Implementation Guide

Guidance for Deploying Cross-Region Disaster Recovery with AWS Elastic Disaster Recovery

Guidance for Deploying Cross-Region Disaster Recovery with AWS Elastic Disaster Recovery: Implementation Guide

Table of Contents

Guidance for Deploying Cross-Region Disaster Recovery with AWS Elastic Disaster Recovery

This user guide provides prescriptive instructions to deploy AWS Elastic Disaster Recovery for AWS customers who are looking to protect applications that currently operate within an Amazon Web Services (AWS) Region and choose to recover to another Region for disaster recovery. This guide serves to complement the publicly available Elastic Disaster Recovery documentation, available on the AWS documentation library. It introduces important concepts, provides specific guidance on configuration of Elastic Disaster Recovery for a cross-Region need, and step-by-step instructions on how to design, deploy, and manage Elastic Disaster Recovery.

By following this guide, you will be able to:

- Familiarize yourself with the concepts of Elastic Disaster Recovery
- Learn how to think about your overall disaster recovery design, and where Elastic Disaster Recovery fits
- Deploy Elastic Disaster Recovery to protect applications across AWS Regions
- Recover source servers to a different AWS Region using best practices
- Learn how to think about your Elastic Disaster Recovery testing process
- Understand how to recover your servers during a disaster
- Understand how to failback your servers after you have alleviated the disaster in your source Availability Zone (AZ)
- If needed: properly clean-up and remove servers from Elastic Disaster Recovery

What is AWS Elastic Disaster Recovery?

Elastic Disaster Recovery minimizes downtime and data loss with fast, reliable recovery of on-premises and cloud-based applications running on Amazon Elastic Compute Cloud (EC2) and Amazon Elastic Block Store (EBS) using affordable storage, minimal compute, and point-in-time recovery. Elastic Disaster Recovery continuously replicates source servers to your recovery target within AWS, allowing you to prepare your environment to recover within minutes from unexpected infrastructure or application outages, human error, data corruption, ransomware, or other disruptions.

Elastic Disaster Recovery provides a unified process to test, recover, and fail back any application running on a supported Operating Systems (OS). Elastic Disaster Recovery supports large, heterogeneous environments with mission-critical workloads, and can support recovery point objectives (RPO) of seconds, with recovery time objectives (RTO) of minutes, reducing overall disaster recovery costs.

Core concepts

Below is a high-level overview of the Core Concepts that are incorporated in Elastic Disaster Recovery. We recommend readers to also familiarize themselves with core AWS functionality such as Amazon Identity and Access Management (IAM), Networking Essentials, Amazon Elastic Compute Cloud (EC2); and general disaster recovery concepts.

The main goal of disaster recovery (DR) is to help your business prepare and recover from unexpected events in an acceptable amount of time. This means you need to determine which applications deliver the core functionality required for your business to be available, and define the appropriate recovery time objective (RTO) and recovery point objective (RPO) required for these applications.

Region

AWS has the concept of a Region, which is a physical location where we cluster data centers around the world. We call each group of logical data centers an Availability Zone (AZ). Each AWS Region consists of a minimum of three, isolated, and physically separate AZs within a geographic area. Unlike other cloud providers, who often define a Region as a single data center, the multiple AZ design of every AWS Region offers advantages for customers. Each AZ has independent power, cooling, and physical security and is connected through redundant, ultra-low-latency networks. AWS customers focused on high availability can design their applications to run in multiple AZs to achieve even greater fault-tolerance. AWS infrastructure Regions meet the highest levels of security, compliance, and data protection.

Availability Zone (AZ)

An Availability Zone (AZ) is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. AZs give customers the ability to operate production applications and databases that are more highly available, fault tolerant, and scalable than would be possible from a single data center. All AZs 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 AZs. All traffic between AZs is encrypted. The network performance is sufficient to accomplish synchronous replication between AZs. AZs make partitioning applications for high availability easy. If an application is partitioned across AZs, companies are better isolated and protected from issues such as power outages, lightning strikes, tornadoes, earthquakes, and more. AZs are physically separated by a meaningful distance, many kilometers, from any other AZ, although all are within 100 km (60 miles) of each other.

Recovery Point Objective (RPO)

RPO is defined by how much data loss your application can tolerate, and determines what is considered an acceptable loss of data between the last recovery point and the interruption of service.

Recovery Time Objective (RTO)

Defined by the organization, RTO is the maximum acceptable time between the interruption of service and restoration of service. This determines what is considered an acceptable time window when service is unavailable after a disaster.

Source server

The Source server refers to the instance or server that you want to protect and recover in the event of a disaster. Elastic Disaster Recovery can be used to recover Amazon EC2 instances (referred to as Recovery Instances on Elastic Disaster Recovery) in a different Availability Zone within the same Region, or a different AWS Region. Elastic Disaster Recovery can also protect applications hosted on physical infrastructure, VMware vSphere, Microsoft Hyper-V, and cloud infrastructure from other cloud providers.

Recovery subnet

The Recovery subnet is the virtual network segment hosted within an Availability Zone and hosts the recovered Source Servers in the event of a disaster.

AWS Replication Agent

The AWS Replication Agent is a lightweight software package. It must be installed on each source EC2 instance that you want to protect using Elastic Disaster Recovery. The agent performs two main tasks:

1. Initial block-level replication of disks by copying the state of the disk on the source server and transmitting this data to the staging environment where this data is persisted on Elastic Block Storage (EBS) volumes that logically map to the source disks.
2. Real-time monitoring and replication of all block-level changes once the agent has completed the initial synchronization process.

Staging area subnet

In the selected AWS account and Region, the subnet selected to host the Replication Server is referred to as the Staging area subnet. The Elastic Disaster Recovery service utilizes low-cost compute and storage hosted on the Staging Area subnet to keep the data in sync with the source environment. Replication resources consist of Replication Servers, Staging volumes, and EBS snapshots.

Replication server

The Replication Server is responsible for receiving and storing the replicated data from the Source Server. The Replication Server is an EC2 instance to which Staging EBS Volumes are attached. The AWS Replication Agent sends data from the Source Server to the Replication Server during the initial synchronization process or when blocks change on the Source Server. Replication Servers will take snapshots of the staging EBS Volumes attached to them.

Point in time snapshots (PiT snapshots)

These are periodic backups taken by the Replication Server at specific intervals to capture the state of the Source Server and its data. The intervals are:

1. Once every 10 minutes for the last hour.
2. Once an hour for the last 24 hours.
3. Once a day for the last 7 days (unless a different retention period is configured, 1-365 days).

These PiT snapshots are used during recovery or recovery drill to recover the source server to a particular point in time.

Conversion Server

The Conversion Server is a component that makes all the necessary modifications to allow the target instance to boot and run, including pre- and post-boot scripts. The conversion server is launched within the staging area subnet and managed by the Elastic Disaster Recovery service. Conversion Servers are ephemeral resources and will only last for minutes in order to complete the conversion process.

Drills

Drills refer to scheduled or ad-hoc tests performed to validate the effectiveness of your disaster recovery plan. Elastic Disaster Recovery allows you to conduct drills to simulate recovery scenarios without impacting the production environment or replication state.

Recovery instance

During an actual recovery, a recovery instance is provisioned in the recovery subnet. The recovery instance is an EC2 instance and a fully functional copy of the source server that allows you to recover operations in the selected Region.

Drill Instance

A Drill Instance is an instance that has been launched using Elastic Disaster Recovery for the purpose of a drill or `test.` The goal of launching a drill instance is to test and validate your disaster recovery plan before an actual disaster. This instance is meant to be launched while your source server remains active. You may choose to activate this instance for production use by shutting down the source server and redirecting traffic to this instance.

Failover

Failover is the process of initiating a recovery in Elastic Disaster Recovery, launching an EC2 instance, and restoring your data based on the PiT snapshot selected to an EBS volume. This process would include failing over to a new Region, launching the recovery instance, and validating the application is ready to receive traffic. Additional steps are often required to prepare the recovery environment for a failover and these are often documented and executed as part of a DR runbook.

Failback

Failback is the process of returning to normal operations at your source site. This includes replicating data back to the source Region, bringing the source servers back online, and redirecting user traffic back to these machines (redirection of traffic, as well as other configuration operations, are handled outside of the AWS Elastic Disaster Recovery service)

Setting up

AWS Elastic Disaster Recovery is only one part of a larger disaster recovery strategy, and being prepared for these unforeseen events requires proper coordination across people, processes, and technology. The recovery plan should be documented and clearly define the stakeholders with roles and responsibilities, along with the steps that should be taken in the event of a real disaster. Below is a checklist of key concepts to consider as part of the planning process.

Sign up for AWS

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

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

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

Who are the stakeholders?

Identify all individuals and stakeholders who should be involved and informed when a disaster occurs. Consider using tools such as a responsibility matrix that provide a method to define who is responsible, accountable, consulted, and informed during a disaster. In many situations, we tend to focus on technical stakeholders who are involved with responding to the actual disaster, but we should also consider other stakeholders, such as vendors, third-party suppliers, public relations, marketing teams, and even key customers. We recommend keeping a registry of all stakeholders with their defined responsibilities and contact information. One of the most critical roles when preparing for a disaster is defining the individual(s) who will make the final decision on declaring a disaster and initiating the Business Continuity/Disaster Recovery Plan.

Establish Communication Channels

Once you have identified and documented all relevant stakeholders, it will be necessary to define the proper communication channels to keep everyone informed. Part of this process should be establishing a chain of command and defining well-understood escalation paths. We generally recommend the use of dedicated communication channels and hubs, such as an on-site situation room where everyone will gather to respond to the disaster. Video conferencing and instant messaging can also be used to facilitate virtual meeting rooms. It is highly advised that executive leadership is kept informed throughout the process.

Maintain Up to Date Documentation

Disaster might be hard to predict, but how we respond to these types of events should be predictable. Once it has been determined that you will be activating your disaster recovery response, it is critical to follow the procedures that have been tried and tested. In all cases, this should start with up to date documentation detailing all steps to be followed. Although your operations and engineering teams are skilled and knowledgeable, the pressure that comes with a disaster is high.

The documentation should include: information on configuration state (mapped network connections), with functioning devices and their configurations.

Furthermore, documentation should include the entire setup of systems and their usage: operating system (OS) and configuration, applications versions, storage and databases (how and where the data is saved, how backups are restored, how the data is verified for accuracy),architecture diagrams, vendor support contacts, and the responsibility matrix. It should contain everything IT related that your business relies upon. Keep hard copies of the documentation, as outages may knock your internal systems offline.

When to Activate the Disaster Recovery Plan

It is critical to quickly detect when your workloads are not meeting business objectives. 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 point 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 DRplan), declare a disaster, and recover within an hour.

Key Performance indicators (KPIs) are quantifiable measurements that help you understand how well you’re performing. It is critical to define and track KPIs in order to determine when your business processes are impaired and determine the cause. In this way, you can quickly declare a disaster and recover from an unexpected event. For aggressive recovery objectives, the time to detect an event, declare a disaster, and respond with your recovery plan will determine if your recovery objectives can be met.

Define action response procedure and verification process

After declaring a disaster, the recovery environment should be activated as soon as possible. An action response procedure outlines all of the necessary steps for recovering at the disaster recovery site. Ensure that your action response procedure is documented and provides details on how the necessary services will be started, verified, and controlled. It is recommended that automation be used whenever possible to minimize the impact of human error. Having all services up in the recovery site is not enough to declare success. It is critical to have a verification process that tests that all of the required data is in place, network traffic has been redirected, and all of the required business applications are functioning properly.

Perform Regular Disaster Recovery Drills

Many organizations do not perform disaster recovery drills on a regular basis because their failover procedures are too complex and they have concerns that failover tests will lead to a disruption of their production environment (and possibly data loss). Despite these concerns, it is important to schedule frequent disaster recovery drills to build confidence in the plan, build comfort within the team, and identify gaps. People will play a large part in any disaster recovery plan, and only by rehearsing the steps and procedures can we ensure that they can respond quickly and accurately to a real event. Furthermore, as the state and configuration of our systems change over time, only by conducting such exercises can we identify unexpected impact. In many cases, planned drills can be scoped down to focus on specific parts of the response plan. When using Elastic Disaster Recovery, these drills can be conducted in an isolated manner, in such a way that production is not impacted.

Stay up to date

Many companies maintain a risk register that tracks and quantifies potential risks to the business. They often include an analysis of current threats, previous disasters, and lessons learned. The risk register should have stakeholders that extend outside of the technology and operations teams and include the business, risk, and executive leadership roles. It is important to be aware of how you handled previous disasters, as well as how you performed during more recent drills. All documentation should be up to date reflecting the current environment, processes, and procedures.

Recovery Operations

In a cross-Region use case, most customers will want to return to the primary Region once they have confidence that the Region is no longer impaired and is considered stable. The process to return to the primary Region should be scheduled in advance and should be done during a planned maintenance window.

Technical prerequisites

Technical prerequisites

Implementing Elastic Disaster Recovery is a critical step in ensuring business continuity and resilience against unexpected disruptions. To achieve a successful deployment, it is essential to meet specific technical requirements that encompass various aspects of the system. These requirements range from network settings and communication protocols to supported operating systems, Regions, and installation prerequisites.

The following sections provide a detailed overview of the technical requirements necessary for the implementation of Elastic Disaster Recovery. They include guidelines for staging area subnets, network requirements, Amazon S3 bucket access, operational subnets, supported AWS Regions, general installation requirements, and specific considerations for Windows and Linux systems.

1. **Administrative rights** - Elastic Disaster Recovery can only be initialized by the Admin user of your AWS Account.
    a. If you are using Single Sign On (SSO), refer to Authenticating with identities in AWS Elastic Disaster Recoveryfor more information
2. **Multi-Account Requirements** Reference
    - **Staging Account Planning and Limitations**: Due to AWS account wide API limitations, Elastic Disaster Recovery is limited to protecting 300 source servers per AWS account. In order to replicate more than 300 servers, you would be required to create multiple staging area AWS accounts. It would still be possible to recover all of your servers into a single recovery environment. Elastic Disaster Recovery can recover up to 3,000 servers into a single target AWS account.
3. **Network Requirements https://docs.aws.amazon.com/drs/latest/userguide/Network-Requirements.htmlReference**
    - **Preparation**: Create a dedicated staging subnet for data replication from source servers to AWS.
        - This subnet should have a Classless Inter Domain Routing (CIDR) range that meets the following criteria:
            - Not overlap with the source server CIDR ranges.
            - Have enough IP addresses for 1 replication server per 15 source volumes, or dedicated replication servers for highly transactional sources.

    - Support 1 conversion server per source server to be launched.

- **Staging subnet access requirements**: The staging area subnet requires outbound internet access to the Amazon EC2, Amazon S3, and Elastic Disaster Recovery endpoints within the Target Region. You can create private link endpoints, or use public internet access to communicate with these AWS services.
- **Communication over TCP Port 443**: All communication is encrypted with TLS. All control plane traffic is handled over TCP port 443 and should be permitted for the following:
    - Between the source servers and Elastic Disaster Recovery Service
    - Between the staging area subnet and AWS Elastic Disaster Recovery
    - The Elastic Disaster Recovery AWS Region-specific Console address: *example: drs.eu-west-1.amazonaws.com*
    - Amazon S3 service URLs (required for downloading AWS Elastic Disaster Recovery software)
    - The AWS Replication Agent installer should have access to the S3 bucket URL of the AWS Region you are using with Elastic Disaster Recovery.
    - The staging area subnet should have access to the Regional S3 endpoint.
    - The staging area subnet requires outbound access to the Amazon EC2 endpoint of its AWS Region.
- **Communication over TCP Port 1500**: All data replication traffic is transmitted between the Source servers and the staging area subnet using TCP Port 1500; this communication is also encrypted.
- **Bandwidth Requirements**: The average network bandwidth must exceed the peak write rate of the source servers to ensure successful replication in the AWS Elastic Disaster Recovery service. Adequate network capacity is critical to maintain continuous data protection and meet your recovery point objectives.

4. **Amazon S3 Buckets Reference**
    - **Access Requirements**: Agent installation and replication server components require Amazon S3 bucket access.
    - **VPC Endpoint Policy**: Ensure that the relevant VPC endpoint policy includes access to all required Amazon S3 buckets. Refer to the example policy for replicating to us-east-1 and Amazon S3 documentation for policy requirements.
5. **Operational Subnets https://docs.aws.amazon.com/drs/latest/userguide/Network-Settings-Preparations.html Reference**

- **Drill and Recovery Subnets**: Create Recovery subnets (and optionally Drill subnets), before attempting to launch Recovery Instances. Instances are launched in a subnet specified in the Amazon EC2 launch template associated with each source server.

6. **Supported Elastic Disaster Recovery AWS Regions Reference**
   - Refer to AWS Elastic Disaster Recovery supported Regions reference for an up to date list of all supported Regions.
7. **Supported Operating Systems https://docs.aws.amazon.com/drs/latest/userguide/Supported-Operating-Systems.html Reference**
   - Elastic Disaster Recovery supports many versions of Windows and Linux operating systems, some of which are not natively supported by Amazon EC2. Refer to Supported Operating Systems for up-to-date versions of supported operating systems.
8. **Windows Installation Requirements https://docs.aws.amazon.com/drs/latest/userguide/installation-requiremets.html#windows-requirements Reference**
   - **Supported Operating Systems**: Ensure that your source server operating system is supported.
   - **Free Disk Space**: At least 4 GB of free disk space on the root directory (C:by default).
   - **Free RAM**: At least 300 MB of free RAM.
   - **MAC Address Stability**: Ensure that the MAC addresses of the source servers do not change upon a reboot or any other common changes in your network environment. The AWS Replication Agent may use the MAC address in its process to link the source server to its replication infrastructure.
9. **Linux Installation Requirements https://docs.aws.amazon.com/drs/latest/userguide/installation-requiremets.html#linux-requirements Reference**
   - **Supported Operating Systems**: Ensure that your source server operating system is supported (referenced above)
   - **MAC Address Stability**: Ensure that the MAC addresses of the source servers do not change upon a reboot or any other common changes in your network environment. The AWS Replication Agent may use the MAC address in its process to link the source server to its replication infrastructure.
     a. **Python**: Python 2 (2.4 or above) or Python 3 (3.0 or above) must be installed on the server.
     b. **Free Disk Space**: At least 4 GB on the root directory (/), 500 MB on the /tmp directory.
     c. **GRUB Bootloader**: The active bootloader software must be GRUB 1 or 2.
     d. **/tmp Directory**: Mounted as read+write and with the exec option.

e. **Sudoers List**: The Linux account that is installing AWS Elastic Disaster Recovery needs to be in the sudoers list.

f. **dhclient Package**: Ensure that the dhclient package is installed.

g. **Kernel Headers**: Verify that kernel-devel/linux-headers are installed and match the running kernel version.

h. **Symbolic Link Considerations**: Ensure that the content of the kernel-devel/linux-headers is not a symbolic link.

   i. Sometimes, the content of the kernel-devel/linux-headers, which match the version of the kernel, is actually a symbolic link. In this case, you will need to remove the link before installing the required package.

      A. To verify that the folder that contains the kernel-devel/linux-headers is not a symbolic link, run the following command:

         I. On RHEL/CENTOS/Oracle: ``ls -l /usr/src/kernels` ``

         II. On Debian/Ubuntu/SUSE: ``ls -l /usr/src` ``

   ii. If you found that the content of the kernel-devel/linux-headers, which matches the version of the kernel, is a symbolic link, you need to delete the link.

      A. Run the following command: ``rm /usr/src/` ``

         I. For example: ``rm /usr/src/linux-headers-4.4.1` ``

i. **Kernel Headers Installation**: For the agent to operate properly, you need to install a kernel headers package with the exact same version number of the running kernel.

   i. To install the correct kernel-devel/linux-headers, run the following commands:

      A. On RHEL/CENTOS/Oracle/SUSE: `sudo yum install kernel-devel-`+uname -r+`` ` ``

      B. On Debian/Ubuntu: s__`udo apt-get install linux-headers-`+uname -r+`` ` ``

   ii. If no matching package was found on the repositories configured on your server, you can download it manually from the Internet and then install it. To download the matching kernel-devel/linux-headers package, navigate to the following sites:

      I. RHEL, CENTOS, Oracle, and SUSE package directory

      II. Debian package directory

      III. Ubuntu package directory

10**AWS Specific Considerations**

a. Number of disks per server

   i. Elastic Disaster Recovery uses Amazon Elastic Block Store and Amazon Elastic Compute Cloud for the replication infrastructure. Because of this, Elastic Disaster Recovery is limited to the amount of disks that can be added to the replication servers.

      I. For Nitro replication instances (such as t3.small), we are limited to source servers with less than 26 volumes

      II. For Xen replication instances (such as t2.small), the limitation is 40 volumes per source server

b. Maximum source disk size

   i. Elastic Disaster Recovery uses Amazon Elastic Block Store and Amazon Elastic Compute Cloud for the replication infrastructure. Because of this, Elastic Disaster Recovery is limited to the 16TB for each disk on the source machines being protected.

Design Guidance

Cross-Region

When deploying Elastic Disaster Recovery as a cross-Region approach, you will be protecting applications that are hosted in your primary Region by replicating data to a secondary Region that would also be used for recovery during a drill or disaster. When used for a drill, there will be no impact to the resources in your primary Region. You will continue to serve all users with your production resources with no changes.

However, when failing over during an actual disaster, your production resources will be active in the recovery Region, and you will have to adjust your network path through routing and DNS to redirect all requests to this new recovery Region. During this time, all new writes or changes to the data will occur in the recovery Region.

Once the disaster has been remediated, you may choose to failback to the primary Region. In order to support this failback operation, it is required that the Elastic Disaster Recovery service be initialized in the primary Region. We recommend that the primary Region be initialized for this operation when you are in the processes of setting up and initializing Elastic Disaster Recovery service in the recovery Region. This is so there is no added delay to the failback process. In the situation where you choose to maintain operations in the recovery Region, we still recommend that you identify a new Region that would be used to replicate data and protect this recovery region from any possible failures.

Security

Security needs to be a high priority, especially when it comes to your disaster recovery approach. Elastic Disaster Recovery has several options built directly into the service, however it does not provide a full security solution, and you should work with your security teams to validate your security posture.

Encryption In Transit

All data replicated by Elastic Disaster Recovery is encrypted in transit using TLS 1.2 or later.

Encryption at Rest

When AWS Elastic Disaster Recovery replicates data to the target AWS Region, it creates Amazon Elastic Block Store(EBS) volumes in a staging area. These volumes are automatically encrypted using an AWS Key Management Service (AWS KMS) encryption key that the service creates in your AWS account by default, providing data protection at rest. You can also choose an existing Customer Managed Key (CMK) or create one for this purpose if needed. The chosen key must be selected in the EBS encryption section of the replication settings for Elastic Disaster Recovery to use it. EBS volumes that are launched during a Drill or Recovery will be encrypted using the same key, unless otherwise specified in the EC2 Launch Template.

If you have specific compliance requirements, you can also use Customer Managed Keys instead of the default keys created by Elastic Disaster Recovery to handle the encryption of the staging volumes, as well as the volumes of Drill or Recovery instances.

Separate DR account

A best practice for Elastic Disaster Recovery is to use separate AWS accounts for the Elastic Disaster Recovery staging network (VPC and subnet) and recovery network. Using a separate AWS account specifically for your disaster recovery solution allows for better segmentation and separation of your critical replicated data.

Networking

Network Connectivity

Elastic Disaster Recovery can utilize multiple networking options when supporting a cross-Region use case. These options can include VPC peering, Transit Gateway, and Internet routing. Of these options, we only recommend that you consider VPC peering or Transit Gateway, as both of these networking solutions can support connecting VPCs together while providing access to services across Regions with better performance and greater security. Of these options, we recommend VPC peering as a simpler and low-cost way to connect your primary Region to the recovery Region while allowing replication traffic to travel between the source server and staging environment.

Network bandwidth

AWS Elastic Disaster Recovery will utilize as much of the network as possible when replicating the data from your source environment. Due to this, you will want to ensure you have enough bandwidth to support your source change rate (ensuring you can maintain Continuous Data

Protection). You will want to monitor your network to ensure there is no congestion being caused by the replication process. If you need to throttle the Elastic Disaster Recovery service, you can do so at the service or machine level. In order to calculate the bandwidth required for your particular workloads, refer to Elastic Disaster Recovery Calculating Bandwidth.

Installation

Ensure Instance Profile Permissions

- Verify that the instances you want to protect have an instance profile with the following policies:
  - AmazonSSMManagedInstanceCore
  - AWSElasticDisasterRecoveryEC2InstancePolicy
- If the instance profile is not present, you can create the default instance profile by following these steps:

  a. Go to the **Instance profile** role **installation** section.

  b. Select the **Install default IAM role** button to create the default instance profile.

Assign Instance Profiles:

- In the **Instance profiles** section, verify that all the instances you want to protect have the required instance profile assigned.
- If any instances do not have an instance profile, you can assign the default instance profile by selecting **Attach profiles to all instances**.

Set Target Disaster Recovery Region:

- In the **Target disaster recovery region** section, select the AWS Region where you want to set up the disaster recovery.
- If the selected Region is not initialized for Elastic Disaster Recovery, select **Initialize and configure Elastic Disaster Recovery** to set it up.

NOTE: As this guide is based on a cross-Region deployment pattern, the Recovery Region you select should be different than the source Region where your source EC2 instances are deployed

Protect instances with Elastic Disaster Recovery:

- In the **Add instances** section, select **Add instances**.
- Elastic Disaster Recovery will list all the instances that are currently managed by AWS Systems Manager and will attempt to install the AWS replication agent on them.

- Once the AWS replication agent is successfully installed, the instances will be added as source servers to Elastic Disaster Recovery.

Monitor the Process:

- In the **Add instances result** page, you can view the progress and status of the AWS replication agent installation on the instances.
- For instances where the installation was successful, you can find a link to the source servers page in the **Details** column.
- For instances where the installation failed, you can find a link to the run log on the AWS Systems Manager console.

Verify Instance Management by AWS SSM:

- After attaching the instance profile, allow a few minutes for Elastic Disaster Recovery to detect if the instances are managed by AWS Systems Manager.
- The marker near the instance ID will change to indicate if the instance is currently managed by AWS Systems Manager.

Remember, if there are instances that are not managed by AWS Systems Manager, you will need to install the Systems Manager agent on those instances and then attach the appropriate instance profile before adding them to Elastic Disaster Recovery. Should you not wish to utilize Systems Manager for this process, refer to Adding Servers for instructions on how to manually install the AWS Replication agent. If you are using a third-party software deployment process, consult with the team that manages it to find if it can be used to deploy AWS Elastic Disaster Recovery.

- When installing the AWS Replication Agent, you may run into unforeseen installation issues based on multiple factors. Refer to **Advanced topics** in the Resources section of this guide for common issues that can be encountered during the installation process.
- If you are unable to resolve the using the information provided, create an AWS support ticket and include the following:
  - What part of the installation process is failing
  - Confirmation that you have followed the troubleshooting guide

- Attach the agent log from that specific server. The agent log can be located at the [following locations](#):

  a. **Linux**: /var/log/awsdrs-agent/agent.log

  b. **Windows**: C:.log

- Once the installation has completed, the Elastic Disaster Recovery console will show the following stages:
  - Initiating
    - This shows that the agent has been installed successfully on the source server, and Elastic Disaster Recovery is now moving on to the next steps of configuring replication for that server. To see what step the service is currently on, select the server name, and check under *Data replication status,* as shown in the user guide under [Recovery Dashboard](#).
  - Initial sync | time left
    - This is the amount of the known blocks that will be replicated.
      - Note that you may see the time left for replication fluctuate by large margins. This is due to how reading block storage is accomplished and we are unable to predict how many future blocks may need to be replicated.
      - You can estimate the amount of time required to complete this step by analyzing the amount of storage that needs to be replicated and the available bandwidth available to transmit this data.
    - During this initial sync process, you may see *backlog* in the same line.
      - Backlog is the amount of new data that has been written and waiting to be added after initial sync. Once initial sync has completed, you will see the backlog amount start to reduce as the agent replicates those newer blocks.
  - Initial sync 100% done | Creating Snapshot
    - All blocks have been replicated from the source machine to the staging area and we are now creating the baseline EBS snapshot for that volume.
      - **Note**: if the service is stuck in the stage for a long time, confirm the replication server has 443 outbound access to regional EC2 endpoint.
  - Healthy
    - All data has been replicated to the staging area, and the replication server has enough bandwidth to replicate the changes being generated at the source environment. *Should there be an issue with the replication process after you have completed the initial sync phase, you will see an error in the same location.*

- Other states you might see are:
  - Rescan
    - This means that something has interrupted the agents ability to validate the block map, usually caused by an unplanned reboot of the source machine (such as a power outage, pulling the plug, or terminating an EC2 instance)
  - Lag
    - Lag is the amount of time since the server was last in continuous data protection (CDP) mode. Lag typically leads to backlog, which is the amount of data that has accumulated and still needs to be replicated. The longer the lag, the larger the backlog that needs to be cleared.
    - This can be caused by many items, and troubleshooting steps can be found in the userguide under Replication Lag Issues.
    - Potential solutions:
      - Make sure that the source server is up and running.
      - Make sure that AWS Elastic Disaster Recovery services are running on the source server.
      - Make sure that TCP Port 1500 is not blocked outbound from the Source server to the replication server.
      - If the MAC address of the Source had changed, that would require a reinstallation of the AWS Replication Agent.
      - If the source machine had a spike of write operations, the lag will grow until AWS Elastic Disaster Recovery service manages to flush all the written data to the drill or recovery instance replication server.
  - Backlog
    - Backlog is the amount of data that was written to the disk and still needs to be replicated in order to reach CDP mode. backlog can also occur without lag. This can happen due to various reasons, such as:
      - Temporary network interruptions or bandwidth limitations that prevent the data from being replicated in real-time.
      - Spikes in data volume that exceed the processing capacity of the system, leading to a backlog.
      - Scheduled maintenance or other operational activities that temporarily pause the replication process.

- Even if there is no lag, meaning the server or service is in the desired state, a backlog of data can still build up that needs to be processed. For example, a server generating traffic at a lower rate than the network bandwidth, resulting in no lag, but there could still be a backlog of data that needs to be replicated. *Once the installation process has been completed across all needed servers, you can move on to the next section, where you will configure monitoring and notifications*.

Monitoring AWS Elastic Disaster Recovery

Monitoring resources

Monitoring will play a critical role when defining a DR strategy. The ability to observe, monitor, and alert on resources and system performance at multiple levels is required to operationalize your plan.

Configure replication monitoring and alerting

Elastic Disaster Recovery can utilize Amazon CloudWatch to assist with monitoring. CloudWatch is a monitoring service that helps you monitor AWS resources as they are being consumed within your account. When these two services are integrated, you can monitor Elastic Disaster Recovery events with CloudWatch to build a customizable and detailed dashboard for Elastic Disaster Recovery. These services can be extended further with Amazon EventBridge and Amazon Simple Notification Service (Amazon SNS), to get real-time alerts and automate responses.

Creating CloudWatch dashboards to monitor Elastic Disaster Recovery

You can visualize and share your metrics using CloudWatch dashboards.There are many metrics available within CloudWatch to help you monitor and manage the state of your disaster recovery operations. With CloudWatch, you can include metrics to monitor your source server count, time since last successful test, and lag of source servers (when the Elastic Disaster Recovery service is no longer in continuous data protection mode and should be investigated for root cause). We recommend using CloudWatch to setup dashboards and notifications to alert you on any possible replication issues. Follow the steps below:

1. Navigate to the Amazon CloudWatch dashboard.
2. Under Dashboards, select **Automatic dashboards**
3. Filter for and select **Elastic Recovery Service**
   a. You will be taken to a default dashboard that monitors several aspects of Elastic Disaster Recovery. These metrics are based on the replication instances you have running in the AWS Region you currently have selected:
      i. LagDuration: Average

      A. This is the average time of "Lag" on your replication severs. Anything higher than 0 should be investigated for possible issues, but we recommend monitoring for lags larger than an hour (or your RPO if close to an hour).

    ii. Backlog: Average

      A. This is the average amount of "backlog". Backlog is generated when the service is unhealthy, but is still seeing data being written to source, that it is unable to replicate

    iii. DurationSinceLastSuccessfulRecoveryLaunch: Maximum

      A. This is the maximum amount of time since the last successful launch of DRS machines

    iv. ElapsedReplicationDuration: Maximum

      A. This is the amount of time Elastic Disaster Recovery has been replicating data

    v. ActiveSourceServerCount: Average

      A. This is how many source servers have Elastic Disaster Recovery installed

    vi. TotalSourceServerCount: Average

      A. This is how many source servers have Elastic Disaster Recovery installed

    vii. ProtectedSourceServerCount: Average

      A. This is how many source servers have Elastic Disaster Recovery installed excluding disconnected and stopped servers

4. Choose **Add to dashboard**

   a. You can either select an existing dashboard, or choose **Create new**

      i. If you decide to create a new dashboard, you will be taken to the next screen to enter a name and select **Create**

   b. Select **Add to dashboard**

You will now have a dashboard monitoring Elastic Disaster Recovery under your **Custom dashboards** section in CloudWatch.

Configuring your Amazon Simple Notification Service Topic

Amazon SNS will be used to alert a specific inbox or distribution list when any AWS Elastic Disaster Recovery source machines are experiencing a stalled replication that must be addressed. Doing so will help to identify and remediate issues quicker, so that your RPO goals can be maintained. Stalled replication is the main indicator of replication issues and can indicate multiple issues.

1. Navigate to Amazon Simple Notification Service.

2. Choose **Create Topic**.
    a. Under **Details** and **Type**, choose **Standard**.
    b. Under **Name** enter a name for this topic. (for example: drs-replication-monitoring)
        - *Optional:* Enter a display name for SMS messages to mobile devices.
            - Note: As of June 1, 2021, US telecom providers no longer support person-to-person long codes for applications-to-person communications. See the Amazon SNS Developer Guidefor more information.
        - *Optional:* For Tags, enter a key-value pair for easy identification later.
            - Select **Create topic**.
            - Once the topic is created, select *drs-replication-monitoring* from the list.
            - Choose **Create subscription**.
            - Validate that the Topic ARN under **Details** is the same as drs-in-lag.
            - From the Protocol dropdown, choose **email**.
            - Under Endpoint add the email or distribution list to receive these alerts.
            - Choose **Create subscription**.

Create a rule using the console

The next step is to configure Amazon EventBridge to monitor for specific Elastic Disaster Recovery events related to replication health. Should EventBridge receive an event for unhealthy replication status for Elastic Disaster Recovery, it will notify the Amazon SNS topic. This, in turn, notifies the subscribers of that topic.

1. Open Amazon EventBridge
2. Choose **Create rule**.
3. Under **Name** and **Description**, enter the name for this rule (we use "drs-replication-monitoring" for this step).
4. Under Define pattern, choose **Event pattern**.
5. Select Pre-defined pattern by service.
6. From the dropdown menu for Service provider, choose **AWS**.
7. Under the Service name dropdown, choose **Elastic Disaster Recovery Service**.
8. Under Event type, choose **DRS Source Server Data Replication Stalled Change**.
9. Under Select targets and Target, choose **SNS topic**.

10. For Topic, choose the SNS topic created earlier: **drs-replication-monitoring**.
11. Choose **Create**.

You have now created a dashboard that will monitor your Elastic Disaster Recovery replication infrastructure, and notify you if there are any stalled replication servers that would cause you to miss your RPO.

Monitoring Costs

Configure cost monitoring

There are several configuration strategies possible with Elastic Disaster Recovery. Understanding what makes up the associated costs of using Elastic Disaster Recovery is an important consideration when deciding how to further optimize the system for performance vs cost while maintaining your resilience objectives. This may include decisions on retention periods, Region selection, network design, and infrastructure configurations.

The following section provides the steps to activate cost allocation tags, creating and saving a custom report, and exporting the report data. This will provide insight into the overall costs of the Amazon EC2, Amazon EBS, and EBS snapshot resources provisioned by Elastic Disaster Recovery.

Activate cost allocation tags This section walks through the process of enabling user-defined cost allocation tags for Elastic Disaster Recovery. Once enabled, you can use these tags on your cost allocation report to track costs.

1. Log in to the AWS Management Console and search for **Billing and Cost Management**.
2. Locate and select **Cost Allocation Tags**.
3. Under **User-defined cost allocation tags**, find the AWSElasticDisasterRecoveryManaged tag.
4. Select the checkbox for this tag, and choose **Activate**.
5. Choose **Activate** from the pop up. It may take a couple of hours before the tags are available.

Where to optimize

- Use default instance types for replication servers unless source servers are often in lag
- Use automated disk type
- Lower snapshot retention to minimal needs

Elastic Disaster Recovery should not be used for long term retention; use a backup solution for long term storage and archiving

Create cost categories

This section walks through the process of creating cost categories. This allows you to map Elastic Disaster Recovery costs and usage into meaningful categories using a rules-based engine.

Add rule

1. Log in to the AWS Management Console and search for **Billing and Cost Management**.
2. Locate and select **Cost Categories** and select **Create cost category**.
3. Provide a Name (for example, DRSCost) to the cost category and select **Next**.
4. Choose Rule type as Inherited value, and Dimension as Cost Allocation Tag, Tag key as AWSElasticDisasterRecoveryManaged, and select **Add rule**.

Modify rule

1. Choose **Rule type** as **Regular** and **DRSCost *Value*** as **License**.
2. Under **Dimension 1**, choose **Service**. For **Operator** select **Is** and for **Service Code** choose **AWSElasticDisasterRecovery**.
3. Select **Next**.
4. Select **Create cost category**. It will take up to 24 hours for the cost category to be available in AWS Cost Explorer.

Create a Cost Explorer report

This section walks through the steps to create a customized Cost Explorer report for Elastic Disaster Recovery. It uses the filters and cost categories created in the preceding section.

Create report

1. Log in to the AWS Management Console and search for AWS Cost Explorer.
2. Open the AWS Cost Management dashboard and select **Cost Explorer**.
3. Locate **Cost Organization** and select Cost Category.
4. Select the cost category (for example: DRSCost) which was created in the Create Cost Categories section.

5. Select two checkboxes: License and drs.amazonaws.com
6. Select **Apply Filters**.

Select filters . On the top left, select the **Group by: Usage Type**. . Locate time ranges and select the time for which you would like to see the data. In the following example, we set it to **Last 7 Days** with the time granularity as **Daily**. . Select **Bar** style type for the chart. You will see the cost breakdown of the staging area. This includes the cost of replication servers, Amazon EBS volumes, Amazon EBS snapshots, as well as other services and AWS resources. . Locate and select **Save as** and assign the new report a name. For example, `0`. . Select **Save Report**.

View and export saved Cost Explorer report

This section walks through the steps to view the Cost Explorer report and export it to a CSV file.

1. Log in to the AWS Management Console.
2. Search for AWS Cost Explorer and open the **AWS Cost Management** dashboard.
3. Select **Reports**.
4. Select the report that was previously saved. The total cost of Elastic Disaster Recovery is included in that report.
   a. The report can be further customized by using the **Group by** options near the top or any of the other filters available in AWS Cost Explorer.
5. Select **Download CSV** to export your data for further analysis.

How can you optimize costs?

- Utilize the default replication server instance types at first. Allow Elastic Disaster Recovery to replicate your initial dataset, then ensure no Source servers are stating that they are in "Lag". If there are any source servers that are in lag, follow the **Advanced Topics** section at the end of this userguide. This section may conclude that you need to increase the size and performance of the replication server or provide a dedicated replication server.
- Use the "Auto volume type selection" option for your replication servers
- When choosing **Auto volume type selection**, the service will dynamically switch between performance or cost optimized volume type according to the replicated disk write throughput.
- Lower snapshot retention to minimal length requirements. Based on the changed rate of your dataset, this can have a large impact on overall Elastic Disaster Recovery costs.

Note that if you have compliance requirements and require your snapshots for long-term retention, you should use a long-term storage and backup solution like AWS Backup. Elastic Disaster Recovery is not intended to act as a backup or archive storage service, and hence is not a suitable solution for long-term retention of your snapshots and other data

Cost Optimization

There are multiple configuration strategies possible with Elastic Disaster Recovery. Understanding what makes up the associated costs of using Elastic Disaster Recovery is a key step in targeting efforts to reduce cost without sacrificing resilience. This includes things like using the most relevant resilience strategy and retention periods, driving redundancy, selecting the Region, and right-sizing your infrastructure.

The method to reduce operational costs when using Elastic Disaster Recovery is to perform a combination of the following:

❏ Evaluate the retention period required for point-in-time snapshots. How far into the past do you need to retain the ability to do a full server restore, as opposed to restoring from a backup? Make sure to consider applicable compliance and regulatory requirements.

❏ For those servers being covered by Elastic Disaster Recovery, consider whether there are redundant drives mounted that are no longer in use and do not need to be replicated. These can either be unmounted or excluded when installing the replication agent.

❏ Right-size the target failover infrastructure by selecting the appropriate Amazon EC2 instance type in the EC2 launch template. You can use the instance right-sizing feature to map to an instance type that closely follows the source infrastructure, however you should use operational data in the source environment to right size these resources.

The size of the underlying disks (for example, the entire disk and not just partitions) directly dictates the amount of data that is replicated over into AWS during the initial sync process. As a result, right sizing and being selective of workloads, as per RPO and RTO objectives, gives the benefit of both monetary and saving time.

Resources:

- AWS Well-Architected Framework: Cost Optimization Pillar
- AWS Well Architected Framework: Performance Efficiency

Drill Planning for AWS Elastic Disaster Recovery

Drills vs Planned DR Events: In some situations, the disaster recovery test will be a failover of the production environment in a planned event, with apps running in production in the recovery Region. It is advised to do a full production test on an annual basis to capture any blockers, and to be familiar with the process in the event of an actual disaster.

Testing your disaster recovery implementation is the only way to validate that your RPO and RTO objectives can be met when a real disaster occurs. Elastic Disaster Recovery natively supports the ability to launch drills without affecting your production environment. However, conducting a drill and launching a server as an EC2 instance is not adequate to declare success. It's important to test at an application or business process level to ensure that the end-to-end service can be delivered when the disaster recovery plan is activated. It is a best practice to perform drills regularly. There are a few things to note before launching a Elastic Disaster Recovery drill:

- When launching a drill or recovery, you can launch up to 500 source servers in a single operation. Additional source servers can be launched in subsequent operations.
- It is a best practice to perform drills regularly. After launching drill instances, use either SSH (Linux) or RDP (Windows) to connect to your instance and ensure that everything is working correctly.
- Take into consideration that once a drill instance is launched, actual resources will be created in your AWS account and you will be billed for these resources. You can terminate the operation of launched Recovery instances once you verify that they are working properly without impact to data replication.
- We recommend that you test as often as possible. Test once a year at a minimum, even if it means reducing scope and testing a portion of the application or business function portfolio. This ensures the team is comfortable with the disaster recovery plan while also allowing them to identify any issues or required changes.

When preparing for a disaster recovery test, it is critical to ensure that your Drill environment is configured properly. A drill will be conducted while the production environment remains intact. In order to minimize impact to the production environment we recommend the following:

- Network Considerations
  - Subnet configuration
    - CIDR range

        - You will want to ensure that your Drill subnet is configured with the same CIDR range size as your Failover subnet. This will ensure that the subnets are sized properly and any IP adjustments to the Drill/Failover machines remains the same.
        - With this in mind, you will want to ensure that the subnet where you are launching Drill instances is in an isolated network with no route to the source environment or production systems. This will ensure there are no IP address or routing conflicts during testing. We also recommend configuring security groups and access control lists to further reinforce these boundaries.
    - Routing
        - If your Drill requires access to services or dependencies outside of the Drill subnet, you should ensure the appropriate routing policies and rules are configured in the Drill subnet to support this connectivity.
        - Updating Launch Template to the Drill subnet
            - By default, you will want to have the Launch Templates configured for your Failover subnet. During a Drill, you will need to change that section of the Launch Template to the Drill subnet. Refer to the EC2 launch template for steps to complete. Additionally, launch settings can be changed for a single server or for multiple servers through the Elastic Disaster Recovery console. This option allows you to quickly make changes to multiple servers at once. Refer to Configuring launch settings in AWS Elastic Disaster Recovery for more details on making bulk changes to your Launch Templates.
- Infrastructure Services (such as AD and DNS)
    - Depending on the criteria for a successful Drill, you may need your Drill servers to connect to services such as Active Directory or other infrastructure services in order to complete a Drill. This might require additional scripting (or usage of appropriate SSM documents to automate the usage of AD after launch)
        - With Elastic Disaster Recovery, you can replicate all applications and services, including Active Directory. With this approach, it is recommended to launch the drill version of AD first and wait until the service is up and running. Once the service is up, you can start to launch the other applications or servers. This will ensure that the AD servers are ready to provide critical functions and services like authentication and authorization.
        - An alternative approach is to extend Active Directory to the Drill subnet. It is advised to work with your system administrators to define the best method for your use case.

Prior to launching a drill instance, ensure that your source servers are ready for testing by looking for the following indicators on the **Source servers** page:

1. Under the **Ready for Recovery** column, the server should show **Ready**. This means that the initial sync has been completed and all data from the source server has been replicated to AWS.
2. Under the **Data Replication Status** column, the server should show the **Healthy** status, but you can also launch the source server if the system is undergoing **Lag** or even **Stall**, but in that case the data may not be up to date. You can still launch a drill instance from a previous Point In Time.
3. Under the **Pending Actions** column, the server should show **Initiative Recovery Drill** if no drill instances have ever been launched for the server. Otherwise, the column will be blank. This helps you identify whether the server has had a recent drill launch.

Launching drill instances

To launch a drill instance for a single source server or multiple source servers:

1. Go to the **Source servers** page and check the box to the left of each server for which you want to launch a drill instance.
2. Open the **Initiate recovery job** menu and select **Initiate drill**.
3. Select the Point in time snapshot from which to launch the drill instance for the selected source server. You can either select the **Use most recent data** option to use the latest snapshot available or select an earlier specific Point-in-time snapshot. You may opt to select an earlier snapshot in case you wish to return to a specific server configuration before a disaster occurred.
4. After you have selected the Point in Time snapshot, select **Initiate drill**.

The Elastic Disaster Recovery Console will indicate **Recovery job is creating drill instance for X source servers** when the drill has started.

Choose **View job details** on the dialog box to view the specific Job for the test launch in the **Recovery job history** tab.

Successful drill instance launch indicators

You can tell that the Drill instance launch started successfully through several indicators on the **Source servers** page.

1. The **Last recovery result** column will show the status of the recovery launch and the time of the launch. A successful drill instance launch will show the **Successful** status. A launch that is still in progress will show the **Pending** status.
2. The launched Drill instance will also appear on the **Recovery instances** page.

Recovery planning

In order to launch your recovery instances quickly, you should pre-configure how those instances are to be launched. Also, perform drills in order to make sure that all of your network and application settings are properly configured. You can configure how your instances will be launched by editing the Launch settings for each source server. Launch settings can be configured immediately when a source server has been added to Elastic Disaster Recovery, there is no need to wait for the initial sync process to finalize. Performing frequent drills is key for failover preparedness. Elastic Disaster Recovery makes it easy for you to launch drill instances as frequently as you want. Drills are non-disruptive and do not impact the source server or ongoing data replication. If you experience a disaster in the middle of a drill, you can launch a new recovery instance from the source server's current state or keep the instance you launched during the drill.

Preparing for recovery

1. Configure your launch templates for each server you want to protect.
2. Under the **Ready for recovery** column, the server should show **Ready**. This means that the initial sync has been completed and all data from the source server has been replicated to AWS.
3. Under the **Data replication status** column, the server should show the **Healthy** status, but you can also launch the source server if the system is undergoing **Lag** or even **Stall**, but in that case the data may not be up to date. You can still launch a drill instance from a previous Point In Time.
4. Under the **Pending actions** column, the server should show **Initiative recovery drill** if no drill instances have ever been launched for the server. Otherwise, the column will be blank. This helps you identify whether the server has had a recent drill or recovery launch.

Performing recovery

Prior to launching a Recovery instance, ensure that your source servers are ready for a Recovery by looking for the following indicators on the **Source Servers** page:

1. Under the **Ready for recovery** column, the server should show **Ready**.
2. Under the **Data replication status** column, the server should show **Healthy** status.
3. Under the **Last recovery result** column, there should be an indication of a successful drill or recovery instance launch sometime in the past. The column should state **Successful** and show

when the last successful launch occurred. This column may be empty if a significant amount of time passed since your last drill instance launch.

To launch a recovery instance for a single source server or multiple source servers, go to the **Source servers** page and check the box to the left of each server for which you want to launch a recovery instance.

1. Open the **Initiate recovery job** menu and select **Initiate recovery**.
2. Select the Point in time snapshot from which to launch the recovery instance for the selected source server. You can either select the **Use most recent data** option to use the latest snapshot available or select an earlier specific Point-in-time snapshot. You may opt to select an earlier snapshot in case you wish to return to a specific server configuration before a disaster occurred. After you have selected the Point in Time snapshot, choose **Initiate recovery**. Learn more about Point in Time snapshots. in the userguide.
3. The Elastic Disaster Recovery Console will indicate **Recovery job is creating recovery instance for X source servers** when the drill has started.
4. Select **View job details** on the dialog to view the specific Job for the test launch in the **Recovery job history** tab.

Note Elastic Disaster Recovery is only one part of your disaster recovery plan. There are likely to be many other dependencies and services that will play a role in recovering from a disaster, and this should be factored in when conducting a drill or actual failover.

Group Launch Templates

Amazon EC2 launch templates control how instances are launched in AWS and each source server has its own launch template. You can edit the launch templates for multiple source servers at once by selecting the relevant servers on the Source servers page, then choosing **Edit EC2 launch template** from the **Actions** dropdown menu.

Important - to edit the launch template, automated launch settings, or to conduct Instance type right-sizing, the DRS launch settings must first be set to **Inactive** else you will receive an error.

Note: The DRS Template Manager is an open source solution available on GitHub that can automate management of launch templates with the use of a single JSON file as a baseline template. This file can be replicated, edited, and used for each source server tagged with a corresponding key in the DRS console.

Failback with AWS Elastic Disaster Recovery

Prerequisites

1. To ensure operational continuity, initialize the AWS DRS in advance in both the source and target AWS Regions, and conduct regular failover and failback drills.
2. Assign *AWSElasticDisasterRecoveryRecoveryInstancePolicy* to the IAM Role of our EC2 Instances. This IAM Policy is used to secure the permission policies needed to communicate with *Elastic Disaster Recovery Service* API during failback.
3. Before starting a failback, make sure the EC2 recovered instances have a network interface while meeting the specified network requirements.
4. Access to EC2 instance metadata is required. If you have a custom network setup that modifies the operating system route, ensure that access to the metadata is intact. Learn how to verify metadata access for Linux and for Windows.
5. EC2 Instances that have failed over must resolve through DNS the Regional DRS endpoint of the failback Region. The resolved endpoint must be accessible from the EC2 Instance through TCP 443.

Initializing Failback

To initialize failback, you need to **start reverse replication** process from the DR Region by following the below steps:

1. Go to the recovery AWS Region.
2. Choose the **AWS Elastic Disaster Recovery** service.
3. Navigate to the **Recovery instances** page.
4. Select the servers that you want to protect and select **Start reversed replication**.
5. You should now see a new Source server in the DRS Console in the source Region.

Note: . All server data is transferred over the wire during this step, resulting in cross-Region data transfer costs. . Starting a reversed replication creates additional replication resources. To avoid double billing, you can stop replicating the source instances by navigating to the AWS DRS source server in the recovery Region and selecting **Stop replication** in the replication drop-down menu. . **If replication is stopped, all previous points in time are deleted.** This is done to minimize costs.

Complete Failback

After the **Reversed direction launch state** is marked as **Ready**, take the following steps to complete the failback:

1. Find the relevant source servers by selecting the **Replicating to source server** link in the recovery instance **(or)** by directly navigating to the **Source servers** page in AWS DRS console at the source Region.
2. If the state is **Ready** (or **Ready with lag**), select **Launch for failback** under **Initiate recovery job**.
3. Redirect traffic to failed back instances, which will now become your new primary instances. Traffic redirection is not conducted using DRS.
4. Choose a service according to your preferences (consider using Amazon Route 53).

Note:

1. Make sure that your applications are working as expected. If you run into any issues, you can relaunch the instances and try again. Until you opt to failback, your recovery instances will continue to run in your recovery AWS Region to ensure business continuity.

Protect new Failed back instances

Do not perform this step when performing a **drill**. This step **replaces** the instances that AWS DRS replicates (from the Source instances to the failed back instances). In a drill, the source instances are still your production environment.

The newly launched failed-back instances are not protected. In order to protect them, follow these steps:

1. Navigate to the recovery instance in the source Region.
2. Select **Start reversed replication**. This step will replace the Instances that the Source Server protects.

Advanced topics

Recovery Plans: AWS Step Functions + Elastic Disaster Recovery API + AWS Lambda

When performing a disaster recovery at scale, there are often servers that have dependencies on other servers in the environment. For example, application servers that connect to a database on boot or servers that require authentication and need to connect to a domain controller on boot to start services. With AWS Lambda, AWS Step Functions, and Elastic Disaster Recovery API you can sequence your disaster recovery launch.

You can sequence your disaster recovery launch to work based on a single API call to execute the state machine. On this architecture, Lambda functions are used to call on the Elastic Disaster Recovery API and launch the recovery instances. Tagged servers being protected by Elastic Disaster Recovery are used by Step Functions to trigger the launch sequence.

Step Functions is a serverless orchestration service that lets you combine AWS Lambda functions and other AWS services to build business-critical applications. Through the graphical Step Functions console, you see your application's workflow as a series of event-driven steps.

Network Replication

The network replication feature in AWS Elastic Disaster Recovery automatically tracks and replicates changes to your network configurations, such as security groups, network ACLs, and routing tables, between your source and recovery environments. This helps prevent configuration mismatches during recovery, ensuring your recovery instances are launched with the correct network settings.

For example, if you update a security group to allow additional access, the network replication feature will automatically apply that change to the corresponding security group in your recovery environment. This maintains consistency between your source and recovery environments, enhancing security and reducing the risk of issues during failover. Beyond security groups, the feature also replicates changes to other network resources, like network ACLs and routing tables. By automating these updates, AWS Elastic Disaster Recovery helps you maintain compliance and avoid the need to manually configure individual launch templates for your recovery instances. Steps to implement the replication of your source network can be found in the Adding source networks to Elastic Disaster Recovery part of the user guide.

Post Launch Validation Automation

Post-launch Actions in Elastic Disaster Recovery allow you to automate actions after a Drill or Recovery instance is launched. These settings are based on the Default post-launch actions. Available post-launch actions include:

- Process status validation: Helps ensure critical processes (such as database and application services) are in a running state after the instance is launched. You can specify a list of processes to verify; you can also specify how long the system should wait before testing.
- EC2 connectivity checks: Conducts network connectivity checks to a predefined list of ports and hosts to ensure the instance can communicate as expected.
- Volume integrity validation: Helps ensure the launched EBS volumes are the same size as the source (rounded up), properly mounted on the EC2 instance, and accessible.

You can also run any available SSM document, including public, custom, or shared documents. To create, edit, or delete custom actions, make sure post-launch actions are activated for the source server. Custom actions are automatically added to new source servers.

Network Design using VPC Peering

If you want your disaster recovery setup to operate within a private and secure network, VPC peering is an excellent option. VPC peering enables secure, high-bandwidth, low-latency communication between VPCs in different AWS Regions without traversing the public internet. This design ensures that data replication for AWS Elastic Disaster Recovery is efficient and secure. Using VPC peering enhances disaster recovery capabilities, maintains compliance, and provides seamless failover and failback operations, ensuring robust protection for EC2 instances across Regions.

To set up VPC peering for your VPCs, visit the AWS VPC Peering Guide.

For more insights, refer to these blogs:

1. Cross-Region AWS Elastic Disaster Recovery agent installation in a secured network
2. Private cross-Region disaster recovery with AWS Elastic Disaster Recovery

Network Design using Transit Gateway

For a scalable and centralized network design, AWS Transit Gateway offers a powerful solution. Transit Gateway enables you to connect multiple VPCs across different AWS Regions through a single gateway, streamlining your network architecture.

By using Transit Gateway, your disaster recovery setup benefits from a simplified, hub-and-spoke topology that reduces complexity and enhances security. This design ensures that data replication for AWS Elastic Disaster Recovery is efficient and secure, with the added flexibility to manage and scale your network easily. Using Transit Gateway also provides seamless connectivity and failover capabilities across multiple AWS accounts and Regions, delivering a robust disaster recovery protection for your critical workloads.

To learn more about setting up and configuring Transit Gateway, visit the AWS Transit Gateway Guide.

Conclusion and notices

This guide walked through design patterns for provisioning key Elastic Disaster Recovery resources, such as staging area subnets, recovery subnets, and EBS volumes. It also provides the necessary technical background to understand the core principles and implementation process for the various stages of Elastic Disaster Recovery. This ensures the provisioned resources remain robust, even when facing unforeseen obstacles.

This guide also covers observability best practices, highlighting the importance of leveraging the right tools and services. This includes monitoring replication health through metrics and alerts, as well as optimizing costs by analyzing usage data. These observability practices enable proactive actions based on the insights gained.Furthermore, this guide emphasizes the critical role of testing your disaster recovery strategy through regular drills. It discusses key considerations when performing an actual failover in response to a real disaster, as well as the process of restoring normal operations using the failback functionality.

Overall, this guide aims to serve as a comprehensive foundation for deploying and managing Elastic Disaster Recovery across the entire disaster recovery lifecycle. This empowers businesses to be prepared and resilient against all types of adversity.

Authors

Daniel Covey

Priyam Reddy

Contributors

Sravan Rachiraju

Stuart Lupton

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 AWS current 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. AWS responsibilities and liabilities 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.