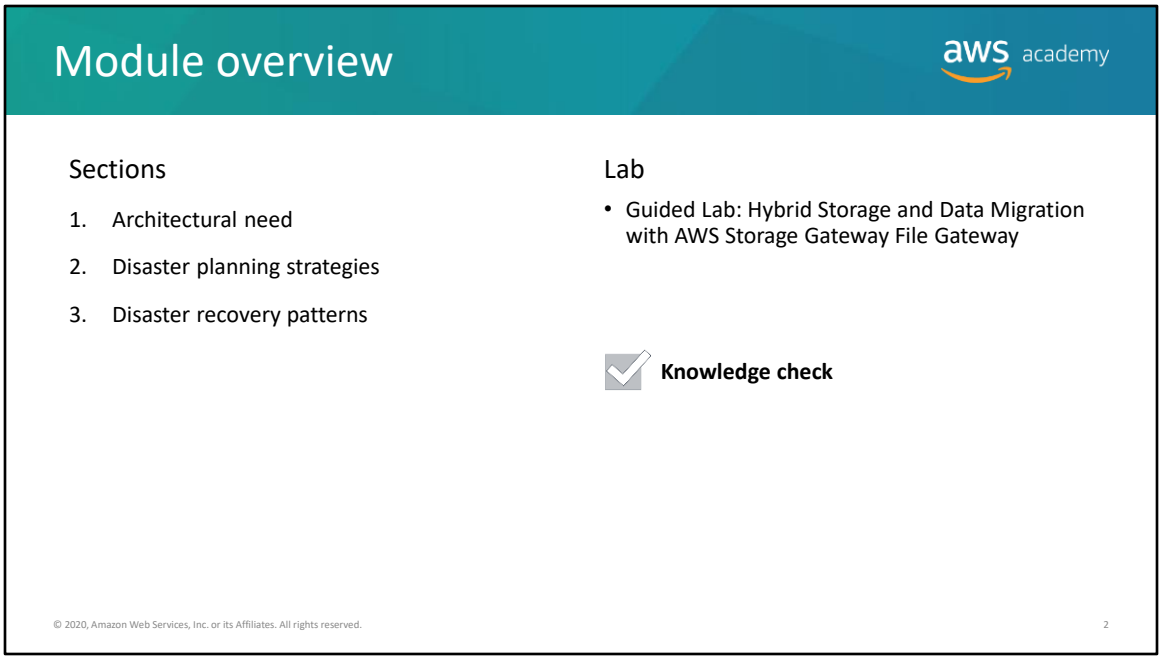
Welcome to Module 14: Planning for Disaster.



This module includes the following sections:

1. Architectural need
2. Disaster planning strategies
3. Disaster recovery patterns

The module also includes a guided lab, where you will enable Amazon S3 cross-Region replication. You will configure a file gateway and mount the file share on an Amazon Elastic Compute Cloud (Amazon EC2) instance.

Finally, you will be asked to complete a knowledge check to test your understanding of key concepts that are covered in this module.

Module objectives

aws academy

At the end of this module, you should be able to:

- Identify strategies for disaster planning
- Define recovery point objective (RPO) and recovery time objective (RTO)
- Describe four common patterns for backup and disaster recovery and how to implement them
- Use AWS Storage Gateway for on-premises-to-cloud backup solutions

At the end of this module, you should be able to:

- Identify strategies for disaster planning
- Define recovery point objective (RPO) and Recovery Time Objective (RTO)
- Describe four common patterns for backup and disaster recovery and how to implement them
- Use AWS Storage Gateway for on-premises-to-cloud backup solutions

Module 14: Planning for Disaster

Section 1: Architectural need

aws academy

Introducing Section 1: Architectural need.

Café business requirement

aws academy

If the café's infrastructure ever becomes unavailable, the staff must be able to get their applications running again within an amount of time that is acceptable to the business. They need an architecture that supports their disaster recovery plans while also optimizing for cost.

By now, the café has implemented several applications that run on AWS. They are also storing a significant amount of business-critical data in the AWS Cloud. Sofía realizes that if the café's infrastructure ever becomes unavailable, they must be able to get their applications running and accessible in an amount of time that's acceptable to the business. Currently, the café's staff hasn't developed any comprehensive disaster recovery plans.

Sofía raised this concern with Frank and Martha. They all agreed that it's important to put backup and disaster recovery plans into place. Their objective is to implement an architecture that supports their disaster recovery time objectives, while it also optimizes for cost. They also agreed that as their revenue grows, they will be able to afford a solution that supports a shorter recovery time objective.

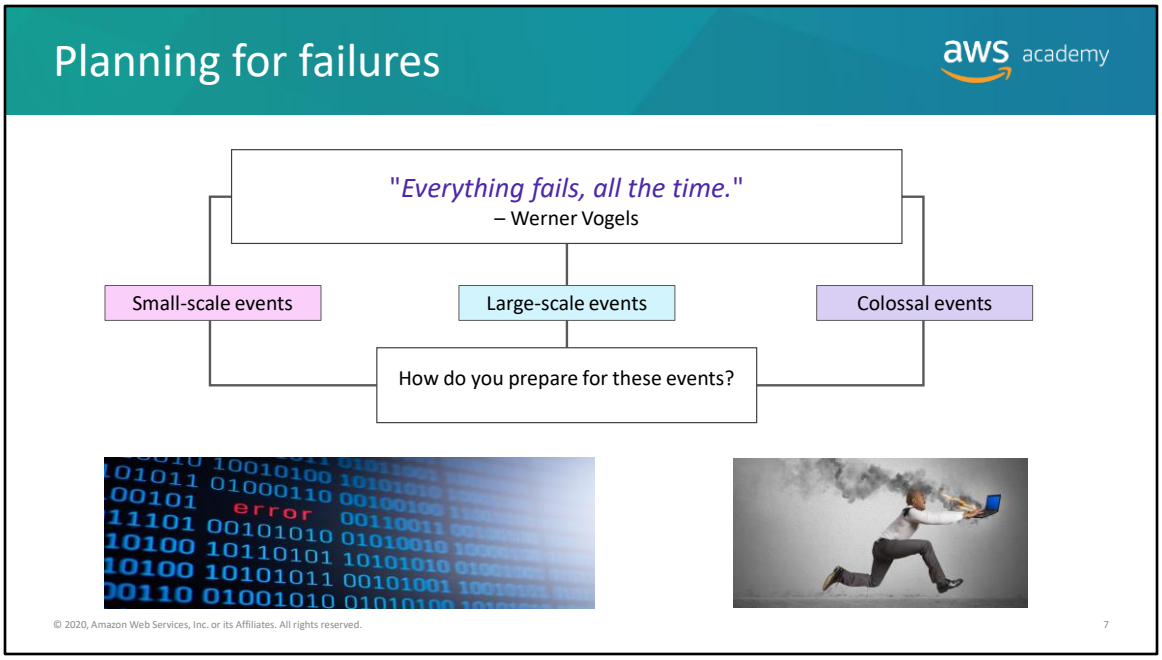
In this module, you will learn about key AWS service features that support data backup and disaster recovery. With an understanding of these features, you should be able to help the café meet this essential business requirement.

Module 14: Planning for Disaster

Section 2: Disaster planning strategies

aws academy

Introducing Section 2: Disaster planning strategies.



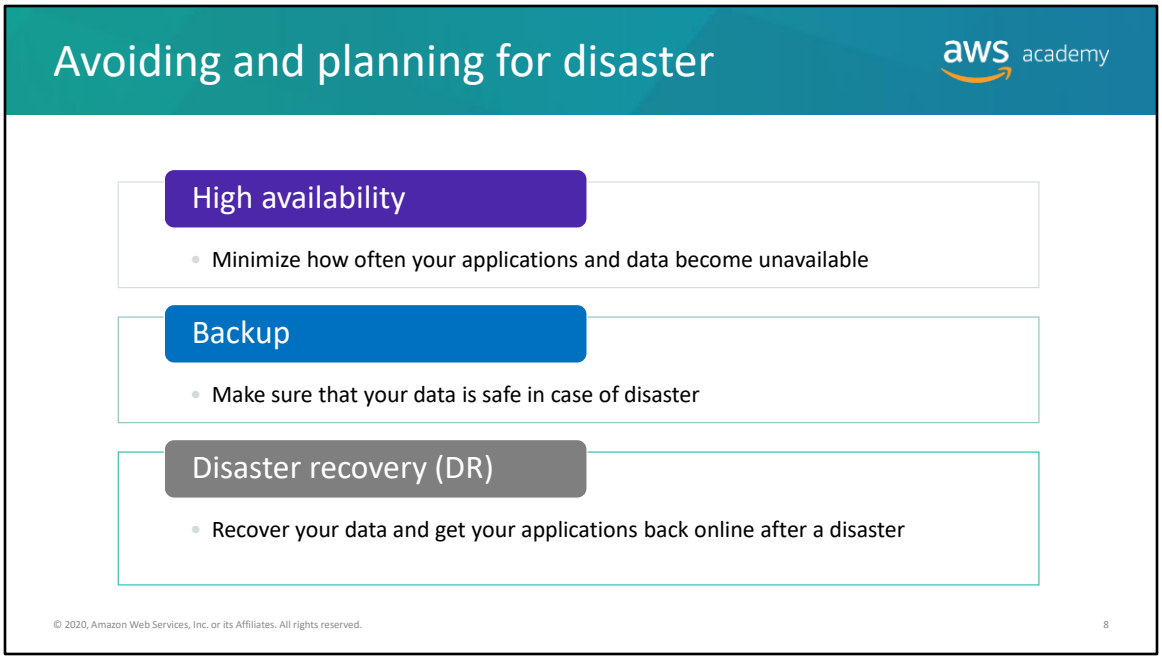
The Chief Technology Officer (CTO) of AWS, Werner Vogels, has famously stated on more than one occasion that, "Everything fails, all the time." His pronouncement has continued to influence cloud computing architectural design for many years, because it speaks to a truism.

Failure should not be thought of as an unlikely aberration. Instead, it should be assumed that failures, both large and small, can—and will—occur. How do you prepare for these events?

A failure can be categorized as one of three types:

- A *small-scale event* – For example, a single server stopped responding or went offline
- A *large-scale event* – In this case, multiple resources were affected, perhaps even across Availability Zones within a Region
- A *colossal scale event* – In this case, the failure is widespread, and it affects a large number of users and systems

To minimize the impact of a disaster, organizations must invest time and resources to plan and prepare, to train employees, and to document and update processes. The amount of investment for disaster planning for a particular system can vary dramatically, depending on the cost of a potential outage.



You can work to avoid and plan for disaster in three ways:

- *High availability* provides redundancy and fault tolerance. A system is highly available when it can withstand failure of an individual or multiple components (for example, hard disks, servers, or network connectivity). Production systems typically have defined uptime requirements.

- *Backup* is critical to protecting data and ensuring business continuity. However, it can be a challenge to implement. The pace at which data is generated is growing exponentially. Meanwhile, the density and durability of local disks are not experiencing the same growth rate. Even so, it is essential to keep your critical data backed up, in case of disaster.

- *Disaster recovery (DR)* is about preparing for and recovering from a disaster. A *disaster* is any event that has a negative impact on a company's business continuity or finances. Such events include hardware or software failure, a network outage, a power outage, or physical damage to a building (like fire or flooding). The cause can be human error, or some other significant event. Disaster recovery is a set of policies and procedures that enable the recovery or continuation of vital technology infrastructure and systems after any disaster.

Selected AWS Well-Architected Framework design principles

aws academy

Operational Excellence pillar

- Anticipate failure
- Refine operational procedures frequently

Reliability pillar

- Test recovery procedures
- Automatically recover from failure

Consider some design principles that relate to the topic of disaster recovery.

The *Operational Excellence* pillar of the AWS Well-Architected Framework states the importance of *anticipating failure*. It recommends that you perform pre-mortem exercises to identify potential sources of failure so that they can be removed or mitigated. You must test your failure scenarios and validate your understanding of their impact. The AWS Well-Architected Framework also describes the benefits of refining your operational procedures frequently so that you can look for opportunities to improve them. Then, as you evolve your workload, you can evolve your procedures accordingly.

The *Reliability* pillar describes the importance of designing your systems. You must be able to recover from infrastructure or service disruptions, and mitigate disruptions such as misconfigurations or transient network issues.

One of the design principles that it mentions is to *test recovery procedures*. Test how your system fails, and validate your recovery procedures. You can use automation to simulate different failures or to recreate scenarios that led to previous failures. This testing exposes failure pathways that you can test and fix before a real failure scenario. It reduces the risk of components that have not been tested before they fail.

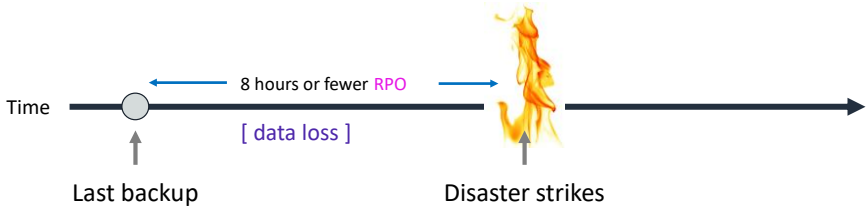
Another principle of design is to *automatically recover from failure*. By monitoring a system for key performance indicators (KPIs), you can trigger automation when a threshold is breached. These KPIs should be a measure of business value, not the technical aspects of how the service operates. Your automation could provide notifications and tracking of failures, and for automated recovery processes that work around or repair the failure.

Recovery point objective (RPO)

Recovery point objective (RPO) is the maximum acceptable amount of data loss, measured in time.

How often must your data be backed up?

Example RPO: *The business can recover from losing (at most) the last 8 hours of data.*



Organizations of all sizes, large and small, often have a Business Continuity Plan (BCP). A typical part of the BCP is to provide for IT Service Continuity, including IT disaster recovery planning.

One of the most important measures of a disaster recovery plan is to define your *recovery point objective (RPO).* To calculate RPO, first determine how much data loss is acceptable, according to your BCP. Then, figure out how quickly that data loss might occur, as a time measurement.

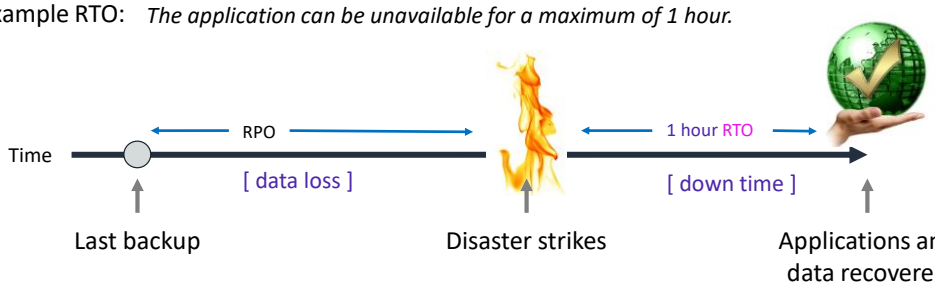
For example, suppose you determine that the data that your application generates is important but not critical, so that losing 800 records would be acceptable. You further calculate that even during peak times, no more than 100 records are created in an hour. In this scenario, you decide that an RPO of 8 hours is sufficient to meet your needs. If you then implement a disaster recovery plan that meets this RPO, you are sure to do data backups at least every 8 hours. Then, if a disaster occurs at 22:00, the system should be able to recover all data that was in the system before 14:00 PM.

Recovery time objective (RTO)

Recovery time objective (RTO) is the maximum acceptable amount of time after disaster strikes that a business process can remain out of commission.

How quickly must your applications and data be recovered?

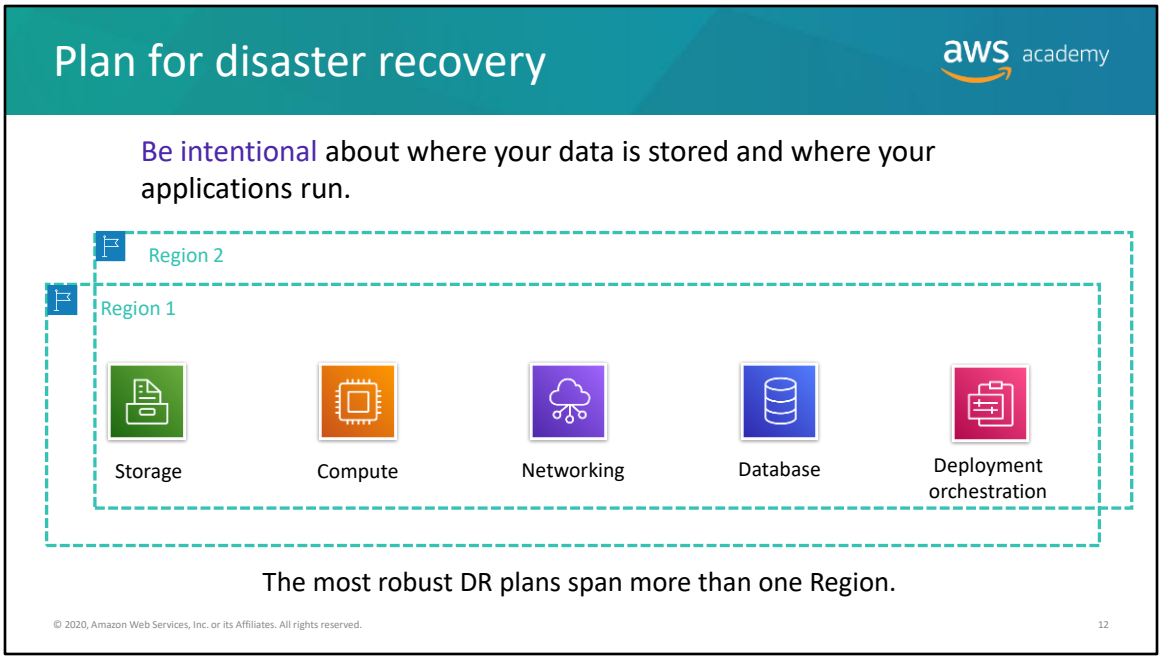
Example RTO: *The application can be unavailable for a maximum of 1 hour.*



Another important measure of a disaster recovery plan is to define the *recovery time objective* (RTO). RTO is the time that it takes *after* a disruption to restore your applications and recover your data. To continue the previous example, suppose a disaster occurs at 22:00 and the RTO is 1 hour. In that scenario, the DR process should restore the business process to the acceptable service level by 23:00.

A company typically decides on acceptable RPO and RTO, and it bases its decision on the financial impact to the business when systems are unavailable. The company determines financial impact by considering many factors. These factors include loss of business and damage to its reputation because of downtime and the lack of systems availability.

IT organizations then plan solutions to provide cost-effective system recovery. The solutions are based on the RPO within the timeline and the service level that the RTO establishes.

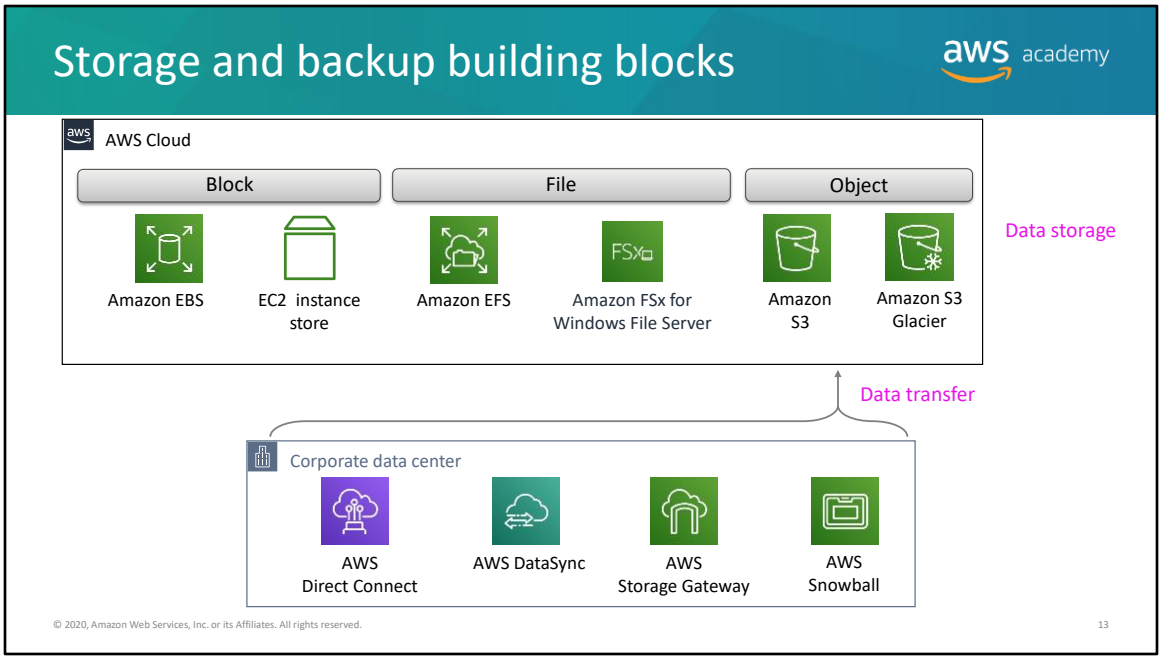


To properly scope your disaster recovery planning, you must look holistically at your use of AWS. Most organizations use a combination of services that can be broadly categorized as encompassing these five service categorizes areas:

- Storage
- Compute
- Networking
- Databases
- Deployment orchestration services

If a disaster occurs, your RPO and RTO will guide your backup-and-restore plans and procedures across each of these service areas. They will also likely affect your production deployment architecture.

It is also important to keep in mind that, although it's unlikely for a Region to be unavailable, it is within the realm of possibility. If some large-scale event affects a Region—for instance, a meteor strike—would your data still be available? Would your applications still be accessible? AWS provides multiple Regions around the world. Thus, you can choose the most appropriate location for your disaster recovery site, in addition to the site where your system is fully deployed.



The following services are referenced in the diagram:

- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Elastic File System (Amazon EFS)
- Amazon Simple Storage Service (Amazon S3)
- Amazon Simple Storage Service Glacier (Amazon S3 Glacier)

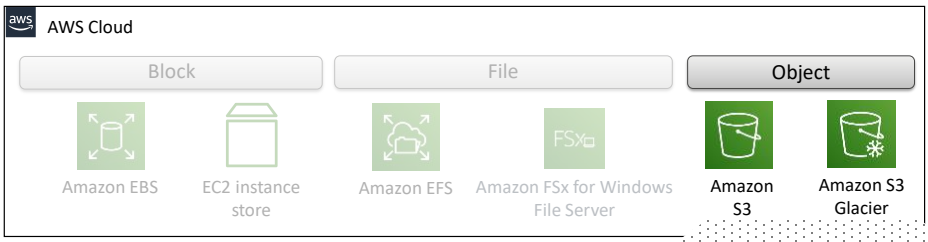
To start your disaster planning in detail, look at the data storage layer (postponing the discussion of database layer for the moment).

Your AWS Cloud storage can consists of a combination of block storage, file system storage, and object storage. Meanwhile, your organization might also use AWS services that connect the on-premises data center to the AWS Cloud.

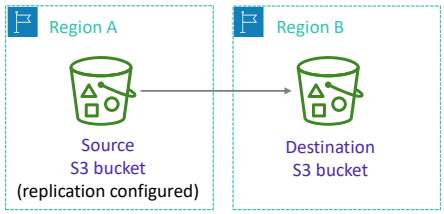
In the next few slides, you will learn about high-level best practices for each of these three areas.

One service that you might not be familiar with is *AWS DataSync*. AWS DataSync provides movement of large amounts of data online between on-premises storage and Amazon S3, Amazon EFS, or Amazon FSx for Windows File Server. It supports scripted copy jobs and scheduled data transfers from on-premises Network File Systems (NFS) and Server Message Block (SMB) storage. It can also optionally use AWS Direct Connect links.

Best practice: S3 Cross-Region Replication



- Most S3 storage classes replicate data across Availability Zones within a single Region
- Configure S3 cross-Region replication for higher-level data security
  - Automatically, asynchronously replicates objects created after you add the replication configuration
  - Can also help meet compliance requirements and reduce latency for users who are accessing objects



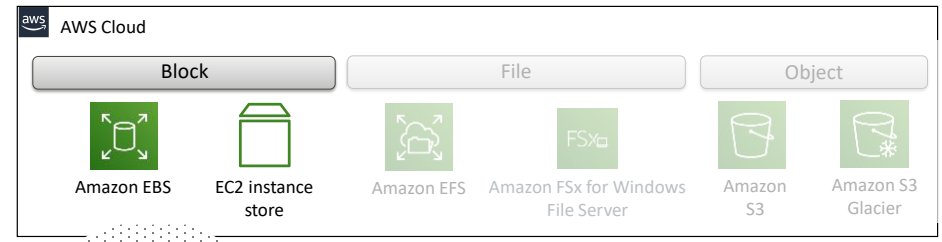
For many organizations, the bulk of their data that is stored on AWS is in Amazon S3, which provides object storage.

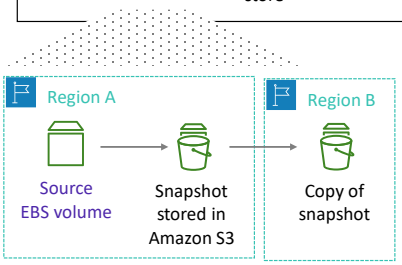
Recall that S3 buckets exist in a specific AWS Region. You choose the Region when you create the bucket. Amazon S3 provides 11 9s (99.999999999 percent) of durability for S3 Standard, S3 Standard-IA, S3 One Zone-IA, and Amazon S3 Glacier storage classes. Amazon S3 Standard, S3 Standard-IA, and Amazon S3 Glacier are all designed to sustain data if an entire Amazon S3 Availability Zone loss occurs. They provide this stability by automatically storing your objects across a minimum of three Availability Zones, each separated miles apart, across a *single* AWS Region.

For critical applications and data scenarios where you want a higher level of data security, it is a best practice to configure S3 cross-Region replication. To enable the replication, you add a replication configuration to your *source bucket*. The minimum configuration must indicate the destination bucket where you want Amazon S3 to replicate all objects, or a subset of all objects. It must also include an AWS Identity and Access Management (IAM) role that grants Amazon S3 permissions to copy the objects to the destination bucket.

Copied objects retain their metadata. The *destination bucket* can belong to another storage class. For example, the contents of an S3 Standard bucket might be replicated to an Amazon S3 Glacier bucket. You can assign different ownership to the objects in the destination bucket. You can also use S3 Replication Time Control (S3 RTC) to replicate your data across different Regions in a predictable time frame. S3 RTC replicates four 9s (99.99 percent) of new objects stored in Amazon S3 within 15 minutes (backed by a service-level agreement).

Best practice: EBS volume snapshots





- Create point-in-time snapshots of EBS volumes
  - Snapshots provide incremental backups (they back up the blocks that changed since the previous snapshot)
- Snapshots enable you to restore data to a new EBS volume
- Use Amazon Data Lifecycle Manager to automate the creation, retention, and deletion of snapshots
- You cannot snapshot instance storage

Regarding to block storage, you can back up the data that is on EBS volumes to Amazon S3 by taking point-in-time *snapshots*. Snapshots are *incremental* backups, which means that it saves only the blocks on the device that have changed since your most recent snapshot. This architecture minimizes the time that is required to create the snapshot, and it saves on storage costs by not duplicating data.

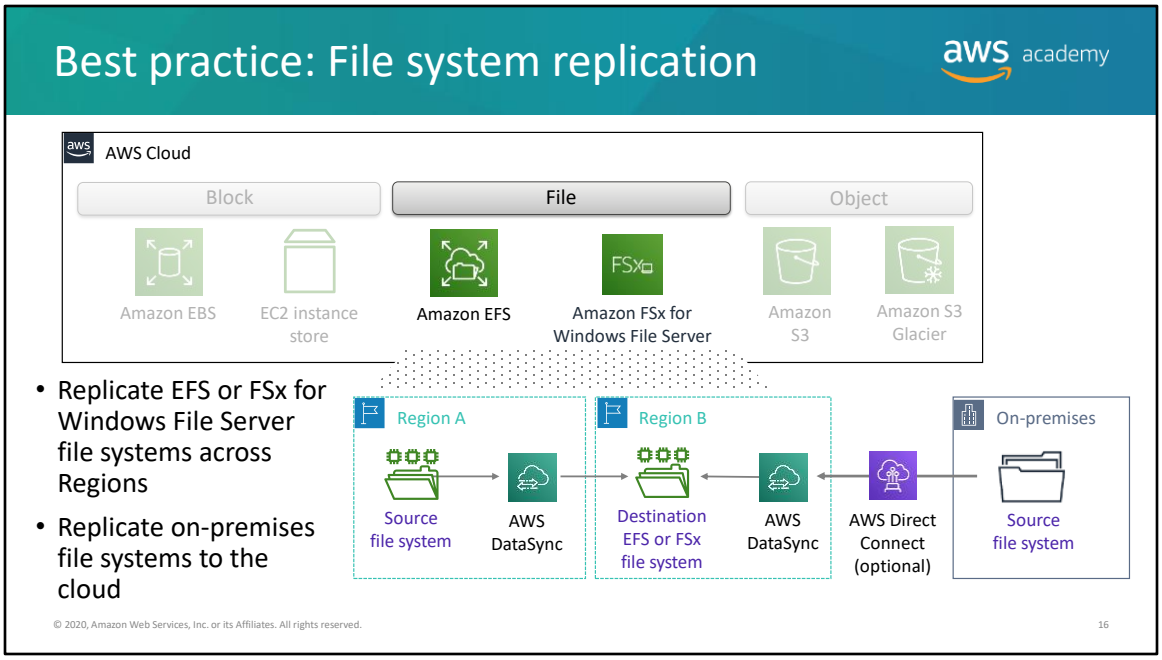
Each snapshot contains all the information that is needed to restore your data (from the moment when the snapshot was taken) to a new EBS volume. When you create an EBS volume that is based on a snapshot, the new volume begins as an exact replica of the original volume. This original volume was used to create the snapshot. The replicated volume loads data in the background so that you can begin to use it immediately. If you access data that has not been loaded yet, the volume immediately downloads the requested data from Amazon S3. Then, it continues to load the rest of the volume's data in the background.

Amazon EBS volumes provide off-instance storage that persists independently from the life of an instance, and is replicated across multiple servers in an Availability Zone. Volumes prevent the loss of data from the failure of any single component. After you create a snapshot, it finishes copying to Amazon S3 (when the snapshot status is completed). Then, you can copy it from one AWS Region to another, or within the same Region.

You can *use Amazon Data Lifecycle Manager to automate* the creation, retention, and deletion of snapshots that back up your EBS volumes. Automating snapshot management helps you to:

- 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

You cannot create snapshots of EC2 instance store volumes. However, if you must back up data from an instance store, you can create a new EBS volume and format it. Then, mount the new volume to the EC2 instance guest OS and copy the data on your instance store volume to the EBS volume. Recall that *instance store* volumes provide temporary block-level storage that works well for information that changes frequently, such as buffers, caches, and scratch data. You might find that you must back up data from an instance store. If so, you might want to rethink why you are storing that data on an instance store volume in the first place.



It is also a best practice to replicate your file storage.

AWS DataSync makes data move faster between two EFS or Amazon FSx Windows File Server file systems, or between on-premises storage and AWS file storage. You can use DataSync to transfer datasets over DX or the internet. Use the service for one-time data migrations or ongoing workflows for data protection and recovery.

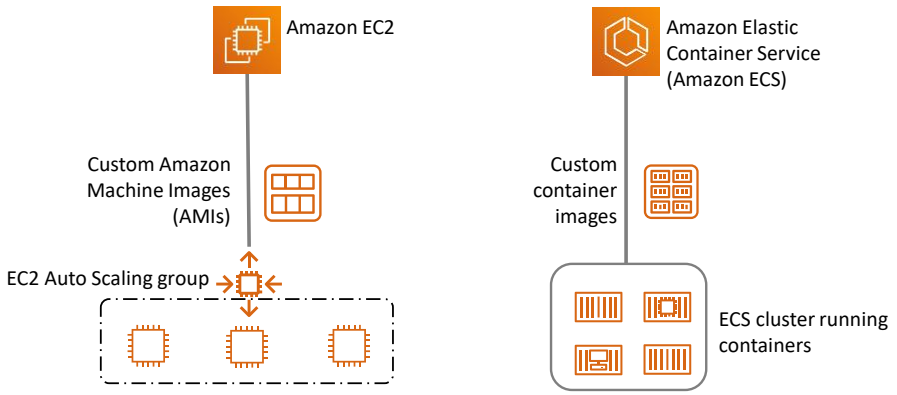
You can learn more about how to use *AWS Backup* to manage EBS volume backups and to automate backups of EFS file systems. See the Scheduling automated backups using Amazon EFS and AWS Backup blog for details.

FSx for Windows File Server takes daily automatic backups of your file systems, and it enables you to take more backups at any point. Amazon FSx stores the backups in Amazon S3. The daily backup window is a 30-minute window that you specify when you create a file system. The daily backup retention period that is specified for your file system determines the number of days that your daily automatic backups are kept. (This number is 7 days by default.)

Like most Amazon S3 storage classes replicate data across Availability Zones, so do Amazon EFS and FSx for Windows File Server file systems. Your disaster recovery requirements might specify that you need a multi-region recovery solution. In that case, it is a best practice to replicate your Amazon EFS and FSx for Windows File Server file systems to a second Region. You can use AWS DataSync to get this replication. To simplify file transfer between two EFS file systems by using DataSync, you can use the AWS DataSync In-Cloud QuickStart and Scheduler.

Compute capacity should be quickly recoverable

Obtain and boot new server instances or containers within minutes.



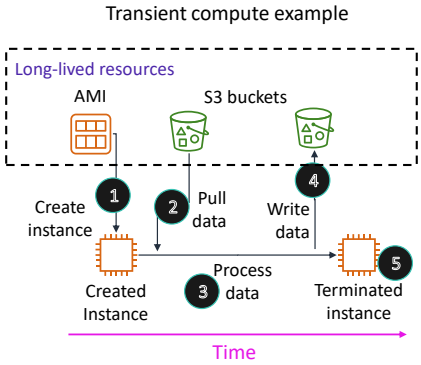
In the context of DR, it's critical that you can rapidly create virtual machines that you control. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location.

You can arrange for automatic recovery of an EC2 instance when a system status check of the underlying hardware fails. The instance is rebooted (on new hardware, if necessary)—but it retains its instance ID, IP addresses, EBS volume attachments, and other configuration details. For a complete recovery, make sure that the instance is configured to automatically start up any services or applications as part of its initialization process.

Amazon Machine Images (AMIs) are preconfigured with operating systems, and some preconfigured AMIs might also include application stacks. You can also configure your own custom AMIs. In the context of DR, AWS recommends that you configure and identify your own AMIs so that they launch as part of your recovery procedure. Such AMIs should be preconfigured with your operating system of choice, in addition to the appropriate pieces of the application stack.

Strategies for compute disaster recovery

- Use the Amazon EC2 snapshot capability for backups
  - Snapshots can be performed manually, or scheduled (for example, by using AWS Lambda)
- Use system or instance level system backups infrequently and as a last resort
  - Drives up the cost of storage that is used quickly
  - Prefer automated rebuild from configuration or code repositories instead
- Cross-region AMI copies
- Cross-region snapshot copies
- Consider transient compute architectures
  - Store essential data off of the instance



For disaster recovery of compute resources, you will probably want to use the Amazon EC2 snapshot capability. Snapshots can be performed manually, or they can be scheduled.

Although you can create system or instance-level system backups, extensive use of this approach increases your storage costs. A better approach is to configure an automated rebuild process, where your source code is stored in a repository.

You might want to replicate Amazon S3 across Regions, and you probably also want to replicate your most critical AMIs and snapshots across Regions.

Finally, consider architecting your use of compute resources to store essential data off of the instances. As you see in the example, your data can be stored in an S3 bucket. When you must do data processing, you can launch one or more EC2 instances from a custom AMI that is preconfigured with application software. As soon as the instance is started, it can pull the needed data from the S3 bucket and process the data. Then, it can write the output data back to Amazon S3 (perhaps to another S3 bucket). After the instance completes its compute tasks, the instance can be terminated. Such an architecture—when it can still meet your business needs—makes it easier to design your disaster recovery strategy. It also can save on costs, because servers that are not in constant use can be terminated and then later re-created when needed.

Networking: Design for resilience, recovery

Amazon Route 53

- Traffic distribution
- Failover

Elastic Load Balancing

- Load balancing
- Health checks and failover

Amazon Virtual Private Cloud (Amazon VPC)

Extend your existing on-premises network topology to the cloud

AWS Direct Connect

Fast, consistent replication and backups of large on-premises environments to the cloud

When you work to recover from a disaster, it's likely that you must modify network settings to fail your system over to another site. AWS offers several services and features that enable you to manage and modify network settings, a few of which are highlighted next.

Amazon Route 53 provides load balancing and network routing capabilities that enable you to distribute network traffic. It also provides the ability to fail over between multiple endpoints and even to a static website that is hosted in Amazon S3.

The *Elastic Load Balancing* service automatically distributes incoming application traffic across multiple EC2 instances. It enables you to achieve fault tolerance in your applications by providing the load-balancing capacity that is needed in response to incoming application traffic. You can pre-allocate a load balancer so that its Domain Name System (DNS) name is already known, which can simplify implementation of your DR plan.

You can use *Amazon Virtual Private Cloud (Amazon VPC)* to extend an existing on-premises network topology to the cloud. This extension can be especially appropriate when you recover enterprise applications that might be hosted on an internal network.

Finally, *AWS Direct Connect* simplifies the setup of a dedicated network connection from an on-premises data center to AWS. Using DX can reduce network costs, increase bandwidth throughput, and provide a more consistent network experience than internet-based connections.

Databases: Features that support recovery

Amazon Relational Database Service (Amazon RDS)

- Take snapshot data and save it in a separate Region
- Combine read replicas with Multi-AZ deployments to build a resilient disaster recovery strategy
- Retain automated backups

Amazon DynamoDB

- Back up entire tables in seconds
- Use point-in-time-recovery to continuously back up tables for up to 35 days
- Initiate backups with a single click in the console or a single application programming interface (API) call
- Use Global Tables to build a multi-region, multi-master database that provides fast local performance for massively scaled globally distributed applications

AWS provides many database services. Some key features of Amazon RDS and Amazon DynamoDB that are relevant to disaster recovery scenarios are explained next.

Consider using *Amazon RDS* in the DR *preparation phase* to store a copy of your critical data in a database that is already running. Then, use Amazon RDS in the DR *recovery phase* to run your production database.

If you implement a multi-region DR plan, Amazon RDS gives you the ability to store snapshot data that was captured from one Region to another Region. You can share a manual snapshot with up to 20 other AWS accounts.

Combining read replicas with Multi-AZ deployments enables you to build a resilient disaster recovery strategy and simplify your database engine upgrade process. By using Amazon RDS read replicas, you can create one or more read-only copies of your database instance. You can create these copies within the same AWS Region, or in a different AWS Region. Updates to the source database are then asynchronously copied to your read replicas. Read replicas can be promoted to become a standalone database instance, when needed.

Use *Amazon DynamoDB* in the preparation phase to copy data to DynamoDB in another Region or to Amazon S3. During the recovery phase of DR, you can scale up in minutes. DynamoDB global tables replicate your DynamoDB tables automatically across your choice of AWS Regions. They resolve update conflicts and enable your applications to stay highly available, even in the unlikely event that an entire Region is isolated or affected by degradation.

Automation services: Quickly replicate or redeploy environments

AWS CloudFormation

- Use templates to quickly deploy collections of resources as needed
- Duplicate production environments in new Region or VPC in minutes

AWS Elastic Beanstalk

- Quickly redeploy your entire stack in only a few clicks

AWS OpsWorks

- Automatic host replacement
- Combine it with AWS CloudFormation in the recovery phase
- Provision a new stack that supports the defined RTO

When you use automation services, you can quickly replicate or redeploy environments.

AWS CloudFormation enables you to model and deploy your entire infrastructure in a text file. This template can become the single source of truth for your infrastructure. When you use AWS CloudFormation to manage your entire infrastructure, it also becomes a powerful tool in your disaster recovery planning toolkit. It enables you to duplicate complex production environments in minutes, for example, to a new Region or a new VPC.

AWS CloudFormation provisions your resources in a repeatable manner, which enables you to build and rebuild your infrastructure and applications. You are not required to perform manual actions or write custom scripts.

If you use *AWS Elastic Beanstalk* to host your applications, you can upload an updated application source bundle and deploy it to your AWS Elastic Beanstalk environment. Alternatively, you can redeploy a previously uploaded version of an application. You can also deploy a previously uploaded version of your application to any of its environments.

Finally, *AWS OpsWorks* is an application management service that makes it easy to deploy and operate applications of all types and sizes. You can define your environment as a series of layers, and configure each layer as a tier of your application. AWS OpsWorks has automatic host replacement, so if you have an instance failure, it is automatically replaced. You can use AWS OpsWorks in the DR *preparation phase* to template your environment and combine it with AWS CloudFormation in the DR recovery phase.

Section 2 key takeaways

- To choose the correct disaster recovery strategy, first identify your recovery point objective (RPO) and recovery time objective (RTO)
- Use features such as S3 Cross-Region Replication, EBS volume snapshots, and Amazon RDS snapshots to protect data
- Use networking features (such as Route 53 failover and Elastic Load Balancing) to improve application availability
- Use automation services (such as AWS CloudFormation) as part of your DR strategy to quickly deploy duplicate environments when needed

Some key takeaways from this section of the module include:

- To choose the correct *disaster recovery* strategy, first identify your recovery point objective (RPO) and recovery time objective (RTO)
- Use features such as *S3 Cross-Region Replication*, *EBS volume snapshots*, and *RDS snapshots* to protect data
- Use networking features—such as *Route 53 failover* and *Elastic Load Balancing*—to improve application availability
- Use automation services—such as *AWS CloudFormation*—as part of your DR strategy to *quickly deploy duplicate environments* when necessary

Module 14: Planning for Disaster

Section 3: Disaster recovery patterns

aws academy

Introducing Section 3: Disaster recovery patterns.

Common disaster recovery patterns on AWS

aws academy

Four disaster recovery patterns

- Backup and restore
- Pilot light
- Warm standby
- Multi-site

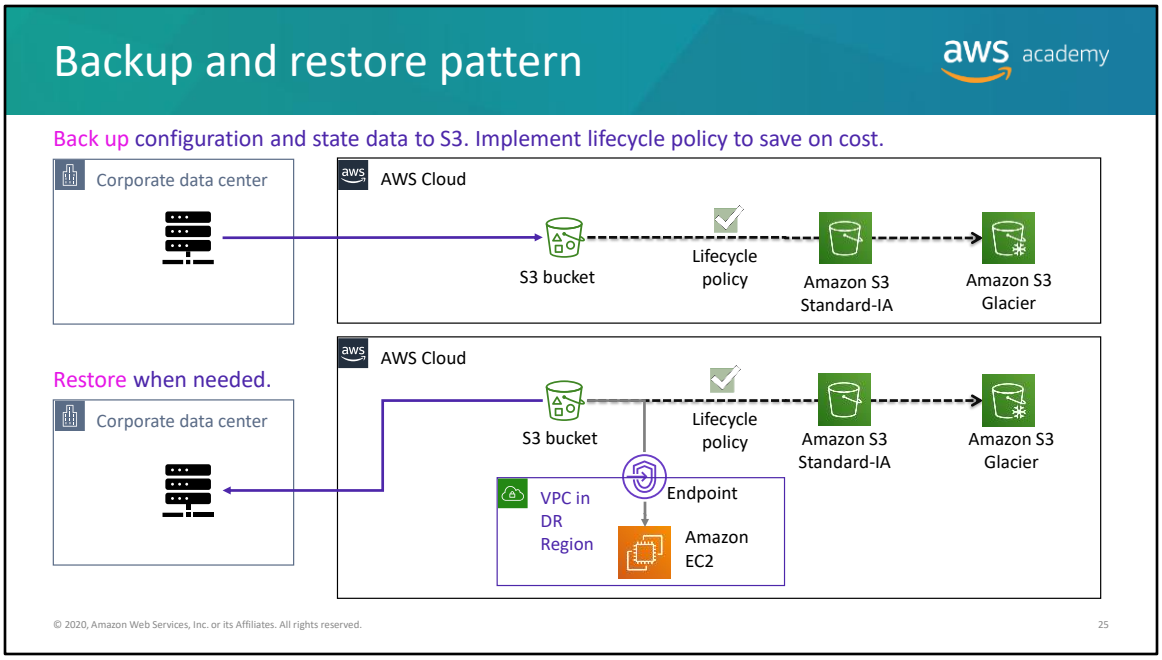
Each pattern is suited to a different combination of:

- Recovery point objective
- Recovery time objective
- Cost-effectiveness

Organizations often use these four common disaster recovery patterns:

- Backup and restore
- Pilot light
- Warm standby
- Multi-site

As you will discover in the details that follow, each pattern is well-suited to different requirements. Some of the patterns offer better cost-effectiveness. Others provide a faster RPO and faster RTO, but cost more to maintain.



The first disaster recovery approach is the *backup and restore pattern*.

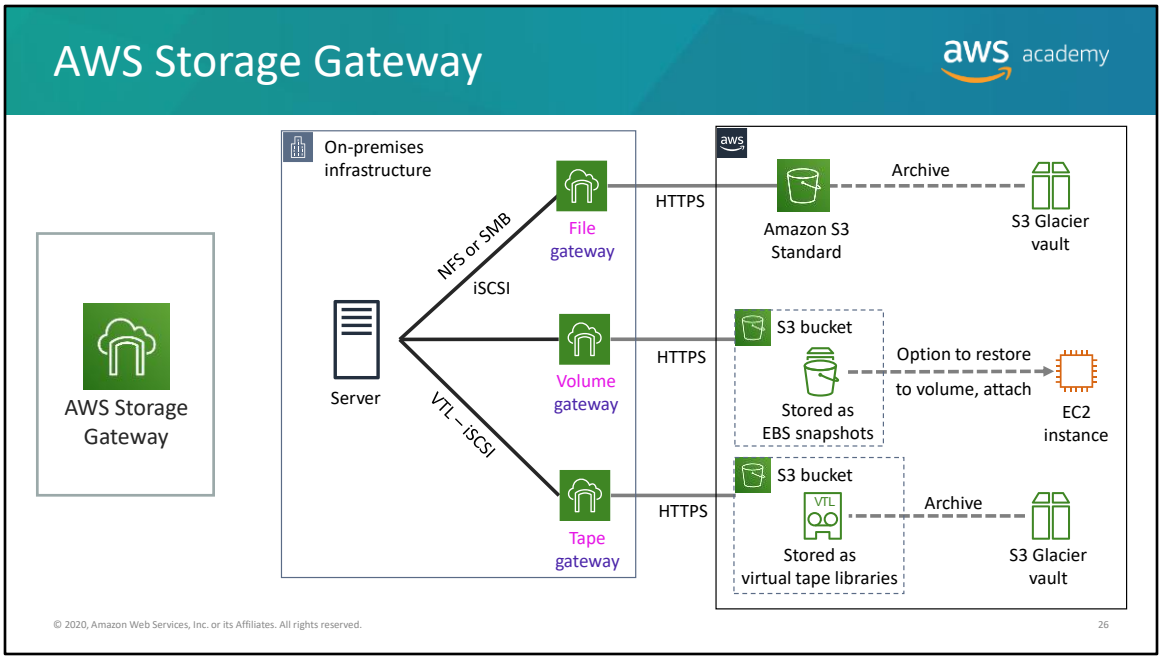
In most traditional environments, data is backed up to tape and sent offsite regularly. If you use this method, it can take a long time to restore your system when a disaster occurs.

Amazon S3 provides a more easily accessible destination for backup data that might be needed quickly to perform a restore. Transferring data to and from Amazon S3 is typically done through the network, and is therefore accessible from any location.

In the example backup scenario, data is copied from the on-premises data center to Amazon S3. AWS DataSync or Amazon S3 Transfer Acceleration can optionally be used as part of this configuration to automate or increase the speed of data transfer. Then, an S3 lifecycle configuration that is applied to the bucket later moves the backup data to less-expensive Amazon S3 storage classes. The backup data moves to Amazon S3 Glacier or Amazon S3 Standard-IA, which saves on cost as the data ages and is not frequently accessed.

In the example restore scenario, the on-premises data might be temporarily or permanently lost. Then, the backup data can be downloaded from Amazon S3 back to the on-premises servers.

If your corporate data center remains offline, you can further ensure the ability to restore your data to your servers. You can have Amazon EC2 servers that are ready to go in a VPC in your designated disaster recovery Region. This Region can connect to the S3 bucket that contains your backup application data. It can read that data, and perhaps temporarily host your applications while you work to restore your data center.



As part of the backup and restore pattern, you might find that it makes sense to use AWS Storage Gateway.

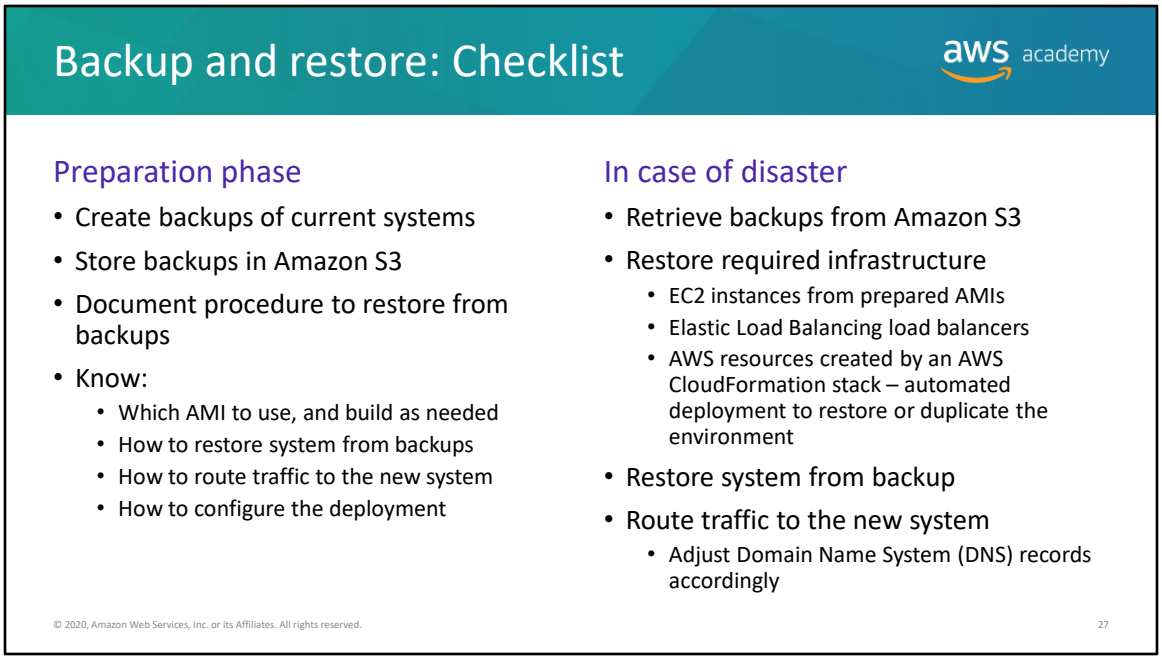
AWS Storage Gateway is a hybrid storage service that enables your on-premises applications to use AWS Cloud storage. You can use the service for backup and archiving, disaster recovery, cloud data processing, storage tiering, and migration.

Your applications connect to the service through a virtual machine or hardware gateway appliance by using standard storage protocols. These protocols include NFS, SMB, Virtual Tape Library (VTL), and Internet Small Computer System Interface (iSCSI). The gateway connects to AWS storage services—such as Amazon S3, Amazon S3 Glacier, and Amazon EBS—which provide storage for files, volumes, and virtual tapes. The service includes an optimized data transfer mechanism. It provides bandwidth management, automated network resilience, and efficient data transfer, in addition to a local cache for low-latency on-premises access to your most active data.

With a *file gateway*, you store and retrieve objects (by using the NFS or SMB protocol) in Amazon S3. You use a local cache for low-latency access to your most recently used data. When your files are transferred to Amazon S3, they are stored as objects and can be accessed through an NFS mount point.

The Storage Gateway *volume* interface presents your applications with block storage disk volumes that can be accessed by using the iSCSI protocol. Data on these volumes is backed up as point-in-time EBS snapshots, which enables you to access it through Amazon EC2, if needed.

The Storage Gateway *tape* interface presents the Storage Gateway to your existing backup application as a virtual tape library. This library consists of a virtual media changer and virtual tape drives. You can continue to use your existing backup applications while you write to a collection of virtual tapes. Each virtual tape is stored in Amazon S3. When you no longer require access to data on virtual tapes, your backup application archives it from the virtual tape library into Amazon S3 Glacier.

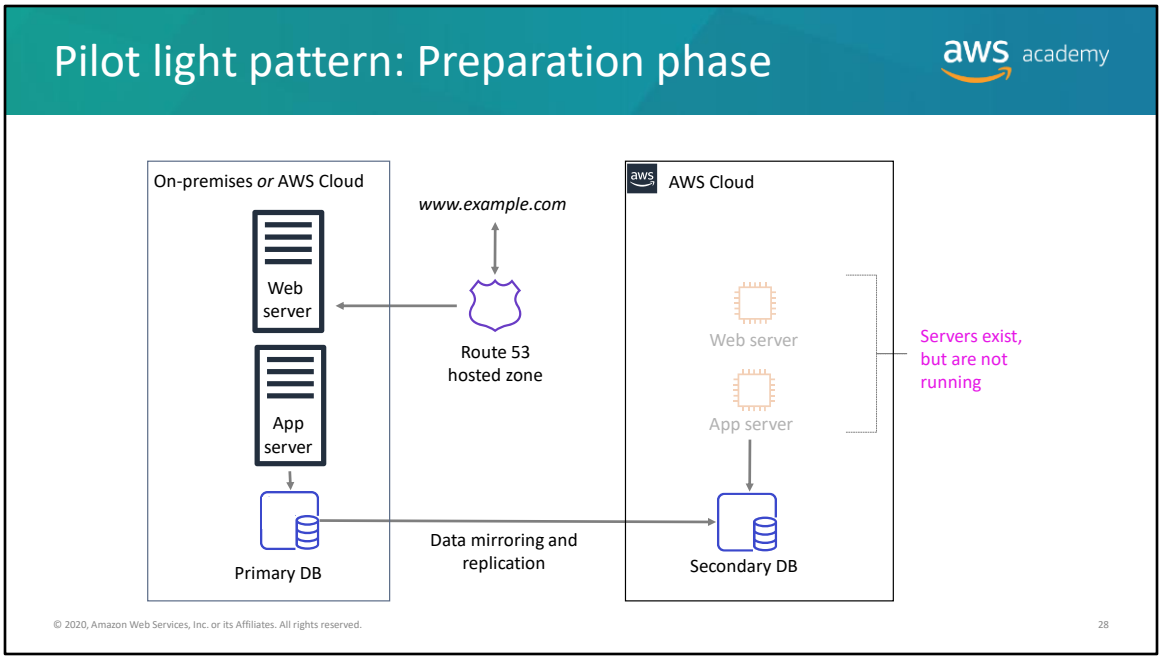


If you implement the backup and restore disaster recovery pattern, the key steps that you should complete during the *preparation phase* are:

- Create backups of current systems
- Store backups in Amazon S3
- Document the procedure to restore from backups

If you implement this pattern, the key steps to complete *in case of disaster* are:

- Retrieve backups from Amazon S3
- Start the required infrastructure
- Restore the system from backups
- Finally, route traffic to the new system



The second disaster recovery approach is the *pilot light pattern*.

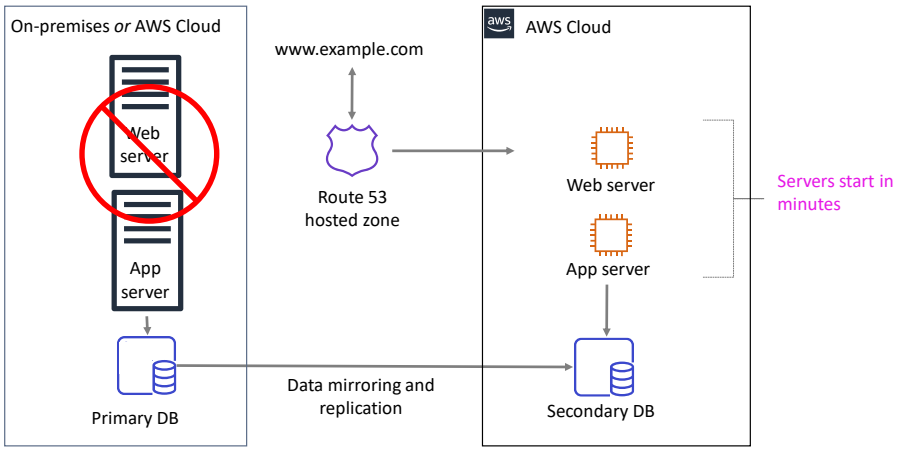
Pilot light describes a disaster recovery pattern where a *minimal* backup version of your environment is *always running*. The pilot light analogy comes from a gas heater: a small flame (or the pilot light) is always on, even when the heater is off. The pilot light can quickly ignite the entire furnace to heat a house. In the example pattern, the pilot light is the secondary database that is always running.

The pilot light scenario is similar to the backup-and-restore scenario. However, recovery time is typically faster because the core pieces of the system are already running and are continually kept up-to-date. When the time comes for recovery, you can rapidly provision a full production environment around the critical core.

Infrastructure elements for the pilot light itself typically include your database servers. This grouping is the critical core of the system (the pilot light). All other infrastructure pieces can quickly be provisioned around it to restore the complete system. To provision the rest of the infrastructure, you typically bundle preconfigured servers as AMIs that are ready to be started at a moment's notice. (Or they might be instances that are in a stopped state.) When recovery begins, these instances start quickly with their pre-defined role, which enables them to connect to the database.

This pattern is relatively inexpensive to implement. Regularly changing data must be replicated to the pilot light, the small core around which the full environment starts in the recovery phase. Your less frequently updated data, such as operating systems and applications, can be periodically updated and stored as AMIs.

Pilot light pattern: In case of disaster



Suppose that disaster strikes, and your primary application goes offline. In this case, you can quickly commission the compute resources to run the application or to orchestrate the failover to pilot light resources in AWS. In this example, the secondary database stores critical data. If there is a disaster, the new web server and app server start up and connect to the secondary database. Amazon Route 53 is configured to then route traffic to the new web server.

The primary environment can exist in an on-premises data center, or in another Region or Availability Zone on AWS. Either way, you can use the pilot light pattern to meet your recovery time objective (RTO).

Pilot light pattern: Checklist

aws academy

Preparation phase

- Configure EC2 instances to replicate or mirror servers
- Ensure that all supporting custom software packages are available on AWS
- Create and maintain AMIs of key servers where fast recovery is needed
- Regularly run these servers, test them, and apply any software updates and configuration changes
- Consider automating the provisioning of AWS resources

In case of disaster

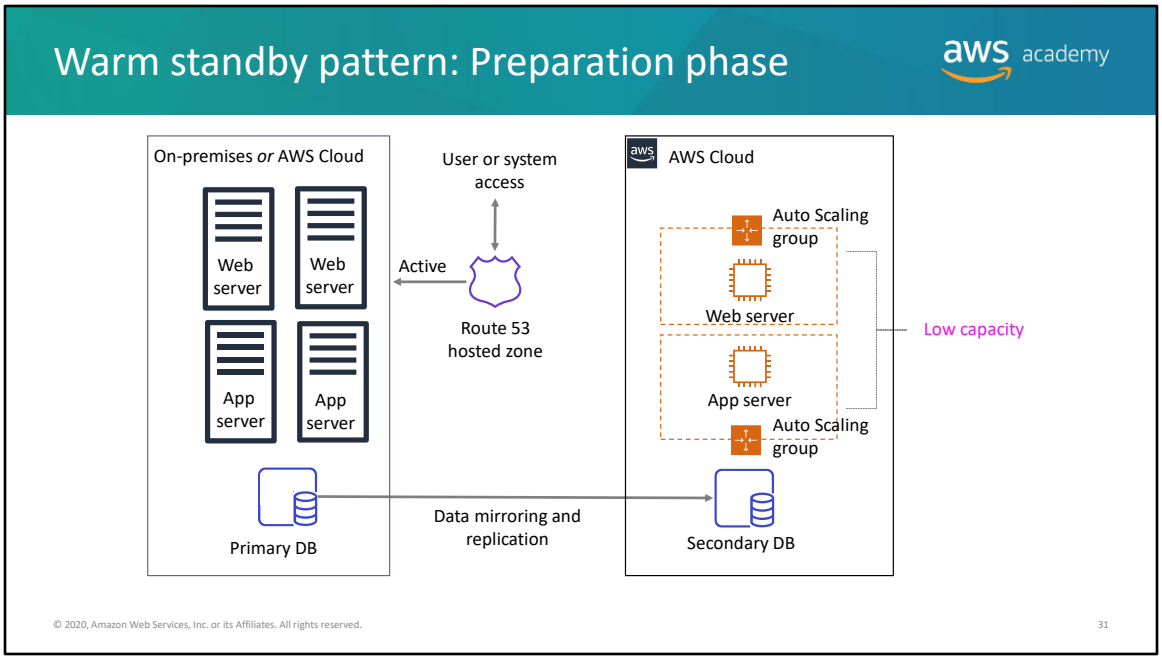
- Automatically bring up resources around the replicated core dataset
- Scale the system as needed to handle current production traffic
- Switch over to the new system
  - Adjust DNS records to point to AWS

If you implement the pilot light disaster recovery pattern, the key steps that you should complete during the *preparation phase* are:

- Configure the EC2 instances
- Ensure that all of the supporting custom software packages are available
- Create and maintain essential AMIs where fast recovery is required
- Regularly run and test servers, and apply software updates and configuration updates
- Consider automating the provisioning of AWS resources

If you implement the pilot light pattern, the key steps to complete *in case of disaster* are:

- Automatically bring up resources around the replicated core dataset
- Scale the system as needed to handle current production traffic
- Switch over to the new system by adjusting the DNS records to point to the backup deployment



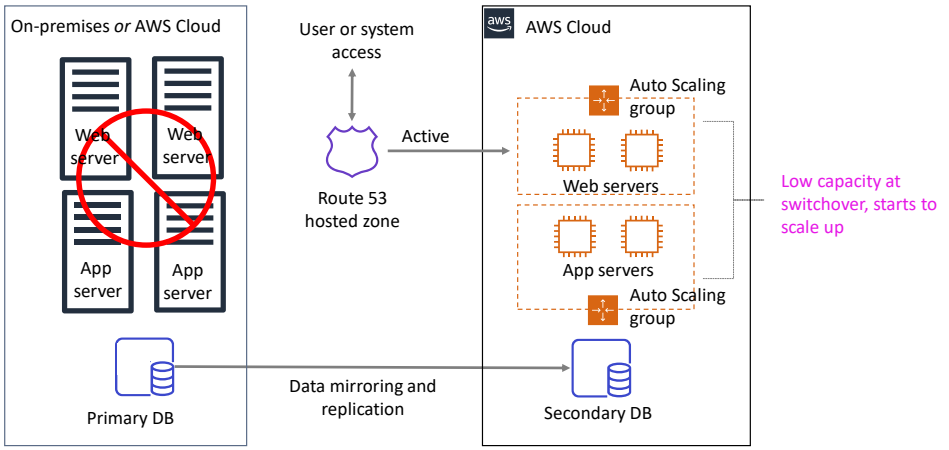
The third disaster recovery approach is the *warm standby pattern*.

The warm standby pattern is like the pilot light, but more resources are already running. The term *warm standby* describes a disaster recovery scenario where a scaled-down version of a fully functional environment is always running in the cloud. The warm standby solution extends the pilot light elements and preparation. It further decreases the recovery time because some services are always running. By identifying your business-critical systems, you can fully duplicate these systems and have them always on.

These servers can be running on a minimum-sized fleet of EC2 instances with the smallest sizes possible. This solution is not yet scaled to take a full production load, but it is fully functional. Though it exists for DR purposes, you can also use it for non-production work, such as testing, quality assurance, and internal use.

In the example, two systems are running. The main system might be running in an on-premises data center or an AWS Region, and a low-capacity system is running on AWS. Use Amazon Route 53 to distribute requests between the main system and the backup system.

Warm standby pattern: In case of disaster



In a disaster, if the primary environment is unavailable, Amazon Route 53 switches over to the secondary system.

The secondary system can then quickly begin to scale up to handle the production load. You can produce this increase by adding more EC2 instances to the load balancer. Alternatively, you can resize the small capacity servers to run on larger EC2 instance types. Horizontal scaling (creating more EC2 instances) is preferred over vertical scaling (increasing the size of existing instances).

Warm standby pattern: Checklist

aws academy

Preparation

- Similar to pilot light
- All necessary components running 24/7, but not scaled for production traffic
- Best practice: Continuous testing
  - Trickle a statistical subset of production traffic to the DR site

In case of disaster

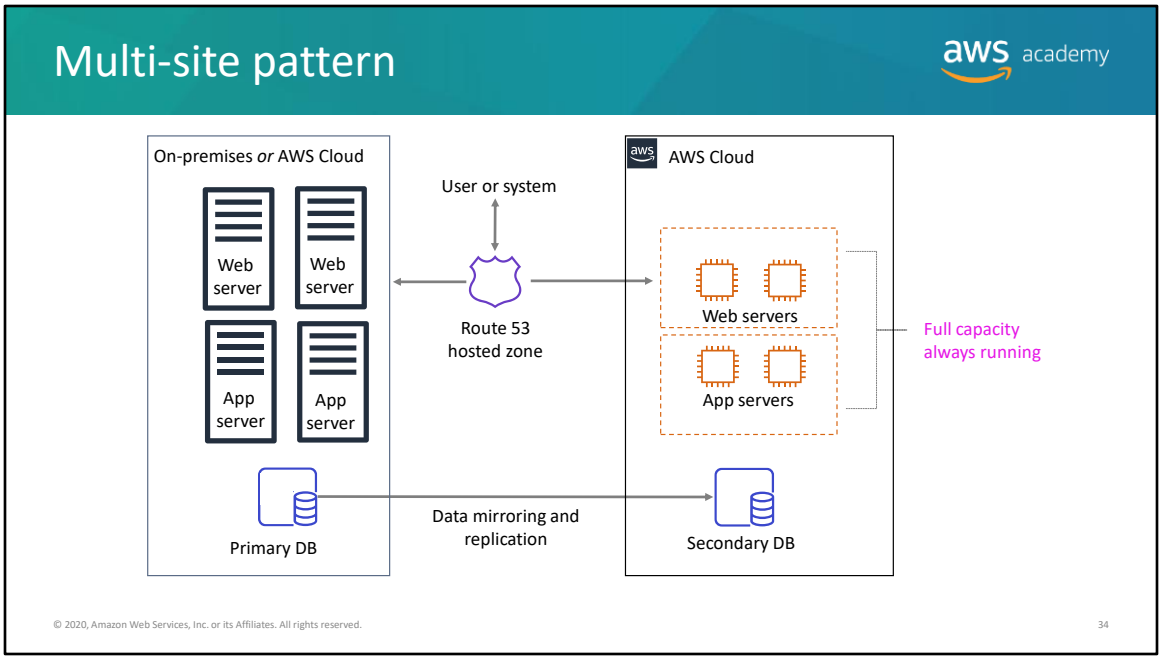
- Immediately fail over most critical production load
  - Adjust DNS records to point to AWS
- (Automatically) Scale the system further to handle all production load

If you implement the warm standby disaster recovery pattern, the *preparation phase* is important. The key steps that you should complete during the preparation phase are similar to the steps that you complete for the pilot light pattern. The most notable difference is that all the necessary components should be left running 24/7, but not scaled for production traffic.

As a best practice, conduct continuous testing. You might also trickle a statistical subset of production traffic to the DR site. Thus, you can verify that it functions for users and systems as seamlessly as the primary system.

With the warm standby pattern, *in case of disaster*, the key steps to complete are:

- Immediately fail over the most critical production load
- Adjust DNS records to point to AWS
- (Automatically) Scale the system further to handle all production load



The fourth and final disaster recovery approach is the *multi-site pattern*. With this pattern, you have a fully functional system that runs in a second Region of AWS. It runs at the same time as the on-premises systems or the systems that run in a different AWS Region.

A multi-site solution runs in an *active-active configuration*. The data replication method that you employ is determined from the recovery point that you choose.

Because both sites can support the full production capacity, you might choose to use a DNS service that supports weighted routing. An example is Amazon Route 53, which routes production traffic to both sites that deliver the same application or service. In this scenario, a proportion of traffic goes to your infrastructure in AWS, and the remainder goes to your on-site infrastructure. (Or if the two environments exist in separate AWS Regions, the traffic is proportioned between these two Regions.)

In an on-site or primary AWS Region disaster situation, you can adjust the DNS weighting and send *all* the traffic to the second deployment. The capacity of the secondary deployment can then be rapidly increased to handle the full production load as needed. You can use Amazon EC2 Auto Scaling to automate this process. You might need some application logic to detect the failure of the primary database services and cut over to the parallel already-running database services.

The cost of this scenario is determined from the volume of production traffic during normal operation. In the recovery phase, you pay for only what you use for the duration that the DR environment is needed at full scale. You can further reduce cost by purchasing Amazon EC2 Reserved Instances for your always-on AWS servers.

Multi-site: Checklist

aws academy

Preparation

- Similar to warm standby
- Configured for full scaling in or scaling out for production load

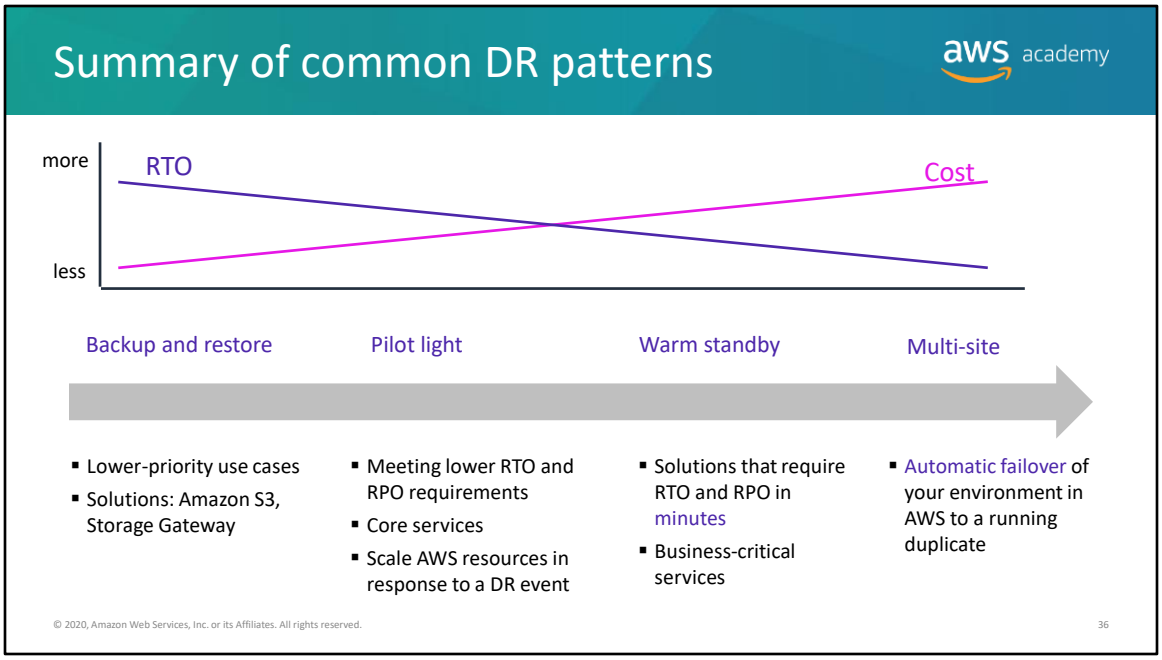
In case of disaster

- Immediately fail over all production load

If you are implementing the multi-site disaster recovery pattern, the key steps to complete during the *preparation phase* are similar to the warm standby pattern. You must configure the backup deployment for full scaling in and out of the production load. You should have the servers running and ready to receive traffic.

With the multi-site pattern, *in case of disaster*, you only need to complete one key step. That step is to immediately fail over all of the production load to the backup site.

The multi-site pattern potentially has the least downtime of all. However, it does have more costs that are associated with it, because more systems are running.



To summarize, each of the four DR patterns offers a different combination of benefits.

The diagram shows a spectrum for the four scenarios, arranged by how quickly a system can be available to users after a DR event.

The backup and restore pattern typically can be accomplished at the lowest cost, but it has a longer RTO. As a result, your systems are likely to be restored more slowly than with the other options.

The warm standby and multi-site patterns support a much faster RTO, but it is costly to have extra servers that are always running.

AWS enables you to cost-effectively operate each of these DR strategies. It's important to realize that these patterns are only examples of possible approaches, and variations and combinations of these patterns are possible. If your application runs on AWS, then you can use multiple Regions, and the same DR strategies still apply.

DR preparation: Best practices

Start simple

Check for software licensing issues

Practice Game Day exercises

Creating a comprehensive disaster recovery plan can be a complex undertaking. However, most organizations recognize—perhaps from past events—that it is worth the effort.

Even though it takes time to develop and implement a full plan, it should not stop you from taking some simple first steps. Start simple, and work your way up. For example, as a first step, create backups of data storage, databases, and critical servers. Then, work to incrementally improve RTO and RPO as a continuous effort.

Software licensing is an issue that can surface while you create backup sites. Look into the software licensing that you have to determine whether your current license contracts support your DR plans. Upgrade your licenses or adjust in other ways as necessary.

Finally, it is a best practice to consistently exercise your DR solution so you can ensure that it works as intended. Some suggested steps include:

- Practice Game Day exercises. These exercises test scenarios when critical systems go offline—or even entire regions. What if an entire fleet crashes?
- Ensure that backups, snapshots, and AMIs are being created, and that they can be used to successfully restore data.
- Monitor your monitoring system.

Test your response procedures to ensure they are effective and that teams are familiar with how to put them into practice. Set up regular Game Days to test workload and team responses to simulated events.

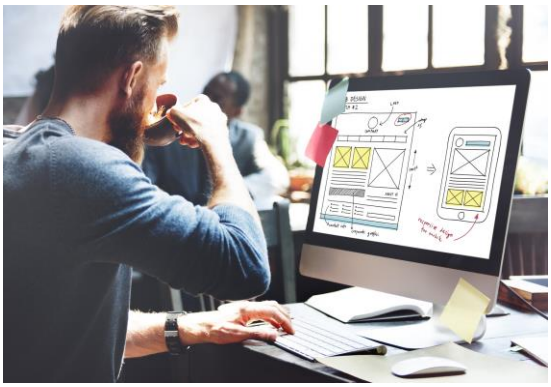
Section 3 key takeaways

- Common disaster recovery patterns on AWS include backup and restore, pilot light, warm standby, and multi-site.
- Backup and restore is the most cost effective approach. However, it has the highest RTO.
- Multi-site provides the fastest RTO. However, it costs the most because it provides a fully running production-ready duplicate.
- AWS Storage Gateway provides three interfaces—file gateway, volume gateway, and tape gateway—for data backup and recovery between on-premises and the AWS Cloud.

Some key takeaways from this section of the module include:

- Common *disaster recovery patterns* on AWS include backup and restore, pilot light, warm standby, and multi-site.
- *Backup and restore* is the most cost effective approach, but it has the highest RTO.
- *Multi-site* provides the fastest RTO, but it costs the most because it provides a fully running production-ready duplicate.
- *AWS Storage Gateway* provides three interfaces—file gateway, volume gateway, and tape gateway—for data backup and recovery between on-premises and the AWS Cloud.

Module 14 – Guided Lab: Hybrid Storage and Data Migration with AWS Storage Gateway File Gateway



You will now complete Module 14 – Guided Lab: Hybrid Storage and Data Migration with AWS Storage Gateway File Gateway.

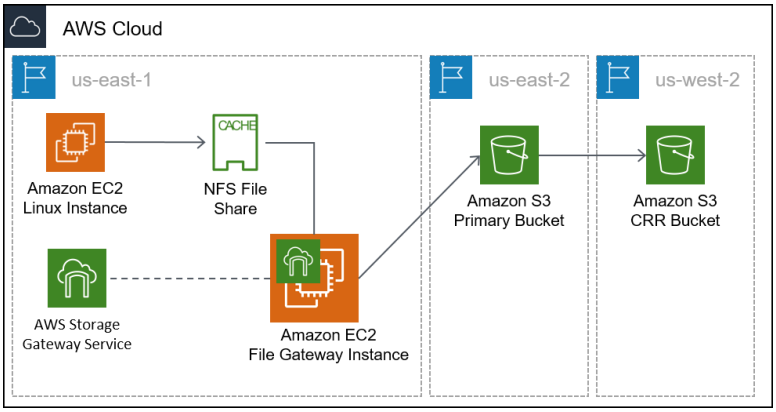
Guided lab: Tasks

1. Reviewing the lab architecture
2. Creating the primary and secondary S3 buckets
3. Enabling Cross-Region Replication
4. Configuring the file gateway and creating an NFS file share
5. Mounting the file share to the Linux instance and migrating the data
6. Verifying that the data is migrated

In this guided lab, you will complete the following tasks:

1. Reviewing the lab architecture
2. Creating the primary and secondary S3 buckets
3. Enabling Cross-Region Replication
4. Configuring the file gateway and creating an NFS file share
5. Mounting the file share to the Linux instance and migrating the data
6. Verifying that the data is migrated

Guided lab: Final product



The diagram summarizes what you will have built after you complete the guided lab. You will have successfully migrated data to Amazon S3 by using the file gateway option in AWS Storage Gateway.

~ 45 minutes

Begin Module 14 – Guided Lab: Hybrid Storage and Data Migration with AWS Storage Gateway File Gateway

It is now time to start the guided lab.

Guided lab debrief: Key takeaways

Your educator might choose to lead a conversation about the key takeaways from this guided lab after you have completed it.

Module 14: Planning for Disaster

Module wrap-up

aws academy

It’s now time to review the module and wrap up with a knowledge check and discussion of a practice certification exam question.

Module summary

aws academy

In summary, in this module, you learned how to:

- Identify strategies for disaster planning
- Define RPO and RTO
- Describe four common patterns for backup and disaster recovery and how to implement them
- Use AWS Storage Gateway for on-premises-to-cloud backup solutions

In summary, in this module, you learned how to:

- Identify strategies for disaster planning
- Define RPO and RTO
- Describe four common patterns for backup and disaster recovery and how to implement them
- Use AWS Storage Gateway for on-premises-to-cloud backup solutions

Complete the knowledge check

It is now time to complete the knowledge check for this module.

Sample exam question

Company salespeople upload their sales figures daily. A Solutions Architect needs a durable storage solution for these documents that also protects against users accidentally deleting important documents.

Which action will protect against unintended user actions?

A. Store data in an EBS volume and create snapshots once a week.
B. Store data in an S3 bucket and enable versioning.
C. Store data in two S3 buckets in different AWS Regions.
D. Store data on EC2 instance storage.

Look at the answer choices and rule them out based on the keywords that were previously highlighted.

The correct answer is B: "Store data in an S3 bucket and enable versioning." With this approach, if a versioned object is deleted, it can still be recovered by retrieving the final version.

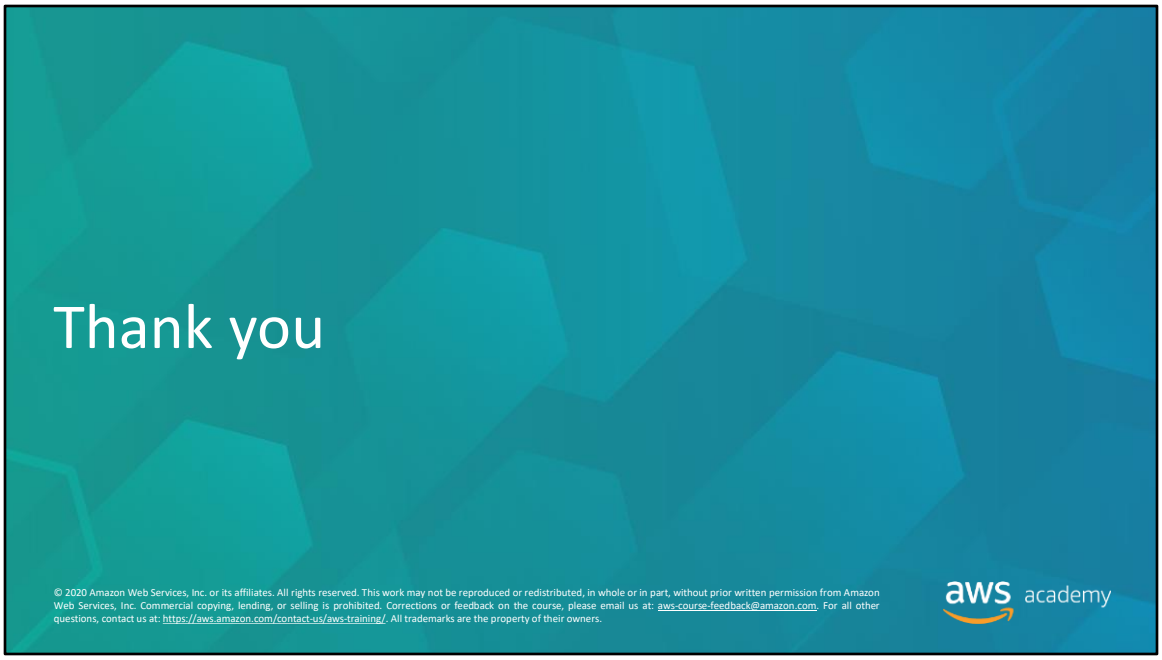
Response A would lose any changes that were committed since the previous snapshot. Response C, storing the data in two S3 buckets, would provide slightly more protection that response A. However, a user might still delete the object from both buckets. Response D is not a good approach, because EC2 instance storage is ephemeral and should never be used for data that requires durability.

Additional resources

- Amazon S3 Replication
- Amazon S3 Object Lifecycle Management
- Amazon EBS Snapshots
- Using AWS Lambda with Scheduled Events
- Backup & Restore resource center
- Disaster Recovery with AWS (video)

If you want to learn more about the topics that are covered in this module, you might find the following resources helpful:

- Amazon S3 Replication
- Amazon S3 Object Lifecycle Management
- Amazon EBS Snapshots
- Using AWS Lambda with Scheduled Events
- Backup & Restore resource center
- Disaster Recovery with AWS (video)



Thank you for completing this module.