AWS Well-Architected Framework

Disaster Recovery of Workloads on AWS: Recovery in the Cloud

Disaster Recovery of Workloads on AWS: Recovery in the Cloud: AWS Well-Architected Framework

Table of Contents

Disaster Recovery of Workloads on AWS: Recovery in the Cloud

Publication date: **February 12, 2021** (*Document history*)

Disaster recovery is the process of preparing for and recovering from a disaster. An event that prevents a workload or system from fulfilling its business objectives in its primary deployed location is considered a disaster. This paper outlines the best practices for planning and testing disaster recovery for any workload deployed to AWS, and offers different approaches to mitigate risks and meet the Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for that workload.

This whitepaper covers how to implement disaster recovery for workloads on AWS. Refer to Disaster Recovery of On-Premises Applications to AWS for information about using AWS as a disaster recovery site for on-premises workloads.

Introduction

Your workload must perform its intended function correctly and consistently. To achieve this, you must architect for *resiliency*. Resiliency is the ability of a workload to recover from infrastructure, service, or application disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions, such as misconfigurations or transient network issues.

Disaster recovery (DR) is an important part of your resiliency strategy and concerns how your workload responds when a disaster strikes (a disaster is an event that causes a serious negative impact on your business). This response must be based on your organization's business objectives which specify your workload's strategy for avoiding loss of data, known as the Recovery Point Objective (RPO), and reducing downtime where your workload is not available for use, known as the Recovery Time Objective (RTO). You must therefore implement resilience in the design of your workloads in the cloud to meet your recovery objectives (RPO and RTO) for a given one-time disaster event. This approach helps your organization to maintain business continuity as part of Business Continuity Planning (BCP).

This paper focuses on how to plan for, design, and implement architectures on AWS that meet the disaster recovery objectives for your business. The information shared here is intended for those in technology roles, such as chief technology officers (CTOs), architects, developers, operations team members, and those tasked with assessing and mitigating risks.

Disaster recovery and availability

Disaster recovery can be compared to *availability*, which is another important component of your resiliency strategy. Whereas disaster recovery measures objectives for one-time events, availability objectives measure mean values over a period of time.

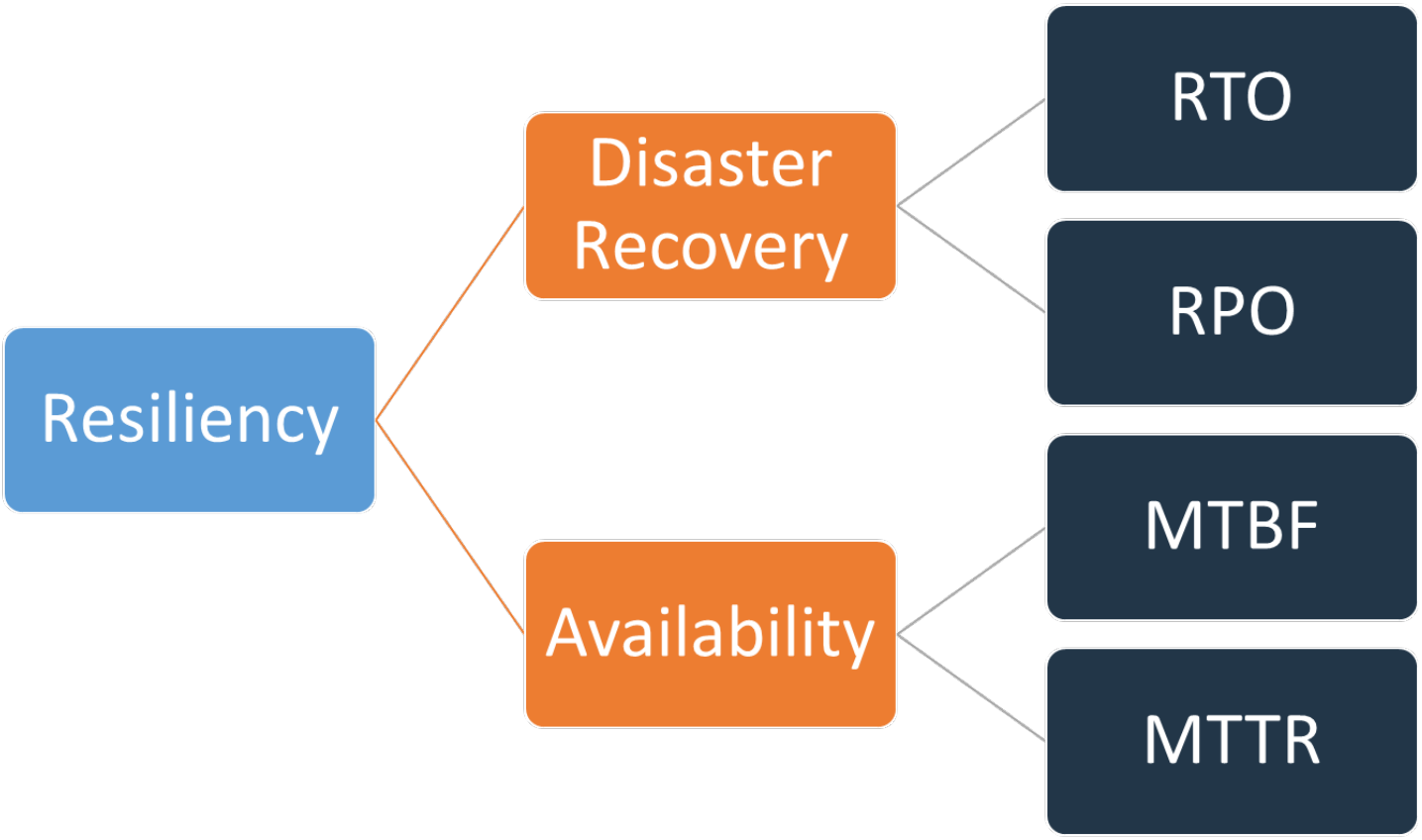

Figure 1 - Resiliency Objectives

Availability is calculated using Mean Time Between Failures (MTBF) and Mean Time to Recover (MTTR):

$$Availability = \frac{Available\ for\ Use\ Time}{Total\ Time} = \frac{MTBF}{MTBF + MTTR}$$

This approach is often referred to as “nines”, where a 99.9% availability target is referred to as “three nines”.

For your workload, it may be easier to count successful and failed requests instead of using a time-based approach. In this case, the following calculation can be used:

$$Availability = \frac{Successful\ Responses}{Valid\ Requests}$$

Disaster recovery focuses on disaster events, whereas availability focuses on more common disruptions of smaller scale such as component failures, network issues, software bugs, and load spikes. The objective of disaster recovery is business continuity, whereas availability concerns maximizing the time that a workload is available to perform its intended business functionality. Both should be part of your resiliency strategy.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

The concepts covered in this whitepaper expand on the best practices contained in the Reliability Pillar whitepaper, specifically question REL 13, "How do you plan for disaster recovery (DR)?". After implementing the practices in this whitepaper, be sure to review (or re-review) your workload using the AWS Well-Architected Tool.

Shared Responsibility Model for Resiliency

Resiliency is a shared responsibility between AWS and you, the customer. It is important that you understand how disaster recovery and availability, as part of resiliency, operate under this shared model.

AWS responsibility "Resiliency of the Cloud"

AWS is responsible for resiliency of the infrastructure that runs all of the services offered in the AWS Cloud. This infrastructure comprises the hardware, software, networking, and facilities that run AWS Cloud services. AWS uses commercially reasonable efforts to make these AWS Cloud services available, ensuring service availability meets or exceeds AWS Service Level Agreements (SLAs).

The AWS Global Cloud Infrastructure is designed to enable customers to build highly resilient workload architectures. Each AWS Region is fully isolated and consists of multiple Availability Zones, which are physically isolated partitions of infrastructure. Availability Zones isolate faults that could impact workload resilience, preventing them from impacting other zones in the Region. But at the same time, all zones in an AWS Region are interconnected with high-bandwidth, low-latency networking, over fully redundant, dedicated metro fiber providing high-throughput, low-latency networking between zones. All traffic between zones is encrypted. The network performance is sufficient to accomplish synchronous replication between zones. When an application is partitioned across AZs, companies are better isolated and protected from issues such as power outages, lightning strikes, tornadoes, hurricanes, and more.

Customer responsibility "Resiliency in the Cloud"

Your responsibility will be determined by the AWS Cloud services that you select. This determines the amount of configuration work you must perform as part of your resiliency responsibilities. For example, a service such as Amazon Elastic Compute Cloud (Amazon EC2) requires the customer to perform all of the necessary resiliency configuration and management tasks. Customers that deploy Amazon EC2 instances are responsible for deploying EC2 instances across multiple locations (such as AWS Availability Zones), implementing self-healing using services like Amazon EC2 Auto Scaling, as well as using resilient workload architecture best practices for applications installed on the instances. For managed services, such as Amazon S3 and Amazon DynamoDB, AWS operates the infrastructure layer, the operating system, and platforms, and customers access the endpoints

to store and retrieve data. You are responsible for managing resiliency of your data including backup, versioning, and replication strategies.

Deploying your workload across multiple Availability Zones in an AWS Region is part of a high availability strategy designed to protect workloads by isolating issues to one Availability Zone, and uses the redundancy of the other Availability Zones to continue serving requests. A Multi-AZ architecture is also part of a DR strategy designed to make workloads better isolated and protected from issues such as power outages, lightning strikes, tornadoes, earthquakes, and more. DR strategies may also make use of multiple AWS Regions. For example in an active/passive configuration, service for the workload will fail over from its active region to its DR region if the active Region can no longer serve requests.

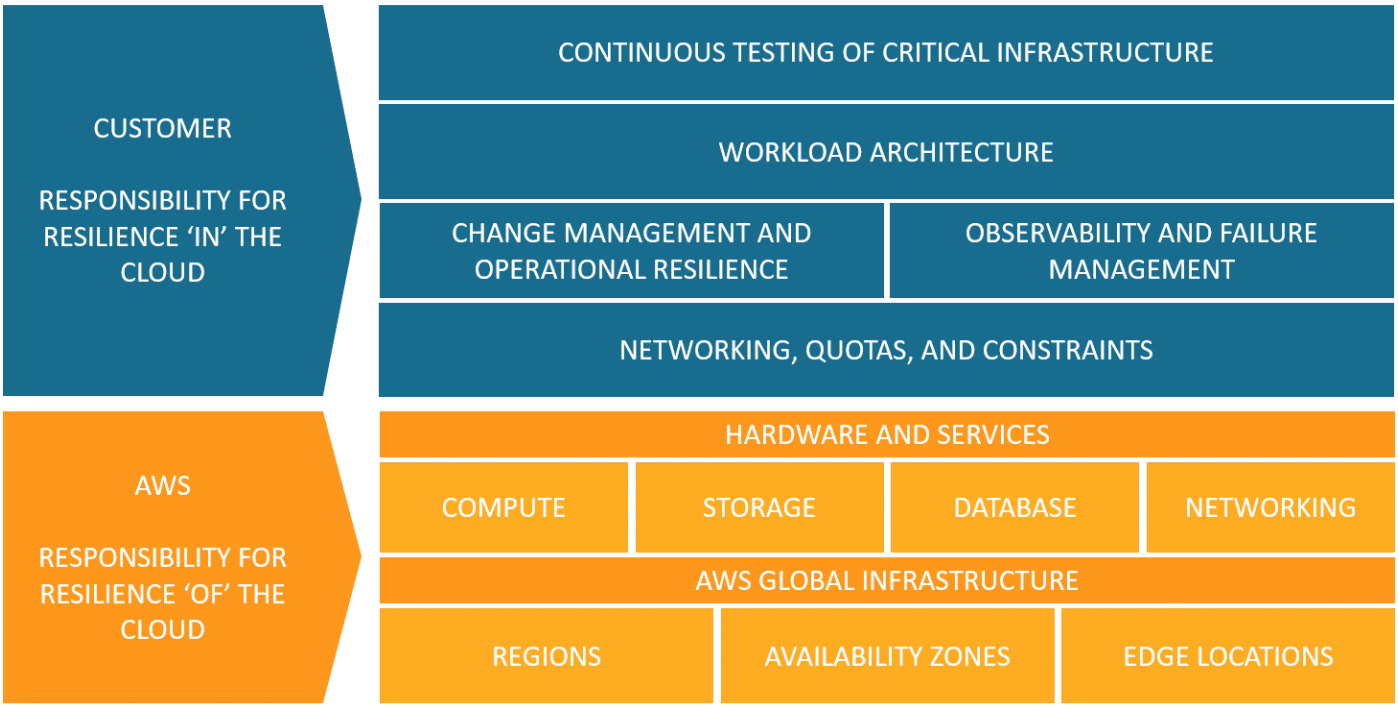

Figure 2 - Resiliency is a shared responsibility between AWS and the customer

What is a disaster?

When planning for disaster recovery, evaluate your plan for these three main categories of disaster:

- Natural disasters, such as earthquakes or floods
- Technical failures, such as power failure or network connectivity
- Human actions, such as inadvertent misconfiguration or unauthorized/outside party access or modification

Each of these potential disasters will also have a geographical impact that can be local, regional, country-wide, continental, or global. Both the nature of the disaster and the geographical impact are important when considering your disaster recovery strategy. For example, you can mitigate a local flooding issue causing a data center outage by employing a Multi-AZ strategy, since it would not affect more than one Availability Zone. However, an attack on production data would require you to invoke a disaster recovery strategy that fails over to backup data in another AWS Region.

High availability is not disaster recovery

Both availability and disaster recovery rely on some of the same best practices, such as monitoring for failures, deploying to multiple locations, and automatic failover. However, Availability focuses on components of the workload, whereas disaster recovery focuses on discrete copies of the entire workload. Disaster recovery has different objectives from Availability, measuring time to recovery after the larger scale events that qualify as disasters. You should first ensure your workload meets your availability objectives, as a highly available architecture will enable you to meet customers' needs in the event of availability impacting events. Your disaster recovery strategy requires different approaches than those for availability, focusing on deploying discrete systems to multiple locations, so that you can fail over the entire workload if necessary.

You must consider the availability of your workload in disaster recovery planning, as it will influence the approach you take. A workload that runs on a single Amazon EC2 instance in one Availability Zone does not have high availability. If a local flooding issue affects that Availability Zone, this scenario requires failover to another AZ to meet DR objectives. Compare this scenario to a highly available workload deployed multi-site active/active, where the workload is deployed across multiple active Regions and all Regions are serving production traffic. In this case, even in the unlikely event a massive disaster makes a Region unusable, the DR strategy is accomplished by routing all traffic to the remaining Regions.

How you approach data is also different between availability and disaster recovery. Consider a storage solution that continuously replicates to another site to achieve high availability (such as a multi-site, active/active workload). If a file or files are deleted or corrupted on the primary storage device, those destructive changes can be replicated to the secondary storage device. In this scenario, despite high availability, the ability to fail over in case of data deletion or corruption will be compromised. Instead, a point-in-time backup is also required as part of a DR strategy.

Business Continuity Plan (BCP)

Your disaster recovery plan should be a subset of your organization's *business continuity plan* (BCP), it should not be a standalone document. There is no point in maintaining aggressive disaster recovery targets for restoring a workload if that workload's business objectives cannot be achieved because of the disaster's impact on elements of your business *other* than your workload. For example an earthquake might prevent you from transporting products purchased on your eCommerce application – even if effective DR keeps your workload functioning, your BCP needs to accommodate transportation needs. Your DR strategy should be based on business requirements, priorities, and context.

Business impact analysis and risk assessment

A *business impact analysis* should quantify the business impact of a disruption to your workloads. It should identify the impact on internal and external customers of not being able to use your workloads and the effect that has on your business. The analysis should help to determine how quickly the workload needs to be made available and how much data loss can be tolerated. However, it is important to note that recovery objectives should not be made in isolation; the probability of disruption and cost of recovery are key factors that help to inform the business value of providing disaster recovery for a workload.

Business impact may be time dependent. You may want to consider factoring this in to your disaster recovery planning. For example, disruption to your payroll system is likely to have a very high impact to the business just before everyone gets paid, but it may have a low impact just after everyone has already been paid.

A *risk assessment* of the type of disaster and geographical impact along with an overview of the technical implementation of your workload will determine the probability of disruption occurring for each type of disaster.

For highly critical workloads, you might consider deploying infrastructure across multiple Regions with data replication and continuous backups in place to minimize business impact. For less critical workloads, a valid strategy may be not to have any disaster recovery in place at all. And for some disaster scenarios, it is also valid not to have any disaster recovery strategy in place as an informed decision based on a low probability of the disaster occurring. Remember that Availability Zones within an AWS Region are already designed with meaningful distance between them, and careful planning of location, such that most common disasters should only impact one zone and not the

others. Therefore, a multi-AZ architecture within an AWS Region may already meet much of your risk mitigation needs.

The cost of the disaster recovery options should be evaluated to ensure that the disaster recovery strategy provides the correct level of business value considering the business impact and risk.

With all of this information, you can document the threat, risk, impact and cost of different disaster scenarios and the associated recovery options. This information should be used to determine your recovery objectives for each of your workloads.

Recovery objectives (RTO and RPO)

When creating a Disaster Recovery (DR) strategy, organizations most commonly plan for the Recovery Time Objective (RTO) and Recovery Point Objective (RPO).

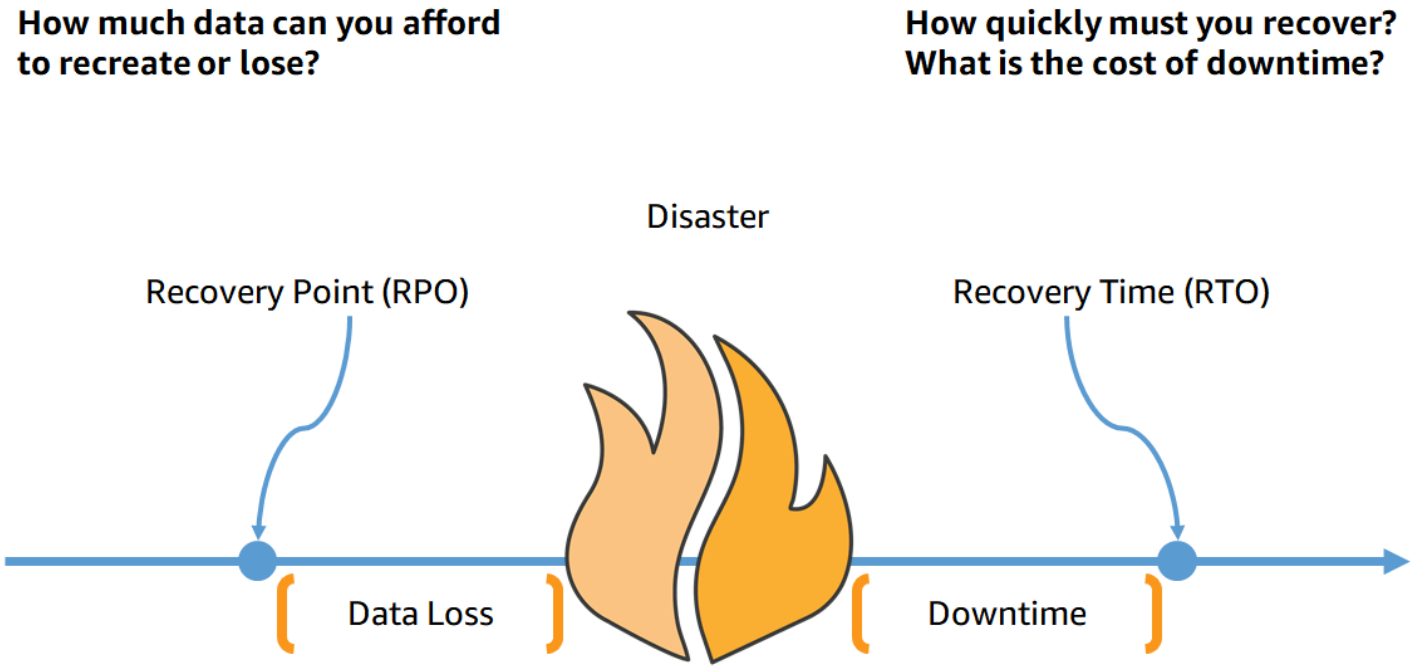

Figure 3 - Recovery objectives

Recovery Time Objective (RTO) is the maximum acceptable delay between the interruption of service and restoration of service. This objective determines what is considered an acceptable time window when service is unavailable and is defined by the organization.

There are broadly four DR strategies discussed in this paper: backup and restore, pilot light, warm standby, and multi-site active/active (see Disaster Recovery Options in the Cloud). In the following

diagram, the business has determined their maximum permissible RTO as well as the limit of what they can spend on their service restoration strategy. Given the business' objectives, the DR strategies Pilot Light or Warm Standby will satisfy both the RTO and the cost criteria.

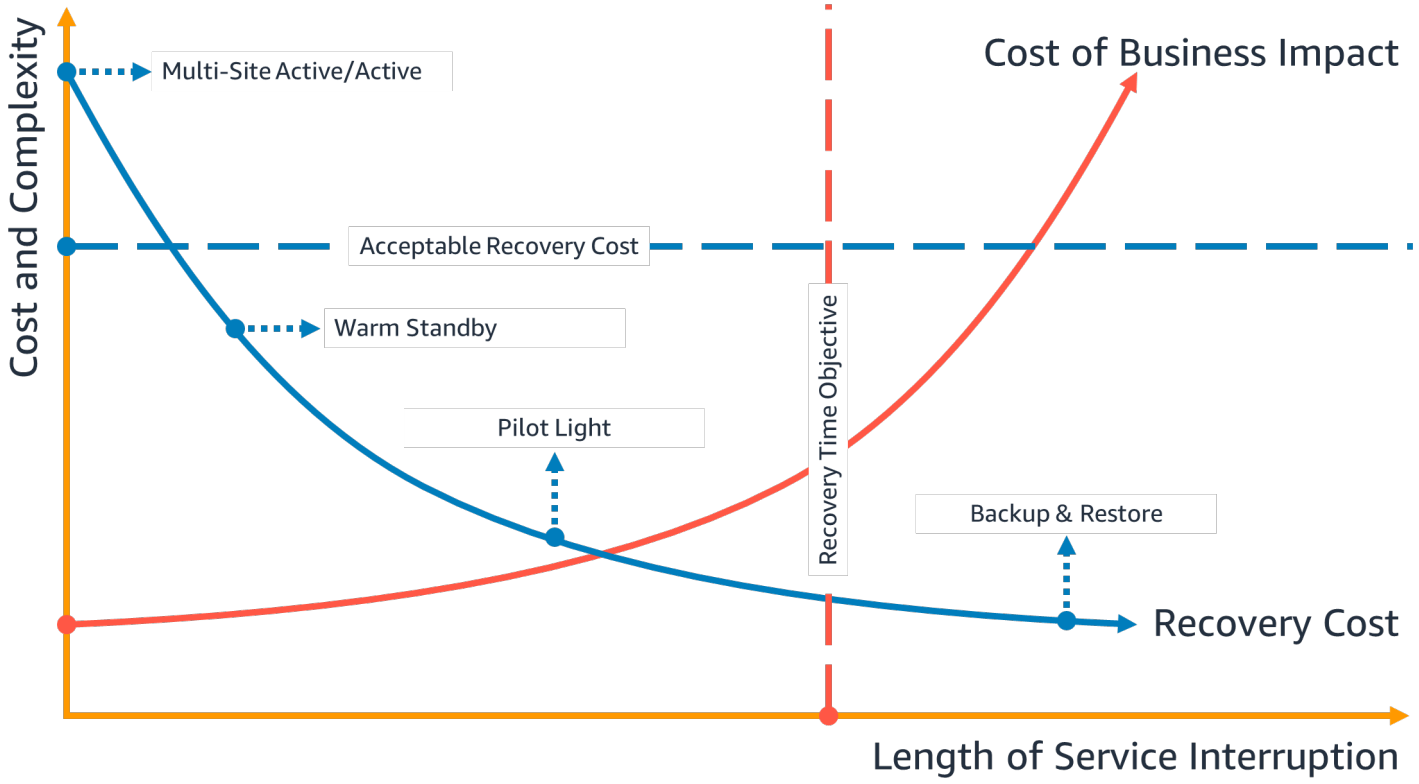

Figure 4 - Recovery time objective

Recovery Point Objective (RPO) is the maximum acceptable amount of time since the last data recovery point. This objective determines what is considered an acceptable loss of data between the last recovery point and the interruption of service and is defined by the organization.

In the following diagram, the business has determined their maximum permissible RPO as well as the limit of what they can spend on their data recovery strategy. Of the four DR strategies, either Pilot Light or Warm Standby DR strategy meet both criteria for RPO and cost.

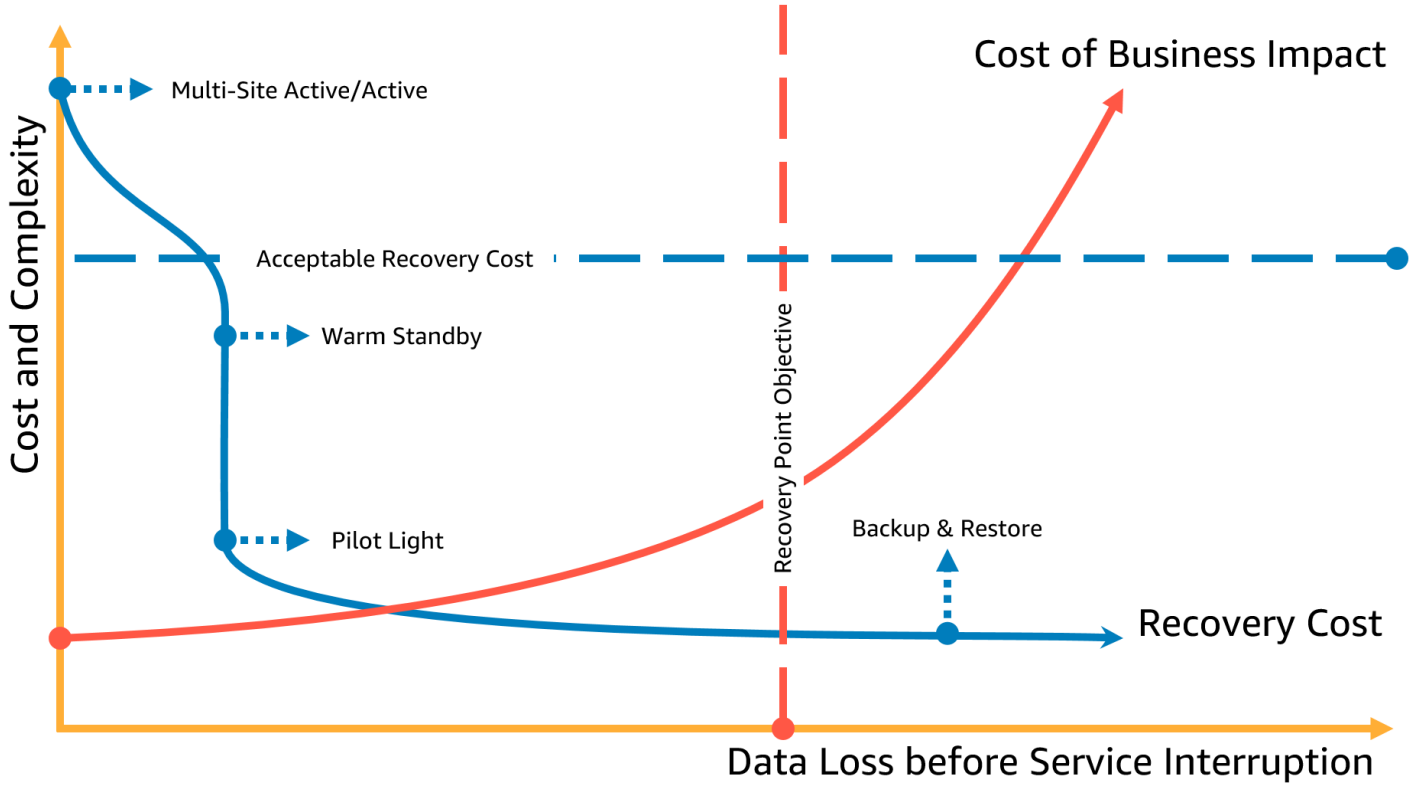

Figure 5 - Recovery point objective

> **Note**
>
> If the cost of the recovery strategy is higher than the cost of the failure or loss, the recovery option should not be put in place unless there is a secondary driver such as regulatory requirements. Consider recovery strategies of varying cost when making this assessment.

Disaster recovery is different in the cloud

Disaster recovery strategies evolve with technical innovation. A disaster recovery plan on-premises may involve physically transporting tapes or replicating data to another site. Your organization needs to re-evaluate the business impact, risk, and cost of its previous disaster recovery strategies in order to fulfill its DR objectives on AWS. Disaster recovery in the AWS Cloud includes the following advantages over traditional environments:

- Recover quickly from a disaster with reduced complexity
- Simple and repeatable testing allow you to test more easily and more frequently
- Lower management overhead decreases operational burden
- Opportunities to automate decrease chances of error and improve recovery time

AWS allows you to trade the fixed capital expense of a physical backup data center for the variable operating expense of a rightsized environment in the cloud, which can significantly reduce cost.

For a lot of organizations, on-premises disaster recovery was based around the risk of disruption to a workload or workloads in a data center and the recovery of backed up or replicated data to a secondary data center. When organizations deploy workloads on AWS, they can implement a well-architected workload and rely on the design of the AWS Global Cloud Infrastructure to help mitigate the effect of such disruptions. See the AWS Well-Architected Framework - Reliability Pillar whitepaper for more information on architectural best practices for designing and operating reliable, secure, efficient, and cost-effective workloads in the cloud. Use the AWS Well-Architected Tool to review your workloads periodically to ensure that they follow the best practices and guidance of the Well-Architected Framework. The tool is available at no charge in the AWS Management Console.

If your workloads are on AWS, you don't need to worry about data center connectivity (with the exception of your ability to access it), power, air conditioning, fire suppression and hardware. All of this is managed for you and you have access to multiple fault-isolated Availability Zones (each made up of one or more discrete data centers).

Single AWS Region

For a disaster event based on disruption or loss of one physical data center, implementing a highly available workload in multiple Availability Zones within a single AWS Region helps mitigate against

natural and technical disasters. Continuous backup of data within this single Region can reduce the risk to human threats, such as an error or unauthorized activity that could result in data loss. Each AWS Region is comprised of multiple Availability Zones, each isolated from faults in the other zones. Each Availability Zone in turn consists of one or more discrete physical data centers. To better isolate impactful issues and achieve high availability, you can partition workloads across multiple zones in the same Region. Availability Zones are designed for physical redundancy and provide resilience, allowing for uninterrupted performance, even in the event of power outages, Internet downtime, floods, and other natural disasters. See AWS Global Cloud Infrastructure to discover how AWS does this.

By deploying across multiple Availability Zones in a single AWS Region, your workload is better protected against failure of a single (or even multiple) data centers. For extra assurance with your single-Region deployment, you can back up data and configuration (including infrastructure definition) to another Region. This strategy reduces the scope of your disaster recovery plan to only include data backup and restoration. Leveraging multi-region resiliency by backing up to another AWS Region is simple and inexpensive relative to the other multi-Region options described in the following section. For example, backing up to Amazon Simple Storage Service (Amazon S3) gives you access to immediate retrieval of your data. However if your DR strategy for portions of your data has more relaxed requirements for retrieval times (from minutes to hours), then using Amazon S3 Glacier or Amazon S3 Glacier Deep Archive will significantly reduce costs of your backup and recovery strategy.

Some workloads may have regulatory data residency requirements. If this applies to your workload in a locality that currently has only one AWS Region, then in addition to designing multi-AZ workloads for high availability as discussed above, you can also use the AZs within that Region as discrete locations, which can be helpful for addressing data residency requirements applicable to your workload within that Region. The DR strategies described in the following sections use multiple AWS Regions, but can also be implemented using Availability Zones instead of Regions.

Multiple AWS Regions

For a disaster event that includes the risk of losing multiple data centers a significant distance away from each other, you should consider disaster recovery options to mitigate against natural and technical disasters that affect an entire Region within AWS. All of the options described in the following sections can be implemented as multi-Region architectures to protect against such disasters.

Disaster recovery options in the cloud

Disaster recovery strategies available to you within AWS can be broadly categorized into four approaches, ranging from the low cost and low complexity of making backups to more complex strategies using multiple active Regions. Active/passive strategies use an active site (such as an AWS Region) to host the workload and serve traffic. The passive site (such as a different AWS Region) is used for recovery. The passive site does not actively serve traffic until a failover event is triggered.

It is critical to regularly assess and test your disaster recovery strategy so that you have confidence in invoking it, should it become necessary. Use AWS Resilience Hub to continuously validate and track the resilience of your AWS workloads, including whether you are likely to meet your RTO and RPO targets.

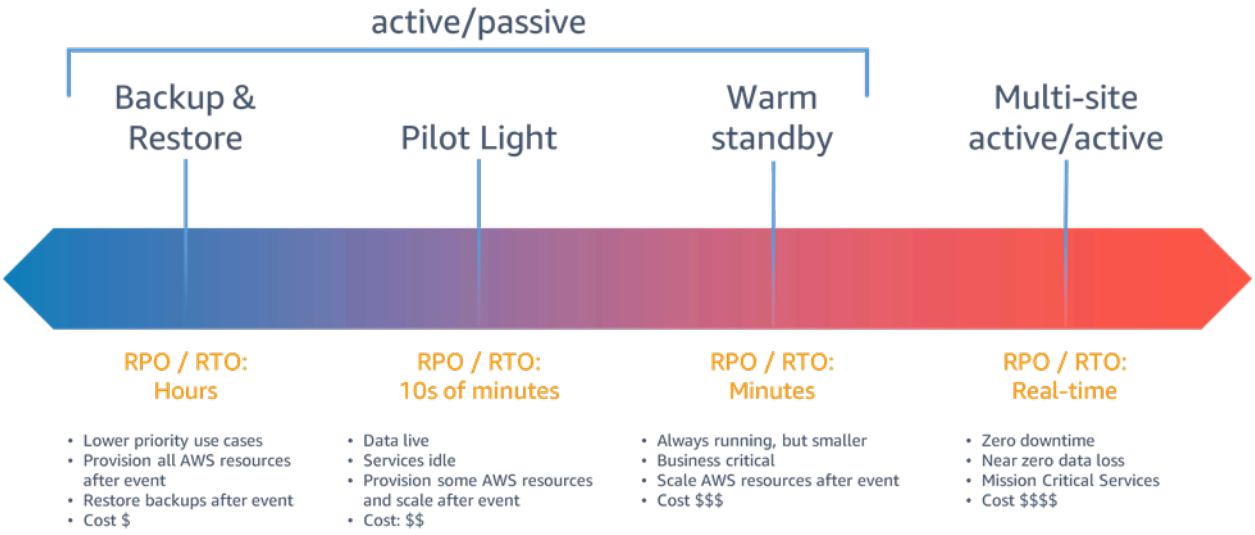

Figure 6 - Disaster recovery strategies

For a disaster event based on disruption or loss of one physical data center for a well-architected, highly available workload, you may only require a backup and restore approach to disaster recovery. If your definition of a disaster goes beyond the disruption or loss of a physical data center to that of a Region or if you are subject to regulatory requirements that require it, then you should consider Pilot Light, Warm Standby, or Multi-Site Active/Active.

When choosing your strategy, and the AWS resources to implement it, keep in mind that within AWS, we commonly divide services into the *data plane* and the *control plane*. The data plane is responsible for delivering real-time service while control planes are used to configure the

environment. For maximum resiliency, you should use only data plane operations as part of your failover operation. This is because the data planes typically have higher availability design goals than the control planes.

Backup and restore

Backup and restore is a suitable approach for mitigating against data loss or corruption. This approach can also be used to mitigate against a regional disaster by replicating data to other AWS Regions, or to mitigate lack of redundancy for workloads deployed to a single Availability Zone. In addition to data, you must redeploy the infrastructure, configuration, and application code in the recovery Region. To enable infrastructure to be redeployed quickly without errors, you should always deploy using infrastructure as code (IaC) using services such as AWS CloudFormation or the AWS Cloud Development Kit (AWS CDK). Without IaC, it may be complex to restore workloads in the recovery Region, which will lead to increased recovery times and possibly exceed your RTO. In addition to user data, be sure to also back up code and configuration, including Amazon Machine Images (AMIs) you use to create Amazon EC2 instances. You can use AWS CodePipeline to automate redeployment of application code and configuration.

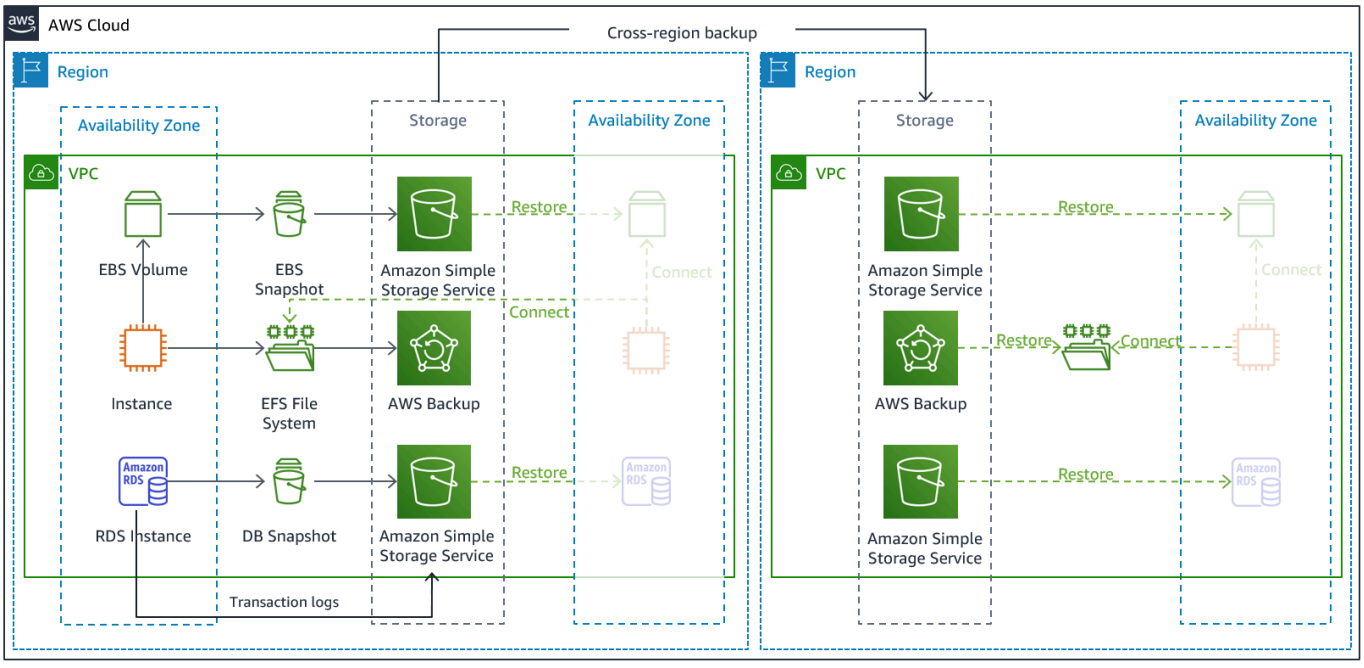

Figure 7 - Backup and restore architecture

AWS services

Your workload data will require a backup strategy that runs periodically or is continuous. How often you run your backup will determine your achievable recovery point (which should align to meet your RPO). The backup should also offer a way to restore it to the point in time in which it was taken. Backup with point-in-time recovery is available through the following services and resources:

- Amazon Elastic Block Store (Amazon EBS) snapshot
- Amazon DynamoDB backup
- Amazon RDS snapshot
- Amazon Aurora DB snapshot
- Amazon EFS backup (when using AWS Backup)
- Amazon Redshift snapshot
- Amazon Neptune snapshot
- Amazon DocumentDB
- Amazon FSx for Windows File Server, Amazon FSx for Lustre, Amazon FSx for NetApp ONTAP, and Amazon FSx for OpenZFS

For Amazon Simple Storage Service (Amazon S3), you can use Amazon S3 Cross-Region Replication (CRR) to asynchronously copy objects to an S3 bucket in the DR region continuously, while providing versioning for the stored objects so that you can choose your restoration point. Continuous replication of data has the advantage of being the shortest time (near zero) to back up your data, but may not protect against disaster events such as data corruption or malicious attack (such as unauthorized data deletion) as well as point-in-time backups. Continuous replication is covered in the AWS Services for Pilot Light section.

AWS Backup provides a centralized location to configure, schedule, and monitor AWS backup capabilities for the following services and resources:

- Amazon Elastic Block Store (Amazon EBS) volumes
- Amazon EC2 instances
- Amazon Relational Database Service (Amazon RDS) databases (including Amazon Aurora databases)
- Amazon DynamoDB tables

- Amazon Elastic File System (Amazon EFS) file systems
- AWS Storage Gateway volumes
- Amazon FSx for Windows File Server, Amazon FSx for Lustre, Amazon FSx for NetApp ONTAP, and Amazon FSx for OpenZFS

AWS Backup supports copying backups across Regions, such as to a disaster recovery Region.

As an additional disaster recovery strategy for your Amazon S3 data, enable S3 object versioning. Object versioning protects your data in S3 from the consequences of deletion or modification actions by retaining the original version before the action. Object versioning can be a useful mitigation for human-error type disasters. If you are using S3 replication to back up data to your DR region, then, by default, when an object is deleted in the source bucket, Amazon S3 adds a delete marker in the source bucket only. This approach protects data in the DR Region from malicious deletions in the source Region.

In addition to data, you must also back up the configuration and infrastructure necessary to redeploy your workload and meet your Recovery Time Objective (RTO). AWS CloudFormation provides Infrastructure as Code (IaC), and enables you to define all of the AWS resources in your workload so you can reliably deploy and redeploy to multiple AWS accounts and AWS Regions. You can back up Amazon EC2 instances used by your workload as Amazon Machine Images (AMIs). The AMI is created from snapshots of your instance's root volume and any other EBS volumes attached to your instance. You can use this AMI to launch a restored version of the EC2 instance. An AMI can be copied within or across Regions. Or, you can use AWS Backup to copy backups across accounts and to other AWS Regions. The cross-account backup capability helps protect from disaster events that include insider threats or account compromise. AWS Backup also adds additional capabilities for EC2 backup—in addition to the instance's individual EBS volumes, AWS Backup also stores and tracks the following metadata: instance type, configured virtual private cloud (VPC), security group, IAM role, monitoring configuration, and tags. However, this additional metadata is only used when restoring the EC2 backup to the same AWS Region.

Any data stored in the disaster recovery Region as backups must be restored at time of failover. AWS Backup offers restore capability, but does not currently enable scheduled or automatic restoration. You can implement automatic restore to the DR region using the AWS SDK to call APIs for AWS Backup. You can set this up as a regularly recurring job or trigger restoration whenever a backup is completed. The following figure shows an example of automatic restoration using Amazon Simple Notification Service (Amazon SNS) and AWS Lambda. Implementing a scheduled periodic data restore is a good idea as data restore from backup is a control plane operation. If this

operation was not available during a disaster, you would still have operable data stores created from a recent backup.

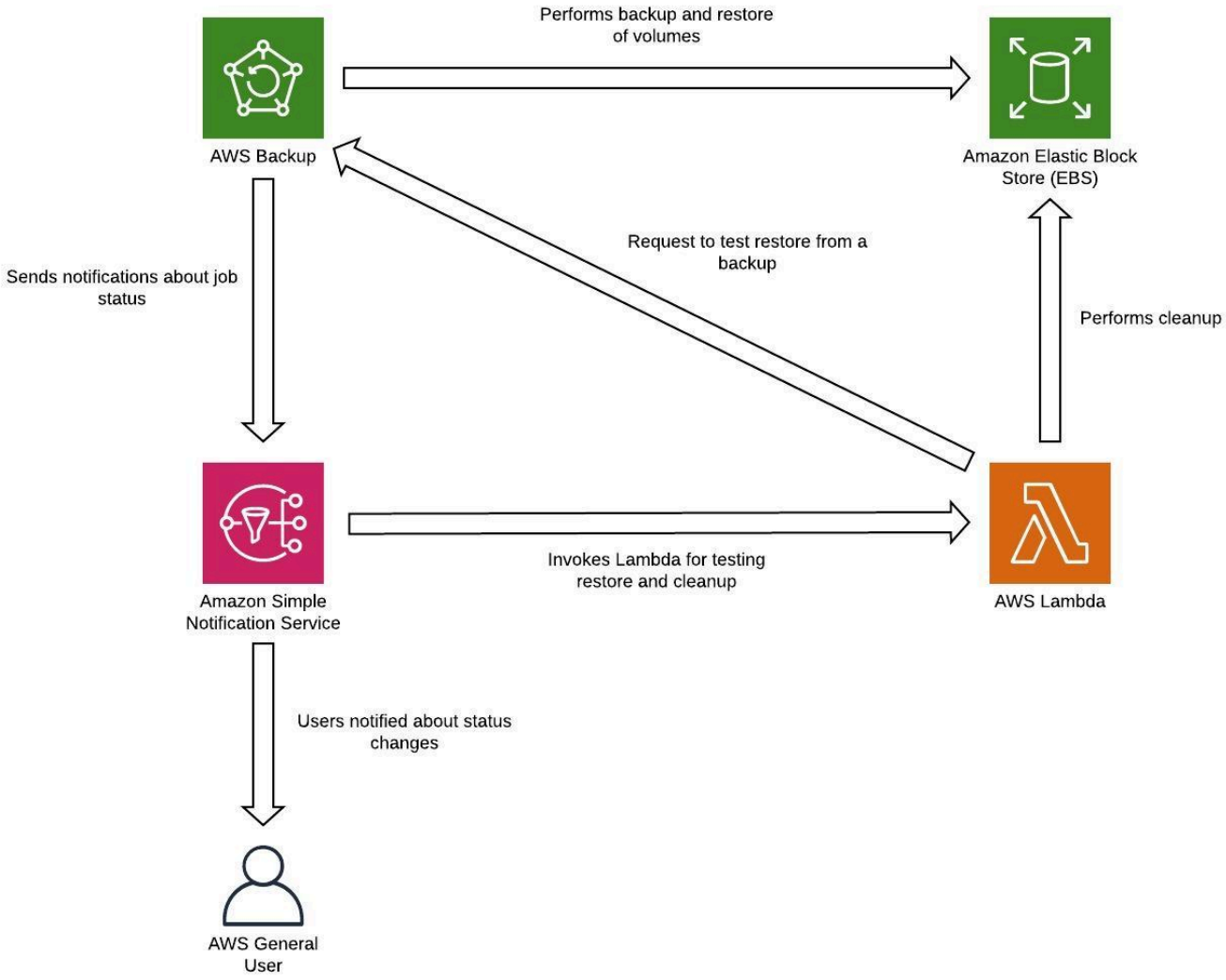

Figure 8 - Restoring and testing backups

> **Note**
>
> Your backup strategy must include testing your backups. See the Testing Disaster Recovery section for more information. Refer to the AWS Well-Architected Lab: Testing Backup and Restore of Data for a hands-on demonstration of implementation.

Pilot light

With the *pilot light* approach, you replicate your data from one Region to another and provision a copy of your core workload infrastructure. Resources required to support data replication and backup, such as databases and object storage, are always on. Other elements, such as application servers, are loaded with application code and configurations, but are "switched off" and are only used during testing or when disaster recovery failover is invoked. In the cloud, you have the flexibility to deprovision resources when you do not need them, and provision them when you do. A best practice for “switched off” is to not deploy the resource, and then create the configuration and capabilities to deploy it (“switch on”) when needed. Unlike the backup and restore approach, your core infrastructure is always available and you always have the option to quickly provision a full scale production environment by switching on and scaling out your application servers.

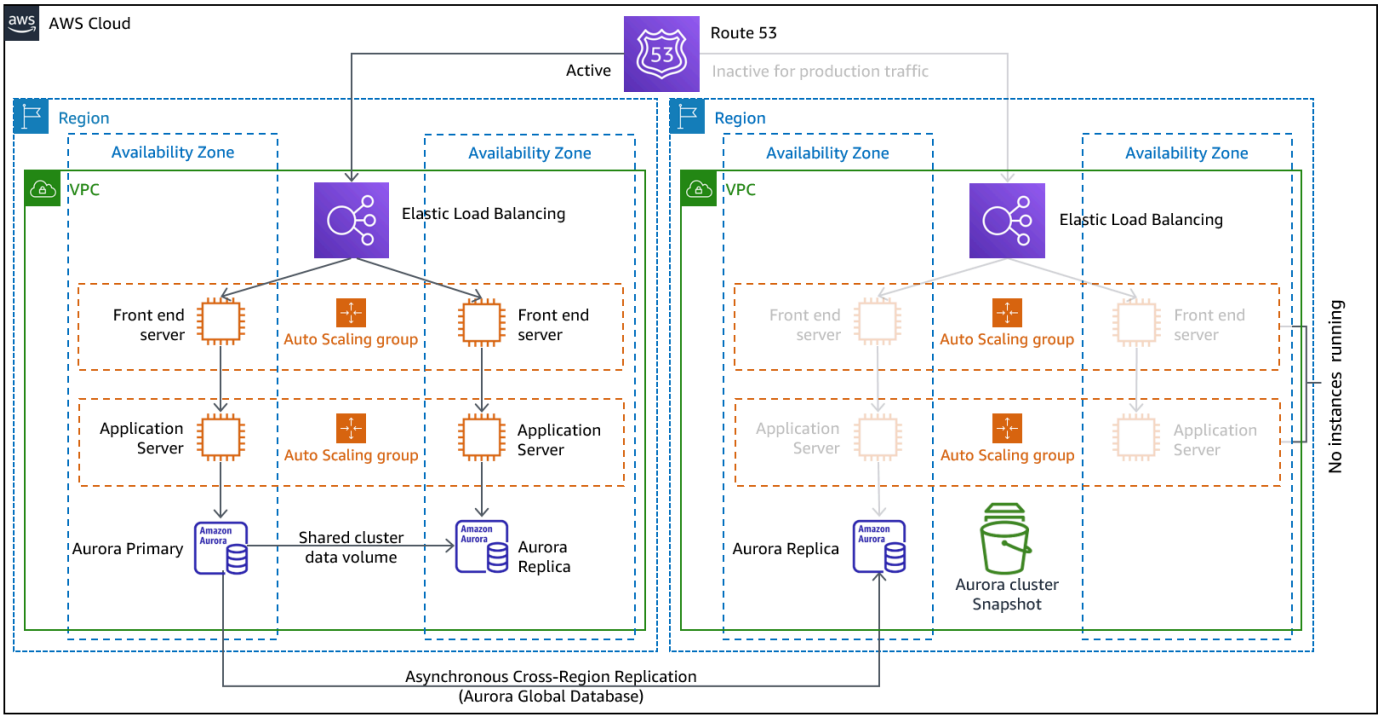

Figure 9 - Pilot light architecture

A pilot light approach minimizes the ongoing cost of disaster recovery by minimizing the active resources, and simplifies recovery at the time of a disaster because the core infrastructure requirements are all in place. This recovery option requires you to change your deployment approach. You need to make core infrastructure changes to each Region and deploy workload (configuration, code) changes simultaneously to each Region. This step can be simplified by automating your deployments and using infrastructure as code (IaC) to deploy infrastructure

across multiple accounts and Regions (full infrastructure deployment to the primary Region and scaled down/switched-off infrastructure deployment to DR regions). It is recommended you use a different account per Region to provide the highest level of resource and security isolation (in the case compromised credentials are part of your disaster recovery plans as well).

With this approach, you must also mitigate against a data disaster. Continuous data replication protects you against some types of disaster, but it may not protect you against data corruption or destruction unless your strategy also includes versioning of stored data or options for point-in-time recovery. You can back up the replicated data in the disaster Region to create point-in-time backups in that same Region.

AWS services

In addition to using the AWS services covered in the Backup and Restore section to create point-in-time backups, also consider the following services for your pilot light strategy.

For pilot light, continuous data replication to live databases and data stores in the DR region is the best approach for low RPO (when used in addition to the point-in-time backups discussed previously). AWS provides continuous, cross-region, asynchronous data replication for data using the following services and resources:

- Amazon Simple Storage Service (Amazon S3) Replication
- Amazon RDS read replicas
- Amazon Aurora global databases
- Amazon DynamoDB global tables
- Amazon DocumentDB global clusters
- Global Datastore for Amazon ElastiCache (Redis OSS)

With continuous replication, versions of your data are available almost immediately in your DR Region. Actual replication times can be monitored using service features like S3 Replication Time Control (S3 RTC) for S3 objects and management features of Amazon Aurora global databases.

When failing over to run your read/write workload from the disaster recovery Region, you must promote an RDS read replica to become the primary instance. For DB instances other than Aurora, the process takes a few minutes to complete and rebooting is part of the process. For Cross-Region Replication (CRR) and failover with RDS, using Amazon Aurora global database provides several

advantages. Global database uses dedicated infrastructure that leaves your databases entirely available to serve your application, and can replicate to the secondary Region with typical latency of under a second (and within an AWS Region is much less than 100 milliseconds). With Amazon Aurora global database, if your primary Region suffers a performance degradation or outage, you can promote one of the secondary regions to take read/write responsibilities in less than one minute even in the event of a complete regional outage. You can also configure Aurora to monitor the RPO lag time of all secondary clusters to make sure that at least one secondary cluster stays within your target RPO window.

A scaled down version of your core workload infrastructure with fewer or smaller resources must be deployed in your DR Region. Using AWS CloudFormation, you can define your infrastructure and deploy it consistently across AWS accounts and across AWS Regions. AWS CloudFormation uses predefined pseudo parameters to identify the AWS account and AWS Region in which it is deployed. Therefore, you can implement condition logic in your CloudFormation templates to deploy only the scaled-down version of your infrastructure in the DR Region. For EC2 instance deployments, an Amazon Machine Image (AMI) supplies information such as hardware configuration and installed software. You can implement an Image Builder pipeline that creates the AMIs you need and copy these to both your primary and backup Regions. This helps to ensure that these *golden AMIs* have everything you need to re-deploy or scale-out your workload in a new region, in case of a disaster event. Amazon EC2 instances are deployed in a scaled-down configuration (less instances than in your primary Region). To scale-out the infrastructure to support production traffic, see Amazon EC2 Auto Scaling in the Warm Standby section.

For an active/passive configuration such as pilot light, all traffic initially goes to the primary Region and switches to the disaster recovery Region if the primary Region is no longer available. This failover operation can be initiated either automatically or manually. Automatically initiated failover based on health checks or alarms should be used with caution. Even using the best practices discussed here, recovery time and recovery point will be greater than zero, incurring some loss of availability and data. If you fail over when you don't need to (false alarm), then you incur those losses. Manually initiated failover is therefore often used. In this case, you should still automate the steps for failover, so that the manual initiation is like the push of a button.

There are several traffic management options to consider when using AWS services.

One option is to use Amazon Route 53. Using Amazon Route 53, you can associate multiple IP endpoints in one or more AWS Regions with a Route 53 domain name. Then, you can route traffic to the appropriate endpoint under that domain name. On failover you need to switch traffic to the recovery endpoint, and away from the primary endpoint. Amazon Route 53 health checks monitor

these endpoints. Using these health checks, you can configure automatically initiated DNS failover to ensure traffic is sent only to healthy endpoints, which is a highly reliable operation done on the data plane. To implement this using manually initiated failover you can use Amazon Application Recovery Controller (ARC). With ARC, you can create Route 53 health checks that do not actually check health, but instead act as on/off switches that you have full control over. Using the AWS CLI or AWS SDK, you can script failover using this highly available, data plane API. Your script toggles these switches (the Route 53 health checks) telling Route 53 to send traffic to the recovery Region instead of the primary Region. Another option for manually initiated failover that some have used is to use a weighted routing policy and change the weights of the primary and recovery Regions so that all traffic goes to the recovery Region. However, be aware this is a control plane operation and therefore not as resilient as the data plane approach using Amazon Application Recovery Controller (ARC).

Another option is to use AWS Global Accelerator. Using AnyCast IP, you can associate multiple endpoints in one or more AWS Regions with the same static public IP address or addresses. AWS Global Accelerator then routes traffic to the appropriate endpoint associated with that address. Global Accelerator health checks monitor endpoints. Using these health checks, AWS Global Accelerator checks the health of your applications and routes user traffic automatically to the healthy application endpoint. For manually initiated failover, you can adjust which endpoint receives traffic using traffic dials, but note this is a control plane operation. Global Accelerator offers lower latencies to the application endpoint since it makes use of the extensive AWS edge network to put traffic on the AWS network backbone as soon as possible. Global Accelerator also avoids caching issues that can occur with DNS systems (like Route 53).

Amazon CloudFront offers origin failover, where if a given request to the primary endpoint fails, CloudFront routes the request to the secondary endpoint. Unlike the failover operations described previously, all subsequent requests still go to the primary endpoint, and failover is done per each request.

AWS Elastic Disaster Recovery

AWS Elastic Disaster Recovery (DRS) continuously replicates server-hosted applications and server-hosted databases from any source into AWS using block-level replication of the underlying server. Elastic Disaster Recovery enables you to use a Region in AWS Cloud as a disaster recovery target for a workload hosted on-premises or on another cloud provider, and its environment. It can also be used for disaster recovery of AWS hosted workloads if they consist only of applications and databases hosted on EC2 (that is, not RDS). Elastic Disaster Recovery uses the Pilot Light strategy, maintaining a copy of data and “switched-off” resources in an Amazon Virtual Private Cloud

(Amazon VPC) used as a staging area. When a failover event is triggered, the staged resources are used to automatically create a full-capacity deployment in the target Amazon VPC used as the recovery location.

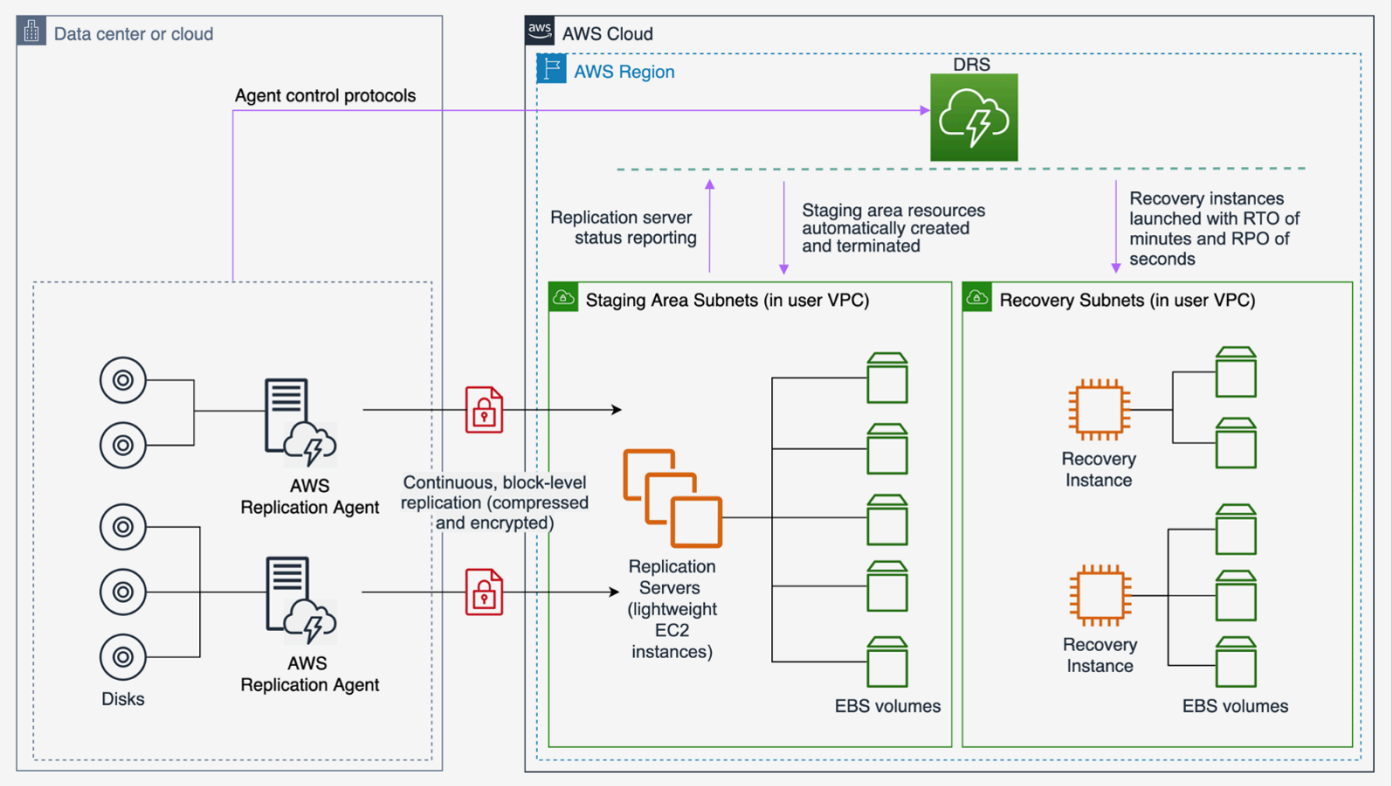

Figure 10 - AWS Elastic Disaster Recovery architecture

Warm standby

The *warm standby* approach involves ensuring that there is a scaled down, but fully functional, copy of your production environment in another Region. This approach extends the pilot light concept and decreases the time to recovery because your workload is always-on in another Region. This approach also allows you to more easily perform testing or implement continuous testing to increase confidence in your ability to recover from a disaster.

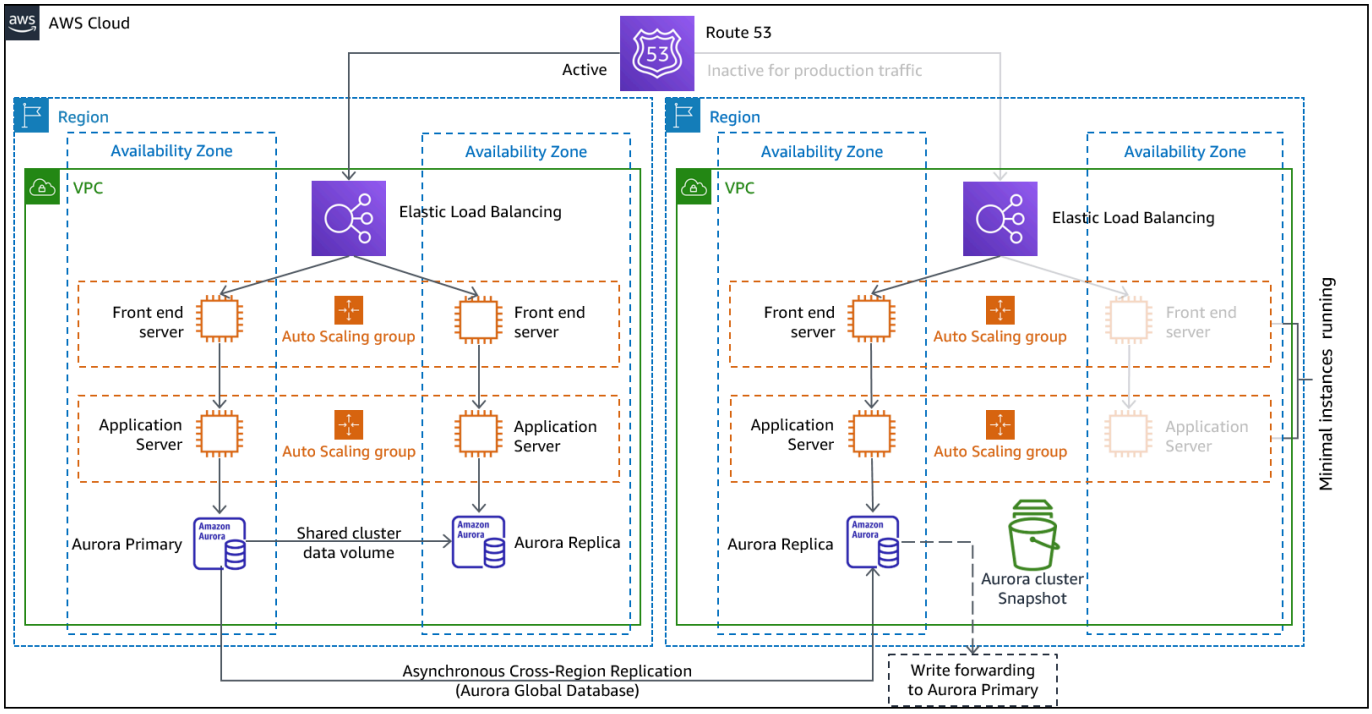

Figure 11 - Warm standby architecture

Note: The difference between pilot light and warm standby can sometimes be difficult to understand. Both include an environment in your DR Region with copies of your primary Region assets. The distinction is that pilot light cannot process requests without additional action taken first, whereas warm standby can handle traffic (at reduced capacity levels) immediately. The pilot light approach requires you to "turn on" servers, possibly deploy additional (non-core) infrastructure, and scale up, whereas warm standby only requires you to scale up (everything is already deployed and running). Use your RTO and RPO needs to help you choose between these approaches.

AWS services

All of the AWS services covered under backup and restore and pilot light are also used in warm standby for data backup, data replication, active/passive traffic routing, and deployment of infrastructure including EC2 instances.

Amazon EC2 Auto Scaling is used to scale resources including Amazon EC2 instances, Amazon ECS tasks, Amazon DynamoDB throughput, and Amazon Aurora replicas within an AWS Region. Amazon EC2 Auto Scaling scales deployment of EC2 instance across Availability Zones within an AWS Region, providing resiliency within that Region. Use Auto Scaling to scale out your DR Region

to full production capability, as part of a pilot light or warm standby strategies. For example, for EC2, increase the *desired capacity* setting on the Auto Scaling group. You can adjust this setting manually through the AWS Management Console, automatically through the AWS SDK, or by redeploying your AWS CloudFormation template using the new desired capacity value. You can use AWS CloudFormation parameters to make redeploying the CloudFormation template easier. Ensure that service quotas in your DR Region are set high enough so as to not limit you from scaling up to production capacity.

Because Auto Scaling is a control plane activity, taking a dependency on it will lower the resiliency of your overall recovery strategy. It is a trade-off. You can choose to provision sufficient capacity such that the recovery Region can handle the full production load as deployed. This statically stable configuration is called *hot standby* (see the next section). Or you may choose to provision fewer resources which will cost less, but take a dependency on Auto Scaling. Some DR implementations will deploy enough resources to handle initial traffic, ensuring low RTO, and then rely on Auto Scaling to ramp up for subsequent traffic.

Multi-site active/active

You can run your workload simultaneously in multiple Regions as part of a *multi-site active/active* or *hot standby active/passive* strategy. Multi-site active/active serves traffic from all regions to which it is deployed, whereas hot standby serves traffic only from a single region, and the other Region(s) are only used for disaster recovery. With a multi-site active/active approach, users are able to access your workload in any of the Regions in which it is deployed. This approach is the most complex and costly approach to disaster recovery, but it can reduce your recovery time to near zero for most disasters with the correct technology choices and implementation (however data corruption may need to rely on backups, which usually results in a non-zero recovery point). Hot standby uses an active/passive configuration where users are only directed to a single region and DR regions do not take traffic. Most customers find that if they are going to stand up a full environment in the second Region, it makes sense to use it active/active. Alternatively, if you do not want to use both Regions to handle user traffic, then Warm Standby offers a more economical and operationally less complex approach.

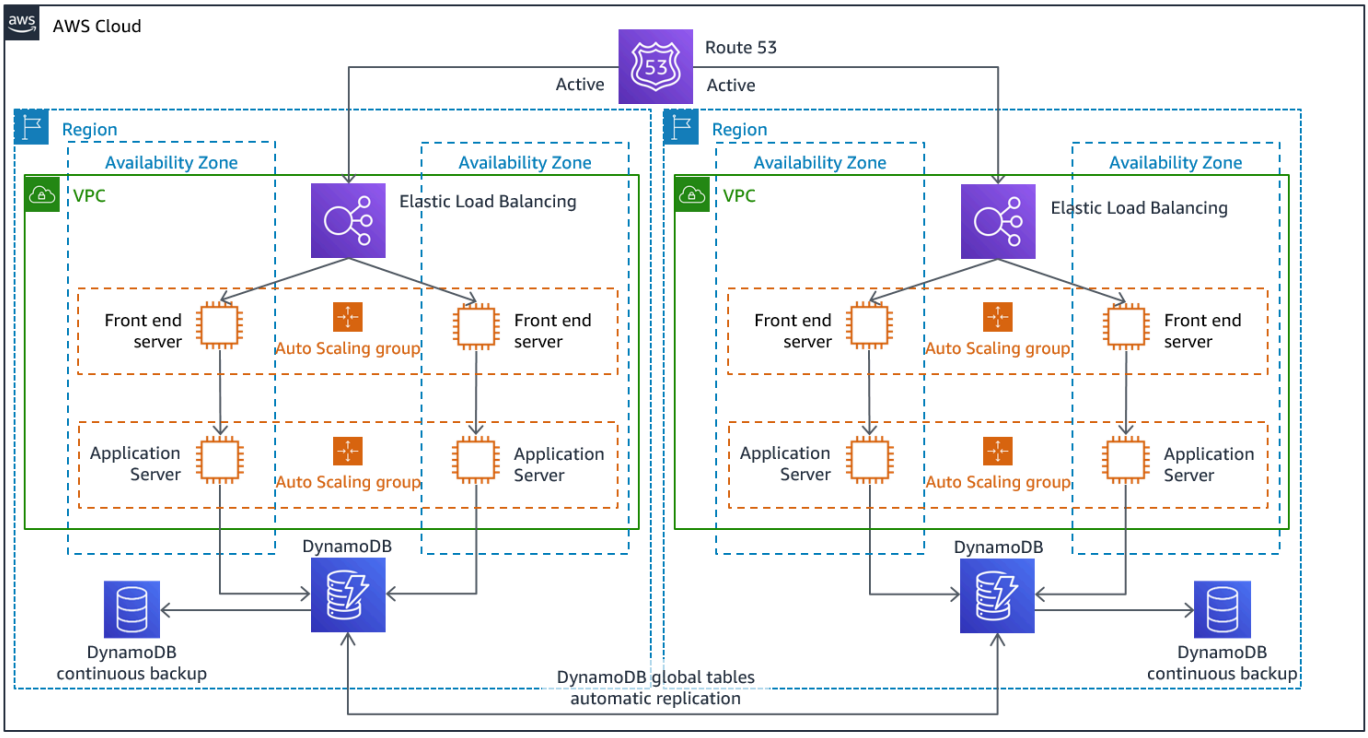

Figure 12 - Multi-site active/active architecture (change one Active path to Inactive for hot standby)

With multi-site active/active, because the workload is running in more than one Region, there is no such thing as failover in this scenario. Disaster recovery testing in this case would focus on how the workload reacts to loss of a Region: Is traffic routed away from the failed Region? Can the other Region(s) handle all the traffic? Testing for a data disaster is also required. Backup and recovery are still required and should be tested regularly. It should also be noted that recovery times for a data disaster involving data corruption, deletion, or obfuscation will always be greater than zero and the recovery point will always be at some point before the disaster was discovered. If the additional complexity and cost of a multi-site active/active (or hot standby) approach is required to maintain near zero recovery times, then additional efforts should be made to maintain security and to prevent human error to mitigate against human disasters.

AWS services

All of the AWS services covered under backup and restore, pilot light, and warm standby also are used here for point-in-time data backup, data replication, active/active traffic routing, and deployment and scaling of infrastructure including EC2 instances.

For the active/passive scenarios discussed earlier (Pilot Light and Warm Standby), both Amazon Route 53 and AWS Global Accelerator can be used for route network traffic to the active region.

For the active/active strategy here, both of these services also enable the definition of policies that determine which users go to which active regional endpoint. With AWS Global Accelerator you set a traffic dial to control the percentage of traffic that is directed to each application endpoint. Amazon Route 53 supports this percentage approach, and also multiple other available policies including geoproximity and latency based ones. Global Accelerator automatically leverages the extensive network of AWS edge servers, to onboard traffic to the AWS network backbone as soon as possible, resulting in lower request latencies.

Asynchronous data replication with this strategy enables near-zero RPO. AWS services like Amazon Aurora global database use dedicated infrastructure that leaves your databases entirely available to serve your application, and can replicate to up to five secondary Region with typical latency of under a second. With active/passive strategies, writes occur only to the primary Region. The difference with active/active is designing how data consistency with writes to each active Region are handled. It is common to design user reads to be served from the Region closest to them, known as *read local*. With writes, you have several options:

- A *write global* strategy routes all writes to a single Region. In case of failure of that Region, another Region would be promoted to accept writes. Aurora global database is a good fit for *write global*, as it supports synchronization with read-replicas across Regions, and you can promote one of the secondary Regions to take read/write responsibilities in less than one minute. Aurora also supports write forwarding, which lets secondary clusters in an Aurora global database forward SQL statements that perform write operations to the primary cluster.
- A *write local* strategy routes writes to the closest Region (just like reads). Amazon DynamoDB global tables enables such a strategy, allowing read and writes from every region your global table is deployed to. Amazon DynamoDB global tables use a *last writer wins* reconciliation between concurrent updates.
- A *write partitioned* strategy assigns writes to a specific Region based on a partition key (like user ID) to avoid write conflicts. Amazon S3 replication configured bi-directionally can be used for this case, and currently supports replication between two Regions. When implementing this approach, make sure to enable replica modification sync on both buckets A and B to replicate replica metadata changes like object access control lists (ACLs), object tags, or object locks on the replicated objects. You can also configure whether or not to replicate delete markers between buckets in your active Regions. In addition to replication, your strategy must also include point-in-time backups to protect against data corruption or destruction events.

AWS CloudFormation is a powerful tool to enforce consistently deployed infrastructure among AWS accounts in multiple AWS Regions. AWS CloudFormation StackSets extends this functionality

by enabling you to create, update, or delete CloudFormation stacks across multiple accounts and Regions with a single operation. Although AWS CloudFormation uses YAML or JSON to define Infrastructure as Code, AWS Cloud Development Kit (AWS CDK) allows you to define Infrastructure as Code using familiar programming languages. Your code is converted to CloudFormation which is then used to deploy resources in AWS.

Detection

It is important to know as soon as possible that your workloads are not delivering the business outcomes that they should be delivering. In this way, you can quickly declare a disaster and recover from an incident. For aggressive recovery objectives, this response time coupled with appropriate information is critical in meeting recovery objectives. If your recovery time objective is one hour, then you need to detect the incident, notify appropriate personnel, engage your escalation processes, evaluate information (if you have any) on expected time to recovery (without executing the DR plan), declare a disaster and recover within an hour.

> **Note**
>
> If stakeholders decide not to invoke DR even though the RTO would be at risk, then re-evaluate DR plans and objectives. The decision not to invoke DR plans may be because the plans are inadequate or there is a lack of confidence in execution.

It is critical to factor in incident detection, notification, escalation, discovery and declaration into your planning and objectives to provide realistic, achievable objectives that provide business value.

AWS publishes our most up-to-the-minute information on service availability on the Service Health Dashboard. Check at any time to get current status information, or subscribe to an RSS feed to be notified of interruptions to each individual service. If you are experiencing a real-time, operational issue with one of our services that is not shown on the Service Health Dashboard, you can create a Support Request.

The AWS Health Dashboard provides information about AWS Health events that can affect your account. The information is presented in two ways: a dashboard that shows recent and upcoming events organized by category, and a full event log that shows all events from the past 90 days.

For the most stringent RTO requirements, you can implement automated failover based on health checks. Design health checks that are representative of user experience and based on Key Performance Indicators. Deep health checks exercise key functionality of your workload and go beyond shallow heartbeat checks. Use deep health checks based on multiple signals. Use caution with this approach so that you do not trigger false alarms because failing over when there is no need to can in itself introduce availability risks.

Testing disaster recovery

Test disaster recovery implementation to validate the implementation and regularly test failover to your workload's DR Region to ensure that RTO and RPO are met.

A pattern to avoid is developing recovery paths that are rarely executed. For example, you might have a secondary data store that is used for read-only queries. When you write to a data store and the primary fails, you might want to fail over to the secondary data store. If you don't frequently test this failover, you might find that your assumptions about the capabilities of the secondary data store are incorrect. The capacity of the secondary, which might have been sufficient when you last tested, might no longer be able to tolerate the load under this scenario, or service quotas in the secondary Region might not be sufficient.

Our experience has shown that the only error recovery that works is the path you test frequently. This is the reason why having a small number of recovery paths is best.

You can establish recovery patterns and regularly test them. If you have a complex or critical recovery path, you still need to regularly execute that failure in production to validate that the recovery path works.

Manage configuration drift at the DR Region. Ensure that your infrastructure, data, and configuration are as needed at the DR Region. For example, check that AMIs and service quotas are up-to-date.

You can utilize AWS Config to continuously monitor and record your AWS resource configurations. AWS Config can detect drift and trigger AWS Systems Manager Automation to fix drift and raise alarms. AWS CloudFormation can additionally detect drift in stacks you have deployed.

Conclusion

Customers are responsible for the availability of their applications in the cloud. It is important to define what a disaster is and to have a disaster recovery plan that reflects this definition and the impact that it may have on business outcomes. Create Recovery Time Objective (RTO) and Recovery Point Objective (RPO) based on impact analysis and risk assessments and then choose the appropriate architecture to mitigate against disasters. Ensure that detection of disasters is possible and timely — it is vital to know when objectives are at risk. Ensure you have a plan and validate the plan with testing. Disaster recovery plans that have not been validated risk not being implemented due to a lack of confidence or failure to meet disaster recovery objectives.

Contributors

Contributors to this document include:

- Alex Livingstone, Practice Lead Cloud Operations, AWS Enterprise Support
- Seth Eliot, Principal Reliability Solutions Architect, Amazon Web Services

Further reading

For additional information, see:

- AWS Architecture Center
- Reliability Pillar, AWS Well-Architected Framework
- Disaster Recovery Plan Checklist
- Implementing Health Checks
- Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud
- Disaster Recovery (DR) Architecture on AWS, Part II: Backup and Restore with Rapid Recovery
- Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby
- Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active
- Creating Disaster Recovery Mechanisms Using Amazon Route 53
- Minimizing Dependencies in a Disaster Recovery Plan
- Hands on AWS Well-Architected Disaster Recovery Labs
- AWS Solutions Implementations: Multi-Region Application Architecture
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Minor updates	Bug fixes and numerous minor changes throughout.	April 1, 2022
Whitepaper updated	Minor editorial updates.	March 21, 2022
Whitepaper updated	Added information on data plane and control plane. Added more details on how to implement active/passive failover. Replaced CloudEndu re Disaster Recovery with AWS Elastic Disaster Recovery.	February 17, 2022
Minor update	AWS Well-Architected Tool information added.	February 11, 2022
Initial publication	Whitepaper first published.	February 12, 2021

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.