AWS Well-Architected Framework

November 2016

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 review 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 five conceptual areas that we define as the *pillars* of the Well-Architected Framework.

Introduction

At Amazon Web Services (AWS) we understand the value of educating our customers on architectural best practices for designing and operating 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 experience, 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 AWS 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 and operating 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 five pillars—security, reliability, performance efficiency, cost optimization, and operational excellence.

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.
Operational Excellence	The ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.

When architecting solutions you make trade-offs between pillars based upon your business context, these business decisions can drive your engineering priorities. You might optimize to reduce cost at the expense of reliability in development environments, or for mission-critical solutions you might optimize reliability with increased costs. In ecommerce solutions, performance can affect revenue and customer propensity to buy. Security and Operational Excellence are generally not traded-off against other pillars.

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 the cloud, you can create a production-scale test 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.
- **Automate to make architectural experimentation easier:** Automation allows you to create and replicate your systems at low cost and avoid the expense of 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 events, with a few major versions of a system during its lifetime. As a business and its context continue to change, these initial decisions might 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 innovations as a standard practice.
- **Data-Driven architectures:** In the cloud you can collect data on how your architectural choices affect the behavior of your workload. This lets you make fact-based decisions on how to improve your workload. Your cloud infrastructure is code, so you can use that data to inform your architecture choices and improvements over time.
- **Improve through game days:** Test how your architecture and processes perform by regularly scheduling game days to simulate events in production. This will help you understand where improvements can be made and can help develop organizational experience in dealing with events.

The Five Pillars of the Well-Architected Framework

Creating a software system is a lot like constructing a building. If the foundation is not solid structural problems could undermine the integrity and function of the building. When architecting technology solutions, if you neglect the five pillars of security, reliability, performance efficiency, cost optimization, and operational excellence 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 five pillars, and includes definitions, best practices, questions, considerations, and key AWS services that are relevant.

Security Pillar

The **Security** pillar includes 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 running security appliances (e.g., firewalls) only 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.
- **Implement a principle of least privilege:** Ensure that authorization is appropriate for each interaction with your AWS resources and implement strong logical access controls directly on resources.

- **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 patched, hardened image of a virtual server, and then use that image automatically on each new server you launch. Create an entire trust zone architecture that is defined and managed in a template via revision control. Automate the response to both routine and anomalous security events.

Definition

There are five best practice areas for Security in the cloud:

1. Identity and access management
2. Detective controls
3. Infrastructure protection
4. Data protection
5. Incident response

Before you architect any system, you need to put in place practices that influence security. You will want to control who can do what. In addition, you want to be able to identify security incidents, protect your systems and services, and maintain the confidentiality and integrity of data through data protection. You should have a well-defined and practiced process for responding to security incidents. These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

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

Identity and Access Management

Identity and access management are key parts of an information security program, ensuring that only authorized and authenticated users are able to access your resources, and only in a manner that is intended. For example, you'll define principals (users, groups, services, and roles that take action in your account), build out policies aligned with these principals, and implement strong credential management. These privilege-management elements form the core concepts of 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 level, avoiding re-use, and using multi-factor authentication (MFA). You can use federation with your existing directory service. For workloads that require systems to have access to AWS, IAM enables secure access through instance profiles, identity federation, and temporary credentials.

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

SEC 1. How are you protecting access to and use of the AWS root account credentials?

SEC 2. How are you defining roles and responsibilities of system users to control human access to the AWS Management Console and API?

SEC 3. How are you limiting automated access to AWS resources? (e.g., applications, scripts, and/or third-party tools or services)

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.

Detective Controls

You can use detective controls to identify a potential security incident. These controls are an essential part of governance frameworks, and can be used to support a quality process, a legal or compliance obligation, and for threat identification and response efforts. There are different types of detective controls. For example, conducting an inventory of 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 you can implement detective controls by processing logs, events and monitoring that allows for auditing, automated analysis, and alarming. AWS CloudTrail logs, AWS API calls, and Amazon CloudWatch provide monitoring of metrics with alarming, and AWS Config provides configuration history. Service level logs are also available, for example you can use Amazon Simple Storage Service (S3) to log access requests. Finally Amazon Glacier provides a 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 4. How are you capturing and analyzing logs?

Log management is important to a well-architected design for reasons ranging from security/forensics to regulatory or legal requirements. It is critical that you

analyze logs and respond to them, so that you can identify potential security incidents. 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.

Infrastructure Protection

Infrastructure protection includes control methodologies, such as defense in depth and multi-factor authentication, which are 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 by 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 5. How are you enforcing network and host-level boundary protection?

SEC 6. How are you leveraging AWS service level security features?

SEC 7. 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, 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.

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 and *encryption* protects data by 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.

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 that contains important content, such as file access and changes, is available.
- 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:

SEC 8. How are you classifying your data?

SEC 9. How are you encrypting and protecting your data at rest?

SEC 10. How are you managing keys?

SEC 11. 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.

Incident response

Even with extremely mature preventive and detective controls, organizations should still put processes in place to respond to and mitigate the potential impact of security incidents. The architecture of your workload will strongly affect the ability of your teams to operate effectively during an incident to isolate or contain systems and to restore operations to a known-good state. Putting in place the tools and access ahead of a security incident, then routinely practicing incident response will make sure the architecture is updated to accommodate timely investigation and recovery.

In AWS, the following practices facilitate effective incident response:

- Detailed logging is available that contains important content, such as file access and changes.

- Events can be automatically processed and trigger scripts that automate run books through the use of AWS APIs.
- You can pre-provision tooling and a "clean room" using AWS CloudFormation. This allows you to carry out forensics in a safe, isolated environment.

The following questions focus on considerations for incident response:

SEC 12. How do you ensure you have the appropriate incident response?

Ensure that you have a way to quickly grant access for your InfoSec team, and automate the isolation of instances as well at the capturing of data and state for forensics.

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:

Identity and access 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.

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.

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.

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.

Incident response: IAM should be used to grant appropriate authorization to incident response teams. Amazon CloudFormation can be used to create a trusted environment for conducting investigations.

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 includes 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.
- **Manage change in automation**: Changes to your infrastructure should be done using automation. The changes that need to be managed are changes to the automation.

Definition

There are three best practice areas for Reliability in the cloud:

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:

REL 1. How do you manage AWS service limits for your accounts?

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

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.

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 3. How does your system adapt to changes in demand?

REL 4. How are you monitoring AWS resources?

REL 5. How are you executing change?

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 6. How are you backing up your data?

REL 7. How does your system withstand component failures?

REL 8. How are you testing for resiliency?

REL 9. How are you planning for disaster 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 resiliency (especially under failure-testing scenarios). Tracking KPIs will 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 provides templates for the 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
- Service Limit Reports Blog

Whitepapers

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

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 at minimal cost.
- **Use *serverless* 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.

- **Mechanical sympathy:** Use the technology approach that aligns best to what you are trying to achieve. For example consider data access patterns when selecting database or storage approaches.

Definition

There are four best practice areas for Performance Efficiency in the cloud:

1. Selection (compute, storage, database, network)
2. Review
3. Monitoring
4. Tradeoffs

Take a data-driven approach to selecting a high performance architecture. Gather data on all aspects of the architecture, from the high level design to the selection and configuration of resource types. By reviewing your choices on a cyclical basis, you will ensure that you are taking advantage of the continually evolving AWS platform. Monitoring will ensure that you are aware of any deviance from expected performance and can take action on it. Finally, your architecture can make tradeoffs to improve performance, such as using compression or caching, or relaxing consistency requirements.

Best Practices

Selection

The optimal solution for a particular system will vary based on the kind of workload you have, often with multiple approaches combined. Well-architected systems use multiple solutions and enable different features to improve performance.

In AWS, resources are virtualized and are available in a number of different types and configurations. This makes it easier to find an approach that closely matches with your needs, and you can also find options that are not easily achievable with on-premises infrastructure. For example, a managed service such as Amazon DynamoDB provides a fully managed NoSQL database with single-digit millisecond latency at any scale.

The following example question focuses on selection considerations:

PERF 1. How do you select the best performing architecture?

When you select the patterns and implementation for your architecture use a data-driven approach for the most optimal solution. AWS Solutions Architects, AWS Reference Architectures, and AWS Partners can help you select an architecture based on what we have learned, but data obtained through benchmarking or load testing will be required to optimize your architecture.

Your architecture will likely combine a number of different architectural approaches (e.g., event driven, ETL, or pipeline). The implementation of your architecture will use the AWS services that are specific to the optimization of your architecture's performance. In the following sections we look at the four main resource types that you should consider (compute, storage, database, and network).

Compute

The optimal compute solution for a particular system may vary based on application design, usage patterns, and configuration settings. Architectures may use different compute solutions for various components and enable different features to improve performance. Selecting the wrong compute solution for an architecture can lead to lower performance efficiency.

In AWS, compute is available in three forms: instances, containers, and functions:

- **Instances** are virtualized servers and, therefore, you can change their capabilities with the click of a button or an API call. Because in the cloud 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, and they offer a wide variety of capabilities, including solid state drives (SSDs) and graphics processing units (GPUs).
- **Containers** are a method of operating system virtualization that allow you to run an application and its dependencies in resource-isolated processes.

- **Functions** abstract the execution environment from the code you want to execute. For example, AWS Lambda allows you to execute code without running an instance.

The following example question focuses on compute considerations:

PERF 2. How do you select your compute solution?

When architecting your use of compute you should take advantage of the elasticity mechanisms available to ensure you have sufficient capacity to sustain performance as demand changes.

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 question focuses on storage considerations for performance efficiency:

PERF 3. How do you select your storage solution?

When you select a storage solution ensuring that it aligns with your access patterns will be critical to achieving the performance you want.

Database

The optimal database solution for a particular system can vary based on requirements for availability, consistency, partition tolerance, latency, durability, scalability and query capability. Many systems use different database solutions for various subsystems 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. 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 question focuses on database considerations for performance efficiency:

PERF 4. How do you select your database solution?

Although a workload's database approach (RDBMS, NoSQL, etc.) has significant impact on performance efficiency, it is often an area that is chosen according to organizational defaults rather than through a data-driven approach. As with storage, it is critical to consider the access patterns of your workload, and also to consider if other non-database solutions could solve the problem more efficiently (such as a using a search engine or data warehouse).

Network

The optimal network solution for a particular system will vary based on latency, throughput requirements, and so on. Physical constraints such as user or on-premises resources will drive location options, which can be offset using edge techniques or resource placement.

In AWS, networking is virtualized and is available in a number of different types and configurations. This makes it easier to match your networking methods more closely with your needs. AWS offers product features (e.g., very high network instance types, Amazon EBS optimized instances, Amazon S3 transfer acceleration, dynamic Amazon CloudFront) to optimize network traffic. AWS also offers networking features (e.g., Amazon Route53 latency routing, Amazon VPC endpoints, and AWS Direct Connect) to reduce network distance or jitter.

The following example question focuses on storage considerations for performance efficiency:

PERF 5. How do you select your network solution?

When you select your network solution, you need to consider location. With AWS you can choose to place resources close to where they will be used to reduce distance. By taking advantage of regions, placement groups, and edge locations you can significantly improve performance.

Review

When architecting solutions, there is a finite set of options that you can choose from. However, over time new technologies and approaches become available that could improve the performance of your architecture.

Using AWS, you can take advantage of our continual innovation, which is driven by customer need. We release new regions, edge location, services, and features regularly. Any of these could positively improve the performance efficiency of your architecture.

The following example question focuses on reviewing performance efficiency:

PERF 6. How do you ensure that you continue to have the most appropriate resource type as new resource types and features are introduced?

Understanding where your architecture is performance constrained will allow you to look out for releases that could alleviate that constraint.

Monitoring

Once you have implemented your architecture you will need to monitor its performance so that you can remediate any issues before your customers are aware. Monitoring metrics should be used to raise alarms when thresholds are breached. The alarm can trigger automated action to work around any badly performing components.

Using AWS, Amazon CloudWatch provides the ability to monitor and send notification alarms, and you can use automation to work around performance issues by triggering actions through Amazon Kinesis, Amazon Simple Queue Service (SQS), and AWS Lambda.

The following example question focuses on monitoring performance efficiency:

PERF 7. How do you monitor your resources post-launch to ensure they are performing as expected?

Ensuring that you do not see too many false positives, or are overwhelmed with data, is key to having an effective monitoring solution. Automated triggers avoid human error and can reduce the time to fix problems. Plan for “game days” where simulations are conducted in the production environment, to test your alarming solution and ensure that it correctly recognizes issues.

Tradeoffs

When you architect solutions, think about trade-offs so you can select an optimal approach. Depending on your situation you could trade consistency, durability, and space versus time or latency, to deliver higher performance.

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. AWS also offers caching solutions such as Amazon ElastiCache, which provides an in-memory data store or cache, and Amazon CloudFront, which caches copies of your static content closer to end-users.

The following example question focuses on space-time trade-offs for Performance Efficiency:

PERF 8. How do you use tradeoffs to improve performance?

Trade-offs can increase the complexity of your architecture, and require load testing to ensure that a measurable benefit is obtained.

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 areas of performance efficiency:

Selection:

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 provisioned input/output operations per second (PIOPS)) that allow you to optimize for your use case. Amazon S3 provides serverless content delivery and Amazon S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances.

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.

Network: Amazon Route 53 provides latency-based routing. Amazon VPC endpoints and Direct Connect can reduce network distance or jitter.

Review: The AWS Blog and the What's New section on the AWS website are resources for learning about newly launched features and services.

Monitoring: Amazon CloudWatch provides metrics, alarms, and notifications that you can integrate with your existing monitoring solution, and that you can use with AWS Lambda to trigger actions.

Tradeoff: Amazon ElastiCache, Amazon CloudFront, and AWS Snowball are services that allow you to improve performance. Read replicas in Amazon RDS can allow you to scale read-heavy workloads.

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
- Amazon EBS Volume Performance

Cost Optimization Pillar

The **Cost Optimization** pillar includes the continual process of refinement and improvement of a system over its entire lifecycle. From the initial design of your very first proof of concept to the ongoing operation of production workloads, adopting the practices in this paper will enable you to build and operate cost-aware systems that achieve business outcomes and minimize costs, thus allowing your business to maximize its return on investment.

Design Principles

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

- **Adopt a consumption model:** Pay only for the computing resources that you consume and increase or decrease usage depending on business requirements, not by using elaborate forecasting. For example, development and test environments are typically only used for eight hours a day during the work week. You can stop these resources when they are not in use for a potential cost savings of 75 percent (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.
- **Analyze and attribute expenditure:** The cloud makes it easier to accurately identify the usage and cost of systems, which then allows transparent attribution of IT costs to individual business owners. This helps measure return on investment (ROI) and gives system owners an opportunity 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 like sending email or managing databases. And because managed services operate at cloud scale, they can offer a lower cost per transaction or service.

Definition

There are four best practice areas for Cost Optimization in the cloud:

1. Cost-effective resources
2. Matching supply with demand

3. Expenditure awareness
4. Optimizing over time

As with the other pillars, there are trade-offs to consider. For example, do you want 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

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 offer 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 1. Are you considering cost when you select AWS services for your solution?

COST 2. Have you sized your resources to meet your cost targets?

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

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.

Matching 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 demand, buffer, and time 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 question focus on matched supply and demand for cost optimization:

COST 4. How do you make sure your capacity matches but does not substantially exceed what you need?

When architecting to match supply against demand, you will want to actively

think about the patterns of usage and the time it takes to provision new resources.

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

COST 6. How are you monitoring usage and spending?

COST 7. Do you decommission resources that you no longer need or stop resources that are temporarily not needed?

COST 8. What access controls and procedures do you have in place to govern AWS usage?

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.

Combining tagged resources with entity lifecycle tracking (employees, projects) makes it possible 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 review 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 9. How do you manage and/or consider the adoption of new services?**

By regularly reviewing your deployments, it is often possible to utilize new AWS services to lower your costs. Also, assess how newer services can 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:

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.

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

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

Optimizing over time: The AWS Blog and the *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 Cloud Economics Center

Tools

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

Operational Excellence Pillar

The **Operational Excellence** pillar includes operational practices and procedures used to manage production workloads. This includes how planned changes are executed, as well as responses to unexpected operational events. Change execution and responses should be automated. All processes and procedures of operational excellence should be documented, tested, and regularly reviewed.

Design Principles

In the cloud, there are a number of principles that drive operational excellence:

- **Perform operations with code:** When there are common repetitive processes or procedures, use automation. For example, consider automating configuration management, changes, and responses to events.
- **Align operations processes to business objectives:** Collect metrics that indicate operational excellence in meeting business objectives. The goal should be to reduce the signal to noise ratio in metrics, so operational monitoring and responses are targeted to support business-critical needs. Collecting metrics that are unnecessary will prevent effective responses to unexpected operational events by complicating monitoring and response.
- **Make regular, small, incremental changes:** Workloads should be designed to allow components to be updated regularly. Changes should be done in small increments, not large batches, and should be able to be rolled back without affecting operations. Put operations procedures in place to allow for the implementation of those changes without downtime for maintenance or the replacement of dependent service components.
- **Test for responses to unexpected events:** Workloads should be tested for component failures and other unexpected operational events. It is important to test and understand procedures for responding to operational events, so that they are followed when operational events occur. Set up game days so you can test responses to simulated operational events and failure injections.

- **Learn from operational events and failures:** Processes should be in place so that all types of operational events and failures are captured, reviewed, and then used for improvements. Regular cross-functional operations reviews should result in process improvements that drive operational excellence.
- **Keep operations procedures current:** Process and procedure guides should be adapted as environments and operations evolve. This includes updating regular operations runbooks (standard operations procedures), as well as playbooks (response plans for unexpected operational events or production failures). Guidance and learnings for operations should be shared between teams to prevent repeated mistakes. Consider using a wiki or an internal knowledge base for this information. The information that should be evaluated includes operations metrics, unexpected anomalies, failed deployments, system failures, and ineffective or inappropriate responses to failures. System and architecture documentation should also be captured and updated using automation as environments and operations evolve.

Definition

There are three best practice areas for Operational Excellence in the cloud:

1. Preparation
2. Operations
3. Responses

To drive operational excellence, preparation is essential. Many operational issues can be avoided by following best practices when designing the workload, and fixes are less expensive to implement in design phases rather than in production. Operations process and procedures must be thoroughly planned, tested, reviewed. Workloads should evolve and be changed in automated and manageable ways. Changes should be small, frequent, and incremental, all without impacting continuous operations. Operations teams must be prepared to respond to operational events and failures, and have processes in place to learn from them.

Best Practices

Preparation

Effective preparation is required to drive operational excellence. Operations checklists will ensure that workloads are ready for production operation, and prevent unintentional production promotion without effective preparation. Workloads should have operations guidance that operations teams can refer to so they can perform normal daily tasks (runbooks), as well as guidance for responding to unexpected operational events (playbooks). Playbooks should include response plans, as well as escalation paths and stakeholder notifications. Routine reviews of business cycle events that can drive changes in operations should also be performed (e.g., marketing events, flash sales, etc.). All runbooks and playbooks should be tested so that gaps or challenges can be identified and any potential risk can be mitigated. Mechanisms to track and learn from failures should be in place. Environments, architecture, and the configuration parameters for resources within them, should be documented in a way that allows components to be easily identified for tracking and troubleshooting. Changes to configuration should also be trackable and automated.

In AWS there are several methods, services, and features that can be used to support operational readiness, and the ability to prepare for normal day-to-day operations as well as unexpected operational events. Operational readiness may still include manual cross-functional or peer reviews to ensure oversight. AWS services such as AWS CloudFormation can be used to ensure that environments contain all required resources when deployed in production, and that the configuration of the environment is based on tested best practices, which reduces the opportunity for human error. Implementing Auto Scaling, or other automated scaling mechanisms, will allow workloads to automatically respond when business-related events affect operational needs. Services like AWS Config with the AWS Config rules feature create mechanisms to automatically track and respond to changes in your AWS workloads and environments. It is also important to use features like tagging to make sure all resources in a workload can be easily identified when needed during operations and responses.

The following questions focus on preparation considerations for operational excellence:

OPS 1. What best practices for cloud operations are you using?

OPS 2. How are you doing configuration management for your workload?

Be sure that documentation does not become stale or out of date as procedures change. Also make sure that it is thorough. Without application designs, environment configurations, resource configurations, response plans, and mitigation plans, documentation is not complete. If documentation is not updated and tested regularly, it will not be useful when unexpected operational events occur. If workloads are not reviewed before production, operations will be affected when undetected issues occur. If resources are not documented, when operational events occur, determining how to respond will be more difficult while the correct resources are identified.

Operations

Operations should be standardized and manageable on a routine basis. The focus should be on automation, small frequent changes, regular quality assurance testing, and defined mechanisms to track, audit, roll back, and review changes. Changes should not be large and infrequent, they should not require scheduled downtime, and they should not require manual execution. A wide range of logs and metrics that are based on key operational indicators for a workload should be collected and reviewed to ensure continuous operations.

In AWS you can set up a continuous integration / continuous deployment (CI/CD) pipeline (e.g., source code repository, build systems, deployment and testing automation). Release management processes, whether manual or automated, should be tested and be based on small incremental changes, and tracked versions. You should be able to revert changes that introduce operational issues without causing operational impact. Change quality assurance should include risk mitigation strategies such as Blue/Green, Canary, and A/B testing. Operations checklists should be used to evaluate a workload's readiness for production. Aggregate logs for centralized monitoring and alerts. Make sure alerts trigger automated responses, including notification and escalations. Also design monitors for anomalies, not just failures.

The following questions focus on how you operate the workload for operational excellence:

OPS 3. How are you evolving your workload while minimizing the impact of change?

OPS 4. How do you monitor your workload to ensure it is operating as expected?

Routine operations, as well as responses to unplanned events, should be automated. Manual processes for deployments, release management, changes, and rollbacks should be avoided. Releases should not be large batches that are done infrequently. Rollbacks are more difficult in large changes, and failing to have a rollback plan, or the ability to mitigate failure impacts, will prevent continuity of operations. Align monitoring to business needs, so that the responses are effective at maintaining business continuity. Monitoring that is ad hoc and not centralized, with responses that are manual, will cause more impact to operations during unexpected events.

Responses

Responses to unexpected operational events should be automated. This is not just for alerting, but also for mitigation, remediation, rollback, and recovery. Alerts should be timely, and should invoke escalations when responses are not adequate to mitigate the impact of operational events. Quality assurance mechanisms should be in place to automatically roll back failed deployments. Responses should follow a pre-defined playbook that includes stakeholders, the escalation process, and procedures. Escalation paths should be defined and include both functional and hierarchical escalation capabilities. Hierarchical escalation should be automated, and escalated priority should result in stakeholder notifications.

In AWS there are several mechanisms to ensure both appropriate alerting and notification in response to unplanned operational events, as well as automated responses. Tools should also be in place to centrally monitor workloads and create effective alerts and notifications based on all available logs and metrics

that relate to key operational indicators. This includes alerts and notifications for out-of-bound anomalies, not just service or component failures. The responses should be enabled for the dependent AWS environment and services, as well as for the application health of the workload. Root cause analysis (RCA) should be performed after operational events, and used to improve both architecture and response plans.

The following questions focus on responding to events for operational excellence:

OPS 5. How do you respond to unplanned operational events?

OPS 6. How is escalation managed when responding to unplanned operational events?

If response plans are done in an ad-hoc way, and undefined, the results will be unpredictable, and often exacerbate the impact of an event. Responses that are based on out-of-date playbooks, or when a playbook is not accessible in the event of issues, will also have unpredictable results. If a process is not in place to review events, then future operational events will be harder to prevent, and also have the same impact.

Key AWS Services

There are two primary services that can be used to drive operational excellence. AWS CloudFormation can be used to create templates based on best practices, and provision resources in an orderly and predictable fashion. Amazon CloudWatch can be used for monitoring metrics, collecting logs, generating alerts, and triggering responses. Other services and features that support the three areas of Operational Excellence are as follows:

Preparation: AWS Config provides a detailed inventory of your AWS resources and configuration, and continuously records configuration changes. AWS Service Catalog helps to create a standardized set of service offerings that are aligned to best practices. Designing workloads that use automation with services like Auto Scaling, and Amazon SQS, are good methods to ensure continuous operations in the event of unexpected operational events.

Operations: AWS CodeCommit, AWS CodeDeploy, and AWS CodePipeline can be used to manage and automate code changes to AWS workloads. Use AWS SDKs or third-party libraries to automate operational changes. Use AWS CloudTrail to audit and track changes made to AWS environments.

Responses: Take advantage of all of the Amazon CloudWatch service features for effective and automated responses. Amazon CloudWatch alarms can be used to set thresholds for alerting and notification, and Amazon CloudWatch events can trigger notifications and automated responses.

Resources

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

Videos

- AWS re:Invent 2015 – DevOps at Amazon
- AWS re:Invent 2015 - Scaling Infrastructure Operations
- AWS Summit 2016 - DevOps, Continuous Integration and Deployment on AWS

Documentation and Blogs

- DevOps and AWS
- What is Continuous Integration
- What is Continuous Delivery
- AWS DevOps Blog

Whitepapers

- AWS Cloud Adoption Framework - Operations Perspective
- Introduction to DevOps on AWS
- AWS Operational Checklist

AWS Support

- AWS Cloud Operations Review
- AWS Premium Support
- AWS Trusted Advisor

Conclusion

The AWS Well-Architected Framework provides architectural best practices across five pillars for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud. The framework provides a set of questions that allows you to review 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, Sr. Manager Well-Architected, Amazon Web Services
- Erin Rifkin, Senior Product Manager, Amazon Web Services
- Max Ramsay, Principal Security Solutions Architect, Amazon Web Services
- Scott Paddock, Security Solutions Architect, Amazon Web Services
- John Steele, Sr. Technical Account Manager, Amazon Web Services
- Callum Hughes, Solutions Architect, Amazon Web Services

Document History

November 20, 2015. Updated the Appendix with current Amazon CloudWatch Logs information.

November 20, 2016. Updated the framework to include Operational Excellence pillar, and revise and update others pillars to reduce duplication and incorporate learnings from carrying out reviews with thousands of customers.

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

Identity and Access Management

SEC 1. 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 the root account 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:

- **MFA and Minimal Use of Root** The AWS root account credentials are only used for only minimal required activities.
- **No use of Root**

SEC 2. 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:

- **Employee Life-Cycle Managed** Employee life-cycle policies are defined and enforced.
- **Least Privilege** Users, groups, and roles are clearly defined and granted only the minimum privileges needed to accomplish business requirements.

SEC 3. How are you limiting automated access to AWS resources? (e.g., applications, scripts, and/or third-party tools or services)

Systematic access should be defined in similar ways because user groups are 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:

- **Static Credentials used for Automated Access** Stored these securely.
- **Dynamic Authentication for Automated Access** Manage using instance profiles or Amazon STS.

Detective Controls

SEC 4. How are you capturing and analyzing 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., CloudWatch Logs, Splunk, Papertrail, etc.) or stored in an Amazon S3 bucket for later processing based on business needs. Common sources of logs are AWS APIs 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. You can use Amazon CloudWatch Logs to monitor, store, and access your log files from Amazon EC2 instances, AWS CloudTrail, or other sources.

Best practices:

- **Activity Monitored Appropriately Amazon CloudWatch logs, events, VPC flow logs, ELB logs, S3 bucket logs, etc.**
- **AWS Cloud Trail Enabled**
- **Monitored Operating System or Application Logs**

Infrastructure protection

SEC 5. How are you enforcing network and host-level boundary protection?

In on-premises data centers, a DMZ approach separates 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 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 define the location-based security in network ACLs (e.g., an Elastic Load Balancing tier in one subnet per Availability Zone, web tier in another subnet per Availability Zone, etc.).

Best practices:

- **Controlled Network Traffic in VPC** For example, use firewalls, security groups, NACLS, a bastion host, etc.
- **Controlled Network Traffic at the Boundary** For example use AWS WAF, host based firewalls, security groups, NACLS, etc.

SEC 6. How are you leveraging AWS service level security features?

AWS services may offer additional security features (e.g., Amazon S3 bucket policies, Amazon SQS, Amazon DynamoDB, KMS key policies, etc.).

Best practices:

- **Using Additional Features Where Appropriate**

SEC 7. 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 Amazon EC2 by using traditional host-based techniques (e.g., OSSEC, Tripwire, Trend Micro Deep Security, etc.).

Best practices:

- **File Integrity** File integrity controls are used for EC2 instances.
- **EC2 Intrusion Detection** Host-based intrusion detection controls are used for EC2 instances.
- **AWS Marketplace or Partner Solution** A solution from the AWS Marketplace or from an APN Partner.
- **Configuration Management Tool** Use of a custom Amazon Machine Image (AMI) or configuration management tools (such as Puppet or Chef) that are secured by default.

Data Protection

SEC8. How are you classifying your data?

Data classification provides a way to categorize organizational data based on levels of sensitivity. This includes what data types are available, where the data is located, access levels, and protection of the data (e.g. through encryption or access control).

Best practices:

- **Using Data Classification Schema**
- **All data is Treated as Sensitive**

SEC 9. 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, operating system-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.

Best practices:

- **Not Required** Data at rest encryption is not required
- **Encrypting at Rest**

SEC 10. How are you managing keys?

Keys 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:

- **AWS CloudHSM** Use AWS CloudHSM.

- **Using AWS Service Controls** data at rest can be 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.).
- **Using Client Side** Data at rest is encrypted using client side techniques.
- **AWS Marketplace or Partner Solution** A solution from the AWS Marketplace or from an APN Partner. (e.g., SafeNet, TrendMicro, etc.).

SEC 11. 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:

- **Not Required** Encryption not required on data in transit.
- **Encrypted Communications** TLS or equivalent is used for communication as appropriate.

Incident Response

SEC 12. How do you ensure that you have the appropriate incident response?

Putting in place the tools and access ahead of a security incident, then routinely practicing incident response will make sure the architecture is updated to accommodate timely investigation and recovery.

Best practices:

- **Pre-Provisioned Access** Infosec has the right access, or means to gain access quickly. This should be pre-provisioned so that an appropriate response can be made to an incident.

- **Pre-Deployed Tools** Infosec has the right tools pre-deployed into AWS so that an appropriate response can be made to an incident
- **Non-Production Game Days** Incident response simulations are conducted regularly in the non-production environment, and lessons learned are incorporated into the architecture and operations.
- **Production Game Days** Incident response simulations are conducted regularly in the production environment, and lessons learned are incorporated into the architecture and operations.

Reliability Pillar

Foundations

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

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 architect around these.
- **Ensure There Is a Sufficient Gap Between Your Service Limit and Your Max Usage to Accommodate for Failover**
- **Service Limits are Considered Across All Relevant Accounts and Regions**

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, Elastic IP/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:

- **Connectivity Back to Data Center not Needed**
- **Highly Available Connectivity Between AWS and On-Premises Environment (as Applicable)** Multiple DX circuits, multiple VPN tunnels, AWS Marketplace appliances as applicable.
- **Highly Available Network Connectivity for the Users of the Workload** Highly available load balancing and/or proxy, DNS-based solution, AWS Marketplace appliances, etc.
- **Non-Overlapping Private IP Address Ranges** The use of 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.

Change Management

REL 3. 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 Tested** Adopt a load testing methodology to measure if scaling activity will meet application requirements.

REL 4. 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.

REL 5. How are you executing change?

Uncontrolled changes to your environment will make predictability of the effect of a change difficult. Controlled changes to provisioned AWS resources and applications is necessary to ensure that the applications and the operating environment are running known software and can be patched or replaced in a predictable manner.

Best practices:

- **Automated** Automate deployments and patching.

Failure Management

REL 6. 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:

- **Automated Backups** Use AWS features, AWS Marketplace solutions, or third-party software to automate backups.

- **Periodic Recovery Testing** Validate that the backup process implementation meets RTO and RPO through a recovery test.

REL 7. 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:

- **Multi-AZ /Region** Distribute application load across multiple Availability Zones /Regions (e.g., DNS, ELB, Application Load Balancer, API Gateway)
- **Loosely Coupled Dependencies** For example use queuing systems, streaming systems, workflows, load balancers, etc.
- **Graceful Degradation** When a component's dependencies are unhealthy, the component itself does not report as unhealthy. It is capable of continuing to serve requests in a degraded manner.
- **Auto Healing** Use automated capabilities to detect failures and perform an action to remediate. Continuously monitor the health of your system and plan to receive notifications of any significant events.

REL 8. How are you testing your resiliency?

When you test your resiliency you might find latent bugs that might only surface in production. Regularly exercising your procedures through game days will help your organization smoothly execute your procedures.

Best practices:

- **Playbook** Have a playbook for failure scenarios.
- **Failure Injection** Regularly test failures (e.g., using Chaos Monkey), ensuring coverage of failure pathways.
- **Schedule Game Days**
- **Root Cause Analysis (RCA)** Perform reviews of system failures based on significant events to evaluate the architecture.

REL 9. How are you planning for disaster recovery?

Data recovery (DR) 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.
- **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

Selection

PERF 1. How do you select the best performing architecture?

The optimal solution for a particular system will vary based on the kind of workload, often with multiple approaches combined. Well-architected systems use multiple solutions and enable different features to improve performance.

Best practices:

- **Benchmarking** Load test a known workload on AWS and use that to estimate the best selection.
- **Load Test** Deploy the latest version of your system on AWS using different resource types and sizes, use monitoring to capture performance metrics, and then make a selection based on a calculation of performance/cost.

PERF 2. How did you select your compute solution?

The optimal compute solution for a particular system may vary based on application design, usage patterns, and configuration settings. Architectures may use different compute solutions for various components and enable different features to improve performance. Selecting the wrong compute solution for an architecture can lead to lower performance efficiency.

Best practices:

- **Consider Options** Consider the different options of using instances, containers, and functions to get the best performance.
- **Instance Configuration Options** If you use instances, consider configuration options such as family, instance sizes, and features (GPU, I/O, burstable).
- **Container Configuration Options** If you use containers, consider configuration options such as memory, CPU, and tenancy configuration of the container.

- **Function Configuration Options** If you use functions, consider configuration options such as memory, runtime, and state.
- **Elasticity** Use elasticity (e.g., Auto Scaling, Amazon EC2 Container Service (ECS), AWS Lambda) to meet changes in demand.

PERF 3. How do you select your storage solution?

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.

Best practices:

- **Consider Characteristics** Consider the different characteristics (e.g., shareable, file size, cache size, access patterns, latency, throughput, persistence of data) you require to select the services you need to use (Amazon S3, Amazon EBS, Amazon Elastic File System (EFS), EC2 instance store)
- **Consider Configuration Options** Considered configuration options such as PIOPS, SSD, magnetic, and Amazon S3 Transfer Acceleration.
- **Consider Access Patterns** Optimize for how you use storage systems based on access pattern (e.g., striping, key distribution, partitioning).

PERF 4. How do you select your database solution?

The optimal database solution for a particular system can vary based on requirements for availability, consistency, partition tolerance, latency, durability, scalability and query capability. 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.

Best practices:

- **Consider Characteristics** Consider the different characteristics (e.g., availability, consistency, partition tolerance, latency, durability, scalability, query capability) so that you can select the most performant database approach to use (relational, No-SQL, warehouse, in-memory).
- **Consider Configuration Options** Consider configuration options such as storage optimization, database level settings, memory, and cache.
- **Consider Access Patterns** Optimize how you use database systems based on your access pattern (e.g., indexes, key distribution, partition, horizontal scaling).
- **Consider Other Approaches** Considered other approaches to providing queryable data such as search indexes, data warehouses, and big data.

PERF 5. How do you configure your networking solution?

The optimal network solution for a particular system will vary based on latency, throughput requirements, and so on. Physical constraints such as user or on-premises resources will drive location options, which can be offset using edge techniques or resource placement.

Best practices:

- **Consider Location** Considered your location options (e.g., region, Availability Zone, placement groups, edge) to reduce network latency.
- **Consider Product Features** Consider product features (e.g., EC2 instance network capability, very high network instance types, Amazon EBS optimized instances, Amazon S3 Transfer Acceleration, Dynamic Amazon CloudFront) to optimize network traffic.
- **Consider Networking Features** Consider networking features (e.g., Amazon Route 53 latency routing, Amazon VPC endpoints, AWS Direct Connect) to reduce network distance or jitter.
- **Appropriate NACLS** Use the minimal set of NACLS to maintain network throughput.

- **Consider Encryption Offload** Consider using load balancing to offload encryption termination (TLS).
- **Consider protocols** Consider which protocols you need to optimize network performance.

Review

PERF 6. How do you ensure that you continue to have the most appropriate resource type as new resource types and features are introduced?

When architecting solutions, there is a finite set of options that you can choose from. However, over time new technologies and approaches become available that could improve the performance of your architecture.

Best practices:

- **Review** Have a process for reviewing new resource types and sizes. Re-run performance tests to evaluate any improvements in performance efficiency.

Monitoring

PERF 7. How do you monitor your resources 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 operating system or application load).

Best practices:

- **Monitoring** Use Amazon CloudWatch, third-party, or custom monitoring tools to monitor performance.
- **Alarm-Based Notifications** Receive an automatic alert from your monitoring systems if metrics are out of safe bounds.

- **Trigger-Based Actions** Set alarms that cause automated actions to remediate or escalate issues.

Trade-offs

PERF 8. How do you use tradeoffs to improve performance?

When architecting solutions, actively thinking about tradeoffs will allow you to select an optimal approach. Often you can trade consistency, durability, and space versus time and latency to deliver higher performance.

Best practices:

- **Consider Services** Use services that improve performance, such as Amazon ElastiCache, Amazon CloudFront, and AWS Snowball.
- **Consider Patterns** Use patterns to improve performance, such as caching, read replicas, sharding, compression, and buffering.

Cost Optimization Pillar

Cost-Effective Resources

COST 1. Are you considering cost when you select AWS services for your solution?

Amazon EC2, Amazon EBS, Amazon S3, etc. are "building-block" AWS services. Managed services such as Amazon RDS, Amazon DynamoDB, etc. are "higher level" AWS services. By selecting the appropriate building-blocks and managed services, you can optimize your architecture for cost. For example, using 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:

- **Select Services for Cost Reduction** Analyze services to see which ones you can use to reduce cost.
- **Optimize for License Costs**
- **Optimize Using Serverless and Container-Based Approach** Use of AWS Lambda, Amazon S3 websites, Amazon DynamoDB, and Amazon ECS to reduce cost.
- **Optimize Using Appropriate Storage Solutions** Use the most cost-effective storage solution based on usage patterns (e.g., Amazon EBS cold storage, Amazon S3 Standard-Infrequent Access, Amazon Glacier, etc.).
- **Optimize Using 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.
- **Optimize Using Other Application-Level Services** Use Amazon Simple Queue Service (SQS), Amazon Simple Notification Service (SNS), and Amazon Simple Email Service (SES) where appropriate.

COST 2. Have you sized your resources to meet your cost targets?

Ensure that you choose the appropriate AWS resource size for the task at hand. AWS encourages the use of benchmarking assessments to ensure that the type you choose is optimized for its workload.

Best practices:

- **Metrics Driven Resource Sizing** Leverage performance metrics to select the right size/type to optimize for cost. Appropriately provision throughput, sizing, and storage for services such as Amazon EC2, Amazon DynamoDB, Amazon EBS (provisioned IOPS), Amazon RDS, Amazon EMR, networking, etc.

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

Use the pricing model that is 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:

- **Reserved Capacity and Commit Deals** Regularly analyze usage and purchase Reserved Instances accordingly (e.g. Amazon EC2, Amazon DynamoDB, Amazon S3, Amazon CloudFront, etc.).
- **Spot** Use Spot Instances (e.g. Spot block, fleet) for select workloads (e.g., batch, EMR, etc.).
- **Consider Region Cost** Factor costs into region selection.

Matching Supply and Demand

COST 4. 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 underutilizing 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.
- **Buffer-Based Approach** Buffer work (e.g. using Amazon Kinesis or Amazon Simple Queue Service (SQS)) to defer work until you have sufficient capacity to process it.
- **Time-based approach** Examples of a time-based approach include following the sun, turning off Development and Test instances over the weekend, following quarterly or annual schedules (e.g., Black Friday).

Expenditure Awareness

COST 5. 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 content delivery network (CDN). Remember that a small yet effective architectural change can drastically reduce your operational costs.

Best practices:

- **Optimize** Architect to optimize data transfer (application design, WAN acceleration, Multi-AZ, region selection, etc.).
- **CDN** Use a CDN where applicable.

- **AWS Direct Connect** Analyze the situation and use AWS Direct Connect where applicable.

COST 6. 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** Tag all taggable resources to be able to correlate changes in your bill to changes in our infrastructure and usage.
- **Leverage Billing and Cost Management Tools** Have a standard process to load and interpret the Detailed Billing Reports or Cost Explorer. Monitor usage and spend regularly using Amazon CloudWatch or a third-party provider where applicable (examples: Cloudability, CloudCheckr, CloudHealth).
- **Notifications** Let key members of your team know if your spend moves outside of well-defined limits.
- **Finance Driven Charge Back/Show Back Method** Use this to allocate instances and resources to cost centers (e.g., using tagging).

COST 7. Do you decommission resources that you no longer need or stop resources that are temporarily not needed?

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 or review Trusted Advisor cost optimization recommendations.

Best practices:

- **Automated** Design your system to gracefully handle resource termination as you identify and decommission non-critical or unrequired resources with low utilization.
- **Defined Process** Have a process in place to identify and decommission orphaned resources.

COST 8. 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 governance mechanisms 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 lifecycle of projects, teams, and environments to avoid using and paying for unnecessary resources.

Optimizing Over Time

COST 9. How do you manage and/or consider the adoption of new services?

As AWS releases new services and features, it is a best practice to review your existing architectural decisions to ensure they continue to be the most cost effective.

Best practices:

- **Establish a Cost Optimization Function**

- **Review** Have a process for reviewing new services, resource types, and sizes. Re-run performance tests to evaluate any reduction in cost.

Operational Excellence Pillar

Prepare

OPS 1. What best practices for cloud operations are you using?

Effective preparation is required to drive operational excellence. Using operations checklists ensures that your workloads are ready for production operation. The use of checklists prevents unintentional promotion to production without effective preparation.

Best practices:

- **Operational Checklist** Create an operational checklist that you use to evaluate if you are ready to operate the workload.
- **Proactive Plan** Have a proactive plan for events (e.g., marketing campaigns, flash sales) that prepares you for both opportunities and risks that could have a material impact on your business (e.g., reputation, finances).
- **Security Checklist** Create a security checklist that you can use to evaluate if you are ready to securely operate the workload (e.g., zero day, DDoS, compromised keys).

OPS 2. How are you doing configuration management for your workload?

Environments, architecture, and the configuration parameters for resources within them, should be documented in a way that allows components to be easily identified for tracking and troubleshooting. Changes to configuration should also be trackable and automated.

Best practices:

- **Resource Tracking** Plan for ways to identify your resources and their function within the workload (e.g., use metadata, tagging).
- **Documentation** Document your architecture (e.g., infrastructure as code, CMDB, diagrams, release notes).

- **Capture Operational Learnings** Captured operational learnings over time (e.g., wiki, knowledge base, tickets).
- **Immutable Infrastructure** Establish an immutable infrastructure so that you redeploy, you don't patch.
- **Automated Change Procedures** Automate your change procedures.
- **Configuration Management Database (CMDB)** Track all changes in a CMDB.

Operate

OPS 3. How are you evolving your workload while minimizing the impact of change?

Your focus should be on automation, small frequent changes, regular quality assurance testing, and defined mechanisms to track, audit, roll back, and review changes.

Best practices:

- **Deployment Pipeline** Put a CI/CD pipeline in place (e.g., source code repository, build systems, deployment and testing automation).
- **Release Management Process** Establish a release management process (e.g., manual or automated).
- **Small Incremental Changes** Ensure that you can release small incremental versions of system components.
- **Revertible Changes** Be prepared to revert changes that introduce operational issues (e.g., roll back, feature toggles).
- **Risk Mitigation Strategies** Use risk mitigation strategies such as Blue/Green, Canary, and A/B testing.

OPS 4. How do you monitor your workload to ensure it is operating as expected?

Your system can degrade over time due to internal and/or external factors. By monitoring the behavior of your systems, you can identify these factors of degradation and remediate them.

Best practices:

- **Monitoring** Use Amazon CloudWatch, third-party, or custom monitoring tools to monitor performance.
- **Aggregate Logs** Aggregate logs from multiple sources (e.g., application logs, AWS service-specific logs, VPC flow logs, CloudTrail).
- **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.

Respond

OPS 5. How do you respond to unplanned operational events?

Be prepared to automate responses to unexpected operational events. This includes not just for alerting, but also mitigation, remediation, rollback, and recovery.

Best practices:

- **Playbook** Have a playbook that you follow (e.g., on call process, workflow chain, escalation process) and update regularly.
- **RCA Process** Have an RCA process to ensure that you can resolve, document, and fix issues so they do not happen in the future.
- **Automated Response** Handle unplanned operational events gracefully through automated responses (e.g., Auto Scaling, Support API).

OPS 6. How do you manage escalation when you respond to unplanned operational events?

Responses to unplanned operational events should follow a pre-defined playbook that includes stakeholders and the escalation process and procedures. Define escalation paths and include both functional and hierarchical escalation capabilities. Hierarchical escalation should be automated, and escalated priority should result in stakeholder notifications.

Best practices:

- **Appropriately Document and Provision** Put necessary stakeholders and systems in place for receiving alerts when escalations occur.
- **Functional Escalation with Queue-based Approach** Escalate between appropriate functional team queues based on priority, impact, and intake mechanisms.
- **Hierarchical Escalation** Use a demand- or time-based approach. As impact, scale, or time to resolution/recovery of incident increases, priority is escalated.
- **External Escalation Path** Include external support, AWS support, AWS Partners, and third-party support engagement in escalation paths.
- **Hierarchical Priority Escalation is Automated** When demand or time thresholds are passed, priority automatically escalates.