channels. AWS will provide security updates, bug fixes, and availability enhancements.

I cannot migrate before May 31, 2026. Can the date be extended?

We are confident that the alternatives that are available for Panorama enable customers to migrate to an alternative solution by May 31, 2026 and we have no plans to extend availability of the service past that date.

Will my edge application continue to function after the service has ended?

No. The Panorama device and applications are dependent on connectivity to the Panorama cloud service. Once that service is discontinued on May 31, 2026, neither the Panorama application nor the Panorama device will continue to function.

Getting started with AWS Panorama

To get started with AWS Panorama, first learn about the service's concepts and the terminology used in this guide. Then you can use the AWS Panorama console to register your AWS Panorama Appliance and create an application. In about an hour, you can configure the device, update its software, and deploy a sample application. To complete the tutorials in this section, you use the AWS Panorama Appliance and a camera that streams video over a local network.

> **Note**
>
> To purchase an AWS Panorama Appliance, visit the AWS Panorama console.

The AWS Panorama sample application demonstrates use of AWS Panorama features. It includes a model that has been trained with SageMaker AI and sample code that uses the AWS Panorama Application SDK to run inference and output video. The sample application include a AWS CloudFormation template and scripts that show how to automate development and deployment workflows from the command line.

The final two topics in this chapter detail requirements for models and cameras, and the hardware specifications of the AWS Panorama Appliance. If you haven't obtained an appliance and cameras yet, or plan on developing your own computer vision models, see these topics first for more information.

Topics

AWS Panorama concepts

In AWS Panorama, you create computer vision applications and deploy them to the AWS Panorama Appliance or a compatible device to analyze video streams from network cameras. You write application code in Python and build application containers with Docker. You use the AWS Panorama Application CLI to import machine learning models locally or from Amazon Simple Storage Service (Amazon S3). Applications use the AWS Panorama Application SDK to receive video input from a camera and interact with a model.

Concepts

The AWS Panorama Appliance

The AWS Panorama Appliance is the hardware that runs your applications. You use the AWS Panorama console to register an appliance, update its software, and deploy applications to it. The software on the AWS Panorama Appliance connects to camera streams, sends frames of video to your application, and displays video output on an attached display.

The AWS Panorama Appliance is an *edge device* powered by Nvidia Jetson AGX Xavier. Instead of sending images to the AWS Cloud for processing, it runs applications locally on optimized hardware. This enables you to analyze video in real time and process the results locally. The appliance requires an internet connection to report its status, to upload logs, and to perform software updates and deployments.

For more information, see Managing the AWS Panorama Appliance.

Compatible devices

In addition to the AWS Panorama Appliance, AWS Panorama supports compatible devices from AWS Partners. Compatible devices support the same features as the AWS Panorama Appliance. You register and manage compatible devices with the AWS Panorama console and API, and build and deploy applications in the same way.

- Lenovo ThinkEdge® SE70 – Powered by Nvidia Jetson Xavier NX

The content and sample applications in this guide are developed with the AWS Panorama Appliance. For more information about specific hardware and software features for your device, refer to the manufacturer's documentation.

Applications

Applications run on the AWS Panorama Appliance to perform computer vision tasks on video streams. You can build computer vision applications by combining Python code and machine learning models, and deploy them to the AWS Panorama Appliance over the internet. Applications can send video to a display, or use the AWS SDK to send results to AWS services.

To build and deploy applications, you use the AWS Panorama Application CLI. The AWS Panorama Application CLI is a command-line tool that generates default application folders and configuration files, builds containers with Docker, and uploads assets. You can run multiple applications on one device.

For more information, see Managing AWS Panorama applications.

Nodes

An application comprises multiple components called *nodes*, which represent inputs, outputs, models, and code. A node can be configuration only (inputs and outputs), or include artifacts (models and code). An application's code node are bundled in *node packages* that you upload to an Amazon S3 access point, where the AWS Panorama Appliance can access them. An *application manifest* is a configuration file that defines connections between the nodes.

For more information, see Application nodes.

Models

A computer vision model is a machine learning network that is trained to process images. Computer vision models can perform various tasks such as classification, detection, segmentation, and tracking. A computer vision model takes an image as input and outputs information about the image or objects in the image.

AWS Panorama supports models built with PyTorch, Apache MXNet, and TensorFlow. You can build models with Amazon SageMaker AI or in your development environment. For more information, see ???.

Setting up the AWS Panorama Appliance

To get started using your AWS Panorama Appliance or compatible device, register it in the AWS Panorama console and update its software. During the setup process, you create an appliance *resource* in AWS Panorama that represents the physical appliance, and copy files to the appliance with a USB drive. The appliance uses these certificates and configuration files to connect to the AWS Panorama service. Then you use the AWS Panorama console to update the appliance's software and register cameras.

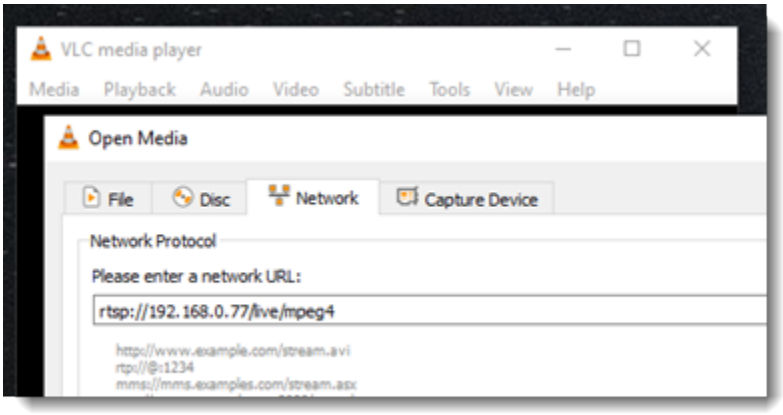
Sections

Prerequisites

To follow this tutorial, you need an AWS Panorama Appliance or compatible device and the following hardware:

- **Display** – A display with HDMI input for viewing the sample application output.
- **USB drive** (included with AWS Panorama Appliance) – A FAT32-formatted USB 3.0 flash memory drive with at least 1 GB of storage, for transferring an archive with configuration files and a certificate to the AWS Panorama Appliance.
- **Camera** – An IP camera that outputs an RTSP video stream.

Use the tools and instructions provided by your camera's manufacturer to identify the camera's IP address and stream path. You can use a video player such as VLC to verify the stream URL, by opening it as a network media source:



The AWS Panorama console uses other AWS services to assemble application components, manage permissions, and verify settings. To register an appliance and deploy the sample application, you need the following permissions:

- AWSPanoramaFullAccess – Provides full access to AWS Panorama, AWS Panorama access points in Amazon S3, appliance credentials in AWS Secrets Manager, and appliance logs in Amazon CloudWatch. Includes permission to create a service-linked role for AWS Panorama.
- **AWS Identity and Access Management (IAM)** – On first run, to create roles used by the AWS Panorama service and the AWS Panorama Appliance.

If you don't have permission to create roles in IAM, have an administrator open the AWS Panorama console and accept the prompt to create service roles.

Register and configure the AWS Panorama Appliance

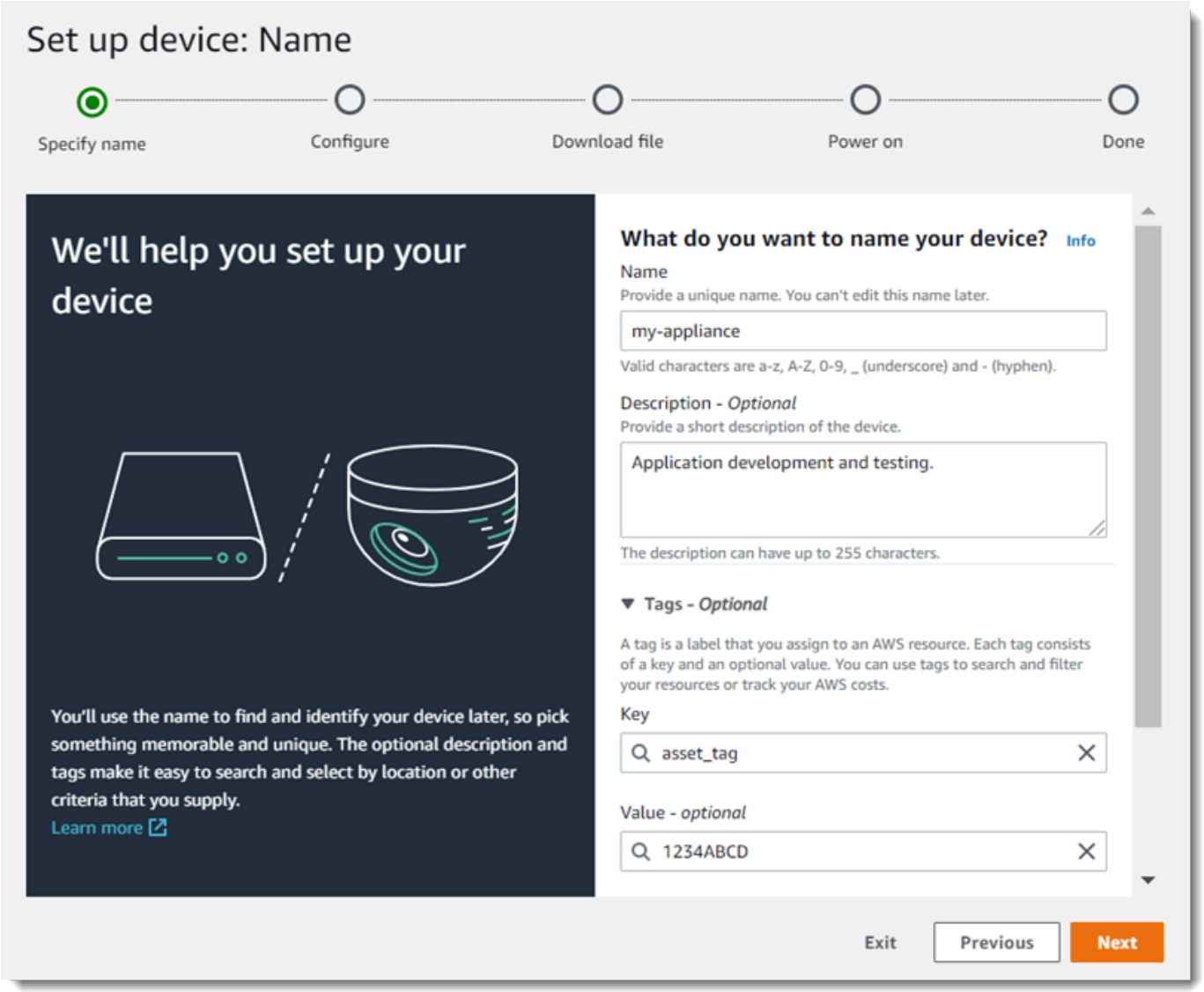
The AWS Panorama Appliance is a hardware device that connects to network-enabled cameras over a local network connection. It uses a Linux-based operating system that includes the AWS Panorama Application SDK and supporting software for running computer vision applications.

To connect to AWS for appliance management and application deployment, the appliance uses a device certificate. You use the AWS Panorama console to generate a provisioning certificate. The appliance uses this temporary certificate to complete initial setup and download a permanent device certificate.

> **Important**
>
> The provisioning certificate that you generate in this procedure is only valid for 5 minutes. If you do not complete the registration process within this time frame, you must start over.

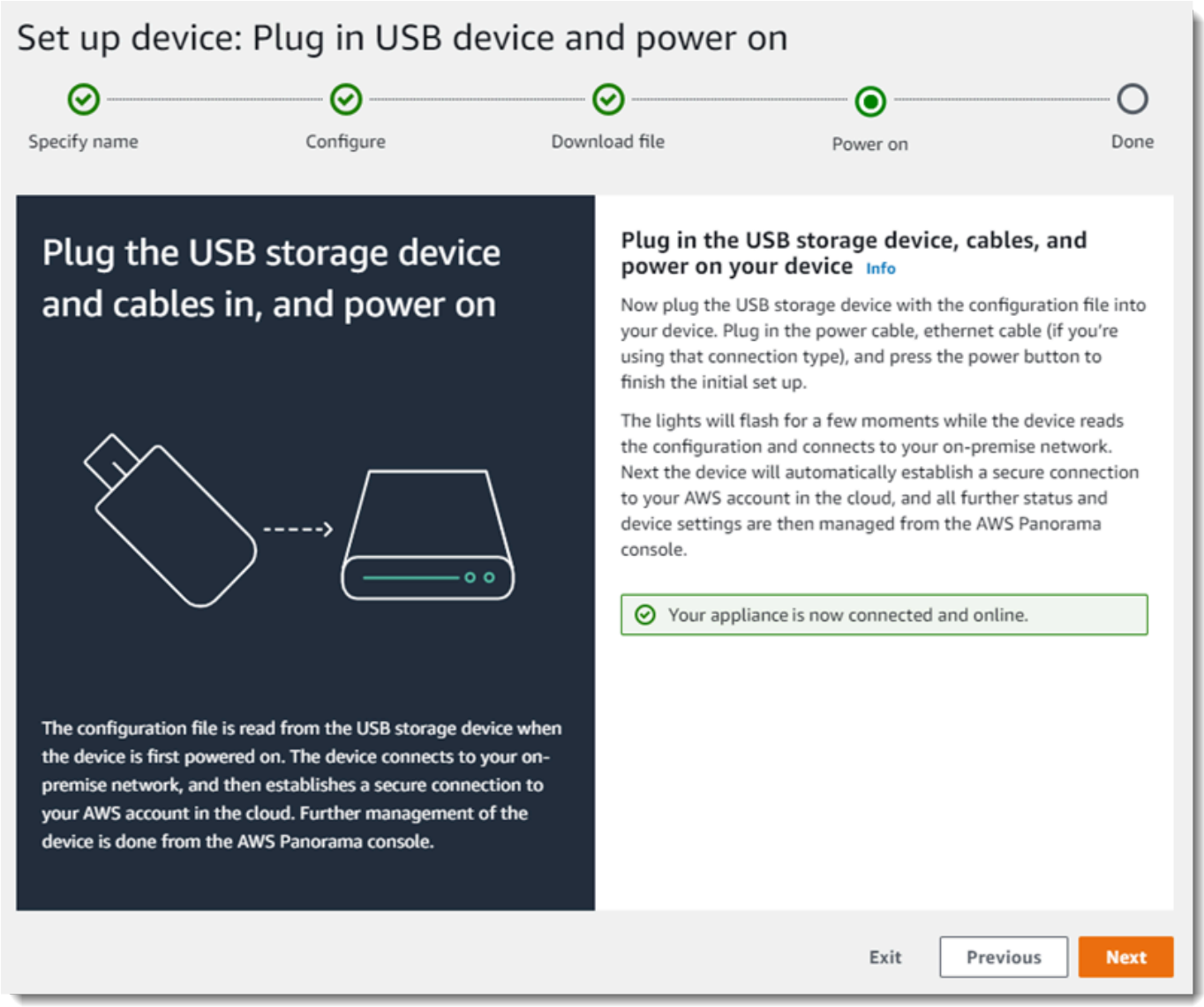
To register a appliance

1. Connect the USB drive to your computer. Prepare the appliance by connecting the network and power cables. The appliance powers on and waits for a USB drive to be connected.
2. Open the AWS Panorama console Getting started page.
3. Choose **Add device**.
4. Choose **Begin setup**.
5. Enter a name and description for the device resource that represents the appliance in AWS Panorama. Choose **Next**



6. If you need to manually assign an IP address, NTP server, or DNS settings, choose **Advanced network settings**. Otherwise, choose **Next**.
7. Choose **Download archive**. Choose **Next**.
8. Copy the configuration archive to the root directory of the USB drive.
9. Connect the USB drive to the USB 3.0 port on the front of the appliance, next to the HDMI port.

   When you connect the USB drive, the appliance copies the configuration archive and network configuration file to itself and connects to the AWS Cloud. The appliance's status light turns from green to blue while it completes the connection, and then back to green.
10. To continue, choose **Next**.



11. Choose **Done**.

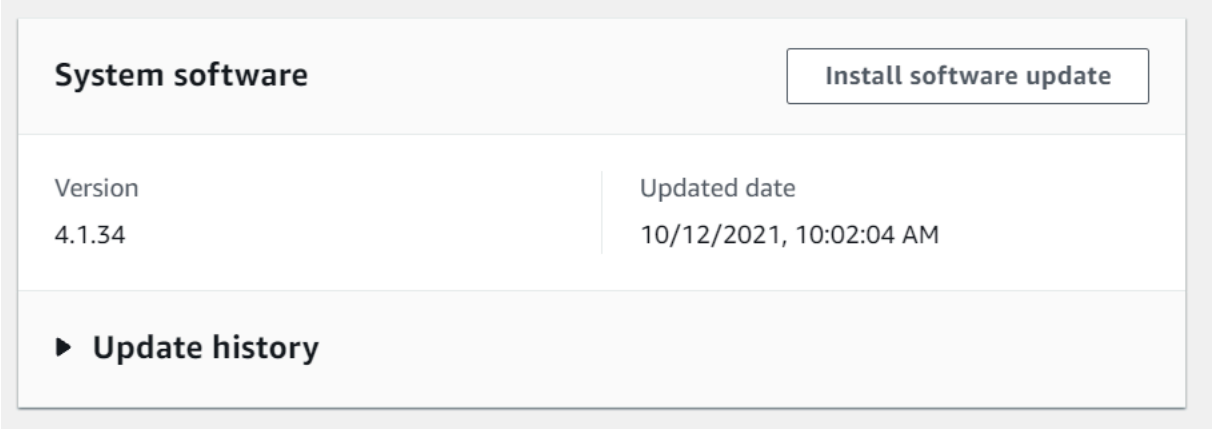
Upgrade the appliance software

The AWS Panorama Appliance has several software components, including a Linux operating system, the AWS Panorama application SDK, and supporting computer vision libraries and frameworks. To ensure that you can use the latest features and applications with your appliance, upgrade its software after setup and whenever an update is available.

To update the appliance software

1. Open the AWS Panorama console Devices page.

2. Choose an appliance.
3. Choose **Settings**
4. Under **System software**, choose **Install software update**.



5. Choose a new version and then choose **Install**.

> **Important**
>
> Before you continue, remove the USB drive from the appliance and format it to delete its contents. The configuration archive contains sensitive data and is not deleted automatically.

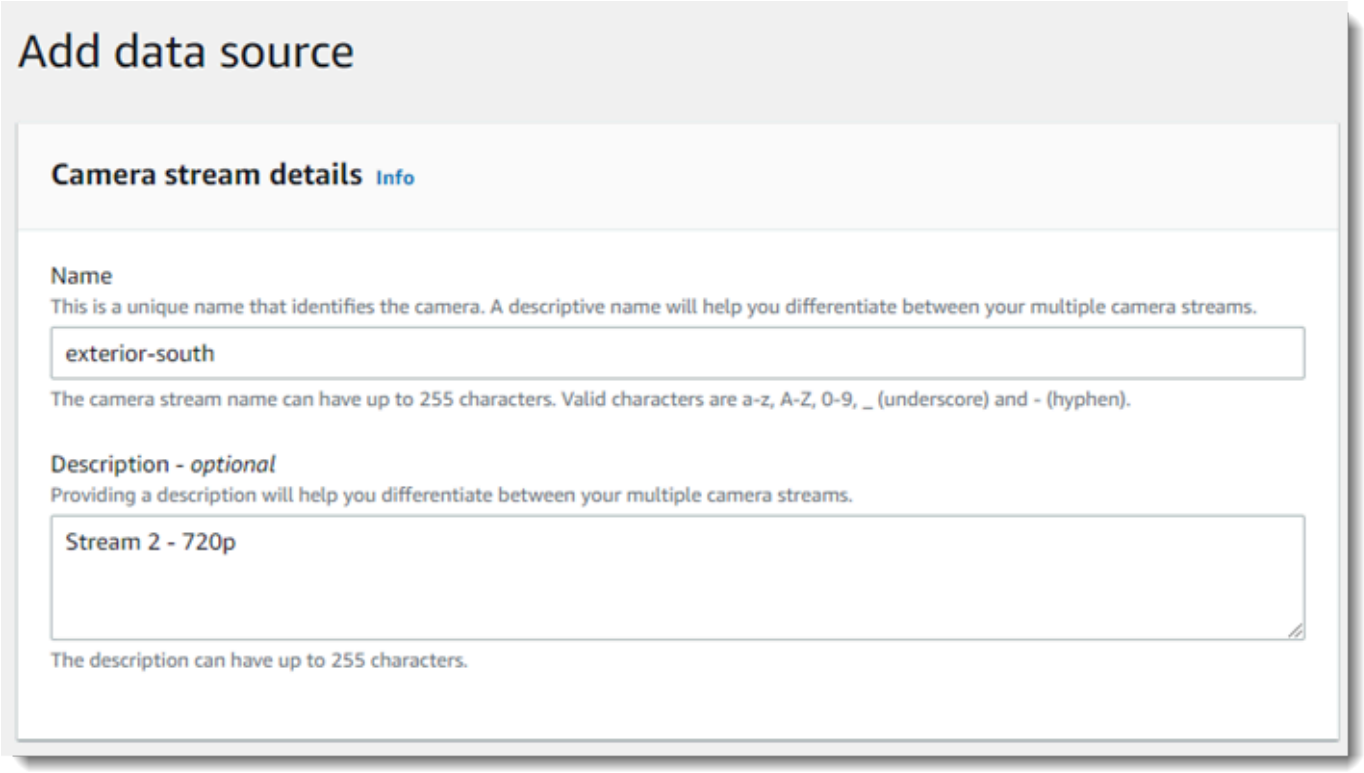
The upgrade process can take 30 minutes or more. You can monitor its progress in the AWS Panorama console or on a connected monitor. When the process completes, the appliance reboots.

Add a camera stream

Next, register a camera stream with the AWS Panorama console.

To register a camera stream

1. Open the AWS Panorama console Data sources page.
2. Choose **Add data source**.



3. Configure the following settings.

   - **Name** – A name for the camera stream.
   - **Description** – A short description of the camera, its location, or other details.
   - **RTSP URL** – A URL that specifies the camera's IP address and the path to the stream. For example, `rtsp://192.168.0.77/live/mpeg4/`
   - **Credentials** – If the camera stream is password protected, specify the username and password.
4. Choose **Save**.

AWS Panorama stores your camera's credentials securely in AWS Secrets Manager. Multiple applications can process the same camera stream simultaneously.

Next steps

If you encountered errors during setup, see Troubleshooting.

To deploy a sample application, continue to the next topic.

Deploying the AWS Panorama sample application

After you've set up your AWS Panorama Appliance or compatible device and upgraded its software, deploy a sample application. In the following sections, you import a sample application with the AWS Panorama Application CLI and deploy it with the AWS Panorama console.

The sample application uses a machine learning model to classify objects in frames of video from a network camera. It uses the AWS Panorama Application SDK to load a model, get images, and run the model. The application then overlays the results on top of the original video and outputs it to a connected display.

In a retail setting, analyzing foot traffic patterns enables you to predict traffic levels. By combining the analysis with other data, you can plan for increased staffing needs around holidays and other events, measure the effectiveness of advertisements and sales promotions, or optimize display placement and inventory management.

Sections

Prerequisites

To follow the procedures in this tutorial, you need a command line terminal or shell to run commands. In the code listings, commands are preceded by a prompt symbol ($) and the name of the current directory, when appropriate.

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

For long commands, we use an escape character (\) to split a command over multiple lines.

On Linux and macOS, use your preferred shell and package manager. On Windows 10, you can install the Windows Subsystem for Linux to get a Windows-integrated version of Ubuntu and Bash. For help setting up a development environment in Windows, see Setting up a development environment in Windows.

You use Python to develop AWS Panorama applications and install tools with pip, Python's package manager. If you don't already have Python, install the latest version. If you have Python 3 but not pip, install pip with your operating system's package manager, or install a new version of Python, which comes with pip.

In this tutorial, you use Docker to build the container that runs your application code. Install Docker from the Docker website: Get Docker

This tutorial uses the AWS Panorama Application CLI to import the sample application, build packages, and upload artifacts. The AWS Panorama Application CLI uses the AWS Command Line Interface (AWS CLI) to call service API operations. If you already have the AWS CLI, upgrade it to the latest version. To install the AWS Panorama Application CLI and AWS CLI, use `pip`.

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

Download the sample application, and extract it into your workspace.

- **Sample application** – aws-panorama-sample.zip

Import the sample application

To import the sample application for use in your account, use the AWS Panorama Application CLI. The application's folders and manifest contain references to a placeholder account number. To update these with your account number, run the `panorama-cli import-application` command.

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

The `SAMPLE_CODE` package, in the `packages` directory, contains the application's code and configuration, including a Dockerfile that uses the application base image, `panorama-application`. To build the application container that runs on the appliance, use the `panorama-cli build-container` command.

```
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
```

The final step with the AWS Panorama Application CLI is to register the application's code and model nodes, and upload assets to an Amazon S3 access point provided by the service. The assets include the code's container image, the model, and a descriptor file for each. To register the nodes and upload assets, run the `panorama-cli package-application` command.

```
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
```

Deploy the application

Use the AWS Panorama console to deploy the application to your appliance.

To deploy the application

1. Open the AWS Panorama console Deployed applications page.
2. Choose **Deploy application**.
3. Paste the contents of the application manifest, `graphs/aws-panorama-sample/graph.json`, into the text editor. Choose **Next**.
4. For **Application name**, enter `aws-panorama-sample`.
5. Choose **Proceed to deploy**.
6. Choose **Begin deployment**.
7. Choose **Next** without selecting a role.
8. Choose **Select device**, and then choose your appliance. Choose **Next**.
9. On the **Select data sources** step, choose **View input(s)**, and add your camera stream as a data source. Choose **Next**.
10. On the **Configure** step, choose **Next**.

11. Choose **Deploy**, and then choose **Done**.
12. In the list of deployed applications, choose **aws-panorama-sample**.

Refresh this page for updates, or use the following script to monitor the deployment from the command line.

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": {}
    }
```

```
]
...
```

If the application doesn't start running, check the application and device logs in Amazon CloudWatch Logs.

View the output

When the deployment is complete, the application starts processing the video stream and sends logs to CloudWatch.

To view logs in CloudWatch Logs

1. Open the Log groups page of the CloudWatch Logs console.
2. Find AWS Panorama application and appliance logs in the following groups:

   - **Device logs** – `/aws/panorama/devices/`*`device-id`*
   - **Application logs** – `/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
```

To view the application's video output, connect the appliance to a monitor with an HDMI cable. By default, the application shows any classification result that has more than 20% confidence.

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",
...
```

The sample model has 1000 classes including many animals, food, and common objects. Try pointing your camera at a keyboard or coffee mug.

For simplicity, the sample application uses a lightweight classification model. The model outputs a single array with a probability for each of its classes. Real-world applications more frequently use object detection models that have multidimensional output. For sample applications with more complex models, see Sample applications, scripts, and templates.

Enable the SDK for Python

The sample application uses the AWS SDK for Python (Boto) to send metrics to Amazon CloudWatch. To enable this functionality, create a role that grants the application permission to send metrics, and redeploy the application with the role attached.

The sample application includes a AWS CloudFormation template that creates a role with the permissions that it needs. To create the role, use the `aws cloudformation deploy` command.

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

To redeploy the application

1. Open the AWS Panorama console Deployed applications page.
2. Choose an application.
3. Choose **Replace**.
4. Complete the steps to deploy the application. In the **Specify IAM role**, choose the role that you created. Its name starts with `aws-panorama-sample-runtime`.
5. When the deployment completes, open the CloudWatch console and view the metrics in the `AWSPanoramaApplication` namespace. Every 150 frames, the application logs and uploads metrics for frame processing and inference time.

Clean up

If you are done working with the sample application, you can use the AWS Panorama console to remove it from the appliance.

To remove the application from the appliance

1. Open the AWS Panorama console Deployed applications page.
2. Choose an application.

3. Choose **Delete from device**.

Next steps

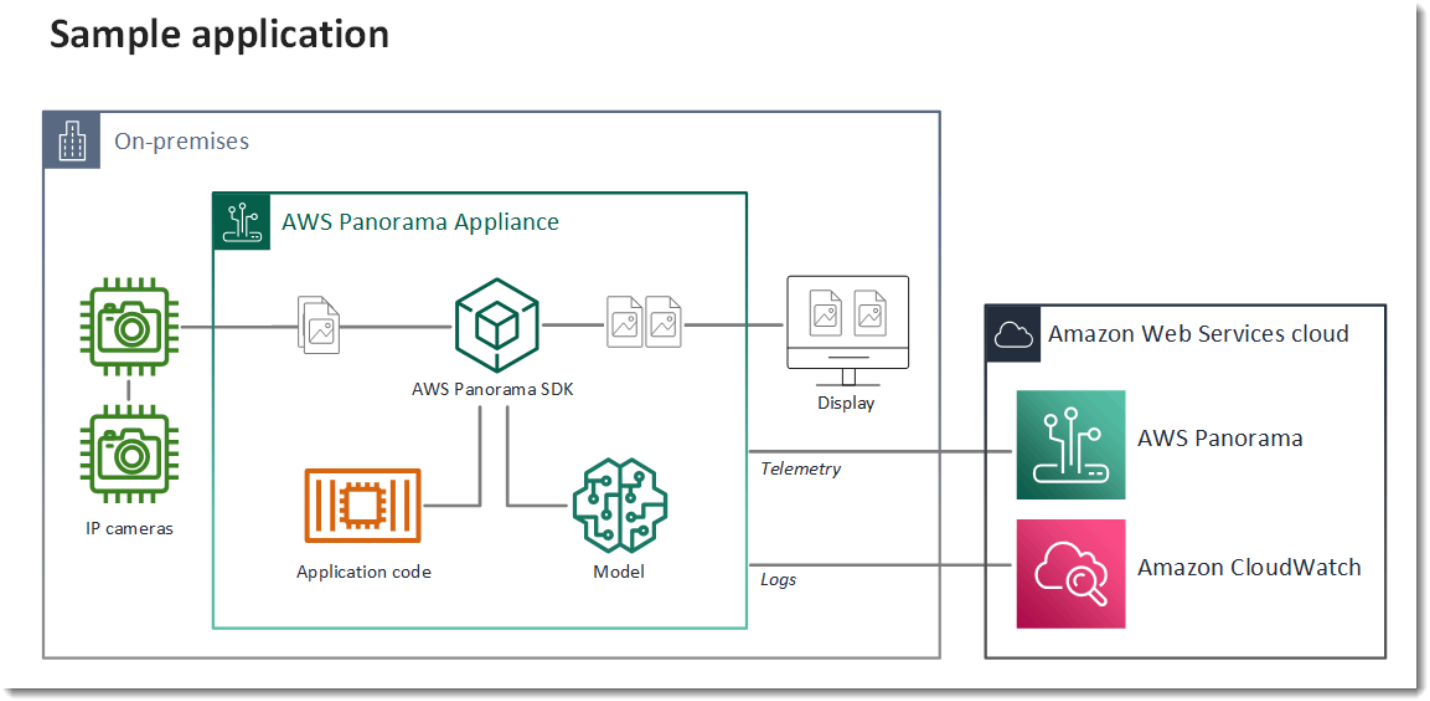
If you encountered errors while deploying or running the sample application, see Troubleshooting.

To learn more about the sample application's features and implementation, continue to the next topic.

Developing AWS Panorama applications

You can use the sample application to learn about AWS Panorama application structure, and as a starting point for your own application.

The following diagram shows the major components of the application running on an AWS Panorama Appliance. The application code uses the AWS Panorama Application SDK to get images and interact with the model, which it doesn't have direct access to. The application outputs video to a connected display but does not send image data outside of your local network.



In this example, the application uses the AWS Panorama Application SDK to get frames of video from a camera, preprocess the video data, and send the data to a computer vision model that detects objects. The application displays the result on an HDMI display connected to the appliance.

Sections

The application manifest

The application manifest is a file named `graph.json` in the `graphs` folder. The manifest defines the application's components, which are packages, nodes, and edges.

Packages are code, configuration, and binary files for application code, models, cameras, and displays. The sample application uses 4 packages:

Example `graphs/aws-panorama-sample/graph.json` – Packages

```
    "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"
        }
    ],
```

The first two packages are defined within the application, in the `packages` directory. They contain the code and model specific to this application. The second two packages are generic camera and display packages provided by the AWS Panorama service. The `abstract_rtsp_media_source` package is a placeholder for a camera that you override during deployment. The `hdmi_data_sink` package represents the HDMI output connector on the device.

Nodes are interfaces to packages, as well as non-package parameters that can have default values that you override at deploy time. The code and model packages define interfaces in `package.json` files that specify inputs and outputs, which can be video streams or a basic data type such as a float, boolean, or string.

For example, the `code_node` node refers to an interface from the `SAMPLE_CODE` package.

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

This interface is defined in the package configuration file, `package.json`. The interface specifies that the package is business logic and that it takes a video stream named `video_in` and a floating point number named `threshold` as inputs. The interface also specifies that the code requires a video stream buffer named `video_out` to output video to a display

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"
```

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

Back in the application manifest, the `camera_node` node represents a video stream from a camera. It includes a decorator that appears in the console when you deploy the application, prompting you to choose a camera stream.

Example `graphs/aws-panorama-sample/graph.json` – Camera node

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

A parameter node, `threshold_param`, defines the confidence threshold parameter used by the application code. It has a default value of 60, and can be overriden during deployment.

Example `graphs/aws-panorama-sample/graph.json` – Parameter node

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

The final section of the application manifest, `edges`, makes connections between nodes. The camera's video stream and the threshold parameter connect to the input of the code node, and the video output from the code node connects to the display.

Example `graphs/aws-panorama-sample/graph.json` – Edges

```
    "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"
        }
    ]
```

Building with the sample application

You can use the sample application as a starting point for your own application.

The name of each package must be unique in your account. If you and another user in your account both use a generic package name such as `code` or `model`, you might get the wrong version of the package when you deploy. Change the name of the code package to one that represents your application.

To rename the code package

1. Rename the package folder: `packages/123456789012-`*`SAMPLE_CODE`*`-1.0/`.
2. Update the package name in the following locations.

   - **Application manifest** – `graphs/aws-panorama-sample/graph.json`
   - **Package configuration** – `packages/123456789012-SAMPLE_CODE-1.0/package.json`
   - **Build script** – `3-build-container.sh`

To update the application's code

1. Modify the application code in `packages/123456789012-SAMPLE_CODE-1.0/src/application.py`.
2. To build the container, run `3-build-container.sh`.

```
aws-panorama-sample$ ./3-build-container.sh
TMPDIR=$(pwd) docker build -t code_asset packages/123456789012-SAMPLE_CODE-1.0
Sending build context to Docker daemon  61.44kB
Step 1/2 : FROM public.ecr.aws/panorama/panorama-application
 ---> 9b197f256b48
Step 2/2 : COPY src /panorama
 ---> 55c35755e9d2
Successfully built 55c35755e9d2
Successfully tagged code_asset:latest
docker export --output=code_asset.tar $(docker create code_asset:latest)
gzip -9 code_asset.tar
Updating an existing asset with the same name
{
    "name": "code_asset",
    "implementations": [
        {
            "type": "container",
            "assetUri":
 "98aaxmpl1c1ef64cde5ac13bd3be5394e5d17064beccee963b4095d83083c343.tar.gz",
            "descriptorUri":
 "1872xmpl129481ed053c52e66d6af8b030f9eb69b1168a29012f01c7034d7a8f.json"
        }
    ]
}
Container asset for the package has been succesfully built at  ~/aws-panorama-
sample-dev/
assets/98aaxmpl1c1ef64cde5ac13bd3be5394e5d17064beccee963b4095d83083c343.tar.gz
```

   The CLI automatically deletes the old container asset from the `assets` folder and updates the package configuration.
3. To upload the packages, run `4-package-application.py`.
4. Open the AWS Panorama console Deployed applications page.
5. Choose an application.
6. Choose **Replace**.

7. Complete the steps to deploy the application. If needed, you can make changes to the application manifest, camera streams, or parameters.

Changing the computer vision model

The sample application includes a computer vision model. To use your own model, modify the model node's configuration, and use the AWS Panorama Application CLI to import it as an asset.

The following example uses an MXNet SSD ResNet50 model that you can download from this guide's GitHub repo: ssd_512_resnet50_v1_voc.tar.gz

To change the sample application's model

1. Rename the package folder to match your model. For example, to `packages/123456789012-SSD_512_RESNET50_V1_VOC-1.0/`.
2. Update the package name in the following locations.

   - **Application manifest** – `graphs/aws-panorama-sample/graph.json`
   - **Package configuration** – `packages/123456789012-SSD_512_RESNET50_V1_VOC-1.0/package.json`
3. In the package configuration file (`package.json`). Change the `assets` value to a blank array.

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

4. Open the package descriptor file (`descriptor.json`). Update the `framework` and `shape` values to match your model.

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

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

The value for **shape**, `1,3,512,512`, indicates the number of images that the model takes as input (1), the number of channels in each image (3--red, green, and blue), and the dimensions of the image (512 x 512). The values and order of the array varies among models.

5. Import the model with the AWS Panorama Application CLI. The AWS Panorama Application CLI copies the model and descriptor files into the `assets` folder with unique names, and updates the package configuration.

```
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. To upload the model, run `panorama-cli package-application`.

```
$ panorama-cli package-application
Uploading package SAMPLE_CODE
Patch Version 1844d5a59150d33f6054b04bac527a1771fd2365e05f990ccd8444a5ab775809
 already registered, ignoring upload
Uploading package SSD_512_RESNET50_V1_VOC
Patch version for the package
 244a63c74d01e082ad012ebf21e67eef5d81ce0de4d6ad1ae2b69d0bc498c8fd
```

```
upload: assets/
b1a1589afe449b346ff47375c284a1998c3e1522b418a7be8910414911784ce1.tar.gz to
 s3://arn:aws:s3:us-west-2:454554846382:accesspoint/panorama-123456789012-
wc66m5eishf4si4sz5jefhx
63a/123456789012/nodePackages/SSD_512_RESNET50_V1_VOC/binaries/
b1a1589afe449b346ff47375c284a1998c3e1522b418a7be8910414911784ce1.tar.gz
upload: assets/
a6a9508953f393f182f05f8beaa86b83325f4a535a5928580273e7fe26f79e78.json to
 s3://arn:aws:s3:us-west-2:454554846382:accesspoint/panorama-123456789012-
wc66m5eishf4si4sz5jefhx63
a/123456789012/nodePackages/SSD_512_RESNET50_V1_VOC/binaries/
a6a9508953f393f182f05f8beaa86b83325f4a535a5928580273e7fe26f79e78.json
{
    "ETag": "\"2381dabba34f4bc0100c478e67e9ab5e\"",
    "ServerSideEncryption": "AES256",
    "VersionId": "KbY5fpESdpYamjWZ0YyGqHo3.LQQWUC2"
}
Registered SSD_512_RESNET50_V1_VOC with patch version
 244a63c74d01e082ad012ebf21e67eef5d81ce0de4d6ad1ae2b69d0bc498c8fd
Uploading package SQUEEZENET_PYTORCH_V1
Patch Version 568138c430e0345061bb36f05a04a1458ac834cd6f93bf18fdacdffb62685530
 already registered, ignoring upload
```

7. Update the application code. Most of the code can be reused. The code specific to the model's response is in the `process_results` method.

```
   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)
```

   Depending on your model, you might also need to update the `preprocess` method.

Preprocessing images

Before the application sends an image to the model, it prepares it for inference by resizing it and normalizing color data. The model that the application uses requires a 224 x 224 pixel image with

three color channels, to match the number of inputs in its first layer. The application adjusts each color value by converting it to a number between 0 and 1, subtracting the average value for that color, and dividing by the standard deviation. Finally, it combines the color channels and converts it to a NumPy array that the model can process.

Example application.py – Preprocessing

```
    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)
```

This process gives the model values in a predictable range centered around 0. It matches the preprocessing applied to images in the training dataset, which is a standard approach but can vary per model.

Uploading metrics with the SDK for Python

The sample application uses the SDK for Python to upload metrics to Amazon CloudWatch.

Example application.py – SDK for Python

```
    def process_streams(self):
        """Processes one frame of video from one or more video streams."""
        ...
            logger.info('epoch length: {:.3f} s ({:.3f} FPS)'.format(epoch_time,
epoch_fps))
            logger.info('avg inference time: {:.3f} ms'.format(avg_inference_time))
```

```
            logger.info('max inference time: {:.3f} ms'.format(max_inference_time))
            logger.info('avg frame processing time: {:.3f}
ms'.format(avg_frame_processing_time))
            logger.info('max frame processing time: {:.3f}
ms'.format(max_frame_processing_time))
            self.inference_time_ms = 0
            self.inference_time_max = 0
            self.frame_time_ms = 0
            self.frame_time_max = 0
            self.epoch_start = time.time()
            self.put_metric_data('AverageInferenceTime', avg_inference_time)
            self.put_metric_data('AverageFrameProcessingTime',
avg_frame_processing_time)

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

```
            logger.warning("CloudWatch client is not available.")
```

It gets permission from a runtime role that you assign during deployment. The role is defined in the `aws-panorama-sample.yml` AWS CloudFormation template.

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/
```

The sample application installs the SDK for Python and other dependencies with pip. When you build the application container, the `Dockerfile` runs commands to install libraries on top of what comes with the base image.

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
```

To use the AWS SDK in your application code, first modify the template to add permissions for all API actions that the application uses. Update the AWS CloudFormation stack by running the `1-create-role.sh` each time you make a change. Then, deploy changes to your application code.

For actions that modify or use existing resources, it is a best practice to minimize the scope of this policy by specifying a name or pattern for the target `Resource` in a separate statement. For details on the actions and resources supported by each service, see Action, resources, and condition keys in the Service Authorization Reference

Next steps

For instructions on using the AWS Panorama Application CLI to build applications and create packages from scratch, see the CLI's README.

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

For more sample code and a test utility that you can use to validate your application code prior to deploying, visit the AWS Panorama samples repository.

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

Supported computer vision models and cameras

AWS Panorama supports models built with PyTorch, Apache MXNet, and TensorFlow. When you deploy an application, AWS Panorama compiles your model in SageMaker AI Neo. You can build models in Amazon SageMaker AI or in your development environment, as long as you use layers that are compatible with SageMaker AI Neo.

To process video and get images to send to a model, the AWS Panorama Appliance connects to an H.264 encoded video stream with the RTSP protocol. AWS Panorama tests a variety of common cameras for compatibility.

Sections

Supported models

When you build an application for AWS Panorama, you provide a machine learning model that the application uses for computer vision. You can use pre-built and pre-trained models provided by model frameworks, a sample model, or a model that you build and train yourself.

> **Note**
>
> For a list of pre-built models that have been tested with AWS Panorama, see Model compatibility.

When you deploy an application, AWS Panorama uses the SageMaker AI Neo compiler to compile your computer vision model. SageMaker AI Neo is a compiler that optimizes models to run efficiently on a target platform, which can be an instance in Amazon Elastic Compute Cloud (Amazon EC2), or an edge device such as the AWS Panorama Appliance.

AWS Panorama supports the versions of PyTorch, Apache MXNet, and TensorFlow that are supported for edge devices by SageMaker AI Neo. When you build your own model, you can use the framework versions listed in the SageMaker AI Neo release notes. In SageMaker AI, you can use the built-in image classification algorithm.

For more information about using models in AWS Panorama, see Computer vision models.

Supported cameras

The AWS Panorama Appliance supports H.264 video streams from cameras that output RTSP over a local network. For camera streams greater than 2 megapixels, the appliance scales down the image to 1920x1080 pixels or an equivalent size that preserves the stream's aspect ratio.

The following camera models have been tested for compatibility with the AWS Panorama Appliance:

- 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

For the appliance's hardware specifications, see AWS Panorama Appliance specifications.

AWS Panorama Appliance specifications

The AWS Panorama Appliance has the following hardware specifications. For other compatible devices, refer to the manufacturer's documentation.

Component	Specification
Processor and GPU	Nvidia Jetson AGX Xavier with 32GB RAM
Ethernet	2x 1000 Base-T (Gigabyte)
USB	1x USB 2.0 and 1x USB 3.0 type-A female
HDMI output	2.0a
Dimensions	7.75″ x 9.6″ x 1.6″ (197mm x 243mm x 40mm)
Weight	3.7lbs (1.7kg)
Power supply	100V-240V 50-60Hz AC 65W
Power input	IEC 60320 C6 (3-pin) receptacle
Dust and liquid protection	IP-62
EMI/EMC regulatory compliance	FCC Part-15 (US)
Thermal touch limits	IEC-62368
Operating temperature	-20°C to 60°C
Operating humidity	0% to 95% RH
Storage temperature	-20°C to 85°C
Storage humidity	Uncontrolled for low temperature. 90% RH at high temperature
Cooling	Forced-air heat extraction (fan)
Mounting options	Rackmount or free standing

Component	Specification
Power cord	6-foot (1.8 meter)
Power control	Push-button
Reset	Momentary switch
Status and network LEDs	Programmable 3-color RGB LED

Wi-Fi, Bluetooth and SD card storage are present on the appliance but are not usable.

The AWS Panorama Appliance includes two screws for mounting on a server rack. You can mount two appliances side-by-side on a 19-inch rack.

Service quotas

AWS Panorama applies quotas to the resources that you create in your account and the applications that you deploy. If you use AWS Panorama in multiple AWS Regions, quotas apply separately to each Region. AWS Panorama quotas are not adjustable.

Resources in AWS Panorama include devices, application node packages, and application instances.

- **Devices** – Up to 50 registered appliances per Region.
- **Node packages** – 50 packages per Region, with up to 20 versions per package.
- **Application instances** – Up to 10 applications per device. Each application can monitor up to 8 camera streams. Deployments are limited to 200 per day for each device.

When you use the AWS Panorama Application CLI, AWS Command Line Interface, or AWS SDK with the AWS Panorama service, quotas apply to the number of API calls that you make. You can make up to 5 requests total per second. A subset of API operations that create or modify resources apply an additional limit of 1 request per second.

For a complete list of quotas, visit the Service Quotas console, or see AWS Panorama endpoints and quotas in the Amazon Web Services General Reference.

AWS Panorama permissions

You can use AWS Identity and Access Management (IAM) to manage access to the AWS Panorama service and resources like appliances and applications. For users in your account that use AWS Panorama, you manage permissions in a permissions policy that you can apply to IAM roles. To manage permissions for an application, you create a role and assign it to the application.

To manage permissions for users in your account, use the managed policy that AWS Panorama provides, or write your own. You need permissions to other AWS services to get application and appliance logs, view metrics, and assign a role to an application.

An AWS Panorama Appliance also has a role that grants it permission to access AWS services and resources. The appliance's role is one of the service roles that the AWS Panorama service uses to access other services on your behalf.

An application role is a separate service role that you create for an application, to grant it permission to use AWS services with the AWS SDK for Python (Boto). To create an application role, you need administrative privileges or the help of an administrator.

You can restrict user permissions by the resource an action affects and, in some cases, by additional conditions. For example, you can specify a pattern for the Amazon Resource Name (ARN) of an application that requires a user to include their user name in the name of applications that they create. For the resources and conditions that are supported by each action, see Actions, resources, and condition keys for AWS Panorama in the Service Authorization Reference.

For more information, see What is IAM? in the IAM User Guide.

Topics

Identity-based IAM policies for AWS Panorama

To grant users in your account access to AWS Panorama, you use identity-based policies in AWS Identity and Access Management (IAM). Apply identity-based policies to IAM roles that are associated with a user. You can also grant users in another account permission to assume a role in your account and access your AWS Panorama resources.

AWS Panorama provides managed policies that grant access to AWS Panorama API actions and, in some cases, access to other services used to develop and manage AWS Panorama resources. AWS Panorama updates the managed policies as needed, to ensure that your users have access to new features when they're released.

- **AWSPanoramaFullAccess** – Provides full access to AWS Panorama, AWS Panorama access points in Amazon S3, appliance credentials in AWS Secrets Manager, and appliance logs in Amazon CloudWatch. Includes permission to create a service-linked role for AWS Panorama. View policy

The `AWSPanoramaFullAccess` policy allows you to tag AWS Panorama resources, but does not have all tag-related permissions used by the AWS Panorama console. To grant these permissions, add the following policy.

- **ResourceGroupsandTagEditorFullAccess** – View policy

The `AWSPanoramaFullAccess` policy does not include permission to purchase devices from the AWS Panorama console. To grant these permissions, add the following policy.

- **ElementalAppliancesSoftwareFullAccess** – View policy

Managed policies grant permission to API actions without restricting the resources that a user can modify. For finer-grained control, you can create your own policies that limit the scope of a user's permissions. Use the full-access policy as a starting point for your policies.

> **Creating service roles**
>
> The first time you use the AWS Panorama console, you need permission to create the service role used by the AWS Panorama Appliance. A service role gives a service permission to manage resources or interact with other services. Create this role before granting access to your users.

For details on the resources and conditions that you can use to limit the scope of a user's permissions in AWS Panorama, see Actions, resources, and condition keys for AWS Panorama in the Service Authorization Reference.

AWS Panorama service roles and cross-service resources

AWS Panorama uses other AWS services to manage the AWS Panorama Appliance, store data, and import application resources. A service role gives a service permission to manage resources or interact with other services. When you sign in to the AWS Panorama console for the first time, you create the following service roles:

- **AWSServiceRoleForAWSPanorama** – Allows AWS Panorama to manage resources in AWS IoT, AWS Secrets Manager, and AWS Panorama.

  Managed policy: AWSPanoramaServiceLinkedRolePolicy
- **AWSPanoramaApplianceServiceRole** – Allows an AWS Panorama Appliance to upload logs to CloudWatch, and to get objects from Amazon S3 access points created by AWS Panorama.

  Managed policy: AWSPanoramaApplianceServiceRolePolicy

To view the permissions attached to each role, use the IAM console. Wherever possible, the role's permissions are restricted to resources that match a naming pattern that AWS Panorama uses. For example, `AWSServiceRoleForAWSPanorama` grants only permission for the service to access AWS IoT resources that have `panorama` in their name.

Sections

Securing the appliance role

The AWS Panorama Appliance uses the `AWSPanoramaApplianceServiceRole` role to access resources in your account. The appliance has permission to upload logs to CloudWatch Logs, read camera stream credentials from AWS Secrets Manager, and to access application artifacts in Amazon Simple Storage Service (Amazon S3) access points that AWS Panorama creates.

> **Note**
>
> Applications don't use the appliance's permissions. To give your application permission to use AWS services, create an application role.

AWS Panorama uses the same service role with all appliances in your account, and does not use roles across accounts. For an added layer of security, you can modify the appliance role's trust policy to enforce this explicitly, which is a best practice when you use roles to grant a service permission to access resources in your account.

To update the appliance role trust policy

1. Open the appliance role in the IAM console: AWSPanoramaApplianceServiceRole
2. Choose **Edit trust relationship**.
3. Update the policy contents and then choose **Update trust policy**.

The following trust policy includes a condition that ensures that when AWS Panorama assumes the appliance role, it is doing so for an appliance in your account. The `aws:SourceAccount` condition compares the account ID specified by AWS Panorama to the one that you include in the policy.

Example trust policy – Specific account

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

If you want to restrict AWS Panorama further, and allow it to only assume the role with a specific device, you can specify the device by ARN. The `aws:SourceArn` condition compares the ARN of the appliance specified by AWS Panorama to the one that you include in the policy.

Example trust policy – Single appliance

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

If you reset and reprovision the appliance, you must remove the source ARN condition temporarily and then add it again with the new device ID.

For more information on these conditions, and security best practices when services use roles to access resources in your account, see The confused deputy problem in the IAM User Guide.

Use of other services

AWS Panorama creates or accesses resources in the following services:

- AWS IoT – Things, policies, certificates, and jobs for the AWS Panorama Appliance
- Amazon S3 – Access points for staging application models, code, and configurations.
- Secrets Manager – Short-term credentials for the AWS Panorama Appliance.

For information about Amazon Resource Name (ARN) format or permission scopes for each service, see the topics in the *IAM User Guide* that are linked to in this list.

Granting permissions to an application

You can create a role for your application to grant it permission to call AWS services. By default, applications do not have any permissions. You create an application role in IAM and assign it to an application during deployment. To grant your application only the permissions that it needs, create a role for it with permissions for specific API actions.

The sample application includes an AWS CloudFormation template and script that create an application role. It is a service role that AWS Panorama can assume. This role grants permission for the application to call CloudWatch to upload metrics.

Example aws-panorama-sample.yml – Application role

```
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/
```

You can extend this script to grant permissions to other services, by specifying a list of API actions or patterns for the value of `Action`.

For more information on permissions in AWS Panorama, see AWS Panorama permissions.

Managing the AWS Panorama Appliance

The AWS Panorama Appliance is the hardware that runs your applications. You use the AWS Panorama console to register an appliance, update its software, and deploy applications to it. The software on the AWS Panorama Appliance connects to camera streams, sends frames of video to your application, and displays video output on an attached display.

After setting up your appliance or another compatible device, you register cameras for use with applications. You manage camera streams in the AWS Panorama console. When you deploy an application, you choose which camera streams the appliance sends to it for processing.

For tutorials that introduce the AWS Panorama Appliance with a sample application, see Getting started with AWS Panorama.

Topics

Managing an AWS Panorama Appliance

You use the AWS Panorama console to configure, upgrade or deregister the AWS Panorama Appliance and other compatible devices.

To set up an appliance, follow the instructions in the getting started tutorial. The setup process creates the resources in AWS Panorama that track your appliance and coordinate updates and deployments.

To register an appliance with the AWS Panorama API, see Automate device registration.

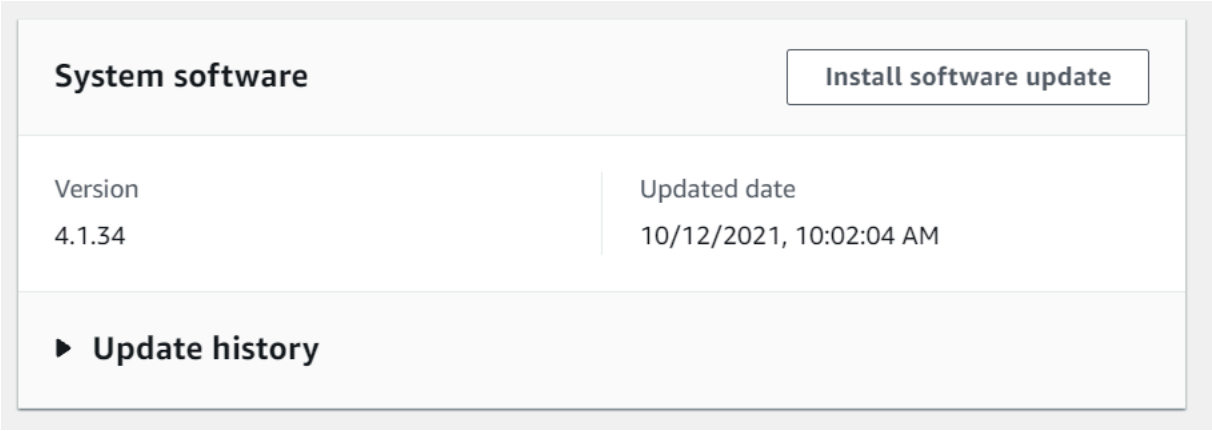
Sections

Update the appliance software

You view and deploy software updates for the appliance in the AWS Panorama console. Updates can be required or optional. When a required update is available, the console prompts you to apply it. You can apply optional updates on the appliance **Settings** page.

To update the appliance software

1. Open the AWS Panorama console Devices page.
2. Choose an appliance.
3. Choose **Settings**
4. Under **System software**, choose **Install software update**.



5. Choose a new version and then choose **Install**.

Deregister an appliance

If you are done working with an appliance, you can use the AWS Panorama console to deregister it and delete the associated AWS IoT resources.

To delete an appliance

1. Open the AWS Panorama console Devices page.
2. Choose the appliance's name.
3. Choose **Delete**.
4. Enter the appliance's name and choose **Delete**.

When you delete an appliance from the AWS Panorama service, data on the appliance is not deleted automatically. A deregistered appliance can't connect to AWS services and can't be registered again until it is reset.

Reboot an appliance

You can reboot an appliance remotely.

To reboot an appliance

1. Open the AWS Panorama console Devices page.
2. Choose the appliance's name.
3. Choose **Reboot**.

The console sends a message to the appliance to reboot it. To receive the signal, the appliance must be able to connect to AWS IoT. To reboot an appliance with the AWS Panorama API, see Reboot appliances.

Reset an appliance

To use an appliance in a different Region or with a different account, you must reset it and reprovision it with a new certificate. Resetting the device applies the most recent required software version and deletes all account data.

To start a reset operation, the appliance must be plugged in and powered down. Press and hold both the power and reset buttons for five seconds. When you release the buttons, the status light blinks orange. Wait until the status light blinks green before provisioning or disconnecting the appliance.

You can also reset the appliance software without deleting certificates from the device. For more information, see Power and reset buttons.

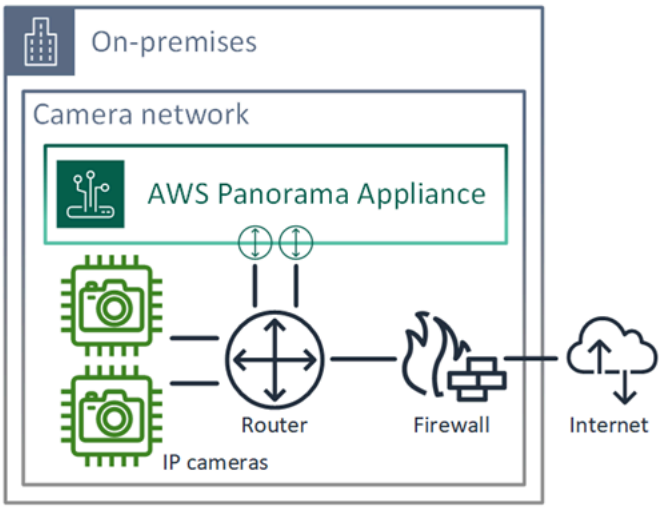
Connecting the AWS Panorama Appliance to your network

The AWS Panorama Appliance requires connectivity to both the AWS cloud and your on-premises network of IP cameras. You can connect the appliance to a single firewall that grants access to both, or connect each of the device's two network interfaces to a different subnet. In either case, you must secure the appliance's network connections to prevent unauthorized access to your camera streams.

Sections

Single network configuration

The appliance has two Ethernet ports. If you route all traffic to and from the device through a single router, you can use the second port for redundancy in case the physical connection to the first port is broken. Configure your router to allow the appliance to connect only to camera streams and the internet, and to block camera streams from otherwise leaving your internal network.

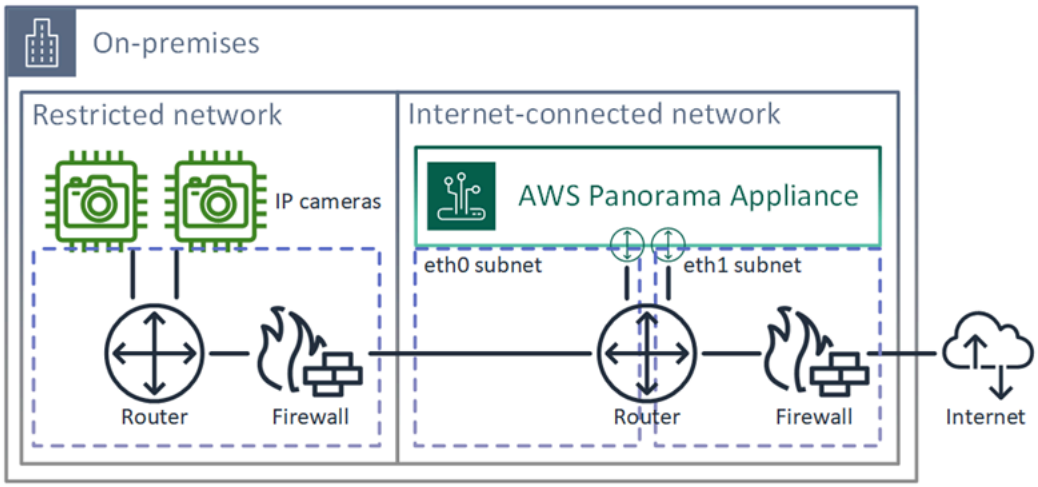


For details on the ports and endpoints that the appliance needs access to, see Configuring service access and Configuring local network access.

Dual network configuration

For an extra layer of security, you can place the appliance in an internet-connected network separate from your camera network. A firewall between your restricted camera network and the appliance's network only allows the appliance to access video streams. If your camera network was previously air-gapped for security purposes, you might prefer this method over connecting the camera network to a router that also grants access to the internet.

The following example shows the appliance connecting to a different subnet on each port. The router places the `eth0` interface on a subnet that routes to the camera network, and `eth1` on a subnet that routes to the internet.



You can confirm the IP address and MAC address of each port in the AWS Panorama console.

Configuring service access

During provisioning, you can configure the appliance to request a specific IP address. Choose an IP address ahead of time to simplify firewall configuration and ensure that the appliance's address doesn't change if it's offline for a long period of time.

The appliance uses AWS services to coordinate software updates and deployments. Configure your firewall to allow the appliance to connect to these endpoints.

Internet access

- **AWS IoT (HTTPS and MQTT, ports 443, 8443 and 8883)** – AWS IoT Core and device management endpoints. For details, see AWS IoT Device Management endpoints and quotas in the Amazon Web Services General Reference.