AWS Well-Architected Framework

October 2015

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. 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.

Contents

Abstract

This paper describes the **AWS Well-Architected Framework**, which enables customers to assess and improve their cloud-based architectures and better understand the business impact of their design decisions. We address general design principles as well as specific best practices and guidance in four conceptual areas that we define as the *pillars* of the Well-Architected Framework.

Introduction

At Amazon Web Services (AWS) we understand the value in educating our customers on architectural best practices for designing reliable, secure, efficient, and cost-effective systems in the cloud. As part of this effort, we developed the AWS Well-Architected Framework, which helps you to understand the pros and cons of decisions you make while building systems on AWS. We believe that having well-architected systems greatly increases the likelihood of business success.

AWS Solutions Architects have years of experience architecting solutions across a wide variety of business verticals and use cases, and we have helped design and review thousands of customers' architectures on AWS. From this, we have identified best practices and core strategies for architecting systems in the cloud. The AWS Well-Architected Framework documents a set of foundational questions that allow you to understand if a specific architecture aligns well with cloud best practices. The framework provides a consistent approach to evaluating systems against the qualities you expect from modern cloud-based systems, and the remediation that would be required to achieve those qualities. As the AWS platform continues to evolve, and we continue to learn more from working with our customers, we will continue to refine the definition of well-architected.

This paper is intended for those in technology roles, such as chief technology officers (CTOs), architects, developers, and operations team members. After reading this paper, you will understand AWS best practices and strategies to use when designing a cloud architecture. This paper does not provide implementation details or architectural patterns; however, it does include references to appropriate resources for this information.

Definition of the AWS Well-Architected Framework

Every day experts at AWS assist customers in architecting systems to take advantage of best practices in the cloud. We work with you on making architectural trade-offs as your designs evolve. As you deploy these systems into

live environments, we learn how well these systems perform, and the consequences of those trade-offs.

Based on what we have learned we have created the AWS Well-Architected Framework, which is a set of questions you can use to evaluate how well an architecture is aligned to AWS best practices.

The AWS Well-Architected Framework is based on four pillars—security, reliability, performance efficiency, and cost optimization, which we define as follows:

Pillar Name	Description
Security	The ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies.
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.
Performance Efficiency	The ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve.
Cost Optimization	The ability to avoid or eliminate unneeded cost or suboptimal resources.

General Design Principles

The Well-Architected Framework identifies a set of general design principles to facilitate good design in the cloud:

- **Stop guessing your capacity needs:** Eliminate guessing your infrastructure capacity needs. When you make a capacity decision before you deploy a system, you might end up sitting on expensive idle resources or dealing with the performance implications of limited capacity. With cloud computing, these problems can go away. You can use as much or as little capacity as you need, and scale up and down automatically.
- **Test systems at production scale**: In a traditional, non-cloud environment, it is usually cost-prohibitive to create a duplicate environment solely for testing. Consequently, most test environments are not tested at live levels of production demand. In the cloud, you can create a duplicate environment on demand, complete your testing, and then decommission the resources. Because you only pay for the test environment when it is running, you can simulate your live environment for a fraction of the cost of testing on premises.
- **Lower the risk of architecture change:** Because you can automate the creation of test environments that emulate your production configurations, you can carry out testing easily. You can also remove the test serialization that occurs in on-premises environments where teams have to queue to use the test resources.
- **Automate to make architectural experimentation easier:** Automation allows you to create and replicate your systems at low cost (no manual effort). You can track changes to your automation, audit the impact, and revert to previous parameters when necessary.
- **Allow for evolutionary architectures:** In a traditional environment, architectural decisions are often implemented as a static, one-time event, with a few major versions of a system during its lifetime. As a business and its context continue to change, these initial decisions may hinder the system's ability to deliver changing business requirements. In the cloud, the capability to automate and test on demand lowers the risk of impact from design changes. This allows systems to evolve over time so that businesses can take advantage of new innovations as a standard practice.

The Four Pillars of the Well-Architected Framework

Building a software system is a lot like constructing a building. If the foundation is not solid there might be structural problems that undermine the integrity and function of the building. When architecting technology solutions, if you neglect the four pillars of security, reliability, performance efficiency, and cost optimization it can become challenging to build a system that delivers on your expectations and requirements. When you incorporate these pillars into your architecture, it will help you produce stable and efficient systems. This will allow you to focus on the other aspects of design, such as functional requirements.

This section describes each of the four pillars, and includes definitions, best practices, questions, considerations, and key AWS services that are relevant.

Security Pillar

The **Security** pillar encompasses the ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies.

Design Principles

In the cloud, there are a number of principles that can help you strengthen your system security.

- **Apply security at all layers:** Rather than just running security appliances (e.g., firewalls) at the edge of your infrastructure, use firewalls and other security controls on all of your resources (e.g., every virtual server, load balancer, and network subnet).
- **Enable traceability:** Log and audit all actions and changes to your environment.
- **Automate responses to security events:** Monitor and automatically trigger responses to event-driven, or condition-driven, alerts.

- **Focus on securing your system:** With the AWS Shared Responsibility Model you can focus on securing your application, data, and operating systems, while AWS provides secure infrastructure and services.
- **Automate security best practices:** Software-based security mechanisms improve your ability to securely scale more rapidly and cost-effectively. Create and save a custom baseline image of a virtual server, and then use that image automatically on each new server you launch. Create an entire infrastructure that is defined and managed in a template.

Definition

Security in the cloud is composed of four areas:

1. Data protection
2. Privilege management
3. Infrastructure protection
4. Detective controls

The AWS Shared Responsibility Model enables organizations that adopt the cloud to achieve their security and compliance goals. Because AWS physically secures the infrastructure that supports our cloud services, AWS customers can focus on using services to accomplish their goals. The AWS cloud also provides greater access to security data and an automated approach to responding to security events.

Best Practices

Data Protection

Before architecting any system, foundational practices that influence security should be in place. For example, *data classification* provides a way to categorize organizational data based on levels of sensitivity; *least privilege* limits access to the lowest level possible while still allowing normal functions; and *encryption* protects data by way of rendering it unintelligible to unauthorized access. These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

Data protection involves using controls and patterns designed to keep your data confidential while preserving its integrity and ensuring that it is available to you when you need it.

In AWS, the following practices facilitate protection of data:

- AWS customers maintain full control over their data.
- AWS makes it easier for you to encrypt your data and manage keys, including regular key rotation, which can be easily automated natively by AWS or maintained by a customer.
- Detailed logging is available that contains important content, such as file access and changes.
- AWS has designed storage systems for exceptional resiliency. As an example, Amazon Simple Storage Service (S3) is designed for 11 nines of durability. (For example, if you store 10,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000,000 years.)
- Versioning, which can be part of a larger data lifecycle-management process, can protect against accidental overwrites, deletes, and similar harm.
- AWS never initiates the movement of data between regions. Content placed in a region will remain in that region unless the customer explicitly enable a feature or leverages a service that provides that functionality.

The following questions focus on considerations for data security (for a list of security question, answers, and best practices, see the Appendix):

SEC 1. How are you encrypting and protecting your data at rest?

SEC 2. How are you encrypting and protecting your data in transit?

AWS provides multiple means for encryption of data at rest and in transit. We build features into our products and services that make it easier to encrypt your data. For example, we have implemented Server Side Encryption (SSE) for Amazon S3 to make it easier for you to store your data in an encrypted form.

You can also arrange for the entire HTTPS encryption and decryption process (generally known as SSL termination) to be handled by Elastic Load Balancing.

Privilege management

Privilege management is a key part of an information security program; it ensures that only authorized and authenticated users are able to access your resources, and only in a manner that is intended. For example, an Access Control List (ACL) is a list of access permissions attached to an object, Role-Based Access Controls (RBAC) is a permission set that is aligned with an end user's role or function, and password management includes complexity requirements and change intervals. These privilege-management elements are critical in an information security architecture, as they represent the core concepts of user authentication and authorization.

In AWS, privilege management is primarily supported by the AWS Identity and Access Management (IAM) service, which allows customers to control access to AWS services and resources for users. You can apply granular policies, which assign permissions to a user, group, role, or resource. You also have the ability to require strong password practices, such as complexity, re-use, and multi-factor authentication (MFA), and you can use federation with your existing directory service.

The following questions focus on privilege-management considerations for security:

SEC 3. How are you protecting access to and use of the AWS root account credentials?

SEC 4. How are you defining roles and responsibilities of system users to control human access to the AWS Management Console and APIs?

SEC 5. How are you limiting automated access (such as from applications, scripts, or third-party tools or services) to AWS resources?

SEC 6. How are you managing keys and credentials?

It is critical to keep root account credentials protected, and to this end AWS recommends attaching MFA to the root account and locking the credentials with the MFA in a physically secured location. The IAM service allows you to create and manage other (non-root) user permissions, as well as establish access levels to resources.

Infrastructure protection

Infrastructure protection encompasses control methodologies, such as defense in depth and multi-factor authentication, necessary to meet best practices and industry or regulatory obligations. Use of these methodologies is critical for successful ongoing operations in either the cloud or on-premises.

In AWS, you can implement stateful and stateless packet inspection, either using AWS native technologies or by using partner products and services available through the AWS Marketplace. You can also use Amazon Virtual Private Cloud (VPC), to create a private, secured, and scalable environment in which you can define your topology—including gateways, routing tables, and public and/or private subnets.

The following questions focus on infrastructure-protection considerations for security:

SEC 7. How are you enforcing network and host-level boundary protection?

SEC 8. How are you enforcing AWS service level protection?

SEC 9. How are you protecting the integrity of the operating systems on your Amazon EC2 instances?

Multiple layers of defense are advisable in any type of environment, and in the case of infrastructure protection, many of the concepts and methods are valid across cloud and on-premises models. Enforcing boundary protection, monitoring points of ingress and egress, and comprehensive logging, monitoring, and alerting are all essential to an effective information security plan.

As mentioned in the *Design Principals* section above, AWS customers are able to tailor, or harden, the configuration of an EC2 instance, and persist this configuration to an immutable Amazon Machine Image (AMI). Then, whether

triggered by Auto Scaling or launched manually, all new virtual servers (instances) launched with this AMI receive the hardened configuration.

Detective Controls

You can use detective controls to detect or identify a security breach. They are a normal part of governance frameworks, and can be used to support a quality process, a legal compliance obligation, and/or threat identification and response efforts. There are different types of detective controls. For example, inventorying assets and their detailed attributes promotes more effective decision making (and lifecycle controls) to help establish operational baselines. Or you can use internal auditing, an examination of controls related to information systems, to ensure that practices meet policies and requirements, and that you have set the correct automated alerting notifications based on defined conditions. These controls are important reactive factors that help organizations identify and understand the scope of anomalous activity.

In AWS, the following services support detective controls:

- **AWS CloudTrail** – A web service that logs API calls, including the identity of the call, the time of the call, source IP address, parameters, and response elements.
- **Amazon CloudWatch** – A monitoring service for AWS resources that logs aspects such as Amazon Elastic Compute Cloud (EC2) CPU, disk, and network activity; Amazon Relational Database Service (RDS) database instances; Amazon Elastic Block Store (EBS) volumes; and more. CloudWatch provides the ability to alarm on these and other metrics.
- **AWS Config** – An inventory and configuration history service that provides information about the configurations and changes in infrastructure over time.
- **Amazon Simple Storage Service (S3)**– Using Amazon S3 data access auditing, customers can configure Amazon S3 buckets to record the details of access requests, including the type, resource, date, and time.
- **Amazon Glacier**– Customers can use the vault lock feature to preserve mission-critical data with compliance controls designed to support auditable long-term retention.

The following question focuses on detective controls considerations for security:

SEC 10. How are you capturing and analyzing AWS logs?

Log management is important to a well-architected design for reasons ranging from security/forensics to regulatory or legal requirements. AWS provides functionality that makes log management easier to implement by giving customers the ability to define a data-retention lifecycle, or define where data will be preserved, archived, and/or eventually deleted. This makes predictable and reliable data handling simpler and more cost effective.

Key AWS Services

The AWS service that is essential to security is AWS Identity and Access Management (IAM), which allows you to securely control access to AWS services and resources for your users. The following services and features support the four areas of security:

Data protection: Services such as Elastic Load Balancing, Amazon Elastic Block Store (EBS), Amazon Simple Storage Service (S3), and Amazon Relational Database Service (RDS) include encryption capabilities to protect your data in transit and at rest. AWS Key Management Service (KMS) makes it easier for customers to create and control keys used for encryption.

Privilege management: IAM enables you to securely control access to AWS services and resources. Multi-factor authentication (MFA), adds an extra layer of protection on top of your user name and password.

Infrastructure protection: Amazon Virtual Private Cloud (VPC) lets you provision a private, isolated section of the AWS cloud where you can launch AWS resources in a virtual network.

Detective controls: AWS CloudTrail records AWS API calls, AWS Config provides a detailed inventory of your AWS resources and configuration, and Amazon CloudWatch is a monitoring service for AWS resources.

Resources

Refer to the following resources to learn more about our best practices for security.

Documentation & Blogs

- AWS Security Center
- AWS Compliance
- AWS Security Blog

Whitepapers

- AWS Security Overview
- AWS Security Best Practices
- AWS Risk and Compliance

Videos

- Security of the AWS Cloud
- Shared Responsibility Overview

Reliability Pillar

The **Reliability** pillar encompasses the ability of a system to recover from infrastructure or service disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues.

Design Principles

In the cloud, there are a number of principles that can help you increase reliability:

- **Test recovery procedures:** In an on-premises environment, testing is often conducted to prove the system works in a particular scenario; testing

is not typically used to validate recovery strategies. In the cloud, you can test how your system fails, and you can validate your recovery procedures. You can use automation to simulate different failures or to recreate scenarios that led to failures before. This exposes failure pathways that you can test and rectify *before* a real failure scenario, reducing the risk of components failing that have not been tested before.

- **Automatically recover from failure:** By monitoring a system for key performance indicators (KPIs), you can trigger automation when a threshold is breached. This allows for automatic notification and tracking of failures, and for automated recovery processes that work around or repair the failure. With more sophisticated automation, it is possible to anticipate and remediate failures before they occur.
- **Scale horizontally to increase aggregate system availability:** Replace one large resource with multiple small resources to reduce the impact of a single failure on the overall system. Distribute requests across multiple, smaller resources to ensure that they don't share a common point of failure.
- **Stop guessing capacity:** A common cause of failure in on-premises systems is resource saturation, when the demands placed on a system exceed the capacity of that system (this is often the objective of denial of service attacks). In the cloud, you can monitor demand and system utilization, and automate the addition or removal of resources to maintain the optimal level to satisfy demand without over- or under-provisioning.

Definition

Reliability in the cloud is composed of three areas:

1. Foundations
2. Change management
3. Failure management

To achieve reliability, a system must have a well-planned foundation and monitoring in place, with mechanisms for handling changes in demand or requirements. The system should be designed to detect failure and automatically heal itself.

Best Practices

Foundations

Before architecting any system, foundational requirements that influence reliability should be in place; for example, you must have sufficient network bandwidth to your data center. These requirements are sometimes neglected

(because they are beyond a single project's scope). This neglect can have a significant impact on the ability to deliver a reliable system. In an on-premises environment, these requirements can cause long lead times due to dependencies and therefore must be incorporated during initial planning.

With AWS, most of these foundational requirements are already incorporated or may be addressed as needed. The cloud is designed to be essentially limitless, so it is the responsibility of AWS to satisfy the requirement for sufficient networking and compute capacity, while you are free to change resource size and allocation, such as the size of storage devices, on demand.

The following questions focus on foundational considerations for reliability (for a full list of reliability questions, answers, and best practices, see the Appendix):

REL 1. How are you managing AWS service limits for your account?

REL 2. How are you planning your network topology on AWS?

REL 3. Do you have an escalation path to deal with technical issues?

AWS sets service limits (an upper limit on the number of each resource your team can request) to protect you from accidently over-provisioning resources. You will need to have governance and processes in place to monitor and change these limits to meet your business needs. As you adopt the cloud, you may need to plan integration with existing on-premises resources (a hybrid approach). A hybrid model enables the gradual transition to an all-in, cloud approach over time, and therefore it's important to have a design for how your AWS and on-premises resources will interact as a network topology. Finally, you will want to ensure your IT team receives training and updated processes to support your public-cloud usage, and that you have partner or support agreements in place when appropriate.

Change Management

Being aware of how change affects a system allows you to plan proactively, and monitoring allows you to quickly identify trends that could lead to capacity issues or SLA breaches. In traditional environments, change-control processes are often manual and must be carefully coordinated with auditing to effectively control who makes changes and when they are made.

Using AWS, you can monitor the behavior of a system and automate the response to KPIs, for example, adding additional servers as a system gains more users. You can control who has permission to make system changes and audit the history of these changes.

The following questions focus on change-related considerations for reliability:

REL 4. How does your system adapt to changes in demand?
REL 5. How are you monitoring AWS resources?
REL 6. How are you executing change management?

When you architect a system to automatically add and remove resources in response to changes in demand, this not only increases reliability but also ensures that business success does not become a burden. With monitoring in place, your team will be automatically alerted when KPIs deviate from expected norms. Automatic logging of changes to your environment allows you to audit and quickly identify actions that might have impacted reliability. Controls on change management ensure that you can enforce the rules that deliver the reliability you need.

Failure Management

In any system of reasonable complexity it is expected that failures will occur, and it is generally of interest to know how to become aware of these failures, respond to them, and prevent them from happening again.

In AWS, we can take advantage of automation to react to monitoring data. For example, when a particular metric crosses a threshold, you can trigger an automated action to remedy the problem. Also, rather than trying to diagnose and fix a failed resource that is part of your production environment, you can replace it with a new one and carry out the analysis on the failed resource out of band. Since the cloud enables you to stand up temporary versions of a whole system at low cost, you can use automated testing to verify full recovery processes.

The following questions focus on failure-management considerations for reliability:

REL 7. How are you backing up your data?

REL 8. How does your system withstand component failures?

REL 9. How are you planning for recovery?

Regularly back up your data, and test your backup files, to ensure you can recover from both logical and physical errors. A key to managing failure is the frequent, and automated testing of systems to failure and through recovery (ideally on a regular schedule and also triggered after significant system changes). Actively track KPIs, such as the recovery time objective (RTO) and recovery point objective (RPO), to assess a system's fitness (especially under failure-testing scenarios) and to help you identify and mitigate single points of failure. The objective is to thoroughly test your system-recovery processes so that you are confident that you can recover all your data and continue to serve your customers, even in the face of sustained problems. Your recovery processes should be as well exercised as your normal production processes.

Key AWS Services

The AWS service that is key to ensuring reliability is Amazon CloudWatch, which monitors run-time metrics. Other services and features that support the three areas of Reliability are as follows:

Foundations: AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources. Amazon VPC lets you provision a private, isolated section of the AWS cloud where you can launch AWS resources in a virtual network.

Change management: AWS CloudTrail records AWS API calls for your account and delivers log files to you for auditing. AWS Config provides a detailed inventory of your AWS resources and configuration, and continuously records configuration changes.

Failure management: AWS CloudFormation enables the template creation of AWS resources and provisions them in an orderly and predictable fashion.

Resources

Refer to the following resources to learn more about our best practices related to reliability.

Video and Analyst Report

- Embracing Failure: Fault-Injection and Service Reliability
- Benchmarking Availability and Reliability in the Cloud

Documentation and Blogs

- Service Limits Documentation
- Service Limit Reports Blog Post

Whitepapers

- Backup Archive and Restore Approach Using AWS Whitepaper
- Managing your AWS Infrastructure at Scale Whitepaper
- AWS Disaster Recovery Whitepaper
- AWS Amazon VPC Connectivity Options Whitepaper

AWS Support

- AWS Premium Support
- Trusted Advisor

Performance Efficiency Pillar

The **Performance Efficiency** pillar focuses on the efficient use of computing resources to meet requirements, and maintaining that efficiency as demand changes and technologies evolve.

Design Principles

In the cloud, there are a number of principles that can help you achieve performance efficiency:

- **Democratize advanced technologies:** Technologies that are difficult to implement can become easier to consume by pushing that knowledge and complexity into the cloud vendor's domain. Rather than having your IT team learn how to host and run a new technology, they can simply consume it as a service. For example, NoSQL databases, media transcoding, and

machine learning are all technologies that require expertise that is not evenly dispersed across the technical community. In the cloud, these technologies become services that your team can consume while focusing on product development rather than resource provisioning and management.

- **Go global in minutes:** Easily deploy your system in multiple regions around the world with just a few clicks. This allows you to provide lower latency and a better experience for your customers simply and at minimal cost.
- **Use *server-less* architectures:** In the cloud, server-less architectures remove the need for you to run and maintain servers to carry out traditional compute activities. For example, storage services can act as static websites, removing the need for web servers; and event services can host your code for you. This not only removes the operational burden of managing these servers, but also can lower transactional costs because these managed services operate at cloud scale.
- **Experiment more often:** With virtual and automatable resources, you can quickly carry out comparative testing using different types of instances, storage, or configurations.

Definition

Performance Efficiency in the cloud is composed of four areas:

1. Compute
2. Storage
3. Database
4. Space-time trade-off

Considerations within each of these areas include a) how to select the optimal approach and resources, b) how to keep that approach current given evolving cloud capabilities, c) how to monitor run-time performance against expectations, and, finally, d) how the resources scale against demand.

Best Practices

Compute

The optimal server configuration for a particular architecture may vary based on application design, usage patterns, and configuration settings. Many systems use different server configurations for various components and enable different features to improve performance. Selecting the wrong server configuration for a use case can lead to lower performance efficiency.

In AWS, servers are virtualized and, therefore, you can change their capabilities with the click of a button or an API call. Because resource decisions are no longer fixed, you can experiment with different server types. At AWS, these virtual server *instances* come in different families and sizes, offering a wide variety of capabilities such as SSDs and GPUs. In AWS, it is also possible to perform server-less computing. For example, AWS Lambda allows you to execute code without running an instance.

The following example questions focus on compute considerations (for a full list of performance efficiency questions, answers, and best practices, see the Appendix):

PERF 1.	**How do you select the appropriate instance type for your system?**
PERF 2.	**How do you ensure that you continue to have the most appropriate instance type as new instance types and features are introduced?**
PERF 3.	**How do you monitor your instances post launch to ensure they are performing as expected?**
PERF 4.	**How do you ensure that the quantity of your instances matches demand?**

When selecting the instance types to use, it is important to have test data that shows which instances types (or server-less approaches) match *that* workload best. These tests should be repeatable (ideally part of the continuous delivery (CD) pipeline) so that you can easily test new instance types or capabilities as they become available. From an operational standpoint, you should have monitoring in place to notify you of any degradation in performance.

Storage

The optimal storage solution for a particular system will vary based on the kind of access method (block, file, or object), patterns of access (random or sequential), throughput required, frequency of access (online, offline, archival), frequency of update (worm, dynamic), and availability and durability constraints. Well architected systems use multiple storage solutions and enable different features to improve performance.

In AWS, storage is virtualized and is available in a number of different types. This makes it easier to match your storage methods more closely with your needs, and also offers storage options that are not easily achievable with on-premises infrastructure. For example, Amazon S3 is designed for 11 nines of durability. You can also change from using magnetic hard drives (HDDs) to solid state drives (SSDs), and easily move virtual drives from one instance to another in seconds.

The following example questions focus on storage considerations for performance efficiency:

PERF 5. How do you select the appropriate storage solution for your system?

PERF 6. How do you ensure that you continue to have the most appropriate storage solution as new storage solutions and features are launched?

PERF 7. How do you monitor your storage solution to ensure it is performing as expected?

PERF 8. How do you ensure that the capacity and throughput of your storage solutions matches demand?

When selecting a storage solution, it is important to have test data that shows which storage solution will deliver the cost/value margin required for *that* workload. These tests should be repeatable (ideally part of the CD pipeline) so that you can easily test new storage solutions or capabilities as they become available. The types of storage (EBS versus instance store, or HDD versus SSD) used for different instances can substantially alter the performance efficiency of your system. From an operational standpoint, you should have monitoring in place to notify you of any degradation in performance.

Database

The optimal database solution for a particular system can vary based on requirements for consistency, availability, partition tolerance, and latency. Many systems use different database solutions for various sub-systems and enable different features to improve performance. Selecting the wrong database solution and features for a system can lead to lower performance efficiency.

In AWS, Amazon Relational Database Service (RDS) provides a fully managed relational database. With Amazon RDS you can scale your database's compute and storage resources, often with no downtime. We also offer other database and storage solutions. Amazon DynamoDB is a fully managed NoSQL database that provides single-digit millisecond latency at any scale. Amazon Redshift is a managed petabyte-scale data warehouse that allows you to change the number or type of nodes as your performance or capacity needs change.

The following example questions focus on database considerations for performance efficiency:

PERF 9. How do you select the appropriate database solution for your system?

PERF 10. How do you ensure that you continue to have the most appropriate database solution and features as new database solution and features are launched?

PERF 11. How do you monitor your databases to ensure performance is as expected?

PERF 12. How do you ensure the capacity and throughput of your databases matches demand?

Although an organization’s database approach (RDBMS, NoSQL, etc.) has significant impact on a system’s performance efficiency, it is often an area that is chosen according to organizational defaults rather than through assessment. During the build and deployment of your database solution, treat the database as code to allow it to evolve over time rather than be a one-time fixed decision. Use test data to identify which database solution matches each workload best. These tests should be repeatable (ideally part of the CD pipeline) so that you can easily test new database solutions or capabilities as they become available. For example, assess whether read-only replicas improve performance efficiency without

violating other non-functional requirements. From an operational standpoint, have monitoring in place to notify you of any degradation in performance.

Space-Time trade-off

When architecting solutions, there is a series of trade-offs where space (memory or storage) is used to reduce processing time (compute), or time is used to reduce space. You can also position resources or cached data closer to end users to reduce time.

Using AWS, you can go global in minutes and deploy resources in multiple locations across the globe to be closer to your end users. You can also dynamically add read-only replicas to information stores such as database systems to reduce the load on the primary database.

Use the global infrastructure of AWS to achieve lower latency and higher throughput, and ensure that your data resides only in the region(s) you specify. Networking solutions such as AWS Direct Connect are designed to provide predictable latency between your on-premises network and AWS infrastructure. AWS also offers caching solutions such as Amazon ElastiCache, which helps improve efficiency, and Amazon CloudFront, which caches copies of your static content closer to end-users.

The following example questions focus on space-time trade-offs for Performance Efficiency:

PERF 13. How do you select the appropriate proximity and caching solutions for your system?

PERF 14. How do you ensure that you continue to have the most appropriate proximity and caching solutions as new solutions are launched?

PERF 15. How do you monitor your proximity and caching solutions to ensure performance is as expected?

PERF 16. How do you ensure that the proximity and caching solutions you have matches demand?

Space-time trade-offs are required to deliver performance efficiency, and it is important to have test data that shows which trade-offs match *that* workload best. These tests should be repeatable (ideally part of the CD pipeline) so that you can easily test new approaches or capabilities as they become available. For example, test to see if using Amazon ElastiCache as a write-aside cache improves performance efficiency without violating other non-functional requirements. From an operational standpoint, you should have monitoring in place to notify you of any degradation in performance. The architecture should scale with demand and maintain its margin.

Key AWS Services

The key AWS service for performance efficiency is Amazon CloudWatch, which monitors your resources and systems, providing visibility into your overall performance and operational health. The following services are important in the four areas of performance efficiency:

Compute: Auto Scaling is key to ensuring that you have enough instances to meet demand and maintain responsiveness.

Storage: Amazon EBS provides a wide range of storage options (such as SSD and PIOPS) that allow you to optimize for your use case. Amazon S3 provides reduced-redundancy storage, lifecycle policies to Amazon Glacier (archival storage), and server-less content delivery.

Database: Amazon RDS provides a wide range of database features (such as provisioned IOPS, and read replicas) that allow you to optimize for your use case. Amazon DynamoDB provides single-digit millisecond latency at any scale.

Space-time trade-off: AWS has regions across the globe, allowing you to choose the optimal location for your resources, data, and processing. Use Amazon CloudFront to cache content even closer to your users.

Resources

Refer to the following resources to learn more about our best practices related to performance efficiency.

Videos

- Performance Channel
- Performance Benchmarking on AWS

Documentation

- Amazon S3 Performance Optimization Documentation
- Amazon EBS Volume Performance Documentation

Cost Optimization Pillar

Use the **Cost Optimization** pillar to assess your ability to avoid or eliminate unneeded costs or suboptimal resources, and use those savings on differentiated benefits for your business. A cost-optimized system allows you to pay the lowest price possible while still achieving your business objectives and meeting, or exceeding, key requirements for the other Well-Architected pillars. You can achieve cost optimization using techniques to select the appropriate architecture, reduce unused resources, and select the most economical approach.

Design Principles

In the cloud you can follow a number of principles that to help you achieve cost optimization:

- **Transparently attribute expenditure:** The cloud makes it easier to identify the cost of a system and attribute IT costs to individual business owners. This helps identify return on investment and, consequently, gives those owners an incentive to optimize their resources and reduce costs.
- **Use managed services to reduce cost of ownership:** In the cloud, managed services remove the operational burden of maintaining servers for tasks such as sending email or managing databases. Additionally, because managed services operate at cloud scale, they can offer a lower cost per transaction or service.
- **Trade capital expense for operating expense:** Instead of investing heavily in data centers and servers before you know how you're going to use them, pay only for the computing resources you consume, when you consume them. For example, development and test environments are typically only used for eight hours a day during the working week, so you can stop these resources when not in use for a potential cost savings of 75% (40 hours versus 168 hours).
- **Benefit from economies of scale:** By using cloud computing, you may achieve a lower variable cost than you could on your own because AWS

can achieve higher economies of scale. Hundreds of thousands of customers are aggregated in the AWS cloud, which translates into lower pay-as-you-go prices.

- **Stop spending money on data center operations:** AWS does the heavy lifting of racking, stacking, and powering servers, so you can focus on your customers and business projects rather than on IT infrastructure.

Definition

Cost Optimization in the cloud is composed of four areas:

1. Matched supply and demand
2. Cost-effective resources
3. Expenditure awareness
4. Optimizing over time

As with the other pillars, there are trade-offs to consider, for example, whether to optimize for speed to market or for cost. In some cases, it's best to optimize for speed—going to market quickly, shipping new features, or simply meeting a deadline—rather than investing in upfront cost optimization. Design decisions are sometimes guided by haste as opposed to empirical data, as the temptation always exists to overcompensate "just in case" rather than spend time benchmarking for the most cost-optimal deployment. This often leads to drastically over-provisioned and under-optimized deployments. The following sections provide techniques and strategic guidance for the initial and ongoing cost optimization of your deployment.

Best Practices

Matched Supply and Demand

Optimally matching supply to demand delivers the lowest costs for a system, but there also needs to be sufficient extra supply to allow for provisioning time and individual resource failures. Demand can be fixed or variable, requiring metrics and automation to ensure that management does not become a significant cost.

In AWS, you can automatically provision resources to match demand. Auto Scaling and time-based, event-driven, and queue-based approaches allow you to add and remove resources as needed. If you can anticipate changes in demand, you can save more money and ensure your resources match your system needs.

The following example questions focus on matched supply and demand for cost optimization (for a full list of cost optimization questions, answers, and best practices, see the Appendix):

COST 1. How do you make sure your capacity matches but does not substantially exceed what you need?

COST 2. How are you optimizing your usage of AWS services?

Monitoring tools and regular benchmarking can help you achieve much greater utilization of resources. The flexibility of on-demand computing, Auto Scaling, and other automated deployment mechanisms facilitate a greater degree of optimization, ensuring that you provision only the resources you need and are able to scale horizontally.

Cost-Effective Resources

Using the appropriate instances and resources for your system is key to cost savings. For example, a reporting process might take five hours to run on a smaller server, but a larger server that is twice as expensive can do it in one hour. Both jobs give you the same outcome, but the smaller server will incur more cost over time.

A well architected system will use the most cost-effective resources, which can have a significant and positive economic impact. You also have the opportunity to use managed services to reduce costs. For example, rather than maintaining servers to deliver email, you can use a service that charges on a per-message basis.

AWS offers a variety of flexible and cost-effective pricing options to acquire Amazon EC2 instances in a way that best fits your needs. *On-Demand instances* allow you to pay for compute capacity by the hour, with no minimum commitments required. *Reserved Instances* (RIs) allow you to reserve capacity and offers savings of up to 75 percent off on-demand pricing. With *Spot instances*, you can bid on unused Amazon EC2 capacity at significant discounts. Spot instances are appropriate where the system can tolerate using a fleet of servers where individual servers can come and go dynamically, such as when using HPC and big data.

The following example questions focus on selecting cost-effective resources for cost optimization:

COST 3. Have you selected the appropriate resource types to meet your cost targets?

COST 4. Have you selected the appropriate pricing model to meet your cost targets?

COST 5. Are there managed services (higher-level services than Amazon EC2, Amazon EBS, and Amazon S3) that you can use to improve your ROI?

By using tools such as AWS Trusted Advisor to regularly review your AWS usage, you can actively monitor your utilization and adjust your deployments accordingly. You can also take advantage of managed AWS services, such as Amazon RDS, Amazon Elastic MapReduce (EMR), and Amazon DynamoDB, which can lower per-item and management costs. Consider CDN solutions such as Amazon CloudFront to potentially reduce your costs associated with network traffic.

Expenditure Awareness

The increased flexibility and agility that the cloud enables encourages innovation and fast-paced development and deployment. It eliminates the manual processes and time associated with provisioning on-premises infrastructure, including identifying hardware specifications, negotiating price quotations, managing purchase orders, scheduling shipments, and then deploying the resources. However, the ease of use and virtually unlimited on-demand capacity may require a new way of thinking about expenditures.

Many businesses are composed of multiple systems run by various teams. The capability to attribute resource costs to the individual business or product owners drives efficient usage behavior and helps reduce waste. Accurate cost attribution also allows you to understand which products are truly profitable, and allows you to make more informed decisions about where to allocate budget.

The following example questions focus on expenditure awareness for cost optimization:

COST 6.	**What access controls and procedures do you have in place to govern AWS costs?**
COST 7.	**How are you monitoring usage and spending?**
COST 8.	**How do you decommission resources that you no longer need, or stop resources that are temporarily not needed?**
COST 9.	**How do you consider data-transfer charges when designing your architecture?**

You can use cost allocation tags to categorize and track your AWS costs. When you apply tags to your AWS resources (such as Amazon EC2 instances or Amazon S3 buckets), AWS generates a cost allocation report with your usage and costs aggregated by your tags. You can apply tags that represent business categories (such as cost centers, system names, or owners) to organize your costs across multiple services.

With this visibility of costs against tagged resources it becomes easier to identify orphaned resources or projects that are no longer generating value to the business and should be decommissioned. You can set up billing alerts to notify you of predicted overspending, and the AWS Simple Monthly Calculator allows you to calculate your data transfer costs.

Optimizing Over Time

As AWS releases new services and features, it is a best practice to reassess your existing architectural decisions to ensure they continue to be the most cost effective. As your requirements change, be aggressive in decommissioning resources and entire services, or systems that you no longer require.

Managed services from AWS can often significantly optimize a solution, so it is good to be aware of new managed services as they become available. For example, running an Amazon RDS database can be cheaper than running your own database on Amazon EC2.

The following example question focuses on cost reassessments for cost optimization:

COST 10. How do you manage and/or consider the adoption of new services?

By regularly reassessing your deployment, it is often possible to utilize new AWS services to lower your costs. Also, assess the applicability of newer services to help save you money; for example, AWS RDS for Aurora could help you reduce costs for relational databases.

Key AWS Services

The key AWS feature that supports cost optimization is cost allocation tags, which help you to understand the costs of a system. The following services and features are important in the four areas of cost optimization:

Matched supply and demand: Auto Scaling allows you to add or remove resources to match demand without overspending.

Cost-effective resources: You can use Reserved Instances and prepaid capacity to reduce your cost. AWS Trusted Advisor can be used to inspect your AWS environment and find opportunities to save money.

Expenditure awareness: Amazon CloudWatch alarms and Amazon Simple Notification Service (SNS) notifications will warn you if you go, or are forecasted to go, over your budgeted amount.

Optimizing over time: The AWS Blog and *What's New* section on the AWS website are resources for learning about newly launched features and services. AWS Trusted Advisor inspects your AWS environment and finds opportunities to save money by eliminating unused or idle resources or committing to Reserved Instance capacity.

Resources

Refer to the following resources to learn more about AWS best practices for cost optimization.

Video

- Cost Optimization on AWS

Documentation

- AWS Economics Center

Tools

- AWS Total Cost of Ownership (TCO) Calculator
- AWS Detailed Billing Reports
- AWS Simple Monthly Calculator
- AWS Cost Explorer

Conclusion

The AWS Well-Architected Framework provides architectural best practices across four pillars for designing reliable, secure, efficient, and cost-effective systems in the cloud. The framework provides a set of questions that allows you to assess an existing or proposed architecture, and also a set of AWS best practices for each pillar. Using the framework in your architecture will help you produce stable and efficient systems, which allows you to focus on your functional requirements.

Contributors

The following individuals and organizations contributed to this document:

- Philip Fitzsimons, Manager Solutions Architecture, Amazon Web Services
- Erin Rifkin, Senior Program Manager, Amazon Web Services
- Callum Hughes, Solutions Architect, Amazon Web Services
- Max Ramsay, Principal Security Solutions Architect, Amazon Web Services
- Scott Paddock, Security Solutions Architect, Amazon Web Services

Appendix: Well-Architected Questions, Answers, and Best Practices

This appendix contains the full list of Well-Architected questions and answers, including best practices, organized by pillar:

Security Pillar

SEC 1. How are you encrypting and protecting your data at rest?

A traditional security control is to encrypt data at rest. AWS supports this using both client-side (e.g., SDK-supported, OS-supported, Windows Bitlocker, dm-crypt, Trend Micro SafeNet, etc.) and server-side (e.g., Amazon S3). You can also use Server-Side Encryption (SSE) and Amazon Elastic Block Store Encrypted Volumes, etc.

Best practices:

- Data at rest is encrypted using AWS service specific controls (e.g., Amazon S3 SSE, Amazon EBS encrypted volumes, Amazon Relational Database Service (RDS) Transparent Data Encryption (TDE), etc.).
- Data at rest is encrypted using client side techniques.
- A solution from the AWS Marketplace or from an APN Partner.

SEC 2. How are you encrypting and protecting your data in transit?

A best practice is to protect data in transit by using encryption. AWS supports using encrypted end-points for the service APIs. Additionally, customers can use various techniques within their Amazon EC2 instances.

Best practices:

- SSL enabled AWS APIs are used appropriately.
- SSL or equivalent is used for communication.
- VPN based solution.
- Private connectivity (e.g., AWS Direct Connect).
- AWS Marketplace solution is being used.

SEC 3. How are you protecting access to and use of the AWS root account credentials?

The AWS root account credentials are similar to root or local admin in other operating systems and should be used very sparingly. The current best practice is to create AWS Identity and Access Management (IAM) users, associate them to an administrator group, and use the IAM user to manage the account. The AWS root account should not have API keys, should have a strong password, and should be associated with a hardware multi-factor authentication (MFA) device; this forces the only use of the root identity to be via the AWS Management Console and does not allow it to be used for application programming interface (API) calls. Note that some resellers or regions do not distribute or support the AWS root account credentials.

Best practices:

- The AWS root account credentials are only used for only minimal required activities.
- There is a MFA hardware device associated with the AWS root account.
- AWS Marketplace solution is being used.

SEC 4. How are you defining roles and responsibilities of system users to control human access to the AWS Management Console and API?

The current best practice is for customers to segregate defined roles and responsibilities of system users by creating user groups. User groups can be defined using several different technologies: Identity and Access Management (IAM) groups, IAM roles for cross-account access, Web Identities, via Security Assertion Markup Language (SAML) integration (e.g., defining the roles in Active Directory), or by using a third-party solution (e.g., Okta, Ping Identity, or another custom technique) which usually integrates via either SAML or AWS Security Token Service (STS). Using a shared account is strongly discouraged.

Best practices:

- IAM users and groups
- SAML integration
- Web Identity Federation
- AWS Security Token Service (STS)
- IAM roles for cross-account access

- A solution from the AWS Marketplace (e.g., Okta, Ping Identity, etc.) or from an APN Partner
- Employee life-cycle policies are defined and enforced
- Users, groups, and roles are clearly defined and granted only the minimum privileges needed to accomplish business requirements

SEC 5. How are you limiting automated access to AWS resources? (e.g., applications, scripts, and/or third-party tool or service)

Systematic access should be defined in similar ways as user groups are for created for people. For Amazon EC2 instances, these groups are called IAM roles for EC2. The current best practice is to use IAM roles for EC2 and an AWS SDK or CLI, which has built-in support for retrieving the IAM roles for EC2 credentials. Traditionally, user credentials are injected into EC2 instances, but hard-coding the credential into scripts and source code is actively discouraged.

Best practices:

- IAM roles for Amazon EC2
- IAM user credential is used, but not hardcoded into scripts and applications
- SAML integration
- AWS Security Token Service (STS)
- OS-specific controls are used for EC2 instances
- AWS Marketplace solution is being used

SEC 6. How are you managing keys and credentials?

Keys and credentials are secrets that should be protected, and an appropriate rotation policy should be defined and used. The best practice is to not hard-code these secrets into management scripts and applications, but it does often occur.

Best practices:

- Appropriate key and credential rotation policy is being used.
- Use AWS CloudHSM.
- AWS server-side techniques are used with AWS managed keys (e.g., Amazon S3 SSE, Amazon EBS encrypted volumes, etc.).

- AWS Marketplace solutions (e.g., SafeNet, TrendMicro, etc.).

SEC 7. How are you enforcing network and host-level boundary protection?

In on-premises datacenters, a DMZ approaches separate systems into trusted and untrusted zones using firewalls. On AWS, both stateful and stateless firewalls are used. Stateful firewalls are called security groups, and stateless firewalls are called network Access Control Lists (ACL) that protect the subnets in an Amazon Virtual Private Cloud (VPC). The current best practice is to run a system in a VPC, and define the role-based security in Security Groups (e.g., web tier, app tier, etc.), and the location-based security in network ACLs (e.g., Elastic Load Balancing tier in one subnet per Availability Zone , web tier in another subnet per Availability Zone, etc.).

Best practices:

- Security groups with minimal authorizations are used to enforce role-based access.
- The system runs in one or more VPCs.
- Trusted VPC access is via a private mechanism (e.g., Virtual Private Network (VPN), IPsec tunnel, AWS Direct Connect, AWS Marketplace solution, etc.).
- Subnets and network ACLs are used appropriately.
- Host-based firewalls with minimal authorizations are used.
- Service-specific access controls are used (e.g., bucket policies).
- Private connectivity to a VPC is used (e.g., VPN, AWS Direct Connect, VPC peering, etc.)
- Bastion host technique is used to manage the instances.
- Security testing is performed regularly.
- AWS Trusted Advisor checks are regularly reviewed.

SEC 8. How are you enforcing AWS service level protection?

Another best practice is to control access to resources. AWS Identity and Access Management (IAM) allows various resource level controls to be defined (e.g., use of encryption, time of day, source IP, etc.) and various services allow additional techniques to be used (e.g., Amazon S3 bucket

policies, etc.). Additionally, customers can use various techniques within their Amazon EC2 instances.

Best practices:

- Credentials configured with the least privilege.
- Separation of duties.
- Periodic auditing of permissions.
- Resource requirements are defined for sensitive API calls, such as requiring MFA authentication and encryption.
- Service-specific requirements are defined and used.
- AWS Marketplace solution is being used.

SEC 9. How are you protecting the integrity of the operating system on your Amazon EC2 instances?

Another traditional control is to protect the integrity of the operating system. This is easily done in EC2 using traditional host-based techniques (e.g., OSSEC, Tripwire, Trend Micro Deep Security, etc.).

Best practices:

- File integrity controls are used for EC2 instances.
- Host-based intrusion detection controls are used for EC2 instances.
- Use of a solution from the AWS Marketplace or an APN Partner.
- Use of a custom AMI or configuration management tools (i.e., Puppet or Chef) that is secured by default.

SEC 10. How are you capturing and analyzing AWS logs?

Capturing logs is critical for investigating everything from performance to security incidents. The current best practice is for the logs to be periodically moved from the source either directly into a log processing system (e.g., Splunk, Papertrail, etc.) or stored in an Amazon S3 bucket for later processing based on business needs. Common sources of logs are AWS API and user-related logs (e.g., AWS CloudTrail), AWS service-specific logs (e.g., Amazon S3, Amazon CloudFront, etc.), Operating system-generated logs, and third-party application-specific logs.

Best practices:

- AWS CloudTrail.
- Elastic Load Balancing (ELB) logs.
- Amazon Virtual Private Cloud (VPC) filter logs.
- Amazon S3 bucket logs.
- Amazon CloudWatch logs.
- Other AWS service-specific log sources.
- Operating system or third-party application logs.
- AWS Marketplace solution is being used.

Reliability Pillar

REL 1. How are you managing AWS Service Limits for your account?

AWS accounts are provisioned with default service limits to prevent new users from accidentally provisioning more resources than they need. AWS customers should evaluate their AWS service needs and request appropriate changes to their limits for each region used.

Best practices:

- **Monitor and manage limits** Evaluate your potential usage on AWS, increase your regional limits appropriately, and allow planned growth in usage.
- **Set up automated monitoring** Implement tools, e.g., SDKs, to alert you when thresholds are being approached.
- **Be aware of fixed service limits** Be aware of unchangeable service limits and architected around these.

REL 2. How are you planning your network topology on AWS?

Applications can exist in one or more environments: EC2 Classic, VPC, or VPC by Default. Network considerations such as system connectivity, EIP/public IP address management, VPC/private address management, and name resolution are fundamental to leveraging resources in the cloud. Well-planned and documented deployments are essential to reduce the risk of overlap and contention.

Best practices:

- **Highly available connectivity to AWS** Multiple DX circuits, multiple VPN tunnels, AWS Marketplace appliances.
- **Highly available connectivity to the system** Highly available load balancing and/or proxy, DNS-based solution, AWS Marketplace appliances, etc.
- **Non-overlapping private IP ranges** The use of your IP address ranges and subnets in your virtual private cloud should not overlap each other, other cloud environments, or your on-premises environments.
- **IP subnet allocation** Individual Amazon VPC IP address ranges should be large enough to accommodate an application's requirements including factoring in future expansion and allocation of IP addresses to subnets across Availability Zones.

REL 3. Do you have an escalation path to deal with technical issues?

Customers should leverage AWS Support or an AWS partner. Regular interaction will help address and prevent known issues, knowledge gaps, and design concerns. This will reduce the risk of implementation failures and also large-scale outages.

Best practices:

- **Planned** Ongoing engagement /relationship with AWS Support or an APN Partner.
- **Leverage AWS Support APIs** Integrate the AWS Support API with your internal monitoring and ticketing systems.

REL 4. How does your system adapt to changes in demand?

A scalable system can provide elasticity to add and remove resources automatically so that they closely match the current demand at any given point in time.

Best practices:

- **Automated scaling** Use automatically scalable services, e.g., Amazon S3, Amazon CloudFront, Auto Scaling, Amazon DynamoDB, AWS Elastic Beanstalk, etc.
- **Load Test** Adopt a load testing methodology to measure if scaling activity will meet application requirements.

REL 5. How are you monitoring AWS resources?

Logs and metrics are a powerful tool for gaining insight into the health of your applications. You can configure your system to monitor logs and metrics and send notifications when thresholds are crossed or significant events occur. Ideally, when low-performance thresholds are crossed or failures occur, the system will have been architected to automatically self-heal or scale in response.

Best practices:

- **Monitoring** Monitor your applications with Amazon CloudWatch or third-party tools.
- **Notification** Plan to receive notifications when significant events occur.
- **Automated Response** Use automation to take action when failure is detected, e.g., to replace failed components.
- **Review** Perform frequent reviews of the system based on significant events to evaluate the architecture.

REL 6. How are you executing change management?

Change management of provisioned AWS resources and applications is necessary to ensure that the applications and operating environment are running known software and can be patched or replaced in a controlled manner.

Best practices:

- **CM Automated** Automate deployments /patching.

REL 7. How are you backing up your data?

Back up data, applications, and operating environments (defined as operating systems configured with applications) to meet requirements for mean time to recovery (MTTR) and recovery point objectives (RPO).

Best practices:

- **Data is Backed Up** Back up important data using Amazon S3, Amazon EBS snapshots, or third-party software to meet RPO.
- **Automated Backups** Use AWS features, AWS Marketplace solutions, or third-party software to automate backups.
- **Backups are Secured and/or Encrypted** See the AWS Security Best Practices whitepaper.
- **Periodic Recovery Testing** Validate that the backup process implementation meets RTO and RPO through a recovery test.

REL 8. How does your system withstand component failures?

Do your applications have a requirement, implicit or explicit, for high availability and low mean time to recovery (MTTR)? If so, architect your applications for resiliency and distribute them to withstand outages. To achieve higher levels of availability, this distribution should span different physical locations. Architect individual layers (e.g., web server, database) for resiliency, which includes monitoring, self-healing, and notification of significant event disruption and failure.

Best practices:

- **Load Balancing** Use a load balancer in front of a pool of resources.
- **Multi-AZ /Region** Distribute applications across multiple Availability Zones /regions.
- **Auto Healing** Use automated capabilities to detect failures and perform an action to remediate.
- **Monitoring** Continuously monitor the health of your system.
- **Notification** Plan to receive notifications of any significant events.

REL 9. How are you planning for recovery?

Data recovery is critical should restoration of data be required from backup methods. Your definition of and execution on the objectives, resources, locations, and functions of this data must align with RTO and RPO objectives.

Best practices:

- **Objectives Defined** Define RTO and RPO.
- **Disaster Recovery** Establish a DR strategy.
- **Configuration Drift** Ensure that Amazon Machine Images (AMIs) and the system configuration state are up-to-date at the DR site/region.
- **Service Limits** Request an increase of service limits with the DR site to accommodate a failover.
- **DR Tested and Validated** Regularly test failover to DR to ensure RTO and RPO are met.
- **Automated Recovery Implemented** Use AWS and/or third-party tools to automate system recovery.

Performance Pillar

PERF 1. How do you select the appropriate instance type for your system?

Amazon EC2 offers a wide selection of instance types optimized to fit different use cases. Instance types are composed of varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload. AWS supports instance-less architectures, such as AWS Lambda, that can radically change the performance efficiency of a workload.

Best practices:

- **Policy/Reference Architecture** Select instance type and size based on predicted resource needs with internal governance standards.
- **Cost/Budget** Select instance type and size based on predicted resource needs with internal cost controls.

- **Benchmarking** Load test a known workload on AWS and use that to estimate the best selection – testing a known performance benchmark vs. a known workload.
- **Guidance from AWS or from a member of the AWS Partner Network (APN)** Make your selections based on best practice advice.
- **Load Test** Deploy the latest version of your system on AWS using different instance types and sizes, use monitoring to capture performance metrics, and then make a selection based on a calculation of performance/cost.

PERF 2. How do you ensure that you continue to have the most appropriate instance type as new instance types and features are introduced?

AWS listens to customer feedback and continues to innovate with new instance types and sizes, providing new combinations of CPU, memory, storage, and networking capacity. This means that a new instance type might be released that offers better performance efficiency than the one you originally selected.

Best practices:

- **Review** Cyclically reselect new instance types and sizes based on predicted resource needs.
- **Benchmarking** After each new instance type is released, carry out a load test of a known workload on AWS, and use that to estimate the best selection.
- **Load Test** After each relevant new instance type is released, deploy the latest version of the system on AWS, use monitoring to capture performance metrics and then make a selection based on a calculation of performance/cost.

PERF 3. How do you monitor your instances post-launch to ensure they are performing as expected?

System performance can degrade over time due to internal and/or external factors. Monitoring the performance of systems allows you to identify this

degradation and remediate internal or external factors (such as the OS or application load).

Best practices:

- **Amazon CloudWatch monitoring** Use CloudWatch to monitor instances.
- **Third-party monitoring** Use third-party tools to monitor systems.
- **Periodic review** Periodically review your monitoring dashboards.
- **Alarm-based notifications** Receive an automatic alert from your monitoring systems if metrics are out of safe bounds.
- **Trigger-based actions** Alarms cause automated actions to remediate or escalate issues.

PERF 4. How do you ensure that the quantity of your instances matches demand?

The amount of demand placed on a system often varies over different cycles: product lifecycle, such as launch or growth; temporal cycles such a time of day, day of the week, or month; unpredictable cycles such as social media visibility; and predictable cycles such as television episodes. Insufficient instances to meet your workload can degrade user experience and, at worst, lead to system failure.

Best practices:

- **Planned** Plan based upon metrics and/or planned events.
- **Automated - Scripted** Use tools for automatic management.
- **Automated - Auto Scaling** Use Auto Scaling for automatic management.

PERF 5. How do you select the appropriate storage solution for your system?

AWS is designed to provide low-cost data storage with high durability and availability. AWS offers storage choices for backup, archiving, and disaster recovery, as well as block, file, and object storage.

Best practices:

- **Policy/Reference Architecture** Select instance type and size based on predicted resource need with internal governance standard.
- **Cost/Budget** Selecting instance type and size based on predicted resource need based on internal cost controls.
- **Benchmarking** Load test a known workload on AWS and use that to estimate the best selection – testing a known performance benchmark vs. a known workload.
- **Guidance from AWS or from an APN Partner** Select a solution based on best practice advice.
- **Load Test** Deploy the latest version of your system on AWS using different instance types and sizes; use monitoring to capture performance metrics; and then make selection based on a calculation of performance/cost.

PERF 6. How do you ensure that you continue to have the most appropriate storage solution as new storage solutions and features are launched?

AWS listens to customer feedback and continues to innovate with new storage solution and features, providing new combinations of capacity, throughput, and durability. This means that a new storage solution might be released that offers better performance efficiency that the one you originally selected.

Best practices:

- **Review** Cyclically reselect new instance type and size based on predicted resource need.
- **Benchmarking** After each new instance type is released, carry out a load test of a known workload on AWS, and use that to estimate the best selection.
- **Load Test** After each relevant new instance type is released deploy the latest version of the system on AWS, use monitoring to capture performance metrics, and then make a selection based on a calculation of performance/cost.

PERF 7. How do you monitor your storage solution to ensure it is performing as expected?

System performance can degrade over time, or for periods of time, due to internal or external factors. Monitoring the performance of systems allows you to identify this degradation and remediate the internal or external factors.

Best practices:

- **Amazon CloudWatch monitoring** Use CloudWatch to monitor storage systems.
- **Third party monitoring** Use third party tools to monitor storage systems.
- **Periodic review** Periodically review your monitoring dashboards.
- **Alarm-based review** Plan for your monitoring systems to automatically alert you if metrics are out of safe bounds.
- **Trigger-based actions** Plan for alarms to cause automated actions to remediate or escalate issues.

PERF 8. How do you ensure that the capacity and throughput of your storage solutions matches demand?

The amount of demand placed on a system often varies over different cycles: product lifecycle, such as launch or growth; temporal cycles such a time of day, day of the week, or month; unpredictable cycles such social media visibility; and predictable cycles such as television episodes. Insufficient storage capacity or throughput to your workload can degrade user experience and, at worst, lead to system failure.

Best practices:

- **Reactive** Manage manually based on metrics.
- **Planned** Plan future capacity and throughput based on metrics and/or planned events.
- **Automated** Automate against metrics.

PERF 9. How do you select the appropriate database solution for your system?

The optimal database solution for a particular system can vary based on requirements for consistency, availability, partition tolerance, and latency. Many systems use different database solutions for different sub-systems and enable different features to improve performance. Selecting the wrong database solution and features for a systems workload can lead to lower performance efficiency.

Best practices:

- **Policy/Reference Architecture** Select instance type and size based on predicted resource needs based on an internal governance standard.
- **Cost/Budget** Select instance type and size based on predicted resource needs based on internal cost controls.
- **Benchmarking** Load test a known workload on AWS and use that to estimate the best selection – testing a known performance benchmark vs. a known workload.
- **Guidance from AWS or from an APN Partner** Select a solution based on best practice advice.
- **Load Test** Deploy the latest version of your system on AWS using different instance types and sizes, use monitoring to capture performance metrics, and then make a selection based on a calculation of performance/cost.

PERF 10. How do you ensure that you continue to have the most appropriate database solution and features as new database solution and features are launched?

AWS listens to customer feedback and continues to innovate with new database solutions and features, providing new combinations of consistency, availability, partition tolerance, and latency. This means that a new database solution or feature might be released that offers better performance efficiency than the one you originally selected.

Best practices:

- **Review** Cyclically reselect new instance type and size based on predicted resource need.

- **Benchmarking** After each new instance type is released, carry out a load test of a known workload on AWS, and use that to estimate the best selection.
- **Load Test** After each relevant new instance type is released, deploy the latest version of the system on AWS, use monitoring to capture performance metrics, and then make a selection based on a calculation of performance/cost.

PERF 11. How do you monitor your databases to ensure performance is as expected?

System performance can degrade over time due to internal or external factors. Monitoring the performance of systems allows you to identify this degradation and remediate the internal or external factors.

Best practices:

- **Amazon CloudWatch monitoring** Use CloudWatch to monitor databases.
- **Third-party monitoring** Use third party tools to monitor databases.
- **Periodic review** Periodically review your monitoring dashboards.
- **Alarm-based notifications** Plan to have your monitoring systems automatically alert you if metrics are out of safe bounds.
- **Trigger-based actions** Plan to have alarms cause automated actions to remediate or escalate issues.

PERF 12. How do you ensure the capacity and throughput of your databases matches demand?

The amount of demand placed on a system often varies over different cycles: product lifecycle such as launch, growth, etc.; temporal cycles such a time of day, weekday or month, etc.; unpredictable cycles such as seen with social media; and predictable cycles such as television episodes. Having insufficient database capacity and throughput to meet workload can degrade user experience and, at its worst, lead to system failure.

Best practices:

- **Planned** Plan for future capacity and throughput based on metrics and/or planned events.
- **Automated** Automate against metrics.

PERF 13. How do you select the appropriate proximity and caching solutions for your system?

Physical distance, network distance, or long running requests can introduce system delays. Unaddressed latency can tie up system resources for longer than required, and introduce both internal and external performance degradation. To reduce latency, consider the end-to-end performance of your entire system from the end-user's perspective, and look for opportunities to adjust the physical proximity of your resources or cache solutions.

Best practices:

- **Policy/Reference Architecture** Select instance type and size based on predicted resource need based on an internal governance standard.
- **Cost/Budget** Selecting instance type and size based on predicted resource need based on internal cost controls.
- **Benchmarking** Load test a known workload on AWS and use that to estimate the best selection; testing a known performance benchmark vs. a known workload.
- **Guidance from AWS or from an APN Partner** Select a proximity and caching solution based on best practice advice.
- **Load Test** Deploy the latest version of your system on AWS using different instance types and sizes, use monitoring to capture performance metrics, and then make a selection based on a calculation of performance/cost.

PERF 14. How do you ensure you continue to have the most appropriate proximity and caching solutions as new solutions are launched?

AWS listens to customer feedback and continues to innovate with new proximity and caching solutions, providing new combinations of proximity, caching, and latency. This means that new proximity and caching solutions might be released that offer better performance efficiency than the one you

originally selected. Look for opportunities to reduce latency and increase performance throughout the system. For example, did you complete a one-time optimization or are you continuing to optimize your system as demand changes over time?

Best practices:

- **Review** Cyclically reselect a new instance type and size based on predicted resource needs.
- **Benchmarking** After each new instance type is released, carry out a load test of a known workload on AWS, and use that to estimate the best selection.
- **Load Test** After each relevant new instance type is released deploy the latest version of the system on AWS, use monitoring to capture performance metrics, and then select based on a calculation of performance/cost.
- **Proactive Monitoring–Amazon Cloud Watch monitoring** Use Amazon CloudWatch to monitor proximity and caching solutions.
- **Proactive Monitoring–Third-party monitoring** Use third-party tools to monitor proximity and caching solutions.
- **Alarm-based notification** Plan for your monitoring systems to automatically alert you if metrics are out of safe bounds.
- **Trigger-based actions** Plan for alarms to cause automated actions to remediate or escalate issues.

PERF 15. How do you monitor your proximity and caching solutions to ensure performance is as expected?

System performance can degrade over time due to internal or external factors. Monitoring the performance of systems allows you to identify this degradation and remediate the internal or external factors.

Best practices:

- **Amazon CloudWatch monitoring** Use CloudWatch to monitor instances.
- **Third-party monitoring** Use third-party tools to monitor systems.
- **Periodic review** Periodically review your monitoring dashboards.

- **Alarm-based notifications** Plan for your monitoring systems to automatically alert you if metrics are out of safe bounds.
- **Trigger-based actions** Plan for alarms to cause automated actions to remediate or escalate issues.

PERF 16. How do you ensure the proximity and caching solutions you have matches demand?

The amount of demand placed on a system often varies over different cycles: product lifecycle such as launch, growth, etc.; temporal cycles such a time of day, weekday or month, etc.; unpredictable cycles such as seen with social media; and predictable cycles such as television episodes. Having the wrong proximity and caching solutions to meet workload can degrade user experience and, at its worst, lead to system failure. This is especially true if you have, or plan to have, a global user base.

Best practices:

- **Planned** Plan future proximity or caching solutions based on metrics and/or planned events.
- **Monitor** Monitor cache usage and demand over time.
- **Periodic Review** Review cache usage and demand over time.

Cost Optimization Pillar

COST 1. How do you make sure your capacity matches but does not substantially exceed what you need?

For an architecture that is balanced in terms of spend and performance, ensure that everything you pay for is used and avoid significantly underutilized instances. A skewed utilization metric in either direction will have an adverse impact on your business in either operational costs (degraded performance due to over-utilization) or wasted AWS expenditures (due to over-provisioning).

Best practices:

- **Demand-based approach** Use Auto Scaling to respond to variable demand.

- **Queue-based approach** Run your own Amazon Simple Queue Service (SQS) queue and spin up/shut down instances based on demand.
- **Time-based approach** Examples: follow the sun, turn off Dev/Test instances over the weekend, follow quarterly or annual schedules (e.g., Black Friday).
- **Appropriately provisioned** Appropriately provision throughput, sizing, and storage for services such as Amazon DynamoDB, Amazon EBS (provisioned IOPS), Amazon RDS, Amazon EMR, etc.

COST 2. How are you optimizing your usage of AWS services?

If you use application-level services, make sure that you use them well. For example, introduce lifecycle policies to control Amazon S3 usage or leverage services such as Amazon RDS and Amazon DynamoDB enable tremendous flexibility. Checks for appropriate usage include verifying multi-AZ deployments for Amazon RDS or verifying that provisioned IOPS are applicable in your Amazon DynamoDB tables.

Best practices:

- **Service-specific optimizations** Examples include minimizing I/O for Amazon EBS; avoiding uploading too many small files into Amazon S3; using Spot instances extensively for Amazon EMR; etc.

COST 3. Have you selected the appropriate resources to meet your cost targets?

Ensure that the Amazon EC2 instances you select are appropriate to the task at hand. AWS encourages the use of benchmarking assessments to ensure that the instance type you chose is optimized for its workload.

Best practices:

- **Match instance profile based on need** For example, match based on workload and instance description –compute, memory, or storage intensive.
- **Third-party products** For example, use third-party products such as CopperEgg or New Relic to determine appropriate instance types.
- **Amazon CloudWatch** Use CloudWatch to determine processor load.

- **Custom Metrics** Load custom memory scripts and inspect memory usage using CloudWatch.
- **Profiled Applications** Profile your applications so you know when to use which type of Amazon EBS (magnetic, general purpose (SSD), provisioned IOPS). Use EBS-Optimized instances only when necessary.

COST 4. Have you selected the appropriate pricing model to meet your cost targets?

Use the pricing model most appropriate for your workload to minimize expense. The optimal deployment could be fully On-Demand instances, a mix of On-Demand and Reserved Instances, or you might include Spot instances, where applicable.

Best practices:

- **Spot** Use Spot instances for select workloads.
- **Analyze Usage** Regularly analyze usage and purchase Reserved Instances accordingly.
- **Sell Reserved Instances** As your needs change, sell Reserved Instances you no longer need on the Reserved Instances Marketplace, and purchase others.
- **Automated Action** Have your architecture allows you to turn off unused instances (e.g., use Auto Scaling to scale down during non-business hours).
- **Consider Cost** Factor costs into region selection.

COST 5. Are there managed services (higher-level services than Amazon EC2, Amazon EBS, Amazon S3) you can use to improve your ROI?

Amazon EC2, Amazon EBS, and Amazon S3 are “building-block” AWS services. Managed services such as Amazon RDS and Amazon DynamoDB are “higher level” AWS services. By using these managed services, you can reduce or remove much of your administrative and operational overhead, freeing you to work on applications and business-related activities.

Best practices:

- **Analyze Services** Analyze application-level services to see which ones you can use.

- **Consider appropriate databases** Use Amazon Relational Database Service (RDS) (Postgres, MySQL, SQL Server, Oracle Server) or Amazon DynamoDB (or other key-value stores, NoSQL alternatives) where it's appropriate.
- **Consider other application level services** Use Amazon Simple Queue Service (SQS), Amazon Simple Notification Service (SNS), Amazon Simple Email Service (SES) where appropriate.
- **Consider AWS CloudFormation, AWS Elastic Beanstalk, or AWS Opsworks** Use AWS CloudFormation templates / AWS Elastic Beanstalk /AWS OpsWorks to achieve the benefits of standardization and cost control.

COST 6. What access controls and procedures do you have in place to govern AWS usage?

Establish policies and mechanisms to make sure that appropriate costs are incurred while objectives are achieved. By employing a checks-and-balances approach through tagging and IAM controls, you can innovate without overspending.

Best practices:

- **Establish groups and roles** (Example: Dev/Test/Prod); use AWS governance mechanisms such as IAM to control who can spin up instances and resources in each group. (This applies to AWS services or third-party solutions.)
- **Track project lifecycle** Track, measure, and audit the life cycle of projects, teams, and environments to avoid using and paying for unnecessary resources.

COST 7. How are you monitoring usage and spending?

Establish policies and procedures to monitor, control, and appropriately assign your costs. Leverage AWS-provided tools for visibility into who is using what—and at what cost. This will provide you with a deeper understanding of your business needs and your teams' operations.

Best practices:

- **Tag all resources** To be able to correlate changes in your bill to changes in our infrastructure and usage.

- **Review Detailed Billing Reports** Have a standard process to load and interpret the Detailed Billing Reports.
- **Cost-efficient architecture** Have a plan for both usage and spending (per unit – e.g., user, gigabyte of data).
- **Monitoring** Monitor usage and spend regularly using Amazon CloudWatch or a third-party provider (examples: Cloudability, CloudCheckr).
- **Notifications** Let key members of our team know if our spending moves outside well-defined limits.
- **Use AWS Cost Explorer**
- **Finance driven charge back method** Use this to allocate instances and resources to cost centers (e.g., tagging).

COST 8. Do you decommission resources that you no longer need or stop resources that are temporarily not needed?

Ensure that you only pay for services that are being used. Implement change control and resource management from project inception to end-of-life so that you can identify necessary process changes or enhancements where appropriate. Work with AWS Support for recommendations on how to optimize your project for your workload: for example, when to use Auto Scaling, AWS OpsWorks, AWS Data Pipeline, or the different Amazon EC2-provisioning approaches.

Best practices:

- Design your system to gracefully handle instance termination as you identify and decommission non-critical or unrequired instances or resources with low utilization.
- Have a process in place to identify and decommission orphaned resources.
- Reconcile decommissioned resources based on either system or process.

COST 9. Did you consider data-transfer charges when designing your architecture?

Ensure that you monitor data-transfer charges so that you can make architectural decisions that might alleviate some of these costs. For example, if you are a content provider and have been serving content directly from an Amazon S3 bucket to your end users, you might be able to significantly reduce your costs if you push your content to the Amazon CloudFront CDN.

Remember that a small yet effective architectural change can drastically reduce your operational costs.

Best practices:

- Use a CDN
- Architect to optimize data transfer (application design, WAN acceleration, etc.).
- Analyze the situation and use AWS Direct Connect to save money and improve performance.
- Balance the data transfer costs of your architecture with your high availability (HA) and reliability needs.

COST 10. How do you manage and/or consider the adoption of new services?

At AWS, our goal is to help you architect as optimally and cost effectively as possible. New services and features might directly reduce your costs. A good example of this is Amazon Glacier, which offers a low-cost, "cold" storage solution for data that is infrequently accessed, yet must be retained for business or legal reasons. Another example is Reduced Redundancy Storage for Amazon S3, which allows you to elect to have fewer copies of your Amazon S3 objects (lower levels of redundancy) for a reduced price. There are implications to consider when making these decisions, for example: "What does it mean to have fewer copies of my data?" or "Will I need to access this data more than I realize?"

Best practices:

- Meet regularly with your AWS Solutions Architect, Consultant, or Account Team, and consider which new services or features you could adopt to save money.