Best Practices for AWS-to-AWS Disaster Recovery Strategies

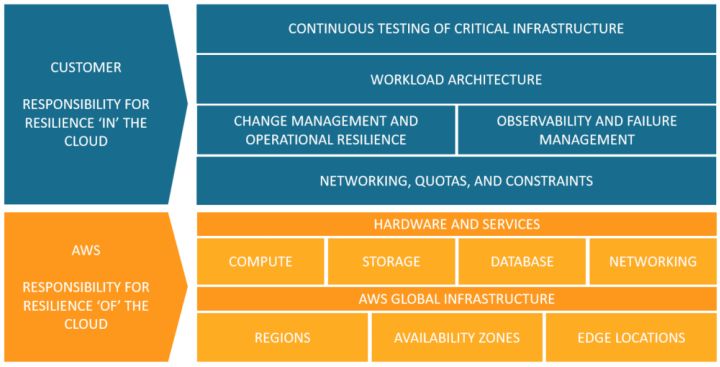
23 June 2023 - 7 min. read

Disaster Recovery (DR) *DR Strategy*

Read Part 1 | Read Part 2

This is the third and final chapter of the series on Disaster Recovery. We will analyze how the concept of disaster recovery varies and what techniques to adopt if our production infrastructure is already on the AWS Cloud.

AWS is responsible for the resilience of the physical infrastructure that underlies all the services offered, while the customer is responsible based on the selected services. For example, a service like Amazon Elastic Compute Cloud (Amazon EC2) requires performing all necessary configurations and management activities for resilience. In fact, it is the customer's responsibility to leverage the Availability Zones provided by AWS to achieve high availability. For managed services like Amazon S3 and Amazon DynamoDB, AWS manages the infrastructure, operating system, and platforms, while customers are responsible for the data, including backup strategies, versioning, and replication. This concept can be summarized in a few words by saying that AWS is responsible for **resilience in the Cloud**, and the **customer** is responsible for r**esilience IN the Cloud**.



Courtesy of AWS

Therefore having our workloads in the Cloud doesn't mean that we can forget about resilience and business continuity. On the contrary, we need to understand how the concept of disaster changes.

Type of Cloud disaster

Normally, different types of disasters are classified into various categories.

1. Hardware or software failures: These can include server failures, storage device failures, network switch failures, or critical system component failures.
2. Network outages: Connectivity disruptions can be caused by network issues, configuration errors, or disruptions from network service providers. In these cases, restoring connectivity is necessary to ensure access to Cloud resources and applications.
3. Natural disasters: Events such as earthquakes, floods, fires, or storms can cause physical damage to buildings or data centers, compromising the IT infrastructure. In these cases, restoring the IT environment in a geographically diverse area is necessary.
4. Human errors: Human errors, such as accidental deletion of critical data or misconfiguration of an application, can cause disruptions and data loss.
5. Cyberattacks: Cyberattacks, such as ransomware attacks, can compromise data and system security.

In the Cloud, in the event of catastrophic events, AWS takes care of restoring its physical infrastructure, while we need to ensure that our services remain active. For certain types of events, such as human errors and cyberattacks, we also need to be responsible for the restoration, as well as business continuity.

In the Cloud, cyberattacks take on a new dimension as this environment is programmable through public API calls. Properly configuring AWS IAM according to best practices (credential rotation, MFA, strong passwords, etc.) is helpful for prevention, but it is not enough. Sometimes, human errors can disrupt the scenario. For example, the theft of credentials by malicious actors who can then use a script to suspend the operation of our services.

By putting all these concepts together, we understand how crucial it is to design a well-thought-out Disaster Recovery plan from Cloud to Cloud.

Disaster Recovery AWS-to-AWS

Implementing disaster recovery strategies on AWS involves utilizing **different AWS Regions** for all disruption events caused by hardware, network, and/or natural disasters. It also means concurrently using **different subscriptions** (referred to as Accounts by AWS) to be resilient against cyberattacks and/or human errors.

Let's first focus on selecting the secondary Region.

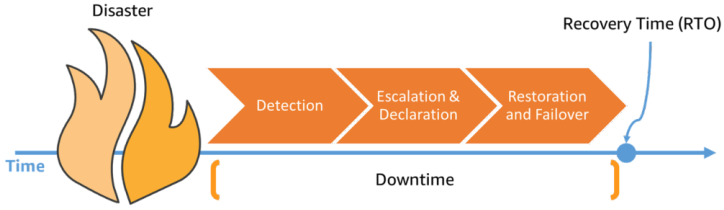
In this case, we need to keep in mind that **not all AWS services are available in every Region**. Therefore, before making a choice, we must ensure that all the components we utilize in our primary site are present.

Another point to consider is **service quotas**. Quotas are in place to prevent accidentally provisioning more resources than needed and to limit request rates on API operations to protect services from illicit use. Quotas can be increased by opening support tickets, providing justifications, and waiting for the provider's intervention. If we have requested quota increases in our production site, we need to remember to request them for the disaster recovery site as well; otherwise, we may risk having a functioning infrastructure that cannot handle the traffic.

Special attention, as always, should be given to **costs**. Both the selection of the Region, which is subject to different state legislations and tax regimes, and the data replication strategy, affect the economic impact.

Recovery phases

A good disaster recovery strategy, therefore, involves evaluating the environment, timing, and costs. The strategy is then divided into several phases: **detection**, **restoration**, and **failover**. For now, let's set aside the failback phase. These three phases should be considered in defining the RTO (Recovery Time Objective): after a disaster, it takes time to become aware of the event, additional time to activate our resources, and further time to redirect the traffic.



Courtesy of AWS

In order to reduce the restoration time, the detection phase should be automated. We shouldn't wait for someone to notice the disaster; we need to proactively notify our customers.

For example, **Amazon EventBridge** supports various types of events across AWS services and also triggers for remediation. Among the supported events are those from CloudWatch Alarms that we can configure on the metrics we define. This allows us to have specific health checks based on the functioning of our workloads.

As mentioned earlier, there are events that directly impact AWS, which are notified to us through the Personal Health Dashboard. EventBridge also supports this type of event.

To restore the infrastructure in the recovery Region, it is highly recommended to use the Infrastructure as Code (IaC) paradigm. This includes **AWS CloudFormation** and **AWS Cloud Development Kit (AWS CDK)** for consistently creating the infrastructure in both the primary and secondary sites.

In the recovery Region (and account), we need to have all the basic configurations, from networking to images of our instances or containers. For EC2 instances, we can leverage Amazon Machine Images (AMIs) to encapsulate the operating system and necessary packages and automate their creation and distribution. Since AMIs contain snapshots of volumes, we can time the process to have a more up-to-date copy. The same principle applies to containers, and by utilizing Amazon Elastic Container Registry (ECR), we can publish the images and replicate them in both environments.

Data replication is also essential for restoration. For example, if we have files on S3, we can configure **cross-account and cross-region bucket replication** to keep the files in sync from both sides.

A small tip: the buckets must have different names, so remember to configure your application to use different endpoints in case of failover.

Regarding databases, we can use a similar technique as with EC2 instances by utilizing snapshots and copying them to the disaster recovery site. If our workload requires a very tight Recovery Point Objective (RPO), then we may need to rely on tools that keep the production database in sync with the secondary one.

In general, an excellent tool for implementing our DR strategy is **AWS Backup**, which allows us to configure backup frequencies and distribute them cross-region and cross-account.

Finally, during the failover phase, traffic is redirected to the DR infrastructure. In this case, **Amazon Route 53** comes to our aid, as we configure the DNS using a failover routing policy with health checks. Using health checks is crucial for automating the redirection of users to the secondary infrastructure.

Conclusioni

We have reached the end of this series of three articles dedicated to Disaster Recovery, where we have explored how to ensure business continuity in both hybrid environments and fully Cloud ecosystems.

In particular, we have seen how the use of AWS services in this context offers numerous advantages, including resilient, scalable, and secure infrastructures, advanced tools for data protection, and simplified resource and cost management.

However, as we have seen from this overview, there is a critical aspect that no provider, service, or technology can handle, which requires special attention from companies: the planning of a truly disruption-proof Disaster Recovery strategy. To plan the best strategy for their organization, it is important not only to understand the useful building blocks but also to fully grasp the concepts on which these strategies are based. This includes understanding the concept of shared responsibility and identifying the responsibilities within each organization, as well as following appropriate best practices to successfully address different types of disasters.

It should also be considered that each implemented strategy requires constant attention and proper risk management, and its maintenance should be an ongoing and evolving process.

Throughout this series, we have emphasized the criticality of these aspects, which are often underestimated in many organizations.

Do you have any experiences you would like to discuss? We look forward to hearing your testimonials!

Stay tuned for a new article on **Proud2beCloud**!

Read Part 1 | Read Part 2

Related resources:

- AWS Well-Architected Framework (White Paper)
- AWS General Reference

- What is a landing zone?
- Best Practices for Organizational Units with AWS Organizations
- What Is AWS Control Tower?
- Customizations for AWS Control Tower
- Single-sign-on con G Suite sulla console di Amazon Web Services.
- Authenticate AWS Client VPN users with AWS IAM Identity Center
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud

---

About Proud2beCloud

Proud2beCloud is a blog by beSharp, an Italian APN Premier Consulting Partner expert in designing, implementing, and managing complex Cloud infrastructures and advanced services on AWS. Before being writers, we are Cloud Experts working daily with AWS services since 2007. We are hungry readers, innovative builders, and gem-seekers. On Proud2beCloud, we regularly share our best AWS pro tips, configuration insights, in-depth news, tips&tricks, how-tos, and many other resources. Take part in the discussion!

Nicola Ferrari

Cloud Infrastructure Line Manager @ beSharp and AWS authorized instructor champion.I live my life one level at a time getting superpowers by collecting caffeine hidden here and there in my daily map. I'm a hardened internet surfer (yes, I surfed the whole internet... twice!) and tech-addicted with a passion for computers and networking. Building great IT things all nice and tidy contribute to achieving my main goal: the pursuit of perfection!

Simone Merlini

CEO and co-founder of beSharp, Cloud Ninja and early adopter of any type of * aaS solution. I divide myself between the PC keyboard and the one with black and white keys; I specialize in deploying gargantuan dinners and testing vintage bottles.
