Building Mission-Critical Financial Services Applications on AWS

April 2019

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's 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's products or services are provided "as is" without warranties, representations, or conditions of any kind, whether express or implied. AWS's 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.

Contents

Abstract

This whitepaper discusses the fundamental design patterns to build highly resilient applications for financial institutions on Amazon Web Services (AWS), to meet mission-critical application recovery requirements.

Resilient applications provide continuous service despite disruption. Events such as natural disasters, hardware failures, and human error can interrupt the continuity of an application or service. Financial institutions that do not design and plan for these failures risk application downtime and data loss. This in turn can result in revenue loss, legal and financial implications, impacts to reputation and brand, and customer dissatisfaction.

Financial institutions rely on AWS to provide resilient infrastructure and services. Our Financial Services customers can build their mission-critical applications using AWS services, in order to plan for potential failures, and to meet resiliency requirements.

Introduction

The technology systems of financial institutions (FIs) are complex, and highly interconnected–to each other, and to non-financial entities. Payment processing, trading and settlement, market data, custody and entitlement management, and financial messaging are examples of the types of programs FIs depend on for the proper functioning of the industry. Disruption to the systems of FIs and the vendors that support them creates risks to financial stability across the industry. FIs are subject to regulatory scrutiny, and this potential for disruption has resulted in stringent resiliency requirements.

FIs, including Systemically Important Financial Institutions (SIFIs),[1] must provably meet regulatory requirements for the resiliency of their mission-critical applications. This is true whether these systems are running in physical data centers or in a cloud environment. As FIs move mission-critical applications to the cloud, they have sought guidance for replicating, and improving the resiliency of, their Tier 1 systems. The applications of FIs are grouped in tiers based on the potential impact the business would experience if there is a disruption. Tier 1 applications are those considered vital to the operations of an organization, such as trading and settlement, transaction processing, and customer relationship management.

One of the tenets of good application design is to design for failure. As Amazon CTO Werner Vogels says, “Everything fails all the time.” Human operators can make mistakes; natural disasters can take data centers and electric grids offline; internet connections can be disrupted; servers, switches, disks, and software can fail. If an event disrupts an FI’s critical applications, the company may need to invoke their disaster recovery (DR) plan. These plans involve stakeholders across the technology, operations, and business teams working to bring the applications to life in an alternate site–restoring service as quickly as possible.

Amazon Web Services (AWS) offers a broad set of compute, storage, database, networks, security, content delivery, analytics, application, and deployment services, available globally, that FIs can use to prepare for disasters by designing highly resilient applications. The inherent application programming interface (API)-driven infrastructure of the AWS Cloud allows FIs to automate the development, deployment, and operation of their application infrastructure. With AWS services, application development teams can shift the organizational response to a disaster event from reactive to automated response and recovery from the failure.

This whitepaper presents technical guidance and thought processes for FIs to build their resilient applications and disaster recovery plans on AWS. This document can be used as a position paper, and can be presented at the CXO or board level to prove the viability of hosting Tier 1 applications on AWS.

Risk and Resiliency in Financial Services

The Financial Services industry is one of the most critical and heavily regulated industries, requiring resilient applications to serve businesses and consumers across the globe. Economies of the world, as well as individual customers, and organizations of all sizes, are dependent on financial systems that are expected to be available even during a disaster event.

According to the Financial Stability Board (FSB), an international standard-setting body that coordinates with other international standard-setters, national central banks, regulators, and finance ministries, "Risk management is a critical first line of defence in the resilience of financial institutions. The FSB, standard-setting bodies (SSBs) and national authorities are working to strengthen risk management practices, including through increased regulatory and supervisory focus as well as additional guidance on firms' risk culture and governance practices."[2]

Modern Resiliency Requirements

After the events of September 11, 2001 led to disruptions of the global financial system, regulators began to introduce significant changes to resiliency requirements. These regulatory changes began in the United States[3] and were later adopted by the broader Financial Services industry globally.[4]

In 2003, U.S. financial regulatory agencies (the Federal Reserve, the Office of the Comptroller of the Currency [OCC], and the Securities and Exchange Commission [SEC]) introduced a required recovery time objective of two hours for the most critical applications.[5] Then, following the 2008 global financial crisis, the FSB created the Systemically Important Financial Institution (SIFI) Framework. This set of policies is intended to reduce the likelihood that a SIFI will fail, and minimize the impact of SIFI failure on the broader economy if such a failure occurred.[6] The Framework's multipronged measures include requirements for higher capital and liquidity, recovery and resolution regimes, intensified supervision, and stronger core financial infrastructures.

In addition to the SIFI Framework, the Basel III: international regulatory framework for banks[7] was developed after the 2008 global financial crisis. The FSB considers Basel III to be the centerpiece set of reforms regarding resilient financial institutions. Basel III was designed to "strengthen the regulation, supervision and risk management of banks," and covers bank capital adequacy, market liquidity risk, and stress testing.

Driven in part by the FSB's SIFI Framework, the Committee on Payments and Market Infrastructures (CPMI) and the International Organization of Securities Commissions (IOSCO) revised international standards for financial market infrastructures in 2012, and also introduced a two-hour recovery time objective for critical systems.[8] Resiliency continues to be an area of major regulatory focus at both international bodies, such as the Basel Committee on Banking Supervision (which developed Basel III), and at the national level, e.g., the Bank of England/Prudential Regulation Authority and Financial Conduct Authority's recent Discussion Paper, Building the UK financial sector's operational resilience.[9]

Industry-wide resiliency requirements that apply to critical applications deployed by FIs include:

- Regulatory requirements regarding an application's recovery time objective (RTO) and recovery point objective (RPO) [See Figure 1]
- Banking requirements regarding business continuity planning (BCP)
- Tests and exercises conducted within institutions, within the industry, and through public-private sector coordination.

Managing Risk

Resiliency in Financial Services is intended to manage risk. While regulatory requirements focus on the FIs with the largest potential impact on the global economy, all FIs regardless of size must manage the risks that come with storing and processing financial data.

AWS conducted research among SIFIs and regulators within the global financial system, to identify specific resiliency metrics that FIs must report to auditors and regulators, described below. According to our research, FIs define and manage resiliency risks based on business considerations, including:

- Financial impact: Calculated as a loss of revenue for every minute an application is down

- Regulatory response: The potential for an adverse enforcement or fine imposed by a regulator
- Business opportunity: The impact of losing customers due to the adverse operating impact of application downtime
- Reputational: A long-term loss to business due to adverse press coverage
- Users impacted: The breadth of the impact to customers due to an outage
- Data loss: The risk of losing highly confidential or critical customer data during a disaster

For Financial Services applications, risk regarding resiliency is primarily defined by the RTO and RPO of a given business process, demonstrated in Figure 1:

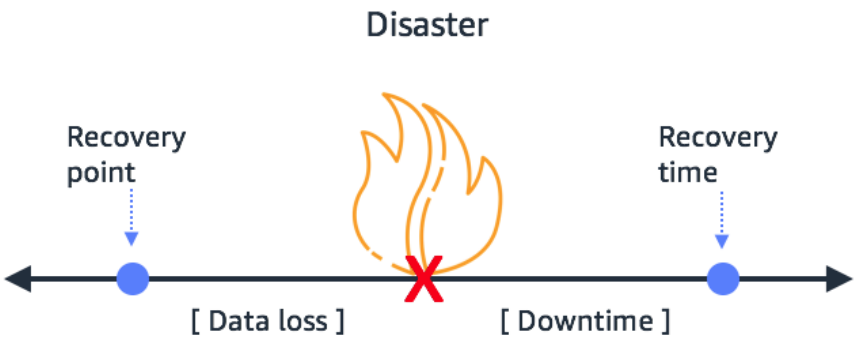

Figure 1

Along with RTO and RPO, availability was highlighted as an additional metric FIs must track and report. FIs need very high levels of availability during the business hours of their systems. For example, in a system processing cash dispensation such as an ATM, the business hours are 24 hours per day, every day, whereas for a trading platform processing trades during U.S. trading hours, the necessary availability would be 12 hours per day, every Monday through Friday. Minimum results required by FIs for RTO, RPO, and availability are presented in Table 1 below based on the tier that applies to a given application:

Table 1

KPI	Platinum or Tier 1	Gold or Tier 2	Silver or Tier 3	Bronze or Tier 4
RTO	2 hours	< 8 hours	24 hours	48+ hours
RPO	< 30 seconds	< 4 hours	24 hours	72 hours

KPI	Platinum or Tier 1	Gold or Tier 2	Silver or Tier 3	Bronze or Tier 4
Availability	99.99%+	99.9%	98%	95%

Financial Services applications are grouped in tiers based on their significance to the business, and the potential impact if the application experiences a disruption. Resiliency and DR planning takes these tiers into account, and prioritizes restoring service to the applications that can have the largest impacts regarding cost and business risks. Figure 2 below demonstrates the spectrum of recovery options, sample use cases for each tier, cost implications, and suggested AWS implementations for these tiers. An example classification of Financial Services applications is provided in Appendix A.

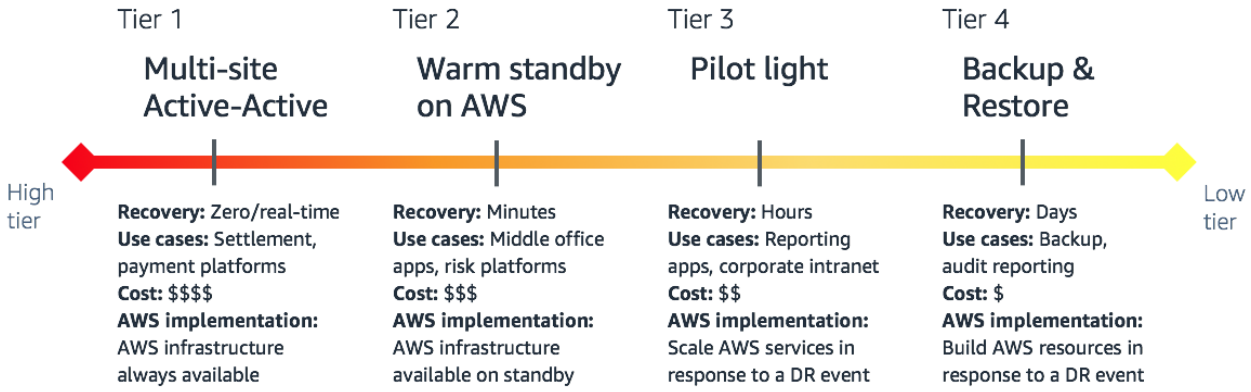

Figure 2

Given the additional cost and complexity required to maintain or restore a mission-critical Tier 1 application during a disaster event, it is important to accurately assess the tier classifications of applications based on substantiated business requirements. FIs may determine that different periods of downtime are acceptable for applications in lower tiers (e.g. Tier 3, Tier 4). Sample application classifications for the four tiers discussed above are outlined in Appendix A.

Principles of Resiliency

Financial institutions that design, test, and deploy their systems on AWS can apply the following principles to ensure high availability of their mission-critical applications.

The AWS Well-Architected Framework

To build a highly resilient application, all aspects of the application must be considered, from design to testing and deployment, from security to operations. The AWS Well-Architected framework[10] has been developed to help cloud architects build the most secure, high-performing, resilient, and efficient infrastructure possible for their applications. This white paper includes components of the AWS Well-Architected Framework that apply to resiliency in Financial Services.

Based on five pillars—operational excellence, security, reliability, performance efficiency, and cost optimization—the Framework provides a consistent approach for customers and partners to evaluate architectures, and implement designs that will scale over time. In addition, the Framework provides guidance to help implement designs that will be secure, highly available, and resilient.

Shared Responsibility

Operating in the AWS Cloud is a shared responsibility. AWS manages security and availability of the cloud, and customers are responsible for application security and availability in the cloud. This means that AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud. AWS infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services.[11] The implementation, configuration, and operation of applications on AWS is the customer's responsibility. FIs that run applications on AWS retain control of the security, resiliency, and availability by choosing the appropriate architecture, policies, and configuration in the AWS Cloud.

Taxonomy of Application Availability

As explained in the reliability pillar of the AWS Well-Architected Framework, "Service availability is commonly defined as the percentage of time that an application is operating normally."[12] When designing an application, many organizations assume that it must be "five nines" (99.999%) available, without appreciating the true cost to build and operate applications at that level of availability.

Doing so requires that all networking, security, and infrastructure—from end customers to the service application, data stores, and all other components—be built and operated to achieve 99.999%. Adding an additional nine to the availability of an application raises the cost of the application. Table 2 below contains the maximum downtime for applications, depending on their availability:

Table 2

Availability	Max disruption (per year)	Max disruption (per month)	Max disruption (per day)
99%	3 days 15 hours	7.31 hours	14.4 minutes
99.9%	8 hours 45 minutes	43.83 minutes	1.44 minutes
99.95%	4 hours 22 minutes	21.92 minutes	43.2 seconds
99.99%	52 minutes	4.38 minutes	8.64 seconds
99.999%	5 minutes	26.3 seconds	864 milliseconds

It is important to note that in complex modern applications using service-oriented architectures, many components of the application may continue to operate normally even when underlying services are not available.

Understanding Application Failure

FIs must consider the issues that cause applications to fail, and understand the probability of such an occurrence. Table 3 below lists possible sources of application failure:

Table 3

Category	Description	Probability
Operator error	Error caused by manual operator	HIGH
Deployment induced	Failure caused directly as a result of a software, hardware, network, or configuration deployment. This includes both automated and manual changes.	HIGH
Load induced	Load related failures can be triggered by a change in behavior, either of a specific caller or in the aggregate, or by the service reaching a tipping point. Load failures can occur in the network.	HIGH
Data induced	An input or entry is accepted by the system that it can't process ("poison pill")	MED

Category	Description	Probability
Credential expiration	Failure caused by the expiration of a certificate or credential.	MED
Hardware failure	Failure of any hardware component in the system, including in hosts, storage, network, or elsewhere.	LOW
Infrastructure	Power supply or environmental condition failure has an impact on hardware availability.	VERY LOW

To design the most reliable systems, FIs should focus on solving for failures with the highest probability of occurring. Planning for failures with the highest probability, utilizing the methods outlined in this white paper and the AWS Well-Architected Framework, will alleviate an FI's risk of failure with lower probability items. For example, automation, testing, and monitoring can solve for credential expiration. Achieving 99.999% availability means planning for the potential interruptions listed in Table 3, and automating human intervention out of processes wherever possible.

Automated Operations

FIs can reduce manual operator errors by automating processes. Performing infrastructure operations as code applies the same engineering discipline that you use for application code to your entire environment. Operations procedures should be captured in runbooks, scripted, tested, and their execution automated to occur in response to observed events when appropriate. Automating operational processes includes activities such as:

- Deploying code
- Maintaining canaries that constantly monitor and test applications
- Performing regular automated fail-over testing to ensure that each part of an application performs properly under these conditions
- Conducting unit-level monitoring and workflow/transaction monitoring of both success and failure scenarios
- Alarm and log analysis

- Automatic system recovery capabilities that include both upstream and downstream dependent service, network connection, and infrastructure between FIs and their customers.

Consistent Development and Deployment

Configuration drift is a high probability cause of failure. One technique to reduce deployment-induced failures is implementing Continuous Integration and Continuous Deployment (CI/CD) pipelines across lifecycle stages. CI/CD allows application development teams to manage and ensure consistent treatment, and automated execution of code and configuration deployments. It's important to ensure that the same code that is tested is what is deployed to production and the recovery environments, with an audit trail to validate.

Predictive Monitoring with Proactive Responses

Performing and documenting an impact analysis of a given application in performance degradation and failure scenarios allows FIs to create predictive monitoring. Teams that implement predictive monitoring, with corresponding proactive responses, can mitigate additional sources of application failure. This situational awareness regarding specific applications should be established in regards to the impact of events, operations activities, and the workload criticality.

Workloads should be designed to emit the necessary telemetry to understand workload health (at the component, service, and business impact level), user experience within the application, and the achievement of business outcomes. This instrumentation will enable the prediction and detection of events, and intervention prior to an undesired impact.

We recommend that FIs work to minimize a workload's Mean Time to Detection (MTTD) in order to provide time for the recovery mechanism to respond, and to increase the availability of your applications.

AWS Cloud

AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world. Financial Services customers can use AWS services to construct highly elastic, highly available, resilient, and scalable solutions with lower costs compared to traditional on-premises IT. Our services are designed to tolerate system or hardware failures with

minimal customer impact. This section outlines the core components of a resilient cloud architecture that AWS provides.

AWS Infrastructure

The AWS Global infrastructure is built around Regions and Availability Zones (AZs). AWS Regions provide multiple, physically separated and isolated Availability Zones which are connected with low latency, high throughput, and highly redundant networking. These Availability Zones offer AWS customers an easier and effective way to design and operate applications and databases, making them more highly available, fault tolerant, and scalable than traditional single data center infrastructures or multi-data center infrastructures. At the time of publication, the AWS Cloud spans 61 Availability Zones (AZs) within 20 geographic Regions.[13]

Compared to the on-premises environments utilized by global financial institutions, the AZ and regional diversity of the AWS infrastructure greatly reduces geographic concentration risk. AWS continues to add new Regions and AZs across the globe, to provide resiliency to financial institutions worldwide.

AWS Regions

Each AWS Region is a separate geographic area where cloud resources can be instantiated. Regions are isolated from each other, meaning that a disruption in one Region does not result in contagion in other Regions. This achieves the greatest possible fault tolerance and stability. Regions are designed to be autonomous, with dedicated copies of services deployed in each Region. AWS Regions are composed of two or more Availability Zones (see Figure 3) that are designed to be independent.

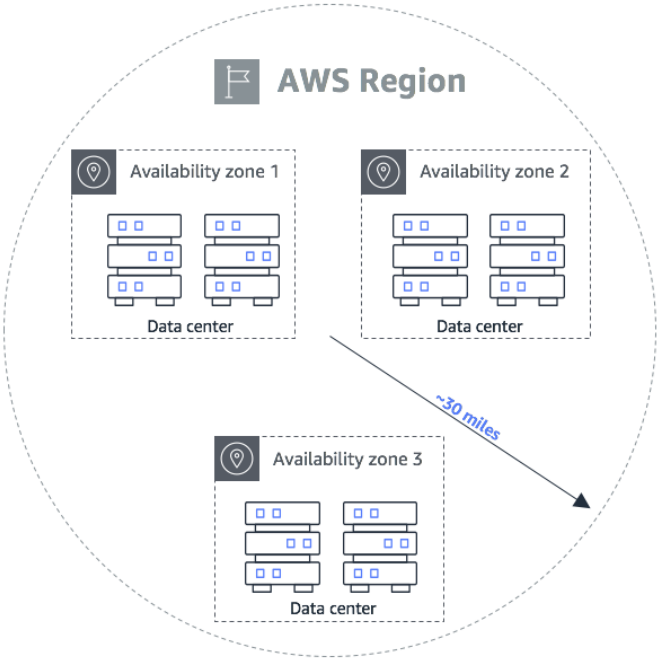

Figure 3

Availability Zones

Each AWS Region has multiple Availability Zones (AZs). An Availability Zone is a collection of one or more data centers which forms a campus. AZs are physically separated and independent, and are built with highly redundant networking. Each AZ is

a distinct location within a Region that is insulated from failures in other AZs, to avoid correlated failure scenarios due to environmental hazards like fires, floods, and tornadoes. Each AZ has independent physical infrastructure: dedicated connections to utility power, standalone backup power sources, independent mechanical services, and independent network connectivity within and beyond the AZ.

Locating AZs within the same Region allows for data replication that provides redundancy without a meaningful impact on latency—an important benefit for FIs that need low latency to run applications. At the same time, AZs are independent in order to ensure services remain available in the event of major incidents. Many AWS services run autonomously within AZs; this means that if one AZ within a single Region loses power or connectivity, the other AZs in the Region are unaffected, or in the case of a software error, the risk of that error propagating is limited.

AWS Services Design

AWS employs multiple application design constructs with different levels of independent, redundant components, to make services highly resilient and reliable. Detailed best practices for designing applications for high availability in the cloud can be found in the reliability pillar of the AWS Well-Architected Framework.[14] The best practices that AWS applies to availability include:

- Fault isolation zones
  - Cell-based architecture
  - Multi AZ architecture
- Micro-services architecture
- Redundant components
- Recovery-Oriented Computing (ROC)
- Distributed systems best practices, including:
  - Throttling
  - Retry with exponential back off
  - Fail fast (load shedding)
  - Use of idempotency tokens
  - Constant work

- Circuit breaker
- Static stability

According to the reliability pillar of the AWS Well-Architected Framework, “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 example, Amazon EC2 instances, Amazon RDS databases, and Amazon DynamoDB table read/write operations are all data plane operations. In contrast, launching new EC2 instances or RDS databases, or adding or changing table meta-data in DynamoDB are all considered control plane operations. While high levels of availability are important for all of these capabilities, the data planes typically have higher availability design goals than the control planes.”

The application design components from the AWS Well-Architected Framework that are most relevant to FIs in their resiliency and DR planning are described below.

Cell-Based Architecture

To make AWS services highly resilient at the lowest levels, to strengthen data plane availability, AWS partitions resources and requests via a specific dimension such as a resource ID. These partitions (which we refer to as “cells” but others call “shards” or “stripes”) are designed to be independent and further contain faults to within a single cell. Cells are multiple instantiations of a service that are fully isolated from each other; these internal service structures are invisible to customers. In a cell-based architecture, resources and requests are partitioned into cells, which are capped in size. This design minimizes the chance that a disruption in one cell—for example, one subset of customers—would disrupt other cells. By reducing the blast radius of a given failure within a service based on cells, overall availability increases and continuity of service remains. A rough analogy is a set of watertight bulkheads on a ship: enough bulkheads, appropriately designed, can contain water in case the ship’s hull is breached and, will allow the ship to remain afloat.

Multi-AZ Architecture

To avoid single points of failure, AWS deploys service API endpoints across multiple AZs within a Region. The control plane of AWS service offerings is designed to be Multi-AZ, that is, servers hosting the control plane are spread across three or more AZs within a region.

Micro-Service Architecture

At AWS, we have built our systems using a concept called micro-services. While micro-services have several attractive qualities, the most important benefit for availability is that micro-services are smaller and simpler. They allow FIs to differentiate the availability required of different services, and thereby focus investments specifically to the micro-services that have the greatest availability needs. Further deployments and updates only impact a small portion of the overall service. This reduces the blast radius of the updates and increases availability of the services.

AWS Services Scope

Having a full understanding of the scope and availability of AWS services will allow FIs to best architect for reliability and availability. It is critical to understand services that offer cross-regional replications, versus services that offer multi-AZ availability, in order to build applications that meet the resiliency requirements of financial institutions.

AWS services scoped to a single AZ

Some services are designed to provide resources that are instantiated in a specific AZ. These are services which provide resources tied to underlying physical entities such as compute and storage. For example: An Amazon Elastic Compute Cloud (EC2) instance, and its associated Elastic Block Storage (EBS), live within a single AZ. Service scope for all AWS services is provided in Appendix C.

Regional AWS services

AWS designs services to be inherently resilient and highly available on their data and control planes. Regional services have their resources spread across multiple AZs within the Region. As a user you deploy these services without having affinity to any specific AZ. For example, Amazon Simple Storage Service (Amazon S3) is designed to provide a regional API endpoint that is highly available and spread across multiple AZs in a region. Amazon S3 replicates data redundantly, durably, and reliably across multiple AZs for every object you store in Amazon S3.

Global and edge-based services

AWS Edge locations are points of presence across the globe where AWS provides specific services such as content distribution and DNS resolution. AWS has five services at the time of publication, including Amazon Route 53 and Amazon CloudFront, that are hosted at AWS Edge Network Locations.[15] These points of presence are

outside of the AWS Regions and AZs and provide services with high levels of availability SLA.[16] For a list of published AWS SLAs please see Appendix E.

Services with cross-regional capability

One of the design principles of AWS is to provide regional isolation. AWS does not automatically replicate customer data across multiple regions. Many AWS services offer features to the customer to replicate their data across multiple regions. Customers can use these features to build highly available and reliable multi-region architectures. Examples include:

- Copy Amazon EC2 Amazon Machine Images (AMIs) between regions.[17]
- Copy Amazon Elastic Block Store (Amazon EBS) snapshots between regions.[18]
- Perform Cross-Region Replication (CRR) with Amazon S3.[19]
- Use engines in Amazon Relational Database Service (Amazon RDS) to support read replicas in a different region.[20]
- Use Amazon Aurora MySQL to support read replicas in different regions.[21]
- Copy Amazon RDS database (DB) snapshots or DB cluster snapshots between regions.[22]
- Use the Global Tables feature in Amazon DynamoDB to create a table with endpoints in two or more regions.[23]
- Copy Amazon Neptune DB Cluster snapshots between regions.[24]
- Copy Amazon Redshift snapshots between regions.[25]
- Use Amazon Virtual Private Cloud (Amazon VPC) for inter-region VPC peering (with limitations in comparison to intra-region VPC peering).
- Access AWS services and your VPCs in remote regions using an AWS Direct Connect circuit.
- Manage resources in multiple regions with AWS Data Pipeline.
- Copy AWS CloudHSM backups across regions.[26]

Note: Appendix A of the reliability pillar of the AWS Well-Architected Framework lists the designed-for availability of select AWS services. This information is also included in Appendix B of this document (updated as of the time of publication). Appendix C of this document shows how scope is defined for select AWS services. Not all services are available in all regions. Please refer to the AWS region table[27] to ensure services used in your architecture are available in the desired regions.

Design Patterns for Critical Applications

In this section we discuss best practices and share reference blueprints for specific resilient application design on AWS. A supplemental checklist for resiliency and DR planning is included in Appendix D.

Design Practices

Network Access

To enable recovery across the globe it is necessary to ensure that the network supporting your applications is appropriately redundant, always available, and seamlessly routed. AWS provides a global infrastructure with 20 Regions and 61 Availability Zones (at the time of publication). For application designers this is akin to having multiple data centers across the globe where an application can be hosted.

When designing a highly resilient application, the network access to the AWS global infrastructure should be transparent and resilient. AWS Direct Connect can be used to provide this connectivity, and connect customer data centers to the AWS Cloud. When connecting from their corporate data centers, financial institutions should ensure that they have redundant independent network paths to establish connectivity. To accomplish this, we can use the pattern in Figure 4 below as an example of a completely redundant and highly available network connectivity design.

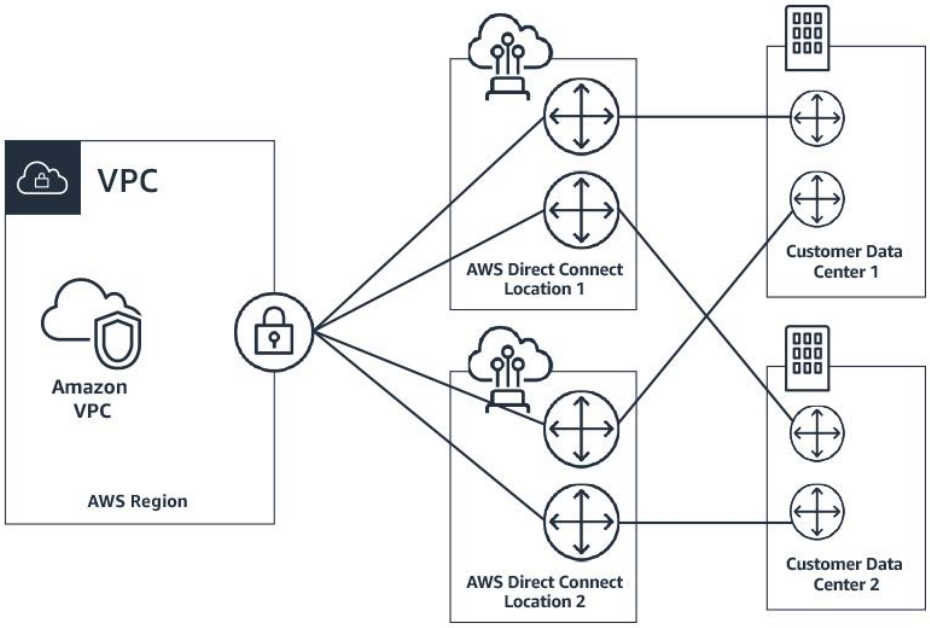

Figure 4

As shown, the customer has two data centers which are connected to an AWS Region via redundant paths with no overlapping single points of failure. The Amazon VPC Virtual Private Gateway (VPG) shown in the figure is a highly redundant component, which is implemented using multiple underlying devices. This eliminates any single points of failure on the VPG and allows for diverse paths in the event of a point of presence (POP) failure.

To enable the application to use multiple AWS regions, it is also important to establish direct connections to multiple regions. Application traffic should automatically be routed to multiple regions. FIs can accomplish this by either using multiple direct connects in different regions, or using AWS Direct Connect Gateways to seamlessly connect their data centers to all regions, using the AWS global network as a backbone as shown in Figure 5 below.

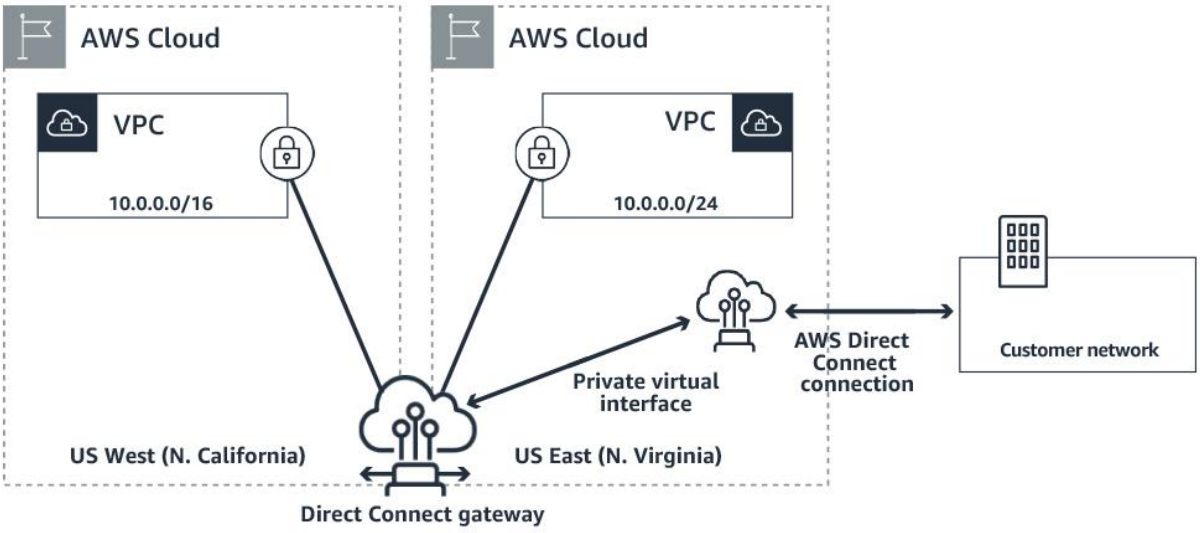

Figure 5

When hosting applications entirely on AWS, you can achieve connectivity between workload components in different regions by using the AWS-provided VPC In-Region Peering[28] and Inter-Region VPC Peering feature as shown in Figure 6. This pattern should be used when the number of VPCs being connected is less than 10.

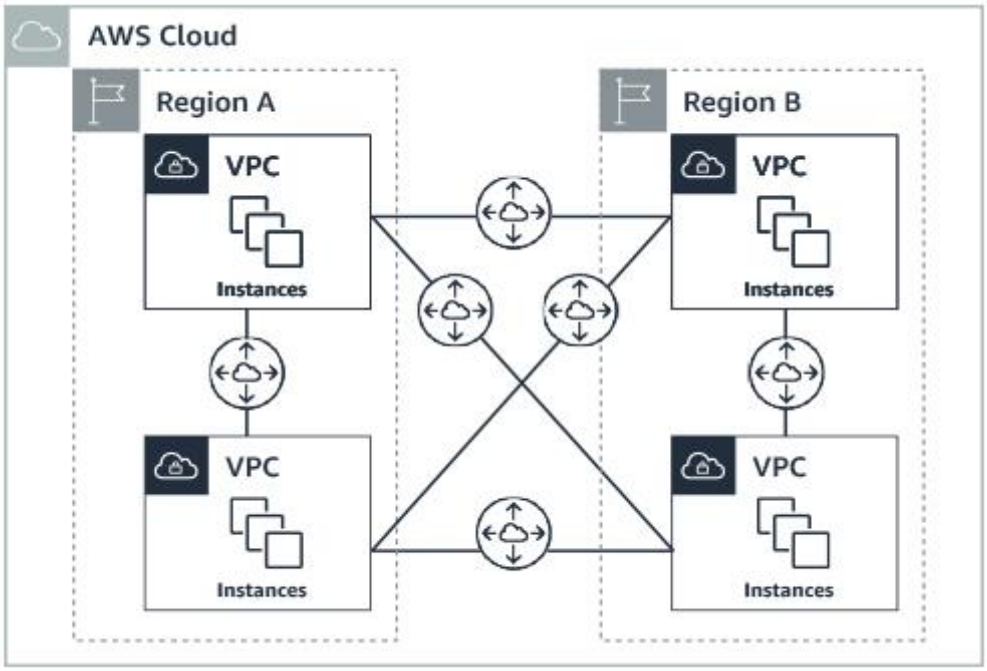

Figure 6

For a more scalable and future-proof solution where application components are dispersed across many VPCs you should use the AWS Transit Gateway to connect the VPCs (Figure 7).

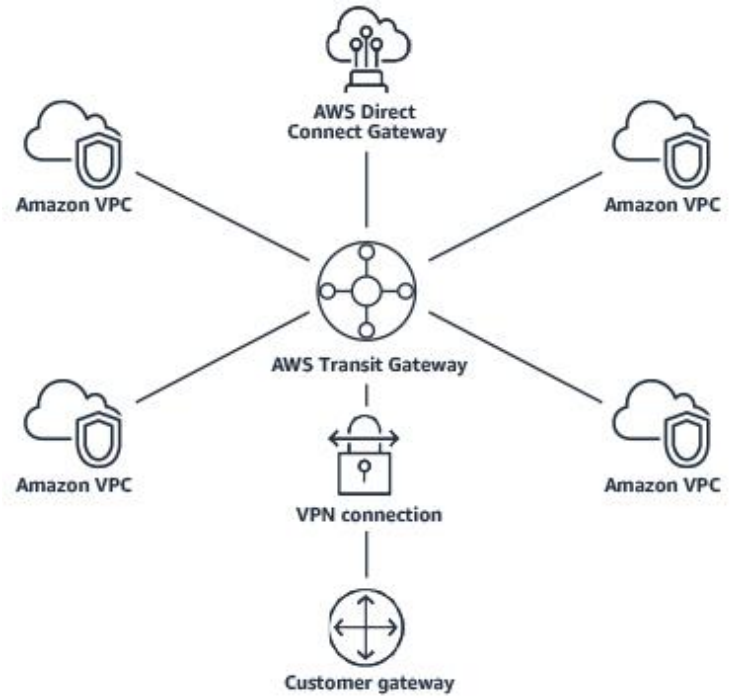

Figure 7

For hybrid scenarios where components of an application can be running in corporate data centers and on AWS, two options are available:

1. AWS Transit Gateway: The AWS Transit Gateway allows customers to connect their Amazon VPCs and their on-premises networks to a single gateway. As your number of workloads running on AWS increases, you need to be able to scale your networks across multiple accounts and Amazon VPCs to keep up with the growth. With AWS Transit Gateway, you only have to create and manage a single connection from the central gateway in to each Amazon VPC, on-premises data center, or remote office across your network. AWS Transit Gateway acts as a hub that controls how traffic is routed among all the connected networks, which act like spokes. This hub and spoke model simplifies management and reduces operational costs because each network only has to connect to the Transit Gateway and not to every other network

2. The Transit VPC Solution: The Transit VPC[29] can be used to enable connectivity between various VPC's in different regions and customer data centers (Figure 8). You can use this to connect multiple VPCs that are geographically disparate and/or running in separate AWS accounts, to a common VPC that serves as a global network transit center. This network topology simplifies network management and minimizes the number of connections that you need to set up. Further information on global connectivity options are available on the AWS Global Transit Network website.[30]

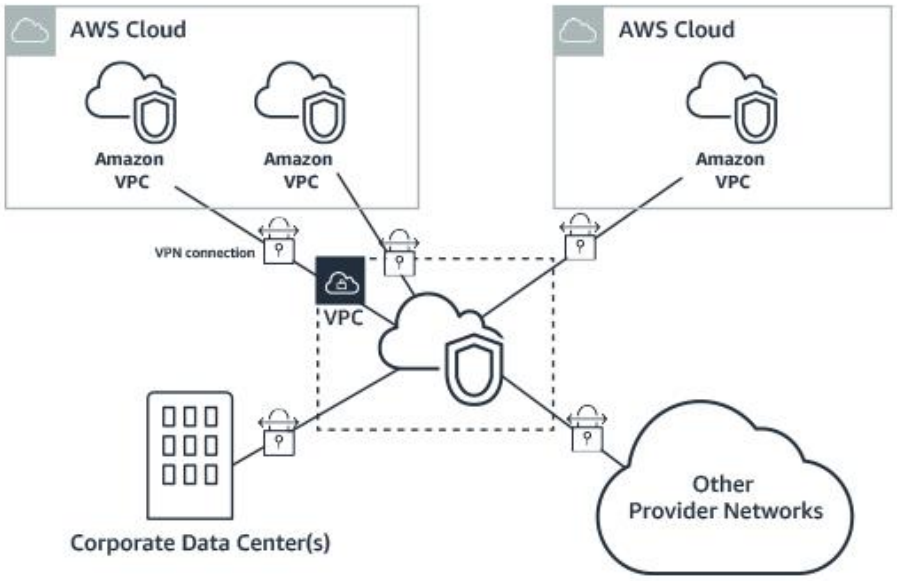

Figure 8

Of the two options presented above, the AWS Transit Gateway along with the AWS Direct Connect Gateway is the recommended future-proof pattern for designing hybrid connectivity.

Data Availability

Applications store data in different forms, including files on a filesystem, block storage, in databases, and in memory caches. Making this data available in alternate regions is key to any disaster recovery strategy. The rate of data change and the distance between regions can constrain the RPO achievable by your application.

Therefore, when designing solutions for very high availability and resiliency, we need to understand these constraints and the impact on RPO. If a financial institution requires an RPO of zero, we should choose the appropriate AWS service or technology, a single

AWS Region, or consider the distance between AWS Regions, so as to minimize the impact on performance. For example, if we choose EU-East1 and EU-West2 as our application hosting regions, having RPO of zero would mean that each transaction would have additional 10 milliseconds latency, while within a single AWS Region you can achieve this with multiple AZs. This requirement would lead to choosing a design using alternate mechanisms to achieve high performance and resiliency. We discuss specific options to achieve this in the Application Resiliency Blueprints section below.

AWS services natively store data within a single region using multiple AZs, except for Amazon EBS, Amazon S3 Reduced Redundancy Storage (RRS), and Amazon S3 ONEZONE_IA.[31] Appendix C: Service Capabilities outlines the scoping of many of our larger services. This gives the customer control of their data and allows them to meet the regulatory requirements of data residency.

To enable customers to design applications which are spread over multiple regions, AWS services provide features to enable replication to an alternate region, as discussed above. In this case, AWS does the heavy lifting of data replication. The RPO for these replications varies based on replication lag. Replication lag depends on various factors such as: the size of data being copied, the distance to the target region being used, and the rate of data change. Customers should monitor this replication lag to ensure that they can meet the RPO objective of the application. Customer should implement mechanisms in the application to throttle or back off to keep the application performance within the RPO window.

Depending on the type of data in question, the rate of change, and its origin, there are multiple options to have data copied over multiple regions. For OS images, when using Amazon EC2 and Amazon EBS, you must initiate the copy and ensure that the appropriate Amazon Machine Images (AMIs) are copied and available in the alternate region. For your application data, you must initiate and ensure the EBS Snapshots of your data volumes are configured for cross-region copy.

For static application data stored in Amazon S3 you can leverage Cross-Region Replication (CRR),[32] which allows for your data to be available elsewhere, as displayed in Figure 9 below. CRR is a core functional piece of an FI's data availability strategy.

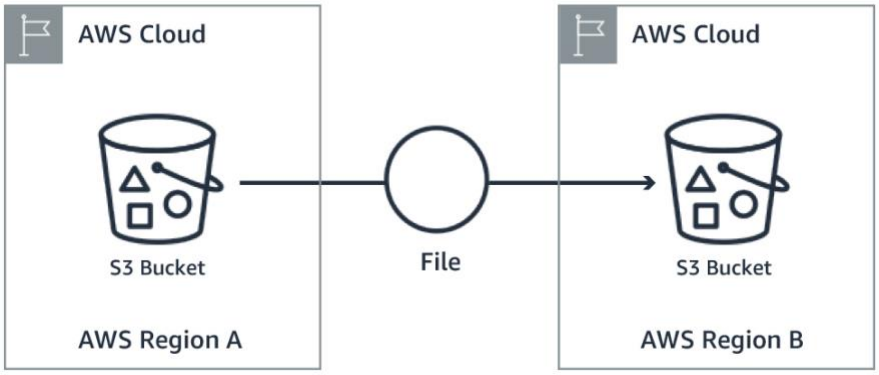

Figure 9

Similar to Amazon S3, Amazon Elastic File System (Amazon EFS) is another regionally scoped storage service. Amazon EFS allows you to mount a single volume to multiple instances, and is stored redundantly across multiple AZ's within that single region. To ensure high data availability, you can employ EFS File Sync[33] to quickly replicate files and their corresponding metadata to another region.

By contrast, Amazon EBS volumes are scoped to an individual AZ. For Amazon EBS volumes attached to your compute resources, snapshotting to another region will permit data from your local volumes to be available in another region. To ensure you have quicker recovery points, data persistence, and expected Amazon EBS capacity, you can replicate your EBS Snapshots to another region and then create (unattached) volumes from them. By creating unattached volumes you block the capacity, effectively creating a reserved allocation for storage in the alternate region.

In the event of a failure, having these volumes readily available will help reduce RTO. For workloads which absolutely require deterministic RPOs, it is advised to build the replication methodology into the application. There are AWS Partner Network (APN) Partners who provide custom-built data replication technologies using point to point systems which can provide the RPOs required. These APN Partner solutions can be found in the AWS Marketplace, including CloudEndure Disaster Recovery,[34] Attunity Replicate,[35] WANdisco Fusion,[36] NetApp Cloud Sync,[37] and Zerto Virtual Replication.[38]

For data stored in databases, Amazon RDS Read Replicas provide enhanced performance and durability for database instances. When building database workloads for Tier 1 Financial Services applications, you can create one or more replicas of a given source DB instance, as demonstrated in Figure 10 below.

Read replicas can be promoted when needed to become standalone DB instances. Read replicas are available in Amazon RDS for MySQL, MariaDB, and PostgreSQL as well as Amazon Aurora. It is important to note that there may be replication latency depending on the distance of the target region from the source region. You can monitor the replication lag using use Amazon CloudWatch when you implement any of these use cases.

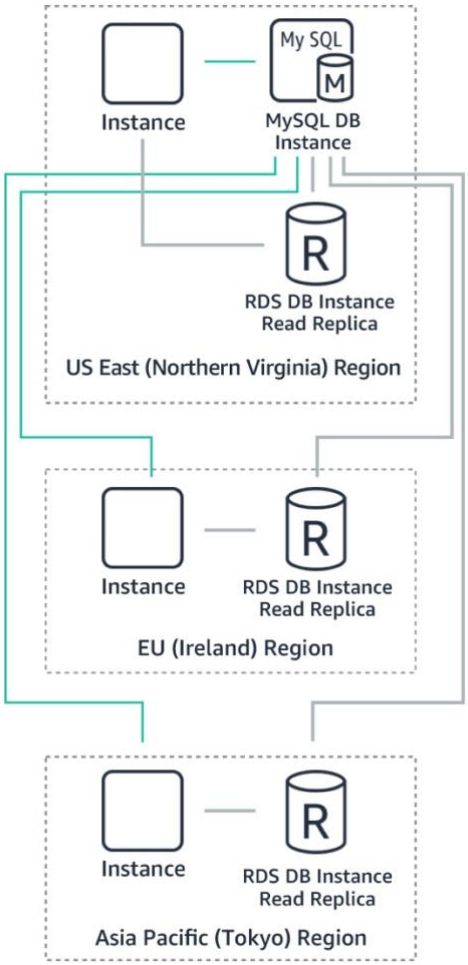

Figure 10

For data coming into AWS from outside sources such as trade data from business partners or market data from exchanges, you should create data subscribers in different regions to store data locally within the region where the subscriber is running. This ensure that data is available in the region where the application will be recovered. Another important point to consider when replicating data is to appropriately sequence the data records so you can discretely determine which records must be replayed, versus which are already committed by the application. Having a well-defined, global, record-based sequencing methodology will greatly reduce the risk of data loss across the systems and provide for faster transaction reconciliation and recovery. Data reconciliation tools are often custom built to applications, and so FIs should invest in building and maintaining these tools if they are not already place.

Build Self-healing and Stateless Applications

When building highly resilient applications, you want to design in the ability to self-heal. Applications should be able to monitor and recover from a common failure. Primarily decoupling application interdependencies and removing states from application components is used to achieve the self-healing behavior. AWS provides services such as queuing, load balancing and autoscaling which should be used in your application architecture. Using Elastic Load Balancing (ELB) you can decouple the client from the servers. With this design, if a server fails, the client connection can be routed to another working server (Figure 11).

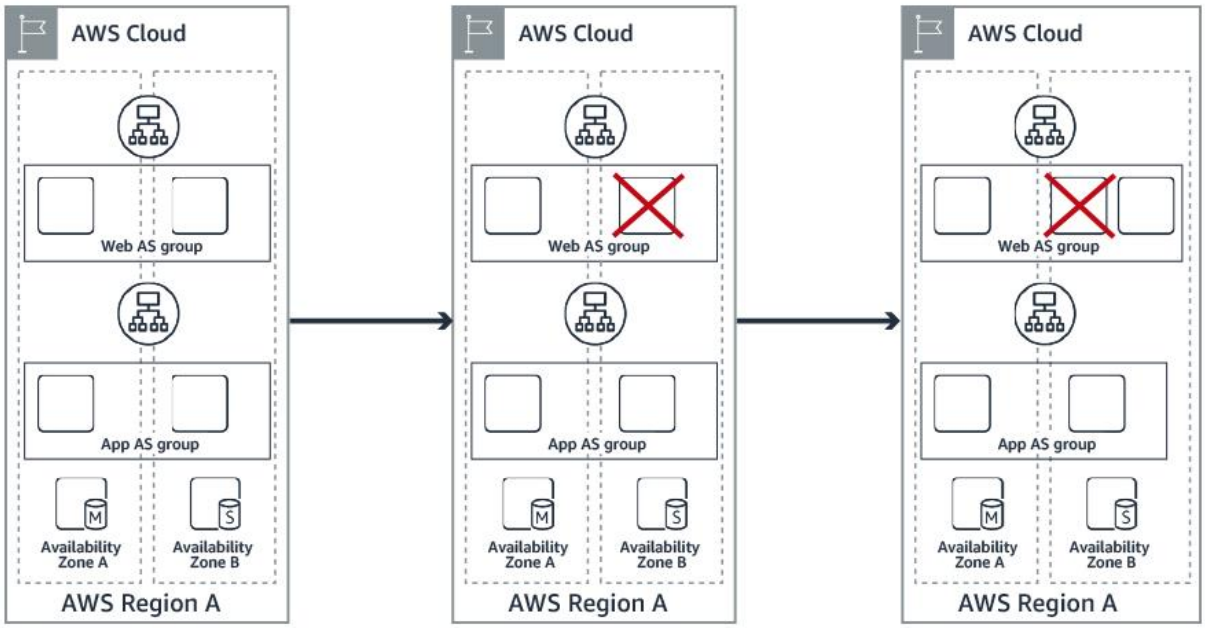

Figure 11

By using the load balancer, the application has self-healed from a failure and that failure did not propagate to the customer. In the background, we can use AWS Auto Scaling to relaunch a server and reintegrate the server into the cluster. With this pattern we have self-healed the capacity that is needed to serve end-users. Statelessness of compute instances is a core part of this design practice.

Application Resiliency Blueprints

Most applications can achieve high levels of resiliency with a standard multi-AZ pattern.[39] For applications requiring even higher resiliency than offered by a single-region pattern, this section describes patterns, or blueprints, to build Tier 1 applications requiring five nines (99.999%) of uptime. A key necessity for applications requiring five nines is to have the capacity available to continue processing in an alternate location when a disaster happens. In the blueprints that follow, it is important to recognize that there is an implicit understanding; when a resource is being used by a customer, it is not taken away from that customer. Once an Amazon EC2 instance or Amazon EBS volume is in use by an AWS customer, it is not re-allocated elsewhere. Therefore, the pre-allocation of the stack provides capacity at the time of recovery.

Active – Standby (Static Resiliency)

This is the traditional pattern used to host Tier 1 applications (Figure 12). In this pattern a complete stack is allocated in the alternate region, but not used until the disaster event. The backup stack is a complete replica of the primary application stack and is kept in sync with the primary by continuous data sync from primary to secondary. This pattern can also be implemented for applications with low RTO and RPO requirements. This pattern is useful for fast recovery of applications which cannot take advantage of native AWS features, and requires the least amount of changes to the application development or deployment. This is also called static resiliency.

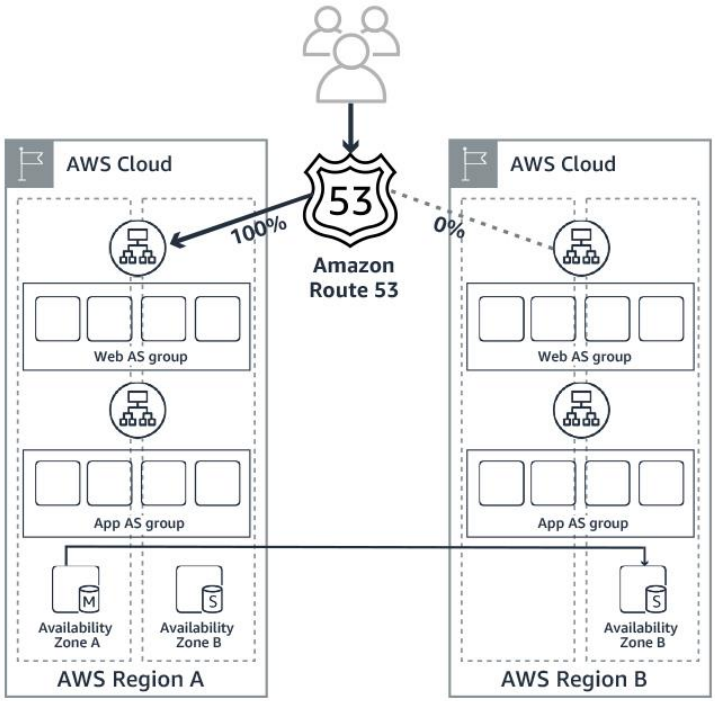

Figure 12

Having a complete parallel stack on standby at an alternate region provides for the fast recovery needed by the application. To achieve recovery a global traffic manager such as Amazon Route 53 or AWS Global Accelerator can be used, which are capable of monitoring application availability and routing the traffic based on availability. Amazon Route 53 can monitor the health of the application endpoints and direct traffic to a primary region. When a failure occurs in the primary region Amazon Route 53 can automatically switch traffic to the alternate region.

This pattern provides a finite low RPO and RTO. The RPO is based on the replication lag between the primary and standby region.

Active – Active (Distributed Resiliency)

For applications requiring zero RTO, you can use the Active-Active pattern (Figure 13 below). This resiliency blueprint is an augmentation of the Active-Standby. Here both of the application stacks are used simultaneously to service production activity. As the secondary stack is also active there is no failover time. In normal operation, AWS Global Accelerator or Amazon Route 53 will split traffic 50-50 between both regions.

When Amazon Route 53 detects a failure of the application in a region, it automatically directs 100% of the traffic to the surviving site.

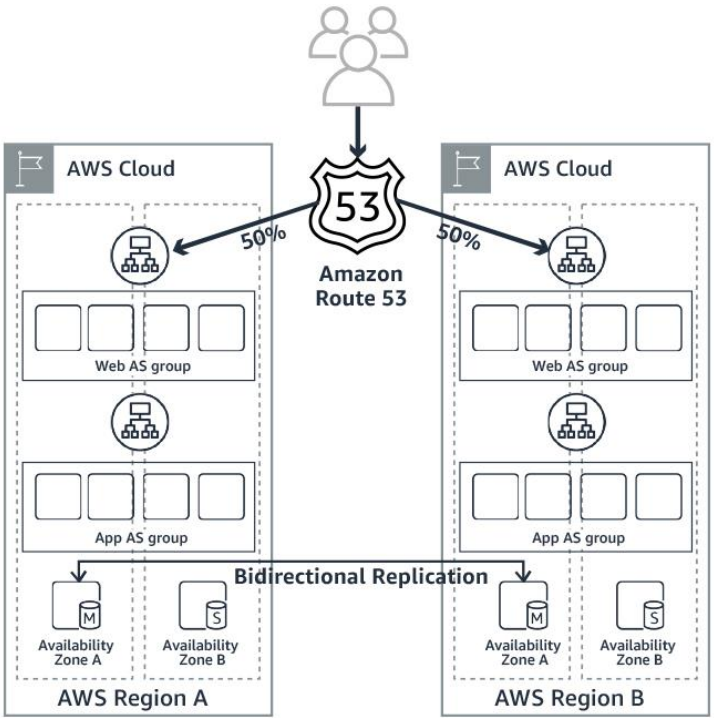

Figure 13

In such scenarios the data replication is critical and needs to be handled at the application level. Since both stacks are being used actively, data replication will need to happen in both directions. The RPO is based on the replication lag between the two sites. This pattern is effective for applications where separate homes can be established for different customer bases. Services such as Amazon DynamoDB Global Tables Multi-master, or Multi-region MySQL deployment in AWS,[40] can be used to achieve the replication required by the application. You can also choose storage layer replication technology from APN Partners such as CloudEndure Disaster Recovery, Attunity Replicate, WANdisco Fusion, NetApp Cloud Sync, to achieve disk-based replication.

Dual Write (Parallel Resiliency)

For applications requiring zero RPO, you can use the Dual Write pattern (Figure 14 below). In this pattern, there is a shared nothing architecture where two independent stacks are setup in different regions, and process each transaction or action in parallel. In this pattern, 100 percent of the traffic flows to both of the regions, and the application processes the data simultaneously. For downstream applications the output of only one region is used at any given time. The application will need to have various safeguards to avoid transaction duplication. You will need to implement checkpointing and run reconciliation jobs to ensure that both sites are in sync and producing the same results.

The RPO of this pattern is zero as data is simultaneously written to both regions. The RTO will depend on the time it takes to switch over the output for downstream applications.

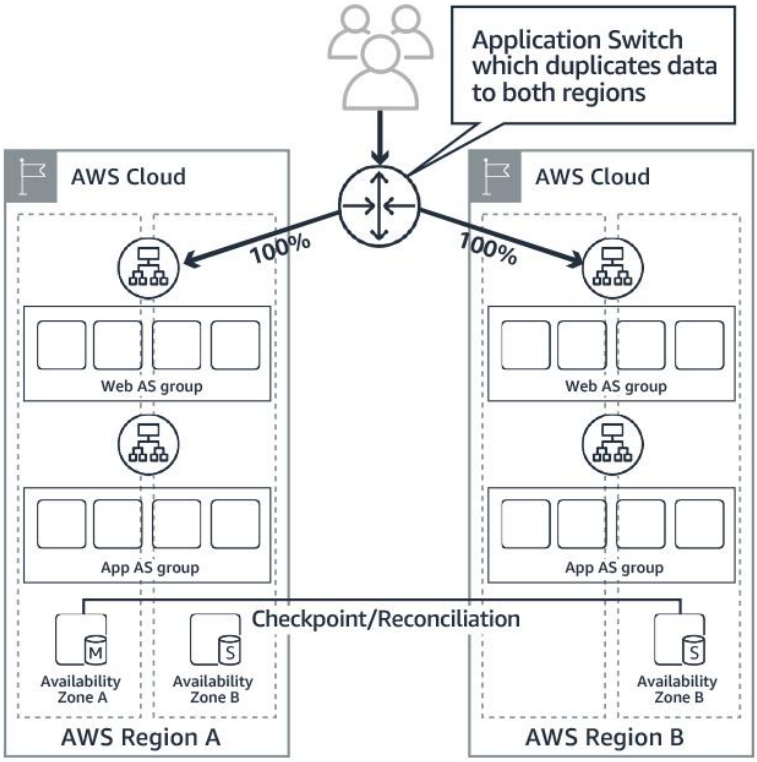

Figure 14

Satellite Region

In addition, for applications requiring zero RPO, and low RTO, a design pattern consisting of three regions can also be used. Of the three regions, one region is in close proximity to the primary region forming a satellite region (Figure 15 below). The application infrastructure is setup in the Active-Standby pattern, with additional infrastructure in the satellite region to provide syncronous replication. All data written to the primary region is committed synchronously to the satellite Region. The satellite region is used only to bunker the data and provide a sync point. From the satellite region, data is forward synched to the secondary region (far region) asynchronously.

From an operational perspective, during normal operations, only the primary region is active. All applications are in stopped mode at the secondary region. At the time of disaster, you will need to monitor the replication status from the satellite region to the secondary region. Once the replication status syncs up, you can start all applications in the secondary region and continue operations from the secondary region. Since the

infrastructure is already in place in the secondary region, the RTO is limited to the time it takes for the data to synchronize from the satellite region and start up the applications.

Figure 15

Operational Resilience

Even the best design requires operational excellence to ensure high availability, mitigate downtime, and to compensate for limitations of design. The operations focus of resiliency in the cloud is on successful implementation of change, execution of processes, insight to operational health, insight to achievement of business outcomes, and timely and effective responses to events impacting the application.

With mission-critical financial systems, increased fidelity of insight and timing are required to move from responding after the fact, to predicting the event and intervening to mitigate or avert the event prior to it causing an undesired impact. This means applications must be designed with a high degree of awareness of user activity, business activity, adverse conditions, and threats, to allow operations and business teams to take them into account.

Design Principles

There are six design principles[41] for operational excellence in the AWS Cloud:

- **Perform operations as code**: In the cloud, you apply the same engineering discipline that you use for application code to your entire environment. You define your entire workload (applications, infrastructure, etc.) as code and update it with code. You script your operations procedures and automate their execution by triggering them in response to events. By performing operations as code, you limit human error and enable consistent responses to events.
- **Annotated documentation**: In an on-premises environment, documentation is created by hand, used by humans, and hard to keep in sync with the pace of change. In the cloud, you can automate the creation of annotated documentation after every build (or automatically annotate hand-crafted documentation). Annotated documentation can be used by humans and systems, and can be used as an input to your operations code.
- **Make frequent, small, reversible changes**: Design workloads to allow components to be updated regularly to increase the flow of beneficial changes into your workload. Make changes in small increments that can be reversed if they fail to aid in the identification and resolution of issues introduced to your environment (without affecting customers when possible). Use canary deployments to detect errors early and lower the impact of failures.
- **Refine operations procedures frequently**: As you use operations procedures, look for opportunities to improve them. As you evolve your workload, evolve your procedures appropriately. Set up regular game days to review and validate that all procedures are effective and that teams are familiar with them.
- **Anticipate failure**: Perform “pre-mortem” exercises to identify potential sources of failure so that they can be removed or mitigated. Test your failure scenarios and validate your understanding of their impact. Test your response procedures to ensure they are effective and that teams are familiar with their execution. Set up regular game days to test workload and team responses to simulated events.
- **Learn from operational failures**: Drive improvement through lessons learned from all operational events and failures. Share what is learned across teams and through the entire organization

Monitoring

High availability for the applications of FIs requires the ability to detect failures and quickly recover from them. Applications should be configured to emit the relevant telemetry to detect failures, and operations processes should be in place to capture and react to the events. Amazon CloudWatch and CloudWatch dashboards provide useful tools to capture, react to, and display application health.

FIs can use the AWS Personal Health Dashboard, which provides alerts and remediation guidance when AWS is experiencing events that may impact your workloads. The dashboard displays relevant and timely information to help manage events in progress, and provides proactive notification to help plan for scheduled activities. With Personal Health Dashboard, alerts are triggered by changes in the health of AWS resources being used in your applications, giving you event visibility, and guidance to help quickly diagnose and resolve issues. Enterprise support and Business support customers have access to the AWS Health API, and use this API to integrate existing in-house or third-party IT management tools with the information in the Personal Health Dashboard.

Drift between primary and secondary sites can lead to failure in recovery during a disaster event. FIs can monitor changes to application infrastructure by using AWS CloudTrail and AWS Config. These services provide the capability to monitor activity within your AWS account, including actions taken through the AWS Management Console, AWS SDKs, command line tools, and other AWS services. Once detected, you can automate the reactive action by using Amazon CloudWatch Events integration. Amazon CloudWatch events can trigger the execution of pre-defined workflows when events are detected.

For application-level insights you can use AWS X-Ray to monitor your application. AWS X-Ray enables analysis of the behavior of distributed applications by providing request tracing, exception collection, and profiling capabilities to provide insight to your workloads. Additional recommendations regarding monitoring and alarms can be found in the reliability pillar of the AWS Well-Architected Framework.

Automation

The benefit of cloud and infrastructure as code is the ability to build and tear down entire environments programmatically and automatically. If architected with resiliency in mind, a recovery environment can be stood up in minutes via AWS CloudFormation templates or AWS Systems Manager automation (Figure 16 below). Automation is

critical to maintaining high availability and fast recovery. AWS offers a wide breadth of automation tools to accomplish this.

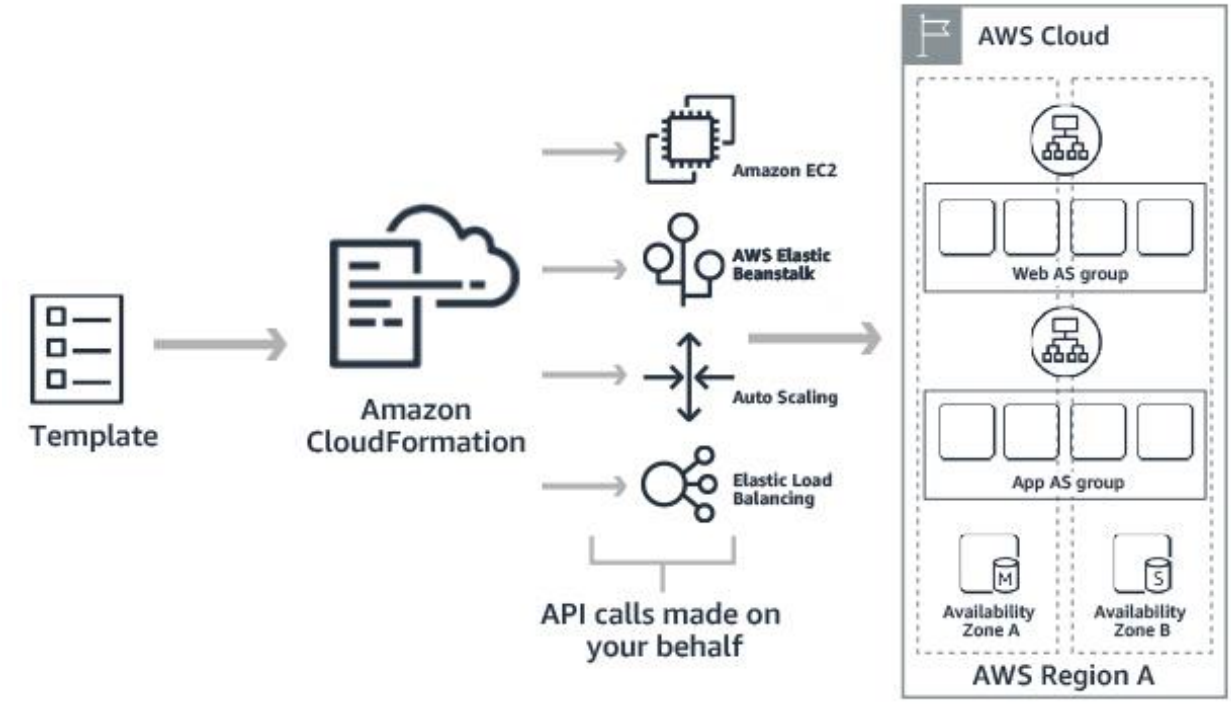

Figure 16

AWS Systems Manager can help automate complete runbooks that are used in recovery of an application during a disaster. You can sequence a complete set of operations to automatically execute on the detection of an event. With Systems Manager automation documents you can manage these runbooks similar to code. You can version them and update them along with every release of code. This helps keep your recovery plan in sync with released code and updates to infrastructure.

Implementation of code-based management practices across your infrastructure, applications, and operational procedures enables the high degree of version control, testing, validation, and mitigation of human error that are necessary to limit the introduction of errors into your environment, and reduce the RTO of the recovery.

Application Deployment

Improper code and failures in deployment are one of the most common causes of failure in an application. For Financial Services applications, it is essential to have a well-defined software development lifecycle, combined with automated deployment pipelines. This reduces the risk of failures due to software deployments. The pipelines

should be carefully designed and include tests and validation to ensure that proper code is deployed into production and recovery sites. On AWS, using infrastructure as code, you can integrate both infrastructure changes and application changes to be tested and validated before they reach production (Figure 17).

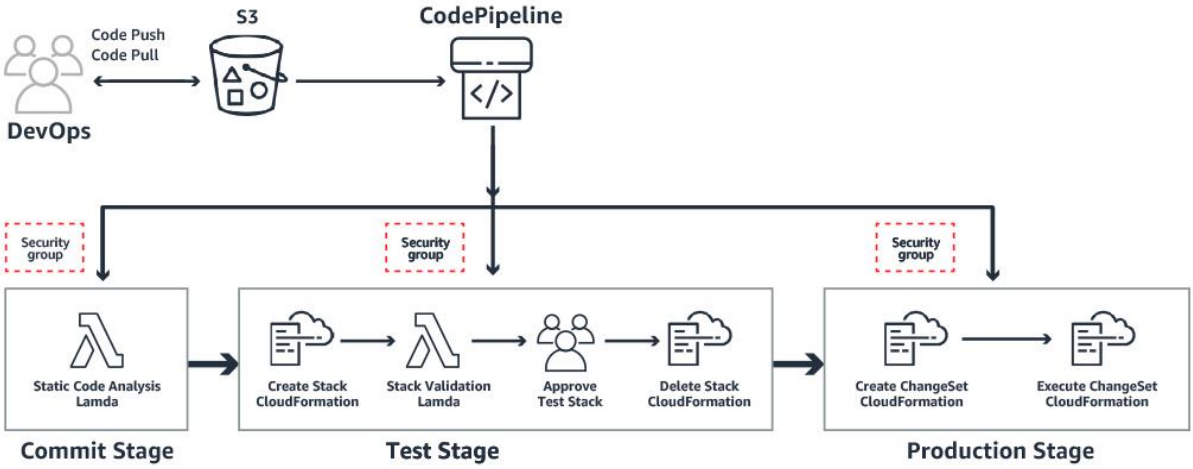

Figure 17

AWS Developer Tools including AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline are designed to provide an integrated set of tools which allow you to build pipelines integrating infrastructure and code deployments. These services include best practice operations procedures such as staged deployment, canary deployments, isolation zone deployments, and automatic roll back which allow for the development and deployment of complex patterns for Tier 1 and other applications.

Cost Optimization Practices

The economic model for resiliency requires a commitment from FIs to understand that there are necessary costs to minimize the risk of downtime in the event of a business continuity or disaster recovery event.

Designing applications for high levels of availability typically comes with increased costs, so it's appropriate to identify the true availability needs before embarking on application design. As discussed above, adding an additional "nine" to an application's availability adds costs to an application's total cost of ownership (TCO).

The global footprint of AWS, efficiencies from automation, and economies of scale allow us to pass cost savings to FIs on a continuous basis.[42] Additionally, we provide multiple

pricing options that help reduce TCO for AWS customers. This is achieved via volume discounts[43] and reserved instances.

Reserved Instances

Amazon EC2 Reserved Instances (RI) provide a significant discount (up to 75%) compared to On-Demand pricing and provide a capacity reservation when used in a specific Availability Zone. To provide the capacity guarantee that FI's need for regulatory requirements, customers can purchase zonal Reserved Instances (zonal RI). These RI's are specific to an instance type and assigned to a specific Availability Zone. The Amazon EC2 instances bought via these RIs are guaranteed to be available to you, irrespective of other customer demands for capacity.

- RIs are employed for a variety of use cases, including but not limited to:
- Cost savings for known steady-state workloads
- Reserved capacity for future projects
- Reserved capacity for parallel blue/green deployments
- Reserved capacity for disaster recovery/failover/continuity-of-operations scenarios

RIs operate on a per-minute resolution, so if one instance is shut down, another instance can take advantage of that now-available RI capacity almost immediately. In addition, customers that purchase a large number of Amazon EC2 RIs in an AWS region automatically receive discounted upfront fees and hourly fees for future purchases of Standard RIs in that AWS region.[44]

Maximizing Return

Guaranteeing capacity in other regions can be accomplished via Reserved Instances and specifically via zonal RIs. Critical Tier 1 applications should use zonal RIs to guarantee capacity in their multi-region architectures.

However, since DR events and tests are infrequent, this reserved capacity may go underutilized unless effort is made to maximize usage. Several techniques are available to minimize the time instances are left idle, and maximize the return on investment of an RI purchase:

Run Lower Priority Workloads on the Reserved Capacity

This is the most common way customers take advantage of idle capacity—running lower priority workloads on this excess reserved capacity. FIs can identify lower priority workloads that can be shut down or moved when the capacity is needed for the originating event. Typical examples include:

- Development/Test/QA Environments
- Nodes for distributed computing workloads, such as High Performance Computing (HPC)
- Other workloads that will not materially affect the core business if disabled during the triggering event, such as lower priority report generation, machine learning model training, or internal and back-office systems
- Alternative revenue or charity workloads (e.g. cryptocurrency mining)

Customers can develop runbooks, policies, and automation to disable the low priority workloads with little to no notice, and transition that capacity to the reserving workload when a triggering event occurs.

If corporate processes allow, FIs can consider implementing an internal market. This means providing unused RI capacity to internal teams at a price point that minimizes unused capacity, and can employ an auction-style pricing or airline/hotel-style pricing, where rates are determined by time blocks reserved in advance. Tagging resources used by different workloads can help with internal cost accounting and facilitate adjusting team charge backs appropriately. This also can lower the TCO for applications on AWS.

Application Testing and Certification

Operational resiliency in the cloud is a new approach for many FIs, which have traditionally conducted IT operations on premises. Working with cloud services, infrastructure as code, and automation, is a new way of reacting to events for many enterprises. Technology operation teams within FIs need to develop the skills for building and supporting highly resilient applications on AWS. AWS plans, and encourage its customers to plan and test for failures.

A resilient system continues to operate successfully in the presence of failures. To prepare for a real DR event, FIs can create multiple test scenarios and understand the impact of the failure on an application. The test scenarios and plans should be based on a careful study of the failure modes of the application and hardware. This practice is

called Failure Mode Effect Analysis (FMEA). FMEA is an industry standard [ISO] engineering technique which estimates a risk priority number (RPN) between 1 and 1000, by ranking probability, severity, and observability on a 1-10 scale, where 1 is good and 10 is bad, and multiplying them. A perfectly low probability, low impact, easy to measure risk has an RPN of 1. An extremely frequent, permanently damaging, impossible to detect risk has an RPN of 1000. By listing and rating failure modes, FIs can see which one to focus on. After rating failure modes, FIs can record the expected effect of a mitigation strategy, which should reduce the overall RPN, and shift the focus to the new highest RPN, until there aren't any high values left. In practice, the easiest way to reduce RPN is to add observability, so users aren't working blind.

Users can then get empirical measurements of probability, as visibility allows users to see how often a failure occurs. An example of using failure modes is to create a failure impact analysis, as shown in Table 4. Additional details on FMEA are provided in Appendix F: Failure Modes and Effects Analysis

Table 4

Failure	Effect	Mitigation	Result
Failure of an AZ	Temporary capacity reduction	Automatic failover to secondary AZ	Temporary performance degradation
Total failure of satellite region	Data replication offline	Repair/reconfigure replication using alternate region	No service interruption
Partition of network between regions	Data replication offline	Auto recovery when network is available	No service interruption
Total failure of primary region	Service offline	Failover to secondary region	Service restored within two hours

Application testing should also include tabletop exercises where development and operations teams can hypothesize failure scenarios and evaluate how the applications and the organization will react to the failure. Conducting these exercises will help FIs to understand and create the necessary automation and recovery scripts required to support application recovery during a DR event. The important starting points for tabletop exercises cover foundational IT services:

- Simulating hardware failure at the Amazon EC2 instance level.
- Simulating the failure of storage – Amazon EBS or Amazon S3.
- Simulating the failure of network components such as security groups, VPC peering, VPN connection, Direct Connect connections.
- Simulating the failure of security components such as password lockouts, expiration of application security certificates, misconfiguration of AWS Identity and Access Management (IAM).

After these base-level exercises are completed, you can model advanced scenarios such as the failure of an AZ or Region, data corruption, virus infection, data leak, or distributed denial of service (DDOS) attacks.

Having understood the failure scenarios and defined the responses of the application and organization, FIs then need to put these results into action. This next step could take the form of game days where development teams and operations teams practice the failure scenarios in a non-production environment. With AWS, teams can quickly set up a new duplicate environment for the game day, practice the scenarios. and then completely tear down and dispose of the infrastructure. When you have parallel environments, you can evaluate multiple scenarios. For example, what happens to the application if you kill an application process? What happens to the application when you apply additional load? What happens to the application if you introduce delays in application component responses? What does the user experience look like during these failures?

FIs can test additional scenarios such as load testing across all the applications, by introducing load at various entry points and understanding the performance of applications. How are the databases keeping up with the increased volume of transactions? What are the other bottlenecks in the processing pipeline? A useful mantra to repeat during this stage is that “a chain is only as strong as the weakest link.” This testing model can be extended to incorporate end-user behavior, and industry providers by conducting an industry-wide test scenario. Running mock trades across industry systems while simultaneously testing failures will arm you with the necessary information and operations practices to handle complex failure scenarios.

We recommend that FIs conduct tabletop and game day exercise for individual applications as well as plan for larger groupings of applications. A common practice is to hold continual company-wide tests. During this test, the core assumption is that the primary data center is down and applications are failed over to an alternate location. This is called the disaster recovery test.

Chaos Engineering

Up to this point, we have discussed testing scenarios. Mostly, when we consider these scenarios we think of single failures. Today's applications are building on new platforms such as web, mobile, and Internet of Things (IoT). They are built using distributed technologies and distributed development practices. Even when each individual service within a distributed system is functioning properly, the interactions between those services can cause unpredictable outcomes. How would your application behave if there were multiple failures?

A new method of testing enabled by the cloud is emerging, called "Chaos Engineering," to specifically address the uncertainty of distributed systems at scale. As defined on the Principles of Chaos Engineering website,[45] "Chaos Engineering can be thought of as the facilitation of experiments to uncover systemic weaknesses." These experiments, according to the Chaos Engineering website, follow the principles of:

Building a hypothesis around steady state behavior

Focus on the measurable output of a system, rather than internal attributes of the system. Measurements of that output over a short period of time constitute a proxy for the system's steady state. The overall system's throughput, error rates, latency percentiles, etc. could all be metrics of interest representing steady state behavior. By focusing on systemic behavior patterns during experiments, Chaos Engineering verifies that the system works, rather than simply validating how it works.

Applying variations to simulate real world events

Chaos variables reflect real-world events. Prioritize events either by potential impact or estimated frequency. Consider events that correspond to hardware failures like servers dying, software failures like malformed responses, and non-failure events like a spike in traffic or a scaling event. Any event capable of disrupting steady state is a potential variable in a Chaos experiment.

Run experiments in production

Systems behave differently depending on environment and traffic patterns. Since the behavior of utilization can change at any time, sampling real traffic is the only way to reliably capture the request path. To guarantee both authenticity of the way in which the system is exercised, and relevance to the current deployed system, Chaos Engineering strongly prefers to experiment directly on production traffic.

Automate the experiments to run continuously

Running experiments manually is labor-intensive and ultimately unsustainable. With automated experiments that run continuously, Chaos Engineering builds automation into the system to drive both orchestration and analysis.

Minimize blast radius of failures

Experimenting in production has the potential to cause unnecessary customer pain. While experimenting ensure the fallout from experiments are minimized and contained.

Mastering the appropriate skills through testing brings applications in control over the infrastructure, versus the application being heavily dependent on expensive infrastructure or complex operations. With learnings and automation in place, a financial institution can strive to run applications independently in any region. To test this, we recommend that customers create procedures and automation to migrate all applications to another region without any customer visible impact. A good example of this practice is demonstrated by the steps Netflix[46] has taken for region evacuation.

Conclusion

AWS presents an opportunity for higher levels of resiliency, better security, and more cost-effective operations for the applications of Financial Institutions. The global payment system and its endpoints can become more secure, trading and settlement can become faster and simpler to regulate, dormant DR systems can become active processing capacity, and the data systems that fuel the industry can become more globally available.

AWS is designed to enable many variations for architecting resilient systems. This paper highlights best practices and provides recommendations. Designing for failure is the foundational lens through which the architecture of critical systems are viewed. And FIs can work with an AWS Solutions Architect to collaboratively validate the resiliency of their application using the AWS Well-Architected Framework.

AWS provides solutions, ranging from backup and restore to fault tolerant, multi-site deployments, that FIs can architect as part of their resiliency plans. AWS provides fine-grained control and building blocks to create the appropriate DR solution in the cloud, given an FI's unique resiliency requirements, recovery objectives (RTO and RPO), and budget. AWS services are available across the globe and include mechanisms to reserve capacity. This is a key advantage for resiliency, where significant infrastructure is needed quickly in the event of a disaster. With AWS, FIs can build highly resilient

applications while taking advantage of flexible, cost-effective infrastructure solutions. Financial Institutions that build mission-critical applications on AWS can employ these solutions to shift from a reactive approach to disaster events, to an automated, prepared approach where FIs maintain high levels of resiliency.

Contributors

The following individuals contributed to this document:

- Pawan Agnihotri, Senior Manager Solutions Architecture, Amazon Web Services, Financial Services
- Adrian Cockcroft, VP Cloud Architecture Strategy, Amazon Web Services

Additional contributors:

- Shuja Sohrawardy, Solutions Delivery Manager, Amazon Web Services. Financial Services
- Peter Voshall, VP/Distinguished Engineer, Amazon Web Services
- Rodney Lester, Principle Solutions Architect, Well Architected Framework Resiliency Pillar
- Tim Griesbach, Manager, Public Sector Solutions Architecture, Amazon Web Services
- Harsha Nippani, Public Sector Solutions Architect, Amazon Web Services
- Ashish Palekar, Director of Product Management, Elastic Block Store, Amazon Web Services
- Tony Petrossian, GM DynamoDB, Amazon Web Services
- Dave Brown, VP EC2 Networking, Amazon Web Services
- Emily Smykal, Manager, Financial Services Content, Amazon Web Services
- Robert Kissell, Senior Manager, Solutions Architecture, Amazon Web Services

Further Reading

For additional information, see the following AWS resources:

AWS Documentation

- AWS Well-Architected Framework
- Reliability Pillar of the AWS Well-Architected Framework
- AWS Technical Whitepapers
- AWS Documentation

AWS Presentations: Disaster Recovery

- AWS re:Invent 2017: Disaster Recovery with AWS: Tiered Approaches to Balance Cost (ENT322)
- AWS re:Invent 2015 | (STG304) Deploying a Disaster Recovery Site on AWS
- Netflix: Multi-Regional Resiliency and Amazon Route 53
- The Secret to a Highly Resilient AWS Environment
- AWS re:Invent 2017: How to Design a Multi-Region Active-Active Architecture (ARC319)
- AWS re:Invent 2017: Building Resilient, Multi-Region Serverless Applications (SRV313)
- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- AWS re:Invent 2018: Breaking Containers: Chaos Engineering for Modern Applications on AWS (CON310)

Document Revisions

Date	Description
October 2018	First draft for confidential review
April 2019	Public Release

Appendix A: Financial Services Applications

Examples of Financial Services applications in various tiers:

Table 5

Platinum or Tier 1	Gold or Tier 2	Silver or Tier 3	Bronze or Tier 4
Settlement Applications	Risk systems	Public website	Intranet
Clearing Applications	Compliance systems	Middle office platforms	Back office platforms
Payment Applications	Email / Communications	ERP	Cold storage
Exchange/Trading Platforms		CRM	Internal websites

Appendix B: Designed-For Availability for Select AWS Services

Below, we provide the availability that select AWS services were designed to achieve, from the reliability pillar of the AWS Well-Architected, as of the time of publication of this document. For the most up to date availability figures, readers should consult the reliability pillar. These values do not represent a Service Level Agreement or guarantee, but rather provide insight to the design goals of each service. In certain cases, we differentiate portions of the service where there's a meaningful difference in the availability design goal.

This list is not comprehensive for all AWS services, and we expect to periodically update with information about additional services. Amazon CloudFront, Amazon Route53, and the Identity & Access Management Control Plane provide global service, and the component availability goal is stated accordingly. Other services provide services within an AWS Region and the availability goal is stated accordingly. Many services provide independence between AZs; in these cases we provide the availability design goal for a single AZ, and when any two (or more) AZs are used.

Note: The numbers in the table below do not refer to durability (long term retention of data); they are availability numbers (access to data or functions).

Table 6

Service	Component	Availability Design Goal
Amazon API Gateway	Control Plane	99.950%
	Data Plane	99.990%
Amazon Aurora	Control Plane	99.950%
	Single AZ Data Plane	99.950%
	Multi AZ Data Plane	99.990%
AWS CloudFormation	Service	99.950%
Amazon CloudFront	Control Plane	99.900%
	Data Plane (content delivery)	99.990%
Amazon CloudSearch	Control Plane	99.950%
	Data Plane	99.950%
Amazon CloudWatch	CW Metrics (service)	99.990%
	CW Events (service)	99.990%
	CW Logs (service)	99.950%
AWS Data Pipeline	Service	99.990%
Amazon DynamoDB	Service (standard)	99.990%
	Service (Global Tables)	99.999%
Amazon EC2	Control Plane	99.950%
	Single AZ Data Plane	99.950%
	Multi AZ Data Plane	99.990%
Amazon ElastiCache	Service	99.990%

Service	Component	Availability Design Goal
Amazon Elastic Block Store	Control Plane	99.950%
	Data Plane (volume availability)	99.999%
Amazon Elasticsearch	Control Plane	99.950%
	Data Plane	99.950%
Amazon EMR	Control Plane	99.950%
Amazon Glacier	Service	99.900%
AWS Glue	Service	99.990%
Amazon Kinesis Streams	Service	99.990%
Amazon RDS	Control Plane	99.950%
	Single AZ Data Plane	99.950%
	Multi AZ Data Plane	99.990%
Amazon Rekognition	Service	99.980%
Amazon Redshift	Control Plane	99.950%
	Data Plane	99.950%
Amazon Route53	Control Plane	99.950%
	Data Plane (query resolution)	100.000%
Amazon SageMaker	Data Plane (Model Hosting)	99.990%
	Control Plane	99.950%
Amazon S3	Service (Standard)	99.990%
AWS Auto Scaling	Control Plane	99.900%
	Data Plane	99.990%
AWS Batch	Control Plane	99.900%
	Data Plane	99.950%

Service	Component	Availability Design Goal
AWS CloudHSM	Control Plane	99.900%
	Single AZ Data Plane	99.900%
	Multi AZ Data Plane	99.990%
AWS CloudTrail	Control Plane (config)	99.900%
	Data Plane (data events)	99.990%
	Data Plane (management events)	99.999%
AWS Config	Service	99.950%
AWS Direct Connect	Control Plane	99.900%
	Single Location Data Plane	99.900%
	Multi Location Data Plane	99.990%
AWS Elastic File Store	Control Plane	99.950%
	Data Plane	99.990%
AWS Identity & Access Management	Control Plane	99.900%
	Data Plane (authentication)	99.995%
AWS Lambda	Function Invocation	99.950%
AWS Shield	Control Plane	99.500%
	Data Plane (detection)	99.000%
	Data Plane (mitigation)	99.900%
AWS Storage Gateway	Control Plane	99.950%
	Data Plane	99.950%
AWS X-Ray	Control Plane (console)	99.900%
	Data Plane	99.950%
EC2 Container Service	Control Plane	99.900%
	EC2 Container Registry	99.990%

Service	Component	Availability Design Goal
	EC2 Container Service	99.990%
Elastic Load Balancing	Control Plane	99.950%
	Data Plane	99.990%
Key Management System (KMS)	Control Plane	99.990%
	Data Plane	99.995%

Appendix C: Service Capabilities

Table 7

Service Area	AWS Service Name	Service Scope	Capacity Reservation	CRR support?
Database				
	RDS MySQL	Regional	Yes (Reserved Instances)	Yes
	RDS MariaDB	Regional	Yes (Reserved Instances)	Yes
	RDS PostgreSQL	Regional	Yes (Reserved Instances)	Yes
	RDS Oracle	Regional	Yes (Reserved Instances)	Yes
	RDS SQL Server	Regional	Yes (Reserved Instances)	Yes
	Aurora MySQL	Regional	Yes (Reserved Instances)	Yes
	Aurora PostgreSQL	Regional	Yes (Reserved Instances)	
	Redshift	Regional	Yes (Reserved Instances)	Yes (snapshot)

Service Area	AWS Service Name	Service Scope	Capacity Reservation	CRR support?
	DynamoDB	Regional	Yes (Reserved Instances)	Yes
	Neptune	Regional	Yes (Reserved Instances)	Yes
	ElastiCache	Regional	Yes (Reserved Instances)	Yes (snapshot)
	DocumentDB	Regional		
Compute and components				
	EC2 (Resource Identifiers)	Regional		
	EC2 (EIP)	Regional		
	EC2 Auto Scaling	Regional		
	EC2 (Security Groups)	Regional		
	EC2 (Elastic Load Balancers)	Regional		
	EC2 (Placement Group)	Availability Zone		
	EC2 (Instances)	Availability Zone	Yes (Reserved Instances)	
	EC2 (EBS Volumes)	Availability Zone		
	EC2 (EBS snapshots	Regional		
	Oracle on EC2		Yes (Reserved Instances)	Yes (manual)
	EC2 AMIs	Regional		Yes
	Elastic Beanstalk	Regional	Yes (Reserved Instances)	Yes

Service Area	AWS Service Name	Service Scope	Capacity Reservation	CRR support?
	ECS	Regional	Yes (Reserved Instances)	
	EKS	Regional	Yes (Reserved Instances)	
	Fargate	Regional	Yes (Reserved Instances)	
	Lambda	Regional		
	Batch	Regional		
	ECR	Regional		
	ELB	Regional		
Storage				
	EBS Volumes	Availability Zone		Yes
	S3	Regional		Yes
	EFS	Regional		Yes
	Storage Gateway	Regional		Yes
	FSx	Regional		
	AWS Backup	Regional		
Networking & Content Delivery				
	VPC	Regional		
	VPC (Security Groups, Endpoints)	Regional		
	VPC (Subnets)	Availability Zone		

Service Area	AWS Service Name	Service Scope	Capacity Reservation	CRR support?
	VPC (inter-region peering)	Regional		Yes
	Route 53 (DNS)	Global		Yes
	Direct Connect Gateway			Yes
	CloudFront	Global		
	Storage Gateway	Regional		
	API Gateway	Regional		
	Direct Connect	Regional		
	Cloud Map	Regional		
	Global Accelerator	Global		
Management & Governance				
	CloudFormation	Regional		Yes
	CloudWatch	Regional		
	CloudWatch Events	Regional		
	CloudWatch Logs	Regional		
	CloudTrail	Regional		
	Config	Regional		
	Service Catalog	Regional		
	Systems Manager	Regional		
	Control Tower	Regional		
	OpsWorks	Regional		
	Auto Scaling	Regional		
Application Integration				

Service Area	AWS Service Name	Service Scope	Capacity Reservation	CRR support?
	Simple Queue Services (SQS)	Regional		
	Simple Notification Service (SNS)	Regional		
	Simple Queue Services (SQS)	Regional		
	Amazon MQ	Regional		
Security, Identity & Compliance				
	IAM (Users, Groups, Roles, Accounts)	Global (except China)		Yes
	IAM (Key pairs)	Regional		
	Organizations	Global		
	KMS	Regional		
	GuardDuty	Regional		
	WAF & Shield	Regional		
	Secrets Manager	Regional		
	Certificate Manager	Regional		
	Cognito	Regional		
	Directory Service	Regional		
	AD Connector	Regional		
	Inspector	Regional		
Migration & Transfer				
	Migration Hub	Regional		

Service Area	AWS Service Name	Service Scope	Capacity Reservation	CRR support?
	Application Discovery Service	Regional		
	Database Migration Service	Regional		
	Server Migration Service	Regional		
	Transfer for SFTP	Regional		
	Snowball	Regional		
Analytics				
	Athena	Regional		
	EMR	Regional		
	ElasticSearch	Regional		
	Kinesis	Regional		
	QuickSight	Regional		
	Glue	Regional		
	CloudSearch	Regional		
Developer Tools				
	CodeCommit	Regional		
	CodeBuild	Regional		
	CodeDeploy	Regional		
	CodePipeline	Regional		
	Cloud9	Regional		
	CodeStart	Regional		
Machine Learning				

Service Area	AWS Service Name	Service Scope	Capacity Reservation	CRR support?
	SageMaker	Regional		
	Comprehend	Regional		
	Lex	Regional		
	Polly	Regional		
	Rekognition	Regional		
	Transcribe	Regional		
	Translate	Regional		
	Forecast	Regional		
	Textract	Regional		

Appendix D: Disaster Recovery Checklist

The checklist below is a suggested guide for FIs that are building resiliency and DR plans in the AWS cloud:

Application Readiness

Dependencies

- ☐ Have all upstream/downstream dependencies, applications, repositories/databases been identified?
- ☐ Have the teams responsible for those entities been engaged, and interfaces for the target region been identified?
- ☐ Has the move back been considered/planned? How will data written to former read-replicas now be synced back to the original region primary source? Will any instances in the target environment need to be spun down for cost considerations?

Configuration

- ☐ How will any upstream or downstream application initiating communication know to contact the target location instance instead?

- ☐ Does the application examine metadata to determine in which region is it currently running, and access resources appropriately?
- ☐ Does the application programmatically use appropriate local endpoints depending upon location for these services?
    - ✓ Amazon S3 (Application data, application configuration, and all bootstrap/startup/user data scripts)
    - ✓ Directory Connectors
    - ✓ Amazon SES Endpoints
    - ✓ Amazon SQS
    - ✓ Amazon SNS
    - ✓ AWS Lambda
    - ✓ Amazon API Gateway
    - ✓ Elements Endpoints
    - ✓ Amazon Elasticsearch domains
    - ✓ Interfaces for upstream/downstream dependencies
- ☐ If using DNS for service discovery:
    - ✓ Have DNS caching issues (such as permanent caching within the Java JVM) been addressed?
    - ✓ Are DNS TTL Settings set appropriately?
- ☐ Is automation (such as Amazon Route 53 health checks) or are manual runbooks in place to redirect user or system usage to the target environment? The following services may need considered:
    - ✓ AWS Lambda event triggers
    - ✓ Amazon Route 53 DNS
    - ✓ Amazon S3
    - ✓ Amazon RDS

- ✓ Amazon DynamoDB
- ✓ Amazon EFS volumes

Some services, such as Amazon RDS, Amazon S3, and Amazon EFS, may need additional work to promote former read-replicas to write access as well.

Environment Readiness – DR Region

Configuration/Automation

- ☐ Since the source region and the target region have a differing number of availability zones (US-East-1 has five; US-West-2 has three), has the target region architecture been adjusted to account for this?
- ☐ Are all service limits configured identical in the DR Region, such as the number of M4/C4 Amazon EC2 instance limits, EBS Volume limits, etc.?
- ☐ Have all Amazon Auto Scaling, AWS Cloud Formation Templates, or user generated/third party scripts been adjusted for the new AMI IDs for the target region?
- ☐ Have all services had their features and configurations set up identically? For example, have Amazon DynamoDB indexes been configured identically, and do all Amazon Kinesis streams have the same number of shards configured?
- ☐ Have region-based services (such as Amazon SQS or AWS Lambda) been duplicated in the target environment?
- ☐ AWS is working on matching service and API limits between regions. Has a process been put in place to ensure all future increases ask for both regions to keep them in parity?

Compute

- ☐ Are all Amazon EC2/RDS instances needed in the target environment reserved so they will be available when needed?
- ☐ Have all needed startup data/bootstrap scripts been replicated to the target environment?
- ☐ Are appropriate SSH keys available in both regions?

- ☐ Is Auto Scaling configured for all environments in the target region?
- ☐ Are all Amazon EC2 AMIs patched and synced with the DR Region?
- ☐ Have the known hosts file for SSH been synced and updated on all systems?
- ☐ Are OS-level UIDs matched across both regions? (This may be an issue with replicating between two hosts that do not share a common user database.)

Network

- ☐ Is the target environment already in place, or are AWS CloudFormation templates prepared for creating the target environment when needed?
- ☐ Are all needed load balancers (elastic/network/application) created or templated in the target region?
- ☐ Has provisioned capacity been enabled for ELBs to ensure capacity when needed?
- ☐ Are NACLs and Security Groups configured identically for both regions?
- ☐ Are the VPC IDs, Security Group IDs, Subnet IDs, etc. appropriately configured in automation scripts such as AWS CloudFormation, Auto Scaling Launch Configurations, etc.
- ☐ Have all IP address space configurations, such as firewall rules or licenses, been matched across both regions, including with third parties? This may be an issue with communication with third parties such as Concur, Bloomberg, or Journal & Library services as they often restrict access and/or licensing based upon source IP.

Storage

- ☐ Are Amazon EBS performance characteristics, such as provisioned IOPs, confirmed identical?
- ☐ Are replications configured for all needed services? Areas for consideration include:
    - ✓ Amazon S3 buckets
    - ✓ Amazon EFS shares
    - ✓ Directories
- ☐ Are all backup/snapshots of data volumes used for failover done in a way that ensures a consistent state, such as quiescing a database before the snapshot?

- ☐ Are replications configured for all needed services? Areas for consideration include:
    - ✓ Databases (Amazon RDS, Amazon DynamoDB, Amazon ElastiCache, Amazon Redshift, or self-managed)

Security & Logging

- ☐ Are all TLS/SSL certificates verified and validated in the DR Region? Are needed certificate authentication chains also available?
- ☐ Are all needed secret stores (such as those used for control administrator or DB access) available in the DR regions?
- ☐ Is the same Amazon CloudWatch/AWS CloudTrail logging enabled in the target region? Have dashboards been configured the same across both regions?
- ☐ Do all services have logging enabled in the target environment? Examples for consideration include:
    - ✓ Amazon S3
    - ✓ Elastic Load Balancing
    - ✓ Amazon RDS (event plus any special engine logging options)
    - ✓ Amazon CloudWatch logs
- ☐ Are all needed Active Directory replication or connectors also available in the target region?
- ☐ While IAM policies are global, ensure all rules and conditions with region, IP address, or timeframe limits configured are adjusted to also apply to the target region.
- ☐ If using a separate account in the target region, have all needed IAM roles and permissions been created in the separate account? Have all upstream or downstream dependencies also adjusted their IAM roles and policies?
- ☐ Are all management integrations, such as SIEM, Datadog, etc., in place on the target environment?

Is encryption key management done appropriately across both regions? Are all customer master keys available in both regions, or are multi-region CMK masters in use?

Appendix E: List of Service Level Agreements for AWS Services

This list is accurate as of April 11, 2019. Please visit the AWS Service Level Agreements public website[47] for the most up to date information. Users should review the specific SLA link for exact details of each SLA.

Table 8

Service Name	SLA* (see SLA URL)	SLA Last Update	SLA URL for Details
Alexa for Business	99.9%	March 19, 2019	https://aws.amazon.com/alexaforbusiness/sla/
Amazon API Gateway	99.95%	March 20, 2019	https://aws.amazon.com/api-gateway/sla/
Amazon AppStream 2.0	99.9%	March 14, 2019	https://aws.amazon.com/appstream2/amazon-appstream-2-0-service-level-agreement/
Amazon Athena	99.9%	March 14, 2019	https://aws.amazon.com/athena/sla/
Amazon Aurora	99.99%	March 21, 2019	https://aws.amazon.com/rds/aurora/sla/
Amazon Chime	99.9%	March 19, 2019	https://aws.amazon.com/chime/sla/
Amazon Cloud Directory	99.9%	March 14, 2019	https://aws.amazon.com/cloud-directory/sla/
Amazon CloudFront	99.9%	March 14, 2019	https://aws.amazon.com/cloudfront/sla/
Amazon CloudSearch	99.9%	March 20, 2019	https://aws.amazon.com/cloudsearch/sla/
Amazon CloudWatch	99.9%	March 19, 2019	https://aws.amazon.com/cloudwatch/sla/
Amazon Cognito	99.9%	March 6, 2019	https://aws.amazon.com/cognito/sla/

Service Name	SLA* (see SLA URL)	SLA Last Update	SLA URL for Details
Amazon Compute	99.99%	March 19, 2019	https://aws.amazon.com/compute/sla/
Amazon Connect	99.99%	March 21, 2019	https://aws.amazon.com/connect/sla/
Amazon Database Migration	99.9%	March 6, 2019	https://aws.amazon.com/dms/sla/
Amazon DocumentDB (with MongoDB compatibility)	99.9%	March 20, 2019	https://aws.amazon.com/documentdb/sla/
Amazon DynamoDB Global Tables	99.999%	March 14, 2019	https://aws.amazon.com/dynamodb/sla/
Amazon DynamoDB Standard	99.99%	March 14, 2019	https://aws.amazon.com/dynamodb/sla/
Amazon EC2	99.99%	March 19, 2019	https://aws.amazon.com/compute/sla/
Amazon EFS	99.9%	March 19, 2019	https://aws.amazon.com/efs/sla/
Amazon EKS	99.9%	March 19, 2019	https://aws.amazon.com/eks/sla/
Amazon Elastic Block Store (Amazon EBS)	99.99%	March 19, 2019	https://aws.amazon.com/compute/sla/
Amazon Elastic Container Registry	99.9%	March 15, 2019	https://aws.amazon.com/ecr/sla/
Amazon Elastic Container Service (Amazon ECS)	99.99%	March 19, 2019	https://aws.amazon.com/compute/sla/
Amazon Elastic Load Balancing	99.99%	March 13, 2019	https://aws.amazon.com/elasticloadbalancing/sla/
Amazon Elastic Transcoder	99.9%	March 13, 2019	https://aws.amazon.com/elastictranscoder/sla/
Amazon ElastiCache	99.9%	March 20, 2019	https://aws.amazon.com/elasticache/sla/

Service Name	SLA* (see SLA URL)	SLA Last Update	SLA URL for Details
Amazon Elasticsearch Service	99.9%	March 20, 2019	https://aws.amazon.com/elasticsearch-service/sla/
Amazon EMR	99.9%	March 12, 2019	https://aws.amazon.com/emr/sla/
Amazon Fargate for Amazon ECS (Amazon Fargate)	99.99%	March 19, 2019	https://aws.amazon.com/compute/sla/
Amazon FSx	99.9%	March 15, 2019	https://aws.amazon.com/fsx/sla/
Amazon GuardDuty	99.9%	March 13, 2019	https://aws.amazon.com/guardduty/sla/
Amazon Inspector	99.9%	March 16, 2019	https://aws.amazon.com/inspector/sla/
Amazon Kinesis Data Firehose	99.9%	March 20, 2019	https://aws.amazon.com/kinesis/data-firehose/sla/
Amazon Kinesis Data Streams	99.9%	March 20, 2019	https://aws.amazon.com/kinesis/data-streams/sla/
Amazon Kinesis Video Streams	99.9%	March 20, 2019	https://aws.amazon.com/kinesis/video-streams/sla/
Amazon Lightsail Instance and Block Storage	99.99%	March 15, 2019	https://aws.amazon.com/lightsail/sla-lightsail-instances-and-block-storage/
Amazon Lightsail Managed Databases	99.95%	March 15, 2019	https://aws.amazon.com/lightsail/sla-lightsail-managed-databases/
Amazon Machine Learning Language	99.9%	March 14, 2019	https://aws.amazon.com/machine-learning/language/sla/
Amazon Macie	99.9%	March 15, 2019	https://aws.amazon.com/macie/sla/
Amazon Messaging (SQS, SNS)	99.9%	March 19, 2019	https://aws.amazon.com/messaging/sla/

Service Name	SLA* (see SLA URL)	SLA Last Update	SLA URL for Details
Amazon MQ	99.9%	March 19, 2019	https://aws.amazon.com/amazon-mq/sla/
Amazon Neptune	99.9%	March 8, 2019	https://aws.amazon.com/neptune/sla/
Amazon QuickSight	99.9%	March 19, 2019	https://aws.amazon.com/quicksight/sla/
Amazon RDS	99.95%	March 21, 2019	https://aws.amazon.com/rds/sla/
Amazon Redshift	99.9%	March 19, 2019	https://aws.amazon.com/redshift/sla/
Amazon Rekognition	99.9%	March 20, 2019	https://aws.amazon.com/rekognition/sla/
Amazon Route 53	100.0%	November 21, 2018	https://aws.amazon.com/route53/sla/
Amazon S3	99.9%	March 20, 2019	https://aws.amazon.com/s3/sla/
Amazon SageMaker	99.95%	March 20, 2019	https://aws.amazon.com/sagemaker/sla/
Amazon Simple Workflow	99.9%	March 19, 2019	https://aws.amazon.com/swf/sla/
Amazon SimpleDB	99.9%	March 21, 2019	https://aws.amazon.com/simpledb/sla/
Amazon Storage Gateway	99.9%	March 15, 2019	https://aws.amazon.com/transfer/sla/
Amazon User Engagement (Pinpoint, SES)	99.9%	March 18, 2019	https://aws.amazon.com/pinpoint/sla/
Amazon VPC NAT Gateway	99.9%	March 14, 2019	https://aws.amazon.com/vpc/sla/
Amazon WorkDocs	99.9%	March 14, 2019	https://aws.amazon.com/workdocs/sla/

Service Name	SLA* (see SLA URL)	SLA Last Update	SLA URL for Details
Amazon WorkLink	99.9%	March 19, 2019	https://aws.amazon.com/worklink/amazon-worklink-service-level-agreement/
Amazon WorkMail	99.9%	March 19, 2019	https://aws.amazon.com/workmail/amazon-workmail-service-level-agreement/
Amazon WorkSpaces	99.9%	March 19, 2019	https://aws.amazon.com/workspaces/sla/
AWS Amplify Console	99.95%	March 19, 2019	https://aws.amazon.com/amplify/console/sla/
AWS AppSync	99.95%	March 19, 2019	https://aws.amazon.com/appsync/sla/
AWS Backup	99.9%	March 18, 2019	https://aws.amazon.com/backup/sla/
AWS Budgets	99.9%	March 8, 2019	https://aws.amazon.com/aws-cost-management/aws-budgets/sla/
AWS Certificate Manager Private Certificate Authority	99.9%	March 15, 2019	https://aws.amazon.com/certificate-manager/private-certificate-authority/sla/
AWS Client VPN	99.9%	March 13, 2019	https://aws.amazon.com/vpn/client-vpn-sla/
AWS Cloud Map	99.9%	March 12, 2019	https://aws.amazon.com/cloud-map/sla/
AWS CloudHSM	99.95%	March 6, 2019	https://aws.amazon.com/cloudhsm/sla/
AWS CloudTrail	99.9%	March 18, 2019	https://aws.amazon.com/cloudtrail/sla/
AWS CodeBuild	99.9%	March 14, 2019	https://aws.amazon.com/codebuild/sla/
AWS CodeCommit	99.9%	March 14, 2019	https://aws.amazon.com/codecommit/sla/

Service Name	SLA* (see SLA URL)	SLA Last Update	SLA URL for Details
AWS CodeDeploy	99.9%	March 14, 2019	https://aws.amazon.com/codedeploy/sla/
AWS CodePipeline	99.9%	March 21, 2019	https://aws.amazon.com/codepipeline/sla/
AWS Config	99.9%	March 12, 2019	https://aws.amazon.com/config/sla/
AWS Cost Explorer API	99.9%	March 8, 2019	https://aws.amazon.com/aws-cost-management/aws-cost-explorer/sla/
AWS Data Pipeline	99.9%	March 19, 2019	https://aws.amazon.com/datapipeline/sla/
AWS DataSync	99.9%	March 15, 2019	https://aws.amazon.com/transfer/sla/
AWS Device Farm	99.9%	March 19, 2019	https://aws.amazon.com/device-farm/sla/
AWS Direct Connect	99.99%	March 20, 2019	https://aws.amazon.com/directconnect/sla/
AWS Directory Service	99.9%	March 6, 2019	https://aws.amazon.com/directoryservice/sla/
AWS Elemental MediaConnect	99.9%	March 13, 2019	https://aws.amazon.com/mediaconnect/sla/
AWS Elemental MediaConvert	99.9%	March 13, 2019	https://aws.amazon.com/mediaconvert/sla/
AWS Elemental MediaLive	99.9%	March 13, 2019	https://aws.amazon.com/medialive/sla/
AWS Elemental MediaPackage	99.9%	March 13, 2019	https://aws.amazon.com/mediapackage/sla/
AWS Elemental MediaStore	99.9%	March 13, 2019	https://aws.amazon.com/mediastore/sla/
AWS Elemental MediaTailor	99.9%	March 13, 2019	https://aws.amazon.com/mediatailor/sla/

Service Name	SLA* (see SLA URL)	SLA Last Update	SLA URL for Details
AWS Firewall Manager	99.9%	March 6, 2019	https://aws.amazon.com/firewall-manager/sla/
AWS GameLift	99.9%	March 15, 2019	https://aws.amazon.com/gamelift/sla/
AWS Global Accelerator	99.99%	March 14, 2019	https://aws.amazon.com/global-accelerator/sla/
AWS Glue	99.9%	March 20, 2019	https://aws.amazon.com/glue/sla/
AWS Hybrid Storage and Data Transfer	99.9%	March 15, 2019	https://aws.amazon.com/transfer/sla/
AWS IoT 1-Click	99.9%	March 14, 2019	https://aws.amazon.com/iot-1-click/sla/
AWS IoT Analytics	99.9%	March 19, 2019	https://aws.amazon.com/iot-analytics/sla/
AWS IoT Core	99.9%	March 19, 2019	https://aws.amazon.com/iot-core/sla/
AWS IoT Device Defender	99.9%	March 19, 2019	https://aws.amazon.com/iot-device-defender/sla/
AWS IoT Device Management	99.9%	March 19, 2019	https://aws.amazon.com/iot-device-management/sla/
AWS IoT Device Management	99.9%	March 19, 2019	https://aws.amazon.com/iot-device-management/sla/
AWS IoT Greengrass	99.9%	March 19, 2019	https://aws.amazon.com/greengrass/sla/
AWS Key Management Service	99.9%	March 6, 2019	https://aws.amazon.com/kms/sla/
AWS Lambda	99.95%	March 19, 2019	https://aws.amazon.com/lambda/sla/
AWS OpsWorks	99.9%	March 7, 2019	https://aws.amazon.com/opsworks/sla/

Service Name	SLA* (see SLA URL)	SLA Last Update	SLA URL for Details
AWS PrivateLink	99.9%	March 20, 2019	https://aws.amazon.com/privatelink/sla/
AWS RoboMaker	99.9%	March 14, 2019	https://aws.amazon.com/robomaker/sla/
AWS Secrets Manager	99.9%	March 6, 2019	https://aws.amazon.com/secrets-manager/sla/
AWS Security Hub	99.9%	March 20, 2019	https://aws.amazon.com/security-hub/sla/
AWS Service Catalog	99.9%	March 14, 2019	https://aws.amazon.com/servicecatalog/sla/
AWS Shield Advanced	See SLA URL	March 6, 2019	https://aws.amazon.com/shield/sla/
AWS Site-to-Site VPN	99.95%	March 13, 2019	https://aws.amazon.com/vpn/site-to-site-vpn-sla/
AWS Step Functions	99.9%	March 19, 2019	https://aws.amazon.com/step-functions/sla/
AWS Systems Manager	99.9%	March 6, 2019	https://aws.amazon.com/systems-manager/sla/
AWS Transfer for SFTP	99.9%	March 15, 2019	https://aws.amazon.com/transfer/sla/
AWS Transit Gateway	99.95%	March 14, 2019	https://aws.amazon.com/transit-gateway/sla/
AWS WAF	99.95%	March 6, 2019	https://aws.amazon.com/waf/sla/

Appendix F: Failure Modes and Effects Analysis

The Failure Modes and Effects Analysis (FMEA) spreadsheet below is used to capture and prioritize risks based on severity, probability and detectability where each is rated on a 1 to 10 scale. A standard model for each follows, including suggested rankings

Severity starts with several high levels that destroy things, in other words, irreversible failures like death or incapacitation of a person, destruction of machinery, flood, or fire in a data center. The subsequent levels are temporary incapacitation, recoverable with degradation of performance, and finally ratings of minor or no effect.

Table 9

Effect	Severity of effect	Ranking
Hazardous without warning	Very high severity ranking when a potential failure mode affects safe system operation without warning	10
Hazardous with warning	Very high severity ranking when a potential failure mode affects safe system operation with warning	9
Very High	System inoperable with destructive failure without compromising safety	8
High	System inoperable with equipment damage	7
Moderate	System inoperable with minor damage	6
Low	System inoperable without damage	5
Very Low	System operable with significant degradation of performance	4
Minor	System operable with some degradation of performance	3
Very Minor	System operable with minimal interference	2
None	No effect	1

For probability, we use an exponential scale, from almost inevitable and repeated observed failures down through occasional failures and failures that haven't been seen in practice. The probabilities are estimates during the design phase, but should be measured in real life when a system is operating, and the risk updated based on what is seen in practice.

Table 10

Probability of Failure	Failure Probability	Ranking
Very High: Failure is almost inevitable	>1 in 2	10

Probability of Failure	Failure Probability	Ranking
	1 in 3	9
High: Repeated failures	1 in 8	8
	1 in 20	7
Moderate: Occasional failures	1 in 80	6
	1 in 400	5
	1 in 2,000	4
Low: Relatively few failures	1 in 15,000	3
	1 in 150,000	2
Remote: Failure is unlikely	<1 in 1,500,000	1

Detectability is focused on design control, metrics that track behavior of the system, and alerting rules that can initiate an incident to investigate a fault. If there is no way to detect the failure, it gets a high score. If we have a robust and well-tested alert for the issue and a clear incident handling process in place, it gets the lowest score.

Table 11

Detection	Likelihood of Detection by Design Control	Ranking
Absolute Uncertainty	Design control cannot detect potential cause/mechanism and subsequent failure mode	10
Very Remote	Very remote chance the design control will detect potential cause/mechanism and subsequent failure mode	9
Remote	Remote chance the design control will detect potential cause/mechanism and subsequent failure mode	8
Very Low	Very low chance the design control will detect potential cause/mechanism and subsequent failure mode	7
Low	Low chance the design control will detect potential cause/mechanism and subsequent failure mode	6
Moderate	Moderate chance the design control will detect potential cause/mechanism and subsequent failure mode	5

Detection	Likelihood of Detection by Design Control	Ranking
Moderately High	Moderately High chance the design control will detect potential cause/mechanism and subsequent failure mode	4
High	High chance the design control will detect potential cause/mechanism and subsequent failure mode	3
Very High	Very high chance the design control will detect potential cause/mechanism and subsequent failure mode	2
Almost Certain	Design control will detect potential cause/mechanism and subsequent failure mode	1

The spreadsheet is organized into sections, listing failure modes for each function. There is also a recommended action, listing who is responsible and when, actions taken and the updated severity, occurrence, and detectability that lead to a planned reduction in the RPN. The rows are shown split below for readability. The only formula needed is RPN=Sev*Prob*Det.

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	Sev	Potential Cause(s)/ Mechanism(s) of Failure	Prob	Current Design Controls	Det	RPN

Recommended Action(s)	Responsibility & Target Completion Date	Action Results				
		Actions Taken	New Sev	New Occ	New Det	New RPN

Application Layer FMEA

The first FMEA models the application layer, assuming it is implementing a web page or network accessed API. Each step in the access protocol is modelled as a possible failure mode, starting with authentication, then the access itself. This is followed by common code related failure modes. For a specific application team, these should be discussed, prioritized, and have additional failure modes added.

Table 12

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
Authentication	Client can't authenticate	Can't connect application	5	Certificate timeout, version mismatch, account not setup, credential changed	3	Log and alert on authentication failures	3	**45**	
	Slow or unreliable authentication	Slow start for application	4	Auth service overloaded, high error and retry rate	3	Log and alert on high authentication latency and errors	4	**48**	
								0	
Client Request to API Endpoint	Service unknown, address un-resolvable	Delay while discovery or DNS times out, slow fallback response	5	DNS configuration error, denial of service attack, or provider failure	1	Customer eventually complains via call center	10	**50**	Dual redundant DNS, fallback to local cache, hardcoded IP addresses. Endpoint monitoring and alerts

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
	Service unreachable, request undeliverable	Fast fail, no response	4	Network route down or no service instances running	1	Autoscaler maintains a number of healthy instances	1	**4**	Endpoint monitoring and alerts
	Service reachable, request undeliverable	Connect timeout, slow fail, no response	4	Service frozen/not accepting connection	1	Retry request on different instance. Healthcheck failure instances removed. Log and alert.	2	**8**	
	Request delivered, no response - stall	Application request timeout, slow fail, no response	4	Broken service code, overloaded CPU or slow dependencies	1	Retry request on different instance. Healthcheck failure instances removed. Log and alert.	2	**8**	
	Response undeliverable	Application request timeout, slow fail, no response	4	Network return route failure, dropped packets	1	Retry request on different instance. Healthcheck failure instances removed. Log and alert.	2	**8**	

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
	Response received in time but empty or unintelligible	Fast fail, no response	3	Version mismatch or exception in service code	2	Retry request on different instance. Healthcheck failure instances removed. Log and alert.	2	**12**	
	Request delivered, response delayed beyond spec	Degraded response arrives too late, slow fallback response	6	Service overloaded or GC hit, dependent services responding slowly	2	Retry request on different instance. Healthcheck failure instances removed. Log and alert.	2	**24**	
	Request delivered, degraded response delivered in time	Degraded timely response	2	Service overloaded or GC hit, dependent services responding slowly	2	Log and alert on high service latency and errors	2	**8**	
Time Bombs	Internal application counter wraparound								Test long running operations of code base

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
	Memory leak								Monitor process sizes and garbage collection intervals over time
Date Bombs	Leap year, leap second, epoch wrap around, "Y2K"								Test across date boundaries
Content Bombs	Incoming data that crashes the app								Fuzz the input with generated random and structured data to show it doesn't crash.
Configuration Errors	Configuration file syntax errors or incorrect values								Canary test deployments incrementally. Chaos testing.

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
Versioning Errors	Incompatible interface versions								Canary test deployments incrementally
Retry Storms	Too many retries, too large timeout values								Chaos testing applications under stress
Excessive Logging	Cascading overload								Chaos testing applications under stress

Software Stack FMEA

The software stack starts along the same lines, with authentication and a request response sequence. However, the more specific failure modes relate to the control planes for services hosted in cloud regions. In general, a good way to avoid customer-visible issues caused by control plane failure modes is to pre-allocate network, compute, and database structures wherever possible. The cost of failure should be weighed against the cost of mitigation.

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
Authentication to cloud services	Client can't authenticate	Can't connect application	5	Certificate timeout, version mismatch, account not setup, credential changed	3	Log and alert on authentication failures	3	**45**	
	Slow or unreliable authentication	Slow start for application	4	Auth service overloaded, high error and retry rate	3	Log and alert on high authentication latency and errors	4	**48**	
								0	
Client request to cloud service endpoint	Service unknown, address un-resolvable	Delay while discovery or DNS times out, slow fallback response	5	DNS configuration error, denial of service attack, or provider failure	1			**0**	
	Service unreachable, request undeliverable	Fast fail, no response	4	Network route down or no service instances running	1			**0**	

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
	Service reachable, request undeliverable	Connect timeout, slow fail, no response	4	Service frozen/not accepting connection	1			**0**	
	Request delivered, no response - stall	Application request timeout, slow fail, no response	4	Broken service code, overloaded CPU or slow dependencies	1			**0**	
	Response undeliverable	Application request timeout, slow fail, no response	4	Network return route failure, dropped packets	1			**0**	
	Response received in time but empty or unintelligible	Fast fail, no response	3	Version mismatch or exception in service code	2			**0**	

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
	Request delivered, response delayed beyond spec	Degraded response arrives too late, slow fallback response	6	Service overloaded or GC hit, dependent services responding slowly	2			**0**	
	Request delivered, degraded response delivered in time	Degraded timely response	2	Service overloaded or GC hit, dependent services responding slowly	2			**0**	
EC2 Control Plane	Instance request refused, direct or via autoscaler	Capacity limited or control plane failure		Limit reached, or Insufficient Capacity Exception					Service call for increased limit. Try a different instance type, different zone, or different region
	Instance created but fails to start	Bad instance hardware						0	Retry via autoscaler

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
	Instance slow to start								
EC2 Network Control Plane	Configuration request refused	Capacity limited or control plane failure		Limit reached, or Insufficient Capacity Exception					Service call for increased limit. Try a different zone, or different region
	Network creation started but operation fails							0	Pre-allocate all network structures in all regions
Database Control Plane (DynamoDB or Aurora)	Configuration request refused							0	Service call for increased limit. Try a different zone, or different region
	Database table creation started but operation fails							0	Pre-allocate all database tables in all regions

Infrastructure FMEA

Service-specific control plane outages are part of the software stack FMEA. If a data center building is destroyed by fire or flood, this is a different kind of failure compared to a temporary power outage or cooling system failure, and that's also different from losing connectivity to a building where all the systems are still running, but isolated. In practice, we can expect individual machines to fail randomly with very low probability, groups of similar machines to fail in a correlated way due to bad batches of components and firmware bugs, and extremely rare availability zone-scoped events caused by weather, earthquake, fire, and flood.

Table 13

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
Availability Zone Durability	Permanent destruction of zone	Total data loss in zone	8	Fire or flood inside building or destruction of data center building	2	Cross zone synchronous replication to over 10Km away	1	16	Ensure that system can run on two out of three zones
	Temporary loss of zone	Loss of compute capacity and non-durable state in zone	5	Power or cooling outage causes reboot of part or all of a data center building	3	Cross zone synchronous replication to over 10Km away	1	15	Ensure that system can run on two out of three zones
								0	

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
Region Connectivity	Address un-resolvable	Delay while DNS times out, slow fallback response	5	DNS configuration error, denial of service attack, or provider failure	1			0	Dual redundant DNS, fallback to local cache, hardcoded IP addresses. Endpoint monitoring and alerts
	Unreachable, request undeliverable	Fast fail, no response	4	Network route down	1			0	Failover to secondary region
	Request undeliverable	Connect timeout, slow fail, no response	4	Router frozen/not accepting connection	1			0	Failover to secondary region
	Request delivered, no response - stall	Application request timeout, slow fail, no response	4	Broken router, overloaded network or slow dependencies	1			0	Failover to secondary region
	Response undeliverable	Application request timeout, slow fail,	4	Network return route failure, dropped packets	1			0	Failover to secondary region

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s)/ Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN	Recommended Action(s)
		no response							
	Response received in time but empty or unintelligible	Fast fail, no response	3	Network response failure	2			0	Failover to secondary region
	Request delivered, response delayed beyond spec	Degraded response arrives too late, slow fallback response	6	Network overloaded dependent services responding slowly	2			0	Failover to secondary region
	Request delivered, degraded response delivered in time	Degraded timely response	2	Service overloaded, dependent services responding slowly	2			0	Alert operators

Operations and Observability FMEA

Misleading and confusing monitoring systems can cause failures to be magnified rather than mitigated. It is imperative that monitoring be centralized to provide overall health of the system. Operations rely upon monitoring to take the corresponding action (based upon runbooks and other scenario-based drill exercises). Failure of monitoring or monitoring systems can exacerbate any problem. Using AWS native monitoring tools

such as CloudWatch and CloudWatch Logs (using custom metrics), and having a monitoring strategy (e.g. agent-based monitoring vs. centralized monitoring systems) can help mitigate the impact.

Table 14

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	SEV	Potential Cause(s) / Mechanism(s) of Failure	PROB	Current Design Controls	DET	RPN
Authentication	Monitoring agent can't authenticate	Can't monitor application	5	Certificate timeout, version mismatch, account not setup, credential changed	3	Log and alert on authentication failures	3	**45**
	Monitoring tool end user operator can't authenticate	Can't monitor system, increased MTTR	5	Certificate timeout, version mismatch, account not setup, credential changed	3	Log and alert on authentication failures	3	**45**
	Slow or unreliable authentication	Errors and delays in observability and alerts	4	Auth service overloaded, high error and retry rate	3	Log and alert on high authentication latency and errors	4	**48**

Notes

[1] http://www.fsb.org/work-of-the-fsb/policy-development/systematically-important-financial-institutions-sifis/

SIFIs are defined by the Financial Stability Board (FSB) as “financial institutions whose distress or disorderly failure, because of their size, complexity and systemic interconnectedness, would cause significant disruption to the wider financial system and economic activity.”

[2] http://www.fsb.org/work-of-the-fsb/policy-development/building-resilience-of-financial-institutions/

[3] https://www.newyorkfed.org/medialibrary/media/banking/circulars/11522.pdf

Additional authorities have introduced their own resiliency policies. Under the auspices of the Federal Financial Institution Examination Council (FFIEC), U.S. banking agencies expect FIs to conduct due diligence and oversee the ability of their third-party providers to provide continuity of service through (a) contractual terms, (b) monitoring mechanisms, and (c) if warranted by the level and criticality of services provided, BCP testing with third parties. The FFIEC BCP guidelines also cover, among other things, data center recovery models (e.g., active-active), capacity planning, and cyber resilience. The FFIEC states that “[i]t is incumbent on financial institutions and third-party service providers to identify and prepare for potentially-significant disruptive events, including those that may have a low probability of occurring but would have a high impact on the institution.”

Additionally, financial institutions across the industry that perform critical economic functions through payments, clearing, and settlement conduct institution-specific, annual tests of their business continuity plans, including both application- and data center-level testing. Industry-wide, approximately 172 organizations (including securities firms, banks, exchanges, and market utilities) also participate in the annual Securities Industry and Financial Markets Association (SIFMA) Industry Business Continuity Test . In the SIFMA Test, participants activate their backup sites to transmit and confirm dummy orders to markets, conduct test payments transactions, and receive and verify market data.

And FIs participate in additional table-top exercises and facilitated discussions, to prepare for different scenarios, including with U.S. regulatory agencies through the U.S. Treasury Department’s Hamilton cybersecurity exercise series.

[4] http://www.fsb.org/what-we-do/policy-development/building-resilience-of-financial-institutions/, https://www.bis.org/cpmi/publ/d146.pdf

[5] https://www.sec.gov/divisions/marketreg/lessonslearned.htm, https://www.occ.treas.gov/news-issuances/bulletins/2003/OCC2003-14a.pdf

[6] http://www.fsb.org/work-of-the-fsb/policy-development/systematically-important-financial-institutions-sifis/

[7] https://www.bis.org/bcbs/basel3.htm

[8] https://www.bis.org/cpmi/publ/d101a.pdf

[9] https://www.bankofengland.co.uk/-/media/boe/files/prudential-regulation/discussion-paper/2018/dp118.pdf

[10] https://aws.amazon.com/architecture/well-architected/

[11] https://aws.amazon.com/compliance/shared-responsibility-model/

[12] https://d1.awsstatic.com/whitepapers/architecture/AWS-Reliability-Pillar.pdf

[13] https://aws.amazon.com/about-aws/global-infrastructure/

[14] https://d1.awsstatic.com/whitepapers/architecture/AWS-Reliability-Pillar.pdf

[15] https://aws.amazon.com/about-aws/global-infrastructure/

[16] https://aws.amazon.com/route53/sla/

[17] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/CopyingAMIs.html

[18] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-copy-snapshot.html

[19] https://docs.aws.amazon.com/AmazonS3/latest/dev/crr.html

[20] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html

[21] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Replication.html

[22] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html

[23] https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GlobalTables.html

[24] https://docs.aws.amazon.com/neptune/latest/userguide/API_CopyDBClusterSnapshot.html

[25] https://docs.aws.amazon.com/redshift/latest/mgmt/working-with-snapshots.html

[26] https://docs.aws.amazon.com/cloudhsm/latest/userguide/copy-backup-to-region.html

[27] https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

[28] https://docs.aws.amazon.com/vpc/latest/peering/what-is-vpc-peering.html

[29] https://docs.aws.amazon.com/aws-technical-content/latest/aws-vpc-connectivity-options/transit-vpc.html

[30] https://aws.amazon.com/answers/networking/aws-global-transit-network/

[31] https://aws.amazon.com/s3/reduced-redundancy/

[32] https://docs.aws.amazon.com/AmazonS3/latest/dev/crr.html

[33] https://docs.aws.amazon.com/efs/latest/ug/gs-step-four-sync-files.html

[34] https://aws.amazon.com/marketplace/pp/B073V2KBXM?qid=1537469325453

[35] https://aws.amazon.com/marketplace/pp/B00K3ALGT6?qid=1537469471064

[36] https://aws.amazon.com/marketplace/pp/B06VV22NYN?qid=1537469471064

[37] https://aws.amazon.com/marketplace/pp/B01LZV5DUJ?qid=1537469619748

[38] https://www.zerto.com/solutions/use-cases/complete-bcdr-solution-business-continuity-disaster-recovery/

[39] https://media.amazonwebservices.com/AWS_Building_Fault_Tolerant_Applications.pdf

[40] https://www.youtube.com/watch?v=FkzW1UCc7B4

[41] https://d1.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf

[42] https://aws.amazon.com/economics/

[43] https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/useconsolidatedbilling-discounts.html

[44] https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/

[45] https://principlesofchaos.org/

[46] https://medium.com/netflix-techblog/project-nimble-region-evacuation-reimagined-d0d0568254d4

[47] https://aws.amazon.com/legal/service-level-agreements/