Unable to locate subtitle

AWS Snowball Edge Developer Guide

AWS Snowball Edge Developer Guide: ***Unable to locate subtitle***

Table of Contents

What is AWS Snowball Edge?

AWS Snowball Edge is a type of Snowball device with on-board storage and compute power for select AWS capabilities. Snowball Edge can do local processing and edge-computing workloads in addition to transferring data between your local environment and the AWS Cloud.

Each Snowball Edge device can transport data at speeds faster than the internet. This transport is done by shipping the data in the appliances through a regional carrier. The appliances are rugged, complete with E Ink shipping labels.

Snowball Edge devices have four options for device configurations—*Storage Optimized*, *Compute Optimized*, *Compute Optimized with GPU*, and *Import virtual tapes into AWS Storage Gateway*. When this guide refers to Snowball Edge devices, it's referring to all options of the device. When specific information applies only to one or more optional configurations of devices (such as how the Snowball Edge with GPU has an on-board GPU), it is called out specifically. For more information, see Snowball Edge device options.

Topics

AWS Snowball Edge features

Snowball Edge devices have the following features:

- Large amounts of storage capacity or compute functionality for devices. This depends on the options you choose when you create your job.
- Network adapters with transfer speeds of up to 100 Gbit/second.
- Encryption is enforced, protecting your data at rest and in physical transit.

- You can import or export data between your local environments and Amazon S3, and physically transport the data with one or more devices without using the internet.
- Snowball Edge devices are their own rugged box. The built-in E Ink display changes to show your shipping label when the device is ready to ship.
- Snowball Edge devices come with an on-board LCD display that can be used to manage network connections and get service status information.
- You can cluster Snowball Edge devices for local storage and compute jobs to achieve data durability across 3 to 16 devices and locally grow or shrink storage on demand.
- You can use Amazon EKS Anywhere on Snowball Edge devices for Kubernetes workloads.
- Snowball Edge devices have Amazon S3 and Amazon EC2 compatible endpoints available, enabling programmatic use cases.
- Snowball Edge devices support the new `sbe1`, `sbe-c`, and `sbe-g` instance types, which you can use to run compute instances on the device using Amazon Machine Images (AMIs).
- Snowball Edge supports these data transfer protocols for data migration:
  - NFSv3
  - NFSv4
  - NFSv4.1
  - Amazon S3 over HTTP or HTTPS (via API compatible with AWS CLI version 1.16.14 and earlier)

Prerequisites for using Snow Family devices

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign

administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create an administrative user

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create an administrative user

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to an administrative user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the administrative user

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Prerequisites for using Amazon S3 adapter on Snow Family devices for import and export jobs

You will use S3 adapter on Snow Family devices when you are using the devices to move data from on-premises data sources to the cloud or from the cloud to on-premises data storage.

> **Note**
>
> You must select S3 adapter on Snow when you order devices. See Step 2: Choose your compute and storage options in this guide.

The Amazon S3 bucket associated with the job must use the Amazon S3 standard storage class. Before creating your first job, keep the following in mind.

For jobs that import data into Amazon S3, follow these steps:

- Confirm that the files and folders to transfer are named according to the object key naming guidelines for Amazon S3. Any files or folders with names that don't meet these guidelines aren't imported into Amazon S3.
- Plan what data you want to import into Amazon S3. For more information, see Planning your large transfer.

Before exporting data from Amazon S3, follow these steps:

- Understand what data is exported when you create your job. For more information, see Using Export Ranges.
- For any files with a colon (:) in the file name, change the file names in Amazon S3 before you create the export job to get these files. Files with a colon in the file name fail export to Microsoft Windows Server.

Prerequisites for using Amazon S3 compatible storage on Snow Family devices

You will use Amazon S3 compatible storage on Snow Family devices when you are storing data on the device at your edge location and using the data for local compute operations. To migrate data to or from AWS, set up an export or import job and use the Amazon S3 adapter.

When ordering a Snow device for local compute and storage with Amazon S3 compatible storage, keep the following in mind.

- You will provision Amazon S3 storage capacity when you order the device. So consider your storage need before ordering a device.
- You can create Amazon S3 buckets on the device after you receive it rather than while ordering a Snow Family device.
- You will need to download the latest version of the AWS CLI (v2.11.15 or higher), Snowball Edge client, or AWS OpsHub and install it on your computer to use Amazon S3 compatible storage on Snow Family devices.
- After receiving your device, configure, start, and use Amazon S3 compatible storage on Snow Family devices according to Using Amazon S3 compatible storage on Snow Family devices in this guide.

Prerequisites for using compute instances on Snow Family devices

For jobs using compute instances, before you can add any AMIs to your job, you must have an AMI in your AWS account and it must be a supported image type. Currently, supported AMIs are based on these operating systems:

- Amazon Linux 2
- CentOS 7 (x86_64) - with Updates HVM
- Ubuntu 16.04 LTS - Xenial (HVM)
- Ubuntu 20.04 LTS - Focal
- Ubuntu 22.04 LTS - Jammy
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016

- Microsoft Windows Server 2019

> **Note**
>
> Ubuntu 16.04 LTS - Xenial (HVM) images are no longer supported in the AWS Marketplace, but still supported for use on Snowball Edge devices through Amazon EC2 VM Import/ Export and running locally in AMIs.

You can get these images from AWS Marketplace.

If you're using SSH to connect to the instances running on a Snowball Edge, you can use your own key pair or you can create one on the Snowball Edge. To use AWS OpsHub to create a key pair on the device, see Working with key pairs. To use the AWS CLI to create a key pair on the device, see `create-key-pair` in List of Supported Amazon EC2-compatible AWS CLI Commands on a Snowball Edge. For more information on key pairs and Amazon Linux 2, see Amazon EC2 key pairs and Linux instances in the Amazon EC2 User Guide for Linux Instances.

For information specific to using compute instances on a device, see Using Amazon EC2-compatible compute instances.

Services related to AWS Snowball Edge

You can use an AWS Snowball Edge device with the following related AWS services:

- **Amazon S3 adapter** — Use for programmatic data transfer in to and out of AWS using the Amazon S3 API for Snowball Edge, which supports a subset of Amazon S3 API operations. In this role, data is transferred to the Snow device by AWS on your behalf and the device is shipped to you (for an export job), or AWS ships an empty Snow device to you and you transfer data from your on-premises sources to the device and ship it back to AWS (for an import job)"
- **Amazon S3 compatible storage on Snow Family devices** — Use to support the data needs of compute services such as Amazon EC2, Amazon EKS Anywhere on Snow, and others. This feature is available on Snowball Edge devices and provides an expanded Amazon S3 API set and features such as increased resiliency with flexible cluster setup for 3 to 16 nodes, local bucket management, and local notifications.
- **Amazon EC2** – Run compute instances on a Snowball Edge device using the Amazon EC2 compatible endpoint, which supports a subset of the Amazon EC2 API operations. For more

information about using Amazon EC2 in AWS, see Getting started with Amazon EC2 Linux instances.

- **Amazon EKS Anywhere on Snow** – Create and operate Kubernetes clusters on Snow Family devices. See Using Amazon EKS Anywhere on AWS Snow.
- **AWS Lambda powered by AWS IoT Greengrass** – Invoke Lambda functions based on Amazon S3 compatible storage on Snow Family devices storage actions made on an AWS Snowball Edge device. For more information about using Lambda, see Using AWS Lambda with an AWS Snowball Edge and the AWS Lambda Developer Guide.
- **Amazon Elastic Block Store (Amazon EBS)** – Provide block-level storage volumes for use with EC2-compatible instances. For more information, see Amazon Elastic Block Store (Amazon EBS).
- **AWS Identity and Access Management (IAM)** – Use this service to securely control access to AWS resources. For more information, see What is IAM?
- **AWS Security Token Service (AWS STS)** – Request temporary, limited-privilege credentials for IAM users or for users that you authenticate (federated users). For more information, see Temporary security credentials in IAM.
- **Amazon EC2 Systems Manager** – Use this service to view and control your infrastructure on AWS. For more information, see What is AWS Systems Manager?

Accessing the service

You can either use the AWS Snow Family Management Console or the job management API to create and manage jobs. For information about the job management API, see Job Management API Reference for AWS Snowball.

Accessing an AWS Snowball Edge device

After your Snowball Edge device is onsite, you can configure it with an IP address using the LCD screen then you can unlock the device using the Snowball Edge client or AWS OpsHub for Snow Family. Then, you run can perform data transfer or edge compute tasks. For more information, see Using an AWS Snowball Edge Device.

Pricing for the AWS Snowball Edge

For information about the pricing and fees associated with the service and its devices, see AWS Snowball Edge Pricing.

Device monitoring

AWS will monitor the Snow device and may collect metrics and usage information when the Snow device is connected to an AWS Region. If the Snow device is not connected to the AWS Region, then AWS will not monitor the Snow device.

If AWS detects an irreparable issue, and there is a need to replace physical equipment, AWS will notify you. You can then place a replacement job that we will ship to your site. There is no additional charge for this, as Snow device monitoring is included as part of the Snow device service fee.

Are you a first-time AWS Snowball user?

If you are a first-time user of the AWS Snow Family service, we recommend that you read the following sections in order:

1. For information about device types and options, see AWS Snowball Edge device differences.
2. To learn more about the types of jobs, see Understanding AWS Snowball Edge Jobs.
3. For an end-to-end overview of how to use an AWS Snowball Edge device, see How AWS Snowball Edge works.
4. When you're ready to get started, see Getting Started.
5. For information about using compute instances on a device, see Using Amazon EC2-compatible compute instances.

AWS Snowball Edge device differences

This guide contains documentation for Snowball Edge devices. You can use these devices to move terabytes of data into and out of Amazon S3. You can order them using the job management API or the AWS Snow Family console. For frequently asked questions and pricing information, see AWS Snowball.

Topics

- Snowball Edge device options
- AWS Snow Family use case differences
- AWS Snow Family Tool Differences

Snowball Edge device options

Snowball Edge devices have the following options for device configurations:

- **Snowball Edge storage-optimized (for data transfer)** – This Snowball Edge device option has 80 TB of usable storage capacity.
- **Snowball Edge storage-optimized 210 TB** – This Snowball Edge device option has 210 TB of usable storage capacity.
- **Snowball Edge storage-optimized (with EC2-compatible compute functionality)** – This Snowball Edge device option has up to 80 TB of usable storage capacity, 40 vCPUs, and 80 GB of memory for compute functionality. It also comes with 1 TB of additional SSD storage capackty for block volumes attached to Amazon EC2-compatible AMIs.
- **Snowball Edge compute-optimized** – This Snowball Edge device (with AMD EPYC Gen2) has the most compute functionality, with up to 104 vCPUs, 416 GB of memory, and 28 TB of dedicated NVMe SSD for compute instances.

  Snowball Edge compute-optimized (with AMD EPYC Gen1) has up to 52 vCPUs, 208 GB of memory, 39.5 TB of usable storage capacity, and 7.68 TB of dedicated NVMe SSD for compute instances.
- **Snowball Edge compute-optimized with GPU** – This Snowball Edge device option is identical to the compute-optimized (with AMD EPYC Gen1) option and includes an installed graphics processing unit (GPU). The GPU is equivalent to the one available in the P3 Amazon EC2-compatible instance type.

> **Note**
>
> When using Amazon S3 compatible storage on Snow Family devices on these devices, usable storage will vary. See Using Amazon S3 compatible storage on Snow Family devices on Snow Family devices for storage capacity with Amazon S3 compatible storage on Snow Family devices.

For more information about the compute functionality of these three options, see Using Amazon EC2-compatible compute instances. Job creation and disk capacity differences in terabytes are described here.

Note

When we refer to *Snowball Edge devices*, this includes all optional variants of the device. When information applies to one or more specific optional configurations (such as how the Snowball Edge compute-optimized with GPU option has an on-board GPU peripheral), we mention this explicitly.

The following table summarizes the differences between the various device options. For hardware specification information, see AWS Snowball Edge Device Specifications.

	Snowball Edge storage-optimized (for data transfer)	**Snowball Edge storage-optimized 210 TB**	**Snowball Edge storage-optimized (with EC2 compute functionality)**	**Snowball Edge compute-optimized with AMD EPYC Gen2 and NVME**	**Snowball Edge compute-optimized with AMD EPYC Gen1, HDD, and optional GPU**
CPU	AMD Naples, 32 cores, 3.4Ghz	AMD Rome, 64 cores, 2 GHz	AMD Naples, 32 cores, 3.4Ghz	AMD Rome, 64 cores, 2 GHz	AMD Naples, 32 cores, 3.4Ghz
vCPUs	40	104	40	104	52
Usable memory	80 GB	416 GB	80 GB	416 GB	208 GB
Security card	Yes	Yes	Yes	Yes	Yes
GPU (optional)	None	None	None	None	NVidia V100
SSD	1 TB SATA	210 TB NVMe	1 TB SATA	28 TB NVMe	7.68 TB NVMe

	Snowball Edge storage-optimized (for data transfer)	**Snowball Edge storage-optimized 210 TB**	**Snowball Edge storage-optimized (with EC2 compute functionality)**	**Snowball Edge compute-optimized with AMD EPYC Gen2 and NVME**	**Snowball Edge compute-optimized with AMD EPYC Gen1, HDD, and optional GPU**
Usable HDD	80 TB	Not applicable	80 TB	Not applicable	39.5 TB usable
Network interfaces	• 2x 10 Gbit – RJ45 (one usable) • 1x 25 Gbit – SFP28 • 1x 100 Gbit – QSFP28	• 2x 10 Gbit – RJ45 (one usable) • 1x 25 Gbit – SFP28 • 1x 100 Gbit – QSFP28	• 2x 10 Gbit – RJ45 (one usable) • 1x 25 Gbit – SFP28 • 1x 100 Gbit – QSFP28	• 2x 10 Gbit – RJ45 (one usable) • 1x 25 Gbit – SFP28 • 1x 100 Gbit – QSFP28	• 2x 10 Gbit – RJ45 (one usable) • 1x 25 Gbit – SFP28 • 1x 100 Gbit – QSFP28

	Snowball Edge storage-optimized (for data transfer)	Snowball Edge storage-optimized 210 TB	Snowball Edge storage-optimized (with EC2 compute functionality)	Snowball Edge compute-optimized with AMD EPYC Gen2 and NVME	Snowball Edge compute-optimized with AMD EPYC Gen1, HDD, and optional GPU
Physical security features	• Hidden magnetic screws • Intrusion switches • NFC tags • Anti-tamper inserts • Android app for tamper detection • GPS and cellular • Conformal coating	• Hidden magnetic screws • Intrusion switches • NFC tags • Anti-tamper inserts • Android app for tamper detection • Conformal coating	• Hidden magnetic screws • Intrusion switches • NFC tags • Anti-tamper inserts • Android app for tamper detection • GPS and cellular • Conformal coating	• Hidden magnetic screws • Intrusion switches • NFC tags • Anti-tamper inserts • Android app for tamper detection • Conformal coating	• Hidden magnetic screws • Intrusion switches • NFC tags • Anti-tamper inserts • Android app for tamper detection • Conformal coating

AWS Snow Family use case differences

The following table shows the use cases for the different AWS Snow Family devices.

Use case	Snowball Edge	AWS Snowcone	
Import data into Amazon S3	✓	✓	
Export from Amazon S3	✓		
Durable local storage	✓		
Local compute with AWS Lambda	✓	✓	
Local compute instances	✓	✓	
Durable Amazon S3 storage in a cluster of devices	✓		
Use with AWS IoT Greengrass (IoT)	✓	✓	
Transfer files through NFS with a GUI	✓	✓	
GPU workloads	✓		

Note

Workloads that need GPU support require the Snowball Edge compute-optimized with GPU option.
Snowball Edge storage-optimized 210TB supports data transfer via NFS, S3 adapter, and Amazon S3 compatible storage on Snow Family devices and does not support the Tape Gateway software.

AWS Snow Family Tool Differences

The following outlines the different tools used with the Snow Family devices, and how they are used.

Snowball Edge Tools

AWS OpsHub for Snow Family

- Snow Family devices now offer a user-friendly tool called AWS OpsHub for Snow Family, which you can use to manage your devices and local AWS services. You can use AWS OpsHub on a client computer to perform tasks such as unlocking and configuring single or clustered devices, transferring files, and launching and managing instances running on Snow Family devices. For more information, see Using AWS OpsHub for Snow Family to Manage Snowball Devices.

Snowball Edge client with Snowball Edge

- Download the Snowball Edge client from the AWS Snowball Edge Resources page and install it on your own computer.
- Use the Snowball Edge client to unlock the Snowball Edge or the cluster of Snowball Edge devices. For more information, see Using the Snowball Edge Client.
- You can't use the Snowball Edge client to transfer data to or from Snow Family devices.

Amazon S3 adapter with Snowball Edge

- Use the Amazon S3 adapter for data transfer to or from AWS.
- Is already installed on the Snowball Edge by default for export or import jobs. It does not need to be downloaded or installed.
- Can transfer data to or from the Snowball Edge. For more information, see Transferring files using the Amazon S3 adapter for data migration.
- Encrypts data on the Snowball Edge while the data is transferred to the device.

Amazon S3 compatible storage on Snow Family devices

- Use Amazon S3 compatible storage on Snow Family devices for edge compute and storage operations.

- The Amazon S3 compatible storage on Snow Family devices service is installed on a Snowball Edge device when chosen during job creation. To configure, start, and use the service, see Amazon S3 compatible storage on Snow Family devices in this guide.

AWS IoT Greengrass console with Snowball Edge

- With a Snowball Edge, you can use the AWS IoT Greengrass console to update your AWS IoT Greengrass group and the core running on the Snowball Edge.

Items Provided for Snowball Edge

The following outlines the network adapters, cables used, and cables provided for the Snowball Edge device.

Network interface	Snowball Edge support	Cables provided with device
RJ45	✓	Not provided.
SFP28	✓	Not provided.
SFP28 (with optic connector)	✓	No cables provided. No optic connector provided for Snowball Edge devices.
QSFP	✓	No cables or optics provided.

For more information about the network interfaces, cables, and connectors, see Supported Network Hardware.

How AWS Snowball Edge works

AWS Snowball Edge devices are owned by AWS, and they reside at your on-premises location while they're in use.

There are four job types you can use with an AWS Snowball Edge device. Although the job types differ in their use cases, every job type has the same workflow for how you order, receive, and return devices. Regardless of the job type, every job follows a data erasure of the National Institute of Standards and Technology (NIST) 800-88 standard after the job completes.

The shared workflow

1. **Create the job** – Each job is created in the AWS Snow Family Management Console or programmatically through the job management API. The status for a job can be tracked in the console or through the API.
2. **A device is prepared for your job** – We prepare an AWS Snowball Edge device for your job, and the status of your job is now **Preparing Snowball**.
3. **A device is shipped to you by your region's carrier** – The carrier takes over from here, and the status of your job is now **In transit to you**. You can find your tracking number and a link to the tracking website on the console or with the job management API. For information about who your region's carrier is, see Shipping considerations for Snow Family devices.
4. **Receive the device** – A few days later, your region's carrier delivers the AWS Snowball Edge device to the address that you provided when you created the job, and the status of your job changes to **Delivered to you**. When it arrives, you'll notice that it didn't arrive in a box, because the device is its own shipping container.
5. **Get your credentials and download the Snowball Edge client** – Get ready to start transferring data by getting your credentials, your job manifest, and the manifest's unlock code, and then downloading the Snowball Edge client.
   - The Snowball Edge client is the tool that you use to manage the flow of data from the device to your on-premises data destination.

     You can download and install the Snowball Edge client from the AWS Snowball resources page.

     You must download the Snowball Edge client from the AWS Snowball Edge Resources page and install on a powerful workstation that you own.

- The manifest is used to authenticate your access to the device, and it is encrypted so that only the unlock code can decrypt it. You can get the manifest from the console or with the job management API when the device is on-premises at your location.
- The unlock code is a 29-character code used to decrypt the manifest. You can get the unlock code from the console or with the job management API. We recommend that you keep the unlock code saved somewhere separate from the manifest to prevent unauthorized access to the device while it's at your facility.

6. **Position the hardware** – Move the device into your data center and open it following the instructions on the case. Connect the device to power and your local network.
7. **Power on the device** – Next, power on the device by pressing the power button above the LCD display. Wait a few minutes, and the **Ready** screen appears.
8. **Get the IP address for the device** – The LCD display has a **CONNECTION** tab on it. Tap this tab and get the IP address for the AWS Snowball Edge device.
9. **Use the Snowball Edge client to unlock the device** – When you use the Snowball Edge client to unlock the AWS Snowball Edge device, enter the IP address of the device, the path to your manifest, and the unlock code. The Snowball Edge client decrypts the manifest and uses it to authenticate your access to the device.
10. **Use the device** – The device is up and running. You can use it to transfer data with the Amazon S3 adapter or the Network File System (NFS) mount point or for local compute and storage with Amazon S3 compatible storage on Snow Family devices.
11. **Prepare the device for its return trip** – After you're done with the device in your on-premises location, press the power button above the LCD display. It takes about 20 seconds or so for the device to power off. Unplug the device and its power cables into the cable nook on top of the device, and shut all three of the device's doors. The device is now ready to be returned.
12. **Your region's carrier returns the device to AWS** – When the carrier has the AWS Snowball Edge device, the status for the job becomes **In transit to AWS**.

> **Note**
>
> There are additional steps for export and cluster jobs. For more information, see How export jobs work and How a clustered local compute and storage job works.

Topics

- How import jobs work

- How export jobs work
- How local compute and storage jobs work
- Snowball Edge videos and blogs

How import jobs work

Each import job uses a single Snowball appliance. After you create a job to order a Snow Family device in the AWS Snow Family Management Console or the job management API, we ship a Snowball to you. When it arrives in a few days, you connect the Snowball Edge device to your network and transfer the data that you want imported into Amazon S3 onto the device. When you're done transferring data, ship the Snowball back to AWS, and we import your data into Amazon S3.

How export jobs work

Each export job can use any number of AWS Snowball Edge devices. If the listing contains more data than can fit on a single device, multiple devices are provided to you. Each job part has exactly one device associated with it. After your job parts are created, your first job part enters the **Preparing Snowball** status.

> **Note**
>
> The listing operation used to split your job into parts is a function of Amazon S3, and you are billed for it the same way as any Amazon S3 operation.

Soon after that, we start exporting your data onto a device. The time required to export your data will vary based on the the nature of your data set. For example, exporting many small files (less than 10 MB) takes significantly longer. When the export is done, AWS gets the device ready for pickup by your region's carrier. When it arrives, you connect the AWS AWS Snowball Edge device to your network and transfer the data from the device to storage on your network.

When you're done transferring data, ship the device back to AWS. When we receive the device for your export job part, we erase it completely. This erasure follows the National Institute of Standards and Technology (NIST) 800-88 standards. This step marks the completion of that particular job part.

- For keylisting

  Before we export the objects in the S3 bucket, we scan the bucket. If the bucket is altered after the scan, the job could encounter delays because we scan for missing or altered objects.

- For S3 Glacier Flexible Retrieval

  It is important to note that AWS Snowball cannot export objects that are in S3 Glacier storage class. These objects must be restored before AWS Snowball can successfully export the objects in the bucket.

How local compute and storage jobs work

You can use the local compute and storage functionality of an AWS Snowball Edge device by running AWS EC2-compatible compute instances or Kubernetes containers in Amazon EKS Anywhere on Snow. For compute functionality, data storage is provided by Amazon S3 compatible storage on Snow Family devices.

You can create Amazon S3 buckets on the Snowball Edge devices to store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. Amazon S3 compatible storage on Snow Family devices provides a new storage class, `SNOW`, which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple Snowball Edge devices. You can use the same APIs and features on Snowball Edge buckets that you do on Amazon S3, including bucket lifecycle policies, encryption, and tagging. When the device or devices are returned to AWS, all data created or stored in Amazon S3 compatible storage on Snow Family devices is erased. For more information, see Local Compute and Storage Only Jobs.

For more information, see Local Compute and Storage Only Jobs.

How a clustered local compute and storage job works

A cluster job is a special kind of job for local storage and compute only. It is for those workloads that require increased data durability and storage capacity. For more information, see Local Cluster Option.

> **Note**
>
> Like standalone local storage and compute jobs, the data stored in a cluster can't be imported into Amazon S3 without ordering additional devices as a part of separate import

jobs. If you order these devices, you can transfer the data from the cluster to the devices and import the data when you return the devices for the import jobs.

Clusters have 3 to 16 AWS Snowball Edge devices, called *nodes*. When you receive the nodes from your regional carrier, connect all the nodes to power and your network to obtain their IP addresses. You use these IP addresses to unlock all the nodes of the cluster at once with a single unlock command, using the IP address of one of the nodes. For more information, see Using the Snowball Edge Client.

You can write data to an unlocked cluster by using or using Amazon S3 compatible storage on Snow Family devices and the data distributed among the other nodes.

When you're done with your cluster, ship all the nodes back to AWS. When we receive the cluster node, we perform a complete erasure of the Snowball. This erasure follows the National Institute of Standards and Technology (NIST) 800-88 standards.

Snowball Edge videos and blogs

- Migrating mixed file sizes with the snow-transfer-tool on AWS Snowball Edge devices
- AWS Snowball Edge Data Migration
- AWS OpsHub for Snow Family
- Novetta delivers IoT and Machine Learning to the edge for disaster response
- Enable large-scale database migrations with DMS and AWS Snowball
- Data Migration Best Practices with AWS Snowball Edge
- AWS Snowball resources
- Amazon S3 Compatible Storage on AWS Snowball Edge Compute Optimized Devices Now Generally Available
- Getting started with Amazon S3 compatible storage on Snow Family devices on AWS Snowball Edge devices

AWS Snowball Edge Device Specifications

In this section, you can find specifications for AWS Snowball Edge device types and the hardware.

Topics

Snowball Edge Storage Optimized (for Data Transfer) specifications

The following table contains hardware specifications for Snowball Edge Storage Optimized devices.

Item	Snowball Edge Storage Optimized (for Data Transfer) specifications
Storage specifications	
HDD storage capacity	80 TB of usable
Power supply specifications	
Power	In AWS Regions in the US: NEMA 5–15p 100–220 volts. In all AWS Regions, a power cable is included
Power consumption	304 watts for an average use case, though the power supply is rated for 1200 watts.
Voltage	100 – 240V AC
Frequency	47/63 Hz

Item	Snowball Edge Storage Optimized (for Data Transfer) specifications
Data and network connections	2x 10 Gbit – RJ45 (one usable) 1x 25 Gbit – SFP28 1x 100 Gbit – QSFP28
Cables	Each AWS Snowball Edge device ships country-specific power cables. No other cables or optics are provided. For more information, see Supported Network Hardware.
Thermal requirements	AWS Snowball Edge devices are designed for office operations, and are ideal for data center operations.
Decibel output	On average, an AWS Snowball Edge device produces 68 decibels of sound, typically quieter than a vacuum cleaner or living-room music.
Dimensions and weight specifications	
Weight	49.7 pounds (22.54 Kg)
Height	15.5 inches (394 mm)
Width	10.6 inches (265 mm)
Length	28.3 inches (718 mm)
Environment specifications	
Vibration	Non-operational use equivalent to ASTM D4169 Truck level I 0.73 GRMS

Item	Snowball Edge Storage Optimized (for Data Transfer) specifications
Shock	Operational use equivalent to 70G (MIL-S-901) Non-operational use equivalent to 50G (ISTA-3A)
Altitude	Operational use equivalent to 0–3,000 meters (0–10,000 feet) Non-operational use equivalent to 0–12,000 meters
Temperature range	0–45°C (operational)

Snowball Edge Storage Optimized 210 TB specifications

The following table contains hardware specifications for Snowball Edge Storage Optimized 210 TB devices.

Item	Snowball Edge Storage Optimized 210 TB specifications
Compute and memory specifications	
CPU	104 vCPUs
RAM	416 GB
Storage specifications	
NVME storage capacity	210 TB usable (for object and NFS data transfer)
SSD storage capacity	None

Item	Snowball Edge Storage Optimized 210 TB specifications
Power supply specifications	
Power	In AWS Regions in the US: NEMA 5–15p 100–220 volts. In all AWS Regions, a power cable is included
Power consumption	304 watts for an average use case, though the power supply is rated for 1200 watts
Voltage	100 – 240V AC
Frequency	47/63 Hz
Data and network connections	2x 10 Gbit – RJ45 (one usable) 1x 25 Gbit – SFP28 1x 100 Gbit – QSFP28
Cables	Each AWS Snowball Edge device ships country-specific power cables. No other cables or optics are provided. For more information, see Supported Network Hardware.
Thermal requirements	AWS Snowball Edge devices are designed for office operations, and are ideal for data center operations.
Decibel output	On average, an AWS Snowball Edge device produces 68 decibels of sound, typically quieter than a vacuum cleaner or living-room music.
Dimensions and weight specifications	
Weight	49.7 pounds (22.54 Kg)
Height	15.5 inches (394 mm)
Width	10.6 inches (265 mm)
Length	28.3 inches (718 mm)

Item	Snowball Edge Storage Optimized 210 TB specifications
Environment specifications	
Vibration	Non-operational use equivalent to ASTM D4169 Truck level I 0.73 GRMS
Shock	Operational use equivalent to 70G (MIL-S-901) Non-operational use equivalent to 50G (ISTA-3A)
Altitude	Operational use equivalent to 0–3,000 meters (0–10,000 feet) Non-operational use equivalent to 0–12,000 meters
Temperature range	0–30°C (operational)

Snowball Edge Storage Optimized (with EC2) specifications

The following table contains hardware specifications for Snowball Edge Storage Optimized (with EC2) devices.

Item	Snowball Edge Storage Optimized (with EC2) specifications
Compute and memory specifications	
CPU	40 vCPUs
RAM	80 GiB
Storage specifications	
HDD storage capacity	80 TB usable (for object and block storage)
SSD storage capacity	1 TB usable SATA SSD storage (for block storage)

Item	Snowball Edge Storage Optimized (with EC2) specifications
Power supply specifications	
Power	In AWS Regions in the US: NEMA 5–15p 100–220 volts. In all AWS Regions, a power cable is included
Power consumption	304 watts for an average use case, though the power supply is rated for 1200 watts
Voltage	100 – 240V AC
Frequency	47/63 Hz
Data and network connections	2x 10 Gbit – RJ45 (one usable) 1x 25 Gbit – SFP28 1x 100 Gbit – QSFP28
Cables	Each AWS Snowball Edge device ships country-specific power cables. No other cables or optics are provided. For more information, see Supported Network Hardware.
Thermal requirements	AWS Snowball Edge devices are designed for office operations, and are ideal for data center operations.
Decibel output	On average, an AWS Snowball Edge device produces 68 decibels of sound, typically quieter than a vacuum cleaner or living-room music.
Dimensions and weight specifications	
Weight	49.7 pounds (22.54 Kg)
Height	15.5 inches (394 mm)
Width	10.6 inches (265 mm)
Length	28.3 inches (718 mm)

Item	Snowball Edge Storage Optimized (with EC2) specifications
Environment specifications	
Vibration	Non-operational use equivalent to ASTM D4169 Truck level I 0.73 GRMS
Shock	Operational use equivalent to 70G (MIL-S-901) Non-operational use equivalent to 50G (ISTA-3A)
Altitude	Operational use equivalent to 0–3,000 meters (0–10,000 feet) Non-operational use equivalent to 0–12,000 meters
Temperature range	0–45°C (operational)

Snowball Edge Compute Optimized device specifications

Item	Snowball Edge Compute Optimized specifications
Compute and memory specifications	
CPU	Up to 104 vCPUs (available in configurations of 52 or 104 vCPUs)
RAM	512 GB RAM (Up to 416 GB RAM - Customer usable)
GPU	nVidia V100 (available in Compute Optimized with GPU configuration - only offered with 52 vCPU)
Storage specifications	
SSD storage capacity	28 TB NVMe SSD or 42 TB HDD (39.5 TB usable)
Power supply specifications	

Item	Snowball Edge Compute Optimized specifications
Power	In AWS Regions in the US: NEMA 5–15p 100–220 volts. In all AWS Regions, a power cable is included
Power consumption	304 watts for an average use case, though the power supply is rated for 1200 watts
Voltage	100 – 240V AC
Frequency	47/63 Hz
Data and network connections	2x 10 Gbit – RJ45 (one usable) 1x 25 Gbit – SFP28 1x 100 Gbit – QSFP28
Cables	Each AWS Snowball Edge device ships country-specific power cables. No other cables or optics are provided. For more information, see Supported Network Hardware.
Thermal requirements	AWS Snowball Edge devices are designed for office operations, and are ideal for data center operations.
Decibel output	On average, an AWS Snowball Edge device produces 68 decibels of sound, typically quieter than a vacuum cleaner or living-room music.
Dimensions and weight specifications	
Weight	49.7 pounds (22.54 Kg)
Height	15.5 inches (394 mm)
Width	10.6 inches (265 mm)
Length	28.3 inches (718 mm)
Environment specifications	

Item	Snowball Edge Compute Optimized specifications
Vibration	Non-operational use equivalent to ASTM D4169 Truck level I 0.73 GRMS
Shock	Operational use equivalent to 70G (MIL-S-901) Non-operational use equivalent to 50G (ISTA-3A)
Altitude	Operational use equivalent to 0–3,000 meters (0–10,000 feet) Non-operational use equivalent to 0–12,000 meters
Temperature range	0–45°C (operational)

Supported Network Hardware

To use the AWS Snowball Edge device, you need your own network cables. For RJ45 cables, there are no specific recommendations. SFP+ and QSFP+ cables and modules from Mellanox and Finisar have been verified to be compatible with the device.

After you open the back panel of the AWS Snowball Edge device, you see the network ports similar to the ports shown in the following screenshot.

Only one network interface on the AWS Snowball Edge device can be used at a time. Hence use any one of the ports to support the following network hardware.

SFP

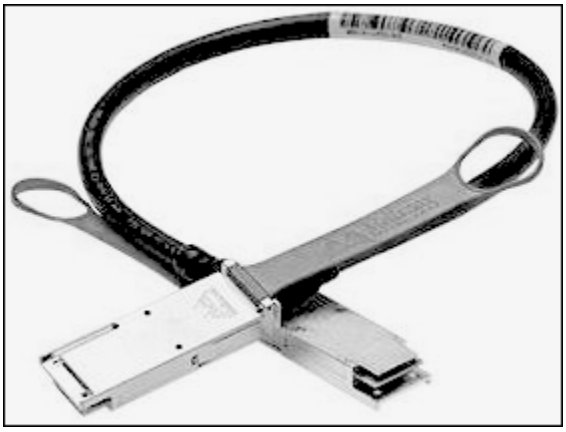
This port provides a 10G/25G SFP28 interface compatible with SFP28 and SFP+ transceiver modules and direct-attach copper (DAC) cables. You need to provide your own transceivers or DAC cables.

- For 10G operation, you can use any SFP+ option. Examples include:
  - 10Gbase-LR (single mode fiber) transceiver
  - 10Gbase-SR (multi-mode fiber) transceiver
  - SFP+ DAC cable
- For 25G operation, you can use any SFP28 option. Examples include:
  - 25Gbase-LR (single mode fiber) transceiver
  - 25Gbase-SR (multi-mode fiber) transceiver
  - SFP28 DAC cable

QSFP

This port provides a 40G QSFP+ interface on storage optimized devices and a 40/50/100G QSFP + interface on compute optimized devices. Both are compatible with QSFP+ transceiver modules and DAC cables. You need to provide your own transceivers or DAC cables. Examples include the following:

- 40Gbase-LR4 (single mode fiber) transceiver
- 40Gbase-SR4 (multi-mode fiber) transceiver
- QSFP+ DAC

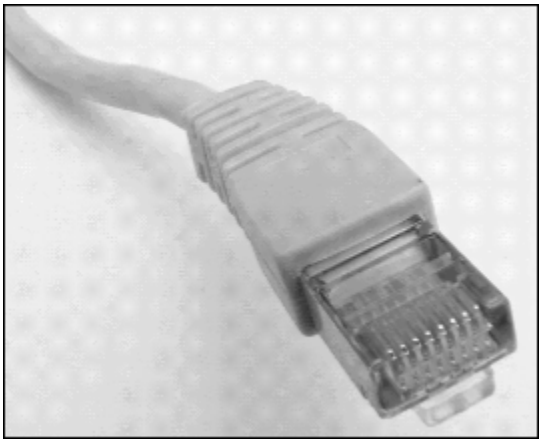


RJ45

This port provides 1Gbase-TX/10Gbase-TX operation. It is connected via UTP cable terminated with a RJ45 connector. Snowball Edge devices have two RJ45 ports. Choose one port to use.

1G operation is indicated by a blinking amber light. 1G operation is not recommended for large-scale data transfers to the Snowball Edge device, as it dramatically increases the time it takes to transfer data.

10G operation is indicated by a blinking green light. It requires a Cat6A UTP cable with a maximum operating distance of 180 feet (55 meters).



Long-term pricing for Snowball Edge devices

When ordering a Snowball Edge device, you can choose the pricing option to best fit your use case. Pricing is available in two ways: on-demand for each day you have the device or prepaid, long-term pricing in monthly, one-, or three-year terms based on the device type. You may choose to automatically renew your long-term pricing option for one- or three-year terms so that a new prepaid period begins when the previous period ends to avoid interruption of your use of the device. The monthly long-term pricing option will automatically renew while the device is in your possession. For more information about ordering a device, see Creating a job to order a Snow Family device in this guide.

In addition to budgetary convenience, long-term pricing allows you to swap Snowball Edge devices during the pricing period when you operational requirements change. For example, you can request to swap devices so that the new device includes a new AMI or new data from Amazon S3 or to replace a failed device. See Swapping devices during the long-term pricing period.

> **Note**
>
> If you request to swap or replace a Snowball Edge device under 1 Year or 3 Year Commit Pricing Plan for any reason other than hardware or a software issue attributed to AWS Snow service, you will be charged a Device Cycling Fee. This Device Cycling Fee is determined as the Monthly fee (for Snowball Edge Compute Optimized) or On-Demand Job Fee for your configuration.

For more information on long-term pricing, see Optimizing cost with long-term pricing options for AWS Snowball. For AWS Snowball pricing for your AWS Region, see AWS Snowball Pricing.

Swapping devices during the long-term pricing period

Swapping Snowball Edge devices during the long-term pricing period involves ordering a new device and immediately returning the current device.

1. Create a new job for the replacement Snowball Edge device. The replacement device must be for the same job type and have the same compute and storage options as the device you have. See Creating a job to order a Snow Family device in this guide.

2. Immediately return the device you have. See Powering off the Snowball Edge and Returning the Snowball Edge Device. AWS will manage the device replacement logistics, and there will be a device cycling fee assessed for this swap.

Setting up your AWS access for AWS Snowball Edge

> **Note**
>
> In the Asia Pacific (Mumbai) AWS Region service is provided by Amazon on Internet Services Private Limited (AISPL). For information on signing up for Amazon Web Services in the Asia Pacific (Mumbai) AWS Region, see Signing Up for AISPL.

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including AWS Snow Family. You are charged only for the services that you use. For more information about pricing and fees, see AWS Snowball Edge Pricing. AWS Snowball Edge is not free to use. For more information on what AWS services are free, see AWS Free Usage Tier.

Note your AWS account number, because you'll need it to create a job to order a Snowball Edge.

Services in AWS, such as AWS Snowball Edge, require that you provide credentials when you access them, so that the service can determine whether you have permission to access its resources. AWS recommends not using the root credentials of your AWS account to make requests. Instead, create an AWS Identity and Access Management (IAM) user, and grant that user full access. We refer to these users as IAM users with administrator-level credentials.

You can use the administrator user credentials, instead of root credentials of your account, to interact with AWS and perform tasks, such as to create an Amazon S3 bucket, create users, and grant them permissions. For more information, see Comparing AWS account root user credentials and IAM user credentials in the *AWS General Reference* and IAM Best Practices in *IAM User Guide*.

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create an administrative user

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create an administrative user

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to an administrative user.

For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the administrative user

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Before you order a Snowball Edge device

AWS Snowball Edge is a region-specific service. So before you plan your job, be sure that the service is available in your AWS Region. Ensure that your location and Amazon S3 bucket are within the same AWS Region or the same country because it will impact your ability to order the device.

To use Amazon S3 compatible storage on Snow Family devices with compute optimized devices for local edge compute and storage jobs, you need to provision S3 capacity on the device or devices when you order. Amazon S3 compatible storage on Snow Family devices supports local bucket management, so you can create S3 buckets on the device or cluster after you receive the device or devices.

As part of the order process, you create an AWS Identity and Access Management (IAM) role and an AWS Key Management Service (AWS KMS) key. The KMS key is used to encrypt the unlock code for your job. For more information about creating IAM roles and KMS keys, see Creating a job to order a Snow Family device.

Topics

Questions about the local environment

Understanding your dataset and how the local environment is set up will help you complete your data transfer. Consider the following before placing your order.

What data are you transferring?

Transferring a large number of small files does not work well with AWS Snowball Edge. This is because Snowball Edge encrypts each individual object. Small files include files under 1 MB in size. We recommend that you zip them up before transferring them onto the AWS Snowball Edge device. We also recommend that you have no more than 500,000 files or directories within each directory.

Will the data be accessed during the transfer?

It is important to have a static dataset, (that is, no users or systems are accessing the data during transfer). If not, the file transfer can fail due to a checksum mismatch. The files won't be transferred and the files will be marked as `Failed`.

To prevent corrupting your data, don't disconnect an AWS Snowball Edge device or change its network settings while transferring data. Files should be in a static state while being written to the device. Files that are modified while they are being written to the device can result in read/write conflicts.

Will the network support AWS Snowball data transfer?

Snowball Edge supports the *RJ45*, *SFP+*, or *QSFP+* networking adapters. Verify that your switch is a gigabit switch. Depending on the brand of switch, it might say **gigabit** or **10/100/1000**. Snowball Edge devices do not support a megabit switch, or 10/100 switch.

Working with filenames that contain special characters

It's important to note that if the names of your objects contain special characters, you might encounter errors. Although Amazon S3 allows special characters, we highly recommend that you avoid the following characters:

- Backslash ("\")
- Left curly brace ("{")
- Right curly brace ("}")
- Left square bracket ("[")
- Right square bracket ("]")
- 'Less Than' symbol ("<")
- 'Greater Than' symbol (">")
- Non-printable ASCII characters (128–255 decimal characters)
- Caret ("^")
- Percent character ("%")
- Grave accent / back tick ("`")
- Quotation marks

- Tilde ("~")
- 'Pound' character ("#")
- Vertical bar / pipe ("|")

If your files have one or more of these characters in object names, rename the objects before you copy them to the AWS Snowball Edge device. Windows users who have spaces in their file names should be careful when copying individual objects or running a recursive command. In commands, surround the names of objects that include spaces in the names with quotation marks. The following are examples of such files.

Operating system	File name: test file.txt
Windows	`“C:\Users\<username>\desktop\test file.txt”`
iOS	`/Users/<username>/test\ file.txt`
Linux	`/home/<username>/test\ file.txt`

Note

The only object metadata that is transferred is the object name and size.

Using Amazon EC2 on Snow Family devices

This section provides an overview of using Amazon EC2-compatible compute instances on an AWS Snowball Edge device. It includes conceptual information, procedures, and examples.

Note

These Amazon EC2 features on AWS Snowball are not supported in the Asia Pacific (Mumbai) and Europe (Paris) AWS Regions.

You can run Amazon EC2-compatible compute instances hosted on an AWS Snowball Edge with the `sbe1`, `sbe-c`, and `sbe-g` instance types:

- The `sbe1` instance type works on devices with the Snowball Edge Storage Optimized option.
- The `sbe-c` instance type works on devices with the Snowball Edge Compute Optimized option.
- Both the `sbe-c` and `sbe-g` instance types work on devices with the Snowball Edge Compute Optimized with GPU option.

All the compute instance types supported on Snowball Edge device options are unique to AWS Snowball Edge devices. Like their cloud-based counterparts, these instances require Amazon Machine Images (AMIs) to launch. You choose the AMI for an instance before you create your Snowball Edge job.

To use a compute instance on a Snowball Edge, create a job to order a Snow Family device and specify your AMIs. You can do this using the AWS Snowball Management Console, the AWS Command Line Interface (AWS CLI), or one of the AWS SDKs. Typically, to use your instances, there are some housekeeping prerequisites that you must perform before creating your job.

After your device arrives, you can start managing your AMIs and instances. You can manage your compute instances on a Snowball Edge through an Amazon EC2 endpoint. This type of endpoint supports many of the Amazon EC2 CLI commands and actions for the AWS SDKs. You can't use the AWS Management Console on the Snowball Edge to manage your AMIs and compute instances.

When you're done with your device, return it to AWS. If the device was used in an import job, the data transferred using the Amazon S3 adapter or the NFS interface is imported into Amazon S3. Otherwise, we perform a complete erasure of the device when it is returned to AWS. This erasure follows the National Institute of Standards and Technology (NIST) 800-88 standards.

> **Important**
>
> Data in compute instances running on a Snowball Edge isn't imported into AWS.

Difference between Amazon EC2 and Amazon EC2-compatible instances on Snow Family devices

AWS Snow Family EC2-compatible instances allow customers to use and manage Amazon EC2-compatible instances using a subset of EC2 APIs and a subset of AMIs.

Pricing for Compute Instances on Snowball Edge

There are additional costs associated with using compute instances. For more information, see AWS Snowball Edge Pricing.

Prerequisites

Before creating your job, keep the following information in mind:

- Before you add any AMIs to your job request, make sure that you have created an AMI that is supported in your AWS account. Currently, supported AMIs are based on the CentOS 7 (x86_64) - with Updates HVM and Ubuntu 16.04 LTS - Xenial (HVM) images. You can get these images from the AWS Marketplace website.
- All AMIs must be based on Amazon Elastic Block Store (Amazon EBS), with a single volume.
- If you are connecting to a compute instance running on a Snowball Edge, you must use Secure Shell (SSH). To do so, you first add the key pair. For more information, see Configuring an AMI to Use SSH to Connect to Compute Instances Launched on the Device.

Creating a Linux AMI from an Instance

You can create an AMI using the AWS Management Console or the command line. Start with an existing AMI, launch an instance, customize it, create a new AMI from it, and finally, launch an instance of your new AMI.

To create an AMI from an instance using the console

1. Select an appropriate EBS-backed AMI as a starting point for your new AMI, and configure it as needed before launch. For more information, see Launching an instance using the Launch Instance Wizard in the *Amazon EC2 User Guide for Linux Instances*.
2. Choose **Launch** to launch an instance of the EBS-backed AMI that you selected. Accept the default values as you step through the wizard. For more information, see Launching an instance using the Launch Instance Wizard.
3. While the instance is running, connect to it. You can perform the following actions on your instance to customize it for your needs:
   - Install software and applications.
   - Copy data.

- Reduce start time by deleting temporary files, defragmenting your hard drive, and zeroing out free space.
- Attach additional Amazon EBS volumes.

4. (Optional) Create snapshots of all the volumes attached to your instance. For more information about creating snapshots, see Creating Amazon EBS snapshots in the *Amazon EC2 User Guide for Linux Instances*.
5. In the navigation pane, choose **Instances**, and choose your instance. Choose **Actions**, choose **Image**, and then choose **Create image**.

> **Tip**
>
> If this option isn't available, your instance isn't an Amazon EBS-backed instance.

6. In the **Create Image** dialog box, specify the following information, and then choose **Create image**.

- **Image name** - A unique name for the image.
- **Image description** - An optional description of the image, up to 255 characters.
- **No reboot** - This option is not selected by default. Amazon EC2 shuts down the instance, takes snapshots of any attached volumes, creates and registers the AMI, and then reboots the instance. Select **No reboot** to avoid having your instance shut down.

> **Warning**
>
> If you select **No reboot**, we can't guarantee the file system integrity of the created image.

- **Instance Volumes** - The fields in this section enable you to modify the root volume, and add more Amazon EBS and instance store volumes. For information about each field, pause on the **i** icon next to each field to display field tooltips. Some important points are listed following:
  - To change the size of the root volume, locate **Root** in the **Volume Type** column. For **Size (GiB)**, enter the required value.
  - If you select **Delete on Termination**, when you terminate the instance created from this AMI, the Amazon EBS volume is deleted. If you clear **Delete on Termination**, when you terminate the instance, the Amazon EBS volume is not deleted. For more information, see

Preserving Amazon EBS volumes on instance termination in the *Amazon EC2 User Guide for Linux Instances*.

- To add an Amazon EBS volume, choose **Add New Volume** (which adds a new row). For **Volume Type**, choose **EBS**, and fill in the fields in the row. When you launch an instance from your new AMI, additional volumes are automatically attached to the instance. Empty volumes must be formatted and mounted. Volumes based on a snapshot must be mounted.
- To add an instance store volume, see Adding instance store volumes to an AMI in the *Amazon EC2 User Guide for Linux Instances*. When you launch an instance from your new AMI, additional volumes are automatically initialized and mounted. These volumes don't contain data from the instance store volumes of the running instance on which you based your AMI.

7. To view the status of your AMI while it is being created, in the navigation pane, choose **AMIs**. Initially, the status is pending but should change to available after a few minutes.

   (Optional) To view the snapshot that was created for the new AMI, choose **Snapshots**. When you launch an instance from this AMI, we use this snapshot to create its root device volume.
8. Launch an instance from your new AMI. For more information, see Launching an instance using the Launch Instance Wizard in the *Amazon EC2 User Guide for Linux Instances*.
9. The new running instance contains all of the customizations that you applied in previous steps.

To Create an AMI from an Instance Using the Command Line

You can use one of the following commands. For more information about these command line interfaces, see Accessing Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances*.

- create-image (AWS CLI)
- New-EC2Image (AWS Tools for Windows PowerShell)

Creating a Linux AMI from a Snapshot

If you have a snapshot of the root device volume of an instance, you can create an AMI from this snapshot using the AWS Management Console or the command line.

To create an AMI from a snapshot using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Elastic Block Store**, choose **Snapshots**.
3. Choose the snapshot, choose **Actions**, and then choose **Create image**.
4. In the **Create image from EBS snapshot** dialog box, complete the fields to create your AMI. Then choose **Create**. If you're re-creating a parent instance, choose the same options as the parent instance.

   - **Architecture** – Choose **i386** for 32-bit or **x86_64** for 64-bit.
   - **Root device name** – Enter the appropriate name for the root volume. For more information, see Device naming on Linux instances in the *Amazon EC2 User Guide for Linux Instances*.
   - **Virtualization type** – Choose whether instances launched from this AMI use paravirtual (PV) or hardware virtual machine (HVM) virtualization. For more information, see Linux AMI virtualization types.
   - (PV virtualization type only) **Kernel ID** and **RAM disk ID** – Choose the AKI and ARI from the lists. If you choose the default AKI, or you don't choose an AKI, you must specify an AKI every time you launch an instance using this AMI. In addition, your instance might fail the health checks if the default AKI is incompatible with the instance.
   - (Optional) **Block Device Mappings** – Add volumes or expand the default size of the root volume for the AMI. For more information about resizing the file system on your instance for a larger volume, see Extending a Linux File system after resizing a volume in the *Amazon EC2 User Guide for Linux Instances*.

To Create an AMI from a Snapshot Using the Command Line

To create an AMI from a snapshot, you can use one of the following commands. For more information about these command line interfaces, see Accessing Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances*.

- register-image (AWS CLI)
- Register-EC2Image (AWS Tools for Windows PowerShell)

Using Amazon S3 on Snowball Edge

As part of the order process, you are asked to create an AWS Identity and Access Management (IAM) role and AWS Key Management Service (AWS KMS) key. The KMS key is used for encrypting the data at rest on the Snowball Edge device. For more information about creating IAM roles and KMS keys, see Creating a job to order a Snow Family device.

Important

If the imported data must be encrypted in the S3 bucket using Server-Side Encryption with keys stored in AWS KMS (SSE-KMS), see Amazon S3 encryption with AWS KMS.
If the imported data must be encrypted in the S3 bucket using Server-Side Encryption with Amazon S3 managed keys (SSE-S3), see Amazon S3 encryption with server-side encryption.

How import works

Each import job uses a single Snowball Edge device. After you create a job to order a Snow Family device, we ship a Snowball Edge device to you. When it arrives, you connect the Snowball Edge device to your network and transfer the data that you want to import to Amazon S3 onto that Snowball Edge. When you're done transferring data, ship the Snowball Edge back to AWS. We then import your data into Amazon S3.

Important

Snowball Edge cannot write to buckets if you have turned on S3 Object Lock. We also cannot write to your bucket if IAM policies on the bucket prevent writing to the bucket.

How export works

Each export job can use any number of AWS Snowball Edge devices. After you create a job, a listing operation starts in Amazon S3. This listing operation splits your job into parts. Each job part has exactly one device associated with it. After your job parts are created, your first job part enters the **Preparing** Snowball status.

> **Note**
>
> The listing operation to split your job into parts is a function of Amazon S3, and you are billed the same as Amazon S3 operation.

We then start exporting your data onto a device. Typically, exporting data takes one business day. However, this process can take longer. When the export is done, AWS gets the device ready for your regional carrier to pick up.

When the device arrives at your site, you connect it to your network and transfer the data that you want to import into Amazon S3 onto the device. When you're done transferring the data, ship the device back to AWS. When we receive the returned device, we erase it completely. This erasure follows the National Institute of Standards and Technology (NIST) 800-88 standards.

This step marks the completion of that particular job part. If there are more job parts, the next job part now is prepared for shipping.

> **Important**
>
> Snowball Edge is unable to export files that are in S3 Glacier storage class. These objects must be restored before we can export the files. If we encounter files in S3 Glacier storage class, we contact you to let you know, but this might add delays to your export job.

Using Amazon S3 compatible storage on Snow Family devices for edge compute and storage jobs

Amazon S3 compatible storage on Snow Family devices delivers secure object storage with increased resiliency, scale, and expanded Amazon S3 API feature-set to the rugged, mobile edge, and disconnected environments. Amazon S3 compatible storage on Snow Family devices enables customers to store data and run highly available applications on Snow Family devices for edge compute use case.

You can create Amazon S3 buckets on the Snowball Edge devices to store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. Amazon S3 compatible storage on Snow Family devices provides a new storage class, `SNOW`, which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple

Snowball Edge devices. You can use the same APIs and features on Snowball Edge buckets that you do on Amazon S3, including bucket lifecycle policies, encryption, and tagging. When the device or devices are returned to AWS, all data created or stored in Amazon S3 compatible storage on Snow Family devices is erased. For more information, see Local Compute and Storage Only Jobs.

Amazon S3 compatible storage on Snow Family devices can be deployed in standalone configuration or cluster configuration. In standalone configuration you can provision usable S3 capacity on device and the balance will be available as block storage. In cluster setup all data disk capacity will be utilzied for S3 storage. Depending on the size of cluster, S3 service is designed to sustain device fault tolerance of 1 or 2 devices. For more information about cluster fault tolerance, see Clustering overview.

To set up and use Amazon S3 compatible storage on Snow Family devices see Amazon S3 compatible storage on Snow Family devices in this guide.

Amazon S3 encryption with AWS KMS

You can use the default AWS managed or customer managed encryption keys to protect your data when importing or exporting data.

Using Amazon S3 default bucket encryption with AWS KMS managed keys

To enable AWS managed encryption with AWS KMS

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the Amazon S3 bucket that you want to encrypt.
3. In the wizard that appears on the right side, choose **Properties**.
4. In the **Default encryption** box, choose **Disabled** (this option is grayed out) to enable default encryption.
5. Choose **AWS-KMS** as the encryption method, and then choose the KMS key that you want to use. This key is used to encrypt objects that are PUT into the bucket.
6. Choose **Save**.

After the Snowball Edge job is created, and before the data is imported, add a statement to the existing IAM role policy. This is the role you created during the ordering process. Depending on the job type, the default role name looks similar to `Snowball-import-s3-only-role` or `Snowball-export-s3-only-role`.

The following are examples of such a statement.

For importing data

If you use server-side encryption with AWS KMS managed keys (SSE-KMS) to encrypt the Amazon S3 buckets associated with your import job, you also need to add the following statement to your IAM role.

Example Example Snowball import IAM role

```
{
    "Effect": "Allow",
    "Action": [
        "kms: GenerateDataKey",
 "kms: Decrypt"
    ],
    "Resource":"arn:aws:kms:us-west-2:123456789012:key/abc123a1-abcd-1234-
efgh-111111111111"
}
```

For exporting data

If you use server-side encryption with AWS KMS managed keys to encrypt the Amazon S3 buckets associated with your export job, you also must add the following statement to your IAM role.

Example Snowball export IAM role

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt"
    ],
    "Resource":"arn:aws:kms:us-west-2:123456789012:key/abc123a1-abcd-1234-
efgh-111111111111"
}
```

Using S3 default bucket encryption with AWS KMS customer keys

You can use the default Amazon S3 bucket encryption with your own KMS keys to protect data you are importing and exporting.

For importing data

To enable customer managed encryption with AWS KMS

1. Sign in to the AWS Management Console and open the AWS Key Management Service (AWS KMS) console at https://console.aws.amazon.com/kms.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. In the left navigation pane, choose **Customer managed keys**, and then choose the KMS key associated with the buckets that you want to use.
4. Expand **Key Policy** if it is not already expanded.
5. In the **Key Users** section, choose **Add** and search for the IAM role. Choose the IAM role, and then choose **Add**.
6. Alternatively, you can choose **Switch to Policy view** to display the key policy document and add a statement to the key policy. The following is an example of the policy.

Example of a policy for the AWS KMS customer managed key

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::111122223333:role/snowball-import-s3-only-role"
    ]
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*"
}
```

After this policy has been added to the AWS KMS customer managed key, it is also needed to update the IAM role associated with the Snowball job. By default, the role is `snowball-import-s3-only-role`.

Example of the Snowball import IAM role

```
{
  "Effect": "Allow",
  "Action": [
```

```
    "kms: GenerateDataKey",
    "kms: Decrypt"
  ],
  "Resource": "arn:aws:kms:us-west-2:123456789012:key/abc123a1-abcd-1234-
efgh-111111111111"
}
```

For more information, see Using Identity-Based Policies (IAM Policies) for AWS Snowball.

The KMS key that is being used looks like the following:

“Resource”:“arn:aws:kms:*region*:*AccoundID*:key/*”

For exporting data

Example of a policy for the AWS KMS customer managed key

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::111122223333:role/snowball-import-s3-only-role"
    ]
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*"
}
```

After this policy has been added to the AWS KMS customer managed key, it is also needed to update the IAM role associated with the Snowball job. By default, the role looks like the following:

`snowball-export-s3-only-role`

Example of the Snowball export IAM role

```
{
  "Effect": "Allow",
  "Action": [
```

```
    "kms: GenerateDataKey",
    "kms: Decrypt"
  ],
  "Resource": "arn:aws:kms:us-west-2:123456789012:key/abc123a1-abcd-1234-
efgh-111111111111"
}
```

After this policy has been added to the AWS KMS customer managed key, it is also needed to update the IAM role associated with the Snowball job. By default, the role is `snowball-export-s3-only-role`.

Amazon S3 encryption with server-side encryption

AWS Snowball supports server-side encryption with Amazon S3 managed encryption keys (SSE-S3). Server-side encryption is about protecting data at rest, and SSE-S3 has strong, multifactor encryption to protect your data at rest in Amazon S3. For more information about SSE-S3, see Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3) in the *Amazon Simple Storage Service User Guide*.

Note

Currently, AWS Snowball doesn't support server-side encryption with customer-provided keys (SSE-C). However, you might want to use that SSE type to protect data that has been imported, or you might already use it on data you want to export. In these cases, keep the following in mind:

- **Import** – If you want to use SSE-C to encrypt the objects that you've imported into S3, copy those objects into another bucket that has SSE-KMS or SSE-S3 encryption established as a part of that bucket's bucket policy.
- **Export** – If you want to export objects that are encrypted with SSE-C, first copy those objects to another bucket that either has no server-side encryption, or has SSE-KMS or SSE-S3 specified in that bucket's bucket policy.

Snowball Edge Clusters

For the AWS Snowball service, a cluster is a collective of Snowball Edge devices, used as a single logical unit, for local storage and compute purposes.

A *cluster* is a logical grouping of AWS Snowball Edge devices, in groups of 3 to 16 devices. A cluster is created with a single job. A cluster offers increased durability and storage capacity. This section provides information about Snowball Edge clusters with Amazon S3 compatible storage on Snow Family devices.

Considerations for Cluster Jobs for AWS Snowball Edge

Keep the following considerations in mind when planning to use a cluster of Snowball Edge devices:

- We recommend that you have a redundant power supply to reduce potential performance and stability issues for your cluster.
- Like standalone local storage and compute jobs, the data stored in a cluster can't be imported into Amazon S3 without ordering additional devices as a part of separate import jobs. If you order these devices, you can transfer the data from the cluster to the devices and import the data when you return the devices for the import jobs.
- To get data onto a cluster from Amazon S3, create a separate export job and copy the data from the devices of the export job onto the cluster.
- You can use the console, the AWS CLI, or AWS SDK to create a cluster job.
- Cluster nodes have node IDs. A *node ID* is the same as the job ID for a device that you can get from the console, the AWS CLI, the AWS SDKs, or the Snowball Edge client. You can use node IDs to remove old nodes from clusters. You can get a list of node IDs by using the `snowballEdge describe-device` command on an unlocked device or the `describe-cluster` on an unlocked cluster.
- The lifespan of a cluster is limited by the security certificate granted to the cluster devices when the cluster is provisioned.
- When AWS receives a returned device that was part of a cluster, we perform a complete erasure of the device. This erasure follows the National Institute of Standards and Technology (NIST) 800-88 standards.

Shipping considerations for Snow Family devices

When you create a job to order a Snow Family device, you provide a shipping address and choose shipping speed. Note that the shipping speed doesn't indicate how soon you can expect to receive the device from the day you created the job. Rather, it indicates the time that the device is in transit between AWS and your shipping address. Before the device ships, AWS processes the device for the job. The amount of time that's required to process your job depends on factors like job type and size. Also, shipping carriers generally only pick up outgoing Snow Family devices once a day and carriers don't pick up outgoing devices on weekends. Thus, processing before shipping can take a day or more. While AWS is preparing your device to ship and after it receives the device after you return it, you can monitor the status of your job through the AWS Snow Family Management Console. For more information, see Job Statuses.

> **Note**
>
> The shipping speed that you choose applies when AWS sends the device to you and when you return the device to AWS.
> Snowball Edge devices can only be used to import or export data within the AWS Region where the devices are ordered.

For more information on choosing shipping speed and entering your shipping address when creating a job to order a Snow Family device, see Step 4: Choose security, shipping, and notification preferences. For more information about returning a Snow Family device to AWS, see Returning the Snowball Edge Device.

For information about shipping charges, see AWS Snowball Edge Pricing.

Region-based shipping restrictions

Before you create a job to order a Snow Family device, you should sign in to the console from the same AWS Region as your Amazon S3 data. AWS does not ship Snow Family devices between countries within the same AWS Region—for example, from Asia Pacific (India) to Asia Pacific (Australia).

An exception to shipping between countries is among European Union (EU) member countries. For data transfers in the European AWS Regions, we only ship devices to the EU member countries listed:

Austria, Belgium, Bulgaria, Croatia, Republic of Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Italy, Ireland, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Poland, Portugal, Romania, Slovakia, Slovenia, Spain and Sweden.

Snow Family devices can only be returned to the same AWS Region where the devices were ordered.

Shipments domestically within the same country are permitted. Examples:

- For data transfers in the United Kingdom Region, we ship devices domestically within the UK.
- For data transfers in Asia Pacific (Mumbai), we ship devices within India.

> **Note**
>
> AWS doesn't ship Snow Family devices to post office boxes.

Getting Started

With an AWS Snowball Edge device, you can access the storage and compute power of the AWS Cloud locally and cost effectively in places where connecting to the internet might not be an option. You can also transfer hundreds of terabytes or petabytes of data between your on-premises data centers and Amazon Simple Storage Service (Amazon S3).

Following, you can find general instructions for creating and completing your first AWS Snowball Edge device job in the AWS Snow Family Management Console. The console presents the most common workflows, separated into job types. You can find more information about specific components of the AWS Snowball Edge device in this documentation. For an overview of the service as a whole, see How AWS Snowball Edge works.

The getting started exercises assume that you use the AWS Snow Family Management Console to create your job, the AWS OpsHub for Snow Family to unlock and manage the AWS Snowball Edge device, and the Amazon S3 interface to read and write data. If you'd rather create your job programmatically with more options for the jobs you're creating, you can use the job management API. For more information, see AWS Snowball API Reference.

Before you can get started, you must create an AWS account and an administrator user in AWS Identity and Access Management (IAM). For information, see Setting up your AWS access for AWS Snowball Edge.

Topics

- Return shipping for Snow Family devices
- Monitoring the Import Status
- Getting your job completion report and logs on the console

Creating a job to order a Snow Family device

To order a Snow Family device, you create a job to order a Snow Family device in the AWS Snow Family Management Console. A *job* is a term that AWS uses to describe the lifecycle of the use of a Snow Family device by a customer. A job begins when you order a device, continues when AWS prepares the device and ships it to you and you use it, and completes after AWS receives and processes the device after you return it. Jobs are categorized by type: export, import, local compute and storage, and virtual tape transfer. For more information, see Understanding AWS Snowball Edge jobs.

After you create the job to order a device, you can use the AWS Snow Family Management Console to view the job status and monitor the progress of the device you ordered as AWS prepares the device to ship to you and after it is returned. For more information, see Job Statuses. After the device is returned and processed by AWS, you can access a job completion report and logs through the AWS Snow Family Management Console. For more information, see Getting your job completion report and logs on the console.

You can also create and manage jobs using the job management API. For more information, see the AWS Snowball API Reference.

Topics

- Step 1: Choose a job type
- Step 2: Choose your compute and storage options
- Step 3: Choose your features and options
- Step 4: Choose security, shipping, and notification preferences
- Step 5: Review job summary and create your job
- Download AWS OpsHub

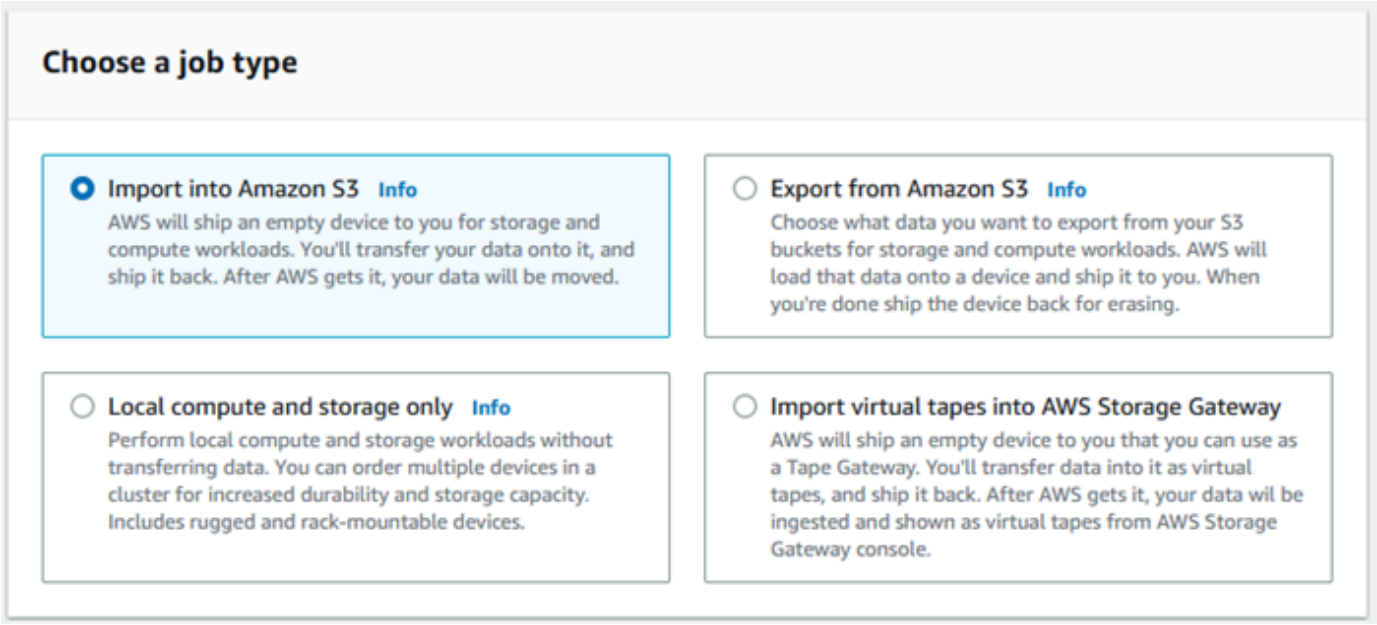
Step 1: Choose a job type

The first step in creating a job is to determine the type of job that you need and to start planning it using the AWS Snow Family Management Console.

To choose your job type

1. Sign in to the AWS Management Console, and open the AWS Snow Family Management Console. If this is your first time creating a job in this AWS Region, you will see the **AWS Snow Family** page. Otherwise you will see the list of existing jobs.
2. If this is your first job, choose **Order an AWS Snow Family device**. If you're expecting multiple jobs to migrate over 500 TB of data, choose **Create your large data migration plan greater than 500 TB**. Otherwise, choose **Create Job** in the left navigation bar. Choose **Next step** to open the **Plan your job** page.
3. In the **Job name** section, provide a name for your job in the **Job name** box.
4. Depending on your need, choose one of the following job types:

   - **Import into Amazon S3** – Choose this option to have AWS ship an empty Snowball Edge device to you. You connect the device to your local network and run the Snowball Edge client. You copy data onto the device using NFS share or the S3 adapter, ship it back to AWS, and your data is uploaded to AWS.
   - **Export from Amazon S3** – Choose this option to export data from your Amazon S3 bucket to your device. AWS loads your data on the device and ships it to you. You connect the device to your local network and run the Snowball Edge client. You copy data from your device to your servers. When you are done, ship the device to AWS, and your data is erased from the device.
   - **Local compute and storage only** – Perform compute and storage workloads on the device without transferring data.
   - **Import virtual tapes into AWS Storage Gateway** – AWS ships an empty Snow Family device configured as a Tape Gateway. You transfer the data to the device in the form of virtual tapes, and ship it back. AWS ingests the data and displays it as virtual tapes in the AWS Storage Gateway console. Your data is then erased from the device.



5. Choose **Next** to continue.

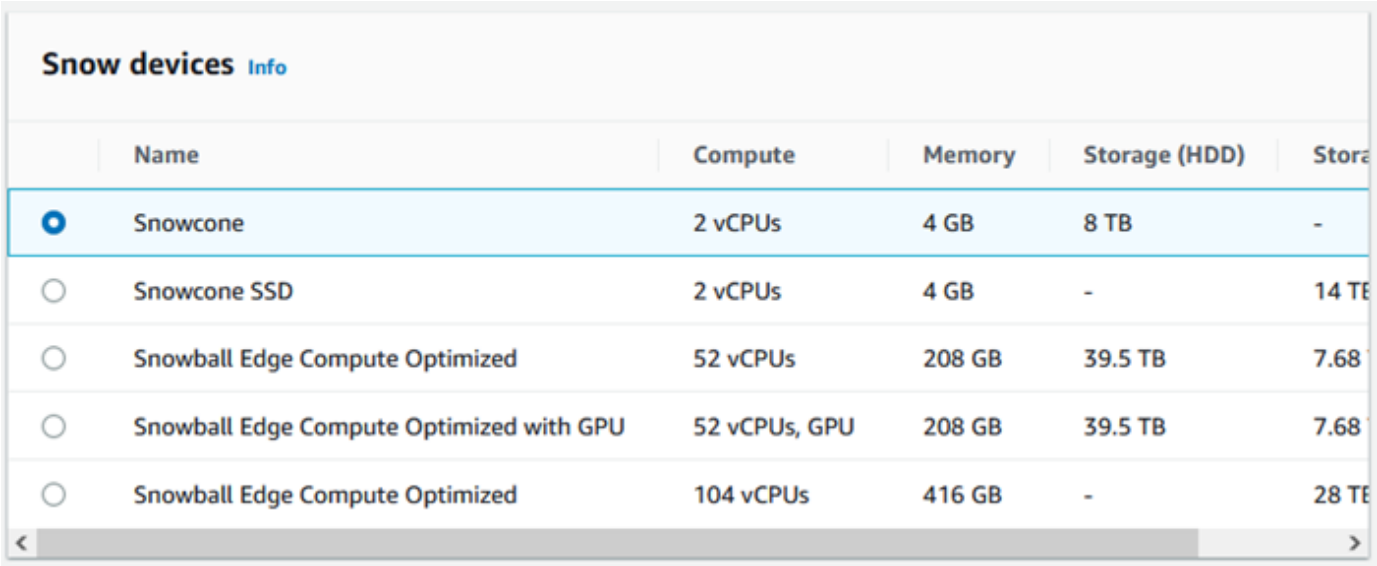
Step 2: Choose your compute and storage options

Choose the hardware specifications for your Snow Family device, which of your Amazon EC2-compatible instances to include on it, how data will be stored, and pricing.

To choose your device's compute and storage options

1. In the **Snow devices** section, choose the Snow Family device to order.

> **Note**
>
> Some Snow Family devices might not be available depending on the AWS Region you are ordering from and the job type you chose.



2. In the **Choose your pricing option** section, from the **Choose your pricing option** menu, choose the type of pricing to apply to this job. If you choose 1 year or 3 year commit upfront pricing, in **Auto-renew**, choose **On** to automatically renew the pricing when the current period ends or **Off** to not automatically renew the pricing when the current period ends. For more information about long-term pricing options for Snowball Edge devices, see Long-term pricing for Snowball Edge devices in this guide. For device pricing for your AWS Region, see AWS Snowball Pricing.
3. In the **Select the storage type** section, make a choice according to your need:

   - **S3 Adapter**: Use the S3 adapter to transfer data programmatically to and from Snow Family devices using Amazon S3 REST API actions.
   - **Amazon S3 compatible storage**: Use Amazon S3 compatible storage to deploy S3 compatible durable, scalable object storage on single Snowball Edge device or in a multi-device cluster.
   - **NFS based data transfer**: Use Network File System (NFS) based data transfer to drag and drop files from your computer into Amazon S3 buckets on Snow Family devices.

   > **Warning**
   >
   > NFS based data transfer doesn't support the S3 adapter. If you proceed with NFS based data transfer, you must mount the NFS share to transfer objects. Using the AWS CLI to transfer objects will fail.

See Using NFS for Offline Data Transfer in the AWS Snowball Edge Developer Guide for more information.

Note

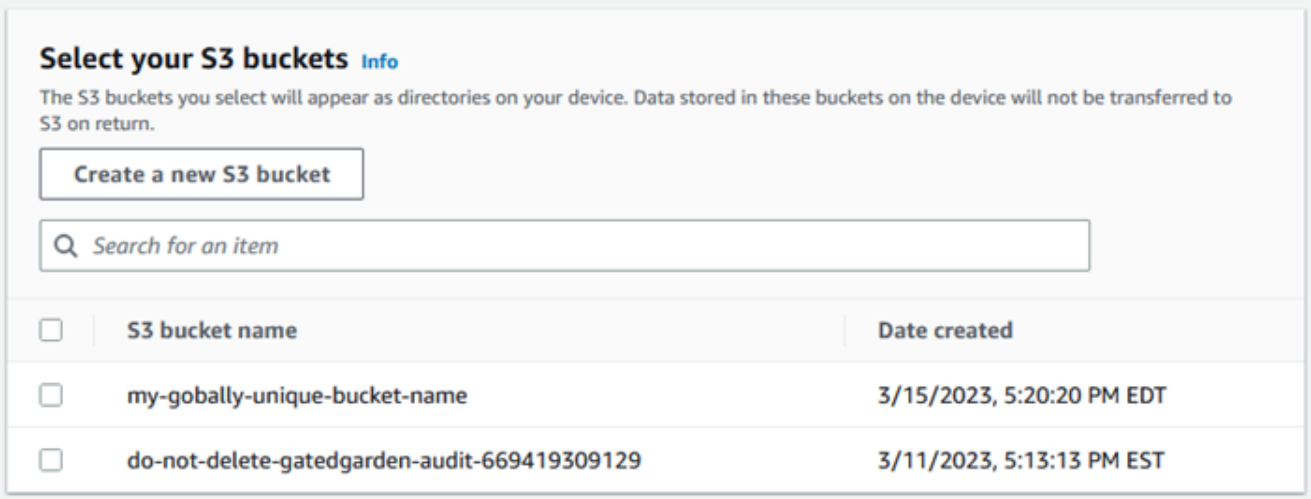
Storage type options available depend on the job type and Snow device you chose.

4. If you selected *S3 Adapter* as the storage type or if you selected a device that supports block storage, do the following to select one or more S3 buckets to include on the device:

   - In the **Select your S3 buckets** section, do one or more of the following to select one or more S3 buckets:

     1. Choose the S3 bucket that you want to use in the **S3 bucket name** list.
     2. In the **Search for an item** field, enter all or part a bucket name to filter the list of available buckets on your entry, then choose the bucket.
     3. Choose the **Create a new S3 bucket** to create a new S3 bucket. The new bucket name appears in the **Bucket name** list. Choose it.

     You can include one or more S3 buckets. These buckets appear on your device as local S3 buckets.

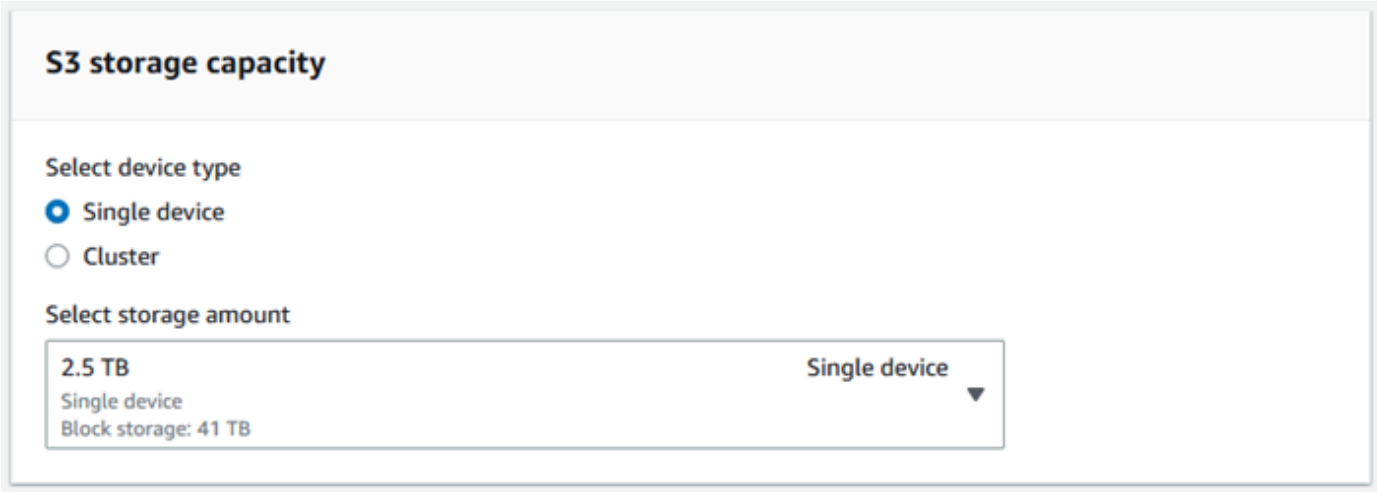


5. If you selected *Amazon S3 compatible storage* as the storage type, in the **S3 storage capacity** section, do the following:

a. Select to use Amazon S3 compatible storage on Snow Family devices on a single device or a cluster of devices. See Using an AWS Snowball Edge cluster in this guide.

b. Select the amount of device storage to use for Amazon S3 compatible storage on Snow Family devices.

> **Note**
>
> When using Amazon S3 compatible storage on Snow Family devices, you can manage and create Amazon S3 buckets after you receive the device, so you don't need to choose them while ordering. See Amazon S3 compatible storage on Snow Family devices in this guide.



6. If you selected *NFS based data transfer* as the storage type, in the **Select your S3 buckets** section, do one or more of the following to select one or more S3 buckets:

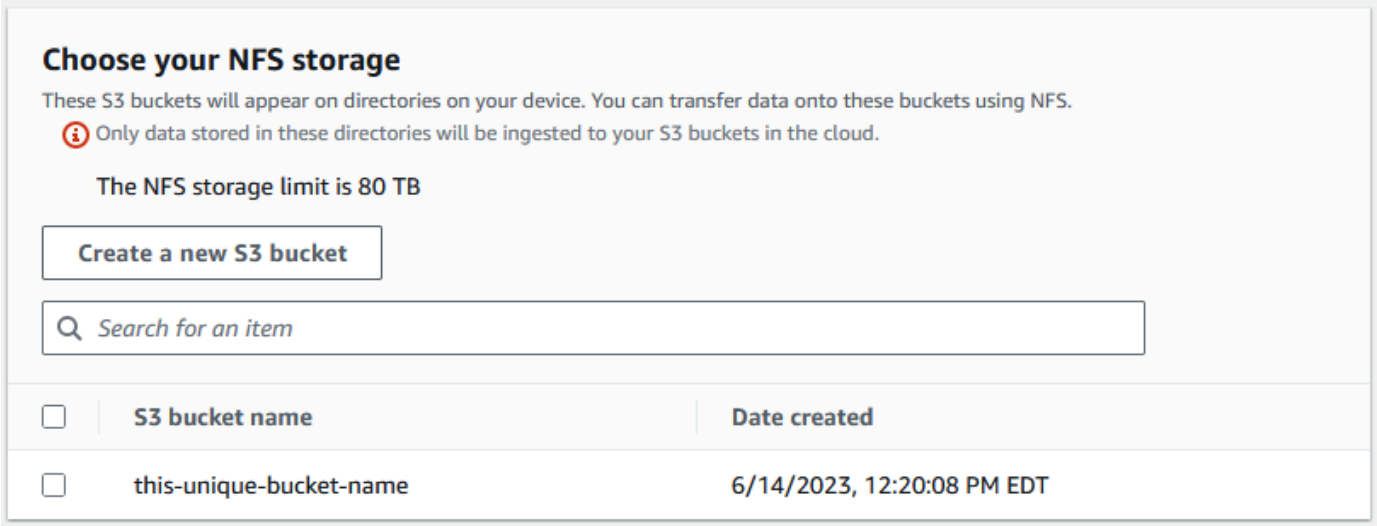
   a. Choose the S3 bucket that you want to use in the **S3 bucket name** list.

   b. In the **Search for an item** field, enter all or part a bucket name to filter the list of available buckets on your entry, then choose the bucket.

   c. Choose the **Create a new S3 bucket** to create a new S3 bucket. The new bucket name appears in the **Bucket name** list. Choose it.

   d. After choosing S3 buckets to use with NFS data transfer, also choose an S3 bucket to use as block storage for AMIs. See the steps to choose an S3 bucket.

You can include one or more S3 buckets. These buckets appear on your device as local S3 buckets.



7. In the **Compute using EC2-compatible instances - *optional*** section, choose Amazon EC2-compatible AMIs from your account to include on the device. Or, in the search field, enter all or part the name of an AMI to filter the list of available AMIs on your entry, then choose the AMI.

   For more information, see Adding an AMI When Ordering Your Device in this guide.

   This feature incurs additional charges. For more information, see AWS Snowball Edge Pricing.

8. Choose the **Next** button.

Step 3: Choose your features and options

Choose the features and options to include in your AWS Snow Family device job, including Amazon EKS Anywhere for Snow, an AWS IoT Greengrass instance, and remote device management capability.

To choose your features and options

1. In the **Amazon EKS Anywhere on AWS Snow** section, to include Amazon EKS Anywhere on AWS Snow, select **Include Amazon EKS Anywhere on Snow** and then do the following.

> **Note**
>
> We recommend that you create your Kubernetes cluster with the latest available Kubernetes version supported by Amazon EKS Anywhere. For more information, see Amazon EKS-Anywhere Versioning. If your application requires a specific version of Kubernetes, use any version of Kubernetes offered in standard or extended support by Amazon EKS. Consider the release and support dates of Kubernetes versions when planning the lifecycle of your deployment. This will help you avoid the potential loss of support for the version of Kubernetes you intend to use. For more information, see Amazon EKS Kubernetes release calendar.

   a. In the **Build your own AMI** section, choose the AMIs you have built for Amazon EKS Anywhere. See Actions to complete before ordering a Snowball Edge device for Amazon EKS Anywhere on AWS Snow.
   b. In the **High availability** section, to operate Amazon EKS Anywhere clusters across multiple Snowball Edge devices, choose the number of devices to include in your order.
2. In the **AWS IoT Greengrass on Snow** section, to include a validated AMI for IoT workloads, select **Install AWS IoT Greengrass validated AMI on my Snow device**.
3. To enable remote management of your Snow Family device by AWS OpsHub or Snowball Edge Client, select **Manage your Snow device remotely with AWS OpsHub or Snowball client**.
4. Select the **Next** button.

Step 4: Choose security, shipping, and notification preferences

Topics

- Choose security preferences
- Choose your shipping preferences
- Choose your notification preferences

Choose security preferences

Setting security adds the permissions and encryption settings for your AWS Snow Family devices job to help protect your data while in transit.

Topics

- Restricting access to the Snow role policy

To set security for your job

1. In the **Encryption** section, choose the **KMS key** that you want to use.
   - If you want to use the default AWS Key Management Service (AWS KMS) key, choose **AWS/importexport (default)**. This is the default key that protects your import and export jobs when no other key is defined.
   - If you want to provide your own AWS KMS key, choose **Enter a key ARN**, provide the Amazon Resource Name (ARN) in the **key ARN** box, and choose **Use this KMS key**. The key ARN will be added to the list.
2. In the **Choose service access type** section, do one of the following:
   - Choose **Snow console will create and use a service-linked role to access AWS resources on your behalf.** to grant AWS Snow Family permissions to use Amazon S3 and Amazon Simple Notification Service (Amazon SNS) on your behalf. The role grants AWS Security Token Service (AWS STS) AssumeRole trust to the Snow service
   - Choose **Add an existing service role to use**, to specify the **Role ARN** that you want, or you can use the default role.

Example of `Condition` object to restrict Snow service actions

Example of restricting Snow service actions by ARN and account IDs.

```
"Condition": {
    "StringEquals": {
        "aws:SourceAccount": "AWS_ACCOUNT_ID"
    }
    "ArnLike": {
        "aws:SourceArn": "arn:aws:snowball:REGION:AWS_ACCOUNT_ID:RESOURCE_ID"
    }
}
```

The following shows these conditions included in a policy.

```
    {
    "Version": "2012-10-17",
    "Id": "__default_policy_ID",
    "Statement": [
        {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
      "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish"
      ],
     "Resource": "arn:aws:sns:us-east-1:123456789012:my-sns-topic",
     "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111122223333"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:sns:us-east-1:555555555555:my-sns-topic"
            }
        }
    }
    ]
    }
```

Example policies for Snowball Edge devices

The following is an example of an Amazon S3 import-only role policy.

```
{
        "Version": "2012-10-17",
```

```
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:GetBucketPolicy",
                    "s3:GetBucketLocation",
                    "s3:ListBucketMultipartUploads",
                    "s3:ListBucket",
                    "s3:PutObject",
                    "s3:AbortMultipartUpload",
                    "s3:ListMultipartUploadParts",
                    "s3:PutObjectAcl",
                    "s3:GetObject"
                ],
                "Resource": [
                    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1",
                    "arn:aws:s3:::DOC-EXAMPLE-BUCKET2/*"
                 ]
            }
        ]
    }
```

```
    {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "importexport.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

The following is an example of an IAM trust relationship for import and export role policies.

```
{
        "Version": "2012-10-17",
        "Statement": [
```

```
            {
                "Effect": "Allow",
                "Action": [
                    "s3:GetBucketLocation",
                    "s3:GetObject",
                    "s3:ListBucket",
                    "s3:GetBucketPolicy"
                ],
                "Resource": [
                    "arn:aws:s3:::DOC-EXAMPLE-BUCKET3/*",
                    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
                ]
            }
        ]
    }
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "importexport.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

> **Note**
>
> You can modify the trust relationship and restrict access to this role based on the customer account number and source arn. See **Restricting Access to the Snow Role Policy** on how to modify the trust relationship to restrict access.

3. Choose **Next.** If the selected **IAM role** has defined a restricted access, the **Create Job** procedure will fail if the access criteria is not met.
4. Choose **Allow**.

5. Choose **Next**.

Restricting access to the Snow role policy

You can restrict access to the selected role based on the customer account number and source ARN.

1. In the navigation pane of the IAM console, choose **Roles**. The console displays the roles for your account.
2. Choose the name of the role that you want to modify, and select the **Trust relationships** tab on the details page.
3. Choose **Edit trust relationships**. Update the trust policy to one of the following:

   To restrict access by **customer account number**:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "importexport.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition":{
          "StringEquals":{
          "aws:SourceAccount":"111122223333"
          }
      }
    }
  ]
}
```

   To restrict access by **source ARN**:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
```

```
            "Service": "importexport.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": "arn:aws:snowball:REGION:555555555555:RESOURCE_ID"
            }
        }
    }]
}
```

To restrict access by both **customer account number** and **source ARN**:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "importexport.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111122223333"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:snowball:REGION:111122223333:RESOURCE_ID"
            }
        }
    }]
}
```

Choose your shipping preferences

Receiving and returning a Snow Family device involves shipping the device back and forth, so it's important that you provide accurate shipping information.

To provide shipping details

1. In the **Shipping Address** section, choose an existing address or add a new address.

- If you choose **Use recent address**, the addresses on file are displayed. Carefully choose the address that you want from the list.
- If you choose **Add a new address**, provide the requested address information. The AWS Snow Family Management Console saves your new shipping information.

> **Note**
>
> The country that you provide in the address must match the destination country for the device and must be valid for that country.

2. In the **Shipping speed** section, choose a shipping speed for the job. This speed shows how quickly the device ships between destinations and doesn't reflect how soon it will arrive after today's date. The shipping speeds you can choose are:

   - **One-Day Shipping (1 business day)**
   - **Two-Day Shipping (2 business days)**
   - **See Shipping Carriers.**

Choose your notification preferences

Notifications update you on the latest status of your AWS Snow Family devices jobs. You create an SNS topic and receive emails from Amazon Simple Notification Service (Amazon SNS) as your job status changes.

To set up notifications

- In the **Set notifications** section, do one of the following:
  - If you want to use an existing SNS topic, choose **Use an existing SNS topic**, and choose the topic Amazon Resource Name (ARN) from the list.
  - If you want to create a new SNS topic, choose **Create a new SNS topic**. Enter a name for your topic and provide an email address.

Notifications will be about one of the following states of your job:

- Job created

- Preparing device
- Preparing shipment
- In transit to you
- Delivered to you
- In transit to AWS
- At sorting facility
- At AWS
- Importing
- Completed
- Canceled

For more information about job status change notifications and encrypted SNS topics, see Notifications for Snow Family devices in this guide.

Select the **Next**.

Step 5: Review job summary and create your job

After you provide all the necessary job details for your AWS Snow Family devices job, review the job and create it.

1. In the **Review job order** page, review all the sections before you create the job. If you want to make changes, choose **Edit** for the appropriate section, and edit the information.
2. When you are done reviewing and editing, choose **Create job**. After you create a job to order a Snow Family device, you can cancel it while it is in the *Job created* state without incurring any charges.

   Jobs are subject to export control laws in specific countries and might require an export license. US export and re-export laws also apply. Diversion from the country and US laws and regulations is prohibited.

   > **Note**
   >
   > Snowcone devices are not provided with power cords, and one must be provided separately.

After your job is created, you can see the status of the job in the **Job status** section. For detailed information about job statuses, see Job Statuses.

Download AWS OpsHub

The AWS Snow Family devices offer a user-friendly tool, AWS OpsHub for Snow Family, that you can use to manage your devices and local AWS services.

With AWS OpsHub installed on your client computer, you can perform tasks such as the following:

- Unlocking and configuring single or clustered devices
- Transferring files
- Launching and managing instances running on Snow Family devices.

For more information, see Using AWS OpsHub for Snow Family to Manage Devices.

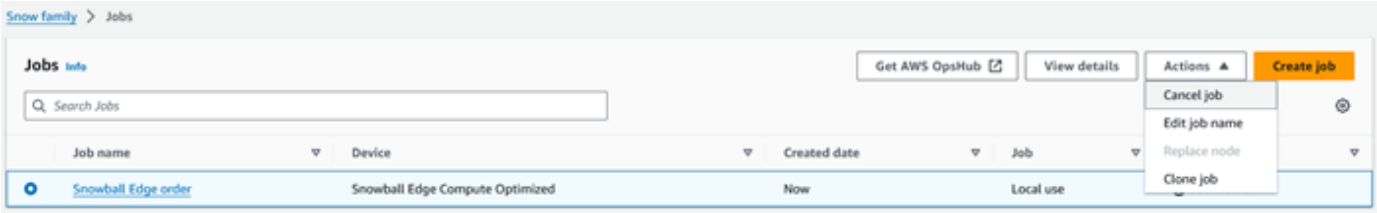
To download and install AWS OpsHub for Snow Family

1. In the AWS Snowball resources, click AWS OpsHub. In the AWS OpsHub section with the Download links, choose the appropriate download link to install AWS OpsHub for your operating system.
2. In the **AWS OpsHub** section, choose **Download** for your operating system, and follow the installation steps. When you are finished, choose **Next**.

Cancelling a job through the AWS Snow Family Management Console

After creating a job to order a Snow Family device, you can cancel the job through the AWS Snow Family Management Console. If you cancel the job, you won't receive the device you ordered. You can only cancel the job while the job status is *Job created*. After the job progresses past this status, you cannot cancel the job. For more information, see Job Statuses.

1. Log in to the AWS Snow Family Management Console.
2. Choose the job to cancel.
3. Choose **Actions**. From the menu that appears, choose **Cancel job**.



4. The **Cancel job** window appears. To confirm cancelling the job, enter the **`job name`** and choose **Cancel job**. In the list of jobs, **Cancelled** appears in the **Status** column.

Receiving the Snowball Edge

When you receive the AWS Snowball Edge device, you might notice that it doesn't come in a box. The device is its own physically rugged shipping container. When the device first arrives, inspect it for damage or obvious tampering. If you notice anything that looks suspicious about the device, don't connect it to your internal network. Instead, contact AWS Support and inform them of the issue so that a new device can be shipped to you.

> **Important**
>
> The AWS Snowball Edge device is the property of AWS. Tampering with an AWS Snowball Edge device is a violation of the AWS Acceptable Use Policy. For more information, see AWS Acceptable Use Policy.

The device looks like the following image.

If you're ready to connect the device to your internal network, see the next section.

Next: Connecting to Your Local Network

Connecting to Your Local Network

Using the following procedure, you connect the AWS Snowball Edge device to your local network. The device doesn't need to be connected to the internet. The device has three doors: a front, a back, and a top.

To connect the device to your network

1. Open the front and back doors, sliding them inside the device door slots. Doing this gives you access to the touch screen on the LCD display embedded in the front of the device, and the power and network ports in the back.

   > **Note**
   >
   > Don't close the front and back doors while you're using the Snowball Edge device. The open doors allow air to cool the device. Closing the doors while using the device may cause the device to shut down to prevent overheating.

2. Open the top door and remove the provided power cable from the cable nook, and plug the device into power.
3. Choose one of your RJ45, SFP+, or QSFP+ network cables, and plug the device into your network. The network ports are on the back of the device.
4. Power on the AWS Snowball Edge device by pressing the power button above the LCD display.
5. When the device is ready, the LCD display shows a short video while the device is getting ready to start. After about 10 minutes, the device is ready to be unlocked.
6. (Optional) Change the default network settings through the LCD display by choosing **CONNECTION**.

   You can change your IP address to a different static address, which you provide by using the following procedure.

To troubleshoot boot-up problems, see Troubleshooting boot-up problems.

To change the IP address of an AWS Snowball Edge device

1. On the LCD display, choose **CONNECTION**.

   A screen appears that shows you the current network settings for the AWS Snowball Edge device. The IP address below the drop-down box is automatically updated to reflect the DHCP address that the AWS Snowball Edge device requested.
2. (Optional) Change the IP address to a static IP address. You can also keep it as is.

The device is now connected to your network.

Important

To prevent corrupting your data, don't disconnect the AWS Snowball Edge device or change its connection settings while it's in use.

Next: Getting credentials to access a Snow Family device

Getting credentials to access a Snow Family device

Each job has a set of credentials that you must get from the AWS Snow Family Management Console or the job management API to authenticate your access to the Snow Family device. These credentials are an encrypted manifest file and an associated unlock code. The manifest file contains important information about the job and permissions associated with it.

Note

You get your credentials after the device is in transit to you. You can see the status of your job in the AWS Snow Family Management Console. For more information, see Job Statuses.

To get your credentials using the console

1. Sign in to the AWS Management Console and open the AWS Snow Family Management Console.
2. On the console, search the table for the specific job to download the job manifest for, and then choose that job.
3. Expand that **Job status** pane, and choose **View job details**.
4. In the details pane that appears, expand **Credentials** and then do the following:
   - Make a note of the unlock code (including the hyphens), because you need to provide all 29 characters to unlock the device.
   - In the dialog box, choose **Download manifest**, and follow the instructions to download the job manifest file to your computer. The name of your manifest file includes your **Job ID**.

> **Note**
>
> We recommend that you don't save a copy of the unlock code in the same location in the computer as the manifest for that job. For more information, see Best practices for using the Snowball Edge device.

Now that you have your credentials, the next step is to download the Snowball Edge client, which is used to unlock the AWS Snowball Edge device.

Next: Downloading and Installing the Snowball Edge client

Downloading and Installing the Snowball Edge client

The Snowball Edge client is the tool that you use to unlock the AWS Snowball Edge device. We recommend that you use the AWS OpsHub for Snow Family application. For instructions, see Using AWS OpsHub for Snow Family to Manage Devices.

You can download and install the Snowball Edge client from AWS Snowball resources page to a powerful workstation that you own.

Next: Unlocking the Snow Family device

Unlocking the Snow Family device

This section describes unlocking the Snow Family device using the Snowball Edge CLI. To unlock the device using AWS OpsHub, a graphical user interface (GUI) tool for Snow Family devices, see Unlock a device.

Before using a Snow Family device device to transfer data or perform edge compute tasks, you need to unlock the device. When unlocking the device, you authenticate your ability to access it by providing two forms of credentials: a 29-digit unlock code and a manifest file. After you unlock the device, you can further configure the device, move data to or from it, set up and use Amazon EC2-compatible instances, and more.

Before unlocking a device, the device must be plugged in to power and network, turned on, and an IP address assigned. See Connecting to Your Local Network. You will need the following information about the Snow Family device:

- Download and install the Snowball Edge client. For more information, see Using the Snowball Edge Client.
- Get the credentials from the AWS Snow Family Management Console. For one or more standalone devices, the unlock codes and manifest file for each Snow Family device. For a cluster of Snowball Edge devices, the one unlock code and one manifest file for the cluster. For more information on downloading credentials, see Getting credentials to access a Snow Family device.
- Power on each device and connect it to your network. For more information, see Connecting to Your Local Network.

To unlock a standalone device with the Snowball Edge client

1. Find the IP address for the AWS Snowball Edge device on the LCD display of the AWS Snowball Edge device, under the **Connections** tab. Make a note of that IP address.
2. Use the `unlock-device` command to authenticate your access to the Snow Family device with the IP address of the Snow Family device and your credentials, as follows.

```
  snowballEdge unlock-device --endpoint https://ip-address-of-device --manifest-
file /Path/to/manifest/file.bin --unlock-code 29-character-unlock-code
```

The device indicates it was unlocked successfully with the following message.

```
Your Snowball Edge device is unlocking. You may determine the unlock state of
 your device using the describe-device command. Your Snowball Edge device will be
 available for use when it is in the UNLOCKED state.
```

If the command returns `connection refused`, see Troubleshooting unlocking a Snow Family device.

Example of `unlock-device` command

In this example, the IP address for the device is `192.0.2.0`, the manifest file name is `JID2EXAMPLE-0c40-49a7-9f53-916aEXAMPLE81-manifest.bin`, and the 29-character unlock code is `12345-abcde-12345-ABCDE-12345`.

```
  snowballEdge unlock-device --endpoint https://192.0.2.0 --manifest-file /
Downloads/JID2EXAMPLE-0c40-49a7-9f53-916aEXAMPLE81-manifest.bin /
    --unlock-code 12345-abcde-12345-ABCDE-12345
```

To unlock a cluster of Snowball Edge devices with the Snowball Edge client

1. Find the IP address of each of the devices in the cluster on the LCD display of each AWS Snowball Edge device, under the **Connections** tab. Make a note of the IP addresses.
2. Use the `snowballEdge unlock-cluster` command to authenticate your access to the cluster of AWS Snowball Edge device devices with the IP address of one of the devices in the cluster, your credentials, and the IP addresses of all devices in the cluster as follows.

```
snowballEdge unlock-cluster --endpoint https://ip-address-of-device --manifest-
file Path/to/manifest/file.bin --unlock-code 29-character-unlock-code --device-ip-
addresses ip-address-of-cluster-device-1 ip-address-of-cluster-device-2 ip-address-
of-cluster-device-3
```

The cluster of devices indicates it was unlocked successfully with the following message.

```
Your Snowball Edge Cluster is unlocking. You may determine the unlock state of your
 cluster using the describe-cluster command. Your Snowball Edge Cluster will be
 available for use when your Snowball Edge devices are in the UNLOCKED state.
```

If the command returns `connection refused`, see Troubleshooting unlocking a Snow Family device.

Example of `unlock-cluster` command

In this example for a cluster of five devices, the IP address for one of the devices in the cluster is `192.0.2.0`, the manifest file name is `JID2EXAMPLE-0c40-49a7-9f53-916aEXAMPLE81-manifest.bin`, and the 29-character unlock code is `12345-abcde-12345-ABCDE-12345`.

```
snowballEdge unlock-cluster --endpoint https://192.0.2.0 --manifest-file /
Downloads/JID2EXAMPLE-0c40-49a7-9f53-916aEXAMPLE81-manifest.bin /

  --unlock-code 12345-abcde-12345-ABCDE-12345 --device-ip-addresses 192.0.2.0
 192.0.2.1 192.0.2.2 192.0.2.3 192.0.2.4
```

Troubleshooting unlocking a Snow Family device

If the `unlock-device` command returns `connection refused`, you may have mistyped the command syntax or the configuration of your computer or network may be preventing the command from reaching the Snow device. Take these actions to resolve the situation:

1. Ensure the command was entered correctly.
   a. Use the LCD screen on the device to verify the IP addressed used in the command is correct.
   b. Ensure that the path to the manifest file used in the command is correct, including the file name.
   c. Use the AWS Snow Family Management Console to verify the unlock code used in the command is correct.
2. Ensure the computer you are using is on the same network and subnet as the Snow device.
3. Ensure the computer you are using and the network are configured to allow access to the Snow device. Use the `ping` command for your operating system to determine if the computer can reach the Snow device over the network. Check the configurations of antivirus software, firewall configuration, virtual private network (VPN), or other configurations of your computer and network.

Now you can begin using the Snow Family device.

Next: Setting Up Local Users

Setting Up Local Users

Following are steps to set up a local administrator on your AWS Snowball Edge device.

1. **Retrieve your root user credentials**

   Use the `snowballEdge list-access-keys` and `snowballEdge get-secret-access-key` to get your local credentials. For more information, see Getting Credentials.

2. **Configure the root user credential using `aws configure`**

   Supply the `AWS Access Key ID`, `AWS Secret Access Key`, and `Default region name`. The region name must be `snow`. Optionally supply a `Default output format`. For more information about configuring the AWS CLI, see Configuring the AWS CLI in the *AWS Command Line Interface User Guide*.

3. **Create one or more local users on your device**

   Use the `create-user` command to add users to your device.

   ```
   aws iam create-user --endpoint endpointIPaddress:6078 --profile ProfileID --region
    snow --user-name UserName
   ```

   After you add users according to your business needs, you can store your AWS root credentials in a safe location and only use them for account and service management tasks. For more information about creating IAM users, see Creating an IAM user in your AWS account in the *IAM User Guide*.

4. **Create an access key for your user**

   > **Warning**
   >
   > This scenario requires IAM users with programmatic access and long-term credentials, which presents a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed. Access keys can be updated if necessary. For more information, see Updating access keys in the *IAM User Guide*.

Use the `create-access-key` command to create an access key for your user.

```
aws iam create-access-key --endpoint endpointIPaddress:6078 --profile ProfileID --
region snow --user-name UserName
```

Save the access key information to a file and distribute to your users.

5. **Create an access policy**

   You might want different users to have different levels of access to functionality on your device. The following example creates a policy document named `s3-only-policy` and attaches it to a user.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "*"
    }
  ]
}
```

```
aws iam create-policy --endpoint endpointIPaddress:6078 --profile ProfileID --
region snow --policy-name s3-only-policy --policy-document file://s3-only-policy
```

6. **Attach the policy to your user**

   Use the `attach-user-policy` to attach the s3-only-policy to a user.

```
  aws iam attach-user-policy --endpoint endpointIPaddress:6078 --profile ProfileID
 --region snow --user-name UserName --policy-arn arn:aws:iam::AccountID:policy/
POLICYNAME
```

For more information about using IAM locally, see Using IAM Locally.

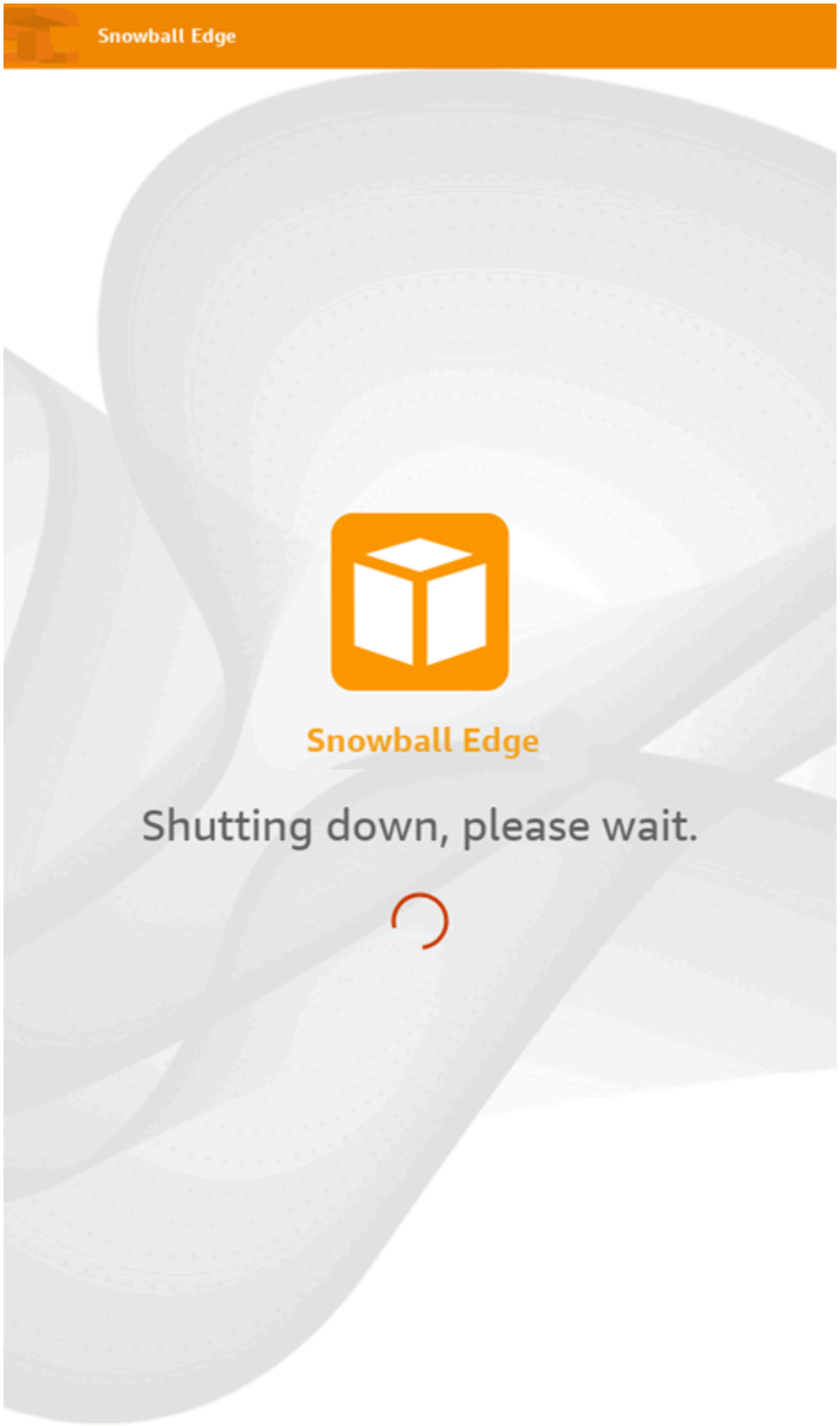
Next: Using an AWS Snowball Edge Device

Rebooting the Snow Family device

Before you reboot a Snow Family device, make sure that all data transfer to the device has stopped.

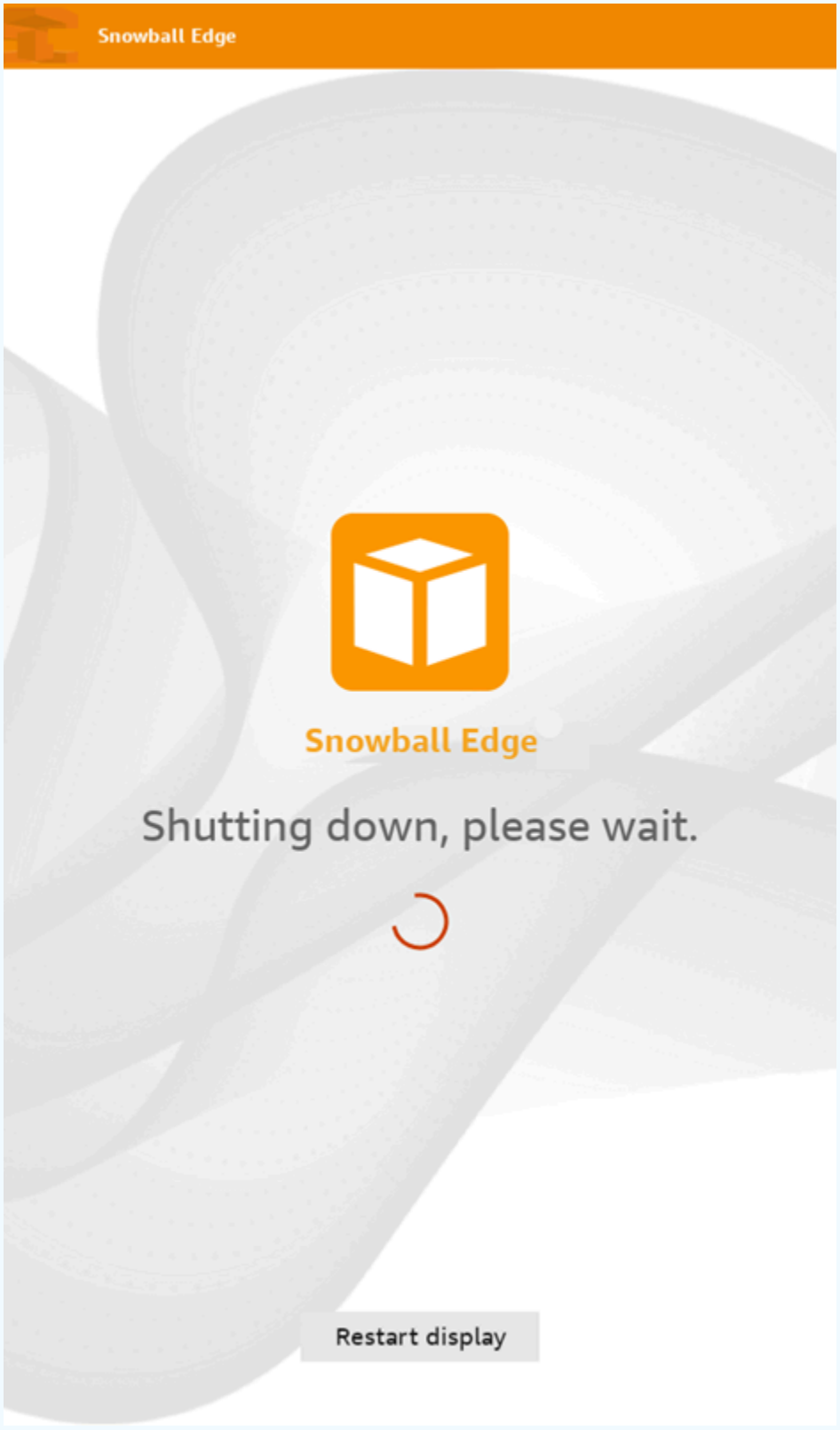
To reboot the device using the power button:

1. When all communication with the device has ended, turn it off by pressing the power button located above the LCD screen. It takes about 20 seconds for the device to shut down. While the device is shutting down, the LCD screen displays a message indicating the device is shutting down.


Snowball Edge
Snowball Edge
Shutting down, please wait.

Note

If the LCD screen is displaying the shutdown message when the device is not actually being shut down, press the **Restart display** button on the screen to return the screen to normal operation.



2. Press the power button. When the device is ready, the LCD display shows a short video while the device is getting ready to start. After about 10 minutes, the device is ready to be unlocked.

3. Unlock the device. See Unlocking the Snow Family device.

To reboot the device using the Snowball Edge client:

1. When all communication with the device has ended, use the `reboot-device` command to reboot it. When the device is ready, the LCD display shows a short video while the device is getting ready to start. After about 10 minutes, the device is ready to be unlocked.

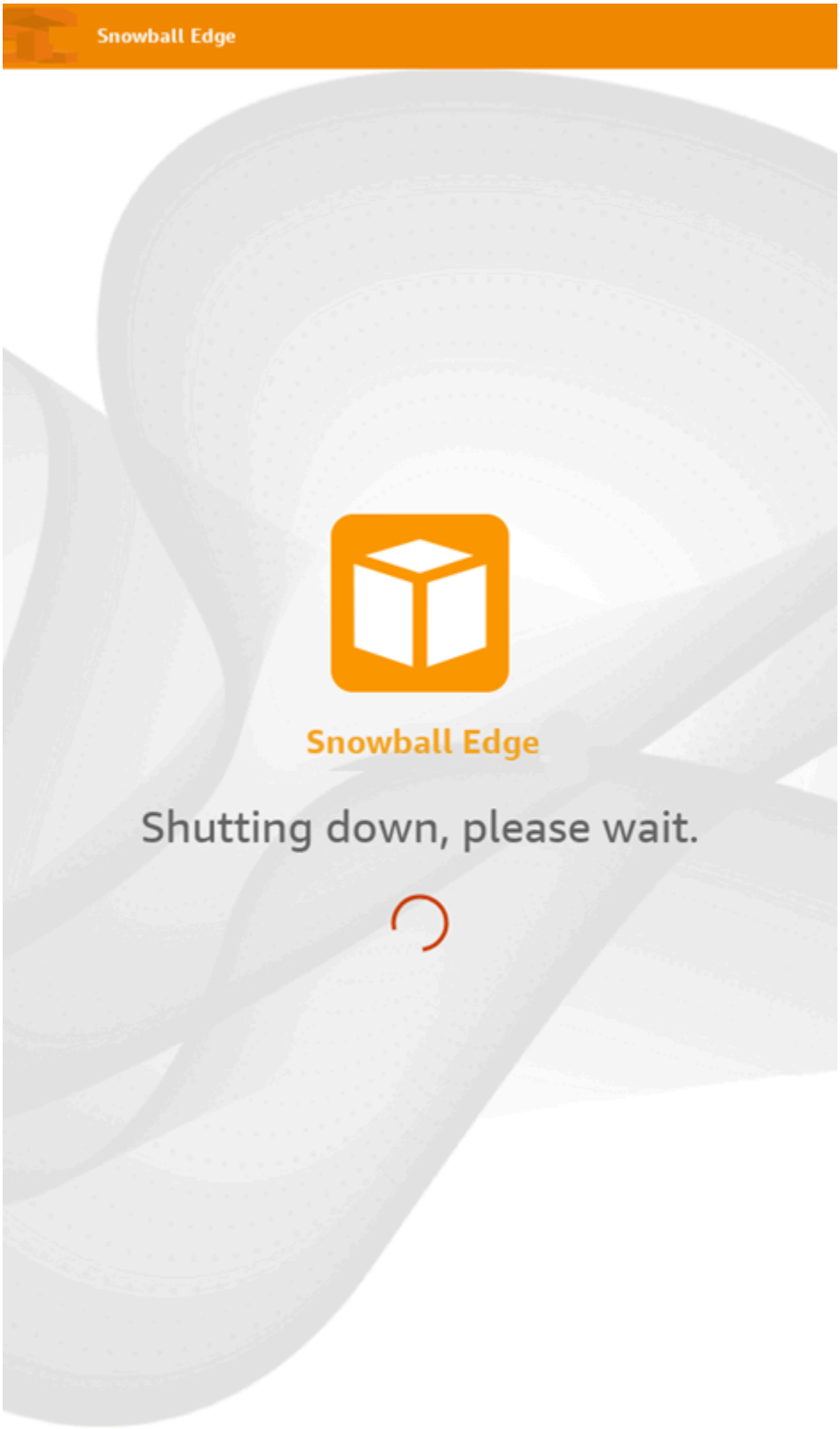
```
snowballEdge reboot-device
```

2. Unlock the device. See Unlocking the Snow Family device.

Powering off the Snowball Edge

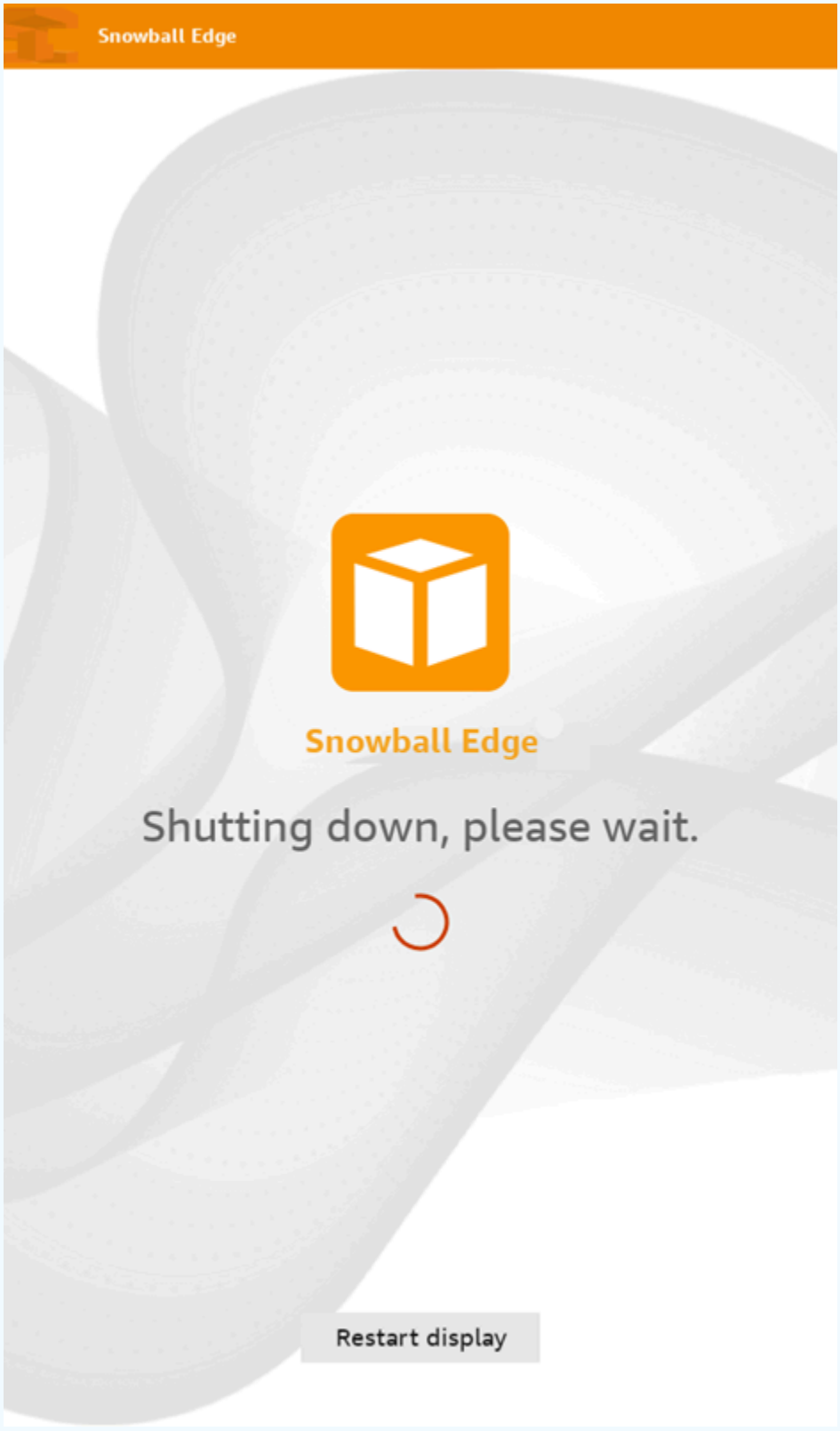
When you've finished transferring data on to the AWS Snowball Edge device, prepare it for its return trip to AWS. Before you continue, make sure that all data transfer to the device has stopped. If you were using the NFS interface to transfer data, disable it before you power off the device. For more information, see Managing the NFS interface.

When all communication with the device has ended, turn it off by pressing the power button located above the LCD screen. It takes about 20 seconds for the device to shut down. While the device is shutting down, the LCD screen displays a message indicating the device is shutting down.


Snowball Edge
Snowball Edge
Shutting down, please wait.

Note

If the LCD screen is displaying the shutdown message when the device is not actually being shut down, press the **Restart display** button on the screen to return the screen to normal operation.



After the device shuts down, the shipping information appears on the E Ink display.

Next: Returning the Snowball Edge Device

Returning the Snowball Edge Device

The prepaid shipping information on the E Ink display contains the address to return the AWS Snowball Edge device. For information about with which carrier to return the device, see Shipping carriers.

> **Note**
>
> Once you return the Snow device for import into Amazon S3, AWS will start ingestion of the data after ensuring the device has not been tampered with and that the device is healthy. In case you do not want the data on the device to be ingested to your destination S3 bucket, you can request to cancel the Snow job. If you cancel the job, we will skip the data transfer and securely erase the device following the established processes. We are not able to hold a device containing your data at our facilities due to our strict chain of custody and operating procedures.

The device is delivered to an AWS sorting facility and forwarded to the AWS data center. The carrier automatically provides a tracking number for your job to the AWS Snow Family Management Console. You can access the tracking number and a link to the carrier's tracking website by viewing the job's status details in the console or by making calls to the job management API.

You can track the status changes of your job through the AWS Snow Family Management Console as AWS processes the device. You can use Amazon SNS notifications if you selected that option during job creation, or you can make calls to the job management API. For more information about this API, see AWS Snowball API Reference.

The final status values include when the AWS Snowball Edge device has been received by AWS, when data import begins, and when the job is completed.

Preparing an AWS Snowball Edge device for shipping

The following explains how to prepare an AWS Snowball Edge device and ship it back to AWS.

To prepare an AWS Snowball Edge device for shipping

1. Disconnect and stow the power cable in the cable nook on top of the AWS Snowball Edge device.
2. Close the doors on the back, top, and front of the AWS Snowball Edge device. Press in until you hear and feel them click.

You don't need to pack the AWS Snowball Edge device in a container, because the device itself is its own physically rugged shipping container. The E Ink display on the top of the AWS Snowball Edge device displays the return shipping information when the device is turned off.

Job-Type Specific Consideration

> **Important**
>
> If you are importing data, don't delete your local copies of the transferred data until the import to Amazon S3 is successful at the end of the process, and you can verify the results of the data transfer.

Return shipping for Snow Family devices

The AWS Snowball Edge device is shipped from and delivered to an AWS data center. The prepaid shipping information on the E Ink screen on the device includes the address to return the AWS Snowball Edge device. The shipping speed for the return matches the original shipping speed when you received the device. You can track status changes using the AWS Snow Family Management Console, and track the package's progress through your region's carrier.

For more information about how to return your AWS Snowball Edge device, see Shipping carriers.

> **Important**
>
> Unless instructed otherwise by AWS, never affix a separate shipping label to the AWS Snowball Edge device. Always use the shipping information that's displayed on the AWS Snowball Edge device E Ink display.

Shipping carriers

When you create a job to order a Snow Family device, you provide the address to ship the AWS Snowball Edge device to. The carrier that supports your region handles the shipping of devices from AWS to you, and from you back to AWS. You can see the outbound shipping information when your job reaches the **Preparing shipment** status.

There's a tracking number for every AWS Snowball Edge device that's shipped. You can find the tracking number and a link to the tracking website using the AWS Snow Family Management Console job dashboard or the job management API.

These carriers are supported for AWS Snowball Edge devices:

- For India, Blue Dart is the carrier.
- For Korea, Japan, Australia, and Indonesia, Kuehne + Nagel, is the carrier.
- For China and Hong Kong, S.F. Express is the carrier.
- For all other regions, UPS is the carrier.

Topics

AWS Snowball Edge UPS pickups in the EU, US, UK, South Africa, and Canada

UPS can often pick up your device in the EU, US, UK, South Africa, and Canada. Here are some helpful guidelines:

- Schedule a pickup with UPS directly, or take the AWS Snowball Edge device to a UPS package drop-off facility to be shipped to AWS.
- The prepaid UPS shipping label on the E Ink display contains the return address for the AWS Snowball Edge device.
- The AWS Snowball Edge device is delivered to an AWS sorting facility and forwarded to a AWS data center. UPS provides you with a tracking number.

Important

Unless instructed otherwise by AWS, never affix a separate shipping label to the AWS Snowball Edge device. Always use the shipping information that is displayed on the device's E Ink display.

UPS ships Snowball Edge devices to the following EU member countries: Austria, Belgium, Bulgaria, Croatia, Republic of Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Italy, Ireland, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, and Sweden.

Note

Orders between the United Kingdom and European Union countries are now considered to be international, and require approval through a special international process. If you need to ship your device between the UK and the EU, email us at `<snowball-shipping@amazon.com>` to request a commercial invoice prior to arranging pick up or drop off with UPS.

UPS services for Snow family of products are domestic only within a country.

AWS Snowball Pickups in UK

In the United Kingdom, keep the following information in mind for UPS to pick up a Snowball Edge:

- You arrange for UPS to pick up the AWS Snowball Edge device by scheduling a pickup with UPS directly, or take the AWS Snowball Edge device to a UPS package drop-off facility to be shipped to AWS.

- The prepaid UPS shipping label on the E Ink display contains the correct address to return the AWS Snowball Edge device.
- The AWS Snowball Edge device is delivered to an AWS sorting facility and forwarded to the AWS data center. UPS automatically reports back a tracking number for your job.

> **Important**
>
> Unless personally instructed otherwise by AWS, never affix a separate shipping label to the AWS Snowball Edge device. Always use the shipping information that is displayed on the device's E Ink display.

UPS services for Snow family of products is domestic only within a country.

> **Note**
>
> Since January 2021, UK is no longer a part of EU. Orders between UK and other EU countries are international orders, a non-general Availability process only approved through a special international process. If a customer has been approved and is returning a device from an EU-country back to LHR or from UK back to an EU-country, they must first request a return to `<snowball-shipping@amazon.com>` so a Commercial Invoice can be provided prior to arranging pick up/drop off with UPS.

AWS Snowball pickups in Brazil

Here are some guidelines for UPS to pick up a Snowball Edge device in Brazil:

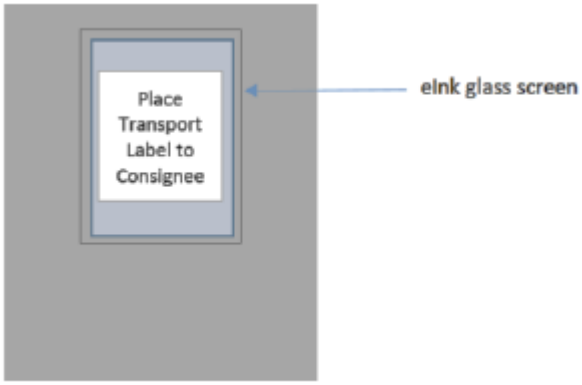
- When you're ready to return a Snowball Edge device, call 0800-770-9035 to schedule a pickup with UPS.
- Snowball Edge is available domestically within Brazil, which includes 26 states and the Distrito Federal.
- If you have a Cadastro Nacional de Pessoa Juridica (CNPJ) tax ID, be sure that you know this ID before you create your job.
- You should issue the appropriate document to return the Snowball Edge device. Confirm with your tax department which of the following documents is required in your state, according to your Imposto sobre Circulação de Mercadorias e Serviços (ICMS) registration:

- **Within São Paulo –** A non-ICMS declaration and an Electronic Tax Invoice (NF-e) are usually required.
- **Outside São Paulo** – The following are usually required:
  - A non-ICMS declaration
  - A nota fiscal avulsa
  - An Electronic Tax Invoice (NF-e)

> **Note**
>
> For non-ICMS taxpayer declaration, we recommend that you generate four copies of the declaration: one for your records, and the other three for transport.

AWS Snowball pickups in Australia

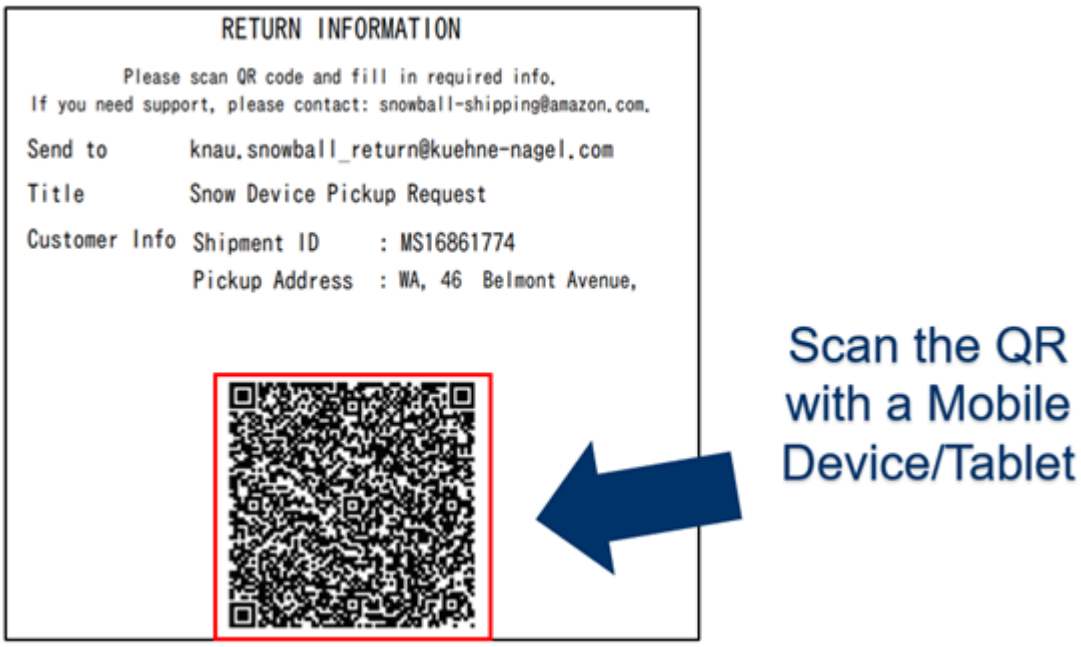
In Australia, if you're shipping an AWS Snowball Edge device back to AWS, place the return transport label (found in the pouch containing these instructions) over the E Ink label on the Snow device.



> **Note**
>
> If you didn't receive a return label with your device, email knau.snowball_return@kuehne-nagel.com with your device serial number or your reference number.

To arrange the return of the Snow Family device, scan the QR code on the return instructions with your mobile device. On your device, a hyperlink to an email message appears. The message

contains information such as email address, subject, and control number or consignment number. Fill in the pickup date, name, and contact details, or provide a new pickup address if there are any changes.



AWS Snowball pickups in India

In India, Blue Dart picks up the Snowballdevice. When you are ready to return your Snowball device, turn it off and prepare it for return shipping. To schedule pickup, email snowball-pickup@amazon.com with **Snowball Pickup Request** in the subject line. In the email, include the following information:

- **Job ID** – The job ID associated with the Snowball that you want returned to AWS.
- **AWS account ID** – The ID for the AWS account that created the job.
- **Earliest Pickup Time** (your local time) – The earliest time of day that you want the Snowball picked up.
- **Latest Pickup Time** (your local time) – The latest time of day that you want the Snowball picked up.
- **Special Instructions** (optional) – Any special instructions for picking up the Snowball, including contact details for coordinating pickup.

The Snowball team arranges the pickup with Blue Dart and sends you a confirmation email. Blue Dart provides you with a paper shipping label and picks up the Snowball device.

Important

When using a Snowball in India, remember to file all relevant tax paperwork with your state.

AWS Snowball Edge pickups in Korea

In Korea, Kuehne + Nagel handles your pickups. When you are ready to return your device, send an email to snowball-shipping@amazon.com with *Snowball Pickup Request* in the subject line so we can schedule the pickup for you. In the body of the email, include the following information:

- **Job ID** – The job ID associated with the Snowball that you want returned to AWS.
- **Pickup Address** -The address where the device is picked up.
- **Pickup Date** - The earliest day you would like the device picked up.
- **Point of contact details** – The name, email address, and local phone number that Kuehne + Nagel can use to get in touch with you if needed.

Soon, you will get a follow-up email from the Snowball team with information regarding the pickup at the address your device you provided. Power cycle the device and be ready for pickup usually between 1300 and 1500.

AWS Snowball Edge pickups in Hong Kong

In Hong Kong, S.F. Express handles your pickups. When you are ready to return your device, send an email to snowball-shipping-ap-east-1@amazon.com with *Snowball Pickup Request* in the subject line so we can schedule the pickup for you. In the body of the email, include the following information:

- Job ID
- AWS account ID
- Contact name
- Contact phone number
- Contact email address
- The day you want the device(s) picked up
- Earliest pickup time

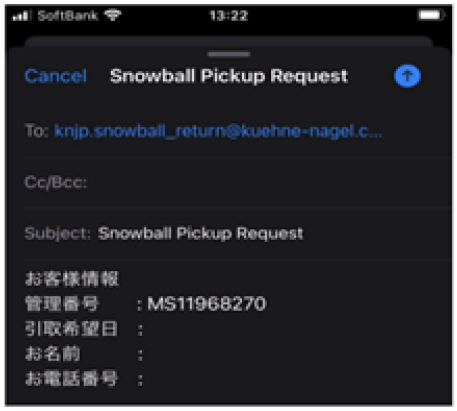
- Latest pickup time
- Pickup address

Once you arrange a pickup date with S.F. Express, it can't be rescheduled.

The device will be delivered to AWS by S.F. Express. The S.F. Express tracking number for the return shipment tells you when it was delivered.

AWS Snowball Pickups in Singapore, Japan, and Indonesia

In Singapore, Japan, and Indonesia, when you are ready to return your device, scan the QR code displayed on the return E Ink label with your mobile phone. This will take you directly onto an email template. Please fill in pick up date/time and contact details.



> **Note**
>
> If your pickup address is different from the address where the device was delivered, please add the new address in the email body so the appointed carrier can be informed.

Note

In Japan, the shipping company charges a shipping fee of $120.00. The description of the fee indicates Snowball, but the fee applies to shipping all Snow Family devices.

AWS Snowball receiving and returning in Dubai, United Arab Emirates

Here are some guidelines that you must follow when receiving or returning an AWS Snowball Edge device in Dubai.

Receiving a Snowball Edge device

When receiving a Snowball Edge device in a free zone, when you are notified by UPS that the package is ready for delivery, apply for, obtain, and share the gate pass for your free zone.

If you are in a free zone or in the Mainland, sign the proof of delivery (POD) when you receive the device.

Returning a Snowball Edge device

When returning a Snowball Edge device, arrange for UPS to pick up the device by scheduling a pickup with UPS directly on 600 544 743 or via the UPS website. Ensure the return shipping information is displayed on the E Ink display before the device is picked up. See Returning the Snowball Edge Device. In a free zone, when you are notified that a UPS driver is assigned for pickup of the device, apply for, obtain, and share the Gate Pass for your free zone.

The prepaid UPS shipping information on the E Ink display contains the correct address to return the Snowball Edge device.

The Snowball Edge device is delivered to an AWS sorting facility and forwarded to the AWS data center. UPS automatically provides a tracking number for your job.

Important

Unless personally instructed otherwise by AWS, never affix a separate shipping label to the Snowball Edge device. Always use the shipping label that is displayed on the device's E Ink display.

UPS services for Snow family of products is domestic only within a country.

Shipping speeds

Each country has different shipping speeds available. These shipping speeds are based on the country in which you're shipping an AWS Snowball Edge device. Shipping speeds are as follows:

- **Australia, Japan, Singapore, Indonesia, S.Korea** – When shipping within these countries, you have access to the standard shipping speed of 1 – 3 days.
- **Brazil** – When shipping within Brazil, you have access to UPS Domestic Express Saver shipping, which delivers within two business days during commercial hours. Shipping speeds might be affected by interstate border delays.
- **European Union (EU)** – When shipping to any of the countries within the EU, you have access to express shipping. Typically, AWS Snowball Edge devices shipped express are delivered in about a day. In addition, most countries in the EU have access to standard shipping, which typically takes less than a week, one way.
- **Hong Kong** – When shipping within Hong Kong, you have access to express shipping.
- **India** – When shipping within India, Snowball Edge devices are sent out within 7 working days of AWS receiving all related tax documents.
- **Dubai, United Arab Emirates** - You have access to Courier Express Saver shipping.
- **United Kingdom (UK)** – When shipping within the UK, you have access to express shipping. Typically, Snowball Edge devices shipped express are delivered in about a day. In addition, you have access to standard shipping, which typically takes less than a week, one way.
- **United States of America (US) and Canada** – When shipping in the US or in Canada, you have access to one-day shipping and two-day shipping.

Monitoring the Import Status

To monitor the status of your import job in the console, sign in to the AWS Snow Family Management Console in the AWS Region where the job was created. Choose the job you want to track from the table, or search for it by your chosen parameters in the search bar above the table. After you select the job, detailed information appears for that job within the table, including a bar that shows real-time status of your job.

> **Note**
>
> If we are unable to import data to our data centers from the Snow device due to any issue with access permissions you have configured, we will attempt to notify you and you will

have 30 days from the date we provide the notification to resolve the issue. If the issue is not resolved, we may cancel your AWS Snow Family job and delete data from the device.

After your device arrives at AWS, your job status changes from **In transit to AWS** to **At AWS**. On average, it takes a day for your data import into Amazon S3 to begin. When it does, the status of your job changes to **Importing**. It will take approximately the same amount of time for AWS to import your data from the Snow Family device as it did for you to move it to the Snow Family device. After your data is imported, the job status changes to **Completed** status.

Now your first data import job into Amazon S3 using AWS Snowball is complete. You can get a report about the data transfer from the console. To access this report from the console, select the job from the table, and expand it to reveal the job's detailed information. Choose **Get report** to download your job completion report as a PDF file. For more information, see Getting your job completion report and logs on the console.

Next: Getting your job completion report and logs on the console

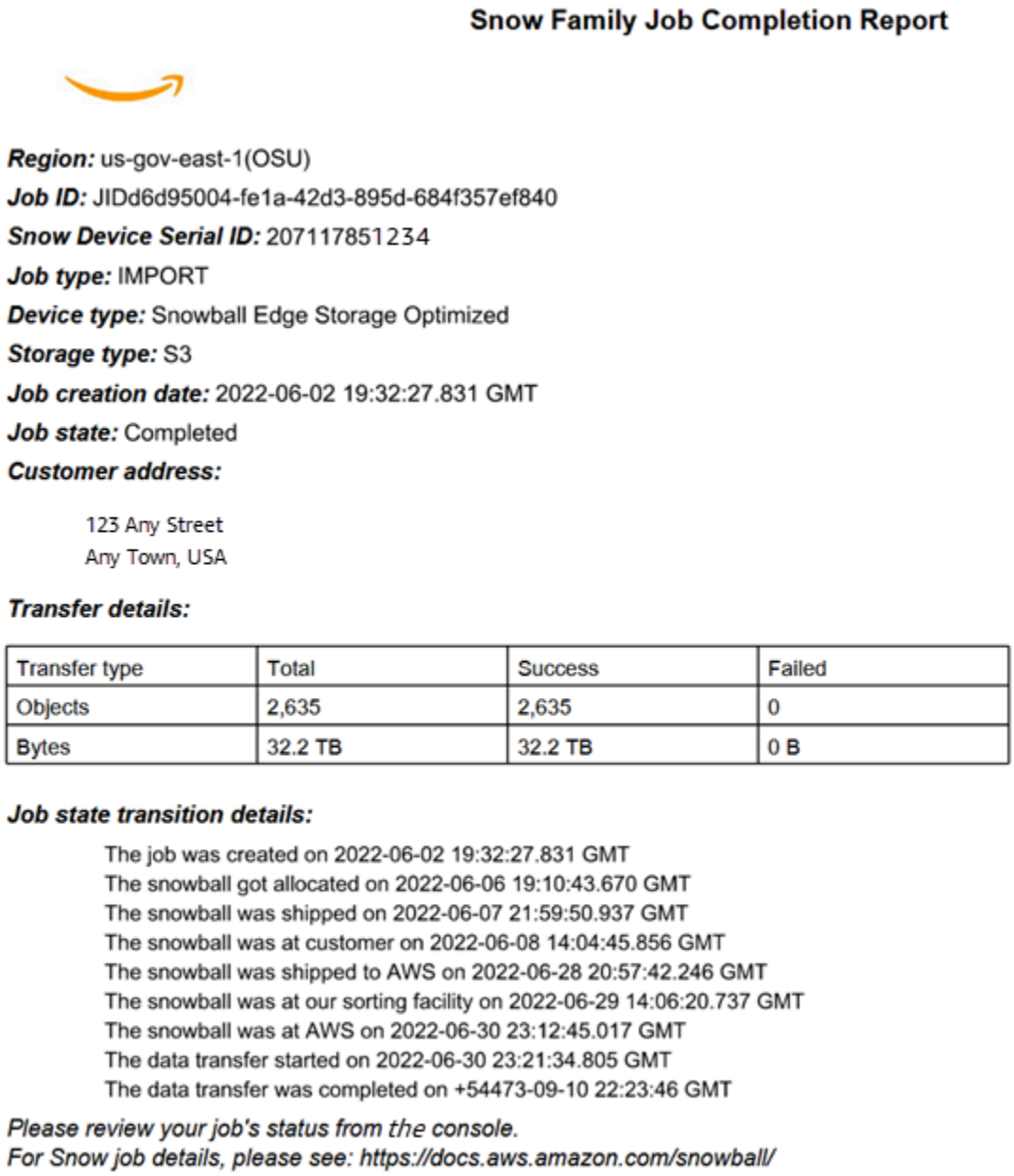
Getting your job completion report and logs on the console

When data is imported into or exported out of Amazon S3, you get a downloadable PDF job report. For import jobs, this report becomes available at the very end of the import process. For export jobs, your job report typically becomes available for you while the AWS Snowball Edge device for your job part is being delivered to you. There are no job completion reports available for *Local Use* job type.

The job report provides you insight into the state of your Amazon S3 data transfer. The report includes details about your job or job part for your records. The job report also includes a table that provides a high-level overview of the total number of objects and bytes transferred between the device and Amazon S3.

For deeper visibility into the status of your transferred objects, you can look at the two associated logs: a success log and a failure log. The logs are saved in comma-separated value (CSV) format, and the name of each log includes the ID of the job or job part that the log describes.

You can download the report and the logs from the AWS Snow Family Management Console. Below is a sample report.



Snow Family Job Completion Report

Region: us-gov-east-1(OSU)
Job ID: JIDd6d95004-fe1a-42d3-895d-684f357ef840
Snow Device Serial ID: 207117851234
Job type: IMPORT
Device type: Snowball Edge Storage Optimized
Storage type: S3
Job creation date: 2022-06-02 19:32:27.831 GMT
Job state: Completed
Customer address:

123 Any Street
Any Town, USA

Transfer details:

Transfer type	Total	Success	Failed
Objects	2,635	2,635	0
Bytes	32.2 TB	32.2 TB	0 B

Job state transition details:

The job was created on 2022-06-02 19:32:27.831 GMT
The snowball got allocated on 2022-06-06 19:10:43.670 GMT
The snowball was shipped on 2022-06-07 21:59:50.937 GMT
The snowball was at customer on 2022-06-08 14:04:45.856 GMT
The snowball was shipped to AWS on 2022-06-28 20:57:42.246 GMT
The snowball was at our sorting facility on 2022-06-29 14:06:20.737 GMT
The snowball was at AWS on 2022-06-30 23:12:45.017 GMT
The data transfer started on 2022-06-30 23:21:34.805 GMT
The data transfer was completed on +54473-09-10 22:23:46 GMT

Please review your job's status from the console.
For Snow job details, please see: https://docs.aws.amazon.com/snowball/

To get your job report and logs

1. Sign in to the AWS Management Console and open the AWS Snow Family Management Console.
2. Choose your job or job part from the table and expand the status pane.

Three options appear for getting your job report and logs: **Get job report**, **Download success log**, and **Download failure log**.

3. Choose the log you want to download.

The following list describes the possible values for the report:

- **Completed** – The transfer was completed successfully. You can find more detailed information in the success log.
- **Completed with errors** – Some or all of your data was not transferred. You can find more detailed information in the failure log.

Next: Using an AWS Snowball Edge Device

Large data migration with AWS Snow Family devices

Large data migration from on-premises locations requires careful planning, orchestration, and execution to ensure that your data is successfully migrated to AWS.

We recommend that you have a data migration strategy in place before starting your migration to avoid the potential for missed deadlines, exceeding budgets and migration failures. AWS Snow services helps you to place, order, and track your large data migration projects via the Snow Family Large Data Migration Manager (LDMM) feature in the AWS Snow Family Management Console.

The topics, Planning your large transfer and Calibrating a large transfer describe a manual data migration process. You can streamline the manual steps using the Snow Family LDMM migration plan.

Topics

Planning your large transfer

We recommend that you plan and calibrate large data transfers between the AWS Snowball Edge devices that you have on site and your servers using the guidelines in the following sections.

Topics

Step 1: Understand what you're moving to the cloud

Before you create your first job using the AWS Snow Family Management Console, ensure that you assess the volume of data you need to transfer, where it is currently stored, and the destination that you want to transfer it to. For data transfers that are a petabyte in scale or larger, this administrative housekeeping makes it much easier when your Snow Family devices arrive.

If you're migrating data into the AWS Cloud for the first time, we recommend that you design a cloud migration model. Cloud migration doesn't happen overnight. It requires a careful planning process to ensure that all systems work as expected.

When you're done with this step, you should know the total amount of data that you're going to move into the cloud.

Step 2: Calculate your target transfer rate

It's important to estimate how quickly you can transfer data to the Snow Family devices that are connected to each of your servers. This estimated speed in MB/Sec determines how fast you can transfer the data from your data source to Snowball Edge devices using your local network infrastructure.

> **Note**
>
> For large data transfers, we recommend using the Amazon S3 data transfer method. You must select this option when the you order devices in the AWS Snow Family Management Console.

To determine a baseline transfer rate, transfer a small subset of your data to the Snowball Edge device, or transfer a 10 GB sample file and observe the throughput.

While determining your target transfer speed, keep in mind that you can improve the throughput by tuning your environment, including network configuration, by changing the network speed, the size of the files being transferred, and the speed at which data can be read from your local servers. The Amazon S3 adapter copies data to Snow Family devices as quickly as your conditions allow.

Step 3: Determine how many Snow Family devices you need

Using the total amount of data that you plan to move into the cloud, the estimated transfer speed, and the number of days that you want to allow to move the data into AWS, determine how many

Snow Family devices you need for your large-scale data migration. Depending on the device type, Snowball Edge devices have approximately 39.5 TB, 80 TB, or 210 TB of usable storage space. For example, if you want to move 300 TB of data to AWS over 10 days and you have a transfer speed of 250 MB/s, you need 4 Snowball Edge devices. With less than 40 TB of data remaining to transfer, AWS Snowcone devices (with 14TB of usable space) will be recommended.

> **Note**
>
> The AWS Snow Family devices LDMM provides a wizard to estimate the number of AWS Snow Family devices that can be supported concurrently. For more information, see Creating a large data migration plan.

Step 4: Create your jobs

After you know how many Snow Family devices you need, you need to create an import job for each device. Creation of multiple jobs are simplified by the Snow Family LDMM. For more information, see Placing your next job order.

> **Note**
>
> You can place your next job order and automatically add it to your plan directly from the **Recommended job ordering** schedule. For more information, see Recommended job ordering schedule.

Step 5: Separate your data into transfer segments

As a best practice for large data transfers involving multiple jobs, we recommend that you logically split your data into a number of smaller, more manageable data sets. This allows you to transfer each partition at a time, or multiple partitions in parallel. When planning your partitions, make sure that the data for the partitions combined fit on the Snow Family devices for the job. For example, you can separate your transfer into partitions in any of the following ways:

- You can create 10 partitions of 8 TB each for a Snowball Edge.
- For large files, each file can be an individual partition up to the 5 TB size limit for objects in Amazon S3.

- Each partition can be a different size, and each individual partition can be made up of the same kind of data—for example, small files in one partition, compressed archives in another, large files in another partition, and so on. This approach can help you to determine your average transfer rate for different types of files.

> **Note**
>
> Metadata operations are performed for each file that's transferred. Regardless of a file's size, this overhead remains the same. Therefore, you get faster performance by compressing small files into a larger bundle, batching your files, or transferring larger individual files.

Creating data transfer segments can make it easier for you to quickly resolve transfer issues because trying to troubleshoot a large, heterogeneous transfer after the transfer runs for a day or more can be complex.

When you've finished planning your petabyte-scale data transfer, we recommend that you transfer a few segments onto the Snow Family device from your server to calibrate your speed and total transfer time.

Calibrating a large transfer

You can calibrate the transfer performance by transferring a representative set of your data partitions. Choose multiple partitions that you have defined and transfer them to a Snow Family device. Make a record of the transfer speed and total transfer time for each operation. If the calibration's results are less than the target transfer rate, you may be able to copy multiple parts of your data transfer at the same time. In this case, repeat the calibration with the additional partitions of your data set.

Continue adding parallel copy operations during calibration until you see diminishing returns in the sum of the transfer speed of all instances currently transferring data. End the last active instance and make a note of your new target transfer rate.

You can transfer data faster to Snow Family devices by transferring data in parallel using one of the following scenarios:

- Using multiple sessions of the S3 adapter on a workstation against a single Snow Family device.

- Using multiple sessions of the S3 adapter on multiple workstations against a single Snow Family device.
- Using multiple sessions of the S3 interface (using a single or multiple workstations) targeting multiple Snow Family devices.

When you complete these steps, you should know how quickly you can transfer data to a Snow Family device.

Creating a large data migration plan

The AWS Snow Family large data migration plan feature enables you to plan, track, monitor, and manage large data migrations from 500 TB to multiple petabytes using multiple Snow Family service products.

Use the large data migration plan feature to collect information about data migration goals, such as the size of the data to move to AWS and the number of Snow Family devices needed to migrate the data simultaneously. Use the plan to create a projected schedule for your data migration project and the recommended job ordering schedule to meet your goals.

> **Note**
>
> Currently, the data migration plan is available for import jobs larger than 500 TB.

Topics

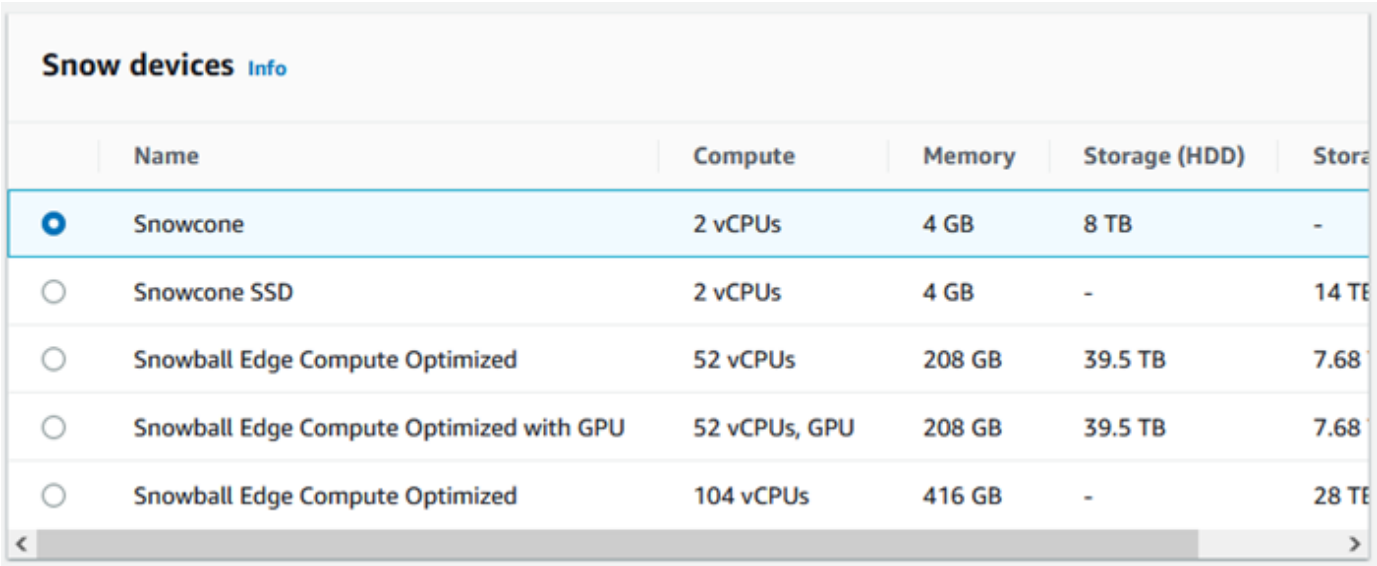
Step 1: Choose your migration details

> **Note**
>
> A large data migration plan is available for data migrations larger than 500 TB. Create job orders individually on Snow Family devices for your data transfer projects that are less

than 500 TB. For more information, see Creating a job to order a Snow Family device in this guide.

1. Sign in to the AWS Snow Family Management Console. If this is your first time using the AWS Snow Family Management Console in this AWS Region, you see the AWS Snow Family page. Otherwise, you see the list of existing jobs.
2. If this is your first data migration plan, choose **Create your large data migration plan** from the main page. Otherwise, choose **Large data migration plan**. Choose **Create data migration plan** to open the plan creation wizard.
3. In **Name your data migration plan**, provide a **Data migration plan name**. The plan name can have up to 64 characters. Valid characters are A-Z, a-z, 0-9, and . - (hyphen). A plan name must not start with **`aws:`**.
4. For **Total data to be migrated to AWS**, enter the amount of data that you want to migrate to AWS.
5. In **Snow devices**, choose a Snow Family device.

> **Note**
>
> Supported device options might vary based on device availability in certain AWS Regions.



6. For **Concurrent devices**, enter the number of Snow Family devices to which you can simultaneously copy data at your location. If you're not sure, skip to the next section for information about using the concurrent devices estimator wizard to determine this.
7. Choose **Next**.

Using the concurrent devices estimator wizard

The concurrent devices estimator wizard helps you to determine the number of concurrent devices that you can use during large data migrations.

Prerequisites:

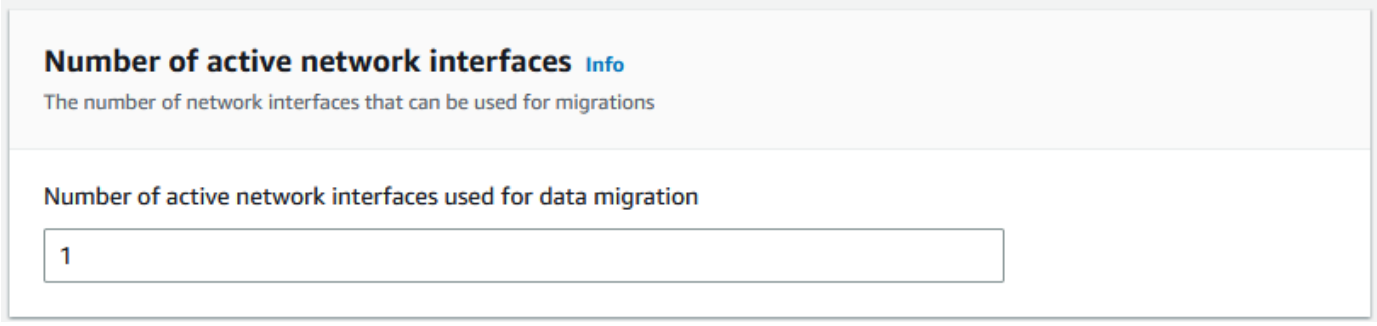
- You performed a proof of concept to test your data transfer methodology and measured performance with a Snow Family device in your environment.
- You know about the network and connection to back-end storage.

Step 1: Enter data source information

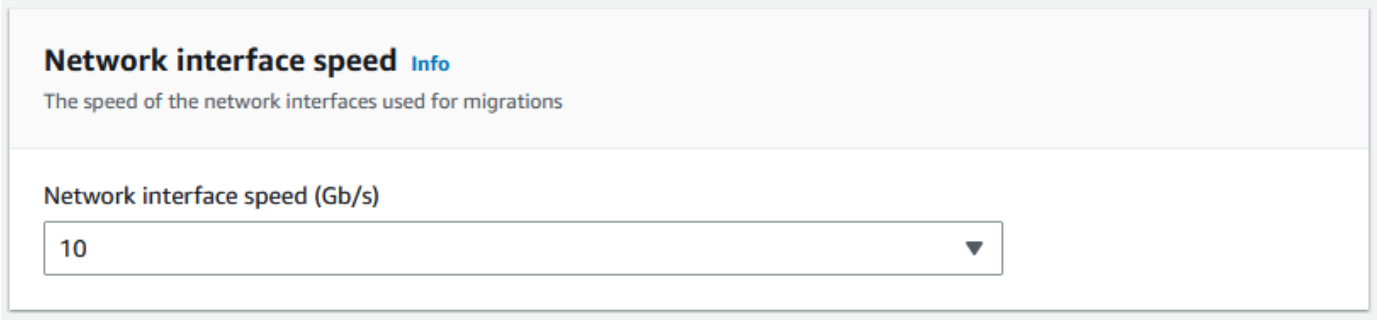
First, determine the maximum theoretical throughput for copying data from your storage source.

1. For **Total data to be migrated**, enter the amount of data that you plan to migrate.

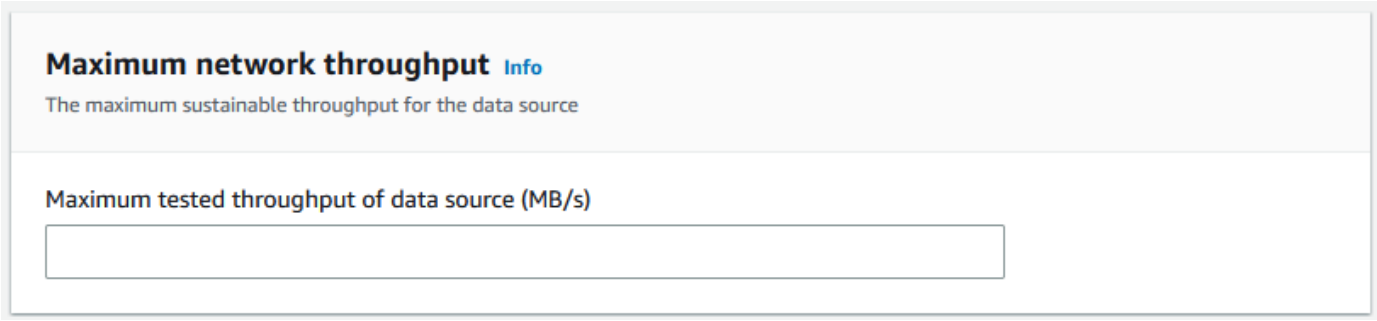
   For **Unit**, choose the unit of measurement (GB or TB) for the amount of data you plan to migrate.
2. For **Number of active network interfaces**, enter the number of active network interfaces that you have available for data migration from the storage source.



3. For **Network interface speed**, choose the speed of the network interface for the storage source. Network speeds are in Gb/s.



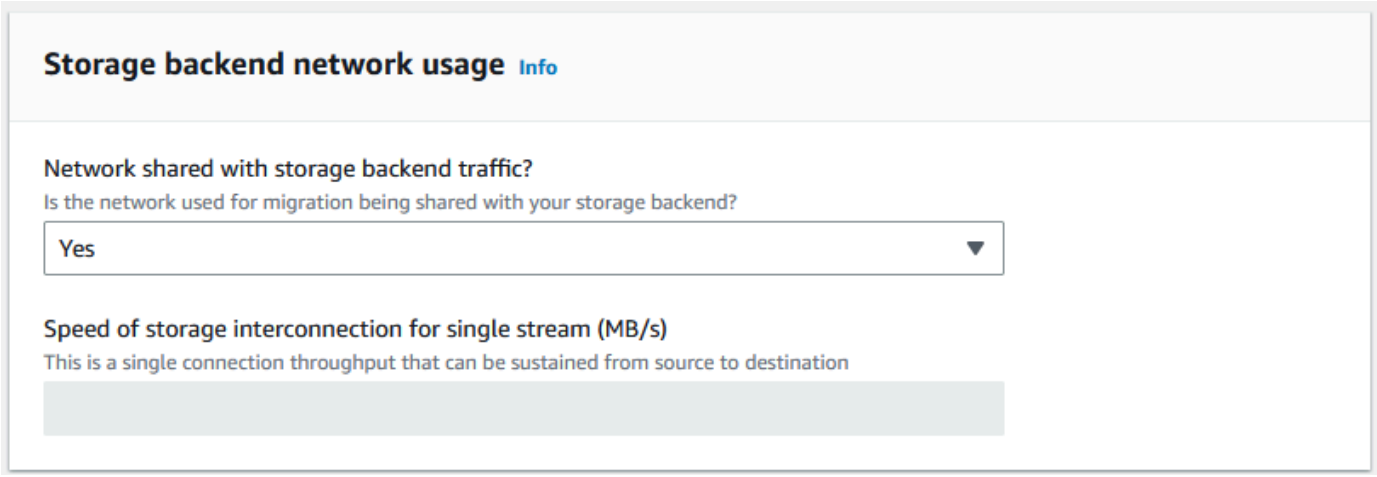
4. For **Maximum network throughput**, enter the maximum tested network throughput to your storage source that you determined during the proof of concept. Throughput is in MB/S.



5. For **Storage backend network usage**, indicate whether the storage source shares a network with the back-end storage.

   - Choose **Yes** if the network is not shared. You don't need to enter the speed of the storage interconnection for a single stream.
   - Choose **No** if the network is shared. Enter the speed of the storage interconnection for a single stream in MB/s.

   Based on your choice, the wizard updates the **Max migration throughput for the data source (MB/s)** value at the bottom of the page.



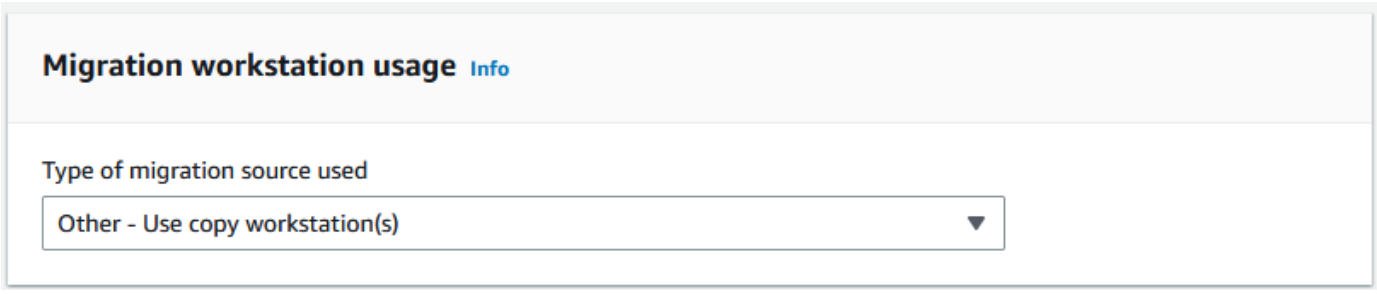
6. Choose **Next**.

Step 2: Input migration workstation parameters

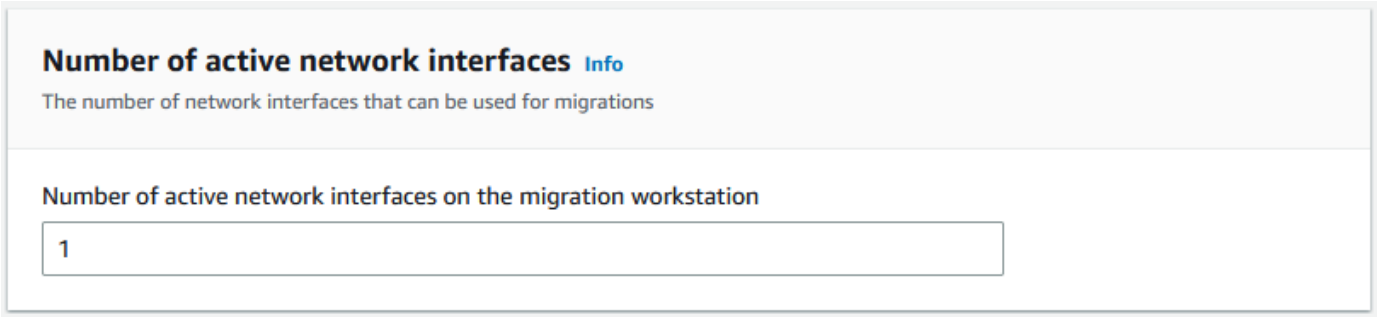
You can connect yourSnow Family devices directly to your storage source (a Microsoft Windows server, for example). You might choose instead to connect yourSnow Family devices to one or more workstations to copy data from the storage source.

1. For **Migration workstation usage**, indicate your workstation usage choice.

   - Choose **None - Use data source directly** to transfer data directly from a data source without using a workstation, and then choose **Next**.
   - Choose **Other - Use copy workstation(s)** to use one or more workstations for transferring data.



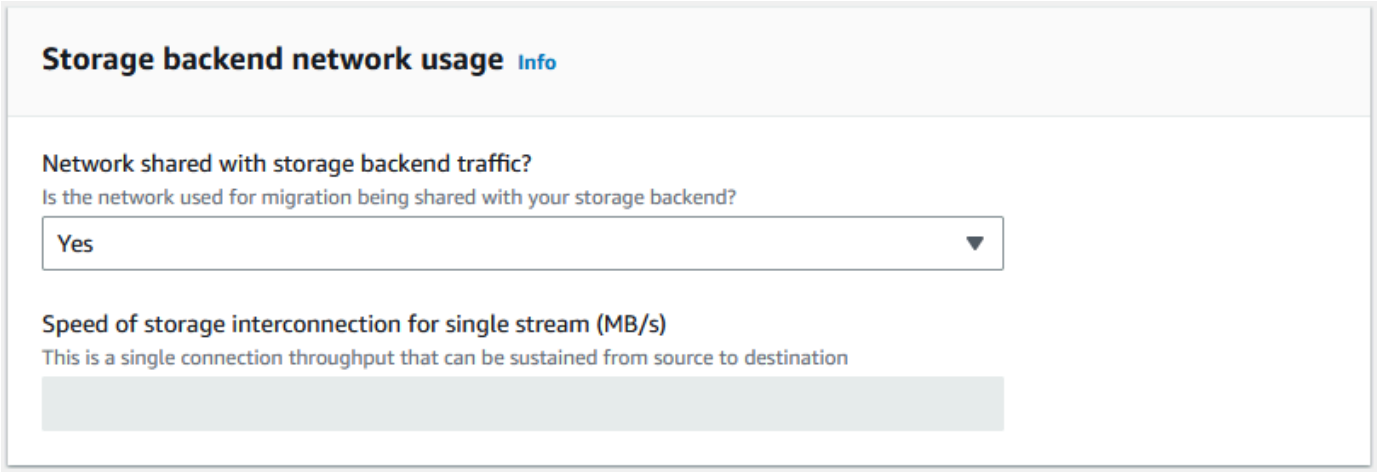
2. For **Number of active network interfaces**, enter the number of ports to use for data migration.



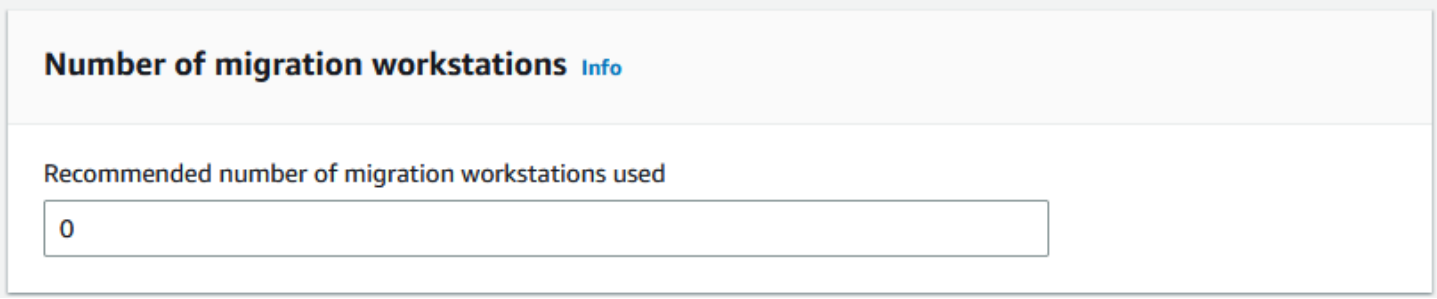
3. For **Network interface speed**, choose the speed in Gb/s of the network interfaces.

4. In **Storage backend network usage**, indicate whether the network that the workstations are on is shared with back-end storage.

   - Choose **Yes** if it's shared.
   - Choose **No** if it's not shared. Enter the speed of the storage interconnection for a single stream in MB/s.

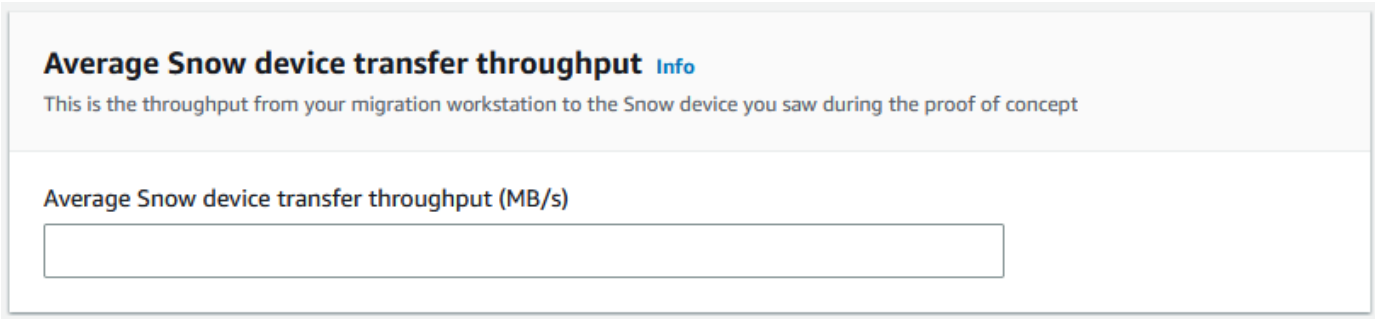


Based on your input, the wizard displays a recommendation in **Number of migration workstations**. You can manually change the number if you disagree with the recommendation. This number will appear in **Concurrent devices** in the large data migration plan.



Step 3: Input average transfer throughput of Snow Family devices

1. In the **Average Snow device transfer throughput** field, enter the transfer throughput in MB/s that you saw during your proof of concept.

   Based on your average throughput, the wizard updates the **Recommended number of concurrent Snow devices** and **Maximum number of concurrent devices** in the migration plan details.

2. Choose **Use this number** to continue and return to choosing your migration details. Choose **Next** and go the next step (Step 2: Choose your shipping, security, and notification preferences).

> **Note**
>
> You can use up to 5 concurrent Snow devices.

Step 2: Choose your shipping, security, and notification preferences

1. In the **Shipping Address** section, choose an existing address or create a new one.

   - > **Note**
     >
     > The country in the address must match the destination country for the device, and must be valid for that country.

2. In **Choose service access type**, do one of the following:

   - Allow Snow Family to create a new service-linked role for you with all of the necessary permissions to publish CloudWatch metrics and Amazon SNS notifications for your Snow Family jobs.
   - Add an existing service role that has the necessary permissions. For an example of how to set up this role, see Example 4: Expected Role Permissions and Trust Policy.

3. For **Send notifications**, choose whether to send notifications. Note that if you choose **Do not send notification about data migration plans**, you won't receive notifications from this plan, but you will still receive job notifications.

4. For **Set notifications**,

   - choose **Use an existing SNS topic**
   - or **Create a new SNS topic**.

Step 3: Review and create your plan

1. Review your information in **Plan details** and **Shipping, security, and notification preferences**, and edit if necessary.
2. Choose **Create data migration plan** to create the plan.

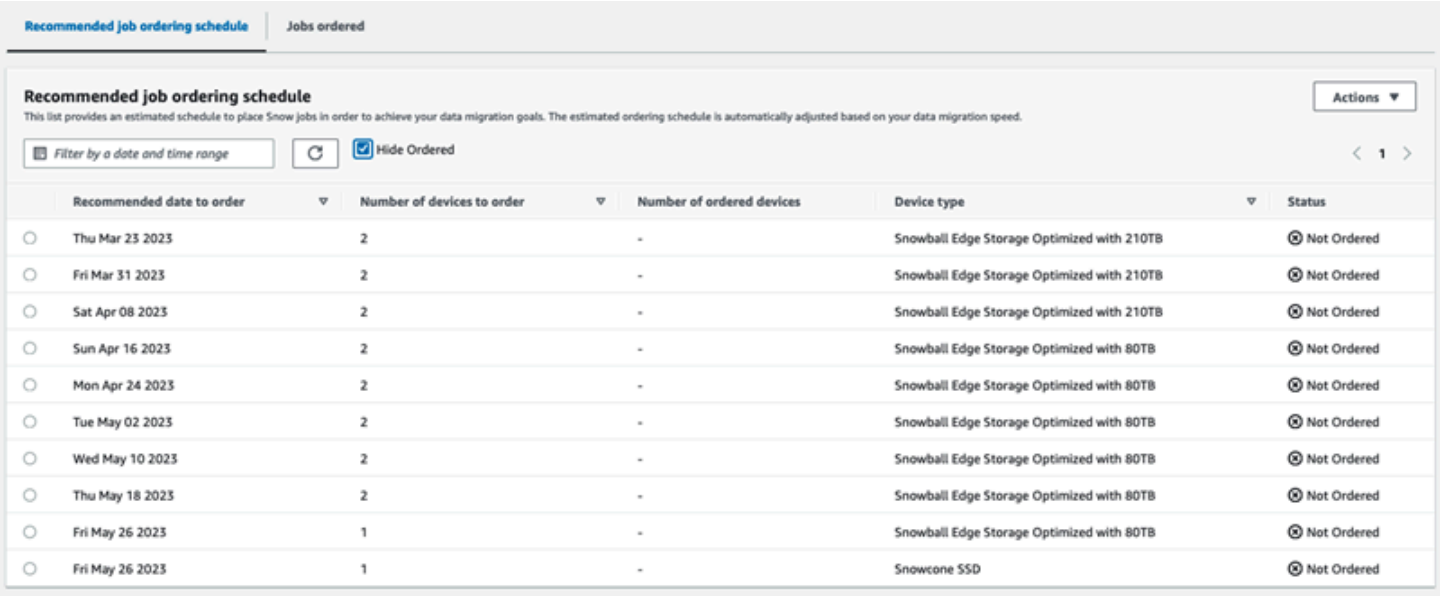
Using the large data migration plan

After you create your large data migration plan, you can use the resulting schedule and dashboard to guide you through the rest of the migration process.

Recommended job ordering schedule

After you create an AWS Snow Family devices large migration plan, you can use the recommended job ordering schedule to create new jobs.

> **Note**
>
> Manual updates that you make to the data size or number of concurrent devices cause the schedule to adjust. The schedule automatically adjusts if a job has not been ordered by the recommended order date or has been ordered before the recommended order date. If a job is returned before the recommended order date, the schedule automatically adjusts.



Placing your next job order

To place you next order, instead of manually creating a job and then adding it to your plan, you have the option to either clone a previously ordered job or create a pre-populated one.

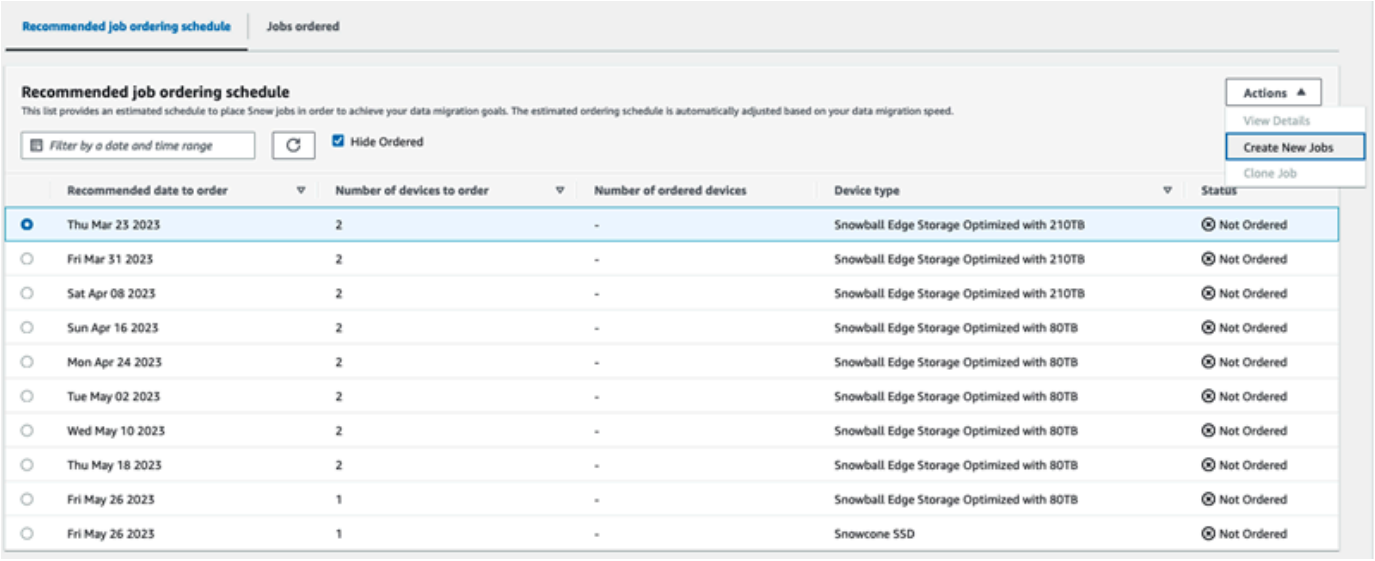
To clone a job:

1. Choose the next order (the first recommendation with a **Not Ordered** status) from the **Recommended job ordering** schedule, then choose **Clone Job** from the **Actions** menu. The **Clone Job** window appears.
2. In the **Clone Job** window, in the **Jobs ordered** section, choose the job to clone.

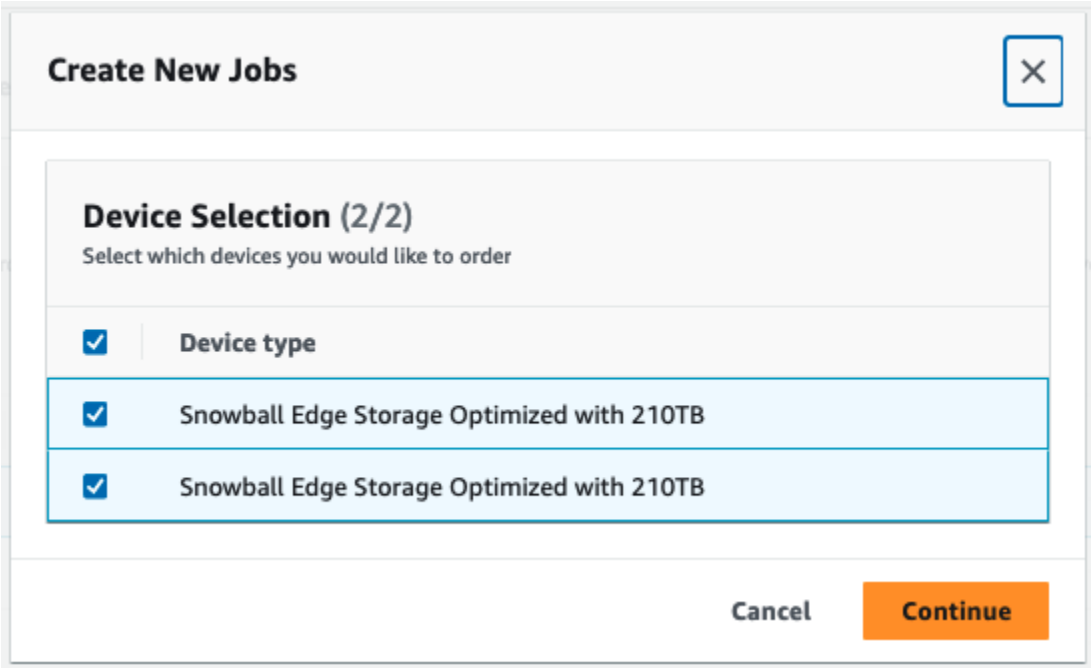
3. In the **New jobs details** section, choose the devices you want to order. For each device chosen, the **Job name** will automatically populate based on the chosen job. You can overwrite the job name.
4. Choose **Confirm** to place the job order for the chosen devices. The system clones the job for each device.

To create new jobs:

1. Choose the next order (the first recommendation with a **Not Ordered** status) from the **Recommended job ordering** schedule, then choose **Create New Jobs** from the **Actions** menu. The **Create new jobs** window appears.



2. In the **Device Selection** section, choose the devices you want to order. Choose **Continue**.



3. The **Create new** page appears. Most parameters, such as the job type, shipping address, and the device type are set based on the plan. The system creates the job for each device.

You can see whether the job or jobs were successfully created or not. Successfully created jobs are automatically added to the plan.

Jobs ordered list

Each plan displays a job ordered list. This is empty at first. When you start to order jobs, you can add jobs to your plan by selecting **Add job** from the **Actions** menu. Jobs that you add here are tracked on the monitoring dashboard.

Similarly, you may remove the job from the job ordered list by selecting **Remove job** from the **Actions** menu.

We recommend using the job ordering schedule provided in the plan for a smooth data migration.

Monitoring dashboard

After you add jobs to your plan, you can see metrics on the dashboard as the jobs return to AWS for ingestion. These metrics can help you to track your progress:

- **Data migrated to AWS** – The amount of data that's been migrated to AWS so far..
- **Average data migrated per job** – The average amount of data per job in terabytes.

- **Total Snow Jobs** – The number of Snowball Edge jobs ordered compared to the remaining jobs to be ordered.
- **Average duration for a migration job** – The average duration of a job in days.
- **Snow Job Status** – The number of jobs in each status.

Using AWS OpsHub for Snow Family to Manage Devices

The Snow Family devices now offer a user-friendly tool, AWS OpsHub for Snow Family, that you can use to manage your devices and local AWS services. You use AWS OpsHub on a client computer to perform tasks such as unlocking and configuring single or clustered devices, transferring files, and launching and managing instances running on Snow Family devices. You can use AWS OpsHub to manage both the Storage Optimized and Compute Optimized Snow device types. The AWS OpsHub application is available at no additional cost to you.

AWS OpsHub takes all the existing operations available in the Snowball API and presents them as a graphical user interface. This interface helps you quickly migrate data to the AWS Cloud and deploy edge computing applications on Snow Family devices.

AWS OpsHub provides a unified view of the AWS services that are running on Snow Family devices and automates operational tasks through AWS Systems Manager. With AWS OpsHub, users with different levels of technical expertise can manage a large number of Snow Family devices. With a few clicks, you can unlock devices, transfer files, manage Amazon EC2-compatible instances, and monitor device metrics.

When your Snow device arrives at your site, you download, install, and launch the AWS OpsHub application on a client machine, such as a laptop. After installation, you can unlock the device and start managing it and using supported AWS services locally. AWS OpsHub provides a dashboard that summarizes key metrics such as storage capacity and active instances on your device. It also provides a selection of AWS services that are supported on the Snow Family devices. Within minutes, you can begin transferring files to the device.

Topics

Download AWS OpsHub for Snow Family devices

To download AWS OpsHub

1. Navigate to the AWS Snowball resources website.

2. In the **AWS OpsHub** section, choose **Download** for your operating system, and follow the installation steps.

Unlocking a device

When your device arrives at your site, the first step is to connect and unlock it. AWS OpsHub lets you sign in, unlock, and manage devices using the following methods:

- **Locally** – To sign in to a device locally, you must power on the device and connect it to your local network. Then provide an unlock code and a manifest file.
- **Remotely** – To sign in to a device remotely, you must power on the device and make sure that it can connect to *`device-order-region`*`.amazonaws.com` through your network. Then provide the AWS Identity and Access Management (IAM) credentials (access key and secret key) for the AWS account that is linked to your device.

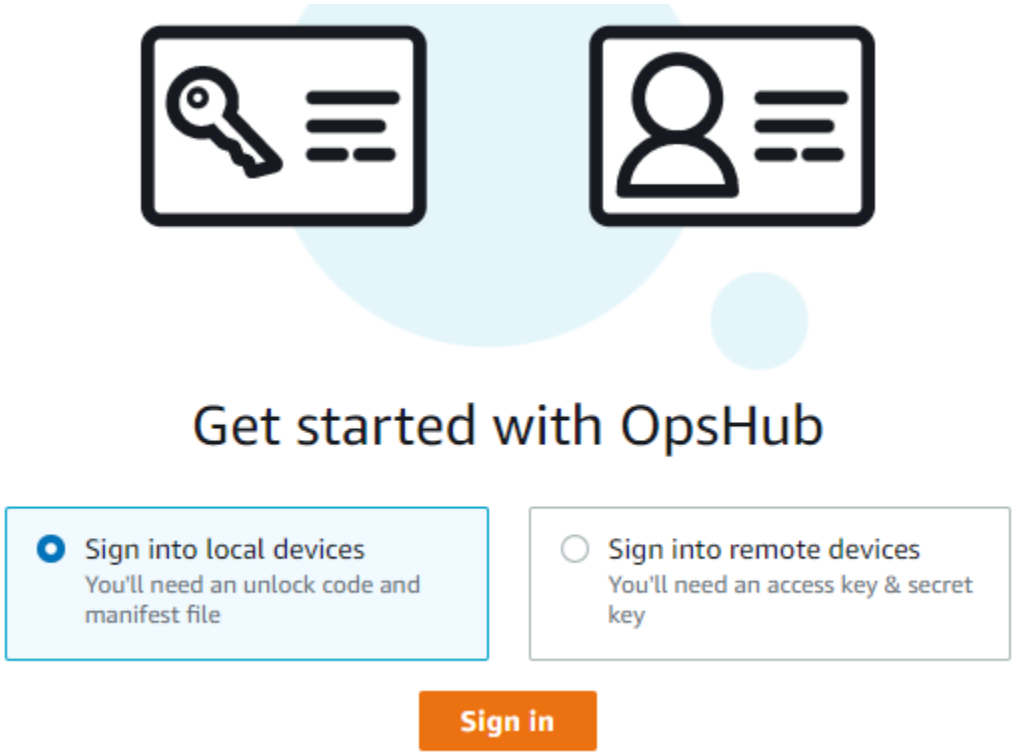
  For information on enabling remote management and creating an associated account, see Activating Snow Device Management.

Topics

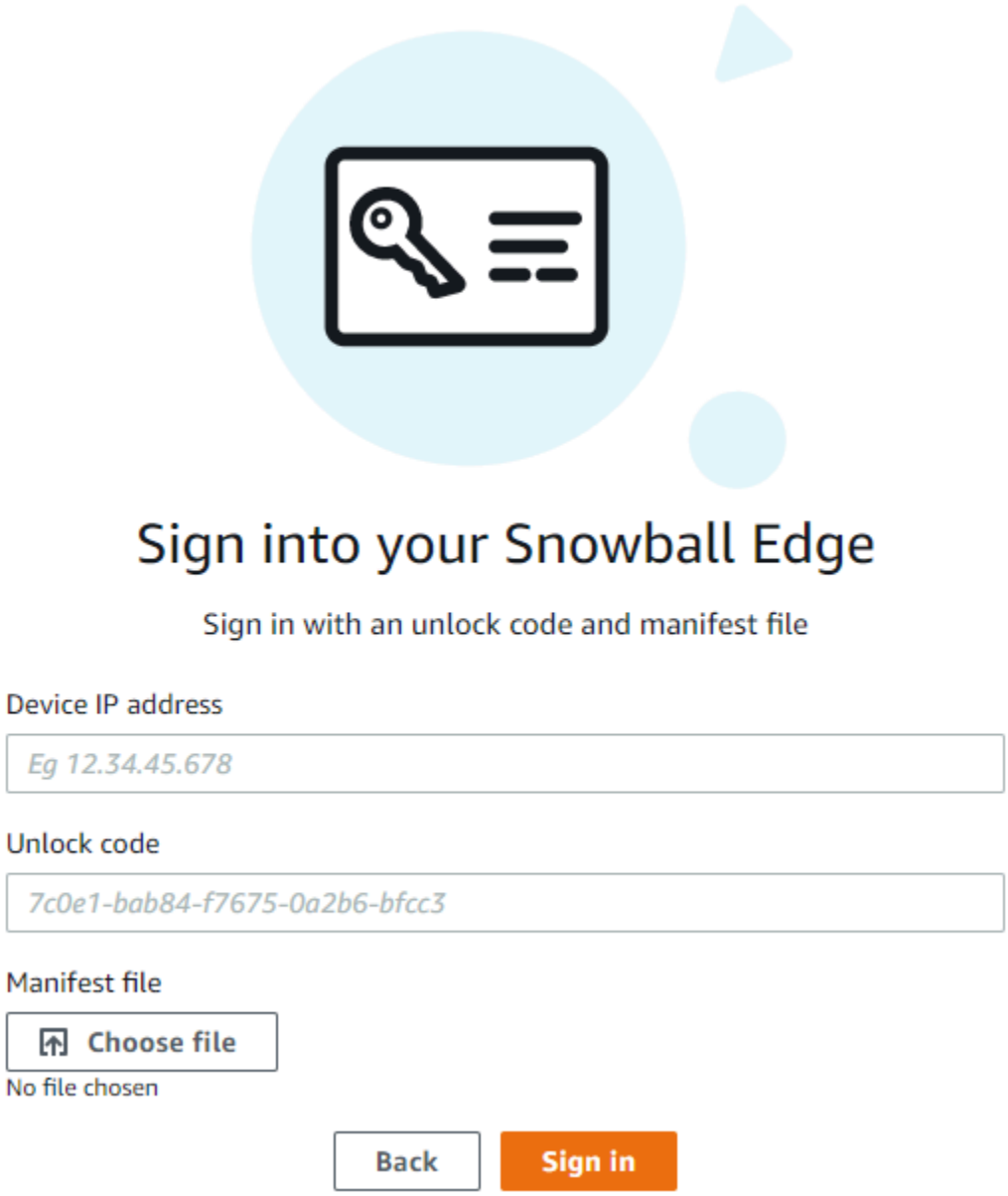
Unlocking a device locally

To connect and unlock your device locally

1. Open the flap on your device, locate the power cord, and connect it to a power source.
2. Connect the device to your network using a network cable (typically an Ethernet RJ45 cable), then open the front panel and power on the device.
3. Open the AWS OpsHub application. If you are a first-time user, you are prompted to choose a language. Then choose **Next**.
4. On the **Get started with OpsHub** page, choose **Sign in to local devices**, and then choose **Sign in**.



5. On the **Sign in to local devices** page, choose your Snow Family devices type, and then choose **Sign in**.
6. On the **Sign in** page, enter the **Device IP address** and **Unlock code**. To select the device manifest, choose **Choose file**, and then choose **Sign in**.



7. (Optional) Save your device's credentials as a *profile*. Name the profile and choose **Save profile name**. For more information about profiles, see Managing profiles.
8. On the **Local devices** tab, choose a device to see its details, such as the network interfaces and AWS services that are running on the device. You can also see details for clusters from this tab, or manage your devices just as you do with the AWS Command Line Interface (AWS CLI). For more information, see Managing AWS services on your device.

   For devices that have AWS Snow Device Management installed, you can choose **Enable remote management** to turn on the feature. For more information, see Using AWS Snow Device Management to Manage Devices.

Unlocking a device remotely

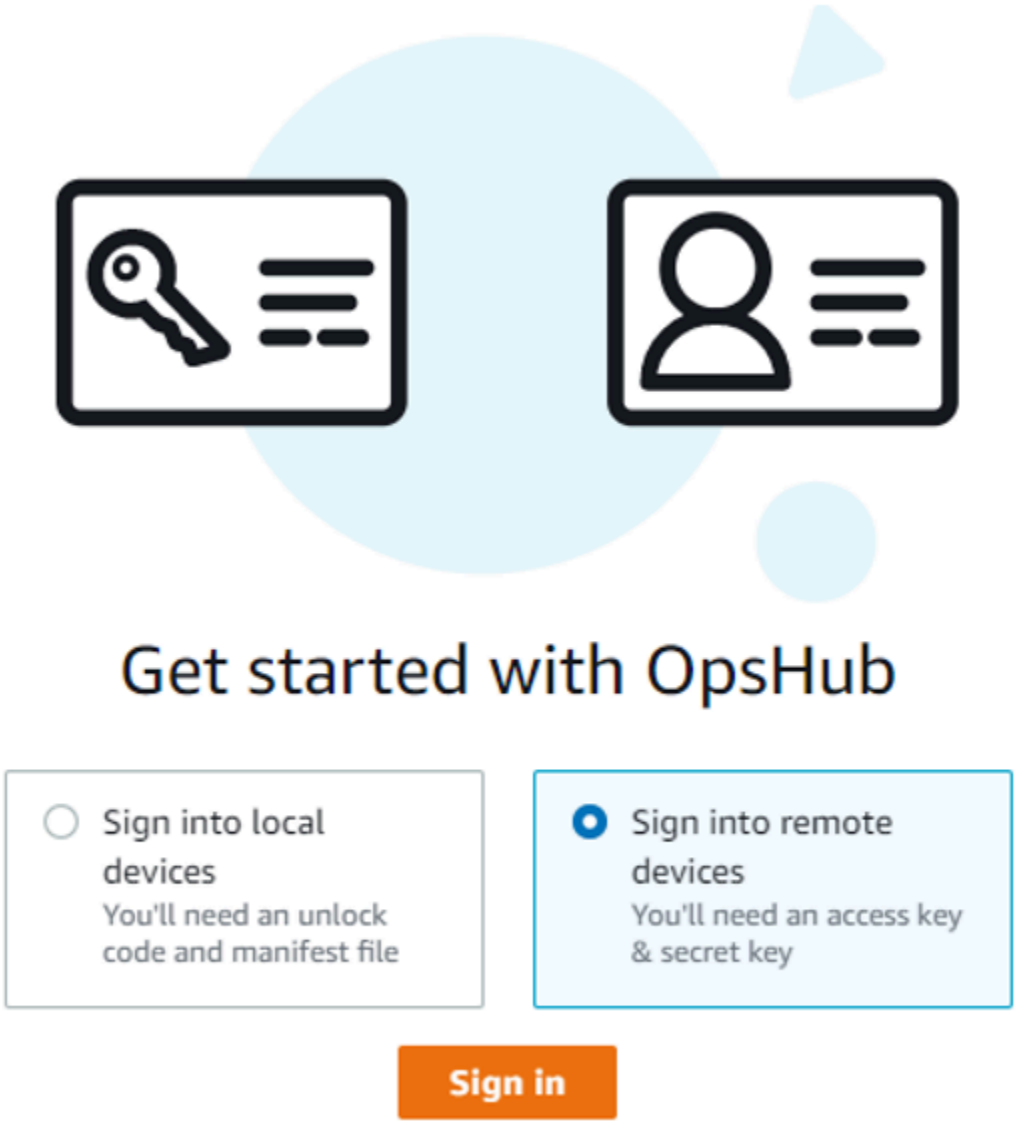
To unlock a Snow Family device not

To connect and unlock your device remotely

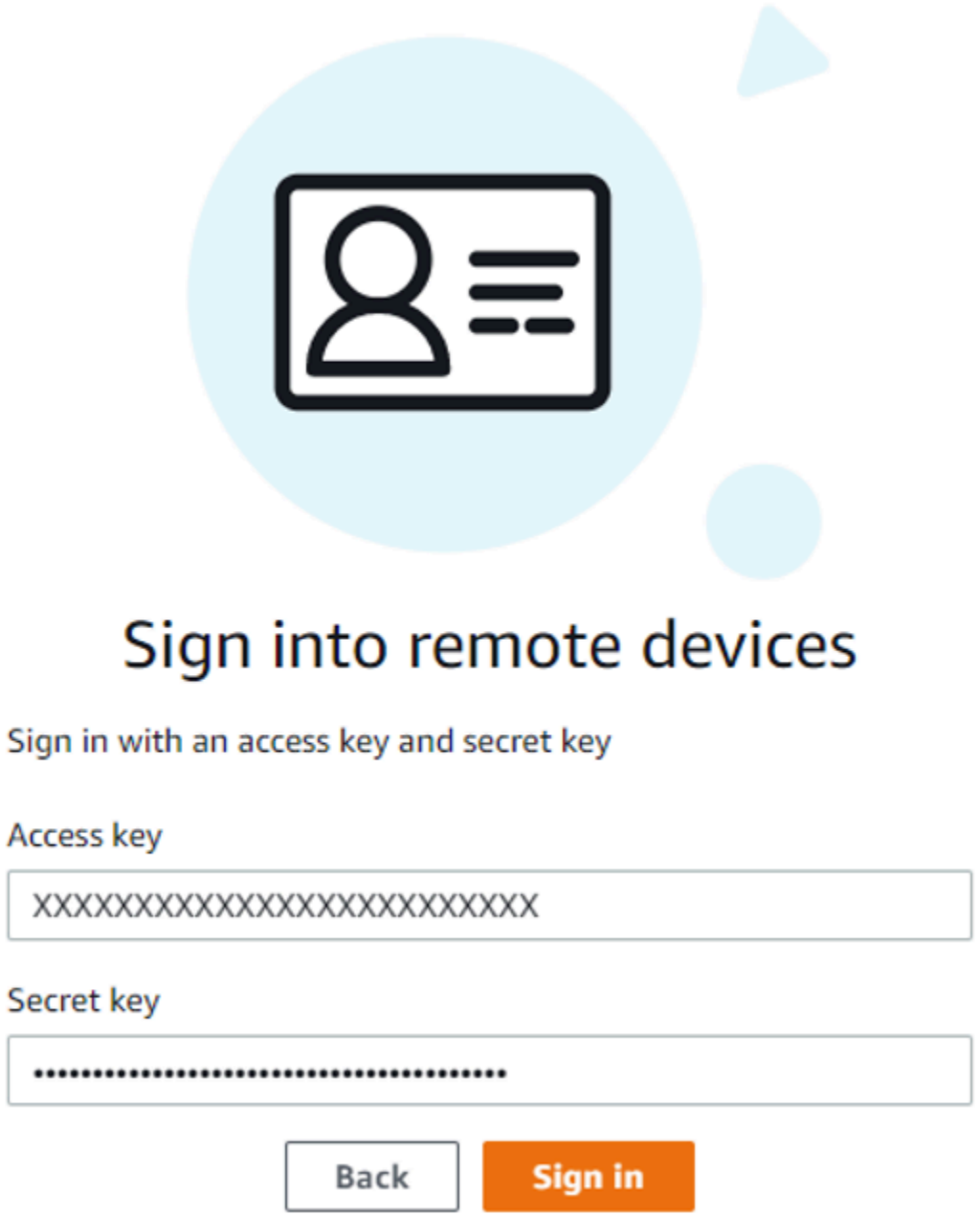
1. Open the flap on your device, locate the power cord, and connect it to a power source.
2. Connect the device to your network using an Ethernet cable (typically an RJ45 cable), then open the front panel and power on the device.

   > **Note**
   >
   > To be unlocked remotely, your device must be able to connect to *`device-order-region`*`.amazonaws.com`.

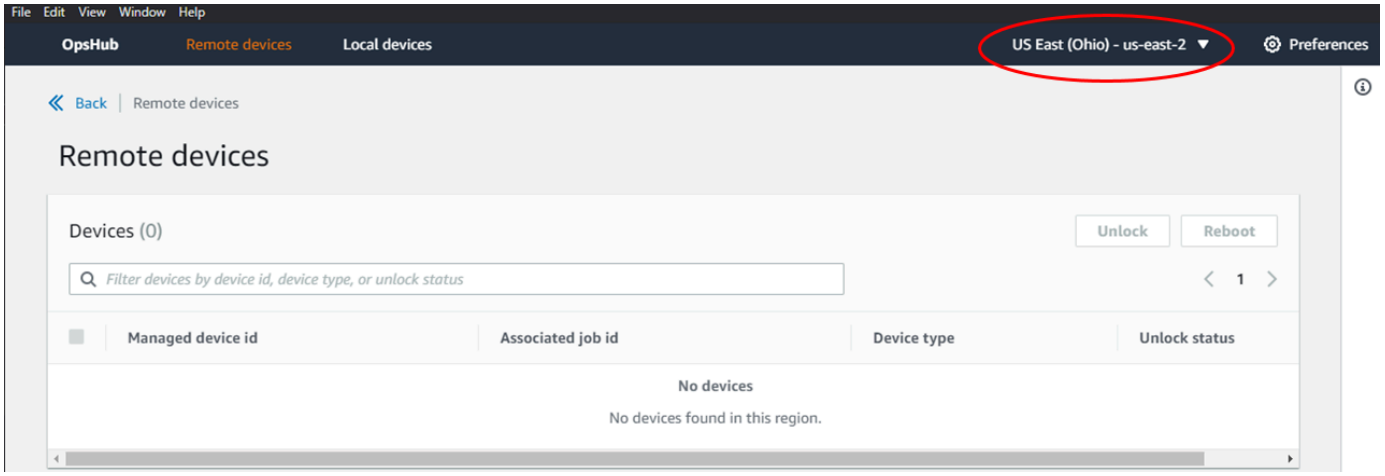
3. Open the AWS OpsHub application. If you are a first-time user, you are prompted to choose a language. Then choose **Next**.
4. On the **Get started with OpsHub** page, choose **Sign into remote devices**, and then choose **Sign in**.



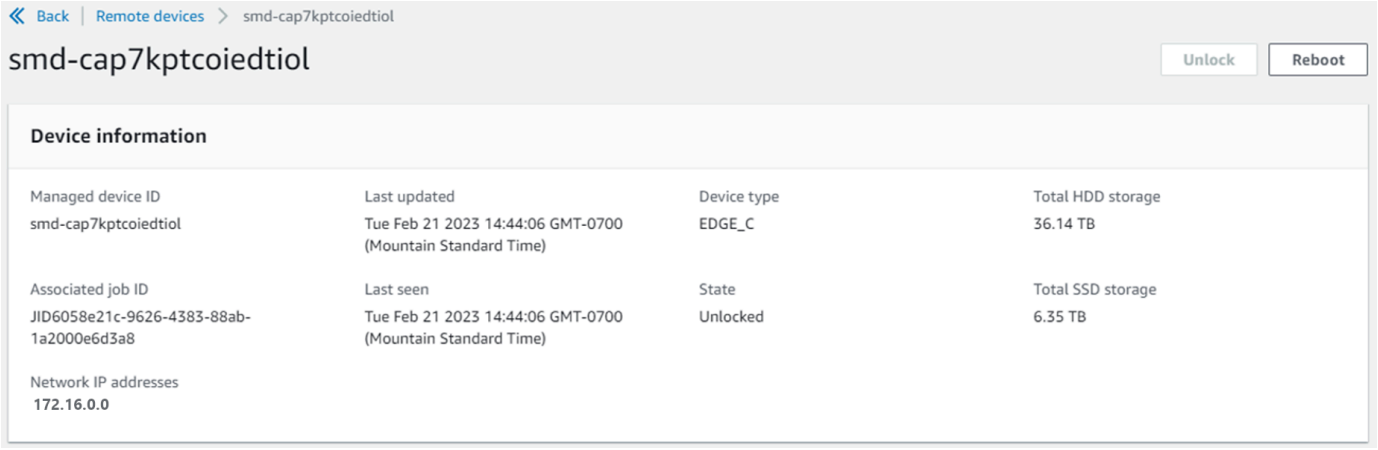
5. On the **Sign in to remote devices** page, enter the AWS Identity and Access Management (IAM) credentials (access key and secret key) for the AWS account that is linked to your device, and then choose **Sign in**.



6. At the top of the **Remote devices** tab, choose the region of the Snow device to unlock remotely.



7. On the **Remote devices** tab, choose your device to see its details, such as its state and network interfaces. Then choose **Unlock** to unlock the device.



From the remote device's details page, you can also reboot your devices and manage them just as you do with the AWS Command Line Interface (AWS CLI). To view remote devices in different AWS Regions, choose the current Region on the navigation bar, and then choose the Region that you want to view. For more information, see Managing AWS services on your device.

Verifying the PGP signature of AWS OpsHub (optional)

The AWS OpsHub application installer package for the Linux operating system are cryptographically signed. You can use a public key to verify that the installer package is original and unmodified. If the files are damaged or altered, the verification fails. You can verify the signature of the installer package using GNU Privacy Guard (GPG). This verification is optional. If you choose to verify the signature of the application, you can do it at any time.

You can download the SIGNATURE file for the Linux operating system installer from AWS Snowcone Resources or Snowball Edge Resources.

To verify the AWS OpsHub install package on for the Linux operating system

1. Copy the following public key, save it to a file, and name the file. For example, `opshub-public-key.pgp`.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
xsFNBF/hGf8BEAC9HCDV8uljDX02Jxspi6kmPu4xqf4ZZLQsSqJcHU61oL/c
/zAN+mUqJT9aJ1rr0QFGVD1bMogecUPflTWlDkEEpG8ZbX5P8vR+EElO/rW/
WtqizSudy6qy59ZRK+YVSDx7DZyuJmIO7j00UADCL+95ZQN9vqwHNjBHsgfQ
l/1Tqhy81ozTZXcI/+u+99YLaugJIP6ZYIeDfpxnghqyVtaappBFTAyfG67Y
N/5mea1VqJzd8liFpIFQnl+X7U2x6emDbM01yJWV3aMmPwhtQ7iBdt5a4x82
EF5bZJ8HSRMvANDILD/9VTN8VfUQGKFjFY2GdX9ERwvfTb47bbv9Z28Vl284
4lw2w1Bl007FoO2v/Y0ukrN3VHCpmJQS1IiqZbYRa0DVK6UR5QNvUlj5fwWs
4qW9UDPhT/HDuaMrMFCejEn/7wvRUrGVtzCT9F56Al/dwRSxBejQQEb1AC8j
uuyi7gJaPdyNntROEFTD7iO2L6X2jB4YLfvGxP7Xeq1Y37t8NKF8CYTpOry/
Wvw0iKZFbo4AkiI0aLyBCk9HBXhUKa9x06gOnhh1UFQrPGrk60RPQKqL76HA
E2ewzGDa90wlRBUAt2nRQpyNYjoASBvz/cAr3e0nuWsIzopZIenrxI5ffcjY
f6UWA/OK3ITHtYHewVhseDyEqTQ4MUIWQS4NAwARAQABzTlBV1MgT3BzSHVi
IGZvciBTbm93IEZhbWlseSA8YXdzLW9wc2h1Yi1zaWduZXJAYW1hem9uLmNv
bT7CwY0EEAEIACAFAl/hGf8GCwkHCAMCBBUICgIEFgIBAAIZAQIbAwIeAQAh
CRAhgc9adPNF8RYhBDcvpelIaY930bOvqiGBz1p080XxGbcP+gPZX7LzKc1Y
w9CT3UHgkAIawOSXYktujzoYVxAz8/j3jEkCY0dKnfyqvWZDiJAXnzmxWWbg
cxg1g0GXNXCM4lAd68CmbAOLoLTaWSQX30ZbswzhbtX2ADAlopV8RLBik7fm
bS9FyuubDRhfYRQq0fpjUGXFiEgwg6aMFxsrGLlv4QD7t+6ftFIe/mxLbjR4
iMgtr8FIPXbgn05YYY/LeF4NIgX4iLEqRbAnfWjPzqQ1spFWAotIzDmZqby+
WdWThrH4K1rwtYM8sDhqRnMnqJrGFZzk7aDhVPwF+FOVMmPeEN5JRazEeUrl
VZaSw6mu0n4FMGSXuwGgdvmkqnMe6I5/xLdU4IOPNhp0UmakDWOq/a1dREDE
ZLMQDMINphmeQno4inGmwbRo63gitD4ZNR5sWwfuwty25lo8Ekv7jkkp3mSv
pdxn5tptttnPaSPcSIX/4EDl19Tu0i7aup+v30t7eikYDSZG6g9+jHB3Va9e
/VWShFSgy8Jm2+qq/ujUQDAGTCfSuY9jg1ITsog6ayEZa/2upDJ1m+4OHK4p
8DrEzP/3jTahT8q5ofFWSRDL17d3lTSU+JBmPE3mz311FNXgiO8w+taY320z
+irHtb3iSiiukbjS8s0maVgzszRqS9mhaEn4LL0zoqrUicmXgTyFB7n2LuYv
O7vxMO5xxhGQwsF2BBABCAAJBQJf4RoCAhsDACEJEBFZvzT/tDi5FiEEi+O9
V+UAYN9Gnw36EVm/NP+0OLnnEQ/+J4C0Mn8j0AebXrwBiFs83sQo2q+WHL1S
MRc1g5gRFDXs6h1Gv+TGXRen7j1oeaddWvgOtUBxqmCOjr+8AKH0OtiBWSuO
lsS8JU5rindEsKUrKTwcG2wyZFoe1zlE8xPkLRSRN5ZbbgKsTz16l1HgCCId
Do+WJdDkWGWxmtDvzjM32EI/PVBd108ga9aPwXdhLwOdKAjZ4JrJXLUQJjRI
IVDSyMObEHOUM6a/+mWNZazNfo0LsGWqGVa6Xn5WJWlwR1S78vPNfO3BQYuO
YRjaVQR+kPtB9aSAZNi5sWfk6NrRNd1Q78d067uhhejsjRt7Mja2fEL4Kb1X
nK4U/ps7Xl03o/VjblneZOhJK6kAKU172tnPJTJ31JbOxX73wsMWDYZRZVcK
```

```
9X9+GFrpwhKHWKKPjpMOt/FRxNepvqRl72TkgBPqGH2TMOFdB1f/uQprvqge
PBbS0JrmBIH9/anIqgtMdtcNQB/0erLdCDqI5afOuD1OLcLwdJwG9/bSrfwT
TVEE3WbXmJ8pZgMzlHUiZE6V2DSadV/YItk50IOjjrOVHOHvlFMwGCEAIFzf
9P/pNi8hpEmlRphRiOVVcdQ30bH0M0gPHu5V9flIhyCL1zU3LjYTHkq0yJD5
YDA1xO1MYq3DcSM513OVBbLmuVS2GpcsTCYqlgQA6h/zzMwz+/7OwU0EX+EZ
/wEQAOAY8ULmcJIQWIr14V0jylpJeD3qwj7wd+QsBzJ+mOpOB/3ZFAhQiNOl
9yCDlHeiZeAmWYX9OIXrNiIdcHy+WTAp4G+NaMpqE52qhbDjz+IbvLpl1yDH
bYEHPjnTHXEy2lbvKAJOKkw/2RcQOi4dodGnq5icyYj+9gcuHvnVwbrQ96Ia
0D7c+b5T+bzFqk90nIcztrMRuhDLJnJpi7OjpvQwfq/TkkZA+mzupxfSkq/Y
N9qXNEToT/VI2gn/LS0X4Ar1l2KxBjzNEsQkwGSiWSYtMA5J+Tj5ED0uZ/qe
omNblAlD4bm7Na8NAoLxCtAiDq/f3To9Xb18lHsndOmfLCb/BVgP4edQKTIi
C/OZHy9QJlfmN0aq7JVLQAuvQNEL88RKW6YZBqkPd3P6zdc7sWDLTMXMOd3I
e6NUvU7pW0E9NyRfUF+oT4s9wAJhAodinAi8Zi9rEfhK1VCJ76j7bcQqYZe0
jXD3IJ7T+X2XA8M/BmypwMW0Soljzhwh044RAasr/fAzpKNPB318JwcQunIz
u2N3CeJ+zrsomjcPxzehwsSVq1lzaL2ureJBLOKkBgYxUJYXpbS01ax1TsFG
09ldANOs9Ej8CND37GsNnuygjOgWXbX6MNgbvPs3H3zi/AbMunQ1VBlw07JX
zdM1hBQZh6w+NeiEsK1T6wHi7IhxABEBAAHCwXYEGAEIAAkFAl/hGf8CGwwA
IQkQIYHPWnTzRfEWIQQ3L6XpSGmPd9Gzr6ohgc9adPNF8TMBD/9TbU/+PVbF
ywKvwi3GLOlpY7BXn8lQaHyunMGuavmO8OfaRROynkH0ZqLHCp6bIajFOfvF
b7c0Jamzx8Hg+SIdl6yRpRY+fA4RQ6PNnnmT93ZgWW3EbjPyJGlm0/rt03SR
+0yn4/ldlg2KfBX4pqMoPCMKUdWxGrmDETXsGihwZ0gmCZqXe8lK122PYkSN
JQQ+LlfjKvCaxfPKEjXYTbIbfyyhCR6NzAOVZxCrzSz2xDrYWp/V002Klxda
0ix6r2aEHf+xYEUhOaBt8OHY5nXTuRReCVU789MUVtCMqD2u6amdo4BR0kWA
QNg4yavKwV+LVtyYh2Iju9VSyv4xL1Q4xKHvcAUrSH73bHG7b7jkUJckD0f4
twhjJk/Lfwe6RdnVo2WoeTvE93w+NAq2FXmvbiG7eltl0XfQecvQU3QNbRvH
U8B96W0w8UXJdvTKg4f0NbjSw7iJ3x5naixQ+rA8hLV8xOgn2LX6wvxT/SEu
mn20KX+fPtJELK7v/NheFLX1jsKLXYo4jHrkfIXNsNUhg/x2E71kAjbeT3s+
t9kCtxt2iXDDZvpIbmGO4QkvLFvoROaSmN6+8fupe3e+e2yN0e6xGTuE60gX
I2+X1p1g9IduDYTpoI2OXleHyyMqGEeIb4gOiiSloTp5oi3EuAYRGflXuqAT
VA19bKnpkBsJ0A==
=tD2T
-----END PGP PUBLIC KEY BLOCK-----
```

2. Import the public key into your keyring, and note the returned key value.

   GPG

   ```
   gpg --import opshub-public-key.pgp
   ```

   Example output

   ```
   gpg: key 1655BBDE2B770256: public key "AWS OpsHub for Snow Family <aws-opshub-
   signer@amazon.com>" imported
   ```

```
gpg: Total number processed: 1
gpg:               imported: 1
```

3. Verify the fingerprint. Be sure to replace *key-value* with the value from the preceding step. We recommend that you use GPG to verify the fingerprint.

   ```
   gpg --fingerprint key-value
   ```

   This command returns output similar to the following.

   ```
   pub   rsa4096 2020-12-21 [SC]
         372F A5E9 4869 8F77 D1B3  AFAA 2181 CF5A 74F3 45F1
   uid           [ unknown] AWS OpsHub for Snow Family <aws-opshub-signer@amazon.com>
   sub   rsa4096 2020-12-21 [E]
   ```

   The fingerprint should match the following:

   ```
   372F A5E9 4869 8F77 D1B3 AFAA 2181 CF5A 74F3 45F1
   ```

   If the fingerprint doesn't match, don't install the AWS OpsHub application. Contact AWS Support.

4. Verify the installer package, and download the SIGNATURE file according to your instance's architecture and operating system if you haven't already done so.

5. Verify the installer package signature. Be sure to replace *signature-filename* and *OpsHub-download-filename* with the values that you specified when downloading the SIGNATURE file and AWS OpsHub application.

   GPG

   ```
   gpg --verify signature-filename OpsHub-download-filename
   ```

   This command returns output similar to the following.

   GPG

   ```
   gpg: Signature made Mon Dec 21 13:44:47 2020 PST
   gpg:                using RSA key 1655BBDE2B770256
   gpg: Good signature from "AWS OpsHub for Snow Family <aws-opshub-
   signer@amazon.com>" [unknown]
   ```

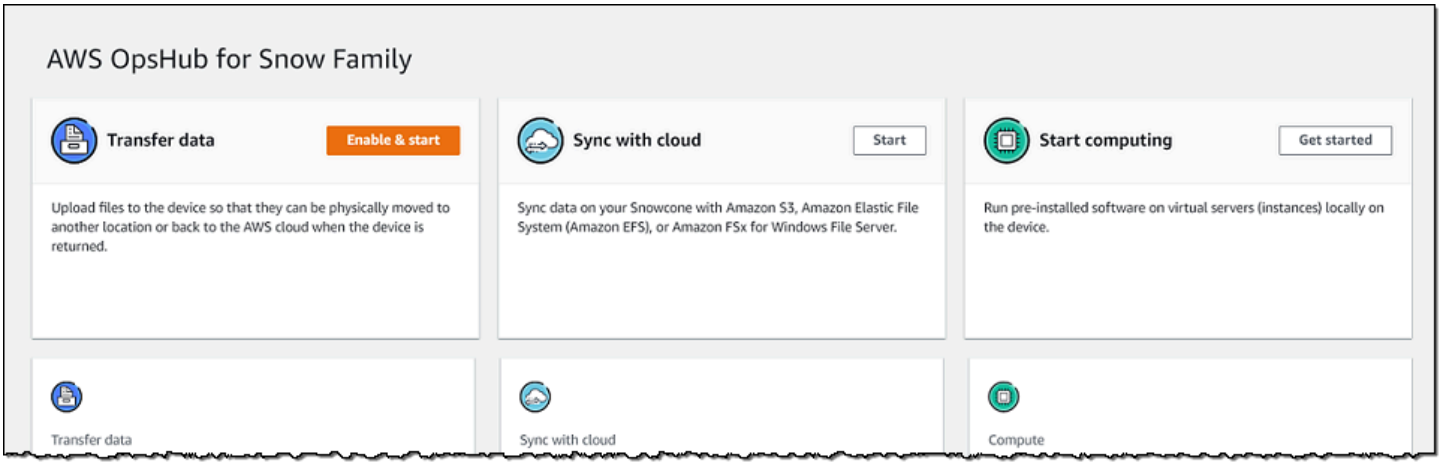
```
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 9C93 4C3B 61F8 C434 9F94  5CA0 1655 BBDE 2B77 0256
```

When using GPG, if the output includes the phrase `BAD signature`, check whether you performed the procedure correctly. If you continue to get this response, contact AWS Support and don't install the agent. The warning message about the trust doesn't mean that the signature is not valid, only that you have not verified the public key. A key is trusted only if you or someone who you trust has signed it.

Managing AWS services on your device

With AWS OpsHub, you can use and manage AWS services on your Snow Family devices. Currently, AWS OpsHub supports the following resources:

- Amazon Elastic Compute Cloud (Amazon EC2) instances – Use Amazon EC2-compatible instances to run software installed on a virtual server without sending it to the AWS Cloud for processing.
- Network File System (NFS) – Use file shares to move data to your device. You can ship the device to AWS to transfer your data to the AWS Cloud, or use DataSync to transfer to other AWS Cloud locations.
- Amazon S3 compatible storage on Snow Family devices – Delivers secure object storage with increased resiliency, scale, and an expanded Amazon S3 API feature-set to rugged, mobile edge, and disconnected environments. Using Amazon S3 compatible storage on Snow Family devices, you can store data and run highly available applications on Snow Family device for edge computing.



Topics

- Using Amazon EC2-compatible compute instances locally
- Managing an Amazon EC2 cluster
- Set up Amazon S3 compatible storage on Snow Family devices
- Managing Amazon S3 adapter storage
- Managing the NFS interface

Using Amazon EC2-compatible compute instances locally

You can use AWS OpsHub to run pre-installed software on virtual servers (instances) locally on your device, and also to manage Amazon EC2 instances on your device.

Topics

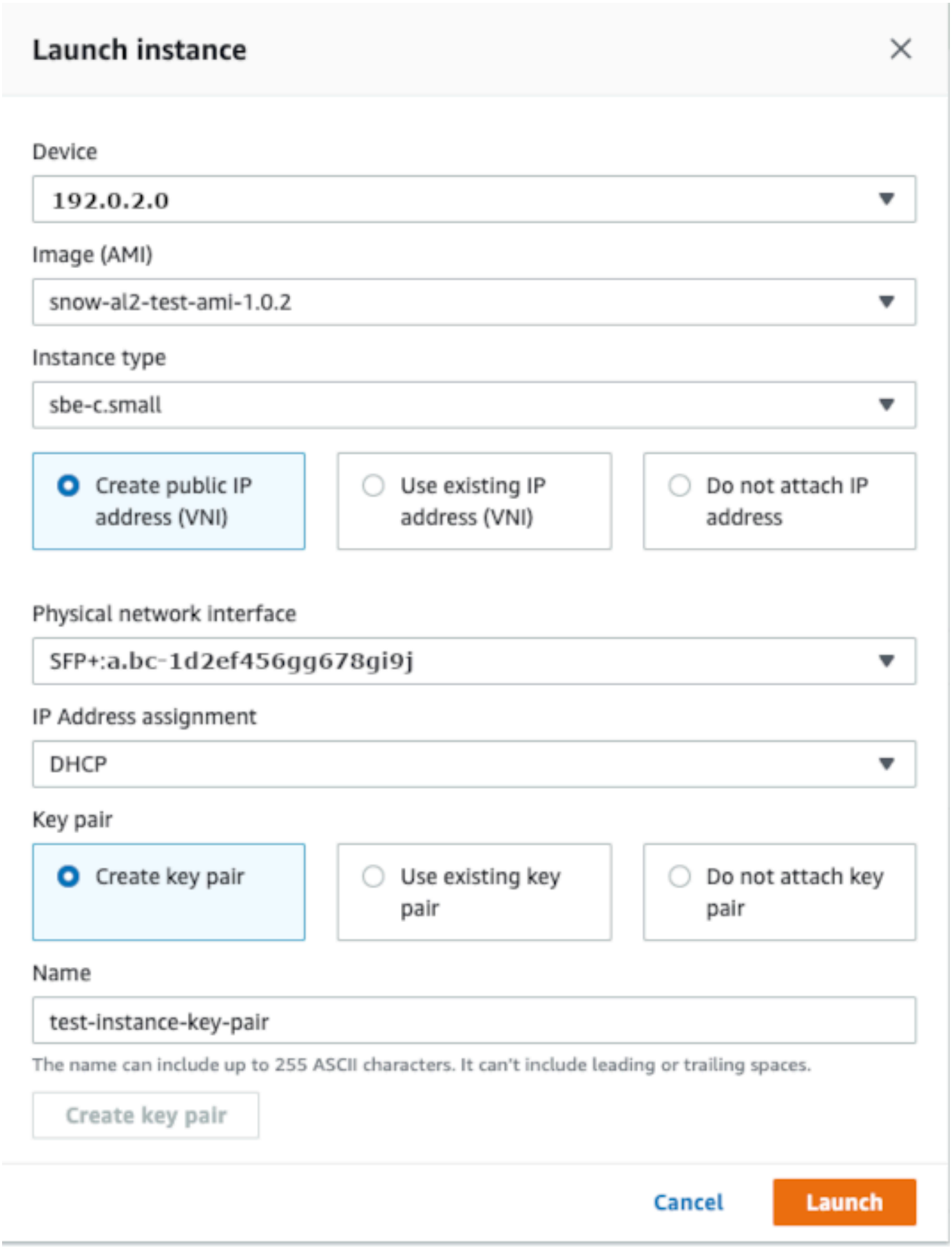
- Launching an Amazon EC2-compatible instance
- Stopping an Amazon EC2-compatible instance
- Starting an Amazon EC2-compatible instance
- Working with key pairs
- Terminating an Amazon EC2-compatible instance
- Using storage volumes locally
- Importing an image into your device as an Amazon EC2-compatible AMI
- Deleting a snapshot
- Deregistering an AMI

Launching an Amazon EC2-compatible instance

Follow these steps to launch an Amazon EC2-compatible instance using AWS OpsHub.

To launch an Amazon EC2-compatible instance

1. Open the AWS OpsHub application.
2. In the **Start computing** section on the dashboard, choose **Get started**. Or, choose the **Services** menu at the top, and then choose **Compute (EC2)** to open the **Compute** page. All your compute resources appear in the **Resources** section.
3. If you have Amazon EC2-compatible instances running on your device, they appear in the **Instance name** column under **Instances**. You can see details of each instance on this page.
4. Choose **Launch instance**. The launch instance wizard opens.
5. For **Device**, choose the Snow device that you want to launch the Amazon EC2-compatible.


Launch instance
Device
192.0.2.0
Image (AMI)
snow-al2-test-ami-1.0.2
Instance type
sbe-c.small
Create public IP address (VNI)
Use existing IP address (VNI)
Do not attach IP address
Physical network interface
SFP+:a.bc-1d2ef456gg678gi9j
IP Address assignment
DHCP
Key pair
Create key pair
Use existing key pair
Do not attach key pair
Name
test-instance-key-pair
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.
Create key pair
Cancel
Launch

6. For **Image (AMI)**, choose an Amazon Machine Image (AMI) from the list. This AMI is used to launch your instance.
7. For **Instance type**, choose one from the list.
8. Choose how you want to attach an IP address to the instance. You have the following options:

   - **Create public IP address (VNI)** – Choose this option to create a new IP address using a physical network interface. Choose a physical network interface and IP address assignment.
   - **Use existing IP address (VNI)** – Choose this option to use an existing IP address and then use existing virtual network interfaces. Choose a physical network interface and a virtual network interface.
   - **Do not attach IP address** – Choose this option if you don't want to attach an IP address.
9. Choose how you want to attach a key pair to the instance. You have the following options:

   Create key pair – Choose this option to create a new key pair and launch the new instance with this key pair.

   Use existing key pair – Choose this option to use an existing key pair to launch the instance.

   Do not attach IP address – Choose this option if you don't want to attach a key pair. You must acknowledge that you won't able to connect to this instance unless you already know the password that is built into this AMI.

   For more information, see Working with key pairs.
10. Choose **Launch**. You should see your instance launching in the **Compute instances** section. The **State** is **Pending** and then changes to **Running** when done.

Stopping an Amazon EC2-compatible instance

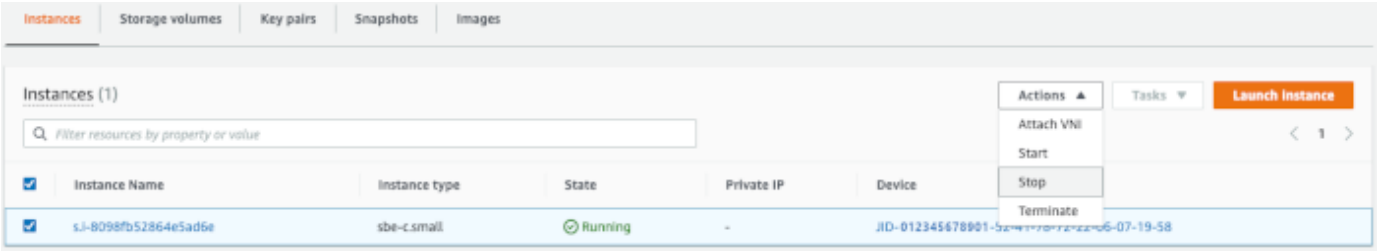
Use the following steps to use AWS OpsHub to stop an Amazon EC2-compatible instance.

To stop an Amazon EC2-compatible instance

1. Open the AWS OpsHub application.
2. In the **Start computing** section of the dashboard, choose **Get started**. Or, choose the **Services** menu at the top, and then choose **Compute (EC2)** to open the **Compute** page.

   All your compute resources appear in the **Resources** section.

3. If you have Amazon EC2-compatible instances running on your device, they appear in the **Instance name** column under **Instances**.
4. Choose the instance that you want to stop, choose the **Actions** menu, and choose **Stop**. The **State** changes to **Stopping**, and then to **Stopped** when done.



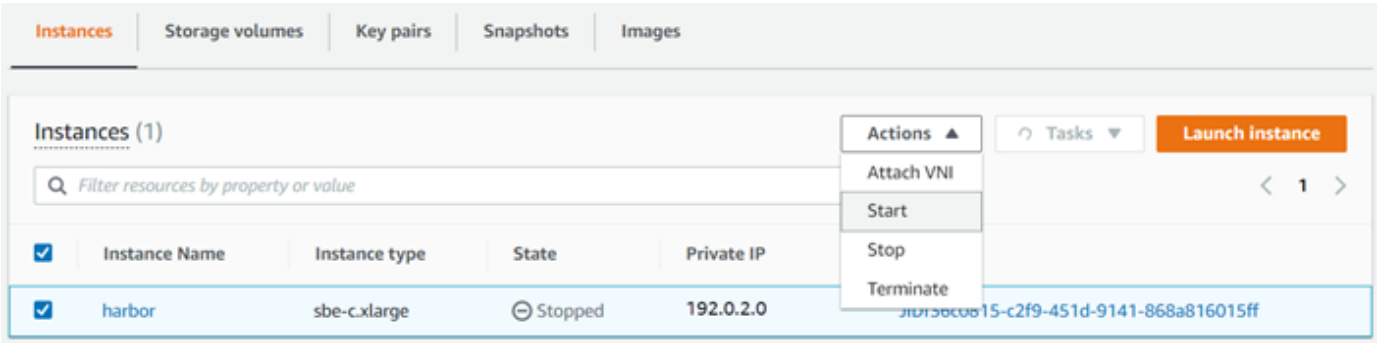
Starting an Amazon EC2-compatible instance

Use these steps to start an Amazon EC2-compatible instance using AWS OpsHub.

To start an Amazon EC2-compatible instance

1. Open the AWS OpsHub application.
2. In the **Start computing** section of the dashboard, choose **Get started**. Or, choose the **Services** menu at the top, and then choose **Compute (EC2)** to open the **Compute** page.

   Your compute resources appear in the **Resources** section.
3. In the **Instance name** column, under **Instances**, find the instance that you want to start.
4. Choose the instance, and then choose **Start**. The **State** changes to **Pending**, and then changes to **Running** when done.



Working with key pairs

When you launch an Amazon EC2-compatible instance and intend to connect to it using SSH, you have to provide a key pair. You can use Amazon EC2 to create a new key pair, or you can import an existing key pair or manage your key pairs.

To create, import, or manage key pairs

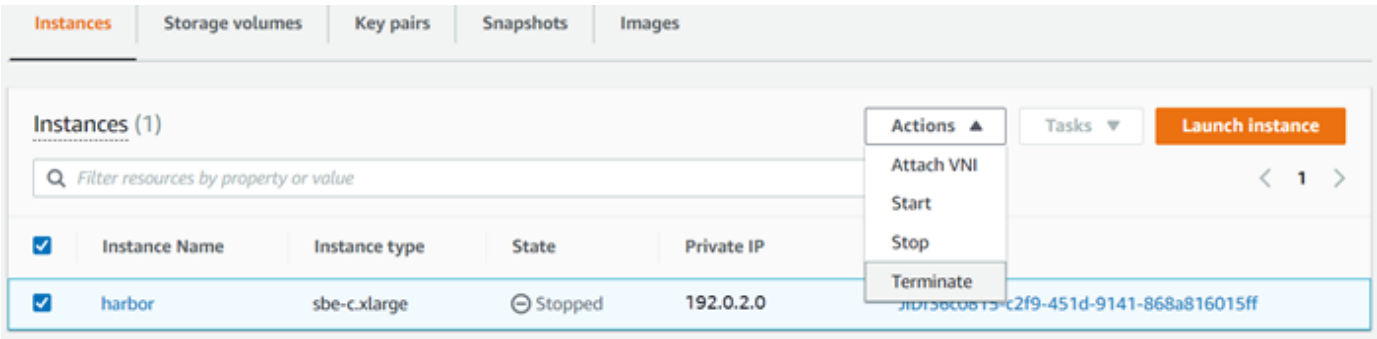
1. Open **Compute** on the AWS OpsHub dashboard.
2. In the navigation pane, choose the **Compute (EC2)** page, and then choose the **Key Pairs** tab. You are redirected to the Amazon EC2 console where you can create, import, or manage your key pairs.
3. For instructions on how to create and import key pairs, see Amazon EC2 key pairs and Linux instances in the *Amazon EC2 User Guide for Linux Instances*.

Terminating an Amazon EC2-compatible instance

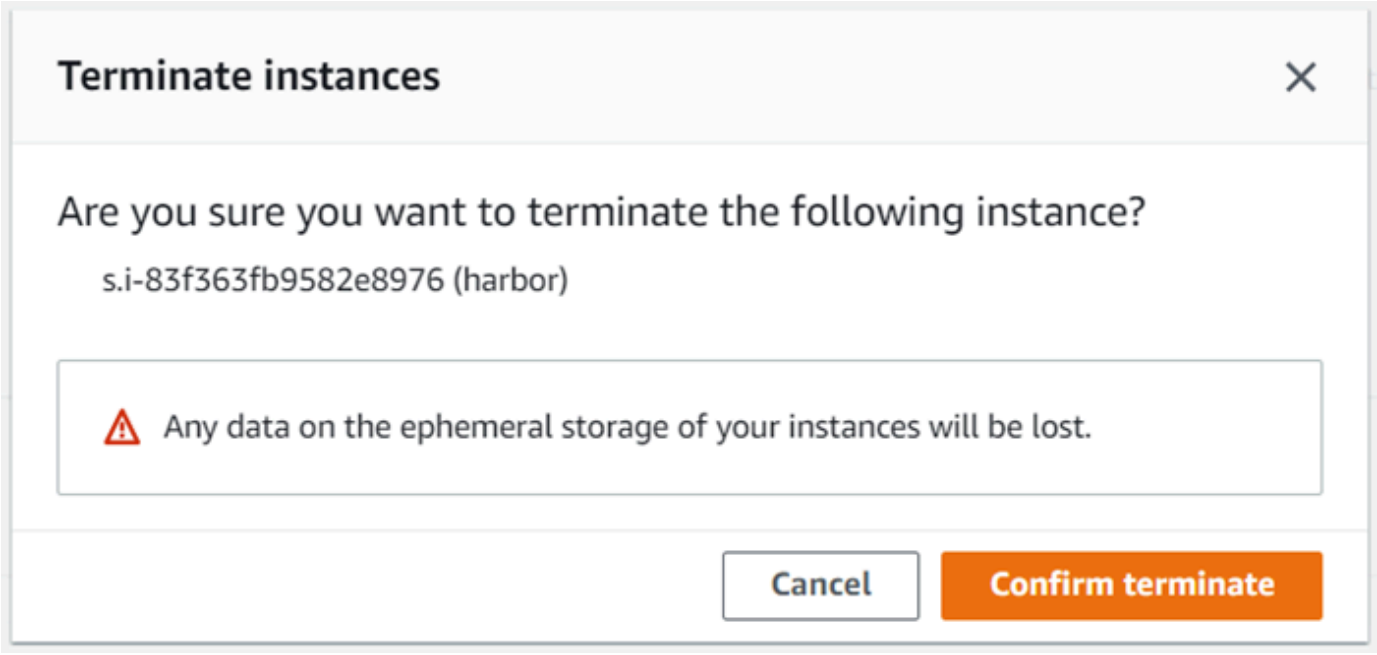
After you terminate an Amazon EC2-compatible instance, you can't restart the instance.

To terminate an Amazon EC2-compatible instance

1. Open the AWS OpsHub application.
2. In the **Start computing** section on the dashboard, choose **Get started**. Or, choose the **Services** menu at the top, and then choose **Compute (EC2)** to open the **Compute** page. You can see all your compute resources in the **Resources** section.
3. In the **Instance name** column, under **Instances**, find the instance that you want to terminate.
4. Choose the instance, and choose the **Actions**menu. From the **Actions** menu, choose **Terminate**.



5. In the **Terminate instances window, choose Confirm terminate**.



> **Note**
>
> After the instance is terminated, you can't restart it.

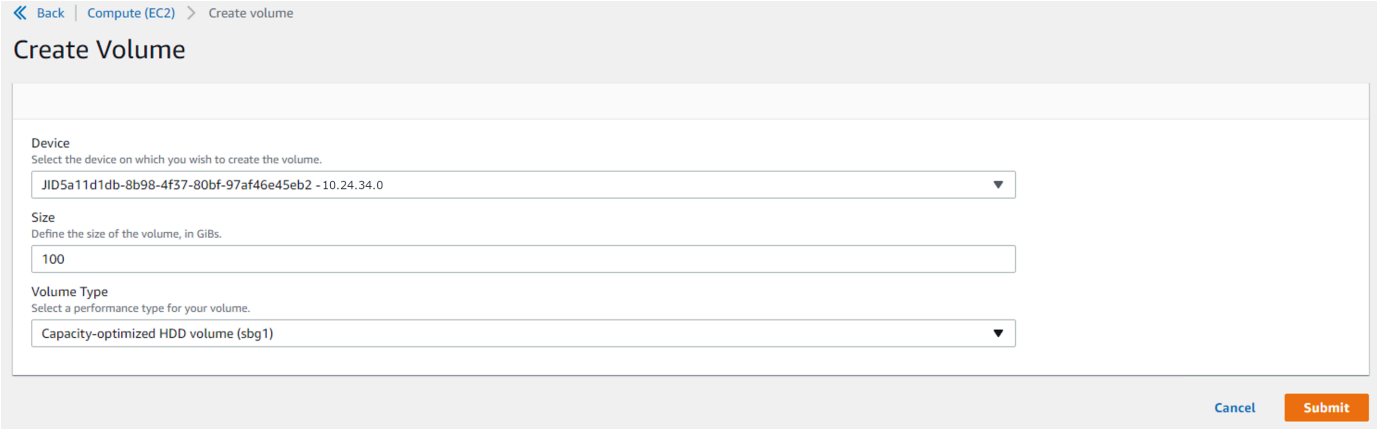
The **State** changes to **Terminating**, and then to **Terminated** when done.

Using storage volumes locally

Amazon EC2-compatible instances use Amazon EBS volumes for storage. In this procedure, you create a storage volume and attach it to your instance using AWS OpsHub.

To create a storage volume

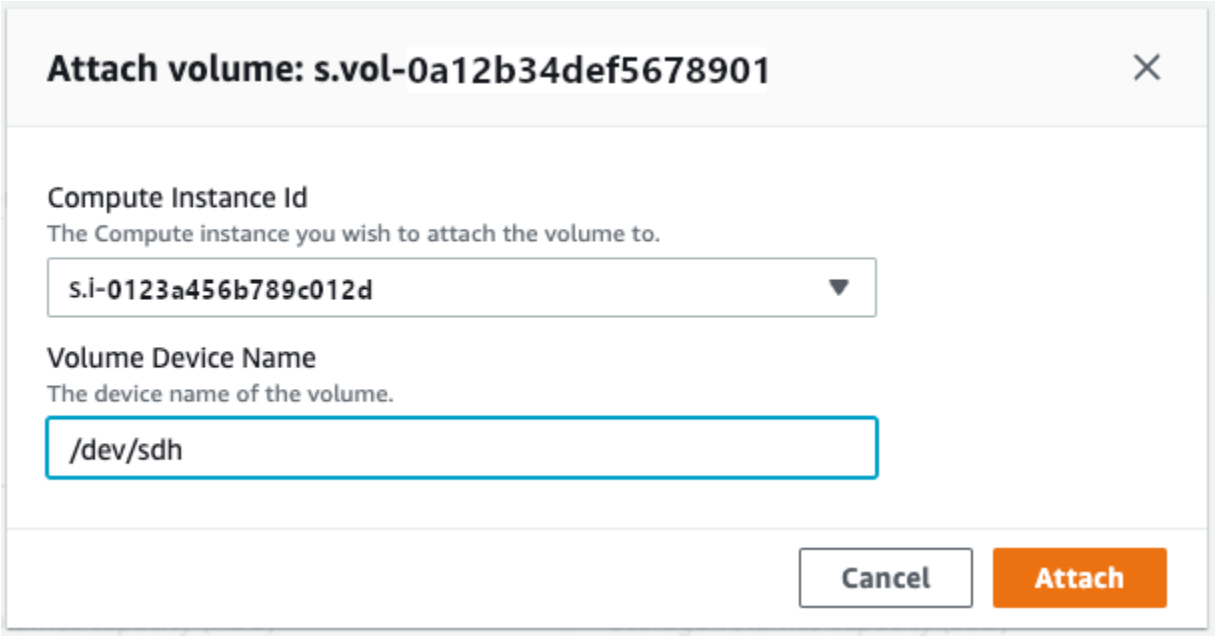
1. Open the AWS OpsHub application.
2. In the **Start computing** section on the dashboard, choose **Get started**. Or, choose the **Services** menu at the top, and then choose **Compute (EC2)** to open the **Compute** page.
3. Choose the **Storage volumes** tab. If you have storage volumes on your device, the details about the volumes appear under **Storage volumes**.
4. Choose **Create volume** to open the **Create volume** page.



5. Choose the device that you want to create the volume on, enter the size (in GiBs) that you want to create, and choose the type of volume.
6. Choose **Submit**. The **State** is **Creating**, and changes to **Available** when done. You can see your volume and details about it in the **Volumes** tab.

To attach a storage volume to your instance

1. Choose the volume that you created, and then choose **Attach volume**.



2. For **Compute instance Id**, choose the instance you want to attach the volume to.
3. For **Volume Device Name**, enter the device name of your volume (for example, **`/dev/sdh`** or **`xvdh`**).
4. Choose **Attach**.

If you no longer need the volume, you can detach it from the instance and then delete it.

Importing an image into your device as an Amazon EC2-compatible AMI

You can import a snapshot of your image into your Snowball Edge device and register it as an Amazon EC2-compatible Amazon Machine Image (AMI). A snapshot is basically a copy of your storage volume that you can use to create an AMI or another storage volume. By doing this, you can bring your own image from an external source onto your device and launch it as an Amazon EC2-compatible instance.

Follow these steps to complete the import of your image.

1. Upload your snapshot into an Amazon S3 bucket on your device.
2. Set up the required permissions to grant access to Amazon S3, Amazon EC2, and VM Import/Export, the feature that is used to import and export snapshots.
3. Import the snapshot from the S3 bucket into your device as an image.
4. Register the image as an Amazon EC2-compatible AMI.
5. Launch the AMI as an Amazon EC2-compatible instance.

> **Note**
>
> Be aware of the following limitations when uploading snapshots to Snow Family devices.
>
> - Snow Family devices currently only support importing snapshots that are in the RAW image format.
> - Snow Family devices currently only support importing snapshots with sizes from 1 GB to 1 TB.

Step 1: Upload a snapshot into an S3 bucket on your device

You must upload your snapshot to Amazon S3 on your device before you import it. This is because snapshots can only be imported from Amazon S3 available on your device or cluster. During the import process, you choose the S3 bucket on your device to store the image in.

To upload a snapshot to Amazon S3

- To create an S3 bucket, see Creating Amazon S3 Storage.

To upload a snapshot to an S3 bucket, see Uploading Files to Amazon S3 Storage.

Step 2: Import the snapshot from an S3 bucket

When your snapshot is uploaded to Amazon S3, you can import it to your device. All snapshots that have been imported or are in the process of being imported are shown in **Snapshots** tab.

To import the snapshot to your device

1. Open the AWS OpsHub application.
2. In the **Start computing** section on the dashboard, choose **Get started**. Or, choose the **Services** menu at the top, and then choose **Compute (EC2)** to open the **Compute** page. All your compute resources appear in the **Resources** section.
3. Choose the **Snapshots** tab to see all the snapshots that have been imported to your device. The image file in Amazon S3 is a .raw file that is imported to your device as a snapshot. You can filter by snapshot ID or the state of the snapshot to find specific snapshots. You can choose a snapshot ID to see details of that snapshot.
4. Choose the snapshot that you want to import, and choose **Import snapshot** to open the **Import snapshot** page.
5. For **Device**, choose the IP address of the Snow Family device that you want to import to.
6. For **Import description** and **Snapshot description**, enter a description for each.
7. In the **Role** list, choose a role to use for the import. Snow Family devices use VM Import/Export to import snapshots. AWS assumes this role and uses it to import the snapshot on your behalf. If you don't have a role configured on your AWS Snowball Edge, open the AWS Identity and Access Management (IAM in AWS OpsHub where you can create a local IAM role. The role also needs a policy that has the required VM Import/Export permissions to perform the import. You must attach this policy to the role. For more details on this please refer to Using IAM Locally.

   The following is an example of the policy.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
```

```
            "Service":"vmie.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

The role you create should have minimum permissions to access Amazon S3 The following is example of a minimum policy.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:GetMetadata"
         ],
         "Resource":[
            "arn:aws:s3:::import-snapshot-bucket-name",
            "arn:aws:s3:::import-snapshot-bucket-name/*"
         ]
      }
   ]
}
```

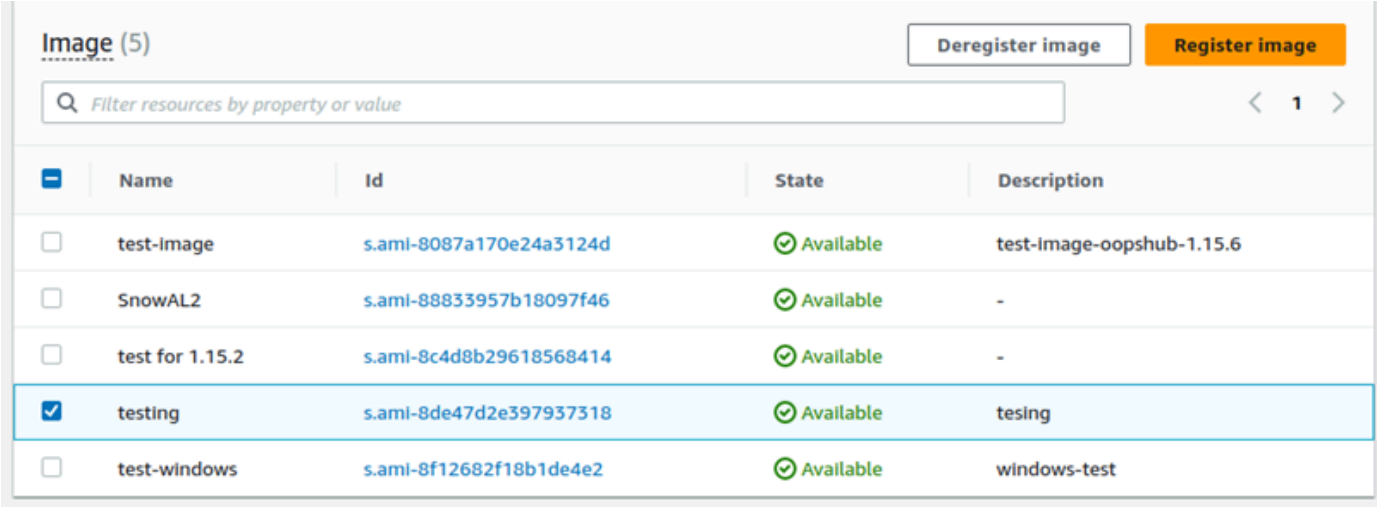
8. Choose **Browse S3** and choose the S3 bucket that contains the snapshot that you want to import. Choose the snapshot, and choose **Submit**. The snapshot begins to download onto your device. You can choose the snapshot ID to see the details. You can cancel the import process from this page.

Step 3: Register the snapshot as an Amazon EC2-compatible AMI

The process of creating an Amazon EC2-compatible AMI from an image imported as a snapshot is known as *registering*. Images that are imported to your device must be registered before they can be launched as Amazon EC2-compatible instances.

To register an image imported as a snapshot

1. Open the AWS OpsHub application.
2. In the **Start computing** section on the dashboard, choose **Get started**. Or, choose the **Services** menu at the top, and then choose **Compute (EC2)** to open the **Compute** page. All your compute resources appear in the **Resources** section.
3. Choose the **Images** tab. You can filter the images by name, ID, or state to find a specific image.
4. Choose the image that you want to register, and choose **Register image**.



5. On the **Register image** page, provide a **Name** and **Description**.
6. For **Root volume**, specify the name of the root device.

   In the **Block device** section, you can change the size of the volume and the volume type.
7. If you want the volume to be deleted when the instance is terminated, choose **Delete on termination**.
8. If you want to add more volumes, choose **Add new volume**.
9. When you are done, choose **Submit**.

Step 4: Launch the Amazon EC2-compatible AMI

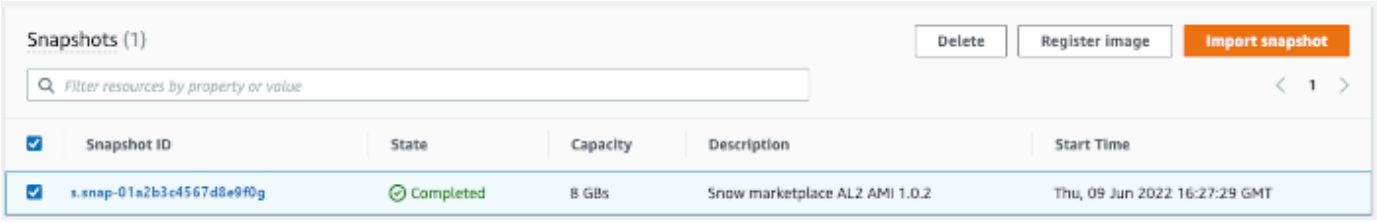
- For more information, see Launching an Amazon EC2-compatible instance.

Deleting a snapshot

If you no longer need a snapshot, you can delete it from your device. The image file in Amazon S3 is a .raw file that is imported to your device as a snapshot. If the snapshot that you are deleting is used by an image, it can't be deleted. After import is completed, you can also delete the .raw file that you uploaded to Amazon S3 on your device.

To delete a snapshot

1. Open the AWS OpsHub application.
2. In the **Start computing** section on the dashboard, choose **Get started**. Or, choose the **Services** menu at the top, and then choose **Compute (EC2)** to open the **Compute** page. All your compute resources appear in the **Resources** section.
3. Choose the **Snapshot** tab to see all snapshots that have been imported. You can filter by snapshot ID or state of the snapshot to find specific snapshots.
4. Choose the snapshot that you want to delete, and choose **Delete**. You can choose multiple snapshots.



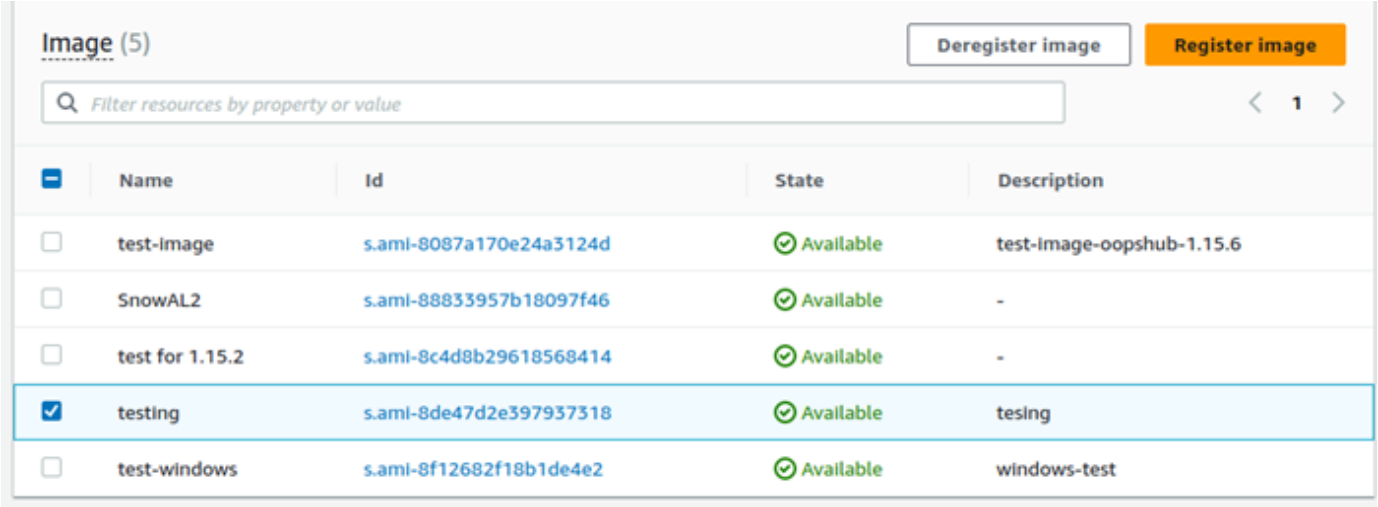
5. In the **Delete snapshot confirmation** box, choose **Delete snapshot**. If your deletion is successful, the snapshot is removed from the list under the **Snapshots** tab.

Deregistering an AMI

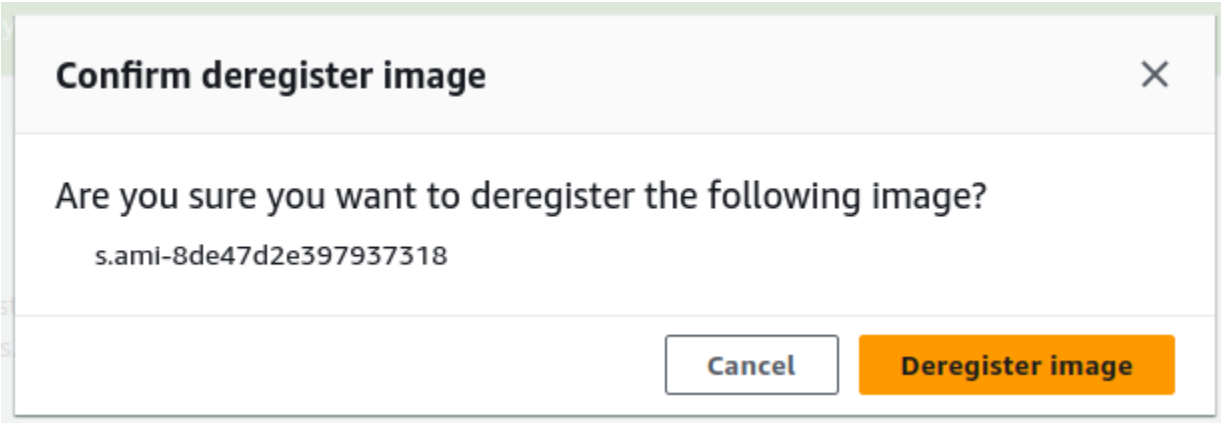
To deregister an AMI

1. Open the AWS OpsHub application.
2. In the **Start computing** section on the dashboard, choose **Get started**. Or, choose the **Services** menu at the top, and then choose **Compute (EC2)** to open the **Compute** page. All your compute resources appear in the **Resources** section.

3. Choose the **Images** tab. All your images are listed. You can filter the images by name, ID, or state to find a specific image.
4. Choose the image that you want to deregister, and choose **Deregister**.



5. In the **Confirm deregister image** window, confirm the image ID and choose **Deregister image**. When deregistering is successful, the image is removed from the list of images.



Managing an Amazon EC2 cluster

An Amazon EC2 *cluster* is a group of devices that provision together as a cluster of devices. To use a cluster, the AWS services on your device must be running at your default endpoint. You also must choose the specific device in the cluster that you want to talk to. You use a cluster on a per-device basis.

To create an Amazon EC2 cluster

1. Connect and log in to your Snow device. For instructions on how to log in to your device, see Unlocking a device.
2. On the **Choose device** page, choose **Snowball Edge cluster**, and then choose **Next**.
3. On the **Connect to your device** page, provide the IP address of the device and the IP addresses of other devices in the cluster.
4. Choose **Add another** device to add more devices, and then choose **Next**.
5. On the **Provide the keys** page, enter the device client unlock code, upload the device manifest, and choose **Unlock device**.

   Snowball Edge devices use 256-bit encryption to help ensure both security and full chain-of-custody for your data.
6. (Optional) Enter a name to create a profile, and then choose **Save profile name**. You are directed to the dashboard, where you see all your clusters.

   You can now start using AWS services and managing your cluster. You manage instances in the cluster the same way you manage individual instances. For instructions, see Managing AWS services on your device or Managing Your Devices.

Set up Amazon S3 compatible storage on Snow Family devices

The Amazon S3 compatible storage on Snow Family devices service is not active by default. To start the service on a device or cluster, you must create two virtual network interfaces (VNICs) on each device to attach to the `s3control` and `s3api` endpoints.

Topics

Prerequisites

Before you can set up your device or cluster using AWS OpsHub for Snow Family, do the following:

- Power on your Snowball Edge device and connect it to your network.
- On your local machine, download and install the latest version of AWS OpsHub. Connect to the device or cluster to unlock it with a manifest file. For more information, see unlocking a device.

Using the simple setup option

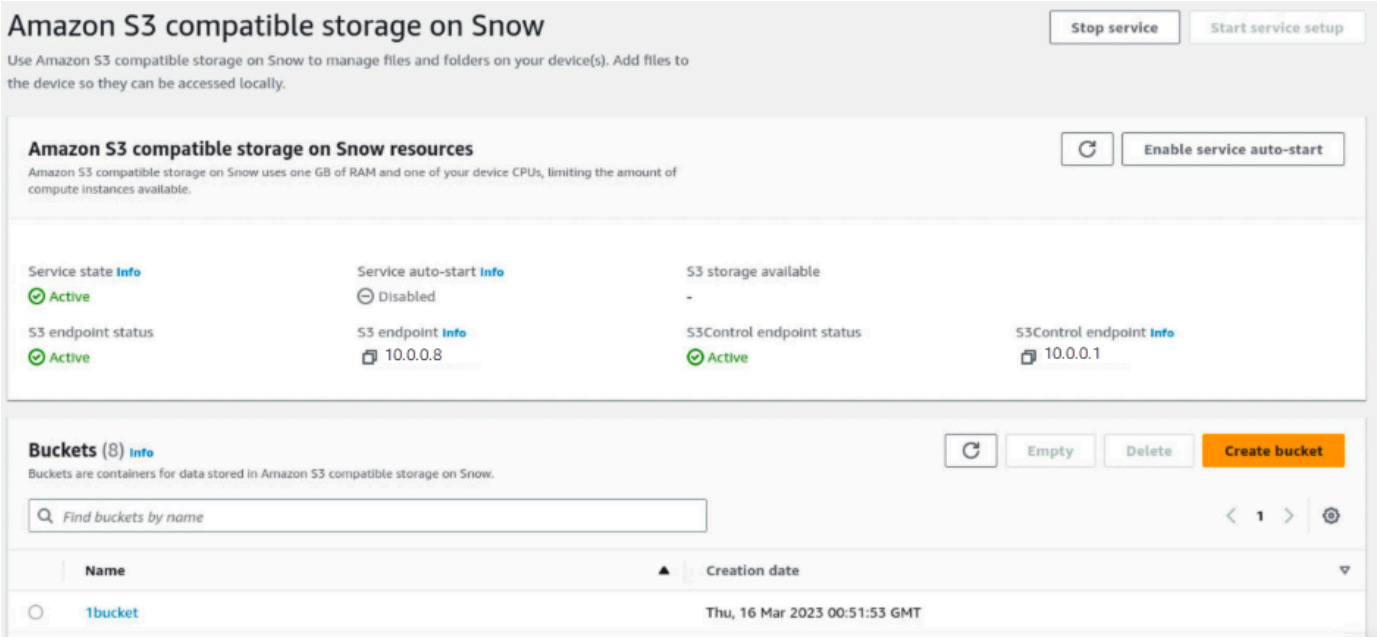
Use the simple setup option if your network uses DHCP. With this option, the VNICs are created automatically on each device when you start the service.

1. Log in to AWS OpsHub, then choose **Manage Storage**.

   This takes you to the Amazon S3 compatible storage on Snow Family devices landing page.
2. For **Start service setup type**, choose **Simple**.
3. Choose **Start service**.

   > **Note**
   >
   > This takes a few minutes to complete and depends on the number of devices you're using.

   After the service starts, the Service state is active, and there are endpoints.



Using the advanced setup option

Use the advanced setup option if your network uses static IP addresses or if you want to reuse existing VNIs. With this option, you create VNICs for each device manually.

1. Log in to AWS OpsHub, then choose **Manage Storage**.

   This takes you to the Amazon S3 compatible storage on Snow Family devices landing page.
2. For **Start service setup type**, choose **Advanced**.
3. Select the devices that you need to create VNICs for.

   For clusters, you need a minimum quorum of devices to start the Amazon S3 compatible storage on Snow Family devices service. The quorum is two for a three-node cluster.

   > **Note**
   >
   > For the initial start of the service in a cluster setup, you must have all devices in the cluster configured and available for the service to start. For subsequent starts, you can use a subset of the devices if you meet quorum, but the service will start in a degraded state.

4. For each device, choose an existing VNIC or select **Create VNI**.

Each device needs a VNIC for the **S3 endpoint** for object operations and another for the **S3Control endpoint** for bucket operations.

5. If you're creating a VNIC, choose a physical network interface and enter the status IP address and subnet mask, then choose **Create virtual network interface**.
6. After you create your VNICS, choose **Start service**.

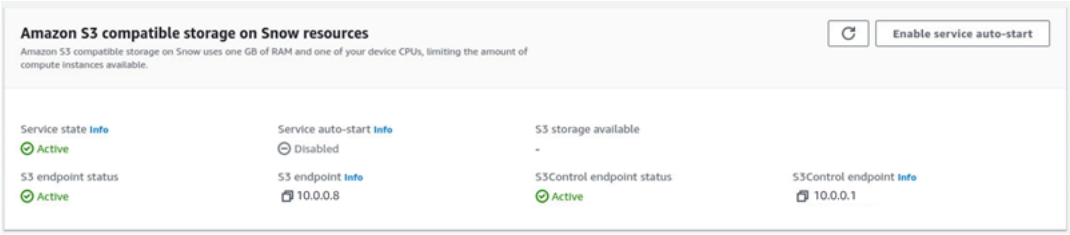
> **Note**
>
> This takes a few minutes to complete and depends on the number of devices you're using.

After the service starts, the Service state is active, and there are endpoints.

Configuring the Amazon S3 compatible storage on Snow Family devices service to autostart

1. Log in to AWS OpsHub, then choose **Manage Storage**.

   This takes you to the Amazon S3 compatible storage on Snow Family devices landing page.
2. In **Amazon S3 compatible storage on Snow resources**, choose **Enable service auto-start**. The system configures the service to automatically start in the future.

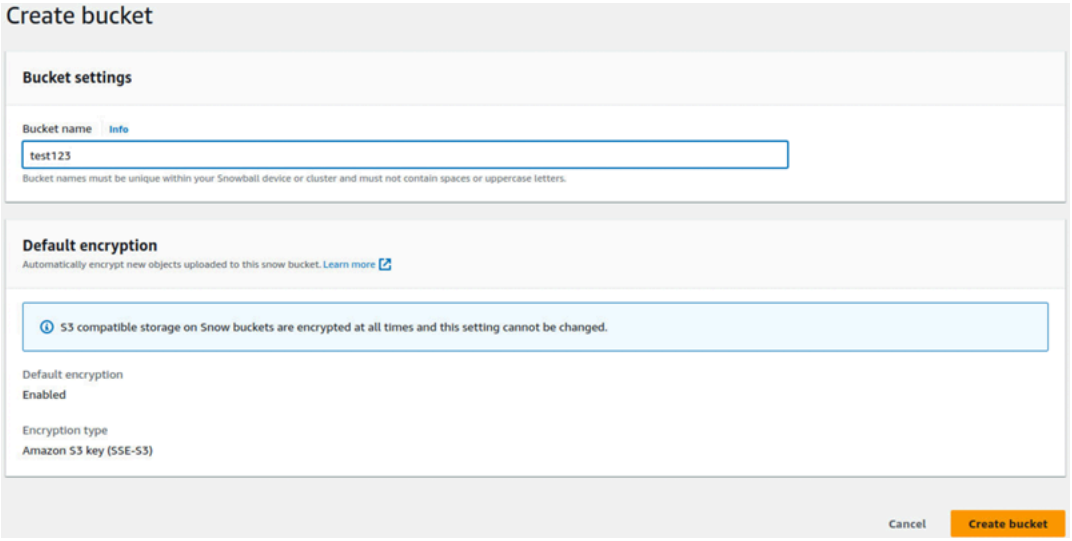


Creating a bucket in Amazon S3 compatible storage on Snow Family devices

Use the AWS OpsHub interface to create an Amazon S3 bucket on your Snow Family device.

1. Open AWS OpsHub.
2. In **Manage storage**, choose **Get started**. The **Amazon S3 compatible storage on Snow** page appears.

3. In **Buckets**, choose **Create bucket**. The **Create bucket** screen appears.



4. In **Bucket name**, enter a name for the bucket.

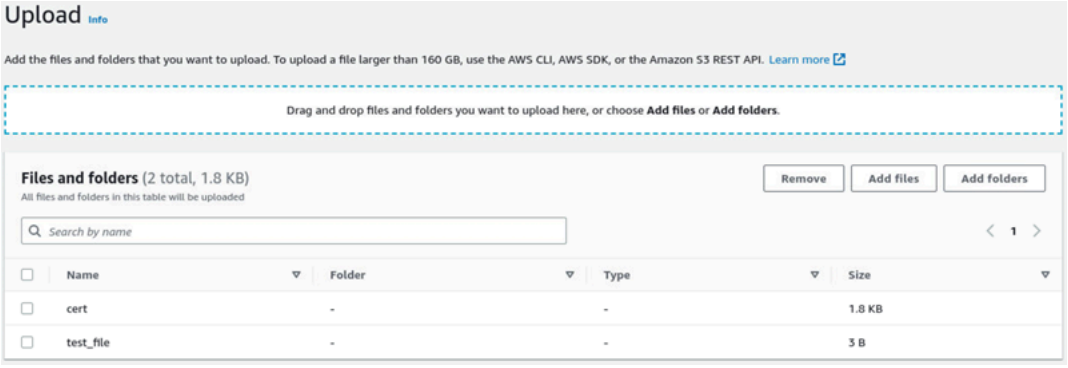
> **Note**
>
> Bucket names must be unique within your Snowball device or cluster and must not contain spaces or uppercase letters.

5. Choose **Create bucket**. The system creates the bucket and it appears in **Buckets** in the **Amazon S3 compatible storage on Snow** page.

Upload files and folders to Amazon S3 compatible storage on Snow Family devices buckets

Use the AWS OpsHub interface to upload files and folders to Amazon S3 compatible storage on Snow Family devices buckets. Files and folders may be uploaded separately or together.

1. Open AWS OpsHub
2. In **Manage storage**, in **Buckets**, choose a bucket in which to upload files. The page for that bucket appears.
3. In the bucket page, choose **Upload files**. The **Upload** page appears.



4. Upload files or folders by dragging them from an operating system file manager to the AWS OpsHub window or do the following:

   a. Select **Add files** or **Add folders**.

   b. Select one or more files or folders to upload. Select **Open**.

   The system uploads the selected files and folders to the bucket on the device. After the upload is complete, the names of the files and folders appear in the **Files and folders** list.

Remove files and folders from Amazon S3 compatible storage on Snow Family devices buckets

Use the AWS OpsHub interface to remove and permanently delete files and folders from buckets on the Snow Family device.

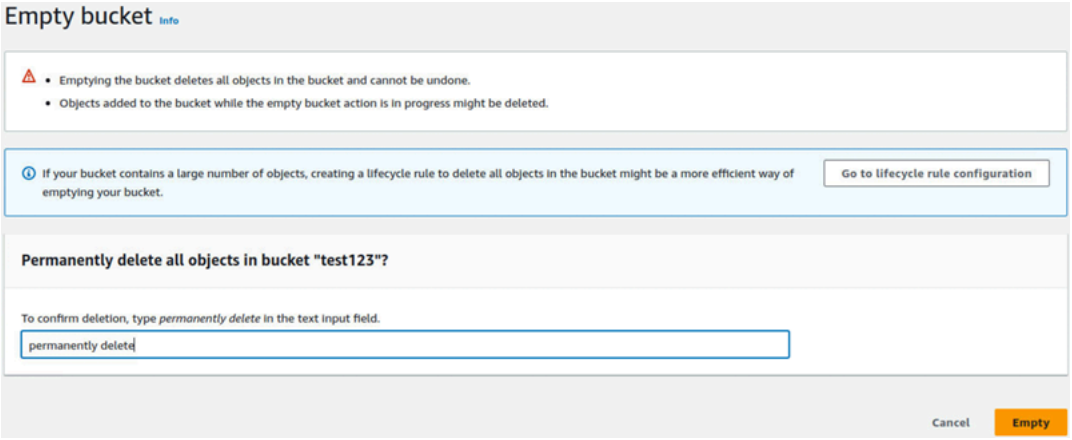
1. Open AWS OpsHub.
2. In **Manage storage**, in **Buckets**, select the name of a bucket from which to delete files and folders. The page for that bucket appears.
3. In **Files and folders** select the check boxes of the files and folders to permanently delete.
4. Select **Remove**. The system removes the files or folders from the bucket on the device.

Delete buckets from Amazon S3 compatible storage on Snow Family devices

Before you can delete a bucket from a device, the bucket must be empty. Either remove files and folders from the bucket or use the empty bucket tool. To remove files and folders, see Remove files and folders from Amazon S3 compatible storage on Snow Family devices buckets.

To use the empty bucket tool

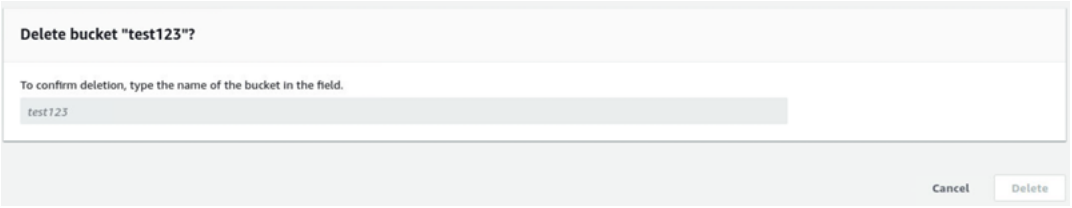
1. Open AWS OpsHub.
2. In **Manage storage**, in **Buckets**, select the radio button of the bucket to empty.
3. Select **Empty**. The **Empty bucket** page appears.



4. In the text box in the **Empty bucket** page, type **`permanently delete`**.
5. Select **Empty**. The system empties the bucket.

To delete an empty bucket

1. In **Manage storage**, in **Buckets**, select the radio button of the bucket to delete.
2. Select **Delete**. The **Delete bucket** page appears.



3. In the text box in the **Delete bucket** page, type the name of the bucket.
4. Select **Delete**. The system deletes the bucket from the device.

Managing Amazon S3 adapter storage

You can use AWS OpsHub to create and manage Amazon Simple Storage Service (Amazon S3) storage on your Snow Family devices using the S3 adapter for import and export jobs.

Topics

Accessing Amazon S3 Storage

You can upload files to your device and access the files locally. You can physically move them to another location on the device, or import them back to the AWS Cloud when the device is returned.

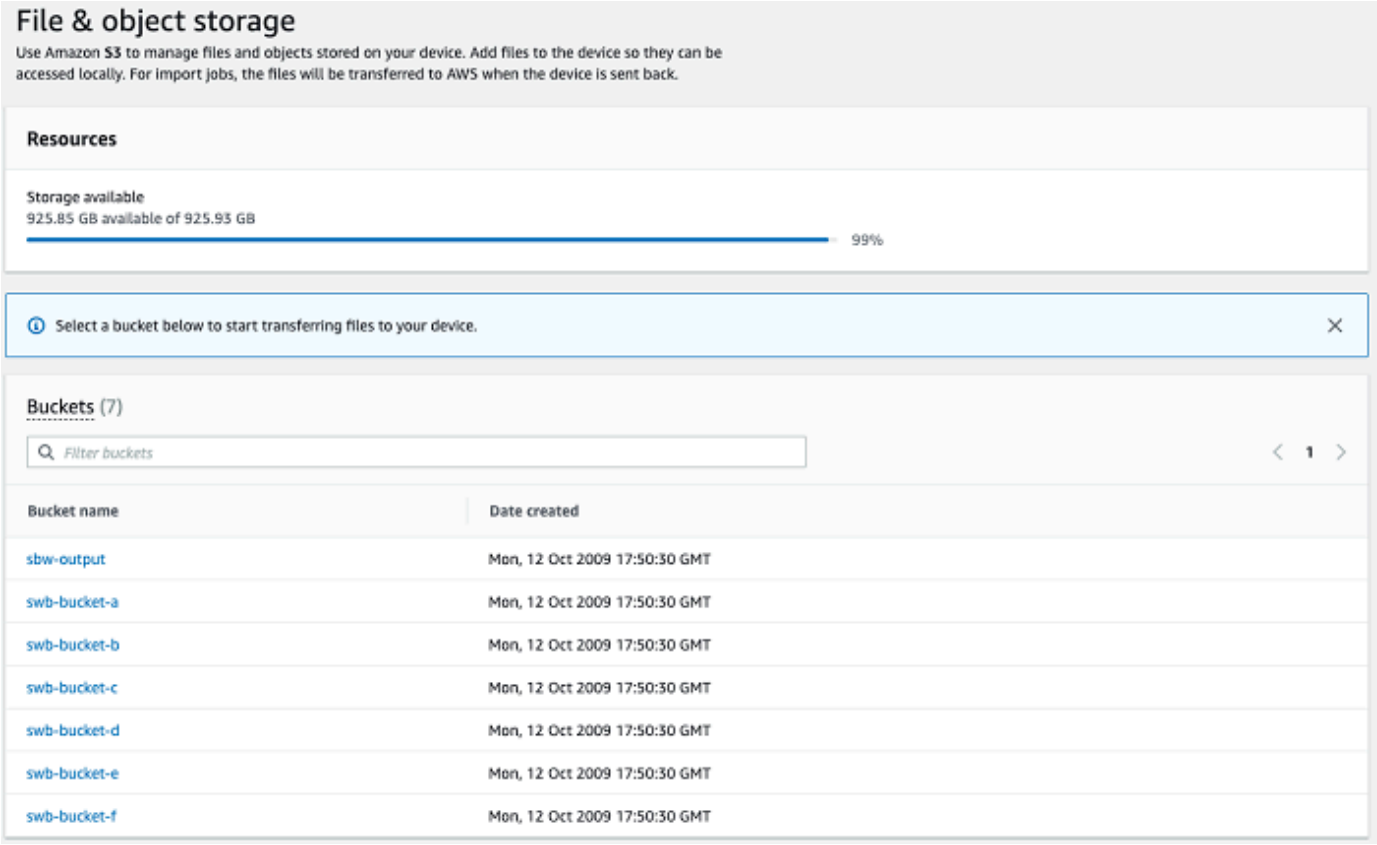
Snow Family devices use Amazon S3 buckets to store and manage files on your device.

To access an S3 bucket

1. Open the AWS OpsHub application.
2. In the **Manage file storage** section of the dashboard, choose **Get started**.

   If your device has been ordered with the Amazon S3 transfer mechanism, they appear in the **Buckets** section of the **File & object storage** page. On the **File & object storage** page, you can see details of each bucket.

   > **Note**
   >
   > If the device was ordered with the NFS transfer mechanism, the bucket name will appear under the mount points section after NFS service is configure and activated. For more information on using the file interface, see Managing the NFS interface.



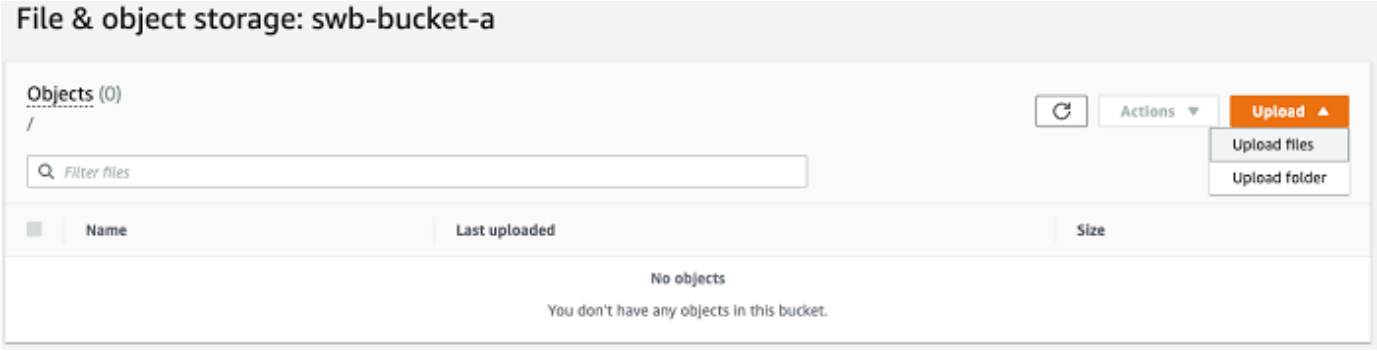
Uploading files to Amazon S3 storage

To upload a file

1. In the **Manage file storage** section on the dashboard, choose **Get started**.

   If you have Amazon S3 buckets on your device, they appear in the **Buckets** section on the **File storage** page. You can see details of each bucket on the page.

2. Choose the bucket that you want to upload files into.
3. Choose **Upload** then **Upload files** or drag and drop the files in the bucket, and choose **OK**.

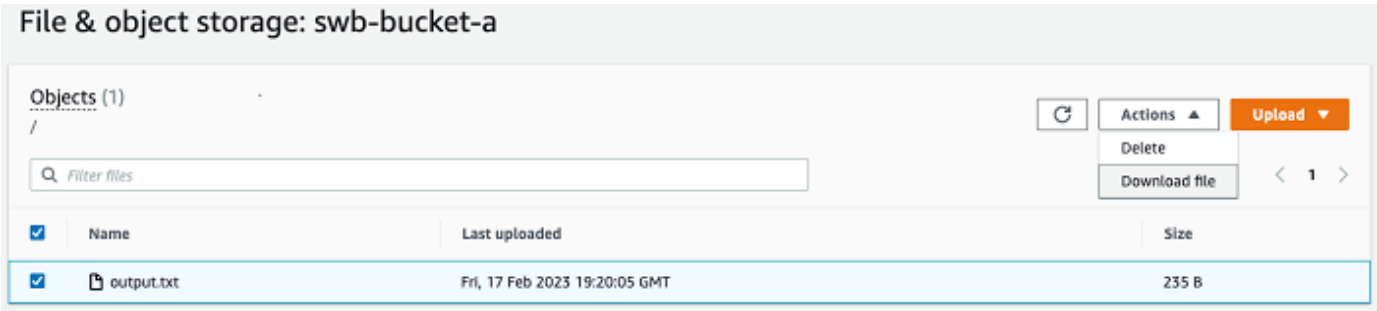


> **Note**
>
> To upload larger files, you can use the multipart upload feature in Amazon S3 using the AWS CLI. For learning more about configuring S3 CLI settings, see CLI S3 Configuration. For information on multipart upload, see Multipart Upload Overview in the *.Amazon Simple Storage Service User Guide*
> Uploading a folder from a local machine to Snowball Edge using the AWS OpsHub is supported. If the folder size is very large, it takes some time to read the file/folder selection. Currently, there is no progress tracker for file reading. However, a progress tracker is displayed once the upload process kicks off.

Downloading files from Amazon S3 storage

To download a file

1. In the **Manage file storage** section of the dashboard, choose **Get started**. If you have S3 buckets on your device, they appear in the **Buckets** section on the **File storage** page. You can see details of each bucket on the page.
2. Choose the bucket that you want to download files from and navigate to the file that you want to download. Choose one or more files.



3. In the **Actions** menu, choose **Download**.
4. Choose a location to download the file to, and choose **OK**.

Deleting files from Amazon S3 storage

If you no longer need a file, you can delete it from your Amazon S3 bucket.

To delete a file

1. In the **Manage file storage** section of the dashboard, choose **Get started**. If you have Amazon S3 buckets on your device, they appear in the **Buckets** section on the **File storage** page. You can see details of each bucket on the page.
2. Choose the bucket you want to delete files from, and navigate to the file that you want to delete.
3. On the **Actions** menu, choose **Delete**.
4. In the dialog box that appears, choose **Confirm delete**.

Managing the NFS interface

Use the Network File System (NFS) interface to upload files to the Snow Family device as if the device is local storage to your operating system. This allows for a more user-friendly approach to transferring data because you can use features of your operating system, like copying files, dragging and dropping them, or other graphical user interface features. Each S3 bucket on the device is available as an NFS interface endpoint and can be mounted to copy data to. The NFS interface is available for import jobs.

You can use the NFS interface if the Snowball Edge device was configured to include it when the job to order the device was created. If the device is not configured to include the NFS interface, use the S3 adapter or Amazon S3 compatible storage on Snow Family devices to transfer data. For more information about the S3 adapter, see Managing Amazon S3 adapter storage. For more information about Amazon S3 compatible storage on Snow Family devices, see Set up Amazon S3 compatible storage on Snow Family devices.

When started, the NFS interface uses 1 GB of memory and 1 CPU. This may limit the number of other services running on the Snow Family device or the number of EC2-compatible instances that can run.

Data transferred through the NFS interface is not encrypted in transit. When configuring the NFS interface, you can provide CIDR blocks and the Snow Family device will restrict access to the NFS interface from client computers with addresses in those blocks.

Files on the device will be transferred to Amazon S3 when it is returned to AWS. For more information, see Importing Jobs into Amazon S3.

For more information about using NFS with your computer operating system, see the documentation for your operating system.

Keep the following details in mind when using the NFS interface.

- File names are object keys in your local S3 bucket on the Snow Family device. The key name is a sequence of Unicode characters whose UTF-8 encoding is at most 1,024 bytes long. We recommend using NFSv4.1 where possible and encode file names with Unicode UTF-8 to ensure a successful data import. File names that are not encoded with UTF-8 might not be uploaded to S3 or might be uploaded to S3 with a different file name depending on the NFS encoding you use.
- Ensure that the maximum length of your file path is less than 1024 characters. Snow Family devices do not support file paths that are greater that 1024 characters. Exceeding this file path length will result in file import errors.
- For more information, see Object keys in the Amazon Simple Storage Service User Guide.
- For NFS based transfers, standard POSIX style meta-data will be added to your objects as they get imported to Amazon S3 from Snow Family devices. In addition, you will see meta-data "x-amz-meta-user-agent aws-datasync" as we currently use AWS DataSync as part of the internal import mechanism to Amazon S3 for Snow Family device import with NFS option.
- You can transfer up to 40M files using a single Snowball Edge device. If you require to transfer more than 40M files in a single job, please batch the files in order to reduce the file numbers per each transfer. Individual files can be of any size with a maximum file size of 5 TB for Snowball Edge devices with the enhanced NFS interface or the S3 interface.

You can also configure and manage the NFS interface with the Snowball Edge client, a command line interface (CLI) tool. For more information, see Managing the NFS interface.

Topics

- Starting the NFS service on a Windows operating system
- Configuring the NFS interface automatically

- Configuring the NFS interface manually
- Managing NFS endpoints on the Snow Family device
- Mounting NFS endpoints on client computers
- Stopping the NFS interface

Starting the NFS service on a Windows operating system

If your client computer is using the Windows 10 Enterprise or Windows 7 Enterprise operating system, start the NFS service on the client computer before configuring NFS in the AWS OpsHub application.

1. On your client computer, open **Start**, choose **Control Panel** and choose **Programs**.
2. Choose **Turn Windows features on or off**.

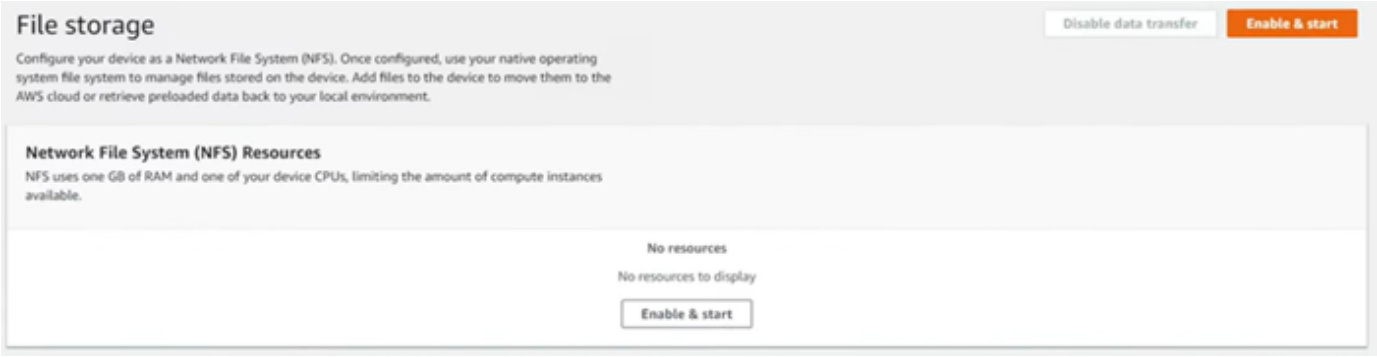
> **Note**
>
> To turn Windows features on, you may need to provide an admin user name and password for your computer.

3. Under **Services for NFS**, choose **Client for NFS** and choose **OK**.

Configuring the NFS interface automatically

The NFS interface is not running on the Snow Family device by default, so you need to start it to enable data transfer on the device. With a few clicks, your Snow Family device can quickly and automatically configure the NFS interface for you. You can also configure the NFS interface yourself. For more information, see Configuring the NFS interface manually.

1. In the **Transfer data** section on the dashboard, choose **Enable & start**. This could take a minute or two to complete.



2. When the NFS service is started, the IP address of the NFS interface is shown on the dashboard and the **Transfer data** section indicates that the service is active.
3. Choose **Open in Explorer** (if using a Windows or a Linux operating system) to open the file share in your operating system's file browser and start transferring files to the Snow Family device. You can copy and paste or drag and drop files from your client computer into the file share. In Windows operating system, your file share looks like the following `buckets(\\12.123.45.679)(Z:)`.

> **Note**
>
> In Linux operating systems, mounting NFS endpoints requires root permissions.

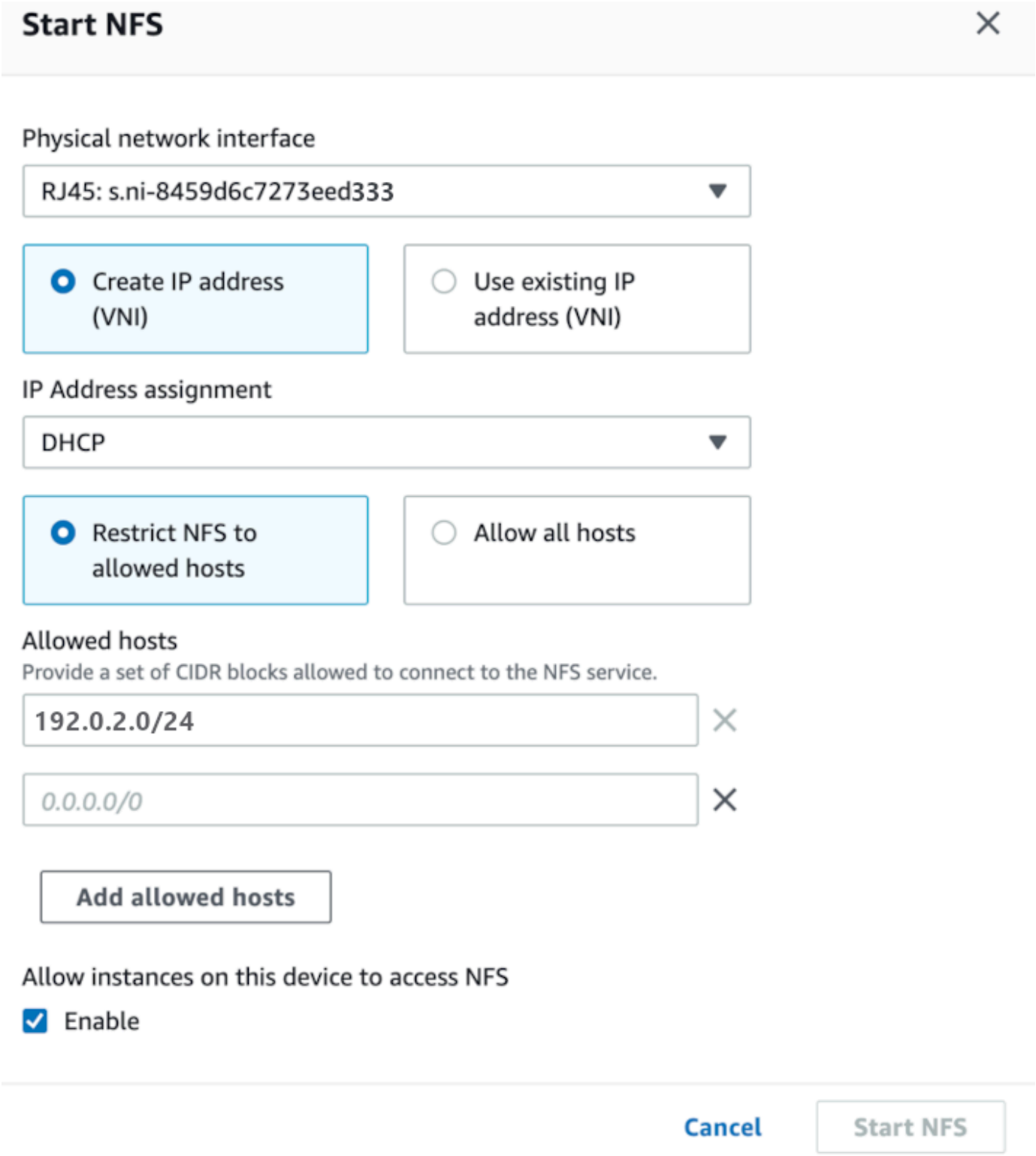
Configuring the NFS interface manually

The NFS interface is not running on the Snow Family device by default, so you need to start it to enable data transfer on the device. You can manually configure the NFS interface by providing the IP address of a Virtual Network Interface (VNI) running on the Snow Family device and restricting access to your file share, if required. Before configuring the NFS interface manually, set up a virtual network interface (VNI) on your Snow Family device. For more information, see Network Configuration for Compute Instances.

You can also have the Snow Family device configure the NFS interface automatically. For more information, see Configuring the NFS interface automatically.

1. At the bottom of **Transfer data** section, on the dashboard, choose **Configure manually**.

2. Choose **Enable & start** to open the **Start NFS** wizard. The **Physical network interface** field is populated.



3. Choose **Create IP address (VNI)** or choose **Use existing IP address**.

4. If you choose **Create IP address (VNI)**, then choose **DHCP** or **Static IP** in the **IP Address assignment** list box.

   > **Important**
   >
   > If you use a DHCP network, it is possible that the NFS interface's IP address could be reassigned by the DCHP server. This can happen after the device has been disconnected and the IP addresses are recycled. If you set an allowed host range and the address of the client changes, another client can pick up that address. In this case, the new client will have access to the share. To prevent this, use DHCP reservations or static IP addresses.

   If you choose **Use existing IP address**, then choose a virtual network interface from the **Virtual network interface** list box.

5. Choose to restrict access to the NFS interface and provide a block of allowed network addresses, or allow any devices on the network to access the NFS interface on the Snow Family device.

   - To restrict access to the NFS interface on the Snow Family device, choose **Restrict NFS to allowed hosts**. In **Allowed hosts** enter a set of CIDR blocks. If you want to allow access to more than one CIDR block, enter another set of blocks. To remove a set of blocks, choose **X** next to the field containing the blocks. Choose **Add allowed hosts**.

     > **Note**
     >
     > If you choose **Restrict NFS to allowed hosts** and do not provide allowed CIDR blocks, the Snow Family device will deny all requests to mount the NFS interface.

   - To allow any device on the network to access the NFS interface, choose **Allow all hosts**.

6. To allow EC2-compatible instances running on the Snow Family device to access the NFS adapter, choose **Enable**.

7. Choose **Start NFS**. It could take about a minute or two to start.

   > **Important**
   >
   > Don't turn off the Snow Family device while the NFS interface is starting.

From the **Network File System (NFS) Resources** section, the **State** of the NFS interface shows as **Active**. You will need the IP address listed to mount the interface as local storage on client computers.

Managing NFS endpoints on the Snow Family device

Each S3 bucket on the Snow Family device is represented as an endpoint and listed in **Mount paths**. After the NFS interface is started, mount an endpoint to transfer files to or from that endpoint. Only one endpoint can be mounted at a time. To mount a different endpoint, unmount the current endpoint first.

To mount an endpoint

1. In the **Mount paths** section, do one of the following to select an endpoint:
   - In the **Filter endpoints** field, enter all or part a bucket name to filter the list of available endpoints on your entry, then choose the endpoint.
   - Choose the endpoint to mount in the **Mount paths** list.
2. Choose **Mount NFS endpoint**. The Snow Family device mounts the endpoint for use.

To unmount an endpoint

1. In the **Mount paths** section, choose the endpoint to unmount.
2. Choose **Unmount endpoint**. The Snow Family device unmounts the endpoint and it is no longer available for use.

> **Note**
>
> Before unmounting an endpoint, ensure no data is being copied from or to it.

Mounting NFS endpoints on client computers

After the NFS interface is started and an endpoint mounted, mount the endpoint as local storage on client computers.

1. In **Mount paths**, choose the copy icon of the endpoint to mount. Paste it in your operating system when mounting the endpoint.
2. The following are the default mount commands for Windows, Linux, and macOS operating systems.

   - Windows:

     ```
     mount -o nolock rsize=128 wsize=128 mtype=hard nfs-interface-ip-address:/
     buckets/BucketName *
     ```

   - Linux:

     ```
     mount -t nfs nfs-interface-ip-address:/buckets/BucketName mount_point
     ```

   - macOS:

     ```
     mount -t nfs -o vers=3,rsize=131072,wsize=131072,nolocks,hard,retrans=2 nfs-
     interface-ip-address:/buckets/$bucketname mount_point
     ```

Stopping the NFS interface

Stop the NFS interface on the Snow Family device when you are done transferring files to or from it.

1. From the dashboard, choose **Services** and then choose **File Storage**.
2. On the **File Storage** page, choose **Disable data transfer**. It usually takes up to 2 minutes for the NFS endpoints to disappear from the dashboard.

Managing Your Devices

You use the AWS OpsHub to manage your Snow Family devices. On the **Device details** page, you can perform the same tasks that you do using the AWS CLI, including changing the alias of your device, rebooting the device, and checking for updates.

Topics

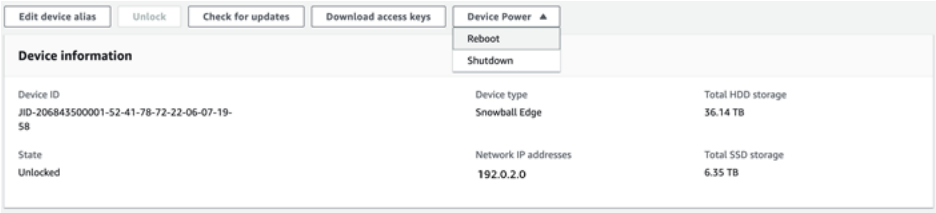
Rebooting your device

Follow these steps to use AWS OpsHub to reboot your Snow device.

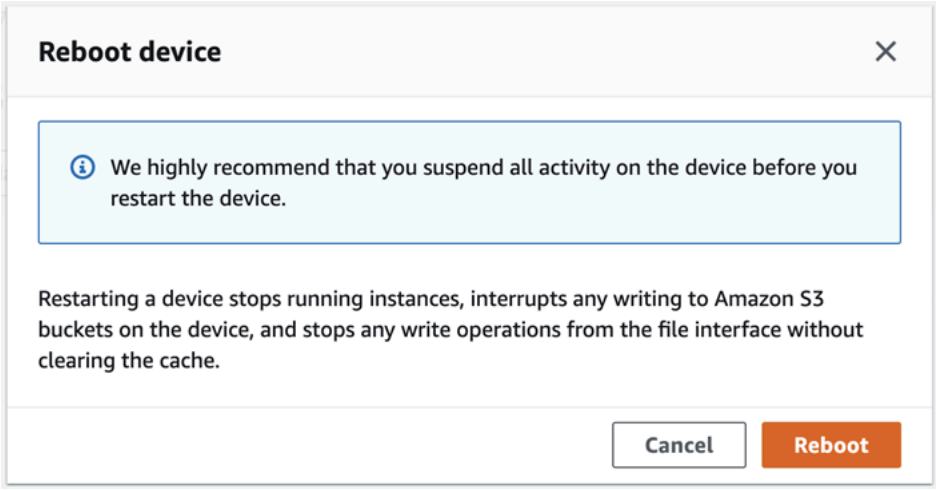
> **Important**
>
> We highly recommend that you suspend all activities on the device before you reboot the device. Rebooting a device stops running instances and interrupts any writing to Amazon S3 buckets on the device.

To reboot a device

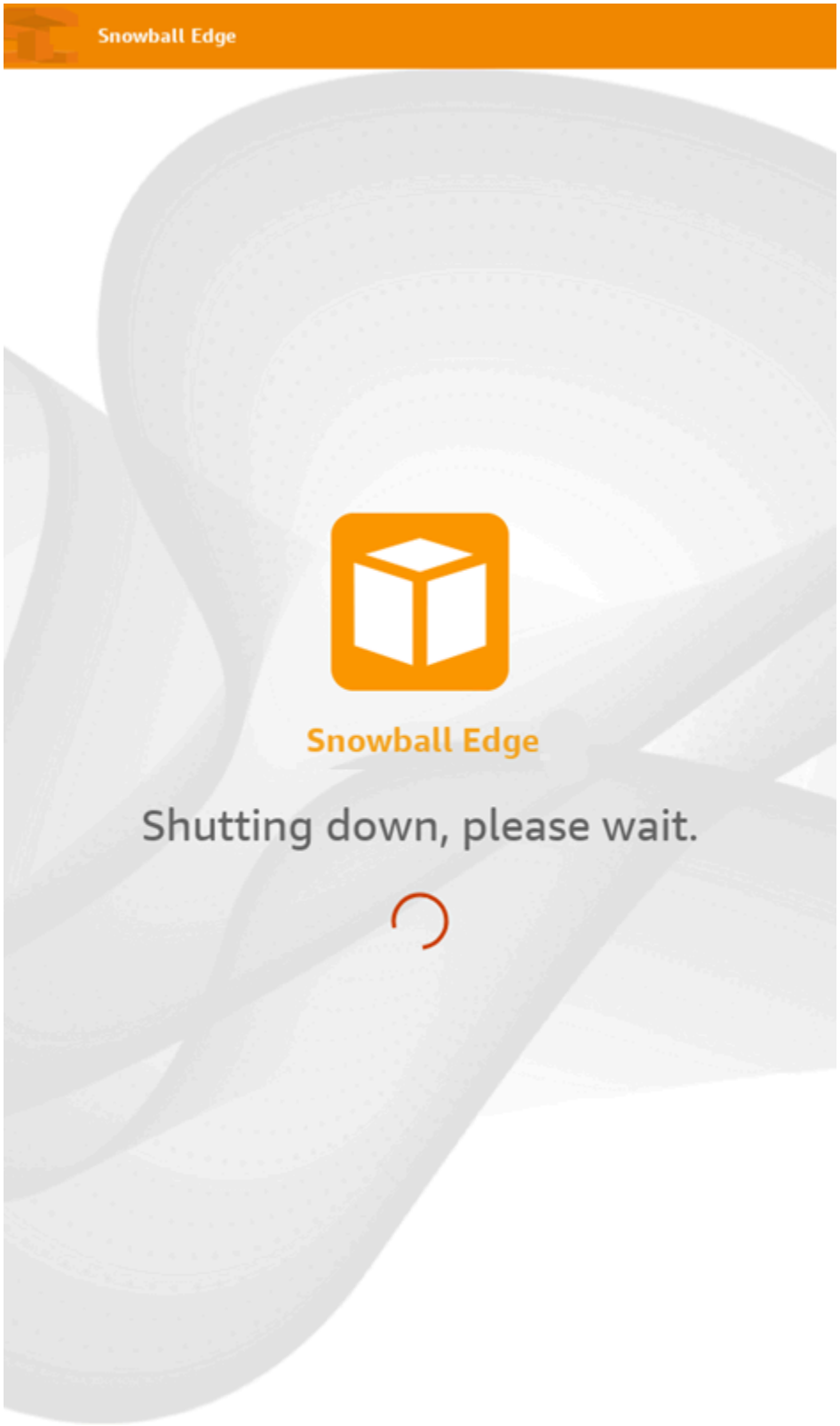
1. On the AWS OpsHub dashboard, find your device under **Devices**. Then choose the device to open the device details page.
2. Choose the **Device Power** menu, then choose **Reboot**. A dialog box appears.



3. In the dialog box, choose **Reboot**. Your device starts to reboot.



While the device shuts down, the LCD screen displays a message indicating the device is shutting down.


Snowball Edge
Snowball Edge
Shutting down, please wait.

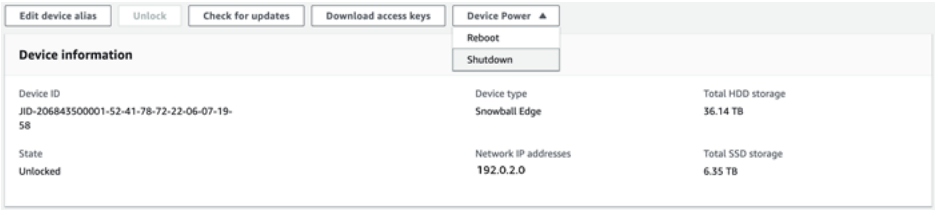
Shutting down your device

Follow these steps to use AWS OpsHub to shut down your Snow device.

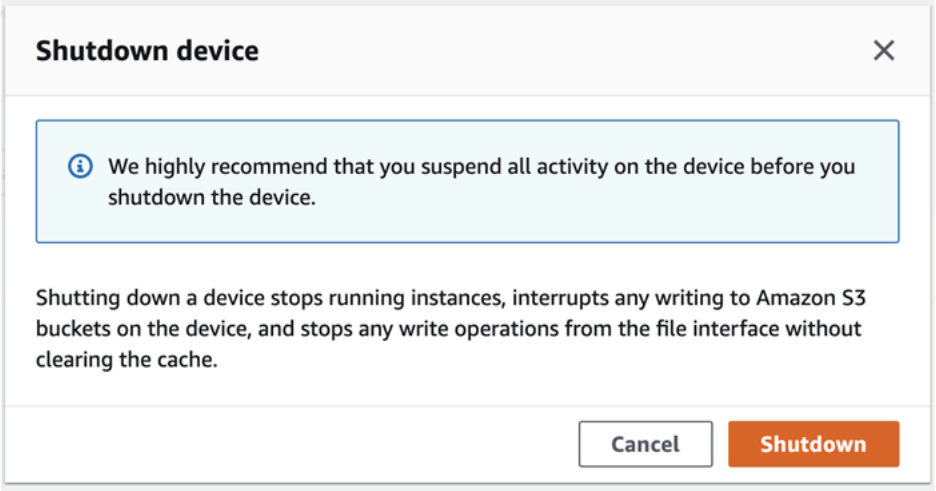
> **Important**
>
> We highly recommend that you suspend all activities on the device before you shut down the device. Shutting down a device stops running instances and interrupts any writing to Amazon S3 buckets on the device.

To shut down a device

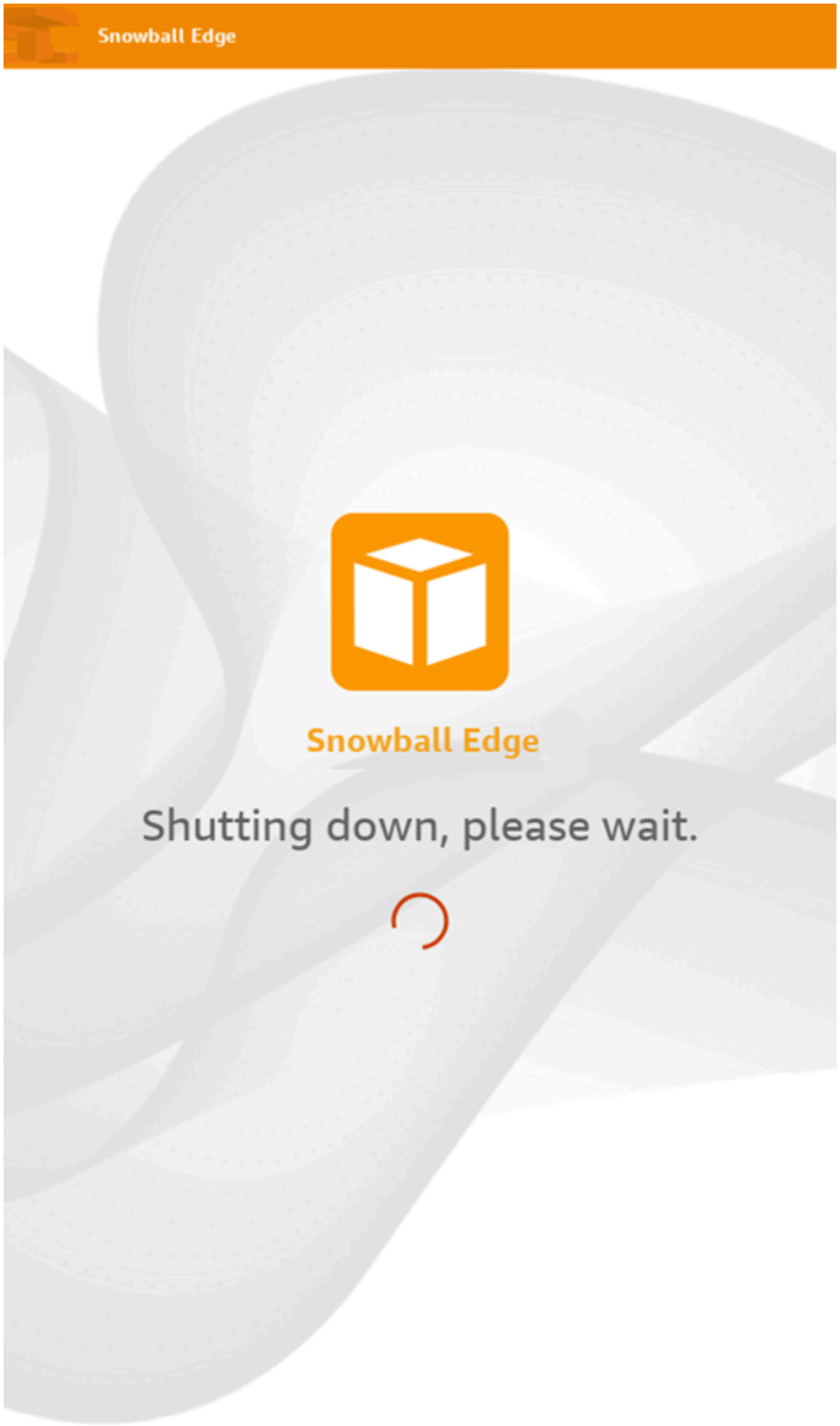
1. On the AWS OpsHub dashboard, find your device under **Devices**. Then choose the device to open the device details page.
2. Choose the **Device Power** menu, then choose **Shutdown**. A dialog box appears.



3. In the dialog box, choose **Shutdown**. Your device starts to shut down.



While the device shuts down, the LCD screen displays a message indicating the device is shutting down.

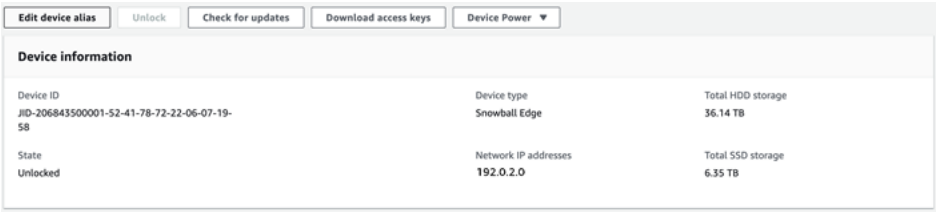

Snowball Edge
Snowball Edge
Shutting down, please wait.

Editing Your Device Alias

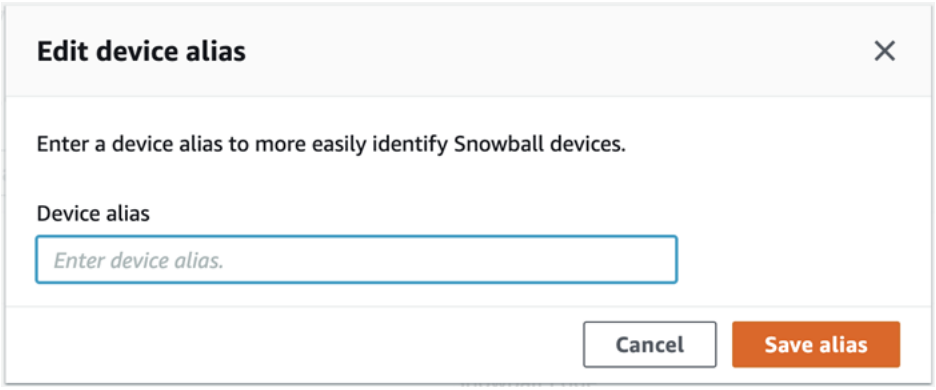
Use these steps to edit your device alias using AWS OpsHub.

To edit your device's alias

1. On the AWS OpsHub dashboard, find your device under **Devices**. Choose the device to open the device details page.
2. Choose the **Edit device alias** tab.



3. For **Device alias**, enter a new name, and choose **Save alias**.



Managing public key certificates using OpsHub

You can securely interact with AWS services running on a Snowball Edge device or a cluster of Snowball Edge devices through the HTTPS protocol by providing a public key certificate. You can use the HTTPS protocol to interact with AWS services such as IAM, Amazon EC2, S3 adapter, Amazon S3 compatible storage on Snow Family devices, Amazon EC2 Systems Manager, and AWS STS on Snowball Edge devices. In the case of a cluster of devices, a single certificate is required and can be generated by any device in the cluster. Once a Snowball Edge device generates the certificate and you unlock the device, you can use Snowball Edge client commands to list, get, and delete the certificate.

A Snowball Edge device generates a certificate when the following events occur:

- The Snowball Edge device or cluster is unlocked for the first time.
- The Snowball Edge device or cluster is unlocked after deleting the certificate (using the `delete-certificate` command or **Renew certificate** in AWS OpsHub).
- The Snowball Edge device or cluster is rebooted and unlocked after the certificate expires.

Whenever a new certificate is generated, the old certificate is no longer valid. A certificate is valid for a period of one year from the day it was generated.

You can also use the Snowball Edge client to manage public key certificates. For more information, see Managing public key certificates.

Topics

Download the public key certificate using OpsHub

You can download the active public key certificate to your computer.

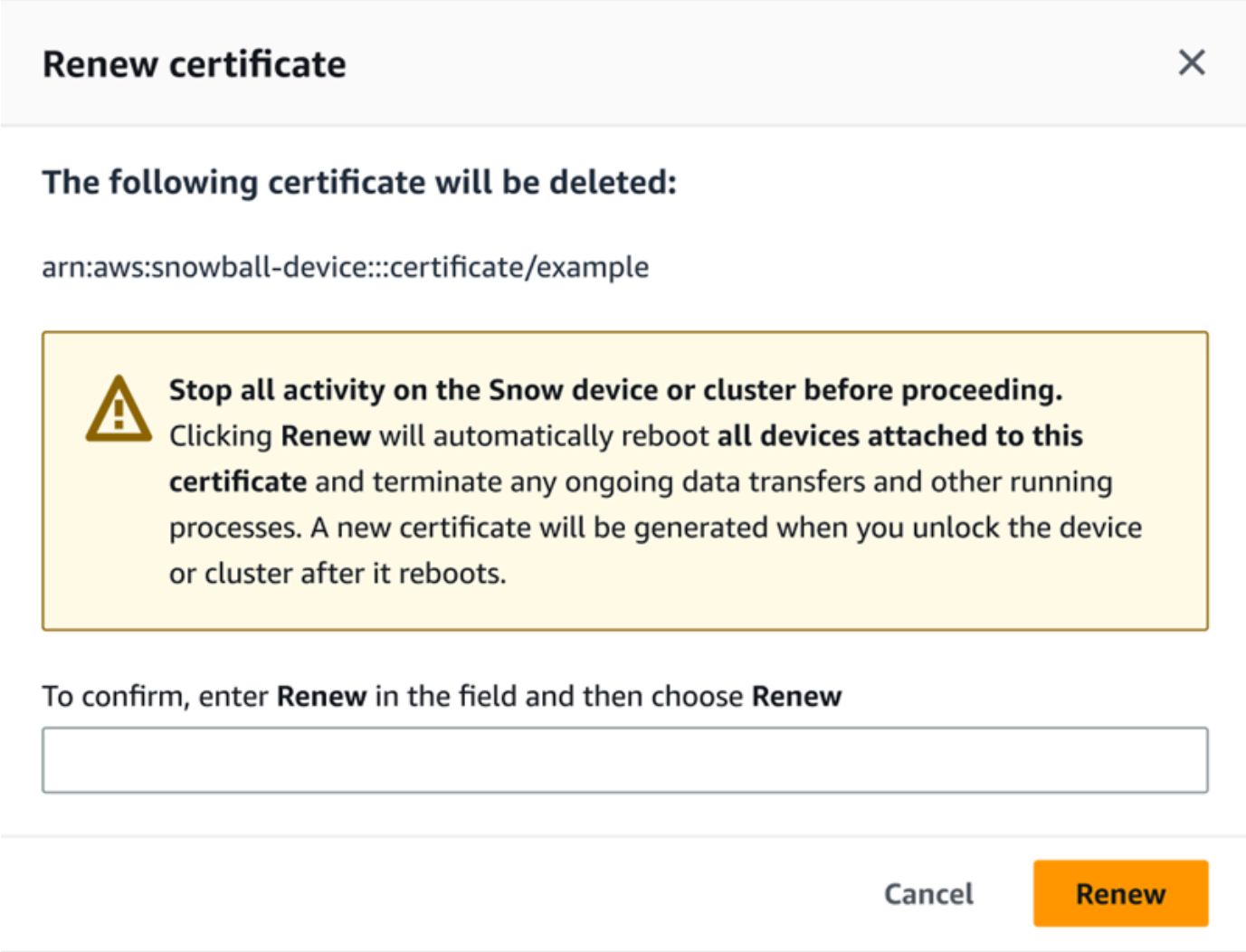
1. On the AWS OpsHub dashboard, find your device under **Devices**. Choose the device to open the device details page.
2. In the device details page, choose the **Manage certificate** menu. From the menu, choose **Download certificate**.
3. A window appears in which you can name the certificate file to download and choose the location on your computer where it will be downloaded. Choose **Save**.

Renewing the public key certificate using OpsHub

Before renewing the public key certificate, stop all data transfers to or from the Snow Family device and stop any EC2-compatible that are running. For more information, see Stopping an Amazon EC2-compatible instance in this guide.

1. On the AWS OpsHub dashboard, find your device under **Devices**. Choose the device to open the device details page.
2. In the device details page, choose the **Manage certificate** menu. From the menu, choose **Renew certificate**.

3. In the **Renew certificate** window, enter **Renew** in the field and choose **Renew**. The Snow Family device deletes the existing public key certificate and reboots the device or cluster.



Getting Updates for Your Device and the AWS OpsHub Application

You can check for updates for your device and install them. You can also configure AWS OpsHub to automatically update the application to the latest version.

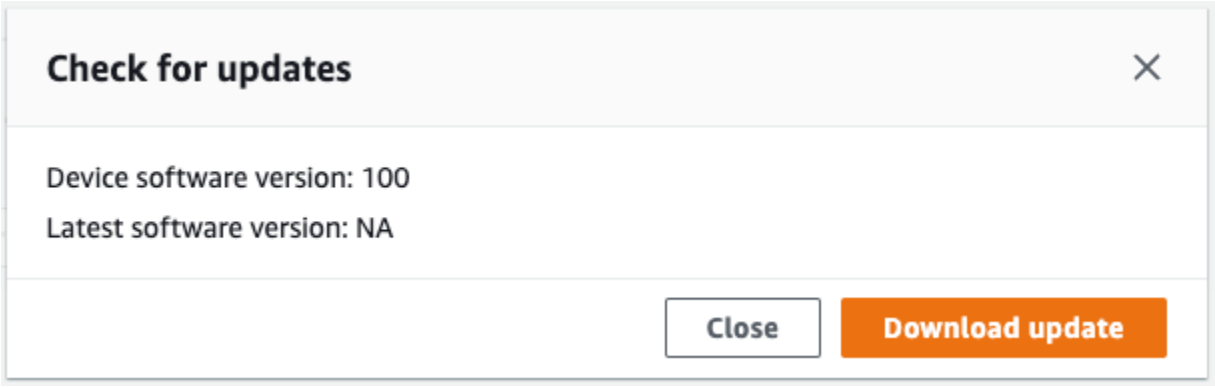
Updating your device

Follow these steps to use AWS OpsHub to update your Snow device.

To update your device

1. On the AWS OpsHub dashboard, find your device under **Devices**. Choose the device to open the device details page.
2. Choose the **Check for updates** tab.

   The **Check for updates** page displays the current software version on your device and the latest software version, if there is one.



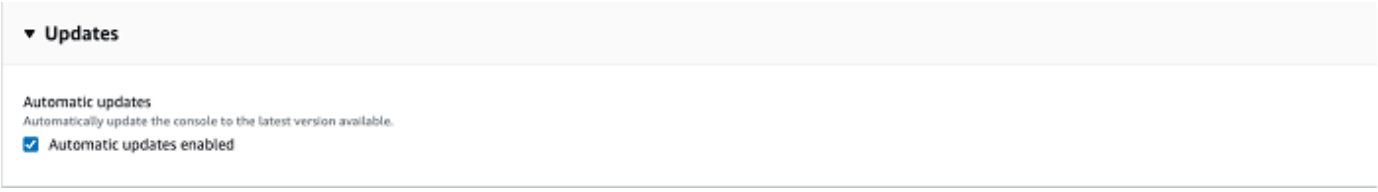
3. If there is an update, choose **Download update**. Otherwise, choose **Close**.

Updating your AWS OpsHub application

AWS OpsHub automatically updates the application to the latest version. Follow these steps to verify that automatic update is enabled.

To verify that automatic updates are enabled for AWS OpsHub

1. On the AWS OpsHub dashboard, choose **Preferences**.
2. Open the **Updates** tab.
3. Verify that **Automatic updates enabled** is selected. Automatic update is enabled by default.



   If **Automatic updates enabled** is not selected, you will not get the latest version of the AWS OpsHub application.

Managing profiles

You can create a *profile* for persistent storage of your credentials on your local file system. Using AWS OpsHub, you have the option to create a new profile any time you unlock the device using the device IP address, unlock code, and manifest file.

You can also use the Snowball Edge Client to create a profile at any time. See Configuring a Profile for the Snowball Edge Client.

To edit or delete profiles, edit the profile file in a text editor.

Example Example `snowball-edge.config` file

This example shows a profile file containing three profiles—`SnowDevice1profile`, `SnowDevice2profile`, and `SnowDevice3profile`.

```
{"version":1,"profiles":
    {
    "SnowDevice1profile":
        {
            "name":"SnowDevice1profile",
            "jobId":"JID12345678-136f-45b4-b5c2-847db8adc749",
            "unlockCode":"db223-12345-dbe46-44557-c7cc2",
            "manifestPath":"C:\\Users\\Administrator\\.aws\\ops-hub\\manifest\
\JID12345678-136f-45b4-b5c2-847db8adc749_manifest-1670622989203.bin",
            "defaultEndpoint":"https://10.16.0.1",
            "isCluster":false,
            "deviceIps":[]
        },
    },
    "SnowDevice2profile":
    {
        "name":"SnowDevice2profile",
        "jobId":"JID12345678-fdb2-436a-a4ff-7c510dec1bae",
        "unlockCode":"b893b-54321-0f65c-6c5e1-7f748",
        "manifestPath":"C:\\Users\\Administrator\\.aws\\ops-hub\\manifest\\JID12345678-
fdb2-436a-a4ff-7c510dec1bae_manifest-1670623746908.bin",
        "defaultEndpoint":"https://10.16.0.2",
        "isCluster":false,
        "deviceIps":[]
    },
```

```
    "SnowDevice3profile":
    {
        "name":"SnowDevice3profile",
        "jobId":"JID12345678-c384-4a5e-becd-ab5f38888463",
        "unlockCode":"64c89-13524-4d054-13d93-c1b80",
        "manifestPath":"C:\\Users\\Administrator\\.aws\\ops-hub\\manifest\\JID12345678-
c384-4a5e-becd-ab5f38888463_manifest-1670623999136.bin",
        "defaultEndpoint":"https://10.16.0.3",
        "isCluster":false,
        "deviceIps":[]
    }
}
```

To create a profile

1. Unlock your device locally and sign in according to the instructions in Unlocking a device.
2. Name the profile and choose **Save profile name**.

To edit a profile

1. In a text editor, open `snowball-edge.config` from `home directory\.aws\snowball \config`.
2. Edit the file as necessary. For example, to change the IP address of a device in the profile, change the `defaultEndpoint` entry.
3. Save and close the file.

To delete a profile

1. Using a text editor, open `snowball-edge.config` from `home directory\.aws\snowball \config`.
2. Delete the line that contains the profile name, the curly brackets { }that follow the profile name, and the contents within the those brackets.
3. Save and close the file.

Automating Your Management Tasks

You can use AWS OpsHub to automate operational tasks that you perform frequently on your Snow Family devices. You can create a task for reoccurring actions that you might want to perform on resources, such as restarting virtual servers, stopping Amazon EC2-compatible instances, and so on. You provide an automation document that safely performs operational tasks and runs the operation on AWS resources in bulk. You can also schedule common IT workflows.

> **Note**
>
> Automating tasks is not supported on clusters.
> To use tasks, the Amazon EC2 Systems Manager service must be started first. To start a service on your Snowball Edge, see Starting a Service on Your Snowball Edge.

Topics

Creating and Starting a Task

When you create a task, you specify the types of resources that the task should run on, and then provide a task document that contains the instructions that run the task. The task document is either in YAML or JSON format. You then provide the required parameters for the task and start the task.

To create a task

1. In the **Launch tasks** section of the dashboard, choose **Get started** to open the **Tasks** page. If you have created tasks, they appear under **Tasks**.
2. Choose **Create task** and provide details for the task.
3. For **Name**, enter a unique name for the task.

> **Tip**
>
> The name must be between 3 and 128 characters. Valid characters are `a-z`, `A-Z`, `0-9`, `.`, `_`, and `-`.

4. Optionally, you can choose a target type from the **Target type-optional** list. This is the type of resource that you want the task to run on.

   For example, you can specify **`/AWS::EC2::Instance`** for the tasks to run on an Amazon EC2-compatible instance or **`/`** to run on all resource types.
5. In the **Content** section, choose **YAML** or **JSON**, and provide the script that performs the task. You have two options, YAML or JSON format. For examples, see Task Examples.
6. Choose **Create**. The task that you created then appears on the **Tasks** page.

To start a task

1. In the **Launch tasks** section of the dashboard, choose **Get started** to open the **Tasks** page. Your tasks appear under **Tasks**.
2. Choose your task to open the **Start task** page.
3. Choose **Simple execution** to run on targets.

   Choose **Rate control** to run safely on multiple targets and define concurrency and error thresholds. For this option, you provide the additional target and error threshold information in the **Rate control** section.
4. Provide the required input parameters, and choose **Start task**.

   The status of the task is **Pending**, and changes to **Success** when the task has run successfully.

Task Examples

The following example restarts an Amazon EC2-compatible instance. It requires two input parameters: `endpoint` and `instance ID`.

YAML example

```
description: Restart EC2 instance
```

```
schemaVersion: '0.3'
parameters:
  Endpoint:
    type: String
    description: (Required) EC2 Service Endpoint URL
  Id:
    type: String
    description: (Required) Instance Id
mainSteps:
  - name: restartInstance
    action: aws:executeScript
    description: Restart EC2 instance step
    inputs:
      Runtime: python3.7
      Handler: restart_instance
      InputPayload:
        Endpoint: "{{ Endpoint }}"
        Id: "{{ Id }}"
      TimeoutSeconds: 30
      Script: |-
        import boto3
        import time
        def restart_instance(payload, context):
            ec2_endpoint = payload['Endpoint']
            instance_id = payload['Id']
            ec2 = boto3.resource('ec2', endpoint_url=ec2_endpoint)
            instance = ec2.Instance(instance_id)
            if instance.state['Name'] != 'stopped':
                instance.stop()
                instance.wait_until_stopped()
            instance.start()
            instance.wait_until_running()
            return {'InstanceState': instance.state}
```

JSON example

```
{
  "description" : "Restart EC2 instance",
  "schemaVersion" : "0.3",
  "parameters" : {
    "Endpoint" : {
      "type" : "String",
```

```
      "description" : "(Required) EC2 Service Endpoint URL"
    },
    "Id" : {
      "type" : "String",
      "description" : "(Required) Instance Id"
    }
  },
  "mainSteps" : [ {
    "name" : "restartInstance",
    "action" : "aws:executeScript",
    "description" : "Restart EC2 instance step",
    "inputs" : {
      "Runtime" : "python3.7",
      "Handler" : "restart_instance",
      "InputPayload" : {
        "Endpoint" : "{{ Endpoint }}",
        "Id" : "{{ Id }}"
      },
      "TimeoutSeconds" : 30,
      "Script" : "import boto3\nimport time\ndef restart_instance(payload, context):\n

            ec2_endpoint = payload['Endpoint']\n    instance_id = payload['Id']\n
            ec2 = boto3.resource('ec2', endpoint_url=ec2_endpoint)\n
            instance = ec2.Instance(instance_id)\n
            if instance.state['Name'] != 'stopped':\n
            instance.stop()\n
            instance.wait_until_stopped()\n
            instance.start()\n
            instance.wait_until_running()\n
            return {'InstanceState': instance.state}"
    }
  } ]
}
```

Viewing Details of a Task

You can view details of a management task, such as the description and the parameters that are required to run the task.

To view details of a task

1. In the **Launch tasks** section of the dashboard, choose **Get started** to open the **Tasks** page.
2. On the **Tasks** page, locate and choose the task that you want to see details of.

3. Choose **View details**, and choose one of the tabs to see the details. For example, the **Parameters** tab shows you the input parameters in the script.

Deleting a Task

Follow these steps to delete a management task.

To delete a task

1. In the **Launch tasks** section of the dashboard, choose **Get started** to open the **Tasks** page.
2. Locate the task that you want to delete. Choose the task, and then choose **Delete**.

Setting the NTP time servers for your device

Follow these steps to view and update which time servers your device must synchronize time with.

To check time sources

1. On the AWS OpsHub dashboard, find your device under **Devices**. Choose the device to open the device details page.
2. You will see a list of time sources that your device is synchronizing time with in the **Time sources** table.

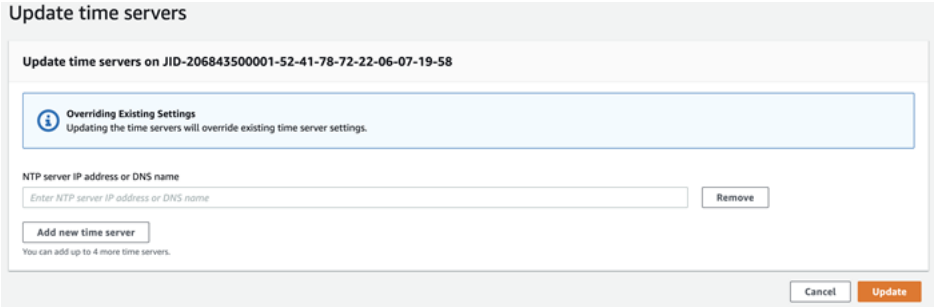
   The **Time sources** table has four columns:

   - **Address**: The DNS name / IP address of the time source
   - **State**: The current connection status between the device and that time source, there are 5 possible states:
     - **CURRENT**: Time source is currently being used to synchronize time
     - **COMBINED**: Time source is combined with the current source
     - **EXCLUDED**: Time source is excluded by the combining algorithm
     - **LOST**: Connection with the time source has been lost
     - **UNAVAILABILITY**: An invalid time source where the combining algorithm has deemed to be either a falseticker or has too much variability

- **Type**: Network Time Protocol (NTP) sources can be a server or peer. A server can be set by the user using the **update-time-server** command, whereas a peer can only be set up using other Snowball Edge devices in the cluster and are automatically set up when the cluster is associated.
- **Stratum**: The stratum of the source. **Stratum 1** indicates a source with a locally attached reference clock. A source that is synchronized to a Stratum 1 source is set at **Stratum 2**. A source that is synchronized to a stratum 2 source is set at **Stratum 3**, and so on.

To update the time servers

1. On the AWS OpsHub dashboard, find your device under **Devices**. Choose the device to open the device details page.
2. You will see a list of time sources that your device is synchronizing time with in the **Time sources** table.
3. Choose **Update time servers** on the **Time sources** table.
4. Provide the DNS name or the IP address of the time servers you would like your device to synchronize time with, and choose **Update**.



Supported NTP device types and software versions

NTP isn't available on any version 2 storage and compute device types. Snowball Edge version 3 storage and compute device types with software version 77 or later support NTP, however. To check if NTP is enabled, use the Snowball Edge CLI command `describe-time-sources`.

Using an AWS Snowball Edge Device

Following, you can find an overview of the AWS Snowball Edge device. Snowball Edge is a physically rugged device protected by AWS Key Management Service (AWS KMS) that you use for local storage and compute, or to transfer data between your on-premises servers and Amazon Simple Storage Service (Amazon S3).

For information about unlocking an AWS Snowball Edge device, see Using the Snowball Edge Client.

When the device first arrives, inspect it for damage or obvious tampering.

> **Warning**
>
> If you notice anything that looks suspicious about the device, don't connect it to your internal network. Instead, contact AWS Support, and a new one will be shipped to you.

The following image shows what the AWS Snowball Edge device looks like.

The device has three doors—a front, a back, and a top—that all can be opened by latches. The top door contains the power cable for the device. The other two doors can be opened and slid inside the device so that they're out of the way while you're using it. By opening the doors, you get access to the LCD E Ink display embedded in the front side of the device, and the power and network ports in the back.

After your device arrives and is powered on, you're ready to use it.

Topics

- Using the Snowball Edge Client
- Transferring files using the Amazon S3 adapter for data migration
- Managing the NFS interface
- Using an AWS Snowball Edge device with a Tape Gateway
- Using AWS IoT Greengrass to run pre-installed software on Amazon EC2-compatible instances

- Using AWS Lambda with an AWS Snowball Edge
- Using Amazon EC2-compatible compute instances
- Using Amazon S3 compatible storage on Snow Family devices
- Using Amazon EKS Anywhere on AWS Snow
- Using IAM Locally
- Using AWS Security Token Service
- Managing public key certificates
- Ports Required to Use AWS Services on an AWS Snowball Edge Device

Using the Snowball Edge Client

Following, you can find information about how to get and use the Snowball Edge client with your AWS Snowball Edge device. The Snowball Edge client is a standalone terminal application that you run on your local server to unlock the device and get credentials, logs, and status information. You can also use the client for administrative tasks for a cluster. While using the Snowball Edge client, you can get additional support information by running the `snowballEdge help` command.

When you read and write data to the AWS Snowball Edge device, you use the Amazon S3 adapter or the file interface.

Downloading and Installing the Snowball Edge Client

You can download and install the Snowball Edge client from AWS Snowball Edge Resources. On that page, you can find the installation package for your operating system. Follow the instructions to install the Snowball Edge client. Running the Snowball Edge client from a terminal in your workstation might require using a specific path, depending on your operating system:

- **Microsoft Windows** – When the client has been installed, you can run it from any directory without any additional preparation.
- **Linux** – The Snowball Edge client must be run from the `~/snowball-client-linux-`*`build_number`*`/bin/` directory. The Snowball Edge client is only supported on 64-bit Linux distributions.
- **macOS** – The `install.sh` script copies folders from the Snowball Edge client .tar file to the `/usr/local/bin/snowball` directory. If you run this script, you can then run the Snowball Edge

client from any directory if `/usr/local/bin` is a path in your `bash_profile`. You can verify your path using the `echo $PATH` command.

Commands for the Snowball Edge Client

Following, you can find information on the Snowball Edge client commands, including examples of use and sample outputs.

Topics

- Deleting a Direct Network Interface
- Checking feature status
- Setting Time Servers
- Checking Time Sources

Configuring a Profile for the Snowball Edge Client

Every time you run a command for the Snowball Edge client, you provide your manifest file, unlock code, and an IP address. You can get the first two of these from the AWS Snow Family Management Console or the job management API. For more information about getting your manifest and unlock code, see Getting credentials to access a Snow Family device.

You have the option of using the `snowballEdge configure` command to store the path to the manifest, the 29-character unlock code, and the endpoint as a profile. After configuration, you can use other Snowball Edge client commands without having to manually enter these values for a particular job. After you configure the Snowball Edge client, the information is saved in a plaintext JSON format to *`home directory`*`/.aws/snowball/config/snowball-edge.config`.

The endpoint is the IP address, with `https://` added to it. You can locate the IP address for the AWS Snowball Edge device on the AWS Snowball Edge device LCD display. When the AWS Snowball Edge device is connected to your network for the first time, it automatically gets a DHCP IP address, if a DHCP server is available. If you want to use a different IP address, you can change it from the LCD display. For more information, see Using an AWS Snowball Edge Device.

> **Important**
>
> Anyone who can access the configuration file can access the data on your Snowball Edge devices or clusters. Managing local access control for this file is one of your administrative responsibilities.

Usage

You can use this command in two ways: inline, or when prompted. This usage example shows the prompted method.

```
snowballEdge configure
```

Example Output

```
Configuration will stored at home directory\.aws\snowball\config\snowball-edge.config
Snowball Edge Manifest Path: /Path/to/manifest/file
Unlock Code: 29 character unlock code
Default Endpoint: https://192.0.2.0
```

You can have multiple profiles if you have multiple jobs at once, or if you want the option of managing a cluster from different endpoints. For more information about multiple AWS CLI profiles, see Named profiles in the *AWS Command Line Interface User Guide*.

Getting Your QR Code for NFC Validation

You can use this command to generate a device-specific QR code for use with the AWS Snowball Edge Verification App. For more information about NFC validation, see Validating NFC Tags.

Usage

```
snowballEdge get-app-qr-code --output-file ~/downloads/snowball-qr-code.png
```

Example Output

```
QR code is saved to ~/downloads/snowball-qr-code.png
```

Snowball Edge client version

Use the `version` command to see the version of the Snowball Edge command line interface (CLI) client.

Usage

```
snowballEdge version
```

Example output

```
Snowball Edge client version: 1.2.0  Build 661
```

Unlocking Snowball Edge Devices

To unlock a standalone AWS Snowball Edge device, run the `snowballEdge unlock-device` command. To unlock a cluster, use the `snowballEdge unlock-cluster` command. These commands authenticate your access to the AWS Snowball Edge device.

> **Note**
>
> To unlock the devices associated with your job, the devices must be on-site, plugged into power and the network, and turned on. In addition, the LCD display on the front of the AWS Snowball Edge device must indicate that the device is ready for use.

Usage

```
snowballEdge unlock-device --endpoint https://192.0.2.0 --manifest-file Path/to/
manifest/file --unlock-code 01234-abcde-ABCDE-01234
```

Example Single Device Unlock Input

```
snowballEdge unlock-device --endpoint https://192.0.2.0 --manifest-file /usr/home/
manifest.bin --unlock-code 01234-abcde-ABCDE-01234
```

Example Single Device Unlock Output

```
Your Snowball Edge device is unlocking. You may determine the unlock state of your
 device using the describe-device command. Your Snowball Edge device will be available
 for use when it is in the UNLOCKED state.
```

Cluster Usage

When you unlock a cluster, you provide the endpoint for one of your nodes, and all the IP addresses for the other devices in your cluster.

```
snowballEdge unlock-cluster --endpoint https://192.0.2.0 --manifest-file Path/to/
manifest/file --unlock-code 01234-abcde-ABCDE-01234 --device-ip-addresses 192.0.2.0
 192.0.2.1 192.0.2.2 192.0.2.3 192.0.2.4
```

Example Cluster Unlock Output

```
Your Snowball Edge Cluster is unlocking. You may determine the unlock state of
 your cluster using the describe-device command. Your Snowball Edge Cluster will be
 available for use when your Snowball Edge devices are in the UNLOCKED state.
```

Updating a Snowball Edge

Use the following commands to download and install updates for your Snowball Edge device. For procedures that use these commands, see Updating software on Snowball Edge devices.

`snowballEdge check-for-updates` – Returns version information about the Snowball Edge software available in the cloud, and the current version installed on the device.

Usage (configured Snowball Edge client)

```
snowballEdge check-for-updates
```

Example Output

```
Latest version: 102
Installed version: 101
```

`snowballEdge describe-device-software` – Returns the current software version and expiry date of the SSL certificate of the device. Additionally, if a software update is being downloaded or installed, the state is also displayed. Following is a list of possible outputs:

- NA – No software updates are currently in progress.
- `Downloading` – New software is being downloaded.
- `Installing` – New software is being installed.
- `Requires Reboot` – New software has been installed, and the device needs to be restarted.

> **Warning**
>
> We highly recommend that you suspend all activity on the device before you restart the device. Restarting a device stops running instances and interrupts any writing to Amazon S3 buckets on the device. All of these processes can result in lost data.

Usage (configured Snowball Edge client)

```
snowballEdge describe-device-software
```

Example Output

```
Installed version: 101
Installing version: 102
Install State: Downloading
CertificateExpiry: Thur Jan 01 00:00:00 UTC 1970
```

`snowballEdge download-updates` – Starts downloading the latest software updates for your Snowball Edge.

Usage (configured Snowball Edge client)

```
snowballEdge download-updates
```

Example Output

```
Download started. Run describe-device-software API for additional information.
```

`snowballEdge install-updates` – Starts installing the latest software updates for your Snowball Edge that were already downloaded.

Usage (configured Snowball Edge client)

```
snowballEdge install-updates
```

Example Output

```
Installation started.
```

`snowballEdge reboot-device` – Reboots the device.

> **Warning**
>
> We highly recommend that you suspend all activity on the device before you restart the device. Restarting a device stops running instances and interrupts any writing to Amazon S3 buckets on the device. All of these processes can result in lost data.

Usage (configured Snowball Edge client)

```
snowballEdge reboot-device
```

Example Output

```
Rebooting device now.
```

`snowballEdge configure-auto-update-strategies` – Configures an automatic update strategy.

Usage (configured Snowball Edge client)

```
snowballEdge configure-auto-update-strategy --auto-check autoCheck [--auto-check-
frequency
autoCheckFreq] --auto-download autoDownload
[--auto-download-frequency autoDownloadFreq]
--auto-install autoInstall
[--auto-install-frequency autoInstallFreq]
--auto-reboot autoReboot [--endpoint
endpoint]
```

Example Output

```
Successfully configured auto update strategy. Run describe-auto-update-strategies for
 additional information.
```

`snowballEdge describe-auto-update-strategies` – Returns any currently configured automatic update strategy.

Usage (configured Snowball Edge client)

```
snowballEdge describe-auto-update-strategies
```

Example Output

```
auto-update-strategy {[
auto-check:true,
auto-check-frequency: "0 0 * * FRI", // CRON Expression String, Every Friday at
 midnight
auto-download:true,
auto-download-frequency: "0 0 * * SAT", // CRON Expression String, Every Saturday at
midnight
auto-install:true,
auto-install-frequency: "0 13 * * Sun", // CRON Expression String, Every Saturday at
midnight
auto-reboot: false;
]}
```

Getting Credentials

Using the `snowballEdge list-access-keys` and `snowballEdge get-secret-access-key` commands, you can get the credentials of the admin user of your AWS account on Snowball Edge. You can use these credentials to create AWS Identity and Access Management (IAM users) and roles, and to authenticate your requests when using the AWS CLI or with an AWS SDK. These credentials are only associated with an individual job for Snowball Edge, and you can use them only on the device or cluster of devices. The device or devices don't have any IAM permissions in the AWS Cloud.

> **Note**
>
> If you're using the AWS CLI with the Snowball Edge, you must use these credentials when you configure the CLI. For information about configuring credentials for the AWS CLI, see Configuring the AWS CLI in the *AWS Command Line Interface User Guide*.

Usage (configured Snowball Edge client)

```
snowballEdge list-access-keys
```

Example Output

```
{
  "AccessKeyIds" : [ "AKIAIOSFODNN7EXAMPLE" ]
```

```
}
```

Usage (configured Snowball Edge client)

```
snowballEdge get-secret-access-key --access-key-id Access Key
```

Example Output

```
[snowballEdge]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Starting a Service on Your Snowball Edge

Snowball Edge devices support multiple services, in addition to Amazon S3. These include compute instances, the file interface, and AWS IoT Greengrass. Amazon S3 and Amazon EC2 are always on by default, and can't be stopped or restarted with the Snowball Edge client. However, the file interface and AWS IoT Greengrass can be started with the `snowballEdge start-service` command. To get the service ID for each service, you can use the `snowballEdge list-services` command.

Before you run this command, create a single virtual network interface to bind to the service that you're starting. For more information, see Creating a Virtual Network Interface.

Usage (configured Snowball Edge client)

```
snowballEdge start-service --service-id service_id --virtual-network-interface-
arns virtual-network-interface-arn
```

Example Output

```
Starting the AWS service on your Snowball Edge. You can determine the status of the AWS
 service using the describe-service command.
```

Stopping a Service on Your Snowball Edge

To stop a service running on your Snowball Edge, you can use the `snowballEdge stop-service` command.

The Amazon S3 adapter, Amazon EC2, AWS STS, and IAM services cannot be stopped.

> **Warning**
>
> Data loss can occur if the file interface is stopped before remaining buffered data is written to the device. For more information on using the file interface, see Managing the NFS interface.

> **Note**
>
> Stopping the Amazon S3 compatible storage on Snow Family devices service disables access to the data stored in your S3 buckets on the device or cluster. Access is restored when the Amazon S3 compatible storage on Snow Family devices is started again.
> For devices enabled with Amazon S3 compatible storage on Snow Family devices, it is recommended to start the service after the Snowball Edge device is powered up. See Setting up Snowball Edge in this guide.

Usage (configured Snowball Edge client)

```
snowballEdge stop-service --service-id service_id
```

Example Output

```
Stopping the AWS service on your Snowball Edge. You can determine the status of the AWS
 service using the describe-service command.
```

Starting NFS and Restricting Access

> **Important**
>
> Don't start the NFS service if you intend to use Amazon Elastic Block Store (Amazon EBS). The first time NFS is started, all storage is allocated to NFS. It is not possible to reallocate NFS storage to Amazon EBS, even if the NFS service is stopped.

> **Note**
>
> You can provide CIDR blocks for IP ranges that are allowed to mount the NFS shares exposed by the device. For example, `10.0.0.0/16`. If you don't provide allowed CIDR blocks, all mount requests will be denied.
> Be aware that data transferred through NFS is not encrypted in transit.
> Other than the allowed hosts by CIDR blocks, Snowcone doesn't provide an authentication or authorization mechanism for the NFS shares.

Start NFS with the `snowballEdge start-service` command. To get the service ID for the NFS service, you can use the `snowballEdge list-services` command.

Before you run this command, create a single virtual network interface to bind to the service that you're starting. For more information, see Creating a Virtual Network Interface. You can restrict access to your file shares and data in your Amazon S3 buckets and see what restrictions are currently in place. You do this by allocating CIDR blocks for allowed hosts that can access your file share and S3 buckets when you start the NFS service.

Usage (configured Snowball Edge client)

```
snowballEdge start-service --service-id nfs --virtual-network-interface-arns
 arn:aws:snowball-device:::interface/s.ni-12345fgh45678j --service-configuration
 AllowedHosts=ip address-1/32,ip address-2/24
```

Example Example Output

```
Starting the service on your Snowball Edge. You can determine the status of the service
 using the describe-service command.
```

Restricting Access to NFS Shares When NFS is Running

You can restrict access to your file shares and data in your Amazon S3 buckets after you have started NFS. You can see what restrictions are currently in place, and give each bucket different access restrictions. You do this by allocating CIDR blocks for hosts that can access your file share and S3 buckets when you start the NFS service. The following is an example command.

Usage (configured Snowball Edge client)

```
snowballEdge start-service \
```

```
    --service-id nfs \
    --virtual-network-interface-arns virtual-network-interface-arn --service-
configuration AllowedHosts=ip-address-1/32,ip-address-1/24
```

To see the current restrictions, use the `describe-service` command.

```
snowballEdge describe-service --service-id nfs
```

AWS Snowball Edge Logs

When you transfer data between your on-premises data center and a Snowball Edge, logs are automatically generated. If you encounter unexpected errors during data transfer to the device, you can use the following commands to save a copy of the logs to your local server.

There are three commands related to logs:

- `list-logs` – Returns a list of logs in JSON format. This list reports the size of the logs in bytes, the ARN for the logs, the service ID for the logs, and the type of logs.

  Usage (configured Snowball Edge client)

  ```
  snowballEdge list-logs
  ```

  Example Output

  ```
  {
    "Logs" : [ {
      "LogArn" : "arn:aws:snowball-device:::log/s3-storage-JIEXAMPLE2f-1234-4953-a7c4-
  dfEXAMPLE709",
      "LogType" : "SUPPORT",
      "ServiceId" : "s3",
      "EstimatedSizeBytes" : 53132614
    }, {
      "LogArn" : "arn:aws:snowball-device:::log/fileinterface-JIDEXAMPLEf-1234-4953-
  a7c4-dfEXAMPLE709",
      "LogType" : "CUSTOMER",
      "ServiceId" : "fileinterface",
      "EstimatedSizeBytes" : 4446
    }]
  }
  ```

- `get-log` – Downloads a copy of a specific log from the Snowball Edge to your server at a specified path. `CUSTOMER` logs are saved in the `.zip` format, and you can extract this type of log to view its contents. `SUPPORT` logs are encrypted and can only be read by AWS Support engineers. You have the option of specifying a name and a path for the log.

  Usage (configured Snowball Edge client)

  ```
  snowballEdge get-log --log-arn arn:aws:snowball-device:::log/fileinterface-
  JIDEXAMPLEf-1234-4953-a7c4-dfEXAMPLE709
  ```

  Example Output

  ```
  Logs are being saved to download/path/snowball-edge-logs-1515EXAMPLE88.bin
  ```

- `get-support-logs` – Downloads a copy of all the `SUPPORT` type of logs from the Snowball Edge to your service at a specified path.

  Usage (configured Snowball Edge client)

  Snowball Edge client

  ```
  snowballEdge get-support-logs
  ```

  Example Output

  ```
  Logs are being saved to download/path/snowball-edge-logs-1515716135711.bin
  ```

> **Important**
>
> `CUSTOMER` type might contain sensitive information about your own data. To protect this potentially sensitive information, we strongly suggest that you delete these logs once you're done with them.

Getting Device Status

You can determine the status and general health of your Snowball Edge devices with the following Snowball Edge client commands:

- `describe-device`

Usage (configured Snowball Edge client)

```
snowballEdge describe-device
```

Example Output

```
{
  "DeviceId" : "JID-EXAMPLE12345-123-456-7-890",
  "UnlockStatus" : {
    "State" : "UNLOCKED"
  },
  "ActiveNetworkInterface" : {
    "IpAddress" : "192.0.2.0"
  },
  "PhysicalNetworkInterfaces" : [ {
    "PhysicalNetworkInterfaceId" : "s.ni-EXAMPLEd9ecbf03e3",
    "PhysicalConnectorType" : "RJ45",
    "IpAddressAssignment" : "STATIC",
    "IpAddress" : "0.0.0.0",
    "Netmask" : "0.0.0.0",
    "DefaultGateway" : "192.0.2.1",
    "MacAddress" : "EX:AM:PL:E0:12:34"
  }, {
    "PhysicalNetworkInterfaceId" : "s.ni-EXAMPLE4c3840068f",
    "PhysicalConnectorType" : "QSFP",
    "IpAddressAssignment" : "STATIC",
    "IpAddress" : "0.0.0.0",
    "Netmask" : "0.0.0.0",
    "DefaultGateway" : "192.0.2.2",
    "MacAddress" : "EX:AM:PL:E0:56:78"
  }, {
    "PhysicalNetworkInterfaceId" : "s.ni-EXAMPLE0a3a6499fd",
    "PhysicalConnectorType" : "SFP_PLUS",
    "IpAddressAssignment" : "DHCP",
    "IpAddress" : "192.168.1.231",
    "Netmask" : "255.255.255.0",
    "DefaultGateway" : "192.0.2.3",
    "MacAddress" : "EX:AM:PL:E0:90:12"
  } ]
}
```

- `describe-cluster`

Usage (configured Snowball Edge client)

```
snowballEdge describe-cluster
```

Example Output

```
{
  "ClusterId" : "CIDEXAMPLE7-5402-4c19-9feb-7c9EXAMPLEd5",
  "Devices" : [ {
    "DeviceId" : "JIDEXAMPLE2-bc53-4618-a538-917EXAMPLE94",
    "UnlockStatus" : {
      "State" : "UNLOCKED"
    },
    "ActiveNetworkInterface" : {
      "IpAddress" : "192.0.2.0"
    },
    "ClusterAssociation" : {
      "State" : "ASSOCIATED",
      "ClusterId" : "CIDEXAMPLE7-5402-4c19-9feb-7c9EXAMPLEd5"
    },
    "NetworkReachability" : {
      "State" : "REACHABLE"
    }
  }, {
    "DeviceId" : "JIDEXAMPLE2-bc53-4618-a538-917EXAMPLE94",
    "UnlockStatus" : {
      "State" : "UNLOCKED"
    },
    "ActiveNetworkInterface" : {
      "IpAddress" : "192.0.2.1"
    },
    "ClusterAssociation" : {
      "State" : "ASSOCIATED",
      "ClusterId" : "CIDEXAMPLE7-5402-4c19-9feb-7c9EXAMPLEd5"
    },
    "NetworkReachability" : {
      "State" : "REACHABLE"
    }
  }, {
    "DeviceId" : "JIDEXAMPLE2-bc53-4618-a538-917EXAMPLE94",
    "UnlockStatus" : {
```

```
      "State" : "UNLOCKED"
    },
    "ActiveNetworkInterface" : {
      "IpAddress" : "192.0.2.2"
    },
    "ClusterAssociation" : {
      "State" : "ASSOCIATED",
      "ClusterId" : "CIDEXAMPLE7-5402-4c19-9feb-7c9EXAMPLEd5"
    },
    "NetworkReachability" : {
      "State" : "REACHABLE"
    }
  }, {
    "DeviceId" : "JIDEXAMPLE2-bc53-4618-a538-917EXAMPLE94",
    "UnlockStatus" : {
      "State" : "UNLOCKED"
    },
    "ActiveNetworkInterface" : {
      "IpAddress" : "192.0.2.3"
    },
    "ClusterAssociation" : {
      "State" : "ASSOCIATED",
      "ClusterId" : "CIDEXAMPLE7-5402-4c19-9feb-7c9EXAMPLEd5"
    },
    "NetworkReachability" : {
      "State" : "REACHABLE"
    }
  }, {
    "DeviceId" : "JIDEXAMPLE2-bc53-4618-a538-917EXAMPLE94",
    "UnlockStatus" : {
      "State" : "UNLOCKED"
    },
    "ActiveNetworkInterface" : {
      "IpAddress" : "192.0.2.4"
    },
    "ClusterAssociation" : {
      "State" : "ASSOCIATED",
      "ClusterId" : "CIDEXAMPLE7-5402-4c19-9feb-7c9EXAMPLEd5"
    },
    "NetworkReachability" : {
      "State" : "REACHABLE"
    }
  } ]
```

```
}
```

Getting Service Status

You can determine the status and general health of the services running on Snowball Edge devices with the `describe-service` command. You can first run the `list-services` command to see what services are running.

- `list-services`

  Usage (configured Snowball Edge client)

  ```
  snowballEdge list-services
  ```

  Example Output

  ```
  {
    "ServiceIds" : [ "greengrass", "fileinterface", "s3", "ec2", "s3-snow" ]
  }
  ```

- `describe-service`

  This command returns a status value for a service. It also includes state information that might be helpful in resolving issues you encounter with the service. Those states are as follows.

  - `ACTIVE` – The service is running and available for use.
  - `ACTIVATING` – The service is starting up, but it is not yet available for use.
  - `DEACTIVATING` – The service is in the process of shutting down.
  - `DEGRADED` – For Amazon S3 compatible storage on Snow Family devices, this status indicates one or more disks or devices in cluster is down. The Amazon S3 compatible storage on Snow Family devices service is running uninterrupted, but you should recover or replace the affected device before the cluster quorum is lost to minimize the risk of lost data. See Clustering overview in this guide.
  - `INACTIVE` – The service is not running and is not available for use.

  Usage (configured Snowball Edge client)

  ```
  snowballEdge describe-service --service-id service-id
  ```

Example Output

```
{
"ServiceId" : "s3",
  "Status" : {
    "State" : "ACTIVE"
  },
"Storage" : {
"TotalSpaceBytes" : 99608745492480,
"FreeSpaceBytes" : 99608744468480
},
"Endpoints" : [ {
"Protocol" : "http",
"Port" : 8080,
"Host" : "192.0.2.0"
}, {
"Protocol" : "https",
"Port" : 8443,
"Host" : "192.0.2.0",
"CertificateAssociation" : {
"CertificateArn" : "arn:aws:snowball-
device:::certificate/6d955EXAMPLEdb71798146EXAMPLE3f0"
}
} ]
}
```

Example Amazon S3 compatible storage on Snow Family devices service output

The `describe-service` command provides the following output for the **`s3-snow`** value of the `service-id` parameter.

```
{
  "ServiceId" : "s3-snow",
  "Autostart" : false,
  "Status" : {
    "State" : "ACTIVE"
  },
  "ServiceCapacities" : [ {
    "Name" : "S3 Storage",
    "Unit" : "Byte",
    "Used" : 640303104,
```

```
    "Available" : 219571981512
  } ],
  "Endpoints" : [ {
    "Protocol" : "https",
    "Port" : 443,
    "Host" : "10.0.2.123",
    "CertificateAssociation" : {
      "CertificateArn" : "arn:aws:snowball-device:::certificate/
a65ba817f2c5ac9683fc3bc1ae123456"
    },
    "Description" : "s3-snow bucket API endpoint",
    "DeviceId" : "JID6ebd4c50-c3a1-4b16-b32c-b254f9b7f2dc",
    "Status" : {
      "State" : "ACTIVE"
    }
  }, {
    "Protocol" : "https",
    "Port" : 443,
    "Host" : "10.0.3.202",
    "CertificateAssociation" : {
      "CertificateArn" : "arn:aws:snowball-device:::certificate/
a65ba817f2c5ac9683fc3bc1ae123456"
    },
    "Description" : "s3-snow object API endpoint",
    "DeviceId" : "JID6ebd4c50-c3a1-4b16-b32c-b254f9b7f2dc",
    "Status" : {
      "State" : "ACTIVE"
    }
  }, {
    "Protocol" : "https",
    "Port" : 443,
    "Host" : "10.0.3.63",
    "CertificateAssociation" : {
      "CertificateArn" : "arn:aws:snowball-device:::certificate/
a65ba817f2c5ac9683fc3bc1ae123456"
    },
    "Description" : "s3-snow bucket API endpoint",
    "DeviceId" : "JID2a1e0deb-38b1-41f8-b904-a396c62da70d",
    "Status" : {
      "State" : "ACTIVE"
    }
  }, {
    "Protocol" : "https",
    "Port" : 443,
```

```
    "Host" : "10.0.2.243",
    "CertificateAssociation" : {
      "CertificateArn" : "arn:aws:snowball-device:::certificate/
a65ba817f2c5ac9683fc3bc1ae123456"
    },
    "Description" : "s3-snow object API endpoint",
    "DeviceId" : "JID2a1e0deb-38b1-41f8-b904-a396c62da70d",
    "Status" : {
      "State" : "ACTIVE"
    }
  }, {
    "Protocol" : "https",
    "Port" : 443,
    "Host" : "10.0.2.220",
    "CertificateAssociation" : {
      "CertificateArn" : "arn:aws:snowball-device:::certificate/
a65ba817f2c5ac9683fc3bc1ae123456"
    },
    "Description" : "s3-snow bucket API endpoint",
    "DeviceId" : "JIDcc45fa8f-b994-4ada-a821-581bc35d8645",
    "Status" : {
      "State" : "ACTIVE"
    }
  }, {
    "Protocol" : "https",
    "Port" : 443,
    "Host" : "10.0.2.55",
    "CertificateAssociation" : {
      "CertificateArn" : "arn:aws:snowball-device:::certificate/
a65ba817f2c5ac9683fc3bc1ae123456"
    },
    "Description" : "s3-snow object API endpoint",
    "DeviceId" : "JIDcc45fa8f-b994-4ada-a821-581bc35d8645",
    "Status" : {
      "State" : "ACTIVE"
    }
  }, {
    "Protocol" : "https",
    "Port" : 443,
    "Host" : "10.0.3.213",
    "CertificateAssociation" : {
      "CertificateArn" : "arn:aws:snowball-device:::certificate/
a65ba817f2c5ac9683fc3bc1ae123456"
    },
```

```
    "Description" : "s3-snow bucket API endpoint",
    "DeviceId" : "JID4ec68543-d974-465f-b81d-89832dd502db",
    "Status" : {
      "State" : "ACTIVE"
    }
  }, {
    "Protocol" : "https",
    "Port" : 443,
    "Host" : "10.0.3.144",
    "CertificateAssociation" : {
      "CertificateArn" : "arn:aws:snowball-device:::certificate/
a65ba817f2c5ac9683fc3bc1ae123456"
    },
    "Description" : "s3-snow object API endpoint",
    "DeviceId" : "JID4ec68543-d974-465f-b81d-89832dd502db",
    "Status" : {
      "State" : "ACTIVE"
    }
  }, {
    "Protocol" : "https",
    "Port" : 443,
    "Host" : "10.0.2.143",
    "CertificateAssociation" : {
      "CertificateArn" : "arn:aws:snowball-device:::certificate/
a65ba817f2c5ac9683fc3bc1ae123456"
    },
    "Description" : "s3-snow bucket API endpoint",
    "DeviceId" : "JID6331b8b5-6c63-4e01-b3ca-eab48b5628d2",
    "Status" : {
      "State" : "ACTIVE"
    }
  }, {
    "Protocol" : "https",
    "Port" : 443,
    "Host" : "10.0.3.224",
    "CertificateAssociation" : {
      "CertificateArn" : "arn:aws:snowball-device:::certificate/
a65ba817f2c5ac9683fc3bc1ae123456"
    },
    "Description" : "s3-snow object API endpoint",
    "DeviceId" : "JID6331b8b5-6c63-4e01-b3ca-eab48b5628d2",
    "Status" : {
      "State" : "ACTIVE"
    }
```

```
  } ]
}
```

Removing a Node from a Cluster

The `disassociate-device` command removes a node from a Snowball Edge cluster. If you want to replace an unhealthy node, use this command. For more information about clusters, see Custering overview in this guide.

> **Important**
>
> Use the `disassociate-device` command only when you are removing an unhealthy node. This command fails and returns an error if you try to remove a healthy node.

Don't use this command to remove a node that was accidentally powered off or disconnected from the network and is therefore temporarily unavailable to the rest of the cluster. Nodes removed with this command can't be added to any cluster, and must be returned to AWS.

If a node was accidentally powered off or disconnected from the network, plug the node back into power and the network, and use the `associate-device` command. You can't use the `disassociate-device` command to disassociate a node if it's powered on and healthy.

Usage (configured Snowball Edge client)

```
snowballEdge disassociate-device --device-id Job ID for the Device
```

Example Output

```
Disassociating your Snowball Edge device from the cluster. Your Snowball Edge device
 will be disassociated from the cluster when it is in the "DISASSOCIATED" state. You
 can use the describe-cluster command to determine the state of your cluster.
```

Adding a Node to a Cluster

The `associate-device` command adds a node to a cluster of Snowball Edge devices. If you power off a node, it reverts from being unlocked to being locked. To unlock that node, you can use this command. Use this command to replace an unavailable node with a new node that you

ordered as a replacement. For more information about clusters, see Clustering overview in this guide.

Usage (configured Snowball Edge client)

```
snowballEdge associate-device --device-ip-address IP Address
```

Example Output

```
Associating your Snowball Edge device with the cluster. Your Snowball Edge device will
 be associated with the cluster when it is in the ASSOCIATED state. You can use the
 describe-cluster command to determine the state of your cluster.
```

Creating Tags for Your Device

Adds or overwrites the specified tags on your device. You can create a maximum of 50 tags. Each tag consists of a key-value pair. The value is optional.

> **Note**
>
> Don't put sensitive data in your tags.

Usage (configured Snowball Edge client)

```
snowballEdge create-tags --tag Key=Name,Value=user-test --tag Key=Stage,Value=beta
```

For more information, run the `describe-tags` command.

Example Output

```
Tag(s) [Key=Name,Value=test, Key=Stage,Value=beta] created.
```

Deleting Tags from Your Device

The `delete-tags` command deletes the specified tags from your Snowball Edge device.

Usage (configured Snowball Edge client)

```
snowballEdge delete-tags --tag Key=Stage,Value=beta
```

```
    Tag(s) [Key=Stage,Value=beta] deleted.
```

For more information, run the `describe-tags` command.

> **Note**
>
> If you want to delete multiple tags at the same time, you can specify multiple key-value pairs, like the following:
> `delete-tags --tag Key=Name,Value=test --tag Key=Stage,Value=Beta`
> If you specify a tag key without a tag value, any tag with this key regardless of its value is deleted. If you specify a tag key with an empty string as the tag value, only tags that have an empty string as the value are deleted.

Describing Tags on Your Device

The `describe-tags` command describes the tags on your Snowball Edge device.

Usage (configured Snowball Edge client)

```
snowballEdge describe-tags
```

For more information, run the `describe-tags` command.

Example Output

```
{
  "Tags" : [ {
    "Key" : "Name",
    "Value" : "user-test"
  }, {
    "Key" : "Stage",
    "Value" : "beta"
  } ]
}
```

Creating a Direct Network Interface

- `create-direct-network-interface` – Creates a direct network interface (DNI). Creates a direct network interface to use with Amazon EC2-compatible compute instances on your device.

You can find the direct network interfaces available on your device by using the `describe-direct-network-interfaces` command.

Usage (configured Snowball Edge client)

```
snowballEdge create-direct-network-interface [--endpoint endpoint] [--instance-
id instanceId] [--mac macAddress]
                                [--manifest-file manifestFile] [--physical-network-
interface-id physicalNetworkInterfaceId]
                                [--profile profile] [--unlock-code unlockCode] [--
vlan vlanId]
```

Getting Information About a Direct Network Interface

- `describe-direct-network-interface` – Gets the direct network interfaces on your device. A direct network interface can be used to configure networking for Amazon EC2-compatible compute instances and services on your device. You can create a new direct network interface by using the `create-direct-network-interface` command.

Usage (configured Snowball Edge client)

```
snowballEdge describe-direct-network-interfaces [--endpoint endpoint] [--manifest-
file manifestFile] [--profile profile] [--unlock-code unlockCode]
```

Updating a Direct Network Interface

- `update-direct-network-interface` – Updates a direct network interface. Use this command to update a direct network interface that will be used with Amazon EC2-compatible compute instances on the device. You can find the direct network interfaces that are available on your device by using the `describe-direct-network`-interfaces command. When you are modifying a network interface that is attached to an Amazon EC2-compatible instance, the interface will first be detached.

Usage (configured Snowball Edge client)

```
snowballEdge update-direct-network-interface [--direct-network-interface-
arn directNetworkInterfaceArn] [--endpoint endpoint]
                                [--mac macAddress]
```

```
                                [--manifest-file manifestFile] [--profile profile] [--
unlock-code unlockCode]
                                [--vlan vlanId] [--attach-instance-id instanceId | --
detach]
```

Deleting a Direct Network Interface

- `delete-direct-network-interface` – Deletes a direct network interface that is no longer in use. To delete a direct network interface associated with your Amazon EC2-compatible compute instance, you must first disassociate the direct network interface from your instance.

Usage (configured Snowball Edge client)

```
snowballEdge delete-direct-network-interface [--direct-network-interface-
arn directNetworkInterfaceArn] [--endpoint endpoint]
                                [--manifest-file manifestFile] [--profile profile] [--
unlock-code unlockCode]
```

Checking feature status

To list the status of features available on your device use the `describe-features` command.

`RemoteManagementState` indicates the status of Snow Device Management and returns one of the following states:

- `INSTALLED_ONLY` – The feature is installed but not enabled.
- `INSTALLED_AUTOSTART` – The feature is enabled and the device will attempt to connect to its AWS Region when it is powered on.
- `NOT_INSTALLED` – The device does not support the feature or was already in the field before its launch.

Usage (configured Snowball Edge client)

```
snowballEdge describe-features \
  --manifest-file manifest.bin path \
  --unlock-code unlock-code \
  --endpoint https://device-local-ip:9091
```

Example Output

```
{
  "RemoteManagementState" : String
}
```

Setting Time Servers

You can set up an external Network Time Protocol (NTP) server. You can use the NTP CLI commands when the device is in both locked and unlocked states. The manifest and unlock code are required. You can set these either with the `snowballEdge configure` command or by using the `--manifest-file` and `--unlock-code` options. Note that you can use the `snowballEdge` CLI on both AWS Snowcone Edge and AWS Snowcone.

It is your responsibility to provide a secure NTP time server. To set which NTP time servers the device connects to, use the `update-time-servers` CLI command.

> **Note**
>
> The `update-time-servers` command will override the previous NTP time servers settings.

Supported NTP device types and software versions

NTP isn't available on any version 2 storage and compute device types. Snowball Edge version 3 storage and compute device types with software version 77 or later support NTP, however. To check if NTP is enabled, use the Snowball Edge CLI command `describe-time-sources`.

Usage

```
snowballEdge update-time-servers time.google.com
```

Example Example Output

```
Updating time servers now.
```

Checking Time Sources

To see which NTP time sources the device are currently connected to, use the `describe-time-sources` Snowball Edge CLI command.

Usage

```
snowballEdge describe-time-sources
```

Example Example Output

```
{
  "Sources" : [ {
    "Address" : "172.31.2.71",
    "State" : "LOST",
    "Type" : "PEER",
    "Stratum" : 10
  }, {
    "Address" : "172.31.3.203",
    "State" : "LOST",
    "Type" : "PEER",
    "Stratum" : 10
  }, {
    "Address" : "172.31.0.178",
    "State" : "LOST",
    "Type" : "PEER",
    "Stratum" : 10
  }, {
    "Address" : "172.31.3.178",
    "State" : "LOST",
    "Type" : "PEER",
    "Stratum" : 10
  }, {
    "Address" : "216.239.35.12",
    "State" : "CURRENT",
    "Type" : "SERVER",
    "Stratum" : 1
  } ]
}
```

The `describe-time-sources` command returns a list of time source states. Each time source state contains the `Address`, `State`, `Type`, and `Stratum` fields. Following are the meanings of these fields.

- `Address` – The DNS name / IP address of the time source.
- `State` – The current connection status between the device and that time source. There are five possible states:.
  - `CURRENT` – The time source is currently being used to synchronize time.
  - `COMBINED` – The time source is combined with the current source.
  - `EXCLUDED` – The time source is excluded by the combining algorithm.
  - `LOST` – The connection with the time source has been lost.
  - `UNACCEPTABLE` – An invalid time source where the combining algorithm has deemed to be either a falseticker or has too much variability.
- `Type` – An NTP time source can be either a server or a peer. Servers can be set by the `update-time-servers` command. Peers can only be other Snowball Edge devices in the cluster and are automatically set up when the cluster is associated.
- `Stratum` – This field shows the stratum of the source. Stratum 1 indicates a source with a locally attached reference clock. A source that is synchronized to a stratum 1 source is at stratum 2. A source that is synchronized to a stratum 2 source is at stratum 3, and so on..

An NTP time source can either be a server or a peer. A server can be set by the user with the `update-time-servers` command, whereas a peer could only be other Snowball Edge devices in the cluster. In the example output, `describe-time-sources` is called on a Snowball Edge that is in a cluster of 5. The output contains 4 peers and 1 server. The peers have a stratum of 10 while the server has a stratum of 1; therefore, the server is selected to be the current time source.

Transferring files using the Amazon S3 adapter for data migration

Following is an overview of the Amazon S3 adapter, which you can use to transfer data programmatically to and from S3 buckets already on the AWS Snowball Edge device using Amazon S3 REST API actions. This Amazon S3 REST API support is limited to a subset of actions. You can use this subset of actions with one of the AWS SDKs to transfer data programmatically. You can

also use the subset of supported AWS Command Line Interface (AWS CLI) commands for Amazon S3 to transfer data programmatically.

If your solution uses the AWS SDK for Java version 1.11.0 or newer, you must use the following `S3ClientOptions`:

- `disableChunkedEncoding()` – Indicates that chunked encoding is not supported with the interface.
- `setPathStyleAccess(true)` – Configures the interface to use path-style access for all requests.

For more information, see Class S3ClientOptions.Builder in the *Amazon AppStream SDK for Java*.

Important

We recommend that you use only one method at a time to read and write data to a local bucket on an AWS Snowball Edge device. Using both the file interface and the Amazon S3 adapter on the same bucket at the same time can result in read/write conflicts.
AWS Snowball Edge Quotas details the limits.
For AWS services to work properly on a Snowball Edge, you must allow the ports for the services. For details, see Ports Required to Use AWS Services on an AWS Snowball Edge Device.

Topics

Downloading and installing the AWS CLI version 1.16.14 for use with the Amazon S3 adapter

Currently, Snowball Edge devices support only version 1.16.14 and earlier of the AWS CLI for use with the Amazon S3 adapter. Newer versions of the AWS CLI are not compatible with the Amazon S3 adapter because they do not support all of the functionality of the S3 adapter.

> **Note**
>
> If you are using Amazon S3 compatible storage on Snow Family devices, you can use the latest version of the AWS CLI. To download and use the latest version, see AWS Command Line Interface User Guide.

Install the AWS CLI on Linux operating systems

Run this chained command:

```
curl "https://s3.amazonaws.com/aws-cli/awscli-bundle-1.16.14.zip" -o "awscli-
bundle.zip";unzip awscli-bundle.zip;sudo ./awscli-bundle/install -i /usr/local/aws -b /
usr/local/bin/aws;/usr/local/bin/aws --version;
```

Install the AWS CLI on Windows operating systems

Download and run the installer file for your operating system:

- 32-bit
- 64-bit

Using the AWS CLI and API operations on Snowball Edge devices

When using the AWS CLI or API operations to issue IAM, Amazon S3, and Amazon EC2 commands on Snowball Edge, you must specify the Region as "snow." You can do this using `AWS configure` or within the command itself, as in the following examples.

```
aws configure --profile abc
```

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: 1234567
Default region name [None]: snow
Default output format [None]: json
```

Or

```
aws s3 ls --profile snowballEdge --endpoint http://192.0.2.0:8080 --region snow
```

Authorization with the Amazon S3 API interface for AWS Snowball

When you use the Amazon S3 adapter, every interaction is signed with the AWS Signature Version 4 algorithm by default. This authorization is used only to verify the data traveling from its source to the interface. All encryption and decryption happens on the device. Unencrypted data is never stored on the device.

When using the interface, keep the following in mind:

- To get the local Amazon S3 credentials to sign your requests to the AWS Snowball Edge device, run the `snowballEdge list-access-keys` and `snowballEdge get-secret-access-keys` Snowball Edge client commands. For more information, see Using the Snowball Edge Client. These local Amazon S3 credentials include a pair of keys: an access key and a secret key. These keys are only valid for the devices associated with your job. They can't be used in the AWS Cloud because they have no AWS Identity and Access Management (IAM) counterpart.
- The encryption key is not changed by what AWS credentials you use. Signing with the Signature Version 4 algorithm is only used to verify the data traveling from its source to the interface. Thus, this signing never factors into the encryption keys used to encrypt your data on the Snowball.

Getting and using local Amazon S3 credentials

Every interaction with a Snowball Edge is signed with the AWS Signature Version 4 algorithm. For more information about the algorithm, see Signature Version 4 Signing Process in the *AWS General Reference*.

You can get the local Amazon S3 credentials to sign your requests to the Snowball Edge client Edge device by running the `snowballEdge list-access-keys` and `snowballEdge get-secret-`

`access-key` Snowball Edge client information, see Getting Credentials. These local Amazon S3 credentials include a pair of keys: an access key ID and a secret key. These credentials are only valid for the devices that are associated with your job. They can't be used in the AWS Cloud because they have no IAM counterpart.

You can add these credentials to the AWS credentials file on your server. The default credential profiles file is typically located at `~/.aws/credentials`, but the location can vary per platform. This file is shared by many of the AWS SDKs and by the AWS CLI. You can save local credentials with a profile name as in the following example.

```
[snowballEdge]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Specifying the S3 adapter as the AWS CLI endpoint

When you use the AWS CLI to issue a command to the AWS Snowball Edge device, you specify that the endpoint is the Amazon S3 adapter. You have the choice of using the HTTPS endpoint, or an unsecured HTTP endpoint, as shown following.

HTTPS secured endpoint

```
aws s3 ls --profile snowballEdge --endpoint https://192.0.2.0:8443 --ca-bundle path/to/certificate
```

HTTP unsecured endpoint

```
aws s3 ls --profile snowballEdge --endpoint http://192.0.2.0:8080
```

If you use the HTTPS endpoint of 8443, your data is securely transferred from your server to the Snowball Edge. This encryption is ensured with a certificate that's generated by the Snowball Edge when it gets a new IP address. After you have your certificate, you can save it to a local `ca-bundle.pem` file. Then you can configure your AWS CLI profile to include the path to your certificate, as described following.

To associate your certificate with the interface endpoint

1. Connect the Snowball Edge to power and the network, and turn it on.

2. After the device has finished booting up, make a note of its IP address on your local network.
3. From a terminal on your network, make sure you can ping the Snowball Edge.
4. Run the `snowballEdge get-certificate` command in your terminal. For more information on this command, see Managing public key certificates.
5. Save the output of the `snowballEdge get-certificate` command to a file, for example `ca-bundle.pem`.
6. Run the following command from your terminal.

```
aws configure set profile.snowballEdge.ca_bundle /path/to/ca-bundle.pem
```

After you complete the procedure, you can run CLI commands with these local credentials, your certificate, and your specified endpoint, as in the following example.

```
aws s3 ls --profile snowballEdge --endpoint https://192.0.2.0:8443
```

Unsupported Amazon S3 features for the Amazon S3 adapter

Using the Amazon S3 adapter, you can programmatically transfer data to and from a Snowball Edge with Amazon S3 API actions. However, not all Amazon S3 transfer features and API actions are supported for use with a Snowball Edge device when using the Amazon S3 adapter. For example, the following features and actions are not supported for use with Snowball Edge:

- TransferManager – This utility transfers files from a local environment to Amazon S3 with the SDK for Java. Consider using the supported API actions or AWS CLI commands with the interface instead.
- GET Bucket (List Objects) Version 2 – This implementation of the GET action returns some or all (up to 1,000) of the objects in a bucket. Consider using the GET Bucket (List Objects) Version 1 action or the ls AWS CLI command.
- ListBuckets – The ListBuckets with the object endpoint is not supported. The following command does not work with Amazon S3 compatible storage on Snow Family devices:

```
aws s3 ls --endpoint https://192.0.2.0 --profile profile
```

Batching small files

Each copy operation has some overhead because of encryption. To speed up the process of transferring small files to your AWS Snowball Edge device, you can batch them together in a single archive. When you batch files together, they can be auto-extracted when they are imported into Amazon S3, if they were batched in one of the supported archive formats.

Typically, files that are 1 MB or smaller should be included in batches. There's no hard limit on the number of files you can have in a batch, though we recommend that you limit your batches to about 10,000 files. Having more than 100,000 files in a batch can affect how quickly those files import into Amazon S3 after you return the device. We recommend that the total size of each batch be no larger than 100 GB.

Batching files is a manual process, which you manage. After you batch your files, transfer them to a Snowball Edge device using the AWS CLI `cp` command with the `--metadata snowball-auto-extract=true` option. Specifying `snowball-auto-extract=true` automatically extracts the contents of the archived files when the data is imported into Amazon S3, so long as the size of the batched file is no larger than 100 GB.

> **Note**
>
> Any batches larger than 100 GB are not extracted when they are imported into Amazon S3.

To batch small files

1. Decide on what format you want to batch your small files in. The auto-extract feature supports the TAR, `ZIP`, and `tar.gz` formats.
2. Identify which small files you want to batch together, including their size and the total number of files that you want to batch together.
3. Batch your files on the command line as shown in the following examples.

   - For Linux, you can batch the files in the same command line used to transfer your files to the device.

   ```
   tar -cf - /Logs/April | aws s3 cp - s3://mybucket/batch01.tar --metadata
    snowball-auto-extract=true --endpoint http://192.0.2.0:8080
   ```

> **Note**
>
> Alternatively, you can use the archive utility of your choice to batch the files into one or more large archives. However, this approach requires extra local storage to save the archives before you transfer them to the Snowball.

- For Windows, use the following example command to batch the files when all files are in the same directory from which the command is run:

```
7z a -tzip -so "test" | aws s3 cp - s3://mybucket/batch01.zip --metadata
 snowball-auto-extract=true --endpoint http://192.0.2.0:8080
```

  To batch files from a different directory from which the command is run, use the following example command:

```
7z a -tzip -so "test" "c:\temp" | aws s3 cp - s3://mybucket/batch01.zip --
metadata snowball-auto-extract=true --endpoint http://10.x.x.x:8080
```

> **Note**
>
> For Microsoft Windows 2016, tar is not available, but you can download it from the *Tar for Windows* website.
> You can download 7 ZIP from the 7ZIP website.

4. Repeat until you've archived all the small files that you want to transfer to Amazon S3 using a Snowball Edge.
5. Transfer the archived files to the Snowball. If you want the data to be auto-extracted, and you used one of the supported archive formats mentioned previously in step 1, use the AWS CLI `cp` command with the `--metadata snowball-auto-extract=true` option.

> **Note**
>
> If there are non-archive files, don't use this command.

When creating the archive files, the extraction will maintain the current data structure. This means if you create an archive file that contains files and folders, Snowball Edge will recreate this during the ingestion to Amazon S3 process.

The archive file will be extracted in the same directory it is stored in and the folder structures will be built out accordingly. Keep in mind that when copying archive files, it is important to set the flag `--metadata snowball-auto-extract=true`. Otherwise, Snowball Edge will not extract the data when it's imported into Amazon S3.

Using the example in step 3, if you have the folder structure of /Logs/April/ that contains files `a.txt`, `b.txt` and `c.txt`. If this archive file was placed in the root of /mybucket/ then the data would look like the following after the extraction:

```
/mybucket/Logs/April/a.txt
/mybucket/Logs/April/b.txt
/mybucket/Logs/April/c.txt
```

If the archive file was placed into /mybucket/Test/, then the extraction would look as like the following:

```
/mybucket/Test/Logs/April/a.txt
/mybucket/Test/Logs/April/b.txt
/mybucket/Test/Logs/April/c.txt
```

Supported AWS CLI commands

Following, you can find information about how to specify the Amazon S3 adapter or Amazon S3 compatible storage on Snow Family devices as the endpoint for applicable AWS Command Line Interface (AWS CLI) commands. You can also find the list of AWS CLI commands for Amazon S3 that are supported for transferring data to the AWS Snowball Edge device with the adapter or Amazon S3 compatible storage on Snow Family devices.

> **Note**
>
> For information about installing and setting up the AWS CLI, including specifying what Regions you want to make AWS CLI calls against, see AWS Command Line Interface User Guide.

Currently, Snowball Edge devices support only version 1.16.14 and earlier of the AWS CLI when using the Amazon S3 adapter. See Snowball Edge client version. If you are using Amazon S3 compatible storage on Snow Family devices, you can use the lastest version of the AWS CLI. To download and use the latest version, see AWS Command Line Interface User Guide.

> **Note**
>
> Be sure to install version 2.6.5+ or 3.4+ of Python before you install version 1.16.14 of the AWS CLI.

Supported AWS CLI commands for Amazon S3

Following is a description of the subset of AWS CLI commands and options for Amazon S3 that the AWS Snowball Edge device supports. If a command or option isn't listed, it's not supported. You can declare some unsupported options, like `--sse` or `--storage-class`, along with a command. However, these are ignored and have no impact on how data is imported.

- cp – Copies a file or object to or from the AWS Snowball Edge device. The following are options for this command:
  - `--dryrun` (Boolean) – The operations that would be performed using the specified command are displayed without being run.
  - `--quiet` (Boolean) – Operations performed by the specified command are not displayed.
  - `--include` (string) – Don't exclude files or objects in the command that match the specified pattern. For details, see Use of Exclude and Include Filters in the *AWS CLI Command Reference*.
  - `--exclude` (string) – Exclude all files or objects from the command that matches the specified pattern.
  - `--follow-symlinks | --no-follow-symlinks` (Boolean) – Symbolic links (symlinks) are followed only when uploading to Amazon S3 from the local file system. Amazon S3 doesn't support symbolic links, so the contents of the link target are uploaded under the name of the link. When neither option is specified, the default is to follow symlinks.
  - `--only-show-errors` (Boolean) – Only errors and warnings are displayed. All other output is suppressed.
  - `--recursive` (Boolean) – The command is performed on all files or objects under the specified directory or prefix.

- `--page-size` (integer) – The number of results to return in each response to a list operation. The default value is 1000 (the maximum allowed). Using a lower value might help if an operation times out.
- `--metadata` (map) – A map of metadata to store with the objects in Amazon S3. This map is applied to every object that is part of this request. In a sync, this functionality means that files that haven't changed don't receive the new metadata. When copying between two Amazon S3 locations, the `metadata-directive` argument defaults to `REPLACE` unless otherwise specified.

- ls – Lists objects on the AWS Snowball Edge device. The following are options for this command:
  - `--human-readable` (Boolean) – File sizes are displayed in human-readable format.
  - `--summarize` (Boolean) – Summary information is displayed. This information is the number of objects and their total size.
  - `--recursive` (Boolean) – The command is performed on all files or objects under the specified directory or prefix.
  - `--page-size` (integer) – The number of results to return in each response to a list operation. The default value is 1000 (the maximum allowed). Using a lower value might help if an operation times out.
- rm – Deletes an object on the AWS Snowball Edge device. The following are options for this command:
  - `--dryrun` (Boolean) – The operations that would be performed using the specified command are displayed without being run.
  - `--include` (string) – Don't exclude files or objects in the command that match the specified pattern. For details, see Use of Exclude and Include Filters in the *AWS CLI Command Reference*.
  - `--exclude` (string) – Exclude all files or objects from the command that matches the specified pattern.
  - `--recursive` (Boolean) – The command is performed on all files or objects under the specified directory or prefix.
  - `--page-size` (integer) – The number of results to return in each response to a list operation. The default value is 1000 (the maximum allowed). Using a lower value might help if an operation times out.
  - `--only-show-errors` (Boolean) – Only errors and warnings are displayed. All other output is suppressed.
  - `--quiet` (Boolean) – Operations performed by the specified command are not displayed.

- sync – Syncs directories and prefixes. This command copies new and updated files from the source directory to the destination. This command only creates directories in the destination if they contain one or more files.

  > **Important**
  >
  > Syncing from one directory to another directory on the same Snowball Edge isn't supported.
  > Syncing from one AWS Snowball device to another AWS Snowball device isn't supported. You can only use this option to sync the contents between your on-premises data storage and a Snowball Edge.

  - `--dryrun` (Boolean) – The operations that would be performed using the specified command are displayed without being run.
  - `--quiet` (Boolean) – Operations performed by the specified command are not displayed.
  - `--include` (string) – Don't exclude files or objects in the command that match the specified pattern. For details, see Use of Exclude and Include Filters in the *AWS CLI Command Reference*.
  - `--exclude` (string) – Exclude all files or objects from the command that matches the specified pattern.
  - `--follow-symlinks` or `--no-follow-symlinks` (Boolean) – Symbolic links (symlinks) are followed only when uploading to Amazon S3 from the local file system. Amazon S3 doesn't support symbolic links, so the contents of the link target are uploaded under the name of the link. When neither option is specified, the default is to follow symlinks.
  - `--only-show-errors` (Boolean) – Only errors and warnings are displayed. All other output is suppressed.
  - `--no-progress` (Boolean) – File transfer progress is not displayed. This option is only applied when the `--quiet` and `--only-show-errors` options are not provided.
  - `--page-size` (integer) – The number of results to return in each response to a list operation. The default value is 1000 (the maximum allowed). Using a lower value might help if an operation times out.
  - `--metadata` (map) – A map of metadata to store with the objects in Amazon S3. This map is applied to every object that is part of this request. In a sync, this functionality means that files that haven't changed don't receive the new metadata. When copying between two Amazon

S3 locations, the `metadata-directive` argument defaults to REPLACE unless otherwise specified.

> **Important**
>
> Syncing from one directory to another directory on the same Snowball Edge isn't supported.
> Syncing from one AWS Snowball device to another AWS Snowball device isn't supported.
> You can only use this option to sync the contents between your on-premises data storage and a Snowball Edge.

- `--size-only` (Boolean) – With this option, the size of each key is the only criterion used to decide whether to sync from source to destination.
- `--exact-timestamps` (Boolean) – When syncing from Amazon S3 to local storage, same-sized items are ignored only when the timestamps match exactly. The default behavior is to ignore same-sized items unless the local version is newer than the Amazon S3 version.
- `--delete` (Boolean) – Files that exist in the destination but not in the source are deleted during sync.

You can work with files or folders with spaces in their names, such as `my photo.jpg` or `My Documents`. However, make sure that you handle the spaces properly in the AWS CLI commands. For more information, see Specifying parameter values for the AWS CLI in the *AWS Command Line Interface User Guide*.

Supported REST API actions

Following, you can find REST API actions that you can use with an AWS Snowball Edge device and Amazon S3.

Topics

- Supported REST API actions for Snowball Edge devices
- Supported REST API actions for the Amazon S3 adapter

Supported REST API actions for Snowball Edge devices

HEAD Snowball Edge

Description

Currently, there's only one Snowball Edge REST API operation, which you can use to return status information for a specific device. This operation returns the status of a Snowball Edge. This status includes information that can be used by AWS Support for troubleshooting purposes.

You can't use this operation with the AWS SDKs or the AWS CLI. We recommend that you use `curl` or an HTTP client. The request doesn't need to be signed for this operation.

Request

In the following example, the IP address for the Snowball Edge is *`192.0.2.0`*. Replace this value with the IP address of your actual device.

```
curl -X HEAD http://192.0.2.0:8080
```

Response

```
<Status xsi:schemaLocation="http://s3.amazonaws.com/doc/2006-03-01/" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
    <snowballIp>127.0.0.1</snowballIp>
    <snowballPort>8080</snowballPort>
    <snowballId>device-id</snowballId>
    <totalSpaceInBytes>499055067136</totalSpaceInBytes>
    <freeSpaceInBytes>108367699968</freeSpaceInBytes>
    <jobId>job-id</jobId>
    <snowballServerVersion>1.0.1</snowballServerVersion>
    <snowballServerBuild>DevBuild</snowballServerBuild>
    <snowballClientVersion>Version 1.0</snowballClientVersion>
    <snowballRoundTripLatencyInMillis>33</snowballRoundTripLatencyInMillis>
</Status>
```

Supported REST API actions for the Amazon S3 adapter

Following, you can find the list of Amazon S3 REST API actions that are supported for using the Amazon S3 adapter. The list includes links to information about how the API actions work with

Amazon S3. The list also covers any differences in behavior between the Amazon S3 API action and the AWS Snowball Edge device counterpart. All responses coming back from an AWS Snowball Edge device declare `Server` as `AWSSnowball`, as in the following example.

```
HTTP/1.1 201 OK
x-amz-id-2: JuKZqmXuiwFeDQxhD7M8KtsKobSzWA1QEjLbTMTagkKdBX2z7Il/jGhDeJ3j6s80
x-amz-request-id: 32FE2CEB32F5EE25
Date: Fri, 08 2016 21:34:56 GMT
Server: AWSSnowball
```

Amazon S3 REST API calls require SigV4 signing. If you're using the AWS CLI or an AWS SDK to make these API calls, the SigV4 signing is handled for you. Otherwise, you need to implement your own SigV4 signing solution. For more information, see Authenticating requests (AWS Signature Version 4) in the *Amazon Simple Storage Service User Guide.*

- GET Bucket (List Objects) version 1 – Supported. However, in this implemetation of the GET operation, the following is not supported:
  - Pagination
  - Markers
  - Delimiters
  - When the list is returned, the list is not sorted

  Only version 1 is supported. GET Bucket (List Objects) version 2 is not supported.
- GET Service
- HEAD Bucket
- HEAD Object
- GET Object – is a DOWNLOAD of an object from the Snow device's S3 bucket.
- PUT Object – When an object is uploaded to an AWS Snowball Edge device using `PUT Object`, an ETag is generated.

  The ETag is a hash of the object. The ETag reflects changes only to the contents of an object, not its metadata. The ETag might or might not be an MD5 digest of the object data. For more information about ETags, see Common Response Headers in the *Amazon Simple Storage Service API Reference.*
- DELETE Object

- Initiate Multipart Upload – In this implementation, initiating a multipart upload request for an object already on the AWS Snowball Edge device first deletes that object. It then copies it in parts to the AWS Snowball Edge device.
- List Multipart Uploads
- Upload Part
- Complete Multipart Upload
- Abort Multipart Upload

> **Note**
>
> Any Amazon S3 adapter REST API actions not listed here are not supported. Using any unsupported REST API actions with your Snowball Edge returns an error message saying that the action is not supported.

Managing the NFS interface

Use the Network File System (NFS) interface to upload files to the Snow Family device as if the device is local storage to your operating system. This allows for a more user-friendly approach to transferring data because you can use features of your operating system, like copying files, dragging and dropping them, or other graphical user interface features. Each S3 bucket on the device is available as an NFS interface endpoint and can be mounted to copy data to. The NFS interface is available for import jobs.

You can use the NFS interface if the Snowball Edge device was configured to include it when the job to order the device was created. If the device is not configured to include the NFS interface, use the S3 adapter or Amazon S3 compatible storage on Snow Family devices to transfer data. For more information about the S3 adapter, see Managing Amazon S3 adapter storage. For more information about Amazon S3 compatible storage on Snow Family devices, see Set up Amazon S3 compatible storage on Snow Family devices.

When started, the NFS interface uses 1 GB of memory and 1 CPU. This may limit the number of other services running on the Snow Family device or the number of EC2-compatible instances that can run.

Data transferred through the NFS interface is not encrypted in transit. When configuring the NFS interface, you can provide CIDR blocks and the Snow Family device will restrict access to the NFS interface from client computers with addresses in those blocks.

Files on the device will be transferred to Amazon S3 when it is returned to AWS. For more information, see Importing Jobs into Amazon S3.

For more information about using NFS with your computer operating system, see the documentation for your operating system.

Keep the following details in mind when using the NFS interface.

- File names are object keys in your local S3 bucket on the Snow Family device. The key name is a sequence of Unicode characters whose UTF-8 encoding is at most 1,024 bytes long. We recommend using NFSv4.1 where possible and encode file names with Unicode UTF-8 to ensure a successful data import. File names that are not encoded with UTF-8 might not be uploaded to S3 or might be uploaded to S3 with a different file name depending on the NFS encoding you use.
- Ensure that the maximum length of your file path is less than 1024 characters. Snow Family devices do not support file paths that are greater that 1024 characters. Exceeding this file path length will result in file import errors.
- For more information, see Object keys in the Amazon Simple Storage Service User Guide.
- For NFS based transfers, standard POSIX style meta-data will be added to your objects as they get imported to Amazon S3 from Snow Family devices. In addition, you will see meta-data "x-amz-meta-user-agent aws-datasync" as we currently use AWS DataSync as part of the internal import mechanism to Amazon S3 for Snow Family device import with NFS option.
- You can transfer up to 40M files using a single Snowball Edge device. If you require to transfer more than 40M files in a single job, please batch the files in order to reduce the file numbers per each transfer. Individual files can be of any size with a maximum file size of 5 TB for Snowball Edge devices with the enhanced NFS interface or the S3 interface.

You can also configure and manage the NFS interface with AWS OpsHub, a GUI tool. For more information, see Managing the NFS interface.

NFS configuration for Snow Family devices

The NFS interface is not running on the Snow Family device by default, so you need to start it to enable data transfer to the device. You can configure the NFS interface by providing the IP address

of a Virtual Network Interface (VNI) running on the Snow Family device and restricting access to your file share, if required. Before configuring the NFS interface, set up a virtual network interface (VNI) on your Snow Family device. For more information, see Network Configuration for Compute Instances.

Configure Snow Family devices for the NFS interface

- Use the `describe-service` command to determine if the NFS interface is active.

```
snowballEdge describe-service --service-id nfs
```

The command will return the state of the NFS service, `ACTIVE` or `INACTIVE`.

```
{
  "ServiceId" : "nfs",
  "Status" : {
  "State" : "ACTIVE"
  }
}
```

If the value of the `State` name is `ACTIVE`, the NFS interface service is active and you can mount the Snow Family device NFS volume. For more information, see

After the NFS interface is started, mount the endpoint as local storage on client computers.

The following are the default mount commands for Windows, Linux, and macOS operating systems.

- Windows:

```
mount -o nolock rsize=128 wsize=128 mtype=hard nfs-interface-ip-address:/
buckets/BucketName *
```

- Linux:

```
mount -t nfs nfs-interface-ip-address:/buckets/BucketName mount_point
```

- macOS:

```
mount -t nfs -o vers=3,rsize=131072,wsize=131072,nolocks,hard,retrans=2 nfs-interface-ip-address:/buckets/$bucketname mount_point
```

. If the value is `INACTIVE`, you have to start the service.

Starting the NFS service on the Snow Family device

Start a virtual network interface (VNI), if necessary, then start the NFS service on the Snow Family device. If necessary, when starting the NFS service, provide a block of allowed network addresses. If you don't provide any addresses, access to the NFS endpoints will be unrestricted.

1. Use the `describe-virtual-network-interface` command to see the VNIs available on the Snow Family device.

   ```
   snowballEdge describe-virtual-network-interfaces
   ```

   If one or more VNIs are active on the Snow Family device, the command returns the following.

   ```
   snowballEdge describe-virtual-network-interfaces
   [
     {
       "VirtualNetworkInterfaceArn" : "arn:aws:snowball-device:::interface/
   s.ni-8EXAMPLE8EXAMPLE8",
       "PhysicalNetworkInterfaceId" : "s.ni-8EXAMPLEaEXAMPLEd",
       "IpAddressAssignment" : "DHCP",
       "IpAddress" : "192.0.2.0",
       "Netmask" : "255.255.255.0",
       "DefaultGateway" : "192.0.2.1",
       "MacAddress" : "EX:AM:PL:E1:23:45"
   ```

```
  },{
    "VirtualNetworkInterfaceArn" : "arn:aws:snowball-device:::interface/
s.ni-1EXAMPLE1EXAMPLE1",
    "PhysicalNetworkInterfaceId" : "s.ni-8EXAMPLEaEXAMPLEd",
    "IpAddressAssignment" : "DHCP",
    "IpAddress" : "192.0.2.2",
    "Netmask" : "255.255.255.0",
    "DefaultGateway" : "192.0.2.1",
    "MacAddress" : "12:34:5E:XA:MP:LE"
  }
]
```

Note the value of the `VirtualNetworkInterfaceArn` name of the VNI to use with the NFS interface.

2. If no VNIs are available, use the `create-virtual-network-interface` command to create a VNI for the NFS interface. For more information, see Setting up a Virtual Network Interface (VNI).
3. Use the `start-service` command to start the NFS service and associate it with the VNI. To restrict access to the NFS interface, include the `service-configuration` and `AllowedHosts` parameters in the command.

```
snowballEdge start-service --virtual-network-interface-arns arn-of-vni --service-id
 nfs --service-configuration AllowedHosts=CIDR-address-range
```

4. Use the `describe-service` command to check the service status. It is running when the value of the `State` name is `ACTIVE`.

```
snowballEdge describe-service --service-id nfs
```

The command returns the service state, as well as the IP address and port number of the NFS endpoint and the CIDR ranges allowed to access the endpoint.

```
{
 "ServiceId" : "nfs",
```

```
 "Status" : {
 "State" : "ACTIVE"
 },
 "Endpoints" : [ {
 "Protocol" : "nfs",
 "Port" : 2049,
 "Host" : "192.0.2.0"
 } ],
 "ServiceConfiguration" : {
 "AllowedHosts" : [ "10.24.34.0/23", "198.51.100.0/24" ]
 }
}
```

Mounting NFS endpoints on client computers

After the NFS interface is started, mount the endpoint as local storage on client computers.

The following are the default mount commands for Windows, Linux, and macOS operating systems.

- Windows:

```
mount -o nolock rsize=128 wsize=128 mtype=hard nfs-interface-ip-address:/
buckets/BucketName *
```

- Linux:

```
mount -t nfs nfs-interface-ip-address:/buckets/BucketName mount_point
```

- macOS:

```
mount -t nfs -o vers=3,rsize=131072,wsize=131072,nolocks,hard,retrans=2 nfs-
interface-ip-address:/buckets/$bucketname mount_point
```

Stopping the NFS interface

When you are finished transferring files through the NFS interface and before powering off the Snow Family device, use the `stop-service` command to stop the NFS service.

```
snowballEdge stop-service --service-id nfs
```

Using an AWS Snowball Edge device with a Tape Gateway

Using an AWS Snowball Edge device with a Tape Gateway provides a secure, offline solution for migrating your tape data. A Snowball Edge device with a Tape Gateway lets you to migrate petabytes of data stored on physical tapes to AWS without changing your existing tape-based backup workflows, and without extreme network infrastructure or bandwidth-usage requirements. A standard Tape Gateway temporarily stores your tape data in the gateway cache and uses your network connection to transfer the data asynchronously to the AWS Cloud. However, a Snowball Edge device with a Tape Gateway stores your tape data on the device itself until you return it to AWS, and uses your network connection only for device-management traffic.

A Tape Gateway is a type of AWS Storage Gateway—a virtual appliance that emulates a physical tape library and works with your existing backup software to help you transfer data into the AWS Cloud. After you receive your Snowball Edge device, you unlock it, set up a Tape Gateway on it, copy your tape data to it, and then ship it back to AWS. AWS then stores your tape data in secure, durable, and low-cost Amazon S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage. With this combination of technologies, you can migrate your tape data to AWS in situations where you have network-connectivity limitations, bandwidth constraints, or high connection costs. Moving tape data to AWS helps you decrease your physical-tape infrastructure expenses and gain online access to your tape-based data at any time.

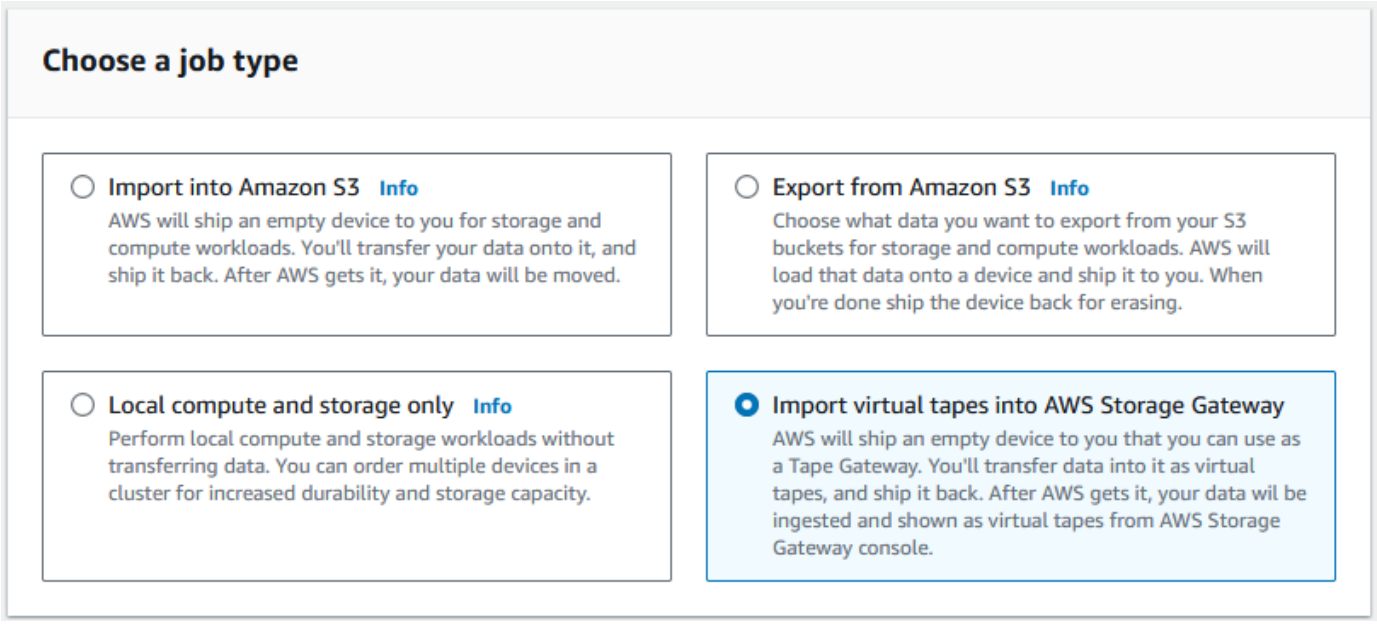
The following sections provide detailed instructions on ordering, deploying, using, and troubleshooting a Snowball Edge device with a Tape Gateway. For more information about creating and managing a Tape Gateway on your Snowball Edge device, see Using a Tape Gateway on an AWS Snowball Edge device in the *AWS Storage Gateway User Guide*.

Ordering a Snowball Edge device with a Tape Gateway

Use the following procedure to order a Snowball Edge device preinstalled with a Tape Gateway and the hardware specifications necessary to back up your tape data. For more detailed information about ordering Snow Family devices, see Creating a job to order a Snow Family device.

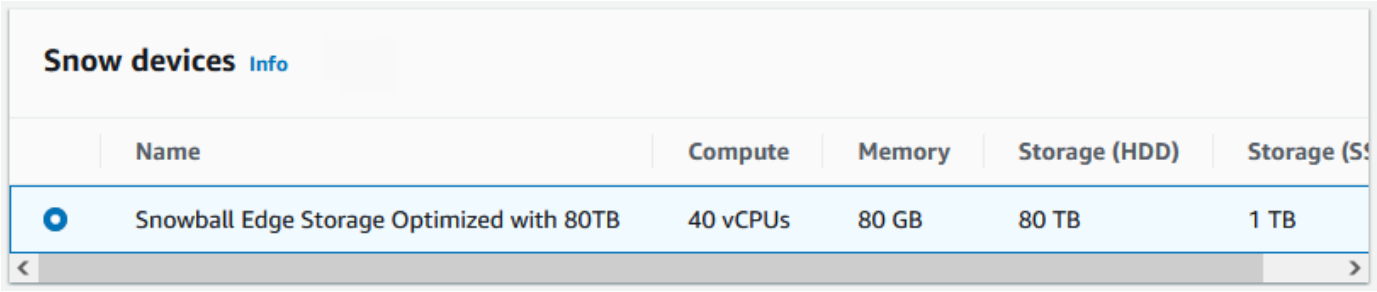
To order your device

1. Sign in to the AWS Management Console, and open the AWS Snow Family Management Console. If this is your first time creating a job in this AWS Region, you will see the **AWS Snow Family** page. Otherwise you will see the list of existing jobs.
2. If this is your first job, choose **Order an AWS Snow Family device**. Otherwise, choose **Create Job** in the left navigation bar.
3. In the **Job name** section, provide a name for your job in the **Job name** box.
4. Choose the **Import virtual tapes into AWS Storage Gateway** job type.



5. Choose **Next** to continue.
6. In the **Snow devices** section, the **Snowball Edge Storage Optimized with 80TB** device chosen for you. Choose **Next** to continue.

> **Note**
>
> **Snowball Edge Storage Optimized with 80TB** is the only Snow Family device that supports a Tape Gateway.



7. To finish creating the job, follow the procedures for Step 4: Choose security, shipping, and notification preferences and Step 5: Review job summary and create your job.

Now that your order has been placed, you can track it on the **Jobs** page of the AWS Snow Family Management Console. From there, you can also download and install the Snow Family management software—AWS OpsHub for Snow Family—while you wait for your Snowball Edge device to arrive.

Deploying a Snowball Edge device with a Tape Gateway

After you receive your Snowball Edge device with a Tape Gateway, use the following procedure to connect and unlock it, launch the Tape Gateway application service on it, and obtain the IP address for the Tape Gateway. The Tape Gateway IP address is different from the IP address of the Snowball Edge device as a whole, and is required for activating the Tape Gateway in the AWS Storage Gateway console. You must activate the Tape Gateway before you can start backing up your tape data.

To deploy your device

1. Connect the device to your local network and note its IP address. For more information, see Connecting to Your Local Network.
2. On a computer connected to the same local network as your device, download and install the latest version of AWS OpsHub for Snow Family from the **Jobs** page of the AWS Snow Family Management Console. For more information, see Using AWS OpsHub for Snow Family to Manage Devices.
3. Obtain the job manifest and unlock code for your device from the AWS Snow Family Management Console. For more information, see Getting credentials to access a Snow Family device.

4. Using AWS OpsHub for Snow Family and the credentials and IP address that you obtained in the previous steps, sign in and unlock the device. For more information, see Unlocking a device.
5. In AWS OpsHub for Snow Family, do the following to start the Tape Gateway application service on your device:

    a. From the **Local devices** page, choose the job name associated with your device to view its management dashboard.

    b. Under **Services**, choose **Tape Gateway**, then choose **Start service**.

    c. In the dialog box that appears, choose a virtual network interface for the Tape Gateway application service to use. Note the IP address of the interface that you select. This is the Tape Gateway IP address, which you will need when you activate the Tape Gateway in the AWS Storage Gateway console.

    d. Choose **Start service**.

Now that your Snowball Edge device is deployed, the Tape Gateway application service is running, and you've obtained the Tape Gateway IP address, you must activate the gateway in the AWS Storage Gateway console. To do this, choose **Open Storage Gateway management console** from your device's AWS OpsHub management dashboard, then follow the procedures described in Using a Tape Gateway on an AWS Snowball Edge device in the *AWS Storage Gateway User Guide*.

> **Note**
>
> Your computer must have network connectivity to the Snowball Edge device on port 80.

Troubleshooting and best practices for a Snowball Edge device with a Tape Gateway

To avoid problems and keep your Snowball Edge device with a Tape Gateway running smoothly, refer to the following tips and guidelines:

- After you order, receive, and deploy your Snowball Edge device, you must create and activate your Tape Gateway from the same AWS Region. Attempting to create and activate the Tape Gateway in any AWS Region other than where the Snowball Edge was ordered is not supported, and will not work.

- To back up your tape data using your Snowball Edge device with a Tape Gateway, connect the virtual tape devices on the gateway to a Windows or Linux client on your network, and then access them using your preferred backup software. For more information about connecting the gateway to a client and testing it with your backup software, see Using Your Tape Gateway in the *AWS Storage Gateway User Guide*.
- When a virtual tape is in the **Available** state in the Storage Gateway console, it is ready to be mounted using your preferred backup software, and its full capacity is reserved in physical storage on the Snowball Edge device. When you eject a virtual tape, its status changes from **Available** to **In transit to VTS**, and only the specific amount of data written to the tape remains reserved on the Snowball Edge device. You don't need to eject virtual tapes from your backup software before shipping your Snowball Edge device back to AWS. Any virtual tapes left in the **Available** state are automatically ejected during the data-transfer process at the AWS facility.
- After you copy the data to your Snowball Edge device, you can schedule a pickup appointment to ship the device back to AWS. The E Ink shipping label is automatically updated to ensure that the device is sent to the correct AWS facility. For more information, see Return shipping for Snow Family devices.

  You can track the device by using Amazon SNS generated text messages and emails.
- After your tape data is successfully transferred to the AWS Cloud and your Snowball Edge job is complete, you must manually delete the associated Tape Gateway using the Storage Gateway console.
- In rare cases, data corruption or other technical difficulties might prevent AWS from transferring specific virtual tapes to the AWS Cloud after receiving your Snowball Edge device. In such a case, you must use the Storage Gateway console to delete the virtual tapes that failed to transfer before you can re-attempt the transfer on another Snowball Edge device.
- A Snowball Edge device with a Tape Gateway supports only importing virtual tape data to AWS, and cannot be used to access data that has already been imported. To access your imported tape data, set up a standard Tape Gateway hosted on a virtual machine, hardware appliance, or Amazon EC2-compatible instance, and transfer the data from AWS over a network connection.
- A Snowball Edge device that is configured for a Tape Gateway is not intended for use with other Snowball Edge services or resources, such as Amazon S3, Network File System (NFS) file systems, AWS Lambda, or Amazon EC2. To use those services or resources, you must create a new Snowball Edge job to order a separate, appropriately configured device. For instructions, see Creating a job to order a Snow Family device.

- To troubleshoot a Snowball Edge device with a Tape Gateway, or if directed to do so by AWS Support, you might need to connect to your gateway's local console. The local console is a configuration interface that runs on the Snowball Edge device that's hosting your gateway. You can use this local console to perform maintenance tasks specific to the gateway on that device. For more information, see Performing Maintenance Tasks on the Local Console in the *AWS Storage Gateway User Guide*.

  To access the local console for the Tape Gateway running on your Snowball Edge device:

  1. From the command prompt on a computer connected to the same local network as your device, run the following command:

     ```
     ssh user_name@xxx.xxx.xxx.xxx
     ```

     To run this command, replace *user_name* with your local console user name, and replace *xxx.xxx.xxx.xxx* with the Tape Gateway IP address that you obtained when you launched the Tape Gateway on your Snowball Edge device. For instructions on how to obtain the Tape Gateway IP address, see Deploying a Snowball Edge device with a Tape Gateway.

  2. Enter your password.

     Note

     If this is your first time logging in to the local console on this device, the default user name is `admin`, and the default password is `password`. Change the default password immediately after you log in. For more information, see Logging in to the Local Console Using Default Credentials in the *AWS Storage Gateway User Guide*.

Using the Snowball Edge client with a Snowball Edge device with a Tape Gateway

The Snowball Edge client and AWS Command Line Interface (AWS CLI) work the same for a Snowball Edge device with a Tape Gateway as they do for a standard Snowball Edge device, with the following exceptions:

- After you receive your Snowball Edge device and use the `unlock-device --manifest-file` AWS CLI command to unlock it, you can use the `list-services` command to return the

`ServiceId` for the Tape Gateway application service. You must provide this value to start the Tape Gateway application service on your device.

```
snowballEdge list-services
{
    "ServiceIds" : [ "tapegateway" ]
}
```

- You can use the `describe-device` command to return the `PhysicalNetworkInterfaceId` for each physical network interface on your Snowball Edge device. You must provide this value when you create the virtual network interface for the Tape Gateway application service.

```
snowballEdge describe-device

{
  "DeviceId" : "JID-EXAMPLE12345-123-456-7-890",
  "UnlockStatus" : {
    "State" : "UNLOCKED"
  },
  "ActiveNetworkInterface" : {
    "IpAddress" : "192.0.2.0"
  },
  "PhysicalNetworkInterfaces" : [ {
    "PhysicalNetworkInterfaceId" : "s.ni-EXAMPLEd9ecbf03e3",
    "PhysicalConnectorType" : "RJ45",
    "IpAddressAssignment" : "STATIC",
    "IpAddress" : "0.0.0.0",
    "Netmask" : "0.0.0.0",
    "DefaultGateway" : "192.0.2.1",
    "MacAddress" : "EX:AM:PL:E0:12:34"
  }, {
    "PhysicalNetworkInterfaceId" : "s.ni-EXAMPLE4c3840068f",
    "PhysicalConnectorType" : "QSFP",
    "IpAddressAssignment" : "STATIC",
    "IpAddress" : "0.0.0.0",
    "Netmask" : "0.0.0.0",
    "DefaultGateway" : "192.0.2.2",
    "MacAddress" : "EX:AM:PL:E0:56:78"
  }, {
    "PhysicalNetworkInterfaceId" : "s.ni-abcd1234",
    "PhysicalConnectorType" : "SFP_PLUS",
    "IpAddressAssignment" : "DHCP",
    "IpAddress" : "192.168.1.231",
```

```
    "Netmask" : "255.255.255.0",
    "DefaultGateway" : "192.0.2.3",
    "MacAddress" : "EX:AM:PL:E0:90:12"
  } ]
}
```

- You can use the `create-virtual-network-interface` command to create the virtual network interface for the Tape Gateway application service. You must provide the `PhysicalNetworkInterfaceId` and specify the method of IP address assignment, such as DHCP. This command returns the `VirtualNetworkInterfaceArn` that you must provide when you start the Tape Gateway application service on your device.

```
snowballEdge create-virtual-network-interface \ --physical-network-interface-id s.ni-abcd1234 \ --ip-address-assignment DHCP

{
     "VirtualNetworkInterface" : {
         "VirtualNetworkInterfaceArn" : "arn:aws:snowball-device:::interface/s.ni-abcd1234",
         "PhysicalNetworkInterfaceId" : "s.ni-abcd1234",
         "IpAddressAssignment" : "DHCP",
         "IpAddress" : "192.0.2.0",
         "Netmask" : "255.255.255.0",
         "DefaultGateway" : "192.0.2.10",
         "MacAddress" : "1a:2b:3c:4d:5e:6f"
     }
}
```

- You can use the `start-service` command to start the Tape Gateway application service on your Snowball Edge device. You must provide the `ServiceId` and `VirtualNetworkInterfaceArn` for the Tape Gateway application service.

```
snowballEdge start-service \
--service-id tapegateway \
--virtual-network-interface-arns arn:aws:snowball-device:::interface/s.ni-abcd1234abcd1234
```

- You can use the `describe-service` command to check whether the Tape Gateway application service is active. You must provide the Tape Gateway `ServiceId`.

```
snowballEdge describe-service –service-id service-id
```

```
                    {
"ServiceId" : "tapegateway",
  "Status" : {
    "State" : "ACTIVE"
  },
"Storage" : {
"TotalSpaceBytes" : 99608745492480,
"FreeSpaceBytes" : 99608744468480
},
"Endpoints" : [ {
"Protocol" : "iSCSI",
"Port" : 860,
"Host" : "192.0.2.0"
}, {
"Protocol" : "iSCSI",
"Port" : 3260,
"Host" : "192.0.2.0",
} ]
}
```

For more detailed information about the Snow Family API, see the AWS Snow Family API Reference.

Using AWS IoT Greengrass to run pre-installed software on Amazon EC2-compatible instances

AWS IoT Greengrass is an open source Internet of Things (IoT) edge runtime and cloud service that helps you build, deploy, and manage IoT applications on your devices. You can use AWS IoT Greengrass to build software that enables your devices to act locally on the data that they generate, run predictions based on machine learning models, and filter and aggregate device data. For detailed information about AWS IoT Greengrass, see What is AWS IoT Greengrass? in the *AWS IoT Greengrass Version 2 Developer Guide*.

By using AWS IoT Greengrass on your Snow Family device, you enable the device to collect and analyze data closer to where it is generated, react autonomously to local events, and communicate securely with other devices on the local network.

Setting up your Amazon EC2-compatible instance

> **Note**
>
> To install AWS IoT Greengrass Version 2 on a Snow Family device, make sure that your device is connected to the internet. After installation, the internet is not required for a Snow Family device to work with AWS IoT Greengrass.

To set up an EC2-compatible instance for AWS IoT Greengrass V2

1. Launch the AWS IoT Greengrass validated AMI with a public IP Address and an SSH key:

   a. Using the AWS CLI: run-instances.

   b. Using AWS OpsHub: Launching an Amazon EC2-compatible instance.

   > **Note**
   >
   > Take note of the public IP address and SSH key name that are associated with the instance.

2. Connect to the EC2-compatible instance using SSH. To do so, run the following command on the computer that is connected to your device. Replace *ssh-key* with the key you used to launch the EC2-compatible instance. Replace *public-ip-address* with the public IP address of the EC2-compatible instance.

   ```
   ssh -i ssh-key ec2-user@ public-ip-address
   ```

   > **Important**
   >
   > If your computer uses an earlier version of Microsoft Windows, you might not have the SSH command, or you might have SSH but can't connect to your EC2-compatible instance. To connect to your EC2-compatible instance, you can install and configure PuTTY, which is a no-cost, open source SSH client. You must convert the SSH key from `.pem` format to PuTTY format and connect to your EC2 instance. For instructions

on how to convert from `.pem` to PuTTY format, see Convert your private key using PuTTYgen in the Amazon EC2 User Guide for Linux Instances.

Installing AWS IoT Greengrass

Next, you set up your EC2-compatible instance as an AWS IoT Greengrass Core device that you can use for local development.

To install AWS IoT Greengrass

1. Use the following command to install the prerequisite software for AWS IoT Greengrass. This command installs the AWS Command Line Interface (AWS CLI) v2, Python 3, and Java 8.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
 && unzip awscliv2.zip && sudo ./aws/install && sudo yum -y install python3
 java-1.8.0-openjdk
```

2. Grant the root user permission to run the AWS IoT Greengrass software and modify the root permission from `root ALL=(ALL) ALL` to `root ALL=(ALL:ALL) ALL` in the sudoers config file.

```
sudo sed -in 's/root\tALL=(ALL)/root\tALL=(ALL:ALL)/' /etc/sudoers
```

3. Use the following command to download the AWS IoT Greengrass Core software.

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
latest.zip > greengrass-nucleus-latest.zip && unzip greengrass-nucleus-latest.zip -
d GreengrassCore && rm greengrass-nucleus-latest.zip
```

4. Use the following commands to provide credentials to allow you to install AWS IoT Greengrass Core software. Replace the example values with your credentials:

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

> **Note**
>
> These are credentials from the IAM user in the AWS Region, not the Snow Family device.

5. Use the following command to install the AWS IoT Greengrass Core software. The command creates AWS resources that the core software requires to operate and sets up the core software as a system service that runs when the AMI boots up.

   Replace the following parameters in the command:

   - `region`: The AWS Region in which to find or create resources.
   - `MyGreengrassCore`: The name of the AWS IoT thing for your AWS IoT Greengrass core device.
   - `MyGreengrassCoreGroup`: The name of the AWS IoT thing group for your AWS IoT Greengrass core device.

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
    -jar ./GreengrassInstaller/lib/Greengrass.jar \
    --aws-region region \
    --thing-name MyGreengrassCore \
    --thing-group-name MyGreengrassCoreGroup \
    --thing-policy-name GreengrassV2IoTThingPolicy \
    --tes-role-name GreengrassV2TokenExchangeRole \
    --tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias \
    --component-default-user ggc_user:ggc_group \
    --provision true \
    --setup-system-service true \
    --deploy-dev-tools true
```

> **Note**
>
> This command is for an Amazon EC2-compatible instance running an Amazon Linux 2 AMI. For a Windows AMI, see Install the AWS IoT Greengrass Core software.

When you are finished, you will have an AWS IoT Greengrass core running on your Snow Family device for your local use.

Using AWS Lambda with an AWS Snowball Edge

AWS Lambda powered by AWS IoT Greengrass is a compute service that lets you to run serverless code (Lambda functions) locally on Snowball Edge devices. You can use Lambda to invoke Lambda functions on a Snowball Edge device with Message Queuing Telemetry Transport (MQTT) messages, run Python code in Lambda functions, and use them to call public AWS service endpoints in the cloud. To use Lambda functions with Snowball Edge devices, you must create your Snowball Edge jobs in an AWS Region supported by AWS IoT Greengrass. For a list of valid AWS Regions, see AWS IoT Greengrass in the AWS General Reference. Lambda on Snowball Edge is available in Regions where Lambda and Snowball Edge devices are available.

> **Note**
>
> If you allocate the minimum recommendation of 128 MB of memory for each of your functions, you can have up to seven Lambda functions in a single job.

Topics

Before You Start

Before you create a Lambda function in the Python language to run on your Snowball Edge, we recommend that you familiarize yourself with the following services, concepts, and related topics.

Prerequisites for AWS IoT Greengrass

AWS IoT Greengrass is software that extends AWS Cloud capabilities to local devices. AWS IoT Greengrass makes it possible for local devices to collect and analyze data closer to the source of information, while also securely communicating with each other on local networks. More specifically, developers who use AWS IoT Greengrass can author serverless code (Lambda functions) in the AWS Cloud. They can then conveniently deploy this code to devices for local execution of applications.

The following AWS IoT Greengrass concepts are important to understand when using AWS IoT Greengrass with a Snowball Edge:

- **AWS IoT Greengrass requirements** – For a full list of AWS IoT Greengrass requirements, see Requirements in the *AWS IoT Greengrass Version 2 Developer Guide*.
- **AWS IoT Greengrass core** – Download the AWS IoT Greengrass core software and install it on an EC2 instance running on the device. See Using AWS IoT Greengrass on Amazon EC2 instances in this guide.

  To use Lambda functions on a Snowball Edge device, you must first install AWS IoT Greengrass Core software on an Amazon EC2 instance on the device. The Lambda functions you plan to use on the Snowball Edge device must be created by the same account you will use to install AWS IoT Greengrass on the Snowball Edge device. For information about installing AWS IoT Greengrass on your Snowball Edge device, see Using AWS IoT Greengrass to run pre-installed software on Amazon EC2-compatible instances.
- **AWS IoT Greengrass group** – A Snowball Edge device is part of an AWS IoT Greengrass group as the group's core device. For more information about groups, see AWS Greengrass IoT Groups in the *AWS IoT Greengrass Developer Guide*.
- **MQTT** – AWS IoT Greengrass uses the industry-standard, lightweight MQTT protocol to communicate within a group. Any device or software compatible with MQTT in your AWS IoT Greengrass group can invoke MQTT messages. These messages can invoke Lambda functions, if you define the related MQTT message to do so.

Prerequisites for AWS Lambda

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. The following Lambda concepts are important to understand when using Lambda with a Snowball Edge:

- **Lambda functions** – Your custom code, uploaded and published to Lambda and used on a Snowball Edge. For more information, see Lambda Functions in the *AWS Lambda Developer Guide*.
- **Lambda console** – The console in which you upload, update, and publish your Python-language Lambda functions for use on a Snowball Edge. For more information about the Lambda console, see Lambda console in the *AWS Lambda Developer Guide*.

- **Python** – The high-level programming language used for your Lambda functions powered by AWS IoT Greengrass on a Snowball Edge. AWS IoT Greengrass supports Python version 3.8.x.

Deploy a Lambda function to a Snowball Edge device

To run a Lambda function on a Snowball Edge device in an AWS IoT Greengrass group, import the function as a component. For complete information about importing a function as a component using the AWS IoT Greengrass console, see Import a Lambda function as a component (console) in the AWS IoT Greengrass Version 2 Developer Guide.

1. In the AWS IoT console, on the **Greengrass components** page, choose **Create component**.
2. In **Component source**, choose **Import Lambda function**. In **Lambda function**, choose the name of your function. In **Lambda function version**, choose the version of your function.
3. To subscribe the function to messages on which it can act, choose **Add event source** and choose the event. In **Timeout (seconds)**, provide a timeout period in seconds.
4. In **Pinned**, choose whether or not to pin your function.
5. Choose **Create component**
6. Choose **Deploy**.
7. In **Deployment**, choose **Add to existing deployment**, then choose your Greengrass group. Choose **Next**.
8. In **Public components**, choose these components:
   - **aws.greengrass.Cli**
   - **aws.greengrass.LambdaLauncher**
   - **aws.greengrass.LambdaManager**
   - **aws.greengrass.LambdaRuntimes**
   - **aws.greengrass.Nucleus**
9. Choose **Deploy**.

Using Amazon EC2-compatible compute instances

This section provides an overview of using Amazon EC2-compatible compute instances on an AWS Snowball Edge device, including conceptual information, procedures, and examples.

Topics

Overview

You can run Amazon EC2-compatible compute instances hosted on a Snowball Edge with the `sbe1`, `sbe-c`, and `sbe-g` instance types. The `sbe1` instance type works on devices with the Snowball Edge Storage Optimized option. The `sbe-c` instance type works on devices with the Snowball Edge Compute Optimized option. Both the `sbe-c` and `sbe-g` instance types work on devices with the Snowball Edge Compute Optimized with GPU option. For a list of supported instance types, see Quotas for Compute Instances on a Snowball Edge Device.

All three compute instance types supported for use on Snowball Edge device options are unique to Snowball Edge devices. Like their cloud-based counterparts, these instances require Amazon

Machine Images (AMIs) to launch. You choose the AMI to be that base image for an instance in the cloud, before you create your Snowball Edge job.

To use a compute instance on a Snowball Edge, create a job to order a Snow Family device and specify your AMIs. You can do this using the AWS Snow Family Management Console, the AWS CLI, or one of the AWS SDKs. Typically, there are some housekeeping prerequisites that you must perform before creating your job, to use your instances.

After your device arrives, you can start managing your AMIs and instances. You can manage your compute instances on a Snowball Edge through an Amazon EC2-compatible endpoint. This type of endpoint supports many of the Amazon EC2-compatible CLI commands and actions for the AWS SDKs. You can't use the AWS Management Console on the Snowball Edge to manage your AMIs and compute instances.

When you're done with your device, return it to AWS. If the device was used in an import job, the data transferred using the Amazon S3 adapter or the NFS interface is imported into Amazon S3. Otherwise, we perform a complete erasure of the device when it is returned to AWS. This erasure follows the National Institute of Standards and Technology (NIST) 800-88 standards.

Important

- Using encrypted AMIs on Snowball Edge devices is not supported.
- Data in compute instances running on a Snowball Edge isn't imported into AWS.

Difference between Amazon EC2 and Amazon EC2-compatible instances on Snow Family devices

AWS Snow Family EC2-compatible instances allow customers to use and manage Amazon EC2-compatible instances using a subset of EC2 APIs and a subset of AMIs.

Pricing for Compute Instances on Snowball Edge

There are additional costs associated with using compute instances. For more information, see AWS Snowball Edge Pricing.

Using an Amazon EC2-compatible AMI on Snow Family devices

To use an Amazon Machine Image (AMI) on your AWS Snow Family device, you must first add it to the device. You can add an AMI in the following ways:

- Upload the AMI when you order the device.
- Add the AMI when your device arrives at your site.

Amazon EC2 compute instances that come with your Snow Family devices are launched based on the Amazon EC2 AMIs that you add to your device. Amazon EC2-compatible AMIs support both Linux and Microsoft Windows operating systems.

Linux

The following Linux operating systems are supported:

- Amazon Linux 2 for Snow Family

  > **Note**
  >
  > The latest version of this AMI will be provided at the time your Snow Family device is being prepared to ship by AWS. To determine the version of this AMI on the device when you receive it, see Determining the version of the Amazon Linux 2 AMI for Snow Family.

- CentOS 7 (x86_64) - with Updates HVM
- Ubuntu 16.04 LTS - Xenial (HVM)

  > **Note**
  >
  > Ubuntu 16.04 LTS - Xenial (HVM) images are no longer supported in the AWS Marketplace, but still supported for use on Snowball Edge devices through Amazon EC2 VM Import/Export and running locally in AMIs.

- Ubuntu 20.04 LTS - Focal
- Ubuntu 22.04 LTS - Jammy

As a best-practice for security, keep your Amazon Linux 2 AMIs up-to-date on Snow Family devices as new Amazon Linux 2 AMIs are released. See Updating your Amazon Linux 2 AMIs on Snow Family devices.

Windows

The following Windows operating systems are supported:

- Windows Server 2012 R2
- Windows Server 2016
- Windows Server 2019

You can add Windows AMIs to your device by importing your Windows virtual machine (VM) image into AWS using VM Import/Export. Or, you can import the image into your device right after the device is deployed to your site. For more information, see Adding a Microsoft Windows AMI.

> **Note**
>
> Windows AMIs that originated in AWS can't be added to your device.
> AMIs imported locally must be in BIOS boot mode as UEFI is not supported.

Snow Family supports the Bring Your Own License (BYOL) model. For more information, see Adding a Microsoft Windows AMI.

> **Note**
>
> AWS Snow Family EC2-compatible instances allow customers to use and manage Amazon EC2-compatible instances using a subset of EC2 APIs and a subset of AMIs.

Topics

- Adding an AMI When Ordering Your Device
- Adding an AMI from AWS Marketplace
- Adding an AMI Locally
- Adding a Microsoft Windows AMI
- Importing a VM Image into Your Device

- Exporting the latest Amazon Linux 2 AMI

Adding an AMI When Ordering Your Device

When you order your device, you can add AMIs to the device by choosing them in the **Compute using EC2 instances - optional** section in the AWS Snow Family Management Console. The **Compute using EC2 instances - optional** lists all of the AMIs that can be loaded onto your device. The AMIs fall into the following categories:

- **AMIs from AWS Marketplace** — These are AMIs created from the list of supported AMIs. For information about creating an AMI from the supported AMIs from AWS Marketplace, see Adding an AMI from AWS Marketplace.
- **AMIs uploaded using VM Import/Export** — When you order your device, the AMIs that were uploaded using VM Import/Export are listed in the console. For more information, see Importing a VM as an Image Using VM Import/Export in the *VM Import/Export User Guide*. For information about supported virtualization environments, see VM Import/Export Requirements.

Adding an AMI from AWS Marketplace

You can add many AMIs from AWS Marketplace to your Snow Family device by launching the AWS Marketplace instance, creating an AMI from it, and configuring the AMI in the same region from which you'll order the Snow device. Then, you can choose to include the AMI on the device when you create a job to order the device. When choosing an AMI from the Marketplace, make sure it has a supported product code and platform.

Topics

- Checking product codes and platform details of AWS Marketplace AMIs
- Determining the version of the Amazon Linux 2 AMI for Snow Family
- Configure the AMI for the Snow Family device

Checking product codes and platform details of AWS Marketplace AMIs

Before you begin the process to add an AMI from AWS Marketplace to your Snow Family device, ensure the product code and platform details of the AMI are supported in your AWS Region.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. From the navigation bar, select the Region in which to launch your instances and from which you will create the job to order the Snow Family device. You can select any Region that is available to you, regardless of your location.
3. In the navigation pane, choose **AMIs**.
4. Use the filter and search options to scope the list of displayed AMIs to see only the AMIs that match your criteria. For example, AMIs provided by the AWS Marketplace, choose **Public images**. Then use the search options to further scope the list of displayed AMIs:

   - (New console) Choose the **Search** bar and, from the menu, choose **Owner alias**, then the = operator, and then the value **amazon**.
   - (Old console) Choose the **Search** bar and, from the menu, choose **Owner** and then the value **Amazon images**.

   > **Note**
   >
   > AMIs from AWS Marketplace include **aws-marketplace** in the **Source** column.

5. In the **AMI ID** column, choose the AMI ID of the AMI.
6. In the **Image summary** of the AMI, ensure the **Product codes** are supported by your Region. For more information, see the table below.

   Supported AWS Marketplace AMI product codes

AMI operating system	Product code
Ubuntu Server 14.04 LTS	b3dl4415quatdndl4qa6kcu45
CentOS 7 (x86_64)	aw0evgkw8e5c1q413zgy5pjce
Ubuntu 16.04 LTS	csv6h7oyg29b7epjzg7qdr7no
Amazon Linux 2	avyfzznywektkgl5qv5f57ska
Ubuntu 20.04 LTS	a8jyynf4hjutohctm41o2z18m
Ubuntu 22.04 LTS	47xbqns9xujfkkjt189a13aqe

7. Then, also ensure the **Platform details** contains one of entries from the list below.

- Amazon Linux, Ubuntu, or Debian
- Red Hat Linux bring-your-own-license
- Amazon RDS for Oracle bring-your-own-license
- Windows bring-your-own-license

Determining the version of the Amazon Linux 2 AMI for Snow Family

Use the following procedure to determine the version of the Amazon Linux 2 AMI for Snow Family on the Snow Family device. Install the latest version of the AWS CLI before continuing. For more information, see Install or update to the latest version of the AWS CLI in the AWS Command Line Interface User Guide.

- Use the `describe-images` AWS CLI command to see the description of the AMI. The version is contained in the description. Provide the public key certificate from the previous step. For more information, see describe-images in the AWS CLI Command Reference.

```
aws ec2 describe-images --endpoint http://snow-device-ip:8008 --region snow
```

Example of output of the `describe-images` command

```
{
    "Images": [
        {
            "CreationDate": "2024-02-12T23:24:45.705Z",
            "ImageId": "s.ami-02ba84cb87224e16e",
            "Public": false,
            "ProductCodes": [
                {
                    "ProductCodeId": "avyfzznywektkgl5qv5f57ska",
                    "ProductCodeType": "marketplace"
                }
            ],
            "State": "AVAILABLE",
            "BlockDeviceMappings": [
                {
```

```
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "DeleteOnTermination": true,
                        "Iops": 0,
                        "SnapshotId": "s.snap-0efb49f2f726fde63",
                        "VolumeSize": 8,
                        "VolumeType": "sbp1"
                    }
                }
            ],
            "Description": "Snow Family Amazon Linux 2 AMI 2.0.20240131.0 x86_64
 HVM gp2",
            "EnaSupport": false,
            "Name": "amzn2-ami-snow-family-hvm-2.0.20240131.0-x86_64-gp2-
b7e7f8d2-1b9e-4774-a374-120e0cd85d5a",
            "RootDeviceName": "/dev/xvda"
        }
    ]
}
```


In this example, the version of the Amazon Linux 2 AMI for Snow Family is **`2.0.20240131.0`**. It is found in the value of the `Description` name.

Configure the AMI for the Snow Family device

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Launch a new instance of a supported AMI in AWS Marketplace.

> **Note**
>
> When you launch your instance, make sure that the storage size you assign to the instance is appropriate for your use case. In the Amazon EC2 console, you do this in the **Add storage** step.

3. Install and configure the applications that you want to run on the Snowball Edge, and make sure that they work as expected.

Important

- Only single volume AMIs are supported.
- The EBS volume in your AMI should be 10 TB or less. We recommend that you provision the EBS volume size needed for the data in the AMI. This will help decrease the time it takes to export your AMI and load it into your device. You can resize or add more volumes to your instance after your device is deployed.
- The EBS snapshot in your AMI must not be encrypted.

4. Make a copy of the PEM or PPK file that you used for the SSH key pair when you created this instance. Save this file to the server that you plan to use to communicate with the Snowball Edge device. Make a note of the path to this file because you will need it when you use SSH to connect to the EC2-compatible instance on your device.

   Important

   If you don't follow this procedure, you can't connect to your instances with SSH when you receive your Snowball Edge device.

5. Save the instance as an AMI. For more information, see Amazon EC2 User Guide for Linux Instances in the *Amazon EC2 User Guide for Linux Instances.*
6. Repeat steps 1–4 for each of the instances that you want to connect to using SSH. Be sure to make copies of each of the SSH key pairs, and keep track of the AMIs that they're associated with.
7. Now, when you order your device, these AMIs are available to add to your device.

Adding an AMI Locally

When the device arrives on your site, you can add new AMIs to it. For instructions, see Importing a virtual machine image to a Snow Family device. Keep in mind that although all VMs are supported, only supported AMIs have been tested for full functionality.

Note

When you use VM Import/Export to add AMIs to your device or import a VM after your device is deployed, you can add VMs that use any operating system. However, only

supported operating systems have been tested and validated on Snow Family devices. You are responsible for adhering to the terms and conditions of any operating system or software that is in the virtual image that you import onto your device.

> **Important**
>
> For AWS services to function properly on a Snowball Edge, you must allow the ports for the services. For details, see Ports Required to Use AWS Services on an AWS Snowball Edge Device.

Adding a Microsoft Windows AMI

For virtual machines (VMs) that use a supported Windows operating system, you can add the AMI by importing your Windows VM image into AWS using VM Import/Export, or by importing it into your device directly after it is deployed to your site.

Bring Your Own License (BYOL)

Snowball Edge supports importing Microsoft Windows AMIs onto your device with your own license. Bring Your Own License (BYOL) is the process of bringing an AMI that you own with its on-premises license to AWS. AWS provides both shared and dedicated deployment options for the BYOL option.

You can add your Windows VM image to your device by importing it into AWS using VM Import/Export or by importing it into your device directly after it is deployed to your site. You can't add Windows AMIs that originated in AWS. Therefore, you must create and import your own Windows VM image and bring your own license if you want to use the AMI on your Snow Family device. For more information about Windows licensing and BYOL, see Amazon Web Services and Microsoft: Frequently Asked Questions.

Creating a Windows VM Image to Import into Your Device

To create a Windows VM image, you need a virtualization environment, such as VirtualBox, which is supported for the Windows and macOS operating systems. When you create a VM for Snow devices, we recommend that you allocate at least two cores with at least 4 GB of RAM. When the VM is up and running, you must install your operating system (Windows Server 2012, 2016, or 2019). To install the required drivers for the Snow Family device, follow the instructions in this section.

For a Windows AMI to run on a Snow device, you must add the VirtIO, FLR, NetVCM, Vioinput, Viorng, Vioscsi, Vioserial, and VioStor drivers. You can download a Microsoft Software Installer (virtio-win-guest-tools-installer) for installing these drivers on Windows images from the virtio-win-pkg-scripts repository on GitHub.

> **Note**
>
> If you plan to import your VM image directly to your deployed Snow device, the VM image file must be in the RAW format.

To create a Windows image

1. On your Microsoft Windows computer, choose **Start** and enter **`devmgmt.msc`** to open **Device Manager**.
2. In the main menu, choose **Actions**, and then choose **Add legacy hardware**.
3. In the wizard, choose **Next**.
4. Choose **Install the hardware that I manually select from a list (advanced)**, and choose **Next**.
5. Choose **Show All Devices** and choose **Next**.
6. Choose **Have Disk**, open the **Copy manufacturer's files from** list, and browse to the ISO file.
7. In the ISO file, browse to the `Driver\W2K8R2\amd64` directory, and then find the `.INF` file.
8. Choose the **.INF** file, choose **Open**, and then choose **OK**.
9. When you see the driver name, choose **Next**, and then choose **Next** two more times. Then choose **Finish**.

   This installs a device using the new driver. The actual hardware doesn't exist, so you will see a yellow exclamation mark that indicates an issue on the device. You must fix this issue.

To fix the hardware issue

1. Open the context (right-click) menu for the device that has the exclamation mark.
2. Choose **Uninstall**, clear **Delete the driver software for this device**, and choose **OK**.

   The driver is installed, and you are ready to launch the AMI on your device.

Importing a VM Image into Your Device

After you prepare your VM image, you can use one of the options to import the image to your device.

- **In the cloud using VM Import/Export** — When you import your VM image into AWS and register it as an AMI, you can add it to your device when you place an order from the AWS Snow Family Management Console. For more information, see Importing a VM as an Image Using VM Import/Export in the *VM Import/Export User Guide*.
- **Locally on your device that is deployed at your site** — You can import your VM image directly into your device using AWS OpsHub for Snow Family or the AWS Command Line Interface (AWS CLI).

  For information about using AWS OpsHub, see Using Amazon EC2-compatible compute instances locally.

  For information about using the AWS CLI, see Importing a virtual machine image to a Snow Family device.

Exporting the latest Amazon Linux 2 AMI

To update your Amazon Linux 2 AMIs to the latest version, first export the latest Amazon Linux 2 VM image from AWS Marketplace, then import that VM image into the Snow device.

1. Use the `ssm get-parameters` AWS CLI command to find the latest image ID of the Amazon Linux 2 AMI in the AWS Marketplace.

   ```
   aws ssm get-parameters --names /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2 --query 'Parameters[0].[Value]' --region your-region
   ```

   The command returns the latest image ID of the AMI. For example, `ami-0ccb473bada910e74`.

2. Export the latest Amazon Linux 2 image. See Exporting a VM directly from an Amazon Machine Image (AMI) in the Amazon EC2 User Guide for Linux Instances. Use the latest image ID of the Amazon Linux 2 AMI as the value of the `image-id` parameter of the `ec2 export-image` command.

3. Import the VM image into the Snow device using the AWS CLI or AWS OpsHub.

   - For information about using AWS CLI, see Importing a virtual machine image to a Snow Family device.
   - For information about using AWS OpsHub, see Importing an image into your device as an Amazon EC2-compatible AMI .

Importing a virtual machine image to a Snow Family device

You can use the AWS CLI and the VM Import/Export service to import a virtual machine (VM) image to the Snow Family device as an Amazon Machine Image (AMI). After importing a VM image, register the image as an AMI and launch it as an Amazon EC2-compatible instance.

You can add AMIs from Amazon EC2 to the device when creating a job to order a Snow Family device. Use this procedure after you have received the Snow Family device. For more information, see Step 2: Choose your compute and storage options.

You can also use AWS OpsHub to upload the VM image file. For more information, see Importing an image into your device as an Amazon EC2-compatible AMI in this guide.

Topics

Step 1: Prepare the VM image and upload it to the Snow Family device

Prepare the VM image by exporting a VM image from an Amazon EC2 AMI or instance in the AWS Cloud using VM Import/Export or by generating the VM image locally using your choice of virtualization platform.

To export an Amazon EC2 instance as a VM image using VM Import/Export, see Exporting an instance as a VM using VM Import/Export in the VM Import/Export User Guide. To export an

Amazon EC2 AMI as a VM image using VM Import/Export, see Exporting a VM directly from an Amazon Machine Image (AMI) in the VM Import/Export User Guide.

If generating a VM image from your local environment, ensure the image is configured for use as an AMI on the Snow Family device. You may need to configure the following items, depending on your environment.

- Configure and update the operating system.
- Set a hostname.
- Ensure network time protocol (NTP) is configured.
- Include SSH public keys, if necessary. Make local copies of the key pairs. For more information, see Using SSH to Connect to Your Compute Instances on a Snowball Edge.
- Install and configure any software you will use on the Snow Family device.

> **Note**
>
> Be aware of the following limitations when preparing a disk snapshot for a Snow Family device.
>
> - Snow Family devices currently only support importing snapshots that are in the RAW image format.
> - Snow Family devices currently only support importing snapshots with sizes from 1 GB to 1 TB.

Uploading a VM image to an Amazon S3 bucket on the Snow Family device

After preparing a VM image, upload it to an S3 bucket on the Snow Family device or cluster. You can use the S3 adapter or Amazon S3 compatible storage on Snow Family devices to upload the snapshot.

To upload the virtual machine image using the S3 adapter

- Use the `cp` command to copy the VM image file to a bucket on the device.

```
aws s3 cp image-path s3://S3-bucket-name --endpoint http://S3-object-API-endpoint:443 --profile profile-name
```

For more information, see Supported AWS CLI commands in this guide.

To upload the VM image using Amazon S3 compatible storage on Snow Family devices

- Use the `put-object` command to copy the snapshot file to a bucket on the device.

```
aws s3api put-object --bucket bucket-name --key path-to-snapshot-file --
body snapshot-file --profile your-profile --endpoint-url s3api-endpoint-ip
```

For more information, see Working with S3 objects on a Snowball Edge device.

Step 2: Set up required permissions

For the import to be successful, you must set up permissions for VM Import/Export on the Snow Family device, Amazon EC2, and the user.

> **Note**
>
> The service roles and policies that provide these permissions are located on the Snow Family device.

Permissions required for VM Import/Export

Before you can start the import process, you must create an IAM role with a trust policy that allows VM Import/Export on the Snow Family device to assume the role. Additional permissions are given to the role to allow VM Import/Export on the device to access the image stored in the S3 bucket on the device.

Create a trust policy json file

Following is an example trust policy required to be attached to the role so that VM Import/Export can access the snapshot that needs to be imported from the S3 bucket.

```
{
   "Version":"2012-10-17",
```

```
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"vmie.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

Create a role with the trust policy json file

The role name can be vmimport. You can change it by using the --role-name option in the command:

```
aws iam create-role --role-name role-name --assume-role-policy-document file:///trust-policy-json-path --profile profile-name --endpoint http://snowball-ip:6078 --region
 snow
```

The following is an example output from the `create-role` command.

```
{
   "Role":{
      "AssumeRolePolicyDocument":{
         "Version":"2012-10-17",
         "Statement":[
            {
               "Action":"sts:AssumeRole",
               "Effect":"Allow",
               "Principal":{
                  "Service":"vmie.amazonaws.com"
               }
            }
         ]
      },
      "MaxSessionDuration":3600,
      "RoleId":"AROACEMGEZDGNBVGY3TQOJQGEZAAAABQBB6NSGNAAAABPSVLTREPY3FPAFOLKJ3",
      "CreateDate":"2022-04-19T22:17:19.823Z",
      "RoleName":"vmimport",
      "Path":"/",
      "Arn":"arn:aws:iam::123456789012:role/vmimport"
   }
```

```
}
```

Create a policy for the role

The following example policy has the minimum required permissions to access Amazon S3. Change the Amazon S3 bucket name to the one which has your images. For a standalone Snowball Edge device, change *snow-id* to your job ID. For a cluster of devices, change *snow-id* to the cluster ID. You also can use prefixes to further narrow down the location where VM Import/Export can import snapshots from. Create a policy json file like this.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:GetMetadata"
         ],
         "Resource":[
            "arn:aws:s3:snow:account-id:snow/snow-id/bucket/import-snapshot-bucket-
name",
            "arn:aws:s3:snow:account-id:snow/snow-id/bucket/import-snapshot-bucket-
name/*"
            ]
      }
   ]
}
```

Create a policy with the policy file:

```
aws iam create-policy --policy-name policy-name --policy-document file:///policy-json-
file-path --profile profile-name --endpoint http://snowball-ip:6078 --region snow
```

The following is an output example from the create-policy command.

```
{
   "Policy":{
      "PolicyName":"vmimport-resource-policy",
```

```
      "PolicyId":"ANPACEMGEZDGNBVGY3TQOJQGEZAAAABOOEE3IIHAAAABWZJPI2VW4UUTFEDBC2R",
      "Arn":"arn:aws:iam::123456789012:policy/vmimport-resource-policy",
      "Path":"/",
      "DefaultVersionId":"v1",
      "AttachmentCount":0,
      "IsAttachable":true,
      "CreateDate":"2020-07-25T23:27:35.690000+00:00",
      "UpdateDate":"2020-07-25T23:27:35.690000+00:00"
   }
}
```

Attach the policy to the role

Attach a policy to the preceding role and grant permissions to access the required resources. This allows the local VM Import/Export service to download the snapshot from Amazon S3 on the device.

```
aws iam attach-role-policy --role-name role-name --policy-arn
 arn:aws:iam::123456789012:policy/policy-name --profile profile-name --endpoint
 http://snowball-ip:6078 --region snow
```

Permissions required by the caller

In addition to the role for the Snowball Edge VM Import/Export to assume, you also must ensure that the user has the permissions that allow them to pass the role to VMIE. If you use the default root user to perform the import, the root user already has all the permissions required, so you can skip this step, and go to step 3.

Attach the following two IAM permissions to the user that is doing the import.

- `pass-role`
- `get-role`

Create a policy for the role

The following is an example policy that allows a user to perform the `get-role` and `pass-role` actions for the IAM role.

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
        {
            "Effect":"Allow",
            "Action": "iam:GetRole",
            "Resource":"*"
        },
        {
            "Sid": "iamPassRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "importexport.amazonaws.com"
                }
            }
        }
    ]
}
```

Create a policy with the policy file:

```
aws iam create-policy --policy-name policy-name --policy-document file:///policy-json-file-path --profile profile-name --endpoint http://snowball-ip:6078 --region snow
```

The following is an output example from the create-policy command.

```
{
   "Policy":{
      "PolicyName":"caller-policy",
      "PolicyId":"ANPACEMGEZDGNBVGY3TQOJQGEZAAAABOOOTUOE3AAAAAAPPBEUM7Q7ARPUE53C6R",
      "Arn":"arn:aws:iam::123456789012:policy/caller-policy",
      "Path":"/",
      "DefaultVersionId":"v1",
      "AttachmentCount":0,
      "IsAttachable":true,
      "CreateDate":"2020-07-30T00:58:25.309000+00:00",
      "UpdateDate":"2020-07-30T00:58:25.309000+00:00"
   }
}
```

After the policy has been generated, attach the policy to the IAM users that will call the Amazon EC2 API or CLI operation to import the snapshot.

```
aws iam attach-user-policy --user-name your-user-name --policy-arn
 arn:aws:iam::123456789012:policy/policy-name --profile profile-name --endpoint
 http://snowball-ip:6078 --region snow
```

Permissions Required to call Amazon EC2 APIs on your device

To import a snapshot, the IAM user must have the `ec2:ImportSnapshot` permissions. If restricting access to the user is not required, you can use the `ec2:*` permissions to grant full Amazon EC2 access. The following are the permissions that can be granted or restricted for Amazon EC2 on your device. Create a policy file with the content shown:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ec2:ImportSnapshot",
            "ec2:DescribeImportSnapshotTasks",
            "ec2:CancelImportTask",
            "ec2:DescribeSnapshots",
            "ec2:DeleteSnapshot",
            "ec2:RegisterImage",
            "ec2:DescribeImages",
            "ec2:DeregisterImage"
         ],
         "Resource":"*"
      }
   ]
}
```

Create a policy with the policy file:

```
aws iam create-policy --policy-name policy-name --policy-document file:///policy-json-
file-path --profile profile-name --endpoint http://snowball-ip:6078 --region snow
```

The following is an output example from the create-policy command.

```
{
    "Policy":
        {
```

```
            "PolicyName": "ec2-import.json",
            "PolicyId":
 "ANPACEMGEZDGNBVGY3TQOJQGEZAAAABQBGPDQC5AAAAATYN62UNBFYTF5WVCSCZS",
            "Arn": "arn:aws:iam::123456789012:policy/ec2-import.json",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 0,
            "IsAttachable": true,
            "CreateDate": "2022-04-21T16:25:53.504000+00:00",
            "UpdateDate": "2022-04-21T16:25:53.504000+00:00"
        }
}
```

After the policy has been generated, attach the policy to the IAM users that will call the Amazon EC2 API or CLI operation to import the snapshot.

```
aws iam attach-user-policy --user-name your-user-name --policy-arn
 arn:aws:iam::123456789012:policy/policy-name --profile profile-name --endpoint
 http://snowball-ip:6078 --region snow
```

Step 3: Import the VM image as a snapshot on the device

The next step is to import the VM image as a snapshot on the device. The value of the `S3Bucket` parameter is the name of the bucket which contains the VM image. The value of the `S3Key` parameter is the path to the VM image file in this bucket.

```
aws ec2 import-snapshot --disk-container "Format=RAW,UserBucket={S3Bucket=bucket-
name,S3Key=image-file}" --profile profile-name --endpoint http://snowball-ip:8008 --
region snow
```

For more information, see import-snapshot in the AWS CLI Command Reference.

This command doesn't support the following switches.

- [--client-data `value`]
- [--client-token `value`]
- [--dry-run]
- [--no-dry-run]
- [--encrypted]
- [--no-encrypted]

- [--kms-key-id `value`]
- [--tag-specifications `value`]

Example output of `import-snapshot` command

```
{
   "ImportTaskId":"s.import-snap-1234567890abc",
   "SnapshotTaskDetail":{
      "DiskImageSize":2.0,
      "Encrypted":false,
      "Format":"RAW",
      "Progress":"3",
      "Status":"active",
      "StatusMessage":"pending",
      "UserBucket":{
         "S3Bucket":"bucket",
         "S3Key":"vmimport/image01"
      }
   }
}
```

Note

Snow Family devices currently only allow one active import job to run at a time, per device. To start a new import task, either wait for the current task to finish, or choose another available node in a cluster. You can also choose to cancel the current import if you want. To prevent delays, don't reboot the Snow Family device while the import is in progress. If you reboot the device, the import will fail, and progress will be deleted when the device becomes accessible. To check the status of your snapshot import task status, use the following command:

```
aws ec2 describe-import-snapshot-tasks --import-task-ids id --profile profile-name --endpoint http://snowball-ip:8008 --region snow
```

Step 4: Register the snapshot as an AMI

When the snapshot import to the device is successful, you can register it using the `register-image` command.

Note

You can only register an AMI when all of its snapshots are available.

For more information, see register-image in the AWS CLI Command Reference.

Example of the `register-image` command

```
aws ec2 register-image \
--name ami-01 \
--description my-ami-01 \
--block-device-mappings "[{\"DeviceName\": \"/dev/sda1\",\"Ebs\":{\"Encrypted\":false,
\"DeleteOnTermination\":true,\"SnapshotId\":\"snapshot-id\",\"VolumeSize\":30}}]" \
--root-device-name /dev/sda1 \
--profile profile-name \
--endpoint http://snowball-ip:8008 \
--region snow
```

Following is an example of block device mapping JSON. For more information, see the block-device-mapping parameter of register-image in the AWS CLI Command Reference.

```
[
    {
        "DeviceName": "/dev/sda",
        "Ebs":
            {
                "Encrypted": false,
                "DeleteOnTermination": true,
                "SnapshotId": "snapshot-id",
                "VolumeSize": 30
            }
    }
]
```

Example of the `register-image` command

```
{
    "ImageId": "s.ami-8de47d2e397937318"
 }
```

Step 5: Launch an instance from the AMI

To launch an instance, see run-instances in the AWS CLI Command Reference.

The value of the `image-id` parameter is the value of the `ImageId` name as the output of the `register-image` command.

```
aws ec2 run-instances --image-id image-id --instance-type instance-type --
profile profile-name --endpoint http://snowball-ip:8008 --region snow
```

```
{
   "Instances":[
      {
         "SourceDestCheck":false,
         "CpuOptions":{
            "CoreCount":1,
            "ThreadsPerCore":2
         },
         "InstanceId":"s.i-12345a73123456d1",
         "EnaSupport":false,
         "ImageId":"s.ami-1234567890abcdefg",
         "State":{
            "Code":0,
            "Name":"pending"
         },
         "EbsOptimized":false,
         "SecurityGroups":[
            {
               "GroupName":"default",
               "GroupId":"s.sg-1234567890abc"
            }
         ],
         "RootDeviceName":"/dev/sda1",
         "AmiLaunchIndex":0,
         "InstanceType":"sbe-c.large"
      }
   ],
   "ReservationId":"s.r-1234567890abc"
}
```

> **Note**
>
> You can also use AWS OpsHub to launch the instance. For more information, see Launching an Amazon EC2-compatible instance in this guide.

Additional AMI actions

You can use additional AWS CLI commands to monitor snapshot import status, get details on snapshots that have been imported, canceling importing a snapshot, and deleting or deregistering snapshots after they have been imported.

Monitoring snapshot import status

To see the current state of the import progress, you can run the Amazon EC2 `describe-import-snapshot-tasks` command. This command supports pagination and filtering on the `task-state`.

Example of the `describe-import-snapshot-tasks` command

```
aws ec2 describe-import-snapshot-tasks --import-task-ids id --profile profile-name --
endpoint http://snowball-ip:8008 --region snow
```

Example of `describe-import-snapshot-tasks` command output

```
{
        "ImportSnapshotTasks": [
            {
                "ImportTaskId": "s.import-snap-8f6bfd7fc9ead9aca",
                "SnapshotTaskDetail": {
                    "Description": "Created by AWS-Snowball-VMImport service for
 s.import-snap-8f6bfd7fc9ead9aca",
                    "DiskImageSize": 8.0,
                    "Encrypted": false,
                    "Format": "RAW",
                    "Progress": "3",
                    "SnapshotId": "s.snap-848a22d7518ad442b",
                    "Status": "active",
                    "StatusMessage": "pending",
                    "UserBucket": {
                        "S3Bucket": "bucket1",
```

```
                        "S3Key": "image1"
                    }
                }
            }
        ]
}
```

> **Note**
>
> This command only shows output for tasks that have successfully completed or been marked as deleted within the last 7 days. Filtering only supports `Name=task-state, Values=active | deleting | deleted | completed`

This command doesn't support the following parameters.

- [--dry-run]
- [--no-dry-run]

Canceling an import task

To cancel an import task, run the `cancel-import-task` command.

Example of the `cancel-import-task` command

```
aws ec2 cancel-import-task --import-task-id import-task-id --profile profile-name --
endpoint http://snowball-ip:8008 --region snow
```

Example of `cancel-import-task` command output

```
{
        "ImportTaskId": "s.import-snap-8234ef2a01cc3b0c6",
        "PreviousState": "active",
        "State": "deleting"
}
```

> **Note**
>
> Only tasks that are not in a completed state can be canceled.

This command doesn't support the following parameters.

- [--dry-run]
- [--no-dry-run]

Describing snapshots

After a snapshot is imported, you can use this command to describe it. To filter the snapshots, you can pass in `snapshot-ids` with the snapshot ID from the previous import task response. This command supports pagination and filter on `volume-id`, `status`, and `start-time`.

Example of `describe-snapshots` command

```
aws ec2 describe-snapshots --snapshot-ids snapshot-id --profile profile-name --endpoint
 http://snowball-ip:8008 --region snow
```

Example of `describe-snapshots` command output

```
{
    "Snapshots": [
        {
            "Description": "Created by AWS-Snowball-VMImport service for s.import-
snap-8f6bfd7fc9ead9aca",
            "Encrypted": false,
            "OwnerId": "123456789012",
            "SnapshotId": "s.snap-848a22d7518ad442b",
            "StartTime": "2020-07-30T04:31:05.032000+00:00",
            "State": "completed",
            "VolumeSize": 8
        }
    ]
 }
```

This command doesn't support the following parameters.

- [--restorable-by-user-ids `value`]
- [--dry-run]
- [--no-dry-run]

Deleting a snapshot from a Snow Family device

To remove snapshots that you own and you no longer need, you can use the `delete-snapshot` command.

Example of the `delete-snapshot` command

```
aws ec2 delete-snapshot --snapshot-id snapshot-id --profile profile-name --endpoint
 http://snowball-ip:8008 --region snow
```

> **Note**
>
> Snowball Edge does not support deleting snapshots that are in a **PENDING** state or if it is designated as a root device for an AMI.

This command doesn't support the following parameters.

- [--dry-run]
- [--no-dry-run]

Deregistering an AMI

To deregister AMIs that you no longer need, you can run the `deregister-image` command. Deregistering an AMI that is in the **Pending** state is not currently supported.

Example of the `deregister-image` command

```
aws ec2 deregister-image --image-id image-id --profile profile-name --endpoint
 http://snowball-ip:8008 --region snow
```

This command doesn't support the following parameters.

- [--dry-run]
- [--no-dry-run]

Using the AWS CLI and API Operations on Snowball Edge

When using the AWS Command Line Interface (AWS CLI) or API operations to issue IAM, Amazon S3 and Amazon EC2 commands on Snowball Edge, you must specify the `region` as "snow." You can do this using `AWS configure` or within the command itself, as in the following examples.

```
aws configure --profile ProfileName
AWS Access Key ID [None]: defgh
AWS Secret Access Key [None]: 1234567
Default region name [None]: snow
Default output format [None]: json
```

Or

```
aws s3 ls --profile ProfileName --endpoint http://192.0.2.0:8080 --region snow
```

Quotas for Compute Instances on a Snowball Edge Device

The following are storage quotas and shared resource limitations for compute resources on an AWS Snowball Edge device.

Storage Quotas

The storage available for compute resources is a separate resource from the dedicated Amazon S3 storage on a Snowball Edge device. The quotas for storage are as follows:

- **Storage quotas for the Snowball Edge Storage Optimized option** – The total available storage for Amazon S3 is between 60 TB and 80 TB depending on whether you're using compute instances on the device. If you are using compute instances, then total available dedicated storage for `sbe1` compute instances for the Snowball Edge Storage Optimized option is 1,000 GB.
- **Storage quotas for the Snowball Edge Compute Optimized and with GPU options** – The total available dedicated storage for `sbe-c` and `sbe-g` instances is 7.68 TB. The total available storage remaining is 42 TB.

The following tables outline the available compute resources for Snowball Edge devices.

Feature	Limitation
Number of AMIs on a single Snowball Edge Storage Optimized option	10
Number of AMIs on a single Snowball Edge Compute Optimized option	20
Number of AMIs on a single Snowball Edge Compute Optimized with GPU option	20
Number of volumes per instance	10
Concurrently running (or stopped) instances	Varies depending on available resources

Instance type	vCPU cores	Memory (GiB)	GPUs	Supported device option
`sbe1.small`	1	1	0	storage optimized
`sbe1.medium`	1	2	0	storage optimized
`sbe1.large`	2	4	0	storage optimized
`sbe1.xlarge`	4	8	0	storage optimized
`sbe1.2xlarge`	8	16	0	storage optimized
`sbe1.4xlarge`	16	32	0	storage optimized
`sbe1.6xlarge`	24	32	0	storage optimized
`sbe-c.small`	1	2	0	compute optimized
`sbe-c.medium`	1	4	0	compute optimized
`sbe-c.large`	2	8	0	compute optimized
`sbe-c.xlarge`	4	16	0	compute optimized

Instance type	vCPU cores	Memory (GiB)	GPUs	Supported device option
sbe-c.2xlarge	8	32	0	compute optimized
sbe-c.4xlarge	16	64	0	compute optimized
sbe-c.8xlarge	32	128	0	compute optimized
sbe-c.12xlarge	48	192	0	compute optimized
sbe-c.16xlarge	64	256	0	compute optimized
sbe-c.24xlarge	96	384	0	compute optimized
sbe-g.small	1	2	1	with GPU
sbe-g.medium	1	4	1	with GPU
sbe-g.large	2	8	1	with GPU
sbe-g.xlarge	4	16	1	with GPU
sbe-g.2xlarge	8	32	1	with GPU
sbe-g.4xlarge	16	64	1	with GPU
sbe-g.8xlarge	32	128	1	with GPU
sbe-g.12xlarge	48	192	1	with GPU

Shared Compute Resource Limitations

All services on a Snowball Edge device use some of the finite resources on the device. A Snowball Edge device with its available compute resources maximized can't launch new compute resources. For example, if you try to start the NFS interface while also running a `sbe1.4xlarge` compute instance on a storage optimized device, the NFS interface service doesn't start. The following outlines the available resources on the different device options as well as resource requirements for each service.

- If no compute services are ACTIVE:
  - On a storage optimized option, you have 24 vCPUs and 32 GiB of memory for your compute instances.
  - On a compute optimized option, you have 52 vCPUs and 208 GiB of memory for your compute instances. This is also true for the with GPU option.
- While AWS IoT Greengrass and AWS Lambda powered by AWS IoT Greengrass are ACTIVE:
  - On a storage optimized option, these services use 4 vCPU cores and 8 GiB of memory.
  - On a compute optimized option, these services use 1 vCPU core and 1 GiB of memory. This is also true for the GPU option.
  - While the NFS interface is ACTIVE, it uses 8 vCPU cores and 16 GiB of memory on a Snowball Edge device.
  - While Amazon S3 compatible storage on Snow Family devices is ACTIVE:
    - On a Snowball Edge Compute Optimized with AMD EPYC Gen2 and NVME, for a single node with the minimum configuration of 3 TB of Amazon S3 compatible storage on Snow Family devices, it uses 8 vCPU cores and 16 GB of memory. For a single node with more than 3 TB of Amazon S3 compatible storage on Snow Family devices, it uses 20 vCPU cores and 40 GB of memory. For a cluster, it uses 20 vCPU cores and 40 GB of memory.
    - On a Snowball Edge Compute Optimized with AMD EPYC Gen1, HDD, and optional GPU, for a single node it uses 8 vCPU cores and 16 GB of memory. for a cluster, it uses 20 vCPU cores and 40 GB of memory.

You can determine whether a service is ACTIVE on a Snowball Edge by using the command `snowballEdge describe-service` on the Snowball Edge client. For more information, see Getting Service Status.

Creating a Compute Job

In this section, you create your first Amazon EC2-compatible compute instance job for an AWS Snowball Edge device.

Important

Be aware of the following points before you create your job:

- Make sure that the vCPU, memory, and storage values associated with your AMI match the type of instance that you want to create.

- If you're going to use Secure Shell (SSH) to connect to the instance *after* you launch the instance on your Snowball Edge, you must first perform the following procedure. You can't update the AMIs on your Snowball Edge after the fact. You must do this step before creating the job.

Configuring an AMI to Use SSH to Connect to Compute Instances Launched on the Device

To use Secure Shell (SSH) to connect to your compute instances on Snowball Edge devices, you must perform the following procedure. This procedure adds the SSH key to the AMI before creating your job. We also recommend that you use this procedure to set up your applications on the instance that you plan to use as the AMI for your job.

Important

If you don't follow this procedure, you can't connect to your instances with SSH when you receive your Snowball Edge device.

To put an SSH key into an AMI

1. Launch a new instance in the AWS Cloud based on the CentOS 7 (x86_64) - with Updates HVM, Ubuntu 16.04 LTS - Xenial (HVM), and Amazon Linux 2 AMI image, or Windows.

   When you launch your instance, make sure that the storage size that you assign to the instance is appropriate for your later use on the Snowball Edge. In the Amazon EC2 console, you do this in **Step 4: Add Storage**. For a list of the supported sizes for compute instance storage volumes on a Snowball Edge, see Quotas for Compute Instances on a Snowball Edge Device.
2. Install and configure the applications that you want to run on the Snowball Edge, and test that they work as expected.
3. Make a copy of the PEM/PPK file that you used for the SSH key pair to create this instance. Save this file to the server that you plan to use to communicate with the Snowball Edge. This file is required for using SSH to connect to the launched instance on your device, so make a note of the path to this file.
4. Save the instance as an AMI. For more information, see Creating an Amazon EBS-Backed Linux AMI in the *Amazon EC2 User Guide for Linux Instances.*

5. Repeat this procedure for each of the instances that you want to connect to using SSH. Make sure that you make copies of your different SSH key pairs and take note of the AMIs they're associated with.

Creating Your Job in the Console

Your next step is to create a job to order a Snow Family device. Your job can be of any job type, including a cluster. Using the AWS Snow Family Management Console, follow the instructions provided in see Creating a job to order a Snow Family device. When you get to the **Step 3: Give job details** page in the job creation wizard, perform the following additional steps.

1. Choose **Enable compute with EC2**.
2. Choose **Add an AMI**.
3. In the dialog box that opens, choose an AMI and then choose **Save**.
4. Add up to 20 total AMIs to your job, depending on device type.
5. Continue creating your job as normal.

Creating Your Job in the AWS CLI

You can also create your job using the AWS CLI. To do this, open a terminal and run the following command, replacing the red text with your actual values.

```
aws snowball create-job --job-type IMPORT --resources '{"S3Resources":
[{"BucketArn":"arn:aws:s3:::bucket-name"}],"Ec2AmiResources":
[{"AmiId":"ami-12345678"}]}' --description Example --address-
id ADIEXAMPLE60-1234-1234-5678-41fEXAMPLE57  --kms-key-arn arn:aws:kms:us-
west-2:012345678901:key/eEXAMPLE-1234-1234-5678-5b4EXAMPLE8e --role-
arn arn:aws:iam::012345678901:role/snowball-local-s3-lambda-us-west-2-role --snowball-
capacity-preference T100 --shipping-option SECOND_DAY --snowball-type EDGE
```

After it arrives and you unlock your device, use the Snowball Edge client to get your local credentials. For more information, see Getting Credentials.

Network Configuration for Compute Instances

After you launch your compute instances on a Snow Family device, you must provide it with an IP address by creating a network interface. Snow Family devices support two kinds of network interfaces, a virtual network interface and a direct network interface.

Virtual network interface (VNI)

A virtual network interface is the standard network interface for connecting to an EC2-compatible instance on your Snow Family device. You must create a VNI for each of your EC2-compatible instances regardless of whether you also use a direct network interface or not. The traffic passing through a VNI is protected by the security groups that you set up. You can only associate VNIs with the physical network port you use to control your Snow Family device.

> **Note**
>
> VNI will use the same physical interface (RJ45, SFP+, or QSFP) that is used to managed the Snow Family device. Creating a VNI on a different physical interface than the one being used for device management could lead to unexpected results.

Direct network interface (DNI)

A direct network interface (DNI) is an advanced network feature that enables use cases like multicast streams, transitive routing, and load balancing. By providing instances with layer 2 network access without any intermediary translation or filtering, you can gain increased flexibility over the network configuration of your Snow Family device and improved network performance. DNIs support VLAN tags and customizing the MAC address. Traffic on DNIs is not protected by security groups.

On Snowball Edge devices, DNIs can be associated with the RJ45, SFP, or QSFP ports. Each physical port supports a maximum of 63 DNIs. DNIs do not have to be associated to the same physical network port that you use to manage the Snow Family device.

> **Note**
>
> Snowball Edge storage optimized (with EC2 compute functionality) devices don't support DNIs.

Topics

Prerequisites

Before you configure a VNI or a DNI, be sure that you've done the following prerequisites.

1. Make sure there's power to your device and that one of your physical network interfaces, like the RJ45 port, is connected with an IP address.
2. Get the IP address associated with the physical network interface that you're using on the Snow Family device.
3. Configure your Snowball Edge client. For more information, see Configuring a Profile for the Snowball Edge Client.
4. Unlock the device. We recommend using AWS OpsHub for Snow Family to unlock your device. For instructions, see Unlocking a Device.

   If you want to use the CLI command, run the following command, and provide the information that appears in the dialog box.

   ```
   snowballEdge configure
   ```

   ```
   Snowball Edge Manifest Path: manifest.bin

   Unlock Code: unlock code

   Default Endpoint: https://device ip
   ```
5. Run the following command.

   ```
   snowballEdge unlock-device
   ```

   The device display update indicates that it is unlocked.
6. Launch an EC2-compatible instance on the device. You will associate the VNI with this instance.
7. Run the `snowballEdge describe-device` command to get the list of physical network interface IDs.
8. Identify the ID for the physical network interface that you want to use, and make a note of it.

Setting Up a Virtual Network Interface (VNI)

After you have identified the ID for your physical network interface, you can set up a virtual network interface (VNI). Use the following procedure set up a VNI. Make sure that you perform the prerequisite tasks before you create a VNI.

Create a VNI and associate IP address

1. Run the `snowballEdge create-virtual-network-interface` command. The following examples show running this command with the two different IP address assignment methods, either `DHCP` or `STATIC`. The `DHCP` method uses Dynamic Host Configuration Protocol (DHCP).

```
snowballEdge create-virtual-network-interface \
--physical-network-interface-id s.ni-abcd1234 \
--ip-address-assignment DHCP

        //OR//

snowballEdge create-virtual-network-interface \
--physical-network-interface-id s.ni-abcd1234 \
--ip-address-assignment STATIC \
--static-ip-address-configuration IpAddress=192.0.2.0,Netmask=255.255.255.0
```

   The command returns a JSON structure that includes the IP address. Make a note of that IP address for the `ec2 associate-address` AWS CLI command later in the process.

   Anytime you need this IP address, you can use the `snowballEdge describe-virtual-network-interfaces` Snowball Edge client command, or the `aws ec2 describe-addresses` AWS CLI command to get it.

2. To associate your newly created IP address with your instance, use the following command, replacing the red text with your values:

```
aws ec2 associate-address --public-ip 192.0.2.0 --instance-id s.i-01234567890123456
 --endpoint http://Snow Family device physical IP address:8008
```

Setting Up a Direct Network Interface (DNI)

> **Note**
>
> The direct network interface feature is available on or after January 12, 2021 and is available in all AWS Regions where Snow Family devices are available.

Prerequisites

Before you set up a direct network interface (DNI), you must perform the tasks in the prerequisites section.

1. Perform the prerequisite tasks before setting up the DNI. For instructions, see Prerequisites.
2. Additionally, you must launch an instance on your device, create a VNI, and associate it with the instance. For instructions, see Setting Up a Virtual Network Interface (VNI).

> **Note**
>
> If you added direct networking to your existing device by performing an in-the-field software update, you must restart the device twice to fully enable the feature.

Create a DNI and associate IP address

1. Create a direct network interface and attach it to the Amazon EC2-compatible instance by running the following command. You will need the MAC address of the device for the next step.

```
create-direct-network-interface [--endpoint endpoint] [--instance-id instanceId]
 [--mac macAddress]
                                [--physical-network-interface-
id physicalNetworkInterfaceId]
                                [--unlock-code unlockCode] [--vlan vlanId]
```

   OPTIONS

`--endpoint <endpoint>` The endpoint to send this request to. The endpoint for your devices will be a URL using the `https` scheme followed by an IP address. For example, if the IP address for your device is 123.0.1.2, the endpoint for your device would be https://123.0.1.2.

`--instance-id <instanceId>` The EC2-compatible instance ID to attach the interface to (optional).

`--mac <macAddress>` Sets the MAC address of the network interface (optional).

`--physical-network-interface-id <physicalNetworkInterfaceId>` The ID for the physical network interface on which to create a new virtual network interface. You can determine the physical network interfaces available on your Snowball Edge using the `describe-device` command.

`--vlan <vlanId>` Set the assigned VLAN for the interface (optional). When specified, all traffic sent from the interface is tagged with the specified VLAN ID. Incoming traffic is filtered for the specified VLAN ID, and has all VLAN tags stripped before being passed to the instance.

2. If you didn't associate your DNI with an instance in step 1, you can associate it by running the Updating a Direct Network Interface command.
3. After you create a DNI and associate it with your EC2-compatible instance, you must make two configuration changes inside your Amazon EC2-compatible instance.

   - The first is to change ensure that packets meant for the VNI associated with the EC2-compatible instance are sent through eth0.
   - The second change configures your direct network interface to use either DCHP or static IP when booting.

   The following are examples of shell scripts for Amazon Linux 2 and CentOS Linux that make these configuration changes.

   Amazon Linux 2

```
# Mac address of the direct network interface.
# You got this when you created the direct network interface.
DNI_MAC=[MAC ADDRESS FROM CREATED DNI]

# Configure routing so that packets meant for the VNI always are sent through
 eth0.
```

```
PRIVATE_IP=$(curl -s http://169.254.169.254/latest/meta-data/local-ipv4)
PRIVATE_GATEWAY=$(ip route show to match 0/0 dev eth0 | awk '{print $3}')
ROUTE_TABLE=10001
echo "from $PRIVATE_IP table $ROUTE_TABLE" > /etc/sysconfig/network-scripts/
rule-eth0
echo "default via $PRIVATE_GATEWAY dev eth0 table $ROUTE_TABLE" > /etc/
sysconfig/network-scripts/route-eth0
echo "169.254.169.254 dev eth0" >> /etc/sysconfig/network-scripts/route-eth0

# Query the persistent DNI name, assigned by udev via ec2net helper.
#   changable in /etc/udev/rules.d/70-persistent-net.rules
DNI=$(ip --oneline link | grep -i $DNI_MAC | awk -F ': ' '{ print $2 }')

# Configure DNI to use DHCP on boot.
cat << EOF > /etc/sysconfig/network-scripts/ifcfg-$DNI
DEVICE="$DNI"
NAME="$DNI"
HWADDR=$DNI_MAC
ONBOOT=yes
NOZEROCONF=yes
BOOTPROTO=dhcp
TYPE=Ethernet
MAINROUTETABLE=no
EOF

# Make all changes live.
systemctl restart network
```

CentOS Linux

```
# Mac address of the direct network interface. You got this when you created the
 direct network interface.
DNI_MAC=[MAC ADDRESS FROM CREATED DNI]
# The name to use for the direct network interface. You can pick any name that
 isn't already in use.
DNI=eth1

# Configure routing so that packets meant for the VNIC always are sent through
 eth0
PRIVATE_IP=$(curl -s http://169.254.169.254/latest/meta-data/local-ipv4)
PRIVATE_GATEWAY=$(ip route show to match 0/0 dev eth0 | awk '{print $3}')
ROUTE_TABLE=10001
```

```
echo from $PRIVATE_IP table $ROUTE_TABLE > /etc/sysconfig/network-scripts/rule-
eth0
echo default via $PRIVATE_GATEWAY dev eth0 table $ROUTE_TABLE > /etc/sysconfig/
network-scripts/route-eth0

# Configure your direct network interface to use DHCP on boot.
cat << EOF > /etc/sysconfig/network-scripts/ifcfg-$DNI
DEVICE="$DNI"
NAME="$DNI"
HWADDR="$DNI_MAC"
ONBOOT=yes
NOZEROCONF=yes
BOOTPROTO=dhcp
TYPE=Ethernet
EOF

# Rename DNI device if needed.
CURRENT_DEVICE_NAME=$(LANG=C ip -o link | awk -F ': ' -vIGNORECASE=1 '!/link\/
ieee802\.11/ && /'"$DNI_MAC"'/ { print $2 }')
ip link set $CURRENT_DEVICE_NAME name $DNI

# Make all changes live.
systemctl restart network
```

Using SSH to connect to compute instances on a Snow Family device

To use Secure Shell (SSH) to connect to compute instances on a Snow Family device, you have the following options for providing or creating an SSH key.

- You can provide the SSH key for the Amazon Machine Image (AMI) when you create a job to order a device. For more information, see Configuring an AMI to Use SSH to Connect to Compute Instances Launched on the Device.
- You can provide the SSH key for the AMI when you create a virtual machine image to import to a Snow Family device. For more information, see Importing a virtual machine image to a Snow Family device.
- You can create a key pair on the Snow Family device and choose to launch an instance with that locally generated public key. For more information, see Create a key pair using Amazon EC2 in the Amazon EC2 User Guide for Linux Instances.

To connect to an instance through SSH

1. Make sure that your device is powered on, connected to the network, and unlocked. For more information, see Connecting to Your Local Network.
2. Make sure that you have your network settings configured for your compute instances. For more information, see Network Configuration for Compute Instances.
3. Check your notes to find the PEM or PPK key pair that you used for this specific instance. Make a copy of those files somewhere on your computer. Make a note of the path to the PEM file.
4. Connect to your instance through SSH as in the following example command. The IP address is the IP address of the virtual network interface (VNIC) that you set up in Network Configuration for Compute Instances.

   ```
   ssh -i path/to/PEM/key/file instance-user-name@192.0.2.0
   ```

   For more information, see Connecting to Your Linux Instance Using SSH in the *Amazon EC2 User Guide for Linux Instances.*

Transferring Data from EC2-compatible Compute Instances to S3 Buckets on the Same Snowball Edge

You can transfer data between compute instances and Amazon S3 buckets on the same Snowball Edge device. You do this by using the supported AWS CLI commands and the appropriate endpoints. For example, assume that you want to move data from a directory in my `sbe1.xlarge` instance into the Amazon S3 bucket, `myBucket` on the same device. Assume that you're using the Amazon S3 compatible storage on Snow Family devices endpoint `https://S3-object-API-endpoint:443`. You use the following procedure.

> **Note**
>
> This procedure only works if you've followed the instructions in Configuring an AMI to Use SSH to Connect to Compute Instances Launched on the Device.

To transfer data between a compute instance and a bucket on the same Snowball Edge

1. Use SSH to connect to your compute instance.

2. Download and install the AWS CLI. If your instance doesn't already have the AWS CLI, download and install it. For more information, see Installing the AWS Command Line Interface.
3. Configure the AWS CLI on your compute instance to work with the Amazon S3 endpoint on the Snowball Edge. For more information, see Getting and using local Amazon S3 credentials.
4. Use the supported Amazon S3 compatible storage on Snow Family devices commands to transfer data. For example:

```
aws s3 cp ~/june2018/results s3://myBucket/june2018/results --recursive --endpoint
 https://S3-object-API-endpoint:443
```

Snowball Edge Client Commands for Compute Instances

The Snowball Edge client is a standalone terminal application that you can run on your local server. You can use it to perform some administrative tasks on your Snowball Edge device or cluster of devices. For more information about how to use the Snowball Edge client, including how to start and stop services with it, see Using the Snowball Edge Client.

Following, you can find information about the Snowball Edge client commands that are specific to compute instances, including examples of use.

For a list of Amazon EC2-compatible commands you can use on your AWS Snowball Edge device, see Supported Amazon EC2-compatible AWS CLI Commands on a Snowball Edge.

Creating a Launch Configuration to Autostart Amazon EC2-compatible Instances

To automatically start Amazon EC2-compatible compute instances on your AWS Snowball Edge device after it is unlocked, you can create a launch configuration. To do so, use the `snowballEdge create-autostart-configuration` command, as shown following.

Usage

```
snowballEdge create-autostart-configuration --physical-connector-type [SFP_PLUS or RJ45
 or QSFP] --ip-address-assignment [DHCP or STATIC] [--static-ip-address-configuration
 IpAddress=[IP address],NetMask=[Netmask]] --launch-template-id [--launch-template-
version]
```

Updating a Launch Configuration to Autostart EC2-compatible Instances

To update an existing launch configuration on your Snowball Edge, use the `snowballEdge update-autostart-configuration` command. You can find its usage following. To enable or disable a launch configuration, specify the `--enabled` parameter.

Usage

```
snowballEdge update-autostart-configuration --autostart-configuration-arn [--physical-connector-type [SFP_PLUS or RJ45 or QSFP]] [--ip-address-assignment [DHCP or STATIC]]
 [--static-ip-address-configuration IpAddress=[IP address],NetMask=[Netmask]][--launch-template-id] [--launch-template-version] [--enabled]
```

Deleting a Launch Configuration to Autostart EC2-compatible Instances

To delete a launch configuration that's no longer in use, use the `snowballEdge delete-autostart-configuration` command, as follows.

Usage

```
snowballEdge delete-autostart-configuration --autostart-configuration-arn
```

Listing Launch Configurations to Autostart EC2-compatible Instances

To list the launch configurations that you have created on your Snowball Edge, use the `describe-autostart-configurations` command, as follows.

Usage

```
snowballEdge describe-autostart-configurations
```

Creating a Virtual Network Interface

To run a compute instance on your Snowball Edge or start the NFS interface on your Snowball Edge, you first create a virtual network interface (VNIC). Each Snowball Edge has three network interfaces (NICs), the physical network interface controllers for the device. These are the RJ45, SFP, and QSFP ports on the back of the device.

Each VNIC is based on a physical one, and you can have any number of VNICs associated with each NIC. To create a virtual network interface, use the `snowballEdge create-virtual-network-interface` command.

> **Note**
>
> The `--static-ip-address-configuration` parameter is valid only when using the `STATIC` option for the `--ip-address-assignment` parameter.

Usage

You can use this command in two ways: with the Snowball Edge client configured, or without the Snowball Edge client configured. The following usage example shows the method with the Snowball Edge client configured.

```
snowballEdge create-virtual-network-interface --ip-address-assignment [DHCP or STATIC]
 --physical-network-interface-id [physical network interface id] --static-ip-address-
configuration IpAddress=[IP address],NetMask=[Netmask]
```

The following usage example shows the method without the Snowball Edge client configured.

```
snowballEdge create-virtual-network-interface --endpoint https://[ip address]
 --manifest-file /path/to/manifest --unlock-code [unlock code] --ip-address-
assignment [DHCP or STATIC] --physical-network-interface-id [physical network interface
 id] --static-ip-address-configuration IpAddress=[IP address],NetMask=[Netmask]
```

Example Example: Creating VNICs (Using DHCP)

```
snowballEdge create-virtual-network-interface --ip-address-assignment dhcp --physical-
network-interface-id s.ni-8EXAMPLEaEXAMPLEd
{
  "VirtualNetworkInterface" : {
    "VirtualNetworkInterfaceArn" : "arn:aws:snowball-device:::interface/
s.ni-8EXAMPLE8EXAMPLEf",
    "PhysicalNetworkInterfaceId" : "s.ni-8EXAMPLEaEXAMPLEd",
    "IpAddressAssignment" : "DHCP",
    "IpAddress" : "192.0.2.0",
    "Netmask" : "255.255.255.0",
    "DefaultGateway" : "192.0.2.1",
    "MacAddress" : "EX:AM:PL:E1:23:45"
  }
}
```

Describing Your Virtual Network Interfaces

To describe the VNICs that you previously created on your device, use the `snowballEdge describe-virtual-network-interfaces` command. You can find its usage following.

Usage

You can use this command in two ways: with the Snowball Edge client configured, or without the Snowball Edge client configured. The following usage example shows the method with the Snowball Edge client configured.

```
snowballEdge describe-virtual-network-interfaces
```

The following usage example shows the method without the Snowball Edge client configured.

```
snowballEdge describe-virtual-network-interfaces --endpoint https://[ip address] --
manifest-file /path/to/manifest --unlock-code [unlock code]
```

Example Example: Describing VNICs

```
snowballEdge describe-virtual-network-interfaces
[
  {
    "VirtualNetworkInterfaceArn" : "arn:aws:snowball-device:::interface/
s.ni-8EXAMPLE8EXAMPLE8",
    "PhysicalNetworkInterfaceId" : "s.ni-8EXAMPLEaEXAMPLEd",
    "IpAddressAssignment" : "DHCP",
    "IpAddress" : "192.0.2.0",
    "Netmask" : "255.255.255.0",
    "DefaultGateway" : "192.0.2.1",
    "MacAddress" : "EX:AM:PL:E1:23:45"
  },{
    "VirtualNetworkInterfaceArn" : "arn:aws:snowball-device:::interface/
s.ni-1EXAMPLE1EXAMPLE1",
    "PhysicalNetworkInterfaceId" : "s.ni-8EXAMPLEaEXAMPLEd",
    "IpAddressAssignment" : "DHCP",
    "IpAddress" : "192.0.2.2",
    "Netmask" : "255.255.255.0",
    "DefaultGateway" : "192.0.2.1",
    "MacAddress" : "12:34:5E:XA:MP:LE"
  }
```

```
]
```

Updating a Virtual Network Interface

After creating a virtual network interface (VNIC), you can update its configuration using the `snowballEdge update-virtual-network-interface` command. After providing the Amazon Resource Name (ARN) for a particular VNIC, you provide values only for whatever elements you are updating.

Usage

You can use this command in two ways: with the Snowball Edge client configured, or without the Snowball Edge client configured. The following usage example shows the method with the Snowball Edge client configured.

```
snowballEdge update-virtual-network-interface --virtual-network-interface-arn [virtual
 network-interface-arn] --ip-address-assignment [DHCP or STATIC] --physical-network-
interface-id [physical network interface id] --static-ip-address-configuration
 IpAddress=[IP address],NetMask=[Netmask]
```

The following usage example shows the method without the Snowball Edge client configured.

```
snowballEdge update-virtual-network-interface --endpoint https://[ip address] --
manifest-file /path/to/manifest --unlock-code [unlock code] --virtual-network-
interface-arn [virtual network-interface-arn] --ip-address-assignment [DHCP or STATIC]
 --physical-network-interface-id [physical network interface id] --static-ip-address-
configuration IpAddress=[IP address],NetMask=[Netmask]
```

Example Example: Updating a VNIC (Using DHCP)

```
snowballEdge update-virtual-network-interface --virtual-network-interface-arn
 arn:aws:snowball-device:::interface/s.ni-8EXAMPLEbEXAMPLEd --ip-address-assignment
 dhcp
```

Deleting a Virtual Network Interface

To delete a virtual network interface, you can use the `snowballEdge delete-virtual-network-interface` command.

Usage

You can use this command in two ways: with the Snowball Edge client configured, or without the Snowball Edge client configured. The following usage example shows the method with the Snowball Edge client configured.

```
snowballEdge delete-virtual-network-interface --virtual-network-interface-arn [virtual
 network-interface-arn]
```

The following usage example shows the method without the Snowball Edge client configured.

```
snowballEdge delete-virtual-network-interface --endpoint https://[ip address] --
manifest-file /path/to/manifest --unlock-code [unlock code] --virtual-network-
interface-arn [virtual network-interface-arn]
```

Example Example: Deleting a VNIC

```
snowballEdge delete-virtual-network-interface --virtual-network-interface-arn
 arn:aws:snowball-device:::interface/s.ni-8EXAMPLEbEXAMPLEd
```

Using the Amazon EC2-compatible Endpoint

Following, you can find an overview of the Amazon EC2-compatible endpoint. Using this endpoint, you can manage your Amazon Machine Images (AMIs) and compute instances programmatically using Amazon EC2-compatible API operations.

Specifying the Amazon EC2-compatible Endpoint as the AWS CLI Endpoint

When you use the AWS CLI to issue a command to the AWS Snowball Edge device, you can specify that the endpoint is the Amazon EC2-compatible endpoint. You have the choice of using the HTTPS endpoint, or an unsecured HTTP endpoint, as shown following.

HTTPS secured endpoint

```
aws ec2 describe-instances --endpoint https://192.0.2.0:8243 --ca-bundle path/to/
certificate
```

HTTP unsecured endpoint

```
aws ec2 describe-instances --endpoint http://192.0.2.0:8008
```

If you use the HTTPS endpoint of 8243, your data in transit is encrypted. This encryption is ensured with a certificate that's generated by the Snowball Edge when it is unlocked. After you have your certificate, you can save it to a local `ca-bundle.pem` file. Then you can configure your AWS CLI profile to include the path to your certificate, as described following.

To associate your certificate with the Amazon EC2-compatible endpoint

1. Connect the Snowball Edge to power and network, and turn it on.
2. After the device has finished unlocking, make a note of its IP address on your local network.
3. From a terminal on your network, make sure you can ping the Snowball Edge.
4. Run the `snowballEdge get-certificate` command in your terminal. For more information on this command, see Managing public key certificates.
5. Save the output of the `snowballEdge get-certificate` command to a file, for example `ca-bundle.pem`.
6. Run the following command from your terminal.

```
aws configure set profile.snowballEdge.ca_bundle /path/to/ca-bundle.pem
```

After you complete the procedure, you can run CLI commands with these local credentials, your certificate, and your specified endpoint.

Supported Amazon EC2-compatible AWS CLI Commands on a Snowball Edge

You can manage your compute instances on a Snow Family device through an Amazon EC2-compatible endpoint. This type of endpoint supports many of the Amazon EC2 CLI commands and actions of the AWS SDKs. For information about installing and setting up the AWS CLI, including specifying which AWS Regions you want to make AWS CLI calls against, see the AWS Command Line Interface User Guide.

List of Supported Amazon EC2-compatible AWS CLI Commands on a Snowball Edge

Following, you can find a description of the subset of AWS CLI commands and options for Amazon EC2 that are supported on Snowball Edge devices. If a command or option isn't listed following, it's not supported. You can declare some unsupported options along with a command. However, these are ignored.

- associate-address – Associates a virtual IP address with an instance for use on one of the three physical network interfaces on the device:
  - --instance-id – The ID of a single `sbe` instance.
  - --public-ip – The virtual IP address that you want to use to access your instance.
- attach-volume – Attaches an Amazon EBS volume to a stopped or running instance on your device and exposes it to the instance with the specified device name.
  - --device `value` – The device name.
  - --instance-id – The ID of a target Amazon EC2-compatible instance.
  - --volume-id `value` – The ID of the EBS volume.
- authorize-security-group-egress – Adds one or more egress rules to a security group for use with a Snowball Edge device. Specifically, this action permits instances to send traffic to one or more destination IPv4 CIDR address ranges. For more information, see Security Groups in Snowball Edge Devices.
  - --group-id `value` – The ID of the security group
  - [--ip-permissions `value`] – One or more sets of IP permissions.
- authorize-security-group-ingress – Adds one or more ingress rules to a security group. When calling `authorize-security-group-ingress`, you must specify a value either for `group-name` or `group-id`.
  - [--group-name `value`] – The name of the security group.
  - [--group-id `value`] – The ID of the security group
  - [--ip-permissions `value`] – One or more sets of IP permissions.
  - [--protocol `value`] The IP protocol. Possible values are `tcp`, udp, and `icmp`. The `--port` argument is required unless the "all protocols" value is specified (-1).
  - [--port `value`] – For TCP or UDP, the range of ports to allow. This value can be a single integer or a range (minimum–maximum).

    For ICMP, a single integer or a range (`type-code`) in which `type` represents the ICMP type number and `code` represents the ICMP code number. A value of -1 indicates all ICMP codes for all ICMP types. A value of -1 just for `type` indicates all ICMP codes for the specified ICMP type.
  - [--cidr `value`] – The CIDR IP range.
- create-launch-template – Creates a launch template. A *launch template* contains the parameters to launch an instance. When you launch an instance using `RunInstances`, you can specify a

launch template instead of providing the launch parameters in the request. You can create up to 100 templates per device.

- --launch-template-name `string` – A name for the launch template.
- --launch-template-data `structure` – The information for the launch template. The following attributes are supported:
  - `ImageId`
  - `InstanceType`
  - `SecurityGroupIds`
  - `TagSpecifications`
  - `UserData`

  JSON syntax:

```
{
    "ImageId":"string",
    "InstanceType":"sbe-c.large",
    "SecurityGroupIds":["string", ...],
    "TagSpecifications":[{"ResourceType":"instance","Tags":
[{"Key":"Name","Value":"Test"},
    {"Key":"Stack","Value":"Gamma"}]}],
    "UserData":"this is my user data"
}
```

- [--version-description `string`] – A description for the first version of the launch template.
- --endpoint `snowballEndpoint` – A value that enables you to manage your compute instances programmatically using Amazon EC2-compatible API operations. For more information, see Specifying the Amazon EC2-compatible Endpoint as the AWS CLI Endpoint.

- create-launch-template-version – Creates a new version for a launch template. You can specify an existing version of a launch template from which to base the new version. Launch template versions are numbered in the order in which they are created. You can't specify, change, or replace the numbering of launch template versions. You can create up to 100 versions of each launch template.

  Specify either the launch template ID or launch template name in the request.

  - --launch-template-id `string` – The ID of the launch template.
  - --launch-template-name `string` – A name for the launch template.

- --launch-template-data `structure` – The information for the launch template. The following attributes are supported:
  - `ImageId`
  - `InstanceType`
  - `SecurityGroupIds`
  - `TagSpecifications`
  - `UserData`

  JSON syntax:

```
{
    "ImageId":"string",
    "InstanceType":"sbe-c.large",
    "SecurityGroupIds":["string", ...],
    "TagSpecifications":[{"ResourceType":"instance","Tags":
[{"Key":"Name","Value":"Test"},
     {"Key":"Stack","Value":"Gamma"}]}],
    "UserData":"this is my user data"
}
```

- [--source-version `string`] – The version number of the launch template on which to base the new version. The new version inherits the same launch parameters as the source version, except for parameters that you specify in `launch-template-data`.
- [--version-description `string`] – A description for the first version of the launch template.
- --endpoint `snowballEndpoint` – A value that enables you to manage your compute instances programmatically using Amazon EC2-compatible API operations. For more information, see Specifying the Amazon EC2-compatible Endpoint as the AWS CLI Endpoint.

- create-tags – Adds or overwrites one or more tags for the specified resource. Each resource can have a maximum of 50 tags. Each tag consists of a key and optional value. Tag keys must be unique for a resource. The following resources are supported:
  - AMI
  - Instance
  - Launch template
  - Security group
  - Key pair

- create-security-group – Creates a security group on your Snowball Edge. You can create up to 50 security groups. When you create a security group, you specify a friendly name of your choice:
  - --group-name `value` – The name of the security group.
  - --description `value` – A description of the security group. This is informational only. This value can be up to 255 characters in length.
- create-volume – Creates an Amazon EBS volume that can be attached to an instance on your device.
  - [--size `value`] – The size of the volume in GiBs, which can be from 1 GiB to 1 TB (1000 GiBs).
  - [--snapshot-id `value`] – The snapshot from which to create the volume.
  - [--volume-type `value`] – The volume type. If no value is specified, the default is `sbg1`. Possible values include the following:
    - `sbg1` for magnetic volumes
    - `sbp1` for SSD volumes
  - [--tag-specification `value`] – A list of tags to apply to the volume during creation.
- delete-launch-template – Deletes a launch template. Deleting a launch template deletes all of its versions.

  Specify either the launch template ID or launch template name in the request.
  - --launch-template-id `string` – The ID of the launch template.
  - --launch-template-name `string` – A name for the launch template.
  - --endpoint `snowballEndpoint` – A value that enables you to manage your compute instances programmatically using Amazon EC2-compatible API operations. For more information, see Specifying the Amazon EC2-compatible Endpoint as the AWS CLI Endpoint.
- delete-launch-template-version – Deletes one or more versions of a launch template. You can't delete the default version of a launch template; you must first assign a different version as the default. If the default version is the only version for the launch template, delete the entire launch template by using the `delete-launch-template` command.

  Specify either the launch template ID or launch template name in the request.
  - --launch-template-id `string` – The ID of the launch template.
  - --launch-template-name `string` – A name for the launch template.
  - --versions (list) `"string" "string"` – The version numbers of one or more launch template versions to delete.

  - --endpoint `snowballEndpoint` – A value that enables you to manage your compute instances programmatically using Amazon EC2-compatible API operations. For more information, see Specifying the Amazon EC2-compatible Endpoint as the AWS CLI Endpoint.
- delete-security-group – Deletes a security group.

  If you attempt to delete a security group that is associated with an instance, or is referenced by another security group, the operation fails with `DependencyViolation`.

  - --group-name `value` – The name of the security group.
  - --description `value` – A description of the security group. This is informational only. This value can be up to 255 characters in length.
- delete-tags – Deletes the specified set of tags from the specified resource (AMI, compute instance, launch template, or security group).
- delete-volume – Deletes the specified Amazon EBS volume. The volume must be in the `available` state (not attached to an instance).
  - --volume-id `value` – The ID of the volume.
- describe-addresses – Describes one or more of your virtual IP addresses associated with the same number of `sbe` instances on your device.
  - --public-ips – One or more of the virtual IP addresses associated with your instances.
- describe-images – Describes one or more of the images (AMIs) available to you. Images available to you are added to the Snowball Edge device during job creation.
  - --image-id – The Snowball AMI ID of the AMI.
- describe-instance-attribute – Describes the specified attribute of the specified instance. You can specify only one attribute at a time. The following attributes are supported:
  - `instanceInitiatedShutdownBehavior`
  - `instanceType`
  - `userData`
- describe-instances – Describes one or more of your instances. The response returns any security groups that are assigned to the instances.
  - --instance-ids – The IDs of one or more `sbe` instances that were stopped on the device.
  - --page-size – The size of each page to get in the call. This value doesn't affect the number of items returned in the command's output. Setting a smaller page size results in more calls to the device, retrieving fewer items in each call. Doing this can help prevent the calls from timing out.

- --max-items – The total number of items to return in the command's output. If the total number of items available is more than the value specified, `NextToken` is provided in the command's output. To resume pagination, provide the `NextToken` value in the `starting-token` argument of a subsequent command.
- --starting-token – A token to specify where to start paginating. This token is the `NextToken` value from a previously truncated response.

- describe-instance-status – Describes the status of the specified instances or all of your instances. By default, only running instances are described, unless you specifically indicate to return the status of all instances. Instance status includes the following components:
  - **Status checks** – Snow device performs status checks on running Amazon EC2-compatible instances to identify hardware and software issues.
  - **Instance state** – You can manage your instances from the moment you launch them through their termination.

  With this command following filters are supported.

  - `[--filters]` (list)

    The filters.

    - `instance-state-code` – The code for the instance state, as a 16-bit unsigned integer. The high byte is used for internal service reporting purposes and should be ignored. The low byte is set based on the state represented. The valid values are 0 (pending), 16 (running), 32 (shutting-down), 48 (terminated), 64 (stopping), and 80 (stopped).
    - `instance-state-name` – The state of the instance (`pending` | `running` | `shutting-down` | `terminated` | `stopping` | `stopped`).
    - `instance-status.reachability` – Filters on instance status where the name is `reachability` (`passed` | `failed` | `initializing` | `insufficient-data`).
    - `instance-status.status` – The status of the instance (ok | `impaired` | `initializing` | `insufficient-data` | `not-applicable`).
    - `system-status.reachability` – Filters on system status where the name is reachability (`passed` | `failed` | `initializing` | `insufficient-data`).
    - `system-status.status` – The system status of the instance (ok | `impaired` | `initializing` | `insufficient-data` | `not-applicable`).
  - JSON Syntax:

```
[
  {
    "Name": "string",
    "Values": ["string", ...]
  }
  ...
]
```

- `[--instance-ids]` (list)

  The instance IDs.

  Default: Describes all of your instances.

- `[--dry-run|--no-dry-run]` (boolean)

  Checks whether you have the required permissions for the action, without actually making the request, and provides an error response. If you have the required permissions, the error response is `DryRunOperation`.

  Otherwise, it is `UnauthorizedOperation`.

- `[--include-all-instances | --no-include-all-instances]` (boolean)

  When `true`, includes the health status for all instances. When `false`, includes the health status for running instances only.

  Default: `false`

- `[--page-size]` (integer) – The size of each page to get in the call. This value doesn't affect the number of items returned in the command's output. Setting a smaller page size results in more calls to the device, retrieving fewer items in each call. Doing this can help prevent the calls from timing out.
- `[--max-items]` (integer) – The total number of items to return in the command's output. If the total number of items available is more than the value specified, `NextToken` is provided in the command's output. To resume pagination, provide the `NextToken` value in the `starting-token` argument of a subsequent command.
- `[--starting-token]` (string) – A token to specify where to start paginating. This token is the `NextToken` value from a previously truncated response.

- describe-launch-templates – Describes one or more launch templates. The `describe-launch-templates` command is a paginated operation. You can make multiple calls to retrieve the entire dataset of results.

  Specify either the launch template IDs or launch template names in the request.

  - --launch-template-ids (list) `"string" "string"` – A list of IDs of the launch templates.
  - --launch-template-names (list) `"string" "string"` – A list of names for the launch templates.
  - --page-size – The size of each page to get in the call. This value doesn't affect the number of items returned in the command's output. Setting a smaller page size results in more calls to the device, retrieving fewer items in each call. Doing this can help prevent the calls from timing out.
  - --max-items – The total number of items to return in the command's output. If the total number of items available is more than the value specified, `NextToken` is provided in the command's output. To resume pagination, provide the `NextToken` value in the `starting-token` argument of a subsequent command.
  - --starting-token – A token to specify where to start paginating. This token is the `NextToken` value from a previously truncated response.
  - --endpoint `snowballEndpoint` – A value that enables you to manage your compute instances programmatically using Amazon EC2-compatible API operations. For more information, see Specifying the Amazon EC2-compatible Endpoint as the AWS CLI Endpoint.
- describe-launch-template-versions – Describes one or more versions of a specified launch template. You can describe all versions, individual versions, or a range of versions. The `describe-launch-template-versions` command is a paginated operation. You can make multiple calls to retrieve the entire dataset of results.

  Specify either the launch template IDs or launch template names in the request.

  - --launch-template-id `string` – The ID of the launch template.
  - --launch-template-name `string` – A name for the launch template.
  - [--versions (list) `"string" "string"`] – The version numbers of one or more launch template versions to delete.
  - [--min-version `string`] – The version number after which to describe launch template versions.
  - [--max-version `string`] – The version number up to which to describe launch template versions.

- --page-size – The size of each page to get in the call. This value doesn't affect the number of items returned in the command's output. Setting a smaller page size results in more calls to the device, retrieving fewer items in each call. Doing this can help prevent the calls from timing out.
- --max-items – The total number of items to return in the command's output. If the total number of items available is more than the value specified, `NextToken` is provided in the command's output. To resume pagination, provide the `NextToken` value in the `starting-token` argument of a subsequent command.
- --starting-token – A token to specify where to start paginating. This token is the `NextToken` value from a previously truncated response.
- --endpoint `snowballEndpoint` – A value that enables you to manage your compute instances programmatically using Amazon EC2-compatible API operations. For more information, see Specifying the Amazon EC2-compatible Endpoint as the AWS CLI Endpoint.

- describe-security-groups – Describes one or more of your security groups.

  The `describe-security-groups` command is a paginated operation. You can issue multiple API calls to retrieve the entire dataset of results.

  - [--group-name `value`] – The name of the security group.
  - [--group-id `value`] – The ID of the security group.
  - [--page-size `value`] – The size of each page to get in the AWS service call. This size doesn't affect the number of items returned in the command's output. Setting a smaller page size results in more calls to the AWS service, retrieving fewer items in each call. This approach can help prevent the AWS service calls from timing out. For usage examples, see Pagination in the *AWS Command Line Interface User Guide*.
  - [--max-items `value`] – The total number of items to return in the command's output. If the total number of items available is more than the value specified, `NextToken` is provided in the command's output. To resume pagination, provide the `NextToken` value in the `starting-token` argument of a subsequent command. Don't use the `NextToken` response element directly outside of the AWS CLI. For usage examples, see Pagination in the *AWS Command Line Interface User Guide*.
  - [--starting-token `value`] – A token to specify where to start paginating. This token is the `NextToken` value from a previously truncated response. For usage examples, see Pagination in the *AWS Command Line Interface User Guide*.

- describe-tags – Describes one or more of the tags for specified resource (`image`, `instance`, or security group). With this command, the following filters are supported:
  - launch-template
  - resource-id
  - resource-type – `image` or `instance`
  - key
  - value
- describe-volumes – Describes the specified Amazon EBS volumes.
  - [--max-items `value`] – The total number of items to return in the command's output. If the total number of items available is more than the value specified, `NextToken` is provided in the command's output. To resume pagination, provide the `NextToken` value in the `starting-token` argument of a subsequent command.
  - [--starting-token `value`] – A token to specify where to start paginating. This token is the `NextToken` value from a previously truncated response.
  - [--volume-ids `value`] – One or more volume IDs.
- detach-volume – Detaches an Amazon EBS volume from a stopped or running instance.
  - [--device `value`] – The device name.
  - [--instance-id] – The ID of a target Amazon EC2 instance.
  - --volume-id `value` – The ID of the volume.
- disassociate-address – Disassociates a virtual IP address from the instance it's associated with.
  - --public-ip – The virtual IP address that you want to disassociate from your instance.
- get-launch-template-data – Retrieves the configuration data of the specified instance. You can use this data to create a launch template.
  - --instance-id – The ID of a single `sbe` instance.
  - --endpoint `snowballEndpoint` – A value that enables you to manage your compute instances programmatically using Amazon EC2-compatible API operations. For more information, see Specifying the Amazon EC2-compatible Endpoint as the AWS CLI Endpoint.
- modify-launch-template – Modifies a launch template. You can specify which version of the launch template to set as the default version. When you launch an instance without specifying a launch template version, the default version of the launch template applies.

  Specify either the launch template ID or launch template name in the request.
  - --launch-template-id `string` – The ID of the launch template.

  - --launch-template-name `string` – A name for the launch template.
  - --default-version `string` – The version number of the launch template to set as the default version.
  - --endpoint `snowballEndpoint` – A value that enables you to manage your compute instances programmatically using Amazon EC2-compatible API operations. For more information, see Specifying the Amazon EC2-compatible Endpoint as the AWS CLI Endpoint.
- modify-instance-attribute – Modifies an attribute of the specified instance. The following attributes are supported:
  - `instanceInitiatedShutdownBehavior`
  - `userData`
- revoke-security-group-egress – Removes one or more egress rules from a security group:
  - [--group-id `value`] – The ID of the security group
  - [--ip-permissions `value`] – One or more sets of IP permissions.
- revoke-security-group-ingress – Revokes one or more ingress rules to a security group. When calling `revoke-security-group-ingress`, you must specify a value for either `group-name` or `group-id`.
  - [--group-name `value`] – The name of the security group.
  - [--group-id `value`] – The ID of the security group.
  - [--ip-permissions `value`] – One or more sets of IP permissions.
  - [--protocol `value`] The IP protocol. Possible values are `tcp`, udp, and `icmp`. The `--port` argument is required unless the "all protocols" value is specified (-1).
  - [--port `value`] – For TCP or UDP, the range of ports to allow. A single integer or a range (minimum–maximum).

    For ICMP, a single integer or a range (`type-code`) in which `type` represents the ICMP type number and `code` represents the ICMP code number. A value of -1 indicates all ICMP codes for all ICMP types. A value of -1 just for `type` indicates all ICMP codes for the specified ICMP type.
  - [--cidr `value`] – The CIDR IP range.
- run-instances – Launches a number of compute instances by using a Snowball AMI ID for an AMI.

> **Note**
>
> It can take up to an hour and a half to launch a compute instance on a Snowball Edge, depending on the size and type of the instance.

- [--block-device-mappings `(list)`] – The block device mapping entries. The parameters `DeleteOnTermination`, `VolumeSize`, and `VolumeType` are supported. Boot volumes must be type `sbg1`.

  The JSON syntax for this command is as follows.

  ```
  {
     "DeviceName": "/dev/sdh",
     "Ebs":
     {
        "DeleteOnTermination": true|false,
        "VolumeSize": 100,
        "VolumeType": "sbp1"|"sbg1"
     }
  }
  ```

- --count – Number of instances to launch. If a single number is provided, it is assumed to be the minimum to launch (defaults to 1). If a range is provided in the form `min:max`, then the first number is interpreted as the minimum number of instances to launch and the second is interpreted as the maximum number of instances to launch.
- --image-id – The Snowball AMI ID of the AMI, which you can get by calling `describe-images`. An AMI is required to launch an instance.
- --InstanceInitiatedShutdownBehavior – By default, when you initiate a shutdown from your instance (using a command such as shutdown or poweroff), the instance stops. You can change this behavior so that it terminates instead. The parameters `stop` and `terminate` are supported. The default is `stop`. For more information, see Changing the instance initiated shutdown behavior in the *Amazon EC2 User Guide for Linux Instances*.
- --instance-type – The `sbe` instance type.
- --launch-template `structure` – The launch template to use to launch the instances. Any parameters that you specify in the `run-instances` command override the same parameters

in the launch template. You can specify either the name or ID of a launch template, but not both.

```
{
   "LaunchTemplateId": "string",
   "LaunchTemplateName": "string",
   "Version": "string"
}
```

- --security-group-ids – One or more security group IDs. You can create a security group using CreateSecurityGroup. If no value is provided, the ID for the default security group is assigned to created instances.
- --tag-specifications – The tags to apply to the resources during launch. You can only tag instances on launch. The specified tags are applied to all instances that are created during launch. To tag a resource after it has been created, use `create-tags`.
- --user-data – The user data to make available to the instance. If you are using the AWS CLI, base64-encoding is performed for you, and you can load the text from a file. Otherwise, you must provide base64-encoded text.
- --key-name (string) – The name of the key pair. You can create a key pair using `CreateKeyPair` or `ImportKeyPair`.

> **Warning**
>
> If you don't specify a key pair, you can't connect to the instance unless you choose an AMI that is configured to allow users another way to log in.

- start-instances – Starts an `sbe` instance that you've previously stopped. All resources attached to the instance persist through starts and stops, but are erased if the instance is terminated.
  - --instance-ids – The IDs of one or more `sbe` instances that were stopped on the device.
- stop-instances – Stops an `sbe` instance that is running. All resources attached to the instance persist through starts and stops, but are erased if the instance is terminated.
  - --instance-ids – The IDs of one or more `sbe` instances to be stopped on the device.
- terminate-instances – Shuts down one or more instances. This operation is idempotent; if you terminate an instance more than once, each call succeeds. All resources attached to the instance persist through starts and stops, but data is erased if the instance is terminated.

> **Note**
>
> By default, when you use a command like `shutdown` or `poweroff` to initiate a shutdown from your instance, the instance stops. However, you can use the `InstanceInitiatedShutdownBehavior` attribute to change this behavior so that these commands terminate your instance. For more information, see Changing the instance initiated shutdown behavior in the *Amazon EC2 User Guide for Linux Instances*.

  - --instance-ids – The IDs of one or more `sbe` instances to be terminated on the device. All associated data stored for those instances will be lost.
- create-key-pair – Creates a 2048-bit RSA key pair with the specified name. Amazon EC2 stores the public key and displays the private key for you to save to a file. The private key is returned as an unencrypted PEM-encoded PKCS#1 private key. If a key with the specified name already exists, Amazon EC2 returns an error.
  - --key-name (string) – A unique name for the key pair.

    Constraints: Up to 255 ASCII characters.
  - [--tag-specifications] (list) – The tags to apply to the new key pair.

```
{
    "ResourceType": "image"|"instance"|"key-pair"|"launch-template"|"security-group",
    "Tags": [
      {
        "Key": "string",
        "Value": "string"
      }
      ...
    ]
  }
  ...
```

- import-key-pair –
  - --key-name (string) – A unique name for the key pair.

    Constraints: Up to 255 ASCII characters.
  - --public-key-material (blob) – The public key. For API calls, the text must be base64-encoded. For command line tools, base64-encoding is performed for you.

- [--tag-specifications] (list) – The tags to apply to the new key pair.

```
{
    "ResourceType": "image"|"instance"|"key-pair"|"launch-template"|"security-group",
    "Tags": [
      {
        "Key": "string",
        "Value": "string"
      }
      ...
    ]
  }
```

- describe-key-pairs –

  [--filters] (list) – The filters.

  - key-pair-id – The ID of the key pair.
  - key-name – The name of the key pair.
  - tag-key – The key of a tag assigned to the resource. Use this filter to find all resources assigned a tag with a specific key, regardless of the tag value.
  - [--tag-specifications] (list) – The tags to apply to the new key pair.
  - tag :key – The key/value combination of a tag assigned to the resource. Use the tag key in the filter name and the tag value as the filter value. For example, to find all resources that have a tag with the key `Owner` and the value `Team A`, specify `tag:Owner` for the filter name and `Team A` for the filter value.

```
{
    "Name": "string",
    "Values": ["string", ...]
  }
  ...
```

- [--key-names] (list) – The key pair names.

  Default: Describes all your key pairs.

- [--key-pair-ids] (list) – The IDs of the key pairs.
- delete-key-pair –
  - [--key-name] (string) – The name of the key pair.

- [--key-pair-id] (string) – The ID of the key pair.

Supported Amazon EC2-compatible API Operations

Following, you can find Amazon EC2-compatible API operations that you can use with a Snowball Edge, with links to their descriptions in the *Amazon EC2 API Reference.* Amazon EC2-compatible API calls require Signature Version 4 (SigV4) signing. If you're using the AWS CLI or an AWS SDK to make these API calls, the SigV4 signing is handled for you. Otherwise, you need to implement your own SigV4 signing solution. For more information, see Getting and using local Amazon S3 credentials.

- AssociateAddress – Associates an Elastic IP address with an instance or a network interface.
- AttachVolume – The following request parameters are supported:
  - `Device`
  - `InstanceId`
  - `VolumeId`
- AuthorizeSecurityGroupEgress – Adds one or more egress rules to a security group for use with a Snowball Edge device. Specifically, this action permits instances to send traffic to one or more destination IPv4 CIDR address ranges.
- AuthorizeSecurityGroupIngress – Adds one or more ingress rules to a security group. When calling AuthorizeSecurityGroupIngress, you must specify a value either for `GroupName` or `GroupId`.
- CreateVolume – The following request parameters are supported:
  - `SnapshotId`
  - `Size`
  - `VolumeType`
  - `TagSpecification.N`
- CreateLaunchTemplate – The following request parameters are supported:
  - `ImageId`
  - `InstanceType`
  - `SecurityGroupIds`
  - `TagSpecifications`
  - `UserData`

- CreateLaunchTemplateVersion
- CreateTags – The following request parameters are supported:
  - `AMI`
  - `Instance`
  - `Launch template`
  - `Security group`
- CreateSecurityGroup – Creates a security group on your Snowball Edge. You can create up to 50 security groups. When you create a security group, you specify a friendly name of your choice.
- DeleteLaunchTemplate
- DeleteLaunchTemplateVersions
- DeleteSecurityGroup – Deletes a security group. If you attempt to delete a security group that is associated with an instance, or is referenced by another security group, the operation fails with `DependencyViolation`.
- DeleteTags – Deletes the specified set of tags from the specified set of resources.
- DeleteVolume – The following request parameters are supported:
  - `VolumeId`
- DescribeAddresses
- DescribeImages
- DescribeInstanceAttribute – The following attributes are supported:
  - `instanceType`
  - `userData`
- DescribeInstanceStatus
- DescribeLaunchTemplates
- DescribeLaunchTemplateVersions
- DescribeInstances
- DescribeSecurityGroups – Describes one or more of your security groups. `DescribeSecurityGroups` is a paginated operation. You can issue multiple API calls to retrieve the entire dataset of results.
- DescribeTags – With this command, the following filters are supported:
  - `resource-id`
  - `resource-type` – AMI or compute instance only

  - `key`
  - `value`
- DescribeVolume – The following request parameters are supported:
  - `MaxResults`
  - `NextToken`
  - `VolumeId.N`
- DetachVolume – The following request parameters are supported:
  - `Device`
  - `InstanceId`
  - `VolumeId`
- DisassociateAddress
- GetLaunchTemplateData
- ModifyLaunchTemplate
- ModifyInstanceAttribute – Only the `userData` attribute is supported.
- RevokeSecurityGroupEgress – Removes one or more egress rules from a security group.
- RevokeSecurityGroupIngress – Revokes one or more ingress rules to a security group. When calling RevokeSecurityGroupIngress, you must specify a value either for `group-name` or `group-id`.
- RunInstances –

  > **Note**
  >
  > It can take up to an hour and a half to launch a compute instance on a Snowball Edge, depending on the size and type of the instance.

- StartInstances
- StopInstances – Resources associated with a stopped instance persist. You can terminate the instance to free up these resources. However, any associated data is deleted.
- TerminateInstances

Autostarting Amazon EC2-compatible Instances with Launch Templates

You can automatically start your Amazon EC2-compatible instances on your AWS Snowball Edge device using launch templates and Snowball Edge client launch configuration commands.

A *launch template* contains the configuration information necessary to create an Amazon EC2-compatible instance on your Snowball Edge. You can use a launch template to store launch parameters so you don't have to specify them every time that you start an EC2-compatible instance on your Snowball Edge.

When you use autostart configurations on your Snowball Edge, you configure the parameters that you want your Amazon EC2-compatible instance to start with. After your Snowball Edge is configured, when you reboot and unlock it, it uses your autostart configuration to launch an instance with the parameters that you specified. If an instance that you launched using an autostart configuration is stopped, the instance starts running when you unlock your device.

> **Note**
>
> After you first configure an autostart configuration, restart your device to launch it. All subsequent instance launches (after planned or unplanned reboots) happen automatically after your device is unlocked.

A launch template can specify the Amazon Machine Image (AMI) ID, instance type, user data, security groups, and tags for an Amazon EC2-compatible instance when you launch that instance. For a list of supported instance types, see Quotas for Compute Instances on a Snowball Edge Device.

To automatically launch EC2-compatible instances on your Snowball Edge, take the following steps:

1. When you order your AWS Snowball Edge device, create a job to order a Snow Family device with compute instances. For more information, see Creating a Compute Job.
2. After receiving your Snowball Edge, unlock it.
3. Use the EC2-compatible API command `aws ec2 create-launch-template` to create a launch template.
4. Use the Snowball Edge client command `snowballEdge create-autostart-configuration` to bind your EC2-compatible instance launch template to your network

configuration. For more information, see Creating a Launch Configuration to Autostart Amazon EC2-compatible Instances.

5. Reboot, then unlock your device. Your EC2-compatible instances are automatically started using the attributes specified in your launch template and your Snowball Edge client command `create-autostart-configuration`.

To view the status of your running instances, use the EC2-compatible API command `describe-autostart-configurations`.

> **Note**
>
> There is no console or job management API for AWS Snowball support for launch templates. You use EC2-compatible and Snowball Edge client CLI commands to automatically start EC2-compatible instances on your AWS Snowball Edge device.

Using Instance Metadata Service for Snow with Amazon EC2-compatible instances

IMDS for Snow provides Instance Metadata Service (IMDS) for Amazon EC2-compatible instances on Snow. Instance metadata is categories of information about instances. It includes categories such as *host name*, *events*, and *security groups*. Using IMDS for Snow, you can use instance metadata to access user data that you specified when launching your Amazon EC2-compatible instance. For example, you can use IMDS for Snow to specify parameters for configuring your instance, or include these parameters in a simple script. You can build generic AMIs and use user data to modify the configuration files supplied at launch time.

To learn about instance metadata and user data and Snow EC2-compatible instances, see Supported Instance Metadata and User Data in this guide.

> **Important**
>
> Although you can only access instance metadata and user data from within the instance itself, the data is not protected by authentication or cryptographic methods. Anyone who has direct access to the instance, and potentially any software running on the instance, can view its metadata. Therefore, you should not store sensitive data, such as passwords or long-lived encryption keys, as user data.

> **Note**
>
> The examples in this section use the IPv4 address of the instance metadata service: 169.254.169.254. We do not support the retrieval of instance metadata using the link-local IPv6 address.

Topics

IMDS versions

You can access instance metadata from a running instance using IMDS version 2 or IMDS version 1:

- Instance Metadata Service version 2 (IMDSv2), a session-oriented method
- Instance Metadata Service version 1 (IMDSv1), a request-response method

Depending on the version of your Snow software, you can use IMDSv1, IMDSv2, or both. This also depends on the type of AMI running in the EC2-compatible instance. Some AMIs, such as those running Ubuntu 20.04, require IMDSv2. The instance metadata service distinguishes between IMDSv1 and IMDSv2 requests based on the presence of `PUT` or `GET` headers. IMDSv2 uses both of these headers. IMDSv1 uses only the `GET` header.

AWS encourages the use of IMDSv2 rather than IMDSv1 because IMDSv2 includes higher security. For more information, see Add defense in depth against open firewalls, reverse proxies, and SSRF vulnerabilities with enhancements to the EC2 Instance Metadata Service.

IMDSv2

IMDSv2 uses session-oriented requests. With session-oriented requests, you create a session token that defines the session duration. Session duration can be a minimum of one second and a maximum of six hours. During this duration, you can use the same session token for subsequent requests. After this duration expires, you must create a new session token for future requests.

The following example uses a Linux shell script and IMDSv2 to retrieve the top-level instance metadata items. This example:

1. Creates a session token lasting six hours (21,600 seconds) using the PUT request.
2. Stores the session token header in a variable named `TOKEN`.
3. Requests the top-level metadata items using the token.

You can run two commands separately, or combine them.

Separate commands

First, generate a token using the following command.

> **Note**
>
> `X-aws-ec2-metadata-token-ttl-seconds` is a required header. If this header is not included, you will receive an **400 - Missing or Invalid Parameters** error code.

```
[ec2-user ~]$ TOKEN=curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"
```

Then, use the token to generate top-level metadata items using the following command.

```
[ec2-user ~]$ curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/
```

Combined commands

You can store the token and combine the commands. The following example combines the above two commands and stores the session token header in a variable named `TOKEN`.

> **Note**
>
> If there is an error in creating the token, an error message is stored in the variable instead of a valid token and the command will not work.

Example of combined commands

```
    [ec2-user ~]$ TOKEN=curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-
aws-ec2-metadata-token-ttl-seconds: 21600" \
    && curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/
meta-data/
```

After you've created a token, you can reuse it until it expires. The following example command gets the ID of the AMI used to launch the instance and stores it in the `$TOKEN` created in the previous example.

Example of reusing a token

```
    [ec2-user ~]$ curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/
latest/meta-data/ami-id
```

When you use IMDSv2 to request instance metadata, the request must follow these rules:

1. Use a `PUT` request to initiate a session to the instance metadata service. The `PUT` request returns a token that must be included in subsequent `GET` requests to the instance metadata service. The token is required to access metadata using IMDSv2.
2. Include the token in all `GET` requests to the instance metadata service.
   a. The token is an instance-specific key. The token is not valid on other EC2-compatible instances and will be rejected if you attempt to use it outside of the instance on which it was generated.
   b. The `PUT` request must include a header that specifies the time to live (TTL) for the token, in seconds, up to a maximum of six hours (21,600 seconds). The token represents a logical session. The TTL specifies the length of time that the token is valid and, therefore, the duration of the session.
   c. After a token expires, to continue accessing instance metadata, you must create a new session using another `PUT` request.
   d. You can choose to reuse a token or create a new token with every request. For a small number of requests, it might be easier to generate and immediately use a token each time you need to access the instance metadata service. But for efficiency, you can specify a longer duration

for the token and reuse it rather than having to write a PUT request every time you need to request instance metadata. There is no practical limit on the number of concurrent tokens, each representing its own session.

HTTP GET and HEAD methods are allowed in IMDSv2 instance metadata requests. PUT requests are rejected if they contain an `X-Forwarded-For` header.

By default, the response to PUT requests has a response hop limit (time to live) of 1 at the IP protocol level. IMDS for Snow does not have ability to modify the hop limit on PUT responses.

IMDSv1

IMDSv1 uses the request-response model. To request instance metadata, you send a GET request to the instance metadata service.

```
[ec2-user ~]$ curl http://169.254.169.254/latest/meta-data/
```

Retrieve instance metadata

Your instance metadata is available from your running instance, so you do not need to use Amazon EC2 console or the AWS CLI to access it. This can be helpful when you're writing scripts to run from your instance. For example, you can access the local IP address of your instance from instance metadata to manage a connection to an external application. Instance metadata is divided into categories. For a description of each instance metadata category, see Supported Instance Metadata and User Data in this guide.

To view all categories of instance metadata from within a running instance, use the following IPv4 URI:

```
http://169.254.169.254/latest/meta-data/
```

The IP addresses are link-local addresses and are valid only from the instance. For more information, see Link-local address on Wikipedia.

Responses and error messages

All instance metadata is returned as text (HTTP content type `text/plain`).

A request for a specific metadata resource returns the appropriate value, or an **404 - Not Found** HTTP error code, if the resource is not available.

A request for a general metadata resource (when the URI ends with a / character) returns a list of available resources, or an **404 - Not Found** HTTP error code if there is no such resource. The list items are on separate lines, terminated by line feeds (ASCII character code 10).

For requests made using IMDSv1, the following HTTP error codes can be returned:

- **400 - Missing or Invalid Parameters** — The PUT request is not valid.
- **401 - Unauthorized** — The GET request uses an invalid token. The recommended action is to generate a new token.
- **403 - Forbidden** — The request is not allowed or the instance metadata service is turned off.

Examples of retrieving instance metadata using IMDSv1 and IMDSv2

The following examples provide commands that you can use on a Linux instance.

Example of getting the available versions of the instance metadata

This example gets the available versions of the instance metadata. Each version refers to an instance metadata build when new instance metadata categories were released. The earlier versions are available to you in case you have scripts that rely on the structure and information present in a previous version.

IMDSv2

```
   [ec2-user ~]$ TOKEN=`curl -X PUT "http://192.0.2.0/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600"` && curl -H "X-aws-ec2-metadata-token: $TOKEN" -v
 http://192.0.2.0/
   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
 Dload  Upload   Total   Spent    Left  Speed
   100        56         100      56      0       0       3733     0    --:--:--
 --:--:-- --:--:--  3733
   *   Trying 192.0.2.0...
   * TCP_NODELAY set
   * Connected to 192.0.2.0 (192.0.2.0) port 80 (#0)
   > GET / HTTP/1.1
   > Host: 192.0.2.0
   > User-Agent: curl/7.61.1
```

```
    > Accept: */*
    > X-aws-ec2-metadata-token:
MDAXcxNFLbAwJIYx8KzgNckcHTdxT4Tt69TzpKExlXKTULHIQnjEtXvD
    >
    * HTTP 1.0, assume close after body
    < HTTP/1.0 200 OK
    < Date: Mon, 12 Sep 2022 21:58:03 GMT
    < Content-Length: 274
    < Content-Type: text/plain
    < Server: EC2ws
    <
    1.0
    2007-01-19
    2007-03-01
    2007-08-29
    2007-10-10
    2007-12-15
    2008-02-01
    2008-09-01
    2009-04-04
    2011-01-01
    2011-05-01
    2012-01-12
    2014-02-25
    2014-11-05
    2015-10-20
    2016-04-19
    2016-06-30
    2016-09-02
    2018-03-28
    2018-08-17
    2018-09-24
    2019-10-01
    2020-10-27
    2021-01-03
    2021-03-23
    * Closing connection 0
```

IMDSv1

```
    [ec2-user ~]$ curl http://192.0.2.0/
```

```
    1.0
    2007-01-19
    2007-03-01
    2007-08-29
    2007-10-10
    2007-12-15
    2008-02-01
    2008-09-01
    2009-04-04
    2011-01-01
    2011-05-01
    2012-01-12
    2014-02-25
    2014-11-05
    2015-10-20
    2016-04-19
    2016-06-30
    2016-09-02
    2018-03-28
    2018-08-17
    2018-09-24
    2019-10-01
    2020-10-27
    2021-01-03
    2021-03-23
    latest
```

Example of getting the top-level metadata items

This example gets the top-level metadata items. For information on top-level metadata items, see Supported Instance Metadata and User Data in this guide.

IMDSv2

```
    [ec2-user ~]$ TOKEN=`curl -X PUT "http://192.0.2.0/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600"` && curl -H "X-aws-ec2-metadata-token: $TOKEN" -v
 http://192.0.2.0/latest/meta-data/
    ami-id
    hostname
    instance-id
    instance-type
```

```
    local-hostname
    local-ipv4
    mac
    network/
    reservation-id
    security-groups
```

IMDSv1

```
    [ec2-user ~]$ curl http://192.0.2.0/latest/meta-data/
    ami-id
    hostname
    instance-id
    instance-type
    local-hostname
    local-ipv4
    mac
    network/
    reservation-id
    security-groups
```

Example of getting values of top-level metadata

The following examples get the values of some of the top-level metadata items that were obtained in the preceding example. The IMDSv2 requests use the stored token that was created in the preceding example command, assuming it has not expired.

`ami-id` IMDSv2

```
    curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://192.0.2.0/latest/meta-data/
ami-id ami-0abcdef1234567890
```

`ami-id` IMDSv1

```
    curl http://192.0.2.0/latest/meta-data/ami-id ami-0abcdef1234567890
```

```
```

`reservation-id` IMDSv2

```
    [ec2-user ~]$ curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://192.0.2.0/
latest/meta-data/reservation-id r-0efghijk987654321
```

`reservation-id` IMDSv1

```
    [ec2-user ~]$ curl http://192.0.2.0/latest/meta-data/reservation-id \
    r-0efghijk987654321
```

`local-hostname` IMDSv2

```
    [ec2-user ~]$ curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://192.0.2.0/
latest/meta-data/local-hostname ip-00-000-00-00
```

`local-hostname` IMDSv1

```
    [ec2-user ~]$ curl http://192.0.2.0/latest/meta-data/local-hostname ip-00-000-00-00
```

Using Block Storage with Your Amazon EC2-compatible Instances

With block storage on Snowball Edge, you can add or remove block storage based on the needs of your applications. Volumes that are attached to an Amazon EC2-compatible instance are exposed as storage volumes that persist independently from the life of the instance. You can manage block storage using the familiar Amazon EBS API.

Certain Amazon EBS commands are supported by using the EC2-compatible endpoint. Supported commands include `attach-volume`, `create-volume`, `delete-volume`, `detach-volume`, and `describe-volumes`. For more information about these commands, see List of Supported Amazon EC2-compatible AWS CLI Commands on a Snowball Edge.

> **Important**
>
> Be sure to unmount any file systems on the device within your operating system before detaching the volume. Failure to do so can potentially result in data loss.

Following, you can find Amazon EBS volume quotas and differences between Amazon EBS volumes on your device and Amazon EBS volumes in the cloud:

- Amazon EBS volumes are only available to EC2-compatible instances running on the device hosting the volumes.
- Volume types are limited to either capacity-optimized HDD (`sbg1`) or performance-optimized SSD (`sbp1`). The default volume type is `sbg1`.
- Snowball Edge shares HDD memory between Amazon S3 objects and Amazon EBS. If you use HDD-based block storage on AWS Snowball Edge, it reduces the amount of memory available for Amazon S3 objects. Likewise, Amazon S3 objects reduce the amount of memory available for Amazon EBS block storage on HDD volumes.
- Amazon EC2-compatible root volumes always use the IDE driver. Additional Amazon EBS volumes will preferentially use the Virtio driver if available. If the Virtio driver isn't available, SBE defaults to the IDE driver. The Virtio driver allows for better performance and is recommended.
- When creating Amazon EBS volumes, the `encrypted` parameter isn't supported. However, all data on your device is encrypted by default. .
- Volumes can be from 1 GB to 10 TB in size.
- Up to 10 Amazon EBS volumes can be attached to a single EC2-compatible instance.
- There is no formal limit to the number of Amazon EBS volumes you can have on your AWS Snowball Edge device. However, total Amazon EBS volume capacity is limited by the available space on your device.

Security Groups in Snowball Edge Devices

A *security group* acts as a virtual firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. You can add rules to each security group to allow traffic to or from its associated instances. For more information, see Amazon EC2 security groups for Linux instances in the *Amazon EC2 User Guide for Linux Instances*.

Security groups in Snowball Edge devices are similar to security groups in the AWS Cloud. Virtual private clouds (VPCs) aren't supported on Snowball Edge devices.

Following, you can find the other differences between Snowball Edge security groups and EC2-VPC security groups:

- Each Snowball Edge has a limit of 50 security groups.
- The default security group allows all inbound and outbound traffic.
- Traffic between local instances can use either the private instance IP address or a public IP address. For example, suppose that you want to connect using SSH from instance A to instance B. In this case, your target IP address can be either the public IP or private IP address of instance B, if the security group rule allows the traffic.
- Only the parameters listed for AWS CLI actions and API calls are supported. These typically are a subset of those supported in EC2-VPC instances.

For more information about supported AWS CLI actions, see List of Supported Amazon EC2-compatible AWS CLI Commands on a Snowball Edge. For more information on supported API operations, see Supported Amazon EC2-compatible API Operations.

Supported Instance Metadata and User Data

Instance metadata is data about your instance that you can use to configure or manage the running instance. Snowball Edge supports a subset of instance metadata categories for your compute instances. For more information, see Instance metadata and user data in the *Amazon EC2 User Guide for Linux Instances.*

The following categories are supported. Using any other categories returns a `404` error message.

Supported instance metadata categories on a Snowball Edge

Data	Description
`ami-id`	The AMI ID used to launch the instance.
`hostname`	The private IPv4 DNS hostname of the instance.
`instance-id`	The ID of this instance.

Data	Description
`instance-type`	The type of instance.
`local-hostname`	The private IPv4 DNS hostname of the instance.
`local-ipv4`	The private IPv4 address of the instance.
`mac`	The instance's media access control (MAC) address.
`network/interfaces/macs/` *`mac`*`/local-hostname`	The interface's local hostname.
`network/interfaces/macs/` *`mac`*`/local-ipv4s`	The private IPv4 addresses associated with the interface.
`network/interfaces/macs/` *`mac`*`/mac`	The instance's MAC address.
`network/interfaces/macs/` *`mac`*`/public-ipv4s`	The Elastic IP addresses associated with the interface.
`public-ipv4`	The public IPv4 address.
`public-keys/0/openssh-key`	Public key. Only available if supplied at instance launch time.
`reservation-id`	The ID of the reservation.
userData	Shell scripts to send instructions to an instance at launch.

Supported instance dynamic data categories on a Snowball Edge

Data	Description
instance-identity/document	JSON containing instance attributes. Only `instanceId` , `imageId`, `privateIp` , and `instanceType` have values, and the

Data	Description
	other returned attributes are null. For more information, see Instance Identity Documents in the *Amazon EC2 User Guide for Linux Instances*.

User Data in Snowball Compute Instances

User data is supported for use with shell scripts for compute instances on a Snowball Edge. Using shell scripts, you can send instructions to an instance at launch. You can change user data with the `modify-instance-attribute` AWS CLI command, or the `ModifyInstanceAttribute` API action.

To change user data

1. Stop your compute instance with the `stop-instances` AWS CLI command.
2. Using the `modify-instance-attribute` AWS CLI command, modify the `userData` attribute.
3. Restart your compute instance with the `start-instances` AWS CLI command.

Only shell scripts are supported with compute instances. There is no support for `cloud-init` package directives on compute instances running on a Snowball Edge. For more information about working with AWS CLI commands, see the *AWS CLI Command Reference*.

Stopping EC2-compatible Instances

To avoid accidentally deleting the Amazon EC2-compatible instances that you create on your device, don't shut down your instances from the operating system. For example, don't use the `shutdown` or `reboot` commands. Shutting down an instance from within the operating system has the same effect as calling the terminate-instances command.

Instead, use the stop-instances command to suspend Amazon EC2-compatible instances that you want to preserve.

Troubleshooting Compute Instances on Snowball Edge Devices

Following, you can find troubleshooting tips for Snowball Edge jobs with compute instances.

Topics

Virtual Network Interface Has an IP Address of 0.0.0.0

This issue can occur if the physical network interface (NIC) you associated with your virtual network interface (VNIC) also has an IP address of 0.0.0.0. This effect can happen if the NIC wasn't configured with an IP address (for instance, if you've just powered on the device). It can also happen if you're using the wrong interface. For example, you might be trying to get the IP address of the SFP+ interface, but it's the RJ45 interface that's connected to your network.

Action to Take

If this occurs, you can do the following:

- Create a new VNIC, associated with a NIC that has an IP address. For more information, see Network Configuration for Compute Instances.
- Update an existing VNIC. For more information, see Updating a Virtual Network Interface.

Snowball Edge Hangs When Launching a Large Compute Instance

It can appear that your Snowball Edge has stopped launching an instance. This is generally not the case. However, it can take an hour or more for the largest compute instances to launch.

To check the status of your instances, use the AWS CLI command `aws ec2 describe-instances` run against the HTTP or HTTPS Amazon EC2-compatible endpoint on the Snowball Edge.

My Instance Has One Root Volume

Instances have one root volume by design. All `sbe` instances have a single root volume, but with Snowball Edge, you can add or remove block storage based on the needs of your applications. For more information, see Using Block Storage with Your Amazon EC2-compatible Instances.

Unprotected Private Key File Error

This error can occur if your .pem file on your compute instance has insufficient read/write permissions.

Action to Take

You can resolve this by changing the permissions for the file with the following procedure:

1. Open a terminal and navigate to the location that you saved your .pem file to.
2. Enter the following command.

```
chmod 400 filename.pem
```

Using Amazon S3 compatible storage on Snow Family devices

Amazon S3 compatible storage on Snow Family devices delivers secure object storage with increased resiliency, scale, and an expanded Amazon S3 API feature-set to rugged, mobile edge, and disconnected environments. Using Amazon S3 compatible storage on Snow Family devices, you can store data and run highly available applications on Snow Family devices for edge computing.

You can create Amazon S3 buckets on the Snowball Edge devices to store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. Amazon S3 compatible storage on Snow Family devices provides a new storage class, `SNOW`, which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple Snowball Edge devices. You can use the same APIs and features on Snowball Edge buckets that you do on Amazon S3, including bucket lifecycle policies, encryption, and tagging. When the device or devices are returned to AWS, all data created or stored in Amazon S3 compatible storage on Snow Family devices is erased. For more information, see Local Compute and Storage Only Jobs.

You can deploy Amazon S3 compatible storage on Snow Family devices in standalone configuration or in cluster configuration. In standalone configuration, you can provision S3 capacity on the device and the balance is available as block storage. In cluster configuration, all data disk capacity is used for S3 storage. A cluster may consist of a minimum of 3 devices up to a maximum of 16 devices. Depending on the size of cluster, S3 service is designed to sustain device fault tolerance of 1 or 2 devices.

With AWS DataSync, you can transfer objects between Amazon S3 compatible storage on Snow Family devices on a Snowball Edge device and AWS storage services. For more information, see Configuring transfers with S3 compatible storage on Snowball Edge in the AWS DataSync User Guide.

Following is the Amazon S3 compatible storage on Snow Family devices storage capacity and block storage capacity for a standalone device using Amazon S3 compatible storage on Snow Family devices. For fault tollerence and storage capacity of clusters, see this table.

Snowball Edge Compute Optimized and Compute Optimized with GPU

Storage capacity of Amazon S3 compatible storage on Snow Family devices and block storage of Snowball Edge Compute Optimized (with AMD EPYC Gen1, HDD, and optional GPU) devices

Amazon S3 compatible storage on Snow Family devices storage capacity (in TB)	Block storage capacity (in TB)
2.5	41
5.5	37
8.5	33
11	29
14	25
17	21
19.5	17
22.5	13
25.5	9
28.5	5
31	1

Snowball Edge Compute Optimized with NVMe storage

Storage capacity of Amazon S3 compatible storage on Snow Family devices and block storage of Snowball Edge Compute Optimized (Compute Optimized with AMD EPYC Gen2 and NVMe) devices

Amazon S3 compatible storage on Snow Family devices storage capacity (in TB)	Block storage capacity (in TB)
3	17.5
5.5	14.5
10.5	8.5
12	6.5
13	5.5
16.5	1.5

Snowball Edge storage optimized 210 TB

Storage capacity of Amazon S3 compatible storage on Snow Family devices and block storage of Snowball Edge storage optimized 210 TB devices

Amazon S3 compatible storage on Snow Family devices storage capacity (in TB)	Block storage capacity (in TB)
20	206
40	182
60	158
80	134
100	110
120	86

Amazon S3 compatible storage on Snow Family devices storage capacity (in TB)	Block storage capacity (in TB)
140	62
160	38
180	14
190	2

Amazon S3 compatible storage on Snow Family devices specifications:

- The maximum number of Snow Family device buckets is 100 per device or per cluster.
- The S3 on Snow Family device bucket owner account owns all objects in the bucket.
- Only the S3 on Snow Family device bucket owner account can perform operations on the bucket.
- Object size limitations are consistent with those in Amazon S3.
- All objects stored on S3 on Snow Family devices have SNOW as the storage class.
- By default, all objects stored in the SNOW storage class are stored using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). You can also explicitly choose to store objects by using server-side encryption with customer-provided encryption keys (SSE-C).
- If there is not enough space to store an object on your Snow Family device, the API returns an insufficient capacity exception (ICE).

Topics

Order Amazon S3 compatible storage on Snow Family devices

Ordering a device for Amazon S3 compatible storage on Snow Family devices is very similar to the process for ordering a Snowball Edge. To order, see Creating a job to order a Snow Family device in this guide and keep these items in mind during the ordering process:

- For **Choose a job type**, choose **Local compute and storage only**.
- Under **Snow devices**, choose **Snowball Edge Compute Optimized**
- Under **Select the storage type**, select **Amazon S3 compatible storage on Snow Family devices**.
- For a standalone device, under **Storage capacity**, choose **Single device** and then select your desired storage amount.
- For a cluster, under **Storage capacity** select **Cluster** and then select your desired storage capacity and fault tolerance.

Setting up Amazon S3 compatible storage on Snow Family devices

Install and configure software tools from AWS to your local environment to interact with the Snowball Edge device or cluster of devices and Amazon S3 compatible storage on Snow Family devices. Then, use these tools to set up the Snowball Edge device or cluster and start Amazon S3 compatible storage on Snow Family devices.

Prerequisites

Amazon S3 compatible storage on Snow Family devices requires you to have the Snowball Edge Client and the AWS CLI installed to your local environment. You can also use AWS SDK for .NET and AWS Tools for Windows PowerShell to work with Amazon S3 compatible storage on Snow Family devices. AWS recommends using the following versions of these tools:

- **Snowball Edge Client** – Use the latest version. For more information, see Downloading and Installing the Snowball Edge Client in this guide.
- **AWS CLI** – Version 2.11.15 or newer. For more information, see Installing, updating, and uninstalling the AWS CLI in the AWS Command Line Interface User Guide.
- **AWS SDK for .NET** – AWSSDK.S3Control 3.7.304.8 or newer. For more information, see AWS SDK for .NET.
- **AWS Tools for Windows PowerShell** – Version 4.1.476 or newer. For more information, see AWS Tools for Windows PowerShell User Guide.

Setting up your local environment

This section describes how to set up and configure the Snowball Edge Client and your local environment for use with Amazon S3 compatible storage on Snow Family devices.

To set up your environment

1. Download and install the latest version of the Snowball Edge Client. For more information, see Downloading and Installing the Snowball Edge Client in this guide.
2. Run the following commands to configure your folders.

```
chmod u+x new_cli/bin/snowballEdge
chmod u+x new_cli/jre/bin/java
```

3. Add `new_cli/bin` to your $PATH.
4. Run the command `snowballEdge configure`. You receive a response similar to the following:

```
Configuration will be stored at /home/user/.aws/snowball/config/snowball-
edge.config
```

5. Enter the following information:

   - The manifest path.
   - An unlock code.
   - The default endpoint. For standalone Snowball Edge devices, use the device's IP address. For a cluster of device,s specify the IP address for any device in the cluster. To test if the default endpoints are available from the client, use a command similar to the following. For the port number, use 9091 (activation port), 22 (SSH), and 8080 (HTTP endpoint for s3).

```
telnet snowball_ip port_number
```

6. If you are using AWS SDK for .NET, set the `clientConfig.AuthenticationRegion` parameter value as follows:

```
    clientConfig.AuthenticationRegion = "snow"
```

Setting up your Snowball Edge device

Set up your Snowball Edge device according to Receiving the Snowball Edge in this guide.

After your device is set up and running, configure and start Amazon S3 compatible storage on Snow Family devices. See Setting up Amazon S3 compatible storage on Snow Family devices.

Setting up IAM on the Snowball Edge

AWS Identity and Access Management (IAM) helps you to enable granular access to AWS resources that run on your Snowball Edge devices. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

IAM is supported locally on the Snowball Edge. You can use the local IAM service to create roles and attach IAM policies to them. You can use these policies to allow the access necessary to perform assigned tasks.

The following example allows full access to the Amazon S3 API:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"

        }
    ]
}
```

For more IAM policy examples, see the AWS Snowball Edge Developer Guide.

Starting the Amazon S3 compatible storage on Snow Family devices service

Use the following instructions to start the Amazon S3 compatible storage on Snow Family devices service on a Snowball Edge device or cluster.

> **Note**
>
> If you prefer a more user-friendly experience, you can start the Amazon S3 compatible storage on Snow Family devices service for a standalone device or cluster of devices using AWS OpsHub. See Set up Amazon S3 compatible storage on Snow Family devices.

1. Unlock your Snowball Edge device or cluster of devices by running the following command:

   - For a single device:

     ```
     snowballEdge unlock-device --endpoint https://snow-device-ip
     ```

   - For a cluster:

     ```
     snowballEdge unlock-cluster
     ```

2. Run the following command and make sure that the Snowball Edge device or cluster of devices are unlocked:

   - For a single device:

     ```
     snowballEdge describe-device --endpoint https://snow-device-ip
     ```

   - For a cluster:

     ```
     snowballEdge describe-cluster --device-ip-addresses [snow-device-1-ip] [snow-device-2-ip] /
         [snow-device-3-ip] [snow-device-4-ip] [snow-device-5-ip] /
         [snow-device-6-ip]
     ```

3. For each device (whether you have one or a cluster), to start Amazon S3 compatible storage on Snow Family devices, do the following:

   a. Fetch the device's `PhysicalNetworkInterfaceId` by running the following `describe-device` command:

```
snowballEdge describe-device --endpoint https://snow-device-ip
```

b. Run the following `create-virtual-network-interface` command twice to create the virtual network interfaces (VNIs) for the `s3control` (for bucket operations) and `s3api` (for object operations) endpoints.

```
snowballEdge create-virtual-network-interface --ip-address-assignment
dhcp --manifest-file manifest --physical-network-interface-id
"PhysicalNetworkInterfaceId" --unlock-code unlockcode --endpoint https://snow-
device-ip
```

For details about these commands, see Creating a Virtual Network Interface.

> **Note**
>
> Starting Amazon S3 compatible storage on Snow Family devices consumes device resources.

4. Start the Amazon S3 compatible storage on Snow Family devices service by running the following `start-service` command. which includes the IP addresses of your devices and the Amazon Resource Names (ARNs) of the VNIs that you created for the `s3control` and `s3api` endpoints:

To start the service on a single device:

```
snowballEdge start-service --service-id s3-snow --device-ip-addresses snow-
device-1-ip --virtual-network-interface-arns vni-arn-1 vni-arn-2
```

To start the service on a cluster:

```
snowballEdge start-service --service-id s3-snow --device-ip-addresses snow-
device-1-ip snow-device-2-ip snow-device-3-ip --virtual-network-interface-arns vni-
arn-1 vni-arn-2 vni-arn-3  vni-arn-4 vni-arn-5 vni-arn-6
```

For `--virtual-network-interface-arns`, include ARNs for all the VNIs that you created in the previous step. Separate each ARN using a space.

5. Run the following `describe-service` command for a single device:

```
snowballEdge describe-service --service-id s3-snow
```

Wait until service status is `Active`.

Run the following `describe-service` command for a cluster:

```
snowballEdge describe-service --service-id s3-snow \
  --device-ip-addresses snow-device-1-ip snow-device-2-ip snow-device-3-ip
```

Working with S3 buckets on a Snowball Edge device

You can create Amazon S3 buckets on your Snowball Edge devices to store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. Amazon S3 compatible storage on Snow Family devices provides a new storage class, `SNOW`, which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple Snowball Edge devices. You can use the same APIs and features on Snowball Edge buckets that you do on Amazon S3, including bucket lifecycle policies, encryption, and tagging.

Using the AWS CLI

Follow these instructions to work with Amazon S3 buckets on your device using the AWS CLI.

To set up the AWS CLI

1. Create a profile for object endpoints in `~/.aws/config`.

```
[profile your-profile]
aws_access_key_id = your-access-id
aws_secret_access_key = your-access-key
region = snow
ca_bundle = dev/apps/ca-certs/your-ca_bundle
```

2. Obtain a certificate from your device. For information, see the *Snowball Edge Developer Guide*.
3. If you installed the SDK in a virtual environment, activate it using the following command:

```
source your-virtual-environment-name/bin/activate
```

After you set up your operations, you can access them using API calls with the AWS CLI. In the following examples, *cert* is the device certificate you just obtained using IAM.

Accessing object operations

```
aws s3api --profile your-profile list-objects-v2 --endpoint-url
https://s3api-endpoint-ip
```

Accessing bucket operations

```
aws s3control --profile your-profile list-regional-buckets --account-id
bucket-owner --endpoint-url https://s3ctrlapi-endpoint-ip
```

Using the Java SDK

Use the following example to work with Amazon S3 objects using the Java SDK.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;

import java.net.URI;

AwsBasicCredentials creds = AwsBasicCredentials.create(accessKey, secretKey); // set
 creds by getting Access Key and Secret Key from snowball edge
SdkHttpClient httpClient =
 ApacheHttpClient.builder().tlsTrustManagersProvider(trustManagersProvider).build(); //
 set trust managers provider with client certificate from snowball edge
String s3SnowEndpoint = "10.0.0.0"; // set s3-snow object api endpoint from describe
 service
```

```
S3Client s3Client =
 S3Client.builder().httpClient(httpClient).region(Region.of("snow")).endpointOverride(new
 URI(s3SnowEndpoint)).credentialsProvider(StaticCredentialsProvider.create(creds)).build();
```

Bucket ARN format

You can use the Amazon Resource Name (ARN) format listed here to identify an Amazon S3 bucket on a Snowball Edge device:

```
arn:partition:s3:snow:account-id:device/device-id/bucket/bucket-name
```

Where *partition* is the partition of the Region where you ordered your Snowball Edge device. *device-id* is the job_id if the device is a standalone Snowball Edge device, or the *cluster_id* if you have a Snowball Edge cluster.

Creating an S3 bucket on a Snowball Edge device

You can create Amazon S3 buckets on your Snowball Edge device to store and retrieve objects at the edge for applications that require local data access, local data processing, and data residency. Amazon S3 compatible storage on Snow Family devices provides a new storage class, SNOW, which uses Amazon S3 and is designed to store data durably and redundantly across multiple devices . You can use the same APIs and features as you do on Amazon S3 buckets, including bucket lifecycle policies, encryption, and tagging.

The following example creates an Amazon S3 bucket for a Snowball Edge device using the AWS CLI. To run this command, replace the user input placeholders with your own information.

```
aws s3control --profile your-profile create-bucket --bucket your-snow-bucket --
endpoint-url https://s3ctrlapi-endpoint-ip
```

Creating and managing an object lifecycle configuration using the AWS CLI

You can use Amazon S3 Lifecycle to optimize storage capacity for Amazon S3 compatible storage on Snow Family devices. You can create lifecycle rules to expire objects as they age or are replaced by newer versions. You can create, enable, disable, or delete a lifecycle rule. For more information about Amazon S3 Lifecycle, see Managing your storage lifecycle.

> **Note**
>
> The AWS account that creates the bucket owns it and is the only one that can create, enable, disable, or delete a lifecycle rule.

To create and manage a lifecycle configuration for an Amazon S3 compatible storage on Snow Family devices bucket using the AWS Command Line Interface (AWS CLI), see the following examples.

PUT a lifecycle configuration on a Snowball Edge bucket

The following AWS CLI example puts a lifecycle configuration policy on a Snowball Edge bucket. This policy specifies that all objects that have the flagged prefix (*myprefix*) and tags expire after 10 days. To use this example, replace each user input placeholder with your own information.

First, save the lifecycle configuration policy to a JSON file. For this example, the file is named **lifecycle-example.json**.

```
{
    "Rules": [{
        "ID": "id-1",
        "Filter": {
            "And": {
                "Prefix": "myprefix",
                "Tags": [{
                        "Value": "mytagvalue1",
                        "Key": "mytagkey1"
                    },
                    {
                        "Value": "mytagvalue2",
                        "Key": "mytagkey2"
                    }
                ],
            }
        },
        "Status": "Enabled",
        "Expiration": {
            "Days": 10
        }
    }]
```

```
}
```

After you save the file, submit the JSON file as part of the `put-bucket-lifecycle-configuration` command. To use this command, replace each user input placeholder with your own information.

```
aws s3control put-bucket-lifecycle-configuration --bucket
                example-snow-bucket --profile your-profile
                --lifecycle-configuration file://lifecycle-example.json --endpoint-url
 https://s3ctrlapi-endpoint-ip
```

For more information about this command, see put-bucket-lifecycle-configuration in the *AWS CLI Command Reference*.

Working with S3 buckets on a Snowball Edge device

With Amazon S3 compatible storage on Snow Family devices, you can create Amazon S3 buckets on your Snowball Edge devices to store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. Amazon S3 compatible storage on Snow Family devices provides a new storage class, SNOW, which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple Snowball Edge devices. You can use the same APIs and features on Snowball Edge buckets that you do on Amazon S3, including bucket lifecycle policies, encryption, and tagging. You can use Amazon S3 compatible storage on Snow Family devices using the AWS Command Line Interface (AWS CLI) or AWS SDKs.

Determine whether you can access an Amazon S3 compatible storage on Snow Family devices bucket

The following example uses the `head-bucket` command to determine if an Amazon S3 bucket exists and you have permissions to access it using the AWS CLI. To use this command, replace each user input placeholder with your own information.

```
aws s3api head-bucket --bucket sample-bucket --profile your-profile --endpoint-url
 https://s3api-endpoint-ip
```

Retrieve a list of buckets or regional buckets

Use the `list-regional-buckets` or `list buckets` to list Amazon S3 compatible storage on Snow Family devices buckets using the AWS CLI.

```
aws s3control list-regional-buckets --account-id 123456789012 --profile your-profile --
endpoint-url https://s3ctrlapi-endpoint-ip
```

For more information about the `list-regional-buckets` command, see list-regional-buckets in the *AWS CLI Command Reference*.

```
 aws s3 list-buckets --account-id 123456789012 --endpoint-url https://s3api-endpoint-ip
```

For more information about the `list-buckets` command, see list-buckets in the *AWS CLI Command Reference*

The following SDK for Java example gets a list of buckets on Snowball Edge devices. For more information, see ListBuckets in the Amazon Simple Storage Service API Reference.

```
  import com.amazonaws.services.s3.model.*;
  public void listBuckets() {
    ListBucketsRequest reqListBuckets = new ListBucketsRequest()
    .withAccountId(AccountId)
    ListBucketsResult respListBuckets = s3APIClient.RegionalBuckets(reqListBuckets);
    System.out.printf("ListBuckets Response: %s%n", respListBuckets.toString());
  }
```

The following PowerShell example gets a list of buckets on Snowball Edge devices.

```
   Get-S3CRegionalBucketList -AccountId 012345678910 -Endpoint "https://snowball_ip" -
Region snow
```

The following .NET example gets a list of buckets on Snowball Edge devices.

```
using Amazon.S3Control;
using Amazon.S3Control.Model;

namespace SnowTest;

internal class Program
```

```
{
    static async Task Main(string[] args)
    {
        var config = new AmazonS3ControlConfig
        {
            ServiceURL = "https://snowball_ip",
            AuthenticationRegion = "snow" // Note that this is not RegionEndpoint
        };

        var client = new AmazonS3ControlClient(config);

        var response = await client.ListRegionalBucketsAsync(new
 ListRegionalBucketsRequest()
        {
            AccountId = "012345678910"
        });
    }
}
```

Get a bucket

The following example gets an Amazon S3 compatible storage on Snow Family devices bucket using the AWS CLI. To use this command, replace each user input placeholder with your own information.

```
aws s3control get-bucket --account-id 123456789012 --bucket DOC-EXAMPLE-BUCKET --
profile your-profile --endpoint-url https://s3ctrlapi-endpoint-ip
```

For more information about this command, see get-bucket in the *AWS CLI Command Reference*.

The following Amazon S3 compatible storage on Snow Family devices example gets a bucket using the SDK for Java. For more information, see GetBucket in the Amazon Simple Storage Service API Reference.

```
import com.amazonaws.services.s3control.model.*;

public void getBucket(String bucketName) {

    GetBucketRequest reqGetBucket = new GetBucketRequest()
            .withBucket(bucketName)
```

```
            .withAccountId(AccountId);

    GetBucketResult respGetBucket = s3ControlClient.getBucket(reqGetBucket);
    System.out.printf("GetBucket Response: %s%n", respGetBucket.toString());
}
```

Delete a bucket

Important

- The AWS account that creates the bucket owns it and is the only one that can delete it.
- Snow Family devices buckets must be empty before they can be deleted.
- You cannot recover a bucket after it has been deleted.

The following example deletes an Amazon S3 compatible storage on Snow Family devices bucket using the AWS CLI. To use this command, replace each user input placeholder with your own information.

```
aws s3control delete-bucket --account-id 123456789012 --bucket DOC-EXAMPLE-BUCKET --
profile your-profile --endpoint-url https://s3ctrlapi-endpoint-ip
```

For more information about this command, see delete-bucket in the *AWS CLI Command Reference*.

Working with S3 objects on a Snowball Edge device

This section describes various operations you can perform with objects on Amazon S3 compatible storage on Snow Family devices devices.

Copy an object to an Amazon S3 compatible storage on Snow Family devices bucket

The following example uploads a file named *sample-object.xml* to an Amazon S3 compatible storage on Snow Family devices bucket that you have write permissions for using the AWS CLI. To use this command, replace each user input placeholder with your own information.

```
aws s3api put-object --bucket sample-bucket --key sample-object.xml --body sample-
object.xml --profile your-profile --endpoint-url s3api-endpoint-ip
```

The following Amazon S3 compatible storage on Snow Family devices example copies an object into a new object in the same bucket using the SDK for Java. To use this command, replace each user input placeholder with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;
add : import java.io.IOException;

public class CopyObject {
    public static void main(String[] args) {
        String bucketName = "*** Bucket name ***";
        String sourceKey = "*** Source object key ***";
        String destinationKey = "*** Destination object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(sourceKey,
 destinationKey);
            s3Client.copyObject(copyObjectRequest);
            CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                    .sourceKey(sourceKey)
                    .destinationKey(destKey)
                    .build();
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
```

```
    }
}
```

Get an object from a bucket

The following example gets an object named *sample-object.xml* from an Amazon S3 compatible storage on Snow Family devices bucket using the AWS CLI. The SDK command is `s3-snow:GetObject`. To use this command, replace each user input placeholder with your own information.

```
aws s3api get-object --bucket sample-bucket --key sample-object.xml --profile your-profile --endpoint-url s3api-endpoint-ip
```

For more information about this command, see get-object in the *AWS CLI Command Reference*.

The following Amazon S3 compatible storage on Snow Family devices example gets an object using the SDK for Java. To use this command, replace each user input placeholder with your own information. For more information, see GetObject in the Amazon Simple Storage Service API Reference.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ResponseHeaderOverrides;
import com.amazonaws.services.s3.model.S3Object;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class GetObject {
    public static void main(String[] args) throws IOException {
        String bucketName = "*** Bucket name ***";
        String key = "*** Object key ***";

        S3Object fullObject = null, objectPortion = null, headerOverrideObject = null;
        try {
```

```
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .build());

s3Client.getObject(getObjectRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        } finally {
            // To ensure that the network connection doesn't remain open, close any
 open input streams.
            if (fullObject != null) {
                fullObject.close();
            }
            if (objectPortion != null) {
                objectPortion.close();
            }
            if (headerOverrideObject != null) {
                headerOverrideObject.close();
            }
        }
    }

    private static void displayTextInputStream(InputStream input) throws IOException {
        // Read the text input stream one line at a time and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line = null;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
```

```
}
```

List objects in a bucket

The following example lists objects in an Amazon S3 compatible storage on Snow Family devices bucket using the AWS CLI. The SDK command is `s3-snow:ListObjectsV2`. To use this command, replace each user input placeholder with your own information.

```
aws s3api list-objects-v2 --bucket sample-bucket --profile your-profile --endpoint-
url s3api-endpoint-ip
```

For more information about this command, see list-objects-v2 in the *AWS CLI Command Reference*.

The following Amazon S3 compatible storage on Snow Family devices example lists objects in a bucket using the SDK for Java. To use this command, replace each user input placeholder with your own information.

This example uses ListObjectsV2, which is the latest revision of the ListObjects API operation. We recommend that you use this revised API operation for application development. For backward compatibility, Amazon S3 continues to support the prior version of this API operation.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsV2Request;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;

public class ListObjectsV2 {

    public static void main(String[] args) {
        String bucketName = "*** Bucket name ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
```

```
                    .enableUseArnRegion()
                    .build();

            System.out.println("Listing objects");

            // maxKeys is set to 2 to demonstrate the use of
            // ListObjectsV2Result.getNextContinuationToken()
            ListObjectsV2Request req = new
 ListObjectsV2Request().withBucketName(bucketName).withMaxKeys(2);
            ListObjectsV2Result result;

            do {
                result = s3Client.listObjectsV2(req);

                for (S3ObjectSummary objectSummary : result.getObjectSummaries()) {
                    System.out.printf(" - %s (size: %d)\n", objectSummary.getKey(),
 objectSummary.getSize());
                }
                // If there are more than maxKeys keys in the bucket, get a
 continuation token
                // and list the next objects.
                String token = result.getNextContinuationToken();
                System.out.println("Next Continuation Token: " + token);
                req.setContinuationToken(token);
            } while (result.isTruncated());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Delete objects in a bucket

You can delete one or more objects from an Amazon S3 compatible storage on Snow Family devices bucket. The following example deletes an object named *sample-object.xml* using the AWS CLI. To use this command, replace each user input placeholder with your own information.

```
aws s3api delete-object --bucket sample-bucket --key key --profile your-profile --
endpoint-url s3api-endpoint-ip
```

For more information about this command, see delete-object in the *AWS CLI Command Reference*.

The following Amazon S3 compatible storage on Snow Family devices example deletes an object in a bucket using the SDK for Java. To use this example, specify the key name for the object that you want to delete. For more information, see DeleteObject in the Amazon Simple Storage Service API Reference.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

public class DeleteObject {
    public static void main(String[] args) {
        String bucketName = "*** Bucket name ***";
        String keyName = "*** key name ****";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build()));
            s3Client.deleteObject(deleteObjectRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

Supported REST API actions for Amazon S3 compatible storage on Snow Family devices

The following lists show the API operations that are supported by Amazon S3 compatible storage on Snow Family devices, including links to the related operations for Amazon S3 in AWS Regions.

Supported bucket API operations:

- CreateBucket
- DeleteBucket
- DeleteBucketLifecycle
- GetBucket
- GetBucketLifecycleConfiguration
- ListBuckets
- PutBucketLifecycleConfiguration

Supported object API operations:

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects
- DeleteObjectTagging
- GetObject
- GetObjectTagging

- HeadBucket
- HeadObject
- ListMultipartUploads
- ListObjects
- ListObjectsV2
- ListParts
- PutObject
- PutObjectTagging
- UploadPart
- UploadPartCopy

Clustering overview

For the AWS Snowball service, a cluster is a collective of Snowball Edge devices used as a single logical unit for local storage and compute purposes.

A cluster offers two primary benefits over a standalone Snowball Edge device for local storage and computing:

- **Increased durability** – The data stored in a cluster of Snowball Edge devices enjoys increased data durability over a single device. In addition, the data on the cluster remains as safe and viable as it was previously, despite possible Snowball Edge outages in the cluster. Clusters can withstand the loss of one device in clusters of 3 and 4 devicess and up to two devices in clusters of 5 to 16 devices before the data is in danger. You can also add or replace nodes.
- **Increased storage** – With Snowball Edge storage optimized devices, you can create a single, 16 node cluster with up to 2.6 PB of usable S3-compatible storage capacity. With Snowball Edge compute optimized devices, you can create a single, 16 node cluster of up to 501 TB of usable S3-compatible storage capacity.

Amazon S3 compatible storage on Snow Family devices cluster fault tolerance and storage capacity

Cluster size	Fault tolerance	Storage capacity of Snowball Edge Compute Optimized (with AMD EPYC Gen1, HDD, and optional GPU) devices	Storage capacity of Snowball Edge Compute Optimized (Compute Optimized with AMD EPYC Gen2 and NVMe) devices	Storage capacity of Snowball Edge storage optimized 210 TB devices
3	Loss of up to 1 node	83	38	438
4	Loss of up to 1 node	125	57	657
5	Loss of up to 2 nodes	125	57	657
6	Loss of up to 2 nodes	167	76	904
7	Loss of up to 2 nodes	209	95	1096
8	Loss of up to 2 nodes	250	114	1315
9	Loss of up to 2 nodes	292	133	1534
10	Loss of up to 2 nodes	334	152	1754

Cluster size	Fault tolerance	Storage capacity of Snowball Edge Compute Optimized (with AMD EPYC Gen1, HDD, and optional GPU) devices	Storage capacity of Snowball Edge Compute Optimized (Compute Optimized with AMD EPYC Gen2 and NVMe) devices	Storage capacity of Snowball Edge storage optimized 210 TB devices
11	Loss of up to 2 nodes	370	165	1970
12	Loss of up to 2 nodes	376	171	1973
13	Loss of up to 2 nodes	418	190	2192
14	Loss of up to 2 nodes	459	209	2411
15	Loss of up to 2 nodes	495	225	2625
16	Loss of up to 2 nodes	501	228	2631

A cluster of Snowball Edge devices is made of leaderless nodes. Any node can write data to and read data from the entire cluster, and all nodes are capable of performing the behind-the-scenes management of the cluster.

Snowball Edge cluster quorums

A *quorum* represents the minimum number of Snowball Edge devices in a cluster that must be communicating with each other to maintain a read/write quorum.

Suppose that you upload your data to a cluster of Snowball Edge devices. With all devices healthy, you have a *read/write quorum* for your cluster. If one or two of those nodes goes offline, you reduce the operational capacity of the cluster. However, you can still read and write to the cluster. In that sense, with the cluster operating all but one or two nodes, the cluster still has a read/write quorum. The number of nodes that can go offline before the operational capacity of the cluster is affected is found in this table.

Finally, quorom may be breached if a cluster loses more than the number of nodes indicated in this table. When a quorom is breached, the cluster is offline and the data in the cluster is unavailable. You might be able fix this, or the data might be permanently lost, depending on the severity of the event. If it is a temporary external power event, and you can power the three Snowball Edge devices back on and unlock all the nodes in the cluster, your data is available again.

> **Important**
>
> If a minimum quorum of healthy nodes doesn't exist, contact AWS Support.

You can determine the quorum state of your cluster by determining your node's lock state and network reachability. The `snowballEdge describe-cluster` command reports back the lock and network reachability state for every node in an unlocked cluster. Ensuring that the devices in your cluster are healthy and connected is an administrative responsibility that you take on when you create the cluster job. For more information about the different client commands, see Commands for the Snowball Edge Client.

Considerations for cluster jobs for Snowball Edge devices

Keep the following considerations in mind when planning to use a cluster of Snowball Edge devices:

- We recommend that you have a redundant power supply to reduce potential performance and stability issues for your cluster.
- As with standalone local storage and compute jobs, the data stored in a cluster can't be imported into Amazon S3 without ordering additional devices as a part of separate import jobs. If you order additional devices as import jobs, you can transfer the data from the cluster to the import job devices.
- To get data onto a cluster from Amazon S3, create a separate export job and copy the data from the devices of the export job onto the cluster.

- You can create a cluster job from the console, the AWS CLI, or one of the AWS SDKs. For a guided walkthrough of creating a job, see Getting Started.
- Cluster nodes have node IDs. A *node ID* is the same as the job ID for a device that you can get from the console, the AWS CLI, the AWS SDKs, and the Snowball Edge client. You can use node IDs to remove old nodes from clusters. You can get a list of node IDs by using the `snowballEdge describe-device` command on an unlocked device or the `describe-cluster` on an unlocked cluster.
- The lifespan of a cluster is limited by the security certificate granted to the cluster devices when the cluster is provisioned. By default, Snowball Edge devices can be used for up to 360 days before they need to be returned. At the end of that time, the devices stop responding to read/write requests. If you need to keep one or more devices for longer than 360 days, contact AWS Support.
- When AWS receives a returned device that was part of a cluster, we perform a complete erasure of the device. This erasure follows the National Institute of Standards and Technology (NIST) 800-88 standards.

Administering a cluster

Reading and writing data to a cluster

After you unlock a cluster, you're ready to store and access data on that cluster. You can use the Amazon S3 compatible endpoint to read from and write data to a cluster.

To read from or write data to a cluster, you must have a read/write quorum with no more than the allowed number of unavailable nodes in your cluster of devices.

Reconnecting an unavailable cluster node

A *node*, or device within a cluster, can become temporarily unavailable due to an issue like power or network loss without damaging the data on the node. When this happens, it affects the status of your cluster. A node's network reachability and lock status is reported in the Snowball Edge client by using the `snowballEdge describe-cluster` command.

We recommend that you physically position your cluster so you have access to the front, back, and top of all nodes. This way, you can access power and network cables on the back, shipping labels on the top for node IDs, and LCD screens on the front of the devices for the IP addresses and other administrative information.

When you detect that a node is unavailable, we recommend that you try one of the following procedures, depending on the scenario that caused the node to become unavailable.

To reconnect an unavailable node

1. Ensure that the node is powered on.
2. Ensure that the node is connected to the same internal network that the rest of the cluster is connected to.
3. If you need to power up the node, wait up to 20 minutes for it to finish.
4. Run the `snowballEdge unlock-cluster` command or the `snowballEdge associate-device` command. For an example, see Unlocking Snowball Edge devices.

To reconnect an unavailable node that lost network connectivity, but didn't lose power

1. Ensure that the node is connected to the same internal network that the rest of the cluster is on.
2. Run the `snowballEdge describe-device` command to see when the previously unavailable node is added back to the cluster. For an example, see Getting Device Status.

After you perform the preceding procedures, your nodes should be working normally. You should also have a read/write quorum. If that's not the case, then one or more of your nodes might have a more serious issue and might need to be removed from the cluster.

Adding or replacing a node in a cluster

You can add a new node after you have removed an unhealthy node from a cluster. You can also add a new node to increase local storage.

To add a new node, you first need to order a replacement. You can order a replacement node from the console, the AWS CLI, or one of the AWS SDKs. If you're ordering a replacement node from the console, you can order replacements for any job that hasn't been canceled or completed.

To order a replacement node from the console

1. Sign in to the AWS Snow Family Management Console.
2. Find and choose a job for a node that belongs to the cluster that you created from the Job dashboard.

3. For **Actions**, choose **Replace node**.

   Doing this opens the final step of the job creation wizard, with all settings identical to how the cluster was originally created.

4. Choose **Create job**.

Your replacement Snowball Edge is now on its way to you. When it arrives, use the following procedure to add it to your cluster.

To add a replacement node

1. Position the new node for the cluster such that you have access to the front, back, and top of all nodes.
2. Ensure that the node has power.
3. Ensure that the node is connected to the same internal network that the rest of the cluster is on.
4. Wait for the node to finish powering up (if it needed to be powered up).
5. Run the `snowballEdge associate-device` command. For an example, see Adding a Node to a Cluster.

Configuring Amazon S3 compatible storage on Snow Family devices event notifications

Amazon S3 compatible storage on Snow Family devices supports Amazon S3 event notifications for object API calls based on the MQTT protocol.

You can use Amazon S3 compatible storage on Snow Family devices to receive notifications when certain events happen in your S3 bucket. To enable notifications, add a notification configuration that identifies the events that you want the service to publish.

Amazon S3 compatible storage on Snow Family devices supports the following notification types:

- New object created events
- Object removal events
- Object tagging events

Configure Amazon S3 event notifications

1. Before you begin, you must have MQTT infrastructure in your network.
2. In your Snowball Edge client, run the `snowballEdge configure` command to set up the Snowball Edge device.

   When prompted, enter the following information:

   - The path to your manifest file.
   - The device's unlock code.
   - The device's endpoint (for example, **`https://10.0.0.1`**).
3. Run the following `put-notification-configuration` command to send notifications to an external broker.

   ```
   snowballEdge put-notification-configuration --broker-endpoint ssl://mqtt-broker-ip-address:8883 --enabled true --service-id s3-snow --ca-certificate file:path-to-mqtt-broker-ca-cert
   ```
4. Run the following `get-notification-configuration` command to verify that everything is set up correctly:

   ```
   snowballEdge get-notification-configuration --service-id s3-snow
   ```

   This returns the broker endpoint and enabled field.

After you configure the entire cluster to send notifications to the MQTT broker in the network, every object API call will result in an event notification.

> **Note**
>
> You need to subscribe to the topic s3SnowEvents/*`Device ID`* (or *`Cluster Id`* if it is a cluster)/bucketName. You can also use wildcards, for example topic name can be *`#`* or *`s3SnowEvents/#`*.

The following is an example Amazon S3 compatible storage on Snow Family devices event log:

```
{
"eventDetails": {
"additionalEventData": {
"AuthenticationMethod": "AuthHeader",
"CipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
"SignatureVersion": "SigV4",
"bytesTransferredIn": 1205,
"bytesTransferredOut": 0,
"x-amz-id-2": "uLdTfvdGTKlX6TBgCZtDd9Beef8wzUurA+Wpht7rKtfdaNsnxeLILg=="
},
"eventName": "PutObject",
"eventTime": "2023-01-30T14:13:24.772Z",
"requestAuthLatencyMillis": 40,
"requestBandwidthKBs": 35,
"requestID": "140CD93455CB62B4",
"requestLatencyMillis": 77,
"requestLockLatencyNanos": 1169953,
"requestParameters": {
"Content-Length": "1205",
"Content-MD5": "GZdTUOhYHvHgQgmaw2gl4w==",
"Host": "10.0.2.251",
"bucketName": "buckett",
"key": "file-key"
},
"requestTTFBLatencyMillis": 77,
"responseElements": {
"ETag": ""19975350e8581ef1e042099ac36825e3"",
"Server": "AmazonS3",
"x-amz-id-2": "uLdTfvdGTKlX6TBgCZtDd9Beef8wzUurA+Wpht7rKtfdaNsnxeLILg==",
"x-amz-request-id": "140CD93455CB62B4"
},
"responseStatusCode": 200,
"sourceIPAddress": "172.31.37.21",
"userAgent": "aws-cli/1.27.23 Python/3.7.16 Linux/4.14.301-224.520.amzn2.x86_64
 botocore/1.29.23",
"userIdentity": {
"identityType": "IAMUser",
"principalId": "531520547609",
"arn": "arn:aws:iam::531520547609:root",
"userName": "root"
}
}
}
```

For more information about Amazon S3 event notifications, see Amazon S3 Event Notifications.

Configuring local SMTP notifications

You can set up local notifications for your Snowball Edge devices with Simple Mail Transfer Protocol (SMTP). The local notifications send emails to configured servers when the service state (Active, Degraded, Inactive) changes, or if you cross capacity utilization thresholds of 80%, 90%, or 100%.

Prerequisites

Before you begin, confirm that:

- You have access to the latest Snowball Edge client.
- Your device is unlocked and ready to use.
- Your device can connect to the internet (if using Amazon Simple Email Service or external SMTP server) or to a local SMTP server.

Configuring the device

Set up your device to send you email notifications.

To configure the device for SMTP notifications

1. Run the following command to add an SMTP configuration to your device:

```
# If you don't specify a port, port 587 is the default.
SMTP_ENDPOINT=your-local-smtp-server-endpoint:port

# For multiple email recepients, separate with commas
RECIPIENTS_LIST=your-email-address

snowballEdge put-notification-configuration \
  --service-id local-monitoring \
  --enabled true \
  --type smtp \
  --broker-endpoint "$SMTP_ENDPOINT" \
  --sender example-sender@domain.com \
  --recipients "$RECIPIENTS_LIST"
```

You receive a test email from **example-sender@domain.com** if you're successful.

2. Test the configuration by running the following `get-notification-configuration` command:

```
snowballEdge get-notification-configuration \
  --service-id local-monitoring
```

The response doesn't include a password or certificate, even if you provide them.

Remote monitoring for Amazon S3 compatible storage on Snow Family devices

Remote monitoring allows AWS to monitor Amazon S3 compatible storage on Snow Family devices on Snowball Edge devices that are connected to an AWS Region. When remote monitoring is enabled, it triggers periodic service log uploads to the AWS Region. AWS monitors this information and can proactively notify you when we detect issues with the service. When remote monitoring is not enabled or if the Snowball Edge device or cluster are not connected to an AWS Region, the remote monitoring service will not attempt to publish internal device or service telemetry to the cloud. Remote monitoring is available for standalone Snowball Edge devices and clusters of Snowball Edge devices.

> **Note**
>
> Remote monitoring only enables monitoring for the Amazon S3 compatible storage on Snow Family devices service at this time.

You can use the `describe-features` command to see if the remote monitoring service is running or not. For more information, see Checking feature status in this guide.

To enable remote monitoring for a standalone device

- Use the `set-features` command and set the value of the `remote-monitoring-state` parameter to `INSTALLED_AUTOSTART`.

```
snowballEdge set-features /
```

```
    --remote-monitoring-state INSTALLED_AUTOSTART
    --manifest-file path/to/manifest.bin
    --unlock-code unlock-code
    --endpoint https://snow-device-local-ip
```

> **Note**
>
> For more information about the manifest file and unlock code of the Snow Family device, see Getting Your Credentials and Tools in this guide.

The command returns the following.

```
{
    "RemoteMonitoringState" : INSTALLED_AUTOSTART
}
```

To enable remote monitoring for a cluster of devices

- Use the `set-features` command and set the value of the `remote-monitoring-state` parameter to `INSTALLED_AUTOSTART` for each Snow Family device in the cluster.

```
snowballEdge set-features /
  --remote-monitoring-state INSTALLED_AUTOSTART
  --manifest-file path/to/manifest.bin
  --unlock-code unlock-code
  --endpoint https://snow-device-1-local-ip
snowballEdge set-features /
  --remote-monitoring-state INSTALLED_AUTOSTART
  --manifest-file path/to/manifest.bin
  --unlock-code unlock-code
  --endpoint https://snow-device-2-local-ip
snowballEdge set-features /
  --remote-monitoring-state INSTALLED_AUTOSTART
  --manifest-file path/to/manifest.bin
  --unlock-code unlock-code
```

```
    --endpoint https://snow-device-3-local-ip
```

> **Note**
>
> For more information about the manifest file and unlock code of the Snow Family device, see Getting Your Credentials and Tools in this guide.

Each time you run the command, it returns the following.

```
{
    "RemoteMonitoringState" : INSTALLED_AUTOSTART
}
```

To disable remote monitoring for a standalone device

- Use the `set-features` command and set the value of the `remote-monitoring-state` parameter to `INSTALLED_ONLY`. The Snow Family device will no longer periodically upload logs and AWS will not monitor nor notify you if issues with the service occur while remote monitoring is disabled.

```
  snowballEdge set-features /
    --remote-monitoring-state INSTALLED_ONLY
    --manifest-file path/to/manifest.bin
    --unlock-code unlock-code
    --endpoint https://snow-device-local-ip
```

The command returns the following.

```
{
    "RemoteMonitoringState" : INSTALLED_ONLY
}
```

To disable remote monitoring for a cluster of devices

- Use the `set-features` command and set the value of the `remote-monitoring-state` parameter to `INSTALLED_ONLY` for each Snow Family device in the cluster.

```
snowballEdge set-features /
  --remote-monitoring-state INSTALLED_ONLY
  --manifest-file path/to/manifest.bin
  --unlock-code unlock-code
  --endpoint https://snow-device-1-local-ip
snowballEdge set-features /
  --remote-monitoring-state INSTALLED_ONLY
  --manifest-file path/to/manifest.bin
  --unlock-code unlock-code
  --endpoint https://snow-device-2-local-ip
snowballEdge set-features /
  --remote-monitoring-state INSTALLED_ONLY
  --manifest-file path/to/manifest.bin
  --unlock-code unlock-code
  --endpoint https://snow-device-3-local-ip
```

Each time you run the command, it returns the following.

```
{
    "RemoteMonitoringState" : INSTALLED_ONLY
}
```

Using Amazon EKS Anywhere on AWS Snow

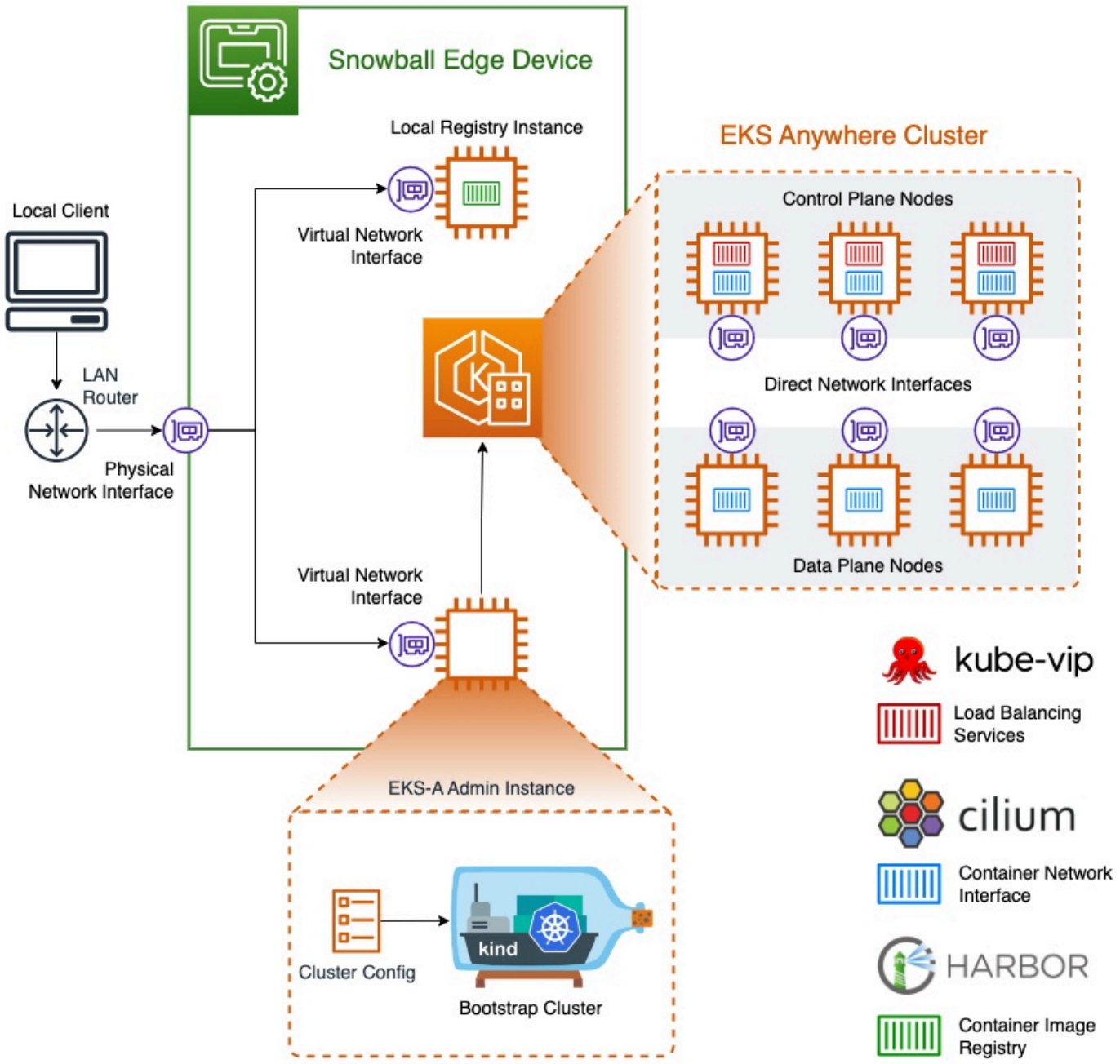
Amazon EKS Anywhere on AWS Snow helps you to create and operate Kubernetes clusters on Snow Family devices. Kubernetes is open-source software that's used for automating deployment, scaling, and management of containerized applications. You can use Amazon EKS Anywhere on a Snowball Edge device with or without an external network connection. To use Amazon EKS

Anywhere on a device without an external network connection, provide a container registry to run on the Snowball Edge device. For general information about Amazon EKS Anywhere, see the Amazon EKS Anywhere documentation.

Using Amazon EKS Anywhere on AWS Snow provides you with these capabilities:

- Provision a Kubernetes (K8s) cluster with Amazon EKS Anywhere CLI (eksctl anywhere) on Snowball Edge compute-optimized devices. You can provision Amazon EKS Anywhere on a single Snowball Edge device or three or more devices for high availability.
- Support for Cilium Container Network Interface (CNI).
- Support for Ubuntu 20.04 as the node operating system.

This diagram illustrates an Amazon EKS Anywhere cluster deployed on a Snowball Edge device.



We recommend that you create your Kubernetes cluster with the latest available Kubernetes version supported by Amazon EKS Anywhere. For more information, see Amazon EKS-Anywhere Versioning. If your application requires a specific version of Kubernetes, use any version of Kubernetes offered in standard or extended support by Amazon EKS. Consider the release and support dates of Kubernetes versions when planning the lifecycle of your deployment. This will help you avoid the potential loss of support for the version of Kubernetes you intend to use. For more information, see Amazon EKS Kubernetes release calendar.

For more information about Amazon EKS Anywhere on AWS Snow, see the Amazon EKS Anywhere documentation.

Topics

Actions to complete before ordering a Snowball Edge device for Amazon EKS Anywhere on AWS Snow

At this time, Amazon EKS Anywhere is compatible with Snowball Edge compute-optimized and compute-optimized with graphics processing unit (GPU) devices. Before you order a Snowball Edge device, there are a few things you should do to prepare.

- Build and supply an operating system image to use to create virtual machines on the device.
- Your network must have a static IP address available for the K8s control plane endpoint and allow Address Resolution Protocol (ARP).
- Your Snowball Edge device must have specific ports open. For more information about ports, see Ports and protocols in the Amazon EKS Anywhere documentation.

Topics

Create an Ubuntu EKS Distro AMI

To create the Ubuntu EKS Distro AMI, see Build Snow node images.

The name of the generated AMI will follow the pattern `capa-ami-ubuntu-20.04-version-timestamp`. For example, `capa-ami-ubuntu-20.04-v1.24-1672424524`.

Build a Harbor AMI

Set up a Harbor private registry AMI to include on the Snowball Edge device so you can use Amazon EKS Anywhere on the device without an external network connection. If you won't be using Amazon EKS Anywhere while the Snowball Edge device is disconnected from the external network, or if you have a private Kubernetes registry in an AMI to use on the device, you can skip this section.

To create the Harbor local registry AMI, see Build a Harbor AMI.

Ordering a Snowball Edge device for use with Amazon EKS Anywhere on AWS Snow

To order your Snowball Edge compute optimized or compute optimized with GPU device, see Creating a job to order a Snow Family device in this guide and keep these items in mind during the ordering process:

- In step 1, choose the **Local compute and storage only** job type.
- In step 2, choose the **Snowball Edge Compute Optimized** or **Snowball Edge Compute Optimized with GPU** device type.
- In step 3, choose **Amazon EKS Anywhere on AWS Snow**, then choose the Kubernetes version that you need.

> **Note**
>
> In order to deliver the latest software, we may configure the device with a version of ESK Anywhere newer than the one that is currently available. For more info, Versioning in the *Amazon EKS User Guide*.
> We recommend that you create your Kubernetes cluster with the latest available Kubernetes version supported by Amazon EKS Anywhere. For more information, see Amazon EKS-Anywhere Versioning. If your application requires a specific version of Kubernetes, use any version of Kubernetes offered in standard or extended support by Amazon EKS. Consider the release and support dates of Kubernetes versions when planning the lifecycle of your deployment. This will help you avoid the potential loss of support for the version of Kubernetes you intend to use. For more information, see Amazon EKS Kubernetes release calendar.

- Choose AMIs to include on your device, including the EKS Distro AMI (see Create an Ubuntu EKS Distro AMI) and, optionally, the Harbor AMI that you built (see Build a Harbor AMI).
- If you need multiple Snowball Edge devices for high availability, choose the number of devices that you need from **High Availability**.

After you receive your Snowball Edge device or devices, configure Amazon EKS Anywhere according to Configuring and running Amazon EKS Anywhere on Snowball Edge devices.

Configuring and running Amazon EKS Anywhere on Snowball Edge devices

Follow these procedures to configure and start Amazon EKS Anywhere on your Snowball Edge devices. Then, to configure Amazon EKS Anywhere to operate on disconnected devices, complete additional procedures before disconnecting those devices from the external network. For more information, see Configuring Amazon EKS Anywhere on AWS Snow for disconnected operation.

Topics

Initial setup

Perform the initial setup on each Snowball Edge device by connecting the device to your local network, downloading the Snowball Edge client, getting credentials, and unlocking the device.

Perform initial setup

1. Download and install the Snowball Edge client. For more information, see Downloading and Installing the Snowball Edge client.
2. Connect the device to your local network. For more information, see Connecting to Your Local Network.
3. Get credentials to unlock your device. For more information, see Getting credentials to access a Snow Family device.
4. Unlock the device. For more information, see Unlocking the Snow Family device. You can also use a script tool instead of unlocking devices manually. See Unlock devices.

Configuring and running Amazon EKS Anywhere on Snowball Edge devices automatically

You can use sample script tools to set up the environment and run an Amazon EKS Anywhere admin instance or you can do so manually. To use the script tools, see Unlock devices and setup environment for Amazon EKS Anywhere. After the environment is set up and the Amazon EKS Anywhere admin instance is running, if you need to configure Amazon EKS Anywhere to operate on the Snowball Edge device while disconnected from a network, see Configuring Amazon EKS Anywhere on AWS Snow for disconnected operation. Otherwise, see Create, upgrade, and delete Amazon EKS Anywhere clusters on Snowball Edge devices.

To manually set up the environment and run an Amazon EKS Anywhere admin instance, see Configuring and running Amazon EKS Anywhere on Snowball Edge devices manually.

Configuring and running Amazon EKS Anywhere on Snowball Edge devices manually

Topics

Create an AWS CLI profile

Create an AWS CLI profile to store credentials for use throughout the process of configuring Snowball Edge devices and the Amazon EKS Anywhere admin instance. For more information about AWS CLI profiles, see Named profiles for the AWS CLI in the AWS Command Line Interface User Guide.

You can use a sample script tool to automatically create the AWS CLI profile and the Amazon EKS Anywhere local IAM user. See Create credentials and certificates file. After using the script, resume with (Optional) Create and import a Secure Shell key. Otherwise, follow this procedure and then the procedures in Create an Amazon EKS Anywhere IAM local user.

> **Note**
>
> Do this for each Snowball Edge device that you configure.

```
PATH_TO_Snowball_Edge_CLIENT/bin/snowballEdge list-access-keys --endpoint
 https://snowball-ip --manifest-file path-to-manifest-file --unlock-code unlock-code
{
 "AccessKeyIds" : [ "xxxx" ]
}
```

Use the value of `AccessKeyIds` as the value of the `access-key-id` parameter of the `get-secret-access-key` command.

```
PATH_TO_Snowball_Edge_CLIENT/bin/snowballEdge get-secret-access-key --access-key-
id ACCESS_KEY_ID --endpoint https://snowball-ip --manifest-file path-to-manifest-file
 --unlock-code unlock-code
[snowballEdge]
aws_access_key_id = xxx
aws_secret_access_key = xxx
```

Use the value of `aws_access_key_id` and `aws_secret_access_key` as the values of `AWS Access Key ID` and `AWS Secret Access Key` of the AWS CLI profile.

```
aws configure --profile profile-name
AWS Access Key ID [None]: aws_access_key_id
AWS Secret Access Key [None]: aws_secret_access_key
Default region name [None]: snow
```

Create an Amazon EKS Anywhere IAM local user

For best security practices, create a local IAM user for Amazon EKS Anywhere on the Snowball Edge device. You can do this by manually using the following procedures.

> **Note**
>
> Do this for each Snowball Edge device that you use.

Create a local user

Use the `create-user` command to create the Amazon EKS Anywhere IAM user.

```
aws iam create-user --user-name user-name --endpoint http://snowball-ip:6078 --
profile profile-name
    {
        "User": {
            "Path": "/",
            "UserName": "eks-a-user",
            "UserId": "AIDACKCEVSQ6C2EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/eks-a-user",
            "CreateDate": "2022-04-06T00:13:35.665000+00:00"
        }
    }
```

Create a policy for the local user

Create a policy document, use it to create an IAM policy, and attach that policy to the Amazon EKS Anywhere local user.

To create a policy document and attach it to the Amazon EKS Anywhere local user

1. Create a policy document and save it to your computer. Copy the policy below to the document.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "snowballdevice:DescribeDevice",
        "snowballdevice:CreateDirectNetworkInterface",
```

```
        "snowballdevice:DeleteDirectNetworkInterface",
        "snowballdevice:DescribeDirectNetworkInterfaces",
        "snowballdevice:DescribeDeviceSoftware"
      ],
      "Resource": ["*"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:RunInstances",
        "ec2:DescribeInstances",
        "ec2:TerminateInstances",
        "ec2:ImportKeyPair",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeInstanceTypes",
        "ec2:DescribeImages",
        "ec2:DeleteTags"
      ],
      "Resource": ["*"]
    }
  ]
}
```

2. Use the `create-policy` command to create an IAM policy based on the policy document. The value of the `--policy-document` parameter should use the absolute path to the policy file. For example, `file:///home/user/policy-name.json`

```
aws iam create-policy --policy-name policy-name --policy-document file:///home/
user/policy-name.json --endpoint http://snowball-ip:6078 --profile profile-name
{
    "Policy": {
        "PolicyName": "policy-name",
        "PolicyId":
 "ANPACEMGEZDGNBVGY3TQOJQGEZAAAABP76TE5MKAAAABCCOTR2IJ43NBTJRZBU",
        "Arn": "arn:aws:iam::123456789012:policy/policy-name",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "CreateDate": "2022-04-06T04:46:56.907000+00:00",
        "UpdateDate": "2022-04-06T04:46:56.907000+00:00"
```

```
    }
}
```

3. Use the `attach-user-policy` command to attach the IAM policy to the Amazon EKS Anywhere local user.

```
aws iam attach-user-policy --policy-arn policy-arn --user-name user-name --endpoint
 http://snowball-ip:6078 --profile profile-name
```

Create an access key and a credential file

Create an access key for the Amazon EKS Anywhere IAM local user. Then, create a credential file and include in it the values of `AccessKeyId` and `SecretAccessKey` generated for the local user. The credential file will be used by the Amazon EKS Anywhere admin instance later.

1. Use the `create-access-key` command to create an access key for the Amazon EKS Anywhere local user.

```
aws iam create-access-key --user-name user-name --endpoint http://snowball-ip:6078
 --profile profile-name
    {
        "AccessKey": {
            "UserName": "eks-a-user",
            "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "Status": "Active",
            "SecretAccessKey": "RTT/wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
            "CreateDate": "2022-04-06T04:23:46.139000+00:00"
        }
    }
```

2. Create a credential file. In it, save the `AccessKeyId` and `SecretAccessKey` values in the following format.

```
[snowball-ip]
aws_access_key_id = ABCDEFGHIJKLMNOPQR2T
```

```
aws_secret_access_key = AfSD7sYz/TBZtzkReBl6PuuISzJ2WtNkeePw+nNzJ
region = snow
```

> **Note**
>
> If you're working with multiple Snowball Edge devices, the order of the credentials in the file doesn't matter, but the credentials for all devices do need to be in one file.

Create a certificates file for the admin instance

The Amazon EKS Anywhere admin instance needs the certificates of the Snowball Edge devices in order to run on them. Create a certificates file holding the certificate to access Snowball Edge devices for use later by the Amazon EKS Anywhere admin instance.

To create a certificates file

1. Use the `list-certificates` command to get certificates for each Snowball Edge device that you plan to use.

```
PATH_TO_Snowball_Edge_CLIENT/bin/snowballEdge list-certificates --endpoint
 https://snowball-ip --manifest-file path-to-manifest-file --unlock-code unlock-
code
{
  "Certificates" : [ {
    "CertificateArn" : "arn:aws:snowball-device:::certificate/xxx",
    "SubjectAlternativeNames" : [ "ID:JID-xxx" ]
  } ]
}
```

2. Use the value of `CertificateArn` as the value for the `--certificate-arn` parameter of the `get-certificate` command.

```
PATH_TO_Snowball_Edge_CLIENT/bin/snowballEdge get-certificate --certificate-arn ARN
 --endpoint https://snowball-ip --manifest-file path-to-manifest-file --unlock-
code unlock-code
```

3. Create a device certificate file. Put the output of `get-certificate` into the certificate file. Following is an example of how to save the output.

> **Note**
>
> If you're working with multiple Snowball Edge devices, the order of the credentials in the file doesn't matter, but the credentials for all devices do need to be in one file.

```
-----BEGIN CERTIFICATE-----
ZWtzYSBzbm93IHRlc3QgY2VydGlmaWNhdGUgZWtzYSBzbm93IHRlc3QgY2VydGlm
aWNhdGVla3NhIHNub3cgdGVzdCBjZXJ0aWZpY2F0ZWVrc2Egc25vdyB0ZXN0IGNl
cnRpZmljYXRlZWtzYSBzbm93IHRlc3QgY2VydGlmaWNhdGVla3NhIHNub3cgdGVz
dCBjZXJ0aWZpY2F0ZQMIIDXDCCAkSgAwIBAgIJAISM0nTVmbj+MA0GCSqGSIb3DQ
...
-----END CERTIFICATE-----
```

4. Repeat Create an Amazon EKS Anywhere IAM local user to create an IAM local user for Amazon EKS Anywhere on all Snowball Edge devices.

(Optional) Create and import a Secure Shell key

Use this optional procedure to create a Secure Shell (SSH) key to access all Amazon EKS Anywhere node instances and to import the public key to all Snowball Edge devices. Keep and secure this key file.

If you skip this procedure, Amazon EKS Anywhere will create and import an SSH key automatically when necessary. This key will be stored on the admin instance in `${PWD}/${CLUSTER_NAME}/eks-a-id_rsa`.

Create an SSH key and import it to the Amazon EKS Anywhere instance

1. Use the `ssh-keygen` command to generate a SSH key.

```
ssh-keygen -t rsa -C "key-name" -f path-to-key-file
```

2. Use the `import-key-pair` command to import the key from your computer to the Snowball Edge device.

> **Note**
>
> The value of the `key-name` parameter must be the same when you import the key to all devices.

```
aws ec2 import-key-pair --key-name key-name --public-key-material fileb:///path/to/
key-file --endpoint http://snowball-ip:8008 --profile profile-name
{
    "KeyFingerprint": "5b:0c:fd:e1:a0:69:05:4c:aa:43:f3:3b:3e:04:7f:51",
    "KeyName": "default",
    "KeyPairId": "s.key-85edb5d820c92a6f8"
}
```

Run an Amazon EKS Anywhere admin instance and transfer credential and certificate files to it

Run an Amazon EKS Anywhere admin instance

Follow this procedure to manually run an Amazon EKS Anywhere admin instance, configure a Virtual Network Interface (VNI) for the admin instance, check the status of the instance, create an SSH key, and connect to the admin instance with it. You can use a sample script tool to automate creating an Amazon EKS Anywhere admin instance and transferring credential and certificate files to this instance. See Create Amazon EKS Anywhere admin instance. After the script tool completes, you can ssh into the instance and create clusters by referring to Create, upgrade, and delete Amazon EKS Anywhere clusters on Snowball Edge devices. If you want to set up the Amazon EKS Anywhere instance manually, use the following steps..

> **Note**
>
> If you're using more than one Snowball Edge devices to provision the cluster, you can launch an Amazon EKS Anywhere admin instance on any of the Snowball Edge devices.

To run an Amazon EKS Anywhere admin instance

1. Use the `create-key-pair` command to create a SSH key for the Amazon EKS Anywhere admin instance. The command saves the key to `$PWD/key-file-name`.

```
aws ec2 create-key-pair --key-name key-name --query 'KeyMaterial' --output text --
endpoint http://snowball ip:8008 --profile profile-name > key-file-name
```

2. Use the `describe-images` command to find the image name that begins with `eks-anywhere-admin` from the output.

```
aws ec2 describe-images --endpoint http://snowball-ip:8008 --profile profile-name
```

3. Use the `run-instance` command to start an eks-a admin instance with the Amazon EKS Anywhere admin image.

```
aws ec2 run-instances --image-id eks-a-admin-image-id --key-name key-name --
instance-type sbe-c.xlarge --endpoint http://snowball-ip:8008 --profile profile-
name
```

4. Use the `describe-instances` command to check the status of the Amazon EKS Anywhere instance. Wait until the command indicates the instances state is `running` before continuing.

```
aws ec2 describe-instances --instance-id instance-id --endpoint http://snowball-
ip:8008 --profile profile-name
```

5. From the output of the `describe-device` command, note the value of `PhysicalNetworkInterfaceId` for the physical network interface that is connected to your network. You will use this to create a VNI.

```
PATH_TO_Snowball_Edge_CLIENT/bin/snowballEdge describe-device --endpoint
 https://snowball-ip --manifest-file path-to-manifest-file --unlock-code unlock-
code
```

6. Create a VNI for the Amazon EKS Anywhere admin instance. Use the value of `PhysicalNetworkInterfaceId` as the value of the `physical-network-interface-id` parameter.

```
PATH_TO_Snowball_Edge_CLIENT/bin/snowballEdge create-virtual-network-interface
 --ip-address-assignment dhcp --physical-network-interface-id PNI --endpoint
 https://snowball-ip --manifest-file path-to-manifest-file --unlock-code unlock-
code
```

7. Use the value of `IpAddress` as the value of the `public-ip` parameter of the `associate-address` command to associate the public address to the Amazon EKS Anywhere admin instance.

```
aws ec2 associate-address --instance-id instance-id --public-ip VNI-IP --endpoint
 http://snowball-ip:8008 --profile profile-name
```

8. Connect to the Amazon EKS Anywhere admin instance by SSH.

```
ssh -i path-to-key ec2-user@VNI-IP
```

Transfer certificate and credential files to the admin instance

After the Amazon EKS Anywhere admin instance is running, transfer the credentials and certificates of your Snowball Edge devices to the admin instance. Run the following command from the same directory where you saved the credentials and certificates files in Create an access key and a credential file and Create a certificates file for the admin instance.

```
scp -i path-to-key path-to-credentials-file path-to-certificates-file ec2-user@eks-
admin-instance-ip:~
```

Verify the contents of the files on the Amazon EKS Anywhere admin instance. Following are examples of the credential and certificate files.

```
[192.168.1.1]
aws_access_key_id = EMGEZDGNBVGY3TQOJQGEZB5ULEAAIWHWUJDXEXAMPLE
aws_secret_access_key = AUHpqjO0GZQHEYXDbN0neLNlfR0gEXAMPLE
region = snow

[192.168.1.2]
aws_access_key_id = EMGEZDGNBVGY3TQOJQGEZG5O7F3FJUCMYRMI4KPIEXAMPLE
aws_secret_access_key = kY4Cl8+RJAwq/bu28Y8fUJepwqhDEXAMPLE
region = snow
```

```
-----BEGIN CERTIFICATE-----
ZWtzYSBzbm93IHRlc3QgY2VydGlmaWNhdGUgZWtzYSBzbm93IHRlc3QgY2VydGlm
aWNhdGVla3NhIHNub3cgdGVzdCBjZXJ0aWZpY2F0ZWVrc2Egc25vdyB0ZXN0IGNl
cnRpZmljYXRlZWtzYSBzbm93IHRlc3QgY2VydGlmaWNhdGVla3NhIHNub3cgdGVz
dCBjZXJ0aWZpY2F0ZQMIIDXDCCAkSgAwIBAgIJAISM0nTVmbj+MA0GCSqGSIb3DQ
...
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
KJ0FPl2PAYPEjxr81/PoCXfZeARBzN9WLUH5yz1ta+sYUJouzhzWuLJYA1xqcCPY
mhVlkRsN4hVdlBNRnCCpRF766yjdJeibKVzXQxoXoZBjrOkuGwqRy3d3ndjK77h4
OR5Fv9mjGf7CjcaSjk/4iwmZvRSaQacb0YG5GVeb4mfUAuVtuFoMeYfnAgMBAAGj
azBpMAwGA1UdEwQFMAMBAf8wHQYDVR0OBBYEFL/bRcnBRuSM5+FcYFa8HfIBomdF
...
-----END CERTIFICATE-----
```

Configuring Amazon EKS Anywhere on AWS Snow for disconnected operation

Complete this additional configuration of Amazon EKS Anywhere on the Snowball Edge device while it's connected to a network to prepare Amazon EKS Anywhere to run in an environment without an external network connection.

To configure Amazon EKS Anywhere for disconnected use with your own local, private Kubernetes registry, see Registry Mirror configuration in the EKS Anywhere documentation.

If you created a Harbor private registry AMI, follow the procedures in this section.

Topics

- Configure the Harbor registry on a Snowball Edge device
- Use the Harbor registry on the Amazon EKS Anywhere admin instance

Configure the Harbor registry on a Snowball Edge device

See Configure Harbor on a Snowball Edge device.

Use the Harbor registry on the Amazon EKS Anywhere admin instance

See Import Amazon EKS Anywhere container images to the local Harbor registry on a Snowball Edge device.

Create, upgrade, and delete Amazon EKS Anywhere clusters on Snowball Edge devices

If you want to create a cluster in a static IP address range, ensure that you don't create other clusters on your Snowball Edge devices in the same IP address range. If you want to create another cluster using DHCP addressing on your Snowball Edge devices, ensure that all static IP address ranges that you use for clusters are not in the DHCP pool subnet.

> **Note**
>
> Best practice: Create a cluster and wait until it is successfully provisioned and running, then create another one.

To create, upgrade, and delete Amazon EKS Anywhere clusters, refer to Snow getstarted.

To upgrade an Amazon EKS Anywhere admin AMI or EKS Distro AMI, contact AWS Support. AWS Support will provide a Snowball Edge update containing the upgraded AMI. Then, download and install the Snowball Edge update. See Downloading updates and Installing updates.

After you upgrade your Amazon EKS Anywhere AMI, you need to start a new Amazon EKS Anywhere admin instance. See Run an Amazon EKS Anywhere admin instance. Then, copy key files, the cluster folder, credentials, and certificates from the previous admin instance to the upgraded instance. These are in a folder that's named for the cluster.

Clean up resources

If you create multiple clusters on your Snowball Edge devices and don't delete them correctly or if there is a problem in the cluster and the cluster creates replacement nodes after resuming, there will be resource leak. You can use a sample script tool to clean your Amazon EKS Anywhere admin instance and your Snowball Edge devices. See Amazon EKS Anywhere on AWS Snow cleanup tools.

Using IAM Locally

AWS Identity and Access Management (IAM) helps you securely control access to AWS resources that run on your AWS Snowball Edge device. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

IAM is supported locally on your device. You can use the local IAM service to create new users and attach IAM policies to them. You can use these policies to allow the access necessary to perform assigned tasks. For example, you can give a user the ability to transfer data, but limit their ability to create new Amazon EC2-compatible instances.

Additionally, you can create local, session-based credentials using AWS Security Token Service (AWS STS) on your device. For information about the IAM service, see Getting started in the *IAM User Guide*.

Your device's root credentials can't be disabled, and you can't use policies within your account to explicitly deny access to the AWS account root user. We recommend that you secure your root user access keys and create IAM user credentials for everyday interaction with your device.

> **Important**
>
> The documentation in this section applies to using IAM locally on an AWS Snowball Edge device. For information about using IAM in the AWS Cloud, see Identity and Access Management in AWS Snowball.
>
> For AWS services to work properly on a Snowball Edge, you must allow the ports for the services. For details, see Ports Required to Use AWS Services on an AWS Snowball Edge Device.

Topics

- Using the AWS CLI and API Operations on Snowball Edge
- List of Supported IAM AWS CLI Commands on a Snowball Edge

- IAM Policy Examples
- TrustPolicy Example

Using the AWS CLI and API Operations on Snowball Edge

When using the AWS CLI or API operations to issue IAM, AWS STS, Amazon S3, and Amazon EC2 commands on Snowball Edge, you must specify the `region` as "snow." You can do this using `aws configure` or within the command itself, as in the following examples.

```
aws configure --profile abc
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: 1234567
Default region name [None]: snow
Default output format [None]: json
```

Or

```
aws iam list-users --profile snowballEdge --endpoint http://192.0.2.0:6078 --region
 snow
```

Note

The access key ID and access secret key that are used locally on AWS Snowball Edge can't be interchanged with the keys in the AWS Cloud.

List of Supported IAM AWS CLI Commands on a Snowball Edge

Following is a description of the subset of AWS CLI commands and options for IAM that are supported on Snowball Edge devices. If a command or option isn't listed following, it's not supported. Unsupported parameters for commands are noted in the description.

- attach-role-policy – Attaches the specified managed policy to the specified IAM role.
- attach-user-policy – Attaches the specified managed policy to the specified user.

- create-access-key – Creates a new local IAM secret access key and corresponding AWS access key ID for the specified user.
- create-policy – Creates a new IAM managed policy for your device.
- create-role – Creates a new local IAM role for your device. The following parameters are **not** supported:
  - `Tags`
  - `PermissionsBoundary`
- create-user – Creates a new local IAM user for your device. The following parameters are **not** supported:
  - `Tags`
  - `PermissionsBoundary`
- delete-access-key – Deletes a new local IAM secret access key and corresponding AWS access key ID for the specified user.
- delete-policy – Deletes the specified managed policy.
- delete-role – Deletes the specified role.
- delete-user – Deletes the specified user.
- detach-role-policy – Removes the specified managed policy from the specified role.
- detach-user-policy – Removes the specified managed policy from the specified user.
- get-policy – Retrieves information about the specified managed policy, including the policy's default version and the total number of local IAM users, groups, and roles to which the policy is attached.
- get-policy-version – Retrieves information about the specified version of the specified managed policy, including the policy document.
- get-role – Retrieves information about the specified role, including the role's path, GUID, ARN, and the role's trust policy that grants permission to assume the role.
- get-user – Retrieves information about the specified IAM user, including the user's creation date, path, unique ID, and ARN.
- list-access-keys – Returns information about the access key IDs associated with the specified IAM user.
- list-attached-role-policies – Lists all managed policies that are attached to the specified IAM role.
- list-attached-user-policies – Lists all managed policies that are attached to the specified IAM user.

- list-entities-for-policy – Lists all local IAM users, groups, and roles that the specified managed policy is attached to.
  - `--EntityFilter`: Only the `user` and `role` values are supported.
- list-policies – Lists all the managed policies that are available in your local AWS account. The following parameter is **not** supported:
  - `--PolicyUsageFilter`
- list-roles – Lists the local IAM roles that have the specified path prefix.
- list-users – Lists the IAM users that have the specified path prefix.
- update-access-key – Changes the status of the specified access key from Active to Inactive, or vice versa.
- update-assume-role-policy – Updates the policy that grants an IAM entity permission to assume a role.
- update-role – Updates the description or maximum session duration setting of a role.
- update-user – Updates the name and/or the path of the specified IAM user.

Supported IAM API Operations

Following are the IAM API operations that you can use with a Snowball Edge, with links to their descriptions in the IAM API Reference.

- AttachRolePolicy – Attaches the specified managed policy to the specified IAM role.
- AttachUserPolicy – Attaches the specified managed policy to the specified user.
- CreateAccessKey – Creates a new local IAM secret access key and corresponding AWS access key ID for the specified user.
- CreatePolicy – Creates a new IAM managed policy for your device.
- CreateRole – Creates a new local IAM role for your device.
- CreateUser – Creates a new local IAM user for your device.

  The following parameters are **not** supported:
  - `Tags`
  - `PermissionsBoundary`
- DeleteAccessKey– Deletes the specified access key.
- DeletePolicy – Deletes the specified managed policy.

- DeleteRole – Deletes the specified role.
- DeleteUser – Deletes the specified user.
- DetachRolePolicy – Removes the specified managed policy from the specified role.
- DetachUserPolicy – Removes the specified managed policy from the specified user.
- GetPolicy – Retrieves information about the specified managed policy, including the policy's default version and the total number of local IAM users, groups, and roles to which the policy is attached.
- GetPolicyVersion – Retrieves information about the specified version of the specified managed policy, including the policy document.
- GetRole – Retrieves information about the specified role, including the role's path, GUID, ARN, and the role's trust policy that grants permission to assume the role.
- GetUser – Retrieves information about the specified IAM user, including the user's creation date, path, unique ID, and ARN.
- ListAccessKeys – Returns information about the access key IDs associated with the specified IAM user.
- ListAttachedRolePolicies – Lists all managed policies that are attached to the specified IAM role.
- ListAttachedUserPolicies – Lists all managed policies that are attached to the specified IAM user.
- ListEntitiesForPolicy – Retrieves information about the specified IAM user, including the user's creation date, path, unique ID, and ARN.
  - `--EntityFilter`: Only the `user` and `role` values are supported.
- ListPolicies – Lists all the managed policies that are available in your local AWS account. The following parameter is **not** supported:
  - `--PolicyUsageFilter`
- ListRoles – Lists the local IAM roles that have the specified path prefix.
- ListUsers – Lists the IAM users that have the specified path prefix.
- UpdateAccessKey – Changes the status of the specified access key from Active to Inactive, or vice versa.
- UpdateAssumeRolePolicy – Updates the policy that grants an IAM entity permission to assume a role.
- UpdateRole – Updates the description or maximum session duration setting of a role.
- UpdateUser – Updates the name and/or the path of the specified IAM user.

Supported IAM Policy Version and Grammar

Following is the local IAM support version 2012-10-17 of the IAM policy and a subset of the policy grammar.

Policy type	Supported grammar
Identity-based policies (user/role policy)	"Effect", "Action" and "Resource" **Note** Local IAM doesn't support "Condition ", "NotAction ", "NotResource " and "Principal ".
Resource-based policies (role trust policy)	"Effect", "Action" and "Principal " **Note** For Principal, only AWS account ID or principal ID is allowed.

IAM Policy Examples

Note

AWS Identity and Access Management (IAM) users need "snowballdevice:*" permissions to use the AWS OpsHub for Snow Family application to manage Snow Family devices.

The following are examples of policies that grant permissions to a Snowball Edge device.

Example 1: Allows the GetUser call for a sample user through the IAM API

Use the following policy to allow the GetUser call for a sample user through the IAM API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:GetUser",
            "Resource": "arn:aws:iam:::user/example-user"
        }
    ]
}
```

Example 2: Allows Full Access to the Amazon S3 API

Use the following policy to allow full access to the Amazon S3 API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"

        }
    ]
}
```

Example 3: Allows Read and Write Access to a Specific Amazon S3 Bucket

Use the following policy to allow read and write access to a specific bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::bucket-name"
        },
```

```
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource": "arn:aws:s3:::bucket-name/*"
        }
    ]
}
```

Example 4: Allows List, Get, and Put Access to a Specific Amazon S3 Bucket

Use the following policy to allow List, Get, and Put Access to a specific S3 bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:List*"
            ],
            "Resource": "arn:aws:s3:::examplebucket/*"
        }
    ]
}
```

Example 5: Allows Full Access to the Amazon EC2 API

Use the following policy to allow full access to Amazon EC2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        }
    ]
}
```

Example 6: Allows Access to Start and Stop Amazon EC2-compatible Instances

Use the following policy to allow access to start and stop Amazon EC2 instances.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

Example 7: Denies Calls to DescribeLaunchTemplates but Allows All Calls to DescribeImages

Use the following policy to deny calls to `DescribeLaunchTemplates` but allow all calls to `DescribeImages`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ec2:DescribeLaunchTemplates"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Example 8: Policy for API Calls

Lists all the managed policies that are available on your Snow device, including your own customer-defined managed policies. More details in list-policies.

```
aws iam list-policies --endpoint http://ip-address:6078 --profile snowballEdge --region
 snow
{
    "Policies": [
        {
            "PolicyName": "Administrator",
            "Description": "Root user admin policy for Account 123456789012",
            "CreateDate": "2020-03-04T17:44:59.412Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "policy-id",
            "DefaultVersionId": "v1",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/Administrator",
            "UpdateDate": "2020-03-04T19:10:45.620Z"
        }
    ]
}
```

TrustPolicy Example

A trust policy returns a set of temporary security credentials that you can use to access AWS resources that you might normally not have access to. These temporary credentials consist of an access key ID, a secret access key, and a security token. Typically, you use `AssumeRole` in your account for cross-account access.

The following is an example of a trust policy. For more information about trust policy, see AssumeRole in the *AWS Security Token Service API Reference*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": [
                    "arn:aws:iam::AccountId:root" //You can use the Principal ID
 instead of the account ID.
                ]
            },
            "Action": [
                "sts:AssumeRole"
            ]
        }
    ]
}
```

Using AWS Security Token Service

The AWS Security Token Service (AWS STS) helps you request temporary, limited-privilege credentials for IAM users.

> **Important**
>
> For AWS services to work properly on a Snowball Edge, you must allow the ports for the services. For details, see Ports Required to Use AWS Services on an AWS Snowball Edge Device.

Topics

- Using the AWS CLI and API Operations on Snowball Edge
- Supported AWS STSAWS CLI Commands on a Snowball Edge
- Supported AWS STS API Operations

Using the AWS CLI and API Operations on Snowball Edge

When using the AWS CLI or API operations to issue IAM, AWS STS, Amazon S3, and Amazon EC2 commands on Snowball Edge device, you must specify the `region` as "snow." You can do this using `AWS configure` or within the command itself, as in the following examples.

```
aws configure --profile snowballEdge
AWS Access Key ID [None]: defgh
```

```
AWS Secret Access Key [None]: 1234567
Default region name [None]: snow
Default output format [None]: json
```

Or

```
aws iam list-users --profile snowballEdge --endpoint http://192.0.2.0:6078 --region
 snow
```

> **Note**
>
> The access key ID and access secret key that are use locally on AWS Snowball Edge can't be interchanged with the keys in the AWS Cloud.

Supported AWS STSAWS CLI Commands on a Snowball Edge

Only the assume-role command is supported locally.

The following parameters are supported for `assume-role`:

- `role-arn`
- `role-session-name`
- `duration-seconds`

Example Command

To assume a role, use the following command.

```
    aws sts assume-role --role-arn "arn:aws:iam::123456789012:role/example-role" --
role-session-name AWSCLI-Session  --endpoint http://snow-device-IP-address:7078
```

For more information about using the `assume-role` command, see How do I assume an IAM role using the AWS CLI?

For more information about using AWS STS, see Using Temporary Security Credentials in the *IAM User Guide*.

Supported AWS STS API Operations

Only the AssumeRole API is supported locally.

The following parameters are supported for `AssumeRole`:

- `RoleArn`
- `RoleSessionName`
- `DurationSeconds`

Example

To assume a role, use the following.

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=AssumeRole
&RoleSessionName=session-example
&RoleArn=arn:aws:iam::123456789012:role/demo
&DurationSeconds=3600
```

Managing public key certificates

You can securely interact with AWS services running on a Snowball Edge device or a cluster of Snowball Edge devices through the HTTPS protocol by providing a public key certificate. You can use the HTTPS protocol to interact with AWS services such as IAM, Amazon EC2, S3 adapter, Amazon S3 compatible storage on Snow Family devices, Amazon EC2 Systems Manager, and AWS STS on Snowball Edge devices. In the case of a cluster of devices, a single certificate is required and can be generated by any device in the cluster. Once a Snowball Edge device generates the certificate and you unlock the device, you can use Snowball Edge client commands to list, get, and delete the certificate.

A Snowball Edge device generates a certificate when the following events occur:

- The Snowball Edge device or cluster is unlocked for the first time.
- The Snowball Edge device or cluster is unlocked after deleting the certificate (using the `delete-certificate` command or **Renew certificate** in AWS OpsHub).
- The Snowball Edge device or cluster is rebooted and unlocked after the certificate expires.

Whenever a new certificate is generated, the old certificate is no longer valid. A certificate is valid for a period of one year from the day it was generated.

You can also use AWS OpsHub for Snow Family to manage public key certificates. For more information, see Managing public key certificates using OpsHub in this guide.

Topics

Listing the certificate

Use the `list-certificates` command to see the Amazon Resource Names (ARNs) for the current certificate.

```
snowballEdge list-certificates
```

Example of `list-certificates` output

```
{
 "Certificates" : [ {
   "CertificateArn" : "arn:aws:snowball-
device:::certificate/78EXAMPLE516EXAMPLEf538EXAMPLEa7",
   "SubjectAlternativeNames" : [ "192.0.2.0" ]
 } ]
}
```

Getting certificates

Use the `get-certificate` command to see the content of the certificate based on the ARN provided. Use the `list-certificates` command to obtain the ARN of the certificate to use as the `certificate-arn` parameter.

```
  snowballEdge get-certificate --certificate-arn arn:aws:snowball-
device:::certificate/78EXAMPLE516EXAMPLEf538EXAMPLEa7
```

Example of `get-certificate` output

```
  -----BEGIN CERTIFICATE-----
  Certificate
  -----END CERTIFICATE-----
```

For information about configuring your certificate, see Specifying the S3 adapter as the AWS CLI endpoint.

Deleting certificates

Use the `delete-certificate` command to delete the current certificate. Use the `list-certificates` command to obtain the ARN of the certificate to use as the `certificate-arn` parameter. To generate a new certificate, reboot the Snowball Edge or each Snowball Edge in a cluster. See Rebooting the Snow Family device or use the `snowballEdge reboot-device` command.

```
  snowballEdge delete-certificate --certificate-arn arn:aws:snowball-
device:::certificate/78EXAMPLE516EXAMPLEf538EXAMPLEa7
```

Example of `delete-certificate` output

```
  The certificate has been deleted from your Snow device. Please reboot your Snowball
 Edge or Snowball Edge cluster to generate a new certificate.
```

Ports Required to Use AWS Services on an AWS Snowball Edge Device

For AWS services to work properly on an AWS Snowball Edge device, you must allow the network ports for the service.

The following is a list of network ports that are required for each AWS service.

Port	Protocol	Comment
22 (HTTP)	TCP	Device health check and for EC2 SSH
443 (HTTPS)	TCP	S3 API and S3 Control API HTTPS endpoint
2049 (HTTP)	TCP	NFS endpoint
6078 (HTTP)	TCP	IAM HTTP endpoint
6089 (HTTPS)	TCP	IAM HTTPS endpoint
7078 (HTTP)	TCP	STS HTTP endpoint
7089 (HTTPS)	TCP	STS HTTPS endpoint
8080 (HTTP)	TCP	S3 adapter HTTP endpoint
8008 (HTTP)	TCP	EC2 HTTP endpoint
8243 (HTTPS)	TCP	EC2 HTTPS endpoint
9091 (HTTP)	TCP	Endpoint for device management
9092	TCP	Inbound for EKS Anywhere and CAPAS device controller
8242	TCP	Inbound for EC2 HTTPS endpoint for EKS Anywhere
6443	TCP	Inbound for EKS Anywhere Kubernetes API endpoint
2379	TCP	Inbound for EKS Anywhere Etcd API endpoint

Port	Protocol	Comment
2380	TCP	Inbound for EKS Anywhere Etcd API endpoint

Using AWS Snow Device Management to Manage Devices

AWS Snow Device Management allows you to manage your Snow Family device and local AWS services remotely. All Snow Family devices support Snow Device Management, and it comes installed on new devices in most AWS Regions where Snow Family devices are available.

With Snow Device Management, you can perform the following tasks:

- Create a task
- Check task status
- Check task metadata
- Cancel a task
- Check device info
- Check Amazon EC2-compatible instance state
- List commands and syntax
- List remote-manageable devices
- List task status across devices
- List available resources
- List tasks by status
- List device or task tags
- Apply tags
- Remove tags

Topics

Choosing the Snow Device Management state when ordering a Snow Family device

When you create a job to order a Snow device, you can choose which state Snow Device Management will be in when you receive the device: installed but not activated or installed and activated. If it is installed but not activated, you will need to use AWS OpsHub or the Snowball Edge client to activate it before using it. If it is installed and activated, you can use Snow Device Management after receiving the device and connecting it to your local network. You can choose the Snow Device Management state when creating a job to order a device through the AWS Snow Family Management Console, the Snowball Edge client, the AWS CLI, or the Snow job management API.

To choose the Snow Device Management state from the AWS Snow Family Management Console

1. To choose for Snow Device Management to be installed and activated, choose **Manage your Snow device remotely with AWS OpsHub or Snowball client**.
2. To choose for Snow Device Management to be installed but not activated, do not select **Manage your Snow device remotely with AWS OpsHub or Snowball client**.

For more information, see Step 3: Choose your features and options in this guide.

To choose the Snow Device Management state from the AWS CLI, Snowball Edge client, or Snow job management API:

- Use the `remote-management` parameter to specify the Snow Device Management state. The `INSTALLED_ONLY` value of the parameter means Snow Device Management is installed but not activated. The `INSTALLED_AUTOSTART` value of the parameter means Snow Device Management is installed and activated. If you don't specify a value for this parameter, `INSTALLED_ONLY` is the default value.

Example of the syntax of the `remote-management` parameter of the `create-job` command

```
aws snowball create-job \
    --job-type IMPORT \
    --remote-management INSTALLED_AUTOSTART
```

```
      --device-configuration '{"SnowconeDeviceConfiguration": {"WirelessConnection":
 {"IsWifiEnabled": false} } }' \
      --resources '{"S3Resources":[{"BucketArn":"arn:aws:s3:::bucket-name"}]}' \
      --description "Description here" \
      --address-id ADID00000000-0000-0000-0000-000000000000 \
      --kms-key-arn arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
      --role-arn arn:aws:iam::000000000000:role/SnowconeImportGamma \
      --snowball-capacity-preference T8 \
      --shipping-option NEXT_DAY \
      --snowball-type SNC1_HDD \
      --region us-west-2 \
```

For more information, see Job Management API Reference in the AWS Snowball API Reference.

Activating Snow Device Management

Follow this procedure to activate Snow Device Management using the Snowball Edge client.

Before using this procedure, do the following:

- Download and install the latest version of the Snowball Edge client. For more information, see Downloading and Installing the Snowball Client.
- Download the manifest file and get the unlock code for the Snow Family device. For more information, see Getting Your Credentials and Tools.
- Connect the Snow Family device to your local network. For more information, see Connecting to Your Local Network.
- Unlock the Snow Family device. For more information, see Unlocking the Snowball Edge.

```
 snowballEdge set-features /
   --remote-management-state INSTALLED_AUTOSTART /
   --manifest-file JID1717d8cc-2dc9-4e68-aa46-63a3ad7927d2_manifest.bin /
   --unlock-code 7c0e1-bab84-f7675-0a2b6-f8k33 /
   --endpoint https://192.0.2.0:9091
```

The Snowball Edge client returns the following when the command is successful.

```
{
  "RemoteManagementState" : "INSTALLED_AUTOSTART"
}
```

Adding permissions for Snow Device Management to an IAM role

On the AWS account from which the device was ordered, create an AWS Identity and Access Management (IAM) role, and add the following policy to the role. Then, assign the role to the IAM user who will log in to remotely manage your device with Snow Device Management. For more information, see Creating IAM roles and Creating an IAM user in your AWS account.

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "snow-device-management:ListDevices",
                "snow-device-management:DescribeDevice",
                "snow-device-management:DescribeDeviceEc2Instances",
                "snow-device-management:ListDeviceResources",
                "snow-device-management:CreateTask",
                "snow-device-management:ListTasks",
                "snow-device-management:DescribeTask",
                "snow-device-management:CancelTask",
                "snow-device-management:DescribeExecution",
                "snow-device-management:ListExecutions",
                "snow-device-management:ListTagsForResource",
                "snow-device-management:TagResource",
                "snow-device-management:UntagResource"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Snow Device Management CLI commands

This section describes the AWS CLI commands that you can use to manage your Snow Family devices remotely with Snow Device Management. You can also perform some remote management tasks using AWS OpsHub for Snow Family. For more information, see Managing AWS services on your device.

> **Note**
>
> Before managing your device, make sure it is powered on, connected to your network, and can connect to the AWS Region where it was provisioned.

Topics

Create a task

To instruct one or more target devices to perform a task, such as unlocking or rebooting, use `create-task`. You specify target devices by providing a list of managed device IDs with the `--targets` parameter, and specify the tasks to perform with the `--command` parameter. Only a single command can be run on a device at a time.

Supported commands:

- `unlock` (no arguments)
- `reboot` (no arguments)

To create a task to be run by the target devices, use the following command. Replace each *`user input placeholder`* with your own information.

Command

```
aws snow-device-management create-task
--targets smd-fictbgr3rbcjeqa5
--command reboot={}
```

Exceptions

```
ValidationException
ResourceNotFoundException
InternalServerException
ThrottlingException
AccessDeniedException
ServiceQuotaExceededException
```

Output

```
{
    "taskId": "st-ficthmqoc2pht111",
    "taskArn": "arn:aws:snow-device-management:us-west-2:000000000000:task/st-
cjkwhmqoc2pht111"
```

```
}
```

Check task status

To check the status of a remote task running on one or more target devices, use the `describe-execution` command.

A task can have one of the following states:

- `QUEUED`
- `IN_PROGRESS`
- `CANCELED`
- `FAILED`
- `COMPLETED`
- `REJECTED`
- `TIMED_OUT`

To check the status of a task, use the following command. Replace each *`user input placeholder`* with your own information.

Command

```
aws snow-device-management describe-execution \
--taskId st-ficthmqoc2phtlef \
--managed-device-id smd-fictqic6gcldf111
```

Output

```
{
    "executionId": "1",
    "lastUpdatedAt": "2021-07-22T15:29:44.110000+00:00",
    "managedDeviceId": "smd-fictqic6gcldf111",
    "startedAt": "2021-07-22T15:28:53.947000+00:00",
    "state": "SUCCEEDED",
    "taskId": "st-ficthmqoc2pht111"
}
```

Check device info

To check device-specific information, such as the device type, software version, IP addresses, and lock status, use the `describe-device` command. The output also includes the following:

- `lastReachedOutAt` – When the device last contacted the AWS Cloud. Indicates that the device is online.
- `lastUpdatedAt` – When data was last updated on the device. Indicates when the device cache was refreshed.

To check device info, use the following command. Replace each *`user input placeholder`* with your own information.

Command

```
aws snow-device-management describe-device \
--managed-device-id smd-fictqic6gcldf111
```

Exceptions

```
ValidationException
ResourceNotFoundException
InternalServerException
ThrottlingException
AccessDeniedException
```

Output

```
{
    "associatedWithJob": "JID2bf11d5a-ea1e-414a-b5b1-3bf7e6a6e111",
    "deviceCapacities": [
        {
            "available": 158892032000,
```

```
            "name": "HDD Storage",
            "total": 158892032000,
            "unit": "Byte",
            "used": 0
        },
        {
            "available": 0,
            "name": "SSD Storage",
            "total": 0,
            "unit": "Byte",
            "used": 0
        },
        {
            "available": 3,
            "name": "vCPU",
            "total": 3,
            "unit": "Number",
            "used": 0
        },
        {
            "available": 5368709120,
            "name": "Memory",
            "total": 5368709120,
            "unit": "Byte",
            "used": 0
        },
        {
            "available": 0,
            "name": "GPU",
            "total": 0,
            "unit": "Number",
            "used": 0
        }
    ],
    "deviceState": "UNLOCKED",
    "deviceType": "SNC1_HDD",
    "lastReachedOutAt": "2021-07-23T21:21:56.120000+00:00",
    "lastUpdatedAt": "2021-07-23T21:21:56.120000+00:00",
    "managedDeviceId": "smd-fictqic6gcldf111",
    "managedDeviceArn": "arn:aws:snow-device-management:us-west-2:000000000000:managed-
device/smd-fictqic6gcldf111"
    "physicalNetworkInterfaces": [
        {
            "defaultGateway": "10.0.0.1",
```

```
            "ipAddress": "10.0.0.2",
            "ipAddressAssignment": "DHCP",
            "macAddress": "ab:cd:ef:12:34:56",
            "netmask": "255.255.252.0",
            "physicalConnectorType": "RJ45",
            "physicalNetworkInterfaceId": "s.ni-530f866d526d4b111"
        },
        {
            "defaultGateway": "10.0.0.1",
            "ipAddress": "0.0.0.0",
            "ipAddressAssignment": "STATIC",
            "macAddress": "ab:cd:ef:12:34:57",
            "netmask": "0.0.0.0",
            "physicalConnectorType": "RJ45",
            "physicalNetworkInterfaceId": "s.ni-8abc787f0a6750111"
        }
    ],
    "software": {
        "installState": "NA",
        "installedVersion": "122",
        "installingVersion": "NA"
    },
    "tags": {
        "Project": "PrototypeA"
    }
}
```

Check Amazon EC2-compatible instance state

To check the current state of the Amazon EC2 instance, use the `describe-ec2-instances` command. The output is similar to that of the `describe-device` command, but the results are sourced from the device cache in the AWS Cloud and include a subset of the available fields.

To check the state of the Amazon EC2-compatible instance, use the following command. Replace each *`user input placeholder`* with your own information.

Command

```
aws snow-device-management describe-device-ec2-instances \
--managed-device-id smd-fictbgr3rbcje111 \
```

```
--instance-ids s.i-84fa8a27d3e15e111
```

Exceptions

```
ValidationException
ResourceNotFoundException
InternalServerException
ThrottlingException
AccessDeniedException
```

Output

```
{
    "instances": [
        {
            "instance": {
                "amiLaunchIndex": 0,
                "blockDeviceMappings": [
                    {
                        "deviceName": "/dev/sda",
                        "ebs": {
                            "attachTime": "2021-07-23T15:25:38.719000-07:00",
                            "deleteOnTermination": true,
                            "status": "ATTACHED",
                            "volumeId": "s.vol-84fa8a27d3e15e111"
                        }
                    }
                ],
                "cpuOptions": {
                    "coreCount": 1,
                    "threadsPerCore": 1
                },
                "createdAt": "2021-07-23T15:23:22.858000-07:00",
                "imageId": "s.ami-03f976c3cadaa6111",
                "instanceId": "s.i-84fa8a27d3e15e111",
                "state": {
                    "name": "RUNNING"
                },
                "instanceType": "snc1.micro",
```

```
                "privateIpAddress": "34.223.14.193",
                "publicIpAddress": "10.111.60.160",
                "rootDeviceName": "/dev/sda",
                "securityGroups": [
                    {
                        "groupId": "s.sg-890b6b4008bdb3111",
                        "groupName": "default"
                    }
                ],
                "updatedAt": "2021-07-23T15:29:42.163000-07:00"
            },
            "lastUpdatedAt": "2021-07-23T15:29:58.
071000-07:00"
        }
    ]
}
```

Check task metadata

To check the metadata for a given task on a device, use the `describe-task` command. The metadata for a task includes the following items:

- The target devices
- The status of the task
- When the task was created
- When data was last updated on the device
- When the task was completed
- The description (if any) that was provided when the task was created

To check a task's metadata, use the following command. Replace each *`user input placeholder`* with your own information.

Command

```
aws snow-device-management describe-task \
--task-id st-ficthmqoc2pht111
```

Exceptions

```
ValidationException
ResourceNotFoundException
InternalServerException
ThrottlingException
AccessDeniedException
```

Output

```
{
    "completedAt": "2021-07-22T15:29:46.758000+00:00",
    "createdAt": "2021-07-22T15:28:42.613000+00:00",
    "lastUpdatedAt": "2021-07-22T15:29:46.758000+00:00",
    "state": "COMPLETED",
    "tags": {},
    "targets": [
        "smd-fictbgr3rbcje111"
    ],
    "taskId": "st-ficthmqoc2pht111",
    "taskArn": "arn:aws:snow-device-management:us-west-2:000000000000:task/st-
ficthmqoc2pht111"
}
```

Cancel a task

To send a cancel request for a specific task, use the `cancel-task` command. You can cancel only tasks in the QUEUED state that have not yet run. Tasks that are already running can't be canceled.

> **Note**
>
> A task that you're attempting to cancel might still run if it is processed from the queue before the `cancel-task` command changes the task's state.

To cancel a task, use the following command. Replace each *user input placeholder* with your own information.

Command

```
aws snow-device-management cancel-task \
--task-id st-ficthmqoc2pht111
```

Exceptions

```
ValidationException
ResourceNotFoundException
InternalServerException
ThrottlingException
AccessDeniedException
```

Output

```
{
    "taskId": "st-ficthmqoc2pht111"
}
```

List commands and syntax

To return a list of all supported commands for the Snow Device Management API, use the `help` command. You can also use the `help` command to return detailed information about and syntax for a given command.

To list all the supported commands, use the following command.

Command

```
aws snow-device-management help
```

To return detailed information and syntax for a command, use the following command. Replace *command* with the name of the command that you're interested in.

Command

```
aws snow-device-management command help
```

List remote-manageable devices

To return a list of all devices on your account that have Snow Device Management enabled in the AWS Region where the command is run, use the `list-devices` command. `--max-results` and `--next-token` are optional. For more information, see Using AWS CLI pagination options in the "AWS Command Line Interface User Guide".

To list remote-manageable devices, use the following command. Replace each *user input placeholder* with your own information.

Command

```
aws snow-device-management list-devices \
--max-results 10
```

Exceptions

```
ValidationException
InternalServerException
ThrottlingException
AccessDeniedException
```

Output

```
{
    "devices": [
```

```
        {
            "associatedWithJob": "ID2bf11d5a-ea1e-414a-b5b1-3bf7e6a6e111",
            "managedDeviceId": "smd-fictbgr3rbcjeqa5",
            "managedDeviceArn": "arn:aws:snow-device-management:us-
west-2:000000000000:managed-device/smd-fictbgr3rbcje111"
            "tags": {}
        }
    ]
}
```

List task status across devices

To return the status of tasks for one or more target devices, use the `list-executions` command. To filter the return list to show tasks that are currently in a single specific state, use the `--state` parameter. `--max-results` and `--next-token` are optional. For more information, see Using AWS CLI pagination options in the "AWS Command Line Interface User Guide".

A task can have one of the following states:

- `QUEUED`
- `IN_PROGRESS`
- `CANCELED`
- `FAILED`
- `COMPLETED`
- `REJECTED`
- `TIMED_OUT`

To list task status across devices, use the following command. Replace each *`user input placeholder`* with your own information.

Command

```
aws snow-device-management list-executions \
--taskId st-ficthmqoc2phtlef \
--state SUCCEEDED \
--max-results 10
```

Exceptions

```
ValidationException
InternalServerException
ThrottlingException
AccessDeniedException
```

Output

```
{
    "executions": [
        {
            "executionId": "1",
            "managedDeviceId": "smd-fictbgr3rbcje111",
            "state": "SUCCEEDED",
            "taskId": "st-ficthmqoc2pht111"
        }
    ]
}
```

List available resources

To return a list of the AWS resources available for a device, use the `list-device-resources` command. To filter the list by a specific type of resource, use the `--type` parameter. Currently, Amazon EC2-compatible instances are the only supported resource type. `--max-results` and `--next-token` are optional. For more information, see Using AWS CLI pagination options in the "AWS Command Line Interface User Guide".

To list the available resources for a device, use the following command. Replace each *`user input placeholder`* with your own information.

Command

```
aws snow-device-management list-device-resources \
--managed-device-id smd-fictbgr3rbcje111 \
--type AWS::EC2::Instance
```

```
--next-
token YAQGPwAT9l3wVKaGYjt4yS34MiQLWvzcShe9oIeDJr05AT4rXSprqcqQhhBEYRfcerAp0YYbJmRT=
--max-results 10
```

Exceptions

```
ValidationException
InternalServerException
ThrottlingException
AccessDeniedException
```

Output

```
{
    "resources": [
        {
            "id": "s.i-84fa8a27d3e15e111",
            "resourceType": "AWS::EC2::Instance"
        }
    ]
}
```

List device or task tags

To return a list of tags for a managed device or task, use the `list-tags-for-resource` command.

To list the tags for a device, use the following command. Replace the example Amazon Resource Name (ARN) with the ARN for your device.

Command

```
aws snow-device-management list-tags-for-resource
--resource-arn arn:aws:snow-device-management:us-west-2:123456789012:managed-device/
smd-fictbgr3rbcjeqa5
```

Exceptions

```
AccessDeniedException
InternalServerException
ResourceNotFoundException
ThrottlingException
```

Output

```
{
    "tags": {
        "Project": "PrototypeA"
    }
}
```

List tasks by status

Use the `list-tasks` command to return a list of tasks from the devices in the AWS Region where the command is run. To filter the results by `IN_PROGRESS`, `COMPLETED`, or `CANCELED` status, use the `--state` parameter. `--max-results` and `--next-token` are optional. For more information, see Using AWS CLI pagination options in the "AWS Command Line Interface User Guide".

To list tasks by status, use the following command. Replace each *`user input placeholder`* with your own information.

Command

```
aws snow-device-management list-tasks \
--state IN_PROGRESS \
--next-token K8VAMqKiP2Cf4xGkmH8GMyZrgOF8FUb+d1OKTP9+P4pUb+8PhW+6MiXh4= \
--max-results 10
```

Exceptions

```
ValidationException
InternalServerException
```

```
ThrottlingException
AccessDeniedException
```

Output

```
{
    "tasks": [
        {
            "state": "IN_PROGRESS",
            "tags": {},
            "taskId": "st-ficthmqoc2phtlef",
            "taskArn": "arn:aws:snow-device-management:us-west-2:000000000000:task/st-
ficthmqoc2phtlef"

        }
    ]
}
```

Apply tags

To add or replace a tag for a device, or for a task on a device, use the `tag-resource` command. The `--tags` parameter accepts a comma-separated list of `Key=Value` pairs.

To apply tags to a device, use the following command. Replace each *`user input placeholder`* with your own information.

Command

```
aws snow-device-management tag-resource \
--resource-arn arn:aws:snow-device-management:us-west-2:123456789012:managed-device/
smd-fictbgr3rbcjeqa5 \
--tags Project=PrototypeA
```

Exceptions

```
AccessDeniedException
InternalServerException
ResourceNotFoundException
```

```
ThrottlingException
```

Remove tags

To remove a tag from a device, or from a task on a device, use the `untag-resources` command.

To remove tags from a device, use the following command. Replace each *`user input placeholder`* with your own information.

Command

```
aws snow-device-management untag-resources \
--resource-arn arn:aws:snow-device-management:us-west-2:123456789012:managed-device/
smd-fictbgr3rbcjeqa5 \
--tag-keys Project
```

Exceptions

```
AccessDeniedException
InternalServerException
ResourceNotFoundException
ThrottlingException
```

Understanding AWS Snowball Edge Jobs

A *job* in AWS Snowball is a discrete unit of work, defined when you create it in the console or the job management API. With the AWS Snowball Edge device, there are three different job types, all of which are capable of local storage and compute functionality. This functionality uses the file interface or the Amazon S3 interface to read and write data. It triggers Lambda functions based on Amazon S3 PUT object API actions running locally on the AWS Snowball Edge device.

- Importing Jobs into Amazon S3 – The transfer of 80 TB or less of your local data copied onto a single device, and then moved into Amazon S3. For import jobs, Snowball devices and jobs have a one-to-one relationship. Each job has exactly one device associated with it. If you need to import more data, you can create new import jobs or clone existing ones. When you return a device of this job type, that data on it is imported into Amazon S3.
- Exporting Jobs from Amazon S3 – The transfer of any amount of data (located in Amazon S3), copied onto any number of Snowball Edge devices, and then moved one AWS Snowball Edge device at a time into your on-premises data destination. When you create an export job, it's split into job parts. Each job part is no more than 80 TB in size, and each job part has exactly one AWS Snowball Edge device associated with it. When you return a device of this job type, it's erased.
- Local Compute and Storage Only Jobs – These jobs involve one AWS Snowball Edge device, or multiple devices used in a cluster. These jobs don't start with data in their buckets like an export job, and they can't have data imported into Amazon S3 at the end like an import job. When you return a device of this job type, it's erased. With this job type, you also have the option of creating a cluster of devices. A cluster improves local storage durability and you can scale up or down with local storage capacity.

  In Regions where Lambda is not available, this job type is called *Local storage only*.

Job Details

Before creating a job, ensure the prerequisites are met. Each job is defined by the details that you specify when it's created. The following table describes all the details of a job.

Console identifier	API identifier	Detail description
Job name	`Description`	A name for the job, containin g alphanumeric character

Console identifier	API identifier	Detail description
		s, spaces, and any Unicode special characters.
Job type	`JobType`	The type of job, either import, export, or local compute and storage.
Job ID	`JobId`	A unique 39-character label that identifies your job. The job ID appears at the bottom of the shipping label that appears on the E Ink display, and in the name of a job's manifest file.
Address	`AddressId`	The address that the device will be shipped to. In the case of the API, this is the ID for the address data type.
Created date	CreationDate	The date that you created this job.
Shipping speed	`ShippingOption`	Speed options are based on region. For more information, see Shipping speeds.

Console identifier	API identifier	Detail description
IAM role ARN	RoleARN	This Amazon Resource Name (ARN) is the AWS Identity and Access Management (IAM) role that is created during job creation with write permissions for your Amazon S3 buckets. The creation process is automatic, and the IAM role that you allow AWS Snowball to assume is only used to copy your data between your S3 buckets and the Snowball. For more information, see Permissions Required to Use the AWS Snowball Console.
AWS KMS key	`KmsKeyARN`	In AWS Snowball, AWS Key Management Service (AWS KMS) encrypts the keys on each Snowball. When you create your job, you also choose or create an ARN for an AWS KMS encryption key that you own. For more information, see AWS Key Management Service in AWS Snowball Edge.
Snowball capacity	`SnowballCapacityPreference`	The storage capacity of the AWS Snowball device ordered in this job. The available size depends on your AWS Region.

Console identifier	API identifier	Detail description
Storage service	N/A	The AWS storage service associated with this job, in this case Amazon S3.
Resources	`Resources`	The AWS storage service resources associated with your job. In this case, these are the Amazon S3 buckets that your data is transferred to or from.
Job type	`JobType`	The type of job, either import, export, or local compute and storage.
Snowball type	`SnowballType`	The type of Snow Family device ordered in this job.
Cluster ID	`ClusterId`	A unique 39-character label that identifies your cluster.

Job Statuses

Each AWS Snowball Edge device job has a *status*, which changes to denote the current state of the job. This job status information doesn't reflect the health, the current processing state, or the storage used for the associated devices.

To see the status of a job

1. Log into the AWS Snow Family Management Console.
2. On the **Job dashboard**, choose the job.
3. Click on your job name within the console.
4. The Job Status pane will be located near the top and reflects the status of the job.

AWS Snowball Edge device job statuses

Console Identifier	API Identifier	Status Description
Job created	`New`	Your job has just been created. This status is the only one during which you can cancel a job or its job parts, if the job is an export job.
Preparing appliance	`PreparingAppliance`	AWS is preparing a device for your job.
Exporting	`InProgress`	AWS is exporting your data from Amazon S3 onto a device.
Preparing shipment	`PreparingShipment`	AWS is preparing to ship a device to you. The expected shipping tracking information is provided for customers in the status.
In transit to you	`InTransitToCustomer`	The device has been shipped to the address you provided during job creation.
Delivered to you	`WithCustomer`	The device has arrived at the address you provided during job creation.

Console Identifier	API Identifier	Status Description
In transit to AWS	`InTransitToAWS`	You have shipped the device back to AWS.
At sorting facility	`WithAWSSortingFacility`	The device for this job is at our internal sorting facility. Any additional processin g for import jobs into Amazon S3 will begin soon, typically within 2 days.
At AWS	`WithAWS`	Your shipment has arrived at AWS. If you're importing data, your import typically begins within a day of its arrival.
Importing	`InProgress`	AWS is importing your data into Amazon Simple Storage Service (Amazon S3).
Completed	`Complete`	Your job or a part of your job has completed successfu lly.
Canceled	`Cancelled`	Your job has been canceled.

Cluster Statuses

Each cluster has a *status*, which changes to denote the current general progress state of the cluster. Each individual node of the cluster has its own job status.

This cluster status information doesn't reflect the health, the current processing state, or the storage used for the cluster or its nodes.

Console Identifier	API Identifier	Status Description
Awaiting Quorum	`AwaitingQuorum`	The cluster hasn't been created yet, because there aren't enough nodes to begin processing the cluster request. For a cluster to be created, it must have at least five nodes.
Pending	`Pending`	Your cluster has been created, and we're getting its nodes ready to ship out. You can track the status of each node with that node's job status.
Delivered to you	`InUse`	At least one node of the cluster is at the address you provided during job creation.
Completed	`Complete`	All the nodes of the cluster have been returned to AWS.

Console Identifier	API Identifier	Status Description
Canceled	`Cancelled`	The request to make a cluster was canceled. Cluster requests can only be canceled before they enter the Pending state.

Importing Jobs into Amazon S3

With an import job, your data is copied to the AWS Snowball Edge device with the built-in Amazon S3 adapter or NFS mount point. Your data source for an import job should be on-premises. In other words, the storage devices that hold the data to be transferred should be physically located at the address that you provided when you created the job.

When you import files, each file becomes an object in Amazon S3 and each directory becomes a prefix. If you import data into an existing bucket, any existing objects with the same names as newly imported objects are overwritten. The import job type is also capable of local storage and compute functionality. This functionality uses the file interface or Amazon S3 adapter to read and write data, and triggers Lambda functions based off of Amazon S3 PUT object API actions running locally on the AWS Snowball Edge device.

When all of your data has been imported into the specified Amazon S3 buckets in the AWS Cloud, AWS performs a complete erasure of the device. This erasure follows the NIST 800-88 standards.

After your import is complete, you can download a job report. This report alerts you to any objects that failed the import process. You can find additional information in the success and failure logs.

Important

Don't delete your local copies of the transferred data until you can verify the results of the job completion report and review your import logs.

Exporting Jobs from Amazon S3

> **Note**
>
> Tags and metadata are NOT currently supported, in other words, all tags and metadata would be removed when exporting objects from S3 buckets.

Your data source for an export job is one or more Amazon S3 buckets. After the data for a job part is moved from Amazon S3 to an AWS Snowball Edge device, you can download a job report. This report alerts you to any objects that failed the transfer to the device. You can find more information in your job's success and failure logs.

You can export any number of objects for each export job, using as many devices as it takes to complete the transfer. Each AWS Snowball Edge device for an export job's job parts is delivered one after another, with subsequent devices shipping to you after the previous job part enters the **In transit to AWS** status.

When you copy objects into your on-premises data destination from a device using the Amazon S3 adapter or the NFS mount point, those objects are saved as files. If you copy objects into a location that already holds files, any existing files with the same names are overwritten. The export job type is also capable of local storage and compute functionality. This functionality uses the file interface or Amazon S3 adapter to read and write data, and triggers Lambda functions based off of Amazon S3 PUT object API actions running locally on the AWS Snowball Edge device.

When AWS receives a returned device, we completely erase it, following the NIST 800-88 standards.

> **Important**
>
> Data you want to export to a Snow device must be in Amazon S3. Any data in Amazon S3 Glacier that you plan to export to the Snow device will have to be thawed or moved into the S3 storage class before it can be exported. Do this before creating the Snow export job. Don't change, update, or delete the exported Amazon S3 objects until you can verify that all of your contents for the entire job have been copied to your on-premises data destination.

When you create an export job, you can export an entire Amazon S3 bucket or a specific range of objects keys.

Using Export Ranges

When you create an export job in the AWS Snow Family Management Console or with the job management API, you can export an entire Amazon S3 bucket or a specific range of objects keys. Object key names uniquely identify objects in a bucket. If you export a range, you define the length of the range by providing either an inclusive range beginning, an inclusive range ending, or both.

Ranges are UTF-8 binary sorted. UTF-8 binary data is sorted in the following way:

- The numbers 0–9 come before both uppercase and lowercase English characters.
- Uppercase English characters come before all lowercase English characters.
- Lowercase English characters come last when sorted against uppercase English characters and numbers.
- Special characters are sorted among the other character sets.

For more information about the specifics of UTF-8, see UTF-8 on Wikipedia.

Export Range Examples

Assume that you have a bucket containing the following objects and prefixes, sorted in UTF-8 binary order:

- 01
- Aardvark
- Aardwolf
- Aasvogel/apple
- Aasvogel/arrow/object1
- Aasvogel/arrow/object2
- Aasvogel/banana
- Aasvogel/banker/object1
- Aasvogel/banker/object2
- Aasvogel/cherry

- Banana
- Car

Specified range beginning	Specified range ending	Objects in the range that will be exported
(none)	(none)	All of the objects in your bucket
(none)	Aasvogel	01 Aardvark Aardwolf Aasvogel/apple Aasvogel/arrow/object1 Aasvogel/arrow/object2 Aasvogel/banana Aasvogel/banker/object1 Aasvogel/banker/object2 Aasvogel/cherry
(none)	Aasvogel/banana	01 Aardvark Aardwolf

Specified range beginning	Specified range ending	Objects in the range that will be exported
		Aasvogel/apple Aasvogel/arrow/object1 Aasvogel/arrow/object2 Aasvogel/banana
Aasvogel	(none)	Aasvogel/apple Aasvogel/arrow/object1 Aasvogel/arrow/object2 Aasvogel/banana Aasvogel/banker/object1 Aasvogel/banker/object2 Aasvogel/cherry Banana Car

Specified range beginning	Specified range ending	Objects in the range that will be exported
Aardwolf	(none)	Aardwolf Aasvogel/apple Aasvogel/arrow/object1 Aasvogel/arrow/object2 Aasvogel/banana Aasvogel/banker/object1 Aasvogel/banker/object2 Aasvogel/cherry Banana Car

Specified range beginning	Specified range ending	Objects in the range that will be exported
Aar	(none)	Aardvark Aardwolf Aasvogel/apple Aasvogel/arrow/object1 Aasvogel/arrow/object2 Aasvogel/banana Aasvogel/banker/object1 Aasvogel/banker/object2 Aasvogel/cherry Banana Car

Specified range beginning	Specified range ending	Objects in the range that will be exported
car	(none)	No objects are exported, and you get an error message when you try to create the job. Note that *car* is sorted below *Car* according to UTF-8 binary values.
Aar	Aarrr	Aardvark Aardwolf
Aasvogel/arrow	Aasvogel/arrox	Aasvogel/arrow/object1 Aasvogel/arrow/object2
Aasvogel/apple	Aasvogel/banana	Aasvogel/apple Aasvogel/arrow/object1 Aasvogel/arrow/object2 Aasvogel/banana

Specified range beginning	Specified range ending	Objects in the range that will be exported
Aasvogel/apple	Aasvogel/banker	Aasvogel/apple Aasvogel/arrow/object1 Aasvogel/arrow/object2 Aasvogel/banana Aasvogel/banker/object1 Aasvogel/banker/object2
Aasvogel/apple	Aasvogel/cherry	Aasvogel/apple Aasvogel/arrow/object1 Aasvogel/arrow/object2 Aasvogel/banana Aasvogel/banker/object1 Aasvogel/banker/object2 Aasvogel/cherry

Assume you have these three buckets and want to copy all objects from **folder2**.

- s3://bucket/folder1/
- s3://bucket/folder2/
- s3://bucket/folder3/

Specified range beginning	Specified range ending	Objects in the range that will be exported
folder2/	folder2/	All of the objects in bucket **folder2**.

Export Jobs Best Practices

- Ensure data is in Amazon S3, batch small files before ordering the job
- Ensure key ranges are specified in the export job definition if you have millions of objects in your bucket
- Update object keys to remove slash in the name as objects with trailing slashes in their names (/ or \) are not transferred to Snowball Edge
- For S3 buckets, the object length limitation is 255 characters.
- For S3 buckets that are version-enabled, only the current version of objects are exported.
- Delete markers are not exported.

Local Compute and Storage Only Jobs

Local compute and storage jobs enable you to use Amazon S3 compatible storage on Snow Family devices locally, without an internet connection. You can't export data from Amazon S3 to the device or import data into Amazon S3 when the device is returned.

Topics

- Local Storage Jobs

- Local Cluster Option

Local Storage Jobs

You can read and write objects to an AWS Snowball Edge device using Amazon S3 compatible storage on Snow Family devices or the S3 adapter. When you order a device, if you choose to use the S3 adapter, you also choose which Amazon S3 buckets will be included on the device when you receive it. If you choose to use Amazon S3 compatible storage on Snow Family devices, no Amazon S3 buckets are included on the device when you receive it.

You can create Amazon S3 buckets on the Snowball Edge devices to store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. Amazon S3 compatible storage on Snow Family devices provides a new storage class, `SNOW`, which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple Snowball Edge devices. You can use the same APIs and features on Snowball Edge buckets that you do on Amazon S3, including bucket lifecycle policies, encryption, and tagging. When the device or devices are returned to AWS, all data created or stored in Amazon S3 compatible storage on Snow Family devices is erased. For more information, see Local Compute and Storage Only Jobs.

For more information, see Amazon S3 compatible storage on Snow Family devices in this guide.

When you've finished using the device, return it to AWS, and the device will be erased. This erasure follows the National Institute of Standards and Technology (NIST) 800-88 standards.

Local Cluster Option

A cluster is a logical grouping of Snowball Edge devices, in groups of 3 to 16 devices. A cluster is created as a single job, which offers increased durability and storage size when compared to other AWS Snowball job offerings. For more information about cluster jobs, see Clustering overview in this guide.

Cloning a Job in the Console

When you first create an import job or a local compute and storage job, you might discover that you need more than one AWS Snowball Edge device. Because import jobs and local compute and storage jobs are associated with a single device, requiring more than one device means that you need to create more than one job. When creating additional jobs, you can go through the job creation wizard again in the console, or you can clone an existing job.

> **Note**
>
> Cloning a job is a shortcut available in the console to make creating additional jobs easier. If you're creating jobs with the job management API, you can simply run the job creation command again.

Cloning a job means re-creating it exactly, except for an automatically modified name. Cloning is a simple process.

To clone a job in the console

1. In the AWS Snow Family Management Console, choose your job from the table.
2. For **Actions**, choose **Clone job**.

   The **Create job** wizard opens to the last page, **Step 6: Review**.
3. Review the information and make any changes you want by choosing the appropriate **Edit** button.
4. To create your cloned job, choose **Create job**.

Cloned jobs are named in the format ***Job Name*-clone-*number***. The number is automatically added to the job name and represents the number of times you've cloned this job after the first time you clone it. For example, **AprilFinanceReports-clone** represents the first cloned job of **AprilFinanceReports** job, and **DataCenterMigration-clone-42** represents the forty-second clone of the **DataCenterMigration** job.

Best practices for using the Snowball Edge device

To help get the maximum benefit and satisfaction with your AWS Snowball Edge device, we recommend that you follow these best practices.

Security

The following are recommendations and best practices for maintaining security while working with an AWS Snowball Edge device.

General Security

- If you notice anything that looks suspicious about the AWS Snowball Edge device, don't connect it to your internal network. Instead, contact AWS Support, and a new AWS Snowball Edge device will be shipped to you.
- We recommend that you don't save a copy of the unlock code in the same location on the workstation as the manifest for that job. Saving these in different locations helps prevent unauthorized parties from gaining access to the AWS Snowball Edge device. For example, you can save a copy of the manifest to your local server, and email the code to a user that unlocks the device. This approach limits access to the AWS Snowball Edge device to individuals who have access to files saved on the server and the user's email address.
- The credentials displayed, when you run the Snowball Edge client commands list-access-keys and get-secret-access-key, are a pair of access keys used to access your device.

  These keys are only associated with the job and the local resources on the device. They don't map to your AWS account or any other AWS account. If you try to use these keys to access services and resources in the AWS Cloud, they will fail because they only work for the local resources associated with your job.
- If you feel your credentials are lost or have been compromised, request a new manifest file and unlock code by following the process to update the device's SSL certificate. See Updating the SSL certificate.

For information about how to use AWS Identity and Access Management (IAM) policies to control access, see AWS-Managed (Predefined) Policies for AWS Snowball Edge.

Network Security

- We recommend that you only use one method at a time for reading and writing data to a local bucket on an AWS Snowball Edge device. Using both the file interface and the Amazon S3 adapter on the same Amazon S3 bucket at the same time can result in read/write conflicts.
- To prevent corrupting your data, don't disconnect the AWS Snowball Edge device or change its network settings while transferring data.
- Files that are being written to on the device should be in a static state. Files that are modified while they are being written to can result in read/write conflicts.
- For more information about improving performance of your AWS Snowball Edge device, see Performance.

Resource Management

Consider the following best practices for managing jobs and resources on your AWS Snowball Edge device.

- The 10 free days for performing your on-premises data transfer start the day after the AWS Snowball Edge device arrives at your data center. This applies only to Snowball Edge device types.
- The **Job created** status is the only status in which you can cancel a job. When a job changes to a different status, you can't cancel the job. This applies to clusters.
- For import jobs, don't delete your local copies of the transferred data until the import into Amazon S3 is successful. As part of your process, be sure to verify the results of the data transfer.

Performance

> **Note**
>
> The data transfer performance you experience will vary based on the network environment, operating systems, copy method, protocol, source data read performance, and dataset characteristics such as file size. To determine the accurate data transfer rates and data transfer times, we recommend you to measure performance through proof-of-concept testing in your environment.

Following, you can find recommendations and information about AWS Snowball Edge device performance. This section describes performance in general terms, because on-premises environments have a different way of doing things—different network technologies, different hardware, different operating systems, different procedures, and so on.

The following table outlines how your network's transfer rate impacts how long it takes to fill a Snowball Edge device with data. Transferring smaller files reduces your transfer speed due to increased overhead. If you have many small files, we recommend that you zip them up into larger archives before transferring them onto a Snowball Edge device.

Rate (MB/s)	82 TB transfer time
800	1.22 days
450	2.11 days
400	2.37 days
300	3.16 days
277	3.42 days
200	4.75 days
100	9.49 days
60	15.53 days
30	31.06 days
10	85.42 days

To provide meaningful guidance about performance, the following sections describe how to determine when to use the AWS Snowball Edge device and how to get the most out of the service.

Topics

Performance Recommendations

The following practices are highly recommended, because they have the largest impact on improving the performance of your data transfer:

- We recommend that you have no more than 500,000 files or directories within each directory.
- We recommend that all files transferred to a Snowball Edge device be no smaller than 1 MB in size.
- If you have many files smaller than 1 MB in size, we recommend that you zip them up into larger archives before transferring them onto a Snowball Edge device.

Speeding Up Data Transfer

One of the best ways that you can improve the performance of an AWS Snowball Edge device is to speed up the transfer of data going to and from a device. In general, you can improve the transfer speed from your data source to the device in the following ways. This following list is ordered from largest to smallest positive impact on performance:

1. **Perform multiple write operations at one time** – To do this, run each command from multiple terminal windows on a computer with a network connection to a single AWS Snowball Edge device.
2. **Transfer small files in batches** – Each copy operation has some overhead because of encryption. To speed up the process, batch files together in a single archive. When you batch files together, they can be auto-extracted when they are imported into Amazon S3. For more information, see Batching small files.
3. **Don't perform other operations on files during transfer** – Renaming files during transfer, changing their metadata, or writing data to the files during a copy operation has a negative impact on transfer performance. We recommend that your files remain in a static state while you transfer them.
4. **Reduce local network use** – Your AWS Snowball Edge device communicates across your local network. So you can improve data transfer speeds by reducing other local network traffic between the AWS Snowball Edge device, the switch it's connected to, and the computer that hosts your data source.
5. **Eliminate unnecessary hops** – We recommend that you set up your AWS Snowball Edge device, your data source, and the computer running the terminal connection between them so that

they're the only machines communicating across a single switch. Doing so can improve data transfer speeds.

Updating software on Snowball Edge devices

AWS will notify you when new software is available for Snow Family devices you have. The notification is provided through email, AWS Health Dashboard, and as a CloudWatch event. The email notification is sent from Amazon Web Services, Inc. to the email address attached to the AWS account used to order the Snow Family device. When you receive the notification, follow the instructions in this topic and download and install the update as soon as possible to avoid interruption of your use of the device. For more information about AWS Health Dashboard, see AWS Health User Guide. For more information about CloudWatch Events, see Amazon CloudWatch Events User Guide.

You can download software updates from AWS and install them on Snowball Edge devices in your on-premises environments. These updates happen in the background. You can continue to use your devices as normal while the latest software is downloaded securely from AWS to your device. However, to apply downloaded updates, you must stop workloads on the device and restart it.

Software updates provided by AWS for Snowball Edge/Snowcone devices (Appliances) are Appliance Software as per Section 9 of the Service Terms.

The software updates are provided solely for the purpose of installing the software updates on the applicable Appliance on behalf of AWS. You will not (or attempt to), and will not permit or authorize third parties to (or attempt to) (i) make any copies of the software updates other than those necessary to install the software updates on the applicable Appliance, or (ii) circumvent or disable any features or measures in the software updates, including, but not limited to, any encryption applied to the software update. Once the software updates have been installed on the applicable Appliance, you agree to delete the software updates from any and all media utilized in installing the software updates to the Appliance.

> **Warning**
>
> We highly recommend that you suspend all activity on your device before installing the update. Updating the device and restarting will stop running instances and interrupt any writes to local Amazon S3 buckets.

Topics

Prerequisites

Before you can update your device, the following prerequisites must be met:

- You've created your job, have the device on-premises, and you've unlocked it. For more information, see Getting Started.
- Updating Snowball Edge devices is done through the Snowball Edge client. The latest version of the Snowball Edge client must be downloaded and installed on a computer in your local environment that has a network connection to the device you want to update. For more information, see Using the Snowball Edge Client.
- (Optional) We recommend that you configure a profile for the Snowball Edge client. For more information, see Configuring a Profile for the Snowball Edge Client.
- For Amazon S3 compatible storage on Snow Family devices on clustered Snowball Edge devices, stop the S3-Snow service and disable auto-start for it. See Configuring the Amazon S3 compatible storage on Snow Family devices service to autostart.

> **Note**
>
> For clustered devices, all commands have to be run for each device.

After you complete these tasks, you can download and install updates for Snowball Edge devices.

Downloading updates

There are two primary ways that you can download an update for Snow Family devices:

- You can trigger manual updates at any time using specific Snowball Edge client commands.
- You can programmatically determine a time to automatically update the device.

The following procedure outlines the process of manually downloading updates. For information about automatically updating your Snowball Edge device, see `configure-auto-update-strategy` in Updating a Snowball Edge.

> **Note**
>
> If your device has no access to the internet, you can download an update file using the GetSoftwareUpdates API. Then point to a local file location when you call `download-updates` using the `uri` parameter, as in the following example.
>
> ```
> snowballEdge download-updates --uri file:///tmp/local-update
> ```
>
> For Windows operating systems, format the value of the `uri` parameter as follows:
>
> ```
> snowballEdge download-updates --uri file:/C:/path/to/local-update
> ```

To check for and download Snowball Edge software updates for standalone devices

1. Open a terminal window, and ensure that the Snowball Edge device is unlocked using the `describe-device` command. If the device is locked, use the `unlock-device` command to unlock it. For more information, see Unlocking the Snow Family device.
2. When the device is unlocked, run the `snowballEdge check-for-updates` command. This command returns the latest available version of the Snowball Edge software, and also the current version installed on the device.
3. If your device software is out of date, run the `snowballEdge download-updates` command.

   > **Note**
   >
   > If your device is not connected to the internet, first download an update file using the GetSoftwareUpdates API. Then run the `snowballEdge download-updates` command using the `uri` parameter with a local path to the file that you downloaded, as in the following example.
   >
   > ```
   > snowballEdge download-updates --uri file:///tmp/local-update
   > ```

For Windows operating systems, format the value of the `uri` parameter as follows:

```
snowballEdge download-updates --uri file:/C:/path/to/local-update
```

4. You can check the status of this download with the `snowballEdge describe-device-software` command. While an update is downloading, you display the status using this command.

   Example output of `describe-device-software` command

```
Install State: Downloading
```

To check for and download Snowball Edge software updates for clusters of devices

1. Open a terminal window, and ensure that all of the Snowball Edge devices in the cluster are unlocked using the `snowballEdge describe-device` command. If the devices are locked, use the `snowballEdge unlock-cluster` command to unlock it. For more information, see Unlocking the Snowball Edge.
2. When all of the devices in the cluster are unlocked, for each device in the cluster, run the `check-for-updates` command. This command returns the latest available version of the Snowball Edge software, and also the current version installed on the device.

```
snowballEdge check-for-updates --unlock-code 29-character-unlock-code --manifest-
file path/to/manifest/file.bin --endpoint https://ip-address-of-snow-device
```

> **Note**
>
> The unlock code and manifest file are the same for all devices in the cluster.

Example of `check-for-updates` command

```
{
"InstalledVersion" : "118",
"LatestVersion" : "119"
}
```

If the value of the `LatestVersion` name is greater than the value of the `InstalledVersion` name, an update is available.

3. For each device in the cluster, use the `download-updates` command to download the update.

```
snowballEdge download-updates --uri file:///tmp/local-update
```

> **Note**
>
> For Windows operating systems, format the value of the `uri` parameter as follows:
>
> ```
> snowballEdge download-updates --uri file:/C:/path/to/local-update
> ```

4. To check the status of this download for each device in the cluster, use the `describe-device-software` command.

```
snowballEdge describe-device-software --unlock-code 29-character-unlock-code --
manifest-file path/to/manifest/file.bin --endpoint https://ip-address-of-snow-
device
```

Example of output of the `describe-device-software` command

```
{
"InstalledVersion" : "118",
"InstallingVersion" : "119",
"InstallState" : "DOWNLOADED",
```

```
"CertificateExpiry" : "Sat Mar 30 16:47:51 UTC 2024"
}
```

If the value of the `InstallState` name is `DOWNLOADED`, the update is done downloading and available to install.

Installing updates

After downloading updates, you must install them and restart your device for the updates to take effect. The following procedure guides you through manually installing updates.

For clusters of Snowball Edge devices, the update must be downloaded to and installed for each device in the cluster.

> **Note**
>
> Suspend all activity on the device before you install software updates. Installing updates stops running instances and interrupts any writes to Amazon S3 buckets on the device. This can result in lost data

To install software updates that were already downloaded to standalone Snow Family devices

1. Open a terminal window, and ensure that the Snowball Edge device is unlocked using the `describe-device` command. If the device is locked, use the `unlock-device` command to unlock it. For more information, see Unlocking the Snowball Edge.
2. Run the `list-services` command to see the services available on the device. The command returns the service IDs of each service available on the device.

```
snowballEdge list-services
```

Example of output of `list-services` command

```
{
```

```
  "ServiceIds" : [ "greengrass", "fileinterface", "s3", "ec2", "s3-snow" ]
}
```

3. For each service ID identified by the `list-services` command, run the `describe-service` command to see the status. Use this information to identify services to stop.

```
snowballEdge describe-service --service-id service-id
```

Example of output of `describe-service` command

```
{
"ServiceId" : "s3",
  "Status" : {
    "State" : "ACTIVE"
  },
"Storage" : {
"TotalSpaceBytes" : 99608745492480,
"FreeSpaceBytes" : 99608744468480
},
"Endpoints" : [ {
"Protocol" : "http",
"Port" : 8080,
"Host" : "192.0.2.0"
}, {
"Protocol" : "https",
"Port" : 8443,
"Host" : "192.0.2.0",
"CertificateAssociation" : {
"CertificateArn" : "arn:aws:snowball-
device:::certificate/6d955EXAMPLEdb71798146EXAMPLE3f0"
  }
 } ]
}
```

This output shows that the `s3` service is active and must be stopped using the `stop-service` command.

4. Use the `stop-service` command to stop each service where the value of the `State` name is `ACTIVE` in the output of the `list-services` command. If more than one service is running, stop each one before continuing.

   > **Note**
   >
   > The Amazon S3 adapter, Amazon EC2, AWS STS, and IAM services cannot be stopped. If Amazon S3 compatible storage on Snow Family devices is running, stop it before installing updates. Amazon S3 compatible storage on Snow Family devices has `s3-snow` as the `serviceId`.

   ```
   snowballEdge stop-service --service-id service-id --device-ip-addresses snow-device-1-ip-address snow-device-device-2-ip-address snow-device-3-ip-address --manifest-file path/to/manifest/file.bin --unlock-code 29-character-unlock-code --endpoint https://snow-device-ip-address
   ```

   Example of output of the `stop-service` command

   ```
   Stopping the AWS service on your Snowball Edge. You can determine the status of the
    AWS service using the describe-service command.
   ```

5. Run the `snowballEdge install-updates` command.

6. You can check the status of this installation with the `snowballEdge describe-device-software` command. While an update is installing, you display the status with this command.

   Example output

   ```
   Install State: Installing //Possible values[NA, Installing, Requires Reboot]
   ```

   You've successfully installed a software update for your Snowball Edge device. Installing an update does not automatically apply the update to the device. To finish installing the update, the device must be restarted.

> **Warning**
>
> Restarting your Snow Family device without stopping all activity on the device can result in lost data.

7. When all the services on the device have stopped, reboot the device, unlock the device, and reboot it again. This completes installation of the downloaded software updates. For more information about rebooting the device, see Rebooting the Snow Family device. For more information about unlocking the device, see Unlocking the Snowball Edge.
8. When the device powers on after the second reboot, unlock the device.
9. Run the `check-for-updates` command. This command returns the latest available version of the Snowball Edge software, and also the current version that is installed on the device.

To install software updates that were already downloaded to a cluster of Snowball Edge devices

1. For each device in the cluster, run the `describe-device` command to determine if the devices are unlocked. If the devices are locked, use the `unlock-cluster` command to unlock it. For more information, see Unlocking the Snowball Edge.
2. For each device in the cluster, run the `list-services` command to see the services available on the device. The command returns the service IDs of each service available on the device.

```
snowballEdge list-services
```

Example of output of `list-services` command

```
{
  "ServiceIds" : [ "greengrass", "fileinterface", "s3", "ec2", "s3-snow" ]
}
```

3. For each service ID identified by the `list-services` command, run the `describe-service` command to see the status. Use this information to identify services to stop.

```
snowballEdge describe-service --service-id service-id
```

Example of output of `describe-service` command

```
{
"ServiceId" : "s3",
  "Status" : {
    "State" : "ACTIVE"
  },
"Storage" : {
"TotalSpaceBytes" : 99608745492480,
"FreeSpaceBytes" : 99608744468480
},
"Endpoints" : [ {
"Protocol" : "http",
"Port" : 8080,
"Host" : "192.0.2.0"
}, {
"Protocol" : "https",
"Port" : 8443,
"Host" : "192.0.2.0",
"CertificateAssociation" : {
"CertificateArn" : "arn:aws:snowball-
device:::certificate/6d955EXAMPLEdb71798146EXAMPLE3f0"
  }
 } ]
}
```

This output shows that the `s3` service is active and must be stopped using the `stop-service` command.

4. For each device in the cluster, use the `stop-service` command to stop each service where the value of the `State` name is `ACTIVE` in the output of the `list-services` command. If more than one service is running, stop each one before continuing.

> **Note**
>
> The Amazon S3 adapter, Amazon EC2, AWS STS, and IAM services cannot be stopped. If Amazon S3 compatible storage on Snow Family devices is running, stop it before installing updates. Amazon S3 compatible storage on Snow Family devices has `s3-snow` as the `serviceId`.

```
snowballEdge stop-service --service-id service-id --device-ip-addresses snow-device-1-ip-address snow-device-device-2-ip-address snow-device-3-ip-address --manifest-file path/to/manifest/file.bin --unlock-code 29-character-unlock-code --endpoint https://snow-device-ip-address
```

Example of output of the `stop-service` command

```
Stopping the AWS service on your Snowball Edge. You can determine the status of the
 AWS service using the describe-service command.
```

5. For each device in the cluster, run the `install-updates` command.

```
snowballEdge install-updates
```

6. You can check the status of this installation with the `describe-device-software` command.

```
snowballEdge describe-device-software
```

Example of output of the `describe-device-service` command

```
Install State: Installing //Possible values[NA, Installing, Requires Reboot]
```

When the `Install State` is `Requires Reboot`, you've successfully installed the software update for your Snowball Edge device. Installing an update does not automatically apply the update to the device. To finish installing the update, the device must be restarted.

> **Warning**
>
> Restarting the Snowball Edge device without stopping all activity on the device can result in lost data.

7. Reboot all devices in the cluster, unlock the cluster, and reboot all devices in the cluster again. This completes installation of the downloaded software updates. For more information about rebooting the devices, see Rebooting the Snow Family device. For more information about unlocking the cluster of devices, see Unlocking the Snowball Edge.
8. After each device in the cluster has rebooted twice, unlock the cluster then use the `check-for-updates` command to verify the device was updated. This command returns the latest available version of the Snowball Edge software, and also the current version that is installed on the device. If the current version and the latest available version are the same, the device was updated successfully.

You have now successfully updated the Snow Family device or cluster of devices and confirmed that the update to the latest Snow Family software.

Updating the SSL certificate

If you plan to keep your Snow Family device for more than 360 days, you will need to update the Secure Sockets Layer (SSL) certificate on the device to avoid interruption of your use of the device. If the certificate expires, you will not be able to use the device and will have to return it to AWS.

AWS will notify you 30 days before the SSL certificate expires for Snow Family devices you have. The notification is provided through email, AWS Health Dashboard, and as a CloudWatch event. The email notification is sent from Amazon Web Services, Inc. to the email address attached to the AWS account used to order the Snow Family device. When you receive the notification, follow the instructions in this topic and request an update as soon as possible to avoid interruption of your use of the device. For more information about AWS Health Dashboard, see AWS Health User Guide. For more information about CloudWatch Events, see Amazon CloudWatch Events User Guide.

Updating the SSL certificate is done through the Snowball Edge client. The latest version of the Snowball Edge client must be downloaded and installed on a computer in your local environment that has a network connection to the device you want to update. For more information, see Using the Snowball Edge Client.

This topic explains how to determine when the certificate will expire and how to update your device.

1. Use the `snowballEdge describe-device-software` command to determine when the certificate will expire. In the output of the command, the value of `CertificateExpiry` includes the date and time at which the certificate will expire.

   Example of `describe-device-software` output

   ```
   Installed version: 101
   Installing version: 102
   Install State: Downloading
   CertificateExpiry : Thur Jan 01 00:00:00 UTC 1970
   ```

2. Contact AWS Support and request an SSL certificate update.
3. AWS Support will provide an update file. Download and install the update file.
4. Use the new unlock code and manifest file when Unlocking the Snowball Edge.

Updating your Amazon Linux 2 AMIs on Snow Family devices

As a best-practice for security, keep your Amazon Linux 2 AMIs up-to-date on Snow Family devices. Regularly check the Amazon Linux 2 AMI (HVM), SSD Volume Type (64-bit x86) in the AWS Marketplace for updates. When you identify the need to update your AMI, import the latest Amazon Linux 2 image to the Snow device. See Importing an Image into Your Device as an Amazon EC2-compatible AMI.

You can also get the latest Amazon Linux 2 image ID using the `ssm get-parameters` command in the AWS CLI.

```
  aws ssm get-parameters --names /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2 --query 'Parameters[0].[Value]' --region your-region
```

The command returns the latest image ID of the AMI. For example:

```
ami-0ccb473bada910e74
```

Security for AWS Snowball Edge

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to AWS Snowball, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS Snowball. The following topics show you how to configure AWS Snowball to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS Snowball resources.

Topics

Data Protection in AWS Snowball Edge

AWS Snowball conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs

all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

Topics

- Protecting Data in the Cloud
- Protecting Data On Your Device

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.

- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with AWS Snowball or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into AWS Snowball or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

Protecting Data in the Cloud

AWS Snowball protects your data when you're importing or exporting data into Amazon S3, when you create a job to order a Snow Family device, and when your device is updated. The following sections describe how you can protect your data when you use Snowball Edge and are online or interacting with AWS in the cloud.

Topics

Encryption for AWS Snowball Edge

When you're using a Snowball Edge to import data into S3, all data transferred to a device is protected by SSL encryption over the network. To protect data at rest, AWS Snowball Edge uses server side-encryption (SSE).

Server-Side Encryption in AWS Snowball Edge

AWS Snowball Edge supports server-side encryption with Amazon S3–managed encryption keys (SSE-S3). Server-side encryption is about protecting data at rest, and SSE-S3 has strong, multifactor encryption to protect your data at rest in Amazon S3. For more information on SSE-S3, see Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3) in the *Amazon Simple Storage Service User Guide*.

Currently, AWS Snowball Edge doesn't offer server-side encryption with customer-provided keys (SSE-C). Amazon S3 compatible storage on Snow Family devices offers SSS-C for local compute and storage jobs. However, you might want to use that SSE type to protect data that has been imported, or you might already use it on data you want to export. In these cases, keep the following in mind:

- **Import** –

  If you want to use SSE-C to encrypt the objects that you've imported into Amazon S3, you should consider using SSE-KMS or SSE-S3 encryption instead established as a part of that bucket's bucket policy. However, if you have to use SSE-C to encrypt the objects that you've imported into Amazon S3, then you will have to copy the object within your bucket to encrypt with SSE-C. A sample CLI command to achieve this is shown below:

```
aws s3 cp s3://mybucket/object.txt s3://mybucket/object.txt --sse-c --sse-c-key
 1234567891SAMPLEKEY
```

or

```
aws s3 cp s3://mybucket s3://mybucket --sse-c --sse-c-key 1234567891SAMPLEKEY --
recursive
```

- **Export** – If you want to export objects that are encrypted with SSE-C, first copy those objects to another bucket that either has no server-side encryption, or has SSE-KMS or SSE-S3 specified in that bucket's bucket policy.

Enabling SSE-S3 for Data Imported into Amazon S3 from a Snowball Edge

Use the following procedure in the Amazon S3 Management Console to enable SSE-S3 for data being imported into Amazon S3. No configuration is necessary in the AWS Snow Family Management Console or on the Snowball device itself.

To enable SSE-S3 encryption for the data that you're importing into Amazon S3, simply set the bucket policies for all the buckets that you're importing data into. You update the policies to deny upload object (`s3:PutObject`) permission if the upload request doesn't include the `x-amz-server-side-encryption` header.

To enable SSE-S3 for data imported into Amazon S3

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the bucket that you're importing data into from the list of buckets.
3. Choose **Permissions**.
4. Choose **Bucket Policy**.
5. In the **Bucket policy editor**, enter the following policy. Replace all the instances of *`YourBucket`* in this policy with the actual name of your bucket.

```
{
  "Version": "2012-10-17",
  "Id": "PutObjPolicy",
  "Statement": [
```

```
    {
      "Sid": "DenyIncorrectEncryptionHeader",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::YourBucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
      }
    },
    {
      "Sid": "DenyUnEncryptedObjectUploads",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::YourBucket/*",
      "Condition": {
        "Null": {
          "s3:x-amz-server-side-encryption": "true"
        }
      }
    }
  ]
}
```

6. Choose **Save**.

You've finished configuring your Amazon S3 bucket. When your data is imported into this bucket, it is protected by SSE-S3. Repeat this procedure for any other buckets, as necessary.

AWS Key Management Service in AWS Snowball Edge

AWS Key Management Service (AWS KMS) is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data. AWS KMS uses hardware security modules (HSMs) to protect the security of your keys. Specifically, the Amazon Resource Name (ARN) for the AWS KMS key that you choose for a job in AWS Snowball Edge is associated with a KMS key. That KMS key is used to encrypt the unlock code for your job. The unlock code is used to decrypt the top layer of encryption on your manifest file. The encryption keys stored within the manifest file are used to encrypt and de-encrypt the data on the device.

In AWS Snowball Edge, AWS KMS protects the encryption keys used to protect data on each AWS Snowball Edge device. When you create your job, you also choose an existing KMS key. Specifying the ARN for an AWS KMS key tells AWS Snowball which AWS KMS keys to use to encrypt the unique keys on the AWS Snowball Edge device. For more information on AWS Snowball Edge supported Amazon S3 server-side-encryption options , see Server-Side Encryption in AWS Snowball Edge.

Using the Managed Customer AWS KMS keys for Snowball Edge

If you'd like to use the managed customer AWS KMS keys for Snowball Edge created for your account, follow these steps.

To select the AWS KMS keys for your job

1. On the AWS Snow Family Management Console, choose **Create job**.
2. Choose your job type, and then choose **Next**.
3. Provide your shipping details, and then choose **Next**.
4. Fill in your job's details, and then choose **Next**.
5. Set your security options. Under **Encryption**, for **KMS key** either choose the AWS managed key or a custom key that was previously created in AWS KMS, or choose **Enter a key ARN** if you need to enter a key that is owned by a separate account.

   > **Note**
   >
   > The AWS KMS key ARN is a globally unique identifier for customer managed keys.

6. Choose **Next** to finish selecting your AWS KMS key.
7. Give the Snow device IAM user access to the KMS key.

   a. In the IAM console (https://console.aws.amazon.com/iam/), go to **Encryption Keys** and open the KMS key you chose to use to encrypt the data on the device.

   b. Under **Key Users**, select **Add**, search for the Snow device IAM user and select **Attach**.

Creating a Custom KMS Envelope Encryption Key

You have the option of using your own custom AWS KMS envelope encryption key with AWS Snowball Edge. If you choose to create your own key, that key must be created in the same region that your job was created in.

To create your own AWS KMS key for a job, see Creating Keys in the *AWS Key Management Service Developer Guide*.

Protecting Data On Your Device

Securing your AWS Snowball Edge

Following are some security points that we recommend you consider when using AWS Snowball Edge, and also some high-level information on other security precautions that we take when a device arrives at AWS for processing.

We recommend the following security approaches:

- When the device first arrives, inspect it for damage or obvious tampering. If you notice anything that looks suspicious about the device, don't connect it to your internal network. Instead, contact AWS Support, and a new device will be shipped to you.
- You should make an effort to protect your job credentials from disclosure. Any individual who has access to a job's manifest and unlock code can access the contents of the device sent for that job.
- Don't leave the device sitting on a loading dock. Left on a loading dock, it can be exposed to the elements. Although each AWS Snowball Edge device is rugged, weather can damage the sturdiest of hardware. Report stolen, missing, or broken devices as soon as possible. The sooner such an issue is reported, the sooner another one can be sent to complete your job.

> **Note**
>
> The AWS Snowball Edge devices are the property of AWS. Tampering with a device is a violation of the AWS Acceptable Use Policy. For more information, see http://aws.amazon.com/aup/.

We perform the following security steps:

- When transferring data with the Amazon S3 adapter, object metadata is not persisted. The only metadata that remains the same is `filename` and `filesize`. All other metadata is set as in the following example: `-rw-rw-r-- 1 root root [filesize] Dec 31 1969 [path/filename]`

- When transferring data with the file interface, object metadata is persisted.
- When a device arrives at AWS, we inspect it for any signs of tampering and to verify that no changes were detected by the Trusted Platform Module (TPM). AWS Snowball Edge uses multiple layers of security designed to protect your data, including tamper-resistant enclosures, 256-bit encryption, and an industry-standard TPM designed to provide both security and full chain of custody for your data.
- Once the data transfer job has been processed and verified, AWS performs a software erasure of the Snowball device that follows the National Institute of Standards and Technology (NIST) guidelines for media sanitization.

Validating NFC Tags

Snowball Edge Compute Optimized and Snowball Edge Storage Optimized (for data transfer) devices have NFC tags built into them. You can scan these tags with the AWS Snowball Edge Verification App, available on Android. Scanning and validating these NFC tags can help you verify that your device has not been tampered with before you use it.

Validating NFC tags includes using the Snowball Edge client to generate a device-specific QR code to verify that the tags you're scanning are for the right device.

The following procedure describes how to validate the NFC tags on a Snowball Edge device. Before you get started, make sure you've performed the following first five steps of the getting started exercise:

1. Create your Snowball Edge job. For more information, see Creating a job to order a Snow Family device
2. Receive the device. For more information, see Receiving the Snowball Edge.
3. Connect to your local network. For more information, see Connecting to Your Local Network.
4. Get your credentials and tools. For more information, see Getting credentials to access a Snow Family device.
5. Download and install the Snowball Edge client. For more information, see Downloading and Installing the Snowball Edge client.

To validate the NFC tags

1. Run the `snowballEdge get-app-qr-code` Snowball Edge client command. If you run this command for a node in a cluster, provide the serial number (`--device-sn`) to get a QR code for a single node. Repeat this step for each node in the cluster. For more information on using this command, see Getting Your QR Code for NFC Validation.

   The QR code is saved to a location of your choice as a .png file.
2. Navigate to the .png file that you saved, and open it so that you can scan the QR code with the app.
3. You can scan these tags using the AWS Snowball Edge Verification App on Android.

   > **Note**
   >
   > The AWS Snowball Edge Verification App is not available to download, but if you have a device with the app already installed, you can use the app.
4. Start the app, and follow the on-screen instructions.

You've now successfully scanned and validated the NFC tags for your device.

If you encounter issues while scanning, try the following:

- Confirm that your device has the Snowball Edge Compute Optimized options (with or without GPU).
- If you have the app on another device, try using that device.
- Move the device to an isolated area of the room, away from interference from other NFC tags, and try again.
- If issues persist, contact AWS Support.

Identity and Access Management in AWS Snowball

Every AWS Snowball job must be authenticated. You do this by creating and managing the IAM users in your account. Using IAM, you can create and manage users and permissions in AWS.

AWS Snowball users must have certain IAM-related permissions to access the AWS Snowball AWS Management Console to create jobs. An IAM user that creates an import or export job must

also have access to the right Amazon Simple Storage Service (Amazon S3) resources, such as the Amazon S3 buckets to be used for the job, AWS KMS resources, Amazon SNS topic, and Amazon EC2-compatible AMI for edge compute jobs.

> **Important**
>
> For information about using IAM locally on your device, see Using IAM Locally.

Topics

- Access Control for Snow Family Console and Creating Jobs

Access Control for Snow Family Console and Creating Jobs

As with all AWS services, access to AWS Snowball requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such an Amazon S3 bucket or an AWS Lambda function. AWS Snowball differs in two ways:

1. Jobs in AWS Snowball do not have Amazon Resource Names (ARNs).
2. Physical and network access control for a device on-premises is your responsibility.

See Identity and Access Management for AWS Snow Family for details on how you can use AWS Identity and Access Management (IAM) and AWS Snowball to help secure your resources by controlling who can access them in the AWS Cloud, and also local access control recommendations.

Identity and Access Management for AWS Snow Family

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS Snow Family resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies

- How AWS Snow Family works with IAM
- Identity-based policy examples for AWS Snow Family
- Troubleshooting AWS Snow Family identity and access

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS Snow Family.

Service user – If you use the AWS Snow Family service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS Snow Family features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS Snow Family, see Troubleshooting AWS Snow Family identity and access.

Service administrator – If you're in charge of AWS Snow Family resources at your company, you probably have full access to AWS Snow Family. It's your job to determine which AWS Snow Family features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS Snow Family, see How AWS Snow Family works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS Snow Family. To view example AWS Snow Family identity-based policies that you can use in IAM, see Identity-based policy examples for AWS Snow Family.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For

information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using

an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including

each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS Snow Family works with IAM

Before you use IAM to manage access to AWS Snow Family, learn what IAM features are available to use with AWS Snow Family.

IAM features you can use with AWS Snow Family

IAM feature	AWS Snow Family support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Forward access sessions (FAS)	Yes

IAM feature	AWS Snow Family support
Service roles	Yes
Service-linked roles	No

To get a high-level view of how AWS Snow Family and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for AWS Snow Family

Supports identity-based policies	Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for AWS Snow Family

To view examples of AWS Snow Family identity-based policies, see Identity-based policy examples for AWS Snow Family.

Resource-based policies within AWS Snow Family

Supports resource-based policies	Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific

resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Policy actions for AWS Snow Family

Supports policy actions	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of AWS Snow Family actions, see Actions defined by AWS Snow Family in the *Service Authorization Reference*.

Policy actions in AWS Snow Family use the following prefix before the action:

```
snowball
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "snowball:action1",
      "snowball:action2"
         ]
```

To view examples of AWS Snow Family identity-based policies, see Identity-based policy examples for AWS Snow Family.

Policy resources for AWS Snow Family

Supports policy resources	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of AWS Snow Family resource types and their ARNs, see Resources defined by AWS Snow Family in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS Snow Family.

To view examples of AWS Snow Family identity-based policies, see Identity-based policy examples for AWS Snow Family.

Policy condition keys for AWS Snow Family

Supports service-specific policy condition keys	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of AWS Snow Family condition keys, see Condition keys for AWS Snow Family in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Snow Family.

To view examples of AWS Snow Family identity-based policies, see Identity-based policy examples for AWS Snow Family.

ACLs in AWS Snow Family

Supports ACLs	No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with AWS Snow Family

Supports ABAC (tags in policies)	Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with AWS Snow Family

Supports temporary credentials	Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Forward access sessions for AWS Snow Family

Supports forward access sessions (FAS)	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for AWS Snow Family

Supports service roles	Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break AWS Snow Family functionality. Edit service roles only when AWS Snow Family provides guidance to do so.

Service-linked roles for AWS Snow Family

Supports service-linked roles	No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for AWS Snow Family

By default, users and roles don't have permission to create or modify AWS Snow Family resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by AWS Snow Family, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Snow Family in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the AWS Snow Family console
- Allow users to view their own permissions

Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS Snow Family resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on

specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.

- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the AWS Snow Family console

To access the AWS Snow Family console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the AWS Snow Family resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the AWS Snow Family console, also attach the AWS Snow Family `ConsoleAccess` or `ReadOnly` AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "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": "*"
        }
    ]
}
```

Troubleshooting AWS Snow Family identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Snow Family and IAM.

Topics

I am not authorized to perform an action in AWS Snow Family

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `snowball:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 snowball:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `snowball:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to AWS Snow Family.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS Snow Family. However, the action requires the service to have

permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

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

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my AWS Snow Family resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS Snow Family supports these features, see How AWS Snow Family works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Access Control in the AWS Cloud

You can have valid credentials to authenticate your requests in AWS. However, unless you have permissions you cannot create or access AWS resources. For example, you must have permissions to create a job to order a Snow Family device.

The following sections describe how to manage cloud-based permissions for AWS Snowball. We recommend that you read the overview first.

- Overview of Managing Access Permissions to Your Resources in the AWS Cloud
- Using Identity-Based Policies (IAM Policies) for AWS Snowball

Overview of Managing Access Permissions to Your Resources in the AWS Cloud

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

> **Note**
>
> An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see IAM Best Practices in the *IAM User Guide*.

Topics

- Resources and Operations
- Understanding Resource Ownership
- Managing Access to Resources in the AWS Cloud
- Specifying Policy Elements: Actions, Effects, and Principals
- Specifying Conditions in a Policy

Resources and Operations

In AWS Snowball, the primary resource is a *job*. AWS Snowball also has devices like the Snowball and the AWS Snowball Edge device, however, you can only use those devices in the context of an existing job. Amazon S3 buckets and Lambda functions are resources of Amazon S3 and Lambda respectively.

As mentioned previously, jobs don't have Amazon Resource Names (ARNs) associated with them. However, other services' resources, like Amazon S3 buckets, do have unique (ARNs) associated with them as shown in the following table.

AWS Snowball provides a set of operations to create and manage jobs. For a list of available operations, see the AWS Snowball API Reference.

Understanding Resource Ownership

The AWS account owns the resources that are created in the account, regardless of who created the resources. Specifically, the resource owner is the AWS account of the principal entity (that is, the root account, an IAM user, or an IAM role) that authenticates the resource creation request. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create a S3 bucket, your AWS account is the owner of the resource (in AWS Snowball, the resource is the job).
- If you create an IAM user in your AWS account and grant permissions to create a job to order a Snow Family device to that user, the user can create a job to order a Snow Family device. However, your AWS account, to which the user belongs, owns the job resource.
- If you create an IAM role in your AWS account with permissions to create a job, anyone who can assume the role can create a job to order a Snow Family device. Your AWS account, to which the role belongs, owns the job resource.

Managing Access to Resources in the AWS Cloud

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

> **Note**
>
> This section discusses using IAM in the context of AWS Snowball. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What Is IAM? in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM polices) and policies attached to a resource are referred to as *resource-based* policies. AWS Snowball supports only identity-based policies (IAM policies).

Topics

- Resource-Based Policies

Resource-Based Policies

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. AWS Snowball doesn't support resource-based policies.

Specifying Policy Elements: Actions, Effects, and Principals

For each job (see Resources and Operations), the service defines a set of API operations (see AWS Snowball API Reference) to create and manage said job. To grant permissions for these API operations, AWS Snowball defines a set of actions that you can specify in a policy. For example, for a job, the following actions are defined: `CreateJob`, `CancelJob`, and `DescribeJob`. Note that, performing an API operation can require permissions for more than one action.

The following are the most basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For more information, see Resources and Operations.

  > **Note**
  >
  > This is supported for Amazon S3, Amazon EC2, AWS Lambda, AWS KMS, and many other services.
  > Snowball does not support specifying a resource ARN in the `Resource` element of an IAM policy statement. To allow access to Snowball, specify `"Resource": "*"` in your policy.

- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, depending on the specified `Effect`, `snowball:*` either allows or denies the user permissions to perform all operations.

  > **Note**
  >
  > This is supported for Amazon EC2, Amazon S3, and IAM.

- **Effect** – You specify the effect when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.

> **Note**
>
> This is supported for Amazon EC2, Amazon S3, and IAM.

- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). AWS Snowball doesn't support resource-based policies.

To learn more about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

For a table showing all of the AWS Snowball API actions, see AWS Snowball API Permissions: Actions, Resources, and Conditions Reference.

Specifying Conditions in a Policy

When you grant permissions, you can use the IAM policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see Condition in the *IAM User Guide*.

To express conditions, you use predefined condition keys. There are no condition keys specific to AWS Snowball. However, there are AWS-wide condition keys that you can use as appropriate. For a complete list of AWS-wide keys, see Available Keys for Conditions in the *IAM User Guide*.

Using Identity-Based Policies (IAM Policies) for AWS Snowball

This topic provides examples of identity-based policies that demonstrate how an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles). These policies thereby grant permissions to perform operations on AWS Snowball resources in the AWS Cloud.

> **Important**
>
> We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your AWS Snowball resources. For more

information, see Overview of Managing Access Permissions to Your Resources in the AWS Cloud.

The sections in this topic cover the following:

- Permissions Required to Use the AWS Snowball Console
- AWS-Managed (Predefined) Policies for AWS Snowball Edge
- Customer Managed Policy Examples

The following shows an example of a permissions policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
       "Effect": "Allow",
       "Action": [
          "snowball:*",
          "importexport:*"
       ],
       "Resource": "*"
    }
  ]
}
```

The policy has two statements:

- The first statement grants permissions for three Amazon S3 actions (`s3:GetBucketLocation`, `s3:GetObject`, and `s3:ListBucket`) on all Amazon S3 buckets using the *Amazon Resource*

Name (ARN) of `arn:aws:s3:::*`. The ARN specifies a wildcard character (*) so the user can choose any or all Amazon S3 buckets to export data from.

- The second statement grants permissions for all AWS Snowball actions. Because these actions don't support resource-level permissions, the policy specifies the wildcard character (*) and the `Resource` value also specifies a wild card character.

The policy doesn't specify the `Principal` element because in an identity-based policy you don't specify the principal who gets the permission. When you attach a policy to a user, the user is the implicit principal. When you attach a permissions policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

For a table showing all of the AWS Snowball job management API actions and the resources that they apply to, see AWS Snowball API Permissions: Actions, Resources, and Conditions Reference.

Permissions Required to Use the AWS Snowball Console

The permissions reference table lists the AWS Snowball job management API operations and shows the required permissions for each operation. For more information about job management API operations, see AWS Snowball API Permissions: Actions, Resources, and Conditions Reference.

To use the AWS Snow Family Management Console, you need to grant permissions for additional actions as shown in the following permissions policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetBucketPolicy",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
```

```
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:GetFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": "arn:aws:lambda:*::function:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:ListFunctions"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:RetireGrant",
                "kms:ListKeys",
                "kms:DescribeKey",
                "kms:ListAliases"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreatePolicy",
                "iam:CreateRole",
```

```
                "iam:ListRoles",
                "iam:ListRolePolicies",
                "iam:PutRolePolicy"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "importexport.amazonaws.com"
                }
            }
        },
        {
           "Effect": "Allow",
           "Action": [
                "ec2:DescribeImages",
                "ec2:ModifyImageAttribute"
           ],
           "Resource": [
                "*"
           ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:ListTopics",
                "sns:GetTopicAttributes",
                "sns:SetTopicAttributes",
                "sns:ListSubscriptionsByTopic",
                "sns:Subscribe"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "greengrass:getServiceRoleForAccount"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "snowball:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

The AWS Snowball console needs these additional permissions for the following reasons:

- `ec2:` – These allow the user to describe Amazon EC2-compatible instances and modify their attributes for local compute purposes. For more information, see Using Amazon EC2-compatible compute instances.
- `kms:` – These allow the user to create or choose the KMS key that will encrypt your data. For more information, see AWS Key Management Service in AWS Snowball Edge.
- `iam:` – These allow the user to create or choose an IAM role ARN that AWS Snowball will assume to access the AWS resources associated with job creation and processing.
- `sns:` – These allow the user to create or choose the Amazon SNS notifications for the jobs they create. For more information, see Notifications for Snow Family devices.

AWS-Managed (Predefined) Policies for AWS Snowball Edge

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see AWS Managed Policies in the *IAM User Guide*.

You can use the following AWS-managed policies with AWS Snowball.

Creating an IAM Role Policy for Snowball Edge

An IAM role policy must be created with read and write permissions for your Amazon S3 buckets. The IAM role must also have a trust relationship with Snowball. Having a trust relationship means that AWS can write the data in the Snowball and in your Amazon S3 buckets, depending on whether you're importing or exporting data.

When you create a job to order a Snow Family device in the AWS Snow Family Management Console, creating the necessary IAM role occurs in step 4 in the **Permission** section. This process is automatic. The IAM role that you allow Snowball to assume is only used to write your data to your bucket when the Snowball with your transferred data arrives at AWS. The following procedure outlines that process.

To create the IAM role for your import job

1. Sign in to the AWS Management Console and open the AWS Snowball console at https://console.aws.amazon.com/importexport/.
2. Choose **Create job**.
3. In the first step, fill out the details for your import job into Amazon S3, and then choose **Next**.
4. In the second step, under **Permission**, choose **Create/Select IAM Role**.

   The IAM Management Console opens, showing the IAM role that AWS uses to copy objects into your specified Amazon S3 buckets.
5. Review the details on this page, and then choose **Allow**.

   You return to the AWS Snow Family Management Console, where **Selected IAM role ARN** contains the Amazon Resource Name (ARN) for the IAM role that you just created.
6. Choose **Next** to finish creating your IAM role.

The preceding procedure creates an IAM role that has write permissions for the Amazon S3 buckets that you plan to import your data into. The IAM role that is created has one of the following structures, depending on whether it's for an import job or export job.

IAM Role for an Import Job

```
        {
"Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:ListBucketMultipartUploads"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketPolicy",
        "s3:PutObject",
        "s3:AbortMultipartUpload",
        "s3:ListMultipartUploadParts",
        "s3:PutObjectAcl",
        "s3:ListBucket",
        "s3:HeadBucket"
      ],
      "Resource": "arn:aws:s3:::*"
    }
  ]
}
```

If you use server-side encryption with AWS KMS–managed keys (SSE-KMS) to encrypt the Amazon S3 buckets associated with your import job, you also need to add the following statement to your IAM role.

```
{
     "Effect": "Allow",
     "Action": [
       "kms:GenerateDataKey"
     ],
     "Resource": "arn:aws:kms:us-west-2:123456789012:key/abc123a1-abcd-1234-
efgh-111111111111"
}
```

If the object sizes are larger, the Amazon S3 client that is used for the import process uses multipart upload. If you initiate a multipart upload using SSE-KMS, then all the uploaded parts are encrypted using the specified AWS KMS key. Because the parts are encrypted, they must be decrypted before they can be assembled to complete the multipart upload. So you must have

permission to decrypt the AWS KMS key (`kms:Decrypt`) when you run a multipart upload to Amazon S3 with SSE-KMS.

The following is an example of an IAM role needed for an import job that needs `kms:Decrypt` permission.

```
{
    "Effect": "Allow",
     "Action": [
       "kms:GenerateDataKey","kms:Decrypt"

     ],
     "Resource": "arn:aws:kms:us-west-2:123456789012:key/abc123a1-abcd-1234-
efgh-111111111111"
}
```

The following is an example of an IAM role needed for an export job.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetBucketPolicy",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::*"
    }
  ]
}
```

If you use server-side encryption with AWS KMS–managed keys to encrypt the Amazon S3 buckets associated with your export job, you also need to add the following statement to your IAM role.

```
{
     "Effect": "Allow",
     "Action": [
            “kms:Decrypt”
      ],
```

```
    "Resource": "arn:aws:kms:us-west-2:123456789012:key/abc123a1-abcd-1234-
efgh-111111111111"
}
```

You can create your own custom IAM policies to allow permissions for API operations for AWS Snowball job management. You can attach these custom policies to the IAM users or groups that require those permissions.

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various AWS Snowball job management actions. These policies work when you are using AWS SDKs or the AWS CLI. When you are using the console, you need to grant additional permissions specific to the console, which is discussed in Permissions Required to Use the AWS Snowball Console.

> **Note**
>
> All examples use the us-west-2 region and contain fictitious account IDs.

Examples

- Example 1: Role Policy That Allows a User to Create a Job to order a Snow Family device with the API
- Example 2: Role Policy for Creating Import Jobs
- Example 3: Role Policy for Creating Export Jobs
- Example 4: Expected Role Permissions and Trust Policy
- AWS Snowball API Permissions: Actions, Resources, and Conditions Reference

Example 1: Role Policy That Allows a User to Create a Job to order a Snow Family device with the API

The following permissions policy is a necessary component of any policy that is used to grant job or cluster creation permission using the job management API. The statement is needed as a Trust Relationship policy statement for the Snowball IAM role.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
    {
         "Effect": "Allow",
         "Principal": {
         "Service": "importexport.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
    }
    ]
}
```

Example 2: Role Policy for Creating Import Jobs

You use the following role trust policy for creating import jobs for Snowball Edge that use AWS Lambda powered by AWS IoT Greengrass functions.

```
                {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketPolicy",
                "s3:GetBucketLocation",
                "s3:ListBucketMultipartUploads",
                "s3:ListBucket",
                "s3:HeadBucket",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts",
                "s3:PutObjectAcl",
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "snowball:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
                "iot:CreateKeysAndCertificate",
                "iot:CreatePolicy",
                "iot:CreateThing",
                "iot:DescribeEndpoint",
                "iot:GetPolicy"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:GetFunction"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:CreateCoreDefinition",
                "greengrass:CreateDeployment",
                "greengrass:CreateDeviceDefinition",
                "greengrass:CreateFunctionDefinition",
                "greengrass:CreateGroup",
                "greengrass:CreateGroupVersion",
                "greengrass:CreateLoggerDefinition",
```

```
                "greengrass:CreateSubscriptionDefinition",
                "greengrass:GetDeploymentStatus",
                "greengrass:UpdateGroupCertificateConfiguration",
                "greengrass:CreateGroupCertificateAuthority",
                "greengrass:GetGroupCertificateAuthority",
                "greengrass:ListGroupCertificateAuthorities",
                "greengrass:ListDeployments",
                "greengrass:GetGroup",
                "greengrass:GetGroupVersion",
                "greengrass:GetCoreDefinitionVersion"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Example 3: Role Policy for Creating Export Jobs

You use the following role trust policy for creating export jobs for Snowball Edge that use AWS Lambda powered by AWS IoT Greengrass functions.

```
                    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
           "Effect": "Allow",
           "Action": [
                "snowball:*"
           ],
```

```
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
                "iot:CreateKeysAndCertificate",
                "iot:CreatePolicy",
                "iot:CreateThing",
                "iot:DescribeEndpoint",
                "iot:GetPolicy"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:GetFunction"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:CreateCoreDefinition",
                "greengrass:CreateDeployment",
                "greengrass:CreateDeviceDefinition",
                "greengrass:CreateFunctionDefinition",
                "greengrass:CreateGroup",
                "greengrass:CreateGroupVersion",
                "greengrass:CreateLoggerDefinition",
                "greengrass:CreateSubscriptionDefinition",
                "greengrass:GetDeploymentStatus",
                "greengrass:UpdateGroupCertificateConfiguration",
                "greengrass:CreateGroupCertificateAuthority",
                "greengrass:GetGroupCertificateAuthority",
                "greengrass:ListGroupCertificateAuthorities",
```

```
                "greengrass:ListDeployments",
                "greengrass:GetGroup",
                "greengrass:GetGroupVersion",
                "greengrass:GetCoreDefinitionVersion"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Example 4: Expected Role Permissions and Trust Policy

The following expected role permissions policy is a necessary for an existing service role to use. It is a one time set up.

```
{
    "Version": "2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": ["[[snsArn]]"]
        },
        {
            "Effect": "Allow",
            "Action":
            [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricData",
                "cloudwatch:PutMetricData"
            ],
            "Resource":
            [
                "*"
            ],
            "Condition": {
                    "StringEquals": {
                        "cloudwatch:namespace": "AWS/SnowFamily"
                    }
            }
```

```
        }
    ]
}
```

The following expected role trust policy is a necessary for an existing service role to use. It is a one time set up.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "importexport.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS Snowball API Permissions: Actions, Resources, and Conditions Reference

When you are setting up Access Control in the AWS Cloud and writing a permissions policy that you can attach to an IAM identity (identity-based policies), you can use the following list as a reference. The includes each AWS Snowball job management API operation and the corresponding actions for which you can grant permissions to perform the action. It also includes for each API operation the AWS resource for which you can grant the permissions. You specify the actions in the policy's `Action` field, and you specify the resource value in the policy's `Resource` field.

You can use AWS-wide condition keys in your AWS Snowball policies to express conditions. For a complete list of AWS-wide keys, see Available Keys in the *IAM User Guide*.

> **Note**
>
> To specify an action, use the `snowball:` prefix followed by the API operation name (for example, `snowball:CreateJob`).

Logging and Monitoring in AWS Snowball

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Snowball and your AWS solutions. You should collect monitoring data so that you can more easily debug a multi-point failure if one occurs. AWS provides several tools for monitoring your AWS Snowball resources and responding to potential incidents:

AWS CloudTrail Logs

CloudTrail provides a record of actions taken by a user, role, or an AWS service in the AWS Snowball Job Management API or when using the AWS Console. Using the information collected by CloudTrail, you can determine the API request that was made to AWS Snowball service, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging AWS Snowball Edge API Calls with AWS CloudTrail.

Compliance Validation for AWS Snowball

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance on Amazon Web Services – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

> **Note**
>
> Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.

- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure Security in AWS Snowball

As a managed service, AWS Snow Family is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access AWS Snow Family through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Data Validation with Snowball Edge Jobs

Following, you'll find information about how AWS Snowball Edge validates data transfers, and the manual steps you can take to help ensure data integrity during and after a job.

Topics

Checksum Validation of Transferred Data

When you copy a file from a local data source using the Amazon S3 interface to the Snowball Edge, a number of checksums are created. These checksums are used to automatically validate data as it's transferred.

At a high level, these checksums are created for each file (or for parts of large files). For the Snowball Edge, these checksums are visible when you run the following AWS CLI command against a bucket on the device. The checksums are used to validate the integrity of your data throughout the transfers and help ensure that your data is copied correctly.

```
aws s3api list-objects --bucket bucket-name --endpoint http://ip:8080 --profile edge-
profile
```

When these checksums don't match, the associated data is not imported into Amazon S3.

Local Inventory Creation During Snowball Transfer

Create a local inventory of files copied to the Snowball when using the Amazon S3 adapter or CLI. The content of the local inventory can be used to compare with what is on the local storage or server.

For example,

```
aws s3 cp folder/ s3://bucket --recursive > inventory.txt
```

Common Validation Errors

When a validation error occurs, the corresponding data (a file or a part of a large file) is not written to the destination. The following are common causes for validation errors:

- Trying to copy symbolic links.
- Trying to copy files that are actively being modified. The attempt fails checksum validation and is marked as a failed transfer.
- Trying to copy files that are larger than 5 TB in size.
- Trying to copy part sizes that are larger than 2 GiB in size.
- Trying to copy files to a Snowball Edge device that is already at full data storage capacity.
- Trying to copy files to a Snowball Edge device that doesn't follow the object key naming guidelines for Amazon S3.

When any one of these validation errors occurs, it is logged. You can take steps to manually identify what files failed validation and why. For information, see Manual Data Validation for Snowball Edge After Import into Amazon S3.

Manual Data Validation for Snowball Edge After Import into Amazon S3

After an import job has completed, you have several options to manually validate the data in Amazon S3, as described following.

Check job completion report and associated logs

Whenever data is imported into or exported out of Amazon S3, you get a downloadable PDF job report. For import jobs, this report becomes available at the end of the import process. For more information, see Getting your job completion report and logs on the console.

S3 inventory

If you transferred a huge amount of data into Amazon S3 in multiple jobs, going through each job completion report might not be an efficient use of time. Instead, you can get an inventory of all the objects in one or more Amazon S3 buckets. Amazon S3 inventory provides a comma-separated values (CSV) file showing your objects and their corresponding metadata on a daily or weekly basis.

This file covers objects for an Amazon S3 bucket or a shared prefix (that is, objects that have names that begin with a common string).

When you have the inventory of the Amazon S3 buckets that you've imported data into, you can easily compare it against the files that you transferred on your source data location. In this way, you can quickly identify what files where not transferred.

Use the Amazon S3 sync command

If your workstation can connect to the internet, you can do a final validation of all your transferred files by running the AWS CLI command `aws s3 sync`. This command syncs directories and S3 prefixes. This command recursively copies new and updated files from the source directory to the destination. For more information, see sync in the *AWS CLI Command Reference*.

> **Important**
>
> If you specify your local storage as the destination for this command, make sure that you have a backup of the files that you sync against. These files are overwritten by the contents in the specified Amazon S3 source.

Notifications for Snow Family devices

How Snow uses Amazon SNS

The Snow service is designed to take advantage of the robust notifications delivered by Amazon Simple Notification Service (Amazon SNS). While creating a job to order a Snow device, you can provide email addresses to receive notifications for your job status changes. When you do this, you choose an existing SNS topic or create a new one. If the SNS topic is encrypted, you need to enable customer-managed KMS encryption for the topic and set up customer-managed KMS key policy. See Choose your notification preferences.

After you create your job, every email address that you specified to get Amazon SNS notifications receives an email message from AWS notifications asking for confirmation to the topic subscription. A user of the email account must confirm the subscription by choosing **Confirm subscription**. The Amazon SNS notification emails are tailored for each job status, and include a link to the AWS Snow Family Management Console.

You can also configure Amazon SNS to send text messages for status change notifications from the Amazon SNS console. For more information, see Mobile text messaging (SMS) in the *Amazon Simple Notification Service Developer Guide*.

Encrypting SNS topics for Snow job status changes

Enable customer-managed KMS encryption for the SNS topic for Snow job status change notifications. SNS topics encrypted with AWS-managed encryption cannot receive Snow job status changes because the Snow import IAM role does not have access to the AWS-managed KMS key to perform `Decrypt` and `GenerateDataKey` actions. Additionally, policies of AWS-managed KMS keys cannot be edited.

To enable server-side encryption for an SNS topic using the Amazon SNS management console

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. In the navigation pane, choose **Topics**.
3. In the Topics page, choose the topic used for job status change notifications, then choose **Edit**.
4. Expand the **Encryption** section and do the following:

a. Choose **Enable encryption**.

b. Specify the AWS KMS key. See

c. For each KMS type, the description, account, and KMS ARN are displayed.

5. To use a custom key from your AWS account, choose the **AWS KMS key** field and then choose the custom KMS kms from the list. For instructions on creating custom KMS, see Creating keys in the AWS Key Management Service Developer Guide.

   To use a custom KMS ARN from your AWS account or from another AWS account, enter the KMS key ARN in the **AWS KMS key** field.

6. Choose **Save changes**. Server side encryption is enabled for your topic and the topic page is displayed.

Setting up a customer-managed KMS key policy

After enabling encryption for SNS topics that will receive notifications for Snow job status changes, update the KMS policy for the SNS topic encryption and allow the Snow service principal `"importexport.amazonaws.com"` for `"mks:Decrypt"` and `"mks:GenerateDataKey*"` actions.

To allow the import export service role in the KMS key policy

1. Sign in to the AWS Management Console and open the AWS Key Management Service (AWS KMS) console at https://console.aws.amazon.com/kms.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. At the top-right corner of the console, change the AWS Region of the console to the same region as the Snow device was ordered from.
4. In the navigation pane, choose **Customer managed keys**.
5. IN the list of KMS keys, choose the alias or key ID of the KMS key to update.
6. Choose the **Key policy** tab, in the key policy statements, you can see the principals that have been given access to the KMS key by the key policy, and you can see the actions they can perform.
7. For the Snow service principal `"importexport.amazonaws.com"`, add the following policy statement for `"kms:Decrypt"` and `"kms:GenerateDataKey*"` actions:

```
  {

    "Effect": "Allow",
    "Principal": {
    "Service": "service.amazonaws.com"
  },
  "Action": [
  "kms:Decrypt",
  "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
    "ArnLike": {
    "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-type/
customer-resource-id"
  },
  "StringEquals": {
  "kms:EncryptionContext:aws:sns:topicArn": "arn:aws:sns:your_region:customer-
account-id:your_sns_topic_name"
  }
  }
  }
```

8. Choose **Save Changes** to apply the changes and exit the policy editor.

SNS notification examples

Amazon SNS notifications produce the following email messages when your job status changes. These messages are examples of the `Email-JSON` SNS topic protocol.

Job status	SNS notification JSON
Job created	`{ "Type" : "Notification", "MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162", "TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",`

Job status	SNS notification JSON
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) has been created. More info - https://console.aws.amazon.com/importexport", "Timestamp" : "2023-02-23T00:27:58.831Z", "SignatureVersion" : "1", "Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==", "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem", "UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297" }

Job status	SNS notification JSON

Preparing appliance	{
	"Type" : "Notification",
	"MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
	"TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) is being prepared. More info - https://console.aws.amazon.com/importexport",
	"Timestamp" : "2023-02-23T00:27:58.831Z",
	"SignatureVersion" : "1",
	"Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==",
	"SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
	"UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
	}

Job status	SNS notification JSON

Exporting	{
	"Type" : "Notification",
	"MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
	"TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) is being Exported. More info - https://console.aws.amazon.com/importexport",
	"Timestamp" : "2023-02-23T00:27:58.831Z",
	"SignatureVersion" : "1",
	"Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==",
	"SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
	"UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
	}

Job status	SNS notification JSON

In transit to you	{
	"Type" : "Notification",
	"MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
	"TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) is in transit to you. More info - https://console.aws.amazon.com/importexport",
	"Timestamp" : "2023-02-23T00:27:58.831Z",
	"SignatureVersion" : "1",
	"Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==",
	"SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
	"UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
	}

Job status	SNS notification JSON

Delivered to you	{
	"Type" : "Notification",
	"MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
	"TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) was delivered to you. More info - https://console.aws.amazon.com/importexport",
	"Timestamp" : "2023-02-23T00:27:58.831Z",
	"SignatureVersion" : "1",
	"Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==",
	"SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
	"UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
	}

Job status	SNS notification JSON

In transit to AWS	{
	"Type" : "Notification",
	"MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
	"TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) is in transit to AWS. More info - https://console.aws.amazon.com/importexport",
	"Timestamp" : "2023-02-23T00:27:58.831Z",
	"SignatureVersion" : "1",
	"Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==",
	"SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
	"UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
	}

Job status	SNS notification JSON

At sorting facility	{
	"Type" : "Notification",
	"MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
	"TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) is at AWS sorting facility. More info - https://console.aws.amazon.com/impor texport",
	"Timestamp" : "2023-02-23T00:27:58.831Z",
	"SignatureVersion" : "1",
	"Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==",
	"SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
	"UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
	}

Job status	SNS notification JSON

At AWS	{
	"Type" : "Notification",
	"MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
	"TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) is at AWS. More info - https://console.aws.amazon.com/importexport",
	"Timestamp" : "2023-02-23T00:27:58.831Z",
	"SignatureVersion" : "1",
	"Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==",
	"SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
	"UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
	}

Job status	SNS notification JSON

Importing	{
	"Type" : "Notification",
	"MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
	"TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) is being imported. More info - https://console.aws.amazon.com/importexport",
	"Timestamp" : "2023-02-23T00:27:58.831Z",
	"SignatureVersion" : "1",
	"Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==",
	"SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
	"UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
	}

Job status	SNS notification JSON

Completed	{
	"Type" : "Notification",
	"MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
	"TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) complete.\nThanks for using AWS Snow Family.\nCan you take a quick survey on your experience? Survey here: http://bit.ly/1pLQJMY. More info - https://console.aws.amazon.com/importexport",
	"Timestamp" : "2023-02-23T00:27:58.831Z",
	"SignatureVersion" : "1",
	"Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==",
	"SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
	"UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
	}

Job status	SNS notification JSON

Job status	SNS notification JSON

Cancelled	{
	"Type" : "Notification",
	"MessageId" : "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
	"TopicArn" : "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
	"Message" : "Your job Job-name (JID8bca334a-6c2f-4cd0-97e2-3f5a4dc9bd6d) was canceled. More info - https://console.aws.amazon.com/importexport",
	"Timestamp" : "2023-02-23T00:27:58.831Z",
	"SignatureVersion" : "1",
	"Signature" : "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/LsDr0Iak4ZDpg8dXg==",
	"SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
	"UnsubscribeURL" : "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
	}

Logging AWS Snowball Edge API Calls with AWS CloudTrail

The AWS Snowball or Snow Family service integrates with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or service. CloudTrail captures all API calls for AWS Snow Family service. The calls captured include calls from the AWS Snowball Family console and code calls to the AWS Snowball Family Job Management API. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Snowball Family API calls. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request made with AWS Snowball Family API, the IP address of the request made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

AWS Snowball Edge Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS Snowball Edge, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History in the *AWS CloudTrail User Guide*.

For an ongoing record of events in your AWS account, including events for AWS Snowball Edge, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all AWS Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following topics in the *AWS CloudTrail User Guide*:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All job management actions are documented in the AWS Snowball API Reference and are logged by CloudTrail with the following exceptions:

- The CreateAddress operation is not logged to protect customer sensitive information.
- All read-only API calls (for API operations beginning with the prefix of `Get`, `Describe`, or `List`) don't record response elements.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM user) credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element in the *AWS CloudTrail User Guide*.

Understanding Log File Entries for AWS Snowball Edge

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the DescribeJob operation.

```
  {"Records": [
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "Root",
        "principalId": "111122223333",
        "arn": "arn:aws:iam::111122223333:root",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {"attributes": {
```

```
                "mfaAuthenticated": "false",
                "creationDate": "2019-01-22T21:58:38Z"
            }},
            "invokedBy": "signin.amazonaws.com"
        },
        "eventTime": "2019-01-22T22:02:21Z",
        "eventSource": "snowball.amazonaws.com",
        "eventName": "DescribeJob",
        "awsRegion": "eu-west-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "signin.amazonaws.com",
        "requestParameters": {"jobId": "JIDa1b2c3d4-0123-abcd-1234-0123456789ab"},
        "responseElements": null,
        "requestID": "12345678-abcd-1234-abcd-ab0123456789",
        "eventID": "33c7ff7c-3efa-4d81-801e-7489fe6fff62",
        "eventType": "AwsApiCall",
        "recipientAccountId": "444455556666"
    }
]}
```

AWS Snowball Edge Quotas

Following, you can find information about limitations on using the AWS Snowball Edge device.

> **Important**
>
> When you transfer data into Amazon Simple Storage Service (Amazon S3) using a Snowball Edge, keep in mind that individual Amazon S3 objects can range in size from a minimum of 0 bytes to a maximum of 5 terabytes (TB).

Region Availability for AWS Snowball Edge

The following table highlights the regions where AWS Snowball Edge is available.

Region	Snowball Edge availability
US East (Ohio)	✓
US East (N. Virginia)	✓
US West (N. California)	✓
US West (Oregon)	✓
AWS GovCloud (US-East)	✓
AWS GovCloud (US-West)	✓
Canada (Central)	✓
Asia Pacific (Jakarta)	✓
Asia Pacific (Mumbai)	✓
Asia Pacific (Osaka)	✓
Asia Pacific (Seoul)	✓
Asia Pacific (Singapore)	✓

Region	Snowball Edge availability
Asia Pacific (Sydney)	✓
Asia Pacific (Tokyo)	✓
Europe (Frankfurt)	✓
Europe (Ireland)	✓
Europe (London)	✓
Europe (Milan)	✓
Europe (Paris)	✓
Europe (Stockholm)	✓
Middle East (UAE)	✓
South America (São Paulo)	✓

For information about supported AWS Regions and endpoints, see AWS Snow Family endpoints and quotas in the AWS General Reference

Limitations for AWS Snowball Edge Jobs

The following limitations exist for creating AWS Snowball Edge device jobs:

- For security purposes, jobs using an AWS Snowball Edge device must be completed within 360 days of being prepared. If you need to keep one or more devices for longer than 360 days, see Updating the SSL certificate. Otherwise, after 360 days, the device becomes locked, can no longer be accessed, and must be returned. If the AWS Snowball Edge device becomes locked during an import job, we can still transfer the existing data on the device into Amazon S3.
- AWS Snowball Edge supports server-side encryption with Amazon S3-managed encryption keys (SSE-S3) and server-side encryption with AWS Key Management Service managed keys (SSE-KMS). Amazon S3 compatible storage on Snow Family devices supports SSS-C for local compute and storage jobs. For more information, see Protecting data using server-side encryption in the *Amazon Simple Storage Service User Guide*.

- If you're using an AWS Snowball Edge device to import data, and you need to transfer more data than will fit on a single Snowball Edge device, create additional jobs. Each export job can use multiple Snowball Edge devices.
- The default service limit for the number of Snowball Edge devices you can have at one time is 1 per account, per AWS Region. If you want to increase your service limit or create a cluster job, contact AWS Support.
- Metadata for objects transferred to a device is not persisted. The only metadata that remains the same is `filename` and `filesize`. All other metadata is set as in the following example:

```
-rw-rw-r-- 1 root root [filesize] Dec 31 1969 [path/filename]
```

Rate Limits on AWS Snowball Edge

The Rate Limiter is used to control the rate of requests in a server cluster environment.

Amazon Snow S3 Adapter Connection Limit

The maximum connection limit is 1000 for Snowball Edge on Amazon S3. Any connections beyond 1000 are dropped.

Limitations on Transferring On-Premises Data with a Snowball Edge Device

The following limitations exist for transferring data to or from an AWS Snowball Edge device on-premises:

- Files must be in a static state while being written. Files that are modified while being transferred are not imported into Amazon S3.
- Jumbo frames are not supported—that is, Ethernet frames with more than 1500 bytes of payload.
- When selecting what data to export, keep in mind that objects with trailing slashes in their names (/ or \) are not transferred. Before exporting any objects with trailing slashes, update their names to remove the slash.
- When using multipart data transfer, the maximum part size is 2 GiB.

Limitations on Shipping a Snowball Edge

The following limitations exist for shipping an AWS Snowball Edge device:

- AWS will not ship a Snowball Edge device to a post office box.
- AWS will not ship a Snowball Edge device between non-US Regions—for example, from EU (Ireland) to EU (Frankfurt), or to Asia Pacific (Sydney).
- Moving a Snowball Edge device to an address outside of the country specified when the job was created is not allowed and is a violation of the AWS service terms.

For more information about shipping, see Shipping considerations for Snow Family devices.

Limitations on Processing Your Returned Snowball Edge for Import

To import your data into AWS, the device must meet the following requirements:

- The AWS Snowball Edge device must not be compromised. Except for opening the three doors on the front, back, and top, or to add and replace the optional air filter, don't open the AWS Snowball Edge device for any reason.
- The device must not be physically damaged. You can prevent damage by closing the three doors on the Snowball Edge device until the latches make an audible clicking sound.
- The E Ink display on the Snowball Edge device must be visible. It must also show the return label that was automatically generated when you finished transferring your data onto the AWS Snowball Edge device.

> **Note**
>
> All Snowball Edge devices returned that don't meet these requirements are erased without having any work performed on them.

Troubleshooting AWS Snowball Edge

Keep the following general guidelines in mind when troubleshooting.

- Objects in Amazon S3 have a maximum file size of 5 TB.
- Objects transferred onto an AWS Snowball Edge device have a maximum key length of 933 bytes. Key names that include characters that take up more than 1 byte each still have a maximum key length of 933 bytes. When determining key length, you include the file or object name and also its path or prefixes. Thus, files with short file names within a heavily nested path can have keys longer than 933 bytes. The bucket name is not factored into the path when determining the key length. Some examples follow.

Object name	Bucket name	Path plus bucket name	Key Length
`sunflower-1.jpg`	`pictures`	`sunflower-1.jpg`	15 characters
`receipts.csv`	`MyTaxInfo`	`/Users/Eric/Documents/2016/January/`	47 characters
`bhv.1`	`$7$zWwwXKQj$gLAOoZCj$r8p`	`/.VfV/FqGC3QN$7BXys3KHYePfuIOMNjY83dVx ugPYlxVg/evpcQEJLT/rSwZc$MlVVf/$hwefVISRqwepB$/BiiD/PPF$twRAjrD`	135 characters

Object name	Bucket name	Path plus bucket name	Key Length
		/fIMp/0NY	

- For security purposes, jobs using an AWS Snowball Edge device must be completed within 360 days of being prepared. If you need to keep one or more devices for longer than 360 days, see Updating the SSL certificate. Otherwise, after 360 days,the device becomes locked, can no longer be accessed, and must be returned. If the AWS Snowball Edge device becomes locked during an import job, we can still transfer the existing data on the device into Amazon S3.
- If you encounter unexpected errors using an AWS Snowball Edge device, we want to hear about it. Copy the relevant logs and include them along with a brief description of the issues that you encountered in a message to AWS Support. For more information about logs, see Commands for the Snowball Edge Client.

Topics

How to identify your device

Use the `describe-device` command to find the device type, then look up the returned value of `DeviceType` in the table below to determine the configuration.

```
snowballEdge describe-device
```

Example of `describe-device` output

```
{
"DeviceId" : "JID-206843500001-35-92-20-211-23-06-02-18-24",
"UnlockStatus" : {
  "State" : "UNLOCKED"
},
"ActiveNetworkInterface" : {
  "IpAddress" : "127.0.0.1"
},
"PhysicalNetworkInterfaces" : [ {
  "PhysicalNetworkInterfaceId" : "s.ni-8d0ef958ec860ac7c",
  "PhysicalConnectorType" : "RJ45",
  "IpAddressAssignment" : "DHCP",
  "IpAddress" : "172.31.25.194",
  "Netmask" : "255.255.240.0",
  "DefaultGateway" : "172.31.16.1",
  "MacAddress" : "02:38:30:12:a3:7b"
} ],
"DeviceCapacities" : [ {
  "Name" : "HDD Storage",
  "Unit" : "Byte",
  "Total" : 39736350227824,
  "Available" : 985536581632
}, {
  "Name" : "SSD Storage",
  "Unit" : "Byte",
  "Total" : 6979321856000,
  "Available" : 6979321856000
}, {
  "Name" : "vCPU",
  "Unit" : "Number",
  "Total" : 52,
  "Available" : 52
}, {
  "Name" : "Memory",
  "Unit" : "Byte",
  "Total" : 223338299392,
```

```
    "Available" : 223338299392
  }, {
    "Name" : "GPU",
    "Unit" : "Number",
    "Total" : 0,
    "Available" : 0
  } ],
  "DeviceType" : "EDGE_C"
}
```

`DeviceType` and Snow Family device configurations

`DeviceType` value	Device configuration
EDGE	Snowball Edge storage-optimized (with EC2 compute functionality)
EDGE_C	Snowball Edge compute-optimized with AMD EPYC Gen1 and HDD
EDGE_CG	Snowball Edge compute-optimized with AMD EPYC Gen1, HDD, and GPU
EDGE_S	Snowball Edge storage-optimized
V3_5C	Snowball Edge compute-optimized with AMD EPYC Gen2 and NVME
V3_5S	Snowball Edge storage-optimized 210 TB

For more information about Snowball Edge device configurations, see AWS Snowball Edge device differences.

Troubleshooting boot-up problems

The following information can help you troubleshoot certain issues you might have with booting-up your Snow Family devices.

- Allow 10 minutes for a device to boot up. Avoid moving or using the device during this time.

- Ensure both ends of the cable supplying power are connected securely.
- Replace the cable supplying power with another cable that you know is good.
- Connect the cable supplying power to another source of power that you know is good.

Troubleshooting problems with the LCD display during boot-up

Sometimes, after powering on a Snowball Edge device, the LCD display may encounter a problem.

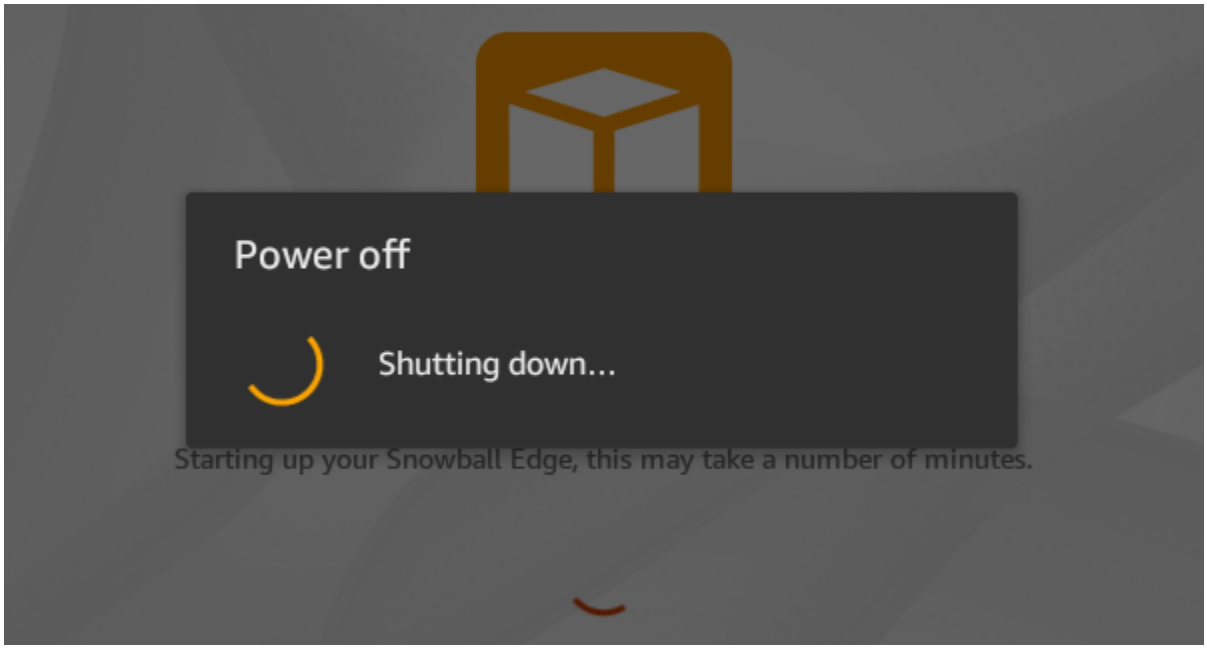
- The LCD screen is black and does not display an image after you connect the Snowball Edge device to power and press the power button above the LCD screen.
- The LCD screen does not advance past the **Setting you your Snowball Edge, this may take a number of minutes.** message and the network configuration screen does not appear.

Action to take when the LCD screen is black after pressing the power button

1. Ensure the Snowball Edge device is connected to a power source and the power source is providing power.
2. Leave the device connected to the power source for 1 to 2 hours. Ensure the doors on the front and back of the device are open.
3. Return to the device and the LCD screen will be ready to use.

Action to take when the Snowball Edge does not advance to the network configuration screen

1. Let the screen stay on the **Setting you your Snowball Edge, this may take a number of minutes.** message for 10 minutes.
2. On the screen, choose the **Restart display** button. The **Shutting down...** message will appear, then the **Setting you your Snowball Edge, this may take a number of minutes.** message will appear and the device will start normally.



If the LCD screen does not advance past the **Setting you your Snowball Edge, this may take a number of minutes.** message after using the **Restart display** button, use the following procedure.

Action to take

1. Above the LCD screen, press the power button to power off the device.
2. Disconnect all cables from the device.
3. Leave the device powered off and disconnected for 20 minutes.
4. Connect the power and network cables.
5. Above the LCD screen, press the power button to power on the device.

If the problem persists, contact AWS Support to return the device and receive a new Snowball Edge device.

Troubleshooting connection problems

The following information can help you troubleshoot certain issues that you might have with connecting to your Snowball Edge:

- Routers and switches that work at a rate of 100 megabytes per second don't work with a Snowball Edge. We recommend that you use a switch that works at a rate of 1 GB per second (or faster).
- If you experience odd connection errors with the device, power off the Snowball Edge, unplug all the cables, and leave it for 10 minutes. After 10 minutes have passed, restart the device, and try again.
- Ensure that no antivirus software or firewalls block the Snowball Edge device's network connection.
- Be aware that the file interface and Amazon S3 interface have different IP addresses.

For more advanced connection troubleshooting, you can take the following steps:

- If you can't communicate with the Snowball Edge, ping the IP address of the device. If the ping returns `no connect`, confirm the IP address for the device and confirm your local network configuration.
- If the IP address is correct and the lights on the back of the device are flashing, use telnet to test the device on ports 22, 9091, and 8080. Testing port 22 determines whether the Snowball Edge is working correctly. Testing port 9091 determines whether the AWS CLI can be used to send commands to the device. Testing port 8080 helps ensure that the device can write to the Amazon S3 buckets on it with S3 adapter only. If you can connect on port 22 but not on port 8080, first power off the Snowball Edge and then unplug all the cables. Leave the device for 10 minutes, and then reconnect it and start again.

Troubleshooting `unlock-device` command problems

If the `unlock-device` command returns `connection refused`, you may have mistyped the command syntax or the configuration of your computer or network may be preventing the command from reaching the Snow device. Take these actions to resolve the situation:

1. Ensure the command was entered correctly.
   a. Use the LCD screen on the device to verify the IP addressed used in the command is correct.

b. Ensure that the path to the manifest file used in the command is correct, including the file name.

c. Use the AWS Snow Family Management Console to verify the unlock code used in the command is correct.

2. Ensure the computer you are using is on the same network and subnet as the Snow device.

3. Ensure the computer you are using and the network are configured to allow access to the Snow device. Use the `ping` command for your operating system to determine if the computer can reach the Snow device over the network. Check the configurations of antivirus software, firewall configuration, virtual private network (VPN), or other configurations of your computer and network.

Troubleshooting manifest file problems

Each job has a specific manifest file associated with it. If you create multiple jobs, track which manifest is for which job.

If you lose a manifest file or if a manifest file is corrupted, you can download the manifest file for a specific job again. You do so using the console, AWS CLI, or one of the AWS APIs.

Troubleshooting credentials problems

Use the following topics to help you resolve credentials issues with the Snowball Edge.

Unable to locate AWS CLI credentials

If you're communicating with the AWS Snowball Edge device through the Amazon S3 interface using the AWS CLI, you might encounter an error message that says Unable to locate credentials. You can configure credentials by running "aws configure".

Action to take

Configure the AWS credentials that the AWS CLI uses to run commands for you. For more information, see Configuring the AWS CLI in the *AWS Command Line Interface User Guide*.

Error message: Check Your Secret Access Key and Signing

When using the Amazon S3 interface to transfer data to a Snowball Edge, you might encounter the following error message.

```
An error occurred (SignatureDoesNotMatch) when calling the CreateMultipartUpload
 operation: The request signature we calculated does not match the signature you
 provided.
Check your AWS secret access key and signing method. For more details go to:
http://docs.aws.amazon.com/AmazonS3/latest/dev/
RESTAuthentication.html#ConstructingTheAuthenticationHeader
```

Action to take

Get your credentials from the Snowball Edge client. For more information, see Getting Credentials.

Troubleshooting NFS interface problems

The Snow Family device may indicate the status of the NFS interface is `DEACTIVATED`. This might occur if the Snow Family device was powered off without first stopping the NFS interface.

Action to take

To correct the problem, stop and restart the NFS service using the following steps.

1. Use the `describe-service` command to determine the status of the service:

   ```
   snowballEdge describe-service --service-id nfs
   ```

   The command returns the following.

   ```
   {
     "ServiceId" : "nfs",
     "Status" : {
     "State" : "DEACTIVATED"
     }
   }
   ```

2. Use the `stop-service` command to stop the NFS service.

   ```
   snowballEdge stop-service --service-id nfs
   ```

3. Use the `start-service` command to start the NFS service. For more information, see Starting the NFS service on the Snow Family device.

```
snowballEdge start-service  --virtual-network-interface-arns vni-arn --service-id
 nfs  --service-configuration  AllowedHosts=0.0.0.0/0
```

4. Use the `describe-service` command to make sure the service is running.

```
snowballEdge describe-service --service-id nfs
```

   If the value of the `State` name is `ACTIVE`, the NFS interface service is active.

```
{
 "ServiceId" : "nfs",
 "Status" : {
 "State" : "ACTIVE"
 },
 "Endpoints" : [ {
 "Protocol" : "nfs",
 "Port" : 2049,
 "Host" : "192.0.2.0"
 } ],
 "ServiceConfiguration" : {
 "AllowedHosts" : [ "10.24.34.0/23", "198.51.100.0/24" ]
 }
}
```

Troubleshooting data transfer problems

If you encounter performance issues while transferring data to or from a Snowball Edge, see Performance for recommendations and guidance on improving transfer performance. The following can help you troubleshoot issues that you might have with your data transfer to or from a Snowball Edge:

- You can't transfer data into the root directory of the Snowball Edge. If you have trouble transferring data into the device, make sure that you're transferring data into a subdirectory. The

top-level subdirectories have the names of the Amazon S3 buckets that you included in the job. Put your data in those subdirectories.

- If you're using Linux and you can't upload files with UTF-8 characters to an AWS Snowball Edge device, it might be because your Linux server doesn't recognize UTF-8 character encoding. You can correct this issue by installing the `locales` package on your Linux server and configuring it to use one of the UTF-8 locales like `en_US.UTF-8`. You can configure the `locales` package by exporting the environment variable `LC_ALL`, for example: `export LC_ALL=en_US.UTF-8`
- When you use the Amazon S3 interface with the AWS CLI, you can work with files or folders with spaces in their names, such as `my photo.jpg` or `My Documents`. However, make sure that you handle the spaces properly. For more information, see Specifying parameter values for the AWS CLI in the *AWS Command Line Interface User Guide*.

Troubleshooting AWS CLI problems

Use the following topics to help you resolve problems when working with an AWS Snowball Edge device and the AWS CLI.

AWS CLI error message: "Profile Cannot Be Null"

When working with the AWS CLI, you might encounter an error message that says Profile cannot be null. You can encounter this error if the AWS CLI hasn't been installed or an AWS CLI profile hasn't been configured.

Action to take

Ensure that you have downloaded and configured the AWS CLI on your workstation. For more information, see Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) in the *AWS Command Line Interface User Guide.*

Null pointer error when transferring data with the AWS CLI

When using the AWS CLI to transfer data, you might encounter a null pointer error. This error can occur in the following conditions:

- If the specified file name is misspelled, for example `flowwer.png` or `flower.npg` instead of `flower.png`
- If the specified path is incorrect, for example `C:\Doccuments\flower.png` instead of `C:\Documents\flower.png`

- If the file is corrupted

Action to take

Confirm that your file name and path are correct, and try again. If you continue to experience this issue, confirm that the file has not been corrupted, abandon the transfer, or attempt repairs to the file.

Troubleshooting import job problems

Sometimes files fail to import into Amazon S3. If the following issue occurs, try the actions specified to resolve your issue. If a file fails import, you might need to try importing it again. Importing it again might require a new job for Snowball Edge.

Files failed import into Amazon S3 due to invalid characters in object names

This problem occurs if a file or folder name has characters that aren't supported by Amazon S3. Amazon S3 has rules about what characters can be in object names. For more information, see Creating object key names in Amazon S3 User Guide.

Action to take

If you encounter this issue, you see the list of files and folders that failed import in your job completion report.

In some cases, the list is prohibitively large, or the files in the list are too large to transfer over the internet. In these cases, you should create a new Snowball import job, change the file and folder names to comply with Amazon S3 rules, and transfer the files again.

If the files are small and there isn't a large number of them, you can copy them to Amazon S3 through the AWS CLI or the AWS Management Console. For more information, see How do I upload files and folders to an S3 bucket? in the *Amazon Simple Storage Service User Guide.*

Troubleshooting export job problems

Sometimes files fail to export into your workstation. If the following issue occurs, try the actions specified to resolve your issue. If a file fails export, you might need to try exporting it again. Exporting it again might require a new job for Snowball Edge.

Files failed export to a Microsoft Windows Server

A file can fail export to a Microsoft Windows Server if it or a related folder is named in a format not supported by Windows. For example, if your file or folder name has a colon (`:`) in it, the export fails because Windows doesn't allow that character in file or folder names.

Action to take

1. Make a list of the names that are causing the error. You can find the names of the files and folders that failed export in your logs. For more information, see AWS Snowball Edge Logs.
2. Change the names of the objects in Amazon S3 that are causing the issue to remove or replace the unsupported characters.
3. If the list of names is prohibitively large, or if the files in the list are too large to transfer over the internet, create a new export job specifically for those objects.

   If the files are small and there isn't a large number of them, copy the renamed objects from Amazon S3 through the AWS CLI or the AWS Management Console. For more information, see How do I download an object from an S3 bucket? in the *Amazon Simple Storage Service User Guide.*

API Reference

In addition to using the console, you can use the AWS Snowball API to programmatically configure and manage AWS Snowball Edge device and its resources. This section describes the device operations and data types and contains the API Reference documentation for AWS Snowball Edge device.

Topics

Document History

- **API version: 1.0**
- **Latest documentation update:** March 1, 2024

The following table describes important changes to the *AWS Snowball Edge Developer Guide* after July 2018. For notifications about documentation updates, you can subscribe to the RSS feed.

Change	Description	Date
File interface deprecated	The file interface is no longer available for data transfer.	March 1, 2024
Amazon S3 compatible storage on Snow Family devices available on Snowball Edge storage-optimized 210 TB devices	Amazon S3 compatible storage on Snow Family devices is available for S3 storage on Snowball Edge storage-optimized 210 TB devices. For more informati on, see Using Amazon S3 compatible storage on Snow Family devices.	February 26, 2024
Include custom AMIs when ordering devices	Custom Amazon Machine Images can now be preloaded while ordering AWS Snow Family jobs. For more information, see Adding an AMI from AWS Marketplace.	November 15, 2023
Amazon S3 compatible storage on Snow Family devices generally available	Amazon S3 compatible storage on Snow Family devices is supported on Snowball Edge compute-o ptimized devices. For more information, see Amazon S3	April 20, 2023

	compatible storage on Snow Family devices.	
New AWS Region supported	AWS Snowball is now supported in the Middle East (UAE) Region. For informati on about endpoints for this region, see Snowball Edge Endpoints and Quotas in the *AWS General Reference*. For information on shipping, see Shipping Considerations for Snowball Edge.	March 6, 2023
New AWS Region supported	AWS Snowball is now supported in the Asia Pacific (Jakarta) Region. For information about endpoints for this region, see Snowball Edge Endpoints and Quotas in the *AWS General Reference* . For information on shipping, see Shipping Considerations for Snowball Edge.	September 7, 2022
Large Data Migration for Snowball Edge	Snowball Edge now supports automating a large data migration plan. For more information see Large Data Migration (manual steps) and Create a Large Data Migration Plan to initiate automation if desired.	April 27, 2022

Introducing AWS Snow Device Management	Snow Device Management allows you to manage your Snowball Edge device and local AWS services remotely. All Snowball Edge devices support Snow Device Management, and it comes pre-installed on new devices in most AWS Regions where Snowball Edge is available. For more information, see Using AWS Snow Device Management to Manage Devices	April 27, 2022
NFS Configuration for Snowball Edge	Added NFS Configuration for Snowball Edge for Storage Optimized devices.	April 21, 2022
Rate Limits for Load Balancer	Snowball Edge now supports Rate Limits to distribute requests in a server cluster environment.	April 19, 2022
Support for Snowball Edge with Tape Gateway	You can now order a Snowball Edge device that is specially configured to host the Tape Gateway service. This combination of technologies facilitates secure offline tape data migration. For more information, see Using AWS Snowball Edge with Tape Gateway.	November 30, 2021

Support for Network Time Protocol (NTP) server configuration	Snowball Edge devices now support external Network Time Protocol (NTP) server configuration.	November 16, 2021
Support for NFS offline data transfer	Snowball Edge devices now support offline data transfer using NFS. For more informati on, see Using NFS for Offline Data Transfer.	August 4, 2021
New AWS Region supported	Snowball Edge devices are now available in the Africa (Cape Town) AWS Region. For more information, see Snowball Edge Endpoints and Quotas in the *AWS General Reference*. For information on shipping, see Shipping Considerations for Snowball Edge.	November 23, 2020
Support for importing your own image into your device	You can now import a snapshot of your image into your Snowball Edge device and register it as an Amazon EC2-compatible Amazon Machine Image (AMI). For more information, see Importing an Image into Your Device as an Amazon EC2 AMI .	November 9, 2020

New AWS Region supported	Snowball Edge devices are now available in the Europe (Milan) AWS Region. For more information, see Snowball Edge Endpoints and Quotas in the *AWS General Reference* . For information on shipping, see Shipping Considerations for Snowball Edge.	September 30, 2020
Content restructure	Created a Getting Started section that aligns with the AWS Snow Family Management Console workflow and updated other sections for clarity. For more information, see Getting Started with an AWS Snowball Edge.	September 17, 2020
Introducing AWS OpsHub for Snow Family	The Snow Family devices now offer a user-friendly tool, AWS OpsHub for Snow Family, that you can use to manage your devices and local AWS services. For more informati on, see Using AWS OpsHub for Snow Family to Manage Snowball Devices.	April 16, 2020

AWS Identity and Access Management (IAM) is now available locally on the AWS Snowball Edge device	You can now use AWS Identity and Access Management (IAM) to securely control access to AWS resources running on your AWS Snowball Edge device. For more information, see Using IAM Locally.	April 16, 2020
Introducing a new Snowball Edge Storage Optimized (for data transfer) device option	Snowball now adds a new storage optimized device based on the current compute-optimized and GPU devices. For more informati on, see Snowball Edge Device Options.	March 23, 2020
NFC tag validation support	Snowball Edge Compute Optimized devices (with or without the GPU) have NFC tags built into them. You can scan these tags with the AWS Snowball Edge Verification App, available on Android. For more information, see Validating NFC Tags.	December 13, 2018
Security groups are now available for compute instances	Security groups in Snowball Edge devices are similar to security groups in the AWS Cloud, with some subtle differences. For more information, see Security Groups in Snowball Edge Devices.	November 26, 2018

Introducing on-premises update	You can now update the software that makes a Snowball Edge device run in your local environment. Note that on-premises updates require an internet connectio n. For more information, see Updating an Snowball Edge.	November 26, 2018
Introducing new device options for Snowball Edge	Snowball Edge devices come in three options – storage optimized, compute optimized, and with GPU. For more information, see Snowball Edge Device Options.	November 15, 2018
New AWS Region supported	Snowball Edge devices are now available in the Asia Pacific (Mumbai). Note that compute instances and AWS Lambda powered by AWS IoT Greengrass are not supported in this region.	September 24, 2018
Introducing support for Amazon EC2-compatible compute instances on Snowball Edge devices	AWS Snowball now supports local jobs using Amazon EC2 compute instances running on Snowball Edge devices.	July 17, 2018
Improved troubleshooting content	The troubleshooting chapter has been updated and reorganized.	July 11, 2018

The following table describes documentation releases for the AWS Snowball Edge device before July 2018.

Change	Description	Date
New AWS Region supported	AWS Snowball is now supported in the Asia Pacific (Singapore) region. For more information on shipping in this AWS Region, see Shipping considerations for Snow Family devices.	April 3, 2018
Automatically extracted batches of small files are now supported	You can now batch many small files together into a larger archive, and specify that those batches are automatically extracted when the data is imported into Amazon S3. Batching small files together can significa ntly improve your transfer performance when moving data from your on-premises server to a Snowball Edge device. For more information, see Batching small files.	March 20, 2018
Major feature revision to the Snowball Edge client and cluster update	The new major feature revision for the Snowball Edge client includes performance improveme nts, profiles, and support for the cluster update. For more information, see Using the Snowball Edge Client. Clusters are now leaderless. All nodes can read and write data to the cluster. For more	February 5, 2018

Change	Description	Date
	information, see Clustering overview.	
New AWS Region supported	AWS Snowball is now supported in the Europe (Paris) region. For more information on shipping in this AWS Region, see Shipping considerations for Snow Family devices.	December 18, 2017
Improved AWS CLI support for the Amazon S3 adapter	You can now use the `s3 sync` command with the Amazon S3 adapter to sync data between a Snowball Edge and your local computer. For more informati on, see Supported AWS CLI commands for Amazon S3.	November 10, 2017
Updated file interface file size support	The file interface can now support files up to 150 GB in size.	October 4, 2017
New AWS Region supported	AWS Snowball Edge is now supported in the Asia Pacific (Tokyo) region, with region-specific shipping options. For more information, see Shipping considerations for Snow Family devices.	September 19, 2017

Change	Description	Date
New AWS Region supported	AWS Snowball Edge is now supported in the South America (São Paulo) region, with region-specific shipping options. For more informati on, see Shipping considera tions for Snow Family devices.	August 8, 2017
Updated AWS IoT Greengrass and Lambda functionality	Lambda functions running on AWS Snowball Edge devices can now be added, updated, removed, or replaced, once the devices are on-premises. In addition, AWS Snowball Edge devices can now be used as AWS IoT Greengrass core devices..	July 25, 2017
New AWS Region supported	AWS Snowball Edge is now supported in the Canada (Central) region, with region-specific shipping options. For more information, see Shipping considerations for Snow Family devices.	June 29, 2017
Updated file interface functionality	With the file interface, you can now choose the Network File System (NFS) clients that are allowed to access the file share on the Snowball Edge, in addition to accessing other support and troubleshooting features..	June 21, 2017

Change	Description	Date
Updated cluster functionality	Clusters can now be created in groups of 5–10 AWS Snowball Edge devices. For more information, see Clustering overview.	June 5, 2017
Documentation update	Documentation navigation has been updated for clarity and consistency, and a regional limitations section has been added. For more information, see Region Availability for AWS Snowball Edge.	May 8, 2017

Change	Description	Date
Updated compute informati on	The following updates have been made for AWS Lambda powered by AWS IoT Greengrass functions: • Event objects are now JSON objects like their cloud-bas ed counterparts. • When you choose a function for a job, you also choose a specific version of the function. Each version of a function now has a separate Amazon Resource Name (ARN). • To improve latency, functions are loaded in memory when a job is created. When creating a compute job, keep in mind that you have a total of 3.744 GB of memory available for all the functions. If you need more functions than the memory can support, you need to create more jobs.	December 6, 2016

Change	Description	Date
Introducing AWS AWS Snowball Edge	The AWS Snowball service now has two devices, the standard Snowball and the AWS Snowball Edge device. With the new Snowball Edge, you can now create local storage and compute jobs, harnessing the power of the AWS Cloud locally, without an internet connection.	November 30, 2016

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.