Fault-Tolerant Components on AWS

November 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 current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2019 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Contents

Abstract

This whitepaper provides an introduction to building fault-tolerant software systems using Amazon Web Services (AWS). You will learn about the diverse array of AWS services at your disposal including compute, storage, networking, and database solutions. By leveraging these solutions, you can set up an infrastructure that refreshes automatically, helping you to avoid degradations and points of failures. The AWS platform can be operated with minimal human interaction and up-front financial investment. In addition, you will learn about the AWS Global Infrastructure, an architecture that provides high availability using AWS Regions and Availability Zones.

This paper is intended for IT managers and system architects looking to deploy or migrate their solutions to the cloud, using a platform that provides highly available, reliable, and fault-tolerant systems.

Introduction

Fault-tolerance is the ability for a system to remain in operation even if some of the components used to build the system fail. Even with very conservative assumptions, a busy e-commerce site may lose thousands of dollars for every minute it is unavailable. This is just one reason why businesses and organizations strive to develop software systems that can survive faults. Amazon Web Services (AWS) provides a platform that is ideally suited for building fault-tolerant software systems. The AWS platform enables you to build fault-tolerant systems that operate with a minimal amount of human interaction and up-front financial investment.

Failures Shouldn't Be THAT Interesting

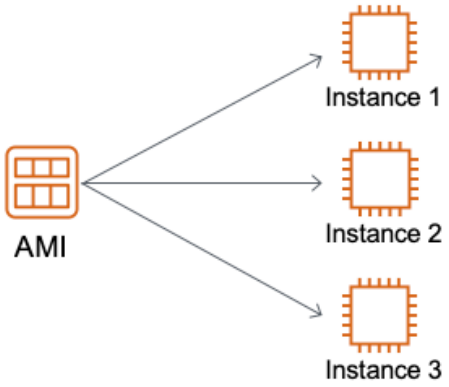
The ideal state in a traditional, on-premises data center environment tends to be one where failure notifications are delivered reliably to a staff of administrators who are ready to take quick and decisive actions in order to solve the problem. Many organizations are able to reach this state of IT nirvana, however, doing so typically requires extensive experience, up-front financial investment, and significant human resources. Amazon Web Services provides services and infrastructure to build reliable, fault-tolerant, and highly available systems in the cloud. As a result, potential failures can be dealt with automatically by the system itself and, as a result, are fairly uninteresting events.

AWS gives you access to a vast amount of IT infrastructure—compute, storage, networking, and databases just to name a few (such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Block Store (Amazon EBS), and Auto Scaling)—that you can allocate automatically (or nearly automatically) to account for almost any kind of failure. You are charged only for resources that you actually use, so there is no up-front financial investment.

Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (Amazon EC2) provides computing resources, literally server instances, that you use to build and host your software systems. Amazon EC2 is a natural entry point to AWS for your application development. You can build a highly reliable and fault-tolerant system using multiple EC2 instances and ancillary services such as Auto Scaling and Elastic Load Balancing.

On the surface, EC2 instances are very similar to traditional hardware servers. EC2 instances use familiar operating systems like Linux or Windows. As such, they can accommodate nearly any kind of software that runs on those operating systems. EC2 instances have IP addresses so the usual methods of interacting with a remote machine (for example, SSH or RDP) can be used. The template that you use to define your service instances is called an Amazon Machine Image (AMI) which contains a defined software configuration (that is, operating system, application server, and applications). From an AMI, you launch an instance, which is a copy of the AMI running as a virtual server in the cloud. You can launch multiple instances of an AMI, as shown in the following figure.



Instance types in Amazon EC2 are essentially hardware archetypes. You choose an instance type that matches the amount of memory (RAM) and computing power (number of CPUs) that you need for your application. Your instances keep running until you stop or terminate them, or until they fail. If an instance fails, you can launch a new one from the AMI.

Amazon publishes many AMIs that contain common software configurations for public use. In addition, members of the AWS developer community have published their own custom AMIs. You can also create your own custom AMI, enabling you to quickly and easily start new instances that contain the software configuration you need.

The first step towards building fault-tolerant applications on AWS is to decide on how the AMIs will be configured. There are two distinct mechanisms to do this, dynamic and static. A dynamic configuration starts with a base AMI and, on launch, deploys the software and data required by the application. A static configuration deploys the required software and data to the base AMI and then uses this to create an application-specific AMI that is used for application deployment. Take the following factors into account when deciding to use either a dynamic or static configuration:

- The frequency of application changes—a dynamic configuration offers greater flexibility for frequent application changes.
- Speed of launch—an application installed on the AMI reduces the time between launch and when the instance becomes available. If this is important then a static configuration minimizes the launch time.
- Audit—when an audit trail of the application configuration is required, then a static configuration combined with a retention policy for AMIs allows past configurations to be recreated.

It is possible to mix dynamic and static configurations. A common pattern is for the application software to be deployed on the AMI while data is deployed once the instance is launched. Your application should be comprised of at least one AMI that you have configured. To start your application, launch the required number of instances from your AMI. For example, if your application is a website or a web service, your AMI could include a web server, the associated static content, and the code for the dynamic pages. As a result, after you launch an instance from this AMI, your web server starts and your application is ready to accept requests.

When the required fleet of instances from the AMI is launched then an instance failure can be addressed by launching a replacement instance that uses the same AMI. This can be done through an API invocation, scriptable command-line tools, or the AWS Management Console. Additionally, an Auto Scaling group can be configured to automatically replace failed or degraded instances. The ability to quickly replace a problematic instance is just the first step towards fault-tolerance. With AWS, an AMI lets you launch a new instance based on the same template, allowing you to quickly recover from failures or problematic behaviors.

To minimize downtime, you have the option to keep a spare instance running, ready to take over in the event of a failure. This can be done efficiently using elastic IP addresses. Failover to a replacement instance or (running) spare instance by remapping your elastic IP address to the new instance.

Elastic Block Store

Amazon Elastic Block Store (Amazon EBS) provides persistent block storage volumes for use with Amazon EC2 instances. EBS volumes can be attached to a running EC2 instance and can persist independently from the instance. EBS volumes are automatically replicated within an Availability Zone, providing high durability and availability, along with protection from component failure.

Amazon EBS is especially suited for applications that require a database, a file system, or access to raw block storage. Typical use cases include big data analytics, relational or NoSQL databases, stream or log processing applications, and data warehousing applications.

Amazon EBS and Amazon EC2 are often used in conjunction with one another when building a fault-tolerant application on the AWS platform. Any application data that needs to be persisted should be stored on EBS volumes. If the EC2 instance fails and needs to be replaced, the EBS volume can simply be attached to the new EC2 instance. Since this new instance is essentially a duplicate of the original, there should be no loss of data or functionality.

Amazon EBS volumes are highly reliable, but to further mitigate the possibility of a failure, backups of these volumes can be created using a feature called snapshots. A robust backup strategy will include an interval (time between backups, generally daily but perhaps more frequently for certain applications), a retention period (dependent on the application and the business requirements for rollback), and a recovery plan. To ensure high durability for backups of EBS volumes, snapshots are stored in Amazon Simple Storage Service (Amazon S3).

EBS snapshots are used to create new Amazon EBS volumes, which are an exact replica of the original volume at the time the snapshot was taken. Because snapshots represent the on-disk state of the application, care must be taken to flush in-memory data to disk before initiating a snapshot.

EBS snapshots are created and managed using the API, AWS Management Console, Amazon Data Lifecycle Manager (DLM), or AWS Backup.

Auto Scaling

An Auto Scaling group contains a collection of Amazon EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management. In the context of a highly available solution, using an Auto Scaling group ensures that an EC2 fleet provides the required capacity. The continuous monitoring of the fleet instance health metrics allows for failures to be automatically detected and for replacement instances to be launched when required.

Where the required size of the EC2 fleet varies Auto Scaling can adjust the capacity using a number of criteria including scheduled and targeted tracking against the value for a specific metric. Multiple scaling criteria can be applied providing a flexible mechanism to manage EC2 capacity.

The requirements of your application and high availability (HA) strategy determines the number of Auto Scaling groups needed. For an application that uses EC2 capacity spread across one or more Availability Zones (AZ) then a single Auto Scaling group suffices. Capacity launches where available and the Auto Scaling group replaces instances as required; but the placement within selected AZs is arbitrary. If the HA strategy requires more precise control of the distribution of EC2 capacity deployments, then using an Auto Scaling group per AZ is the appropriate solution. An example is an application with two instances—production and fail-over—that needs to be deployed in separate Availability Zones. Using two Auto Scaling groups to manage the capacity of each application instance separately ensures that they do not both have capacity in the same Availability Zone.

Failures Can Be Useful

Software systems degrade over time. This is due in part to:

- Software leaking memory and/or resources, including software that you wrote and software that you depend on (such as application frameworks, operating systems, and device drivers).
- File systems fragmenting over time which impacts performance.
- Hardware (particularly storage) devices physically degrading over time.

Disciplined software engineering can mitigate some of these problems, but ultimately, even the most sophisticated software system depends on a number of components that are out of its control (such as the operating system, firmware, and hardware). Eventually, some combination of hardware, system software, and your software will cause a failure and interrupt the availability of your application.

In a traditional IT environment, hardware can be regularly maintained and serviced, but there are practical and financial limits to how aggressively this can be done. However, with Amazon EC2, you can terminate and recreate the resources you need at will.

An application that takes full advantage of the AWS platform can be refreshed periodically with new server instances. This ensures that any potential degradation does not adversely affect your system as a whole. Essentially, you are using what would be considered a failure (such as a server termination) as a forcing function to refresh this resource.

Using this approach, an AWS application is more accurately defined as the service it provides to its clients, rather than the server instance(s) it is comprised of. With this mindset, server instances become immaterial and even disposable.

AWS Global Infrastructure

To build fault-tolerant applications in AWS, it is important to understand the architecture of the AWS Global Infrastructure. The AWS Global Infrastructure is built around Regions and Availability Zones.

AWS Regions and Availability Zones

An AWS Region is a geographical area of the world. Each AWS Region is a collection of data centers that are logically grouped into what we call Availability Zones. AWS Regions provide multiple (typically three) physically separated and isolated Availability Zones which are connected with low latency, high throughput, and highly redundant networking[1]. Each AZ consists of one or more physical data centers. Availability Zones are designed for physical redundancy and provide resilience, enabling uninterrupted performance, even in the event of power outages, Internet downtime, floods, and other natural disasters.

Note: Refer to the Global Infrastructure page for current information about AWS Regions and Availability Zones or our interactive map.

High Availability Through Multiple Availability Zones

Availability Zones are connected to each other with fast, private fiber-optic networking, enabling you to architect applications that automatically fail-over between AZs without interruption. These AZs offer AWS customers an easier and more 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.

[1] Asia Pacific (Osaka) is a Local Region with a single AZ that is available to select AWS customers to provide regional redundancy in Japan.

Building Architectures to Achieve High Availability

You can achieve high availability by deploying your applications to span across multiple Availability Zones. For each application tier (that is, web, application, and database), placing multiple, redundant instances in distinct AZs creates a multi-site solution. Using Elastic Load Balancing (ELB), you get improved fault tolerance as the ELB service automatically balances traffic across multiple instances in multiple Availability Zones, ensuring that only healthy instances receive traffic. The desired goal is to have an independent copy of each application stack in two or more AZs, with automated traffic routing to healthy resources.

Improving Continuity with Replication Between Regions

In addition to replicating applications and data across multiple data centers in the same Region using Availability Zones, you can also choose to increase redundancy and fault tolerance further by replicating data between geographic Regions. You can do so using both private, high speed networking and public internet connections to provide an additional layer of business continuity, or to provide low latency access across the globe.

High Availability Building Blocks

Amazon EC2 and its related services provide a powerful, yet economic platform upon which to deploy and build your applications. However, they are just one aspect of the Amazon Web Services platform. AWS offers a number of other services that can be incorporated into your application development and deployments to increase the availability of your applications.

Elastic IP Addresses

An *Elastic IP address* is a static, public, IPv4 address allocated to your AWS account. With an Elastic IP address, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account. Elastic IPs do not change and remain allocated to your account until you delete them.

An Elastic IP address is allocated from the public AWS IPv4 network ranges in a specific region. If your instance does not have a public IPv4 address, you can associate an Elastic IP address with your instance to enable communication with the internet; for

example, to connect to your instance from your local computer. Elastic IP addresses are mapped via an Internet Gateway to the private address of the instance. Once you associate an Elastic IP address with an instance, it remains associated until you remove the association or associate the address with another resource.

Elastic IP addresses are one method for handling failover, especially for legacy type applications that cannot be scaled horizontally. In the event of a failure of a single server with an associated Elastic IP address, the failover mechanism can re-associate the Elastic IP address to a replacement instance, ideally in an automated fashion. While this scenario may experience downtime for the application, the time may be limited to the time it takes to detect the failure and quickly re-associate the Elastic IP address to the replacement resource.

Where higher availability levels are required, you can use multiple instances and an Elastic Load Balancer.

Elastic Load Balancing

Elastic Load Balancing is an AWS service that automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, and Lambda functions, and ensures only healthy targets receive traffic. It can handle the varying load of your application traffic in a single Availability Zone or across multiple AZs, and supports the ability to load balance across AWS and on-premises resources in the same load balancer.

Elastic Load Balancing offers three types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make your applications fault tolerant.

Application Load Balancer

The Application Load Balancer is best suited for load balancing HTTP and HTTPS traffic and provides advanced request routing targeted at the delivery of modern application architectures, including microservices and containers. Operating at the individual request level (Layer 7), the Application Load Balancer routes traffic to targets within Amazon Virtual Private Cloud (Amazon VPC) based on the content of the request.

Network Load Balancer

Network Load Balancer is best suited for load balancing of Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Transport Layer Security (TLS) traffic where extreme performance is required. Operating at the connection level (Layer 4), Network Load Balancer routes traffic to targets within Amazon VPC and is capable of handling millions of requests per second while maintaining ultra-low latencies. Network Load Balancer is also optimized to handle sudden and volatile traffic patterns.

Benefits of Using Elastic Load Balancing

- **Highly available**—Elastic Load Balancing automatically distributes incoming traffic across multiple targets—Amazon EC2 instances, containers, IP addresses, and Lambda functions—in multiple AZs and ensures only healthy targets receive traffic. The Amazon Elastic Load Balancing Service Level Agreement commitment is 99.99% availability for a load balancer.
- **Secure**—Elastic Load Balancing works with Amazon VPC to provide robust security features, including integrated certificate management, user-authentication, and SSL/TLS decryption. Together, they give you the flexibility to centrally manage TLS settings and offload CPU intensive workloads from your applications.
- **Elastic**—Elastic Load Balancing is capable of handling rapid changes in network traffic patterns. Additionally, deep integration with Auto Scaling ensures sufficient application capacity to meet varying levels of application load without requiring manual intervention.
- **Flexible**—Elastic Load Balancing also allows you to use IP addresses to route requests to application targets. This offers you flexibility in how you virtualize your application targets, allowing you to host more applications on the same instance. This also enables these applications to have individual security groups and use the same network port to further simplify inter-application communication in microservice-based architecture.
- **Robust monitoring & auditing**—Elastic Load Balancing allows you to monitor your applications and their performance in real time with Amazon CloudWatch metrics, logging, and request tracing. This improves visibility into the behavior of your applications, uncovering issues and identifying performance bottlenecks in your application stack at the granularity of an individual request.

- **Hybrid load balancing**—Elastic Load Balancing offers ability to load balance across AWS and on-premises resources using the same load balancer. This makes it easy for you to migrate, burst, or failover on-premises applications to the cloud.

Amazon Simple Queue Service

Amazon Simple Queue Service (Amazon SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. Amazon SQS eliminates the complexity and overhead associated with managing and operating message-oriented middleware, and empowers developers to focus on differentiating work. Using Amazon SQS, you can send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available.

Messages are stored in queues that you create. Each queue is defined as a URL, so it can be accessed by any server that has access to the Internet, subject to the Access Control List (ACL) of that queue. Use Amazon SQS to ensure that your queue is always available; any messages that you send to a queue are retained for up to 14 days.

SQS offers two types of message queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery with best-effort ordering. SQS FIFO queues offer high throughput and are designed to guarantee that messages are processed exactly once, in the exact order that they are sent.

Using Amazon SQS with Other AWS Infrastructure Web Services

Amazon SQS message queuing can be used with other AWS Services such as Amazon Redshift, Amazon DynamoDB, Amazon Relational Database Service (Amazon RDS), Amazon EC2, Amazon Elastic Container Service (Amazon ECS), AWS Lambda, and Amazon S3, to make distributed applications more scalable and reliable. Common design patterns include:

- **Work Queues**—Decouple components of a distributed application that may not all process the same amount of work simultaneously.
- **Buffer and Batch Operations**—Add scalability and reliability to your architecture, and smooth out temporary volume spikes without losing messages or increasing latency.
- **Request Offloading**—Move slow operations off of interactive request paths by enqueuing the request.

- **Fanout**—Combine SQS with Simple Notification Service (SNS) to send identical copies of a message to multiple queues in parallel.
- **Priority**—Use separate queues to provide prioritization of work.
- **Scalability**—Scale up the send or receive rate of messages by adding another process since message queues decouple your processes.
- **Resiliency**—Continue adding messages to the queue even if a process that is reading messages from the queue fails; once the system recovers the queue can be processed since message queues decouple components of your system.

Amazon Simple Storage Service

Amazon S3 is an object storage service that provides highly durable, secure, fault-tolerant data storage. AWS is responsible for maintaining availability and fault-tolerance; you simply pay for the storage that you use. Data is stored as objects within resources called *buckets* and a single object can be up to 5 terabytes in size.

Behind the scenes, Amazon S3 stores objects redundantly on multiple devices across multiple facilities in an AWS Region—so even in the rare case of a failure in an AWS data center, you will still have access to your data. Amazon S3 is designed for 99.999999999% (11 9's) of durability and stores data for millions of applications for companies globally.

Amazon S3 is ideal for any kind of object data storage requirements that your application might have. Amazon S3 can be accessed using the AWS Management Console, by a URL, through a Command Line Interface (CLI), or via API using an SDK with your programming language of choice.

The versioning feature in Amazon S3 allows you to retain prior versions of objects stored in S3 and also protects against accidental deletions initiated by a misbehaving application. Versioning can be enabled for any of your S3 buckets. You can also use either S3 Cross-Region Replication (CRR) to replicate objects in another region or Same-Region Replication (SRR) to replicate objects in the same AWS Region for reduced latency, security, disaster recovery, and other use cases.

In addition to providing highly available storage, Amazon S3 provides multiple storage classes to help reduce storage costs while still providing high availability and durability. Using S3 Lifecycle policies, objects can be transferred to lower cost storage. If you are unsure of your data access patterns, you can select S3 Intelligent-Tiering, which automatically moves your data based on changing access patterns.

By using Amazon S3, you can delegate the responsibility of one critical aspect of fault-tolerance—data storage—to AWS.

Amazon Elastic File System and Amazon FSx for Windows File Server

While Amazon S3 is ideal for applications that can access data as objects, many applications store and access data as files. Amazon Elastic File System (Amazon EFS) and Amazon FSx for Windows File Server (Amazon FSx) are fully-managed AWS services that provide file-based storage for applications.

Amazon EFS provides a simple, scalable, elastic file system for Linux-based workloads. File systems grow and shrink on demand and can scale to petabytes of capacity. Amazon EFS is a regional service storing data within and across multiple Availability Zones for high availability and durability. Applications that need access to shared storage from multiple EC2 instances can store data reliably and securely on Amazon EFS.

Amazon FSx provides a fully managed native Microsoft Windows file system so you can move your Windows-based applications that require file storage to AWS. With Amazon FSx, you can launch highly durable and available Windows file systems that can be accessed from up to thousands of application instances. Amazon FSx is highly available within a single AZ. For applications that require additional levels of availability, Amazon FSx supports the use of Distributed File System (DFS) Replication to enable multi-AZ deployments.

Using either Amazon EFS or Amazon FSx, you can provide highly available, fault-tolerant file storage to your applications running in AWS.

Amazon Relational Database Service

Amazon Relational Database Service guides you in the setup, operation, and scaling a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching, and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security, and compatibility they need.

Amazon RDS is available on several database instance types and is optimized for memory, performance, or I/O. You can choose from six familiar database engines

including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server.

Amazon RDS has many features that enhance reliability for critical production databases, including automated backups, database snapshots, and automatic host replacement.

- **Administration**—Go from project conception to deployment using the Amazon RDS Management Console, the AWS RDS CLI, or API calls to access the capabilities of a production-ready relational database in minutes. No need for infrastructure provisioning or installing and maintaining database software.
- **Scalability**—Scale your database compute and storage resources using the console or an API call, often with no downtime. Many Amazon RDS engine types allow you to launch one or more Read Replicas to offload read traffic from your primary database instance.
- **Availability**—Run on the same highly reliable infrastructure used by other Amazon Web Services. Use Amazon RDS for replication to enhance availability and reliability for production workloads across Availability Zones. Use the Multi-AZ deployment option to run mission-critical workloads with high availability and built-in automated fail-over from your primary database to a synchronously replicated secondary database.
- **Security**—Control network access to your database by running your database instances in an Amazon VPC, which enables you to isolate your database instances and to connect to your existing IT infrastructure through an industry-standard encrypted IPsec VPN. Many Amazon RDS engine types offer encryption at rest and encryption in transit.
- **Cost**—Pay for only the resources you actually consume with no up-front or long-term commitments. You have the flexibility to use on-demand resources or utilize our Reserved Instance pricing to further reduce your costs.

Amazon DynamoDB

Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multi-region, multi-master database with built-in security, backup and restore, and in-memory caching for internet-scale applications.

Amazon DynamoDB is purpose-built for mission-critical workloads. DynamoDB helps secure your data with encryption at rest by default and continuously backs up your data for protection, with guaranteed reliability through a service level agreement. Point-in-time recovery (PITR) helps protect DynamoDB tables from accidental write or delete operations. PITR provides continuous backups of your DynamoDB table data, and you can restore that table to any point in time up to the second during the preceding 35 days.

With Amazon DynamoDB, there are no servers to provision, patch, or manage, and no software to install, maintain, or operate. DynamoDB automatically scales tables to adjust for capacity and maintains performance with zero administration. Availability and fault tolerance are built in, eliminating the need to architect your applications for these capabilities.

Using Serverless Architectures for High Availability

What is Serverless?

Serverless is the native architecture of the cloud that enables you to shift more of your operational responsibilities to AWS, increasing your agility and innovation. Serverless allows you to build and run applications and services without thinking about servers. It eliminates infrastructure management tasks such as server or cluster provisioning, patching, operating system maintenance, and capacity provisioning. Serverless provides built-in availability and fault tolerance. You don't need to architect for these capabilities since the services running the application provide them by default.

Central to many serverless designs is AWS Lambda. AWS Lambda automatically runs your code on highly available, fault-tolerant infrastructure spread across multiple Availability Zones in a single region without requiring you to provision or manage servers. With Lambda, you can run code for virtually any type of application or backend service no administration. Upload your code and Lambda will run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app. AWS Lambda automatically scales your application by running code in response to each trigger. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.

In addition to AWS Lambda, other AWS serverless technologies include:

- AWS Fargate—a serverless compute engine for containers
- Amazon DynamoDB—a fast and flexible NoSQL database
- Amazon Aurora Serverless—a MySQL compatible relational database
- Amazon API Gateway—a service to create, publish, monitor and secure APIs
- Amazon S3—a secure, durable and highly scalable object storage
- Amazon Elastic File System—a simple, scalable, elastic file storage
- Amazon SNS—a fully managed pub/sub messaging service
- Amazon SQS—a fully managed message queuing service

Note: While a full discussion on Serverless capabilities is outside the scope of this paper, you may find additional information about Serverless Computing on our website.

Using Continuous Integration and Continuous Deployment/Delivery to Roll-out Application Changes

What is Continuous Integration?

Continuous integration (CI) is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.

What is Continuous Deployment/Delivery?

Continuous deployment/delivery (CD) is a software development practice where code changes are automatically built, tested, and prepared for production release. It expands on continuous integration by deploying all code changes to a testing environment, a production environment, or both after the build stage has been completed. Continuous delivery can be fully automated with a workflow process or partially automated with manual steps at critical points.

How Does This Help?

Continuous integration and continuous deployment/delivery tools remove the human factor of rolling out application changes and instead add automation as much as possible. Prior to CI/CD tools, scripts were used which required manual intervention or kick off process. Many deployments occurred during the weekends to minimize potential disruptions to the business and could be quickly rolled back if issues arose. Deployment steps were usually documented in runbooks.

AWS CodeBuild, AWS CodePipeline, and AWS CodeDeploy are part of the CI/CD services that DevOps teams use to deploy applications or application changes in their environment. For example, a single pipeline can roll out application changes in one region and, if successful, the same pipeline rolls out the changes in other regions.

With a streamlined CI/CD pipeline, developers can deploy application changes which are transparent to end users. These pipelines can be leveraged to perform multi-region deployments or to quickly deploy a bug fix. If a fault occurs in one environment, users can be redirected to another environment (or region) and updates can be rolled out to the faulty environment. Once the fault has been addressed, you can redirect users back to the original environment.

Utilize Immutable Environment Updates

An *immutable environment* is a type of infrastructure in which resources (that is, servers) are never modified once they have been deployed. Typically, these servers are built from a common image (such as an Amazon Machine Image). The benefit of this type of environment is increased reliability, consistency and a more predictable environment. In AWS, this can be achieved by creating the infrastructure using AWS CloudFormation or AWS Cloud Development Kit (CDK).

Leverage AWS Elastic Beanstalk

With AWS Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control. After uploading your application, Elastic Beanstalk will automatically handle the details of capacity provisioning, load balancing, scaling, and application health monitoring.

Amazon CloudWatch

Amazon CloudWatch is a fully managed monitoring service for AWS resources and the applications that you run on top of them. You can use Amazon CloudWatch to collect and store metrics on a durable platform that is separate and independent from your own infrastructure. You can use these metrics to measure the performance and response times and also capture custom metrics for your applications. These metrics that can be used to do additional actions such as trigger Auto Scaling, Notification, Fan-out, trigger automated tasks, etc.

To capture the custom metrics, you can publish your own metrics to CloudWatch through a simple API request.

Conclusion

Amazon Web Services provides services and infrastructure to build reliable, fault-tolerant, and highly available systems in the cloud. Services that provide basic infrastructure such as Amazon EC2 and Amazon EBS provide specific features, such as availability zones, elastic IP addresses, and snapshots. In particular, Amazon EBS provides durable block storage for applications running on EC2 and an Auto Scaling group ensures your Amazon EC2 fleet operates at the required capacity, automatically detects failures, and replaces instances as needed. Higher-level building blocks such as Amazon S3 provide highly scalable, globally accessible object storage with 11 9s of durability. For durable, fault tolerant file storage for applications running in AWS, you can use Amazon EFS and Amazon FSx for Windows. The wide spectrum of building blocks available give you the flexibility and capability to set up the reliable and highly available environment you need and only pay for the services you consume.

Contributors

Contributors to this document include:

- Jeff Bartley, Solutions Architect, Amazon Web Services
- Lewis Foti, Solutions Architect, Amazon Web Services
- Bert Zahniser, Solutions Architect, Amazon Web Services
- Muhammad Mansoor, Solutions Architect, Amazon Web Services

Further Reading

- Amazon API Gateway
- Amazon Aurora
- Amazon Aurora Serverless
- Amazon CloudWatch
- Amazon DynamoDB
- Amazon Elastic Block Store
- Amazon Elastic Compute Cloud
- Amazon Elastic Container Service
- Amazon Elastic File System
- Amazon FSx for Windows File Server
- Amazon Machine Image
- Amazon Redshift
- Amazon Relational Database Service
- Amazon Simple Notification Service
- Amazon Simple Queue Service
- Amazon Simple Storage Service
- Amazon Virtual Private Cloud
- AWS Auto Scaling
- AWS Cloud Development Kit
- AWS CloudFormation
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline
- AWS Command Line Interface
- AWS Elastic Beanstalk
- AWS Fargate
- AWS Global Infrastructure
- Region and Availability Zones
- AWS Lambda
- AWS Management Console
- Elastic IP Addresses
- Elastic Load Balancing
- Serverless

Document Revisions

Date	Description
November 2019	Refreshed the paper removing outdated references and adding newer AWS services not previously available
October 2011	First publication