AWS Well-Architected Framework

November 2017

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 document describes the AWS Well-Architected Framework, which enables you to review and improve your cloud-based architectures and better understand the business impact of your 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

The AWS Well-Architected Framework helps you understand the pros and cons of decisions you make while building systems on AWS. By using the Framework you will learn architectural best practices for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud. It provides a way for you to consistently measure your architectures against best practices and identify areas for improvement. 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. 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.

Definitions

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 provides a consistent set of best practices for customers and partners to evaluate architectures, and provides 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 — operational excellence, security, reliability, performance efficiency, and cost optimization.

Pillar Name	Description
Operational Excellence	The ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.
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 disruptions, 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.

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 the other pillars.

On Architecture

In on-premises environments customers often have a central team for technology architecture that acts as an overlay to other product or feature teams to ensure they are following best practice. Technology architecture teams are often composed of a set of roles such as Technical Architect (infrastructure), Solutions Architect (software), Data Architect, Networking Architect, and

Security Architect. Often these teams use TOGAF[1] or the Zachman Framework[2] as part of an enterprise architecture capability.

In Amazon, we prefer to distribute capabilities into teams rather than having a centralized team with that capability. There are risks when you choose to distribute decision making authority, for example, ensuring that teams are meeting internal standards. We mitigate these risks in two ways. First, we have *practices*[1] that focus on enabling each team to have that capability, and we provide access to experts who ensure that teams raise the bar on the standards they need to meet. Second, we put in place *mechanisms*[2] that carry out automated checks to ensure standards are being meet. This distributed approach is supported by the Amazon leadership principles,[3] and establishes a culture across all roles that *works back*[3] from the customer. Customer-obsessed teams build products in response to a customer need.

For architecture this means that we expect every team to have the capability to create architectures and to follow best practices. To help new teams gain these capabilities or existing teams to raise their bar, we enable access to a virtual community of principal engineers who can review their designs and help them understand what AWS best practices are. The principal engineering community works to make best practices visible and accessible. One way they do this, for example, is through lunchtime talks that focus on applying best practices to real examples. These talks are recorded and can be used as part of onboarding materials for new team members.

AWS best practices emerge from our experience running thousands of systems at internet scale. We prefer to use data to define best practices, but we also use subject matter experts like principal engineers to set them. As principal engineers see new best practices emerge, they work as a community to ensure that teams follow them. In time, these best practices are formalized into our internal review processes, as well as into mechanisms that enforce compliance.

[1] Ways of doing things, process, standards, and accepted norms.

[2] *"Good intentions never work, you need good mechanisms to make anything happen"* Jeff Bezos. This means replacing humans best efforts with mechanisms (often automated) that check for compliance with rules or process.

[3] Working backward is a fundamental part of our innovation process. We start with the customer and what they want, and let that define and guide our efforts.

Well-Architected is the customer-facing implementation of our internal review process, where we have codified our principal engineering thinking across field roles like Solutions Architecture and internal engineering teams. Well-Architected is a scalable mechanism that lets you take advantage of these learnings.

By following the approach of a principal engineering community with distributed ownership of architecture, we believe that a Well-Architected enterprise architecture can emerge that is driven by customer need. Technology leaders (such as a CTOs or development managers), carrying out Well-Architected reviews across all your workloads will allow you to better understand the risks in your technology portfolio. Using this approach you can identify themes across teams that your organization could address by mechanisms, trainings, or lunchtime talks where your principal engineers can share their thinking on specific areas with multiple teams.

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 about 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’s 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**: Allow for evolutionary architectures. In a traditional environment, architectural decisions are often implemented as 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.
- **Drive architectures using data**: 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 can undermine the integrity and function of the building. When architecting technology solutions, if you neglect the five pillars of operational excellence, security, reliability, performance efficiency, and cost optimization it can become challenging to build a system that delivers on your expectations and requirements. Incorporating these pillars into your architecture will help you produce stable and efficient systems. This will allow you to focus on the other aspects of design, such as functional requirements.

Operational Excellence

The **operational excellence** pillar includes the ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.

The operational excellence pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Operational Excellence Pillar whitepaper.[4]

Design Principles

There are six design principles for operational excellence in the cloud:

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

- **Learn from all operational failures**: Drive improvement through lessons learned from all operational events and failures. Share what is learned across teams and through the entire organization.

Definition

There are three best practice areas for operational excellence in the cloud:

- Prepare
- Operate
- Evolve

Operations teams need to understand their business and customer needs so they can effectively and efficiently support business outcomes. Operations creates and uses procedures to respond to operational events and validates their effectiveness to support business needs. Operations collects metrics that are used to measure the achievement of desired business outcomes. Everything continues to change—your business context, business priorities, customer needs, etc. It’s important to design operations to support evolution over time in response to change and to incorporate lessons learned through their performance.

Best Practices

Prepare

Effective preparation is required to drive operational excellence. Business success is enabled by shared goals and understanding across the business, development, and operations. Common standards simplify workload design and management, enabling operational success. Design workloads with mechanisms to monitor and gain insight into application, platform, and infrastructure components, as well as customer experience and behavior.

Create mechanisms to validate that workloads, or changes, are ready to be moved into production and supported by operations. Operational readiness is validated through checklists to ensure a workload meets defined standards and that required procedures are adequately captured in runbooks and playbooks. Validate that there are sufficient trained personnel to effectively support the workload. Prior to transition, test responses to operational events and failures. Practice responses in supported environments through failure injection and game day events.

AWS enables operations as code in the cloud and the ability to safely experiment, develop operations procedures, and practice failure. Using AWS CloudFormation enables you to have consistent, templated, sandbox development, test, and production environments with increasing levels of operations control. AWS enables visibility into your workloads at all layers through various log collection and monitoring features. Data on use of resources, application programming interfaces (APIs), and network flow logs can be collected using Amazon CloudWatch, AWS CloudTrail, and VPC Flow Logs. You can use the CloudWatch Logs agent, or the collectd plugin, to aggregate information about the operating system into CloudWatch.

The following questions focus on the **prepare** considerations for operational excellence. (For a list of operational excellence questions, answers, and best practices, see the Appendix.)

OPS 1: What factors drive your operational priorities?

OPS 2: How do you design your workload to enable operability?

OPS 3: How do you know that you are ready to support a workload?

Implement the minimum number of architecture standards for your workloads. Balance the cost to implement a standard against the benefit to the workload and the burden upon operations. Reduce the number of supported standards to reduce the chance that lower-than-acceptable standards will be applied by error. Operations personnel are often constrained resources.

Invest in scripting operations activities to maximize the productivity of operations personnel, minimize error rates, and enable automated responses. Adopt deployment practices that take advantage of the elasticity of the cloud to facilitate pre-deployment of systems for faster implementations.

Operate

Successful operation of a workload is measured by the achievement of business and customer outcomes. Define expected outcomes, determine how success will be measured, and identify the workload and operations metrics that will be used in those calculations to determine if operations are successful. Consider that operational health includes both the health of the workload and the health and success of the operations acting upon the workload (for example, deployment and incident response). Establish baselines from which improvement or degradation of operations will be identified, collect and analyze your metrics, and then validate your understanding of operations success and how it changes over time. Use collected metrics to determine if you are satisfying customer and business needs, and identify areas for improvement.

Efficient and effective management of operational events is required to achieve operational excellence. This applies to both planned and unplanned operational events. Use established runbooks for well-understood events, and use playbooks to aid in the resolution of other events. Prioritize responses to events based on their business and customer impact. Ensure that if an alert is raised in response to an event, there is an associated process to be executed, with a specifically identified owner. Define in advance the personnel required to resolve an event and include escalation triggers to engage additional personnel, as it becomes necessary, based on impact (that is, duration, scale, and scope). Identify and engage individuals with the authority to decide on courses of action where there will be a business impact from an event response not previously addressed.

Communicate the operational status of workloads through dashboards and notifications that are tailored to the target audience (for example, customer, business, developers, operations) so that they may take appropriate action, so that their expectations are managed, and so that they are informed when normal operations resume.

Determine the root cause of unplanned events and unexpected impacts from planned events. This information will be used to update your procedures to mitigate future occurrence of events. Communicate root cause with affected communities as appropriate.

In AWS, you can generate dashboard views of your metrics collected from workloads and natively from AWS. You can leverage CloudWatch or third-party applications to aggregate and present business, workload, and operations level views of operations activities. AWS provides workload insights through logging capabilities including AWS X-Ray, CloudWatch, CloudTrail, and VPC Flow Logs enabling the identification of workload issues in support of root cause analysis and remediation.

The following questions focus on **operate** considerations for operational excellence.

OPS 4: What factors drive your understanding of operational health?

OPS 5: How do you manage operational events?

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. Failing to have a rollback plan, or the ability to mitigate failure impacts, will prevent continuity of operations. Align metrics to business needs so that responses are effective at maintaining business continuity. One-time, decentralized metrics with manual responses will result in greater disruption to operations during unplanned events.

Evolve

Evolution of operations is required to sustain operational excellence. Dedicate work cycles to making continuous incremental improvements. Regularly evaluate and prioritize opportunities for improvement (for example, feature requests, issue remediation, and compliance requirements), including both the workload and operations procedures. Include feedback loops within your procedures to rapidly identify areas for improvement and capture learnings from the execution of operations.

Share lessons learned across teams to share the benefits of those lessons. Analyze trends within lessons learned and perform cross-team retrospective analysis of operations metrics to identify opportunities and methods for improvement. Implement changes intended to bring about improvement and evaluate the results to determine success.

With AWS Developer Tools you can implement continuous delivery build, test, and deployment activities that work with a variety of source code, build, testing, and deployment tools from AWS and third parties. The results of deployment activities can be used to identify opportunities for improvement for both deployment and development. You can perform analytics on your metrics data integrating data from your operations and deployment activities, to enable analysis of the impact of those activities against business and customer outcomes. This data can be leveraged in cross-team retrospective analysis to identify opportunities and methods for improvement.

The following question focuses on **evolve** considerations for operational excellence.

OPS 6: How do you evolve operations?

Successful evolution of operations is founded in: frequent small improvements; providing safe environments and time to experiment, develop, and test improvements; and environments in which learning from failures is encouraged. Operations support for sandbox, development, test, and production environments, with increasing level of operational controls, facilitates development and increases the predictability of successful results from changes deployed into production.

Key AWS Services

The AWS service that is essential to operational excellence is AWS CloudFormation, which you can use to create templates based on best practices. This enables you to provision resources in an orderly and consistent fashion from your development through production environments. The following services and features support the three areas of operational excellence:

- **Prepare**: AWS Config and AWS Config rules can be used to create standards for workloads and to determine if environments are compliant with those standards before being put into production.
- **Operate**: Amazon CloudWatch allows you to monitor the operational health of a workload.
- **Evolve**: Amazon Elasticsearch Service (Amazon ES) allows you to analyze your log data to gain actionable insights quickly and securely.

Resources

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

Documentation

- DevOps and AWS[5]

Whitepaper

- Operational Excellence Pillar[6]

Video

- AWS re:Invent 2015 – DevOps at Amazon[7]

Security

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

The security pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Security Pillar whitepaper.[8]

Design Principles

There are six design principles for security in the cloud:

- **Implement a strong identity foundation**: Implement the principle of least privilege and enforce separation of duties with appropriate authorization for each interaction with your AWS resources. Centralize privilege management and reduce or even eliminate reliance on long-term credentials.
- **Enable traceability**: Monitor, alert, and audit actions and changes to your environment in real time. Integrate logs and metrics with systems to automatically respond and take action.
- **Apply security at all layers**: Rather than just focusing on protecting a single outer layer, apply a defense-in-depth approach with other security controls. Apply to all layers, for example, edge network, virtual private cloud (VPC), subnet, load balancer, every instance, operating system, and application.

- **Automate security best practices**: Automated software-based security mechanisms improve your ability to securely scale more rapidly and cost effectively. Create secure architectures, including the implementation of controls that are defined and managed as code in version-controlled templates.
- **Protect data in transit and at rest**: Classify your data into sensitivity levels and use mechanisms, such as encryption and tokenization where appropriate. Reduce or eliminate direct human access to data to reduce risk of loss or modification.
- **Prepare for security events**: Prepare for an incident by having an incident management process that aligns to your organizational requirements. Run incident response simulations and use tools with automation to increase your speed for detection, investigation, and recovery.

Definition

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

- Identity and Access Management
- Detective Controls
- Infrastructure Protection
- Data Protection
- 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, as an AWS customer you can focus on using services to accomplish your 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 should 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 you to control user access to AWS services and resources. You should 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 **identity and access management** considerations for security. (For a list of security questions, answers, and best practices, see the Appendix.)

SEC 1: How are you protecting access to and use of the AWS account root user 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 (for example, applications, scripts, and/or third-party tools or services)?

It is critical to keep root user credentials protected, and to this end AWS recommends attaching MFA to the root user and locking the credentials with the MFA in a physically secured location. IAM 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 can help your organization 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. CloudTrail logs, AWS API calls, and 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 (Amazon 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 or 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 you the ability to define a data-retention lifecycle or define where data will be preserved, archived, 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 MFA, which are necessary to meet best practices and industry or regulatory obligations. Use of these methodologies is critical for successful ongoing operations either in 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 should use Amazon Virtual Private Cloud (Amazon VPC) to create a private, secured, and scalable environment in which you can define your topology—including gateways, routing tables, and public and 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 system?

Multiple layers of defense are advisable in any type of environment. 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.

AWS customers are able to tailor, or harden, the configuration of an Amazon Elastic Compute Cloud (Amazon EC2), Amazon EC2 Container Service (Amazon ECS) container, or AWS Elastic Beanstalk 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:

- As an AWS customer you maintain full control over your data.
- AWS makes it easier for you to encrypt your data and manage keys, including regular key rotation, which can be easily automated by AWS or maintained by you.
- Detailed logging that contains important content, such as file access and changes, is available.
- AWS has designed storage systems for exceptional resiliency. For example, Amazon 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 you explicitly enable a feature or leverage a service that provides that functionality.

The following questions focus on **data protection** considerations for 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 encrypting data at rest and in transit. We build features into our 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 (ELB).

Incident Response

Even with extremely mature preventive and detective controls, your organization 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 runbooks 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 question focuses on **incident response** considerations for security.

> SEC 12: How do you ensure that 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 IAM, which allows you to securely control access to AWS services and resources for your users. The following services and features support the five areas in security:

- **Identity and Access Management**: IAM enables you to securely control access to AWS services and resources. 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 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 ELB, Amazon Elastic Block Store (Amazon EBS), Amazon S3, and Amazon Relational Database Service (Amazon RDS) include encryption capabilities to protect your data in transit and at rest. Amazon Macie automatically discovers, classifies, and protects sensitive data, while AWS Key Management Service (AWS KMS) makes it easy for you to create and control keys used for encryption.
- **Incident Response**: IAM should be used to grant appropriate authorization to incident response teams. AWS 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

- AWS Security Center[9]
- AWS Compliance[10]
- AWS Security Blog[11]

Whitepapers

- Security Pillar[12]
- AWS Security Overview[13]
- AWS Security Best Practices[14]
- AWS Risk and Compliance[15]

Videos

- Security of the AWS Cloud[16]
- Shared Responsibility Overview[17]

Reliability

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.

The reliability pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Reliability Pillar whitepaper.[18]

Design Principles

There are five design principles for reliability in the cloud:

- **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's 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:

- Foundations
- Change Management
- 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 **foundations** considerations for reliability. (For a list of reliability questions, answers, and best practices, see the Appendix.)

REL 1: How are you managing 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. 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, by 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 management** 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 doesn't 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. It is generally of interest to know how to become aware of these failures, respond to them, and prevent them from happening again.

With AWS, you 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 your 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 cause failure, and then observe how they recover. Do this on a regular schedule and ensure that such testing is 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 essential to reliability is Amazon CloudWatch, which monitors runtime metrics. The following services and features support the three areas in reliability:

- **Foundations**: 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. AWS Trusted Advisor provides visibility into service limits. AWS Shield is a managed Distributed Denial of Service (DDoS) protection service that safeguards web applications running on AWS.
- **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. Auto Scaling is a service that will provide an automated demand management for a deployed workload. CloudWatch provides the ability to alert on metrics, including custom metrics.

- **Failure Management**: AWS CloudFormation provides templates for the creation of AWS resources and provisions them in an orderly and predictable fashion. Amazon S3 provides a highly durable service to keep backups. Amazon Glacier provides highly durable archives. AWS KMS provides a reliable key management system that integrates with many AWS services.

Resources

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

Documentation

- Service Limits[19]
- Service Limits Reports Blog[20]
- AWS Shield[21]
- Amazon CloudWatch[22]
- Amazon S3[23]
- AWS KMS[24]

Whitepapers

- Reliability Pillar[25]
- Backup Archive and Restore Approach Using AWS[26]
- Managing your AWS Infrastructure at Scale[27]
- AWS Disaster Recovery[28]
- AWS Amazon VPC Connectivity Options[29]

Videos

- How do I manage my AWS service limits?[30]
- Embracing Failure: Fault-Injection and Service Reliability[31]

Report

- Benchmarking Availability and Reliability in the Cloud[32]

AWS Support

- AWS Premium Support[33]
- Trusted Advisor[34]

Performance Efficiency

The **performance efficiency** pillar includes the ability to use computing resources efficiently to meet system requirements and to maintain that efficiency as demand changes and technologies evolve.

The performance efficiency pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Performance Efficiency Pillar whitepaper.[35]

Design Principles

There are five design principles for performance efficiency in the cloud:

- **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, serverless 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:

- Selection
- Review
- Monitoring
- 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 Cloud. 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 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 question focuses on **selection** considerations for performance efficiency. (For a list of performance efficiency questions, answers, and best practices, see the Appendix.)

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 Partner Network (APN) 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 (for example, 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 question focuses on **compute** considerations for performance efficiency.

PERF 2: How did 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 disk drives (HDDs) to SSDs, and easily move virtual drives from one instance to another in seconds.

The following 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.

Amazon 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 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 (for example, 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 (for example, Amazon Route 53 latency routing, Amazon VPC endpoints, and AWS Direct Connect) to reduce network distance or jitter.

The following question focuses on **network** considerations for performance efficiency.

> PERF 5: How do you configure your networking solution?

When selecting 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 locations, services, and features regularly. Any of these could positively improve the performance efficiency of your architecture.

The following question focuses on **review** considerations for 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

After 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.

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

The following question focuses on **monitoring** considerations for 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 you can conduct simulations in the production environment to test your alarm solution and ensure that it correctly recognizes issues.

Tradeoffs

When you architect solutions, think about tradeoffs 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. Amazon DynamoDB Accelerator (DAX) provides a read-through/write-through distributed caching tier in front of Amazon DynamoDB, supporting the same API, but providing sub-millisecond latency for entities that are in the cache;

The following question focuses on **tradeoffs** considerations for performance efficiency.

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

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

Key AWS Services

The AWS service that is essential to performance efficiency is Amazon CloudWatch, which monitors your resources and systems, providing visibility into your overall performance and operational health. The following services and features support the four areas in 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 PIOPS 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 AWS 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.
- **Tradeoffs**: 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 for Performance Efficiency.

Documentation

- Amazon S3 Performance Optimization[36]
- Amazon EBS Volume Performance[37]

Whitepaper

- Performance Efficiency Pillar[38]

Videos

- AWS re:Invent 2016: Scaling Up to Your First 10 Million Users (ARC201)[39]
- Performance AWS re:Invent 2016: Deep Dive on Amazon EC2 Instances, Featuring Performance Optimization (CMP301)[40]

Cost Optimization

The **cost optimization** pillar includes the ability to avoid or eliminate unneeded cost or suboptimal resources.

The cost optimization pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Cost Optimization Pillar whitepaper.[41]

Design Principles

There are five design principles for cost optimization in the cloud:

- **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% (40 hours versus 168 hours).
- **Measure overall efficiency**: Measure the business output of the system and the costs associated with delivering it. Use this measure to understand the gains you make from increasing output and reducing costs.
- **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:

- Cost-Effective Resources
- Matching Supply and Demand
- Expenditure Awareness
- Optimizing Over Time

As with the other pillars, there are tradeoffs 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 one hour to run on a larger server that is twice as expensive. Both servers give you the same outcome, but the smaller one 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 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 allow you to reserve capacity and offer savings of up to 75% 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 questions focus on **cost-effective resources** considerations for cost optimization. (For a list of cost optimization questions, answers, and best practices, see the Appendix.)

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 cost 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.

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.

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 question focuses on **matching supply and demand** considerations 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 questions focus on **expenditure awareness** considerations for cost optimization.

COST 5: Do 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 EC2 instances or 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, entire services, and 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 question focuses on **optimizing over time** considerations for cost optimization.

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

When regularly reviewing your deployments, assess how newer services can help save you money. For example, Amazon Aurora on RDS could help you reduce costs for relational databases.

Key AWS Services

The AWS service that is essential to cost optimization is cost allocation tags, which helps you understand the costs of a system. The following services and features support the four areas of cost optimization:

- **Cost-Effective Resources**: You can use Reserved Instances and prepaid capacity to reduce your cost. You can use Cost Explorer to see patterns in how much you spend on AWS resources over time, identify areas that need further inquiry, and see trends that you can use to understand your costs.

- **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 (Amazon 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 you money by eliminating unused or idle resources or committing to Reserved Instance capacity.

Resources

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

Documentation

- Analyzing Your Costs with Cost Explorer[42]
- AWS Cloud Economics Center[43]
- AWS Detailed Billing Reports[44]

Whitepaper

- Cost Optimization Pillar[45]

Video

- Cost Optimization on AWS[46]

Tools

- AWS Total Cost of Ownership (TCO) Calculators[47]
- AWS Simple Monthly Calculator[48]

The Review Process

The review of architectures needs to be done in a consistent manner, with a blame-free approach that encourages diving deep. It should be a light-weight process (hours not days) that is a conversation and not an audit. The purpose of reviewing an architecture is to identify any critical issues that might need addressing or areas that could be improved. The outcome of the review is a set of actions that should improve the experience of a customer using the workload.

As discussed in the "On Architecture" section, you will want each team member to take responsibility for the quality of its architecture. We recommend that teams building an architecture using the Well-Architected Framework continually review their architecture, rather than holding a formal review meeting. A continuous approach allows your team members to update answers as the architecture evolves, and improve the architecture as you deliver features.

AWS Well-Architected is aligned to the way that AWS reviews systems and services internally. It is premised on a set of design principles that influences architectural approach, and questions that ensure that people don't neglect areas that often featured in a Root Cause Analysis (RCA). Whenever there is a significant issue with an internal system, AWS service, or customer we look at the RCA to see if we could improve the review processes we use.

Reviews should be applied at multiple times in a workload lifecycle, early on in the design phase to avoid *one-way doors*[4] that are difficult to change, and then before the go live date. Post go live your workload will continue to evolve as you add new features and change technology implementations. The architecture of a workload changes over time. You will need to follow good hygiene practices to stop its architectural characteristics from degrading as you evolve it. As you make significant architecture changes you should follow a set of hygiene processes including a Well-Architected review.

[4] Many decisions are reversible, two-way doors. Those decisions can use a light-weight process. One-way doors are hard or impossible to reverse and require more inspection before making them.

If you want to use the review as a one-time snapshot or independent measurement you will want to ensure you have all the right people in the conversation. Often we find that reviews are the first time that a team truly understands what they have implemented. An approach that works well when reviewing another team's workload is to have a series of informal conversations about their architecture where you can glean the answers to most questions. You can then follow up with one or two meetings where you can gain clarity or dive deep on areas of ambiguity or perceived risk.

Here are some suggested items to facilitate your meetings:

- A meeting room with whiteboards
- Print outs of any diagrams or design notes
- Action list of questions that require out-of-band research to answer (for example, did we enable encryption or not?)

After you have done a review you should have a list of issues that you can prioritize based on your business context. You will also want to take into account the impact of those issues on the day-to-day work of your team. If you address these issues early you could free up time to work on creating business value rather than solving recurring problems. As you address issues you can update your review to see how the architecture is improving.

While the value of a review is clear after you have done one, you may find that a new team might be resistant at first. Here are some objections that can be handled through educating the team on the benefits of a review:

- "*We are too busy!*" (Often said when the team is getting ready for a big launch.)
    - If you are getting ready for a big launch you will want it to go smoothly. The review will allow you to understand any problems you might have missed.
    - We recommend that you carry out reviews early in the design lifecycle to uncover risks and develop a mitigation plan aligned with the feature delivery roadmap.

- "*We don't have time to do anything with the results!*" (Often said when there is an immovable event, such as the Super Bowl, that they are targeting.)
    - These events can't be moved. Do you really want to go into it without knowing the risks in your architecture? Even if you don't address all of these issues you can still have playbooks for handling them if they materialize.
- "*We don't want others to know the secrets of our solution implementation!*"
    - If you point the team at the questions that are in the Appendix of the Well-Architected Framework whitepaper, they will see that none of the questions reveal any commercial or technical propriety information.

As you carry out multiple reviews with teams in your organization you might identify thematic issues. For example, you might see that a group of teams has clusters of issues in a particular pillar or topic. You will want to look at all your reviews in a holistic manner, and identify any mechanisms, training, or principal engineering talks that could help address those thematic issues.

Conclusion

The AWS Well-Architected Framework provides architectural best practices across the 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. It also provides a set of AWS best practices for each pillar. Using the Framework in your architecture will help you produce stable and efficient systems, which allow 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
- Rodney Lester: Reliability Lead Well-Architected, Amazon Web Services
- Brian Carlson: Operations Lead Well-Architected, Amazon Web Services

- Jon Steele: Sr. Technical Account Manager, Amazon Web Services
- Ryan King: Technical Program Manager, Amazon Web Services
- Ben Potter: Security Lead 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
- Callum Hughes: Solutions Architect, Amazon Web Services

Further Reading

For additional information, see the following:

- Operational Excellence Pillar
- Security Pillar
- Reliability Pillar
- Performance Efficiency
- Cost Optimization Pillar

Document Revisions

Date	Description
November 2017	Operational Excellence moved to front of pillars and rewritten so it frames the other pillars. Refreshed other pillars to reflect evolution of AWS.
November 2016	Updated the Framework to include operational excellence pillar, and revised and updated the other pillars to reduce duplication and incorporate learnings from carrying out reviews with thousands of customers.
November 2015	Updated the Appendix with current Amazon CloudWatch Logs information.
October 2015	Original publication.

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

Operational Excellence

Prepare

OPS 1: What factors drive your operational priorities?

Businesses exist to serve the needs of the customer. Operations exists to serve the needs of the business. Operations priorities are shaped by business and customer needs. External factors such as regulatory compliance requirements or industry best practices may also influence operational priorities. Make informed decisions that consider both benefits and risks when setting operational priorities.

Best practices:

- **Business needs**: Involve the business and development teams in setting operational priorities.
- **Compliance requirements**: External factors such as regulatory or industry standards may obligate your business to satisfy specific requirements, for example, SOX regulatory compliance requirement versus PCI industry best practice.
- **Risk management**: Balance the risk of decisions against their potential benefit.

OPS 2: How do you design your workload to enable operability?

Consider operations needs as a part of system design. Design workloads that provide insight to their operating status, customer behavior, and customer experience enabling issue response and the identification of areas for improvement. Implement low-risk, zero-downtime deployment methods that allow for easy identification of failures and fast recovery.

Best practices:

- **Shared design standards**: Share existing best practices, guidance, and governance requirements across teams and include in system design. Procedures exist to request changes, additions, and exceptions to design standards.
- **Design for cloud operations**: Include cloud-enabled capabilities that provide advantages over physical resources in system design.
- **Provides insights into workload behavior**: Build instrumentation into your system design to enable understanding of what is going on within the system and measure performance across individual components.
- **Provides insights into customer behavior**: Build instrumentation into your system design to enable understanding of how the customer uses the system and the quality of their experience.
- **Implement practices that reduce defects, ease remediation, and improve flow**: Adopt approaches that include fast feedback on quality and enable refactoring and bug fixing.
- **Mitigate deployment risks**: Use approaches such as frequent, small, reversible changes, automated deployments, testing, canary/one-box deployments, blue-green, etc.

OPS 3: How do you know that you are ready to support a workload?

Processes are put in place to validate operational readiness to support a workload. Sufficient, appropriately skilled personnel are in place and prepared to support the workload. Operations procedures are accessible, reviewed frequently, and updated as appropriate. Treat operations procedures as code, and automate procedures where appropriate.

Best Practices:

- **Continuous improvement culture**: This governs the way you operate and recognizes that change is constant and that you need to continue to experiment and evolve, acting on opportunities to improve.
- **Shared understanding of the value to the business**: Have cross-team understanding of the value of the workload to the business, and procedures exist to engage additional teams for support.

- **Personnel capability**: Ensure that you have an appropriate number of trained personnel to support your workload needs. Perform regular review of workload demands and train or adjust personnel capacity as necessary.
- **Documented accessible governance and guidance**: Ensure that standards are accessible, readily understood, and are measurable for compliance. A path exists to propose changes to standards and request exceptions.
- **Use checklists**: Use checklists to evaluate if you are ready to operate workloads. These include operational readiness and security checklists.
- **Runbooks**: Have runbooks for well-understood events and procedures.
- **Playbooks**: Have a playbook for failure scenarios.
- **Practice recovery**: Identify potential failure scenarios and test your responses (for example, game days, failure injection).

Operate

OPS 4: What factors drive your understanding of operational health?

Define success in terms of business outcomes, determine measures for success, identify workload and operational execution metrics to satisfy those measures, and create a business-level view of operations success. Establish baselines and perform proactive review to identify trends and determine responses. Validate insights into workload and operational health with cross-functional teams, adjusting responses as appropriate.

Best Practices:

- **Define expected business and customer outcomes**: Have a documented definition of what success looks like for the workload from a business and customer perspective.
- **Identify success metrics**: Define metrics that can be used to measure the behavior of the workload against business and customer expectations.

- **Identify workload metrics**: Define metrics that can be used to measure the status of the workload and its components against the success metrics.
- **Identify operations metrics**: Define metrics that can be used to measure the execution of operations activities (runbooks and playbooks).
- **Establish baselines**: Establish baselines for metrics to provide expected values as the basis for comparison.
- **Collect and analyze** your metrics: Perform regular proactive review to identify trends and determine responses.
- **Validate insights**: Review your analysis results and responses with cross-functional teams and business owners. Adjust responses as appropriate.
- **Business-level view of operations**: Determine if you are satisfying customer needs and can identify areas in need of improvement in order to reach business goals.

OPS 5: How do you manage operational events?

Prioritize operational events by their business impact. Define processes for the response to any event upon which you will generate an alert, and identify the response owners. Automate the execution of these responses where appropriate. Communicate changes in operation status so that affected parties can respond as necessary. Determine the root cause of unplanned events and unexpected impacts from planned events so that the information can be used to mitigate future occurrences. Share this information with affected parties as appropriate.

Best Practices:

- **Determine priority of operational events based on business impact**: When multiple events require intervention, priority is based on business impact.
- **Event, incident, and problem management processes**: Processes are in place to address observed events, events that require intervention (incidents), and events that require intervention and either recur and/or cannot currently be resolved (problems).

- **Process per alert**: Any event upon which you raise an alert should have a well-defined response (runbook or playbook) with a specifically identified owner (for example, individual, team, or role).
- **Define escalation paths**: Runbooks and playbooks should have a definition for what triggers escalation, process for escalation, and specifically identify owners for each action. Escalations may include third parties (for example, vendors, AWS Support).
- **Identify decision makers**: Where actions have potential impacts to business outcomes, decision makers are identified who are empowered to make decisions regarding course of action on the behalf of the organization.
- **Communicate status through dashboards**: Dashboards exist where the current operating status of the business is communicated, tailored to the target audiences (for example, internal technical teams, leadership, and customers). Examples include CloudWatch dashboard, Personal Health Dashboard (PHD), and Service Health Dashboard (SHD).
- **Push notifications**: Communicate with your users when the services they consume are being impacted and when they return to normal operating conditions (for example, via email or SMS).
- **Root cause analysis process**: A process is in place that identifies and documents the root cause of an event
- **Communicate root cause**: Well-understood root causes and the impact of events are communicated as appropriate, tailored to the target audiences.

Evolve

OPS 6: How do you evolve operations?

Dedicate work cycles to making continuous incremental improvements. Evaluate opportunities for improvement captured from feedback loops, lessons learned, analysis, and cross-team reviews. Validate opportunities and prioritize them. Make changes and evaluate outcomes. If the desired results are not met consider trying another approach. Share what you learn, and the benefits, across teams.

Best Practices:

- **Processes for continuous improvement**: Your operations processes include dedicated work cycles to make continuous incremental improvements possible. Opportunities are evaluated and prioritized for action.
- **Drivers for improvement**: Consider desired features, capabilities, and improvements; unacceptable issues, bugs, and vulnerabilities; and updates required to maintain compliance with policy or support from a vendor when evaluating opportunities for improvement.
- **Feedback loops**: Your procedures include feedback loops to identify areas for improvement.
- **Lessons learned**: You have procedures to capture and document the lessons learned from the execution of operations activities so that they can be leveraged by other teams.
- **Share learnings**: You have procedures in place to share lessons learned across teams
- **Analysis of lessons learned**: You have procedures in place to look at trends in learnings and identify areas to investigate for improvement opportunities
- **Operations metrics review**: Perform retrospective analysis of operations metrics with participants spanning the business to determine opportunities and methods for improvement.
- **Making changes**: Implement changes to bring about improvement and evaluate the results to determine success.

Security

Identity and Access Management

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

The AWS account root user 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 IAM users, associate them with an administrator group, and use the IAM user to manage the account. The AWS account root user should not have API keys, should have a strong password, and should be associated with a hardware 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 API calls. Note that some resellers and Regions do not distribute or support the AWS account root user credentials.

Best Practices:

- **MFA and minimal use of root**: The AWS account root user credentials are only used for 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 you to segregate defined roles and responsibilities of system users by creating user groups. User groups can be defined using several different technologies: IAM groups, IAM roles for cross-account access, web identities, via Security Assertion Markup Language (SAML) integration (for example, defining the roles in Active Directory), or by using a third-party solution (for example, 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 (for example, 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 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 have 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**: Store 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 (for example, CloudWatch Logs, Splunk, Papertrail) or stored in an S3 bucket for later processing based on business needs. Common sources of logs are AWS APIs and user-related logs (for example, AWS CloudTrail), AWS service-specific logs (for example, Amazon S3, Amazon CloudFront), operating system-generated logs, and third-party application-specific logs. You can use CloudWatch Logs to monitor, store, and access your log files from EC2 instances, AWS CloudTrail, and other sources.

Best Practices:

- **Activity monitored appropriately**: CloudWatch Logs, events, VPC Flow Logs, ELB logs, S3 bucket logs, etc.
- **Enable AWS CloudTrail**.
- **Monitor 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 (NACLs) that protect the subnets in Amazon VPC. The current best practice is to run a system in a VPC, and define the role-based security in security groups (for example, web tier, app tier, etc.), and define the location-based security in NACLs (for example, an ELB 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 (for example, Amazon S3 bucket policies, Amazon SQS, Amazon DynamoDB, AWS KMS key policies, etc.).

Best Practices:

- **Use additional features where appropriate.**

SEC 7: How are you protecting the integrity of the operating system?

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

Best Practices:

- **File integrity**: Use file integrity controls for EC2 instances.
- **EC2 intrusion detection**: Use host-based intrusion detection controls for EC2 instances.
- **AWS Marketplace or APN Partner solution**: Use a solution from the AWS Marketplace or from an APN Partner.
- **Configuration management tool**: Use a custom AMI or configuration management tools (such as Puppet or Chef) that are secured by default.

Data Protection

SEC 8: 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 how the data is protected (for example, through encryption or access control).

Best Practices:

- **Use a data classification schema**.
- **Treat all data 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 (for example, SDK-supported, operating system-supported, Windows Bitlocker, dm-crypt, Trend Micro, SafeNet) and server-side (for example, Amazon S3). You can also use SSE and Amazon EBS 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. Best practice is to not hard-code these secrets into management scripts and applications, but it does often occur.

Best Practices:

- **Use AWS CloudHSM**.
- **Use AWS service controls**: Encrypt data at rest using AWS service-specific controls (for example, Amazon S3 SSE, Amazon EBS encrypted volumes, Amazon RDS Transparent Data Encryption (TDE)).
- **Use client-side encryption**: Encrypt data at rest using client-side techniques.
- **AWS Marketplace or APN Partner solution**: Use a solution from the AWS Marketplace or from an APN Partner (for example, SafeNet, Trend Micro).

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, you can use various techniques within your EC2 instances.

Best Practices:

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

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

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. You should evaluate your AWS service needs and request appropriate changes to your 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, for example, 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 is not needed**.
- **Highly available connectivity between AWS and on-premises environment (as applicable)**: Use multiple DX circuits, multiple VPN tunnels, AWS Marketplace appliances as applicable.
- **Highly available network connectivity for workload users**: Use a 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 VPC 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, for example, 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, for example, 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 your 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 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 (for example, web server, database) for resiliency, which includes monitoring, self-healing, and notification of significant event disruption and failure.

Best Practices:

- **Multiple Availability Zones and Regions**: Distribute application loads across multiple Availability Zones and Regions (using for example, DNS, ELB, Application Load Balancer, and 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 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 (for example, using Chaos Monkey), ensuring coverage of failure pathways.
- **Schedule game days**: 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 AMIs and the system configuration state are up-to-date at the DR site or Region.
- **DR tested and validated**: Regularly test failover to DR to ensure that RTO and RPO are met.
- **Automated recovery implemented**: Use AWS and/or third-party tools to automate system recovery.

Performance Efficiency

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, and 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 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**: Trigger-based actions set alarms that cause automated actions to remediate or escalate issues.

Tradeoffs

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

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 a serverless and container-based approach**: Use AWS Lambda, Amazon S3, Amazon DynamoDB, and Amazon ECS to reduce cost.
- **Optimize using appropriate storage solutions**: Use the most cost-effective storage solution based on usage patterns (for example, Amazon EBS cold storage, Amazon S3 Standard-Infrequent Access, Amazon Glacier).
- **Optimize using appropriate databases**: Use Amazon RDS (Postgres, MySQL, SQL Server, Oracle Server) or Amazon DynamoDB (or other key-value stores, NoSQL alternatives) where appropriate.
- **Optimize using other application-level services**: Use Amazon SQS, Amazon SNS, and Amazon Simple Email Service (Amazon 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 (PIOPS), 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 (for example, Amazon EC2, Amazon DynamoDB, Amazon S3, Amazon CloudFront).
- **Spot**: Use Spot Instances (for example, Spot block, fleet) for select workloads (for example, batch, EMR).
- **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 (for example, using Amazon Kinesis or Amazon 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 (for example, Black Friday).

Expenditure Awareness

COST 5: Do 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 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 (for example, 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 (for example, 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 AWS 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.

Notes

[1] http://pubs.opengroup.org/architecture/togaf9-doc/arch/

[2] https://www.zachman.com/about-the-zachman-framework

[3] https://www.amazon.com/p/feature/p34qgjcv93n37yd

[4] https://d0.awsstatic.com/whitepapers/architecture/AWS-Operational-Excellence-Pillar.pdf

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

[6] https://d0.awsstatic.com/whitepapers/architecture/AWS-Operational-Excellence-Pillar.pdf

[7] https://www.youtube.com/watch?v=esEFaY0FDKc

[8] https://d0.awsstatic.com/whitepapers/architecture/AWS-Security-Pillar.pdf

[9] http://aws.amazon.com/security/

[10] https://aws.amazon.com/compliance/

[11] http://blogs.aws.amazon.com/security/

[12] https://d0.awsstatic.com/whitepapers/architecture/AWS-Security-Pillar.pdf

[13] https://d0.awsstatic.com/whitepapers/Security/AWS%20Security%20Whitepaper.pdf

[14] https://aws.amazon.com/whitepapers/aws-security-best-practices/

[15] https://d0.awsstatic.com/whitepapers/compliance/AWS_Risk_and_Compliance_Whitepaper.pdf

[16] https://www.youtube.com/watch?v=OEK7mHn4JLs

[17] https://www.youtube.com/watch?v=U632-ND7dKQ

[18] https://d0.awsstatic.com/whitepapers/architecture/AWS-Reliability-Pillar.pdf

[19] http://docs.aws.amazon.com/general/latest/gr/aws_service_limits.html

[20] http://aws.amazon.com/about-aws/whats-new/2014/06/19/amazon-ec2-service-limits-report-now-available/

[21] http://docs.aws.amazon.com/waf/latest/developerguide/shield-chapter.html

[22] http://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html

[23] http://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html

[24] http://docs.aws.amazon.com/kms/latest/developerguide/overview.html

[25] https://d0.awsstatic.com/whitepapers/architecture/AWS-Reliability-Pillar.pdf

[26] http://d0.awsstatic.com/whitepapers/Backup_Archive_and_Restore_Approaches_Using_AWS.pdf

[27] http://d0.awsstatic.com/whitepapers/managing-your-aws-infrastructure-at-scale.pdf

[28] http://media.amazonwebservices.com/AWS_Disaster_Recovery.pdf

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

[30] https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/

[31] https://www.youtube.com/watch?v=wrY7XoOnysg

[3232] https://d0.awsstatic.com/analyst-reports/Benchmarking%20Availability%20and%20Reliability%20in%20the%20Cloud_Nucleus%20Research_2014%20.pdf

[33] https://aws.amazon.com/premiumsupport/

[34] https://aws.amazon.com/premiumsupport/trustedadvisor/

[35] https://d0.awsstatic.com/whitepapers/architecture/AWS-Performance-Efficiency-Pillar.pdf

[36] http://docs.aws.amazon.com/AmazonS3/latest/dev/PerformanceOptimization.html

[37] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSPerformance.html

[38] https://d0.awsstatic.com/whitepapers/architecture/AWS-Performance-Efficiency-Pillar.pdf

[39] https://www.youtube.com/watch?v=n28lDDdlnVg

[40] https://www.youtube.com/watch?v=agQMFIWr2h4

[41] https://d0.awsstatic.com/whitepapers/architecture/AWS-Cost-Optimization-Pillar.pdf

[42] http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/cost-explorer-what-is.html

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

[44] http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/detailed-billing-reports.html

[45] https://d0.awsstatic.com/whitepapers/architecture/AWS-Cost-Optimization-Pillar.pdf

[46] https://www.youtube.com/watch?v=mqY8xfKU5yE

[47] http://aws.amazon.com/tco-calculator/

[48] http://calculator.s3.amazonaws.com/index.html