Infrastructure Event Readiness

AWS Guidelines and Best Practices

December 2018

Notices

Contents

Abstract

This whitepaper describes guidelines and best practices for customers with production workloads deployed on Amazon Web Services (AWS) who want to design and provision their cloud-based applications to handle planned scaling events, such as product launches or seasonal traffic spikes, gracefully and dynamically. We address general design principles as well as provide specific best practices and guidance across multiple conceptual areas of infrastructure event planning. We then describe operational readiness considerations and practices, and post-event activities.

Introduction

Infrastructure event readiness is about designing and preparing for anticipated and significant events that have an impact on your business. During such events, it is critical that the company web service is reliable, responsive, and highly fault tolerant; under all conditions and changes in traffic patterns. Examples of such events are expansion into new territories, new product or feature launches, seasonal events, or significant business announcements or marketing events.

An infrastructure event that is not properly planned can have a negative impact on your company's business reputation, continuity, or finances. Infrastructure event failures can take the form of unanticipated service failures, load-related performance degradations, network latency, storage capacity limitations, system limits (such as API call rates), finite quantities of available IP addresses, poor understanding of the behaviors of components of an application stack due to insufficient monitoring, unanticipated dependencies on a third-party service or component not set up for scale, or some other unforeseen error condition.

To minimize the risk of unanticipated failures during an important event, companies should invest time and resources to plan and prepare, to train employees, and to design and document relevant processes. The amount of investment in infrastructure event planning for a particular cloud-enabled application or set of applications can vary depending on the system's complexity and global reach. Regardless of the scope or complexity of a company's cloud presence, the design principles and best practices guidance provided in this whitepaper are the same.

With Amazon Web Services (AWS), your company can scale up its infrastructure in preparation for a planned scaling event in a dynamic, adaptable, pay-as-you-go basis. Amazon's rich array of elastic and programmable products and services gives your company access to the same highly secure, reliable, and fast infrastructure that Amazon uses to run its own global network and enables your company to nimbly adapt in response to its own rapidly changing business requirements.

This whitepaper outlines best practices and design principles to guide your infrastructure event planning and execution and how you can use AWS services to ensure that your applications are ready to scale up and scale out as your business needs dictate.

Infrastructure Event Readiness Planning

This section describes what constitutes a planned infrastructure event and the kinds of activities that typically occur during such an event.

What is a Planned Infrastructure Event?

A *planned infrastructure event* is a business-driven, anticipated, and scheduled event window during which it is business critical to maintain a highly responsive, highly scalable, and fault-tolerant web service. Such events can be driven by marketing campaigns, news events related to the company's line of business, product launches, territorial expansion, or any similar activity that results in additional traffic to a company's web-based applications and underlying infrastructure.

What Happens During a Planned Infrastructure Event?

The primary concern in most planned infrastructure events is being able to add capacity to your infrastructure to meet higher traffic demands. In a traditional on-premises environment provisioned with physical compute, storage, and networking resources, a company's IT department provisions additional capacity based on their best estimates of a theoretical maximum peak. This incurs the risk of insufficiently provisioning capacity and the company suffering business loss due to overloaded web servers, slow response times, and other run time errors.

Within the AWS Cloud, infrastructure is programmable and elastic. This means it can be provisioned quickly in response to real-time demand. Additionally, infrastructure can be configured to respond to system metrics in an automated, intelligent, and dynamic fashion—growing or shrinking resources such as web server clusters, provisioned throughput, storage capacity, available compute cores, number of streaming shards, and so on, as needed.

Additionally, many AWS services are fully managed, including storage, database, analytic, application, and deployment services. As a result, AWS customers don't have to worry about the complexities of configuring these services for a high-traffic event. AWS fully managed services are designed for scalability and high availability.

Typically, in preparation for a planned infrastructure event, AWS customers conduct a system review to evaluate their application architecture and operational readiness, considering both scalability and fault tolerance. Traffic estimates are considered and compared to normal business activity performance. Capacity metrics and estimates of required additional capacity are determined. Potential bottlenecks and third-party upstream and downstream dependencies are identified and addressed. Geography is also considered, if the planned event includes an expansion of territory or introduction of new audiences. Expansion into additional AWS Regions or Availability Zones is undertaken in advance of the planned event. A review of the customer's AWS dynamic system settings, such as Auto Scaling, load balancing, geo-routing, high availability, and failover measures is also conducted to ensure these are configured to correctly handle the expected increases in volume and transaction rate. Static settings such as AWS resource limits and location of content delivery network (CDN) origin servers are also considered and modified as needed.

Monitoring and notification mechanisms are reviewed and enhanced as needed to provide real-time transparency into events as they occur and for post-mortem analysis after the planned event has completed.

During the planned event, AWS customers can open support cases with AWS for troubleshooting or real-time support (such as a server going down). Customers who subscribe to the AWS Enterprise Support plan have the additional flexibility to talk with support engineers immediately and to raise critical severity cases if rapid response is required.

After the event, AWS resources are designed to automatically scale down to appropriate levels to match traffic levels, or continue to scale up, as events dictate.

Design Principles

Preparation for planned events starts with a design at the beginning of any implementation of a cloud-based application stack or workload that follows best practices.

Discrete Workloads

A design based on best practices is essential to the effective management of planned-event workloads at both normal and elevated traffic levels. From the start, design discrete and independent functional groupings of resources centered on a specific business application or product. This section describes the multiple dimensions to this design goal.

Tagging

Tags are used to label and organize resources. They are an essential component of managing infrastructure resources during a planned infrastructure event. On AWS, tags are customer-managed, key-value labels applied to an individual managed resource, such as a load balancer or an Amazon Elastic Compute Cloud (Amazon EC2) instance. By referencing well-defined tags that have been attached to AWS resources, you can easily identify which resources within your overall infrastructure comprise your planned event workload. Then, using this information, you can analyze it for preparedness. Tags can also be used for cost-allocation purposes.

Tags can be used to organize, for example, Amazon EC2 instances, Amazon Machine Image (AMI) images, load balancers, security groups, Amazon Relational Database Service (Amazon RDS) resources, Amazon Virtual Private Cloud (Amazon VPC) resources, Amazon Route 53 health checks, and Amazon Simple Storage Service (Amazon S3) buckets.

For more information on effective tagging strategies, refer to AWS Tagging Strategies.[1]

For examples of how to create and manage tags, and put them in Resource Groups, see Resource Groups and Tagging for AWS.[2]

Loose Coupling

When architecting for the cloud, design every component of your application stack to operate as independently as possible from each other. This gives cloud-based workloads the advantage of resiliency and scalability.

You can reduce interdependencies between components in a cloud-based application stack by designing each component as a black box with well-defined interfaces for inputs and outputs (for example, RESTful APIs). If the components aren't applications, but are services that together comprise an application, this is known as a *microservices architecture*. For communication and coordination between application components, you can use event-driven notification mechanisms such as AWS message queues to pass messages between the components, as shown in Figure 1.

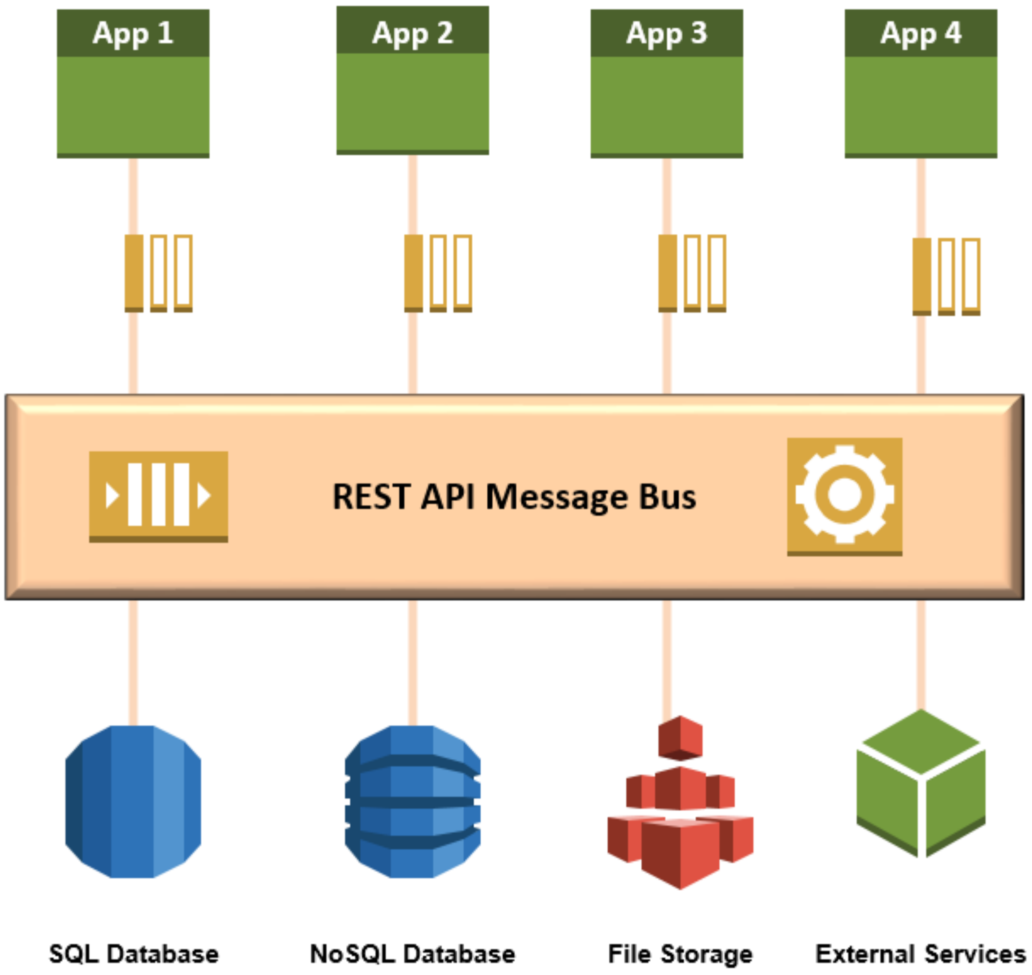


Figure 1. Loose coupling using RESTful interfaces and message queues

Using mechanisms such as that illustrated above, a change or failure in one component has much less chance of cascading to other components. For example, if a server in a multi-tiered application stack becomes unresponsive, applications that are loosely coupled can be designed to bypass the unresponsive tier or switch to degraded mode alternative transactions.

Loosely coupled application components using intermediate message queues can also be designed for asynchronous integration. Because an application's components do not employ direct point-to-point communication but instead use an intermediate and persistent messaging layer (for example, an Amazon Simple Queue Service (SQS) queue or a streaming data mechanism like Amazon Kinesis Streams), they can withstand sudden increases in activity in one component while downstream components process the incoming queue.

If there is a component failure, the messages persist in the queues or streams until the failed component can recover.

For more information on message queueing and notification services offered by AWS, refer to Amazon Simple Queue Service.[3]

Services, Not Servers

Managed services and service endpoints free you from having to worry about security or access, backups or restores, patch management or change control, monitoring or reporting setups, or administration of traditional systems management details. These cloud resources can be provisioned prior to an event for high availability and resilience, using multiple Availability Zone (or, in some cases, multiple Region) configurations. Cloud resources can be scaled up or down, often with no downtime, and you can configure them on the fly through either the AWS Management Console or API/CLI calls.

Managed services and service endpoints can be used to power customer application stacks with capabilities such as relational and NoSQL database systems, data warehousing, event notification, object and file storage, real-time streaming, big data analytics, machine learning, search, transcoding, and many others. An endpoint is a URL that is the entryway for an AWS service. For example, https://dynamodb.us-west-2.amazonaws.com is an entry point for the Amazon DynamoDB service.

By using managed services and their service endpoints, you can leverage the power of production-ready resources as part of your design solution for handling increased volume, reach, and transaction rates during a planned infrastructure event. You don't need to provision and administer your own servers that perform the same functions as managed services.

For more information on AWS service endpoints, see AWS Regions and Endpoints.[4] See also Amazon EMR,[5] Amazon RDS,[6] and Amazon ECS[7] for examples of managed services that have endpoints.

Serverless Architectures

Leverage AWS Lambda as a strategy to effectively respond to dynamically changing processing loads during a planned infrastructure event. Lambda is an event-driven, serverless computing platform. It's a dynamically invoked service that runs Python, Node.js, or Java code in response to events (via notifications) and automatically manages the compute resources specified by that code. Lambda doesn't require provisioning, prior to the event, of Amazon Elastic Compute Cloud (EC2) resources. The Amazon Simple Notification Service (Amazon SNS) can be configured to trigger Lambda functions. See Amazon Simple Notification Service[8] for details.

Lambda serverless functions can execute code that access or invoke other AWS services such as database operations, data transformations, object or file retrieval, or even scaling operations in response to external events or internal system load metrics. AWS Lambda can also generate new notifications or events of its own, and even launch other Lambda functions.

AWS Lambda provides the ability to exercise fine control over scaling operations during a planned infrastructure event. For example, Lambda can be used to extend the functionality of Auto Scaling operations to perform actions such as notifying third-party systems that they also need to scale, or for adding additional network interfaces to new instances as they are provisioned. See Using AWS Lambda with Auto Scaling Lifecycle Hooks[9] for examples of how to use Lambda to customize scaling operations.

For more information on AWS Lambda see What is AWS Lambda?[10]

Automation

Auto Scaling

A critical component of infrastructure event planning is Auto Scaling. Being able to automatically scale an application's capacity up or down according to pre-defined conditions helps to maintain application availability during fluctuations in traffic patterns and volume that occur in a planned infrastructure event.

AWS provides Auto Scaling capability across many of its resources, including EC2 instances, database capacity, containers, etc.

Auto Scaling can be used to scale groupings of instances, such as a fleet of servers that comprise a cloud-based application, so that they scale automatically based on specified criteria. Auto Scaling can also be used to maintain a fixed number of instances even when an instance becomes unhealthy. This automatic scaling and maintaining of the number of instances is the core functionality of the Auto Scaling service.

Auto Scaling maintains the number of instances that you specify by performing periodic health checks on the instances in the group. If an instance becomes unhealthy, the group terminates the unhealthy instance and launches another instance to replace it.

Auto Scaling policies can be used to automatically increase or decrease the number of running EC2 instances in a group of servers to meet changing conditions. When the scaling policy is in effect, the Auto Scaling group adjusts the desired capacity of the group and launches or terminates the instances as needed, either dynamically or alternatively on a schedule, if there is a known and predictable ebb and flow of traffic.

Restarts and Recovery

An important design element in any planned infrastructure event is to have procedures and automation in place to handle compromised instances or servers and to be able to recover or restart them on the fly.

Amazon EC2 instances can be set up to automatically recover when a system status check of the underlying hardware fails. The instance reboots (on new hardware if necessary) but retains its instance ID, IP address, Elastic IP addresses, Amazon Elastic Block Store (EBS) volume attachments, and other configuration details. For more information on auto recovery of EC2 instances, see Auto Recovery of Amazon EC2.[11]

Configuration Management/Orchestration

Integral to a robust, reliable, and responsive planned infrastructure event strategy is the incorporation of configuration management and orchestration tools for individual resource state management and application stack deployment.

Configuration management tools typically handle the provisioning and configuration of server instances, load balancers, Auto Scaling, individual application deployment, and application health monitoring. They also provide the ability to integrate with additional services such as databases, storage volumes, and caching layers.

Orchestration tools, one layer of abstraction above configuration management, provide the means to specify the relationships of these various resources, allowing customers to provision and manage multiple resources as a unified cloud application infrastructure, without worrying about resource dependencies.

Orchestration tools define and describe individual resources as well as their relationships as code. As a result, this code can be version controlled, facilitating the ability to (1) roll back to prior versions or (2) create new branches for testing and development purposes. It is also possible to define orchestrations and configurations optimized for an infrastructure event, and then roll back to the standard configuration following such an event.

Amazon Web Services recommends the following tools to achieve hardware as code deployments and orchestrations:

- **AWS Config with Config Rules** or an AWS Config Partner to provide a detailed, visual, and searchable inventory of AWS resources, configuration history, and resource configuration compliance.
- **AWS CloudFormation** or third-party AWS-resource orchestration tools to manage AWS resource provisioning, update, and termination.
- **AWS OpsWorks, Elastic Beanstalk,** or third-party server configuration management tools to manage operating system (OS) and application configuration changes.

See Infrastructure Configuration Management for more details about ways to manage hardware as code.[12]

Diversity/Resiliency

Remove Single Points of Failure and Bottlenecks

When planning for an infrastructure event, analyze your application stacks for any single points of failure (SPOF) or performance bottlenecks. For example, is there any single instance of a server, data volume, database, NAT gateway, or load balancer that would cause the entire application, or significant portions of it, to stop working if it were to fail?

Secondly, as the cloud-based application scales up in traffic or transaction volume, is there any part of the infrastructure that will encounter a physical limit or constraint, such as network bandwidth, or CPU processing cycles as the volume of data grows along the data flow path?

These risks, once identified, can be mitigated in a variety of ways.

Design for Failure

As mentioned earlier, using loose coupling and message queues with RESTful interfaces is a good strategy for achieving resiliency against individual resource failures or fluctuations in traffic or transaction volume. Another dimension of resilient design is to configure application components to be as stateless as possible.

Stateless applications require no knowledge of prior transactions and have loose dependency on other application components. They store no session information. A stateless application can scale horizontally, as a member of a pool or cluster, since any request can be handled by any instance within the pool or cluster. You can add more resources as needed using Auto Scaling and health check criteria to programmatically handle fluctuating compute, capacity, and throughput requirements. Once an application is designed to be stateless, it could potentially be refactored onto serverless architecture, using Lambda functions in the place of EC2 instances. Lambda functions also have built-in dynamic scaling capability.

In the situation where an application resource such as a web server cannot avoid having state data about transactions, consider designing your applications so that the portions of the application that are stateful are decoupled from the servers themselves. For example, an HTTP cookie or equivalent state data could be stored in a database, such as DynamoDB, or in an S3 bucket or EBS volume.

If you have a complex multistep workflow where there is a need to track the current state of each step in the workflow, Amazon Simple Workflow Service (SWF) can be used to centrally store execution history and make these workloads stateless.

Another resiliency measure is to employ distributed processing. For use cases that require processing large amounts of data in a timely manner where one single compute resource can't meet the need, you can design your workloads so that tasks and data are partitioned into smaller fragments and executed in parallel across a cluster of compute resources. Distributed processing is stateless, since the independent nodes on which the partitioned data and tasks are being processed may fail. In this case, auto-restart of failed tasks on another node of the distributed processing cluster is automatically handled by the distributed processing scheduling engine.

AWS offers a variety of distributed data processing engines such Amazon EMR, Amazon Athena, and Amazon Machine Learning; each of which is a managed service providing endpoints and shielding you from any complexity involving patching, maintenance, scaling, failover, etc.

For real-time processing of streaming data, Amazon Kinesis Streams can partition data into multiple shards that can be processed by multiple consumers of that data, such as Lambda functions or EC2 instances.

For more information on these types of workloads, see Big Data Analytics Options on AWS.[13]

Multi-Zone and Multi-Region

AWS services are hosted in multiple locations worldwide. These locations are composed of Regions and Availability Zones. A Region is a separate geographic area. Each Region has multiple, isolated locations, which are known as Availability Zones. AWS provides customers with the ability to place resources, such as instances, and data in multiple locations.

Design your applications so that they are distributed across multiple Availability Zones and Regions. In conjunction with distributing and replicating resources across Availability Zones and Regions, design your apps using load balancing and failover mechanisms so that your application stacks automatically re-route data flows and traffic to these alternative locations in the event of a failure.

Load Balancing

With the Elastic Load Balancing service (ELB), a fleet of application servers can be attached to a load balancer and yet be distributed across multiple Availability Zones. When the EC2 instances in a particular Availability Zone sitting behind a load balancer fail their health checks, the load balancer stops sending traffic to those nodes. When combined with Auto Scaling, the number of healthy nodes is automatically rebalanced with the other Availability Zones and no manual intervention is required.

It's also possible to have load balancing across Regions by using Amazon Route 53 and latency-based DNS routing algorithms. See Latency Based Routing for more information.[14]

Load Shedding Strategies

The concept of *load shedding* in cloud-based infrastructures consists of redirecting or proxying traffic elsewhere to relieve pressure on the primary systems. In some cases, the load shedding strategy can be a triage exercise, where you choose to drop certain streams of traffic or reduce functionality of your applications to lighten the processing load and to be able to serve at least a subset of the incoming requests.

There are numerous techniques that can be used for load shedding such as caching or latency-based DNS routing. With latency-based DNS routing, the IP addresses of those application servers that are responding with the least latency are returned by the DNS servers in response to name resolution requests. Caching can take place close to the application, using an in-memory caching layer such as Amazon ElastiCache. You can also deploy a caching layer that is closer to the user's edge location, using a global content distribution network such as Amazon CloudFront.

For more information about ElastiCache and CloudFront, see Getting Started with ElastiCache[15] and Amazon CloudFront CDN.[16]

Mitigating Against External Attacks

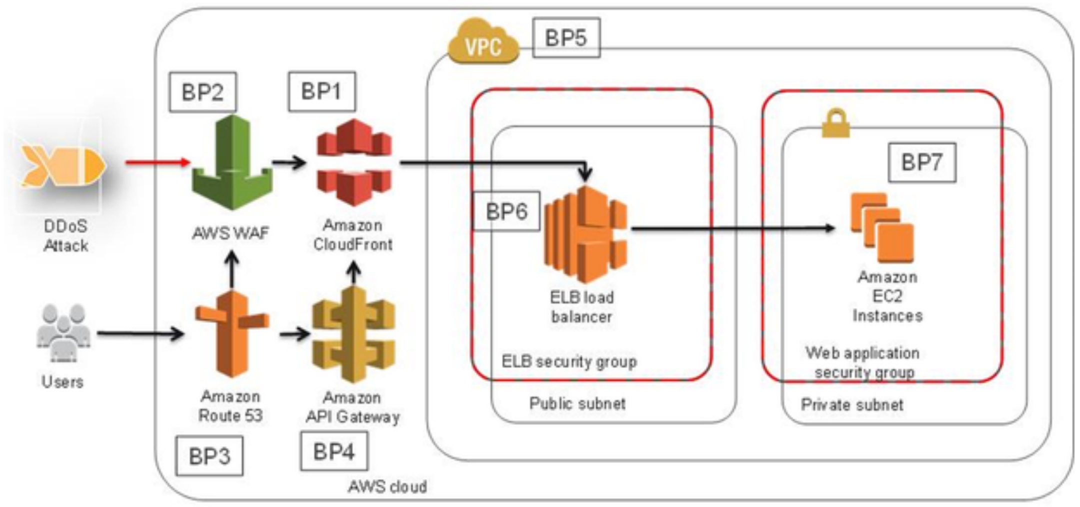
Distributed Denial of Service (DDoS) Attacks

Planned infrastructure events can attract attention which may increase the risk of your application being targeted by a Distributed Denial of Service (DDoS) attack. A DDoS attack is a deliberate attempt to make your application unavailable to users by flooding it with traffic from multiple sources. These attacks include network-layer attacks which aim to saturate the Internet capacity of a network or application, transport-layer attacks which aim to exhaust the connection handling capacity of a device, and application-layer attacks which aim to exhaust the ability of an application to process requests.

There are numerous actions you can take at each of these layers to mitigate against such an attack. For example, you can protect against saturation events by overprovisioning network and server capacity or implementing auto-scaling technologies that are configured to react to attack patterns. You can also make use of purpose-built DDoS mitigation systems such as application firewalls, dynamic load shedding at the edge using Content Distribution Networks (CDNs), network layer threat pattern recognition and filtering, or routing your traffic or requests through a DDoS mitigation provider.

AWS provides automatic DDoS protection as part of the AWS Shield Standard which is included in all AWS services in every AWS Region, at no additional cost. When a network or transport-layer attack is detected it is automatically mitigated at the AWS border, before the traffic is routed to an AWS Region. To make use of this capability it is important to architect your application for DDoS-resiliency.

The optimal DDoS-resiliency is achieved by using services that operate from the AWS Global Edge Network, like Amazon CloudFront and Amazon Route 53, which provides comprehensive protection against all known network and transport-layer attacks. For a reference architecture that includes these services, see Figure 2.



Summary of DDOS Mitigation Best Practices (BP)

	AWS Edge Locations		AWS Regions			
	Amazon CloudFront (BP1) with AWS WAF (BP2)	Amazon Route 53 (BP3)	Elastic Load Balancing (BP6)	Amazon API Gateway (BP4)	Amazon VPC (BP5)	Amazon EC2 with Auto Scaling (BP7)
Layer 3 (for example, UDP reflection) attack mitigation	✓	✓	✓	✓	✓	✓
Layer 4 (for example, SYN flood) attack mitigation	✓	✓	✓	✓		
Layer 6 (for example, TLS) attack mitigation	✓		✓	✓		
Reduce attack surface	✓	✓	✓	✓	✓	

Figure 2: DDoS-resilient reference architecture.

This reference architecture includes several AWS services that can help you improve your web application's resiliency against DDoS attacks.

In addition to architecting for DDoS-resiliency, you can optionally subscribe to AWS Shield Advanced to receive additional features that are useful for monitoring your application, mitigating larger or more complex DDoS attacks, and managing the cost of an attack. With AWS Shield Advanced, you can monitor for DDoS events via the provided APIs and AWS CloudWatch metrics. In case of an attack that causes impact to the availability of your application you can raise a case with AWS Support and, where necessary, receive escalation to the AWS DDoS Response Team (DRT). You also receive AWS WAF for AWS Shield Advanced protected resources and AWS Firewall Manager at no additional cost. If an attack causes an increase in your AWS bill, AWS Shield Advanced allows you to request a limited refund of costs related to the DDoS event. To learn more about using AWS Shield Advanced, see Getting Started with AWS Shield Advanced[17].

Bots and Exploits

To mitigate application-layer attacks consider operating your application at scale and implementing a Web Application Firewall (WAF) which allows you to identify and block unwanted requests. The combination of these techniques can help you mitigate high-volume bots that could otherwise harm the availability of your application and lower-volume bots that could steal content or exploit vulnerabilities. Use these mitigation techniques to significantly reduce the volume of unwanted requests that reach your application and have resilience against unwanted requests that are not blocked.

On AWS, you can implement a WAF from the AWS Marketplace or use AWS WAF which allows you to build your own rules or subscribe to rules managed by Marketplace vendors. With AWS WAF you can use regular rules to block known bad patterns or rate-based rules to temporarily block requests from sources that match conditions you define and exceed a given rate. Deploy these rules using an AWS CloudFormation template. If you have applications distributed across many AWS accounts, deploy and manage AWS WAF rules for your entire organization by using AWS Firewall Manager.

To learn more about deploying preconfigured protections with AWS WAF, see AWS WAF Security Automations[18]. To learn more about rules available from Marketplace vendors, see Managed Rules for AWS WAF[19]. To learn more about managing rules with AWS Firewall Manager, see Getting Started with AWS Firewall Manager[20].

Cost Optimization

Reserved vs Spot vs On-Demand

Controlling the costs of provisioned resources in the cloud is closely tied to the ability to dynamically provision these resources based on systems metrics and other performance and health check criteria. With Auto Scaling, resource utilization can be closely matched to actual processing and storage needs, minimizing wasteful expense and underutilized resources.

Another dimension of cost control in the cloud is being able to choose from the following: On-Demand instances, Reserved Instances (RIs), or Spot Instances. In addition, DynamoDB offers a reservation capacity capability.

With On-Demand instances you pay for only the Amazon EC2 instances you use. On-Demand instances let you pay for compute capacity by the hour with no long-term commitments.

Amazon EC2 Reserved Instances provide a significant discount (up to 75%) compared to On-Demand instance pricing and provide a capacity reservation when used in a specific Availability Zone. Aside from the availability reservation and the billing discount, there is no functional difference between Reserved Instances and On-Demand instances.

Spot Instances allow you to bid on spare Amazon EC2 computing capacity. Spot Instances are often available at a discount compared to On-Demand pricing, which significantly reduces the cost of running your cloud-based applications.

When designing for the cloud, some use cases are better suited for the use of Spot Instances than others. For example, since Spot Instances can be retired at any time once the bid price goes above your bid, you should consider running Spot Instances only for relatively stateless and horizontally scaled application stacks. For stateful applications or expensive processing loads, Reserved Instances or On-Demand instances may be more appropriate. For mission-critical applications where capacity limitations are out of the question, Reserved Instances are the optimal choice.

See Reserved Instances[21] and Spot Instances[22] for more details.

Event Management Process

Planning for an infrastructure event is a group activity involving application developers, administrators, and business stakeholders. Weeks prior to an infrastructure event, establish a cadence of recurring meetings involving the key technical staff who own and operate each of the key infrastructure components of the web service.

Infrastructure Event Schedule

Planning for an infrastructure event should begin several weeks prior to the date of the event. A typical timeline in the planned event lifecycle is shown in Figure 3.

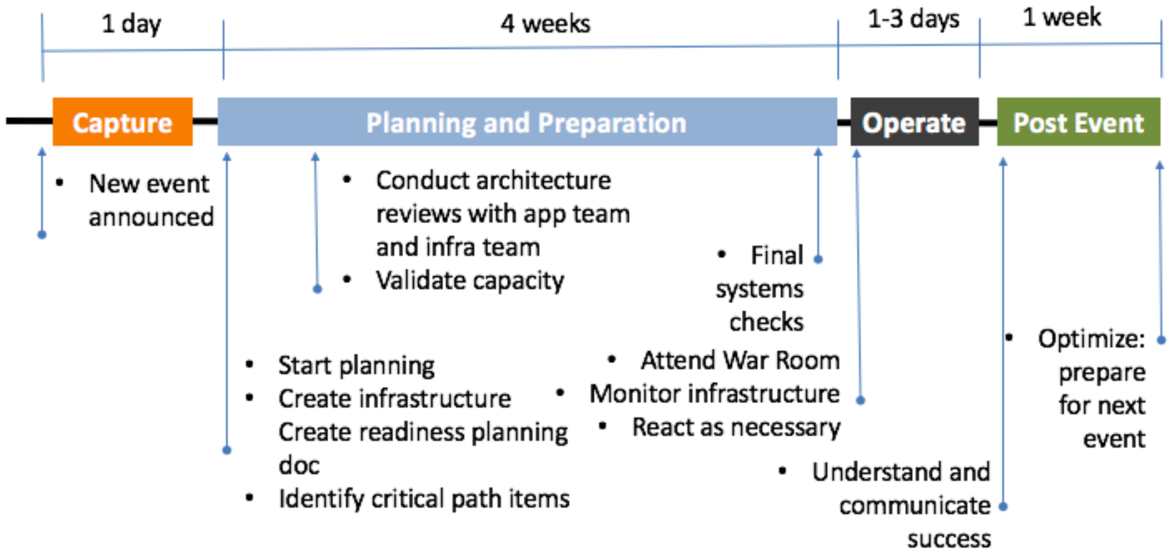


Figure 3. Typical infrastructure event timeline

Planning and Preparation

Schedule

We recommend the following schedule of activities in the weeks leading up to an infrastructure event:

Week 1:

- Nominate a team to drive planning and engineering for the infrastructure event.
- Conduct meetings between stakeholders to understand the parameters of the event (scale, duration, time, geographic reach, affected workloads) and the success criteria.
- Engage any downstream or upstream partners and vendors.

Weeks 2-3:

- Review architecture and adjust as needed.
- Conduct operational review; adjust as needed.
- Follow best practices described in this paper and in footnoted references.
- Identify risks and develop mitigation plans.

- Develop an event runbook.

Week 4:

- Review all cloud vendor services that require scaling based on expected load.
- Check service limits and increase limits as needed.
- Set up monitoring dashboard and alerts on defined thresholds.

Architecture Review

An essential part of your preparation for an infrastructure event is an architectural review of the application stack that will experience the upsurge in traffic. The purpose of the review is to verify and identify potential areas of risk to either the scalability or reliability of the application and to identify opportunities for optimization in advance of the event.

AWS provides its Enterprise Support customers a framework for reviewing customer application stacks that is centered around five design pillars. These are Security, Reliability, Performance Efficiency, Cost Optimization, and Operational Excellence, as described in Table 1.

Table 1: Pillars of well-architected applications

Pillar Name	Pillar Definition	Relevant Area of Interest
Security	The ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies.	Identity Management, Encryption, Monitoring, Logging, Key Management, Dedicated Instances, Compliance, Governance
Reliability	The ability of a system to recover from infrastructure or service failures, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues.	Service Limits, Multiple Availability Zones and Regions, Scalability, Health Check/Monitoring, Backup/Disaster Recovery (DR), Networking, Self-Healing Automation
Performance Efficiency	The ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve.	Right AWS Services, Resource Utilization, Storage Architecture, Caching, Latency Requirements
Cost Optimization	The ability to avoid or eliminate unneeded cost or suboptimal resources.	Spot/Reserved Instances, Environment Tuning, Service Selection, Volume Tuning, Account Management, Consolidated Billing, Decommission Resources

Operational Excellence	The ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.	Runbooks, Playbooks, Continuous Integration/Continuous Deployment (CI/CD), Game Days, Infrastructure as Code, Root Cause Analysis (RCA)s

A detailed checklist of architectural review items, which can be used to review an AWS-based application stack, is available in the Appendix.

Operational Review

In addition to an architectural review, which is more focused on the design components of an application, review your cloud operations and management practices to evaluate how well you are addressing the management of your cloud workloads. The goal of the review is to identify operational gaps and issues and take actions in advance of the event to minimize them.

AWS offers an operational review to its Enterprise Support customers, which can be a valuable tool for preparing for an infrastructure event. The review focuses on assessing the following areas:

- Preparedness–You must have the right mix of organizational structure, processes, and technology. Ensure clear roles and responsibilities are defined for the staff managing your application stack. Define processes in advance to align with the event. Automate procedures where possible.
- Monitoring–Effective monitoring measures how an application is performing. Monitoring is critical to detecting anomalies before they become problems and provides opportunities to minimize impact from adverse events.
- Operations–Operational activities need to be carried out in a timely and reliable way leveraging automation wherever possible, while also dealing with unexpected operational events that require escalations.
- Optimization–Conduct a post-mortem analysis using collected metrics, operational trends, and lessons learned to capture and report opportunities for improvement during future events. Optimization plus preparedness creates a feedback loop to address operational issues and prevent them from reoccurring.

AWS Service Limits

During a planned infrastructure event, it is crucial to avoid exceeding any service limits that may be imposed by a cloud provider while scaling an application or workload.

Cloud services providers typically have limits on the different resources that you can use. Limits are usually imposed on a per-account and per-region basis. The resources affected include instances, volumes, streams, serverless invocations, snapshots, number of VPCs, security rules, and so on. Limits are a safety measure against runaway code or rogue actors attempting to abuse resources and as a control to help minimize billing risk.

Some service limits are raised automatically over time as you expand your footprint in the cloud, though most of these services require that you request limit increases by opening a support case. While some service limits can be increased via support cases, other services have limits that can't be changed.

AWS provides Enterprise and Business support customers with Trusted Advisor, which provides a Limit Check dashboard to allow customers to proactively manage all service limits.

For more information on limits for various AWS services and how to check them, see AWS Service Limits[23] and Trusted Advisor.[24]

Pattern Recognition

Baselines

You should document "back to healthy" values for key metrics prior to the commencement of an infrastructure event. This helps you to determine when an application/service is safely returned to normal levels following the completion/end of the event. For example, identifying that the normal transaction rate through a load balancer is 2,500 requests per second will help determine when it is safe to begin wind down procedures after the event.

Data Flows and Dependencies

Understanding how data flows through the various components of an application helps you identify potential bottlenecks and dependencies. Are the application tiers or components that are consumers of data in a data flow sized appropriately and set up to auto scale correctly if the tiers or components in an application stack that are producers of data scale upwards? In the event of a component failure, can data be queued until that component recovers? Are any downstream or upstream data providers or consumers scalable in response to your event?

Proportionality

Review the proportionality of scaling required by the various components of an application stack when preparing for an infrastructure event. This proportionality is not always one-to-one. For example, a ten-fold increase in transactions per second across a load balancer might require a twenty-fold increase in storage capacity, number of streaming shards, or number of database read and write operations; due to processing that might be taking place in the front-facing application.

Communications Plan

Prior to the event, develop a communications plan. Gather a list of internal stakeholders and support groups and identify who should be contacted at various stages of the event in various scenarios, such as beginning of the event, during the event, end of the event, post-event analysis, emergency contacts, contacts during troubleshooting situations, etc.

Persons and groups to be contacted may include the following:

- Stakeholders
- Operations managers
- Developers
- Support teams
- Cloud service provider teams
- Network operations center (NOC) team

As you gather a list of internal contacts you should also develop a contact list of external stakeholders involved with the continuous live delivery of the application. These stakeholders include partners and vendors supporting key components of the stack, downstream and upstream vendors providing external services, data feeds, authentication services, and so on.

This external contact list should also include the following:

- Infrastructure hosting vendors
- Telecommunications vendors
- Live data streaming partners
- PR marketing contacts
- Advertising partners
- Technical consultants involved with service engineering

Ask for the following information from each provider:

- Live points of contact during time of event
- Critical support contact and escalation process
- Name, telephone number, and email address
- Verification that live technical contacts will be available

AWS customers subscribed to Enterprise Support also have Technical Account Managers (TAMs) assigned to their account who can coordinate and verify that dedicated AWS support staff is aware of and prepared for support of the event. TAMs are also on call during the event, present in the war room, and available to drive support escalations if they are needed.

NOC Preparation

Prior to the event, instruct your operations and/or developer team to create a live metrics dashboard that monitors each critical component of the web service in production as the event occurs. Ideally, the dashboard should automatically present updated metrics every minute or at an interval that is suitable and effective during the event.

Consider monitoring the following components:

- Resource utilization of each server (CPU, disk, and memory utilization)
- Web service response time
- Web traffic metrics (users, page views, sessions)

- Web traffic per visitor region (global customer segments)
- Database server utilization
- Marketing flow conversion funnels, such as conversion rates and fallout percentage
- Application error logs
- Heartbeat monitoring

Amazon CloudWatch provides a means to gather most of these metrics from AWS resources into a single pane of glass using CloudWatch custom dashboards. Additionally, CloudWatch offers the capability to import custom metrics into CloudWatch wherever AWS isn't already providing that metric automatically. See the Monitor section for more details on AWS monitoring tools and capabilities.

Runbook Preparation

You should develop a runbook in preparation for the infrastructure event. A *runbook* is an operational manual containing a compilation of procedures and operations that your operators will carry out during the event. Event runbooks can be outgrowths of existing runbooks used for routine operations and exception handling. Typically, a runbook contains procedures to begin, stop, supervise, and debug a system. It should also describe procedures for handling unexpected events and contingencies.

A runbook should include the following sections:

- **Event details**: Briefly describes of the event, success criteria, media coverage, event dates, and contact details of the main stakeholders from the customer side and AWS.
- **List of AWS services**: Enumerates all AWS services to be used during the event. Also, the expected load on these services, Regions affected, and account IDs.
- **Architecture and application review**: Documents load testing results, any stress points in the infrastructure and application design, resiliency measures for the workload, single points of failure, and potential bottlenecks.

- **Operational review**: Highlights monitoring setup, health criteria, notification mechanisms, and service restoration procedures.
- **Preparedness checklist**: Includes such considerations as service limits checks, pre-warming of application stack components such as load balancers, pre-provisioning of resources such as stream shards, DynamoDB partitions, S3 partitions, and so on. For more information, see the Architecture Review Detailed Checklist in the Appendix.

Monitor

Monitoring Plan

Database, application, and operating system monitoring is crucial to ensure a successful event. Set up comprehensive monitoring systems to effectively detect and respond immediately to serious incidents during the infrastructure event. Incorporate both AWS and customer monitoring data. Ensure that monitoring tools are instrumented at the appropriate level for an application based on its business criticality. Implementing a monitoring plan that collectively gathers monitoring data from all of your AWS solution segments will help in debugging a complex failure if it occurs.

The monitoring plan should address the following questions:

- What monitoring tools and dashboards must be set up for the event?
- What are the monitoring objectives and the allowed thresholds? What events will trigger actions?
- What resources and metrics from these resources will be monitored and how often must they be polled?
- Who will perform the monitoring tasks? What monitoring alerts are in place? Who will be alerted?
- What remediation plans have been set up for common and expected failures? What about unexpected events?
- What is the escalation process in the case of operational failure of any critical systems components?

The following AWS monitoring tools can be used as part of your plan:

- **Amazon CloudWatch**: Provided as an out-of-the-box solution for AWS dashboard metrics, monitoring, alerting, and automated provisioning.

- **Amazon CloudWatch custom metrics**: Used for operating systems, application, and business metrics collection. The Amazon CloudWatch API allows for the collection of virtually any type of custom metric.
- **Amazon EC2 instance health**: Used for viewing status checks and for scheduling events for your instances based on their status, such as auto-rebooting or restarting an instance.
- **Amazon SNS**: Used for setting up, operating, and sending event-driven notifications.
- **AWS X-Ray**: Used to debug and analyze distributed applications and microservices architecture by analyzing data flows across system components.
- **Amazon Elasticsearch Service**: Used for centralized log collection and real-time log analysis. For rapid, heuristic detection of problems.
- **Third-party tools:** Used for a real-time analytics and full stack monitoring and visibility.
- **Standard operating system monitoring tools**: Used for OS-level monitoring.

For more details about AWS monitoring tools, see Automated and Manual Monitoring.[25] See also Using Amazon CloudWatch Dashboards[26] and Publish Custom Metrics.[27]

Notifications

A crucial operational element in your design for infrastructure events is the configuration of alarms and notifications to integrate with your monitoring solutions. These alarms and notifications can be used with services such as AWS Lambda to trigger actions based on the alert. Automating responses to operational events is a key element to enabling mitigation, rollback, and recovery with maximum responsiveness.

Tools should also be in place to centrally monitor workloads and create appropriate alerts and notifications based on available logs and metrics that relate to key operational indicators. This includes alerts and notifications for out-of-bound anomalies, as well as service or component failures. Ideally, when low-performance thresholds are crossed or failures occur, the system has been architected to automatically self-heal or scale in response to such notifications and alerts.

As previously noted, AWS offers services (Amazon Simple Queue Service (SQS) and Amazon SNS) to ensure appropriate alerting and notification in response to unplanned operational events, as well as for enabling automated responses.

Operational Readiness (Day of Event)

Plan Execution

On the day of the event, the core team involved with the infrastructure event should be on a conference call monitoring real-time dashboards. Runbooks should be fully developed and available. Make sure that the communications plan is well defined and known to all support staff and stakeholders, and that a contingency plan is in place.

War Room

During the event, have an open conference bridge with the following participants:

- The responsible application and operations teams
- Operations team leadership
- Technical support resources from external partners directly involved with technical delivery
- Business stakeholders

Throughout most of the event the conversation of this conference bridge should be minimal. If an adverse operational event arises, the key people who can respond to the event will already be on this bridge ready to act and consult.

Leadership Reporting

During the event, send an email hourly to key leadership stakeholders. This update should include the following:

- Status summary: Green (on track), Yellow (issues encountered), Red (major issue)
- Key metrics update

- Issues encountered, status of remedy plan, and the estimated time to resolution (ETA)
- Phone number of the war room conference bridge (so stakeholders may join, if needed)

At the conclusion of the event, a summary email should be sent that follows the following format:

- Overall event summary with synopsis of issues encountered
- Final metrics
- Updated remedy plan that details the issues and resolutions
- Key points-of-contact for any follow-ups that stakeholders may have.

Contingency Plan

Each step in the event's preparation process should have a corresponding contingency action that has been verified in a test environment.

Address the following questions as you put together a contingency plan:

- What are the worst-case scenarios that can occur during the event?
- What types of events would cause a negative public relations impact?
- Which third-party components and services might fail during the event?
- Which metrics should be monitored that would indicate that a worst-case scenario is occurring?
- What is the rollback plan for each identified worst-case scenario?
- How long will each rollback process take? What is the acceptable Recovery Point Objective (RPO) and Recovery Time Objective (RTO)? (See Using AWS for Disaster Recovery[28] for additional information on these concepts.)

Consider the following types of contingency plans:

- **Blue/Green Deployment**: If rolling out a new production app or environment, keep the prior production build online and available (in case a switch back is needed).
- **Warm Pilot**: Launch a minimal environment in a second Region that can quickly scale up if needed. If a failure occurs in the primary Region, scale up the second Region and switch traffic over to it.
- **Maintenance Mode Error Pages**: Check any pre-configured error pages and triggers at each layer of your web service. Be prepared to inject a more specific message into these error pages if any operational failures of any of these layers occurs.

Test the contingency plan for each documented worst-case scenario.

Post-Event Activities

Post-Mortem Analysis

We recommend a post-mortem analysis as part of an infrastructure event management lifecycle. Post mortems allow you to collaborate with each team involved and identify areas that might need further optimization, such as operational procedures, implementation details, failover and recovery procedures, etc. This is especially relevant if an application stack encountered disruptions during the event and a root cause analysis (RCA) is needed. A post-mortem analysis helps provide data points and other essential information needed in an RCA document.

Wind-Down Process

Immediately following the conclusion of the infrastructure event, the wind-down process should begin. During this period, monitor relevant applications and services to ensure traffic has reverted back to normal production levels. Use the health dashboards created during the event's preparation phase to verify the normalization of traffic and transaction rates. Wind-down periods for some events may be linear and straightforward, while others may experience uneven or more gradual reductions in volume. Some traffic patterns from the event may persist. For example, recovering from a surge in traffic generally requires straightforward wind-down procedures, whereas an application deployment or expansion into a new geographical Region may have long-lasting effects requiring you to carefully monitor new traffic patterns as part of the permanent application stack.

At some point following the completion of the event, you must determine when it is safe to end event management operations. Refer to the previously documented "normal" values for key metrics to help determine when to declare that an event is completed or ended. We recommend splitting wind-down activities into two branches, which could have different timelines. Focus the first branch on operational management of the event, such as sending communications to internal and external stakeholders and partners, and the resetting of service limits. Focus the second branch on technical aspects of the wind-down such as scale-down procedures, validation of the health of the environment, and criteria for determining whether architectural changes should be reverted or committed.

The timeline associated with each of those branches can vary depending on the nature of the event, key metrics, and customer comfort. We've outlined some common tasks associated with each branch in Tables 2 and 3 to help you determine the appropriate time-to-end management for an event.

Table 2: First branch: operational wind-down tasks

Task	Description
Communications	Notification to internal and external stakeholders that the event has ended. The time-to-end communication should be aligned with the definition of the completion of the event. Use "back to healthy" metrics to determine when it is appropriate to end communication. Alternatively, you can end communication in tiers. For example, you could end the war room bridge but leave the event escalation procedures intact in case of post-event failures.
Service Limits/Cost Containment	Although it may be tempting to retain an elevated service limit after an event, keep in mind that service limits are also used as a safety net. Service limits protect you and your costs by preventing excess service usage, be that a compromised account or misconfigured automation.
Reporting and Analysis	Data collection and collation of event metrics, accompanied by analytical narratives showing patterns, trends, problem areas, successful procedures, ad-hoc procedures, timeline of event, and whether or not success criteria were met should be developed and distributed to all internal parties identified in the communications plan. A detailed cost analysis should also be developed, to show the operational expense of supporting the event.
Optimization Tasks	Enterprise organizations evolve over time as they continue to improve their operations. Operational optimization requires the constant collection of metrics, operational trends, and lessons learned from events to uncover opportunities for improvement. Optimization ties back with preparation to form a feedback loop to address operational issues and prevent them from reoccurring.

Table 3: Second branch: technical wind-down tasks

Task	Description
Service Limits/Cost Containment	Although it may be tempting to retain elevated service limits after an event, keep in mind that service limits also serve the purpose of being a safety net. Service limits protect your operations and operating costs by preventing excess service usage, either through malicious activity stemming from a compromised account or through misconfigured automation.
Scale Down Procedures	Revert resources that were scaled up during the preparation phase. These items are unique to your architecture but the following examples are common: • EC2/RDS instance size • Auto Scaling configuration • Reserved capacity • Provisioned Input/Output Operations Per Second (PIOPS)
Validation of Health of Environment	Compare to baseline metrics and review production health to verify that after the event and after scale-down procedures have been completed, the systems affected are reporting normal behavior.
Disposition of Architectural Changes	Some changes made in preparation for the event may be worth keeping, depending on the nature of the event and observation of operational metrics. For example, expansion into a new geographical Region might require a permanent increase of resources in that Region, or raising certain service limits or configuration parameters, such as number of partitions in a DB or shards in a stream of PIOPS in a volume, might be a performance tuning measure that should be persisted.

Optimize

Perhaps the most important component of infrastructure event management is the post-event analysis and the identification of operational and architectural challenges observed and opportunities for improvement. Infrastructure events are rarely one-time events. They might be seasonal or coincide with new releases of an application, or they might be part of the growth of the company as it expands into new markets and territories. Thus, every infrastructure event is an opportunity to observe, improve, and prepare more effectively for the next one.

Conclusion

AWS provides building blocks in the form of elastic and programmable products and services that your company can assemble to support virtually any scale of workload. With AWS infrastructure event guidelines and best practices, coupled with our complete set of highly available services, your company can design and prepare for major business events and ensure that scaling demands can be met smoothly and dynamically, ensuring fast response and global reach.

Contributors

The following individuals and organizations contributed to this document:

- Presley Acuna, AWS Enterprise Support Manager
- Kurt Gray, AWS Global Solutions Architect
- Michael Bozek, AWS Sr. Technical Account Manager
- Rovan Omar, AWS Technical Account Manager
- Will Badr, AWS Technical Account Manager
- Eric Blankenship, AWS Sr. Technical Account Manager
- Greg Bur, AWS Technical Account Manager
- Bill Hesse, AWS Sr. Technical Account Manager
- Hasan Khan, AWS Sr. Technical Account Manager
- Varun Bakshi, AWS Sr. Technical Account Manager
- Fatima Ahmed, AWS Specialist Technical Account Manager (Security)
- Jeffrey Lyon, AWS Manager, DDoS Ops Engineering

Further Reading

For additional reading on operational and architectural best practices, see Operational Checklists for AWS.[29] We recommend that readers review AWS Well Architected Framework[30] for a structured approach to evaluating their cloud based application delivery stacks. AWS offers Infrastructure Event Management (IEM) as a premium support offering for customers desiring more direct involvement of AWS Technical Account Manager and Support Engineers in their design, planning and day of event operations. For more details about the AWS IEM premium support offering, please see Infrastructure Event Management.[31]

Appendix

Detailed Architecture Review Checklist

Yes-No-N/A	Security
Y—N—N/A	We rotate our AWS Identity and Access Management (IAM) access keys and user password and the credentials for the resources involved in our application at most every 3 months as per AWS security best practices. We apply password policy in every account, and we use hardware or virtual multifactor authentication (MFA) devices.
Y—N—N/A	We have internal security processes and controls for controlling unique, role-based, least privilege access to AWS APIs leveraging IAM.
Y—N—N/A	We have removed any confidential or sensitive information including embedded public/private instance key pairs and have reviewed all SSH authorized keys files from any customized Amazon Machine Images (AMIs).
Y—N—N/A	We use IAM roles for EC2 instances as convenient instead of embedding any credentials inside AMIs.
Y—N—N/A	We segregate IAM administrative privileges from regular user privileges by creating an IAM administrative role and restricting IAM actions from other functional roles.
Y—N—N/A	We apply the latest security patches on our EC2 instances for either Windows or Linux instances. We use operating system access controls including Amazon EC2 Security Group rules, VPC network access control lists, OS hardening, host-based firewall, intrusion detection/prevention, monitoring software configuration and host inventory.
Y—N—N/A	We ensure that the network connectivity to and from the organization's AWS and corporate environments uses a transport of encryption protocols.
Y—N—N/A	We apply a centralized log and audit management solution to identify and analyze any unusual access patterns or any malicious attacks on the environment.
Y—N—N/A	We have Security event and incident management, correlation, and reporting processes in place.
Y—N—N/A	We make sure that there isn't unrestricted access to AWS resources in any of our security groups.
Y—N—N/A	We use a secure protocol (HTTPS or SSL), up-to-date security policies, and cipher protocols for a front-end connection (client to load balancer). The requests are encrypted between the clients and the load balancer, which is more secure.
Y—N—N/A	We configure our Amazon Route 53 MX resource record set to have a TXT resource record set that contains a corresponding Sender Policy Framework (SPF) value to specify the servers that are authorized to send email for our domain.
Y—N—N/A	We architect our application for DDoS resiliency by using services that operate from the AWS Global Edge Network, like Amazon CloudFront and Amazon Route 53, as well as additional AWS services that mitigate against Layer 3 through 6 attacks (see Summary of DDoS Mitigation Best Practices in the Appendix).

Yes-No-N/A	Reliability
Y—N—N/A	We deploy our application on a fleet of EC2 instances that are deployed into an Auto Scaling group to ensure automatic horizontal scaling based on a pre-defined scaling plans. Learn more.
Y—N—N/A	We use an Elastic Load Balancing health check in our Auto Scaling group configuration to ensure that the Auto Scaling group acts on the health of the underlying EC2 instances. (Applicable only if you use load balancers in Auto Scaling groups.)
Y—N—N/A	We deploy critical components of our applications across multiple Availability Zones, are appropriately replicating data between zones. We test how failure within these components affects application availability using Elastic Load Balancing, Amazon Route 53, or any appropriate third-party tool.
Y—N—N/A	In the database layer we deploy our Amazon RDS instances in multiple Availability Zones to enhance database availability by synchronously replicating to a standby instance in a different Availability Zone.
Y—N—N/A	We define processes for either automatic or manual failover in case of any outage or performance degradation.
Y—N—N/A	We use CNAME records to map our DNS name to our services. We DON'T use A records.
Y—N—N/A	We configure a lower time-to-live (TTL) value for our Amazon Route 53 record set. This avoids delays when DNS resolvers request updated DNS records when rerouting traffic. (For example, this can occur when DNS failover detects and responds to a failure of one of your endpoints.)
Y—N—N/A	We have at least two VPN tunnels configured to provide redundancy in case of outage or planned maintenance of the devices at the AWS endpoint.
Y—N—N/A	We use AWS Direct Connect and have two Direct Connect connections configured at all times to provide redundancy in case a device is unavailable. The connections are provisioned at different Direct Connect locations to provide redundancy in case a location is unavailable. We configure the connectivity to our virtual private gateway to have multiple virtual interfaces configured across multiple Direct Connect connections and locations.
Y—N—N/A	We use Windows instances and ensure that we are using the latest paravirtual (PV) drivers. PV driver helps optimize driver performance and minimize runtime issues and security risks. We ensure that EC2Config agent is running the latest version on our Windows instance.
Y—N—N/A	We take snapshots of our Amazon Elastic Block Store (EBS) volumes to ensure a point-in-time recovery in case of failure.
Y—N—N/A	We use separate Amazon EBS volumes for the operating system and application/database data where appropriate.
Y—N—N/A	We apply the latest kernel, software and drivers patches on any Linux instances.

Yes-No-N/A	Performance Efficiency
Y—N—N/A	We fully test our AWS-hosted application components, including performance testing, prior to going live. We also perform load testing to ensure that we have used the right EC2 instance size, number of IOPS, RDS DB instance size, etc.
Y—N—N/A	We run a usage check report against our services limits and make sure that the current usage across AWS services is at or less than 80% of the service limits. Learn more
Y—N—N/A	We use Content Delivery/Distribution Network (CDN) to utilize caching for our application (Amazon CloudFront) and as a way to optimize the delivery of the content and the automatic distribution of the content to the nearest edge location to the user.
Y—N—N/A	We understand that some dynamic HTTP request headers that Amazon CloudFront receives (User-Agent, Date, etc.) can impact the performance by reducing the cache hit ratio and increasing the load on the origin. Learn more
Y—N—N/A	We ensure that the maximum throughput of an EC2 instance is greater than the aggregate maximum throughput of the attached EBS volumes. We also use EBS-optimized instances with PIOPS EBS volumes to get the expected performance out of the volumes.
Y—N—N/A	We ensure that the solution design doesn't have a bottleneck in the infrastructure or a stress point in the database or the application design.
Y—N—N/A	We deploy monitoring on application resources and configure alarms based on any performance breaches using Amazon CloudWatch or third-party partner tools.
Y—N—N/A	In our designs, we avoid using a large number of rules in security group(s) attached to our application instances. A large number of rules in a security group may degrade performance.

Yes-No-N/A	Cost Optimization
Y—N—N/A	We note whether the infrastructure event may involve over-provisioned capacity that needs to be cleaned up after the event to avoid unnecessary cost.
Y—N—N/A	We use right sizing for all of our infrastructure components including EC2 instance size, RDS DB instance size, caching cluster nodes size and numbers, Redshift Cluster nodes size and numbers, and EBS volume size.
Y—N—N/A	We use Spot Instances when it's convenient. Spot Instances are ideal for workloads that have flexible start and end times. Typical use cases for Spot instances are: batch processing, report generation, and high-performance computing workloads.
Y—N—N/A	We have predictable application capacity minimum requirements and take advantage of Reserved Instances. Reserved Instances allow you to reserve Amazon EC2 computing capacity in exchange for a significantly discounted hourly rate compared to On Demand instance pricing.

Notes

[1] https://aws.amazon.com/answers/account-management/aws-tagging-strategies/

[2] https://aws.amazon.com/blogs/aws/resource-groups-and-tagging/

[3] https://aws.amazon.com/sqs/

[4] http://docs.aws.amazon.com/general/latest/gr/rande.html

[5] https://aws.amazon.com/emr/

[6] https://aws.amazon.com/rds/

[7] https://aws.amazon.com/ecs/

[8] https://aws.amazon.com/sns/

[9] https://aws.amazon.com/blogs/compute/using-aws-lambda-with-auto-scaling-lifecycle-hooks/

[10] http://docs.aws.amazon.com/lambda/latest/dg/welcome.html

[11] https://aws.amazon.com/blogs/aws/new-auto-recovery-for-amazon-ec2/

[12] https://aws.amazon.com/answers/configuration-management/aws-infrastructure-configuration-management/

[13] https://d0.awsstatic.com/whitepapers/Big_Data_Analytics_Options_on_AWS%20.pdf

[14] http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html#routing-policy-latency

[15] https://aws.amazon.com/elasticache/

[16] https://aws.amazon.com/cloudfront/

[17] https://docs.aws.amazon.com/waf/latest/developerguide/getting-started-ddos.html

[18] https://aws.amazon.com/answers/security/aws-waf-security-automations/

[19] https://aws.amazon.com/mp/security/WAFManagedRules/

[20] https://docs.aws.amazon.com/waf/latest/developerguide/getting-started-fms.html

[21] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts-on-demand-reserved-instances.html

[22] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html

[23] https://docs.aws.amazon.com/general/latest/gr/aws_service_limits.html

[24] https://aws.amazon.com/about-aws/whats-new/2014/07/31/aws-trusted-advisor-security-and-service-limits-checks-now-free/

[25] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring_automated_manual.html

[26] http://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch_Dashboards.html

[27] http://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/publishingMetrics.html

[28] https://aws.amazon.com/blogs/aws/new-whitepaper-use-aws-for-disaster-recovery/

[29] http://media.amazonwebservices.com/AWS_Operational_Checklists.pdf

[30] http://d0.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf

[31] https://aws.amazon.com/premiumsupport/iem/