
AWS Prescriptive Guidance

Backup and recovery approaches on AWS



AWS Prescriptive Guidance: Backup and recovery approaches on AWS

Copyright © 2022 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Introduction	1
Why use AWS as a data-protection platform?	1
Targeted business outcomes	3
Choosing AWS services	4
Designing a backup and recovery solution	6
AWS Backup	7
Amazon S3 and Amazon S3 Glacier	9
Amazon S3	9
Standard S3 buckets	10
Maintain rollback history	10
Customized configuration files	10
Custom backup and restore	10
Amazon S3 Glacier	11
Using Amazon S3 Lifecycle object transition	11
Securing backup data	12
Backup and recovery for Amazon EC2 with EBS volumes	14
Amazon EC2 backup and recovery	15
Separate server volumes	15
AMIs or snapshots	16
Instance store volumes	16
Tagging and enforcing standards	17
Create EBS volume backups	17
Preparing an EBS volume	18
Creating snapshots from the console	19
Creating AMIs	19
Amazon Data Lifecycle Manager	20
AWS Backup	20
Multi-volume backups	20
Protecting backups	22
Archiving snapshots	22
Automating snapshot and AMI creation	22
Restoring from a snapshot or AMI	23
Restoring from a snapshot	23
Restoring from an AMI	24
Backup and recovery from on-premises infrastructure to AWS	26
File gateway	26
Volume gateway	27
Tape gateway	27
Backup and recovery of applications	29
Cloud-native AWS services	30
Amazon RDS	30
Using DNS CNAME	31
DynamoDB	31
Hybrid architectures	33
Moving centralized backup management solutions	33
Disaster recovery with AWS	35
On-premises DR to AWS	35
DR for cloud-native workloads	36
DR in a single Availability Zone	37
DR in a regional failure	37
Cleaning up backups	38
Backup and recovery FAQ	39
What backup schedule should I select?	39
Do I need to create backups in my development accounts?	39

Next steps	40
Additional resources	41
Document history	42

Backup and recovery approaches on AWS

Khurram Nizami, Amazon Web Services (AWS)

August 2022 ([document history](#) (p. 42))

This guide discusses how to implement backup and recovery approaches using Amazon Web Services (AWS) services for on-premises, cloud-native, and hybrid architectures. These approaches offer lower costs, higher scalability, and more durability to meet recovery time objective (RTO), recovery point objective (RPO), and compliance requirements.

This guide is intended for technical leaders who are responsible for protecting data in their corporate IT and cloud environments.

This guide covers different backup architectures (cloud-native applications, hybrid, and on-premises environments). It also covers associated Amazon Web Services (AWS) services that can be used to build scalable and reliable data-protection solutions for the non-immutable components of your architecture.

Another approach is to modernize your workloads to use immutable architectures, reducing the need for backup and recovery of components. AWS provides a number of services to implement immutable architectures and reduce the need for backup and recovery, including:

- Serverless with AWS Lambda
- Containers with Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and AWS Fargate
- Amazon Machine Images (AMIs) with Amazon Elastic Compute Cloud (Amazon EC2)

As the growth of enterprise data accelerates, the task of protecting it becomes more challenging. Questions about the durability and scalability of backup approaches are commonplace, including this one: How does the cloud help meet my backup and restore needs?

Why use AWS as a data-protection platform?

AWS is a secure, high-performance, flexible, money-saving, and easy-to-use cloud computing platform. AWS takes care of the undifferentiated heavy lifting required to create, implement, and manage scalable backup and recovery solutions.

There are many advantages to using AWS as part of your data protection strategy:

- **Durability:** Amazon Simple Storage Service (Amazon S3), Amazon S3 Glacier, and S3 Glacier Deep Archive are designed for 99.999999999 percent (11 nines) of durability. Both platforms offer reliable backup of data, with object replication across at least three geographically dispersed Availability Zones. Many AWS services use Amazon S3 for storage and export/import operations. For example, Amazon Elastic Block Store (Amazon EBS) uses Amazon S3 for snapshot storage.
- **Security:** AWS provides a number of options for access control and data encryption while in-transit and at-rest.
- **Global infrastructure:** AWS services are available around the globe, so you can back up and store data in the Region that meets your compliance and workload requirements.

- **Compliance:** AWS infrastructure is certified for compliance with the following standards, so you can easily fit the backup solution into your existing compliance regimen:
 - Service Organization Controls (SOC)
 - Statement on Standards for Attestation Engagements (SSAE) 16
 - International Organization for Standardization (ISO) 27001
 - Payment Card Industry Data Security Standard (PCI DSS)
 - Health Insurance Portability and Accountability Act (HIPAA)
 - SEC1
 - Federal Risk and Authorization Management Program (FedRAMP)
- **Scalability:** With AWS, you don't have to worry about capacity. As your needs change, you can scale your consumption up or down without administrative overhead.
- **Lower total cost of ownership (TCO):** The scale of AWS operations drives down service costs and helps lower the TCO of AWS services. AWS passes these cost savings on to customers through price drops.
- **Pay-as-you-go pricing:** Purchase AWS services as you need them and only for the period that you plan to use them. AWS pricing has no upfront fees, termination penalties, or long-term contracts.

Targeted business outcomes

The goal of this guide is to provide an overview of AWS services that you can use to support backup and recovery approaches for the following:

- On-premises architectures
- Cloud-native architectures
- Hybrid architectures
- AWS native services
- Disaster recovery (DR)

Best practices and considerations are covered along with an overview of services. This guide also provides you with the tradeoffs between using one approach over another for backup and recovery.

Choosing AWS services for data protection

AWS provides a number of storage and complementary services that can be used as part of your backup and recovery approach. These services can support both cloud-native and hybrid architectures. Different services are more effective for different use cases.

- [Amazon S3](#) and [Amazon S3 Glacier](#) and [S3 Glacier Deep Archive](#) are suited for both hybrid and cloud-native use cases. These services provide highly durable, general-purpose object storage solutions that are suitable for backing up individual files, servers, or an entire data center.
- [AWS Storage Gateway](#) is ideal for hybrid use cases. Storage Gateway uses the power of Amazon S3 for common on-premises backup and storage requirements. Your applications connect to the service through a virtual machine (VM) or hardware gateway appliance using the following standard storage protocols:
 - Network File System (NFS)
 - Server Message Block (SMB)
 - Internet Small Computer System Interface (iSCSI)

The gateway bridges these common on-premises protocols to AWS storage services such as the following:

- Amazon S3
- Amazon S3 Glacier
- S3 Glacier Deep Archive
- Amazon EBS

Storage Gateway makes it easier to provide elastic, high-performance storage for [files](#), [volumes](#), snapshots, and [virtual tapes](#) in AWS.

- [AWS Backup](#) is a fully managed backup service for centralizing and automating the backup of data across AWS services. Using AWS Backup, you can centrally configure backup policies and monitor backup activity for AWS resources, such as the following:
 - EBS volumes
 - EC2 instances (including Windows applications)
 - Amazon RDS and Amazon Aurora databases
 - DynamoDB tables
 - Amazon Neptune databases
 - Amazon DocumentDB (with MongoDB compatibility) databases
 - Amazon EFS file systems
 - Amazon FSx for Lustre file systems and Amazon FSx for Windows File Server file systems
 - VMware workloads on premises and in VMware Cloud on AWS
 - Storage Gateway volumes

The cost of AWS Backup is based on the storage that you consume, restore, and transfer in a month. For more information, see the <http://aws.amazon.com/backup/pricing/> AWS Backup pricing.

- [CloudEndure Disaster Recovery](#) continuously replicates your machines into a low-cost staging area in your target AWS account and preferred Region. The replication includes the operating system, system state configuration, databases, applications, and files. CloudEndure Disaster Recovery can be used for on-premises to cloud DR and cross-Region DR.

- [AWS Config](#) provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past. In this view, you can see how the resource configuration and relationships have changed over time.

When you turn on [AWS Config configuration recording](#) for your AWS resources, you maintain a history of your resource relationships over time. This helps to identify and track AWS resource relationships (including deleted resources) for up to seven years. For example, AWS Config can track the relationship of an Amazon EBS snapshot volume and the EC2 instance to which the volume was attached.

- [AWS Lambda](#) can be used to programmatically define and automate your backup and recovery procedures for your workloads. You can use the AWS SDKs to interact with AWS services and their data. You can also use [Amazon CloudWatch Events](#) to run your Lambda functions on a scheduled basis.

AWS services provide specific features for backup and restore. For each AWS service that you are using, consult the AWS documentation to determine the backup, restore, and data protection features provided by the service. You can use the AWS Command Line Interface (AWS CLI), AWS SDKs, and API operations to automate the AWS service-specific features for data backup and recovery.

Designing a backup and recovery solution

When developing a comprehensive strategy for backing up and restoring data, you must first identify possible failure or disaster situations and their potential business impact. In some industries, you must consider regulatory requirements for data security, privacy, and records retention.

Backup and recovery processes should include the appropriate level of granularity to meet recovery time objective (RTO) and recovery point objective (RPO) for the workload and its supporting business processes, including the following:

- File-level recovery (for example, configuration files for an application)
- Application data-level recovery (for example, a specific database within MySQL)
- Application-level recovery (for example, a specific web server application version)
- Amazon EC2 volume-level recovery (for example, an EBS volume)
- EC2 instance-level recovery. (for example, an EC2 instance)
- Managed service recovery (for example, a DynamoDB table)

Be sure to consider all the recovery requirements for your solution and the data dependencies between various components in your architecture. To facilitate a successful restore process, coordinate the backup and recovery between various components in your architecture.

The following topics describe backup and recovery approaches based on the organization of your infrastructure. IT infrastructure can broadly be categorized as on-premises, hybrid, or cloud native.

Backup and recovery using AWS Backup

AWS Backup is a fully managed backup service centralizing and automating the backup of data across AWS services. AWS Backup provides an orchestration layer that integrates Amazon CloudWatch, AWS CloudTrail, AWS Identity and Access Management (IAM), AWS Organizations, and other services. This centralized, AWS Cloud native solution provides global backup capabilities that can help you achieve your disaster recovery and compliance requirements. Using AWS Backup, you can centrally configure backup policies and monitor backup activity for AWS resources.

AWS Backup is an ideal solution for implementing standard backup plans for your AWS resources across your AWS accounts and Regions. Because AWS Backup supports multiple AWS resource types, it makes it easier to maintain and implement a backup strategy for workloads using multiple AWS resources that need to be backed up collectively. AWS Backup also enables you to collectively monitor a backup and restore operation that involves multiple AWS resources.

If you have compliance and audit requirements, you can use the [AWS Backup Audit Manager](#) feature to create audit frameworks and reports to support your compliance requirements. The [AWS Backup Vault Lock](#) feature also supports compliance requirements by enforcing a write-once, read-many (WORM) configuration for all your backups stored in an backup vault in AWS Backup.

A key differentiator for AWS Backup is support for Organizations. Using this support, you can define and manage backup policies at the organization or organizational unit level and automatically have those policies implemented for each related AWS account and Region. As you onboard new AWS accounts and Regions, you don't have to define and manage backup plans separately.

AWS Backup can make it easier for you to implement an organization-wide backup policy by using tags. You can create separate backup plans that each have unique frequency and retention settings and then create unique key-value pair tags that select the resources to include for backup.

For example, you could create a daily backup plan that starts a backup at 05:00 UTC on a daily basis and has a 35-day retention policy. This backup plan can include a [backup resource assignment](#) that specifies that any supported AWS resource with the tag key **backup** and tag value **daily** will be backed up according to this plan. Additionally, you could create a monthly backup plan that starts at 05:00 UTC on the first day of each month and has a 366-day retention policy. This backup plan can include a backup resource assignment that specifies that any supported AWS resource with the tag key **backup** and tag value **monthly** will be backed up according to this plan.

You can then use tag policies and the [required-tags](#) AWS Config rule to ensure that all your AWS supported resources have this tag key and one of these tag values. This approach can help you consistently implement and maintain a standard backup approach in AWS for supported AWS Backup resources. You can extend this approach to standardize backups for your applications and architectural layers that have different recovery point objective (RPO) requirements.

We recommend taking steps to secure your backup vault. For example, you can implement an Organizations service control policy (SCP) that prevents your backup vault from being deleted or from being shared with unintended AWS accounts. For more details and other important security considerations, review the [Top 10 security best practices for securing backups in AWS](#) blog post.

AWS Backup can simplify implementation of your disaster recovery (DR) plan for AWS because it supports multiple AWS resources that can be addressed collectively. For example, you can implement [cross-Region](#) and [cross-account](#) backup for most of the AWS resource types supported by AWS Backup. Cross-account backup improves backup security because a copy is available in a separate account. Cross-

Region backup improves availability because the backups are available in more than one Region. For details about supported AWS resource types, see the [Feature availability by resource](#) table.

You can use the example [Backup and Recovery with AWS Backup open-source solution](#) to implement an infrastructure as code (IaC) and continuous integration and continuous delivery (CI/CD) approach to managing backups for your AWS Organizations organization. This solution includes custom features, such as automatically reapplying AWS tags on restored AWS resources as well as establishing a secondary backup vault in a separate account and Region for DR purposes.

Backup and recovery using Amazon S3 and Amazon S3 Glacier

Amazon S3 and Amazon S3 Glacier are ideal storage services for use in on-premises, hybrid, and cloud-native architectures. These services provide durable, low-cost storage platforms that offer scalable capacity and require no volume or media management as your backup datasets grow. The pay-for-what-you-use model and low cost per GB/month make these services a fit for a broad range of data-protection use cases.

Amazon S3

You can use Amazon S3 to store and retrieve any amount of data, at any time. You can use Amazon S3 as your durable store for your application data and file-level backup and restore processes. For example, you can copy your database backups from a database instance to Amazon S3 with a backup script using the AWS CLI or SDKs.

AWS services use Amazon S3 for highly durable and reliable storage, as in the following examples:

- Amazon EC2 uses Amazon S3 to store Amazon EBS snapshots for EBS volumes and for EC2 instance stores.
- Storage Gateway integrates with Amazon S3 to provide on-premises environments with Amazon S3-backed file shares, volumes, and tape libraries.
- Amazon RDS uses Amazon S3 for database snapshots.

Many third-party backup solutions also use Amazon S3. For example, Arcserve Unified Data Protection supports Amazon S3 for durable backup of on-premises and cloud-native servers.

You can use the Amazon S3-integrated features of these services to simplify your backup and recovery approach. At the same time, you can benefit from the high durability and availability provided by Amazon S3.

Amazon S3 stores data as objects within resources called buckets. You can store as many objects as you want in a bucket. You can write, read, and delete objects in your bucket with fine-grained access control. Single objects can be up to 5 TB in size.

Amazon S3 offers a range of storage classes designed for different use cases, including the following classes:

- **S3 Standard** for general-purpose storage of frequently accessed data (for example, configuration files, unplanned backups, daily backups).
- **S3 Standard-IA** for long-lived, but less frequently accessed data (for example, monthly backups). IA stands for *infrequent access*.

Amazon S3 offers lifecycle policies that you can configure to manage your data throughout its lifecycle. After a policy is set, your data will be migrated to the appropriate storage class without any changes to your application. For more information, see the [Amazon S3 object lifecycle management](#) documentation.

To reduce your costs for backup, use a tiered storage class approach based on your recovery time objective (RTO) and recovery point objective (RPO), as in the following example:

- Daily backups for the past 2 weeks using S3 Standard
- Weekly backups for the past 3 months using S3 Standard-IA
- Quarterly backups for the past year on S3 Glacier Flexible Retrieval
- Yearly backups for the past 5 years on S3 Glacier Deep Archive
- Backups deleted from S3 Glacier Deep Archive after the 5-year mark

You can automate the transition of your backups by using object lifecycle management.

Creating standard S3 buckets for backup and archive

You can create a standard S3 bucket for backup and archive with your corporation's backup and retention policy implemented through S3 lifecycle policies. Cost allocation tagging and reporting for AWS billing is based on the [tags assigned at the bucket level](#). If cost allocation is important, create separate backup and archive S3 buckets for each project or business unit so that you can allocate costs accordingly.

Your backup scripts and applications can use the backup and archive S3 bucket that you create to store point-in-time snapshots for application and workload data. You can create a standard s3 prefix to help you organize your point-in-time data snapshots. For example, if you create hourly backups, consider using a backup prefix such as `YYYY/MM/DD/HH/<WorkloadName>/<files...>`. By doing this, you can quickly retrieve your point-in-time backups manually or programmatically.

Using Amazon S3 versioning to automatically maintain rollback history

You can enable S3 object versioning to maintain a history of object changes, including the ability to revert to a previous version. This is useful for configuration files and other objects that might change more frequently than your point-in-time backup schedule. It's also useful for files that must be reverted individually.

Using Amazon S3 to back up and recover customized configuration files for AMIs

Amazon S3 with object versioning can become your system of record for your workload configuration and option files. For example, you might use a standard AWS Marketplace Amazon EC2 image that is maintained by an ISV. This image might contain software whose configuration is maintained in a number of configuration files. You can maintain your customized configuration files in Amazon S3. When your instance is launched, you can copy these configuration files to your instance as a part of your [instance user data](#). When you apply this approach, you don't need to customize and recreate an AMI to use an updated version.

Using Amazon S3 in your custom backup and restore process

Amazon S3 provides a general-purpose backup store that you can quickly integrate into your existing custom backup processes. You can use the AWS CLI, AWS SDKs, and API operations to integrate your backup and restore scripts and processes that use Amazon S3. For example, you might have a database backup script that performs nightly database exports. You can customize this script to copy your

nightly backups to Amazon S3 for offsite storage. See the [Batch upload files to the cloud](#) tutorial for an overview of how to do this.

You can take a similar approach for exporting and backing up data for different applications based on their individual RPO. Additionally, you can use AWS Systems Manager to run your backup scripts on your managed instances. Systems Manager provides automation, access control, scheduling, logging, and notification for your individual backup processes.

Amazon S3 Glacier

Amazon S3 Glacier is a low-cost, cloud-archive storage service that provides secure and durable storage for data archiving and online backup. To keep costs low, S3 Glacier provides three storage classes from a few milliseconds to hours. S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive provide additional options based on how quickly you need to restore the data. With S3 Glacier, you can reliably store large or small amounts of data at significant savings compared to on-premises solutions. S3 Glacier is well suited for storage of backup data with long or indefinite retention requirements and for long-term data archiving. S3 Glacier provides the following storage classes:

- **S3 Glacier Instant Retrieval** for archiving data that might be needed once per quarter and needs to be restored quickly (milliseconds)
- **S3 Glacier Flexible Retrieval** for archiving data that might infrequently need to be restored, once or twice per year, within a few hours
- **S3 Glacier Deep Archive** for archiving long-term backup cycle data that might infrequently need to be restored within 12 hours

The following table summarizes the archive retrieval options.

Storage class	Expedited	Standard	Bulk
S3 Glacier Instant Retrieval	Not applicable	Not applicable	Not applicable
S3 Glacier Flexible Retrieval	1–5 minutes	3–5 hours	5–12 hours
S3 Glacier Deep Archive	Not available	Within 12 hours	Within 48 hours

Using Amazon S3, you can [set the storage class for each object in your S3 bucket](#) when you create it. After the object is created, you can change the storage class by copying the object to a new object with a different storage class. Or you can enable a lifecycle configuration that will automatically change the storage class of the objects based on the rules you specify.

To automate your backup and restore processes, you can access Amazon S3 Glacier and S3 Glacier Deep Archive via the AWS Management Console, AWS CLI, and AWS SDKs. For more information, see [Amazon S3 Glacier](#).

Using Amazon S3 Lifecycle object transition to Amazon S3 Glacier compared with managing Amazon S3 Glacier archives

Amazon S3 provides convenient transition of S3 objects into Amazon S3 Glacier storage classes, so that you can manage the lifecycle and costs for your backups. However, depending on the size of the objects

and whether you must restore a collection of objects for different components in your architecture, you might want to manage this process yourself.

If you have a large number of small objects that must be restored collectively, consider the cost implications of the following options:

- Using a lifecycle policy to automatically transition objects individually to Amazon S3 Glacier
- Zipping objects into a single file and storing them in Amazon S3 Glacier

Amazon S3 Glacier has minimum capacity charges for each object depending on the storage class you use. For example, S3 Glacier Instant Retrieval has a minimum capacity charge of 128 KB for each object. See the [performance chart](#) for the most up-to-date information.

For each object that you archive to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, Amazon S3 uses 8 KB of storage for the object name and other metadata. Amazon S3 stores this metadata so that you can get a real-time list of your archived objects by using the Amazon S3 API. You are charged S3 Standard rates for this additional storage.

Amazon S3 also adds 32 KB of storage for index and related metadata for each object that is archived to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. This extra data is necessary to identify and restore your object. You are charged Amazon S3 Glacier or S3 Glacier Deep Archive rates for this additional storage.

By zipping your objects into a single file, you can reduce the additional storage used by Amazon S3 Glacier as well as avoid minimum capacity charges for many small objects..

Another important consideration is that lifecycle policies are applied to objects individually. This can impact the integrity of your backup if a collection of objects must be restored collectively from a specific point in time. There is no guarantee that all objects transition at the same time even with the same expiration and lifecycle transition time set across objects. There might be a delay between when the lifecycle rule is satisfied and when the action for the rule is complete. For more information, see the [AWS Knowledge Center](#).

Finally, consider the restoration effort between using archives from lifecycle policies and managing a separate archive that you create. You must initiate a restore for each object from Amazon S3 Glacier separately. This requires you to write a script or use a tool in order to initiate a restore for many objects collectively. You can use [S3 Batch Operations](#) to help reduce the number of individual requests, or you can use the Amazon S3 console.

Securing backup data in Amazon S3 and Amazon Simple Storage Service

Data security is a universal concern, and AWS takes security very seriously. Security is the foundation of every AWS service. Storage services such as Amazon S3 provide strong capabilities for access control and encryption both at rest and in transit. All Amazon S3 and Amazon S3 Glacier API endpoints support Secure Sockets Layer/Transport Layer Security (SSL/TLS) for encrypting data in transit. Amazon S3 Glacier encrypts all data at rest by default. With Amazon S3, you can choose server-side encryption for objects at rest by doing the following:

- Using [server-side encryption with Amazon S3–managed encryption keys](#)
- Using [server-side encryption with AWS Key Management Service \(AWS KMS\) keys stored in AWS KMS](#)

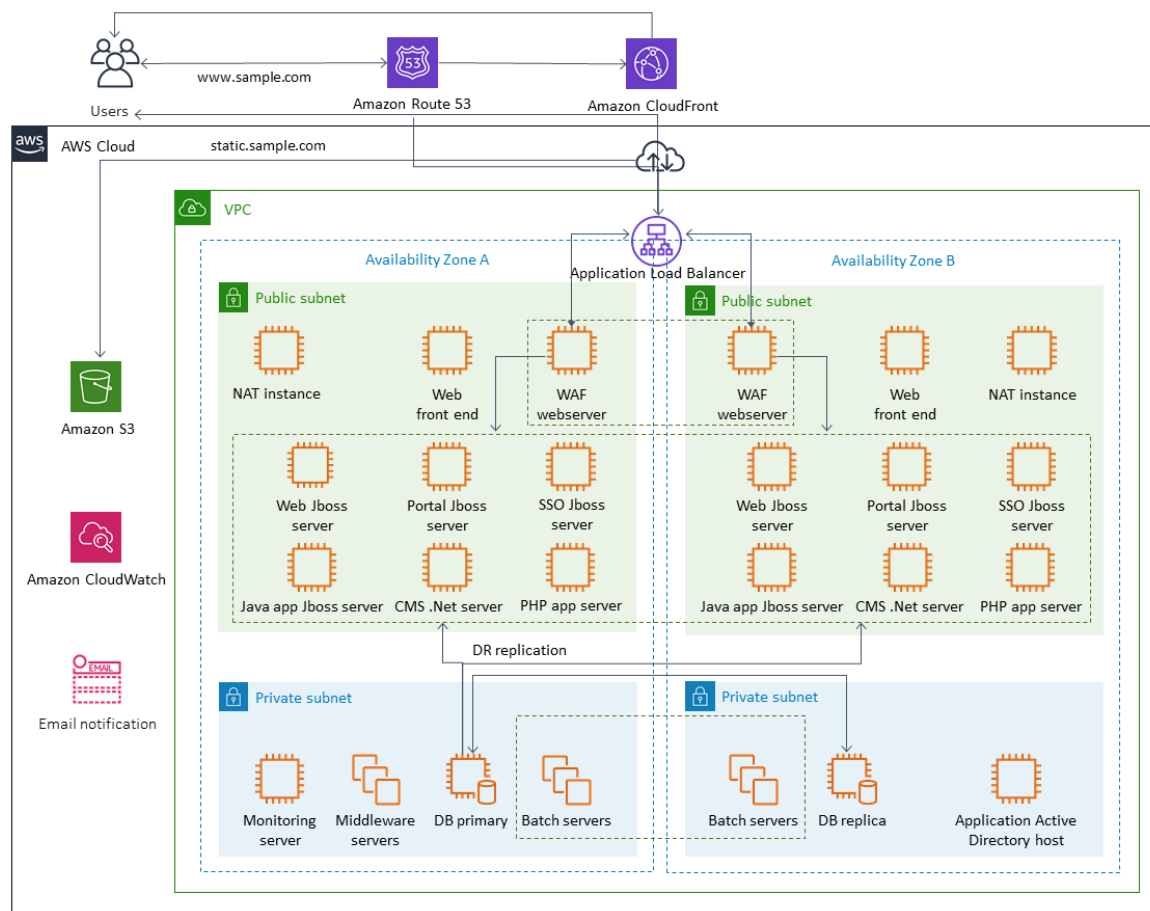
Alternatively, you can encrypt your data before uploading it to AWS. For more information, see the [client-side encryption](#) documentation.

You can use AWS Identity and Access Management (IAM) to control access to S3 objects. IAM provides control over permissions for individual objects and specific prefix paths within an S3 bucket. You can audit access to S3 objects by using [object level logging with AWS CloudTrail](#).

Backup and recovery for Amazon EC2 with EBS volumes

AWS provides multiple methods to back up your Amazon EC2 instances. This section covers different aspects of backing up Amazon Elastic Block Store (Amazon EBS) volumes or instance store volumes for storage. Consider AWS Backup as your first choice for managing backups on AWS if it meets your requirements. Remember that backups are good only if they can be restored to the function for which they were intended. The restore and recovery function should be regularly tested to confirm this.

The solution architecture in the following diagram describes a workload environment that exists entirely on AWS with the majority of the architecture based on Amazon EC2. As the following figure shows, the scenario includes web servers, application servers, monitoring servers, databases, and Active Directory.



AWS provides many fully featured services for many of the Amazon EC2 servers represented in this architecture to perform the undifferentiated work of creating, provisioning, backing up, restoring, and optimizing the instances and storage. Consider whether these services make sense in your architecture to reduce complexity and management. AWS also provides services to improve the availability of your Amazon EC2-based architectures. In particular, consider Amazon EC2 Auto Scaling and Elastic Load Balancing to complement your workloads on Amazon EC2. Using these services can improve the availability and fault tolerance of your architecture and help you to restore impaired instances with minimal user impact.

EC2 instances primarily use Amazon EBS volumes for persistent storage. Amazon EBS provides a number of features for backup and recovery that are covered in detail in this section.

Topics

- [Amazon EC2 backup and recovery with snapshots and AMIs \(p. 15\)](#)
- [Creating EBS volume backups with AMIs and EBS snapshots \(p. 17\)](#)
- [Restoring from an Amazon EBS snapshot or an AMI \(p. 23\)](#)

Amazon EC2 backup and recovery with snapshots and AMIs

EBS volumes are the primary persistent storage option for Amazon EC2. You can use this block storage for structured data, such as databases, or unstructured data, such as files in a file system on a volume.

EBS volumes are placed in a specific Availability Zone. The volumes are replicated across multiple servers to prevent the loss of data from the failure of any single component. Failure refers to a complete or partial loss of the volume, depending on the size and performance of the volume.

EBS volumes are designed for an annual failure rate (AFR) of 0.1-0.2 percent. This makes EBS volumes 20 times more reliable than typical commodity disk drives, which fail with an AFR of around 4 percent. For example, if you have 1,000 EBS volumes running for 1 year, you should expect one or two volumes will have a failure.

Amazon EBS also supports a snapshot feature for taking point-in-time backups of your data. All EBS volume types offer durable snapshot capabilities and are designed for 99.999 percent availability. For more information, see the [Amazon Compute Service Level Agreement](#).

Amazon EBS provides the ability to create snapshots (backups) of any EBS volume. A snapshot is a base feature for creating backups of your EBS volumes. A snapshot takes a copy of the EBS volume and places it in Amazon S3, where it is stored redundantly in multiple Availability Zones. The initial snapshot is a full copy of the volume; ongoing snapshots store incremental block-level changes only. See the [Amazon EC2 documentation](#) for details on how to create Amazon EBS snapshots.

You can perform a restore operation, delete a snapshot, or update the snapshot metadata, such as tags, associated with the snapshot [from the Amazon EC2 console](#) in the same Region that you took the snapshot.

Restoring a snapshot creates a new Amazon EBS volume with full volume data. If you need only a partial restore, you can attach the volume to the running instance under a different device name. Then mount it, and use operating system copy commands to copy the data from the backup volume to the production volume.

Amazon EBS snapshots can also be copied between AWS Regions by using the Amazon EBS snapshot copy capability, as described in the [Amazon EC2 documentation](#). You can use this feature to store your backup in another Region without having to manage the underlying replication technology.

Establishing separate server volumes

You may already use a standard set of separate volumes for the operating system, logs, applications, and data. By establishing separate server volumes, you can reduce the blast radius of application or platform failures due to disk space exhaustion. This risk is usually greater with physical hard drives, because you don't have the flexibility to expand volumes quickly. With physical drives, you must purchase the new drives, back up the data, and then restore the data on the new drives. With AWS, this risk is greatly reduced because you can use Amazon EBS to expand your provisioned volumes. For more information, see the [AWS documentation](#).

Maintain separate volumes for application data, user data, logs, and swap files so that you can use separate backup and restore policies for these resources. By separating volumes for your data, you can also use different volume types based on the performance and storage requirements for the data. You can then optimize and fine-tune your costs for different workloads.

Using AMIs or Amazon EBS snapshots for backups

Consider whether you need to create a full backup of an EC2 instance with an AMI or take a snapshot of an individual volume.

An AMI includes the following:

- One or more snapshots. Instance-store-backed AMIs include a template for the root volume of the instance (for example, an operating system, an application server, and applications).
- Launch permissions that control which AWS accounts can use the AMI to launch instances.
- A block device mapping that specifies the volumes to attach to the instance when it's launched.

You can use AMIs to launch new instances with preconfigured software and data. You can create AMIs when you want to establish a baseline, which is a reusable configuration for launching more instances. When you create an AMI of an existing EC2 instance, a snapshot is taken for all the volumes that are attached to the instance. The snapshot includes the device mappings.

You can't use snapshots to launch a new instance, but you can use them to replace volumes on an existing instance. If you experience data corruption or a volume failure, you can create a volume from a snapshot that you have taken and replace the old volume. You can also use snapshots to provision new volumes and attach them during a new instance launch.

If you are using platform and application AMIs maintained and published by AWS or from the AWS Marketplace, consider maintaining separate volumes for your data. You can back up your data volumes as snapshots that are separate from the operating system and application volumes. Then use the data volume snapshots with newly updated AMIs published by AWS or from the AWS Marketplace. This approach requires careful testing and planning to back up and restore all custom data, including configuration information, on the newly published AMIs.

The restore process is affected by your choice between AMI backups or snapshot backups. If you create AMIs to serve as instance backups, you must launch an EC2 instance from the AMI as a part of your restore process. You might also need to shut down the existing instance to avoid potential collisions. An example of a potential collision is security identifiers (SIDs) for domain-joined Windows instances. The restore process for snapshots might require you to detach the existing volume and attach the newly restored volume. Or you might need to make a configuration change to point your applications to the newly attached volume.

AWS Backup supports both instance-level backups as AMIs and volume-level backups as separate snapshots based on the resource tags.

Considerations for instance store volumes

An instance store provides temporary block-level storage for your instance. This storage is located on disks that are physically attached to the host computer. Instance stores are ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content. They are also preferable for data that are replicated across a fleet of instances, such as a load balanced pool of web servers.

The data in an instance store persists only during the lifetime of its associated instance. If an instance reboots (intentionally or unintentionally), data in the instance store persists. However, data in the instance store is lost under any of the following circumstances.

- The underlying drive fails.
- The instance stops.
- The instance terminates.

Therefore, do not rely on an instance store for valuable, long-term data. Instead, use more durable data storage, such as Amazon S3, Amazon EBS, or Amazon EFS.

A common strategy with instance store volumes is to persist necessary data to Amazon S3 regularly as needed, based on the recovery point objective (RPO) and recovery time objective (RTO). You can then download the data from Amazon S3 to your instance store when a new instance is launched. You can also upload the data to Amazon S3 before an instance is stopped. For persistence, create an EBS volume, attach it to your instance, and copy the data from the instance store volume to the EBS volume on a periodic basis. For more information, see the [AWS Knowledge Center](#).

Tagging and enforcing standards for EBS snapshots and AMIs

Tagging all your AWS resources is an important practice for cost allocation, auditing, troubleshooting, and notification. Tagging is important for EBS volumes so that the pertinent information required to manage and restore volumes is present. Tags are not automatically copied from EC2 instances to AMIs or from source volumes to snapshots. Make sure that your backup process includes the relevant tags from these sources. This helps you to set the snapshot metadata, such as access policies, attachment information, and cost allocation, to use these backups in the future. For more information on tagging your AWS resources, refer to the [tagging best practices technical paper](#).

In addition to the tags you use for all AWS resources, use the following backup-specific tags:

- Source instance ID
- Source volume ID (for snapshots)
- Recovery point description

You can enforce tagging policies by using AWS Config rules and IAM permissions. IAM supports enforced tag usage, so you can write IAM policies that mandate the use of specific tags when acting on Amazon EBS snapshots. If a `CreateSnapshot` operation is attempted without the tags defined in the IAM permissions policy granting rights, the snapshot creation fails with access denied. For more information, see the [blog post on tagging Amazon EBS snapshots on creation and implementing stronger security policies](#).

You can use AWS Config rules to evaluate the configuration settings of your AWS resources automatically. To help you get started, AWS Config provides customizable, predefined rules called managed rules. You can also create your own custom rules. While AWS Config continuously tracks configuration changes among your resources, it checks whether these changes violate any of the conditions in your rules. If a resource violates a rule, AWS Config flags the resource and the rule as *noncompliant*. Note that the [required-tags](#) managed rule does not currently support snapshots and AMIs.

Creating EBS volume backups with AMIs and EBS snapshots

AWS provides a wealth of options for creating and managing AMIs and snapshots. You can use the approach that meets your needs. A common issue that many customers face is managing the snapshot lifecycle and clearly aligning snapshots by purpose, retention policy, etc. Without proper tagging, there

is a risk that snapshots might be deleted accidentally or as part of an automated cleanup process. You might also end up paying for obsolete snapshots that are retained because there is no clear understanding whether they are still needed.

Preparing an EBS volume before creating a snapshot or AMI

Before you take a snapshot or create an AMI, make the necessary preparations to your EBS volume. Creating an AMI results in a new snapshot for each EBS volume that is attached to the instance, so these preparations also apply to AMIs.

You can take a snapshot of an attached EBS volume that is in use. However, snapshots capture only data that has been written to your EBS volume at the time the snapshot command is issued. This might exclude any data that has been cached by applications or the operating system. A best practice is to have the system in a state where it is not performing any I/O. Ideally, the machine isn't accepting traffic and is in a stopped state, but this is rare as 24/7 IT operations become the norm. If you can flush any data from system memory to the disk being used by your applications and pause any file writes to the volume long enough to take a snapshot, your snapshot should be complete.

To make a clean backup, you must quiesce the database or file system. The way in which you do this depends on your database or file system.

The process for a database is as follows:

1. If possible, put the database into hot backup mode.
2. Run the Amazon EBS snapshot commands.
3. Take the database out of hot backup mode or, if using a read replica, terminate the read replica instance.

The process for a file system is similar, but it depends on the capabilities of the operating system or file system. For example, XFS is a file system that can flush its data for a consistent backup. For more information, see [xfs_freeze](#). Alternatively, you can facilitate this process by using a logical volume manager that supports the freezing of I/O.

However, if you can't flush or pause all file writes to the volume, do the following:

1. Unmount the volume from the operating system.
2. Issue the snapshot command.
3. Remount the volume to achieve a consistent and complete snapshot. You can remount and use your volume while the snapshot status is pending.

The snapshot process continues in the background and snapshot creation is fast and captures a point in time. The volumes that you're backing up are unmounted for only a matter of seconds. You can schedule a small backup window where an outage is expected and handled by clients gracefully.

When you create a snapshot for an EBS volume that serves as a root device, stop the instance before you take the snapshot. Windows provides the Volume Shadow Copy Service (VSS) to help create application-consistent snapshots. AWS provides a Systems Manager document that you can run to take image-level backups of VSS-aware applications. The snapshots include data from pending transactions between these applications and the disk. You don't have to shut down your instances or disconnect them when you back up all attached volumes. For more information, see the [AWS documentation](#).

Note

If you are creating a Windows AMI so that you can deploy another similar instance, use [EC2Config](#) or [EC2Launch](#) to [Sysprep](#) your instance. Then create an AMI from the stopped

instance. Sysprep removes unique information from the Amazon EC2 Windows instance, including the SIDs, computer name, and drivers. Duplicate SIDs can cause issues with Active Directory, Windows Server Update Services (WSUS), login issues, Windows volume key activation, Microsoft Office, and third-party products. Do not use Sysprep with your instance if your AMI is for backup purposes and you want to restore the same instance with all its unique information intact.

Creating EBS volume snapshots manually from the console

Create snapshots of the appropriate volumes or the entire instance before you make any major changes that have not been fully tested on the instance. For example, you might want to create a snapshot before you upgrade or patch application or system software on your instance.

You can create a snapshot manually from the console. On the Amazon EC2 console, on the **Elastic Block Store Volumes** page, select the volume that you want to back up. Then on the **Actions** menu, choose **Create Snapshot**. You can search for volumes that are attached to a specific instance by entering the instance ID in the filter box.

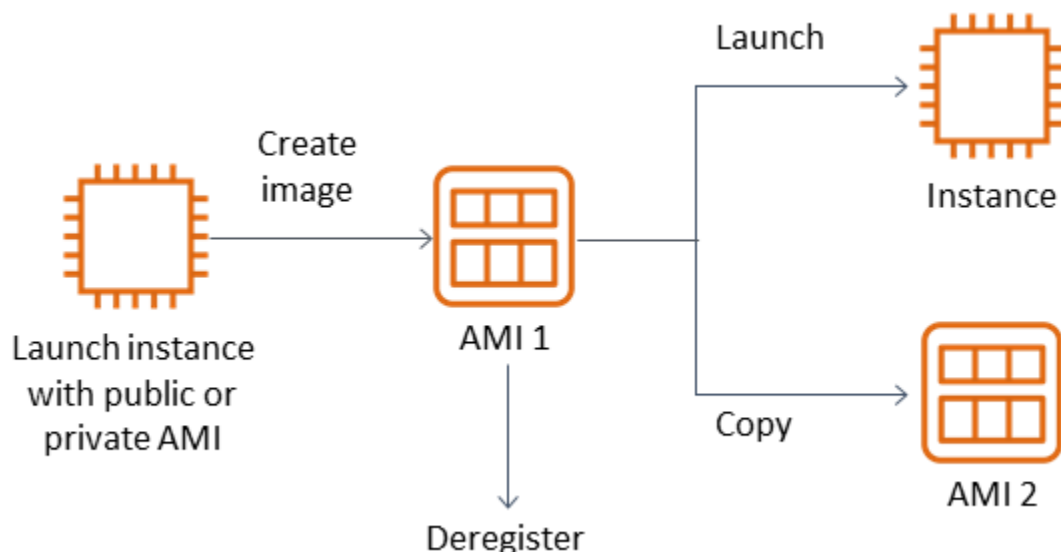
Enter a description and add the appropriate tags. Add a `Name` tag to make it easier to find the volume later. Add any other appropriate tags based on your tagging strategy.

Creating AMIs

An AMI provides the information that is required to launch an instance. The AMI includes the root volume and snapshots of the EBS volumes attached to the instance when the image was created. You can't launch new instances from EBS snapshots alone; you must launch new instances from an AMI.

When you create an AMI, it is created in the account and Region that you are using. The AMI creation process creates Amazon EBS snapshots for each volume attached to the instance, and the AMI refers to these Amazon EBS snapshots. These snapshots reside in Amazon S3 and are highly durable.

After you create an AMI of your EC2 instance, you can use the AMI to re-create the instance or launch more copies of the instance. You can also copy AMIs from one Region to another for application migration or DR.



An AMI must be created from an EC2 instance unless you are migrating a virtual machine, such as a VMWARE virtual machine, to AWS. To create an AMI from the Amazon EC2 console, select the instance, choose **Actions**, choose **Image**, and then choose **Create Image**.

Amazon Data Lifecycle Manager

To automate the creation, retention, and deletion of Amazon EBS snapshots, you can use Amazon Data Lifecycle Manager. Automating snapshot management helps you to do the following:

- Protect valuable data by enforcing a regular backup schedule.
- Retain backups as required by auditors or internal compliance.
- Reduce storage costs by deleting outdated backups.

Using Amazon Data Lifecycle Manager, you can automate the snapshot management process for EC2 instances (and their attached EBS volumes) or separate EBS volumes. It supports options such as cross-Region copy, so you can copy snapshots automatically to other AWS Regions. Copying snapshots to alternative Regions is one approach to support DR efforts and restore options in an alternative Region. You can also use Amazon Data Lifecycle Manager to create a snapshot lifecycle policy that supports [fast snapshot restore](#).

Amazon Data Lifecycle Manager is an included feature of Amazon EC2 and Amazon EBS. There is no charge for Amazon Data Lifecycle Manager.

AWS Backup

AWS Backup is unique from Amazon Data Lifecycle Manager because you can create a backup plan that includes resources across multiple AWS services. You can coordinate your backup to cover the resources you are using together rather than coordinating the backups of the resources individually.

AWS Backup also includes the concept of backup vaults, which can restrict access to the recovery points for your completed backups. Restore operations can be initiated from AWS Backup rather than proceeding to each individual resource and restoring the backup created. AWS Backup also includes a host of additional features, such as audit management and reporting. For more information, see the [Backup and recovery using AWS Backup \(p. 7\)](#) section of this guide.

Performing multi-volume backups

If you want to back up the data on the EBS volumes in a RAID array using snapshots, the snapshots must be consistent. This is because the snapshots of these volumes are created independently. Restoring EBS volumes in a RAID array from snapshots that are out of sync degrades the integrity of the array.



To create a consistent set of snapshots for your RAID array, use the [CreateSnapshots](#) API operation, or log in to the Amazon EC2 console and choose **Elastic Block Store, Snapshots, Create Snapshot**.


AWS Prescriptive Guidance Backup and recovery approaches on AWS Multi-volume backups

Snapshots > Create Snapshot

Create Snapshot

Select resource type Volume Instance

Instance ID*  

Description 

Exclude root volume

Volume ID	Volume Type	Encryption
vol-1111111	Root	Encrypted
vol-2222222	EBS	Not Encrypted
vol-3333333	EBS	Not Encrypted
vol-4444444	EBS	Not Encrypted

Copy tags from volume

Key	Value
This resource currently has no tags	
Choose the Add tag button or click to add a Name tag	

50 remaining (Up to 50 tags maximum)

* Required

Snapshots of instances that have multiple volumes attached in a RAID configuration are taken as a multi-volume snapshot, collectively. Multi-volume snapshots provide point-in-time, data-coordinated, and crash-consistent snapshots across multiple EBS volumes attached to an EC2 instance. You do not have to stop your instance to coordinate between volumes to achieve consistency because snapshots are automatically taken across multiple EBS volumes. After the snapshot for the volumes is initiated (usually a second or two), the file system can continue its operations.

After the snapshots are created, each snapshot is treated as an individual snapshot. You can perform all snapshot operations, such as restore, delete, and cross-Region and account copy, as you would with a single-volume snapshot. You can also tag your multi-volume snapshots as you would a single-volume snapshot. We recommend that you tag your multi-volume snapshots to manage them collectively during restore, copy, or retention. For more information, see the [AWS documentation](#).

You can also perform these backups from a logical volume manager or a file system-level backup. In these cases, using a traditional backup agent enables the data to be backed up over the network. A number of agent-based backup solutions are available on the internet and in the [AWS Marketplace](#).

An alternative approach is to create a replica of the primary system volumes that exist on a single large volume. This simplifies the backup process, because only one large volume must be backed up, and the backup does not take place on the primary system. However, first determine whether the single volume can perform sufficiently during the backup and whether the maximum volume size is appropriate for the application.

Protecting your Amazon EC2 backups

It is important to consider the security of your backups and to prevent accidental or malicious deletion of your backups. You can use a number of approaches collectively to accomplish this. To prevent the loss of your critical backups due to a security breach, we recommend that you copy your backups to another AWS account. If you have multiple AWS accounts, you can designate a separate account as your archive account to which all the other accounts can copy backups. For example, you can accomplish this with a [cross-account backup in AWS Backup](#).

Your disaster recovery plan might also require you to be able to reproduce EC2 instances in another AWS Region in case of a regional failure. You can support this goal by copying your backups to another Region within the same account. This can provide an additional layer of accidental deletion protection as well as support disaster recovery (DR) objectives. AWS Backup provides support for [cross-Region backups](#).

Consider blocking IAM permissions to the [ec2:DeleteSnapshot](#) and [ec2:DeregisterImage](#) actions. Instead, you can let your retention policies and methods manage the lifecycle of EBS snapshots and Amazon EC2 AMIs. Blocking delete actions is one way to implement a write-once, read-many (WORM) strategy for your EBS snapshots. You can also use [AWS Backup Vault Lock](#), which provides support for EBS snapshots and other AWS resources.

Additionally, consider blocking the ability for users to share AMIs and EBS snapshots by blocking the [ec2:ModifyImageAttribute](#) and [ec2:ModifySnapshotAttribute](#) IAM actions. This will prevent your AMIs and snapshots from being shared with AWS accounts that are external to your organization. If you are using AWS Backup, limit users from performing similar operations on backup vaults. For more information, see the [AWS Backup \(p. 7\)](#) section of this guide.

Amazon EC2 includes a [Recycle Bin feature](#) that can help you restore accidentally deleted EBS snapshots. If you allow your users to delete snapshots, turn this feature on so that needed snapshots aren't permanently deleted. Users should be particularly careful about deleting multiple snapshots, because the Amazon EC2 console allows you to select multiple snapshots and delete them in one operation. Additionally, be careful when you use cleanup scripts and automation so that you don't unintentionally delete snapshots you need. The Recycle Bin feature helps provide protection from these types of situations.

Archiving EBS snapshots

[Archiving your EBS snapshots](#) can be a cost-effective method for keeping a copy of a volume for reference purposes that you don't intend to restore for 90 or more days. This can be a good intermediate step before permanently deleting all related snapshots for an EBS volume. For example, you might consider archiving snapshots as an end-of-lifecycle step for EBS volumes that are no longer used. Archiving rather than deleting can also be a more cost effective method of deletion retention instead of using the Recycle Bin.

Automating snapshot and AMI creation with Systems Manager, the AWS CLI, and the AWS SDKs

Your backup approach might require operations before and after a snapshot or AMI is created. For example, you might need to stop and start services to quiesce the file system. Or you might need to stop and start your instance during AMI creation. You might also need to create backups of multiple components in your architecture collectively, each with its own pre-creation and post-creation steps.

You can reduce your maintenance window times for your backups by automating your process and verifying that your backup process is consistently applied. To automate your custom pre-creation and post-creation operations, script your backup process by using the AWS CLI and the SDK.

Your automation can be defined in a Systems Manager runbook that can be run on demand or during a Systems Manager maintenance window. You can grant your users access to run Systems Manager runbooks without the need to grant them permissions to Amazon EC2 disruptive commands. This can also help you verify that your backup process and tags are applied consistently by your users. You can use the [AWS-CreateSnapshot](#) and [AWS-CreateImage](#) runbooks for creating snapshots and AMIs, or you can grant other users permissions to use them. Systems Manager also includes the [AWS-UpdateLinuxAmi](#) and [AWS-UpdateWindowsAmi](#) runbooks to automate the AMI patching and AMI creation.

You can also use the AWS CLI and [AWS Tools for Windows PowerShell](#) to automate your snapshot and AMI creation process. You can use the `aws ec2 create-snapshot` AWS CLI command to create a snapshot of an EBS volume as one step in your automation. You can use the `aws ec2 create-snapshots` command to create crash-consistent, synchronized snapshots of all volumes that are attached to your EC2 instance.

You can use the AWS CLI to create new AMIs. You can use the `aws ec2 register-image` command to create a new image for your EC2 instance. To automate the shutdown, image creation, and restart of your instances, combine this command with the `aws ec2 stop-instances` and `aws ec2 start-instances` commands.

Restoring from an Amazon EBS snapshot or an AMI

You can restore an entire EC2 instance including all of its associated volumes by restoring an AMI backup of your instance. If you need to restore only a single volume attached to an EC2 instance, you can restore that volume separately, detach the existing volume, and attach the restored volume to your EC2 instance.

To reduce the recovery time and impact to dependent applications and processes, your restore process must consider the resource that it is replacing. For best results, regularly test your restore process in lower environments (for example, non-production) to verify that your process meets your recovery point objective (RPO) and recovery time objective (RTO) and that the restore process works as expected. Consider how the restore process will impact applications and services that depend on the instance you are restoring, and then coordinate the restore as necessary. Try to automate and test the restore process as much as possible to reduce the risk of your restore process failing or being implemented inconsistently.

Data from an Amazon EBS snapshot is asynchronously loaded into an EBS volume. If an application accesses the volume where the data is not loaded, there is higher latency than normal while the data is loaded from Amazon S3. To avoid this impact for latency-sensitive applications, you can pre-warm your data from a snapshot into an EBS volume. For an additional charge, Amazon EBS supports [fast snapshot restore](#), which reduces the need to pre-warm your data.

Your workload architecture impacts your restore procedure. For example, if you use Elastic Load Balancing, with multiple instances servicing traffic, you can take a failed or impaired instance out of service. Then you can restore a new instance to replace it while the other instances continue to service traffic without disruption to users.

The following restore processes described are for instances that are not using Elastic Load Balancing.

Restoring an EBS volume from an Amazon EBS snapshot

You can restore a volume attached to an existing EC2 instance by creating a volume from its snapshot and attaching it to your instance. You can use the console, the AWS CLI, or the API operations to create a volume from an existing snapshot. You can then mount the volume to the instance by using the operating system.

If you are replacing a volume that must use the same mount point, unmount that volume so that you can mount the new volume in its place. To unmount the volume, first stop any processes that are using the volume. If you are replacing the root volume, you must stop the instance first before you can detach the root volume.

For example, follow these steps to restore a volume to an earlier point-in-time backup by using the console:

1. On the Amazon EC2 console, on the **Elastic Block Store** menu, choose **Snapshots**.
2. Search for the snapshot that you want to restore, and select it.
3. Choose **Actions**, and then choose **Create Volume**.
4. Create the new volume in the same Availability Zone as your EC2 instance.
5. On the Amazon EC2 console, select the instance.
6. In the instance details, make note of the device name that you want to replace in the **Root device** entry or **Block Devices** entries.
7. Attach the volume. The process differs for root volumes and non-root volumes.

For root volumes:

1. Stop the EC2 instance.
2. On the **EC2 Elastic Block Store Volumes** menu, select the root volume that you want to replace.
3. Choose **Actions**, and then choose **Detach Volume**.
4. On the **EC2 Elastic Block Store Volumes** menu, select the new volume.
5. Choose **Actions**, and then choose **Attach Volume**.
6. Select the instance that you want to attach the volume to, and use the same device name that you noted earlier.

For non-root volumes:

1. On the **EC2 Elastic Block Store Volumes** menu, select the non-root volume that you want to replace.
2. Choose **Actions**, and then choose **Detach Volume**.
3. Attach the new volume by choosing it on the **EC2 Elastic Block Store Volumes** menu and then choosing **Actions, Attach Volume**. Select the instance that you want to attach it to, and then select an available device name.
4. Using the operating system for the instance, unmount the existing volume, and then mount the new volume in its place.

In Linux, you can use the `umount` command. In Windows, you can use a logical volume manager (LVM) such as the Disk Management system utility.

5. Detach any prior volumes that you may be replacing by choosing it on the **EC2 Elastic Block Store Volumes** menu and then choosing **Actions, Detach Volume**.

You can also use the AWS CLI in combination with operating system commands to automate these steps.

Restoring a running instance from an AMI

You can bring up a new instance from your AMI backup to replace an existing, running instance. One approach is to stop the existing instance, keep it offline while you launch a new instance from your AMI, and perform any necessary updates. This approach reduces the risk of conflicts from both instances running simultaneously. It is an acceptable approach if the services that your instance provides are down or you are performing the restore during a maintenance window. After you test your new instance, you

can reassign any Elastic IP addresses that were allocated to the old instance. Then you can update any Domain Name Service (DNS) records to point to the new instance.

However, if during a restore you must minimize the downtime of your in-service instance, consider launching and testing a new instance from your AMI backup. Then replace the existing instance with the new instance.

While both instances are running, you must prevent the new instance from causing any platform-level or application-level collisions. For example, you might run into problems with domain-joined Windows instances that are running with the same SIDs and computer name. You might encounter similar issues with network applications and services that require unique identifiers.

You can use security groups to temporarily block all inbound connections for your new instance except for your own IP address for access and testing. This will prevent other servers and services from connecting to and utilizing your new instance before it is ready. You can also block outbound connections temporarily for the new instance to prevent services and applications from initiating any connections or updates to other resources. When ready, stop the existing instance, and then start services and processes on the new instance.

Backup and recovery from on-premises infrastructure to AWS

You can use AWS for durable, offsite storage of your on-premises infrastructure backups. By using AWS storage services in this scenario, you can focus on backup and archiving tasks. You don't have to worry about storage infrastructure provisioning, scaling, or infrastructure capacity for your backup tasks.

Amazon S3 and Amazon S3 Glacier provide extensive API operations and SDKs for integrating these services into your new and existing backup and recovery approaches. This also gives backup software vendors ways to directly integrate their applications with AWS storage solutions.

In this scenario, backup and archive software that you are using in your on-premises infrastructure directly interfaces with AWS through the API operations. Because the backup software is AWS-aware, it backs up the data from the on-premises servers directly to Amazon S3 or Amazon S3 Glacier.

If your existing backup software does not natively support the AWS Cloud, you can use Storage Gateway. A cloud storage service, Storage Gateway gives your on-premises systems access to scalable cloud storage. It supports open standard storage protocols that work with your existing applications while securely storing your data encrypted in Amazon S3 or Amazon S3 Glacier. You can use Storage Gateway as a part of a backup and recovery approach for your on-premises block-based storage workloads.

Storage Gateway is helpful in hybrid scenarios where you want to transition to cloud-based storage for your backups. Storage Gateway also helps you reduce capital investments in on-premises storage. You deploy Storage Gateway as a VM or a dedicated hardware appliance. This guide focuses on how Storage Gateway applies to backup and recovery.

Storage Gateway provides three different options to satisfy different requirements:

- A file gateway for storing application data files and backup images as durable objects on Amazon S3 cloud storage using SMB-based or NFS-based access.
- A volume gateway for presenting cloud-based iSCSI block storage volumes to your on-premises applications. A volume gateway provides either a local cache or full volumes on premises while also storing full copies of your volumes in the AWS Cloud.
- A tape gateway for pointing trusted backup software at an on-premises storage gateway that, in turn, connects to Amazon S3 and Amazon S3 Glacier. This option delivers the scale and durability of the cloud for safe, long-term retention without disrupting existing investments or processes.

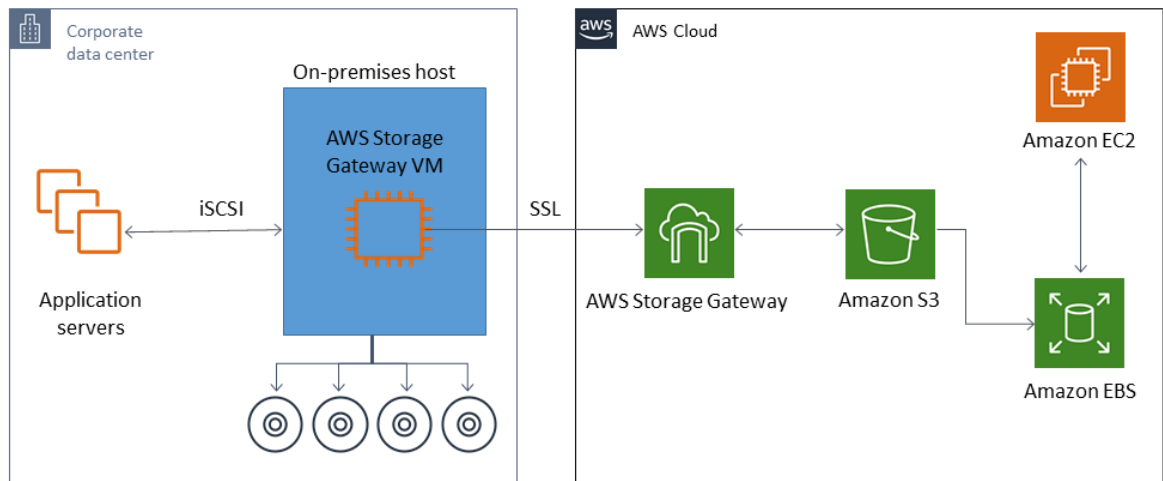
File gateway

Many organizations start their cloud journey by moving secondary and tertiary data, such as backups, to the cloud. A file gateway's SMB and NFS interface support provides a way for IT groups to transition backup jobs from existing on-premises backup systems to the cloud. Backup applications, native database tools, or scripts that can write to SMB or NFS can write to a file gateway. The file gateway stores the backups as Amazon S3 objects of up to 5 TiB in size. With an adequately sized local cache, recent backups can be used for fast on-site recoveries. Long-term retention needs are addressed by tiering backups to low-cost S3 Standard-Infrequent Access and Amazon S3 Glacier storage tiers.

File gateway provides an on ramp for your block-based storage to Amazon S3 for highly durable offsite backups. It is especially useful for scenarios in which a recently backed up file must be restored quickly. Because a file gateway supports the SMB and NFS protocols, users can access files the same way they would access a network file share. You can also take advantage of Amazon S3 object versioning capabilities. Using object versioning, you can restore previous object versions for a file and then easily access them by using SMB or NFS.

Volume gateway

A volume gateway enables you to provision cloud-based iSCSI block storage volumes for your on-premises servers. The volume gateway stores your volume data to Amazon S3 for durable, scalable cloud-based offsite storage. A volume gateway facilitates taking full point-in-time snapshots of your volumes and storing them in the cloud as Amazon EBS snapshots. After they are stored as snapshots, whole volumes can be restored as EBS volumes and attached to EC2 instances, accelerating a cloud-based DR solution. The volumes can also be restored to Storage Gateway, enabling your on-premises applications to revert back to a previous state.



Because a volume gateway integrates with the Amazon EBS volume feature of Amazon EC2, you can use AWS Backup to automate and schedule your snapshot process. A volume gateway provides you with the added benefits of durable, Amazon S3-backed Amazon EBS snapshots and tagging features. For more information, see the [Amazon EBS snapshot documentation](#).

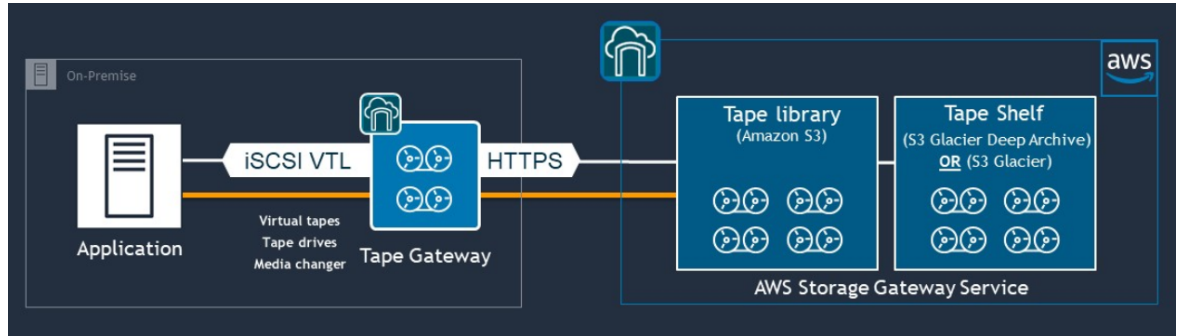
Tape gateway

A tape gateway offers the high durability, low-cost tiered storage, and extensive features of Amazon S3 and Amazon S3 Glacier for your offsite virtual tape backup store. All your virtual tapes stored in Amazon S3 and Amazon S3 Glacier are replicated and stored across at least three geographically dispersed Availability Zones. Your virtual tapes are protected by 11 nines of durability.

AWS also performs fixity checks on a regular basis to confirm that your data can be read and that no errors have been introduced. All tapes stored in Amazon S3 are protected by server-side encryption using default keys or your AWS KMS keys. In addition, you avoid physical security risk associated with tape portability. With a tape gateway, you get correct data, compared to offsite warehousing of tapes, where you might receive an incorrect or broken tape during restore.

You can save on monthly storage costs when storing your data in Amazon S3. You can save even more for your long-term archival requirements by using S3 Glacier Deep Archive.

AWS Prescriptive Guidance Backup
and recovery approaches on AWS
Tape gateway



A tape gateway acts as a virtual tape library (VTL) that spans from your on-premises environment to highly scalable, redundant, and durable storage services: Amazon S3, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive.

The tape gateway presents Storage Gateway to your existing backup application as an open standard iSCSI-based VTL, with a virtual media changer and virtual tape drives. You can continue to use your existing backup applications and workflows while writing to a collection of virtual tapes stored on massively scalable Amazon S3. When you no longer require immediate or frequent access to the data on a virtual tape, your backup application can archive it into S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, further reducing storage costs.

You can retrieve a tape that is archived in S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive typically in 3–5 hours or 12 hours, respectively. The tape gateway can be used with a backup application that is compatible with the iSCSI-based tape library interface for accessing the virtual tapes. Also consider the minimum 100-GB storage size per tape. For more information, review the list of [third-party backup applications](#) that support tape gateways.

Backup and recovery of applications from AWS to your data center

You might have a policy requiring you to implement a scenario such as DR or business continuity for your cloud-based workloads and your on-premises infrastructure. If you already have a data backup framework for your on-premises servers, you can extend it to your AWS resources over a VPN connection or through AWS Direct Connect. You can install the backup agent on the EC2 instances and back up your data and applications according to your data-protection policies. You can also use Amazon S3 as the intermediate service to store your application-level backups. You can then use the API operations, SDKs, or the AWS CLI to restore the data to your on-premises environment.

To back up data in AWS services other than Amazon EC2, use the AWS CLI, SDKs, and API operations to extract the data into your desired format. Then copy the data to Amazon S3, and copy it from Amazon S3 to your on-premises environment. Some services provide direct export to Amazon S3. For example, Amazon RDS supports [native backup](#) of Microsoft SQL Server databases to Amazon S3.

Backup and recovery of cloud-native AWS services

Your backup and recovery approach should cover the AWS services that are used in your workloads. AWS provides service-specific features and options for managing and interacting with your data. You can use the console, the AWS CLI, SDKs, and API operations to implement backup and recovery for the AWS services that you are using. This guide covers [Amazon RDS \(p. 30\)](#) and [Amazon DynamoDB \(p. 31\)](#) as examples. AWS Backup supports both DynamoDB and Amazon RDS and should be used if it satisfies your requirements.

Backup and recovery for Amazon RDS

Amazon RDS includes features for automating database backups. Amazon RDS creates a storage volume snapshot of your database instance, backing up the entire DB instance, not individual databases only. Using Amazon RDS, you can establish a backup window for automated backups, create database instance snapshots, and share and copy snapshots across Regions and accounts.

Amazon RDS provides two different options for backing up and restoring your DB instances:

- **Automated backups** provide point-in-time recovery (PITR) of your DB instance. Automated backups are turned on by default when you create a new DB instance.

Amazon RDS performs a full daily backup of your data during a backup window that you define when you create the DB instance. You can configure a retention period of up to 35 days for the automated backup. Amazon RDS also uploads the transaction logs for DB instances to Amazon S3 every 5 minutes. Amazon RDS uses your daily backups along with your database transaction logs to restore your DB instance. You can restore the instance to any second during your retention period, up to the `LatestRestorableTime` (typically, the last five minutes).

To find the latest restorable time for your DB instances, use the `DescribeDBInstances` API call. Or look on the **Description** tab for the database on the Amazon RDS console.

When you initiate a PITR, transaction logs are combined with the most appropriate daily backup to restore your DB instance to the requested time.

- **DB snapshots** are user-initiated backups that you can use to restore your DB instance to a known state as frequently as you like. You can then restore to that state at any time. You can use the Amazon RDS console or the `CreateDBSnapshot` API call to create DB snapshots. These snapshots are kept until you use the console or the `DeleteDBSnapshot` API call to explicitly delete them.

Both of these backup options are supported for Amazon RDS in AWS Backup, which also provides other features. Consider using AWS Backup to set up a standard backup plan for your Amazon RDS databases, and use the user-initiated instance backup options when your backup plans for a particular database are unique.

Amazon RDS prevents direct access to the underlying storage used by the DB instance. This also prevents you from directly exporting the database on an RDS DB instance to its local disk. In some cases, you can use native backup and restore functions using client utilities. For example, you can use the [mysqldump command with an Amazon RDS MySQL database](#) to export a database to your local client machine. In some cases, Amazon RDS also provides augmented options for performing a native backup and restore of a database. For example, Amazon RDS provides stored procedures to [export and import RDS database backups of SQL Server databases](#).

Be sure to thoroughly test your database restore process and its impact on database clients as a part of your overall backup and restore approach.

Using DNS CNAME records to reduce client impact during a database recovery

When you restore a database by using PITR or an RDS DB instance snapshot, a new DB instance with a new endpoint is created. In this way, you can create multiple DB instances from a specific DB snapshot or point in time. There are special considerations when you restore an RDS DB instance to replace a live RDS DB instance. For example, you must determine how you will redirect your existing database clients to the new instance with minimal interruption and modification. You also must ensure continuity and consistency in the data within the database by considering the restored data time and the recovery time when the new instance begins receiving writes.

You can create a separate DNS CNAME record that points to your DB instance endpoint and have your clients use this DNS name. Then you can update the CNAME to point to new, restored endpoint without having to update your database clients.

Set the Time to Live (TTL) for your CNAME record to an appropriate value. The TTL that you specify determines how long the record is cached with DNS resolvers before another request is made. It is important to note that some DNS resolvers or applications might not honor the TTL, and they might cache the record for longer than the TTL. For Amazon Route 53, if you specify a longer value (for example, 172800 seconds, or two days), you reduce the number of calls that DNS recursive resolvers must make to Route 53 to get the latest information in this record. This reduces latency and reduces your bill for the Route 53 service. For more information, see [How Amazon Route 53 routes traffic for your domain](#).

Applications and client operating systems might also cache DNS information that you have to flush or restart to initiate a new DNS resolution request and retrieve the updated CNAME record.

When you initiate a database restore and shift traffic to your restored instance, verify that all your clients are writing to your restored instance instead of your prior instance. Your data architecture might support restoring your database, updating DNS to shift traffic to your restored instance, and then remediating any data that may still be written to your prior instance. If this isn't the case, you can stop your existing instance before you update the DNS CNAME record. Then all access is from your newly restored instance. This may temporarily cause connection problems for some of your database clients that you can handle individually. To reduce client impact, you can perform the database restore during a maintenance window.

Write your applications to handle database connection failures gracefully with retries using exponential backoff. This enables your application to recover when a database connection becomes unavailable during a restore without causing your application to unexpectedly crash.

After you have completed your restore process, you can keep your prior instance in a stopped state. Or you can use security group rules to limit traffic to your prior instance until you are satisfied that it is no longer needed. For a gradual decommissioning approach, first limit access to a running database by the security group. You can eventually stop the instance when it is no longer needed. Finally, take a snapshot of the database instance and delete it.

Backup and recovery for DynamoDB

DynamoDB provides PITR, which makes nearly continuous backups of your DynamoDB table data. When enabled, DynamoDB maintains incremental backups of your table for the last 35 days until you explicitly turn it off.

You can also create on-demand backups of your DynamoDB table by using the DynamoDB console, the AWS CLI, or the DynamoDB API. For more information, see [Backing up a DynamoDB table](#). You can schedule periodic or future backups by using AWS Backup, or you can customize and automate your backup approach by using Lambda functions. For more information about using Lambda functions for backup of DynamoDB, see the blog post [A serverless solution to schedule your Amazon DynamoDB On-Demand Backup](#). If you don't want to create scheduling scripts and cleanup jobs, you can use AWS Backup to create backup plans. The backup plans include schedules and retention policies for your DynamoDB tables. AWS Backup creates the backups and deletes prior backups based on your retention schedule. AWS Backup also includes advanced DynamoDB backup options that aren't available in the DynamoDB service, including lower-cost tiered storage, and cross-account and cross-Region copy. For more information, see [Advanced DynamoDB backup](#).

You must manually set up the following on a restored DynamoDB table:

- Automatic scaling policies
- IAM policies
- Amazon CloudWatch metrics and alarms
- Tags
- Stream settings
- TTL settings

You can restore only the entire table data to a new table from a backup. You can write to the restored table only after it becomes active.

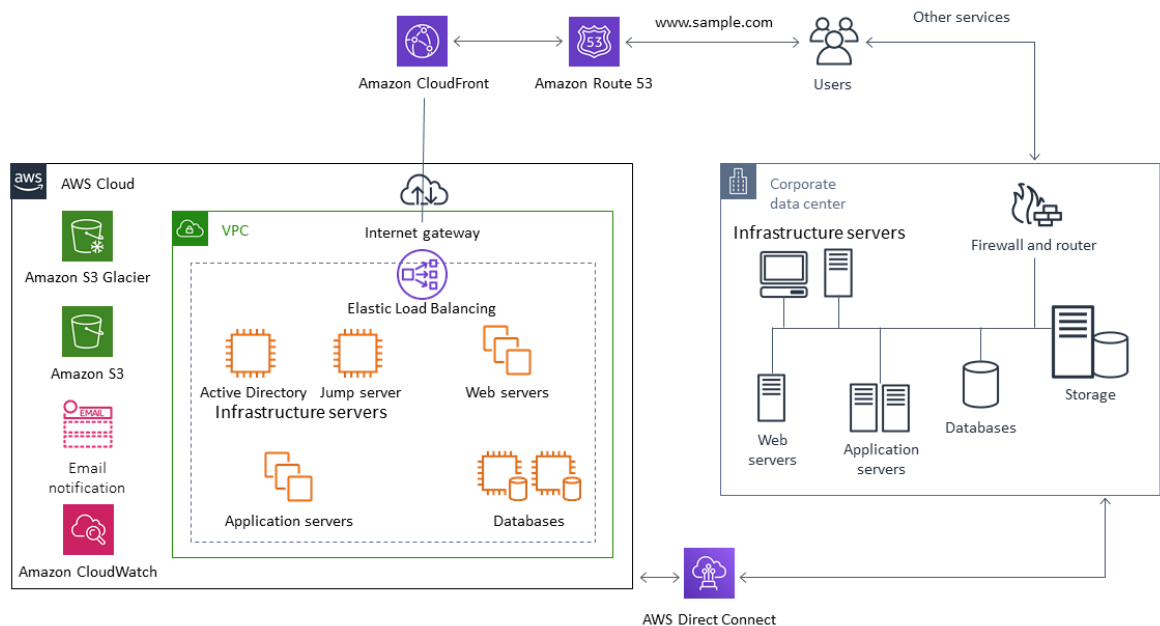
Your restore process must consider how clients will be directed to use the newly restored table name. You can configure your applications and clients to retrieve the DynamoDB table name from a configuration file, AWS Systems Manager Parameter Store value, or another reference that can be updated dynamically to reflect the table name that the client should use.

As a part of the restore process, you should carefully consider your switch-over process. You might choose to deny access to your existing DynamoDB table via IAM permissions and allow access to your new table. You can then update the application and client configuration to use the new table. You might also need to reconcile the differences between your existing DynamoDB table and the newly restored DynamoDB table.

Backup and recovery for hybrid architectures

The cloud-native and on-premises deployments discussed in this guide can be combined into hybrid scenarios where the workload environment has on-premises and AWS infrastructure components. Resources, including web servers, application servers, monitoring servers, databases, and Microsoft Active Directory, are hosted either in the customer data center or on AWS. Applications that are running in the AWS Cloud are connected to applications that are running on premises.

This is becoming a common scenario for enterprise workloads. Many enterprises have data centers of their own and use AWS to augment capacity. These customer data centers are often connected to the AWS network by high-capacity network links. For example, with [AWS Direct Connect](#), you can establish private, dedicated connectivity from your on-premises data center to AWS. This provides the bandwidth and consistent latency to upload data to the cloud for the purposes of data protection. It also provides consistent performance and latency for hybrid workloads. The following diagram provides one example of a hybrid environment approach.



Well-designed data protection solutions typically use a combination of the options described in the cloud-native and on-premises solutions in this guide. Many ISVs provide market leading backup and restore solutions for on-premises infrastructure and have expanded their solutions to support hybrid approaches.

Moving centralized backup management solutions to the cloud for higher availability

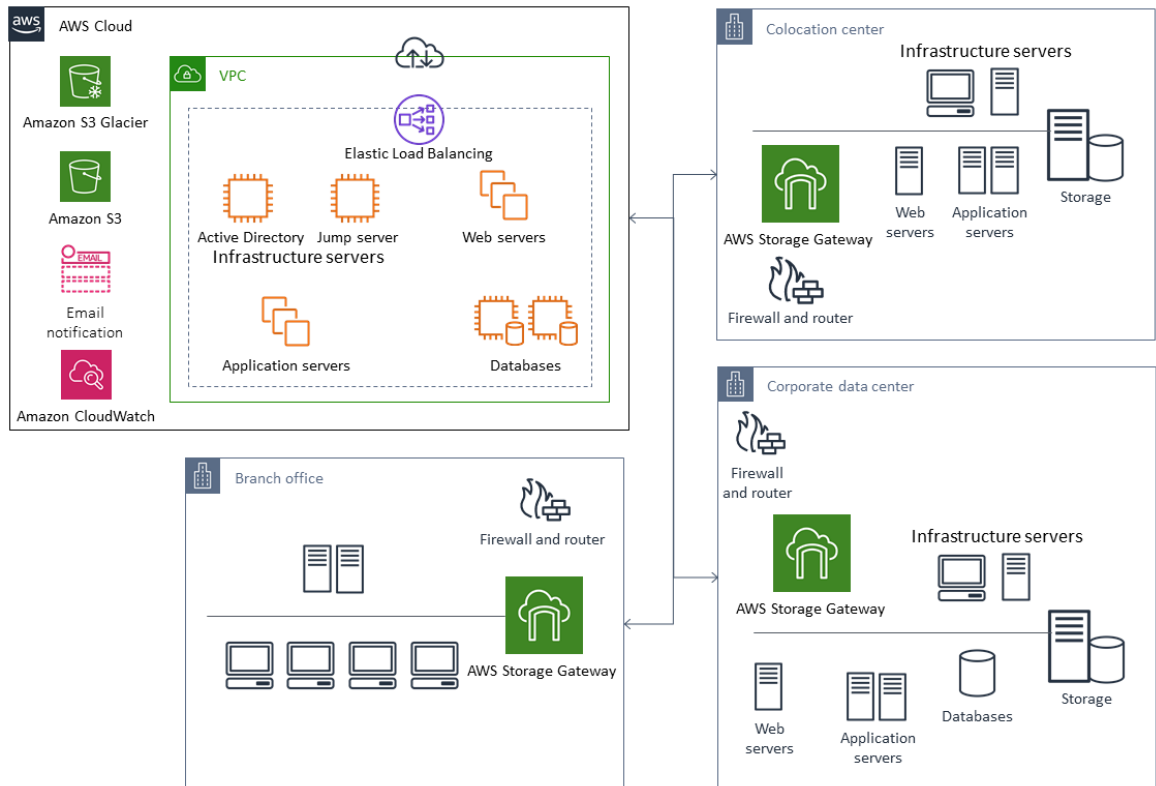
By using your existing backup management solution investments with AWS, you can improve the resilience and architecture of your approach. You might have a primary backup server and one or more

AWS Prescriptive Guidance Backup
and recovery approaches on AWS
Moving centralized backup management solutions

media or storage servers located on-premises across multiple locations close to the servers and services they are protecting. In this case, consider moving the primary backup server to an EC2 instance to protect it from on-premises disasters and for high availability.

To manage the backup data flows, you can create one or more media servers on EC2 instances in the same Region as the servers they will protect. Media servers near the EC2 instances save you money on internet transfer. When you back up to Amazon S3 or Amazon S3 Glacier, media servers increase overall backup and recovery performance.

You can also use Storage Gateway to provide centralized cloud access to data from geographically dispersed data centers and offices. For example, a file gateway gives you on-demand, low-latency access to data stored in AWS for application workflows that can span the globe. You can use features such as cache refresh to refresh data in geographically distributed locations so that content can be easily shared across your offices.



Disaster recovery with AWS

The backup and restore approaches and supporting services and technologies can be used to implement your disaster recovery (DR) solution. Many enterprises are using the AWS Cloud for backup and restore and as a DR site. AWS provides a number of services and features that support DR and business continuity.

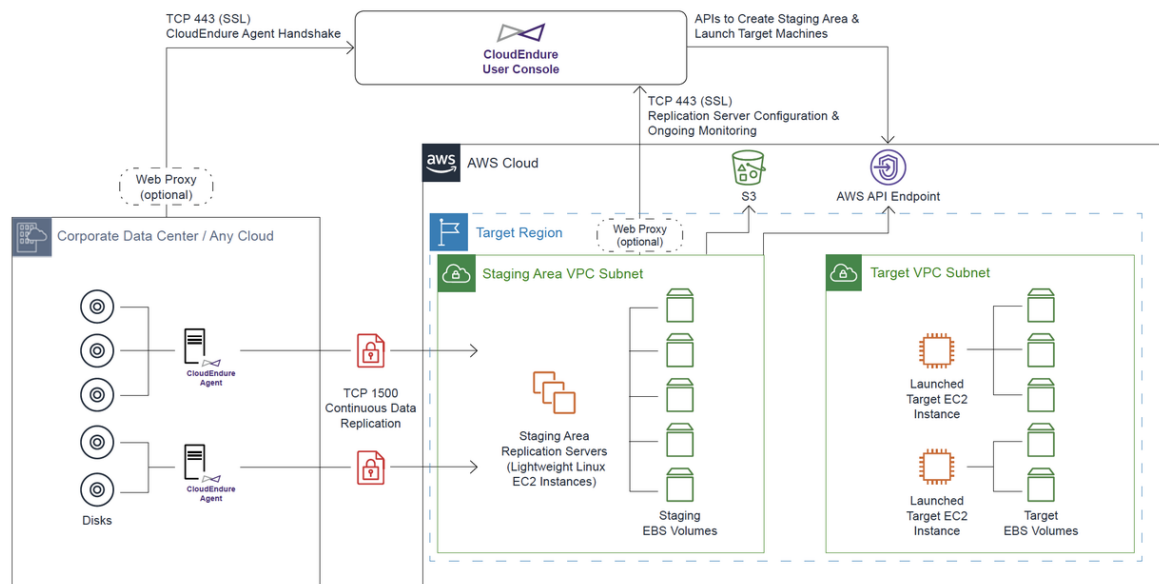
On-premises DR to AWS

Using AWS as an offsite DR environment for on-premises workloads is a common hybrid scenario. Define your DR objectives, including the required recovery time and recovery point objectives, before selecting technologies to use. To help with this definition, you can use the [DR plan checklist](#).

There are a number of options available to help you quickly set up and provision a DR environment on AWS. Be sure that you account for all your workload dependencies, and test your DR plan and solution thoroughly and regularly to verify its integrity.

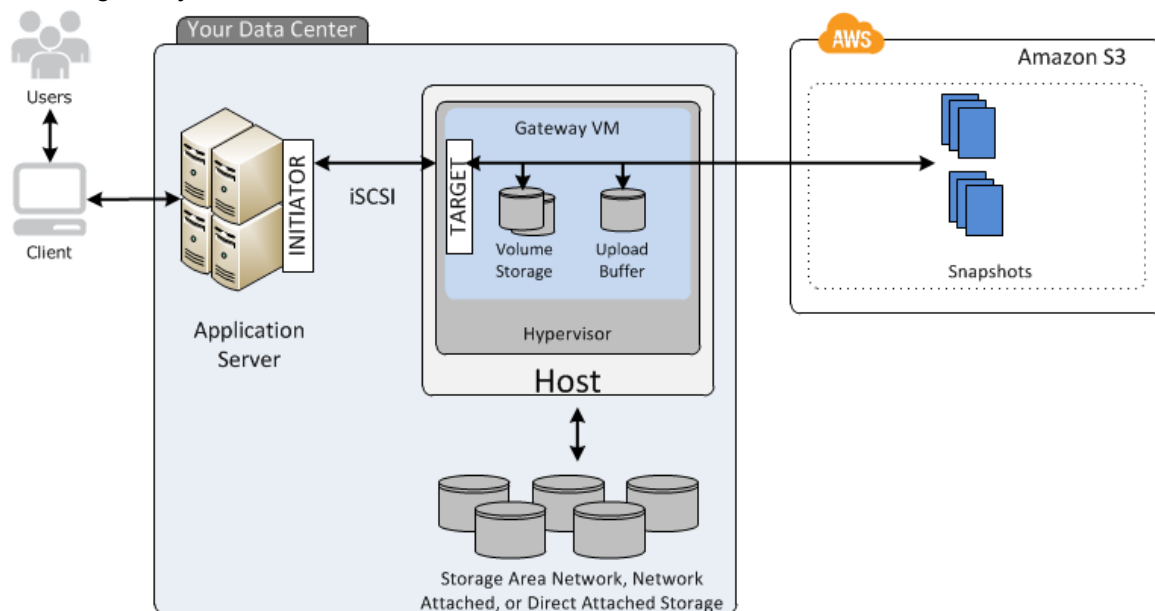
AWS provides [CloudEndure Disaster Recovery](#) for creating a full replica of your on-premises servers, including the root volume and operating system, on AWS. CloudEndure Disaster Recovery continuously replicates your machines into a low-cost staging area in your target AWS account and preferred Region. The replication includes operating system, system state configuration, databases, applications, and files. If there is a disaster, you can instruct CloudEndure Disaster Recovery to automatically launch thousands of your machines in their fully provisioned state in minutes.

CloudEndure Disaster Recovery uses an agent installed on each of your on-premises servers. The agents synchronize the state of your on-premises servers with lower-powered Amazon EC2 equivalents running on AWS. CloudEndure Disaster Recovery automates your DR failover process. It also automates and coordinates the failback process, enabling you to achieve lower recovery times.



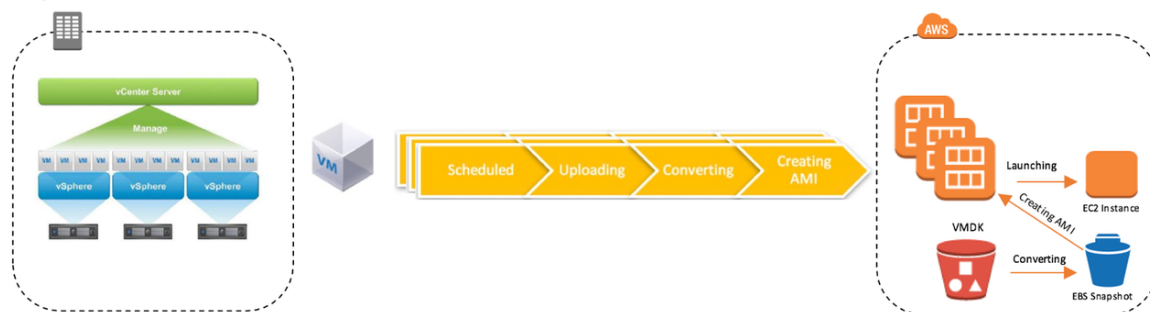
It is important to test the DR process and to verify that the live staging environment does not create conflicts with the on-premises environment. For example, confirm that the appropriate licenses are available and functioning in your on-premises, staging, and initiated DR environment. Also confirm that any worker type processes that may poll and pull work from a central database are configured appropriately to avoid overlaps or conflicts.

You can use a [Storage Gateway volume gateway](#) to provide your on-premises servers with cloud-based volumes. These volumes can also be quickly provisioned for use with Amazon EC2 using Amazon EBS snapshots. In particular, stored volume gateways provide your on-premises applications with low-latency access to their entire datasets. The volume gateways also provide durable snapshot-based backups that can be restored for on premises use or for use with Amazon EC2. You can schedule point-in-time snapshots based on the recovery point objective (RPO) for your workload. It is important to note that volume gateway volumes are intended to be used as data volumes and not as boot volumes.



You can use an Amazon EC2 AMI with a configuration that matches your on-premises servers and specifies your data volumes separately. After you configure and test the AMI, provision the EC2 instances from the AMI along with the data volumes based on the volume gateway snapshots. This approach requires you to test your environment thoroughly to verify that your EC2 instance is operating properly, especially for Windows workloads.

You can also use the [AWS Server Migration Service](#) to create AMIs from your existing on-premises VMs. You can then customize and configure them with appropriate data backup and restore process with your on-premises servers.



DR for cloud-native workloads

Consider how your cloud-native workloads align to your DR objectives. AWS provides multiple Availability Zones in Regions around the world. Many enterprises using the AWS Cloud align their workload architectures and DR objectives to withstand the loss of an Availability Zone. The [Reliability](#)

[Pillar](#) in the AWS Well-Architected Framework supports this best practice. You can architect your workloads and their service and application dependencies to use multiple Availability Zones. You can then automate your DR and achieve your DR objectives with minimal to no intervention.

In practice, however, you might find that you are unable to establish a redundant, active, and automated architecture for all of your components. Examine every layer of your architecture to determine the necessary DR processes to achieve your objectives. This may vary from workload to workload, with different architectural and service requirements. This guide covers considerations and options for Amazon EC2. For other AWS services, you can refer to the [AWS documentation](#) to determine high availability and DR options.

DR for Amazon EC2 in a single Availability Zone

Try to architect your workloads to actively support and service clients from multiple Availability Zones. You can use Amazon EC2 Auto Scaling and Elastic Load Balancing to achieve a Multi-AZ server architecture for Amazon EC2 and other services.

If your architecture has EC2 instances that can't be load balanced and can have only a single instance running at any given moment, you can use either of the following options.

- Create an Auto Scaling group that has a minimum, maximum, and desired size of 1 and is configured for multiple Availability Zones. Create an AMI that can be used to replace the instance if it fails. Make sure that you define the proper automation and configuration so that a newly provisioned instance from the AMI can be automatically configured and provide service. Create a load balancer that points to the Auto Scaling group and is configured for multiple Availability Zones. Optionally, create a Route 53 alias that points to the load balancer endpoint.
- Create a Route 53 record for your active instance and have your clients connect using this record. Create a script that creates a new AMI of your active instance and uses the AMI to provision a new EC2 instance in the stopped state in a separate Availability Zone. Configure the script to run periodically and to terminate the previous stopped instance. If there is an Availability Zone failure, start your backup instance in your alternative Availability Zone. Then update the Route 53 record to point to this new instance.

Test your solution thoroughly by simulating the failure that the solution was designed to protect against. Also consider the updates that your DR solution will need as your workload architecture changes.

DR for Amazon EC2 in a regional failure

Although AWS regional failures are rare, it is possible that an AWS Region could fail at some point in the future. Customers must carefully weigh the complexity, cost, and effort required to establish and maintain a multi-Region DR plan against the benefit. AWS provides features that support multi-Region architectures for global availability, failover, and DR. This guide covers a few of the available features that are specific to backup and recovery for Amazon EC2.

AWS AMIs and Amazon EBS snapshots are regional resources that can be used to provision new instances within a single Region. However, you can copy your snapshots and AMIs to another Region and use them to provision new instances in that Region. To support a regional failure DR plan, you can automate the process of copying AMIs and snapshots to other Regions. AWS Backup and Amazon Data Lifecycle Manager support cross-Region copying as a part of your backup configuration.

You must determine the provisioning, failover, and fallback process to use in the event of an outage. You can use Route 53 with health checks and DNS failover to help support your final solution.

Cleaning up backups

To reduce costs, clean up the backups that are no longer required for recovery or retention purposes. You can use AWS Backup and Amazon Data Lifecycle Manager to automate your retention policy for a portion of your backups. However, even with these tools in place, you still need a cleanup approach for backups that are taken separately.

A tagging strategy is a prerequisite to a cleanup strategy. Use tagging to identify resources that should be cleaned up, notify owners appropriately, and automate your cleanup process. Backups created by AWS have creation dates aligned to them, but tagging is important to correlate backups to your workloads, retention requirements, and restore-point identification.

You can implement a cleanup process for snapshots using automation. For example, you can scan your account for snapshots and determine if the corresponding volumes are in an attached state or an available state. You can further filter the results on a time threshold that you specify. Using the tags attached to the volume, you can automatically send email to snapshot owners, and warning them that their snapshots have been scheduled for deletion. This automated remediation can be implemented by using AWS Config rules, a script using the AWS CLI, or a Lambda function using the AWS SDK.

Systems Manager provides the [AWS-DeleteEBSVolumeSnapshots](#) and [AWS-DeleteSnapshot](#) documents to help you initiate and automate the cleanup of Amazon EBS snapshots. You can also use the AWS CLI and AWS SDK to automate the cleanup of other AWS resources such as Amazon RDS snapshots.

Backup and recovery FAQ

What backup schedule should I select?

Define a backup schedule frequency that aligns to your recovery point objective (RPO). Define a backup time when your workload is under the least amount of load and when user impact can be reduced. Create a point-in-time snapshot whenever you are going to make a significant change to your workload.

Do I need to create backups in my development accounts?

Test potentially breaking changes in your development accounts for your workloads and create backups before performing breaking changes. You might have many more point-in-time recovery (PITR) backups in your development and non-production accounts from development and testing activities.

Next steps

Start by evaluating, implementing, and testing your backup and recovery approach in a non-production environment. It is important to test your recovery process thoroughly and to validate that your restored workloads are operating as expected.

Test the restore process for a single component in your architecture in addition to all components in your architecture. Validate the recovery time for each. Also validate the impact of your backup and restore process on upstream and downstream dependencies. Confirm the impact of any service outage on your upstream dependencies and confirm the downstream impact on your backups.

Additional resources

AWS resources

- [AWS Prescriptive Guidance](#)
- [AWS documentation](#)
- [AWS general reference](#)
- [AWS glossary](#)

AWS services

- [AWS Backup](#)
- [Amazon CloudWatch](#)
- [Amazon CloudWatch Events](#)
- [AWS Config](#)
- [Amazon DynamoDB](#)
- [Amazon EBS](#)
- [Amazon EC2](#)
- [IAM](#)
- [Amazon RDS](#)
- [Amazon S3](#)
- [Amazon S3 Glacier](#)
- [Storage Gateway](#)
- [AWS Systems Manager](#)

Other resources

- [Backup and Recovery with AWS Backup \(solution\)](#)
- [Disaster Recovery of Workloads on AWS: Recovery in the Cloud \(whitepaper\)](#)
- [Disaster Recovery Series \(AWS Architecture blog posts\)](#)
- [DR Plan Checklist](#)
- [Backup and Recovery Approaches Using AWS \(technical paper – archived\)](#)
- [Getting started with AWS Backup](#)
- [AWS Marketplace – Backup and Restore](#)

Document history

The following table describes significant changes to this guide. If you want to be notified about future updates, you can subscribe to an [RSS feed](#).

update-history-change	update-history-description	update-history-date
Updated information (p. 42)	Updated the information about restoring volumes .	August 30, 2022
Updated information and added new section (p. 42)	In the Choosing AWS services for data protection section, added services. Added the section Backup and recovery using AWS Backup . In the Backup and recovery using Amazon S3 and Amazon S3 Glacier section, added information about new Amazon S3 Glacier storage classes. In the Backup and recovery for Amazon EC2 with EBS volumes section, added links to documentation and additional information. In the Backup and recovery of cloud-native AWS services section, added a recommendation to use AWS Backup. In the Additional resources section, added resources.	January 28, 2022
Updated information (p. 42)	Added information about setting storage classes to the S3 Glacier Flexible Retrieval section. Added information about retrieving snapshots to the Amazon EC2 backup and recovery with snapshots and AMIs section.	September 9, 2021
Updated information (p. 42)	In the AWS Backup section, added information about the AWS services that AWS Backup supports.	June 1, 2021
Initial publication (p. 42)	—	July 29, 2020