Unable to locate subtitle

AWS Well-Architected Framework (2023-04-10)

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Well-Architected Framework (2023-04-10): ***Unable to locate subtitle***

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

..... vi
Abstract and introduction 1
Introduction 1
Definitions 2
On architecture 4
General design principles 6
The pillars of the framework 8
Operational excellence 8
Design principles 9
Definition 9
Best practices 10
Resources 18
Security 19
Design principles 19
Definition 20
Best practices 21
Resources 30
Reliability 31
Design principles 31
Definition 32
Best practices 33
Resources 38
Performance efficiency 38
Design principles 39
Definition 39
Best practices 40
Resources 47
Cost optimization 47
Design principles 48
Definition 48
Best practices 49
Resources 55
Sustainability 55
Design principles 56

Definition 57
Best practices 57
Resources 63
The review process 65
Conclusion 67
Contributors 68
Further reading 69
Document revisions 70
Appendix: Questions and best practices 73
Operational excellence 73
Organization 73
Prepare 108
Operate 168
Evolve 207
Security 223
Security foundations 224
Identity and access management 241
Detection 283
Infrastructure protection 293
Data protection 311
Incident response 332
Application security 347
Reliability 366
Foundations 366
Workload architecture 404
Change management 444
Failure management 475
Performance efficiency 562
Selection 562
Review 652
Monitoring 657
Tradeoffs 668
Cost optimization 677
Practice Cloud Financial Management 678
Expenditure and usage awareness 698
Cost-effective resources 737

Manage demand and supply resources 765
Optimize over time 775
Sustainability 783
Region selection 783
Alignment to demand 785
Software and architecture 797
Data 808
Hardware and services 826
Process and culture 836
Notices 844
AWS Glossary 845

This is an earlier version of the AWS Well-Architected Framework. For the latest version, see AWS Well-Architected Framework.

AWS Well-Architected Framework v10

Publication date: **April 10, 2023** (*Document revisions*)

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, cost-effective, and sustainable systems in the cloud.

Introduction

The AWS Well-Architected Framework helps you understand the pros and cons of decisions you make while building systems on AWS. Using the Framework helps you learn architectural best practices for designing and operating secure, reliable, efficient, cost-effective, and sustainable workloads in the AWS Cloud. It provides a way for you to consistently measure your architectures against best practices and identify areas for improvement. The process for reviewing an architecture is a constructive conversation about architectural decisions, and is not an audit mechanism. 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 help 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 framework is intended for those in technology roles, such as chief technology officers (CTOs), architects, developers, and operations team members. It describes AWS best practices and strategies to use when designing and operating a cloud workload, and provides links to further implementation details and architectural patterns. For more information, see the AWS Well-Architected homepage.

AWS also provides a service for reviewing your workloads at no charge. The AWS Well-Architected Tool (AWS WA Tool) is a service in the cloud that provides a consistent process for you to review and measure your architecture using the AWS Well-Architected Framework. The AWS WA Tool provides recommendations for making your workloads more reliable, secure, efficient, and cost-effective.

To help you apply best practices, we have created AWS Well-Architected Labs, which provides you with a repository of code and documentation to give you hands-on experience implementing best practices. We also have teamed up with select AWS Partner Network (APN) Partners, who are members of the AWS Well-Architected Partner program. These AWS Partners have deep AWS knowledge, and can help you review and improve your workloads.

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 six pillars — operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability.

Table 1. The pillars of the AWS Well-Architected Framework

Name	Description
Operational excellence	The ability to support development and run workloads effectively, gain insight into their operations, and to continuously improve supporting processes and procedures to deliver business value.
Security	The security pillar describes how to take advantage of cloud technologies to protect

Name	Description
	data, systems, and assets in a way that can improve your security posture.
Reliability	The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This includes the ability to operate and test the workload through its total lifecycle . This paper provides in-depth, best practice guidance for implementing reliable workloads on AWS.
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 run systems to deliver business value at the lowest price point.
Sustainability	The ability to continually improve sustainab ility impacts by reducing energy consumption and increasing efficiency across all component s of a workload by maximizing the benefits from the provisioned resources and minimizin g the total resources required.

In the AWS Well-Architected Framework, we use these terms:

- A **component** is the code, configuration, and AWS Resources that together deliver against a requirement. A component is often the unit of technical ownership, and is decoupled from other components.
- The term **workload** is used to identify a set of components that together deliver business value. A workload is usually the level of detail that business and technology leaders communicate about.

- We think about **architecture** as being how components work together in a workload. How components communicate and interact is often the focus of architecture diagrams.
- **Milestones** mark key changes in your architecture as it evolves throughout the product lifecycle (design, implementation, testing, go live, and in production).
- Within an organization the **technology portfolio** is the collection of workloads that are required for the business to operate.
- The **level of effort** is categorizing the amount of time, effort, and complexity a task requires for implementation. Each organization needs to consider the size and expertise of the team and the complexity of the workload for additional context to properly categorize the level of effort for the organization.
  - **High:** The work might take multiple weeks or multiple months. This could be broken out into multiple stories, releases, and tasks.
  - **Medium:** The work might take multiple days or multiple weeks. This could be broken out into multiple releases and tasks.
  - **Low:** The work might take multiple hours or multiple days. This could be broken out into multiple tasks.

When architecting workloads, you make trade-offs between pillars based on your business context. These business decisions can drive your engineering priorities. You might optimize to improve sustainability impact and reduce cost at the expense of reliability in development environments, or, for mission-critical solutions, you might optimize reliability with increased costs and sustainability impact. 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 verify they are following best practice. Technology architecture teams typically include a set of roles such as: Technical Architect (infrastructure), Solutions Architect (software), Data Architect, Networking Architect, and Security Architect. Often these teams use TOGAF or the Zachman Framework as part of an enterprise architecture capability.

At AWS, 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, verifying that teams are meeting internal standards. We mitigate these risks in two ways. First, we have *practices* (ways of doing things, process, standards, and accepted norms) that focus on allowing each team to have that capability, and we put in place experts who verify that teams raise the bar on the standards they need to meet. Second, we implement *mechanisms* that carry out automated checks to verify standards are being met.

> "Good intentions never work, you need good mechanisms to make anything happen" — Jeff Bezos.

This means replacing a human's best efforts with mechanisms (often automated) that check for compliance with rules or process. This distributed approach is supported by the Amazon leadership principles, and establishes a culture across all roles that *works back* from the customer. 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. 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 activate 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 practice, 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 verify that teams follow them. In time, these best practices are formalized into our internal review processes, and also into mechanisms that enforce compliance. The Well-Architected Framework 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. The Well-Architected Framework 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 permit 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, training, 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**: If you make a poor capacity decision when deploying a workload, 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 with architectural experimentation in mind**: Automation permits you to create and replicate your workloads 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.
- **Consider evolutionary architectures**: In a traditional environment, architectural decisions are often implemented as static, onetime events, with a few major versions of a system during its lifetime. As a business and its context continue to evolve, 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 permits 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 pillars of the 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 six pillars of operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability, 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.

Pillars

- Operational excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization
- Sustainability

Operational excellence

The Operational Excellence pillar includes the ability to support development and run workloads effectively, gain insight into their operations, and to continuously improve supporting processes and procedures to deliver business value.

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.

Topics

- Design principles
- Definition
- Best practices
- Resources

Design principles

There are five 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) as code and update it with code. You can implement your operations procedures as code and automate their run process by initiating them in response to events. By performing operations as code, you limit human error and achieve consistent responses to events.
- **Make frequent, small, reversible changes**: Design workloads to permit 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 verify that they are effective, and that teams are familiar with their process. 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 four best practice areas for operational excellence in the cloud:

- **Organization**
- **Prepare**
- **Operate**
- **Evolve**

Your organization's leadership defines business objectives. Your organization must understand requirements and priorities and use these to organize and conduct work to support the achievement of business outcomes. Your workload must emit the information necessary to support it. Implementing services to achieve integration, deployment, and delivery of your workload will create an increased flow of beneficial changes into production by automating repetitive processes.

There may be risks inherent in the operation of your workload. Understand those risks and make an informed decision to enter production. Your teams must be able to support your workload. Business and operational metrics derived from desired business outcomes will permit you to understand the health of your workload, your operations activities, and respond to incidents. Your priorities will change as your business needs and business environment changes. Use these as a feedback loop to continually drive improvement for your organization and the operation of your workload.

Best practices

Topics

- Organization
- Prepare
- Operate
- Evolve

Organization

Your teams must have a shared understanding of your entire workload, their role in it, and shared business goals to set the priorities that will achieve business success. Well-defined priorities will maximize the benefits of your efforts. Evaluate internal and external customer needs involving key stakeholders, including business, development, and operations teams, to determine where to focus efforts. Evaluating customer needs will verify that you have a thorough understanding of the support that is required to achieve business outcomes. Verify that you are aware of guidelines or obligations defined by your organizational governance and external factors, such as regulatory compliance requirements and industry standards that may mandate or emphasize specific focus. Validate that you have mechanisms to identify changes to internal governance and external compliance requirements. If no requirements are identified, validate that you have applied due diligence to this determination. Review your priorities regularly so that they can be updated as needs change.

Evaluate threats to the business (for example, business risk and liabilities, and information security threats) and maintain this information in a risk registry. Evaluate the impact of risks, and tradeoffs between competing interests or alternative approaches. For example, accelerating speed to market for new features may be emphasized over cost optimization, or you may choose a relational database for non-relational data to simplify the effort to migrate a system without refactoring. Manage benefits and risks to make informed decisions when determining where to focus efforts. Some risks or choices may be acceptable for a time, it may be possible to mitigate associated risks, or it may become unacceptable to permit a risk to remain, in which case you will take action to address the risk.

Your teams must understand their part in achieving business outcomes. Teams must understand their roles in the success of other teams, the role of other teams in their success, and have shared goals. Understanding responsibility, ownership, how decisions are made, and who has authority to make decisions will help focus efforts and maximize the benefits from your teams. The needs of a team will be shaped by the customer they support, their organization, the makeup of the team, and the characteristics of their workload. It's unreasonable to expect a single operating model to be able to support all teams and their workloads in your organization.

Verify that there are identified owners for each application, workload, platform, and infrastructure component, and that each process and procedure has an identified owner responsible for its definition, and owners responsible for their performance.

Having understanding of the business value of each component, process, and procedure, of why those resources are in place or activities are performed, and why that ownership exists will inform the actions of your team members. Clearly define the responsibilities of team members so that they may act appropriately and have mechanisms to identify responsibility and ownership. Have mechanisms to request additions, changes, and exceptions so that you do not constrain innovation. Define agreements between teams describing how they work together to support each other and your business outcomes.

Provide support for your team members so that they can be more effective in taking action and supporting your business outcomes. Engaged senior leadership should set expectations and measure success. Senior leadership should be the sponsor, advocate, and driver for the adoption of best practices and evolution of the organization. Let team members take action when outcomes are at risk to minimize impact and encourage them to escalate to decision makers and stakeholders when they believe there is a risk so that it can be addressed and incidents avoided. Provide timely, clear, and actionable communications of known risks and planned events so that team members can take timely and appropriate action.

Encourage experimentation to accelerate learning and keep team members interested and engaged. Teams must grow their skill sets to adopt new technologies, and to support changes in demand and responsibilities. Support and encourage this by providing dedicated structured time for learning. Verify that your team members have the resources, both tools and team members, to be successful and scale to support your business outcomes. Leverage cross-organizational diversity to seek multiple unique perspectives. Use this perspective to increase innovation, challenge your assumptions, and reduce the risk of confirmation bias. Grow inclusion, diversity, and accessibility within your teams to gain beneficial perspectives.

If there are external regulatory or compliance requirements that apply to your organization, you should use the resources provided by AWS Cloud Compliance to help educate your teams so that they can determine the impact on your priorities. The Well-Architected Framework emphasizes learning, measuring, and improving. It provides a consistent approach for you to evaluate architectures, and implement designs that will scale over time. AWS provides the AWS Well-Architected Tool to help you review your approach before development, the state of your workloads before production, and the state of your workloads in production. You can compare workloads to the latest AWS architectural best practices, monitor their overall status, and gain insight into potential risks. AWS Trusted Advisor is a tool that provides access to a core set of checks that recommend optimizations that may help shape your priorities. Business and Enterprise Support customers receive access to additional checks focusing on security, reliability, performance, cost-optimization, and sustainability that can further help shape their priorities.

AWS can help you educate your teams about AWS and its services to increase their understanding of how their choices can have an impact on your workload. Use the resources provided by AWS Support (AWS Knowledge Center, AWS Discussion Forums, and AWS Support Center) and AWS Documentation to educate your teams. Reach out to AWS Support through AWS Support Center for help with your AWS questions. AWS also shares best practices and patterns that we have learned through the operation of AWS in The Amazon Builders' Library. A wide variety of other useful information is available through the AWS Blog and The Official AWS Podcast. AWS Training and Certification provides some training through self-paced digital courses on AWS fundamentals. You can also register for instructor-led training to further support the development of your teams' AWS skills.

Use tools or services that permit you to centrally govern your environments across accounts, such as AWS Organizations, to help manage your operating models. Services like AWS Control Tower expand this management capability by allowing you to define blueprints (supporting your operating models) for the setup of accounts, apply ongoing governance using AWS Organizations, and automate provisioning of new accounts. Managed Services providers such as AWS Managed

Services, AWS Managed Services Partners, or Managed Services Providers in the AWS Partner Network, provide expertise implementing cloud environments, and support your security and compliance requirements and business goals. Adding Managed Services to your operating model can save you time and resources, and lets you keep your internal teams lean and focused on strategic outcomes that will differentiate your business, rather than developing new skills and capabilities.

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

OPS 1: How do you determine what your priorities are?

Everyone must understand their part in achieving business success. Have shared goals in order to set priorities for resources. This will maximize the benefits of your efforts.

OPS 2: How do you structure your organization to support your business outcomes?

Your teams must understand their part in achieving business outcomes. Teams must understand their roles in the success of other teams, the role of other teams in their success, and have shared goals. Understanding responsibility, ownership, how decisions are made, and who has authority to make decisions will help focus efforts and maximize the benefits from your teams.

OPS 3: How does your organizational culture support your business outcomes?

Provide support for your team members so that they can be more effective in taking action and supporting your business outcome.

You might find that you want to emphasize a small subset of your priorities at some point in time. Use a balanced approach over the long term to verify the development of needed capabilities and management of risk. Review your priorities regularly and update them as needs change. When responsibility and ownership are undefined or unknown, you are at risk of both not performing necessary action in a timely fashion and of redundant and potentially conflicting efforts emerging to address those needs. Organizational culture has a direct impact on team member job satisfaction and retention. Activate the engagement and capabilities of your team

members to achieve the success of your business. Experimentation is required for innovation to happen and turn ideas into outcomes. Recognize that an undesired result is a successful experiment that has identified a path that will not lead to success.

Prepare

To prepare for operational excellence, you have to understand your workloads and their expected behaviors. You will then be able to design them to provide insight to their status and build the procedures to support them.

Design your workload so that it provides the information necessary for you to understand its internal state (for example, metrics, logs, events, and traces) across all components in support of observability and investigating issues. Iterate to develop the telemetry necessary to monitor the health of your workload, identify when outcomes are at risk, and activate effective responses. When instrumenting your workload, capture a broad set of information to achieve situational awareness (for example, changes in state, user activity, permission access, utilization counters), knowing that you can use filters to select the most useful information over time.

Adopt approaches that improve the flow of changes into production and that achieves refactoring, fast feedback on quality, and bug fixing. These accelerate beneficial changes entering production, limit issues deployed, and activate rapid identification and remediation of issues introduced through deployment activities or discovered in your environments.

Adopt approaches that provide fast feedback on quality and achieves rapid recovery from changes that do not have desired outcomes. Using these practices mitigates the impact of issues introduced through the deployment of changes. Plan for unsuccessful changes so that you are able to respond faster if necessary and test and validate the changes you make. Be aware of planned activities in your environments so that you can manage the risk of changes impacting planned activities. Emphasize frequent, small, reversible changes to limit the scope of change. This results in faster troubleshooting and remediation with the option to roll back a change. It also means you are able to get the benefit of valuable changes more frequently.

Evaluate the operational readiness of your workload, processes, procedures, and personnel to understand the operational risks related to your workload. Use a consistent process (including manual or automated checklists) to know when you are ready to go live with your workload or a change. This will also help you to find any areas that you must make plans to address. Have runbooks that document your routine activities and playbooks that guide your processes for issue resolution. Understand the benefits and risks to make informed decisions to permit changes to enter production.

AWS allows you to view your entire workload (applications, infrastructure, policy, governance, and operations) as code. This means you can apply the same engineering discipline that you use for application code to every element of your stack and share these across teams or organizations to magnify the benefits of development efforts. Use operations as code in the cloud and the ability to safely experiment to develop your workload, your operations procedures, and practice failure. Using CloudFormation allows you to have consistent, templated, sandbox development, test, and production environments with increasing levels of operations control.

The following questions focus on these considerations for operational excellence.

OPS 4: How do you design your workload so that you can understand its state?

Design your workload so that it provides the information necessary across all components (for example, metrics, logs, and traces) for you to understand its internal state. This allows you to provide effective responses when appropriate.

OPS 5: How do you reduce defects, ease remediation, and improve flow into production?

Adopt approaches that improve flow of changes into production that achieve refactoring fast feedback on quality, and bug fixing. These accelerate beneficial changes entering productio n, limit issues deployed, and achieve rapid identification and remediation of issues introduced through deployment activities.

OPS 6: How do you mitigate deployment risks?

Adopt approaches that provide fast feedback on quality and achieve rapid recovery from changes that do not have desired outcomes. Using these practices mitigates the impact of issues introduced through the deployment of changes.

OPS 7: How do you know that you are ready to support a workload?

Evaluate the operational readiness of your workload, processes and procedures, and personnel to understand the operational risks related to your workload.

Invest in implementing operations activities as code to maximize the productivity of operations personnel, minimize error rates, and achieve automated responses. Use “pre-mortems” to anticipate failure and create procedures where appropriate. Apply metadata using Resource Tags and AWS Resource Groups following a consistent tagging strategy to achieve identification of your resources. Tag your resources for organization, cost accounting, access controls, and targeting the running of automated operations activities. Adopt deployment practices that take advantage of the elasticity of the cloud to facilitate development activities, and pre-deployment of systems for faster implementations. When you make changes to the checklists you use to evaluate your workloads, plan what you will do with live systems that no longer comply.

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 metrics that will be used in those calculations to determine if your workload and operations are successful. Operational health includes both the health of the workload and the health and success of the operations activities performed in support of the workload (for example, deployment and incident response). Establish metrics baselines for improvement, investigation, and intervention, 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 investigation and resolution of issues. Prioritize responses to events based on their business and customer impact. Verify that if an alert is raised in response to an event, there is an associated process to run with a specifically identified owner. Define in advance the personnel required to resolve an event and include escalation processes to engage additional personnel, as it becomes necessary, based on urgency and impact. Identify and engage individuals with the authority to make a decision 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.

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 to identify workload issues in support of root cause analysis and remediation.

The following questions focus on these considerations for operational excellence.

OPS 8: How do you understand the health of your workload?

Define, capture, and analyze workload metrics to gain visibility to workload events so that you can take appropriate action.

OPS 9: How do you understand the health of your operations?

Define, capture, and analyze operations metrics to gain visibility to operations events so that you can take appropriate action.

OPS 10: How do you manage workload and operations events?

Prepare and validate procedures for responding to events to minimize their disruption to your workload.

All of the metrics you collect should be aligned to a business need and the outcomes they support. Develop scripted responses to well-understood events and automate their performance in response to recognizing the event.

Evolve

Learn, share, and continuously improve to sustain operational excellence. Dedicate work cycles to making nearly continuous incremental improvements. Perform post-incident analysis of all customer impacting events. Identify the contributing factors and preventative action to limit or prevent recurrence. Communicate contributing factors with affected communities as appropriate. 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 running 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.

On AWS, you can export your log data to Amazon S3 or send logs directly to Amazon S3 for long-term storage. Using AWS Glue, you can discover and prepare your log data in Amazon S3 for analytics, and store associated metadata in the AWS Glue Data Catalog. Amazon Athena, through its native integration with AWS Glue, can then be used to analyze your log data, querying it using standard SQL. Using a business intelligence tool like Amazon Quick, you can visualize, explore, and analyze your data. Discovering trends and events of interest that may drive improvement.

The following question focuses on these considerations for operational excellence.

OPS 11: How do you evolve operations?

Dedicate time and resources for nearly continuous incremental improvement to evolve the effectiveness and efficiency of your 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.

Resources

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

Documentation

- DevOps and AWS

Whitepaper

- Operational Excellence Pillar

Video

- DevOps at Amazon

Security

The Security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security.

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.

Topics

- Design principles
- Definition
- Best practices
- Resources

Design principles

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

- **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 identity management, and aim to eliminate reliance on long-term static credentials.
- **Maintain traceability:** Monitor, alert, and audit actions and changes to your environment in real time. Integrate log and metric collection with systems to automatically investigate and take action.
- **Apply security at all layers:** Apply a defense in depth approach with multiple security controls. Apply to all layers (for example, edge of network, VPC, load balancing, every instance and compute service, operating system, application, and code).

- **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, tokenization, and access control where appropriate.
- **Keep people away from data:** Use mechanisms and tools to reduce or eliminate the need for direct access or manual processing of data. This reduces the risk of mishandling or modification and human error when handling sensitive data.
- **Prepare for security events:** Prepare for an incident by having incident management and investigation policy and processes that align 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 seven best practice areas for security in the cloud:

- Security foundations
- Identity and access management
- Detection
- Infrastructure protection
- Data protection
- Incident response
- Application security

Before you architect any workload, 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 helps 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

Topics

- Security
- Identity and access management
- Detection
- Infrastructure protection
- Data protection
- Incident response
- Application security

Security

The following question focuses on these considerations for security. (For a list of security questions and best practices, see the Appendix.).

SEC 1: How do you securely operate your workload?

To operate your workload securely, you must apply overarching best practices to every area of security. Take requirements and processes that you have defined in operational excellence at an organizational and workload level, and apply them to all areas.

Staying up to date with recommendations from AWS, industry sources, and threat intelligence helps you evolve your threat model and control objectives. Automating security processes, testing, and validation allow you to scale your security operations.

In AWS, segregating different workloads by account, based on their function and compliance or data sensitivity requirements, is a recommended approach.

Identity and access management

Identity and access management are key parts of an information security program, ensuring that only authorized and authenticated users and components are able to access your resources, and only in a manner that you intend. For example, you should define principals (that is, accounts, users, roles, and services that can perform actions in your account), build out policies aligned with these principals, and implement strong credential management. These privilege-management elements form the core 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 and programmatic 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 enforcing multi-factor authentication (MFA). You can use federation with your existing directory service. For workloads that require systems to have access to AWS, IAM allows for secure access through roles, instance profiles, identity federation, and temporary credentials.

The following questions focus on these considerations for security.

SEC 2: How do you manage identities for people and machines?

There are two types of identities you need to manage when approaching operating secure AWS workloads. Understanding the type of identity you need to manage and grant access helps you verify the right identities have access to the right resources under the right conditions.

Human Identities: Your administrators, developers, operators, and end users require an identity to access your AWS environments and applications. These are members of your organization, or external users with whom you collaborate, and who interact with your AWS resources via a web browser, client application, or interactive command line tools.

Machine Identities: Your service applications, operational tools, and workloads require an identity to make requests to AWS services, for example, to read data. These identities include machines running in your AWS environment such as Amazon EC2 instances or AWS Lambda functions. You may also manage machine identities for external parties who need access. Additionally, you may also have machines outside of AWS that need access to your AWS environment.

SEC 3: How do you manage permissions for people and machines?

Manage permissions to control access to people and machine identities that require access to AWS and your workload. Permissions control who can access what, and under what conditions.

Credentials must not be shared between any user or system. User access should be granted using a least-privilege approach with best practices including password requirements and MFA enforced. Programmatic access, including API calls to AWS services, should be performed using temporary and limited-privilege credentials, such as those issued by the AWS Security Token Service.

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use.

Which user needs programmatic access?	To	By
		• For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

AWS provides resources that can help you with identity and access management. To help learn best practices, explore our hands-on labs on managing credentials & authentication, controlling human access, and controlling programmatic access.

Detection

You can use detective controls to identify a potential security threat or incident. They 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. You can also use internal auditing, an examination of controls related to information systems, to verify 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. Amazon GuardDuty is a managed threat detection service that continuously monitors for malicious or unauthorized behavior to help you protect your AWS accounts and workloads. Service-level logs are also available, for example, you can use Amazon Simple Storage Service (Amazon S3) to log access requests.

The following question focuses on these considerations for security.

SEC 4: How do you detect and investigate security events?

Capture and analyze events from logs and metrics to gain visibility. Take action on security events and potential threats to help secure your workload.

Log management is important to a Well-Architected workload 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 encompasses control methodologies, such as defense in depth, necessary to meet best practices and organizational or regulatory obligations. Use of these methodologies is critical for successful, ongoing operations in either the cloud or on-premises.

In AWS, you can implement stateful and stateless packet inspection, either by using AWS-native technologies or by using partner products and services available through the AWS Marketplace. You 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 these considerations for security.

SEC 5: How do you protect your network resources?

Any workload that has some form of network connectivity, whether it's the internet or a private network, requires multiple layers of defense to help protect from external and internal network-based threats.

SEC 6: How do you protect your compute resources?

Compute resources in your workload require multiple layers of defense to help protect from external and internal threats. Compute resources include EC2 instances, containers, AWS Lambda functions, database services, IoT devices, and more.

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 Elastic Container Service (Amazon ECS) container, or AWS Elastic Beanstalk instance, and persist this configuration to an immutable Amazon Machine Image (AMI). Then, whether launched 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 way of rendering it unintelligible to unauthorized access. These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

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 Standard, S3 Standard–IA, S3 One Zone-IA, and Amazon Glacier are all designed to provide 99.999999999% durability of objects over a given year. This durability level corresponds to an average annual expected loss of 0.000000001% of objects.
- 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 use a feature or leverage a service that provides that functionality.

The following questions focus on these considerations for security.

SEC 7: How do you classify your data?

Classification provides a way to categorize data, based on criticality and sensitivity in order to help you determine appropriate protection and retention controls.

SEC 8: How do you protect your data at rest?

Protect your data at rest by implementing multiple controls, to reduce the risk of unauthorized access or mishandling.

SEC 9: How do you protect your data in transit?

Protect your data in transit by implementing multiple controls to reduce the risk of unauthori zed access or loss.

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 strongly affects 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 through game days, will help you verify that your architecture can 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 launch tools that automate responses 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 these considerations for security.

SEC 10: How do you anticipate, respond to, and recover from incidents?

Preparation is critical to timely and effective investigation, response to, and recovery from security incidents to help minimize disruption to your organization.

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

Application security

Application security (AppSec) describes the overall process of how you design, build, and test the security properties of the workloads you develop. You should have appropriately trained people in your organization, understand the security properties of your build and release infrastructure, and use automation to identify security issues.

Adopting application security testing as a regular part of your software development lifecycle (SDLC) and post release processes help validate that you have a structured mechanism to identify, fix, and prevent application security issues entering your production environment.

Your application development methodology should include security controls as you design, build, deploy, and operate your workloads. While doing so, align the process for continuous defect reduction and minimizing technical debt. For example, using threat modeling in the design phase helps you uncover design flaws early, which makes them easier and less costly to fix as opposed to waiting and mitigating them later.

The cost and complexity to resolve defects is typically lower the earlier you are in the SDLC. The easiest way to resolve issues is to not have them in the first place, which is why starting with a threat model helps you focus on the right outcomes from the design phase. As your AppSec program matures, you can increase the amount of testing that is performed using automation, improve the fidelity of feedback to builders, and reduce the time needed for security reviews. All of these actions improve the quality of the software you build, and increase the speed of delivering features into production.

These implementation guidelines focus on four areas: organization and culture, security *of* the pipeline, security *in* the pipeline, and dependency management. Each area provides a set of principles that you can implement. and provides an end-to-end view of how you design, develop, build, deploy, and operate workloads.

In AWS, there are a number of approaches you can use when addressing your application security program. Some of these approaches rely on technology while others focus on the people and organizational aspects of your application security program.

The following question focuses on these considerations for application security.

SEC 11: How do you incorporate and validate the security properties of applications throughout the design, development, and deployment lifecycle?

Training people, testing using automation, understanding dependencies, and validating the security properties of tools and applications help to reduce the likelihood of security issues in production workloads.

Resources

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

Documentation

- AWS Cloud Security

- AWS Compliance
- AWS Security Blog
- AWS Security Maturity Model

Whitepaper

- Security Pillar
- AWS Security Overview
- AWS Risk and Compliance

Video

- AWS Security State of the Union
- Shared Responsibility Overview

Reliability

The Reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This includes the ability to operate and test the workload through its total lifecycle. This paper provides in-depth, best practice guidance for implementing reliable workloads on AWS.

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.

Topics

- Design principles
- Definition
- Best practices
- Resources

Design principles

There are five design principles for reliability in the cloud:

- **Automatically recover from failure**: By monitoring a workload for key performance indicators (KPIs), you can start automation when a threshold is breached. These KPIs should be a measure of business value, not of the technical aspects of the operation of the service. This provides 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.
- **Test recovery procedures**: In an on-premises environment, testing is often conducted to prove that the workload works in a particular scenario. Testing is not typically used to validate recovery strategies. In the cloud, you can test how your workload 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 approach exposes failure pathways that you can test and fix before a real failure scenario occurs, thus reducing risk.
- **Scale horizontally to increase aggregate workload availability**: Replace one large resource with multiple small resources to reduce the impact of a single failure on the overall workload. Distribute requests across multiple, smaller resources to verify that they don't share a common point of failure.
- **Stop guessing capacity**: A common cause of failure in on-premises workloads is resource saturation, when the demands placed on a workload exceed the capacity of that workload (this is often the objective of denial of service attacks). In the cloud, you can monitor demand and workload utilization, and automate the addition or removal of resources to maintain the more efficient level to satisfy demand without over- or under-provisioning. There are still limits, but some quotas can be controlled and others can be managed (see Manage Service Quotas and Constraints).
- **Manage change in automation**: Changes to your infrastructure should be made using automation. The changes that must be managed include changes to the automation, which then can be tracked and reviewed.

Definition

There are four best practice areas for reliability in the cloud:

- Foundations
- Workload architecture
- Change management
- Failure management

To achieve reliability, you must start with the foundations — an environment where Service Quotas and network topology accommodate the workload. The workload architecture of the distributed system must be designed to prevent and mitigate failures. The workload must handle changes in demand or requirements, and it must be designed to detect failure and automatically heal itself.

Best practices

Topics

- Foundations
- Workload architecture
- Change management
- Failure management

Foundations

Foundational requirements are those whose scope extends beyond a single workload or project. 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.

With AWS, most of these foundational requirements are already incorporated or can be addressed as needed. The cloud is designed to be nearly limitless, so it's the responsibility of AWS to satisfy the requirement for sufficient networking and compute capacity, permitting you to change resource size and allocations on demand.

The following questions focus on these considerations for reliability. (For a list of reliability questions and best practices, see the Appendix.).

REL 1: How do you manage Service Quotas and constraints?

For cloud-based workload architectures, there are Service Quotas (which are also referred to as service limits). These quotas exist to prevent accidentally provisioning more resources than you need and to limit request rates on API operations so as to protect services from abuse. There are also resource constraints, for example, the rate that you can push bits down a fiber-optic cable, or the amount of storage on a physical disk.

REL 2: How do you plan your network topology?

Workloads often exist in multiple environments. These include multiple cloud environments (both publicly accessible and private) and possibly your existing data center infrastructure. Plans must include network considerations such as intra- and inter-system connectivity, public IP address management, private IP address management, and domain name resolution.

For cloud-based workload architectures, there are Service Quotas (which are also referred to as service limits). These quotas exist to prevent accidentally provisioning more resources than you need and to limit request rates on API operations to protect services from abuse. Workloads often exist in multiple environments. You must monitor and manage these quotas for all workload environments. These include multiple cloud environments (both publicly accessible and private) and may include your existing data center infrastructure. Plans must include network considerations, such as intrasystem and intersystem connectivity, public IP address management, private IP address management, and domain name resolution.

Workload architecture

A reliable workload starts with upfront design decisions for both software and infrastructure. Your architecture choices will impact your workload behavior across all of the Well-Architected pillars. For reliability, there are specific patterns you must follow.

With AWS, workload developers have their choice of languages and technologies to use. AWS SDKs take the complexity out of coding by providing language-specific APIs for AWS services. These SDKs, plus the choice of languages, permits developers to implement the reliability best practices listed here. Developers can also read about and learn from how Amazon builds and operates software in The Amazon Builders' Library.

The following questions focus on these considerations for reliability.

REL 3: How do you design your workload service architecture?

Build highly scalable and reliable workloads using a service-oriented architecture (SOA) or a microservices architecture. Service-oriented architecture (SOA) is the practice of making software components reusable via service interfaces. Microservices architecture goes further to make components smaller and simpler.

REL 4: How do you design interactions in a distributed system to prevent failures?

Distributed systems rely on communications networks to interconnect components, such as servers or services. Your workload must operate reliably despite data loss or latency in these networks. Components of the distributed system must operate in a way that does not negativel y impact other components or the workload. These best practices prevent failures and improve mean time between failures (MTBF).

REL 5: How do you design interactions in a distributed system to mitigate or withstand failures?

Distributed systems rely on communications networks to interconnect components (such as servers or services). Your workload must operate reliably despite data loss or latency over these networks. Components of the distributed system must operate in a way that does not negatively impact other components or the workload. These best practices permit workloads to withstand stresses or failures, more quickly recover from them, and mitigate the impact of such impairmen ts. The result is improved mean time to recovery (MTTR).

Change management

Changes to your workload or its environment must be anticipated and accommodated to achieve reliable operation of the workload. Changes include those imposed on your workload, such as spikes in demand, and also those from within, such as feature deployments and security patches.

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

The following questions focus on these considerations for reliability.

REL 6: How do you monitor workload resources?

Logs and metrics are powerful tools to gain insight into the health of your workload. You can configure your workload to monitor logs and metrics and send notifications when thresholds are crossed or significant events occur. Monitoring allows your workload to recognize when low-performance thresholds are crossed or failures occur, so it can recover automatically in response.

REL 7: How do you design your workload to adapt to changes in demand?

A scalable workload provides elasticity to add or remove resources automatically so that they closely match the current demand at any given point in time.

REL 8: How do you implement change?

Controlled changes are necessary to deploy new functionality, and to verify that the workloads and the operating environment are running known software and can be patched or replaced in a predictable manner. If these changes are uncontrolled, then it makes it difficult to predict the effect of these changes, or to address issues that arise because of them.

When you architect a workload to automatically add and remove resources in response to changes in demand, this not only increases reliability but also validates 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 permits you to audit and quickly identify actions that might have impacted reliability. Controls on change management certify 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. Reliability requires that your workload be aware of failures as they occur and take action to avoid impact on availability. Workloads must be able to both withstand failures and automatically repair issues.

With AWS, you can take advantage of automation to react to monitoring data. For example, when a particular metric crosses a threshold, you can initiate 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 allows 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 these considerations for reliability.

REL 9: How do you back up data?

Back up data, applications, and configuration to meet your requirements for recovery time objectives (RTO) and recovery point objectives (RPO).

REL 10: How do you use fault isolation to protect your workload?

Fault isolated boundaries limit the effect of a failure within a workload to a limited number of components. Components outside of the boundary are unaffected by the failure. Using multiple fault isolated boundaries, you can limit the impact on your workload.

REL 11: How do you design your workload to withstand component failures?

Workloads with a requirement for high availability and low mean time to recovery (MTTR) must be architected for resiliency.

REL 12: How do you test reliability?

After you have designed your workload to be resilient to the stresses of production, testing is the only way to verify that it will operate as designed, and deliver the resiliency you expect.

REL 13: How do you plan for disaster recovery (DR)?

Having backups and redundant workload components in place is the start of your DR strategy. RTO and RPO are your objectives for restoration of your workload. Set these based on business needs. Implement a strategy to meet these objectives, considering locations and function of workload resources and data. The probability of disruption and cost of recovery are also key factors that help to inform the business value of providing disaster recovery for a workload.

Regularly back up your data and test your backup files to verify that you can recover from both logical and physical errors. A key to managing failure is the frequent and automated testing of workloads to cause failure, and then observe how they recover. Do this on a regular schedule

and verify that such testing is also initiated after significant workload changes. Actively track KPIs, and also the recovery time objective (RTO) and recovery point objective (RPO), to assess a workload'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 workload-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.

Resources

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

Documentation

- AWS Documentation
- AWS Global Infrastructure
- AWS Auto Scaling: How Scaling Plans Work
- What Is AWS Backup?

Whitepaper

- Reliability Pillar: AWS Well-Architected
- Implementing Microservices on AWS

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.

Topics

- Design principles
- Definition

- Best practices
- Resources

Design principles

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

- **Democratize advanced technologies**: Make advanced technology implementation smoother for your team by delegating complex tasks to your cloud vendor. Rather than asking your IT team to learn about hosting and running a new technology, consider consuming the technology as a service. For example, NoSQL databases, media transcoding, and machine learning are all technologies that require specialized expertise. In the cloud, these technologies become services that your team can consume, permitting your team to focus on product development rather than resource provisioning and management.
- **Go global in minutes**: Deploying your workload in multiple AWS Regions around the world permits you to provide lower latency and a better experience for your customers at minimal cost.
- **Use serverless architectures**: Serverless architectures remove the need for you to run and maintain physical servers for traditional compute activities. For example, serverless storage services can act as static websites (removing the need for web servers) and event services can host code. This removes the operational burden of managing physical servers, and can lower transactional costs because 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.
- **Consider mechanical sympathy**: Understand how cloud services are consumed and always use the technology approach that aligns with your workload goals. For example, consider data access patterns when you select 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 building 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.

Reviewing your choices on a regular basis validates that you are taking advantage of the continually evolving AWS Cloud. Monitoring verifies that you are aware of any deviance from expected performance. Make trade-offs in your architecture to improve performance, such as using compression or caching, or relaxing consistency requirements.

Best practices

Topics

- Selection
- Review
- Monitoring
- Tradeoffs

Selection

The more effective solution for a particular workload varies, and solutions often combine multiple approaches. Well-architected workloads use multiple solutions and activate different features to improve performance.

AWS resources are available in many types and configurations so you can find an approach that closely matches your workload needs. You can also find options that are not efficiently 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 these considerations for performance efficiency. (For a list of performance efficiency questions and best practices, see the Appendix.).

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

Often, multiple approaches are required for more effective performance across a workload. Well-architected systems use multiple solutions and features to improve performance.

Use a data-driven approach to select the patterns and implementation for your architecture and achieve a cost effective solution. AWS Solutions Architects, AWS Reference Architectures, and AWS Partner Network (APN) partners can help you select an architecture based on industry knowledge, 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 discuss the four main resource types to consider (compute, storage, database, and network).

Compute

Selecting compute resources that meet your requirements, performance needs, and provide great efficiency of cost and effort will permit you to accomplish more with the same number of resources. When evaluating compute options, be aware of your requirements for workload performance and cost requirements and use this to make informed decisions.

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

- **Instances** are virtualized servers, permitting you to change their capabilities with a button or an API call. Because resource decisions in the cloud aren’t 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 permit you to run an application and its dependencies in resource-isolated processes. AWS Fargate is serverless compute for containers or Amazon EC2 can be used if you need control over the installation, configuration, and management of your compute environment. You can also choose from multiple container orchestration platforms: Amazon Elastic Container Service (ECS) or Amazon Elastic Kubernetes Service (EKS).
- **Functions** abstract the run environment from the code you want to apply. For example, AWS Lambda permits you to run code without running an instance.

The following question focuses on these considerations for performance efficiency.

PERF 2: How do you select your compute solution?

The more efficient compute solution for a workload varies based on application design, usage patterns, and configuration settings. Architectures can use different compute solutions for various components and turn on different features to improve performance. Selecting the wrong compute solution for an architecture can lead to lower performance efficiency.

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

Storage

Cloud storage is a critical component of cloud computing, holding the information used by your workload. Cloud storage is typically more reliable, scalable, and secure than traditional on-premises storage systems. Select from object, block, and file storage services, and cloud data migration options for your workload.

In AWS, storage is available in three forms: object, block, and file:

- **Object Storage** provides a scalable, durable platform to make data accessible from any internet location for user-generated content, active archive, serverless computing, Big Data storage or backup and recovery. Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Amazon S3 is designed for 99.999999999% (11 9's) of durability, and stores data for millions of applications for companies all around the world.
- **Block Storage** provides highly available, consistent, low-latency block storage for each virtual host and is analogous to direct-attached storage (DAS) or a Storage Area Network (SAN). Amazon Elastic Block Store (Amazon EBS) is designed for workloads that require persistent storage accessible by EC2 instances that helps you tune applications with the right storage capacity, performance and cost.
- **File Storage** provides access to a shared file system across multiple systems. File storage solutions like Amazon Elastic File System (Amazon EFS) are ideal for use cases such as large content repositories, development environments, media stores, or user home directories. Amazon FSx makes it efficient and cost effective to launch and run popular file systems so you can leverage the rich feature sets and fast performance of widely used open source and commercially-licensed file systems.

The following question focuses on these considerations for performance efficiency.

PERF 3: How do you select your storage solution?

The more efficient storage solution for a system varies based on the kind of access operation (block, file, or object), patterns of access (random or sequential), required throughput, frequency of access (online, offline, archival), frequency of update (WORM, dynamic), and availability and durability constraints. Well-architected systems use multiple storage solutions and turn on different features to improve performance and use resources efficiently.

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

Database

The cloud offers purpose-built database services that address different problems presented by your workload. You can choose from many purpose-built database engines including relational, key-value, document, in-memory, graph, time series, and ledger databases. By selecting the most effective database to solve a specific problem (or a group of problems), you can break away from restrictive one-size-fits-all monolithic databases and focus on building applications to meet the performance needs of your customers.

In AWS you can choose from multiple purpose-built database engines including relational, key-value, document, in-memory, graph, time series, and ledger databases. With AWS databases, you don't need to worry about database management tasks such as server provisioning, patching, setup, configuration, backups, or recovery. AWS continuously monitors your clusters to keep your workloads up and running with self-healing storage and automated scaling, so that you can focus on higher value application development.

The following question focuses on these considerations for performance efficiency.

PERF 4: How do you select your database solution?

The most effective database solution for a system varies based on requirements for availabil ity, consistency, partition tolerance, latency, durability, scalability, and query capability. Many systems use different database solutions for various subsystems and turn on different features to improve performance. Selecting the wrong database solution and features for a system can lead to lower performance efficiency.

Your workload's database approach has a significant impact on performance efficiency. It's 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 using graph, time series, or in-memory storage database).

Network

Since the network is between all workload components, it can have great impacts, both positive and negative, on workload performance and behavior. There are also workloads that are heavily dependent on network performance such as high performance computing (HPC) where deep network understanding is important to increase cluster performance. Determine the workload requirements for bandwidth, latency, jitter, and throughput.

On AWS, networking is virtualized and is available in a number of different types and configurations. This makes it efficient to match your networking operations with your needs. AWS offers product features (for example, Enhanced Networking, Amazon EBS-optimized instances, Amazon S3 transfer acceleration, and dynamic Amazon CloudFront) to optimize network traffic. AWS also offers networking features (for example, Amazon Route 53 latency routing, Amazon VPC endpoints, AWS Direct Connect, and AWS Global Accelerator) to reduce network distance or jitter.

The following question focuses on these considerations for performance efficiency.

PERF 5: How do you configure your networking solution?

The most efficient network solution for a workload varies based on latency, throughput requirements, jitter, and bandwidth. Physical constraints, such as user or on-premises resources , determine location options. These constraints can be offset with edge locations or resource placement.

You must consider location when deploying your network. You can choose to place resources close to where they will be used to reduce distance. Use networking metrics to make changes to networking configuration as the workload evolves. By taking advantage of Regions, placement groups, and edge services, you can significantly improve performance. Cloud based networks can be quickly re-built or modified, so evolving your network architecture over time is necessary to maintain performance efficiency.

Review

Cloud technologies are rapidly evolving and you must verify that workload components are using the latest technologies and approaches to continually improve performance. You must continually evaluate and consider changes to your workload components to verify you are meeting its performance and cost objectives. New technologies, such as machine learning and artificial intelligence (AI), can permit you to reimagine customer experiences and innovate across all of your business workloads.

Take advantage of the continual innovation at AWS driven by customer need. We release new Regions, edge locations, services, and features regularly. Any of these releases could positively improve the performance efficiency of your architecture.

The following question focuses on these considerations for performance efficiency.

PERF 6: How do you evolve your workload to take advantage of new releases?

When architecting workloads, there are finite options that you can choose from. However, over time, new technologies and approaches become available that could improve the performance of your workload.

Architectures performing poorly are usually the result of a non-existent or broken performance review process. If your architecture is performing poorly, implementing a performance review process will permit you to apply Deming's plan-do-check-act (PDCA) cycle to drive iterative improvement.

Monitoring

After you implement your workload, you must monitor its performance so that you can remediate any issues before they impact your customers. Monitoring metrics should be used to raise alarms when thresholds are breached.

Amazon CloudWatch is a monitoring and observability service that provides you with data and actionable insights to monitor your workload, respond to system-wide performance changes, optimize resource utilization, and get a unified view of operational health. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events from workloads that run on AWS and on-premises servers. AWS X-Ray helps developers analyze and debug production, distributed applications. With AWS X-Ray, you can glean insights into how your application is

performing and discover root causes and identify performance bottlenecks. You can use these insights to react quickly and keep your workload running smoothly.

The following question focuses on these considerations for performance efficiency.

PERF 7: How do you monitor your resources to verify they are performing?

System performance can degrade over time. Monitor system performance to identify degradation and remediate internal or external factors, such as the operating system or application load.

Validating that you do not see false positives is key to an effective monitoring solution. Automated initiation functions avoid human error and can reduce the time it takes to fix problems. Plan for game days, where simulations are conducted in the production environment, to test your alarm solution and verify that it correctly recognizes issues.

Tradeoffs

When you architect solutions, think about tradeoffs to validate a more efficient approach. Depending on your situation, you could trade consistency, durability, and space for 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.

The following question focuses on these considerations for performance efficiency.

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

When architecting solutions, determining tradeoffs permits you to select an more efficient approach. Often you can improve performance by trading consistency, durability, and space for time and latency.

As you make changes to the workload, collect and evaluate metrics to determine the impact of those changes. Measure the impacts to the system and to the end user to understand how your trade-offs impact your workload. Use a systematic approach, such as load testing, to explore whether the tradeoff improves performance.

Resources

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

Documentation

- Amazon S3 Performance Optimization
- Amazon EBS Volume Performance

Whitepaper

- Performance Efficiency Pillar

Video

- AWS re:Invent 2019: Amazon EC2 foundations (CMP211-R2)
- AWS re:Invent 2019: Leadership session: Storage state of the union (STG201-L)
- AWS re:Invent 2019: Leadership session: AWS purpose-built databases (DAT209-L)
- AWS re:Invent 2019: Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- AWS re:Invent 2019: Powering next-gen Amazon EC2: Deep dive into the Nitro system (CMP303-R2)
- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R)

Cost optimization

The Cost Optimization pillar includes the ability to run systems to deliver business value at the lowest price point.

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.

Topics

- Design principles
- Definition
- Best practices

- Resources

Design principles

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

- **Implement Cloud Financial Management**: To achieve financial success and accelerate business value realization in the cloud, invest in Cloud Financial Management and Cost Optimization. Your organization should dedicate time and resources to build capability in this new domain of technology and usage management. Similar to your Security or Operational Excellence capability, you need to build capability through knowledge building, programs, resources, and processes to become a cost-efficient organization.
- **Adopt a consumption model**: Pay only for the computing resources that you require 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 workload and the costs associated with delivering it. Use this measure to know the gains you make from increasing output and reducing costs.
- **Stop spending money on undifferentiated heavy lifting**: AWS does the heavy lifting of data center operations like racking, stacking, and powering servers. It also removes the operational burden of managing operating systems and applications with managed services. This permits you to focus on your customers and business projects rather than on IT infrastructure.
- **Analyze and attribute expenditure**: The cloud makes it simple to accurately identify the usage and cost of systems, which then permits transparent attribution of IT costs to individual workload owners. This helps measure return on investment (ROI) and gives workload owners an opportunity to optimize their resources and reduce costs.

Definition

There are five best practice areas for cost optimization in the cloud:

- **Practice Cloud Financial Management**
- **Expenditure and usage awareness**
- **Cost-effective resources**

- **Manage demand and supply resources**
- **Optimize over time**

As with the other pillars within the Well-Architected Framework, there are tradeoffs to consider, for example, whether to optimize for speed-to-market or for cost. In some cases, it's more efficient to optimize for speed, going to market quickly, shipping new features, or meeting a deadline, rather than investing in upfront cost optimization. Design decisions are sometimes directed by haste rather than data, and the temptation always exists to overcompensate "just in case" rather than spend time benchmarking for the most cost-optimal deployment. This might lead to over-provisioned and under-optimized deployments. However, this is a reasonable choice when you must "lift and shift" resources from your on-premises environment to the cloud and then optimize afterwards. Investing the right amount of effort in a cost optimization strategy up front permits you to realize the economic benefits of the cloud more readily by achieving a consistent adherence to best practices and avoiding unnecessary over provisioning. The following sections provide techniques and best practices for both the initial and ongoing implementation of Cloud Financial Management and cost optimization of your workloads.

Best practices

Topics

- Practice Cloud Financial Management
- Expenditure and usage awareness
- Cost-effective resources
- Manage demand and supply resources
- Optimize over time

Practice Cloud Financial Management

With the adoption of cloud, technology teams innovate faster due to shortened approval, procurement, and infrastructure deployment cycles. A new approach to financial management in the cloud is required to realize business value and financial success. This approach is Cloud Financial Management, and builds capability across your organization by implementing organizational wide knowledge building, programs, resources, and processes.

Many organizations are composed of many different units with different priorities. The ability to align your organization to an agreed set of financial objectives, and provide your organization the

mechanisms to meet them, will create a more efficient organization. A capable organization will innovate and build faster, be more agile and adjust to any internal or external factors.

In AWS you can use Cost Explorer, and optionally Amazon Athena and Amazon QuickSight with the Cost and Usage Report (CUR), to provide cost and usage awareness throughout your organization. AWS Budgets provides proactive notifications for cost and usage. The AWS blogs provide information on new services and features to verify you keep up to date with new service releases.

The following question focuses on these considerations for cost optimization. (For a list of cost optimization questions and best practices, see the Appendix.).

COST 1: How do you implement cloud financial management?

Implementing Cloud Financial Management helps organizations realize business value and financial success as they optimize their cost and usage and scale on AWS.

When building a cost optimization function, use members and supplement the team with experts in CFM and cost optimization. Existing team members will understand how the organization currently functions and how to rapidly implement improvements. Also consider including people with supplementary or specialist skill sets, such as analytics and project management.

When implementing cost awareness in your organization, improve or build on existing programs and processes. It is much faster to add to what exists than to build new processes and programs. This will result in achieving outcomes much faster.

Expenditure and usage awareness

The increased flexibility and agility that the cloud provides encourages innovation and fast-paced development and deployment. It decreases 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 requires 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 organization or product owners drives efficient usage behavior and helps reduce waste. Accurate cost attribution permits you to know which products are truly profitable, and permits you to make more informed decisions about where to allocate budget.

In AWS, you create an account structure with AWS Organizations or AWS Control Tower, which provides separation and assists in allocation of your costs and usage. You can also use resource tagging to apply business and organization information to your usage and cost. Use AWS Cost Explorer for visibility into your cost and usage, or create customized dashboards and analytics with Amazon Athena and Amazon QuickSight. Controlling your cost and usage is done by notifications through AWS Budgets, and controls using AWS Identity and Access Management (IAM), and Service Quotas.

The following questions focus on these considerations for cost optimization.

COST 2: How do you govern usage?

Establish policies and mechanisms to validate that appropriate costs are incurred while objective s are achieved. By employing a checks-and-balances approach, you can innovate without overspending.

COST 3: How do you monitor usage and cost?

Establish policies and procedures to monitor and appropriately allocate your costs. This permits you to measure and improve the cost efficiency of this workload.

COST 4: How do you decommission resources?

Implement change control and resource management from project inception to end-of-life. This facilitates shutting down unused resources to reduce waste.

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

Verify that you use the right level of detail and granularity in cost and usage reporting and monitoring. For high level insights and trends, use daily granularity with AWS Cost Explorer. For deeper analysis and inspection use hourly granularity in AWS Cost Explorer, or Amazon Athena and Amazon Quick with the Cost and Usage Report (CUR) at an hourly granularity.

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 organization and should be decommissioned. You can set up billing alerts to notify you of predicted overspending.

Cost-effective resources

Using the appropriate instances and resources for your workload 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 server incurs more cost over time.

A well-architected workload uses 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 instances from Amazon EC2 and other services in a way that more effectively fits your needs. *On-Demand Instances* permit you to pay for compute capacity by the hour, with no minimum commitments required. *Savings Plans and Reserved Instances* offer savings of up to 75% off On-Demand pricing. With Spot Instances, you can leverage unused Amazon EC2 capacity and offer savings of up to 90% off On-Demand pricing. *Spot Instances* are appropriate where the system can tolerate using a fleet of servers where individual servers can come and go dynamically, such as stateless web servers, batch processing, or when using HPC and big data.

Appropriate service selection can also reduce usage and costs; such as CloudFront to minimize data transfer, or decrease costs, such as utilizing Amazon Aurora on Amazon RDS to remove expensive database licensing costs.

The following questions focus on these considerations for cost optimization.

COST 5: How do you evaluate cost when you select services?

Amazon EC2, Amazon EBS, and Amazon S3 are building-block AWS services. Managed services, such as Amazon RDS and Amazon DynamoDB, are higher level, or application level, AWS services. By selecting the appropriate building blocks and managed services, you can optimize this workload for cost. For example, using managed services, you can reduce or remove much of

COST 5: How do you evaluate cost when you select services?

your administrative and operational overhead, freeing you to work on applications and business-related activities.

COST 6: How do you meet cost targets when you select resource type, size and number?

Verify that you choose the appropriate resource size and number of resources for the task at hand. You minimize waste by selecting the most cost effective type, size, and number.

COST 7: How do you use pricing models to reduce cost?

Use the pricing model that is most appropriate for your resources to minimize expense.

COST 8: How do you plan for data transfer charges?

Verify that you plan and monitor data transfer charges so that you can make architectural decisions to minimize costs. A small yet effective architectural change can drastically reduce your operational costs over time.

By factoring in cost during service selection, and using tools such as Cost Explorer and AWS Trusted Advisor to regularly review your AWS usage, you can actively monitor your utilization and adjust your deployments accordingly.

Manage demand and supply resources

When you move to the cloud, you pay only for what you need. You can supply resources to match the workload demand at the time they're needed, this decreases the need for costly and wasteful over provisioning. You can also modify the demand, using a throttle, buffer, or queue to smooth the demand and serve it with less resources resulting in a lower cost, or process it at a later time with a batch service.

In AWS, you can automatically provision resources to match the workload demand. Auto Scaling using demand or time-based approaches permit you to add and remove resources as needed. If

you can anticipate changes in demand, you can save more money and validate that your resources match your workload needs. You can use Amazon API Gateway to implement throttling, or Amazon SQS to implementing a queue in your workload. These will both permit you to modify the demand on your workload components.

The following question focuses on these considerations for cost optimization.

COST 9: How do you manage demand, and supply resources?

For a workload that has balanced spend and performance, verify that everything you pay for is used and avoid significantly underutilizing instances. A skewed utilization metric in either direction has an adverse impact on your organization, in either operational costs (degraded performance due to over-utilization), or wasted AWS expenditures (due to over-provisioning).

When designing to modify demand and supply resources, actively think about the patterns of usage, the time it takes to provision new resources, and the predictability of the demand pattern. When managing demand, verify you have a correctly sized queue or buffer, and that you are responding to workload demand in the required amount of time.

Optimize over time

As AWS releases new services and features, it's a best practice to review your existing architectural decisions to verify 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.

Implementing new features or resource types can optimize your workload incrementally, while minimizing the effort required to implement the change. This provides continual improvements in efficiency over time and provides you remain on the most updated technology to reduce operating costs. You can also replace or add new components to the workload with new services. This can provide significant increases in efficiency, so it's essential to regularly review your workload, and implement new services and features.

The following questions focus on these considerations for cost optimization.

COST 10: How do you evaluate new services?

As AWS releases new services and features, it's a best practice to review your existing architect ural decisions to verify they continue to be the most cost effective.

When regularly reviewing your deployments, assess how newer services can help save you money. For example, Amazon Aurora on Amazon RDS can reduce costs for relational databases. Using serverless such as Lambda can remove the need to operate and manage instances to run code.

COST 11: How do you evaluate the cost of effort?

Evaluate the cost of effort for operations in the cloud, review your time-consuming cloud operations, and automate them to reduce human efforts and cost by adopting related AWS services, third-party products, or custom tools.

Resources

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

Documentation

- AWS Documentation

Whitepaper

- Cost Optimization Pillar

Sustainability

The Sustainability pillar focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct action to reduce resource usage. You can find prescriptive guidance on implementation in the Sustainability Pillar whitepaper.

Topics

- Design principles
- Definition
- Best practices
- Resources

Design principles

There are six design principles for sustainability in the cloud:

- **Understand your impact:** Measure the impact of your cloud workload and model the future impact of your workload. Include all sources of impact, including impacts resulting from customer use of your products, and impacts resulting from their eventual decommissioning and retirement. Compare the productive output with the total impact of your cloud workloads by reviewing the resources and emissions required per unit of work. Use this data to establish key performance indicators (KPIs), evaluate ways to improve productivity while reducing impact, and estimate the impact of proposed changes over time.
- **Establish sustainability goals:** For each cloud workload, establish long-term sustainability goals such as reducing the compute and storage resources required per transaction. Model the return on investment of sustainability improvements for existing workloads, and give owners the resources they must invest in sustainability goals. Plan for growth, and architect your workloads so that growth results in reduced impact intensity measured against an appropriate unit, such as per user or per transaction. Goals help you support the wider sustainability goals of your business or organization, identify regressions, and prioritize areas of potential improvement.
- **Maximize utilization:** Right-size workloads and implement efficient design to verify high utilization and maximize the energy efficiency of the underlying hardware. Two hosts running at 30% utilization are less efficient than one host running at 60% due to baseline power consumption per host. At the same time, reduce or minimize idle resources, processing, and storage to reduce the total energy required to power your workload.
- **Anticipate and adopt new, more efficient hardware and software offerings:** Support the upstream improvements your partners and suppliers make to help you reduce the impact of your cloud workloads. Continually monitor and evaluate new, more efficient hardware and software offerings. Design for flexibility to permit the rapid adoption of new efficient technologies.
- **Use managed services:** Sharing services across a broad customer base helps maximize resource utilization, which reduces the amount of infrastructure needed to support cloud workloads. For example, customers can share the impact of common data center components like power and networking by migrating workloads to the AWS Cloud and adopting managed services, such as AWS Fargate for serverless containers, where AWS operates at scale and is responsible for their efficient operation. Use managed services that can help minimize your impact, such as automatically moving infrequently accessed data to cold storage with Amazon S3 Lifecycle configurations or Amazon EC2 Auto Scaling to adjust capacity to meet demand.

- **Reduce the downstream impact of your cloud workloads:** Reduce the amount of energy or resources required to use your services. Reduce the need for customers to upgrade their devices to use your services. Test using device farms to understand expected impact and test with customers to understand the actual impact from using your services.

Definition

There are six best practice areas for sustainability in the cloud:

- Region selection
- Alignment to demand
- Software and architecture
- Data
- Hardware and services
- Process and culture

Sustainability in the cloud is a nearly continuous effort focused primarily on energy reduction and efficiency across all components of a workload by achieving the maximum benefit from the resources provisioned and minimizing the total resources required. This effort can range from the initial selection of an efficient programming language, adoption of modern algorithms, use of efficient data storage techniques, deploying to correctly sized and efficient compute infrastructure, and minimizing requirements for high-powered end user hardware.

Best practices

Topics

- Region selection
- Alignment to demand
- Software and architecture
- Data
- Hardware and services
- Process and culture

Region selection

The choice of Region for your workload significantly affects its KPIs, including performance, cost, and carbon footprint. To improve these KPIs, you should choose Regions for your workloads based on both business requirements and sustainability goals.

The following question focuses on these considerations for sustainability. (For a list of sustainability questions and best practices, see the Appendix.)

SUS 1: How do you select Regions for your workload?

The choice of Region for your workload significantly affects its KPIs, including performan ce, cost, and carbon footprint. To improve these KPIs, you should choose Regions for your workloads based on both business requirements and sustainability goals.

Alignment to demand

The way users and applications consume your workloads and other resources can help you identify improvements to meet sustainability goals. Scale infrastructure to continually match demand and verify that you use only the minimum resources required to support your users. Align service levels to customer needs. Position resources to limit the network required for users and applications to consume them. Remove unused assets. Provide your team members with devices that support their needs and minimize their sustainability impact.

The following question focuses on this consideration for sustainability:

SUS 2: How do you align cloud resources to your demand?

The way users and applications consume your workloads and other resources can help you identify improvements to meet sustainability goals. Scale infrastructure to continually match demand and verify that you use only the minimum resources required to support your users. Align service levels to customer needs. Position resources to limit the network required for users and applications to consume them. Remove unused assets. Provide your team members with devices that support their needs and minimize their sustainability impact.

Scale infrastructure with user load: Identify periods of low or no utilization and scale resources to reduce excess capacity and improve efficiency.

Align SLAs with sustainability goals: Define and update service level agreements (SLAs) such as availability or data retention periods to minimize the number of resources required to support your workload while continuing to meet business requirements.

Decrease creation and maintenance of unused assets: Analyze application assets (such as pre-compiled reports, datasets, and static images) and asset access patterns to identify redundancy, underutilization, and potential decommission targets. Consolidate generated assets with redundant content (for example, monthly reports with overlapping or common datasets and outputs) to reduce the resources consumed when duplicating outputs. Decommission unused assets (for example, images of products that are no longer sold) to release consumed resources and reduce the number of resources used to support the workload.

Optimize geographic placement of workloads for user locations: Analyze network access patterns to identify where your customers are connecting from geographically. Select Regions and services that reduce the distance that network traffic must travel to decrease the total network resources required to support your workload.

Optimize team member resources for activities performed: Optimize resources provided to team members to minimize the sustainability impact while supporting their needs. For example, perform complex operations, such as rendering and compilation, on highly used shared cloud desktops instead of on under-utilized high-powered single user systems.

Software and architecture

Implement patterns for performing load smoothing and maintaining consistent high utilization of deployed resources to minimize the resources consumed. Components might become idle from lack of use because of changes in user behavior over time. Revise patterns and architecture to consolidate under-utilized components to increase overall utilization. Retire components that are no longer required. Understand the performance of your workload components, and optimize the components that consume the most resources. Be aware of the devices that your customers use to access your services, and implement patterns to minimize the need for device upgrades.

The following questions focus on these considerations for sustainability:

SUS 3: How do you take advantage of software and architecture patterns to support your sustainability goals?

Implement patterns for performing load smoothing and maintaining consistent high utilizati on of deployed resources to minimize the resources consumed. Components might become idle

SUS 3: How do you take advantage of software and architecture patterns to support your sustainability goals?

from lack of use because of changes in user behavior over time. Revise patterns and architect ure to consolidate under-utilized components to increase overall utilization. Retire component s that are no longer required. Understand the performance of your workload components, and optimize the components that consume the most resources. Be aware of the devices that your customers use to access your services, and implement patterns to minimize the need for device upgrades.

Optimize software and architecture for asynchronous and scheduled jobs: Use efficient software designs and architectures to minimize the average resources required per unit of work. Implement mechanisms that result in even utilization of components to reduce resources that are idle between tasks and minimize the impact of load spikes.

Remove or refactor workload components with low or no use: Monitor workload activity to identify changes in utilization of individual components over time. Remove components that are unused and no longer required, and refactor components with little utilization, to limit wasted resources.

Optimize areas of code that consume the most time or resources: Monitor workload activity to identify application components that consume the most resources. Optimize the code that runs within these components to minimize resource usage while maximizing performance.

Optimize impact on customer devices and equipment: Understand the devices and equipment that your customers use to consume your services, their expected lifecycle, and the financial and sustainability impact of replacing those components. Implement software patterns and architectures to minimize the need for customers to replace devices and upgrade equipment. For example, implement new features using code that is backward compatible with earlier hardware and operating system versions, or manage the size of payloads so they don't exceed the storage capacity of the target device.

Use software patterns and architectures that most effectively supports data access and storage patterns: Understand how data is used within your workload, consumed by your users, transferred, and stored. Select technologies to minimize data processing and storage requirements.

Data

The following question focuses on these considerations for sustainability:

SUS 4: How do you take advantage of data management policies and patterns to support your sustainability goals?

Implement data management practices to reduce the provisioned storage required to support your workload, and the resources required to use it. Understand your data, and use storage technologies and configurations that most effectively supports the business value of the data and how it's used. Lifecycle data to more efficient, less performant storage when requirements decrease, and delete data that's no longer required.

Implement a data classification policy: Classify data to understand its significance to business outcomes. Use this information to determine when you can move data to more energy-efficient storage or safely delete it.

Use technologies that support data access and storage patterns: Use storage that most effectively supports how your data is accessed and stored to minimize the resources provisioned while supporting your workload. For example, solid state devices (SSDs) are more energy intensive than magnetic drives and should be used only for active data use cases. Use energy-efficient, archival-class storage for infrequently accessed data.

Use lifecycle policies to delete unnecessary data: Manage the lifecycle of all your data and automatically enforce deletion timelines to minimize the total storage requirements of your workload.

Minimize over-provisioning in block storage: To minimize total provisioned storage, create block storage with size allocations that are appropriate for the workload. Use elastic volumes to expand storage as data grows without having to resize storage attached to compute resources. Regularly review elastic volumes and shrink over-provisioned volumes to fit the current data size.

Remove unneeded or redundant data: Duplicate data only when necessary to minimize total storage consumed. Use backup technologies that deduplicate data at the file and block level. Limit the use of Redundant Array of Independent Drives (RAID) configurations except where required to meet SLAs.

Use shared file systems or object storage to access common data: Adopt shared storage and single sources of truth to avoid data duplication and reduce the total storage requirements of your workload. Fetch data from shared storage only as needed. Detach unused volumes to release resources. Minimize data movement across networks: Use shared storage and access data

from Regional data stores to minimize the total networking resources required to support data movement for your workload.

Back up data only when difficult to recreate: To minimize storage consumption, only back up data that has business value or is required to satisfy compliance requirements. Examine backup policies and exclude ephemeral storage that doesn't provide value in a recovery scenario.

Hardware and services

Look for opportunities to reduce workload sustainability impacts by making changes to your hardware management practices. Minimize the amount of hardware needed to provision and deploy, and select the most efficient hardware and services for your individual workload.

The following question focuses on these considerations for sustainability:

SUS 5: How do you select and use cloud hardware and services in your architecture to support your sustainability goals?

Look for opportunities to reduce workload sustainability impacts by making changes to your hardware management practices. Minimize the amount of hardware needed to provision and deploy, and select the most efficient hardware and services for your individual workload.

Use the minimum amount of hardware to meet your needs: Using the capabilities of the cloud, you can make frequent changes to your workload implementations. Update deployed components as your needs change.

Use instance types with the least impact: Continually monitor the release of new instance types and take advantage of energy efficiency improvements, including those instance types designed to support specific workloads such as machine learning training and inference, and video transcoding.

Use managed services: Managed services shift responsibility for maintaining high average utilization, and sustainability optimization of the deployed hardware, to AWS. Use managed services to distribute the sustainability impact of the service across all tenants of the service, reducing your individual contribution.

Optimize your use of GPUs: Graphics processing units (GPUs) can be a source of high-power consumption, and many GPU workloads are highly variable, such as rendering, transcoding, and machine learning training and modeling. Only run GPUs instances for the time needed, and decommission them with automation when not required to minimize resources consumed.

Process and culture

Look for opportunities to reduce your sustainability impact by making changes to your development, test, and deployment practices.

The following question focuses on these considerations for sustainability:

SUS 6: How do your organizational processes support your sustainability goals?

Look for opportunities to reduce your sustainability impact by making changes to your development, test, and deployment practices.

Adopt operations that can rapidly introduce sustainability improvements: Test and validate potential improvements before deploying them to production. Account for the cost of testing when calculating potential future benefit of an improvement. Develop low-cost testing operations to drive delivery of small improvements.

Keep your workload up to date: Up-to-date operating systems, libraries, and applications can improve workload efficiency and create adoption of more efficient technologies. Up-to-date software might also include features to measure the sustainability impact of your workload more accurately, as vendors deliver features to meet their own sustainability goals.

Increase utilization of build environments: Use automation and infrastructure as code to bring up pre-production environments when needed and take them down when not used. A common pattern is to schedule periods of availability that coincide with the working hours of your development team members. Hibernation is a useful tool to preserve state and rapidly bring instances online only when needed. Use instance types with burst capacity, Spot Instances, elastic database services, containers, and other technologies to align development and test capacity with use.

Use managed device farms for testing: Managed device farms spread the sustainability impact of hardware manufacturing and resource usage across multiple tenants. Managed device farms offer diverse device types so you can support earlier, less popular hardware, and avoid customer sustainability impact from unnecessary device upgrades.

Resources

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

Whitepaper

- Sustainability Pillar

Video

- The Climate Pledge

The review process

The review of architectures must be done in a consistent manner, with a blame-free approach that encourages diving deep. It should be a lightweight 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 the team members who build an architecture use the Well-Architected Framework to continually review their architecture, rather than holding a formal review meeting. A nearly continuous approach permits your team members to update answers as the architecture evolves, and improve the architecture as you deliver features.

The AWS Well-Architected Framework 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 verify that people don’t neglect areas that often featured in 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 key milestones in the product lifecycle, early on in the design phase to avoid *one-way doors* that are difficult to change, and then before the go-live date. (Many decisions are reversible, two-way doors. Those decisions can use a lightweight process. One-way doors are hard or impossible to reverse and require more inspection before making them.) After you go into production, your workload will continue to evolve as you add new features and change technology implementations. The architecture of a workload changes over time. You must 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.

If you want to use the review as a one-time snapshot or independent measurement, you will want to verify that 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 activate 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 significant launch.)
  - If you are getting ready for a big launch, you will want it to go smoothly. The review will permit you to understand any problems you might have missed.
  - We recommend that you carry out reviews early in the product 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 in the Well-Architected Framework, they will see that none of the questions reveal any commercial or technical proprietary 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 six pillars for designing and operating reliable, secure, efficient, cost-effective, and sustainable 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:

- Brian Carlson, Operations Lead Well-Architected, Amazon Web Services
- Ben Potter, Security Lead Well-Architected, Amazon Web Services
- Seth Eliot, Reliability Lead Well-Architected, Amazon Web Services
- Eric Pullen, Sr. Solutions Architect, Amazon Web Services
- Rodney Lester, Principal Solutions Architect, Amazon Web Services
- Jon Steele, Sr. Technical Account Manager, Amazon Web Services
- Max Ramsay, Principal Security Solutions Architect, Amazon Web Services
- Callum Hughes, Solutions Architect, Amazon Web Services
- Aden Leirer, Content Program Manager Well-Architected, Amazon Web Services

Further reading

AWS Architecture Center

AWS Cloud Compliance

AWS Well-Architected Partner program

AWS Well-Architected Tool

AWS Well-Architected homepage

Operational Excellence Pillar whitepaper

Security Pillar whitepaper

Reliability Pillar whitepaper

Performance Efficiency Pillar whitepaper

Cost Optimization Pillar whitepaper

Sustainability Pillar whitepaper

The Amazon Builders' Library

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Updates for new Framework	Best practices updated with prescriptive guidance and new best practices added. New questions added to the Security and Cost Optimizat ion pillars.	April 10, 2023
Minor update	Added definition for level of effort and updated best practices in the appendix.	October 20, 2022
Whitepaper updated	Added Sustainability Pillar and updated links.	December 2, 2021
Major update	Sustainability Pillar added to the framework.	November 20, 2021
Minor update	Removed non-inclusive language.	April 22, 2021
Minor update	Fixed numerous links.	March 10, 2021
Minor update	Minor editorial changes throughout.	July 15, 2020
Updates for new Framework	Review and rewrite of most questions and answers.	July 8, 2020
Whitepaper updated	Addition of AWS Well-Arch itected Tool, links to AWS Well-Architected Labs, and AWS Well-Architected Partners, minor fixes to	July 1, 2019

	enable multiple language version of framework.	
Whitepaper updated	Review and rewrite of most questions and answers, to ensure questions focus on one topic at a time. This caused some previous questions to be split into multiple questions. Added common terms to definitions (workload , component etc). Changed presentation of question in main body to include descripti ve text.	November 1, 2018
Whitepaper updated	Updates to simplify question text, standardize answers, and improve readability.	June 1, 2018
Whitepaper updated	Operational Excellence moved to front of pillars and rewritten so it frames other pillars. Refreshed other pillars to reflect evolution of AWS.	November 1, 2017
Whitepaper updated	Updated the Framework to include operational excellenc e pillar, and revised and updated the other pillars to reduce duplication and incorporate learnings from carrying out reviews with thousands of customers.	November 1, 2016
Minor updates	Updated the Appendix with current Amazon CloudWatch Logs information.	November 1, 2015

Initial publication	AWS Well-Architected Framework published.	October 1, 2015

Appendix: Questions and best practices

This appendix summarizes all the questions and best practices in the AWS Well-Architected Framework.

Pillars

- Operational excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization
- Sustainability

Operational excellence

The Operational Excellence pillar includes the ability to support development and run workloads effectively, gain insight into your operations, and to continuously improve supporting processes and procedures to deliver business value. You can find prescriptive guidance on implementation in the Operational Excellence Pillar whitepaper.

Best practice areas

- Organization
- Prepare
- Operate
- Evolve

Organization

Questions

- OPS 1. How do you determine what your priorities are?
- OPS 2. How do you structure your organization to support your business outcomes?
- OPS 3. How does your organizational culture support your business outcomes?

OPS 1. How do you determine what your priorities are?

Everyone should understand their part in enabling business success. Have shared goals in order to set priorities for resources. This will maximize the benefits of your efforts.

Best practices

- OPS01-BP01 Evaluate external customer needs
- OPS01-BP02 Evaluate internal customer needs
- OPS01-BP03 Evaluate governance requirements
- OPS01-BP04 Evaluate compliance requirements
- OPS01-BP05 Evaluate threat landscape
- OPS01-BP06 Evaluate tradeoffs
- OPS01-BP07 Manage benefits and risks

OPS01-BP01 Evaluate external customer needs

Involve key stakeholders, including business, development, and operations teams, to determine where to focus efforts on external customer needs. This will ensure that you have a thorough understanding of the operations support that is required to achieve your desired business outcomes.

Common anti-patterns:

- You have decided not to have customer support outside of core business hours, but you haven't reviewed historical support request data. You do not know whether this will have an impact on your customers.
- You are developing a new feature but have not engaged your customers to find out if it is desired, if desired in what form, and without experimentation to validate the need and method of delivery.

Benefits of establishing this best practice: Customers whose needs are satisfied are much more likely to remain customers. Evaluating and understanding external customer needs will inform how you prioritize your efforts to deliver business value.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Understand business needs: Business success is created by shared goals and understanding across stakeholders, including business, development, and operations teams.
  - Review business goals, needs, and priorities of external customers: Engage key stakeholders, including business, development, and operations teams, to discuss goals, needs, and priorities of external customers. This ensures that you have a thorough understanding of the operational support that is required to achieve business and customer outcomes.
  - Establish shared understanding: Establish shared understanding of the business functions of the workload, the roles of each of the teams in operating the workload, and how these factors support your shared business goals across internal and external customers.

Resources

Related documents:

- AWS Well-Architected Framework Concepts – Feedback loop

OPS01-BP02 Evaluate internal customer needs

Involve key stakeholders, including business, development, and operations teams, when determining where to focus efforts on internal customer needs. This will ensure that you have a thorough understanding of the operations support that is required to achieve business outcomes.

Use your established priorities to focus your improvement efforts where they will have the greatest impact (for example, developing team skills, improving workload performance, reducing costs, automating runbooks, or enhancing monitoring). Update your priorities as needs change.

Common anti-patterns:

- You have decided to change IP address allocations for your product teams, without consulting them, to make managing your network easier. You do not know the impact this will have on your product teams.
- You are implementing a new development tool but have not engaged your internal customers to find out if it is needed or if it is compatible with their existing practices.
- You are implementing a new monitoring system but have not contacted your internal customers to find out if they have monitoring or reporting needs that should be considered.

Benefits of establishing this best practice: Evaluating and understanding internal customer needs will inform how you prioritize your efforts to deliver business value.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Understand business needs: Business success is created by shared goals and understanding across stakeholders including business, development, and operations teams.
  - Review business goals, needs, and priorities of internal customers: Engage key stakeholders, including business, development, and operations teams, to discuss goals, needs, and priorities of internal customers. This ensures that you have a thorough understanding of the operational support that is required to achieve business and customer outcomes.
  - Establish shared understanding: Establish shared understanding of the business functions of the workload, the roles of each of the teams in operating the workload, and how these factors support shared business goals across internal and external customers.

Resources

Related documents:

- AWS Well-Architected Framework Concepts – Feedback loop

OPS01-BP03 Evaluate governance requirements

Governance is the set of policies, rules, or frameworks that a company uses to achieve its business goals. Governance requirements are generated from within your organization. They can affect the types of technologies you choose or influence the way you operate your workload. Incorporate organizational governance requirements into your workload. Conformance is the ability to demonstrate that you have implemented governance requirements.

Desired outcome:

- Governance requirements are incorporated into the architectural design and operation of your workload.
- You can provide proof that you have followed governance requirements.
- Governance requirements are regularly reviewed and updated.

Common anti-patterns:

- Your organization mandates that the root account has multi-factor authentication. You failed to implement this requirement and the root account is compromised.
- During the design of your workload, you choose an instance type that is not approved by the IT department. You are unable to launch your workload and must conduct a redesign.
- You are required to have a disaster recovery plan. You did not create one and your workload suffers an extended outage.
- Your team wants to use new instances but your governance requirements have not been updated to allow them.

Benefits of establishing this best practice:

- Following governance requirements aligns your workload with larger organization policies.
- Governance requirements reflect industry standards and best practices for your organization.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify governance requirement by working with stakeholders and governance organizations. Include governance requirements into your workload. Be able to demonstrate proof that you've followed governance requirements.

Customer example

At AnyCompany Retail, the cloud operations team works with stakeholders across the organization to develop governance requirements. For example, they prohibit SSH access into Amazon EC2 instances. If teams need system access, they are required to use AWS Systems Manager Session Manager. The cloud operations team regularly updates governance requirements as new services become available.

Implementation steps

1. Identify the stakeholders for your workload, including any centralized teams.
2. Work with stakeholders to identify governance requirements.
3. Once you've generated a list, prioritize the improvement items, and begin implementing them into your workload.

   a. Use services like AWS Config to create governance-as-code and validate that governance requirements are followed.
   b. If you use AWS Organizations, you can leverage Service Control Policies to implement governance requirements.
4. Provide documentation that validates the implementation.

Level of effort for the implementation plan: Medium. Implementing missing governance requirements may result in rework of your workload.

Resources

Related best practices:

- OPS01-BP04 Evaluate compliance requirements - Compliance is like governance but comes from outside an organization.

Related documents:

- AWS Management and Governance Cloud Environment Guide
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment
- Governance in the AWS Cloud: The Right Balance Between Agility and Safety
- What is Governance, Risk, And Compliance (GRC)?

Related videos:

- AWS Management and Governance: Configuration, Compliance, and Audit - AWS Online Tech Talks
- AWS re:Inforce 2019: Governance for the Cloud Age (DEM12-R1)
- AWS re:Invent 2020: Achieve compliance as code using AWS Config
- AWS re:Invent 2020: Agile governance on AWS GovCloud (US)

Related examples:

- AWS Config Conformance Pack Samples

Related services:

- AWS Config
- AWS Organizations - Service Control Policies

OPS01-BP04 Evaluate compliance requirements

Regulatory, industry, and internal compliance requirements are an important driver for defining your organization's priorities. Your compliance framework may preclude you from using specific technologies or geographic locations. Apply due diligence if no external compliance frameworks are identified. Generate audits or reports that validate compliance.

If you advertise that your product meets specific compliance standards, you must have an internal process for ensuring continuous compliance. Examples of compliance standards include PCI DSS, FedRAMP, and HIPAA. Applicable compliance standards are determined by various factors, such as what types of data the solution stores or transmits and which geographic regions the solution supports.

Desired outcome:

- Regulatory, industry, and internal compliance requirements are incorporated into architectural selection.
- You can validate compliance and generate audit reports.

Common anti-patterns:

- Parts of your workload fall under the Payment Card Industry Data Security Standard (PCI-DSS) framework but your workload stores credit cards data unencrypted.
- Your software developers and architects are unaware of the compliance framework that your organization must adhere to.
- The yearly Systems and Organizations Control (SOC2) Type II audit is happening soon and you are unable to verify that controls are in place.

Benefits of establishing this best practice:

- Evaluating and understanding the compliance requirements that apply to your workload will inform how you prioritize your efforts to deliver business value.

- You choose the right locations and technologies that are congruent with your compliance framework.
- Designing your workload for auditability helps you to prove you are adhering to your compliance framework.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implementing this best practice means that you incorporate compliance requirements into your architecture design process. Your team members are aware of the required compliance framework. You validate compliance in line with the framework.

Customer example

AnyCompany Retail stores credit card information for customers. Developers on the card storage team understand that they need to comply with the PCI-DSS framework. They've taken steps to verify that credit card information is stored and accessed securely in line with the PCI-DSS framework. Every year they work with their security team to validate compliance.

Implementation steps

1. Work with your security and governance teams to determine what industry, regulatory, or internal compliance frameworks that your workload must adhere to. Incorporate the compliance frameworks into your workload.
   a. Validate continual compliance of AWS resources with services like AWS Compute Optimizer and AWS Security Hub CSPM.
2. Educate your team members on the compliance requirements so they can operate and evolve the workload in line with them. Compliance requirements should be included in architectural and technological choices.
3. Depending on the compliance framework, you may be required to generate an audit or compliance report. Work with your organization to automate this process as much as possible.
   a. Use services like AWS Audit Manager to generate validate compliance and generate audit reports.
   b. You can download AWS security and compliance documents with AWS Artifact.

Level of effort for the implementation plan: Medium. Implementing compliance frameworks can be challenging. Generating audit reports or compliance documents adds additional complexity.

Resources

Related best practices:

- SEC01-BP03 Identify and validate control objectives - Security control objectives are an important part of overall compliance.
- SEC01-BP06 Automate testing and validation of security controls in pipelines - As part of your pipelines, validate security controls. You can also generate compliance documentation for new changes.
- SEC07-BP02 Define data protection controls - Many compliance frameworks have data handling and storage policies based.
- SEC10-BP03 Prepare forensic capabilities - Forensic capabilities can sometimes be used in auditing compliance.

Related documents:

- AWS Compliance Center
- AWS Compliance Resources
- AWS Risk and Compliance Whitepaper
- AWS Shared Responsibility Model
- AWS services in scope by compliance programs

Related videos:

- AWS re:Invent 2020: Achieve compliance as code using AWS Compute Optimizer
- AWS re:Invent 2021 - Cloud compliance, assurance, and auditing
- AWS Summit ATL 2022 - Implementing compliance, assurance, and auditing on AWS (COP202)

Related examples:

- PCI DSS and AWS Foundational Security Best Practices on AWS

Related services:

- AWS Artifact
- AWS Audit Manager
- AWS Compute Optimizer
- AWS Security Hub CSPM

OPS01-BP05 Evaluate threat landscape

Evaluate threats to the business (for example, competition, business risk and liabilities, operational risks, and information security threats) and maintain current information in a risk registry. Include the impact of risks when determining where to focus efforts.

The Well-Architected Framework emphasizes learning, measuring, and improving. It provides a consistent approach for you to evaluate architectures, and implement designs that will scale over time. AWS provides the AWS Well-Architected Tool to help you review your approach prior to development, the state of your workloads prior to production, and the state of your workloads in production. You can compare them to the latest AWS architectural best practices, monitor the overall status of your workloads, and gain insight to potential risks.

AWS customers are eligible for a guided Well-Architected Review of their mission-critical workloads to measure their architectures against AWS best practices. Enterprise Support customers are eligible for an Operations Review, designed to help them to identify gaps in their approach to operating in the cloud.

The cross-team engagement of these reviews helps to establish common understanding of your workloads and how team roles contribute to success. The needs identified through the review can help shape your priorities.

AWS Trusted Advisor is a tool that provides access to a core set of checks that recommend optimizations that may help shape your priorities. Business and Enterprise Support customers receive access to additional checks focusing on security, reliability, performance, and cost-optimization that can further help shape their priorities.

Common anti-patterns:

- You are using an old version of a software library in your product. You are unaware of security updates to the library for issues that may have unintended impact on your workload.
- Your competitor just released a version of their product that addresses many of your customers' complaints about your product. You have not prioritized addressing any of these known issues.

- Regulators have been pursuing companies like yours that are not compliant with legal regulatory compliance requirements. You have not prioritized addressing any of your outstanding compliance requirements.

Benefits of establishing this best practice: Identifying and understanding the threats to your organization and workload helps your determination of which threats to address, their priority, and the resources necessary to do so.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Evaluate threat landscape: Evaluate threats to the business (for example, competition, business risk and liabilities, operational risks, and information security threats), so that you can include their impact when determining where to focus efforts.
  - AWS Latest Security Bulletins
  - AWS Trusted Advisor
  - Maintain a threat model: Establish and maintain a threat model identifying potential threats, planned and in place mitigations, and their priority. Review the probability of threats manifesting as incidents, the cost to recover from those incidents and the expected harm caused, and the cost to prevent those incidents. Revise priorities as the contents of the threat model change.

Resources

Related documents:

- AWS Cloud Compliance
- AWS Latest Security Bulletins
- AWS Trusted Advisor

OPS01-BP06 Evaluate tradeoffs

Evaluate the impact of tradeoffs between competing interests or alternative approaches, to help make informed decisions when determining where to focus efforts or choosing a course of action. For example, accelerating speed to market for new features may be emphasized over cost optimization, or you may choose a relational database for non-relational data to simplify the

effort to migrate a system, rather than migrating to a database optimized for your data type and updating your application.

AWS can help you educate your teams about AWS and its services to increase their understanding of how their choices can have an impact on your workload. You should use the resources provided by AWS Support (AWS Knowledge Center, AWS Discussion Forums, and AWS Support Center) and AWS Documentation to educate your teams. Reach out to AWS Support through AWS Support Center for help with your AWS questions.

AWS also shares best practices and patterns that we have learned through the operation of AWS in The Amazon Builders' Library. A wide variety of other useful information is available through the AWS Blog and The Official AWS Podcast.

Common anti-patterns:

- You are using a relational database to manage time series and non-relational data. There are database options that are optimized to support the data types you are using but you are unaware of the benefits because you have not evaluated the tradeoffs between solutions.
- Your investors request that you demonstrate compliance with Payment Card Industry Data Security Standards (PCI DSS). You do not consider the tradeoffs between satisfying their request and continuing with your current development efforts. Instead you proceed with your development efforts without demonstrating compliance. Your investors stop their support of your company over concerns about the security of your platform and their investments.

Benefits of establishing this best practice: Understanding the implications and consequences of your choices helps you to prioritize your options.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Evaluate tradeoffs: Evaluate the impact of tradeoffs between competing interests, to help make informed decisions when determining where to focus efforts. For example, accelerating speed to market for new features might be emphasized over cost optimization.
- AWS can help you educate your teams about AWS and its services to increase their understanding of how their choices can have an impact on your workload. You should use the resources provided by AWS Support (AWS Knowledge Center, AWS Discussion Forums, and AWS Support Center) and AWS Documentation to educate your teams. Reach out to AWS Support through AWS Support Center for help with your AWS questions.

- AWS also shares best practices and patterns that we have learned through the operation of AWS in The Amazon Builders' Library. A wide variety of other useful information is available through the AWS Blog and The Official AWS Podcast.

Resources

Related documents:

- AWS Blog
- AWS Cloud Compliance
- AWS Discussion Forums
- AWS Documentation
- AWS Knowledge Center
- AWS Support
- AWS Support Center
- The Amazon Builders' Library
- The Official AWS Podcast

OPS01-BP07 Manage benefits and risks

Manage benefits and risks to make informed decisions when determining where to focus efforts. For example, it may be beneficial to deploy a workload with unresolved issues so that significant new features can be made available to customers. It may be possible to mitigate associated risks, or it may become unacceptable to allow a risk to remain, in which case you will take action to address the risk.

You might find that you want to emphasize a small subset of your priorities at some point in time. Use a balanced approach over the long term to ensure the development of needed capabilities and management of risk. Update your priorities as needs change

Common anti-patterns:

- You have decided to include a library that does everything you need that one of your developers found on the internet. You have not evaluated the risks of adopting this library from an unknown source and do not know if it contains vulnerabilities or malicious code.

- You have decided to develop and deploy a new feature instead of fixing an existing issue. You have not evaluated the risks of leaving the issue in place until the feature is deployed and do not know what the impact will be on your customers.
- You have decided to not deploy a feature frequently requested by customers because of unspecified concerns from your compliance team.

Benefits of establishing this best practice: Identifying the available benefits of your choices, and being aware of the risks to your organization, helps you to make informed decisions.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Manage benefits and risks: Balance the benefits of decisions against the risks involved.
  - Identify benefits: Identify benefits based on business goals, needs, and priorities. Examples include time-to-market, security, reliability, performance, and cost.
  - Identify risks: Identify risks based on business goals, needs, and priorities. Examples include time-to-market, security, reliability, performance, and cost.
  - Assess benefits against risks and make informed decisions: Determine the impact of benefits and risks based on goals, needs, and priorities of your key stakeholders, including business, development, and operations. Evaluate the value of the benefit against the probability of the risk being realized and the cost of its impact. For example, emphasizing speed-to-market over reliability might provide competitive advantage. However, it may result in reduced uptime if there are reliability issues.

OPS 2. How do you structure your organization to support your business outcomes?

Your teams must understand their part in achieving business outcomes. Teams should understand their roles in the success of other teams, the role of other teams in their success, and have shared goals. Understanding responsibility, ownership, how decisions are made, and who has authority to make decisions will help focus efforts and maximize the benefits from your teams.

Best practices

- OPS02-BP01 Resources have identified owners
- OPS02-BP02 Processes and procedures have identified owners

- OPS02-BP03 Operations activities have identified owners responsible for their performance
- OPS02-BP04 Team members know what they are responsible for
- OPS02-BP05 Mechanisms exist to identify responsibility and ownership
- OPS02-BP06 Mechanisms exist to request additions, changes, and exceptions
- OPS02-BP07 Responsibilities between teams are predefined or negotiated

OPS02-BP01 Resources have identified owners

Resources for your workload must have identified owners for change control, troubleshooting, and other functions. Owners are assigned for workloads, accounts, infrastructure, platforms, and applications. Ownership is recorded using tools like a central register or metadata attached to resources. The business value of components informs the processes and procedures applied to them.

Desired outcome:

- Resources have identified owners using metadata or a central register.
- Team members can identify who owns resources.
- Accounts have a single owner where possible.

Common anti-patterns:

- The alternate contacts for your AWS accounts are not populated.
- Resources lack tags that identify what teams own them.
- You have an ITSM queue without an email mapping.
- Two teams have overlapping ownership of a critical piece of infrastructure.

Benefits of establishing this best practice:

- Change control for resources is straightforward with assigned ownership.
- You can involve the right owners when troubleshooting issues.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Define what ownership means for the resource use cases in your environment. Ownership can mean who oversees changes to the resource, supports the resource during troubleshooting, or who is financially accountable. Specify and record owners for resources, including name, contact information, organization, and team.

Customer example

AnyCompany Retail defines ownership as the team or individual that owns changes and support for resources. They leverage AWS Organizations to manage their AWS accounts. Alternate account contacts are configuring using group inboxes. Each ITSM queue maps to an email alias. Tags identify who own AWS resources. For other platforms and infrastructure, they have a wiki page that identifies ownership and contact information.

Implementation steps

1. Start by defining ownership for your organization. Ownership can imply who owns the risk for the resource, who owns changes to the resource, or who supports the resource when troubleshooting. Ownership could also imply financial or administrative ownership of the resource.
2. Use AWS Organizations to manage accounts. You can manage the alternate contacts for your accounts centrally.
    a. Using company owned email addresses and phone numbers for contact information helps you to access them even if the individuals whom they belong to are no longer with your organization. For example, create separate email distribution lists for billing, operations, and security and configure these as Billing, Security, and Operations contacts in each active AWS account. Multiple people will receive AWS notifications and be able to respond, even if someone is on vacation, changes roles, or leaves the company.
    b. If an account is not managed by AWS Organizations, alternate account contacts help AWS get in contact with the appropriate personnel if needed. Configure the account's alternate contacts to point to a group rather than an individual.
3. Use tags to identify owners for AWS resources. You can specify both owners and their contact information in separate tags.
    a. You can use AWS Config rules to enforce that resources have the required ownership tags.
    b. For in-depth guidance on how to build a tagging strategy for your organization, see AWS Tagging Best Practices whitepaper.

4. For other resources, platforms, and infrastructure, create documentation that identifies ownership. This should be accessible to all team members.

Level of effort for the implementation plan: Low. Leverage account contact information and tags to assign ownership of AWS resources. For other resources you can use something as simple as a table in a wiki to record ownership and contact information, or use an ITSM tool to map ownership.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners - The processes and procedures to support resources depends on resource ownership.
- OPS02-BP04 Team members know what they are responsible for - Team members should understand what resources they are owners of.
- OPS02-BP05 Mechanisms exist to identify responsibility and ownership - Ownership needs to be discoverable using mechanisms like tags or account contacts.

Related documents:

- AWS Account Management - Updating contact information
- AWS Config Rules - required-tags
- AWS Organizations - Updating alternative contacts in your organization
- AWS Tagging Best Practices whitepaper

Related examples:

- AWS Config Rules - Amazon EC2 with required tags and valid values

Related services:

- AWS Config
- AWS Organizations

OPS02-BP02 Processes and procedures have identified owners

Understand who has ownership of the definition of individual processes and procedures, why those specific process and procedures are used, and why that ownership exists. Understanding the reasons that specific processes and procedures are used aids in identification of improvement opportunities.

Benefits of establishing this best practice: Understanding ownership identifies who can approve improvements, implement those improvements, or both.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Process and procedures have identified owners responsible for their definition: Capture the processes and procedures used in your environment and the individual or team responsible for their definition.
  - Identify process and procedures: Identify the operations activities conducted in support of your workloads. Document these activities in a discoverable location.
  - Define who owns the definition of a process or procedure: Uniquely identify the individual or team responsible for the specification of an activity. They are responsible to ensure it can be successfully performed by an adequately skilled team member with the correct permissions, access, and tools. If there are issues with performing that activity, the team members performing it are responsible to provide the detailed feedback necessary for the activitiy to be improved.
  - Capture ownership in the metadata of the activity artifact: Procedures automated in services like AWS Systems Manager, through documents, and AWS Lambda, as functions, support capturing metadata information as tags. Capture resource ownership using tags or resource groups, specifying ownership and contact information. Use AWS Organizations to create tagging polices and ensure ownership and contact information are captured.

OPS02-BP03 Operations activities have identified owners responsible for their performance

Understand who has responsibility to perform specific activities on defined workloads and why that responsibility exists. Understanding who has responsibility to perform activities informs who will conduct the activity, validate the result, and provide feedback to the owner of the activity.

Benefits of establishing this best practice: Understanding who is responsible to perform an activity informs whom to notify when action is needed and who will perform the action, validate the result, and provide feedback to the owner of the activity.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Operations activities have identified owners responsible for their performance: Capture the responsibility for performing processes and procedures used in your environment
  - Identify process and procedures: Identify the operations activities conducted in support of your workloads. Document these activities in a discoverable location.
  - Define who is responsible to perform each activity: Identify the team responsible for an activity. Ensure they have the details of the activity, and the necessary skills and correct permissions, access, and tools to perform the activity. They must understand the condition under which it is to be performed (for example, on an event or schedule). Make this information discoverable so that members of your organization can identify who they need to contact, team or individual, for specific needs.

OPS02-BP04 Team members know what they are responsible for

Understanding the responsibilities of your role and how you contribute to business outcomes informs the prioritization of your tasks and why your role is important. This helps team members to recognize needs and respond appropriately.

Benefits of establishing this best practice: Understanding your responsibilities informs the decisions you make, the actions you take, and your hand off activities to their proper owners.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Ensure team members understand their roles and responsibilities: Identify team members roles and responsibilities and ensure they understand the expectations of their role. Make this information discoverable so that members of your organization can identify who they need to contact, team or individual, for specific needs.

OPS02-BP05 Mechanisms exist to identify responsibility and ownership

Where no individual or team is identified, there are defined escalation paths to someone with the authority to assign ownership or plan for that need to be addressed.

Benefits of establishing this best practice: Understanding who has responsbility or ownership allows you to reach out to the proper team or team member to make a request or transition a task. Having an identified person who has the authority to assign responsbility or ownership or plan to address needs reduces the risk of inaction and needs not being addressed.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Mechanisms exist to identify responsibility and ownership: Provide accessible mechanisms for members of your organization to discover and identify ownership and responsibility. These mechanisms will help them to identify who to contact, team or individual, for specific needs.

OPS02-BP06 Mechanisms exist to request additions, changes, and exceptions

You can make requests to owners of processes, procedures, and resources. Requests include additions, changes, and exceptions. These requests go through a change management process. Make informed decisions to approve requests where viable and determined to be appropriate after an evaluation of benefits and risks.

Desired outcome:

- You can make requests to change processes, procedures, and resources based on assigned ownership.
- Changes are made in a deliberate manner, weighing benefits and risks.

Common anti-patterns:

- You must update the way you deploy your application, but there is no way to request a change to the deployment process from the operations team.
- The disaster recovery plan must be updated, but there is no identified owner to request changes to.

Benefits of establishing this best practice:

- Processes, procedures, and resources can evolve as requirements change.
- Owners can make informed decisions when to make changes.
- Changes are made in a deliberate manner.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To implement this best practice, you need to be able to request changes to processes, procedures, and resources. The change management process can be lightweight. Document the change management process.

Customer example

AnyCompany Retail uses a responsibility assignment (RACI) matrix to identify who owns changes for processes, procedures, and resources. They have a documented change management process that's lightweight and easy to follow. Using the RACI matrix and the process, anyone can submit change requests.

Implementation steps

1. Identify the processes, procedures, and resources for your workload and the owners for each. Document them in your knowledge management system.
   a. If you have not implemented OPS02-BP01 Resources have identified owners, OPS02-BP02 Processes and procedures have identified owners, or OPS02-BP03 Operations activities have identified owners responsible for their performance, start with those first.
2. Work with stakeholders in your organization to develop a change management process. The process should cover additions, changes, and exceptions for resources, processes, and procedures.
   a. You can use AWS Systems Manager Change Manager as a change management platform for workload resources.
3. Document the change management process in your knowledge management system.

Level of effort for the implementation plan: Medium. Developing a change management process requires alignment with multiple stakeholders across your organization.

Resources

Related best practices:

- OPS02-BP01 Resources have identified owners - Resources need identified owners before you build a change management process.
- OPS02-BP02 Processes and procedures have identified owners - Processes need identified owners before you build a change management process.
- OPS02-BP03 Operations activities have identified owners responsible for their performance - Operations activities need identified owners before you build a change management process.

Related documents:

- AWS Prescriptive Guidance - Foundation palybook for AWS large migrations: Creating RACI matrices
- Change Management in the Cloud Whitepaper

Related services:

- AWS Systems Manager Change Manager

OPS02-BP07 Responsibilities between teams are predefined or negotiated

Have defined or negotiated agreements between teams describing how they work with and support each other (for example, response times, service level objectives, or service-level agreements). Inter-team communications channels are documented. Understanding the impact of the teams' work on business outcomes and the outcomes of other teams and organizations informs the prioritization of their tasks and helps them respond appropriately.

When responsibility and ownership are undefined or unknown, you are at risk of both not addressing necessary activities in a timely fashion and of redundant and potentially conflicting efforts emerging to address those needs.

Desired outcome:

- Inter-team working or support agreements are agreed to and documented.
- Teams that support or work with each other have defined communication channels and response expectations.

Common anti-patterns:

- An issue occurs in production and two separate teams start troubleshooting independent of each other. Their siloed efforts extend the outage.
- The operations team needs assistance from the development team but there is no agreed to response time. The request is stuck in the backlog.

Benefits of establishing this best practice:

- Teams know how to interact and support each other.
- Expectations for responsiveness are known.
- Communications channels are clearly defined.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Implementing this best practice means that there is no ambiguity about how teams work with each other. Formal agreements codify how teams work together or support each other. Inter-team communication channels are documented.

Customer example

AnyCompany Retail's SRE team has a service level agreement with their development team. Whenever the development team makes a request in their ticketing system, they can expect a response within fifteen minutes. If there is a site outage, the SRE team takes lead in the investigation with support from the development team.

Implementation steps

1. Working with stakeholders across your organization, develop agreements between teams based on processes and procedures.
   a. If a process or procedure is shared between two teams, develop a runbook on how the teams will work together.
   b. If there are dependencies between teams, agree to a response SLA for requests.
2. Document responsibilities in your knowledge management system.

Level of effort for the implementation plan: Medium. If there are no existing agreements between teams, it can take effort to come to agreement with stakeholders across your organization.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners - Process ownership must be identified before setting agreements between teams.
- OPS02-BP03 Operations activities have identified owners responsible for their performance - Operations activities ownership must be identified before setting agreements between teams.

Related documents:

- AWS Executive Insights - Empowering Innovation with the Two-Pizza Team
- Introduction to DevOps on AWS - Two-Pizza Teams

OPS 3. How does your organizational culture support your business outcomes?

Provide support for your team members so that they can be more effective in taking action and supporting your business outcome.

Best practices

- OPS03-BP01 Executive Sponsorship
- OPS03-BP02 Team members are empowered to take action when outcomes are at risk
- OPS03-BP03 Escalation is encouraged
- OPS03-BP04 Communications are timely, clear, and actionable
- OPS03-BP05 Experimentation is encouraged
- OPS03-BP06 Team members are encouraged to maintain and grow their skill sets
- OPS03-BP07 Resource teams appropriately
- OPS03-BP08 Diverse opinions are encouraged and sought within and across teams

OPS03-BP01 Executive Sponsorship

Senior leadership clearly sets expectations for the organization and evaluates success. Senior leadership is the sponsor, advocate, and driver for the adoption of best practices and evolution of the organization

Benefits of establishing this best practice: Engaged leadership, clearly communicated expectations, and shared goals ensures that team members know what is expected of them. Evaluating success aids in identification of barriers to success so that they can be addressed through intervention by the sponsor advocate or their delegates.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Executive Sponsorship: Senior leadership clearly sets expectations for the organization and evaluates success. Senior leadership is the sponsor, advocate, and driver for the adoption of best practices and evolution of the organization
  - Set expectations: Define and publish goals for your organizations including how they will be measured.
  - Track achievement of goals: Measure the incremental achievement of goals regularly and share the results so that appropriate action can be taken if outcomes are at risk.
  - Provide the resources necessary to achieve your goals: Regularly review if resources are still appropriate, of if additional resources are needed based on: new information, changes to goals, responsibilities, or your business environment.
  - Advocate for your teams: Remain engaged with your teams so that you understand how they are doing and if there are external factors affecting them. When your teams are impacted by external factors, reevaluate goals and adjust targets as appropriate. Identify obstacles that are impeding your teams progress. Act on behalf of your teams to help address obstacles and remove unnecessary burdens.
  - Be a driver for adoption of best practices: Acknowledge best practices that provide quantifiable benefits and recognize the creators and adopters. Encourage further adoption to magnify the benefits achieved.
  - Be a driver for evolution of for your teams: Create a culture of continual improvement. Encourage both personal and organizational growth and development. Provide long term targets to strive for that will require incremental achievement over time. Adjust this vision to compliment your needs, business goals, and business environment as they change.

OPS03-BP02 Team members are empowered to take action when outcomes are at risk

The workload owner has defined guidance and scope empowering team members to respond when outcomes are at risk. Escalation mechanisms are used to get direction when events are outside of the defined scope.

Benefits of establishing this best practice: By testing and validating changes early, you are able to address issues with minimized costs and limit the impact on your customers. By testing prior to deployment you minimize the introduction of errors.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Team members are empowered to take action when outcomes are at risk: Provide your team members the permissions, tools, and opportunity to practice the skills necessary to respond effectively.
  - Give your team members opportunity to practice the skills necessary to respond: Provide alternative safe environments where processes and procedures can be tested and trained upon safely. Perform game days to allow team members to gain experience responding to real world incidents in simulated and safe environments.
  - Define and acknowledge team members' authority to take action: Specifically define team members authority to take action by assigning permissions and access to the workloads and components they support. Acknowledge that they are empowered to take action when outcomes are at risk.

OPS03-BP03 Escalation is encouraged

Team members have mechanisms and are encouraged to escalate concerns to decision makers and stakeholders if they believe outcomes are at risk. Escalation should be performed early and often so that risks can be identified, and prevented from causing incidents.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Encourage early and frequent escalation: Organizationally acknowledge that escalation early and often is the best practice. Organizationally acknowledge and accept that escalations may prove to be unfounded, and that it is better to have the opportunity to prevent an incident then to miss that opportunity by not escalating.

- Have a mechanism for escalation: Have documented procedures defining when and how escalation should occur. Document the series of people with increasing authority to take action or approve action and their contact information. Escalation should continue until the team member is satisfied that they have handed off the risk to a person able to address it, or they have contacted the person who owns the risk and liability for the operation of the workload. It is that person who ultimately owns all decisions with respect to their workload. Escalations should include the nature of the risk, the criticality of the workload, who is impacted, what the impact is, and the urgency, that is, when is the impact expected.
- Protect employees who escalate: Have policy that protects team members from retribution if they escalate around a non-responsive decision maker or stakeholder. Have mechanisms in place to identify if this is occurring and respond appropriately.

OPS03-BP04 Communications are timely, clear, and actionable

Mechanisms exist and are used to provide timely notice to team members of known risks and planned events. Necessary context, details, and time (when possible) are provided to support determining if action is necessary, what action is required, and to take action in a timely manner. For example, providing notice of software vulnerabilities so that patching can be expedited, or providing notice of planned sales promotions so that a change freeze can be implemented to avoid the risk of service disruption. Planned events can be recorded in a change calendar or maintenance schedule so that team members can identify what activities are pending.

Desired outcome:

- Communications provide context, details, and time expectations.
- Team members have a clear understanding of when and how to act in response to communications.
- Leverage change calendars to provide notice of expected changes.

Common anti-patterns:

- An alert happens several times per week that is a false positive. You mute the notification each time it happens.
- You are asked to make a change to your security groups but are not given an expectation of when it should happen.

- You receive constant notifications in chat when systems scale up but no action is necessary. You avoid the chat channel and miss an important notification.
- A change is made to production without informing the operations team. The change creates an alert and the on-call team is activated.

Benefits of establishing this best practice:

- Your organization avoids alert fatigue.
- Team members can act with the necessary context and expectations.
- Changes can be made during change windows, reducing risk.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To implement this best practice, you must work with stakeholders across your organization to agree to communication standards. Publicize those standards to your organization. Identify and remove alerts that are false-positive or always on. Utilize change calendars so team members know when actions can be taken and what activities are pending. Verify that communications lead to clear actions with necessary context.

Customer example

AnyCompany Retail uses chat as their main communication medium. Alerts and other information populate specific channels. When someone must act, the desired outcome is clearly stated, and in many cases, they are given a runbook or playbook to use. They use a change calendar to schedule major changes to production systems.

Implementation steps

1. Analyze your alerts for false-positives or alerts that are constantly created. Remove or change them so that they start when human intervention is required. If an alert is initiated, provide a runbook or playbook.
   a. You can use AWS Systems Manager Documents to build playbooks and runbooks for alerts.
2. Mechanisms are in place to provide notification of risks or planned events in a clear and actionable way with enough notice to allow appropriate responses. Use email lists or chat channels to send notifications ahead of planned events.

   a. Amazon Q Developer in chat applications can be used to send alerts and respond to events within your organizations messaging platform.

3. Provide an accessible source of information where planned events can be discovered. Provide notifications of planned events from the same system.

   a. AWS Systems Manager Change Calendar can be used to create change windows when changes can occur. This provides team members notice when they can make changes safely.

4. Monitor vulnerability notifications and patch information to understand vulnerabilities in the wild and potential risks associated to your workload components. Provide notification to team members so that they can act.

   a. You can subscribe to AWS Security Bulletins to receive notifications of vulnerabilities on AWS.

Resources

Related best practices:

- OPS07-BP03 Use runbooks to perform procedures - Make communications actionable by supplying a runbook when the outcome is known.
- OPS07-BP04 Use playbooks to investigate issues - In the case where the outcome is unknown, playbooks can make communications actionable.

Related documents:

- AWS Security Bulletins
- Open CVE

Related examples:

- Well-Architected Labs: Inventory and Patch Management (Level 100)

Related services:

- Amazon Q Developer in chat applications
- AWS Systems Manager Change Calendar
- AWS Systems Manager Documents

OPS03-BP05 Experimentation is encouraged

Experimentation is a catalyst for turning new ideas into products and features. It accelerates learning and keeps team members interested and engaged. Team members are encouraged to experiment often to drive innovation. Even when an undesired result occurs, there is value in knowing what not to do. Team members are not punished for successful experiments with undesired results.

Desired outcome:

- Your organization encourages experimentation to foster innovation.
- Experiments are used as an opportunity to learn.

Common anti-patterns:

- You want to run an A/B test but there is no mechanism to run the experiment. You deploy a UI change without the ability to test it. It results in a negative customer experience.
- Your company only has a stage and production environment. There is no sandbox environment to experiment with new features or products so you must experiment within the production environment.

Benefits of establishing this best practice:

- Experimentation drives innovation.
- You can react faster to feedback from users through experimentation.
- Your organization develops a culture of learning.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Experiments should be run in a safe manner. Leverage multiple environments to experiment without jeopardizing production resources. Use A/B testing and feature flags to test experiments. Provide team members the ability to conduct experiments in a sandbox environment.

Customer example

AnyCompany Retail encourages experimentation. Team members can use 20% of their work week to experiment or learn new technologies. They have a sandbox environment where they can innovate. A/B testing is used for new features to validate them with real user feedback.

Implementation steps

1. Work with leadership across your organization to support experimentation. Team members should be encouraged to conduct experiments in a safe manner.
2. Provide your team members with an environment where they can safely experiment. They must have access to an environment that is like production.
   a. You can use a separate AWS account to create a sandbox environment for experimentation. AWS Control Tower can be used to provision these accounts.
3. Use feature flags and A/B testing to experiment safely and gather user feedback.
   a. AWS AppConfig Feature Flags provides the ability to create feature flags.
   b. Amazon CloudWatch Evidently can be used to run A/B tests over a limited deployment.
   c. You can use AWS Lambda versions to deploy a new version of a function for beta testing.

Level of effort for the implementation plan: High. Providing team members with an environment to experiment in and a safe way to conduct experiments can require significant investment. You may also need to modify application code to use feature flags or support A/B testing.

Resources

Related best practices:

- OPS11-BP02 Perform post-incident analysis - Learning from incidents is an important driver for innovation along with experimentation.
- OPS11-BP03 Implement feedback loops - Feedback loops are an important part of experimentation.

Related documents:

- An Inside Look at the Amazon Culture: Experimentation, Failure, and Customer Obsession
- Best practices for creating and managing sandbox accounts in AWS
- Create a Culture of Experimentation Enabled by the Cloud
- Enabling experimentation and innovation in the cloud at SulAmérica Seguros

- Experiment More, Fail Less
- Organizing Your AWS Environment Using Multiple Accounts - Sandbox OU
- Using AWS AppConfig Feature Flags

Related videos:

- AWS On Air ft. Amazon CloudWatch Evidently | AWS Events
- AWS On Air San Fran Summit 2022 ft. AWS AppConfig Feature Flags integration with Jira
- AWS re:Invent 2022 - A deployment is not a release: Control your launches w/feature flags (BOA305-R)
- Programmatically Create an AWS account with AWS Control Tower
- Set Up a Multi-Account AWS Environment that Uses Best Practices for AWS Organizations

Related examples:

- AWS Innovation Sandbox
- End-to-end Personalization 101 for E-Commerce

Related services:

- Amazon CloudWatch Evidently
- AWS AppConfig
- AWS Control Tower

OPS03-BP06 Team members are encouraged to maintain and grow their skill sets

Teams must grow their skill sets to adopt new technologies, and to support changes in demand and responsibilities in support of your workloads. Growth of skills in new technologies is frequently a source of team member satisfaction and supports innovation. Support your team members' pursuit and maintenance of industry certifications that validate and acknowledge their growing skills. Cross train to promote knowledge transfer and reduce the risk of significant impact when you lose skilled and experienced team members with institutional knowledge. Provide dedicated structured time for learning.

AWS provides resources, including the AWS Getting Started Resource Center, AWS Blogs, AWS Online Tech Talks, AWS Events and Webinars, and the AWS Well-Architected Labs, that provide guidance, examples, and detailed walkthroughs to educate your teams.

AWS also shares best practices and patterns that we have learned through the operation of AWS in The Amazon Builders' Library and a wide variety of other useful educational material through the AWS Blog and The Official AWS Podcast.

You should take advantage of the education resources provided by AWS such as the Well-Architected labs, AWS Support (AWS Knowledge Center, AWS Discussion Forms, and AWS Support Center) and AWS Documentation to educate your teams. Reach out to AWS Support through AWS Support Center for help with your AWS questions.

AWS Training and Certification provides some free training through self-paced digital courses on AWS fundamentals. You can also register for instructor-led training to further support the development of your teams' AWS skills.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Team members are encouraged to maintain and grow their skill sets: To adopt new technologies, support innovation, and to support changes in demand and responsibilities in support of your workloads continuing education is necessary.
  - Provide resources for education: Provided dedicated structured time, access to training materials, lab resources, and support participation in conferences and professional organizations that provide opportunities for learning from both educators and peers. Provide junior team members' access to senior team members as mentors or allow them to shadow their work and be exposed to their methods and skills. Encourage learning about content not directly related to work in order to have a broader perspective.
  - Team education and cross-team engagement: Plan for the continuing education needs of your team members. Provide opportunities for team members to join other teams (temporarily or permanently) to share skills and best practices benefiting your entire organization
  - Support pursuit and maintenance of industry certifications: Support your team members acquiring and maintaining industry certifications that validate what they have learned, and acknowledge their accomplishments.

Resources

Related documents:

- AWS Getting Started Resource Center
- AWS Blogs
- AWS Cloud Compliance
- AWS Discussion Forms
- AWS Documentation
- AWS Online Tech Talks
- AWS Events and Webinars
- AWS Knowledge Center
- AWS Support
- AWS Training and Certification
- AWS Well-Architected Labs,
- The Amazon Builders' Library
- The Official AWS Podcast.

OPS03-BP07 Resource teams appropriately

Maintain team member capacity, and provide tools and resources to support your workload needs. Overtasking team members increases the risk of incidents resulting from human error. Investments in tools and resources (for example, providing automation for frequently performed activities) can scale the effectiveness of your team, helping them to support additional activities.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Resource teams appropriately: Ensure you have an understanding of the success of your teams and the factors that contribute to their success or lack of success. Act to support teams with appropriate resources.
  - Understand team performance: Measure the achievement of operational outcomes and the development of assets by your teams. Track changes in output and error rate over time. Engage with teams to understand the work related challenges that impact them (for example,

increasing responsibilities, changes in technology, loss of personnel, or increase in customers supported).

- Understand impacts on team performance: Remain engaged with your teams so that you understand how they are doing and if there are external factors affecting them. When your teams are impacted by external factors, reevaluate goals and adjust targets as appropriate. Identify obstacles that are impeding your teams progress. Act on behalf of your teams to help address obstacles and remove unnecessary burdens.
- Provide the resources necessary for teams to be successful: Regularly review if resources are still appropriate, of if additional resources are needed, and make appropriate adjustments to support teams.

OPS03-BP08 Diverse opinions are encouraged and sought within and across teams

Leverage cross-organizational diversity to seek multiple unique perspectives. Use this perspective to increase innovation, challenge your assumptions, and reduce the risk of confirmation bias. Grow inclusion, diversity, and accessibility within your teams to gain beneficial perspectives.

Organizational culture has a direct impact on team member job satisfaction and retention. Foster the engagement and capabilities of your team members to create the success of your business.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Seek diverse opinions and perspectives: Encourage contributions from everyone. Give voice to under-represented groups. Rotate roles and responsibilities in meetings.
  - Expand roles and responsibilities: Provide opportunity for team members to take on roles that they might not otherwise. They will gain experience and perspective from the role, and from interactions with new team members with whom they might not otherwise interact. They will bring their experience and perspective to the new role and team members they interact with. As perspective increases, additional business opportunities may emerge, or new opportunities for improvement may be identified. Have members within a team take turns at common tasks that others typically perform to understand the demands and impact of performing them.
  - Provide a safe and welcoming environment: Have policy and controls that protect team members' mental and physical safety within your organization. Team members should be able to interact without fear of reprisal. When team members feel safe and welcome they are more likely to be engaged and productive. The more diverse your organization the better your understanding can be of the people you support including your customers. When your

team members are comfortable, feel free to speak, and are confident they will be heard, they are more likely to share valuable insights (for example, marketing opportunities, accessibility needs, unserved market segments, unacknowledged risks in your environment).

- Enable team members to participate fully: Provide the resources necessary for your employees to participate fully in all work related activities. Team members that face daily challenges have developed skills for working around them. These uniquely developed skills can provide significant benefit to your organization. Supporting team members with necessary accommodations will increase the benefits you can receive from their contributions.

Prepare

Questions

- OPS 4. How do you design your workload so that you can understand its state?
- OPS 5. How do you reduce defects, ease remediation, and improve flow into production?
- OPS 6. How do you mitigate deployment risks?
- OPS 7. How do you know that you are ready to support a workload?

OPS 4. How do you design your workload so that you can understand its state?

Design your workload so that it provides the information necessary across all components (for example, metrics, logs, and traces) for you to understand its internal state. This allows you to provide effective responses when appropriate.

Best practices

- OPS04-BP01 Implement application telemetry
- OPS04-BP02 Implement and configure workload telemetry
- OPS04-BP03 Implement user activity telemetry
- OPS04-BP04 Implement dependency telemetry
- OPS04-BP05 Implement transaction traceability

OPS04-BP01 Implement application telemetry

Application telemetry is the foundation for observability of your workload. Your application should emit telemetry that provides insight into the state of the application and the achievement of

business outcomes. From troubleshooting to measuring the impact of a new feature, application telemetry informs the way you build, operate, and evolve your workload.

Application telemetry consists of metrics and logs. Metrics are diagnostic information, such as your pulse or temperature. Metrics are used collectively to describe the state of your application. Collecting metrics over time can be used to develop baselines and detect anomalies. Logs are messages that the application sends about its internal state or events that occur. Error codes, transaction identifiers, and user actions are examples of events that are logged.

Desired Outcome:

- Your application emits metrics and logs that provide insight into its health and the achievement of business outcomes.
- Metrics and logs are stored centrally for all applications in the workload.

Common anti-patterns:

- Your application doesn't emit telemetry. You are forced to rely upon your customers to tell you when something is wrong.
- A customer has reported that your application is unresponsive. You have no telemetry and are unable to confirm that the issue exists or characterize the issue without using the application yourself to understand the current user experience.

Benefits of establishing this best practice:

- You can understand the health of your application, the user experience, and the achievement of business outcomes.
- You can react quickly to changes in your application health.
- You can develop application health trends.
- You can make informed decisions about improving your application.
- You can detect and resolve application issues faster.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implementing application telemetry consists of three steps: identifying a location to store telemetry, identifying telemetry that describes the state of the application, and instrumenting the application to emit telemetry.

Customer example

AnyCompany Retail has a microservices based architecture. As part of their architectural design process, they identified application telemetry that would help them understand the state of each microservice. For example, the user cart service emits telemetry about events like add to cart, abandon cart, and length of time it took to add an item to the cart. All microservices log errors, warnings, and transaction information. Telemetry is sent to Amazon CloudWatch for storage and analysis.

Implementation steps

1. Identify a central location for telemetry storage for the applications in your workload. The location should support both collection of telemetry and analysis capabilities. Anomaly detection and automated insights are recommended features.
    a. Amazon CloudWatch provides telemetry collection, dashboards, analysis, and event generation capabilities.
2. To identify what telemetry you need, start by answering this question: what is the state of my application? Your application should emit logs and metrics that collectively answer this question. If you can't answer the questions with the existing application telemetry, work with business and engineering stakeholders to create a list of telemetry requirements.
    a. You can request expert technical advice from your AWS account team as you identify and develop new application telemetry.
3. Once the additional application telemetry has been identified, work with your engineering stakeholders to instrument your application.
    a. The AWS Distro for Open Telemetry provides APIs, libraries, and agents that collect application telemetry. This example demonstrates how to instrument a JavaScript application with custom metrics.
    b. If you want to understand the observability services that AWS offers, work through the One Observability Workshop or request support from your AWS account team.
    c. For a deeper dive into application telemetry, read the Instrumenting distributed systems for operational visibility article in the Amazon Builder's Library, which explains how Amazon

instruments applications and can serve as a guide for developing your own instrumentation guidelines.

Level of effort for the implementation plan: High. Instrumenting your application and centralizing telemetry storage can take significant investment.

Resources

Related best practices:

the section called "OPS04-BP02 Implement and configure workload telemetry" – Application telemetry is a component of workload telemetry. In order to understand the health of the overall workload you need to understand the health of individual applications that make up the workload.

the section called "OPS04-BP03 Implement user activity telemetry" – User activity telemetry is often a subset of application telemetry. User activity like add to cart events, click streams, or completed transactions provide insight into the user experience.

the section called "OPS04-BP04 Implement dependency telemetry" – Dependency checks are related to application telemetry and may be instrumented into your application. If your application relies on external dependencies like DNS or a database your application can emit metrics and logs on reachability, timeouts, and other events.

the section called "OPS04-BP05 Implement transaction traceability" – Tracing transactions across a workload requires each application to emit information about how they process shared events. The way individual applications handle these events is emitted through their application telemetry.

the section called "OPS08-BP02 Define workload metrics" – Workload metrics are the key health indicators for your workload. Key application metrics are a part of workload metrics.

Related documents:

- AWS Builders Library – Instrumenting Distributed Systems for Operational Visibility
- AWS Distro for OpenTelemetry
- AWS Well-Architected Operational Excellence Whitepaper – Design Telemetry
- Creating metrics from log events using filters
- Implementing Logging and Monitoring with Amazon CloudWatch
- Monitoring application health and performance with AWS Distro for OpenTelemetry

- New – How to better monitor your custom application metrics using Amazon CloudWatch Agent
- Observability at AWS
- Scenario – Publish metrics to CloudWatch
- Start Building – How to Monitor your Applications Effectively
- Using CloudWatch with an AWS SDK

Related videos:

- AWS re:Invent 2021 - Observability the open-source way
- Collect Metrics and Logs from Amazon EC2 instances with the CloudWatch Agent
- How to Easily Setup Application Monitoring for Your AWS Workloads - AWS Online Tech Talks
- Mastering Observability of Your Serverless Applications - AWS Online Tech Talks
- Open Source Observability with AWS - AWS Virtual Workshop

Related examples:

- AWS Logging & Monitoring Example Resources
- AWS Solution: Amazon CloudWatch Monitoring Framework
- AWS Solution: Centralized Logging
- One Observability Workshop

Related services:

- Amazon CloudWatch

OPS04-BP02 Implement and configure workload telemetry

Design and configure your workload to emit information about its internal state and current status, for example, API call volume, HTTP status codes, and scaling events. Use this information to help determine when a response is required.

Use a service such as Amazon CloudWatch to aggregate logs and metrics from workload components (for example, API logs from AWS CloudTrail, AWS Lambda metrics, Amazon VPC Flow Logs, and other services).

Common anti-patterns:

- Your customers are complaining about poor performance. There are no recent changes to your application and so you suspect an issue with a workload component. You have no telemetry to analyze to determine what component or components are contributing to the poor performance.
- Your application is unreachable. You lack the telemetry to determine if it's a networking issue.

Benefits of establishing this best practice: Understanding what is going on inside your workload helps you to respond if necessary.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Implement log and metric telemetry: Instrument your workload to emit information about its internal state, status, and the achievement of business outcomes. Use this information to determine when a response is required.
  - Gaining better observability of your VMs with Amazon CloudWatch - AWS Online Tech Talks
  - How Amazon CloudWatch works
  - What is Amazon CloudWatch?
  - Using Amazon CloudWatch metrics
  - What is Amazon CloudWatch Logs?
    - Implement and configure workload telemetry: Design and configure your workload to emit information about its internal state and current status (for example, API call volume, HTTP status codes, and scaling events).
      - Amazon CloudWatch metrics and dimensions reference
      - AWS CloudTrail
      - What Is AWS CloudTrail?
      - VPC Flow Logs

Resources

Related documents:

- AWS CloudTrail
- Amazon CloudWatch Documentation

- Amazon CloudWatch metrics and dimensions reference
- How Amazon CloudWatch works
- Using Amazon CloudWatch metrics
- VPC Flow Logs
- What Is AWS CloudTrail?
- What is Amazon CloudWatch Logs?
- What is Amazon CloudWatch?

Related videos:

- Application Performance Management on AWS
- Gaining Better Observability of Your VMs with Amazon CloudWatch
- Gaining better observability of your VMs with Amazon CloudWatch - AWS Online Tech Talks

OPS04-BP03 Implement user activity telemetry

Instrument your application code to emit information about user activity. Examples of user activity include click streams or started, abandoned, and completed transactions. Use this information to help understand how the application is used, patterns of usage, and to determine when a response is required. Capturing real user activity allows you to build synthetic activity that can be used to monitor and test your workload in production.

Desired outcome:

- Your workload emits telemetry about user activity across all applications.
- You leverage synthetic user activity to monitor your application during off-peak hours.

Common anti-patterns:

- Your developers have deployed a new feature without user telemetry. You cannot tell if your customers are using the feature without asking them.
- After a deployment to your front-end application, you see increased utilization. Because you lack user activity telemetry, it is difficult to identify the exact issue.
- An issue occurs in your application during off-peak hours. You do not notice the issue until the morning when your users come online because you have not configured synthetic user activity.

Benefits of establishing this best practice:

- Understand common user patterns or unexpected behaviors to optimize functionality of the application to fit your business goals.
- Monitor the application from the perspective of your users to detect problems with user experience, such as broken links or slow click responses
- Identify the root cause of issues by tracing the steps your impacted user has taken.
- Synthetic user activity can provide early warning signs of performance degradation during off-peak hours, allowing you to take corrective action before actual users are affected.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Design your application code to emit information about user activity. Use this information to help understand how the application is used, patterns of usage, and to determine when a response is required. Utilize synthetic user activity to provide insight into application performance during off-peak hours.

Customer example

AnyCompany Retail implements user activity telemetry at several layers in their application. The front-end telemetry tracks pointer and movement events while the backend microservices emit telemetry tracking events like adding an item to the user's cart and checking out. Together they provide observability into the user experience. AnyCompany Retail also uses synthetic user telemetry to catch problems when there are fewer users on the workload.

Implementation steps

1. Instrument your application to emit telemetry (metrics, events, logs, and traces) about user activity. Once instrumented, front-end components emit telemetry automatically as the user interacts with the user interface. Backend applications emit telemetry on user events and transactions.
   a. Amazon CloudWatch RUM can provide insight into end user experience for front-end applications.
   b. You can use the AWS Distro for Open Telemetry to instrument and capture telemetry from your applications.

   c. Amazon Pinpoint can analyze user behavior through campaigns, providing insight on user engagement.
   d. Customers with Enterprise Support can request the Building a Monitoring Strategy Workshop from their Technical Account Manager. This workshop helps you build an observability strategy for your workload.
2. Establish synthetic user activity to monitor your application. Synthetic user activity simulates user actions to validate that your application is working properly.
   a. Amazon CloudWatch Synthetics can simulate user activity using canaries.

Level of effort for the implementation plan: High. It may take significant development effort to fully instrument your application to collect user activity telemetry.

Resources

Related best practices:

- OPS04-BP01 Implement application telemetry - Application telemetry is required in order to build in user activity telemetry.
- OPS04-BP02 Implement and configure workload telemetry - Some user activity telemetry may also be considered workload telemetry.

Related documents:

- How to Monitor your Applications Effectively

Related videos:

- AWS re:Invent 2020: Monitoring production services at Amazon
- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM
- Testing and Monitoring APIs on AWS - AWS Online Tech Talks

Related examples:

- Amazon CloudWatch RUM Web Client
- AWS Distro for Open Telemetry

- Implementing Real User Monitoring of Amplify Application using Amazon CloudWatch RUM
- One Observability Workshop

Related services:

- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Amazon Pinpoint

OPS04-BP04 Implement dependency telemetry

Design and configure your workload to emit information about the status of resources it depends on. These are resources that are external to your workload. Examples of external dependencies can include external databases, DNS, and network connectivity. Use this information to determine when a response is required and provide additional context on workload state.

Desired outcome:

- Your workload emits telemetry about the status of external dependencies.
- You are notified when dependencies are unhealthy.

Common anti-patterns:

- Your users cannot reach your site. You are unable to determine if the reason is a DNS issue without manually performing a check to see if your DNS provider is working.
- Your shopping cart application is unable to complete transactions. You are unable to determine if it's a problem with your credit card processing provider without contacting them to verify.

Benefits of establishing this best practice:

- Monitoring external dependencies provides advance notice of issues.
- Awareness of the health of your dependencies assists in troubleshooting.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Work with stakeholders to identify external dependencies that your workload depends on. External dependencies can include external databases, APIs, or network connectivity between your workload and resources in other environments. Develop a monitoring strategy to provide awareness of the health of dependencies and proactively alarm if the status changes.

Customer example

AnyCompany Retail's ecommerce workload relies on a database located in another environment. Every night, data is populated in the database for use in the ecommerce platform. The network connectivity and database support are owned by other teams. The ecommerce team configured several canary alarms to alert them when the network connectivity drops, the database is unreachable, and when the job fails to complete.

Implementation steps

1. Identify external dependencies that your workload relies on. Implement telemetry to track the health or reachability of dependencies.
   a. AWS customers can use the AWS Health Dashboard to monitor the health of AWS services and receive notifications of health events.
   b. Amazon CloudWatch Synthetics can be used to monitor APIs, URLs, and website contents.
2. Set up alerts to notify your organization when a dependency is unhealthy or unreachable.
   a. Customers with Enterprise Support can request the Building a Monitoring Strategy Workshop from their Technical Account Manager. This workshop will help you build an observability strategy for your workload.
3. Identify contacts for dependencies in cases where the dependency is unhealthy. Document how to contact the dependency owner, service agreements, and escalation process.

Level of effort for the implementation plan: Medium. Implementing dependency telemetry may require building custom monitoring solutions.

Resources

Related best practices:

- OPS04-BP01 Implement application telemetry - You may build dependency monitoring into your application telemetry.

Related documents:

- Monitor your private internal endpoints 24x7 using CloudWatch Synthetics

Related videos:

- AWS re:Invent 2018: Monitor All Your Things: Amazon CloudWatch in Action with BBC
- AWS re:Invent 2022 - Developing an observability strategy
- AWS re:Invent 2022 - Observability best practices at Amazon

Related examples:

- One Observability Workshop
- Well-Architected Labs - Dependency Monitoring

Related services:

- Amazon CloudWatch Synthetics
- AWS Health

OPS04-BP05 Implement transaction traceability

Implement your application code and configure your workload components to emit events, which are started as a result of single logical operations and consolidated across various boundaries of your workload. Generate maps to see how traces flow across your workload and services. Gain insight into the relationships between components, and identify and analyze issues. Use the collected information to determine when a response is required and to assist you in identifying the factors contributing to an issue.

Desired outcome:

- Collect transaction traces across your workload to gain insight into the relationship between components.
- Generate maps to gain a better understanding of how transactions and events flow across your workload.

Common anti-patterns:

- You have implemented a serverless microservices architecture spanning multiple accounts. Your customers are experiencing intermittent performance issues. You are unable to discover which function or component is responsible because you lack transaction traceability.
- There is a performance bottleneck in your workload. Because you lack transaction traceability, you are unable to see the relationship between your application components and identify the bottleneck.
- The identifier used for traces is not globally unique, resulting in a tracing collision when analyzing workload behavior.

Benefits of establishing this best practice:

- Understanding the flow of transactions across your workload provides insight into the expected behavior of your workload transactions.
- You can see variations from expected behavior across your workload and you can respond if necessary.
- You can pinpoint transactions by their unique generated identifier independent from where they were generated.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Design your application and workload to emit information about the flow of transactions across system components. Data to include in transactions are a globally unique transaction identifier, transaction stage, active component, and time to complete activity. Use this information to determine what is in progress, what is complete, and what the results of completed activities are.

Customer example

At AnyCompany Retail, all transactions have a globally unique UUID generated. This UUID is passed between microservices during transactions. The UUID is used to create transaction traces as users interact with the workload. A map of the workload topology is generated with the traces and is used to troubleshoot workload issues and improve performance.

Implementation steps

1. Instrument the applications in your workload to emit transaction traces. This can be done by generating a unique identifier for each transaction and passing the identifier between applications.
   a. You can use auto-instrumentation in the AWS Distro for OpenTelemetry to implement traces in your existing applications without modifying your application code.
2. Generate maps of your application topology. Use these maps to improve performance, gain insights, and aid in troubleshooting.
   a. AWS X-Ray can generate maps of the applications in your workload.

Level of effort for the implementation plan: Medium. Implementing transaction traces may require moderate development effort.

Resources

Related best practices:

- OPS04-BP01 Implement application telemetry - Application telemetry covers transaction traceability and handling and needs to be implementing first.

Related documents:

- Discover application issues and get notifications with AWS X-Ray Insights
- How Wealthfront utilizes AWS X-Ray to analyze and debug distributed applications
- New for AWS Distro for OpenTelemetry – Tracing Support is Now Generally Available

Related videos:

- AWS re:Invent 2018: Deep Dive into AWS X-Ray: Monitor Modern Applications (DEV324)
- AWS re:Invent 2022 - Building observable applications with OpenTelemetry (BOA310)
- AWS re:Invent 2022 - Observability the open-source way (COP301-R)
- Capturing Trace Data with the AWS Distro for OpenTelemetry
- Optimize Application Performance with AWS X-Ray

Related examples:

- AWS X-Ray Multi API Gateway Tracing Example

Related services:

- AWS Distro for OpenTelemetry
- AWS X-Ray

OPS 5. How do you reduce defects, ease remediation, and improve flow into production?

Adopt approaches that improve flow of changes into production, that activate refactoring, fast feedback on quality, and bug fixing. These accelerate beneficial changes entering production, limit issues deployed, and achieve rapid identification and remediation of issues introduced through deployment activities.

Best practices

- OPS05-BP01 Use version control
- OPS05-BP02 Test and validate changes
- OPS05-BP03 Use configuration management systems
- OPS05-BP04 Use build and deployment management systems
- OPS05-BP05 Perform patch management
- OPS05-BP06 Share design standards
- OPS05-BP07 Implement practices to improve code quality
- OPS05-BP08 Use multiple environments
- OPS05-BP09 Make frequent, small, reversible changes
- OPS05-BP10 Fully automate integration and deployment

OPS05-BP01 Use version control

Use version control to activate tracking of changes and releases.

Many AWS services offer version control capabilities. Use a revision or source control system such as AWS CodeCommit to manage code and other artifacts, such as version-controlled AWS CloudFormation templates of your infrastructure.

Common anti-patterns:

- You have been developing and storing your code on your workstation. You have had an unrecoverable storage failure on the workstation your code is lost.
- After overwriting the existing code with your changes, you restart your application and it is no longer operable. You are unable to revert to the change.
- You have a write lock on a report file that someone else needs to edit. They contact you asking that you stop work on it so that they can complete their tasks.
- Your research team has been working on a detailed analysis that will shape your future work. Someone has accidentally saved their shopping list over the final report. You are unable to revert the change and will have to recreate the report.

Benefits of establishing this best practice: By using version control capabilities you can easily revert to known good states, previous versions, and limit the risk of assets being lost.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Use version control: Maintain assets in version controlled repositories. Doing so supports tracking changes, deploying new versions, detecting changes to existing versions, and reverting to prior versions (for example, rolling back to a known good state in the event of a failure). Integrate the version control capabilities of your configuration management systems into your procedures.
  - Introduction to AWS CodeCommit
  - What is AWS CodeCommit?

Resources

Related documents:

- What is AWS CodeCommit?

Related videos:

- Introduction to AWS CodeCommit

OPS05-BP02 Test and validate changes

Every change deployed must be tested to avoid errors in production. This best practice is focused on testing changes from version control to artifact build. Besides application code changes, testing should include infrastructure, configuration, security controls, and operations procedures. Testing takes many forms, from unit tests to software component analysis (SCA). Move tests further to the left in the software integration and delivery process results in higher certainty of artifact quality.

Your organization must develop testing standards for all software artifacts. Automated tests reduce toil and avoid manual test errors. Manual tests may be necessary in some cases. Developers must have access to automated test results to create feedback loops that improve software quality.

Desired outcome:

- All software changes are tested before they are delivered.
- Developers have access to test results.
- Your organization has a testing standard that applies to all software changes.

Common anti-patterns:

- You deploy a new software change without any tests. It fails to run in production, which leads to an outage.
- New security groups are deployed with CloudFormation without being tested in a pre-production environment. The security groups make your app unreachable for your customers.
- A method is modified but there are no unit tests. The software fails when it is deployed to production.

Benefits of establishing this best practice:

- The change fail rate of software deployments is reduced.
- Software quality is improved.
- Developers have increased awareness on the viability of their code.
- Security policies can be rolled out with confidence to support organization's compliance
- Infrastructure changes such as automatic scaling policy updates are tested in advance to meet traffic needs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Testing is done on all changes, from application code to infrastructure, as part of your continuous integration practice. Test results are published so that developers have fast feedback. Your organization has a testing standard that all changes must pass.

Customer example

As part of their continuous integration pipeline, AnyCompany Retail conducts several types of tests on all software artifacts. They practice test driven development so all software has unit tests. Once the artifact is built, they run end-to-end tests. After this first round of tests is complete, they run a static application security scan, which looks for known vulnerabilities. Developers receive messages as each testing gate is passed. Once all tests are complete, the software artifact is stored in an artifact repository.

Implementation steps

1. Work with stakeholders in your organization to develop a testing standard for software artifacts. What standard tests should all artifacts pass? Are there compliance or governance requirements that must be included in the test coverage? Do you need to conduct code quality tests? When tests complete, who needs to know?
   a. The AWS Deployment Pipeline Reference Architecture contains an authoritative list of types of tests that can be conducted on software artifacts as part of an integration pipeline.
2. Instrument your application with the necessary tests based on your software testing standard. Each set of tests should complete in under ten minutes. Tests should run as part of an integration pipeline.
   a. Amazon CodeGuru Reviewer can test your application code for defects.
   b. You can use AWS CodeBuild to conduct tests on software artifacts.
   c. AWS CodePipeline can orchestrate your software tests into a pipeline.

Resources

Related best practices:

- OPS05-BP01 Use version control - All software artifacts must be backed by a version-controlled repository.

- OPS05-BP06 Share design standards - Your organizations software testing standards inform your design standards.
- OPS05-BP10 Fully automate integration and deployment - Software tests should be automatically run as part of your larger integration and deployment pipeline.

Related documents:

- Adopt a test-driven development approach
- Automated CloudFormation Testing Pipeline with TaskCat and CodePipeline
- Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST, and DAST tools
- Getting started with testing serverless applications
- My CI/CD pipeline is my release captain
- Practicing Continuous Integration and Continuous Delivery on AWS Whitepaper

Related videos:

- AWS re:Invent 2020: Testable infrastructure: Integration testing on AWS
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development
- Testing Your Infrastructure as Code with AWS CDK

Related resources:

- AWS Deployment Pipeline Reference Architecture - Application
- AWS Kubernetes DevSecOps Pipeline
- Policy as Code Workshop – Test Driven Development
- Run unit tests for a Node.js application from GitHub by using AWS CodeBuild
- Use Serverspec for test-driven development of infrastructure code

Related services:

- Amazon CodeGuru Reviewer
- AWS CodeBuild
- AWS CodePipeline

OPS05-BP03 Use configuration management systems

Use configuration management systems to make and track configuration changes. These systems reduce errors caused by manual processes and reduce the level of effort to deploy changes.

Static configuration management sets values when initializing a resource that are expected to remain consistent throughout the resource's lifetime. Some examples include setting the configuration for a web or application server on an instance, or defining the configuration of an AWS service within the AWS Management Console or through the AWS CLI.

Dynamic configuration management sets values at initialization that can or are expected to change during the lifetime of a resource. For example, you could set a feature toggle to activate functionality in your code through a configuration change, or change the level of log detail during an incident to capture more data and then change back following the incident eliminating the now unnecessary logs and their associated expense.

If you have dynamic configurations in your applications running on instances, containers, serverless functions, or devices, you can use AWS AppConfig to manage and deploy them across your environments.

On AWS, you can use AWS Config to continuously monitor your AWS resource configurations across accounts and Regions. It helps you to track their configuration history, understand how a configuration change would affect other resources, and audit them against expected or desired configurations using AWS Config Rules and AWS Config Conformance Packs.

On AWS, you can build continuous integration/continuous deployment (CI/CD) pipelines using services such as AWS Developer Tools (for example, AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeDeploy, and AWS CodeStar).

Have a change calendar and track when significant business or operational activities or events are planned that may be impacted by implementation of change. Adjust activities to manage risk around those plans. AWS Systems Manager Change Calendar provides a mechanism to document blocks of time as open or closed to changes and why, and share that information with other AWS accounts. AWS Systems Manager Automation scripts can be configured to adhere to the change calendar state.

AWS Systems Manager Maintenance Windows can be used to schedule the performance of AWS SSM Run Command or Automation scripts, AWS Lambda invocations, or AWS Step Functions activities at specified times. Mark these activities in your change calendar so that they can be included in your evaluation.

Common anti-patterns:

- You manually update the web server configuration across your fleet and a number of servers become unresponsive due to update errors.
- You manually update your application server fleet over the course of many hours. The inconsistency in configuration during the change causes unexpected behaviors.
- Someone has updated your security groups and your web servers are no longer accessible. Without knowledge of what was changed you spend significant time investigating the issue extending your time to recovery.

Benefits of establishing this best practice: Adopting configuration management systems reduces the level of effort to make and track changes, and the frequency of errors caused by manual procedures.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Use configuration management systems: Use configuration management systems to track and implement changes, to reduce errors caused by manual processes, and reduce the level of effort.
  - Infrastructure configuration management
  - AWS Config
  - What is AWS Config?
  - Introduction to AWS CloudFormation
  - What is AWS CloudFormation?
  - AWS OpsWorks
  - What is AWS OpsWorks?
  - Introduction to AWS Elastic Beanstalk
  - What is AWS Elastic Beanstalk?

Resources

Related documents:

- AWS AppConfig
- AWS Developer Tools

- AWS OpsWorks
- AWS Systems Manager Change Calendar
- AWS Systems Manager Maintenance Windows
- Infrastructure configuration management
- What is AWS CloudFormation?
- What is AWS Config?
- What is AWS Elastic Beanstalk?
- What is AWS OpsWorks?

Related videos:

- Introduction to AWS CloudFormation
- Introduction to AWS Elastic Beanstalk

OPS05-BP04 Use build and deployment management systems

Use build and deployment management systems. These systems reduce errors caused by manual processes and reduce the level of effort to deploy changes.

In AWS, you can build continuous integration/continuous deployment (CI/CD) pipelines using services such as AWS Developer Tools (for example, AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeDeploy, and AWS CodeStar).

Common anti-patterns:

- After compiling your code on your development system you, copy the executable onto your production systems and it fails to start. The local log files indicates that it has failed due to missing dependencies.
- You successfully build your application with new features in your development environment and provide the code to Quality Assurance (QA). It fails QA because it is missing static assets.
- On Friday, after much effort, you successfully built your application manually in your development environment including your newly coded features. On Monday, you are unable to repeat the steps that allowed you to successfully build your application.
- You perform the tests you have created for your new release. Then you spend the next week setting up a test environment and performing all the existing integration tests followed by

the performance tests. The new code has an unacceptable performance impact and must be redeveloped and then retested.

Benefits of establishing this best practice: By providing mechanisms to manage build and deployment activities you reduce the level of effort to perform repetitive tasks, free your team members to focus on their high value creative tasks, and limit the introduction of error from manual procedures.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Use build and deployment management systems: Use build and deployment management systems to track and implement change, to reduce errors caused by manual processes, and reduce the level of effort. Fully automate the integration and deployment pipeline from code check-in through build, testing, deployment, and validation. This reduces lead time, encourages increased frequency of change, and reduces the level of effort.
  - What is AWS CodeBuild?
  - Continuous integration best practices for software development
  - Slalom: CI/CD for serverless applications on AWS
  - Introduction to AWS CodeDeploy - automated software deployment with Amazon Web Services
  - What is AWS CodeDeploy?

Resources

Related documents:

- AWS Developer Tools
- What is AWS CodeBuild?
- What is AWS CodeDeploy?

Related videos:

- Continuous integration best practices for software development
- Introduction to AWS CodeDeploy - automated software deployment with Amazon Web Services

- Slalom: CI/CD for serverless applications on AWS

OPS05-BP05 Perform patch management

Perform patch management to gain features, address issues, and remain compliant with governance. Automate patch management to reduce errors caused by manual processes, and reduce the level of effort to patch.

Patch and vulnerability management are part of your benefit and risk management activities. It is preferable to have immutable infrastructures and deploy workloads in verified known good states. Where that is not viable, patching in place is the remaining option.

Updating machine images, container images, or Lambda custom runtimes and additional libraries to remove vulnerabilities are part of patch management. You should manage updates to Amazon Machine Images (AMIs) for Linux or Windows Server images using EC2 Image Builder. You can use Amazon Elastic Container Registry with your existing pipeline to manage Amazon ECS images and manage Amazon EKS images. AWS Lambda includes version management features.

Patching should not be performed on production systems without first testing in a safe environment. Patches should only be applied if they support an operational or business outcome. On AWS, you can use AWS Systems Manager Patch Manager to automate the process of patching managed systems and schedule the activity using AWS Systems Manager Maintenance Windows.

Common anti-patterns:

- You are given a mandate to apply all new security patches within two hours resulting in multiple outages due to application incompatibility with patches.
- An unpatched library results in unintended consequences as unknown parties use vulnerabilities within it to access your workload.
- You patch the developer environments automatically without notifying the developers. You receive multiple complaints from the developers that their environment cease to operate as expected.
- You have not patched the commercial off-the-self software on a persistent instance. When you have an issue with the software and contact the vendor, they notify you that version is not supported and you will have to patch to a specific level to receive any assistance.
- A recently released patch for the encryption software you used has significant performance improvements. Your unpatched system has performance issues that remain in place as a result of not patching.

Benefits of establishing this best practice: By establishing a patch management process, including your criteria for patching and methodology for distribution across your environments, you will be able to realize their benefits and control their impact. This will encourage the adoption of desired features and capabilities, the removal of issues, and sustained compliance with governance. Implement patch management systems and automation to reduce the level of effort to deploy patches and limit errors caused by manual processes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Patch management: Patch systems to remediate issues, to gain desired features or capabilities, and to remain compliant with governance policy and vendor support requirements. In immutable systems, deploy with the appropriate patch set to achieve the desired result. Automate the patch management mechanism to reduce the elapsed time to patch, to reduce errors caused by manual processes, and reduce the level of effort to patch.
  - AWS Systems Manager Patch Manager

Resources

Related documents:

- AWS Developer Tools
- AWS Systems Manager Patch Manager

Related videos:

- CI/CD for Serverless Applications on AWS
- Design with Ops in Mind

Related examples:

- Well-Architected Labs – Inventory and Patch Management

OPS05-BP06 Share design standards

Share best practices across teams to increase awareness and maximize the benefits of development efforts. Document them and keep them up to date as your architecture evolves. If shared standards

are enforced in your organization, it's critical that mechanisms exist to request additions, changes, and exceptions to standards. Without this option, standards become a constraint on innovation.

Desired outcome:

- Design standards are shared across teams in your organizations.
- They are documented and kept up to date as best practices evolve.

Common anti-patterns:

- Two development teams have each created a user authentication service. Your users must maintain a separate set of credentials for each part of the system they want to access.
- Each team manages their own infrastructure. A new compliance requirement forces a change to your infrastructure and each team implements it in a different way.

Benefits of establishing this best practice:

- Using shared standards supports the adoption of best practices and to maximizes the benefits of development efforts.
- Documenting and updating design standards keeps your organization up to date with best practices and security and compliance requirements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Share existing best practices, design standards, checklists, operating procedures, guidance, and governance requirements across teams. Have procedures to request changes, additions, and exceptions to design standards to support improvement and innovation. Make teams are aware of published content. Have a mechanism to keep design standards up to date as new best practices emerge.

Customer example

AnyCompany Retail has a cross-functional architecture team that creates software architecture patterns. This team builds the architecture with compliance and governance built in. Teams that adopt these shared standards get the benefits of having compliance and governance built in. They

can quickly build on top of the design standard. The architecture team meets quarterly to evaluate architecture patterns and update them if necessary.

Implementation steps

1. Identify a cross-functional team that will own developing and updating design standards. This team will work with stakeholders across your organization to develop design standards, operating procedures, checklists, guidance, and governance requirements. Document the design standards and share them within your organization.
   a. AWS Service Catalog can be used to create portfolios representing design standards using infrastructure as code. You can share portfolios across accounts.
2. Have a mechanism in place to keep design standards up to date as new best practices are identified.
3. If design standards are centrally enforced, have a process to request changes, updates, and exemptions.

Level of effort for the implementation plan: Medium. Developing a process to create and share design standards can take coordination and cooperation with stakeholders across your organization.

Resources

Related best practices:

- OPS01-BP03 Evaluate governance requirements - Governance requirements influence design standards.
- OPS01-BP04 Evaluate compliance requirements - Compliance is a vital input in creating design standards.
- OPS07-BP02 Ensure a consistent review of operational readiness - Operational readiness checklists are a mechanism to implement design standards when designing your workload.
- OPS11-BP01 Have a process for continuous improvement - Updating design standards is a part of continuous improvement.
- OPS11-BP04 Perform knowledge management - As part of your knowledge management practice, document and share design standards.

Related documents:

- Automate AWS Backups with AWS Service Catalog
- AWS Service Catalog Account Factory-Enhanced
- How Expedia Group built Database as a Service (DBaaS) offering using AWS Service Catalog
- Maintain visibility over the use of cloud architecture patterns
- Simplify sharing your AWS Service Catalog portfolios in an AWS Organizations setup

Related videos:

- AWS Service Catalog – Getting Started
- AWS re:Invent 2020: Manage your AWS Service Catalog portfolios like an expert

Related examples:

- AWS Service Catalog Reference Architecture
- AWS Service Catalog Workshop

Related services:

- AWS Service Catalog

OPS05-BP07 Implement practices to improve code quality

Implement practices to improve code quality and minimize defects. Some examples include test-driven development, code reviews, standards adoption, and pair programming. Incorporate these practices into your continuous integration and delivery process.

Desired outcome:

- Your organization uses best practices like code reviews or pair programming to improve code quality.
- Developers and operators adopt code quality best practices as part of the software development lifecycle.

Common anti-patterns:

- You commit code to the main branch of your application without a code review. The change automatically deploys to production and causes an outage.
- A new application is developed without any unit, end-to-end, or integration tests. There is no way to test the application before deployment.
- Your teams make manual changes in production to address defects. Changes do not go through testing or code reviews and are not captured or logged through continuous integration and delivery processes.

Benefits of establishing this best practice:

- By adopting practices to improve code quality, you can help minimize issues introduced to production.
- Code quality increases using best practices like pair programming and code reviews.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implement practices to improve code quality to minimize defects before they are deployed. Use practices like test-driven development, code reviews, and pair programming to increase the quality of your development.

Customer example

AnyCompany Retail adopts several practices to improve code quality. They have adopted test-driven development as the standard for writing applications. For some new features, they will have developers pair program together during a sprint. Every pull request goes through a code review by a senior developer before being integrated and deployed.

Implementation steps

1. Adopt code quality practices like test-driven development, code reviews, and pair programming into your continuous integration and delivery process. Use these techniques to improve software quality.
   a. Amazon CodeGuru Reviewer can provide programming recommendations for Java and Python code using machine learning.
   b. You can create shared development environments with AWS Cloud9 where you can collaborate on developing code.

Level of effort for the implementation plan: Medium. There are many ways of implementing this best practice, but getting organizational adoption may be challenging.

Resources

Related best practices:

- OPS05-BP06 Share design standards - You can share design standards as part of your code quality practice.

Related documents:

- Agile Software Guide
- My CI/CD pipeline is my release captain
- Automate code reviews with Amazon CodeGuru Reviewer
- Adopt a test-driven development approach
- How DevFactory builds better applications with Amazon CodeGuru
- On Pair Programming
- RENGA Inc. automates code reviews with Amazon CodeGuru
- The Art of Agile Development: Test-Driven Development
- Why code reviews matter (and actually save time!)

Related videos:

- AWS re:Invent 2020: Continuous improvement of code quality with Amazon CodeGuru
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development

Related services:

- Amazon CodeGuru Reviewer
- Amazon CodeGuru Profiler
- AWS Cloud9

OPS05-BP08 Use multiple environments

Use multiple environments to experiment, develop, and test your workload. Use increasing levels of controls as environments approach production to gain confidence your workload will operate as intended when deployed.

Common anti-patterns:

- You are performing development in a shared development environment and another developer overwrites your code changes.
- The restrictive security controls on your shared development environment are preventing you from experimenting with new services and features.
- You perform load testing on your production systems and cause an outage for your users.
- A critical error resulting in data loss has occurred in production. In your production environment, you attempt to recreate the conditions that lead to the data loss so that you can identify how it happened and prevent it from happening again. To prevent further data loss during testing, you are forced to make the application unavailable to your users.
- You are operating a multi-tenant service and are unable to support a customer request for a dedicated environment.
- You may not always test, but when you do it's in production.
- You believe that the simplicity of a single environment overrides the scope of impact of changes within the environment.

Benefits of establishing this best practice: By deploying multiple environments you can support multiple simultaneous development, testing, and production environments without creating conflicts between developers or user communities.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Use multiple environments: Provide developers sandbox environments with minimized controls to aid in experimentation. Provide individual development environments to help work in parallel, increasing development agility. Implement more rigorous controls in the environments approaching production to allow developers to innovate. Use infrastructure as code and configuration management systems to deploy environments that are configured consistent with the controls present in production to ensure systems operate as expected when deployed.

When environments are not in use, turn them off to avoid costs associated with idle resources (for example, development systems on evenings and weekends). Deploy production equivalent environments when load testing to improve valid results.

- What is AWS CloudFormation?
- How do I stop and start Amazon EC2 instances at regular intervals using AWS Lambda?

Resources

Related documents:

- How do I stop and start Amazon EC2 instances at regular intervals using AWS Lambda?
- What is AWS CloudFormation?

OPS05-BP09 Make frequent, small, reversible changes

Frequent, small, and reversible changes reduce the scope and impact of a change. This eases troubleshooting, helps with faster remediation, and provides the option to roll back a change.

Common anti-patterns:

- You deploy a new version of your application quarterly.
- You frequently make changes to your database schema.
- You perform manual in-place updates, overwriting existing installations and configurations.

Benefits of establishing this best practice: You recognize benefits from development efforts faster by deploying small changes frequently. When the changes are small, it is much easier to identify if they have unintended consequences. When the changes are reversible, there is less risk to implementing the change as recovery is simplified.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Make frequent, small, reversible changes: Frequent, small, and reversible changes reduce the scope and impact of a change. This eases troubleshooting, helps with faster remediation, and provides the option to roll back a change. It also increases the rate at which you can deliver value to the business.

OPS05-BP10 Fully automate integration and deployment

Automate build, deployment, and testing of the workload. This reduces errors caused by manual processes and reduces the effort to deploy changes.

Apply metadata using Resource Tags and AWS Resource Groups following a consistent tagging strategy to aid in identification of your resources. Tag your resources for organization, cost accounting, access controls, and targeting the run of automated operations activities.

Common anti-patterns:

- On Friday you, finish authoring the new code for your feature branch. On Monday, after running your code quality test scripts and each of your unit tests scripts, you will check in your code for the next scheduled release.
- You are assigned to code a fix for a critical issue impacting a large number of customers in production. After testing the fix, you commit your code and email change management to request approval to deploy it to production.

Benefits of establishing this best practice: By implementing automated build and deployment management systems, you reduce errors caused by manual processes and reduce the effort to deploy changes helping your team members to focus on delivering business value.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Use build and deployment management systems: Use build and deployment management systems to track and implement change, to reduce errors caused by manual processes, and reduce the level of effort. Fully automate the integration and deployment pipeline from code check-in through build, testing, deployment, and validation. This reduces lead time, encourages increased frequency of change, and reduces the level of effort.
  - What is AWS CodeBuild?
  - Continuous integration best practices for software development
  - Slalom: CI/CD for serverless applications on AWS
  - Introduction to AWS CodeDeploy - automated software deployment with Amazon Web Services
  - What is AWS CodeDeploy?

Resources

Related documents:

- What is AWS CodeBuild?
- What is AWS CodeDeploy?

Related videos:

- Continuous integration best practices for software development
- Introduction to AWS CodeDeploy - automated software deployment with Amazon Web Services
- Slalom: CI/CD for serverless applications on AWS

OPS 6. How do you mitigate deployment risks?

Adopt approaches that provide fast feedback on quality and achieve rapid recovery from changes that do not have desired outcomes. Using these practices mitigates the impact of issues introduced through the deployment of changes.

Best practices

- OPS06-BP01 Plan for unsuccessful changes
- OPS06-BP02 Test and validate changes
- OPS06-BP03 Use deployment management systems
- OPS06-BP04 Test using limited deployments
- OPS06-BP05 Deploy using parallel environments
- OPS06-BP06 Deploy frequent, small, reversible changes
- OPS06-BP07 Fully automate integration and deployment
- OPS06-BP08 Automate testing and rollback

OPS06-BP01 Plan for unsuccessful changes

Plan to revert to a known good state, or remediate in the production environment if a change does not have the desired outcome. This preparation reduces recovery time through faster responses.

Common anti-patterns:

- You performed a deployment and your application has become unstable but there appear to be active users on the system. You have to decide whether to roll back the change and impact the active users or wait to roll back the change knowing the users may be impacted regardless.
- After making a routine change, your new environments are accessible but one of your subnets has become unreachable. You have to decide whether to roll back everything or try to fix the inaccessible subnet. While you are making that determination, the subnet remains unreachable.

Benefits of establishing this best practice: Having a plan in place reduces the mean time to recover (MTTR) from unsuccessful changes, reducing the impact to your end users.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Plan for unsuccessful changes: Plan to revert to a known good state (that is, roll back the change), or remediate in the production environment (that is, roll forward the change) if a change does not have the desired outcome. When you identify changes that you cannot roll back if unsuccessful, apply due diligence prior to committing the change.

OPS06-BP02 Test and validate changes

Test changes and validate the results at all lifecycle stages to confirm new features and minimize the risk and impact of failed deployments.

On AWS, you can create temporary parallel environments to lower the risk, effort, and cost of experimentation and testing. Automate the deployment of these environments using AWS CloudFormation to ensure consistent implementations of your temporary environments.

Common anti-patterns:

- You deploy a cool new feature to your application. It doesn't work. You don't know.
- You update your certificates. You accidentally install the certificates to the wrong components. You don't know.

Benefits of establishing this best practice: By testing and validating changes following deployment you are able to identify issues early providing an opportunity to mitigate the impact on your customers.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Test and validate changes: Test changes and validate the results at all lifecycle stages (for example, development, test, and production), to confirm new features and minimize the risk and impact of failed deployments.
  - AWS Cloud9
  - What is AWS Cloud9?
  - How to test and debug AWS CodeDeploy locally before you ship your code

Resources

Related documents:

- AWS Cloud9
- AWS Developer Tools
- How to test and debug AWS CodeDeploy locally before you ship your code
- What is AWS Cloud9?

OPS06-BP03 Use deployment management systems

Use deployment management systems to track and implement change. This reduces errors caused by manual processes and reduces the effort to deploy changes.

In AWS, you can build Continuous Integration/Continuous Deployment (CI/CD) pipelines using services such as AWS Developer Tools (for example, AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeDeploy, and AWS CodeStar).

Common anti-patterns:

- You manually deploy updates to the application servers across your fleet and a number of servers become unresponsive due to update errors.
- You manually deploy to your application server fleet over the course of many hours. The inconsistency in versions during the change causes unexpected behaviors.

Benefits of establishing this best practice: Adopting deployment management systems reduces the level of effort to deploy changes, and the frequency of errors caused by manual procedures.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Use deployment management systems: Use deployment management systems to track and implement change. This will reduce errors caused by manual processes, and reduce the level of effort to deploy changes. Automate the integration and deployment pipeline from code check-in through testing, deployment, and validation. This reduces lead time, encourages increased frequency of change, and further reduces the level of effort.
  - Introduction to AWS CodeDeploy - automated software deployment with Amazon Web Services
  - What is AWS CodeDeploy?
  - What is AWS Elastic Beanstalk?
  - What is Amazon API Gateway?

Resources

Related documents:

- AWS CodeDeploy User Guide
- AWS Developer Tools
- Try a Sample Blue/Green Deployment in AWS CodeDeploy
- What is AWS CodeDeploy?
- What is AWS Elastic Beanstalk?
- What is Amazon API Gateway?

Related videos:

- Deep Dive on Advanced Continuous Delivery Techniques Using AWS
- Introduction to AWS CodeDeploy - automated software deployment with Amazon Web Services

OPS06-BP04 Test using limited deployments

Test with limited deployments alongside existing systems to confirm desired outcomes prior to full scale deployment. For example, use deployment canary testing or one-box deployments.

Common anti-patterns:

- You deploy an unsuccessful change to all of production all at once. You don't know.

Benefits of establishing this best practice: By testing and validating changes following limited deployment you are able to identify issues early with minimal impact on your customers providing an opportunity to further mitigate the impact on your customers.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Test using limited deployments: Test with limited deployments alongside existing systems to confirm desired outcomes prior to full scale deployment. For example, use deployment canary testing or one-box deployments.
  - AWS CodeDeploy User Guide
  - Blue/Green deployments with AWS Elastic Beanstalk
  - Set up an API Gateway canary release deployment
  - Try a Sample Blue/Green Deployment in AWS CodeDeploy
  - Working with deployment configurations in AWS CodeDeploy

Resources

Related documents:

- AWS CodeDeploy User Guide
- Blue/Green deployments with AWS Elastic Beanstalk
- Set up an API Gateway canary release deployment
- Try a Sample Blue/Green Deployment in AWS CodeDeploy
- Working with deployment configurations in AWS CodeDeploy

OPS06-BP05 Deploy using parallel environments

Implement changes onto parallel environments, and then transition over to the new environment. Maintain the prior environment until there is confirmation of successful deployment. Doing so minimizes recovery time by permitting rollback to the previous environment.

Common anti-patterns:

- You perform a mutable deployment by modifying your existing systems. After discovering that the change was unsuccessful, you are forced to modify the systems again to restore the old version extending your time to recovery.
- During a maintenance window, you decommission the old environment and then start building your new environment. Many hours into the procedure, you discover unrecoverable issues with the deployment. While extremely tired, you are forced to find the previous deployment procedures and start rebuilding the old environment.

Benefits of establishing this best practice: By using parallel environments, you can pre-deploy the new environment and transition over to them when desired. If the new environment is not successful, you can recover quickly by transitioning back to your original environment.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Deploy using parallel environments: Implement changes onto parallel environments, and transition or cut over to the new environment. Maintain the prior environment until there is confirmation of successful deployment. This minimizes recovery time by permitting rollback to the previous environment. For example, use immutable infrastructures with blue/green deployments.
  - Working with deployment configurations in AWS CodeDeploy
  - Blue/Green deployments with AWS Elastic Beanstalk
  - Set up an API Gateway canary release deployment
  - Try a Sample Blue/Green Deployment in AWS CodeDeploy

Resources

Related documents:

- AWS CodeDeploy User Guide
- Blue/Green deployments with AWS Elastic Beanstalk
- Set up an API Gateway canary release deployment
- Try a Sample Blue/Green Deployment in AWS CodeDeploy

- Working with deployment configurations in AWS CodeDeploy

Related videos:

- Deep Dive on Advanced Continuous Delivery Techniques Using AWS

OPS06-BP06 Deploy frequent, small, reversible changes

Use frequent, small, and reversible changes to reduce the scope of a change. This results in easier troubleshooting and faster remediation with the option to roll back a change.

Common anti-patterns:

- You deploy a new version of your application quarterly.
- You frequently make changes to your database schema.
- You perform manual in-place updates, overwriting existing installations and configurations.

Benefits of establishing this best practice: You recognize benefits from development efforts faster by deploying small changes frequently. When the changes are small it is much easier to identify if they have unintended consequences. When the changes are reversible there is less risk to implementing the change as recovery is simplified.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Deploy frequent, small, reversible changes: Use frequent, small, and reversible changes to reduce the scope of a change. This results in easier troubleshooting and faster remediation with the option to roll back a change.

OPS06-BP07 Fully automate integration and deployment

Automate build, deployment, and testing of the workload. This reduces errors cause by manual processes and reduces the effort to deploy changes.

Apply metadata using Resource Tags and AWS Resource Groups following a consistent tagging strategy to aid in identification of your resources. Tag your resources for organization, cost accounting, access controls, and targeting the run of automated operations activities.

Common anti-patterns:

- On Friday, you finish authoring the new code for your feature branch. On Monday, after running your code quality test scripts and each of your unit tests scripts, you will check in your code for the next scheduled release.
- You are assigned to code a fix for a critical issue impacting a large number of customers in production. After testing the fix, you commit your code and email change management to request approval to deploy it to production.

Benefits of establishing this best practice: By implementing automated build and deployment management systems you reduce errors caused by manual processes and reduce the effort to deploy changes helping your team members to focus on delivering business value.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Use build and deployment management systems: Use build and deployment management systems to track and implement change, to reduce errors caused by manual processes, and reduce the level of effort. Fully automate the integration and deployment pipeline from code check-in through build, testing, deployment, and validation. This reduces lead time, encourages increased frequency of change, and reduces the level of effort.
  - What is AWS CodeBuild?
  - Continuous integration best practices for software development
  - Slalom: CI/CD for serverless applications on AWS
  - Introduction to AWS CodeDeploy - automated software deployment with Amazon Web Services
  - What is AWS CodeDeploy?
  - Deep Dive on Advanced Continuous Delivery Techniques Using AWS

Resources

Related documents:

- Try a Sample Blue/Green Deployment in AWS CodeDeploy
- What is AWS CodeBuild?

- What is AWS CodeDeploy?

Related videos:

- Continuous integration best practices for software development
- Deep Dive on Advanced Continuous Delivery Techniques Using AWS
- Introduction to AWS CodeDeploy - automated software deployment with Amazon Web Services
- Slalom: CI/CD for serverless applications on AWS

OPS06-BP08 Automate testing and rollback

Automate testing of deployed environments to confirm desired outcomes. Automate rollback to a previous known good state when outcomes are not achieved to minimize recovery time and reduce errors caused by manual processes.

Common anti-patterns:

- You deploy changes to your workload. After your see that the change is complete, you start post deployment testing. After you see that they are complete, you realize that your workload is inoperable and customers are disconnected. You then begin rolling back to the previous version. After an extended time to detect the issue, the time to recover is extended by your manual redeployment.

Benefits of establishing this best practice: By testing and validating changes following deployment, you are able to identify issues immediately. By automatically rolling back to the previous version, the impact on your customers is minimized.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Automate testing and rollback: Automate testing of deployed environments to confirm desired outcomes. Automate rollback to a previous known good state when outcomes are not achieved to minimize recovery time and reduce errors caused by manual processes. For example, perform detailed synthetic user transactions following deployment, verify the results, and roll back on failure.
  - Redeploy and roll back a deployment with AWS CodeDeploy

Resources

Related documents:

- Redeploy and roll back a deployment with AWS CodeDeploy

OPS 7. How do you know that you are ready to support a workload?

Evaluate the operational readiness of your workload, processes and procedures, and personnel to understand the operational risks related to your workload.

Best practices

- OPS07-BP01 Ensure personnel capability
- OPS07-BP02 Ensure a consistent review of operational readiness
- OPS07-BP03 Use runbooks to perform procedures
- OPS07-BP04 Use playbooks to investigate issues
- OPS07-BP05 Make informed decisions to deploy systems and changes
- OPS07-BP06 Create support plans for production workloads

OPS07-BP01 Ensure personnel capability

Have a mechanism to validate that you have the appropriate number of trained personnel to support the workload. They must be trained on the platform and services that make up your workload. Provide them with the knowledge necessary to operate the workload. You must have enough trained personnel to support the normal operation of the workload and troubleshoot any incidents that occur. Have enough personnel so that you can rotate during on-call and vacations to avoid burnout.

Desired outcome:

- There are enough trained personnel to support the workload at times when the workload is available.
- You provide training for your personnel on the software and services that make up your workload.

Common anti-patterns:

- Deploying a workload without team members trained to operate the platform and services in use.
- Not having enough personnel to support on-call rotations or personnel taking time off.

Benefits of establishing this best practice:

- Having skilled team members helps effective support of your workload.
- With enough team members, you can support the workload and on-call rotations while decreasing the risk of burnout.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Validate that there are sufficient trained personnel to support the workload. Verify that you have enough team members to cover normal operational activities, including on-call rotations.

Customer example

AnyCompany Retail makes sure that teams supporting the workload are properly staffed and trained. They have enough engineers to support an on-call rotation. Personnel get training on the software and platform that the workload is built on and are encouraged to earn certifications. There are enough personnel so that people can take time off while still supporting the workload and the on-call rotation.

Implementation steps

1. Assign an adequate number of personnel to operate and support your workload, including on-call duties.
2. Train your personnel on the software and platforms that compose your workload.
   a. AWS Training and Certification has a library of courses about AWS. They provide free and paid courses, online and in-person.
   b. AWS hosts events and webinars where you learn from AWS experts.
3. Regularly evaluate team size and skills as operating conditions and the workload change. Adjust team size and skills to match operational requirements.

Level of effort for the implementation plan: High. Hiring and training a team to support a workload can take significant effort but has substantial long-term benefits.

Resources

Related best practices:

- OPS11-BP04 Perform knowledge management - Team members must have the information necessary to operate and support the workload. Knowledge management is the key to providing that.

Related documents:

- AWS Events and Webinars
- AWS Training and Certification

OPS07-BP02 Ensure a consistent review of operational readiness

Use Operational Readiness Reviews (ORRs) to validate that you can operate your workload. ORR is a mechanism developed at Amazon to validate that teams can safely operate their workloads. An ORR is a review and inspection process using a checklist of requirements. An ORR is a self-service experience that teams use to certify their workloads. ORRs include best practices from lessons learned from our years of building software.

An ORR checklist is composed of architectural recommendations, operational process, event management, and release quality. Our Correction of Error (CoE) process is a major driver of these items. Your own post-incident analysis should drive the evolution of your own ORR. An ORR is not only about following best practices but preventing the recurrence of events that you've seen before. Lastly, security, governance, and compliance requirements can also be included in an ORR.

Run ORRs before a workload launches to general availability and then throughout the software development lifecycle. Running the ORR before launch increases your ability to operate the workload safely. Periodically re-run your ORR on the workload to catch any drift from best practices. You can have ORR checklists for new services launches and ORRs for periodic reviews. This helps keep you up to date on new best practices that arise and incorporate lessons learned from post-incident analysis. As your use of the cloud matures, you can build ORR requirements into your architecture as defaults.

Desired outcome: You have an ORR checklist with best practices for your organization. ORRs are conducted before workloads launch. ORRs are run periodically over the course of the workload lifecycle.

Common anti-patterns:

- You launch a workload without knowing if you can operate it.
- Governance and security requirements are not included in certifying a workload for launch.
- Workloads are not re-evaluated periodically.
- Workloads launch without required procedures in place.
- You see repetition of the same root cause failures in multiple workloads.

Benefits of establishing this best practice:

- Your workloads include architecture, process, and management best practices.
- Lessons learned are incorporated into your ORR process.
- Required procedures are in place when workloads launch.
- ORRs are run throughout the software lifecycle of your workloads.

Level of risk if this best practice is not established: High

Implementation guidance

An ORR is two things: a process and a checklist. Your ORR process should be adopted by your organization and supported by an executive sponsor. At a minimum, ORRs must be conducted before a workload launches to general availability. Run the ORR throughout the software development lifecycle to keep it up to date with best practices or new requirements. The ORR checklist should include configuration items, security and governance requirements, and best practices from your organization. Over time, you can use services, such as AWS Config, AWS Security Hub CSPM, and AWS Control Tower Guardrails, to build best practices from the ORR into guardrails for automatic detection of best practices.

Customer example

After several production incidents, AnyCompany Retail decided to implement an ORR process. They built a checklist composed of best practices, governance and compliance requirements, and lessons

learned from outages. New workloads conduct ORRs before they launch. Every workload conducts a yearly ORR with a subset of best practices to incorporate new best practices and requirements that are added to the ORR checklist. Over time, AnyCompany Retail used AWS Config to detect some best practices, speeding up the ORR process.

Implementation steps

To learn more about ORRs, read the Operational Readiness Reviews (ORR) whitepaper. It provides detailed information on the history of the ORR process, how to build your own ORR practice, and how to develop your ORR checklist. The following steps are an abbreviated version of that document. For an in-depth understanding of what ORRs are and how to build your own, we recommend reading that whitepaper.

1. Gather the key stakeholders together, including representatives from security, operations, and development.
2. Have each stakeholder provide at least one requirement. For the first iteration, try to limit the number of items to thirty or less.
    - Appendix B: Example ORR questions from the Operational Readiness Reviews (ORR) whitepaper contains sample questions that you can use to get started.
3. Collect your requirements into a spreadsheet.
    - You can use custom lenses in the AWS Well-Architected Tool to develop your ORR and share them across your accounts and AWS Organization.
4. Identify one workload to conduct the ORR on. A pre-launch workload or an internal workload is ideal.
5. Run through the ORR checklist and take note of any discoveries made. Discoveries might not be ok if a mitigation is in place. For any discovery that lacks a mitigation, add those to your backlog of items and implement them before launch.
6. Continue to add best practices and requirements to your ORR checklist over time.

Support customers with Enterprise Support can request the Operational Readiness Review Workshop from their Technical Account Manager. The workshop is an interactive *working backwards* session to develop your own ORR checklist.

Level of effort for the implementation plan: High. Adopting an ORR practice in your organization requires executive sponsorship and stakeholder buy-in. Build and update the checklist with inputs from across your organization.

Resources

Related best practices:

- OPS01-BP03 Evaluate governance requirements – Governance requirements are a natural fit for an ORR checklist.
- OPS01-BP04 Evaluate compliance requirements – Compliance requirements are sometimes included in an ORR checklist. Other times they are a separate process.
- OPS03-BP07 Resource teams appropriately – Team capability is a good candidate for an ORR requirement.
- OPS06-BP01 Plan for unsuccessful changes – A rollback or rollforward plan must be established before you launch your workload.
- OPS07-BP01 Ensure personnel capability – To support a workload you must have the required personnel.
- SEC01-BP03 Identify and validate control objectives – Security control objectives make excellent ORR requirements.
- REL13-BP01 Define recovery objectives for downtime and data loss – Disaster recovery plans are a good ORR requirement.
- COST02-BP01 Develop policies based on your organization requirements – Cost management policies are good to include in your ORR checklist.

Related documents:

- AWS Control Tower - Guardrails in AWS Control Tower
- AWS Well-Architected Tool - Custom Lenses
- Operational Readiness Review Template by Adrian Hornsby
- Operational Readiness Reviews (ORR) Whitepaper

Related videos:

- AWS Supports You | Building an Effective Operational Readiness Review (ORR)

Related examples:

- Sample Operational Readiness Review (ORR) Lens

Related services:

- AWS Config
- AWS Control Tower
- AWS Security Hub CSPM
- AWS Well-Architected Tool

OPS07-BP03 Use runbooks to perform procedures

A *runbook* is a documented process to achieve a specific outcome. Runbooks consist of a series of steps that someone follows to get something done. Runbooks have been used in operations going back to the early days of aviation. In cloud operations, we use runbooks to reduce risk and achieve desired outcomes. At its simplest, a runbook is a checklist to complete a task.

Runbooks are an essential part of operating your workload. From onboarding a new team member to deploying a major release, runbooks are the codified processes that provide consistent outcomes no matter who uses them. Runbooks should be published in a central location and updated as the process evolves, as updating runbooks is a key component of a change management process. They should also include guidance on error handling, tools, permissions, exceptions, and escalations in case a problem occurs.

As your organization matures, begin automating runbooks. Start with runbooks that are short and frequently used. Use scripting languages to automate steps or make steps easier to perform. As you automate the first few runbooks, you'll dedicate time to automating more complex runbooks. Over time, most of your runbooks should be automated in some way.

Desired outcome: Your team has a collection of step-by-step guides for performing workload tasks. The runbooks contain the desired outcome, necessary tools and permissions, and instructions for error handling. They are stored in a central location and updated frequently.

Common anti-patterns:

- Relying on memory to complete each step of a process.
- Manually deploying changes without a checklist.
- Different team members performing the same process but with different steps or outcomes.
- Letting runbooks drift out of sync with system changes and automation.

Benefits of establishing this best practice:

- Reducing error rates for manual tasks.
- Operations are performed in a consistent manner.
- New team members can start performing tasks sooner.
- Runbooks can be automated to reduce toil.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Runbooks can take several forms depending on the maturity level of your organization. At a minimum, they should consist of a step-by-step text document. The desired outcome should be clearly indicated. Clearly document necessary special permissions or tools. Provide detailed guidance on error handling and escalations in case something goes wrong. List the runbook owner and publish it in a central location. Once your runbook is documented, validate it by having someone else on your team run it. As procedures evolve, update your runbooks in accordance with your change management process.

Your text runbooks should be automated as your organization matures. Using services like AWS Systems Manager automations, you can transform flat text into automations that can be run against your workload. These automations can be run in response to events, reducing the operational burden to maintain your workload.

Customer example

AnyCompany Retail must perform database schema updates during software deployments. The Cloud Operations Team worked with the Database Administration Team to build a runbook for manually deploying these changes. The runbook listed each step in the process in checklist form. It included a section on error handling in case something went wrong. They published the runbook on their internal wiki along with their other runbooks. The Cloud Operations Team plans to automate the runbook in a future sprint.

Implementation steps

If you don't have an existing document repository, a version control repository is a great place to start building your runbook library. You can build your runbooks using Markdown. We have provided an example runbook template that you can use to start building runbooks.

```
# Runbook Title
## Runbook Info
```

```
| Runbook ID | Description | Tools Used | Special Permissions | Runbook Author | Last
 Updated | Escalation POC |
|-------|-------|-------|-------|-------|-------|-------|
| RUN001 | What is this runbook for? What is the desired outcome? | Tools | Permissions
 | Your Name | 2022-09-21 | Escalation Name |
## Steps
1. Step one
2. Step two
```

1. If you don't have an existing documentation repository or wiki, create a new version control repository in your version control system.
2. Identify a process that does not have a runbook. An ideal process is one that is conducted semiregularly, short in number of steps, and has low impact failures.
3. In your document repository, create a new draft Markdown document using the template. Fill in `Runbook Title` and the required fields under `Runbook Info`.
4. Starting with the first step, fill in the `Steps` portion of the runbook.
5. Give the runbook to a team member. Have them use the runbook to validate the steps. If something is missing or needs clarity, update the runbook.
6. Publish the runbook to your internal documentation store. Once published, tell your team and other stakeholders.
7. Over time, you'll build a library of runbooks. As that library grows, start working to automate runbooks.

Level of effort for the implementation plan: Low. The minimum standard for a runbook is a step-by-step text guide. Automating runbooks can increase the implementation effort.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners: Runbooks should have an owner in charge of maintaining them.
- OPS07-BP04 Use playbooks to investigate issues: Runbooks and playbooks are like each other with one key difference: a runbook has a desired outcome. In many cases runbooks are initiated once a playbook has identified a root cause.
- OPS10-BP01 Use a process for event, incident, and problem management: Runbooks are a part of a good event, incident, and problem management practice.

- OPS10-BP02 Have a process per alert: Runbooks and playbooks should be used to respond to alerts. Over time these reactions should be automated.
- OPS11-BP04 Perform knowledge management: Maintaining runbooks is a key part of knowledge management.

Related documents:

- Achieving Operational Excellence using automated playbook and runbook
- AWS Systems Manager: Working with runbooks
- Migration playbook for AWS large migrations - Task 4: Improving your migration runbooks
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

Related videos:

- AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response (SEC318-R1)
- How to automate IT Operations on AWS | Amazon Web Services
- Integrate Scripts into AWS Systems Manager

Related examples:

- AWS Systems Manager: Automation walkthroughs
- AWS Systems Manager: Restore a root volume from the latest snapshot runbook
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Gitlab - Runbooks
- Rubix - A Python library for building runbooks in Jupyter Notebooks
- Using Document Builder to create a custom runbook
- Well-Architected Labs: Automating operations with Playbooks and Runbooks

Related services:

- AWS Systems Manager Automation

OPS07-BP04 Use playbooks to investigate issues

Playbooks are step-by-step guides used to investigate an incident. When incidents happen, playbooks are used to investigate, scope impact, and identify a root cause. Playbooks are used for a variety of scenarios, from failed deployments to security incidents. In many cases, playbooks identify the root cause that a runbook is used to mitigate. Playbooks are an essential component of your organization's incident response plans.

A good playbook has several key features. It guides the user, step by step, through the process of discovery. Thinking outside-in, what steps should someone follow to diagnose an incident? Clearly define in the playbook if special tools or elevated permissions are needed in the playbook. Having a communication plan to update stakeholders on the status of the investigation is a key component. In situations where a root cause can't be identified, the playbook should have an escalation plan. If the root cause is identified, the playbook should point to a runbook that describes how to resolve it. Playbooks should be stored centrally and regularly maintained. If playbooks are used for specific alerts, provide your team with pointers to the playbook within the alert.

As your organization matures, automate your playbooks. Start with playbooks that cover low-risk incidents. Use scripting to automate the discovery steps. Make sure that you have companion runbooks to mitigate common root causes.

Desired outcome: Your organization has playbooks for common incidents. The playbooks are stored in a central location and available to your team members. Playbooks are updated frequently. For any known root causes, companion runbooks are built.

Common anti-patterns:

- There is no standard way to investigate an incident.
- Team members rely on muscle memory or institutional knowledge to troubleshoot a failed deployment.
- New team members learn how to investigate issues through trial and error.
- Best practices for investigating issues are not shared across teams.

Benefits of establishing this best practice:

- Playbooks boost your efforts to mitigate incidents.
- Different team members can use the same playbook to identify a root cause in a consistent manner.

- Known root causes can have runbooks developed for them, speeding up recovery time.
- Playbooks help team members to start contributing sooner.
- Teams can scale their processes with repeatable playbooks.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

How you build and use playbooks depends on the maturity of your organization. If you are new to the cloud, build playbooks in text form in a central document repository. As your organization matures, playbooks can become semi-automated with scripting languages like Python. These scripts can be run inside a Jupyter notebook to speed up discovery. Advanced organizations have fully automated playbooks for common issues that are auto-remediated with runbooks.

Start building your playbooks by listing common incidents that happen to your workload. Choose playbooks for incidents that are low risk and where the root cause has been narrowed down to a few issues to start. After you have playbooks for simpler scenarios, move on to the higher risk scenarios or scenarios where the root cause is not well known.

Your text playbooks should be automated as your organization matures. Using services like AWS Systems Manager Automations, flat text can be transformed into automations. These automations can be run against your workload to speed up investigations. These automations can be activated in response to events, reducing the mean time to discover and resolve incidents.

Customers can use AWS Systems Manager Incident Manager to respond to incidents. This service provides a single interface to triage incidents, inform stakeholders during discovery and mitigation, and collaborate throughout the incident. It uses AWS Systems Manager Automations to speed up detection and recovery.

Customer example

A production incident impacted AnyCompany Retail. The on-call engineer used a playbook to investigate the issue. As they progressed through the steps, they kept the key stakeholders, identified in the playbook, up to date. The engineer identified the root cause as a race condition in a backend service. Using a runbook, the engineer relaunched the service, bringing AnyCompany Retail back online.

Implementation steps

If you don't have an existing document repository, we suggest creating a version control repository for your playbook library. You can build your playbooks using Markdown, which is compatible with most playbook automation systems. If you are starting from scratch, use the following example playbook template.

```
# Playbook Title
## Playbook Info
| Playbook ID | Description | Tools Used | Special Permissions | Playbook Author | Last
 Updated | Escalation POC | Stakeholders | Communication Plan |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| RUN001 | What is this playbook for? What incident is it used for? | Tools |
 Permissions | Your Name | 2022-09-21 | Escalation Name | Stakeholder Name | How will
 updates be communicated during the investigation? |
## Steps
1. Step one
2. Step two
```

1. If you don't have an existing document repository or wiki, create a new version control repository for your playbooks in your version control system.
2. Identify a common issue that requires investigation. This should be a scenario where the root cause is limited to a few issues and resolution is low risk.
3. Using the Markdown template, fill in the `Playbook Name` section and the fields under `Playbook Info`.
4. Fill in the troubleshooting steps. Be as clear as possible on what actions to perform or what areas you should investigate.
5. Give a team member the playbook and have them go through it to validate it. If there's anything missing or something isn't clear, update the playbook.
6. Publish your playbook in your document repository and inform your team and any stakeholders.
7. This playbook library will grow as you add more playbooks. Once you have several playbooks, start automating them using tools like AWS Systems Manager Automations to keep automation and playbooks in sync.

Level of effort for the implementation plan: Low. Your playbooks should be text documents stored in a central location. More mature organizations will move towards automating playbooks.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners: Playbooks should have an owner in charge of maintaining them.
- OPS07-BP03 Use runbooks to perform procedures: Runbooks and playbooks are similar, but with one key difference: a runbook has a desired outcome. In many cases, runbooks are used once a playbook has identified a root cause.
- OPS10-BP01 Use a process for event, incident, and problem management: Playbooks are a part of good event, incident, and problem management practice.
- OPS10-BP02 Have a process per alert: Runbooks and playbooks should be used to respond to alerts. Over time, these reactions should be automated.
- OPS11-BP04 Perform knowledge management: Maintaining playbooks is a key part of knowledge management.

Related documents:

- Achieving Operational Excellence using automated playbook and runbook
- AWS Systems Manager: Working with runbooks
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

Related videos:

- AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response (SEC318-R1)
- AWS Systems Manager Incident Manager - AWS Virtual Workshops
- Integrate Scripts into AWS Systems Manager

Related examples:

- AWS Customer Playbook Framework
- AWS Systems Manager: Automation walkthroughs
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Rubix – A Python library for building runbooks in Jupyter Notebooks

- Using Document Builder to create a custom runbook
- Well-Architected Labs: Automating operations with Playbooks and Runbooks
- Well-Architected Labs: Incident response playbook with Jupyter

Related services:

- AWS Systems Manager Automation
- AWS Systems Manager Incident Manager

OPS07-BP05 Make informed decisions to deploy systems and changes

Have processes in place for successful and unsuccessful changes to your workload. A pre-mortem is an exercise where a team simulates a failure to develop mitigation strategies. Use pre-mortems to anticipate failure and create procedures where appropriate. Evaluate the benefits and risks of deploying changes to your workload. Verify that all changes comply with governance.

Desired outcome:

- You make informed decisions when deploying changes to your workload.
- Changes comply with governance.

Common anti-patterns:

- Deploying a change to our workload without a process to handle a failed deployment.
- Making changes to your production environment that are out of compliance with governance requirements.
- Deploying a new version of your workload without establishing a baseline for resource utilization.

Benefits of establishing this best practice:

- You are prepared for unsuccessful changes to your workload.
- Changes to your workload are compliant with governance policies.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use pre-mortems to develop processes for unsuccessful changes. Document your processes for unsuccessful changes. Ensure that all changes comply with governance. Evaluate the benefits and risks to deploying changes to your workload.

Customer example

AnyCompany Retail regularly conducts pre-mortems to validate their processes for unsuccessful changes. They document their processes in a shared Wiki and update it frequently. All changes comply with governance requirements.

Implementation steps

1. Make informed decisions when deploying changes to your workload. Establish and review criteria for a successful deployment. Develop scenarios or criteria that would initiate a rollback of a change. Weigh the benefits of deploying changes against the risks of an unsuccessful change.
2. Verify that all changes comply with governance policies.
3. Use pre-mortems to plan for unsuccessful changes and document mitigation strategies. Run a table-top exercise to model an unsuccessful change and validate roll-back procedures.

Level of effort for the implementation plan: Moderate. Implementing a practice of pre-mortems requires coordination and effort from stakeholders across your organization

Resources

Related best practices:

- OPS01-BP03 Evaluate governance requirements - Governance requirements are a key factor in determining whether to deploy a change.
- OPS06-BP01 Plan for unsuccessful changes - Establish plans to mitigate a failed deployment and use pre-mortems to validate them.
- OPS06-BP02 Test and validate changes - Every software change should be properly tested before deployment in order to reduce defects in production.
- OPS07-BP01 Ensure personnel capability - Having enough trained personnel to support the workload is essential to making an informed decision to deploy a system change.

Related documents:

- Amazon Web Services: Risk and Compliance
- AWS Shared Responsibility Model
- Governance in the AWS Cloud: The Right Balance Between Agility and Safety

OPS07-BP06 Create support plans for production workloads

Enable support for any software and services that your production workload relies on. Select an appropriate support level to meet your production service-level needs. Support plans for these dependencies are necessary in case there is a service disruption or software issue. Document support plans and how to request support for all service and software vendors. Implement mechanisms that verify that support points of contacts are kept up to date.

Desired outcome:

- Implement support plans for software and services that production workloads rely on.
- Choose an appropriate support plan based on service-level needs.
- Document the support plans, support levels, and how to request support.

Common anti-patterns:

- You have no support plan for a critical software vendor. Your workload is impacted by them and you can do nothing to expedite a fix or get timely updates from the vendor.
- A developer that was the primary point of contact for a software vendor left the company. You are not able to reach the vendor support directly. You must spend time researching and navigating generic contact systems, increasing the time required to respond when needed.
- A production outage occurs with a software vendor. There is no documentation on how to file a support case.

Benefits of establishing this best practice:

- With the appropriate support level, you are able to get a response in the time frame necessary to meet service-level needs.
- As a supported customer you can escalate if there are production issues.
- Software and services vendors can assist in troubleshooting during an incident.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Enable support plans for any software and services vendors that your production workload relies on. Set up appropriate support plans to meet service-level needs. For AWS customers, this means activating AWS Business Support or greater on any accounts where you have production workloads. Meet with support vendors on a regular cadence to get updates about support offerings, processes, and contacts. Document how to request support from software and services vendors, including how to escalate if there is an outage. Implement mechanisms to keep support contacts up to date.

Customer example

At AnyCompany Retail, all commercial software and services dependencies have support plans. For example, they have AWS Enterprise Support activated on all accounts with production workloads. Any developer can raise a support case when there is an issue. There is a wiki page with information on how to request support, whom to notify, and best practices for expediting a case.

Implementation steps

1. Work with stakeholders in your organization to identify software and services vendors that your workload relies on. Document these dependencies.
2. Determine service-level needs for your workload. Select a support plan that aligns with them.
3. For commercial software and services, establish a support plan with the vendors.
    a. Subscribing to AWS Business Support or greater for all production accounts provides faster response time from AWS Support and strongly recommended. If you don't have premium support, you must have an action plan to handle issues, which require help from AWS Support. AWS Support provides a mix of tools and technology, people, and programs designed to proactively help you optimize performance, lower costs, and innovate faster. AWS Business Support provides additional benefits, including access to AWS Trusted Advisor and AWS Personal Health Dashboard and faster response times.
4. Document the support plan in your knowledge management tool. Include how to request support, who to notify if a support case is filed, and how to escalate during an incident. A wiki is a good mechanism to allow anyone to make necessary updates to documentation when they become aware of changes to support processes or contacts.

Level of effort for the implementation plan: Low. Most software and services vendors offer opt-in support plans. Documenting and sharing support best practices on your knowledge management system verifies that your team knows what to do when there is a production issue.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners

Related documents:

- AWS Support Plans

Related services:

- AWS Business Support
- AWS Enterprise Support

Operate

Questions

- OPS 8. How do you understand the health of your workload?
- OPS 9. How do you understand the health of your operations?
- OPS 10. How do you manage workload and operations events?

OPS 8. How do you understand the health of your workload?

Define, capture, and analyze workload metrics to gain visibility to workload events so that you can take appropriate action.

Best practices

- OPS08-BP01 Identify key performance indicators
- OPS08-BP02 Define workload metrics
- OPS08-BP03 Collect and analyze workload metrics
- OPS08-BP04 Establish workload metrics baselines

- OPS08-BP05 Learn expected patterns of activity for workload
- OPS08-BP06 Alert when workload outcomes are at risk
- OPS08-BP07 Alert when workload anomalies are detected
- OPS08-BP08 Validate the achievement of outcomes and the effectiveness of KPIs and metrics

OPS08-BP01 Identify key performance indicators

Identify key performance indicators (KPIs) based on desired business outcomes (for example, order rate, customer retention rate, and profit versus operating expense) and customer outcomes (for example, customer satisfaction). Evaluate KPIs to determine workload success.

Common anti-patterns:

- You are asked by business leadership how successful a workload has been serving business needs but have no frame of reference to determine success.
- You are unable to determine if the commercial off-the-shelf application you operate for your organization is cost-effective.

Benefits of establishing this best practice: By identifying key performance indicators you help achieve business outcomes as the test of the health and success of your workload.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Identify key performance indicators: Identify key performance indicators (KPIs) based on desired business and customer outcomes. Evaluate KPIs to determine workload success.

OPS08-BP02 Define workload metrics

Define metrics that measure the health of the workload. Workload health is measured by the achievement of business outcomes (KPIs) and the state of workload components and applications. Examples of KPIs are abandoned shopping carts, orders placed, cost, price, and allocated workload expense. While you may collect telemetry from multiple components, select a subset that provides insight into the overall workload health. Adjust workload metrics over time as business needs change.

Desired outcome:

- You have identified metrics that validate the achievement of KPIs that reflect business outcomes.
- You have metrics that show a consistent view of workload health.
- Workload metrics are evaluated periodically as business needs change.

Common anti-patterns:

- You are monitoring all the applications in your workload but are unable to determine if your workload is achieving business outcomes.
- You have defined workload metrics but they are not associated to any business KPIs.

Benefits of establishing this best practice:

- You can measure your workload against the achievement of business outcomes.
- You know if your workload is in a healthy state or needs intervention.

Level of risk exposed if this best practice is not established: High

Implementation guidance

The goal of this best practice is that you can answer the following question: is my workload healthy? Workload health is determined by the achievement of business outcomes and the state of applications and components in the workload. Work backwards from business KPIs to identify metrics. Identify key metrics from components and applications. Periodically review workload metrics as business needs change.

Customer example

Workload health is determined at AnyCompany Retail by a collection of application and component metrics. Starting with business KPIs, they identify metrics like order rate that can show they are achieving business outcomes. They also include key application metrics like page response and component metrics like open database connections. On a quarterly basis, they re-evaluate workload metrics to make sure they are still valid in determining workload health.

Implementation steps

1. Starting with business KPIs, identify metrics that show you are achieving business outcomes. If there are KPIs that do not have metrics, instrument your workload with additional metrics for any missing business KPIs.

   a. You can publish custom metrics from your applications to Amazon CloudWatch.
   b. The AWS Distro for OpenTelemetry can collect metrics from existing applications and be used to add new metrics.
   c. Customers with Enterprise Support can request the Building a Monitoring Strategy Workshop from their Technical Account Manager. This workshop will help you build an observability strategy for your workload.
2. Identify metrics for applications and components in the workload. What are key metrics that show the health of individual components and applications? Applications and components may emit many different metrics, but choose one to three key metrics that show their overall health.
3. Implement a mechanism to evaluate workload metrics periodically. When business KPIs change, work with stakeholders to update workload metrics. As your workload components and applications evolve, adjust your workload metrics.

Level of effort for the implementation plan: Medium. Adding metrics for business KPIs to applications may require moderate effort.

Resources

Related best practices:

- OPS04-BP01 Implement application telemetry - Your application must emit telemetry that supports business outcomes.
- OPS04-BP02 Implement and configure workload telemetry - You must instrument your workload to emit telemetry before you can define workload metrics that support business outcomes.
- OPS08-BP01 Identify key performance indicators - You must identify key performance indicators first before selecting workload metrics.

Related documents:

- Adding metrics and traces to your application on Amazon EKS with AWS Distro for OpenTelemetry, AWS X-Ray, and Amazon CloudWatch
- Instrumenting distributed systems for operational visibility
- Implementing health checks
- How to Monitor your Applications Effectively

- How to better monitor your custom application metrics using Amazon CloudWatch Agent

Related videos:

- AWS re:Invent 2020: Monitoring production services at Amazon
- AWS re:Invent 2022 - Building observable applications with OpenTelemetry (BOA310)
- How to Easily Setup Application Monitoring for Your AWS Workloads - AWS Online Tech Talks
- Mastering Observability of Your Serverless Applications - AWS Online Tech Talks

Related examples:

- One Observability Workshop

Related services:

- Amazon CloudWatch
- AWS Distro for OpenTelemetry

OPS08-BP03 Collect and analyze workload metrics

Perform regular, proactive reviews of workload metrics to identify trends and determine if a response is necessary and validate the achievement of business outcomes. Aggregate metrics from your workload applications and components to a central location. Use dashboards and analytics tools to analyze telemetry and determine workload health. Implement a mechanism to conduct workload health reviews on periodic basis with stakeholders in your organization.

Desired outcome:

- Workload metrics are collected in a central location.
- Dashboards and analytics tools are used to analyze workload health trends.
- You conduct periodic workload metric reviews with your organization.

Common anti-patterns:

- Your organization collects metrics from the workload in two different observability platforms. You are unable to determine workload health because the platforms are incompatible.

- Error rates for a component of your workload are slowly increasing. You fail to notice this trend because your organization does not conduct periodic workload metric reviews. The component fails after a week, impairing your workload.

Benefits of establishing this best practice:

- You have increased awareness of workload health and the achievement of business outcomes.
- Workload health trends can be developed over time.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Collect workload metrics in a central location. Using dashboards and analytics tools, analyze workload metrics to gain insight into workload health, develop workload health trends, and validate the achievement of business outcomes. Implement a mechanism to conduct periodic reviews of workload metrics.

Customer example

AnyCompany Retail conducts workload metric reviews every week on Wednesday. They gather stakeholders from across the company and go through the previous week's metrics. During the meeting, they highlight trends and insights gleaned from analytics tools. Internal dashboards are published with key workload metrics that any employee can view and search.

Implementation steps

1. Identify the workload metrics that are tied to workload health. Starting with business KPIs, identify the metrics for applications, components, and platforms that provide an overall view of workload health.
   a. You can publish custom metrics to Amazon CloudWatch. You can leverage the Amazon CloudWatch agent to collect metrics and logs from Amazon EC2 instances and on-premises servers.
   b. The AWS Distro for OpenTelemetry can collect metrics from existing applications and be used to add new metrics.
   c. Customers with Enterprise Support can request the Building a Monitoring Strategy Workshop from their Technical Account Manager. This workshop helps you build an observability strategy for your workload.

2. Collect workload metrics in a central platform. If workload metrics are split between different platform, this can make it difficult to analyze and develop trends. The platform should have dashboards and analytic capabilities.
    a. Amazon CloudWatch can collect and store workload metrics. In multi-account topologies, it is recommended to have a central logging and monitoring account, referred to as a *log archive account*.
3. Build a consolidated dashboard of workload metrics. Use this view for metrics reviews and analysis of trends.
    a. You can create custom CloudWatch dashboards to collect your workload metrics in a consolidated view.
4. Implement a workload metric review process. On a weekly, bi-weekly, or monthly basis, review your workload metrics with stakeholders, including technical and non-technical personnel. Use these review sessions to identify trends and gain insight into workload health.

Level of effort for the implementation plan: High. If workload metrics are not centrally collected, it could require significant investment to consolidate them in one platform.

Resources

Related best practices:

- OPS08-BP01 Identify key performance indicators - You must identify key performance indicators first before selecting workload metrics.
- OPS08-BP02 Define workload metrics - You must define workload metrics before collecting and analyzing them.

Related documents:

- Power operational insights with Amazon Quick
- Using Amazon CloudWatch dashboards custom widgets

Related videos:

- Create Cross Account & Cross Region CloudWatch Dashboards
- Monitor AWS Resources Using Amazon CloudWatch Dashboards

Related examples:

- AWS Management and Governance Tools Workshop - CloudWatch Dashboards
- Well-Architected Labs - Level 100: Monitoring with CloudWatch Dashboards

Related services:

- Amazon CloudWatch
- AWS Distro for OpenTelemetry

OPS08-BP04 Establish workload metrics baselines

Establishing a baseline for workload metrics aids in understanding workload health and performance. Using baselines, you can identify under- and over-performing applications and components. A workload baseline adds to your ability to mitigate issues before they become incidents. Baselines are foundational in developing patterns of activity and implementing anomaly detection when metrics deviate from expected values.

Desired outcome:

- You have a baseline level of metrics for your workload under normal conditions.
- You can determine if your workload is functioning normally.

Common anti-patterns:

- After deploying a new feature, there is drop in request latency. A baseline was not established for a composite metric of incoming processed requests and overall latency. You are unable to determine if the change caused an improvement or caused a defect.
- A sudden spike in user activity occurs, but you have not established a metric baseline. The activity spike slowly leads to a memory leak in an application. Eventually this takes your workload offline.

Benefits of establishing this best practice:

- You understand the normal pattern of activity for your workload using metrics for key components and applications.

- You can determine if your workload, its applications, and components, are behaving normally or may require intervention.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Use historical data to establish a baseline of workload metrics for applications and components in your workload. Leverage the metric baseline in metric review meetings and troubleshooting. Periodically review workload performance and adjust the baseline as the architecture evolves.

Customer example

Baselines are established for all components and applications at AnyCompany Retail. Using historical data, AnyCompany Retail developed their workload metric baselines over a two-month metric window. Every two months they re-assess baselines and adjust them based on real-world data.

Implementation steps

1. Working backwards from your workload metrics, establish a metric baseline for key components and applications using historical data. Limit the number of metrics per component or application and avoid monitor fatigue.
   a. You can use Amazon CloudWatch Metrics Insights to query metrics at scale and identify trends and patterns.
   b. Amazon CloudWatch anomaly detection uses machine learning algorithms to identify patterns of behavior for metrics, determine baselines, and surfaces anomalies.
   c. Amazon DevOps Guru provides the ability to detect operational issues with your workload using machine learning.
   d. Customers with Enterprise Support can request the Building a Monitoring Strategy Workshop from their Technical Account Manager. This workshop will help you build an observability strategy for your workload.
2. Put in place a mechanism to periodically review workload metric baselines, especially before significant business events. At least once a quarter, evaluate your workload metric baseline using historical data. Use the baseline in your metric review meetings.

Level of effort for the implementation plan: Low. Having established workload metrics, establishing baselines may require you to collect enough data to identify normal patterns of behavior.

Resources

Related best practices:

- OPS08-BP02 Define workload metrics - Workload metrics must be established first before determining baselines.
- OPS08-BP03 Collect and analyze workload metrics - Collecting and analyzing workload metrics is necessary to have in place before establishing metric baselines.
- OPS08-BP05 Learn expected patterns of activity for workload - This best practice builds on top of the baseline to develop usage trends.
- OPS08-BP06 Alert when workload outcomes are at risk - Metric baselines are necessary to identifying thresholds and developing alerts.
- OPS08-BP07 Alert when workload anomalies are detected - Anomaly detection requires the establishment of metric baselines.

Related documents:

- AWS Observability Best Practices - Alarms
- How to Monitor your Applications Effectively
- How to set up CloudWatch Anomaly Detection to set dynamic alarms, automate actions, and drive online sales
- Operationalizing CloudWatch Anomaly Detection

Related videos:

- AWS re:Invent 2020: Monitoring production services at Amazon
- AWS re:Invent 2021- Get insights from operational metrics at scale with CloudWatch Metrics Insights
- AWS re:Invent 2022 - Developing an observability strategy (COP302)
- AWS Summit DC 2022 - Monitoring and observability for modern applications
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS (COP310)

Related examples:

- AWS CloudTrail and Amazon CloudWatch Integration Workshop

Related services:

- Amazon CloudWatch
- Amazon DevOps Guru

OPS08-BP05 Learn expected patterns of activity for workload

Establish patterns of workload activity to identify anomalous behavior so that you can respond appropriately if required.

CloudWatch through the CloudWatch Anomaly Detection feature applies statistical and machine learning algorithms to generate a range of expected values that represent normal metric behavior.

Amazon DevOps Guru can be used to identify anomalous behavior through event correlation, log analysis, and applying machine learning to analyze your workload telemetry. When unexpected behaviors are detected, it provides the related metrics and events with recommendations to address the behavior.

Common anti-patterns:

- You are reviewing network utilization logs and see that network utilization increased between 11:30am and 1:30pm and then again at 4:30pm through 6:00pm. You are unaware if this should be considered normal or not.
- Your web servers reboot every night at 3:00am. You are unaware if this is an expected behavior.

Benefits of establishing this best practice: By learning patterns of behavior you can recognize unexpected behavior and take action if necessary.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Learn expected patterns of activity for workload: Establish patterns of workload activity to determine when behavior is outside of the expected values so that you can respond appropriately if required.

Resources

Related documents:

- Amazon DevOps Guru
- CloudWatch Anomaly Detection

OPS08-BP06 Alert when workload outcomes are at risk

Raise an alert when workload outcomes are at risk so that you can respond appropriately if necessary.

Ideally, you have previously identified a metric threshold that you are able to alarm upon or an event that you can use to initiate an automated response.

On AWS, you can use Amazon CloudWatch Synthetics to create canary scripts to monitor your endpoints and APIs by performing the same actions as your customers. The telemetry generated and the insight gained can help you to identify issues before your customers are impacted.

You can also use CloudWatch Logs Insights to interactively search and analyze your log data using a purpose-built query language. CloudWatch Logs Insights automatically discovers fields in logs from AWS services, and custom log events in JSON. It scales with your log volume and query complexity and gives you answers in seconds, helping you to search for the contributing factors of an incident.

Common anti-patterns:

- You have no network connectivity. No one is aware. No one is trying to identify why or taking action to restore connectivity.
- Following a patch, your persistent instances have become unavailable, disrupting users. Your users have opened support cases. No one has been notified. No one is taking action.

Benefits of establishing this best practice: By identifying that business outcomes are at risk and alerting for action to be taken you have the opportunity to prevent or mitigate the impact of an incident.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Alert when workload outcomes are at risk: Raise an alert when workload outcomes are at risk so that you can respond appropriately if required.
  - What is Amazon CloudWatch Events?
  - Creating Amazon CloudWatch Alarms
  - Invoking Lambda functions using Amazon SNS notifications

Resources

Related documents:

- Amazon CloudWatch Synthetics
- CloudWatch Logs Insights
- Creating Amazon CloudWatch Alarms
- Invoking Lambda functions using Amazon SNS notifications
- What is Amazon CloudWatch Events?

OPS08-BP07 Alert when workload anomalies are detected

Raise an alert when workload anomalies are detected so that you can respond appropriately if necessary.

Your analysis of your workload metrics over time may establish patterns of behavior that you can quantify sufficiently to define an event or raise an alarm in response.

Once trained, the CloudWatch Anomaly Detection feature can be used to alarm on detected anomalies or can provide overlaid expected values onto a graph of metric data for ongoing comparison.

Common anti-patterns:

- Your retail website sales have increased suddenly and dramatically. No one is aware. No one is trying to identify what led to this surge. No one is taking action to ensure quality customer experiences under the additional load.
- Following the application of a patch, your persistent servers are rebooting frequently, disrupting users. Your servers typically reboot up to three times but not more. No one is aware. No one is trying to identify why this is happening.

Benefits of establishing this best practice: By understanding patterns of workload behavior, you can identify unexpected behavior and take action if necessary.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Alert when workload anomalies are detected: Raise an alert when workload anomalies are detected so that you can respond appropriately if required.
  - What is Amazon CloudWatch Events?
  - Creating Amazon CloudWatch Alarms
  - Invoking Lambda functions using Amazon SNS notifications

Resources

Related documents:

- Creating Amazon CloudWatch Alarms
- CloudWatch Anomaly Detection
- Invoking Lambda functions using Amazon SNS notifications
- What is Amazon CloudWatch Events?

OPS08-BP08 Validate the achievement of outcomes and the effectiveness of KPIs and metrics

Create a business-level view of your workload operations to help you determine if you are satisfying needs and to identify areas that need improvement to reach business goals. Validate the effectiveness of KPIs and metrics and revise them if necessary.

AWS also has support for third-party log analysis systems and business intelligence tools through the AWS service APIs and SDKs (for example, Grafana, Kibana, and Logstash).

Common anti-patterns:

- Page response time has never been considered a contributor to customer satisfaction. You have never established a metric or threshold for page response time. Your customers are complaining about slowness.

- You have not been achieving your minimum response time goals. In an effort to improve response time, you have scaled up your application servers. You are now exceeding response time goals by a significant margin and also have significant unused capacity you are paying for.

Benefits of establishing this best practice: By reviewing and revising KPIs and metrics, you understand how your workload supports the achievement of your business outcomes and can identify where improvement is needed to reach business goals.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Validate the achievement of outcomes and the effectiveness of KPIs and metrics: Create a business level view of your workload operations to help you determine if you are satisfying needs and to identify areas that need improvement to reach business goals. Validate the effectiveness of KPIs and metrics and revise them if necessary.
  - Using Amazon CloudWatch dashboards
  - What is log analytics?

Resources

Related documents:

- Using Amazon CloudWatch dashboards
- What is log analytics?

OPS 9. How do you understand the health of your operations?

Define, capture, and analyze operations metrics to gain visibility to operations events so that you can take appropriate action.

Best practices

- OPS09-BP01 Identify key performance indicators
- OPS09-BP02 Define operations metrics
- OPS09-BP03 Collect and analyze operations metrics
- OPS09-BP04 Establish operations metrics baselines
- OPS09-BP05 Learn the expected patterns of activity for operations

- OPS09-BP06 Alert when operations outcomes are at risk
- OPS09-BP07 Alert when operations anomalies are detected
- OPS09-BP08 Validate the achievement of outcomes and the effectiveness of KPIs and metrics

OPS09-BP01 Identify key performance indicators

Identify key performance indicators (KPIs) based on desired business outcomes (for example, new features delivered) and customer outcomes (for example, customer support cases). Evaluate KPIs to determine operations success.

Common anti-patterns:

- You are asked by business leadership how successful operations is at accomplishing business goals but have no frame of reference to determine success.
- You are unable to determine if your maintenance windows have an impact on business outcomes.

Benefits of establishing this best practice: By identifying key performance indicators you help achieve business outcomes as the test of the health and success of your operations.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Identify key performance indicators: Identify key performance indicators (KPIs) based on desired business and customer outcomes. Evaluate KPIs to determine operations success.

OPS09-BP02 Define operations metrics

Define operations metrics to measure the achievement of KPIs (for example, successful deployments, and failed deployments). Define operations metrics to measure the health of operations activities (for example, mean time to detect an incident (MTTD), and mean time to recovery (MTTR) from an incident). Evaluate metrics to determine if operations are achieving desired outcomes, and to understand the health of your operations activities.

Common anti-patterns:

- Your operations metrics are based on what the team thinks is reasonable.

- You have errors in your metrics calculations that will yield incorrect results.
- You don't have any metrics defined for your operations activities.

Benefits of establishing this best practice: By defining and evaluating operations metrics you can determine the health of your operations activities and measure the achievement of business outcomes.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Define operations metrics: Define operations metrics to measure the achievement of KPIs. Define operations metrics to measure the health of operations and its activities. Evaluate metrics to determine if operations are achieving desired outcomes, and to understand the health of the operations.
  - Publish custom metrics
  - Searching and filtering log data
  - Amazon CloudWatch metrics and dimensions reference

Resources

Related documents:

- AWS Answers: Centralized Logging
- Amazon CloudWatch metrics and dimensions reference
- Detect and React to Changes in Pipeline State with Amazon CloudWatch Events
- Publish custom metrics
- Searching and filtering log data

Related videos:

- Build a Monitoring Plan

OPS09-BP03 Collect and analyze operations metrics

Perform regular, proactive reviews of metrics to identify trends and determine where appropriate responses are needed.

You should aggregate log data from the processing of your operations activities and operations API calls, into a service such as CloudWatch Logs. Generate metrics from observations of necessary log content to gain insight into the performance of operations activities.

On AWS, you can export your log data to Amazon S3 or send logs directly to Amazon S3 for long-term storage. Using AWS Glue, you can discover and prepare your log data in Amazon S3 for analytics, storing associated metadata in the AWSAWS Glue Data Catalog. Amazon Athena, through its native integration with AWS Glue, can then be used to analyze your log data, querying it using standard SQL. Using a business intelligence tool like Quick you can visualize, explore, and analyze your data.

Common anti-patterns:

- Consistent delivery of new features is considered a key performance indicator. You have no method to measure how frequently deployments occur.
- You log deployments, rolled back deployments, patches, and rolled back patches to track you operations activities, but no one reviews the metrics.
- You have a recovery time objective to restore a lost database within fifteen minutes that was defined when the system was deployed and had no users. You now have ten thousand users and have been operating for two years. A recent restore took over two hours. This was not recorded and no one is aware.

Benefits of establishing this best practice: By collecting and analyzing your operations metrics, you gain understanding of the health of your operations and can gain insight to trends that have may an impact on your operations or the achievement of your business outcomes.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Collect and analyze operations metrics: Perform regular proactive reviews of metrics to identify trends and determine where appropriate responses are needed.
  - Using Amazon CloudWatch metrics

- Amazon CloudWatch metrics and dimensions reference
- Collect metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch Agent

Resources

Related documents:

- Amazon Athena
- Amazon CloudWatch metrics and dimensions reference
- Quick
- AWS Glue
- AWSAWS Glue Data Catalog
- Collect metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch Agent
- Using Amazon CloudWatch metrics

OPS09-BP04 Establish operations metrics baselines

Establish baselines for metrics to provide expected values as the basis for comparison and identification of under and over performing operations activities.

Common anti-patterns:

- You have been asked what the expected time to deploy is. You have not measured how long it takes to deploy and can not determine expected times.
- You have been asked what how long it takes to recover from an issue with the application servers. You have no information about time to recovery from first customer contact. You have no information about time to recovery from first identification of an issue through monitoring.
- You have been asked how many support personnel are required over the weekend. You have no idea how many support cases are typical over a weekend and can not provide an estimate.
- You have a recovery time objective to restore lost databases within fifteen minutes that was defined when the system was deployed and had no users. You now have ten thousand users and have been operating for two years. You have no information on how the time to restore has changed for your database.

Benefits of establishing this best practice: By defining baseline metric values you are able to evaluate current metric values, and metric trends, to determine if action is required.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Learn expected patterns of activity for operations: Establish patterns of operations activity to determine when behavior is outside of the expected values so that you can respond appropriately if required.

OPS09-BP05 Learn the expected patterns of activity for operations

Establish patterns of operations activities to identify anomalous activity so that you can respond appropriately if necessary.

Common anti-patterns:

- Your deployment failure rate has increased substantially recently. You address each of the failures independently. You do not realize that the failures correspond to deployments by a new employee who is unfamiliar with the deployment management system.

Benefits of establishing this best practice: By learning patterns of behavior, you can recognize unexpected behavior and take action if necessary.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Learn expected patterns of activity for operations: Establish patterns of operations activity to determine when behavior is outside of the expected values so that you can respond appropriately if required.

OPS09-BP06 Alert when operations outcomes are at risk

Whenever operations outcomes are at risk, an alert must be raised and acted upon. Operations outcomes are any activity that supports a workload in production. This includes everything from deploying new versions of applications to recovering from an outage. Operations outcomes must be treated with the same importance as business outcomes.

Software teams should identify key operations metrics and activities and build alerts for them. Alerts must be timely and actionable. If an alert is raised, a reference to a corresponding runbook or playbook should be included. Alerts without a corresponding action can lead to alert fatigue.

Desired outcome: When operations activities are at risk, alerts are sent to drive action. The alerts contain context on why an alert is being raised and point to a playbook to investigate or a runbook to mitigate. Where possible, runbooks are automated and notifications are sent.

Common anti-patterns:

- You are investigating an incident and support cases are being filed. The support cases are breaching the service level agreement (SLA) but no alerts are being raised.
- A deployment to production scheduled for midnight is delayed due to last-minute code changes. No alert is raised and the deployment hangs.
- A production outage occurs but no alerts are sent.
- Your deployment time consistently runs behind estimates. No action is taken to investigate.

Benefits of establishing this best practice:

- Alerting when operations outcomes are at risk boosts your ability to support your workload by staying ahead of issues.
- Business outcomes are improved due to healthy operations outcomes.
- Detection and remediation of operations issues are improved.
- Overall operational health is increased.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Operations outcomes must be defined before you can alert on them. Start by defining what operations activities are most important to your organization. Is it deploying to production in under two hours or responding to a support case within a set amount of time? Your organization must define key operations activities and how they are measured so that they can be monitored, improved, and alerted on. You need a central location where workload and operations telemetry is stored and analyzed. The same mechanism should be able to raise an alert when an operations outcome is at risk.

Customer example

A CloudWatch alarm was initiated during a routine deployment at AnyCompany Retail. The lead time for deployment was breached. Amazon EventBridge created an OpsItem in AWS Systems Manager OpsCenter. The Cloud Operations team used a playbook to investigate the issue and identified that a schema change was taking longer than expected. They alerted the on-call developer and continued monitoring the deployment. Once the deployment was complete, the Cloud Operations team resolved the OpsItem. The team will analyze the incident during a postmortem.

Implementation steps

1. If you have not identified operations KPIs, metrics, and activities, work on implementing the preceding best practices to this question (OPS09-BP01 to OPS09-BP05).
   - Support customers with Enterprise Support can request the Operations KPI Workshop from their Technical Account Manager. This collaborative workshop helps you define operations KPIs and metrics aligned to business goals, provided at no additional cost. Contact your Technical Account Manager to learn more.
2. Once you have operations activities, KPIs, and metrics established, configure alerts in your observability platform. Alerts should have an action associated to them, like a playbook or runbook. Alerts without an action should be avoided.
3. Over time, you should evaluate your operations metrics, KPIs, and activities to identify areas of improvement. Capture feedback in runbooks and playbooks from operators to identify areas for improvement in responding to alerts.
4. Alerts should include a mechanism to flag them as a false-positive. This should lead to a review of the metric thresholds.

Level of effort for the implementation plan: Medium. There are several best practices that must be in place before implementing this best practice. Once operations activities have been identified and operations KPIs established, alerts should be established.

Resources

Related best practices:

- OPS02-BP03 Operations activities have identified owners responsible for their performance: Every operation activity and outcome should have an identified owner that's responsible. This is who should be alerted when outcomes are at risk.

- OPS03-BP02 Team members are empowered to take action when outcomes are at risk: When alerts are raised, your team should have agency to act to remedy the issue.
- OPS09-BP01 Identify key performance indicators: Alerting on operations outcomes starts with identify operations KPIs.
- OPS09-BP02 Define operations metrics: Establish this best practice before you start generating alerts.
- OPS09-BP03 Collect and analyze operations metrics: Centrally collecting operations metrics is required to build alerts.
- OPS09-BP04 Establish operations metrics baselines: Operations metrics baselines provide the ability to tune alerts and avoid alert fatigue.
- OPS09-BP05 Learn the expected patterns of activity for operations: You can improve the accuracy of your alerts by understanding the activity patterns for operations events.
- OPS09-BP08 Validate the achievement of outcomes and the effectiveness of KPIs and metrics: Evaluate the achievement of operations outcomes to ensure that your KPIs and metrics are valid.
- OPS10-BP02 Have a process per alert: Every alert should have an associated runbook or playbook and provide context for the person being alerted.
- OPS11-BP02 Perform post-incident analysis: Conduct a post-incident analysis after the alert to identify areas for improvement.

Related documents:

- AWS Deployment Pipelines Reference Architecture: Application Pipeline Architecture
- GitLab: Getting Started with Agile / DevOps Metrics

Related videos:

- Aggregate and Resolve Operational Issues Using AWS Systems Manager OpsCenter
- Integrate AWS Systems Manager OpsCenter with Amazon CloudWatch Alarms
- Integrate Your Data Sources into AWS Systems Manager OpsCenter Using Amazon EventBridge

Related examples:

- Automate remediation actions for Amazon EC2 notifications and beyond using Amazon EC2 Systems Manager Automation and AWS Health

- AWS Management and Governance Tools Workshop - Operations 2022
- Ingesting, analyzing, and visualizing metrics with DevOps Monitoring Dashboard on AWS

Related services:

- Amazon EventBridge
- Support Proactive Services - Operations KPI Workshop
- AWS Systems Manager OpsCenter
- CloudWatch Events

OPS09-BP07 Alert when operations anomalies are detected

Raise an alert when operations anomalies are detected so that you can respond appropriately if necessary.

Your analysis of your operations metrics over time may established patterns of behavior that you can quantify sufficiently to define an event or raise an alarm in response.

Once trained, the CloudWatch Anomaly Detection feature can be used to alarm on detected anomalies or can provide overlaid expected values onto a graph of metric data for ongoing comparison.

Amazon DevOps Guru can be used to identify anomalous behavior through event correlation, log analysis, and applying machine learning to analyze your workload telemetry. The insights gained are presented with the relevant data and recommendations.

Common anti-patterns:

- You are applying a patch to your fleet of instances. You tested the patch successfully in the test environment. The patch is failing for a large percentage of instances in your fleet. You do nothing.
- You note that there are deployments starting Friday end of day. Your organization has predefined maintenance windows on Tuesdays and Thursdays. You do nothing.

Benefits of establishing this best practice: By understanding patterns of operations behavior you can identify unexpected behavior and take action if necessary.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Alert when operations anomalies are detected: Raise an alert when operations anomalies are detected so that you can respond appropriately if required.
  - What is Amazon CloudWatch Events?
  - Creating Amazon CloudWatch alarms
  - Invoking Lambda functions using Amazon SNS notifications

Resources

Related documents:

- Amazon DevOps Guru
- CloudWatch Anomaly Detection
- Creating Amazon CloudWatch alarms
- Detect and React to Changes in Pipeline State with Amazon CloudWatch Events
- Invoking Lambda functions using Amazon SNS notifications
- What is Amazon CloudWatch Events?

OPS09-BP08 Validate the achievement of outcomes and the effectiveness of KPIs and metrics

Create a business-level view of your operations activities to help you determine if you are satisfying needs and to identify areas that need improvement to reach business goals. Validate the effectiveness of KPIs and metrics and revise them if necessary.

AWS also has support for third-party log analysis systems and business intelligence tools through the AWS service APIs and SDKs (for example, Grafana, Kibana, and Logstash).

Common anti-patterns:

- The frequency of your deployments has increased with the growth in number of development teams. Your defined expected number of deployments is once per week. You have been regularly deploying daily. When their is an issue with your deployment system, and deployments are not possible, it goes undetected for days.
- When your business previously provided support only during core business hours from Monday to Friday. You established a next business day response time goal for incidents. You have recently started offering 24x7 support coverage with a two hour response time goal. Your overnight staff

are overwhelmed and customers are unhappy. There is no indication that there are issues with incident response times because you are reporting against a next business day target.

Benefits of establishing this best practice: By reviewing and revising KPIs and metrics, you understand how your workload supports the achievement of your business outcomes and can identify where improvement is needed to reach business goals.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Validate the achievement of outcomes and the effectiveness of KPIs and metrics: Create a business level view of your operations activities to help you determine if you are satisfying needs and to identify areas that need improvement to reach business goals. Validate the effectiveness of KPIs and metrics and revise them if necessary.
  - Using Amazon CloudWatch dashboards
  - What is log analytics?

Resources

Related documents:

- Using Amazon CloudWatch dashboards
- What is log analytics?

OPS 10. How do you manage workload and operations events?

Prepare and validate procedures for responding to events to minimize their disruption to your workload.

Best practices

- OPS10-BP01 Use a process for event, incident, and problem management
- OPS10-BP02 Have a process per alert
- OPS10-BP03 Prioritize operational events based on business impact
- OPS10-BP04 Define escalation paths
- OPS10-BP05 Define a customer communication plan for outages

- OPS10-BP06 Communicate status through dashboards
- OPS10-BP07 Automate responses to events

OPS10-BP01 Use a process for event, incident, and problem management

Your organization has processes to handle events, incidents, and problems. *Events* are things that occur in your workload but may not need intervention. *Incidents* are events that require intervention. *Problems* are recurring events that require intervention or cannot be resolved. You need processes to mitigate the impact of these events on your business and make sure that you respond appropriately.

When incidents and problems happen to your workload, you need processes to handle them. How will you communicate the status of the event with stakeholders? Who oversees leading the response? What are the tools that you use to mitigate the event? These are examples of some of the questions you need answer to have a solid response process.

Processes must be documented in a central location and available to anyone involved in your workload. If you don't have a central wiki or document store, a version control repository can be used. You'll keep these plans up to date as your processes evolve.

Problems are candidates for automation. These events take time away from your ability to innovate. Start with building a repeatable process to mitigate the problem. Over time, focus on automating the mitigation or fixing the underlying issue. This frees up time to devote to making improvements in your workload.

Desired outcome: Your organization has a process to handle events, incidents, and problems. These processes are documented and stored in a central location. They are updated as processes change.

Common anti-patterns:

- An incident happens on the weekend and the on-call engineer doesn't know what to do.
- A customer sends you an email that the application is down. You reboot the server to fix it. This happens frequently.
- There is an incident with multiple teams working independently to try to solve it.
- Deployments happen in your workload without being recorded.

Benefits of establishing this best practice:

- You have an audit trail of events in your workload.
- Your time to recover from an incident is decreased.
- Team members can resolve incidents and problems in a consistent manner.
- There is a more consolidated effort when investigating an incident.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implementing this best practice means you are tracking workload events. You have processes to handle incidents and problems. The processes are documented, shared, and updated frequently. Problems are identified, prioritized, and fixed.

Customer example

AnyCompany Retail has a portion of their internal wiki devoted to processes for event, incident, and problem management. All events are sent to Amazon EventBridge. Problems are identified as OpsItems in AWS Systems Manager OpsCenter and prioritized to fix, reducing undifferentiated labor. As processes change, they're updated in their internal wiki. They use AWS Systems Manager Incident Manager to manage incidents and coordinate mitigation efforts.

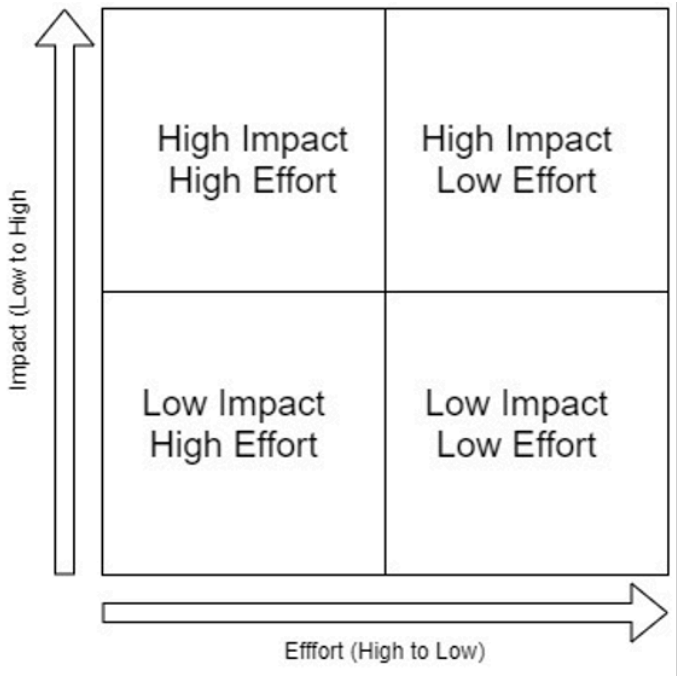
Implementation steps

1. Events
    - Track events that happen in your workload, even if no human intervention is required.
    - Work with workload stakeholders to develop a list of events that should be tracked. Some examples are completed deployments or successful patching.
    - You can use services like Amazon EventBridge or Amazon Simple Notification Service to generate custom events for tracking.
2. Incidents
    - Start by defining the communication plan for incidents. What stakeholders must be informed? How will you keep them in the loop? Who oversees coordinating efforts? We recommend standing up an internal chat channel for communication and coordination.
    - Define escalation paths for the teams that support your workload, especially if the team doesn't have an on-call rotation. Based on your support level, you can also file a case with Support.

- Create a playbook to investigate the incident. This should include the communication plan and detailed investigation steps. Include checking the AWS Health Dashboard in your investigation.
- Document your incident response plan. Communicate the incident management plan so internal and external customers understand the rules of engagement and what is expected of them. Train your team members on how to use it.
- Customers can use Incident Manager to set up and manage their incident response plan.
- Enterprise Support customers can request the Incident Management Workshop from their Technical Account Manager. This guided workshop tests your existing incident response plan and helps you identify areas for improvement.

3. Problems

- Problems must be identified and tracked in your ITSM system.
- Identify all known problems and prioritize them by effort to fix and impact to workload.



- Solve problems that are high impact and low effort first. Once those are solved, move on to problems to that fall into the low impact low effort quadrant.
- You can use Systems Manager OpsCenter to identify these problems, attach runbooks to them, and track them.

Level of effort for the implementation plan: Medium. You need both a process and tools to implement this best practice. Document your processes and make them available to anyone

associated with the workload. Update them frequently. You have a process for managing problems and mitigating them or fixing them.

Resources

Related best practices:

- OPS07-BP03 Use runbooks to perform procedures: Known problems need an associated runbook so that mitigation efforts are consistent.
- OPS07-BP04 Use playbooks to investigate issues: Incidents must be investigated using playbooks.
- OPS11-BP02 Perform post-incident analysis: Always conduct a postmortem after you recover from an incident.

Related documents:

- Atlassian - Incident management in the age of DevOps
- AWS Security Incident Response Guide
- Incident Management in the Age of DevOps and SRE
- PagerDuty - What is Incident Management?

Related videos:

- AWS re:Invent 2020: Incident management in a distributed organization
- AWS re:Invent 2021 - Building next-gen applications with event-driven architectures
- AWS Supports You | Exploring the Incident Management Tabletop Exercise
- AWS Systems Manager Incident Manager - AWS Virtual Workshops
- AWS What's Next ft. Incident Manager | AWS Events

Related examples:

- AWS Management and Governance Tools Workshop - OpsCenter
- AWS Proactive Services – Incident Management Workshop
- Building an event-driven application with Amazon EventBridge
- Building event-driven architectures on AWS

Related services:

- Amazon EventBridge
- Amazon SNS
- AWS Health Dashboard
- AWS Systems Manager Incident Manager
- AWS Systems Manager OpsCenter

OPS10-BP02 Have a process per alert

Have a well-defined response (runbook or playbook), with a specifically identified owner, for any event for which you raise an alert. This ensures effective and prompt responses to operations events and prevents actionable events from being obscured by less valuable notifications.

Common anti-patterns:

- Your monitoring system presents you a stream of approved connections along with other messages. The volume of messages is so large that you miss periodic error messages that require your intervention.
- You receive an alert that the website is down. There is no defined process for when this happens. You are forced to take an ad hoc approach to diagnose and resolve the issue. Developing this process as you go extends the time to recovery.

Benefits of establishing this best practice: By alerting only when action is required, you prevent low value alerts from concealing high value alerts. By having a process for every actionable alert, you create a consistent and prompt response to events in your environment.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Process per alert: Any event for 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) accountable for successful completion. Performance of the response may be automated or conducted by another team but the owner is accountable for ensuring the process delivers the expected outcomes. By having these processes, you ensure effective and prompt responses to operations events and you can prevent actionable events from being obscured by less valuable notifications. For example, automatic scaling might be applied to scale a web front end, but the

operations team might be accountable to ensure that the automatic scaling rules and limits are appropriate for workload needs.

Resources

Related documents:

- Amazon CloudWatch Features
- What is Amazon CloudWatch Events?

Related videos:

- Build a Monitoring Plan

OPS10-BP03 Prioritize operational events based on business impact

Ensure that when multiple events require intervention, those that are most significant to the business are addressed first. Impacts can include loss of life or injury, financial loss, or damage to reputation or trust.

Common anti-patterns:

- You receive a support request to add a printer configuration for a user. While working on the issue, you receive a support request stating that your retail site is down. After completing the printer configuration for your user, you start work on the website issue.
- You get notified that both your retail website and your payroll system are down. You don't know which one should get priority.

Benefits of establishing this best practice: Prioritizing responses to the incidents with the greatest impact on the business notifies your management of that impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Prioritize operational events based on business impact: Ensure that when multiple events require intervention, those that are most significant to the business are addressed first. Impacts can include loss of life or injury, financial loss, regulatory violations, or damage to reputation or trust.

OPS10-BP04 Define escalation paths

Define escalation paths in your runbooks and playbooks, including what initiates escalation, and procedures for escalation. Specifically identify owners for each action to ensure effective and prompt responses to operations events.

Identify when a human decision is required before an action is taken. Work with decision makers to have that decision made in advance, and the action preapproved, so that MTTR is not extended waiting for a response.

Common anti-patterns:

- Your retail site is down. You don't understand the runbook for recovering the site. You start calling colleagues hoping that someone will be able to help you.
- You receive a support case for an unreachable application. You don't have permissions to administer the system. You don't know who does. You attempt to contact the system owner that opened the case and there is no response. You have no contacts for the system and your colleagues are not familiar with it.

Benefits of establishing this best practice: By defining escalations, what initiates the escalation, and procedures for escalation you provide the systematic addition of resources to an incident at an appropriate rate for the impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Define escalation paths: Define escalation paths in your runbooks and playbooks, including what starts escalation, and procedures for escalation. For example, escalation of an issue from support engineers to senior support engineers when runbooks cannot resolve the issue, or when a predefined period of time has elapsed. Another example of an appropriate escalation path is from senior support engineers to the development team for a workload when the playbooks are unable to identify a path to remediation, or when a predefined period of time has elapsed. Specifically identify owners for each action to ensure effective and prompt responses to operations events. Escalations can include third parties. For example, a network connectivity provider or a software vendor. Escalations can include identified authorized decision makers for impacted systems.

OPS10-BP05 Define a customer communication plan for outages

Define and test a communication plan for system outages that you can rely on to keep your customers and stakeholders informed during outages. Communicate directly with your users both when the services they use are impacted and when services return to normal.

Desired outcome:

- You have a communication plan for situations ranging from scheduled maintenance to large unexpected failures, including invocation of disaster recovery plans.
- In your communications, you provide clear and transparent information about systems issues to help customers avoid second guessing the performance of their systems.
- You use custom error messages and status pages to reduce the spike in help desk requests and keep users informed.
- The communication plan is regularly tested to verify that it will perform as intended when a real outage occurs.

Common anti-patterns:

- A workload outage occurs but you have no communication plan. Users overwhelm your trouble ticket system with requests because they have no information on the outage.
- You send an email notification to your users during an outage. It doesn't contain a timeline for restoration of service so users cannot plan around the outage.
- There is a communication plan for outages but it has never been tested. An outage occurs and the communication plan fails because a critical step was missed that could have been caught in testing.
- During an outage, you send a notification to users with too many technical details and information under your AWS NDA.

Benefits of establishing this best practice:

- Maintaining communication during outages ensures that customers are provided with visibility of progress on issues and estimated time to resolution.
- Developing a well-defined communications plan verifies that your customers and end users are well informed so they can take required additional steps to mitigate the impact of outages.

- With proper communications and increased awareness of planned and unplanned outages, you can improve customer satisfaction, limit unintended reactions, and drive customer retention.
- Timely and transparent system outage communication builds confidence and establishes trust needed to maintain relationships between you and your customers.
- A proven communication strategy during an outage or crisis reduces speculation and gossip that could hinder your ability to recover.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Communication plans that keep your customers informed during outages are holistic and cover multiple interfaces including customer facing error pages, custom API error messages, system status banners, and health status pages. If your system includes registered users, you can communicate over messaging channels such as email, SMS or push notifications to send personalized message content to your customers.

Customer communication tools

As a first line of defense, web and mobile applications should provide friendly and informative error messages during an outage as well as have the ability to redirect traffic to a status page. Amazon CloudFront is a fully managed content delivery network (CDN) that includes capabilities to define and serve custom error content. Custom error pages in CloudFront are a good first layer of customer messaging for component level outages. CloudFront can also simplify managing and activating a status page to intercept all requests during planned or unplanned outages.

Custom API error messages can help detect and reduce impact when outages are isolated to discrete services. Amazon API Gateway allows you to configure custom responses for your REST APIs. This allows you to provide clear and meaningful messaging to API consumers when API Gateway is not able to reach backend services. Custom messages can also be used to support outage banner content and notifications when a particular system feature is degraded due to service tier outages.

Direct messaging is the most personalized type of customer messaging. Amazon Pinpoint is a managed service for scalable multichannel communications. Amazon Pinpoint allows you to build campaigns that can broadcast messages widely across your impacted customer base over SMS, email, voice, push notifications, or custom channels you define. When you manage messaging with Amazon Pinpoint, message campaigns are well defined, testable, and can be intelligently

applied to targeted customer segments. Once established, campaigns can be scheduled or started by events and they can easily be tested.

Customer example

When the workload is impaired, AnyCompany Retail sends out an email notification to their users. The email describes what business functionality is impaired and provides a realistic estimate of when service will be restored. In addition, they have a status page that shows real-time information about the health of their workload. The communication plan is tested in a development environment twice per year to validate that it is effective.

Implementation steps

1. Determine the communication channels for your messaging strategy. Consider the architectural aspects of your application and determine the best strategy for delivering feedback to your customers. This could include one or more of the guidance strategies outlined including error and status pages, custom API error responses, or direct messaging.
2. Design status pages for your application. If you've determined that status or custom error pages are suitable for your customers, you'll need to design your content and messaging for those pages. Error pages explain to users why an application is not available, when it may become available again, and what they can do in the meantime. If your application uses Amazon CloudFront you can serve custom error responses or use Lambda at Edge to translate errors and rewrite page content. CloudFront also makes it possible to swap destinations from your application content to a static Amazon S3 content origin containing your maintenance or outage status page .
3. Design the correct set of API error statuses for your service. Error messages produced by API Gateway when it can't reach backend services, as well as service tier exceptions, may not contain friendly messages suitable for display to end users. Without having to make code changes to your backend services, you can configure API Gateway custom error responses to map HTTP response codes to curated API error messages.
4. Design messaging from a business perspective so that it is relevant to end users for your system and does not contain technical details. Consider your audience and align your messaging. For example, you may steer internal users towards a workaround or manual process that leverages alternate systems. External users may be asked to wait until the system is restored, or subscribe to updates to receive a notification once the system is restored. Define approved messaging for multiple scenarios including unexpected outages, planned maintenance, and partial system failures where a particular feature may be degraded or unavailable.

5. Templatize and automate your customer messaging. Once you have established your message content, you can use Amazon Pinpoint or other tools to automate your messaging campaign. With Amazon Pinpoint you can create customer target segments for specific affected users and transform messages into templates. Review the Amazon Pinpoint tutorial to get an understanding of how-to setup a messaging campaign.
6. Avoiding tightly coupling messaging capabilities to your customer facing system. Your messaging strategy should not have hard dependencies on system data stores or services to verify that you can successfully send messages when you experience outages. Consider building the ability to send messages from more than one Availability Zone or Region for messaging availability. If you are using AWS services to send messages, leverage data plane operations over control plane operation to invoke your messaging.

Level of effort for the implementation plan: High. Developing a communication plan, and the mechanisms to send it, can require a significant effort.

Resources

Related best practices:

- OPS07-BP03 Use runbooks to perform procedures - Your communication plan should have a runbook associated with it so that your personnel know how to respond.
- OPS11-BP02 Perform post-incident analysis - After an outage, conduct post-incident analysis to identify mechanisms to prevent another outage.

Related documents:

- Error Handling Patterns in Amazon API Gateway and AWS Lambda
- Amazon API Gateway responses

Related examples:

- AWS Health Dashboard
- Summary of the AWS Service Event in the Northern Virginia (US-EAST-1) Region

Related services:

- AWS Support

- AWS Customer Agreement
- Amazon CloudFront
- Amazon API Gateway
- Amazon Pinpoint
- Amazon S3

OPS10-BP06 Communicate status through dashboards

Provide dashboards tailored to their target audiences (for example, internal technical teams, leadership, and customers) to communicate the current operating status of the business and provide metrics of interest.

You can create dashboards using Amazon CloudWatch Dashboards on customizable home pages in the CloudWatch console. Using business intelligence services such as Quick you can create and publish interactive dashboards of your workload and operational health (for example, order rates, connected users, and transaction times). Create Dashboards that present system and business-level views of your metrics.

Common anti-patterns:

- Upon request, you run a report on the current utilization of your application for management.
- During an incident, you are contacted every twenty minutes by a concerned system owner wanting to know if it is fixed yet.

Benefits of establishing this best practice: By creating dashboards, you create self-service access to information helping your customers to informed themselves and determine if they need to take action.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Communicate status through dashboards: Provide dashboards tailored to their target audiences (for example, internal technical teams, leadership, and customers) to communicate the current operating status of the business and provide metrics of interest. Providing a self-service option for status information reduces the disruption of fielding requests for status by the operations team. Examples include Amazon CloudWatch dashboards, and AWS Health Dashboard.

- CloudWatch dashboards create and use customized metrics views

Resources

Related documents:

- Quick
- CloudWatch dashboards create and use customized metrics views

OPS10-BP07 Automate responses to events

Automate responses to events to reduce errors caused by manual processes, and to ensure prompt and consistent responses.

There are multiple ways to automate runbook and playbook actions on AWS. To respond to an event from a state change in your AWS resources, or from your own custom events, you should create CloudWatch Events rules to initiate responses through CloudWatch targets (for example, Lambda functions, Amazon Simple Notification Service (Amazon SNS) topics, Amazon ECS tasks, and AWS Systems Manager Automation).

To respond to a metric that crosses a threshold for a resource (for example, wait time), you should create CloudWatch alarms to perform one or more actions using Amazon EC2 actions, Auto Scaling actions, or to send a notification to an Amazon SNS topic. If you need to perform custom actions in response to an alarm, invoke Lambda through an Amazon SNS notification. Use Amazon SNS to publish event notifications and escalation messages to keep people informed.

AWS also supports third-party systems through the AWS service APIs and SDKs. There are a number of monitoring tools provided by AWS Partners and third parties that allow for monitoring, notifications, and responses. Some of these tools include New Relic, Splunk, Loggly, SumoLogic, and Datadog.

You should keep critical manual procedures available for use when automated procedures fail

Common anti-patterns:

- A developer checks in their code. This event could have been used to start a build and then perform testing but instead nothing happens.
- Your application logs a specific error before it stops working. The procedure to restart the application is well understood and could be scripted. You could use the log event to invoke a

script and restart the application. Instead, when the error happens at 3am Sunday morning, you are woken up as the on-call resource responsible to fix the system.

Benefits of establishing this best practice: By using automated responses to events, you reduce the time to respond and limit the introduction of errors from manual activities.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Automate responses to events: Automate responses to events to reduce errors caused by manual processes, and to ensure prompt and consistent responses.
  - What is Amazon CloudWatch Events?
  - Creating a CloudWatch Events rule that starts on an event
  - Creating a CloudWatch Events rule that starts on an AWS API call using AWS CloudTrail
  - CloudWatch Events event examples from supported services

Resources

Related documents:

- Amazon CloudWatch Features
- CloudWatch Events event examples from supported services
- Creating a CloudWatch Events rule that starts on an AWS API call using AWS CloudTrail
- Creating a CloudWatch Events rule that starts on an event
- What is Amazon CloudWatch Events?

Related videos:

- Build a Monitoring Plan

Related examples:

Evolve

Question

- OPS 11. How do you evolve operations?

OPS 11. How do you evolve operations?

Dedicate time and resources for nearly continuous incremental improvement to evolve the effectiveness and efficiency of your operations.

Best practices

- OPS11-BP01 Have a process for continuous improvement
- OPS11-BP02 Perform post-incident analysis
- OPS11-BP03 Implement feedback loops
- OPS11-BP04 Perform knowledge management
- OPS11-BP05 Define drivers for improvement
- OPS11-BP06 Validate insights
- OPS11-BP07 Perform operations metrics reviews
- OPS11-BP08 Document and share lessons learned
- OPS11-BP09 Allocate time to make improvements

OPS11-BP01 Have a process for continuous improvement

Evaluate your workload against internal and external architecture best practices. Conduct workload reviews at least once per year. Prioritize improvement opportunities into your software development cadence.

Desired outcome:

- You analyze your workload against architecture best practices at least yearly.
- Improvement opportunities are given equal priority in your software development process.

Common anti-patterns:

- You have not conducted an architecture review on your workload since it was deployed several years ago.
- Improvement opportunities are given a lower priority and stay in the backlog.
- There is no standard for implementing modifications to best practices for the organization.

Benefits of establishing this best practice:

- Your workload is kept up to date on architecture best practices.
- Evolving your workload is done in a deliberate manner.
- You can leverage organization best practices to improve all workloads.

Level of risk exposed if this best practice is not established: High

Implementation guidance

On at least a yearly basis, you conduct an architectural review of your workload. Using internal and external best practices, evaluate your workload and identify improvement opportunities. Prioritize improvement opportunities into your software development cadence.

Customer example

All workloads at AnyCompany Retail go through a yearly architecture review process. They developed their own checklist of best practices that apply to all workloads. Using the AWS Well-Architected Tool's Custom Lens feature, they conduct reviews using the tool and their custom lens of best practices. Improvement opportunities generated from the reviews are given priority in their software sprints.

Implementation steps

1. Conduct periodic architecture reviews of your production workload at least yearly. Use a documented architectural standard that includes AWS-specific best practices.
   a. We recommend you use your own internally defined standards it for these reviews. If you do not have an internal standard, we recommend you use the AWS Well-Architected Framework.
   b. You can use the AWS Well-Architected Tool to create a Custom Lens of your internal best practices and conduct your architecture review.
   c. Customers can contact their AWS Solutions Architect to conduct a guided Well-Architected Framework Review of their workload.
2. Prioritize improvement opportunities identified during the review into your software development process.

Level of effort for the implementation plan: Low. You can use the AWS Well-Architected Framework to conduct your yearly architecture review.

Resources

Related best practices:

- OPS11-BP02 Perform post-incident analysis - Post-incident analysis is another generator for improvement items. Feed lessons learned into your internal list of architecture best practices.
- OPS11-BP08 Document and share lessons learned - As you develop your own architecture best practices, share those across your organization.

Related documents:

- AWS Well-Architected Tool - Custom lenses
- AWS Well-Architected Whitepaper - The review process
- Customize Well-Architected Reviews using Custom Lenses and the AWS Well-Architected Tool
- Implementing the AWS Well-Architected Custom Lens lifecycle in your organization

Related videos:

- Well-Architected Labs - Level 100: Custom Lenses on AWS Well-Architected Tool

Related examples:

- The AWS Well-Architected Tool

OPS11-BP02 Perform post-incident analysis

Review customer-impacting events, and identify the contributing factors and preventative actions. Use this information to develop mitigations to limit or prevent recurrence. Develop procedures for prompt and effective responses. Communicate contributing factors and corrective actions as appropriate, tailored to target audiences.

Common anti-patterns:

- You administer an application server. Approximately every 23 hours and 55 minutes all your active sessions are terminated. You have tried to identify what is going wrong on your application server. You suspect it could instead be a network issue but are unable to get cooperation from the network team as they are too busy to support you. You lack a predefined

process to follow to get support and collect the information necessary to determine what is going on.

- You have had data loss within your workload. This is the first time it has happened and the cause is not obvious. You decide it is not important because you can recreate the data. Data loss starts occurring with greater frequency impacting your customers. This also places addition operational burden on you as you restore the missing data.

Benefits of establishing this best practice: Having a predefined processes to determine the components, conditions, actions, and events that contributed to an incident helps you to identify opportunities for improvement.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Use a process to determine contributing factors: Review all customer impacting incidents. Have a process to identify and document the contributing factors of an incident so that you can develop mitigations to limit or prevent recurrence and you can develop procedures for prompt and effective responses. Communicate root cause as appropriate, tailored to target audiences.

OPS11-BP03 Implement feedback loops

Feedback loops provide actionable insights that drive decision making. Build feedback loops into your procedures and workloads. This helps you identify issues and areas that need improvement. They also validate investments made in improvements. These feedback loops are the foundation for continuously improving your workload.

Feedback loops fall into two categories: *immediate feedback* and *retrospective analysis*. Immediate feedback is gathered through review of the performance and outcomes from operations activities. This feedback comes from team members, customers, or the automated output of the activity. Immediate feedback is received from things like A/B testing and shipping new features, and it is essential to failing fast.

Retrospective analysis is performed regularly to capture feedback from the review of operational outcomes and metrics over time. These retrospectives happen at the end of a sprint, on a cadence, or after major releases or events. This type of feedback loop validates investments in operations or your workload. It helps you measure success and validates your strategy.

Desired outcome: You use immediate feedback and retrospective analysis to drive improvements. There is a mechanism to capture user and team member feedback. Retrospective analysis is used to identify trends that drive improvements.

Common anti-patterns:

- You launch a new feature but have no way of receiving customer feedback on it.
- After investing in operations improvements, you don't conduct a retrospective to validate them.
- You collect customer feedback but don't regularly review it.
- Feedback loops lead to proposed action items but they aren't included in the software development process.
- Customers don't receive feedback on improvements they've proposed.

Benefits of establishing this best practice:

- You can work backwards from the customer to drive new features.
- Your organization culture can react to changes faster.
- Trends are used to identify improvement opportunities.
- Retrospectives validate investments made to your workload and operations.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implementing this best practice means that you use both immediate feedback and retrospective analysis. These feedback loops drive improvements. There are many mechanisms for immediate feedback, including surveys, customer polls, or feedback forms. Your organization also uses retrospectives to identify improvement opportunities and validate initiatives.

Customer example

AnyCompany Retail created a web form where customers can give feedback or report issues. During the weekly scrum, user feedback is evaluated by the software development team. Feedback is regularly used to steer the evolution of their platform. They conduct a retrospective at the end of each sprint to identify items they want to improve.

Implementation steps

1. Immediate feedback
   - You need a mechanism to receive feedback from customers and team members. Your operations activities can also be configured to deliver automated feedback.
   - Your organization needs a process to review this feedback, determine what to improve, and schedule the improvement.
   - Feedback must be added into your software development process.
   - As you make improvements, follow up with the feedback submitter.
     - You can use AWS Systems Manager OpsCenter to create and track these improvements as OpsItems.
2. Retrospective analysis
   - Conduct retrospectives at the end of a development cycle, on a set cadence, or after a major release.
   - Gather stakeholders involved in the workload for a retrospective meeting.
   - Create three columns on a whiteboard or spreadsheet: Stop, Start, and Keep.
     - *Stop* is for anything that you want your team to stop doing.
     - *Start* is for ideas that you want to start doing.
     - *Keep* is for items that you want to keep doing.
   - Go around the room and gather feedback from the stakeholders.
   - Prioritize the feedback. Assign actions and stakeholders to any Start or Keep items.
   - Add the actions to your software development process and communicate status updates to stakeholders as you make the improvements.

Level of effort for the implementation plan: Medium. To implement this best practice, you need a way to take in immediate feedback and analyze it. Also, you need to establish a retrospective analysis process.

Resources

Related best practices:

- OPS01-BP01 Evaluate external customer needs: Feedback loops are a mechanism to gather external customer needs.

- OPS01-BP02 Evaluate internal customer needs: Internal stakeholders can use feedback loops to communicate needs and requirements.
- OPS11-BP02 Perform post-incident analysis: Post-incident analyses are an important form of retrospective analysis conducted after incidents.
- OPS11-BP07 Perform operations metrics reviews: Operations metrics reviews identify trends and areas for improvement.

Related documents:

- 7 Pitfalls to Avoid When Building a CCOE
- Atlassian Team Playbook - Retrospectives
- Email Definitions: Feedback Loops
- Establishing Feedback Loops Based on the AWS Well-Architected Framework Review
- IBM Garage Methodology - Hold a retrospective
- Investopedia – The PDCS Cycle
- Maximizing Developer Effectiveness by Tim Cochran
- Operations Readiness Reviews (ORR) Whitepaper - Iteration
- ITIL CSI - Continual Service Improvement
- When Toyota met e-commerce: Lean at Amazon

Related videos:

- Building Effective Customer Feedback Loops

Related examples:

- Astuto - Open source customer feedback tool
- AWS Solutions - QnABot on AWS
- Fider - A platform to organize customer feedback

Related services:

- AWS Systems Manager OpsCenter

OPS11-BP04 Perform knowledge management

Knowledge management helps team members find the information to perform their job. In learning organizations, information is freely shared which empowers individuals. The information can be discovered or searched. Information is accurate and up to date. Mechanisms exist to create new information, update existing information, and archive outdated information. The most common example of a knowledge management platform is a content management system like a wiki.

Desired outcome:

- Team members have access to timely, accurate information.
- Information is searchable.
- Mechanisms exist to add, update, and archive information.

Common anti-patterns:

- There is no centralized knowledge storage. Team members manage their own notes on their local machines.
- You have a self-hosted wiki but no mechanisms to manage information, resulting in outdated information.
- Someone identifies missing information but there's no process to request adding it the team wiki. They add it themselves but they miss a key step, leading to an outage.

Benefits of establishing this best practice:

- Team members are empowered because information is shared freely.
- New team members are onboarded faster because documentation is up to date and searchable.
- Information is timely, accurate, and actionable.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Knowledge management is an important facet of learning organizations. To begin, you need a central repository to store your knowledge (as a common example, a self-hosted wiki). You must

develop processes for adding, updating, and archiving knowledge. Develop standards for what should be documented and let everyone contribute.

Customer example

AnyCompany Retail hosts an internal Wiki where all knowledge is stored. Team members are encouraged to add to the knowledge base as they go about their daily duties. On a quarterly basis, a cross-functional team evaluates which pages are least updated and determines if they should be archived or updated.

Implementation steps

1. Start with identifying the content management system where knowledge will be stored. Get agreement from stakeholders across your organization.
   a. If you don't have an existing content management system, consider running a self-hosted wiki or using a version control repository as a starting point.
2. Develop runbooks for adding, updating, and archiving information. Educate your team on these processes.
3. Identify what knowledge should be stored in the content management system. Start with daily activities (runbooks and playbooks) that team members perform. Work with stakeholders to prioritize what knowledge is added.
4. On a periodic basis, work with stakeholders to identify out-of-date information and archive it or bring it up to date.

Level of effort for the implementation plan: Medium. If you don't have an existing content management system, you can set up a self-hosted wiki or a version-controlled document repository.

Resources

Related best practices:

- OPS11-BP08 Document and share lessons learned - Knowledge management facilitates information sharing about lessons learned.

Related documents:

- Atlassian - Knowledge Management

Related examples:

- DokuWiki
- Gollum
- MediaWiki
- Wiki.js

OPS11-BP05 Define drivers for improvement

Identify drivers for improvement to help you evaluate and prioritize opportunities.

On AWS, you can aggregate the logs of all your operations activities, workloads, and infrastructure to create a detailed activity history. You can then use AWS tools to analyze your operations and workload health over time (for example, identify trends, correlate events and activities to outcomes, and compare and contrast between environments and across systems) to reveal opportunities for improvement based on your drivers.

You should use CloudTrail to track API activity (through the AWS Management Console, CLI, SDKs, and APIs) to know what is happening across your accounts. Track your AWS developer Tools deployment activities with CloudTrail and CloudWatch. This will add a detailed activity history of your deployments and their outcomes to your CloudWatch Logs log data.

Export your log data to Amazon S3 for long-term storage. Using AWS Glue, you discover and prepare your log data in Amazon S3 for analytics. Use Amazon Athena, through its native integration with AWS Glue, to analyze your log data. Use a business intelligence tool like Quick to visualize, explore, and analyze your data

Common anti-patterns:

- You have a script that works but is not elegant. You invest time in rewriting it. It is now a work of art.
- Your start-up is trying to get another set of funding from a venture capitalist. They want you to demonstrate compliance with PCI DSS. You want to make them happy so you document your compliance and miss a delivery date for a customer, losing that customer. It wasn't a wrong thing to do but now you wonder if it was the right thing to do.

Benefits of establishing this best practice: By determining the criteria you want to use for improvement, you can minimize the impact of event based motivations or emotional investment.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Understand drivers for improvement: You should only make changes to a system when a desired outcome is supported.
  - Desired capabilities: Evaluate desired features and capabilities when evaluating opportunities for improvement.
    - What's New with AWS
  - Unacceptable issues: Evaluate unacceptable issues, bugs, and vulnerabilities when evaluating opportunities for improvement.
    - AWS Latest Security Bulletins
    - AWS Trusted Advisor
  - Compliance requirements: Evaluate updates and changes required to maintain compliance with regulation, policy, or to remain under support from a third party, when reviewing opportunities for improvement.
    - AWS Compliance
    - AWS Compliance Programs
    - AWS Compliance Latest News

Resources

Related documents:

- Amazon Athena
- Quick
- AWS Compliance
- AWS Compliance Latest News
- AWS Compliance Programs
- AWS Glue
- AWS Latest Security Bulletins
- AWS Trusted Advisor
- Export your log data to Amazon S3
- What's New with AWS

OPS11-BP06 Validate insights

Review your analysis results and responses with cross-functional teams and business owners. Use these reviews to establish common understanding, identify additional impacts, and determine courses of action. Adjust responses as appropriate.

Common anti-patterns:

- You see that CPU utilization is at 95% on a system and make it a priority to find a way to reduce load on the system. You determine the best course of action is to scale up. The system is a transcoder and the system is scaled to run at 95% CPU utilization all the time. The system owner could have explained the situation to you had you contacted them. Your time has been wasted.
- A system owner maintains that their system is mission critical. The system was not placed in a high security environment. To improve security, you implement the additional detective and preventative controls that are required for mission critical systems. You notify the system owner that the work is complete and that he will be charged for the additional resources. In the discussion following this notification, the system owner learns there is a formal definition for mission critical systems that this system does not meet.

Benefits of establishing this best practice: By validating insights with business owners and subject matter experts, you can establish common understanding and more effectively guide improvement.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Validate insights: Engage with business owners and subject matter experts to ensure there is common understanding and agreement of the meaning of the data you have collected. Identify additional concerns, potential impacts, and determine a courses of action.

OPS11-BP07 Perform operations metrics reviews

Regularly perform retrospective analysis of operations metrics with cross-team participants from different areas of the business. Use these reviews to identify opportunities for improvement, potential courses of action, and to share lessons learned.

Look for opportunities to improve in all of your environments (for example, development, test, and production).

Common anti-patterns:

- There was a significant retail promotion that was interrupted by your maintenance window. The business remains unaware that there is a standard maintenance window that could be delayed if there are other business impacting events.
- You suffered an extended outage because of your use of a buggy library commonly used in your organization. You have since migrated to a reliable library. The other teams in your organization do not know that they are at risk. If you met regularly and reviewed this incident, they would be aware of the risk.
- Performance of your transcoder has been falling off steadily and impacting the media team. It isn't terrible yet. You will not have an opportunity to find out until it is bad enough to cause an incident. Were you to review your operations metrics with the media team, there would be an opportunity for the change in metrics and their experience to be recognized and the issue addressed.
- You are not reviewing your satisfaction of customer SLAs. You are trending to not meet your customer SLAs. There are financial penalties related to not meeting your customer SLAs. If you meet regularly to review the metrics for these SLAs, you would have the opportunity to recognize and address the issue.

Benefits of establishing this best practice: By meeting regularly to review operations metrics, events, and incidents, you maintain common understanding across teams, share lessons learned, and can prioritize and target improvements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Operations metrics reviews: Regularly perform retrospective analysis of operations metrics with cross-team participants from different areas of the business. Engage stakeholders, including the business, development, and operations teams, to validate your findings from immediate feedback and retrospective analysis, and to share lessons learned. Use their insights to identify opportunities for improvement and potential courses of action.
  - Amazon CloudWatch
  - Using Amazon CloudWatch metrics
  - Publish custom metrics
  - Amazon CloudWatch metrics and dimensions reference

Resources

Related documents:

- Amazon CloudWatch
- Amazon CloudWatch metrics and dimensions reference
- Publish custom metrics
- Using Amazon CloudWatch metrics

OPS11-BP08 Document and share lessons learned

Document and share lessons learned from the operations activities so that you can use them internally and across teams.

You should share what your teams learn to increase the benefit across your organization. You will want to share information and resources to prevent avoidable errors and ease development efforts. This will allow you to focus on delivering desired features.

Use AWS Identity and Access Management (IAM) to define permissions permitting controlled access to the resources you wish to share within and across accounts. You should then use version-controlled AWS CodeCommit repositories to share application libraries, scripted procedures, procedure documentation, and other system documentation. Share your compute standards by sharing access to your AMIs and by authorizing the use of your Lambda functions across accounts. You should also share your infrastructure standards as AWS CloudFormation templates.

Through the AWS APIs and SDKs, you can integrate external and third-party tools and repositories (for example, GitHub, BitBucket, and SourceForge). When sharing what you have learned and developed, be careful to structure permissions to ensure the integrity of shared repositories.

Common anti-patterns:

- You suffered an extended outage because of your use of a buggy library commonly used in your organization. You have since migrated to a reliable library. The other teams in your organization do not know they are at risk. Were you to document and share your experience with this library, they would be aware of the risk.
- You have identified an edge case in an internally shared microservice that causes sessions to drop. You have updated your calls to the service to avoid this edge case. The other teams in your organization do not know that they are at risk. Were you to document and share your experience with this library, they would be aware of the risk.

- You have found a way to significantly reduce the CPU utilization requirements for one of your microservices. You do not know if any other teams could take advantage of this technique. Were you to document and share your experience with this library, they would have the opportunity to do so.

Benefits of establishing this best practice: Share lessons learned to support improvement and to maximize the benefits of experience.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Document and share lessons learned: Have procedures to document the lessons learned from the running of operations activities and retrospective analysis so that they can be used by other teams.
  - Share learnings: Have procedures to share lessons learned and associated artifacts across teams. For example, share updated procedures, guidance, governance, and best practices through an accessible wiki. Share scripts, code, and libraries through a common repository.
    - Delegating access to your AWS environment
    - Share an AWS CodeCommit repository
    - Easy authorization of AWS Lambda functions
    - Sharing an AMI with specific AWS Accounts
    - Speed template sharing with an AWS CloudFormation designer URL
    - Using AWS Lambda with Amazon SNS

Resources

Related documents:

- Easy authorization of AWS Lambda functions
- Share an AWS CodeCommit repository
- Sharing an AMI with specific AWS Accounts
- Speed template sharing with an AWS CloudFormation designer URL
- Using AWS Lambda with Amazon SNS

Related videos:

- Delegating access to your AWS environment

OPS11-BP09 Allocate time to make improvements

Dedicate time and resources within your processes to make continuous incremental improvements possible.

On AWS, you can create temporary duplicates of environments, lowering the risk, effort, and cost of experimentation and testing. These duplicated environments can be used to test the conclusions from your analysis, experiment, and develop and test planned improvements.

Common anti-patterns:

- There is a known performance issue in your application server. It is added to the backlog behind every planned feature implementation. If the rate of planned features being added remains constant, the performance issue will never be addressed.
- To support continual improvement you approve administrators and developers using all their extra time to select and implement improvements. No improvements are ever completed.

Benefits of establishing this best practice: By dedicating time and resources within your processes you make continuous incremental improvements possible.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Allocate time to make improvements: Dedicate time and resources within your processes to make continuous incremental improvements possible. Implement changes to improve and evaluate the results to determine success. If the results do not satisfy the goals, and the improvement is still a priority, pursue alternative courses of action.

Security

The Security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security. You can find prescriptive guidance on implementation in the Security Pillar whitepaper.

Best practice areas

- Security foundations
- Identity and access management
- Detection
- Infrastructure protection
- Data protection
- Incident response
- Application security

Security foundations

Question

- SEC 1. How do you securely operate your workload?

SEC 1. How do you securely operate your workload?

To operate your workload securely, you must apply overarching best practices to every area of security. Take requirements and processes that you have defined in operational excellence at an organizational and workload level, and apply them to all areas. Staying up to date with AWS and industry recommendations and threat intelligence helps you evolve your threat model and control objectives. Automating security processes, testing, and validation permit you to scale your security operations.

Best practices

- SEC01-BP01 Separate workloads using accounts
- SEC01-BP02 Secure account root user and properties
- SEC01-BP03 Identify and validate control objectives
- SEC01-BP04 Keep up-to-date with security threats
- SEC01-BP05 Keep up-to-date with security recommendations
- SEC01-BP06 Automate testing and validation of security controls in pipelines
- SEC01-BP07 Identify threats and prioritize mitigations using a threat model
- SEC01-BP08 Evaluate and implement new security services and features regularly

SEC01-BP01 Separate workloads using accounts

Establish common guardrails and isolation between environments (such as production, development, and test) and workloads through a multi-account strategy. Account-level separation is strongly recommended, as it provides a strong isolation boundary for security, billing, and access.

Desired outcome: An account structure that isolates cloud operations, unrelated workloads, and environments into separate accounts, increasing security across the cloud infrastructure.

Common anti-patterns:

- Placing multiple unrelated workloads with different data sensitivity levels into the same account.
- Poorly defined organizational unit (OU) structure.

Benefits of establishing this best practice:

- Decreased scope of impact if a workload is inadvertently accessed.
- Central governance of access to AWS services, resources, and Regions.
- Maintain security of the cloud infrastructure with policies and centralized administration of security services.
- Automated account creation and maintenance process.
- Centralized auditing of your infrastructure for compliance and regulatory requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS accounts provide a security isolation boundary between workloads or resources that operate at different sensitivity levels. AWS provides tools to manage your cloud workloads at scale through a multi-account strategy to leverage this isolation boundary. For guidance on the concepts, patterns, and implementation of a multi-account strategy on AWS, see Organizing Your AWS Environment Using Multiple Accounts.

When you have multiple AWS accounts under central management, your accounts should be organized into a hierarchy defined by layers of organizational units (OUs). Security controls can then be organized and applied to the OUs and member accounts, establishing consistent preventative controls on member accounts in the organization. The security controls are inherited,

allowing you to filter permissions available to member accounts located at lower levels of an OU hierarchy. A good design takes advantage of this inheritance to reduce the number and complexity of security policies required to achieve the desired security controls for each member account.

AWS Organizations and AWS Control Tower are two services that you can use to implement and manage this multi-account structure in your AWS environment. AWS Organizations allows you to organize accounts into a hierarchy defined by one or more layers of OUs, with each OU containing a number of member accounts. Service control policies (SCPs) allow the organization administrator to establish granular preventative controls on member accounts, and AWS Config can be used to establish proactive and detective controls on member accounts. Many AWS services integrate with AWS Organizations to provide delegated administrative controls and performing service-specific tasks across all member accounts in the organization.

Layered on top of AWS Organizations, AWS Control Tower provides a one-click best practices setup for a multi-account AWS environment with a landing zone. The landing zone is the entry point to the multi-account environment established by Control Tower. Control Tower provides several benefits over AWS Organizations. Three benefits that provide improved account governance are:

- Integrated mandatory security controls that are automatically applied to accounts admitted into the organization.
- Optional controls that can be turned on or off for a given set of OUs.
- AWS Control Tower Account Factory provides automated deployment of accounts containing pre-approved baselines and configuration options inside your organization.

Implementation steps

1. **Design an organizational unit structure:** A properly designed organizational unit structure reduces the management burden required to create and maintain service control policies and other security controls. Your organizational unit structure should be aligned with your business needs, data sensitivity, and workload structure.
2. **Create a landing zone for your multi-account environment:** A landing zone provides a consistent security and infrastructure foundation from which your organization can quickly develop, launch, and deploy workloads. You can use a custom-built landing zone or AWS Control Tower to orchestrate your environment.
3. **Establish guardrails:** Implement consistent security guardrails for your environment through your landing zone. AWS Control Tower provides a list of mandatory and optional controls that can be deployed. Mandatory controls are automatically deployed when implementing Control

Tower. Review the list of highly recommended and optional controls, and implement controls that are appropriate to your needs.

4. **Restrict access to newly added Regions**: For new AWS Regions, IAM resources such as users and roles are only propagated to the Regions that you specify. This action can be performed through the console when using Control Tower, or by adjusting IAM permission policies in AWS Organizations.
5. **Consider AWS CloudFormation StackSets**: StackSets help you deploy resources including IAM policies, roles, and groups into different AWS accounts and Regions from an approved template.

Resources

Related best practices:

- SEC02-BP04 Rely on a centralized identity provider

Related documents:

- AWS Control Tower
- AWS Security Audit Guidelines
- IAM Best Practices
- Use CloudFormation StackSets to provision resources across multiple AWS accounts and regions
- Organizations FAQ
- AWS Organizations terminology and concepts
- Best Practices for Service Control Policies in an AWS Organizations Multi-Account Environment
- AWS Account Management Reference Guide
- Organizing Your AWS Environment Using Multiple Accounts

Related videos:

- Enable AWS adoption at scale with automation and governance
- Security Best Practices the Well-Architected Way
- Building and Governing Multiple Accounts using AWS Control Tower
- Enable Control Tower for Existing Organizations

Related workshops:

- Control Tower Immersion Day

SEC01-BP02 Secure account root user and properties

The root user is the most privileged user in an AWS account, with full administrative access to all resources within the account, and in some cases cannot be constrained by security policies. Deactivating programmatic access to the root user, establishing appropriate controls for the root user, and avoiding routine use of the root user helps reduce the risk of inadvertent exposure of the root credentials and subsequent compromise of the cloud environment.

Desired outcome: Securing the root user helps reduce the chance that accidental or intentional damage can occur through the misuse of root user credentials. Establishing detective controls can also alert the appropriate personnel when actions are taken using the root user.

Common anti-patterns:

- Using the root user for tasks other than the few that require root user credentials.
- Neglecting to test contingency plans on a regular basis to verify the functioning of critical infrastructure, processes, and personnel during an emergency.
- Only considering the typical account login flow and neglecting to consider or test alternate account recovery methods.
- Not handling DNS, email servers, and telephone providers as part of the critical security perimeter, as these are used in the account recovery flow.

Benefits of establishing this best practice: Securing access to the root user builds confidence that actions in your account are controlled and audited.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS offers many tools to help secure your account. However, because some of these measures are not turned on by default, you must take direct action to implement them. Consider these recommendations as foundational steps to securing your AWS account. As you implement these steps it's important that you build a process to continuously assess and monitor the security controls.

When you first create an AWS account, you begin with one identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user. You can sign in as the root user using the email address and password that you used to create the account. Due to the elevated access granted to the AWS root user, you must limit use of the AWS root user to perform tasks that specifically require it. The root user login credentials must be closely guarded, and multi-factor authentication (MFA) should always be used for the AWS account root user.

In addition to the normal authentication flow to log into your root user using a username, password, and multi-factor authentication (MFA) device, there are account recovery flows to log into your AWS account root user given access to the email address and phone number associated with your account. Therefore, it is equally important to secure the root user email account where the recovery email is sent and the phone number associated with the account. Also consider potential circular dependencies where the email address associated with the root user is hosted on email servers or domain name service (DNS) resources from the same AWS account.

When using AWS Organizations, there are multiple AWS accounts each of which have a root user. One account is designated as the management account and several layers of member accounts can then be added underneath the management account. Prioritize securing your management account's root user, then address your member account root users. The strategy for securing your management account's root user can differ from your member account root users, and you can place preventative security controls on your member account root users.

Implementation steps

The following implementation steps are recommended to establish controls for the root user. Where applicable, recommendations are cross-referenced to CIS AWS Foundations benchmark version 1.4.0. In addition to these steps, consult AWS best practice guidelines for securing your AWS account and resources.

Preventative controls

1. Set up accurate contact information for the account.
    a. This information is used for the lost password recovery flow, lost MFA device account recovery flow, and for critical security-related communications with your team.
    b. Use an email address hosted by your corporate domain, preferably a distribution list, as the root user's email address. Using a distribution list rather than an individual's email account provides additional redundancy and continuity for access to the root account over long periods of time.

   c. The phone number listed on the contact information should be a dedicated, secure phone for this purpose. The phone number should not be listed or shared with anyone.

2. Do not create access keys for the root user. If access keys exist, remove them (CIS 1.4).
   a. Eliminate any long-lived programmatic credentials (access and secret keys) for the root user.
   b. If root user access keys already exist, you should transition processes using those keys to use temporary access keys from an AWS Identity and Access Management (IAM) role, then delete the root user access keys.
3. Determine if you need to store credentials for the root user.
   a. If you are using AWS Organizations to create new member accounts, the initial password for the root user on new member accounts is set to a random value that is not exposed to you. Consider using the password reset flow from your AWS Organization management account to gain access to the member account if needed.
   b. For standalone AWS accounts or the management AWS Organization account, consider creating and securely storing credentials for the root user. Use MFA for the root user.
4. Use preventative controls for member account root users in AWS multi-account environments.
   a. Consider using the Disallow Creation of Root Access Keys for the Root User preventative guard rail for member accounts.
   b. Consider using the Disallow Actions as a Root User preventative guard rail for member accounts.
5. If you need credentials for the root user:
   a. Use a complex password.
   b. Turn on multi-factor authentication (MFA) for the root user, especially for AWS Organizations management (payer) accounts (CIS 1.5).
   c. Consider hardware MFA devices for resiliency and security, as single use devices can reduce the chances that the devices containing your MFA codes might be reused for other purposes. Verify that hardware MFA devices powered by a battery are replaced regularly. (CIS 1.6)
      - To configure MFA for the root user, follow the instructions for creating either a virtual MFA or hardware MFA device.
   d. Consider enrolling multiple MFA devices for backup. Up to 8 MFA devices are allowed per account.
      - Note that enrolling more than one MFA device for the root user automatically turns off the flow for recovering your account if the MFA device is lost.

e. Store the password securely, and consider circular dependencies if storing the password electronically. Don't store the password in such a way that would require access to the same AWS account to obtain it.

6. Optional: Consider establishing a periodic password rotation schedule for the root user.
   - Credential management best practices depend on your regulatory and policy requirements. Root users protected by MFA are not reliant on the password as a single factor of authentication.
   - Changing the root user password on a periodic basis reduces the risk that an inadvertently exposed password can be misused.

Detective controls

- Create alarms to detect use of the root credentials (CIS 1.7). Amazon GuardDuty can monitor and alert on root user API credential usage through the RootCredentialUsage finding.
- Evaluate and implement the detective controls included in the AWS Well-Architected Security Pillar conformance pack for AWS Config, or if using AWS Control Tower, the strongly recommended controls available inside Control Tower.

Operational guidance

- Determine who in the organization should have access to the root user credentials.
  - Use a two-person rule so that no one individual has access to all necessary credentials and MFA to obtain root user access.
  - Verify that the organization, and not a single individual, maintains control over the phone number and email alias associated with the account (which are used for password reset and MFA reset flow).
- Use root user only by exception (CIS 1.7).
  - The AWS root user must not be used for everyday tasks, even administrative ones. Only log in as the root user to perform AWS tasks that require root user. All other actions should be performed by other users assuming appropriate roles.
- Periodically check that access to the root user is functioning so that procedures are tested prior to an emergency situation requiring the use of the root user credentials.
- Periodically check that the email address associated with the account and those listed under Alternate Contacts work. Monitor these email inboxes for security notifications you might receive

from <abuse@amazon.com>. Also ensure any phone numbers associated with the account are working.

- Prepare incident response procedures to respond to root account misuse. Refer to the AWS Security Incident Response Guide and the best practices in the Incident Response section of the Security Pillar whitepaper for more information on building an incident response strategy for your AWS account.

Resources

Related best practices:

- SEC01-BP01 Separate workloads using accounts
- SEC02-BP01 Use strong sign-in mechanisms
- SEC03-BP02 Grant least privilege access
- SEC03-BP03 Establish emergency access process
- SEC10-BP05 Pre-provision access

Related documents:

- AWS Control Tower
- AWS Security Audit Guidelines
- IAM Best Practices
- Amazon GuardDuty – root credential usage alert
- Step-by-step guidance on monitoring for root credential use through CloudTrail
- MFA tokens approved for use with AWS
- Implementing break glass access on AWS
- Top 10 security items to improve in your AWS account
- What do I do if I notice unauthorized activity in my AWS account?

Related videos:

- Enable AWS adoption at scale with automation and governance
- Security Best Practices the Well-Architected Way

- Limiting use of AWS root credentials from AWS re:inforce 2022 – Security best practices with AWS IAM

Related examples and labs:

- Lab: AWS account setup and root user

SEC01-BP03 Identify and validate control objectives

Based on your compliance requirements and risks identified from your threat model, derive and validate the control objectives and controls that you need to apply to your workload. Ongoing validation of control objectives and controls help you measure the effectiveness of risk mitigation.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Identify compliance requirements: Discover the organizational, legal, and compliance requirements that your workload must comply with.
- Identify AWS compliance resources: Identify resources that AWS has available to assist you with compliance.
  - https://aws.amazon.com/compliance/
  - https://aws.amazon.com/artifact/

Resources

Related documents:

- AWS Security Audit Guidelines
- Security Bulletins

Related videos:

- AWS Security Hub CSPM: Manage Security Alerts and Automate Compliance
- Security Best Practices the Well-Architected Way

SEC01-BP04 Keep up-to-date with security threats

To help you define and implement appropriate controls, recognize attack vectors by staying up to date with the latest security threats. Consume AWS Managed Services to make it easier to receive notification of unexpected or unusual behavior in your AWS accounts. Investigate using AWS Partner tools or third-party threat information feeds as part of your security information flow. The Common Vulnerabilities and Exposures (CVE) List list contains publicly disclosed cyber security vulnerabilities that you can use to stay up to date.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Subscribe to threat intelligence sources: Regularly review threat intelligence information from multiple sources that are relevant to the technologies used in your workload.
  - Common Vulnerabilities and Exposures List
- Consider AWS Shield Advanced service: It provides near real-time visibility into intelligence sources, if your workload is internet accessible.

Resources

Related documents:

- AWS Security Audit Guidelines
- AWS Shield
- Security Bulletins

Related videos:

- Security Best Practices the Well-Architected Way

SEC01-BP05 Keep up-to-date with security recommendations

Stay up-to-date with both AWS and industry security recommendations to evolve the security posture of your workload. AWS Security Bulletins contain important information about security and privacy notifications.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Follow AWS updates: Subscribe or regularly check for new recommendations, tips and tricks.
  - AWS Well-Architected Labs
  - AWS security blog
  - AWS service documentation
- Subscribe to industry news: Regularly review news feeds from multiple sources that are relevant to the technologies that are used in your workload.
  - Example: Common Vulnerabilities and Exposures List

Resources

Related documents:

- Security Bulletins

Related videos:

- Security Best Practices the Well-Architected Way

SEC01-BP06 Automate testing and validation of security controls in pipelines

Establish secure baselines and templates for security mechanisms that are tested and validated as part of your build, pipelines, and processes. Use tools and automation to test and validate all security controls continuously. For example, scan items such as machine images and infrastructure-as-code templates for security vulnerabilities, irregularities, and drift from an established baseline at each stage. AWS CloudFormation Guard can help you verify that CloudFormation templates are safe, save you time, and reduce the risk of configuration error.

Reducing the number of security misconfigurations introduced into a production environment is critical—the more quality control and reduction of defects you can perform in the build process, the better. Design continuous integration and continuous deployment (CI/CD) pipelines to test for security issues whenever possible. CI/CD pipelines offer the opportunity to enhance security at each stage of build and delivery. CI/CD security tooling must also be kept updated to mitigate evolving threats.

Track changes to your workload configuration to help with compliance auditing, change management, and investigations that may apply to you. You can use AWS Config to record and evaluate your AWS and third-party resources. It allows you to continuously audit and assess the overall compliance with rules and conformance packs, which are collections of rules with remediation actions.

Change tracking should include planned changes, which are part of your organization's change control process (sometimes referred to as MACD—Move, Add, Change, Delete), unplanned changes, and unexpected changes, such as incidents. Changes might occur on the infrastructure, but they might also be related to other categories, such as changes in code repositories, machine images and application inventory changes, process and policy changes, or documentation changes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Automate configuration management: Enforce and validate secure configurations automatically by using a configuration management service or tool.
  - AWS Systems Manager
  - AWS CloudFormation
  - Set Up a CI/CD Pipeline on AWS

Resources

Related documents:

- How to use service control policies to set permission guardrails across accounts in your AWS Organization

Related videos:

- Managing Multi-Account AWS Environments Using AWS Organizations
- Security Best Practices the Well-Architected Way

SEC01-BP07 Identify threats and prioritize mitigations using a threat model

Perform threat modeling to identify and maintain an up-to-date register of potential threats and associated mitigations for your workload. Prioritize your threats and adapt your security

control mitigations to prevent, detect, and respond. Revisit and maintain this in the context of your workload, and the evolving security landscape.

Level of risk exposed if this best practice is not established: High

Implementation guidance

What is threat modeling?

As a definition, “Threat modeling works to identify, communicate, and understand threats and mitigations within the context of protecting something of value.” – The Open Web Application Security Project (OWASP) Application Threat Modeling

Why should you threat model?

Systems are complex, and are becoming increasingly more complex and capable over time, delivering more business value and increased customer satisfaction and engagement. This means that IT design decisions need to account for an ever-increasing number of use cases. This complexity and number of use-case permutations typically makes unstructured approaches ineffective for finding and mitigating threats. Instead, you need a systematic approach to enumerate the potential threats to the system, and to devise mitigations and prioritize these mitigations to make sure that the limited resources of your organization have the maximum impact in improving the overall security posture of the system.

Threat modeling is designed to provide this systematic approach, with the aim of finding and addressing issues early in the design process, when the mitigations have a low relative cost and effort compared to later in the lifecycle. This approach aligns with the industry principle of “shift-left” security. Ultimately, threat modeling integrates with an organization’s risk management process and helps drive decisions on which controls to implement by using a threat driven approach.

When should threat modeling be performed?

Start threat modeling as early as possible in the lifecycle of your workload, this gives you better flexibility on what to do with the threats you have identified. Much like software bugs, the earlier you identify threats, the more cost effective it is to address them. A threat model is a living document and should continue to evolve as your workloads change. Revisit your threat models over time, including when there is a major change, a change in the threat landscape, or when you adopt a new feature or service.

Implementation steps

How can we perform threat modeling?

There are many different ways to perform threat modeling. Much like programming languages, there are advantages and disadvantages to each, and you should choose the way that works best for you. One approach is to start with Shostack's 4 Question Frame for Threat Modeling, which poses open-ended questions to provide structure to your threat modeling exercise:

1. **What are working on?**

   The purpose of this question is to help you understand and agree upon the system you are building and the details about that system that are relevant to security. Creating a model or diagram is the most popular way to answer this question, as it helps you to visualize what you are building, for example, using a data flow diagram. Writing down assumptions and important details about your system also helps you define what is in scope. This allows everyone contributing to the threat model to focus on the same thing, and avoid time-consuming detours into out-of-scope topics (including out of date versions of your system). For example, if you are building a web application, it is probably not worth your time threat modeling the operating system trusted boot sequence for browser clients, as you have no ability to affect this with your design.

2. **What can go wrong?**

   This is where you identify threats to your system. Threats are accidental or intentional actions or events that have unwanted impacts and could affect the security of your system. Without a clear understanding of what could go wrong, you have no way of doing anything about it.

   There is no canonical list of what can go wrong. Creating this list requires brainstorming and collaboration between all of the individuals within your team and relevant personas involved in the threat modeling exercise. You can aid your brainstorming by using a model for identifying threats, such as STRIDE, which suggests different categories to evaluate: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of privilege. In addition, you might want to aid the brainstorming by reviewing existing lists and research for inspiration, including the OWASP Top 10, HiTrust Threat Catalog, and your organization's own threat catalog.

3. **What are we going to do about it?**

   As was the case with the previous question, there is no canonical list of all possible mitigations. The inputs into this step are the identified threats, actors, and areas of improvement from the previous step.

Security and compliance is a shared responsibility between you and AWS. It's important to understand that when you ask "What are we going to do about it?", that you are also asking "Who is responsible for doing something about it?". Understanding the balance of responsibilities between you and AWS helps you scope your threat modeling exercise to the mitigations that are under your control, which are typically a combination of AWS service configuration options and your own system-specific mitigations.

For the AWS portion of the shared responsibility, you will find that AWS services are in-scope of many compliance programs. These programs help you to understand the robust controls in place at AWS to maintain security and compliance of the cloud. The audit reports from these programs are available for download for AWS customers from AWS Artifact.

Regardless of which AWS services you are using, there's always an element of customer responsibility, and mitigations aligned to these responsibilities should be included in your threat model. For security control mitigations for the AWS services themselves, you want to consider implementing security controls across domains, including domains such as identity and access management (authentication and authorization), data protection (at rest and in transit), infrastructure security, logging, and monitoring. The documentation for each AWS service has a dedicated security chapter that provides guidance on the security controls to consider as mitigations. Importantly, consider the code that you are writing and its code dependencies, and think about the controls that you could put in place to address those threats. These controls could be things such as input validation, session handling, and bounds handling. Often, the majority of vulnerabilities are introduced in custom code, so focus on this area.

4. **Did we do a good job?**

   The aim is for your team and organization to improve both the quality of threat models and the velocity at which you are performing threat modeling over time. These improvements come from a combination of practice, learning, teaching, and reviewing. To go deeper and get hands on, it's recommended that you and your team complete the Threat modeling the right way for builders training course or workshop. In addition, if you are looking for guidance on how to integrate threat modeling into your organization's application development lifecycle, see How to approach threat modeling post on the AWS Security Blog.

Resources

Related best practices:

- SEC01-BP03 Identify and validate control objectives
- SEC01-BP04 Keep up-to-date with security threats
- SEC01-BP05 Keep up-to-date with security recommendations
- SEC01-BP08 Evaluate and implement new security services and features regularly

Related documents:

- How to approach threat modeling (AWS Security Blog)
- NIST: Guide to Data-Centric System Threat Modelling

Related videos:

- AWS Summit ANZ 2021 - How to approach threat modelling
- AWS Summit ANZ 2022 - Scaling security – Optimise for fast and secure delivery

Related training:

- Threat modeling the right way for builders – AWS Skill Builder virtual self-paced training
- Threat modeling the right way for builders – AWS Workshop

SEC01-BP08 Evaluate and implement new security services and features regularly

Evaluate and implement security services and features from AWS and AWS Partners that allow you to evolve the security posture of your workload. The AWS Security Blog highlights new AWS services and features, implementation guides, and general security guidance. What's New with AWS? is a great way to stay up to date with all new AWS features, services, and announcements.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Plan regular reviews: Create a calendar of review activities that includes compliance requirements, evaluation of new AWS security features and services, and staying up-to-date with industry news.
- Discover AWS services and features: Discover the security features that are available for the services that you are using, and review new features as they are released.

  - AWS security blog
  - AWS security bulletins
  - AWS service documentation
- Define AWS service on-boarding process: Define processes for onboarding of new AWS services. Include how you evaluate new AWS services for functionality, and the compliance requirements for your workload.
- Test new services and features: Test new services and features as they are released in a non-production environment that closely replicates your production one.
- Implement other defense mechanisms: Implement automated mechanisms to defend your workload, explore the options available.
  - Remediating non-compliant AWS resources by AWS Config Rules

Resources

Related videos:

- Security Best Practices the Well-Architected Way

Identity and access management

Questions

- SEC 2. How do you manage authentication for people and machines?
- SEC 3. How do you manage permissions for people and machines?

SEC 2. How do you manage authentication for people and machines?

There are two types of identities that you must manage when approaching operating secure AWS workloads. Understanding the type of identity you must manage and grant access helps you verify the right identities have access to the right resources under the right conditions.

Human Identities: Your administrators, developers, operators, and end users require an identity to access your AWS environments and applications. These are members of your organization, or external users with whom you collaborate, and who interact with your AWS resources via a web browser, client application, or interactive command line tools.

Machine Identities: Your service applications, operational tools, and workloads require an identity to make requests to AWS services, for example, to read data. These identities include machines running in your AWS environment such as Amazon EC2 instances or AWS Lambda functions. You may also manage machine identities for external parties who need access. Additionally, you may also have machines outside of AWS that need access to your AWS environment.

Best practices

- SEC02-BP01 Use strong sign-in mechanisms
- SEC02-BP02 Use temporary credentials
- SEC02-BP03 Store and use secrets securely
- SEC02-BP04 Rely on a centralized identity provider
- SEC02-BP05 Audit and rotate credentials periodically
- SEC02-BP06 Leverage user groups and attributes

SEC02-BP01 Use strong sign-in mechanisms

Sign-ins (authentication using sign-in credentials) can present risks when not using mechanisms like multi-factor authentication (MFA), especially in situations where sign-in credentials have been inadvertently disclosed or are easily guessed. Use strong sign-in mechanisms to reduce these risks by requiring MFA and strong password policies.

Desired outcome: Reduce the risks of unintended access to credentials in AWS by using strong sign-in mechanisms for AWS Identity and Access Management (IAM) users, the AWS account root user, AWS IAM Identity Center (successor to AWS Single Sign-On), and third-party identity providers. This means requiring MFA, enforcing strong password policies, and detecting anomalous login behavior.

Common anti-patterns:

- Not enforcing a strong password policy for your identities including complex passwords and MFA.
- Sharing the same credentials among different users.
- Not using detective controls for suspicious sign-ins.

Level of risk exposed if this best practice is not established: High

Implementation guidance

There are many ways for human identities to sign-in to AWS. It is an AWS best practice to rely on a centralized identity provider using federation (direct federation or using AWS IAM Identity Center) when authenticating to AWS. In that case, you should establish a secure sign-in process with your identity provider or Microsoft Active Directory.

When you first open an AWS account, you begin with an AWS account root user. You should only use the account root user to set up access for your users (and for tasks that require the root user). It's important to turn on MFA for the account root user immediately after opening your AWS account and to secure the root user using the AWS best practice guide.

If you create users in AWS IAM Identity Center, then secure the sign-in process in that service. For consumer identities, you can use Amazon Cognito user pools and secure the sign-in process in that service, or by using one of the identity providers that Amazon Cognito user pools supports.

If you are using AWS Identity and Access Management (IAM) users, you would secure the sign-in process using IAM.

Regardless of the sign-in method, it's critical to enforce a strong sign-in policy.

Implementation steps

The following are general strong sign-in recommendations. The actual settings you configure should be set by your company policy or use a standard like NIST 800-63.

- Require MFA. It's an IAM best practice to require MFA for human identities and workloads. Turning on MFA provides an additional layer of security requiring that users provide sign-in credentials and a one-time password (OTP) or a cryptographically verified and generated string from a hardware device.
- Enforce a minimum password length, which is a primary factor in password strength.
- Enforce password complexity to make passwords more difficult to guess.
- Allow users to change their own passwords.
- Create individual identities instead of shared credentials. By creating individual identities, you can give each user a unique set of security credentials. Individual users provide the ability to audit each user's activity.

IAM Identity Center recommendations:

- IAM Identity Center provides a predefined password policy when using the default directory that establishes password length, complexity, and reuse requirements.
- Turn on MFA and configure the context-aware or always-on setting for MFA when the identity source is the default directory, AWS Managed Microsoft AD, or AD Connector.
- Allow users to register their own MFA devices.

Amazon Cognito user pools directory recommendations:

- Configure the Password strength settings.
- Require MFA for users.
- Use the Amazon Cognito user pools advanced security settings for features like adaptive authentication which can block suspicious sign-ins.

IAM user recommendations:

- Ideally you are using IAM Identity Center or direct federation. However, you might have the need for IAM users. In that case, set a password policy for IAM users. You can use the password policy to define requirements such as minimum length or whether the password requires non-alphabetic characters.
- Create an IAM policy to enforce MFA sign-in so that users are allowed to manage their own passwords and MFA devices.

Resources

Related best practices:

- SEC02-BP03 Store and use secrets securely
- SEC02-BP04 Rely on a centralized identity provider
- SEC03-BP08 Share resources securely within your organization

Related documents:

- AWS IAM Identity Center Password Policy
- IAM user password policy
- Setting the AWS account root user password

- Amazon Cognito password policy
- AWS credentials
- IAM security best practices

Related videos:

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP02 Use temporary credentials

When doing any type of authentication, it's best to use temporary credentials instead of long-term credentials to reduce or eliminate risks, such as credentials being inadvertently disclosed, shared, or stolen.

Desired outcome: To reduce the risk of long-term credentials, use temporary credentials wherever possible for both human and machine identities. Long-term credentials create many risks, for example, they can be uploaded in code to public GitHub repositories. By using temporary credentials, you significantly reduce the chances of credentials becoming compromised.

Common anti-patterns:

- Developers using long-term access keys from IAM users rather than obtaining temporary credentials from the CLI using federation.
- Developers embedding long-term access keys in their code and uploading that code to public Git repositories.
- Developers embedding long-term access keys in mobile apps that are then made available in app stores.
- Users sharing long-term access keys with other users, or employees leaving the company with long-term access keys still in their possession.
- Using long-term access keys for machine identities when temporary credentials could be used.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Use temporary security credentials instead of long-term credentials for all AWS API and CLI requests. API and CLI requests to AWS services must, in nearly every case, be signed using AWS access keys. These requests can be signed with either temporary or long-term credentials. The only time you should use long-term credentials, also known as long-term access keys, is if you are using an IAM user or the AWS account root user. When you federate to AWS or assume an IAM role through other methods, temporary credentials are generated. Even when you access the AWS Management Console using sign-in credentials, temporary credentials are generated for you to make calls to AWS services. There are few situations where you need long-term credentials and you can accomplish nearly all tasks using temporary credentials.

Avoiding the use of long-term credentials in favor of temporary credentials should go hand in hand with a strategy of reducing the usage of IAM users in favor of federation and IAM roles. While IAM users have been used for both human and machine identities in the past, we now recommend not using them to avoid the risks in using long-term access keys.

Implementation steps

For human identities like employees, administrators, developers, operators, and customers:

- You should rely on a centralized identity provider and require human users to use federation with an identity provider to access AWS using temporary credentials. Federation for your users can be done either with direct federation to each AWS account or using AWS IAM Identity Center (successor to AWS IAM Identity Center) and the identity provider of your choice. Federation provides a number of advantages over using IAM users in addition to eliminating long-term credentials. Your users can also request temporary credentials from the command line for direct federation or by using IAM Identity Center. This means that there are few uses cases that require IAM users or long-term credentials for your users.
- When granting third parties, such as software as a service (SaaS) providers, access to resources in your AWS account, you can use cross-account roles and resource-based policies.
- If you need to grant applications for consumers or customers access to your AWS resources, you can use Amazon Cognito identity pools or Amazon Cognito user pools to provide temporary credentials. The permissions for the credentials are configured through IAM roles. You can also define a separate IAM role with limited permissions for guest users who are not authenticated.

For machine identities, you might need to use long-term credentials. In these cases, you should require workloads to use temporary credentials with IAM roles to access AWS.

- For Amazon Elastic Compute Cloud (Amazon EC2), you can use roles for Amazon EC2.

- AWS Lambda allows you to configure a Lambda execution role to grant the service permissions to perform AWS actions using temporary credentials. There are many other similar models for AWS services to grant temporary credentials using IAM roles.
- For IoT devices, you can use the AWS IoT Core credential provider to request temporary credentials.
- For on-premises systems or systems that run outside of AWS that need access to AWS resources, you can use IAM Roles Anywhere.

There are scenarios where temporary credentials are not an option and you might need to use long-term credentials. In these situations, audit and rotate credentials periodically and rotate access keys regularly for use cases that require long-term credentials. Some examples that might require long-term credentials include WordPress plugins and third-party AWS clients. In situations where you must use long-term credentials, or for credentials other than AWS access keys, such as database logins, you can use a service that is designed to handle the management of secrets, such as AWS Secrets Manager. Secrets Manager makes it simple to manage, rotate, and securely store encrypted secrets using supported services. For more information about rotating long-term credentials, see rotating access keys.

Resources

Related best practices:

- SEC02-BP03 Store and use secrets securely
- SEC02-BP04 Rely on a centralized identity provider
- SEC03-BP08 Share resources securely within your organization

Related documents:

- Temporary Security Credentials
- AWS Credentials
- IAM Security Best Practices
- IAM Roles
- IAM Identity Center

- Identity Providers and Federation
- Rotating Access Keys
- Security Partner Solutions: Access and Access Control
- The AWS Account Root User

Related videos:

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP03 Store and use secrets securely

A workload requires an automated capability to prove its identity to databases, resources, and third-party services. This is accomplished using secret access credentials, such as API access keys, passwords, and OAuth tokens. Using a purpose-built service to store, manage, and rotate these credentials helps reduce the likelihood that those credentials become compromised.

Desired outcome: Implementing a mechanism for securely managing application credentials that achieves the following goals:

- Identifying what secrets are required for the workload.
- Reducing the number of long-term credentials required by replacing them with short-term credentials when possible.
- Establishing secure storage and automated rotation of remaining long-term credentials.
- Auditing access to secrets that exist in the workload.
- Continual monitoring to verify that no secrets are embedded in source code during the development process.
- Reduce the likelihood of credentials being inadvertently disclosed.

Common anti-patterns:

- Not rotating credentials.
- Storing long-term credentials in source code or configuration files.
- Storing credentials at rest unencrypted.

Benefits of establishing this best practice:

- Secrets are stored encrypted at rest and in transit.
- Access to credentials is gated through an API (think of it as a *credential vending machine*).
- Access to a credential (both read and write) is audited and logged.
- Separation of concerns: credential rotation is performed by a separate component, which can be segregated from the rest of the architecture.
- Secrets are automatically distributed on-demand to software components and rotation occurs in a central location.
- Access to credentials can be controlled in a fine-grained manner.

Level of risk exposed if this best practice is not established: High

Implementation guidance

In the past, credentials used to authenticate to databases, third-party APIs, tokens, and other secrets might have been embedded in source code or in environment files. AWS provides several mechanisms to store these credentials securely, automatically rotate them, and audit their usage.

The best way to approach secrets management is to follow the guidance of remove, replace, and rotate. The most secure credential is one that you do not have to store, manage, or handle. There might be credentials that are no longer necessary to the functioning of the workload that can be safely removed.

For credentials that are still required for the proper functioning of the workload, there might be an opportunity to replace a long-term credential with a temporary or short-term credential. For example, instead of hard-coding an AWS secret access key, consider replacing that long-term credential with a temporary credential using IAM roles.

Some long-lived secrets might not be able to be removed or replaced. These secrets can be stored in a service such as AWS Secrets Manager, where they can be centrally stored, managed, and rotated on a regular basis.

An audit of the workload's source code and configuration files can reveal many types of credentials. The following table summarizes strategies for handling common types of credentials:

Credential type	Description	Suggested strategy
IAM access keys	AWS IAM access and secret keys used to assume IAM roles inside of a workload	Replace: Use IAM roles assigned to the compute instances (such as Amazon EC2 or AWS Lambda) instead. For interoperability with third parties that require access to resources in your AWS account, ask if they support AWS cross-account access. For mobile apps, consider using temporary credentia ls through Amazon Cognito identity pools (federated identities). For workloads running outside of AWS, consider IAM Roles Anywhere or AWS Systems Manager Hybrid Activations.
SSH keys	Secure Shell private keys used to log into Linux EC2 instances, manually or as part of an automated process	Replace: Use AWS Systems Manager or EC2 Instance Connect to provide programmatic and human access to EC2 instances using IAM roles.
Application and database credentials	Passwords – plain text string	Rotate: Store credentials in AWS Secrets Manager and establish automated rotation if possible.
Amazon RDS and Aurora Admin Database credentials	Passwords – plain text string	Replace: Use the Secrets Manager integration with Amazon RDS or Amazon Aurora. In addition, some RDS

Credential type	Description	Suggested strategy
		database types can use IAM roles instead of passwords for some use cases (for more detail, see IAM database authentication).
OAuth tokens	Secret tokens – plain text string	Rotate: Store tokens in AWS Secrets Manager and configure automated rotation.
API tokens and keys	Secret tokens – plain text string	Rotate: Store in AWS Secrets Manager and establish automated rotation if possible.

A common anti-pattern is embedding IAM access keys inside source code, configuration files, or mobile apps. When an IAM access key is required to communicate with an AWS service, use temporary (short-term) security credentials. These short-term credentials can be provided through IAM roles for EC2 instances, execution roles for Lambda functions, Cognito IAM roles for mobile user access, and IoT Core policies for IoT devices. When interfacing with third parties, prefer delegating access to an IAM role with the necessary access to your account's resources rather than configuring an IAM user and sending the third party the secret access key for that user.

There are many cases where the workload requires the storage of secrets necessary to interoperate with other services and resources. AWS Secrets Manager is purpose built to securely manage these credentials, as well as the storage, use, and rotation of API tokens, passwords, and other credentials.

AWS Secrets Manager provides five key capabilities to ensure the secure storage and handling of sensitive credentials: encryption at rest, encryption in transit, comprehensive auditing, fine-grained access control, and extensible credential rotation. Other secret management services from AWS Partners or locally developed solutions that provide similar capabilities and assurances are also acceptable.

Implementation steps

1. Identify code paths containing hard-coded credentials using automated tools such as Amazon CodeGuru.
   a. Use Amazon CodeGuru to scan your code repositories. Once the review is complete, filter on `Type=Secrets` in CodeGuru to find problematic lines of code.
2. Identify credentials that can be removed or replaced.
   a. Identify credentials no longer needed and mark for removal.
   b. For AWS Secret Keys that are embedded in source code, replace them with IAM roles associated with the necessary resources. If part of your workload is outside AWS but requires IAM credentials to access AWS resources, consider IAM Roles Anywhere or AWS Systems Manager Hybrid Activations.
3. For other third-party, long-lived secrets that require the use of the rotate strategy, integrate Secrets Manager into your code to retrieve third-party secrets at runtime.
   a. The CodeGuru console can automatically create a secret in Secrets Manager using the discovered credentials.
   b. Integrate secret retrieval from Secrets Manager into your application code.
      i. Serverless Lambda functions can use a language-agnostic Lambda extension.
      ii. For EC2 instances or containers, AWS provides example client-side code for retrieving secrets from Secrets Manager in several popular programming languages.
4. Periodically review your code base and re-scan to verify no new secrets have been added to the code.
   a. Consider using a tool such as git-secrets to prevent committing new secrets to your source code repository.
5. Monitor Secrets Manager activity for indications of unexpected usage, inappropriate secret access, or attempts to delete secrets.
6. Reduce human exposure to credentials. Restrict access to read, write, and modify credentials to an IAM role dedicated for this purpose, and only provide access to assume the role to a small subset of operational users.

Resources

Related best practices:

- SEC02-BP02 Use temporary credentials
- SEC02-BP05 Audit and rotate credentials periodically

Related documents:

- Getting Started with AWS Secrets Manager
- Identity Providers and Federation
- Amazon CodeGuru Introduces Secrets Detector
- How AWS Secrets Manager uses AWS Key Management Service
- Secret encryption and decryption in Secrets Manager
- Secrets Manager blog entries
- Amazon RDS announces integration with AWS Secrets Manager

Related videos:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale
- Find Hard-Coded Secrets Using Amazon CodeGuru Secrets Detector
- Securing Secrets for Hybrid Workloads Using AWS Secrets Manager

Related workshops:

- Store, retrieve, and manage sensitive credentials in AWS Secrets Manager
- AWS Systems Manager Hybrid Activations

SEC02-BP04 Rely on a centralized identity provider

For workforce identities, rely on an identity provider that allows you to manage identities in a centralized place. This makes it easier to manage access across multiple applications and services, because you are creating, managing, and revoking access from a single location. For example, if someone leaves your organization, you can revoke access for all applications and services (including AWS) from one location. This reduces the need for multiple credentials and provides an opportunity to integrate with existing human resources (HR) processes.

For federation with individual AWS accounts, you can use centralized identities for AWS with a SAML 2.0-based provider with AWS Identity and Access Management. You can use any provider — whether hosted by you in AWS, external to AWS, or supplied by the AWS Partner—that is compatible with the SAML 2.0 protocol. You can use federation between your AWS account and your chosen provider to grant a user or application access to call AWS API operations by using a

SAML assertion to get temporary security credentials. Web-based single sign-on is also supported, allowing users to sign in to the AWS Management Console from your sign in website.

For federation to multiple accounts in your AWS Organizations, you can configure your identity source in AWS IAM Identity Center (IAM Identity Center), and specify where your users and groups are stored. Once configured, your identity provider is your source of truth, and information can be synchronized using the System for Cross-domain Identity Management (SCIM) v2.0 protocol. You can then look up users or groups and grant them IAM Identity Center access to AWS accounts, cloud applications, or both.

IAM Identity Center integrates with AWS Organizations, which allows you to configure your identity provider once and then grant access to existing and new accounts managed in your organization. IAM Identity Center provides you with a default store, which you can use to manage your users and groups. If you choose to use the IAM Identity Center store, create your users and groups and assign their level of access to your AWS accounts and applications, keeping in mind the best practice of least privilege. Alternatively, you can choose to Connect to Your External Identity Provider using SAML 2.0, or Connect to Your Microsoft AD Directory using AWS Directory Service. Once configured, you can sign into the AWS Management Console, or the AWS mobile app, by authenticating through your central identity provider.

For managing end-users or consumers of your workloads, such as a mobile app, you can use Amazon Cognito. It provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with sign-in credentials, or through a third party, such as Amazon, Apple, Facebook, or Google.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Centralize administrative access: Create an Identity and Access Management (IAM) identity provider entity to establish a trusted relationship between your AWS account and your identity provider (IdP). IAM supports IdPs that are compatible with OpenID Connect (OIDC) or SAML 2.0 (Security Assertion Markup Language 2.0).
  - Identity Providers and Federation
- Centralize application access: Consider Amazon Cognito for centralizing application access. It lets you add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily. Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Facebook, Google, and Amazon, and enterprise identity providers via SAML 2.0.

- Remove old users and groups: After you start using an identity provider (IdP), remove users and groups that are no longer required.
  - Finding unused credentials
  - Deleting an IAM group

Resources

Related documents:

- IAM Best Practices
- Security Partner Solutions: Access and Access Control
- Temporary Security Credentials
- The AWS Account Root User

Related videos:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale
- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP05 Audit and rotate credentials periodically

Audit and rotate credentials periodically to limit how long the credentials can be used to access your resources. Long-term credentials create many risks, and these risks can be reduced by rotating long-term credentials regularly.

Desired outcome: Implement credential rotation to help reduce the risks associated with long-term credential usage. Regularly audit and remediate non-compliance with credential rotation policies.

Common anti-patterns:

- Not auditing credential use.
- Using long-term credentials unnecessarily.
- Using long-term credentials and not rotating them regularly.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

When you cannot rely on temporary credentials and require long-term credentials, audit credentials to verify that defined controls like multi-factor authentication (MFA) are enforced, rotated regularly, and have the appropriate access level.

Periodic validation, preferably through an automated tool, is necessary to verify that the correct controls are enforced. For human identities, you should require users to change their passwords periodically and retire access keys in favor of temporary credentials. As you move from AWS Identity and Access Management (IAM) users to centralized identities, you can generate a credential report to audit your users.

We also recommend that you enforce and monitor MFA in your identity provider. You can set up AWS Config Rules, or use AWS Security Hub CSPM Security Standards, to monitor if users have configured MFA. Consider using IAM Roles Anywhere to provide temporary credentials for machine identities. In situations when using IAM roles and temporary credentials is not possible, frequent auditing and rotating access keys is necessary.

Implementation steps

- **Regularly audit credentials:** Auditing the identities that are configured in your identity provider and IAM helps verify that only authorized identities have access to your workload. Such identities can include, but are not limited to, IAM users, AWS IAM Identity Center users, Active Directory users, or users in a different upstream identity provider. For example, remove people that leave the organization, and remove cross-account roles that are no longer required. Have a process in place to periodically audit permissions to the services accessed by an IAM entity. This helps you identify the policies you need to modify to remove any unused permissions. Use credential reports and AWS Identity and Access Management Access Analyzer to audit IAM credentials and permissions. You can use Amazon CloudWatch to set up alarms for specific API calls called within your AWS environment. Amazon GuardDuty can also alert you to unexpected activity, which might indicate overly permissive access or unintended access to IAM credentials.
- **Rotate credentials regularly:** When you are unable to use temporary credentials, rotate long-term IAM access keys regularly (maximum every 90 days). If an access key is unintentionally disclosed without your knowledge, this limits how long the credentials can be used to access your resources. For information about rotating access keys for IAM users, see Rotating access keys.

- **Review IAM permissions:** To improve the security of your AWS account, regularly review and monitor each of your IAM policies. Verify that policies adhere to the principle of least privilege.
- **Consider automating IAM resource creation and updates:** IAM Identity Center automates many IAM tasks, such as role and policy management. Alternatively, AWS CloudFormation can be used to automate the deployment of IAM resources, including roles and policies, to reduce the chance of human error because the templates can be verified and version controlled.
- **Use IAM Roles Anywhere to replace IAM users for machine identities:** IAM Roles Anywhere allows you to use roles in areas that you traditionally could not, such as on-premise servers. IAM Roles Anywhere uses a trusted X.509 certificate to authenticate to AWS and receive temporary credentials. Using IAM Roles Anywhere avoids the need to rotate these credentials, as long-term credentials are no longer stored in your on-premises environment. Please note that you will need to monitor and rotate the X.509 certificate as it approaches expiration.

Resources

Related best practices:

- SEC02-BP02 Use temporary credentials
- SEC02-BP03 Store and use secrets securely

Related documents:

- Getting Started with AWS Secrets Manager
- IAM Best Practices
- Identity Providers and Federation
- Security Partner Solutions: Access and Access Control
- Temporary Security Credentials
- Getting credential reports for your AWS account

Related videos:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale
- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

Related examples:

- Well-Architected Lab - Automated IAM User Cleanup
- Well-Architected Lab - Automated Deployment of IAM Groups and Roles

SEC02-BP06 Leverage user groups and attributes

As the number of users you manage grows, you will need to determine ways to organize them so that you can manage them at scale. Place users with common security requirements in groups defined by your identity provider, and put mechanisms in place to ensure that user attributes that may be used for access control (for example, department or location) are correct and updated. Use these groups and attributes to control access, rather than individual users. This allows you to manage access centrally by changing a user's group membership or attributes once with a permission set, rather than updating many individual policies when a user's access needs change.

You can use AWS IAM Identity Center (IAM Identity Center) to manage user groups and attributes. IAM Identity Center supports most commonly used attributes whether they are entered manually during user creation or automatically provisioned using a synchronization engine, such as defined in the System for Cross-Domain Identity Management (SCIM) specification.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- If you are using AWS IAM Identity Center (IAM Identity Center), configure groups: IAM Identity Center provides you with the ability to configure groups of users, and assign groups the desired level of permission.
  - AWS Single Sign-On - Manage Identities
- Learn about attribute-based access control (ABAC): ABAC is an authorization strategy that defines permissions based on attributes.
  - What Is ABAC for AWS?
  - Lab: IAM Tag Based Access Control for EC2

Resources

Related documents:

- Getting Started with AWS Secrets Manager

- IAM Best Practices
- Identity Providers and Federation
- The AWS Account Root User

Related videos:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale
- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

Related examples:

- Lab: IAM Tag Based Access Control for EC2

SEC 3. How do you manage permissions for people and machines?

Manage permissions to control access to people and machine identities that require access to AWS and your workload. Permissions control who can access what, and under what conditions.

Best practices

- SEC03-BP01 Define access requirements
- SEC03-BP02 Grant least privilege access
- SEC03-BP03 Establish emergency access process
- SEC03-BP04 Reduce permissions continuously
- SEC03-BP05 Define permission guardrails for your organization
- SEC03-BP06 Manage access based on lifecycle
- SEC03-BP07 Analyze public and cross-account access
- SEC03-BP08 Share resources securely within your organization
- SEC03-BP09 Share resources securely with a third party

SEC03-BP01 Define access requirements

Each component or resource of your workload needs to be accessed by administrators, end users, or other components. Have a clear definition of who or what should have access to each component, choose the appropriate identity type and method of authentication and authorization.

Common anti-patterns:

- Hard-coding or storing secrets in your application.
- Granting custom permissions for each user.
- Using long-lived credentials.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Each component or resource of your workload needs to be accessed by administrators, end users, or other components. Have a clear definition of who or what should have access to each component, choose the appropriate identity type and method of authentication and authorization.

Regular access to AWS accounts within the organization should be provided using federated access or a centralized identity provider. You should also centralize your identity management and ensure that there is an established practice to integrate AWS access to your employee access lifecycle. For example, when an employee changes to a job role with a different access level, their group membership should also change to reflect their new access requirements.

When defining access requirements for non-human identities, determine which applications and components need access and how permissions are granted. Using IAM roles built with the least privilege access model is a recommended approach. AWS Managed policies provide predefined IAM policies that cover most common use cases.

AWS services, such as AWS Secrets Manager and AWS Systems Manager Parameter Store, can help decouple secrets from the application or workload securely in cases where it's not feasible to use IAM roles. In Secrets Manager, you can establish automatic rotation for your credentials. You can use Systems Manager to reference parameters in your scripts, commands, SSM documents, configuration, and automation workflows by using the unique name that you specified when you created the parameter.

You can use AWS Identity and Access Management Roles Anywhere to obtain temporary security credentials in IAM for workloads that run outside of AWS. Your workloads can use the same IAM policies and IAM roles that you use with AWS applications to access AWS resources.

Where possible, prefer short-term temporary credentials over long-term static credentials. For scenarios in which you need users with programmatic access and long-term credentials, use access key last used information to rotate and remove access keys.

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS*

Which user needs programmatic access?	To	By
		Command Line Interface User Guide. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

Resources

Related documents:

- Attribute-based access control (ABAC)
- AWS IAM Identity Center
- IAM Roles Anywhere
- AWS Managed policies for IAM Identity Center
- AWS IAM policy conditions
- IAM use cases
- Remove unnecessary credentials
- Working with Policies
- How to control access to AWS resources based on AWS account, OU, or organization
- Identify, arrange, and manage secrets easily using enhanced search in AWS Secrets Manager

Related videos:

- Become an IAM Policy Master in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, and CI/CD
- Streamlining identity and access management for innovation

SEC03-BP02 Grant least privilege access

It's a best practice to grant only the access that identities require to perform specific actions on specific resources under specific conditions. Use group and identity attributes to dynamically set permissions at scale, rather than defining permissions for individual users. For example, you can allow a group of developers access to manage only resources for their project. This way, if a developer leaves the project, the developer's access is automatically revoked without changing the underlying access policies.

Desired outcome: Users should only have the permissions required to do their job. Users should only be given access to production environments to perform a specific task within a limited time period, and access should be revoked once that task is complete. Permissions should be revoked when no longer needed, including when a user moves onto a different project or job function. Administrator privileges should be given only to a small group of trusted administrators.

Permissions should be reviewed regularly to avoid permission creep. Machine or system accounts should be given the smallest set of permissions needed to complete their tasks.

Common anti-patterns:

- Defaulting to granting users administrator permissions.
- Using the root user for day-to-day activities.
- Creating policies that are overly permissive, but without full administrator privileges.
- Not reviewing permissions to understand whether they permit least privilege access.

Level of risk exposed if this best practice is not established: High

Implementation guidance

The principle of least privilege states that identities should only be permitted to perform the smallest set of actions necessary to fulfill a specific task. This balances usability, efficiency, and security. Operating under this principle helps limit unintended access and helps track who has access to what resources. IAM users and roles have no permissions by default. The root user has full access by default and should be tightly controlled, monitored, and used only for tasks that require root access.

IAM policies are used to explicitly grant permissions to IAM roles or specific resources. For example, identity-based policies can be attached to IAM groups, while S3 buckets can be controlled by resource-based policies.

When creating an IAM policy, you can specify the service actions, resources, and conditions that must be true for AWS to allow or deny access. AWS supports a variety of conditions to help you scope down access. For example, by using the `PrincipalOrgID` condition key, you can deny actions if the requestor isn't a part of your AWS Organization.

You can also control requests that AWS services make on your behalf, such as AWS CloudFormation creating an AWS Lambda function, using the `CalledVia` condition key. You should layer different policy types to establish defense-in-depth and limit the overall permissions of your users. You can also restrict what permissions can be granted and under what conditions. For example, you can allow your application teams to create their own IAM policies for systems they build, but must also apply a Permission Boundary to limit the maximum permissions the system can receive.

Implementation steps

- **Implement least privilege policies**: Assign access policies with least privilege to IAM groups and roles to reflect the user's role or function that you have defined.
  - **Base policies on API usage**: One way to determine the needed permissions is to review AWS CloudTrail logs. This review allows you to create permissions tailored to the actions that the user actually performs within AWS. IAM Access Analyzer can automatically generate an IAM policy based on activity. You can use IAM Access Advisor at the organization or account level to track the last accessed information for a particular policy.
- **Consider using AWS managed policies for job functions.** When starting to create fine-grained permissions policies, it can be difficult to know where to start. AWS has managed policies for common job roles, for example billing, database administrators, and data scientists. These policies can help narrow the access that users have while determining how to implement the least privilege policies.
- **Remove unnecessary permissions:** Remove permissions that are not needed and trim back overly permissive policies. IAM Access Analyzer policy generation can help fine-tune permissions policies.
- **Ensure that users have limited access to production environments:** Users should only have access to production environments with a valid use case. After the user performs the specific tasks that required production access, access should be revoked. Limiting access to production environments helps prevent unintended production-impacting events and lowers the scope of impact of unintended access.
- **Consider permissions boundaries:** A permissions boundary is a feature for using a managed policy that sets the maximum permissions that an identity-based policy can grant to an IAM entity. An entity's permissions boundary allows it to perform only the actions that are allowed by both its identity-based policies and its permissions boundaries.
- **Consider resource tags for permissions:** An attribute-based access control model using resource tags allows you to grant access based on resource purpose, owner, environment, or other criteria. For example, you can use resource tags to differentiate between development and production environments. Using these tags, you can restrict developers to the development environment. By combining tagging and permissions policies, you can achieve fine-grained resource access without needing to define complicated, custom policies for every job function.
- **Use service control policies for AWS Organizations.** Service control policies centrally control the maximum available permissions for member accounts in your organization. Importantly, service control policies allow you to restrict root user permissions in member accounts. Also consider using AWS Control Tower, which provides prescriptive managed controls that enrich AWS Organizations. You can also define your own controls within Control Tower.

- **Establish a user lifecycle policy for your organization:** User lifecycle policies define tasks to perform when users are onboarded onto AWS, change job role or scope, or no longer need access to AWS. Permission reviews should be done during each step of a user's lifecycle to verify that permissions are properly restrictive and to avoid permissions creep.
- **Establish a regular schedule to review permissions and remove any unneeded permissions:** You should regularly review user access to verify that users do not have overly permissive access. AWS Config and IAM Access Analyzer can help when auditing user permissions.
- **Establish a job role matrix:** A job role matrix visualizes the various roles and access levels required within your AWS footprint. Using a job role matrix, you can define and separate permissions based on user responsibilities within your organization. Use groups instead of applying permissions directly to individual users or roles.

Resources

Related documents:

- Grant least privilege
- Permissions boundaries for IAM entities
- Techniques for writing least privilege IAM policies
- IAM Access Analyzer makes it easier to implement least privilege permissions by generating IAM policies based on access activity
- Delegate permission management to developers by using IAM permissions boundaries
- Refining Permissions using last accessed information
- IAM policy types and when to use them
- Testing IAM policies with the IAM policy simulator
- Guardrails in AWS Control Tower
- Zero Trust architectures: An AWS perspective
- How to implement the principle of least privilege with CloudFormation StackSets
- Attribute-based access control (ABAC)
- Reducing policy scope by viewing user activity
- View role access
- Use Tagging to Organize Your Environment and Drive Accountability
- AWS Tagging Strategies

- Tagging AWS resources

Related videos:

- Next-generation permissions management
- Zero Trust: An AWS perspective

Related examples:

- Lab: IAM permissions boundaries delegating role creation
- Lab: IAM tag based access control for EC2

SEC03-BP03 Establish emergency access process

A process that allows emergency access to your workload in the unlikely event of an automated process or pipeline issue. This will help you rely on least privilege access, but ensure users can obtain the right level of access when they require it. For example, establish a process for administrators to verify and approve their request, such as an emergency AWS cross-account role for access, or a specific process for administrators to follow to validate and approve an emergency request.

Common anti-patterns:

- Not having an emergency process in place to recover from an outage with your existing identity configuration.
- Granting long term elevated permissions for troubleshooting or recovery purposes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Establishing emergency access can take several forms for which you should be prepared. The first is a failure of your primary identity provider. In this case, you should rely on a second method of access with the required permissions to recover. This method could be a backup identity provider or a user. This second method should be tightly controlled, monitored, and notify in the event it is used. The emergency access identity should source from an account specific for this purpose and only have permissions to assume a role specifically designed for recovery.

You should also be prepared for emergency access where temporary elevated administrative access is needed. A common scenario is to limit mutating permissions to an automated process used for deploying changes. In the event that this process has an issue, users might need to request elevated permissions to restore functionality. In this case, establish a process where users can request elevated access and administrators can validate and approve it. The implementation plans detailing the best practice guidance for pre-provisioning access and setting up emergency, *break-glass*, roles are provided as part of SEC10-BP05 Pre-provision access.

Resources

Related documents:

- Monitor and Notify on AWS
- Managing temporary elevated access

Related video:

- Become an IAM Policy Master in 60 Minutes or Less

SEC03-BP04 Reduce permissions continuously

As your teams determine what access is required, remove unneeded permissions and establish review processes to achieve least privilege permissions. Continually monitor and remove unused identities and permissions for both human and machine access.

Desired outcome: Permission policies should adhere to the least privilege principle. As job duties and roles become better defined, your permission policies need to be reviewed to remove unnecessary permissions. This approach lessens the scope of impact should credentials be inadvertently exposed or otherwise accessed without authorization.

Common anti-patterns:

- Defaulting to granting users administrator permissions.
- Creating policies that are overly permissive, but without full administrator privileges.
- Keeping permission policies after they are no longer needed.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

As teams and projects are just getting started, permissive permission policies might be used to inspire innovation and agility. For example, in a development or test environment, developers can be given access to a broad set of AWS services. We recommend that you evaluate access continuously and restrict access to only those services and service actions that are necessary to complete the current job. We recommend this evaluation for both human and machine identities. Machine identities, sometimes called system or service accounts, are identities that give AWS access to applications or servers. This access is especially important in a production environment, where overly permissive permissions can have a broad impact and potentially expose customer data.

AWS provides multiple methods to help identify unused users, roles, permissions, and credentials. AWS can also help analyze access activity of IAM users and roles, including associated access keys, and access to AWS resources such as objects in Amazon S3 buckets. AWS Identity and Access Management Access Analyzer policy generation can assist you in creating restrictive permission policies based on the actual services and actions a principal interacts with. Attribute-based access control (ABAC) can help simplify permissions management, as you can provide permissions to users using their attributes instead of attaching permissions policies directly to each user.

Implementation steps

- **Use AWS Identity and Access Management Access Analyzer:** IAM Access Analyzer helps identify resources in your organization and accounts, such as Amazon Simple Storage Service (Amazon S3) buckets or IAM roles that are shared with an external entity.
- **Use IAM Access Analyzer policy generation:** IAM Access Analyzer policy generation helps you create fine-grained permission policies based on an IAM user or role's access activity.
- **Determine an acceptable timeframe and usage policy for IAM users and roles:** Use the last accessed timestamp to identify unused users and roles and remove them. Review service and action last accessed information to identify and scope permissions for specific users and roles. For example, you can use last accessed information to identify the specific Amazon S3 actions that your application role requires and restrict the role's access to only those actions. Last accessed information features are available in the AWS Management Console and programmatically allow you to incorporate them into your infrastructure workflows and automated tools.
- **Consider logging data events in AWS CloudTrail:** By default, CloudTrail does not log data events such as Amazon S3 object-level activity (for example, `GetObject` and `DeleteObject`) or Amazon DynamoDB table activities (for example, `PutItem` and `DeleteItem`). Consider using

logging for these events to determine what users and roles need access to specific Amazon S3 objects or DynamoDB table items.

Resources

Related documents:

- Grant least privilege
- Remove unnecessary credentials
- What is AWS CloudTrail?
- Working with Policies
- Logging and monitoring DynamoDB
- Using CloudTrail event logging for Amazon S3 buckets and objects
- Getting credential reports for your AWS account

Related videos:

- Become an IAM Policy Master in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, and CI/CD
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive

SEC03-BP05 Define permission guardrails for your organization

Establish common controls that restrict access to all identities in your organization. For example, you can restrict access to specific AWS Regions, or prevent your operators from deleting common resources, such as an IAM role used for your central security team.

Common anti-patterns:

- Running workloads in your Organizational administrator account.
- Running production and non-production workloads in the same account.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

As you grow and manage additional workloads in AWS, you should separate these workloads using accounts and manage those accounts using AWS Organizations. We recommend that you establish common permission guardrails that restrict access to all identities in your organization. For example, you can restrict access to specific AWS Regions, or prevent your team from deleting common resources, such as an IAM role used by your central security team.

You can get started by implementing example service control policies, such as preventing users from turning off key services. SCPs use the IAM policy language and allow you to establish controls that all IAM principals (users and roles) adhere to. You can restrict access to specific service actions, resources and based on specific condition to meet the access control needs of your organization. If necessary, you can define exceptions to your guardrails. For example, you can restrict service actions for all IAM entities in the account except for a specific administrator role.

We recommend you avoid running workloads in your management account. The management account should be used to govern and deploy security guardrails that will affect member accounts. Some AWS services support the use of a delegated administrator account. When available, you should use this delegated account instead of the management account. You should strongly limit access to the Organizational administrator account.

Using a multi-account strategy allows you to have greater flexibility in applying guardrails to your workloads. The AWS Security Reference Architecture gives prescriptive guidance on how to design your account structure. AWS services such as AWS Control Tower provide capabilities to centrally manage both preventative and detective controls across your organization. Define a clear purpose for each account or OU within your organization and limit controls in line with that purpose.

Resources

Related documents:

- AWS Organizations
- Service control policies (SCPs)
- Get more out of service control policies in a multi-account environment
- AWS Security Reference Architecture (AWS SRA)

Related videos:

- Enforce Preventive Guardrails using Service Control Policies

- Building governance at scale with AWS Control Tower
- AWS Identity and Access Management deep dive

SEC03-BP06 Manage access based on lifecycle

Integrate access controls with operator and application lifecycle and your centralized federation provider. For example, remove a user's access when they leave the organization or change roles.

As you manage workloads using separate accounts, there will be cases where you need to share resources between those accounts. We recommend that you share resources using AWS Resource Access Manager (AWS RAM). This service allows you to easily and securely share AWS resources within your AWS Organizations and Organizational Units. Using AWS RAM, access to shared resources is automatically granted or revoked as accounts are moved in and out of the Organization or Organization Unit with which they are shared. This helps ensure that resources are only shared with the accounts that you intend.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Implement a user access lifecycle policy for new users joining, job function changes, and users leaving so that only current users have access.

Resources

Related documents:

- Attribute-based access control (ABAC)
- Grant least privilege
- IAM Access Analyzer
- Remove unnecessary credentials
- Working with Policies

Related videos:

- Become an IAM Policy Master in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, and CI/CD

SEC03-BP07 Analyze public and cross-account access

Continually monitor findings that highlight public and cross-account access. Reduce public access and cross-account access to only the specific resources that require this access.

Desired outcome: Know which of your AWS resources are shared and with whom. Continually monitor and audit your shared resources to verify they are shared with only authorized principals.

Common anti-patterns:

- Not keeping an inventory of shared resources.
- Not following a process for approval of cross-account or public access to resources.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

If your account is in AWS Organizations, you can grant access to resources to the entire organization, specific organizational units, or individual accounts. If your account is not a member of an organization, you can share resources with individual accounts. You can grant direct cross-account access using resource-based policies — for example, Amazon Simple Storage Service (Amazon S3) bucket policies — or by allowing a principal in another account to assume an IAM role in your account. When using resource policies, verify that access is only granted to authorized principals. Define a process to approve all resources which are required to be publicly available.

AWS Identity and Access Management Access Analyzer uses provable security to identify all access paths to a resource from outside of its account. It reviews resource policies continuously, and reports findings of public and cross-account access to make it simple for you to analyze potentially broad access. Consider configuring IAM Access Analyzer with AWS Organizations to verify that you have visibility to all your accounts. IAM Access Analyzer also allows you to preview findings before deploying resource permissions. This allows you to validate that your policy changes grant only the intended public and cross-account access to your resources. When designing for multi-account access, you can use trust policies to control in what cases a role can be assumed. For example, you could use the `PrincipalOrgId` condition key to deny an attempt to assume a role from outside your AWS Organizations.

AWS Config can report resources that are misconfigured, and through AWS Config policy checks, can detect resources that have public access configured. Services such as AWS Control Tower and AWS Security Hub CSPM simplify deploying detective controls and guardrails across AWS Organizations to identify and remediate publicly exposed resources. For example, AWS Control

Tower has a managed guardrail which can detect if any Amazon EBS snapshots are restorable by AWS accounts.

Implementation steps

- **Consider using AWS Config for AWS Organizations:** AWS Config allows you to aggregate findings from multiple accounts within an AWS Organizations to a delegated administrator account. This provides a comprehensive view, and allows you to deploy AWS Config Rules across accounts to detect publicly accessible resources.
- **Configure AWS Identity and Access Management Access Analyzer** IAM Access Analyzer helps you identify resources in your organization and accounts, such as Amazon S3 buckets or IAM roles that are shared with an external entity.
- **Use auto-remediation in AWS Config to respond to changes in public access configuration of Amazon S3 buckets:** You can automatically turn on the block public access settings for Amazon S3 buckets.
- **Implement monitoring and alerting to identify if Amazon S3 buckets have become public:** You must have monitoring and alerting in place to identify when Amazon S3 Block Public Access is turned off, and if Amazon S3 buckets become public. Additionally, if you are using AWS Organizations, you can create a service control policy that prevents changes to Amazon S3 public access policies. AWS Trusted Advisor checks for Amazon S3 buckets that have open access permissions. Bucket permissions that grant, upload, or delete access to everyone create potential security issues by allowing anyone to add, modify, or remove items in a bucket. The Trusted Advisor check examines explicit bucket permissions and associated bucket policies that might override the bucket permissions. You also can use AWS Config to monitor your Amazon S3 buckets for public access. For more information, see How to Use AWS Config to Monitor for and Respond to Amazon S3 Buckets Allowing Public Access. While reviewing access, it's important to consider what types of data are contained in Amazon S3 buckets. Amazon Macie helps discover and protect sensitive data, such as PII, PHI, and credentials, such as private or AWS keys.

Resources

Related documents:

- Using AWS Identity and Access Management Access Analyzer
- AWS Control Tower controls library
- AWS Foundational Security Best Practices standard
- AWS Config Managed Rules

- AWS Trusted Advisor check reference
- Monitoring AWS Trusted Advisor check results with Amazon EventBridge
- Managing AWS Config Rules Across All Accounts in Your Organization
- AWS Config and AWS Organizations

Related videos:

- Best Practices for securing your multi-account environment
- Dive Deep into IAM Access Analyzer

SEC03-BP08 Share resources securely within your organization

As the number of workloads grows, you might need to share access to resources in those workloads or provision the resources multiple times across multiple accounts. You might have constructs to compartmentalize your environment, such as having development, testing, and production environments. However, having separation constructs does not limit you from being able to share securely. By sharing components that overlap, you can reduce operational overhead and allow for a consistent experience without guessing what you might have missed while creating the same resource multiple times.

Desired outcome: Minimize unintended access by using secure methods to share resources within your organization, and help with your data loss prevention initiative. Reduce your operational overhead compared to managing individual components, reduce errors from manually creating the same component multiple times, and increase your workloads' scalability. You can benefit from decreased time to resolution in multi-point failure scenarios, and increase your confidence in determining when a component is no longer needed. For prescriptive guidance on analyzing externally shared resources, see SEC03-BP07 Analyze public and cross-account access.

Common anti-patterns:

- Lack of process to continually monitor and automatically alert on unexpected external share.
- Lack of baseline on what should be shared and what should not.
- Defaulting to a broadly open policy rather than sharing explicitly when required.
- Manually creating foundational resources that overlap when required.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Architect your access controls and patterns to govern the consumption of shared resources securely and only with trusted entities. Monitor shared resources and review shared resource access continuously, and be alerted on inappropriate or unexpected sharing. Review Analyze public and cross-account access to help you establish governance to reduce the external access to only resources that require it, and to establish a process to monitor continuously and alert automatically.

Cross-account sharing within AWS Organizations is supported by a number of AWS services, such as AWS Security Hub CSPM, Amazon GuardDuty, and AWS Backup. These services allow for data to be shared to a central account, be accessible from a central account, or manage resources and data from a central account. For example, AWS Security Hub CSPM can transfer findings from individual accounts to a central account where you can view all the findings. AWS Backup can take a backup for a resource and share it across accounts. You can use AWS Resource Access Manager (AWS RAM) to share other common resources, such as VPC subnets and Transit Gateway attachments, AWS Network Firewall, or Amazon SageMaker AI pipelines.

To restrict your account to only share resources within your organization, use service control policies (SCPs) to prevent access to external principals. When sharing resources, combine identity-based controls and network controls to create a data perimeter for your organization to help protect against unintended access. A data perimeter is a set of preventive guardrails to help verify that only your trusted identities are accessing trusted resources from expected networks. These controls place appropriate limits on what resources can be shared and prevent sharing or exposing resources that should not be allowed. For example, as a part of your data perimeter, you can use VPC endpoint policies and the `AWS:PrincipalOrgId` condition to ensure the identities accessing your Amazon S3 buckets belong to your organization. It is important to note that SCPs do not apply to service-linked roles or AWS service principals.

When using Amazon S3, turn off ACLs for your Amazon S3 bucket and use IAM policies to define access control. For restricting access to an Amazon S3 origin from Amazon CloudFront, migrate from origin access identity (OAI) to origin access control (OAC) which supports additional features including server-side encryption with AWS Key Management Service.

In some cases, you might want to allow sharing resources outside of your organization or grant a third party access to your resources. For prescriptive guidance on managing permissions to share resources externally, see Permissions management.

Implementation steps

1. **Use AWS Organizations.**

   AWS Organizations is an account management service that allows you to consolidate multiple AWS accounts into an organization that you create and centrally manage. You can group your accounts into organizational units (OUs) and attach different policies to each OU to help you meet your budgetary, security, and compliance needs. You can also control how AWS artificial intelligence (AI) and machine learning (ML) services can collect and store data, and use the multi-account management of the AWS services integrated with Organizations.

2. **Integrate AWS Organizations with AWS services.**

   When you use an AWS service to perform tasks on your behalf in the member accounts of your organization, AWS Organizations creates an IAM service-linked role (SLR) for that service in each member account. You should manage trusted access using the AWS Management Console, the AWS APIs, or the AWS CLI. For prescriptive guidance on turning on trusted access, see Using AWS Organizations with other AWS services and AWS services that you can use with Organizations.

3. **Establish a data perimeter.**

   The AWS perimeter is typically represented as an organization managed by AWS Organizations. Along with on-premises networks and systems, accessing AWS resources is what many consider as the perimeter of My AWS. The goal of the perimeter is to verify that access is allowed if the identity is trusted, the resource is trusted, and the network is expected.

   a. Define and implement the perimeters.

      Follow the steps described in Perimeter implementation in the Building a Perimeter on AWS whitepaper for each authorization condition. For prescriptive guidance on protecting network layer, see Protecting networks.

   b. Monitor and alert continually.

      AWS Identity and Access Management Access Analyzer helps identify resources in your organization and accounts that are shared with external entities. You can integrate IAM Access Analyzer with AWS Security Hub CSPM to send and aggregate findings for a resource from IAM Access Analyzer to Security Hub CSPM to help analyze the security posture of your environment. To integrate, turn on both IAM Access Analyzer and Security Hub CSPM in each Region in each account. You can also use AWS Config Rules to audit configuration and alert the appropriate party using Amazon Q Developer in chat applications with AWS Security Hub CSPM. You can then use AWS Systems Manager Automation documents to remediate noncompliant resources.

   c. For prescriptive guidance on monitoring and alerting continuously on resources shared externally, see Analyze public and cross-account access.

4. **Use resource sharing in AWS services and restrict accordingly.**

   Many AWS services allow you to share resources with another account, or target a resource in another account, such as Amazon Machine Images (AMIs) and AWS Resource Access Manager (AWS RAM). Restrict the `ModifyImageAttribute` API to specify the trusted accounts to share the AMI with. Specify the `ram:RequestedAllowsExternalPrincipals` condition when using AWS RAM to constrain sharing to your organization only, to help prevent access from untrusted identities. For prescriptive guidance and considerations, see Resource sharing and external targets.

5. **Use AWS RAM to share securely in an account or with other AWS accounts.**

   AWS RAM helps you securely share the resources that you have created with roles and users in your account and with other AWS accounts. In a multi-account environment, AWS RAM allows you to create a resource once and share it with other accounts. This approach helps reduce your operational overhead while providing consistency, visibility, and auditability through integrations with Amazon CloudWatch and AWS CloudTrail, which you do not receive when using cross-account access.

   If you have resources that you shared previously using a resource-based policy, you can use the `PromoteResourceShareCreatedFromPolicy` API or an equivalent to promote the resource share to a full AWS RAM resource share.

   In some cases, you might need to take additional steps to share resources. For example, to share an encrypted snapshot, you need to share a AWS KMS key.

Resources

Related best practices:

- SEC03-BP07 Analyze public and cross-account access
- SEC03-BP09 Share resources securely with a third party
- SEC05-BP01 Create network layers

Related documents:

- Bucket owner granting cross-account permission to objects it does not own
- How to use Trust Policies with IAM
- Building Data Perimeter on AWS
- How to use an external ID when granting a third party access to your AWS resources
- AWS services you can use with AWS Organizations
- Establishing a data perimeter on AWS: Allow only trusted identities to access company data

Related videos:

- Granular Access with AWS Resource Access Manager
- Securing your data perimeter with VPC endpoints
- Establishing a data perimeter on AWS

Related tools:

- Data Perimeter Policy Examples

SEC03-BP09 Share resources securely with a third party

The security of your cloud environment doesn't stop at your organization. Your organization might rely on a third party to manage a portion of your data. The permission management for the third-party managed system should follow the practice of just-in-time access using the principle of least privilege with temporary credentials. By working closely with a third party, you can reduce the scope of impact and risk of unintended access together.

Desired outcome: Long-term AWS Identity and Access Management (IAM) credentials, IAM access keys, and secret keys that are associated with a user can be used by anyone as long as the credentials are valid and active. Using an IAM role and temporary credentials helps you improve your overall security stance by reducing the effort to maintain long-term credentials, including the management and operational overhead of those sensitive details. By using a universally unique identifier (UUID) for the external ID in the IAM trust policy, and keeping the IAM policies attached to the IAM role under your control, you can audit and verify that the access granted to the third party is not too permissive. For prescriptive guidance on analyzing externally shared resources, see SEC03-BP07 Analyze public and cross-account access.

Common anti-patterns:

- Using the default IAM trust policy without any conditions.
- Using long-term IAM credentials and access keys.
- Reusing external IDs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

You might want to allow sharing resources outside of AWS Organizations or grant a third party access to your account. For example, a third party might provide a monitoring solution that needs to access resources within your account. In those cases, create an IAM cross-account role with only the privileges needed by the third party. Additionally, define a trust policy using the external ID condition. When using an external ID, you or the third party can generate a unique ID for each customer, third party, or tenancy. The unique ID should not be controlled by anyone but you after it's created. The third party must implement a process to relate the external ID to the customer in a secure, auditable, and reproduceable manner.

You can also use IAM Roles Anywhere to manage IAM roles for applications outside of AWS that use AWS APIs.

If the third party no longer requires access to your environment, remove the role. Avoid providing long-term credentials to a third party. Maintain awareness of other AWS services that support sharing. For example, the AWS Well-Architected Tool allows sharing a workload with other AWS accounts, and AWS Resource Access Manager helps you securely share an AWS resource you own with other accounts.

Implementation steps

1. **Use cross-account roles to provide access to external accounts.**

   Cross-account roles reduce the amount of sensitive information that is stored by external accounts and third parties for servicing their customers. Cross-account roles allow you to grant access to AWS resources in your account securely to a third party, such as AWS Partners or other accounts in your organization, while maintaining the ability to manage and audit that access.

   The third party might be providing service to you from a hybrid infrastructure or alternatively pulling data into an offsite location. IAM Roles Anywhere helps you allow third party workloads to securely interact with your AWS workloads and further reduce the need for long-term credentials.

You should not use long-term credentials, or access keys associated with users, to provide external account access. Instead, use cross-account roles to provide the cross-account access.

2. **Use an external ID with third parties.**

   Using an external ID allows you to designate who can assume a role in an IAM trust policy. The trust policy can require that the user assuming the role assert the condition and target in which they are operating. It also provides a way for the account owner to permit the role to be assumed only under specific circumstances. The primary function of the external ID is to address and prevent the confused deputy problem.

   Use an external ID if you are an AWS account owner and you have configured a role for a third party that accesses other AWS accounts in addition to yours, or when you are in the position of assuming roles on behalf of different customers. Work with your third party or AWS Partner to establish an external ID condition to include in IAM trust policy.

3. **Use universally unique external IDs.**

   Implement a process that generates random unique value for an external ID, such as a universally unique identifier (UUID). A third party reusing external IDs across different customers does not address the confused deputy problem, because customer A might be able to view data of customer B by using the role ARN of customer B along with the duplicated external ID. In a multi-tenant environment, where a third party supports multiple customers with different AWS accounts, the third party must use a different unique ID as the external ID for each AWS account. The third party is responsible for detecting duplicate external IDs and securely mapping each customer to their respective external ID. The third party should test to verify that they can only assume the role when specifying the external ID. The third party should refrain from storing the customer role ARN and the external ID until the external ID is required.

   The external ID is not treated as a secret, but the external ID must not be an easily guessable value, such as a phone number, name, or account ID. Make the external ID a read-only field so that the external ID cannot be changed for the purpose of impersonating the setup.

   You or the third party can generate the external ID. Define a process to determine who is responsible for generating the ID. Regardless of the entity creating the external ID, the third party enforces uniqueness and formats consistently across customers.

4. **Deprecate customer-provided long-term credentials.**

Deprecate the use of long-term credentials and use cross-account roles or IAM Roles Anywhere. If you must use long-term credentials, establish a plan to migrate to role-based access. For details on managing keys, see Identity Management. Also work with your AWS account team and the third party to establish risk mitigation runbook. For prescriptive guidance on responding to and mitigating the potential impact of security incident, see Incident response.

5. **Verify that setup has prescriptive guidance or is automated.**

   The policy created for cross-account access in your accounts must follow the least-privilege principle. The third party must provide a role policy document or automated setup mechanism that uses an AWS CloudFormation template or an equivalent for you. This reduces the chance of errors associated with manual policy creation and offers an auditable trail. For more information on using a CloudFormation template to create cross-account roles, see Cross-Account Roles.

   The third party should provide an automated, auditable setup mechanism. However, by using the role policy document outlining the access needed, you should automate the setup of the role. Using a CloudFormation template or equivalent, you should monitor for changes with drift detection as part of the audit practice.

6. **Account for changes.**

   Your account structure, your need for the third party, or their service offering being provided might change. You should anticipate changes and failures, and plan accordingly with the right people, process, and technology. Audit the level of access you provide on a periodic basis, and implement detection methods to alert you to unexpected changes. Monitor and audit the use of the role and the datastore of the external IDs. You should be prepared to revoke third-party access, either temporarily or permanently, as a result of unexpected changes or access patterns. Also, measure the impact to your revocation operation, including the time it takes to perform, the people involved, the cost, and the impact to other resources.

   For prescriptive guidance on detection methods, see the Detection best practices.

Resources

Related best practices:

- SEC02-BP02 Use temporary credentials
- SEC03-BP05 Define permission guardrails for your organization
- SEC03-BP06 Manage access based on lifecycle

- SEC03-BP07 Analyze public and cross-account access
- SEC04 Detection

Related documents:

- Bucket owner granting cross-account permission to objects it does not own
- How to use trust policies with IAM roles
- Delegate access across AWS accounts using IAM roles
- How do I access resources in another AWS account using IAM?
- Security best practices in IAM
- Cross-account policy evaluation logic
- How to use an external ID when granting access to your AWS resources to a third party
- Collecting Information from AWS CloudFormation Resources Created in External Accounts with Custom Resources
- Securely Using External ID for Accessing AWS Accounts Owned by Others
- Extend IAM roles to workloads outside of IAM with IAM Roles Anywhere

Related videos:

- How do I allow users or roles in a separate AWS account access to my AWS account?
- AWS re:Invent 2018: Become an IAM Policy Master in 60 Minutes or Less
- AWS Knowledge Center Live: IAM Best Practices and Design Decisions

Related examples:

- Well-Architected Lab - Lambda cross account IAM role assumption (Level 300)
- Configure cross-account access to Amazon DynamoDB
- AWS STS Network Query Tool

Detection

Question

- SEC 4. How do you detect and investigate security events?

SEC 4. How do you detect and investigate security events?

Capture and analyze events from logs and metrics to gain visibility. Take action on security events and potential threats to help secure your workload.

Best practices

- SEC04-BP01 Configure service and application logging
- SEC04-BP02 Analyze logs, findings, and metrics centrally
- SEC04-BP03 Automate response to events
- SEC04-BP04 Implement actionable security events

SEC04-BP01 Configure service and application logging

Retain security event logs from services and applications. This is a fundamental principle of security for audit, investigations, and operational use cases, and a common security requirement driven by governance, risk, and compliance (GRC) standards, policies, and procedures.

Desired outcome: An organization should be able to reliably and consistently retrieve security event logs from AWS services and applications in a timely manner when required to fulfill an internal process or obligation, such as a security incident response. Consider centralizing logs for better operational results.

Common anti-patterns:

- Logs are stored in perpetuity or deleted too soon.
- Everybody can access logs.
- Relying entirely on manual processes for log governance and use.
- Storing every single type of log just in case it is needed.
- Checking log integrity only when necessary.

Benefits of establishing this best practice: Implement a root cause analysis (RCA) mechanism for security incidents and a source of evidence for your governance, risk, and compliance obligations.

Level of risk exposed if this best practice is not established: High

Implementation guidance

During a security investigation or other use cases based on your requirements, you need to be able to review relevant logs to record and understand the full scope and timeline of the incident. Logs are also required for alert generation, indicating that certain actions of interest have happened. It is critical to select, turn on, store, and set up querying and retrieval mechanisms and alerting.

Implementation steps

- **Select and use log sources.** Ahead of a security investigation, you need to capture relevant logs to retroactively reconstruct activity in an AWS account. Select log sources relevant to your workloads.

  The log source selection criteria should be based on the use cases required by your business. Establish a trail for each AWS account using AWS CloudTrail or an AWS Organizations trail, and configure an Amazon S3 bucket for it.

  AWS CloudTrail is a logging service that tracks API calls made against an AWS account capturing AWS service activity. It's turned on by default with a 90-day retention of management events that can be retrieved through CloudTrail Event history using the AWS Management Console, the AWS CLI, or an AWS SDK. For longer retention and visibility of data events, create a CloudTrail trail and associate it with an Amazon S3 bucket, and optionally with a Amazon CloudWatch log group. Alternatively, you can create a CloudTrail Lake, which retains CloudTrail logs for up to seven years and provides a SQL-based querying facility

  AWS recommends that customers using a VPC turn on network traffic and DNS logs using VPC Flow Logs and Amazon Route 53 resolver query logs, respectively, and streaming them to either an Amazon S3 bucket or a CloudWatch log group. You can create a VPC flow log for a VPC, a subnet, or a network interface. For VPC Flow Logs, you can be selective on how and where you use Flow Logs to reduce cost.

  AWS CloudTrail Logs, VPC Flow Logs, and Route 53 resolver query logs are the basic logging sources to support security investigations in AWS. You can also use Amazon Security Lake to collect, normalize, and store this log data in Apache Parquet format and Open Cybersecurity Schema Framework (OCSF), which is ready for querying. Security Lake also supports other AWS logs and logs from third-party sources.

  AWS services can generate logs not captured by the basic log sources, such as Elastic Load Balancing logs, AWS WAF logs, AWS Config recorder logs, Amazon GuardDuty findings, Amazon

Elastic Kubernetes Service (Amazon EKS) audit logs, and Amazon EC2 instance operating system and application logs. For a full list of logging and monitoring options, see Appendix A: Cloud capability definitions – Logging and Events of the AWS Security Incident Response Guide.

- **Research logging capabilities for each AWS service and application:** Each AWS service and application provides you with options for log storage, each of which with its own retention and life-cycle capabilities. The two most common log storage services are Amazon Simple Storage Service (Amazon S3) and Amazon CloudWatch. For long retention periods, it is recommended to use Amazon S3 for its cost effectiveness and flexible lifecycle capabilities. If the primary logging option is Amazon CloudWatch Logs, as an option, you should consider archiving less frequently accessed logs to Amazon S3.
- **Select log storage:** The choice of log storage is generally related to which querying tool you use, retention capabilities, familiarity, and cost. The main options for log storage are an Amazon S3 bucket or a CloudWatch Log group.

  An Amazon S3 bucket provides cost-effective, durable storage with an optional lifecycle policy. Logs stored in Amazon S3 buckets can be queried using services such as Amazon Athena.

  A CloudWatch log group provides durable storage and a built-in query facility through CloudWatch Logs Insights.
- **Identify appropriate log retention:** When you use an Amazon S3 bucket or CloudWatch log group to store logs, you must establish adequate lifecycles for each log source to optimize storage and retrieval costs. Customers generally have between three months to one year of logs readily available for querying, with retention of up to seven years. The choice of availability and retention should align with your security requirements and a composite of statutory, regulatory, and business mandates.
- **Use logging for each AWS service and application with proper retention and lifecycle policies:** For each AWS service or application in your organization, look for the specific logging configuration guidance:
  - Configure AWS CloudTrail Trail
  - Configure VPC Flow Logs
  - Configure Amazon GuardDuty Finding Export
  - Configure AWS Config recording
  - Configure AWS WAF web ACL traffic
  - Configure AWS Network Firewall network traffic logs
  - Configure Elastic Load Balancing access logs

  - Configure Amazon Route 53 resolver query logs
  - Configure Amazon RDS logs
  - Configure Amazon EKS Control Plane logs
  - Configure Amazon CloudWatch agent for Amazon EC2 instances and on-premises servers
- **Select and implement querying mechanisms for logs:** For log queries, you can use CloudWatch Logs Insights for data stored in CloudWatch log groups, and Amazon Athena and Amazon OpenSearch Service for data stored in Amazon S3. You can also use third-party querying tools such as a security information and event management (SIEM) service.

  The process for selecting a log querying tool should consider the people, process, and technology aspects of your security operations. Select a tool that fulfills operational, business, and security requirements, and is both accessible and maintainable in the long term. Keep in mind that log querying tools work optimally when the number of logs to be scanned is kept within the tool's limits. It is not uncommon to have multiple querying tools because of cost or technical constraints.

  For example, you might use a third-party security information and event management (SIEM) tool to perform queries for the last 90 days of data, but use Athena to perform queries beyond 90 days because of the log ingestion cost of a SIEM. Regardless of the implementation, verify that your approach minimizes the number of tools required to maximize operational efficiency, especially during a security event investigation.
- **Use logs for alerting:** AWS provides alerting through several security services:
  - AWS Config monitors and records your AWS resource configurations and allows you to automate the evaluation and remediation against desired configurations.
  - Amazon GuardDuty is a threat detection service that continually monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads. GuardDuty ingests, aggregates, and analyzes information from sources, such as AWS CloudTrail management and data events, DNS logs, VPC Flow Logs, and Amazon EKS Audit logs. GuardDuty pulls independent data streams directly from CloudTrail, VPC Flow Logs, DNS query logs, and Amazon EKS. You don't have to manage Amazon S3 bucket policies or modify the way you collect and store logs. It is still recommended to retain these logs for your own investigation and compliance purposes.
  - AWS Security Hub CSPM provides a single place that aggregates, organizes, and prioritizes your security alerts or findings from multiple AWS services and optional third-party products to give you a comprehensive view of security alerts and compliance status.

You can also use custom alert generation engines for security alerts not covered by these services or for specific alerts relevant to your environment. For information on building these alerts and detections, see Detection in the AWS Security Incident Response Guide.

Resources

Related best practices:

- SEC04-BP02 Analyze logs, findings, and metrics centrally
- SEC07-BP04 Define data lifecycle management
- SEC10-BP06 Pre-deploy tools

Related documents:

- AWS Security Incident Response Guide
- Getting Started with Amazon Security Lake
- Getting started: Amazon CloudWatch Logs
- Security Partner Solutions: Logging and Monitoring

Related videos:

- AWS re:Invent 2022 - Introducing Amazon Security Lake

Related examples:

- Assisted Log Enabler for AWS
- AWS Security Hub CSPM Findings Historical Export

Related tools:

- Snowflake for Cybersecurity

SEC04-BP02 Analyze logs, findings, and metrics centrally

Security operations teams rely on the collection of logs and the use of search tools to discover potential events of interest, which might indicate unauthorized activity or unintentional change. However, simply analyzing collected data and manually processing information is insufficient to keep up with the volume of information flowing from complex architectures. Analysis and reporting alone don't facilitate the assignment of the right resources to work an event in a timely fashion.

A best practice for building a mature security operations team is to deeply integrate the flow of security events and findings into a notification and workflow system such as a ticketing system, a bug or issue system, or other security information and event management (SIEM) system. This takes the workflow out of email and static reports, and allows you to route, escalate, and manage events or findings. Many organizations are also integrating security alerts into their chat or collaboration, and developer productivity platforms. For organizations embarking on automation, an API-driven, low-latency ticketing system offers considerable flexibility when planning what to automate first.

This best practice applies not only to security events generated from log messages depicting user activity or network events, but also from changes detected in the infrastructure itself. The ability to detect change, determine whether a change was appropriate, and then route that information to the correct remediation workflow is essential in maintaining and validating a secure architecture, in the context of changes where the nature of their undesirability is sufficiently subtle that they cannot currently be prevented with a combination of AWS Identity and Access Management (IAM) and AWS Organizations configuration.

Amazon GuardDuty and AWS Security Hub CSPM provide aggregation, deduplication, and analysis mechanisms for log records that are also made available to you via other AWS services. GuardDuty ingests, aggregates, and analyzes information from sources such as AWS CloudTrail management and data events, VPC DNS logs, and VPC Flow Logs. Security Hub CSPM can ingest, aggregate, and analyze output from GuardDuty, AWS Config, Amazon Inspector, Amazon Macie, AWS Firewall Manager, and a significant number of third-party security products available in the AWS Marketplace, and if built accordingly, your own code. Both GuardDuty and Security Hub CSPM have an Administrator-Member model that can aggregate findings and insights across multiple accounts, and Security Hub CSPM is often used by customers who have an on- premises SIEM as an AWS-side log and alert preprocessor and aggregator from which they can then ingest Amazon EventBridge through a AWS Lambda-based processor and forwarder.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Evaluate log processing capabilities: Evaluate the options that are available for processing logs.
  - Find an AWS Partner that specializes in logging and monitoring solutions
- As a start for analyzing CloudTrail logs, test Amazon Athena.
  - Configuring Athena to analyze CloudTrail logs
- Implement centralize logging in AWS: See the following AWS example solution to centralize logging from multiple sources.
  - Centralize logging solution
- Implement centralize logging with partner: APN Partners have solutions to help you analyze logs centrally.
  - Logging and Monitoring

Resources

Related documents:

- AWS Answers: Centralized Logging
- AWS Security Hub CSPM
- Amazon CloudWatch
- Amazon EventBridge
- Getting started: Amazon CloudWatch Logs
- Security Partner Solutions: Logging and Monitoring

Related videos:

- Centrally Monitoring Resource Configuration and Compliance
- Remediating Amazon GuardDuty and AWS Security Hub CSPM Findings
- Threat management in the cloud: Amazon GuardDuty and AWS Security Hub CSPM

SEC04-BP03 Automate response to events

Using automation to investigate and remediate events reduces human effort and error, and allows you to scale investigation capabilities. Regular reviews will help you tune automation tools, and continuously iterate.

In AWS, investigating events of interest and information on potentially unexpected changes into an automated workflow can be achieved using Amazon EventBridge. This service provides a scalable rules engine designed to broker both native AWS event formats (such as AWS CloudTrail events), as well as custom events you can generate from your application. Amazon GuardDuty also allows you to route events to a workflow system for those building incident response systems (AWS Step Functions), or to a central Security Account, or to a bucket for further analysis.

Detecting change and routing this information to the correct workflow can also be accomplished using AWS Config Rules and Conformance Packs. AWS Config detects changes to in-scope services (though with higher latency than EventBridge) and generates events that can be parsed using AWS Config Rules for rollback, enforcement of compliance policy, and forwarding of information to systems, such as change management platforms and operational ticketing systems. As well as writing your own Lambda functions to respond to AWS Config events, you can also take advantage of the AWS Config Rules Development Kit, and a library of open source AWS Config Rules. Conformance packs are a collection of AWS Config Rules and remediation actions you deploy as a single entity authored as a YAML template. A sample conformance pack template is available for the Well-Architected Security Pillar.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Implement automated alerting with GuardDuty: GuardDuty is a threat detection service that continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads. Turn on GuardDuty and configure automated alerts.
- Automate investigation processes: Develop automated processes that investigate an event and report information to an administrator to save time.
  - Lab: Amazon GuardDuty hands on

Resources

Related documents:

- AWS Answers: Centralized Logging
- AWS Security Hub CSPM
- Amazon CloudWatch
- Amazon EventBridge

- Getting started: Amazon CloudWatch Logs
- Security Partner Solutions: Logging and Monitoring
- Setting up Amazon GuardDuty

Related videos:

- Centrally Monitoring Resource Configuration and Compliance
- Remediating Amazon GuardDuty and AWS Security Hub CSPM Findings
- Threat management in the cloud: Amazon GuardDuty and AWS Security Hub CSPM

Related examples:

- Lab: Automated Deployment of Detective Controls

SEC04-BP04 Implement actionable security events

Create alerts that are sent to and can be actioned by your team. Ensure that alerts include relevant information for the team to take action. For each detective mechanism you have, you should also have a process, in the form of a runbook or playbook, to investigate. For example, when you use Amazon GuardDuty, it generates different findings. You should have a runbook entry for each finding type, for example, if a trojan is discovered, your runbook has simple instructions that instruct someone to investigate and remediate.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Discover metrics available for AWS services: Discover the metrics that are available through Amazon CloudWatch for the services that you are using.
  - AWS service documentation
  - Using Amazon CloudWatch Metrics
- Configure Amazon CloudWatch alarms.
  - Using Amazon CloudWatch Alarms

Resources

Related documents:

- Amazon CloudWatch
- Amazon EventBridge
- Security Partner Solutions: Logging and Monitoring

Related videos:

- Centrally Monitoring Resource Configuration and Compliance
- Remediating Amazon GuardDuty and AWS Security Hub CSPM Findings
- Threat management in the cloud: Amazon GuardDuty and AWS Security Hub CSPM

Infrastructure protection

Questions

- SEC 5. How do you protect your network resources?
- SEC 6. How do you protect your compute resources?

SEC 5. How do you protect your network resources?

Any workload that has some form of network connectivity, whether it's the internet or a private network, requires multiple layers of defense to help protect from external and internal network-based threats.

Best practices

- SEC05-BP01 Create network layers
- SEC05-BP02 Control traffic at all layers
- SEC05-BP03 Automate network protection
- SEC05-BP04 Implement inspection and protection

SEC05-BP01 Create network layers

Group components that share sensitivity requirements into layers to minimize the potential scope of impact of unauthorized access. For example, a database cluster in a virtual private cloud (VPC) with no need for internet access should be placed in subnets with no route to or from the internet. Traffic should only flow from the adjacent next least sensitive resource. Consider a web application sitting behind a load balancer. Your database should not be accessible directly from the load balancer. Only the business logic or web server should have direct access to your database.

Desired outcome: Create a layered network. Layered networks help logically group similar networking components. They also shrink the potential scope of impact of unauthorized network access. A properly layered network makes it harder for unauthorized users to pivot to additional resources within your AWS environment. In addition to securing internal network paths, you should also protect your network edge, such as web applications and API endpoints.

Common anti-patterns:

- Creating all resources in a single VPC or subnet.
- Using overly permissive security groups.
- Failing to use subnets.
- Allowing direct access to data stores such as databases.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Components such as Amazon Elastic Compute Cloud (Amazon EC2) instances, Amazon Relational Database Service (Amazon RDS) database clusters, and AWS Lambda functions that share reachability requirements can be segmented into layers formed by subnets. Consider deploying serverless workloads, such as Lambda functions, within a VPC or behind an Amazon API Gateway. AWS Fargate tasks that have no need for internet access should be placed in subnets with no route to or from the internet. This layered approach mitigates the impact of a single layer misconfiguration, which could allow unintended access. For AWS Lambda, you can run your functions in your VPC to take advantage of VPC-based controls.

For network connectivity that can include thousands of VPCs, AWS accounts, and on-premises networks, you should use AWS Transit Gateway. Transit Gateway acts as a hub that controls how traffic is routed among all the connected networks, which act like spokes. Traffic between Amazon

Virtual Private Cloud (Amazon VPC) and Transit Gateway remains on the AWS private network, which reduces external exposure to unauthorized users and potential security issues. Transit Gateway Inter-Region peering also encrypts inter-Region traffic with no single point of failure or bandwidth bottleneck.

Implementation steps

- **Use Reachability Analyzer to analyze the path between a source and destination based on configuration:** Reachability Analyzer allows you to automate verification of connectivity to and from VPC connected resources. Note that this analysis is done by reviewing configuration (no network packets are sent in conducting the analysis).
- **Use Amazon VPC Network Access Analyzer to identify unintended network access to resources:** Amazon VPC Network Access Analyzer allows you to specify your network access requirements and identify potential network paths.
- **Consider whether resources need to be in a public subnet:** Do not place resources in public subnets of your VPC unless they absolutely must receive inbound network traffic from public sources.
- **Create subnets in your VPCs:** Create subnets for each network layer (in groups that include multiple Availability Zones) to enhance micro-segmentation. Also verify that you have associated the correct route tables with your subnets to control routing and internet connectivity.
- **Use AWS Firewall Manager to manage your VPC security groups:** AWS Firewall Manager helps lessen the management burden of using multiple security groups.
- **Use AWS WAF to protect against common web vulnerabilities:** AWS WAF can help enhance edge security by inspecting traffic for common web vulnerabilities, such as SQL injection. It also allows you to restrict traffic from IP addresses originating from certain countries or geographical locations.
- **Use Amazon CloudFront as a content distribution network (CDN):** Amazon CloudFront can help speed up your web application by storing data closer to your users. It can also improve edge security by enforcing HTTPS, restricting access to geographic areas, and ensuring that network traffic can only access resources when routed through CloudFront.
- **Use Amazon API Gateway when creating application programming interfaces (APIs):** Amazon API Gateway helps publish, monitor, and secure REST, HTTPS, and WebSocket APIs.

Resources

Related documents:

- AWS Firewall Manager
- Amazon Inspector
- Amazon VPC Security
- Reachability Analyzer
- Amazon VPC Network Access Analyzer

Related videos:

- AWS Transit Gateway reference architectures for many VPCs
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield
- AWS re:Inforce 2022 - Validate effective network access controls on AWS
- AWS re:Inforce 2022 - Advanced protections against bots using AWS WAF

Related examples:

- Well-Architected Lab - Automated Deployment of VPC
- Workshop: Amazon VPC Network Access Analyzer

SEC05-BP02 Control traffic at all layers

When architecting your network topology, you should examine the connectivity requirements of each component. For example, if a component requires internet accessibility (inbound and outbound), connectivity to VPCs, edge services, and external data centers.

A VPC allows you to define your network topology that spans an AWS Region with a private IPv4 address range that you set, or an IPv6 address range AWS selects. You should apply multiple controls with a defense in depth approach for both inbound and outbound traffic, including the use of security groups (stateful inspection firewall), Network ACLs, subnets, and route tables. Within a VPC, you can create subnets in an Availability Zone. Each subnet can have an associated route table that defines routing rules for managing the paths that traffic takes within the subnet. You can define an internet routable subnet by having a route that goes to an internet or NAT gateway attached to the VPC, or through another VPC.

When an instance, Amazon Relational Database Service(Amazon RDS) database, or other service is launched within a VPC, it has its own security group per network interface. This firewall is outside the operating system layer and can be used to define rules for allowed inbound and outbound

traffic. You can also define relationships between security groups. For example, instances within a database tier security group only accept traffic from instances within the application tier, by reference to the security groups applied to the instances involved. Unless you are using non-TCP protocols, it shouldn't be necessary to have an Amazon Elastic Compute Cloud(Amazon EC2) instance directly accessible by the internet (even with ports restricted by security groups) without a load balancer, or CloudFront. This helps protect it from unintended access through an operating system or application issue. A subnet can also have a network ACL attached to it, which acts as a stateless firewall. You should configure the network ACL to narrow the scope of traffic allowed between layers, note that you need to define both inbound and outbound rules.

Some AWS services require components to access the internet for making API calls, where AWS API endpoints are located. Other AWS services use VPC endpoints within your Amazon VPCs. Many AWS services, including Amazon S3 and Amazon DynamoDB, support VPC endpoints, and this technology has been generalized in AWS PrivateLink. We recommend you use this approach to access AWS services, third-party services, and your own services hosted in other VPCs securely. All network traffic on AWS PrivateLink stays on the global AWS backbone and never traverses the internet. Connectivity can only be initiated by the consumer of the service, and not by the provider of the service. Using AWS PrivateLink for external service access allows you to create air-gapped VPCs with no internet access and helps protect your VPCs from external threat vectors. Third-party services can use AWS PrivateLink to allow their customers to connect to the services from their VPCs over private IP addresses. For VPC assets that need to make outbound connections to the internet, these can be made outbound only (one-way) through an AWS managed NAT gateway, outbound only internet gateway, or web proxies that you create and manage.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Control network traffic in a VPC: Implement VPC best practices to control traffic.
  - Amazon VPC security
  - VPC endpoints
  - Amazon VPC security group
  - Network ACLs
- Control traffic at the edge: Implement edge services, such as Amazon CloudFront, to provide an additional layer of protection and other features.
  - Amazon CloudFront use cases
  - AWS Global Accelerator

  - AWS Web Application Firewall (AWS WAF)
  - Amazon Route 53
  - Amazon VPC Ingress Routing
- Control private network traffic: Implement services that protect your private traffic for your workload.
  - Amazon VPC Peering
  - Amazon VPC Endpoint Services (AWS PrivateLink)
  - Amazon VPC Transit Gateway
  - AWS Direct Connect
  - AWS Site-to-Site VPN
  - AWS Client VPN
  - Amazon S3 Access Points

Resources

Related documents:

- AWS Firewall Manager
- Amazon Inspector
- Getting started with AWS WAF

Related videos:

- AWS Transit Gateway reference architectures for many VPCs
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield

Related examples:

- Lab: Automated Deployment of VPC

SEC05-BP03 Automate network protection

Automate protection mechanisms to provide a self-defending network based on threat intelligence and anomaly detection. For example, intrusion detection and prevention tools that can adapt to

current threats and reduce their impact. A web application firewall is an example of where you can automate network protection, for example, by using the AWS WAF Security Automations solution (https://github.com/awslabs/aws-waf-security-automations) to automatically block requests originating from IP addresses associated with known threat actors.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Automate protection for web-based traffic: AWS offers a solution that uses AWS CloudFormation to automatically deploy a set of AWS WAF rules designed to filter common web-based attacks. Users can select from preconfigured protective features that define the rules included in an AWS WAF web access control list (web ACL).
  - AWS WAF security automations
- Consider AWS Partner solutions: AWS Partners offer hundreds of industry-leading products that are equivalent, identical to, or integrate with existing controls in your on-premises environments. These products complement the existing AWS services to allow you to deploy a comprehensive security architecture and a more seamless experience across your cloud and on-premises environments.
  - Infrastructure security

Resources

Related documents:

- AWS Firewall Manager
- Amazon Inspector
- Amazon VPC Security
- Getting started with AWS WAF

Related videos:

- AWS Transit Gateway reference architectures for many VPCs
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield

Related examples:

- Lab: Automated Deployment of VPC

SEC05-BP04 Implement inspection and protection

Inspect and filter your traffic at each layer. You can inspect your VPC configurations for potential unintended access using VPC Network Access Analyzer. You can specify your network access requirements and identify potential network paths that do not meet them. For components transacting over HTTP-based protocols, a web application firewall can help protect from common attacks. AWS WAF is a web application firewall that lets you monitor and block HTTP(s) requests that match your configurable rules that are forwarded to an Amazon API Gateway API, Amazon CloudFront, or an Application Load Balancer. To get started with AWS WAF, you can use AWS Managed Rules in combination with your own, or use existing partner integrations.

For managing AWS WAF, AWS Shield Advanced protections, and Amazon VPC security groups across AWS Organizations, you can use AWS Firewall Manager. It allows you to centrally configure and manage firewall rules across your accounts and applications, making it easier to scale enforcement of common rules. It also allows you to rapidly respond to attacks, using AWS Shield Advanced, or solutions that can automatically block unwanted requests to your web applications. Firewall Manager also works with AWS Network Firewall. AWS Network Firewall is a managed service that uses a rules engine to give you fine-grained control over both stateful and stateless network traffic. It supports the Suricata compatible open source intrusion prevention system (IPS) specifications for rules to help protect your workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Configure Amazon GuardDuty: GuardDuty is a threat detection service that continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads. Use GuardDuty and configure automated alerts.
  - Amazon GuardDuty
  - Lab: Automated Deployment of Detective Controls
- Configure virtual private cloud (VPC) Flow Logs: VPC Flow Logs is a feature that allows you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data can be published to Amazon CloudWatch Logs and Amazon Simple Storage Service (Amazon S3). After you've created a flow log, you can retrieve and view its data in the chosen destination.

- Consider VPC traffic mirroring: Traffic mirroring is an Amazon VPC feature that you can use to copy network traffic from an elastic network interface of Amazon Elastic Compute Cloud (Amazon EC2) instances and then send it to out-of-band security and monitoring appliances for content inspection, threat monitoring, and troubleshooting.
  - VPC traffic mirroring

Resources

Related documents:

- AWS Firewall Manager
- Amazon Inspector
- Amazon VPC Security
- Getting started with AWS WAF

Related videos:

- AWS Transit Gateway reference architectures for many VPCs
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield

Related examples:

- Lab: Automated Deployment of VPC

SEC 6. How do you protect your compute resources?

Compute resources in your workload require multiple layers of defense to help protect from external and internal threats. Compute resources include EC2 instances, containers, AWS Lambda functions, database services, IoT devices, and more.

Best practices

- SEC06-BP01 Perform vulnerability management
- SEC06-BP02 Reduce attack surface
- SEC06-BP03 Implement managed services
- SEC06-BP04 Automate compute protection

- SEC06-BP05 Enable people to perform actions at a distance
- SEC06-BP06 Validate software integrity

SEC06-BP01 Perform vulnerability management

Frequently scan and patch for vulnerabilities in your code, dependencies, and in your infrastructure to help protect against new threats.

Desired outcome: Create and maintain a vulnerability management program. Regularly scan and patch resources such as Amazon EC2 instances, Amazon Elastic Container Service (Amazon ECS) containers, and Amazon Elastic Kubernetes Service (Amazon EKS) workloads. Configure maintenance windows for AWS managed resources, such as Amazon Relational Database Service (Amazon RDS) databases. Use static code scanning to inspect application source code for common issues. Consider web application penetration testing if your organization has the requisite skills or can hire outside assistance.

Common anti-patterns:

- Not having a vulnerability management program.
- Performing system patching without considering severity or risk avoidance.
- Using software that has passed its vendor-provided end of life (EOL) date.
- Deploying code into production before analyzing it for security issues.

Level of risk exposed if this best practice is not established: High

Implementation guidance

A vulnerability management program includes security assessment, identifying issues, prioritizing, and performing patch operations as part of resolving the issues. Automation is the key to continually scanning workloads for issues and unintended network exposure and performing remediation. Automating the creation and updating of resources saves time and reduces the risk of configuration errors creating further issues. A well-designed vulnerability management program should also consider vulnerability testing during the development and deployment stages of the software life cycle. Implementing vulnerability management during development and deployment helps lessen the chance that a vulnerability can make its way into your production environment.

Implementing a vulnerability management program requires a good understanding of the AWS Shared Responsibly model and how it relates to your specific workloads. Under the Shared

Responsibility Model, AWS is responsible for protecting the infrastructure of the AWS Cloud. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services. You are responsible for security in the cloud, for example, the actual data, security configuration, and management tasks of Amazon EC2 instances, and verifying that your Amazon S3 objects are properly classified and configured. Your approach to vulnerability management also can vary depending on the services you consume. For example, AWS manages the patching for our managed relational database service, Amazon RDS, but you would be responsible for patching self-hosted databases.

AWS has a range of services to help with your vulnerability management program. Amazon Inspector continually scans AWS workloads for software issues and unintended network access. AWS Systems Manager Patch Manager helps manage patching across your Amazon EC2 instances. Amazon Inspector and Systems Manager can be viewed in AWS Security Hub CSPM, a cloud security posture management service that helps automate AWS security checks and centralize security alerts.

Amazon CodeGuru can help identify potential issues in Java and Python applications using static code analysis.

Implementation steps

- **Configure Amazon Inspector:** Amazon Inspector automatically detects newly launched Amazon EC2 instances, Lambda functions, and eligible container images pushed to Amazon ECR and immediately scans them for software issues, potential defects, and unintended network exposure.
- **Scan source code:** Scan libraries and dependencies for issues and defects. Amazon CodeGuru can scan and provide recommendations to remediating common security issues for both Java and Python applications. The OWASP Foundation publishes a list of Source Code Analysis Tools (also known as SAST tools).
- **Implement a mechanism to scan and patch your existing environment, as well as scanning as part of a CI/CD pipeline build process:** Implement a mechanism to scan and patch for issues in your dependencies and operating systems to help protect against new threats. Have that mechanism run on a regular basis. Software vulnerability management is essential to understanding where you need to apply patches or address software issues. Prioritize remediation of potential security issues by embedding vulnerability assessments early into your continuous integration/continuous delivery (CI/CD) pipeline. Your approach can vary based on the AWS services that you are consuming. To check for potential issues in software running in Amazon EC2 instances, add Amazon Inspector to your pipeline to alert you and stop the build

process if issues or potential defects are detected. Amazon Inspector continually monitors resources. You can also use open source products such as OWASP Dependency-Check, Snyk, OpenVAS, package managers, and AWS Partner tools for vulnerability management.

- **Use AWS Systems Manager:** You are responsible for patch management for your AWS resources, including Amazon Elastic Compute Cloud (Amazon EC2) instances, Amazon Machine Images (AMIs), and other compute resources. AWS Systems Manager Patch Manager automates the process of patching managed instances with both security related and other types of updates. Patch Manager can be used to apply patches on Amazon EC2 instances for both operating systems and applications, including Microsoft applications, Windows service packs, and minor version upgrades for Linux based instances. In addition to Amazon EC2, Patch Manager can also be used to patch on-premises servers.

  For a list of supported operating systems, see Supported operating systems in the Systems Manager User Guide. You can scan instances to see only a report of missing patches, or you can scan and automatically install all missing patches.
- **Use AWS Security Hub CSPM:** Security Hub CSPM provides a comprehensive view of your security state in AWS. It collects security data across multiple AWS services and provides those findings in a standardized format, allowing you to prioritize security findings across AWS services.
- **Use AWS CloudFormation:** AWS CloudFormation is an infrastructure as code (IaC) service that can help with vulnerability management by automating resource deployment and standardizing resource architecture across multiple accounts and environments.

Resources

Related documents:

- AWS Systems Manager
- Security Overview of AWS Lambda
- Amazon CodeGuru
- Improved, Automated Vulnerability Management for Cloud Workloads with a New Amazon Inspector
- Automate vulnerability management and remediation in AWS using Amazon Inspector and AWS Systems Manager – Part 1

Related videos:

- Securing Serverless and Container Services
- Security best practices for the Amazon EC2 instance metadata service

SEC06-BP02 Reduce attack surface

Reduce your exposure to unintended access by hardening operating systems and minimizing the components, libraries, and externally consumable services in use. Start by reducing unused components, whether they are operating system packages or applications, for Amazon Elastic Compute Cloud (Amazon EC2)-based workloads, or external software modules in your code, for all workloads. You can find many hardening and security configuration guides for common operating systems and server software. For example, you can start with the Center for Internet Security and iterate.

In Amazon EC2, you can create your own Amazon Machine Images (AMIs), which you have patched and hardened, to help you meet the specific security requirements for your organization. The patches and other security controls you apply on the AMI are effective at the point in time in which they were created—they are not dynamic unless you modify after launching, for example, with AWS Systems Manager.

You can simplify the process of building secure AMIs with EC2 Image Builder. EC2 Image Builder significantly reduces the effort required to create and maintain golden images without writing and maintaining automation. When software updates become available, Image Builder automatically produces a new image without requiring users to manually initiate image builds. EC2 Image Builder allows you to easily validate the functionality and security of your images before using them in production with AWS-provided tests and your own tests. You can also apply AWS-provided security settings to further secure your images to meet internal security criteria. For example, you can produce images that conform to the Security Technical Implementation Guide (STIG) standard using AWS-provided templates.

Using third-party static code analysis tools, you can identify common security issues such as unchecked function input bounds, as well as applicable common vulnerabilities and exposures (CVEs). You can use Amazon CodeGuru for supported languages. Dependency checking tools can also be used to determine whether libraries your code links against are the latest versions, are themselves free of CVEs, and have licensing conditions that meet your software policy requirements.

Using Amazon Inspector, you can perform configuration assessments against your instances for known CVEs, assess against security benchmarks, and automate the notification of defects.

Amazon Inspector runs on production instances or in a build pipeline, and it notifies developers and engineers when findings are present. You can access findings programmatically and direct your team to backlogs and bug-tracking systems. EC2 Image Builder can be used to maintain server images (AMIs) with automated patching, AWS-provided security policy enforcement, and other customizations. When using containers implement ECR Image Scanning in your build pipeline and on a regular basis against your image repository to look for CVEs in your containers.

While Amazon Inspector and other tools are effective at identifying configurations and any CVEs that are present, other methods are required to test your workload at the application level. Fuzzing is a well-known method of finding bugs using automation to inject malformed data into input fields and other areas of your application.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Harden operating system: Configure operating systems to meet best practices.
  - Securing Amazon Linux
  - Securing Microsoft Windows Server
- Harden containerized resources: Configure containerized resources to meet security best practices.
- Implement AWS Lambda best practices.
  - AWS Lambda best practices

Resources

Related documents:

- AWS Systems Manager
- Replacing a Bastion Host with Amazon EC2 Systems Manager
- Security Overview of AWS Lambda

Related videos:

- Running high-security workloads on Amazon EKS
- Securing Serverless and Container Services

- Security best practices for the Amazon EC2 instance metadata service

Related examples:

- Lab: Automated Deployment of Web Application Firewall

SEC06-BP03 Implement managed services

Implement services that manage resources, such as Amazon Relational Database Service (Amazon RDS), AWS Lambda, and Amazon Elastic Container Service (Amazon ECS), to reduce your security maintenance tasks as part of the shared responsibility model. For example, Amazon RDS helps you set up, operate, and scale a relational database, automates administration tasks such as hardware provisioning, database setup, patching, and backups. This means you have more free time to focus on securing your application in other ways described in the AWS Well-Architected Framework. Lambda lets you run code without provisioning or managing servers, so you only need to focus on the connectivity, invocation, and security at the code level–not the infrastructure or operating system.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Explore available services: Explore, test, and implement services that manage resources, such as Amazon RDS, AWS Lambda, and Amazon ECS.

Resources

Related documents:

- AWS Website
- AWS Systems Manager
- Replacing a Bastion Host with Amazon EC2 Systems Manager
- Security Overview of AWS Lambda

Related videos:

- Running high-security workloads on Amazon EKS

- Securing Serverless and Container Services
- Security best practices for the Amazon EC2 instance metadata service

Related examples:

- Lab: AWS Certificate Manager Request Public Certificate

SEC06-BP04 Automate compute protection

Automate your protective compute mechanisms including vulnerability management, reduction in attack surface, and management of resources. The automation will help you invest time in securing other aspects of your workload, and reduce the risk of human error.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Automate configuration management: Enforce and validate secure configurations automatically by using a configuration management service or tool.
  - AWS Systems Manager
  - AWS CloudFormation
  - Lab: Automated deployment of VPC
  - Lab: Automated deployment of EC2 web application

- Automate patching of Amazon Elastic Compute Cloud (Amazon EC2) instances: AWS Systems Manager Patch Manager automates the process of patching managed instances with both security-related and other types of updates. You can use Patch Manager to apply patches for both operating systems and applications.
  - AWS Systems Manager Patch Manager
  - Centralized multi-account and multi-Region patching with AWS Systems Manager Automation

- Implement intrusion detection and prevention: Implement an intrusion detection and prevention tool to monitor and stop malicious activity on instances.
- Consider AWS Partner solutions: AWS Partners offer hundreds of industry-leading products that are equivalent, identical to, or integrate with existing controls in your on-premises environments. These products complement the existing AWS services to allow you to deploy a comprehensive

security architecture and a more seamless experience across your cloud and on-premises environments.

- Infrastructure security

Resources

Related documents:

- AWS CloudFormation
- AWS Systems Manager
- AWS Systems Manager Patch Manager
- Centralized multi-account and multi-region patching with AWS Systems Manager Automation
- Infrastructure security
- Replacing a Bastion Host with Amazon EC2 Systems Manager
- Security Overview of AWS Lambda

Related videos:

- Running high-security workloads on Amazon EKS
- Securing Serverless and Container Services
- Security best practices for the Amazon EC2 instance metadata service

Related examples:

- Lab: Automated Deployment of Web Application Firewall
- Lab: Automated deployment of Amazon EC2 web application

SEC06-BP05 Enable people to perform actions at a distance

Removing the ability for interactive access reduces the risk of human error, and the potential for manual configuration or management. For example, use a change management workflow to deploy Amazon Elastic Compute Cloud (Amazon EC2) instances using infrastructure-as-code, then manage Amazon EC2 instances using tools such as AWS Systems Manager instead of allowing direct access or through a bastion host. AWS Systems Manager can automate a variety of maintenance and deployment tasks, using features including automation workflows,

documents (playbooks), and the run command. AWS CloudFormation stacks build from pipelines and can automate your infrastructure deployment and management tasks without using the AWS Management Console or APIs directly.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Replace console access: Replace console access (SSH or RDP) to instances with AWS Systems Manager Run Command to automate management tasks.

- AWS Systems Manager Run Command

Resources

Related documents:

- AWS Systems Manager
- AWS Systems Manager Run Command
- Replacing a Bastion Host with Amazon EC2 Systems Manager
- Security Overview of AWS Lambda

Related videos:

- Running high-security workloads on Amazon EKS
- Securing Serverless and Container Services
- Security best practices for the Amazon EC2 instance metadata service

Related examples:

- Lab: Automated Deployment of Web Application Firewall

SEC06-BP06 Validate software integrity

Implement mechanisms (for example, code signing) to validate that the software, code and libraries used in the workload are from trusted sources and have not been tampered with. For example, you should verify the code signing certificate of binaries and scripts to confirm the

author, and ensure it has not been tampered with since created by the author. AWS Signer can help ensure the trust and integrity of your code by centrally managing the code- signing lifecycle, including signing certification and public and private keys. You can learn how to use advanced patterns and best practices for code signing with AWS Lambda. Additionally, a checksum of software that you download, compared to that of the checksum from the provider, can help ensure it has not been tampered with.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Investigate mechanisms: Code signing is one mechanism that can be used to validate software integrity.
  - NIST: Security Considerations for Code Signing

Resources

Related documents:

- AWS Signer
- New – Code Signing, a Trust and Integrity Control for AWS Lambda

Data protection

Questions

- SEC 7. How do you classify your data?
- SEC 8. How do you protect your data at rest?
- SEC 9. How do you protect your data in transit?

SEC 7. How do you classify your data?

Classification provides a way to categorize data, based on criticality and sensitivity in order to help you determine appropriate protection and retention controls.

Best practices

- SEC07-BP01 Identify the data within your workload
- SEC07-BP02 Define data protection controls

- SEC07-BP03 Automate identification and classification
- SEC07-BP04 Define data lifecycle management

SEC07-BP01 Identify the data within your workload

It's critical to understand the type and classification of data your workload is processing, the associated business processes, where the data is stored, and who is the data owner. You should also have an understanding of the applicable legal and compliance requirements of your workload, and what data controls need to be enforced. Identifying data is the first step in the data classification journey.

Benefits of establishing this best practice:

Data classification allows workload owners to identify locations that store sensitive data and determine how that data should be accessed and shared.

Data classification aims to answer the following questions:

- **What type of data do you have?**

  This could be data such as:

  - Intellectual property (IP) such as trade secrets, patents, or contract agreements.
  - Protected health information (PHI) such as medical records that contain medical history information connected to an individual.
  - Personally identifiable information (PII), such as name, address, date of birth, and national ID or registration number.
  - Credit card data, such as the Primary Account Number (PAN), cardholder name, expiration date, and service code number.
  - Where is the sensitive data is stored?
  - Who can access, modify, and delete data?
  - Understanding user permissions is essential in guarding against potential data mishandling.
- **Who can perform create, read, update, and delete (CRUD) operations?**
  - Account for potential escalation of privileges by understanding who can manage permissions to the data.
- **What business impact might occur if the data is disclosed unintentionally, altered, or deleted?**

- Understand the risk consequence if data is modified, deleted, or inadvertently disclosed.

By knowing the answers to these questions, you can take the following actions:

- Decrease sensitive data scope (such as the number of sensitive data locations) and limit access to sensitive data to only approved users.
- Gain an understanding of different data types so that you can implement appropriate data protection mechanisms and techniques, such as encryption, data loss prevention, and identity and access management.
- Optimize costs by delivering the right control objectives for the data.
- Confidently answer questions from regulators and auditors regarding the types and amount of data, and how data of different sensitivities are isolated from each other.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Data classification is the act of identifying the sensitivity of data. It might involve tagging to make the data easily searchable and trackable. Data classification also reduces the duplication of data, which can help reduce storage and backup costs while speeding up the search process.

Use services such as Amazon Macie to automate at scale both the discovery and classification of sensitive data. Other services, such as Amazon EventBridge and AWS Config, can be used to automate remediation for data security issues such as unencrypted Amazon Simple Storage Service (Amazon S3) buckets and Amazon EC2 EBS volumes or untagged data resources. For a complete list of AWS service integrations, see the EventBridge documentation.

Detecting PII in unstructured data such as customer emails, support tickets, product reviews, and social media, is possible by using Amazon Comprehend, which is a natural language processing (NLP) service that uses machine learning (ML) to find insights and relationships like people, places, sentiments, and topics in unstructured text. For a list of AWS services that can assist with data identification, see Common techniques to detect PHI and PII data using AWS services.

Another method that supports data classification and protection is AWS resource tagging. Tagging allows you to assign metadata to your AWS resources that you can use to manage, identify, organize, search for, and filter resources.

In some cases, you might choose to tag entire resources (such as an S3 bucket), especially when a specific workload or service is expected to store processes or transmissions of already known data classification.

Where appropriate, you can tag an S3 bucket instead of individual objects for ease of administration and security maintenance.

Implementation steps

Detect sensitive data within Amazon S3:

1. Before starting, make sure you have the appropriate permissions to access the Amazon Macie console and API operations. For additional details, see Getting started with Amazon Macie.
2. Use Amazon Macie to perform automated data discovery when your sensitive data resides in Amazon S3.
   - Use the Getting Started with Amazon Macie guide to configure a repository for sensitive data discovery results and create a discovery job for sensitive data.
   - How to use Amazon Macie to preview sensitive data in S3 buckets.

     By default, Macie analyzes objects by using the set of managed data identifiers that we recommend for automated sensitive data discovery. You can tailor the analysis by configuring Macie to use specific managed data identifiers, custom data identifiers, and allow lists when it performs automated sensitive data discovery for your account or organization. You can adjust the scope of the analysis by excluding specific buckets (for example, S3 buckets that typically store AWS logging data).
3. To configure and use automated sensitive data discovery, see Performing automated sensitive data discovery with Amazon Macie.
4. You might also consider Automated Data Discovery for Amazon Macie.

Detect sensitive data within Amazon RDS:

For more information on data discovery in Amazon Relational Database Service (Amazon RDS) databases, see Enabling data classification for Amazon RDS database with Macie.

Detect sensitive data within DynamoDB:

- Detecting sensitive data in DynamoDB with Macie explains how to use Amazon Macie to detect sensitive data in Amazon DynamoDB tables by exporting the data to Amazon S3 for scanning.

AWS Partner solutions:

- Consider using our extensive AWS Partner Network. AWS Partners have extensive tools and compliance frameworks that directly integrate with AWS services. Partners can provide you with a tailored governance and compliance solution to help you meet your organizational needs.
- For customized solutions in data classification, see Data governance in the age of regulation and compliance requirements.

You can automatically enforce the tagging standards that your organization adopts by creating and deploying policies using AWS Organizations. Tag policies let you specify rules that define valid key names and what values are valid for each key. You can choose to monitor only, which gives you an opportunity to evaluate and clean up your existing tags. After your tags are in compliance with your chosen standards, you can turn on enforcement in the tag policies to prevent non-compliant tags from being created. For more details, see Securing resource tags used for authorization using a service control policy in AWS Organizations and the example policy on preventing tags from being modified except by authorized principals.

- To begin using tag policies in AWS Organizations, it's strongly recommended that you follow the workflow in Getting started with tag policies before moving on to more advanced tag policies. Understanding the effects of attaching a simple tag policy to a single account before expanding to an entire organizational unit (OU) or organization allows you to see a tag policy's effects before you enforce compliance with the tag policy. Getting started with tag policies provides links to instructions for more advanced policy-related tasks.
- Consider evaluating other AWS services and features that support data classification, which are listed in the Data Classification whitepaper.

Resources

Related documents:

- Getting Started with Amazon Macie
- Automated data discovery with Amazon Macie
- Getting started with tag policies
- Detecting PII entities

Related blogs:

- How to use Amazon Macie to preview sensitive data in S3 buckets.
- Performing automated sensitive data discovery with Amazon Macie.
- Common techniques to detect PHI and PII data using AWS Services
- Detecting and redacting PII using Amazon Comprehend
- Securing resource tags used for authorization using a service control policy in AWS Organizations
- Enabling data classification for Amazon RDS database with Macie
- Detecting sensitive data in DynamoDB with Macie
-

Related videos:

- Event-driven data security using Amazon Macie
- Amazon Macie for data protection and governance
- Fine-tune sensitive data findings with allow lists

SEC07-BP02 Define data protection controls

Protect data according to its classification level. For example, secure data classified as public by using relevant recommendations while protecting sensitive data with additional controls.

By using resource tags, separate AWS accounts per sensitivity (and potentially also for each caveat, enclave, or community of interest), IAM policies, AWS Organizations SCPs, AWS Key Management Service (AWS KMS), and AWS CloudHSM, you can define and implement your policies for data classification and protection with encryption. For example, if you have a project with S3 buckets that contain highly critical data or Amazon Elastic Compute Cloud (Amazon EC2) instances that process confidential data, they can be tagged with a `Project=ABC` tag. Only your immediate team knows what the project code means, and it provides a way to use attribute-based access control. You can define levels of access to the AWS KMS encryption keys through key policies and grants to ensure that only appropriate services have access to the sensitive content through a secure mechanism. If you are making authorization decisions based on tags you should make sure that the permissions on the tags are defined appropriately using tag policies in AWS Organizations.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Define your data identification and classification schema: Identification and classification of your data is performed to assess the potential impact and type of data you store, and who can access it.
  - AWS Documentation
- Discover available AWS controls: For the AWS services you are or plan to use, discover the security controls. Many services have a security section in their documentation.
  - AWS Documentation
- Identify AWS compliance resources: Identify resources that AWS has available to assist.
  - https://aws.amazon.com/compliance/

Resources

Related documents:

- AWS Documentation
- Data Classification whitepaper
- Getting started with Amazon Macie
- AWS Compliance

Related videos:

- Introducing the New Amazon Macie

SEC07-BP03 Automate identification and classification

Automating the identification and classification of data can help you implement the correct controls. Using automation for this instead of direct access from a person reduces the risk of human error and exposure. You should evaluate using a tool, such as Amazon Macie, that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. Amazon Macie recognizes sensitive data, such as personally identifiable information (PII) or intellectual property, and provides you with dashboards and alerts that give visibility into how this data is being accessed or moved.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Use Amazon Simple Storage Service (Amazon S3) Inventory: Amazon S3 inventory is one of the tools you can use to audit and report on the replication and encryption status of your objects.
  - Amazon S3 Inventory
- Consider Amazon Macie: Amazon Macie uses machine learning to automatically discover and classify data stored in Amazon S3.
  - Amazon Macie

Resources

Related documents:

- Amazon Macie
- Amazon S3 Inventory
- Data Classification Whitepaper
- Getting started with Amazon Macie

Related videos:

- Introducing the New Amazon Macie

SEC07-BP04 Define data lifecycle management

Your defined lifecycle strategy should be based on sensitivity level as well as legal and organization requirements. Aspects including the duration for which you retain data, data destruction processes, data access management, data transformation, and data sharing should be considered. When choosing a data classification methodology, balance usability versus access. You should also accommodate the multiple levels of access and nuances for implementing a secure, but still usable, approach for each level. Always use a defense in depth approach and reduce human access to data and mechanisms for transforming, deleting, or copying data. For example, require users to strongly authenticate to an application, and give the application, rather than the users, the requisite access permission to perform action at a distance. In addition, ensure that users come from a trusted network path and require access to the decryption keys. Use tools, such as dashboards and automated reporting, to give users information from the data rather than giving them direct access to the data.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Identify data types: Identify the types of data that you are storing or processing in your workload. That data could be text, images, binary databases, and so forth.

Resources

Related documents:

- Data Classification Whitepaper
- Getting started with Amazon Macie

Related videos:

- Introducing the New Amazon Macie

SEC 8. How do you protect your data at rest?

Protect your data at rest by implementing multiple controls, to reduce the risk of unauthorized access or mishandling.

Best practices

- SEC08-BP01 Implement secure key management
- SEC08-BP02 Enforce encryption at rest
- SEC08-BP03 Automate data at rest protection
- SEC08-BP04 Enforce access control
- SEC08-BP05 Use mechanisms to keep people away from data

SEC08-BP01 Implement secure key management

By defining an encryption approach that includes the storage, rotation, and access control of keys, you can help provide protection for your content against unauthorized users and against unnecessary exposure to authorized users. AWS Key Management Service (AWS KMS) helps you manage encryption keys and integrates with many AWS services. This service provides durable, secure, and redundant storage for your AWS KMS keys. You can define your key aliases as well as

key-level policies. The policies help you define key administrators as well as key users. Additionally, AWS CloudHSM is a cloud-based hardware security module (HSM) that allows you to easily generate and use your own encryption keys in the AWS Cloud. It helps you meet corporate, contractual, and regulatory compliance requirements for data security by using FIPS 140-2 Level 3 validated HSMs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Implement AWS KMS: AWS KMS makes it easy for you to create and manage keys and control the use of encryption across a wide range of AWS services and in your applications. AWS KMS is a secure and resilient service that uses FIPS 140-2 validated hardware security modules to protect your keys.
  - Getting started: AWS Key Management Service (AWS KMS)
- Consider AWS Encryption SDK: Use the AWS Encryption SDK with AWS KMS integration when your application needs to encrypt data client-side.
  - AWS Encryption SDK

Resources

Related documents:

- AWS Key Management Service
- AWS cryptographic services and tools
- Getting started: AWS Key Management Service (AWS KMS)
- Protecting Amazon S3 Data Using Encryption

Related videos:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS

SEC08-BP02 Enforce encryption at rest

You should enforce the use of encryption for data at rest. Encryption maintains the confidentiality of sensitive data in the event of unauthorized access or accidental disclosure.

Desired outcome: Private data should be encrypted by default when at rest. Encryption helps maintain confidentiality of the data and provides an additional layer of protection against intentional or inadvertent data disclosure or exfiltration. Data that is encrypted cannot be read or accessed without first unencrypting the data. Any data stored unencrypted should be inventoried and controlled.

Common anti-patterns:

- Not using encrypt-by-default configurations.
- Providing overly permissive access to decryption keys.
- Not monitoring the use of encryption and decryption keys.
- Storing data unencrypted.
- Using the same encryption key for all data regardless of data usage, types, and classification.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Map encryption keys to data classifications within your workloads. This approach helps protect against overly permissive access when using either a single, or very small number of encryption keys for your data (see SEC07-BP01 Identify the data within your workload).

AWS Key Management Service (AWS KMS) integrates with many AWS services to make it easier to encrypt your data at rest. For example, in Amazon Simple Storage Service (Amazon S3), you can set default encryption on a bucket so that new objects are automatically encrypted. When using AWS KMS, consider how tightly the data needs to be restricted. Default and service-controlled AWS KMS keys are managed and used on your behalf by AWS. For sensitive data that requires fine-grained access to the underlying encryption key, consider customer managed keys (CMKs). You have full control over CMKs, including rotation and access management through the use of key policies.

Additionally, Amazon Elastic Compute Cloud (Amazon EC2) and Amazon S3 support the enforcement of encryption by setting default encryption. You can use AWS Config Rules to check automatically that you are using encryption, for example, for Amazon Elastic Block Store (Amazon EBS) volumes, Amazon Relational Database Service (Amazon RDS) instances, and Amazon S3 buckets.

AWS also provides options for client-side encryption, allowing you to encrypt data prior to uploading it to the cloud. The AWS Encryption SDK provides a way to encrypt your data using

envelope encryption. You provide the wrapping key, and the AWS Encryption SDK generates a unique data key for each data object it encrypts. Consider AWS CloudHSM if you need a managed single-tenant hardware security module (HSM). AWS CloudHSM allows you to generate, import, and manage cryptographic keys on a FIPS 140-2 level 3 validated HSM. Some use cases for AWS CloudHSM include protecting private keys for issuing a certificate authority (CA), and turning on transparent data encryption (TDE) for Oracle databases. The AWS CloudHSM Client SDK provides software that allows you to encrypt data client side using keys stored inside AWS CloudHSM prior to uploading your data into AWS. The Amazon DynamoDB Encryption Client also allows you to encrypt and sign items prior to upload into a DynamoDB table.

Implementation steps

- **Enforce encryption at rest for Amazon S3:** Implement Amazon S3 bucket default encryption.

  Configure default encryption for new Amazon EBS volumes: Specify that you want all newly created Amazon EBS volumes to be created in encrypted form, with the option of using the default key provided by AWS or a key that you create.

  Configure encrypted Amazon Machine Images (AMIs): Copying an existing AMI with encryption configured will automatically encrypt root volumes and snapshots.

  Configure Amazon RDS encryption: Configure encryption for your Amazon RDS database clusters and snapshots at rest by using the encryption option.

  Create and configure AWS KMS keys with policies that limit access to the appropriate principals for each classification of data: For example, create one AWS KMS key for encrypting production data and a different key for encrypting development or test data. You can also provide key access to other AWS accounts. Consider having different accounts for your development and production environments. If your production environment needs to decrypt artifacts in the development account, you can edit the CMK policy used to encrypt the development artifacts to give the production account the ability to decrypt those artifacts. The production environment can then ingest the decrypted data for use in production.

  Configure encryption in additional AWS services: For other AWS services you use, review the security documentation for that service to determine the service's encryption options.

Resources

Related documents:

- AWS Crypto Tools
- AWS Encryption SDK
- AWS KMS Cryptographic Details Whitepaper
- AWS Key Management Service
- AWS cryptographic services and tools
- Amazon EBS Encryption
- Default encryption for Amazon EBS volumes
- Encrypting Amazon RDS Resources
- How do I enable default encryption for an Amazon S3 bucket?
- Protecting Amazon S3 Data Using Encryption

Related videos:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS

SEC08-BP03 Automate data at rest protection

Use automated tools to validate and enforce data at rest controls continuously, for example, verify that there are only encrypted storage resources. You can automate validation that all EBS volumes are encrypted using AWS Config Rules. AWS Security Hub CSPM can also verify several different controls through automated checks against security standards. Additionally, your AWS Config Rules can automatically remediate noncompliant resources.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Data at rest represents any data that you persist in non-volatile storage for any duration in your workload. This includes block storage, object storage, databases, archives, IoT devices, and any other storage medium on which data is persisted. Protecting your data at rest reduces the risk of unauthorized access, when encryption and appropriate access controls are implemented.

Enforce encryption at rest: You should ensure that the only way to store data is by using encryption. AWS KMS integrates seamlessly with many AWS services to make it easier for you

to encrypt all your data at rest. For example, in Amazon Simple Storage Service (Amazon S3) you can set default encryption on a bucket so that all new objects are automatically encrypted. Additionally, Amazon EC2 and Amazon S3 support the enforcement of encryption by setting default encryption. You can use AWS Managed Config Rules to check automatically that you are using encryption, for example, for EBS volumes, Amazon Relational Database Service (Amazon RDS) instances, and Amazon S3 buckets.

Resources

Related documents:

- AWS Crypto Tools
- AWS Encryption SDK

Related videos:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS

SEC08-BP04 Enforce access control

To help protect your data at rest, enforce access control using mechanisms, such as isolation and versioning, and apply the principle of least privilege. Prevent the granting of public access to your data.

Desired outcome: Verify that only authorized users can access data on a need-to-know basis. Protect your data with regular backups and versioning to prevent against intentional or inadvertent modification or deletion of data. Isolate critical data from other data to protect its confidentiality and data integrity.

Common anti-patterns:

- Storing data with different sensitivity requirements or classification together.
- Using overly permissive permissions on decryption keys.
- Improperly classifying data.
- Not retaining detailed backups of important data.
- Providing persistent access to production data.

- Not auditing data access or regularly reviewing permissions.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Multiple controls can help protect your data at rest, including access (using least privilege), isolation, and versioning. Access to your data should be audited using detective mechanisms, such as AWS CloudTrail, and service level logs, such as Amazon Simple Storage Service (Amazon S3) access logs. You should inventory what data is publicly accessible, and create a plan to reduce the amount of publicly available data over time.

Amazon Glacier Vault Lock and Amazon S3 Object Lock provide mandatory access control for objects in Amazon S3—once a vault policy is locked with the compliance option, not even the root user can change it until the lock expires.

Implementation steps

- **Enforce access control**: Enforce access control with least privileges, including access to encryption keys.
- **Separate data based on different classification levels**: Use different AWS accounts for data classification levels, and manage those accounts using AWS Organizations.
- **Review AWS Key Management Service (AWS KMS) policies**: Review the level of access granted in AWS KMS policies.
- **Review Amazon S3 bucket and object permissions**: Regularly review the level of access granted in S3 bucket policies. Best practice is to avoid using publicly readable or writeable buckets. Consider using AWS Config to detect buckets that are publicly available, and Amazon CloudFront to serve content from Amazon S3. Verify that buckets that should not allow public access are properly configured to prevent public access. By default, all S3 buckets are private, and can only be accessed by users that have been explicitly granted access.
- **Use AWS IAM Access Analyzer:** IAM Access Analyzer analyzes Amazon S3 buckets and generates a finding when an S3 policy grants access to an external entity.
- **Use Amazon S3 versioning and object lock when appropriate**.
- **Use Amazon S3 Inventory**: Amazon S3 Inventory can be used to audit and report on the replication and encryption status of your S3 objects.
- **Review Amazon EBS and AMI sharing permissions**: Sharing permissions can allow images and volumes to be shared with AWS accounts that are external to your workload.

- **Review AWS Resource Access Manager Shares periodically to determine whether resources should continue to be shared.** Resource Access Manager allows you to share resources, such as AWS Network Firewall policies, Amazon Route 53 resolver rules, and subnets, within your Amazon VPCs. Audit shared resources regularly and stop sharing resources which no longer need to be shared.

Resources

Related best practices:

- SEC03-BP01 Define access requirements
- SEC03-BP02 Grant least privilege access

Related documents:

- AWS KMS Cryptographic Details Whitepaper
- Introduction to Managing Access Permissions to Your Amazon S3 Resources
- Overview of managing access to your AWS KMS resources
- AWS Config Rules
- Amazon S3 + Amazon CloudFront: A Match Made in the Cloud
- Using versioning
- Locking Objects Using Amazon S3 Object Lock
- Sharing an Amazon EBS Snapshot
- Shared AMIs
- Hosting a single-page application on Amazon S3

Related videos:

- Securing Your Block Storage on AWS

SEC08-BP05 Use mechanisms to keep people away from data

Keep all users away from directly accessing sensitive data and systems under normal operational circumstances. For example, use a change management workflow to manage Amazon Elastic

Compute Cloud (Amazon EC2) instances using tools instead of allowing direct access or a bastion host. This can be achieved using AWS Systems Manager Automation, which uses automation documents that contain steps you use to perform tasks. These documents can be stored in source control, be peer reviewed before running, and tested thoroughly to minimize risk compared to shell access. Business users could have a dashboard instead of direct access to a data store to run queries. Where CI/CD pipelines are not used, determine which controls and processes are required to adequately provide a normally deactivated break-glass access mechanism.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Implement mechanisms to keep people away from data: Mechanisms include using dashboards, such as Quick, to display data to users instead of directly querying.
  - Quick
- Automate configuration management: Perform actions at a distance, enforce and validate secure configurations automatically by using a configuration management service or tool. Avoid use of bastion hosts or directly accessing EC2 instances.
  - AWS Systems Manager
  - AWS CloudFormation
  - CI/CD Pipeline for AWS CloudFormation templates on AWS

Resources

Related documents:

- AWS KMS Cryptographic Details Whitepaper

Related videos:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS

SEC 9. How do you protect your data in transit?

Protect your data in transit by implementing multiple controls to reduce the risk of unauthorized access or loss.

Best practices

- SEC09-BP01 Implement secure key and certificate management
- SEC09-BP02 Enforce encryption in transit
- SEC09-BP03 Automate detection of unintended data access
- SEC09-BP04 Authenticate network communications

SEC09-BP01 Implement secure key and certificate management

Store encryption keys and certificates securely and rotate them at appropriate time intervals with strict access control. The best way to accomplish this is to use a managed service, such as AWS Certificate Manager (ACM). It lets you easily provision, manage, and deploy public and private Transport Layer Security (TLS) certificates for use with AWS services and your internal connected resources. TLS certificates are used to secure network communications and establish the identity of websites over the internet as well as resources on private networks. ACM integrates with AWS resources, such as Elastic Load Balancers (ELBs), AWS distributions, and APIs on API Gateway, also handling automatic certificate renewals. If you use ACM to deploy a private root CA, both certificates and private keys can be provided by it for use in Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and so on.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Implement secure key and certificate management: Implement your defined secure key and certificate management solution.
  - AWS Certificate Manager
  - How to host and manage an entire private certificate infrastructure in AWS
- Implement secure protocols: Use secure protocols that offer authentication and confidentiality, such as Transport Layer Security (TLS) or IPsec, to reduce the risk of data tampering or loss. Check the AWS documentation for the protocols and security relevant to the services that you are using.

Resources

Related documents:

- AWS Documentation

SEC09-BP02 Enforce encryption in transit

Enforce your defined encryption requirements based on your organization's policies, regulatory obligations and standards to help meet organizational, legal, and compliance requirements. Only use protocols with encryption when transmitting sensitive data outside of your virtual private cloud (VPC). Encryption helps maintain data confidentiality even when the data transits untrusted networks.

Desired outcome: All data should be encrypted in transit using secure TLS protocols and cipher suites. Network traffic between your resources and the internet must be encrypted to mitigate unauthorized access to the data. Network traffic solely within your internal AWS environment should be encrypted using TLS wherever possible. The AWS internal network is encrypted by default and network traffic within a VPC cannot be spoofed or sniffed unless an unauthorized party has gained access to whatever resource is generating traffic (such as Amazon EC2 instances, and Amazon ECS containers). Consider protecting network-to-network traffic with an IPsec virtual private network (VPN).

Common anti-patterns:

- Using deprecated versions of SSL, TLS, and cipher suite components (for example, SSL v3.0, 1024-bit RSA keys, and RC4 cipher).
- Allowing unencrypted (HTTP) traffic to or from public-facing resources.
- Not monitoring and replacing X.509 certificates prior to expiration.
- Using self-signed X.509 certificates for TLS.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS services provide HTTPS endpoints using TLS for communication, providing encryption in transit when communicating with the AWS APIs. Insecure protocols like HTTP can be audited and blocked in a VPC through the use of security groups. HTTP requests can also be automatically redirected to HTTPS in Amazon CloudFront or on an Application Load Balancer. You have full control over your computing resources to implement encryption in transit across your services. Additionally, you can use VPN connectivity into your VPC from an external network or AWS Direct Connect to facilitate encryption of traffic. Verify that your clients are making calls to AWS APIs using at least TLS 1.2, as AWS is deprecating the use of earlier versions of TLS in June 2023. AWS

recommends using TLS 1.3. Third-party solutions are available in the AWS Marketplace if you have special requirements.

Implementation steps

- **Enforce encryption in transit:** Your defined encryption requirements should be based on the latest standards and best practices and only allow secure protocols. For example, configure a security group to only allow the HTTPS protocol to an application load balancer or Amazon EC2 instance.
- **Configure secure protocols in edge services:** Configure HTTPS with Amazon CloudFront and use a security profile appropriate for your security posture and use case.
- **Use a VPN for external connectivity:** Consider using an IPsec VPN for securing point-to-point or network-to-network connections to help provide both data privacy and integrity.
- **Configure secure protocols in load balancers:** Select a security policy that provides the strongest cipher suites supported by the clients that will be connecting to the listener. Create an HTTPS listener for your Application Load Balancer.
- **Configure secure protocols in Amazon Redshift:** Configure your cluster to require a secure socket layer (SSL) or transport layer security (TLS) connection.
- **Configure secure protocols:** Review AWS service documentation to determine encryption-in-transit capabilities.
- **Configure secure access when uploading to Amazon S3 buckets:** Use Amazon S3 bucket policy controls to enforce secure access to data.
- **Consider using AWS Certificate Manager:** ACM allows you to provision, manage, and deploy public TLS certificates for use with AWS services.
- **Consider using AWS Private Certificate Authority for private PKI needs:** AWS Private CA allows you to create private certificate authority (CA) hierarchies to issue end-entity X.509 certificates that can be used to create encrypted TLS channels.

Resources

Related documents:

- Using HTTPS with CloudFront
- Connect your VPC to remote networks using AWS Virtual Private Network
- Create an HTTPS listener for your Application Load Balancer
- Tutorial: Configure SSL/TLS on Amazon Linux 2

- Using SSL/TLS to encrypt a connection to a DB instance
- Configuring security options for connections

SEC09-BP03 Automate detection of unintended data access

Use tools such as Amazon GuardDuty to automatically detect suspicious activity or attempts to move data outside of defined boundaries. For example, GuardDuty can detect Amazon Simple Storage Service (Amazon S3) read activity that is unusual with the Exfiltration:S3/AnomalousBehavior finding. In addition to GuardDuty, Amazon VPC Flow Logs, which capture network traffic information, can be used with Amazon EventBridge to detect connections, both successful and denied. Amazon S3 Access Analyzer can help assess what data is accessible to who in your Amazon S3 buckets.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Automate detection of unintended data access: Use a tool or detection mechanism to automatically detect attempts to move data outside of defined boundaries, for example, to detect a database system that is copying data to an unrecognized host.
  - VPC Flow Logs
- Consider Amazon Macie: Amazon Macie is a fully managed data security and data privacy service that uses machine learning and pattern matching to discover and protect your sensitive data in AWS.
  - Amazon Macie

Resources

Related documents:

- VPC Flow Logs
- Amazon Macie

SEC09-BP04 Authenticate network communications

Verify the identity of communications by using protocols that support authentication, such as Transport Layer Security (TLS) or IPsec.

Using network protocols that support authentication, allows for trust to be established between the parties. This adds to the encryption used in the protocol to reduce the risk of communications being altered or intercepted. Common protocols that implement authentication include Transport Layer Security (TLS), which is used in many AWS services, and IPsec, which is used in AWS Virtual Private Network (Site-to-Site VPN).

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Implement secure protocols: Use secure protocols that offer authentication and confidentiality, such as TLS or IPsec, to reduce the risk of data tampering or loss. Check the AWS documentation for the protocols and security relevant to the services you are using.

Resources

Related documents:

- AWS Documentation

Incident response

Question

- SEC 10. How do you anticipate, respond to, and recover from incidents?

SEC 10. How do you anticipate, respond to, and recover from incidents?

Preparation is critical to timely and effective investigation, response to, and recovery from security incidents to help minimize disruption to your organization.

Best practices

- SEC10-BP01 Identify key personnel and external resources
- SEC10-BP02 Develop incident management plans
- SEC10-BP03 Prepare forensic capabilities
- SEC10-BP04 Automate containment capability
- SEC10-BP05 Pre-provision access

- SEC10-BP06 Pre-deploy tools
- SEC10-BP07 Run game days

SEC10-BP01 Identify key personnel and external resources

Identify internal and external personnel, resources, and legal obligations that would help your organization respond to an incident.

When you define your approach to incident response in the cloud, in unison with other teams (such as your legal counsel, leadership, business stakeholders, AWS Support Services, and others), you must identify key personnel, stakeholders, and relevant contacts. To reduce dependency and decrease response time, make sure that your team, specialist security teams, and responders are educated about the services that you use and have opportunities to practice hands-on.

We encourage you to identify external AWS security partners that can provide you with outside expertise and a different perspective to augment your response capabilities. Your trusted security partners can help you identify potential risks or threats that you might not be familiar with.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Identify key personnel in your organization: Maintain a contact list of personnel within your organization that you would need to involve to respond to and recover from an incident.
- Identify external partners: Engage with external partners if necessary that can help you respond to and recover from an incident.

Resources

Related documents:

- AWS Incident Response Guide

Related videos:

- Prepare for and respond to security incidents in your AWS environment

Related examples:

SEC10-BP02 Develop incident management plans

This best practice was updated with new guidance on July 13th, 2023.

The first document to develop for incident response is the incident response plan. The incident response plan is designed to be the foundation for your incident response program and strategy.

Benefits of establishing this best practice: Developing thorough and clearly defined incident response processes is key to a successful and scalable incident response program. When a security event occurs, clear steps and workflows will help you to respond in a timely manner. You might already have existing incident response processes. Regardless of your current state, it's important to update, iterate, and test your incident response processes regularly.

Level of risk exposed if this best practice is not established: High

Implementation guidance

An incident management plan is critical to respond, mitigate, and recover from the potential impact of security incidents. An incident management plan is a structured process for identifying, remediating, and responding in a timely matter to security incidents.

The cloud has many of the same operational roles and requirements found in an on-premises environment. When creating an incident management plan, it is important to factor response and recovery strategies that best align with your business outcome and compliance requirements. For example, if you are operating workloads in AWS that are FedRAMP compliant in the United States, it's useful to adhere to NIST SP 800-61 Computer Security Handling Guide. Similarly, when operating workloads with European personally identifiable information (PII) data, consider scenarios like how you might protect and respond to issues related to data residency as mandated by EU General Data Protection Regulation (GDPR) Regulations.

When building an incident management plan for your workloads operating in AWS, start with the AWS Shared Responsibility Model for building a defense-in-depth approach towards incident response. In this model, AWS manages security of the cloud, and you are responsible for security in the cloud. This means that you retain control and are responsible for the security controls you choose to implement. The AWS Security Incident Response Guide details key concepts and foundational guidance for building a cloud-centric incident management plan.

An effective incident management plan must be continually iterated upon, remaining current with your cloud operations goal. Consider using the implementation plans detailed below as you create and evolve your incident management plan.

Implementation steps

Define roles and responsibilities

Handling security events requires cross-organizational discipline and an inclination for action. Within your organizational structure, there should be many people who are responsible, accountable, consulted, or kept informed during an incident, such as representatives from human resources (HR), the executive team, and legal. Consider these roles and responsibilities, and whether any third parties must be involved. Note that many geographies have local laws that govern what should and should not be done. Although it might seem bureaucratic to build a responsible, accountable, consulted, and informed (RACI) chart for your security response plans, doing so facilitates quick and direct communication and clearly outlines the leadership across different stages of the event.

During an incident, including the owners and developers of impacted applications and resources is key because they are subject matter experts (SMEs) that can provide information and context to aid in measuring impact. Make sure to practice and build relationships with the developers and application owners before you rely on their expertise for incident response. Application owners or SMEs, such as your cloud administrators or engineers, might need to act in situations where the environment is unfamiliar or has complexity, or where the responders don't have access.

Lastly, trusted partners might be involved in the investigation or response because they can provide additional expertise and valuable scrutiny. When you don't have these skills on your own team, you might want to hire an external party for assistance.

Understand AWS response teams and support

- **AWS Support**
  - Support offers a range of plans that provide access to tools and expertise that support the success and operational health of your AWS solutions. If you need technical support and more resources to help plan, deploy, and optimize your AWS environment, you can select a support plan that best aligns with your AWS use case.
  - Consider the Support Center in AWS Management Console (sign-in required) as the central point of contact to get support for issues that affect your AWS resources. Access to Support

is controlled by AWS Identity and Access Management. For more information about getting access to Support features, see Getting started with Support.

- **AWS Customer Incident Response Team (CIRT)**
  - The AWS Customer Incident Response Team (CIRT) is a specialized 24/7 global AWS team that provides support to customers during active security events on the customer side of the AWS Shared Responsibility Model.
  - When the AWS CIRT supports you, they provide assistance with triage and recovery for an active security event on AWS. They can assist in root cause analysis through the use of AWS service logs and provide you with recommendations for recovery. They can also provide security recommendations and best practices to help you avoid security events in the future.
  - AWS customers can engage the AWS CIRT through an Support case.
- **DDoS response support**
  - AWS offers AWS Shield, which provides a managed distributed denial of service (DDoS) protection service that safeguards web applications running on AWS. Shield provides always-on detection and automatic inline mitigations that can minimize application downtime and latency, so there is no need to engage Support to benefit from DDoS protection. There are two tiers of Shield: AWS Shield Standard and AWS Shield Advanced. To learn about the differences between these two tiers, see Shield features documentation.
- **AWS Managed Services (AMS)**
  - AWS Managed Services (AMS) provides ongoing management of your AWS infrastructure so you can focus on your applications. By implementing best practices to maintain your infrastructure, AMS helps reduce your operational overhead and risk. AMS automates common activities such as change requests, monitoring, patch management, security, and backup services, and provides full-lifecycle services to provision, run, and support your infrastructure.
  - AMS takes responsibility for deploying a suite of security detective controls and provides a 24/7 first line of response to alerts. When an alert is initiated, AMS follows a standard set of automated and manual playbooks to verify a consistent response. These playbooks are shared with AMS customers during onboarding so that they can develop and coordinate a response with AMS.

Develop the incident response plan

The incident response plan is designed to be the foundation for your incident response program and strategy. The incident response plan should be in a formal document. An incident response plan typically includes these sections:

- **An incident response team overview:** Outlines the goals and functions of the incident response team.
- **Roles and responsibilities:** Lists the incident response stakeholders and details their roles when an incident occurs.
- **A communication plan:** Details contact information and how you will communicate during an incident.
- **Backup communication methods:** It's a best practice to have out-of-band communication as a backup for incident communication. An example of an application that provides a secure out-of-band communications channel is AWS Wickr.
- **Phases of incident response and actions to take:** Enumerates the phases of incident response (for example, detect, analyze, eradicate, contain, and recover), including high-level actions to take within those phases.
- **Incident severity and prioritization definitions:** Details how to classify the severity of an incident, how to prioritize the incident, and then how the severity definitions affect escalation procedures.

While these sections are common throughout companies of different sizes and industries, each organization's incident response plan is unique. You will need to build an incident response plan that works best for your organization.

Resources

Related best practices:

- SEC04 (How do you detect and investigate security events?)

Related documents:

- AWS Security Incident Response Guide
- NIST: Computer Security Incident Handling Guide

SEC10-BP03 Prepare forensic capabilities

It's important for your incident responders to understand when and how the forensic investigation fits into your response plan. Your organization should define what evidence is collected and what tools are used in the process. Identify and prepare forensic investigation capabilities that are

suitable, including external specialists, tools, and automation. A key decision that you should make upfront is if you will collect data from a live system. Some data, such as the contents of volatile memory or active network connections, will be lost if the system is powered off or rebooted.

Your response team can combine tools, such as AWS Systems Manager, Amazon EventBridge, and AWS Lambda, to automatically run forensic tools within an operating system and VPC traffic mirroring to obtain a network packet capture, to gather non-persistent evidence. Conduct other activities, such as log analysis or analyzing disk images, in a dedicated security account with customized forensic workstations and tools accessible to your responders.

Routinely ship relevant logs to a data store that provides high durability and integrity. Responders should have access to those logs. AWS offers several tools that can make log investigation easier, such as Amazon Athena, Amazon OpenSearch Service (OpenSearch Service), and Amazon CloudWatch Logs Insights. Additionally, preserve evidence securely using Amazon Simple Storage Service (Amazon S3) Object Lock. This service follows the WORM (write-once- read-many) model and prevents objects from being deleted or overwritten for a defined period. As forensic investigation techniques require specialist training, you might need to engage external specialists.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Identify forensic capabilities: Research your organization's forensic investigation capabilities, available tools, and external specialists.
- Automating Incident Response and Forensics

Resources

Related documents:

- How to automate forensic disk collection in AWS

SEC10-BP04 Automate containment capability

Automate containment and recovery of an incident to reduce response times and organizational impact.

Once you create and practice the processes and tools from your playbooks, you can deconstruct the logic into a code-based solution, which can be used as a tool by many responders to automate the

response and remove variance or guess-work by your responders. This can speed up the lifecycle of a response. The next goal is to allow this code to be fully automated by being invoked by the alerts or events themselves, rather than by a human responder, to create an event-driven response. These processes should also automatically add relevant data to your security systems. For example, an incident involving traffic from an unwanted IP address can automatically populate an AWS WAF block list or Network Firewall rule group to prevent further activity.

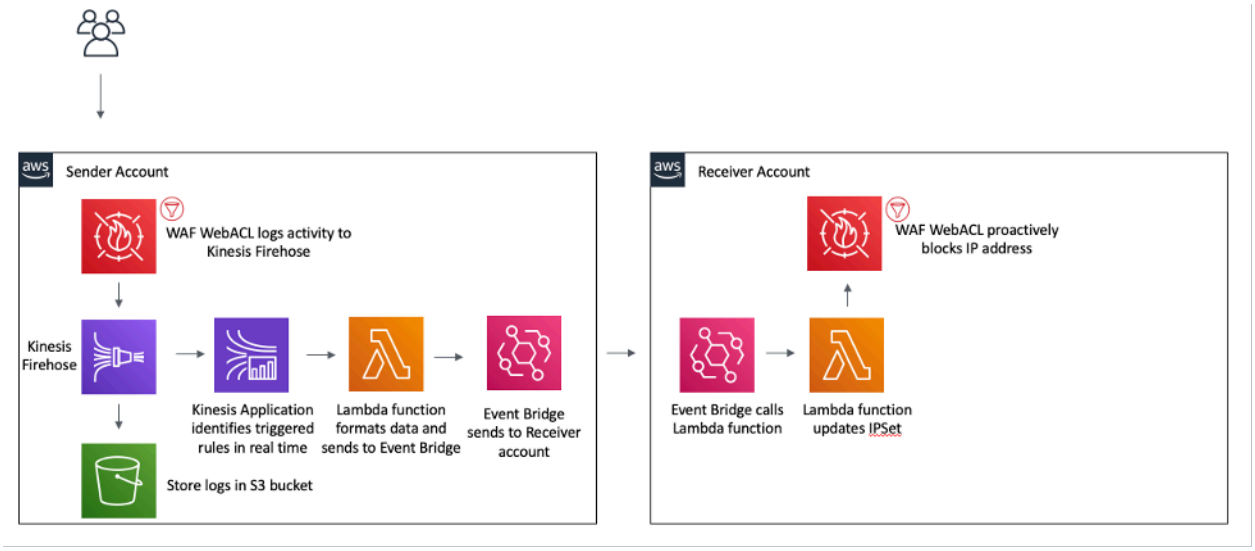


Figure 3: AWS WAF automate blocking of known malicious IP addresses.

With an event-driven response system, a detective mechanism initiates a responsive mechanism to automatically remediate the event. You can use event-driven response capabilities to reduce the time-to-value between detective mechanisms and responsive mechanisms. To create this event-driven architecture, you can use AWS Lambda, which is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. For example, assume that you have an AWS account using the AWS CloudTrail service. If CloudTrail is ever turned off (through the `cloudtrail:StopLogging` API call), you can use Amazon EventBridge to monitor for the specific `cloudtrail:StopLogging` event, and invoke a Lambda function to call `cloudtrail:StartLogging` to restart logging.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Automate containment capability.

Resources

Related documents:

- AWS Incident Response Guide

Related videos:

- Prepare for and respond to security incidents in your AWS environment

SEC10-BP05 Pre-provision access

Verify that incident responders have the correct access pre-provisioned in AWS to reduce the time needed for investigation through to recovery.

Common anti-patterns:

- Using the root account for incident response.
- Altering existing accounts.
- Manipulating IAM permissions directly when providing just-in-time privilege elevation.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS recommends reducing or eliminating reliance on long-lived credentials wherever possible, in favor of temporary credentials and *just-in-time* privilege escalation mechanisms. Long-lived credentials are prone to security risk and increase operational overhead. For most management tasks, as well as incident response tasks, we recommend you implement identity federation alongside temporary escalation for administrative access. In this model, a user requests elevation to a higher level of privilege (such as an incident response role) and, provided the user is eligible for elevation, a request is sent to an approver. If the request is approved, the user receives a set of temporary AWS credentials which can be used to complete their tasks. After these credentials expire, the user must submit a new elevation request.

We recommend the use of temporary privilege escalation in the majority of incident response scenarios. The correct way to do this is to use the AWS Security Token Service and session policies to scope access.

There are scenarios where federated identities are unavailable, such as:

- Outage related to a compromised identity provider (IdP).

- Misconfiguration or human error causing broken federated access management system.
- Malicious activity such as a distributed denial of service (DDoS) event or rendering unavailability of the system.

In the preceding cases, there should be emergency *break glass* access configured to allow investigation and timely remediation of incidents. We recommend that you use a user, group, or role with appropriate permissions to perform tasks and access AWS resources. Use the root user only for tasks that require root user credentials. To verify that incident responders have the correct level of access to AWS and other relevant systems, we recommend the pre-provisioning of dedicated accounts. The accounts require privileged access, and must be tightly controlled and monitored. The accounts must be built with the fewest privileges required to perform the necessary tasks, and the level of access should be based on the playbooks created as part of the incident management plan.

Use purpose-built and dedicated users and roles as a best practice. Temporarily escalating user or role access through the addition of IAM policies both makes it unclear what access users had during the incident, and risks the escalated privileges not being revoked.

It is important to remove as many dependencies as possible to verify that access can be gained under the widest possible number of failure scenarios. To support this, create a playbook to verify that incident response users are created as users in a dedicated security account, and not managed through any existing Federation or single sign-on (SSO) solution. Each individual responder must have their own named account. The account configuration must enforce strong password policy and multi-factor authentication (MFA). If the incident response playbooks only require access to the AWS Management Console, the user should not have access keys configured and should be explicitly disallowed from creating access keys. This can be configured with IAM policies or service control policies (SCPs) as mentioned in the AWS Security Best Practices for AWS Organizations SCPs. The users should have no privileges other than the ability to assume incident response roles in other accounts.

During an incident it might be necessary to grant access to other internal or external individuals to support investigation, remediation, or recovery activities. In this case, use the playbook mechanism mentioned previously, and there must be a process to verify that any additional access is revoked immediately after the incident is complete.

To verify that the use of incident response roles can be properly monitored and audited, it is essential that the IAM accounts created for this purpose are not shared between individuals, and that the AWS account root user is not used unless required for a specific task. If the root user is

required (for example, IAM access to a specific account is unavailable), use a separate process with a playbook available to verify availability of the root user sign-in credentials and MFA token.

To configure the IAM policies for the incident response roles, consider using IAM Access Analyzer to generate policies based on AWS CloudTrail logs. To do this, grant administrator access to the incident response role on a non-production account and run through your playbooks. Once complete, a policy can be created that allows only the actions taken. This policy can then be applied to all the incident response roles across all accounts. You might wish to create a separate IAM policy for each playbook to allow easier management and auditing. Example playbooks could include response plans for ransomware, data breaches, loss of production access, and other scenarios.

Use the incident response accounts to assume dedicated incident response IAM roles in other AWS accounts. These roles must be configured to only be assumable by users in the security account, and the trust relationship must require that the calling principal has authenticated using MFA. The roles must use tightly-scoped IAM policies to control access. Ensure that all `AssumeRole` requests for these roles are logged in CloudTrail and alerted on, and that any actions taken using these roles are logged.

It is strongly recommended that both the IAM accounts and the IAM roles are clearly named to allow them to be easily found in CloudTrail logs. An example of this would be to name the IAM accounts *<USER_ID>*-BREAK-GLASS and the IAM roles BREAK-GLASS-ROLE.

CloudTrail is used to log API activity in your AWS accounts and should be used to configure alerts on usage of the incident response roles. Refer to the blog post on configuring alerts when root keys are used. The instructions can be modified to configure the Amazon CloudWatch metric filter-to-filter on `AssumeRole` events related to the incident response IAM role:

```
{ $.eventName = "AssumeRole" && $.requestParameters.roleArn =
 "<INCIDENT_RESPONSE_ROLE_ARN>" && $.userIdentity.invokedBy NOT EXISTS && $.eventType !
= "AwsServiceEvent" }
```

As the incident response roles are likely to have a high level of access, it is important that these alerts go to a wide group and are acted upon promptly.

During an incident, it is possible that a responder might require access to systems which are not directly secured by IAM. These could include Amazon Elastic Compute Cloud instances, Amazon Relational Database Service databases, or software-as-a-service (SaaS) platforms. It is strongly recommended that rather than using native protocols such as SSH or RDP, AWS Systems Manager

Session Manager is used for all administrative access to Amazon EC2 instances. This access can be controlled using IAM, which is secure and audited. It might also be possible to automate parts of your playbooks using AWS Systems Manager Run Command documents, which can reduce user error and improve time to recovery. For access to databases and third-party tools, we recommend storing access credentials in AWS Secrets Manager and granting access to the incident responder roles.

Finally, the management of the incident response IAM accounts should be added to your Joiners, Movers, and Leavers processes and reviewed and tested periodically to verify that only the intended access is allowed.

Resources

Related documents:

- Managing temporary elevated access to your AWS environment
- AWS Security Incident Response Guide
- AWS Elastic Disaster Recovery
- AWS Systems Manager Incident Manager
- Setting an account password policy for IAM users
- Using multi-factor authentication (MFA) in AWS
- Configuring Cross-Account Access with MFA
- Using IAM Access Analyzer to generate IAM policies
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment
- How to Receive Notifications When Your AWS Account's Root Access Keys Are Used
- Create fine-grained session permissions using IAM managed policies

Related videos:

- Automating Incident Response and Forensics in AWS
- DIY guide to runbooks, incident reports, and incident response
- Prepare for and respond to security incidents in your AWS environment

Related examples:

- Lab: AWS Account Setup and Root User

- Lab: Incident Response with AWS Console and CLI

SEC10-BP06 Pre-deploy tools

Ensure that security personnel have the right tools pre-deployed into AWS to reduce the time for investigation through to recovery.

To automate security engineering and operations functions, you can use a comprehensive set of APIs and tools from AWS. You can fully automate identity management, network security, data protection, and monitoring capabilities and deliver them using popular software development methods that you already have in place. When you build security automation, your system can monitor, review, and initiate a response, rather than having people monitor your security position and manually react to events. An effective way to automatically provide searchable and relevant log data across AWS services to your incident responders is to turn on Amazon Detective.

If your incident response teams continue to respond to alerts in the same way, they risk alert fatigue. Over time, the team can become desensitized to alerts and can either make mistakes handling ordinary situations or miss unusual alerts. Automation helps avoid alert fatigue by using functions that process the repetitive and ordinary alerts, leaving humans to handle the sensitive and unique incidents. Integrating anomaly detection systems, such as Amazon GuardDuty, AWS CloudTrail Insights, and Amazon CloudWatch Anomaly Detection, can reduce the burden of common threshold-based alerts.

You can improve manual processes by programmatically automating steps in the process. After you define the remediation pattern to an event, you can decompose that pattern into actionable logic, and write the code to perform that logic. Responders can then run that code to remediate the issue. Over time, you can automate more and more steps, and ultimately automatically handle whole classes of common incidents.

For tools that run within the operating system of your Amazon Elastic Compute Cloud (Amazon EC2) instance, you should evaluate using the AWS Systems Manager Run Command, which allows you to remotely and securely administrate instances using an agent that you install on your Amazon EC2 instance operating system. It requires the Systems Manager Agent (SSM Agent), which is installed by default on many Amazon Machine Images (AMIs). Be aware, though, that once an instance has been compromised, no responses from tools or agents running on it should be considered trustworthy.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Pre-deploy tools: Ensure that security personnel have the right tools pre-deployed in AWS so that an appropriate response can be made to an incident.
  - Lab: Incident response with AWS Management Console and CLI
  - Incident Response Playbook with Jupyter - AWS IAM
  - AWS Security Automation
- Implement resource tagging: Tag resources with information, such as a code for the resource under investigation, so that you can identify resources during an incident.
  - AWS Tagging Strategies

Resources

Related documents:

- AWS Incident Response Guide

Related videos:

- DIY guide to runbooks, incident reports, and incident response

SEC10-BP07 Run game days

> This best practice was updated with new guidance on July 13th, 2023.

As organizations grow and evolve over time, so does the threat landscape, making it important to continually review your incident response capabilities. Running game days, or simulations, is one method that can be used to perform this assessment. Simulations use real-world security event scenarios designed to mimic a threat actor's tactics, techniques, and procedures (TTPs) and allow an organization to exercise and evaluate their incident response capabilities by responding to these mock cyber events as they might occur in reality.

Benefits of establishing this best practice: Simulations have a variety of benefits:

- Validating cyber readiness and developing the confidence of your incident responders.
- Testing the accuracy and efficiency of tools and workflows.

- Refining communication and escalation methods aligned with your incident response plan.
- Providing an opportunity to respond to less common vectors.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

There are three main types of simulations:

- **Tabletop exercises:** The tabletop approach to simulations is a discussion-based session involving the various incident response stakeholders to practice roles and responsibilities and use established communication tools and playbooks. Exercise facilitation can typically be accomplished in a full day in a virtual venue, physical venue, or a combination. Because it is discussion-based, the tabletop exercise focuses on processes, people, and collaboration. Technology is an integral part of the discussion, but the actual use of incident response tools or scripts is generally not a part of the tabletop exercise.
- **Purple team exercises:** Purple team exercises increase the level of collaboration between the incident responders (blue team) and simulated threat actors (red team). The blue team is comprised of members of the security operations center (SOC), but can also include other stakeholders that would be involved during an actual cyber event. The red team is comprised of a penetration testing team or key stakeholders that are trained in offensive security. The red team works collaboratively with the exercise facilitators when designing a scenario so that the scenario is accurate and feasible. During purple team exercises, the primary focus is on the detection mechanisms, the tools, and the standard operating procedures (SOPs) supporting the incident response efforts.
- **Red team exercises:** During a red team exercise, the offense (red team) conducts a simulation to achieve a certain objective or set of objectives from a predetermined scope. The defenders (blue team) will not necessarily have knowledge of the scope and duration of the exercise, which provides a more realistic assessment of how they would respond to an actual incident. Because red team exercises can be invasive tests, be cautious and implement controls to verify that the exercise does not cause actual harm to your environment.

Consider facilitating cyber simulations at a regular interval. Each exercise type can provide unique benefits to the participants and the organization as a whole, so you might choose to start with less complex simulation types (such as tabletop exercises) and progress to more complex simulation types (red team exercises). You should select a simulation type based on your security maturity,

resources, and your desired outcomes. Some customers might not choose to perform red team exercises due to complexity and cost.

Implementation steps

Regardless of the type of simulation you choose, simulations generally follow these implementation steps:

1. **Define core exercise elements:** Define the simulation scenario and the objectives of the simulation. Both of these should have leadership acceptance.
2. **Identify key stakeholders:** At a minimum, an exercise needs exercise facilitators and participants. Depending on the scenario, additional stakeholders such as legal, communications, or executive leadership might be involved.
3. **Build and test the scenario:** The scenario might need to be redefined as it is being built if specific elements aren't feasible. A finalized scenario is expected as the output of this stage.
4. **Facilitate the simulation:** The type of simulation determines the facilitation used (a paper-based scenario compared to a highly technical, simulated scenario). The facilitators should align their facilitation tactics to the exercise objects and they should engage all exercise participants wherever possible to provide the most benefit.
5. **Develop the after-action report (AAR):** Identify areas that went well, those that can use improvement, and potential gaps. The AAR should measure the effectiveness of the simulation as well as the team's response to the simulated event so that progress can be tracked over time with future simulations.

Resources

Related documents:

- AWS Incident Response Guide

Related videos:

- AWS GameDay - Security Edition

Application security

Question

- SEC 11. How do you incorporate and validate the security properties of applications throughout the design, development, and deployment lifecycle?

SEC 11. How do you incorporate and validate the security properties of applications throughout the design, development, and deployment lifecycle?

Training people, testing using automation, understanding dependencies, and validating the security properties of tools and applications help to reduce the likelihood of security issues in production workloads.

Best practices

- SEC11-BP01 Train for application security
- SEC11-BP02 Automate testing throughout the development and release lifecycle
- SEC11-BP03 Perform regular penetration testing
- SEC11-BP04 Manual code reviews
- SEC11-BP05 Centralize services for packages and dependencies
- SEC11-BP06 Deploy software programmatically
- SEC11-BP07 Regularly assess security properties of the pipelines
- SEC11-BP08 Build a program that embeds security ownership in workload teams

SEC11-BP01 Train for application security

Provide training to the builders in your organization on common practices for the secure development and operation of applications. Adopting security focused development practices helps reduce the likelihood of issues that are only detected at the security review stage.

Desired outcome: Software should be designed and built with security in mind. When the builders in an organization are trained on secure development practices that start with a threat model, it improves the overall quality and security of the software produced. This approach can reduce the time to ship software or features because less rework is needed after the security review stage.

For the purposes of this best practice, *secure development* refers to the software that is being written and the tools or systems that support the software development lifecycle (SDLC).

Common anti-patterns:

- Waiting until a security review, and then considering the security properties of a system.

- Leaving all security decisions to the security team.
- Failing to communicate how the decisions taken in the SDLC relate to the overall security expectations or policies of the organization.
- Engaging in the security review process too late.

Benefits of establishing this best practice:

- Better knowledge of the organizational requirements for security early in the development cycle.
- Being able to identify and remediate potential security issues faster, resulting in a quicker delivery of features.
- Improved quality of software and systems.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Provide training to the builders in your organization. Starting off with a course on threat modeling is a good foundation for helping train for security. Ideally, builders should be able to self-serve access to information relevant to their workloads. This access helps them make informed decisions about the security properties of the systems they build without needing to ask another team. The process for engaging the security team for reviews should be clearly defined and simple to follow. The steps in the review process should be included in the security training. Where known implementation patterns or templates are available, they should be simple to find and link to the overall security requirements. Consider using AWS CloudFormation, AWS Cloud Development Kit (AWS CDK) Constructs, Service Catalog, or other templating tools to reduce the need for custom configuration.

Implementation steps

- Start builders with a course on threat modeling to build a good foundation, and help train them on how to think about security.
- Provide access to AWS Training and Certification, industry, or AWS Partner training.
- Provide training on your organization's security review process, which clarifies the division of responsibilities between the security team, workload teams, and other stakeholders.
- Publish self-service guidance on how to meet your security requirements, including code examples and templates, if available.

- Regularly obtain feedback from builder teams on their experience with the security review process and training, and use that feedback to improve.
- Use game days or bug bash campaigns to help reduce the number of issues, and increase the skills of your builders.

Resources

Related best practices:

- SEC11-BP08 Build a program that embeds security ownership in workload teams

Related documents:

- AWS Training and Certification
- How to think about cloud security governance
- How to approach threat modeling
- Accelerating training – The AWS Skills Guild

Related videos:

- Proactive security: Considerations and approaches

Related examples:

- Workshop on threat modeling
- Industry awareness for developers

Related services:

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK) (AWS CDK) Constructs
- Service Catalog
- AWS BugBust

SEC11-BP02 Automate testing throughout the development and release lifecycle

Automate the testing for security properties throughout the development and release lifecycle. Automation makes it easier to consistently and repeatably identify potential issues in software prior to release, which reduces the risk of security issues in the software being provided.

Desired outcome: The goal of automated testing is to provide a programmatic way of detecting potential issues early and often throughout the development lifecycle. When you automate regression testing, you can rerun functional and non-functional tests to verify that previously tested software still performs as expected after a change. When you define security unit tests to check for common misconfigurations, such as broken or missing authentication, you can identify and fix these issues early in the development process.

Test automation uses purpose-built test cases for application validation, based on the application's requirements and desired functionality. The result of the automated testing is based on comparing the generated test output to its respective expected output, which expedites the overall testing lifecycle. Testing methodologies such as regression testing and unit test suites are best suited for automation. Automating the testing of security properties allows builders to receive automated feedback without having to wait for a security review. Automated tests in the form of static or dynamic code analysis can increase code quality and help detect potential software issues early in the development lifecycle.

Common anti-patterns:

- Not communicating the test cases and test results of the automated testing.
- Performing the automated testing only immediately prior to a release.
- Automating test cases with frequently changing requirements.
- Failing to provide guidance on how to address the results of security tests.

Benefits of establishing this best practice:

- Reduced dependency on people evaluating the security properties of systems.
- Having consistent findings across multiple workstreams improves consistency.
- Reduced likelihood of introducing security issues into production software.
- Shorter window of time between detection and remediation due to catching software issues earlier.

- Increased visibility of systemic or repeated behavior across multiple workstreams, which can be used to drive organization-wide improvements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

As you build your software, adopt various mechanisms for software testing to ensure that you are testing your application for both functional requirements, based on your application's business logic, and non-functional requirements, which are focused on application reliability, performance, and security.

Static application security testing (SAST) analyzes your source code for anomalous security patterns, and provides indications for defect prone code. SAST relies on static inputs, such as documentation (requirements specification, design documentation, and design specifications) and application source code to test for a range of known security issues. Static code analyzers can help expedite the analysis of large volumes of code. The NIST Quality Group provides a comparison of Source Code Security Analyzers, which includes open source tools for Byte Code Scanners and Binary Code Scanners.

Complement your static testing with dynamic analysis security testing (DAST) methodologies, which performs tests against the running application to identify potentially unexpected behavior. Dynamic testing can be used to detect potential issues that are not detectable via static analysis. Testing at the code repository, build, and pipeline stages allows you to check for different types of potential issues from entering into your code. Amazon CodeWhisperer provides code recommendations, including security scanning, in the builder's IDE. Amazon CodeGuru Reviewer can identify critical issues, security issues, and hard-to-find bugs during application development, and provides recommendations to improve code quality.

The Security for Developers workshop uses AWS developer tools, such as AWS CodeBuild, AWS CodeCommit, and AWS CodePipeline, for release pipeline automation that includes SAST and DAST testing methodologies.

As you progress through your SDLC, establish an iterative process that includes periodic application reviews with your security team. Feedback gathered from these security reviews should be addressed and validated as part of your release readiness review. These reviews establish a robust application security posture, and provide builders with actionable feedback to address potential issues.

Implementation steps

- Implement consistent IDE, code review, and CI/CD tools that include security testing.
- Consider where in the SDLC it is appropriate to block pipelines instead of just notifying builders that issues need to be remediated.
- The Security for Developers workshop provides an example of integrating static and dynamic testing into a release pipeline.
- Performing testing or code analysis using automated tools, such as Amazon CodeWhisperer integrated with developer IDEs, and Amazon CodeGuru Reviewer for scanning code on commit, helps builders get feedback at the right time.
- When building using AWS Lambda, you can use Amazon Inspector to scan the application code in your functions.
- When automated testing is included in CI/CD pipelines, you should use a ticketing system to track the notification and remediation of software issues.
- For security tests that might generate findings, linking to guidance for remediation helps builders improve code quality.
- Regularly analyze the findings from automated tools to prioritize the next automation, builder training, or awareness campaign.

Resources

Related documents:

- Continuous Delivery and Continuous Deployment

- AWS DevOps Competency Partners
- AWS Security Competency Partners for Application Security
- Choosing a Well-Architected CI/CD approach
- Monitoring CodeCommit events in Amazon EventBridge and Amazon CloudWatch Events
- Secrets detection in Amazon CodeGuru Review
- Accelerate deployments on AWS with effective governance
- How AWS approaches automating safe, hands-off deployments

Related videos:

- Hands-off: Automating continuous delivery pipelines at Amazon
- Automating cross-account CI/CD pipelines

Related examples:

- Industry awareness for developers
- AWS CodePipeline Governance (GitHub)
- Security for Developers workshop

SEC11-BP03 Perform regular penetration testing

Perform regular penetration testing of your software. This mechanism helps identify potential software issues that cannot be detected by automated testing or a manual code review. It can also help you understand the efficacy of your detective controls. Penetration testing should try to determine if the software can be made to perform in unexpected ways, such as exposing data that should be protected, or granting broader permissions than expected.

Desired outcome: Penetration testing is used to detect, remediate, and validate your application's security properties. Regular and scheduled penetration testing should be performed as part of the software development lifecycle (SDLC). The findings from penetration tests should be addressed prior to the software being released. You should analyze the findings from penetration tests to identify if there are issues that could be found using automation. Having a regular and repeatable penetration testing process that includes an active feedback mechanism helps inform the guidance to builders and improves software quality.

Common anti-patterns:

- Only penetration testing for known or prevalent security issues.
- Penetration testing applications without dependent third-party tools and libraries.
- Only penetration testing for package security issues, and not evaluating implemented business logic.

Benefits of establishing this best practice:

- Increased confidence in the security properties of the software prior to release.
- Opportunity to identify preferred application patterns, which leads to greater software quality.

- A feedback loop that identifies earlier in the development cycle where automation or additional training can improve the security properties of software.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Penetration testing is a structured security testing exercise where you run planned security breach scenarios to detect, remediate, and validate security controls. Penetration tests start with reconnaissance, during which data is gathered based on the current design of the application and its dependencies. A curated list of security-specific testing scenarios are built and run. The key purpose of these tests is to uncover security issues in your application, which could be exploited for gaining unintended access to your environment, or unauthorized access to data. You should perform penetration testing when you launch new features, or whenever your application has undergone major changes in function or technical implementation.

You should identify the most appropriate stage in the development lifecycle to perform penetration testing. This testing should happen late enough that the functionality of the system is close to the intended release state, but with enough time remaining for any issues to be remediated.

Implementation steps

- Have a structured process for how penetration testing is scoped, basing this process on the threat model is a good way of maintaining context.
- Identify the appropriate place in the development cycle to perform penetration testing. This should be when there is minimal change expected in the application, but with enough time to perform remediation.
- Train your builders on what to expect from penetration testing findings, and how to get information on remediation.
- Use tools to speed up the penetration testing process by automating common or repeatable tests.
- Analyze penetration testing findings to identify systemic security issues, and use this data to inform additional automated testing and ongoing builder education.

Resources

Related best practices:

- SEC11-BP01 Train for application security
- SEC11-BP02 Automate testing throughout the development and release lifecycle

Related documents:

- AWS Penetration Testing provides detailed guidance for penetration testing on AWS
- Accelerate deployments on AWS with effective governance
- AWS Security Competency Partners
- Modernize your penetration testing architecture on AWS Fargate
- AWS Fault injection Simulator

Related examples:

- Automate API testing with AWS CodePipeline (GitHub)
- Automated security helper (GitHub)

SEC11-BP04 Manual code reviews

Perform a manual code review of the software that you produce. This process helps verify that the person who wrote the code is not the only one checking the code quality.

Desired outcome: Including a manual code review step during development increases the quality of the software being written, helps upskill less experienced members of the team, and provides an opportunity to identify places where automation can be used. Manual code reviews can be supported by automated tools and testing.

Common anti-patterns:

- Not performing reviews of code before deployment.
- Having the same person write and review the code.
- Not using automation to assist or orchestrate code reviews.
- Not training builders on application security before they review code.

Benefits of establishing this best practice:

- Increased code quality.

- Increased consistency of code development through reuse of common approaches.
- Reduction in the number of issues discovered during penetration testing and later stages.
- Improved knowledge transfer within the team.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The review step should be implemented as part of the overall code management flow. The specifics depend on the approach used for branching, pull-requests, and merging. You might be using AWS CodeCommit or third-party solutions such as GitHub, GitLab, or Bitbucket. Whatever method you use, it's important to verify that your processes require the review of code before it's deployed in a production environment. Using tools such as Amazon CodeGuru Reviewer can make it easier to orchestrate the code review process.

Implementation steps

- Implement a manual review step as part of your code management flow and perform this review before proceeding.
- Consider Amazon CodeGuru Reviewer for managing and assisting in code reviews.
- Implement an approval flow that requires a code review being completed before code can progress to the next stage.
- Verify there is a process to identify issues being found during manual code reviews that could be detected automatically.
- Integrate the manual code review step in a way that aligns with your code development practices.

Resources

Related best practices:

- SEC11-BP02 Automate testing throughout the development and release lifecycle

Related documents:

- Working with pull requests in AWS CodeCommit repositories

- Working with approval rule templates in AWS CodeCommit
- About pull requests in GitHub
- Automate code reviews with Amazon CodeGuru Reviewer
- Automating detection of security vulnerabilities and bugs in CI/CD pipelines using Amazon CodeGuru Reviewer CLI

Related videos:

- Continuous improvement of code quality with Amazon CodeGuru

Related examples:

- Security for Developers workshop

SEC11-BP05 Centralize services for packages and dependencies

Provide centralized services for builder teams to obtain software packages and other dependencies. This allows the validation of packages before they are included in the software that you write, and provides a source of data for the analysis of the software being used in your organization.

Desired outcome: Software is comprised of a set of other software packages in addition to the code that is being written. This makes it simple to consume implementations of functionality that are repeatedly used, such as a JSON parser or an encryption library. Logically centralizing the sources for these packages and dependencies provides a mechanism for security teams to validate the properties of the packages before they are used. This approach also reduces the risk of an unexpected issue being caused by a change in an existing package, or by builder teams including arbitrary packages directly from the internet. Use this approach in conjunction with the manual and automated testing flows to increase the confidence in the quality of the software that is being developed.

Common anti-patterns:

- Pulling packages from arbitrary repositories on the internet.
- Not testing new packages before making them available to builders.

Benefits of establishing this best practice:

- Better understanding of what packages are being used in the software being built.
- Being able to notify workload teams when a package needs to be updated based on the understanding of who is using what.
- Reducing the risk of a package with issues being included in your software.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Provide centralized services for packages and dependencies in a way that is simple for builders to consume. Centralized services can be logically central rather than implemented as a monolithic system. This approach allows you to provide services in a way that meets the needs of your builders. You should implement an efficient way of adding packages to the repository when updates happen or new requirements emerge. AWS services such as AWS CodeArtifact or similar AWS partner solutions provide a way of delivering this capability.

Implementation steps:

- Implement a logically centralized repository service that is available in all of the environments where software is developed.
- Include access to the repository as part of the AWS account vending process.
- Build automation to test packages before they are published in a repository.
- Maintain metrics of the most commonly used packages, languages, and teams with the highest amount of change.
- Provide an automated mechanism for builder teams to request new packages and provide feedback.
- Regularly scan packages in your repository to identify the potential impact of newly discovered issues.

Resources

Related best practices:

- SEC11-BP02 Automate testing throughout the development and release lifecycle

Related documents:

- Accelerate deployments on AWS with effective governance
- Tighten your package security with CodeArtifact Package Origin Control toolkit
- Detecting security issues in logging with Amazon CodeGuru Reviewer
- Supply chain Levels for Software Artifacts (SLSA)

Related videos:

- Proactive security: Considerations and approaches
- The AWS Philosophy of Security (re:Invent 2017)
- When security, safety, and urgency all matter: Handling Log4Shell

Related examples:

- Multi Region Package Publishing Pipeline (GitHub)
- Publishing Node.js Modules on AWS CodeArtifact using AWS CodePipeline (GitHub)
- AWS CDK Java CodeArtifact Pipeline Sample (GitHub)
- Distribute private .NET NuGet packages with AWS CodeArtifact (GitHub)

SEC11-BP06 Deploy software programmatically

Perform software deployments programmatically where possible. This approach reduces the likelihood that a deployment fails or an unexpected issue is introduced due to human error.

Desired outcome: Keeping people away from data is a key principle of building securely in the AWS Cloud. This principle includes how you deploy your software.

The benefits of not relying on people to deploy software is the greater confidence that what you tested is what gets deployed, and that the deployment is performed consistently every time. The software should not need to be changed to function in different environments. Using the principles of twelve-factor application development, specifically the externalizing of configuration, allows you to deploy the same code to multiple environments without requiring changes. Cryptographically signing software packages is a good way to verify that nothing has changed between environments. The overall outcome of this approach is to reduce risk in your change process and improve the consistency of software releases.

Common anti-patterns:

- Manually deploying software into production.
- Manually performing changes to software to cater to different environments.

Benefits of establishing this best practice:

- Increased confidence in the software release process.
- Reduced risk of a failed change impacting business functionality.
- Increased release cadence due to lower change risk.
- Automatic rollback capability for unexpected events during deployment.
- Ability to cryptographically prove that the software that was tested is the software deployed.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Build your AWS account structure to remove persistent human access from environments and use CI/CD tools to perform deployments. Architect your applications so that environment-specific configuration data is obtained from an external source, such as AWS Systems Manager Parameter Store. Sign packages after they have been tested, and validate these signatures during deployment. Configure your CI/CD pipelines to push application code and use canaries to confirm successful deployment. Use tools such as AWS CloudFormation or AWS CDK to define your infrastructure, then use AWS CodeBuild and AWS CodePipeline to perform CI/CD operations.

Implementation steps

- Build well-defined CI/CD pipelines to streamline the deployment process.
- Using AWS CodeBuild and AWS Code Pipeline to provide CI/CD capability makes it simple to integrate security testing into your pipelines.
- Follow the guidance on separation of environments in the Organizing Your AWS Environment Using Multiple Accounts whitepaper.
- Verify no persistent human access to environments where production workloads are running.
- Architect your applications to support the externalization of configuration data.
- Consider deploying using a blue/green deployment model.
- Implement canaries to validate the successful deployment of software.

- Use cryptographic tools such as AWS Signer or AWS Key Management Service (AWS KMS) to sign and verify the software packages that you are deploying.

Resources

Related best practices:

- SEC11-BP02 Automate testing throughout the development and release lifecycle

Related documents:

- AWS CI/CD Workshop
- Accelerate deployments on AWS with effective governance
- Automating safe, hands-off deployments
- Code signing using AWS Certificate Manager Private CA and AWS Key Management Service asymmetric keys
- Code Signing, a Trust and Integrity Control for AWS Lambda

Related videos:

- Hands-off: Automating continuous delivery pipelines at Amazon

Related examples:

- Blue/Green deployments with AWS Fargate

SEC11-BP07 Regularly assess security properties of the pipelines

Apply the principles of the Well-Architected Security Pillar to your pipelines, with particular attention to the separation of permissions. Regularly assess the security properties of your pipeline infrastructure. Effectively managing the security *of* the pipelines allows you to deliver the security of the software that passes *through* the pipelines.

Desired outcome: The pipelines used to build and deploy your software should follow the same recommended practices as any other workload in your environment. The tests that are implemented in the pipelines should not be editable by the builders who are using them. The pipelines should only have the permissions needed for the deployments they are doing and should

implement safeguards to avoid deploying to the wrong environments. Pipelines should not rely on long-term credentials, and should be configured to emit state so that the integrity of the build environments can be validated.

Common anti-patterns:

- Security tests that can be bypassed by builders.
- Overly broad permissions for deployment pipelines.
- Pipelines not being configured to validate inputs.
- Not regularly reviewing the permissions associated with your CI/CD infrastructure.
- Use of long-term or hardcoded credentials.

Benefits of establishing this best practice:

- Greater confidence in the integrity of the software that is built and deployed through the pipelines.
- Ability to stop a deployment when there is suspicious activity.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Starting with managed CI/CD services that support IAM roles reduces the risk of credential leakage. Applying the Security Pillar principles to your CI/CD pipeline infrastructure can help you determine where security improvements can be made. Following the AWS Deployment Pipelines Reference Architecture is a good starting point for building your CI/CD environments. Regularly reviewing the pipeline implementation and analyzing logs for unexpected behavior can help you understand the usage patterns of the pipelines being used to deploy software.

Implementation steps

- Start with the AWS Deployment Pipelines Reference Architecture.
- Consider using AWS IAM Access Analyzer to programmatically generate least privilege IAM policies for the pipelines.
- Integrate your pipelines with monitoring and alerting so that you are notified of unexpected or abnormal activity, for AWS managed services Amazon EventBridge allows you to route data to targets such as AWS Lambda or Amazon Simple Notification Service (Amazon SNS).

Resources

Related documents:

- AWS Deployment Pipelines Reference Architecture
- Monitoring AWS CodePipeline
- Security best practices for AWS CodePipeline

Related examples:

- DevOps monitoring dashboard (GitHub)

SEC11-BP08 Build a program that embeds security ownership in workload teams

Build a program or mechanism that empowers builder teams to make security decisions about the software that they create. Your security team still needs to validate these decisions during a review, but embedding security ownership in builder teams allows for faster, more secure workloads to be built. This mechanism also promotes a culture of ownership that positively impacts the operation of the systems you build.

Desired outcome: To embed security ownership and decision making in builder teams, you can either train builders on how to think about security or you can augment their training with security people embedded or associated with the builder teams. Either approach is valid and allows the team to make higher quality security decisions earlier in the development cycle. This ownership model is predicated on training for application security. Starting with the threat model for the particular workload helps focus the design thinking on the appropriate context. Another benefit of having a community of security focused builders, or a group of security engineers working with builder teams, is that you can more deeply understand how software is written. This understanding helps you determine the next areas for improvement in your automation capability.

Common anti-patterns:

- Leaving all security design decisions to a security team.
- Not addressing security requirements early enough in the development process.
- Not obtaining feedback from builders and security people on the operation of the program.

Benefits of establishing this best practice:

- Reduced time to complete security reviews.
- Reduction in security issues that are only detected at the security review stage.
- Improvement in the overall quality of the software being written.
- Opportunity to identify and understand systemic issues or areas of high value improvement.
- Reduction in the amount of rework required due to security review findings.
- Improvement in the perception of the security function.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Start with the guidance in SEC11-BP01 Train for application security. Then identify the operational model for the program that you think might work best for your organization. The two main patterns are to train builders or to embed security people in builder teams. After you have decided on the initial approach, you should pilot with a single or small group of workload teams to prove the model works for your organization. Leadership support from the builder and security parts of the organization helps with the delivery and success of the program. As you build this program, it's important to choose metrics that can be used to show the value of the program. Learning from how AWS has approached this problem is a good learning experience. This best practice is very much focused on organizational change and culture. The tools that you use should support the collaboration between the builder and security communities.

Implementation steps

- Start by training your builders for application security.
- Create a community and an onboarding program to educate builders.
- Pick a name for the program. Guardians, Champions, or Advocates are commonly used.
- Identify the model to use: train builders, embed security engineers, or have affinity security roles.
- Identify project sponsors from security, builders, and potentially other relevant groups.
- Track metrics for the number of people involved in the program, the time taken for reviews, and the feedback from builders and security people. Use these metrics to make improvements.

Resources

Related best practices:

- SEC11-BP01 Train for application security
- SEC11-BP02 Automate testing throughout the development and release lifecycle

Related documents:

- How to approach threat modeling
- How to think about cloud security governance

Related videos:

- Proactive security: Considerations and approaches

Reliability

The Reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. You can find prescriptive guidance on implementation in the Reliability Pillar whitepaper.

Best practice areas

- Foundations
- Workload architecture
- Change management
- Failure management

Foundations

Questions

- REL 1. How do you manage Service Quotas and constraints?
- REL 2. How do you plan your network topology?

REL 1. How do you manage Service Quotas and constraints?

For cloud-based workload architectures, there are Service Quotas (which are also referred to as service limits). These quotas exist to prevent accidentally provisioning more resources than you

need and to limit request rates on API operations so as to protect services from abuse. There are also resource constraints, for example, the rate that you can push bits down a fiber-optic cable, or the amount of storage on a physical disk.

Best practices

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP03 Accommodate fixed service quotas and constraints through architecture
- REL01-BP04 Monitor and manage quotas
- REL01-BP05 Automate quota management
- REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover

REL01-BP01 Aware of service quotas and constraints

This best practice was updated with new guidance on July 13th, 2023.

Be aware of your default quotas and manage your quota increase requests for your workload architecture. Know which cloud resource constraints, such as disk or network, are potentially impactful.

Desired outcome: Customers can prevent service degradation or disruption in their AWS accounts by implementing proper guidelines for monitoring key metrics, infrastructure reviews, and automation remediation steps to verify that services quotas and constraints are not reached that could cause service degradation or disruption.

Common anti-patterns:

- Deploying a workload without understanding the hard or soft quotas and their limits for the services used.
- Deploying a replacement workload without analyzing and reconfiguring the necessary quotas or contacting Support in advance.
- Assuming that cloud services have no limits and the services can be used without consideration to rates, limits, counts, quantities.
- Assuming that quotas will automatically be increased.

- Not knowing the process and timeline of quota requests.
- Assuming that the default cloud service quota is the identical for every service compared across regions.
- Assuming that service constraints can be breached and the systems will auto-scale or add increase the limit beyond the resource's constraints
- Not testing the application at peak traffic in order to stress the utilization of its resources.
- Provisioning the resource without analysis of the required resource size.
- Overprovisioning capacity by choosing resource types that go well beyond actual need or expected peaks.
- Not assessing capacity requirements for new levels of traffic in advance of a new customer event or deploying a new technology.

Benefits of establishing this best practice: Monitoring and automated management of service quotas and resource constraints can proactively reduce failures. Changes in traffic patterns for a customer's service can cause a disruption or degradation if best practices are not followed. By monitoring and managing these values across all regions and all accounts, applications can have improved resiliency under adverse or unplanned events.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Service Quotas is an AWS service that helps you manage your quotas for over 250 AWS services from one location. Along with looking up the quota values, you can also request and track quota increases from the Service Quotas console or using the AWS SDK. AWS Trusted Advisor offers a service quotas check that displays your usage and quotas for some aspects of some services. The default service quotas per service are also in the AWS documentation per respective service (for example, see Amazon VPC Quotas).

Some service limits, like rate limits on throttled APIs are set within the Amazon API Gateway itself by configuring a usage plan. Some limits that are set as configuration on their respective services include Provisioned IOPS, Amazon RDS storage allocated, and Amazon EBS volume allocations. Amazon Elastic Compute Cloud has its own service limits dashboard that can help you manage your instance, Amazon Elastic Block Store, and Elastic IP address limits. If you have a use case where service quotas impact your application's performance and they are not adjustable to your needs, then contact Support to see if there are mitigations.

Service quotas can be Region specific or can also be global in nature. Using an AWS service that reaches its quota will not act as expected in normal usage and may cause service disruption or degradation. For example, a service quota limits the number of DL Amazon EC2 that be used in an Region and that limit may be reached during a traffic scaling event using Auto Scaling groups (ASG).

Service quotas for each account should be assessed for usage on a regular basis to determine what the appropriate service limits might be for that account. These service quotas exist as operational guardrails, to prevent accidentally provisioning more resources than you need. They also serve to limit request rates on API operations to protect services from abuse.

Service constraints are different from service quotas. Service constraints represent a particular resource's limits as defined by that resource type. These might be storage capacity (for example, gp2 has a size limit of 1 GB - 16 TB) or disk throughput (10,0000 iops). It is essential that a resource type's constraint be engineered and constantly assessed for usage that might reach its limit. If a constraint is reached unexpectedly, the account's applications or services may be degraded or disrupted.

If there is a use case where service quotas impact an application's performance and they cannot be adjusted to required needs, contact Support to see if there are mitigations. For more detail on adjusting fixed quotas, see REL01-BP03 Accommodate fixed service quotas and constraints through architecture.

There are a number of AWS services and tools to help monitor and manage Service Quotas. The service and tools should be leveraged to provide automated or manual checks of quota levels.

- AWS Trusted Advisor offers a service quota check that displays your usage and quotas for some aspects of some services. It can aid in identifying services that are near quota.
- AWS Management Console provides methods to display services quota values, manage, request new quotas, monitor status of quota requests, and display history of quotas.
- AWS CLI and CDKs offer programmatic methods to automatically manage and monitor service quota levels and usage.

Implementation steps

For Service Quotas:

- Review AWS Service Quotas.

- To be aware of your existing service quotas, determine the services (like IAM Access Analyzer) that are used. There are approximately 250 AWS services controlled by service quotas. Then, determine the specific service quota name that might be used within each account and region. There are approximate 3000 service quota names per region.
- Augment this quota analysis with AWS Config to find all AWS resources used in your AWS accounts.
- Use AWS CloudFormation data to determine your AWS resources used. Look at the resources that were created either in the AWS Management Console or with the `list-stack-resources` AWS CLI command. You can also see resources configured to be deployed in the template itself.
- Determine all the services your workload requires by looking at the deployment code.
- Determine the service quotas that apply. Use the programmatically accessible information from Trusted Advisor and Service Quotas.
- Establish an automated monitoring method (see REL01-BP02 Manage service quotas across accounts and regions and REL01-BP04 Monitor and manage quotas) to alert and inform if services quotas are near or have reached their limit.
- Establish an automated and programmatic method to check if a service quota has been changed in one region but not in other regions in the same account (see REL01-BP02 Manage service quotas across accounts and regions and REL01-BP04 Monitor and manage quotas).
- Automate scanning application logs and metrics to determine if there are any quota or service constraint errors. If these errors are present, send alerts to the monitoring system.
- Establish engineering procedures to calculate the required change in quota (see REL01-BP05 Automate quota management) once it has been identified that larger quotas are required for specific services.
- Create a provisioning and approval workflow to request changes in service quota. This should include an exception workflow in case of request deny or partial approval.
- Create an engineering method to review service quotas prior to provisioning and using new AWS services before rolling out to production or loaded environments. (for example, load testing account).

For service constraints:

- Establish monitoring and metrics methods to alert for resources reading close to their resource constraints. Leverage CloudWatch as appropriate for metrics or log monitoring.

- Establish alert thresholds for each resource that has a constraint that is meaningful to the application or system.
- Create workflow and infrastructure management procedures to change the resource type if the constraint is near utilization. This workflow should include load testing as a best practice to verify that new type is the correct resource type with the new constraints.
- Migrate identified resource to the recommended new resource type, using existing procedures and processes.

Resources

Related best practices:

- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP03 Accommodate fixed service quotas and constraints through architecture
- REL01-BP04 Monitor and manage quotas
- REL01-BP05 Automate quota management
- REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover
- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP05 Test resiliency using chaos engineering

Related documents:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase

- Service endpoints and quotas
- Service Quotas User Guide
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN Partner: partners that can help with configuration management
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

Related tools:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager

- AWS Marketplace

REL01-BP02 Manage service quotas across accounts and regions

If you are using multiple accounts or Regions, request the appropriate quotas in all environments in which your production workloads run.

Desired outcome: Services and applications should not be affected by service quota exhaustion for configurations that span accounts or Regions or that have resilience designs using zone, Region, or account failover.

Common anti-patterns:

- Allowing resource usage in one isolation Region to grow with no mechanism to maintain capacity in the other ones.
- Manually setting all quotas independently in isolation Regions.
- Not considering the effect of resiliency architectures (like active or passive) in future quota needs during a degradation in the non-primary Region.
- Not evaluating quotas regularly and making necessary changes in every Region and account the workload runs.
- Not leveraging quota request templates to request increases across multiple Regions and accounts.
- Not updating service quotas due to incorrectly thinking that increasing quotas has cost implications like compute reservation requests.

Benefits of establishing this best practice: Verifying that you can handle your current load in secondary regions or accounts if regional services become unavailable. This can help reduce the number of errors or levels of degradations that occur during region loss.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Service quotas are tracked per account. Unless otherwise noted, each quota is AWS Region-specific. In addition to the production environments, also manage quotas in all applicable non-production environments so that testing and development are not hindered. Maintaining a high degree of resiliency requires that service quotas are assessed continually (whether automated or manual).

With more workloads spanning Regions due to the implementation of designs using *Active/Active*, *Active/Passive – Hot*, *Active/Passive-Cold*, and *Active/Passive-Pilot Light* approaches, it is essential to understand all Region and account quota levels. Past traffic patterns are not always a good indicator if the service quota is set correctly.

Equally important, the service quota name limit is not always the same for every Region. In one Region, the value could be five, and in another region the value could be ten. Management of these quotas must span all the same services, accounts, and Regions to provide consistent resilience under load.

Reconcile all the service quota differences across different Regions (Active Region or Passive Region) and create processes to continually reconcile these differences. The testing plans of passive Region failovers are rarely scaled to peak active capacity, meaning that game day or table top exercises can fail to find differences in service quotas between Regions and also then maintain the correct limits.

Service quota drift, the condition where service quota limits for a specific named quota is changed in one Region and not all Regions, is very important to track and assess. Changing the quota in Regions with traffic or potentially could carry traffic should be considered.

- Select relevant accounts and Regions based on your service requirements, latency, regulatory, and disaster recovery (DR) requirements.
- Identify service quotas across all relevant accounts, Regions, and Availability Zones. The limits are scoped to account and Region. These values should be compared for differences.

Implementation steps

- Review Service Quotas values that might have breached beyond the a risk level of usage. AWS Trusted Advisor provides alerts for 80% and 90% threshold breaches.
- Review values for service quotas in any Passive Regions (in an Active/Passive design). Verify that load will successfully run in secondary Regions in the event of a failure in the primary Region.
- Automate assessing if any service quota drift has occurred between Regions in the same account and act accordingly to change the limits.
- If the customer Organizational Units (OU) are structured in the supported manner, service quota templates should be updated to reflect changes in any quotas that should be applied to multiple Regions and accounts.
  - Create a template and associate Regions to the quota change.

- Review all existing service quota templates for any changes required (Region, limits, and accounts).

Resources

Related best practices:

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP03 Accommodate fixed service quotas and constraints through architecture
- REL01-BP04 Monitor and manage quotas
- REL01-BP05 Automate quota management
- REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover
- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP05 Test resiliency using chaos engineering

Related documents:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy

- AWS for Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN Partner: partners that can help with configuration management
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

Related services:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP03 Accommodate fixed service quotas and constraints through architecture

Be aware of unchangeable service quotas, service constraints, and physical resource limits. Design architectures for applications and services to prevent these limits from impacting reliability.

Examples include network bandwidth, serverless function invocation payload size, throttle burst rate for of an API gateway, and concurrent user connections to a database.

Desired outcome: The application or service performs as expected under normal and high traffic conditions. They have been designed to work within the limitations for that resource's fixed constraints or service quotas.

Common anti-patterns:

- Choosing a design that uses a resource of a service, unaware that there are design constraints that will cause this design to fail as you scale.
- Performing benchmarking that is unrealistic and will reach service fixed quotas during the testing. For example, running tests at a burst limit but for an extended amount of time.
- Choosing a design that cannot scale or be modified if fixed service quotas are to be exceeded. For example, an SQS payload size of 256KB.
- Observability has not been designed and implemented to monitor and alert on thresholds for service quotas that might be at risk during high traffic events

Benefits of establishing this best practice: Verifying that the application will run under all projected services load levels without disruption or degradation.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Unlike soft service quotas or resources that be replaced with higher capacity units, AWS services' fixed quotas cannot be changed. This means that all these type of AWS services must be evaluated for potential hard capacity limits when used in an application design.

Hard limits are show in the Service Quotas console. If the columns shows `ADJUSTABLE = No`, the service has a hard limit. Hard limits are also shown in some resources configuration pages. For example, Lambda has specific hard limits that cannot be adjusted.

As an example, when designing a python application to run in a Lambda function, the application should be evaluated to determine if there is any chance of Lambda running longer than 15 minutes. If the code may run more than this service quota limit, alternate technologies or designs must be considered. If this limit is reached after production deployment, the application will suffer degradation and disruption until it can be remediated. Unlike soft quotas, there is no method to change to these limits even under emergency Severity 1 events.

Once the application has been deployed to a testing environment, strategies should be used to find if any hard limits can be reached. Stress testing, load testing, and chaos testing should be part of the introduction test plan.

Implementation steps

- Review the complete list of AWS services that could be used in the application design phase.
- Review the soft quota limits and hard quota limits for all these services. Not all limits are shown in the Service Quotas console. Some services describe these limits in alternate locations.
- As you design your application, review your workload's business and technology drivers, such as business outcomes, use case, dependent systems, availability targets, and disaster recovery objects. Let your business and technology drivers guide the process to identify the distributed system that is right for your workload.
- Analyze service load across Regions and accounts. Many hard limits are regionally based for services. However, some limits are account based.
- Analyze resilience architectures for resource usage during a zonal failure and Regional failure. In the progression of multi-Region designs using active/active, active/passive – hot, active/passive - cold, and active/passive - pilot light approaches, these failure cases will cause higher usage. This creates a potential use case for hitting hard limits.

Resources

Related best practices:

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP04 Monitor and manage quotas
- REL01-BP05 Automate quota management
- REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover
- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers

- REL12-BP05 Test resiliency using chaos engineering

Related documents:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN Partner: partners that can help with configuration management
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas

- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

Related tools:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP04 Monitor and manage quotas

Evaluate your potential usage and increase your quotas appropriately, allowing for planned growth in usage.

Desired outcome: Active and automated systems that manage and monitor have been deployed. These operations solutions ensure that quota usage thresholds are nearing being reached. These would be proactively remediated by requested quota changes.

Common anti-patterns:

- Not configuring monitoring to check for service quota thresholds
- Not configuring monitoring for hard limits, even though those values cannot be changed.
- Assuming that amount of time required to request and secure a soft quota change is immediate or a short period.
- Configuring alarms for when service quotas are being approached, but having no process on how to respond to an alert.

- Only configuring alarms for services supported by AWS Service Quotas and not monitoring other AWS services.
- Not considering quota management for multiple Region resiliency designs, like active/active, active/passive – hot, active/passive - cold, and active/passive - pilot light approaches.
- Not assessing quota differences between Regions.
- Not assessing the needs in every Region for a specific quota increase request.
- Not leveraging templates for multi-Region quota management.

Benefits of establishing this best practice: Automatic tracking of the AWS Service Quotas and monitoring your usage against those quotas will allow you to see when you are approaching a quota limit. You can also use this monitoring data to help limit any degradations due to quota exhaustion.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

For supported services, you can monitor your quotas by configuring various different services that can assess and then send alerts or alarms. This can aid in monitoring usage and can alert you to approaching quotas. These alarms can be invoked from AWS Config, Lambda functions, Amazon CloudWatch, or from AWS Trusted Advisor. You can also use metric filters on CloudWatch Logs to search and extract patterns in logs to determine if usage is approaching quota thresholds.

Implementation steps

For monitoring:

- Capture current resource consumption (for example, buckets or instances). Use service API operations, such as the Amazon EC2 `DescribeInstances` API, to collect current resource consumption.
- Capture your current quotas that are essential and applicable to the services using:
  - AWS Service Quotas
  - AWS Trusted Advisor
  - AWS documentation
  - AWS service-specific pages
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)

- Use AWS Service Quotas, an AWS service that helps you manage your quotas for over 250 AWS services from one location.
- Use Trusted Advisor service limits to monitor your current service limits at various thresholds.
- Use the service quota history (console or AWS CLI) to check on regional increases.
- Compare service quota changes in each Region and each account to create equivalency, if required.

For management:

- Automated: Set up an AWS Config custom rule to scan service quotas across Regions and compare for differences.
- Automated: Set up a scheduled Lambda function to scan service quotas across Regions and compare for differences.
- Manual: Scan services quota through AWS CLI, API, or AWS Console to scan service quotas across Regions and compare for differences. Report the differences.
- If differences in quotas are identified between Regions, request a quota change, if required.
- Review the result of all requests.

Resources

Related best practices:

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP03 Accommodate fixed service quotas and constraints through architecture
- REL01-BP05 Automate quota management
- REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover
- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP05 Test resiliency using chaos engineering

Related documents:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN Partner: partners that can help with configuration management
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

Related tools:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP05 Automate quota management

Implement tools to alert you when thresholds are being approached. You can automate quota increase requests by using AWS Service Quotas APIs.

If you integrate your Configuration Management Database (CMDB) or ticketing system with Service Quotas, you can automate the tracking of quota increase requests and current quotas. In addition to the AWS SDK, Service Quotas offers automation using the AWS Command Line Interface (AWS CLI).

Common anti-patterns:

- Tracking the quotas and usage in spreadsheets.
- Running reports on usage daily, weekly, or monthly, and then comparing usage to the quotas.

Benefits of establishing this best practice: Automated tracking of the AWS service quotas and monitoring of your usage against that quota allows you to see when you are approaching a quota. You can set up automation to assist you in requesting a quota increase when needed. You might want to consider lowering some quotas when your usage trends in the opposite direction to realize the benefits of lowered risk (in case of compromised credentials) and cost savings.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Set up automated monitoring Implement tools using SDKs to alert you when thresholds are being approached.
  - Use Service Quotas and augment the service with an automated quota monitoring solution, such as AWS Limit Monitor or an offering from AWS Marketplace.
    - What is Service Quotas?
    - Quota Monitor on AWS - AWS Solution
  - Set up automated responses based on quota thresholds, using Amazon SNS and AWS Service Quotas APIs.
  - Test automation.
    - Configure limit thresholds.
    - Integrate with change events from AWS Config, deployment pipelines, Amazon EventBridge, or third parties.
    - Artificially set low quota thresholds to test responses.
    - Set up automated operations to take appropriate action on notifications and contact AWS Support when necessary.
    - Manually start change events.
    - Run a game day to test the quota increase change process.

Resources

Related documents:

- APN Partner: partners that can help with configuration management
- AWS Marketplace: CMDB products that help track limits
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- Quota Monitor on AWS - AWS Solution
- Amazon EC2 Service Limits
- What is Service Quotas?

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas

REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover

When a resource fails or is inaccessible, that resource might still be counted against a quota until it's successfully terminated. Verify that your quotas cover the overlap of failed or inaccessible resources and their replacements. You should consider use cases like network failure, Availability Zone failure, or Regional failures when calculating this gap.

Desired outcome: Small or large failures in resources or resource accessibility can be covered within the current service thresholds. Zone failures, network failures, or even Regional failures have been considered in the resource planning.

Common anti-patterns:

- Setting service quotas based on current needs without accounting for failover scenarios.
- Not considering the principals of static stability when calculating the peak quota for a service.
- Not considering the potential of inaccessible resources in calculating total quota needed for each Region.
- Not considering AWS service fault isolation boundaries for some services and their potential abnormal usage patterns.

Benefits of establishing this best practice: When a service disruption events impact application availability, the cloud allows you to implement strategies to mitigate or recover from these events. Such strategies often include creating additional resources to replace failed or inaccessible ones. Your quota strategy would accommodate these failover conditions and not layer in additional degradations due to service limit exhaustion.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

When evaluating quota limits, consider failover cases that might occur due to some degradation. The following types of failover cases should be considered:

- A VPC that is disrupted or inaccessible.
- A Subnet that is inaccessible.

- An Availability Zone has been degraded sufficiently to impact the accessibility of many resources.
- Various networking routes or ingress and egress points are blocked or changed.
- A Region has been degraded sufficiently to impact the accessibility of many resources.
- There are multiple resources but not all are affected by a failure in a Region or an Availability Zone.

Failures like the ones listed could be the reason to initiate a failover event. The decision to failover is unique for each situation and customer, as the business impact can vary dramatically. However, when operationally deciding to failover application or services, the capacity planning of resources in the failover location and their related quotas must be addressed before the event.

Review the service quotas for each service considering the high than normal peaks that might occur. These peaks might be related to resources that can be reached due to networking or permissions but are still active. Unterminated active resources will still be counted against the service quota limit.

Implementation steps

- Verify that there is enough gap between your service quota and your maximum usage to accommodate for a failover or loss of accessibility.
- Determine your service quotas, accounting for your deployment patterns, availability requirements, and consumption growth.
- Request quota increases if necessary. Plan for necessary time for quota increase requests to be fulfilled.
- Determine your reliability requirements (also known as your number of nines).
- Establish your fault scenarios (for example, loss of a component, an Availability Zone, or a Region).
- Establish your deployment methodology (for example, canary, blue/green, red/black, or rolling).
- Include an appropriate buffer (for example, 15%) to the current limit.
- Include calculations for static stability (Zonal and Regional) where appropriate.
- Plan consumption growth (for example, monitor your trends in consumption).
- Consider the impact of static stability for your most critical workloads. Assess resources conforming to a statically stable system in all Regions and Availability Zones.

- Consider the use of On-Demand Capacity Reservations to schedule capacity ahead of any failover. This can a useful strategy during the most critical business schedules to reduce potential risks of obtaining the correct quantity and type of resources during failover.

Resources

Related best practices:

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP03 Accommodate fixed service quotas and constraints through architecture
- REL01-BP04 Monitor and manage quotas
- REL01-BP05 Automate quota management
- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP05 Test resiliency using chaos engineering

Related documents:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries

- Availability with redundancy
- AWS for Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN Partner: partners that can help with configuration management
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

Related tools:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL 2. How do you plan your network topology?

Workloads often exist in multiple environments. These include multiple cloud environments (both publicly accessible and private) and possibly your existing data center infrastructure. Plans must include network considerations such as intra- and intersystem connectivity, public IP address management, private IP address management, and domain name resolution.

Best practices

- REL02-BP01 Use highly available network connectivity for your workload public endpoints
- REL02-BP02 Provision redundant connectivity between private networks in the cloud and on-premises environments
- REL02-BP03 Ensure IP subnet allocation accounts for expansion and availability
- REL02-BP04 Prefer hub-and-spoke topologies over many-to-many mesh
- REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected

REL02-BP01 Use highly available network connectivity for your workload public endpoints

Building highly available network connectivity to public endpoints of your workloads can help you reduce downtime due to loss of connectivity and improve the availability and SLA of your workload. To achieve this, use highly available DNS, content delivery networks (CDNs), API gateways, load balancing, or reverse proxies.

Desired outcome: It is critical to plan, build, and operationalize highly available network connectivity for your public endpoints. If your workload becomes unreachable due to a loss in connectivity, even if your workload is running and available, your customers will see your system as down. By combining the highly available and resilient network connectivity for your workload's public endpoints, along with a resilient architecture for your workload itself, you can provide the best possible availability and service level for your customers.

AWS Global Accelerator, Amazon CloudFront, Amazon API Gateway, AWS Lambda Function URLs, AWS AppSync APIs, and Elastic Load Balancing (ELB) all provide highly available public endpoints. Amazon Route 53 provides a highly available DNS service for domain name resolution to verify that your public endpoint addresses can be resolved.

You can also evaluate AWS Marketplace software appliances for load balancing and proxying.

Common anti-patterns:

- Designing a highly available workload without planning out DNS and network connectivity for high availability.
- Using public internet addresses on individual instances or containers and managing the connectivity to them with DNS.
- Using IP addresses instead of domain names for locating services.
- Not testing out scenarios where connectivity to your public endpoints is lost.
- Not analyzing network throughput needs and distribution patterns.
- Not testing and planning for scenarios where internet network connectivity to your public endpoints of your workload might be interrupted.
- Providing content (like web pages, static assets, or media files) to a large geographic area and not using a content delivery network.
- Not planning for distributed denial of service (DDoS) attacks. DDoS attacks risk shutting out legitimate traffic and lowering availability for your users.

Benefits of establishing this best practice: Designing for highly available and resilient network connectivity ensures that your workload is accessible and available to your users.

Level of risk exposed if this best practice is not established: High

Implementation guidance

At the core of building highly available network connectivity to your public endpoints is the routing of the traffic. To verify your traffic is able to reach the endpoints, the DNS must be able to resolve the domain names to their corresponding IP addresses. Use a highly available and scalable Domain Name System (DNS) such as Amazon Route 53 to manage your domain's DNS records. You can also use health checks provided by Amazon Route 53. The health checks verify that your application is reachable, available, and functional, and they can be set up in a way that they mimic your user's behavior, such as requesting a web page or a specific URL. In case of failure, Amazon Route 53 responds to DNS resolution requests and directs the traffic to only health endpoints. You can also consider using Geo DNS and Latency Based Routing capabilities offered by Amazon Route 53.

To verify that your workload itself is highly available, use Elastic Load Balancing (ELB). Amazon Route 53 can be used to target traffic to ELB, which distributes the traffic to the target compute instances. You can also use Amazon API Gateway along with AWS Lambda for a serverless solution. Customers can also run workloads in multiple AWS Regions. With multi-site active/active pattern, the workload can serve traffic from multiple Regions. With a multi-site active/passive pattern, the workload serves traffic from the active region while data is replicated to the secondary region and

becomes active in the event of a failure in the primary region. Route 53 health checks can then be used to control DNS failover from any endpoint in a primary Region to an endpoint in a secondary Region, verifying that your workload is reachable and available to your users.

Amazon CloudFront provides a simple API for distributing content with low latency and high data transfer rates by serving requests using a network of edge locations around the world. Content delivery networks (CDNs) serve customers by serving content located or cached at a location near to the user. This also improves availability of your application as the load for content is shifted away from your servers over to CloudFront's edge locations. The edge locations and regional edge caches hold cached copies of your content close to your viewers resulting in quick retrieval and increasing reachability and availability of your workload.

For workloads with users spread out geographically, AWS Global Accelerator helps you improve the availability and performance of the applications. AWS Global Accelerator provides Anycast static IP addresses that serve as a fixed entry point to your application hosted in one or more AWS Regions. This allows traffic to ingress onto the AWS global network as close to your users as possible, improving reachability and availability of your workload. AWS Global Accelerator also monitors the health of your application endpoints by using TCP, HTTP, and HTTPS health checks. Any changes in the health or configuration of your endpoints permit redirection of user traffic to healthy endpoints that deliver the best performance and availability to your users. In addition, AWS Global Accelerator has a fault-isolating design that uses two static IPv4 addresses that are serviced by independent network zones increasing the availability of your applications.

To help protect customers from DDoS attacks, AWS provides AWS Shield Standard. Shield Standard comes automatically turned on and protects from common infrastructure (layer 3 and 4) attacks like SYN/UDP floods and reflection attacks to support high availability of your applications on AWS. For additional protections against more sophisticated and larger attacks (like UDP floods), state exhaustion attacks (like TCP SYN floods), and to help protect your applications running on Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator, and Route 53, you can consider using AWS Shield Advanced. For protection against Application layer attacks like HTTP POST or GET floods, use AWS WAF. AWS WAF can use IP addresses, HTTP headers, HTTP body, URI strings, SQL injection, and cross-site scripting conditions to determine if a request should be blocked or allowed.

Implementation steps

1. Set up highly available DNS: Amazon Route 53 is a highly available and scalable domain name system (DNS) web service. Route 53 connects user requests to internet applications running

on AWS or on-premises. For more information, see configuring Amazon Route 53 as your DNS service.

2. Setup health checks: When using Route 53, verify that only healthy targets are resolvable. Start by creating Route 53 health checks and configuring DNS failover. The following aspects are important to consider when setting up health checks:
   a. How Amazon Route 53 determines whether a health check is healthy
   b. Creating, updating, and deleting health checks
   c. Monitoring health check status and getting notifications
   d. Best practices for Amazon Route 53 DNS
3. Connect your DNS service to your endpoints.
   a. When using Elastic Load Balancing as a target for your traffic, create an alias record using Amazon Route 53 that points to your load balancer's regional endpoint. During the creation of the alias record, set the Evaluate target health option to Yes.
   b. For serverless workloads or private APIs when API Gateway is used, use Route 53 to direct traffic to API Gateway.
4. Decide on a content delivery network.
   a. For delivering content using edge locations closer to the user, start by understanding how CloudFront delivers content.
   b. Get started with a simple CloudFront distribution. CloudFront then knows where you want the content to be delivered from, and the details about how to track and manage content delivery. The following aspects are important to understand and consider when setting up CloudFront distribution:
      i. How caching works with CloudFront edge locations
      ii. Increasing the proportion of requests that are served directly from the CloudFront caches (cache hit ratio)
      iii. Using Amazon CloudFront Origin Shield
      iv. Optimizing high availability with CloudFront origin failover
5. Set up application layer protection: AWS WAF helps you protect against common web exploits and bots that can affect availability, compromise security, or consume excessive resources. To get a deeper understanding, review how AWS WAF works and when you are ready to implement protections from application layer HTTP POST AND GET floods, review Getting started with AWS WAF. You can also use AWS WAF with CloudFront see the documentation on how AWS WAF works with Amazon CloudFront features.

6. Set up additional DDoS protection: By default, all AWS customers receive protection from common, most frequently occurring network and transport layer DDoS attacks that target your web site or application with AWS Shield Standard at no additional charge. For additional protection of internet-facing applications running on Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator, and Amazon Route 53 you can consider AWS Shield Advanced and review examples of DDoS resilient architectures. To protect your workload and your public endpoints from DDoS attacks review Getting started with AWS Shield Advanced.

Resources

Related best practices:

- REL10-BP01 Deploy the workload to multiple locations
- REL10-BP02 Select the appropriate locations for your multi-location deployment
- REL11-BP04 Rely on the data plane and not the control plane during recovery
- REL11-BP06 Send notifications when events impact availability

Related documents:

- APN Partner: partners that can help plan your networking
- AWS Marketplace for Network Infrastructure
- What Is AWS Global Accelerator?
- What is Amazon CloudFront?
- What is Amazon Route 53?
- What is Elastic Load Balancing?
- Network Connectivity capability - Establishing Your Cloud Foundations
- What is Amazon API Gateway?
- What are AWS WAF, AWS Shield, and AWS Firewall Manager?
- What is Amazon Route 53 Application Recovery Controller?
- Configure custom health checks for DNS failover

Related videos:

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator

- AWS re:Invent 2020: Global traffic management with Amazon Route 53
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications
- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure
- AWS re:Invent 2022 - Building resilient networks

Related examples:

- Disaster Recovery with Amazon Route 53 Application Recovery Controller (ARC)
- Reliability Workshops
- AWS Global Accelerator Workshop

REL02-BP02 Provision redundant connectivity between private networks in the cloud and on-premises environments

Use multiple AWS Direct Connect connections or VPN tunnels between separately deployed private networks. Use multiple Direct Connect locations for high availability. If using multiple AWS Regions, ensure redundancy in at least two of them. You might want to evaluate AWS Marketplace appliances that terminate VPNs. If you use AWS Marketplace appliances, deploy redundant instances for high availability in different Availability Zones.

AWS Direct Connect is a cloud service that makes it easy to establish a dedicated network connection from your on-premises environment to AWS. Using Direct Connect Gateway, your on-premises data center can be connected to multiple AWS VPCs spread across multiple AWS Regions.

This redundancy addresses possible failures that impact connectivity resiliency:

- How are you going to be resilient to failures in your topology?
- What happens if you misconfigure something and remove connectivity?
- Will you be able to handle an unexpected increase in traffic or use of your services?
- Will you be able to absorb an attempted Distributed Denial of Service (DDoS) attack?

When connecting your VPC to your on-premises data center via VPN, you should consider the resiliency and bandwidth requirements that you need when you select the vendor and instance size on which you need to run the appliance. If you use a VPN appliance that is not resilient in its implementation, then you should have a redundant connection through a second appliance. For all

these scenarios, you need to define an acceptable time to recovery and test to ensure that you can meet those requirements.

If you choose to connect your VPC to your data center using a Direct Connect connection and you need this connection to be highly available, have redundant Direct Connect connections from each data center. The redundant connection should use a second Direct Connect connection from different location than the first. If you have multiple data centers, ensure that the connections terminate at different locations. Use the Direct Connect Resiliency Toolkit to help you set this up.

If you choose to fail over to VPN over the internet using Site-to-Site VPN, it's important to understand that it supports up to 1.25-Gbps throughput per VPN tunnel, but does not support Equal Cost Multi Path (ECMP) for outbound traffic in the case of multiple AWS Managed VPN tunnels terminating on the same VGW. We do not recommend that you use AWS Managed VPN as a backup for Direct Connect connections unless you can tolerate speeds less than 1 Gbps during failover.

You can also use VPC endpoints to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without traversing the public internet. Endpoints are virtual devices. They are horizontally scaled, redundant, and highly available VPC components. They allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.

Common anti-patterns:

- Having only one connectivity provider between your on-site network and AWS.
- Consuming the connectivity capabilities of your AWS Direct Connect connection, but only having one connection.
- Having only one path for your VPN connectivity.

Benefits of establishing this best practice: By implementing redundant connectivity between your cloud environment and you corporate or on-premises environment, you can ensure that the dependent services between the two environments can communicate reliably.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Ensure that you have highly available connectivity between AWS and on-premises environment. Use multiple AWS Direct Connect connections or VPN tunnels between separately deployed

private networks. Use multiple Direct Connect locations for high availability. If using multiple AWS Regions, ensure redundancy in at least two of them. You might want to evaluate AWS Marketplace appliances that terminate VPNs. If you use AWS Marketplace appliances, deploy redundant instances for high availability in different Availability Zones.

- Ensure that you have a redundant connection to your on-premises environment You may need redundant connections to multiple AWS Regions to achieve your availability needs.
  - AWS Direct Connect Resiliency Recommendations
  - Using Redundant Site-to-Site VPN Connections to Provide Failover
    - Use service API operations to identify correct use of Direct Connect circuits.
      - DescribeConnections
      - DescribeConnectionsOnInterconnect
      - DescribeDirectConnectGatewayAssociations
      - DescribeDirectConnectGatewayAttachments
      - DescribeDirectConnectGateways
      - DescribeHostedConnections
      - DescribeInterconnects
    - If only one Direct Connect connection exists or you have none, set up redundant VPN tunnels to your virtual private gateways.
      - What is AWS Site-to-Site VPN?
- Capture your current connectivity (for example, Direct Connect, virtual private gateways, AWS Marketplace appliances).
  - Use service API operations to query configuration of Direct Connect connections.
    - DescribeConnections
    - DescribeConnectionsOnInterconnect
    - DescribeDirectConnectGatewayAssociations
    - DescribeDirectConnectGatewayAttachments
    - DescribeDirectConnectGateways
    - DescribeHostedConnections
    - DescribeInterconnects
  - Use service API operations to collect virtual private gateways where route tables use them.
    - DescribeVpnGateways

    - DescribeRouteTables
- Use service API operations to collect AWS Marketplace applications where route tables use them.
    - DescribeRouteTables

Resources

Related documents:

- APN Partner: partners that can help plan your networking
- AWS Direct Connect Resiliency Recommendations
- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Multiple data center HA network connectivity
- Using Redundant Site-to-Site VPN Connections to Provide Failover
- Using the Direct Connect Resiliency Toolkit to get started
- VPC Endpoints and VPC Endpoint Services (AWS PrivateLink)
- What Is Amazon VPC?
- What Is a Transit Gateway?
- What is AWS Site-to-Site VPN?
- Working with Direct Connect Gateways

Related videos:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)

REL02-BP03 Ensure IP subnet allocation accounts for expansion and availability

Amazon VPC IP address ranges must be large enough to accommodate workload requirements, including factoring in future expansion and allocation of IP addresses to subnets across Availability Zones. This includes load balancers, EC2 instances, and container-based applications.

When you plan your network topology, the first step is to define the IP address space itself. Private IP address ranges (following RFC 1918 guidelines) should be allocated for each VPC. Accommodate the following requirements as part of this process:

- Allow IP address space for more than one VPC per Region.
- Within a VPC, allow space for multiple subnets that span multiple Availability Zones.
- Always leave unused CIDR block space within a VPC for future expansion.
- Ensure that there is IP address space to meet the needs of any transient fleets of EC2 instances that you might use, such as Spot Fleets for machine learning, Amazon EMR clusters, or Amazon Redshift clusters.
- Note that the first four IP addresses and the last IP address in each subnet CIDR block are reserved and not available for your use.
- You should plan on deploying large VPC CIDR blocks. Note that the initial VPC CIDR block allocated to your VPC cannot be changed or deleted, but you can add additional non-overlapping CIDR blocks to the VPC. Subnet IPv4 CIDRs cannot be changed, however IPv6 CIDRs can. Keep in mind that deploying the largest VPC possible (/16) results in over 65,000 IP addresses. In the base 10.x.x.x IP address space alone, you could provision 255 such VPCs. You should therefore err on the side of being too large rather than too small to make it easier to manage your VPCs.

Common anti-patterns:

- Creating small VPCs.
- Creating small subnets and then having to add subnets to configurations as you grow.
- Incorrectly estimating how many IP addresses a elastic load balancer can use.
- Deploying many high traffic load balancers into the same subnets.

Benefits of establishing this best practice: This ensures that you can accommodate the growth of your workloads and continue to provide availability as you scale up.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Plan your network to accommodate for growth, regulatory compliance, and integration with others. Growth can be underestimated, regulatory compliance can change, and acquisitions or private network connections can be difficult to implement without proper planning.
  - Select relevant AWS accounts and Regions based on your service requirements, latency, regulatory, and disaster recovery (DR) requirements.
  - Identify your needs for regional VPC deployments.
  - Identify the size of the VPCs.
    - Determine if you are going to deploy multi-VPC connectivity.
      - What Is a Transit Gateway?
      - Single Region Multi-VPC Connectivity
    - Determine if you need segregated networking for regulatory requirements.
    - Make VPCs as large as possible. The initial VPC CIDR block allocated to your VPC cannot be changed or deleted, but you can add additional non-overlapping CIDR blocks to the VPC. This however may fragment your address ranges.

Resources

Related documents:

- APN Partner: partners that can help plan your networking
- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Multiple data center HA network connectivity
- Single Region Multi-VPC Connectivity
- What Is Amazon VPC?

Related videos:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)

REL02-BP04 Prefer hub-and-spoke topologies over many-to-many mesh

If more than two network address spaces (for example, VPCs and on-premises networks) are connected via VPC peering, AWS Direct Connect, or VPN, then use a hub-and-spoke model, like that provided by AWS Transit Gateway.

If you have only two such networks, you can simply connect them to each other, but as the number of networks grows, the complexity of such meshed connections becomes untenable. AWS Transit Gateway provides an easy to maintain hub-and-spoke model, allowing the routing of traffic across your multiple networks.

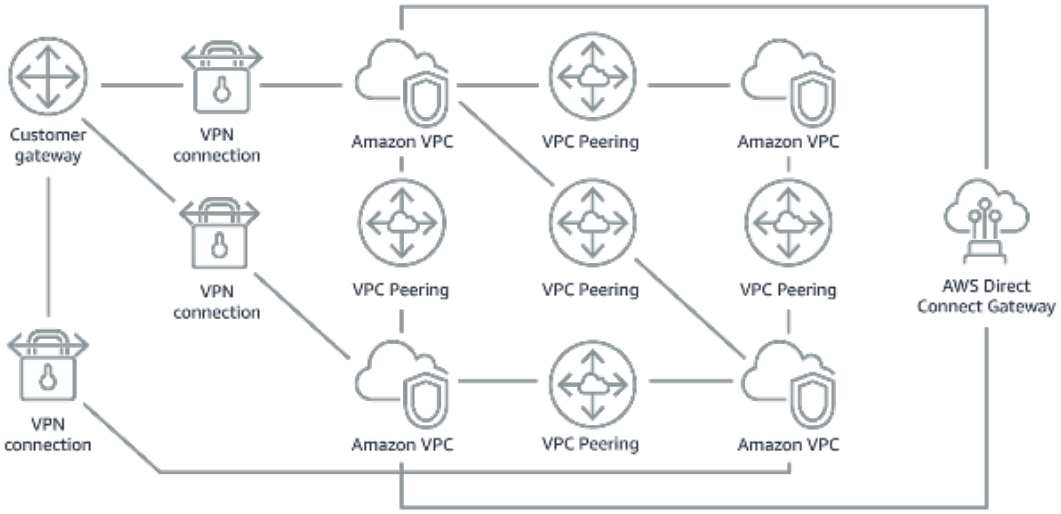


Figure 1: Without AWS Transit Gateway: You need to peer each Amazon VPC to each other and to each onsite location using a VPN connection, which can become complex as it scales.

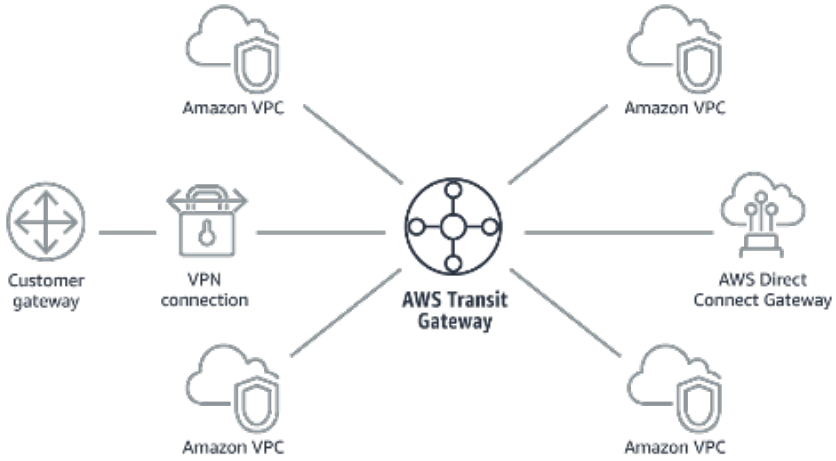


Figure 2: With AWS Transit Gateway: You simply connect each Amazon VPC or VPN to the AWS Transit Gateway and it routes traffic to and from each VPC or VPN.

Common anti-patterns:

- Using VPC peering to connect more than two VPCs.
- Establishing multiple BGP sessions for each VPC to establish connectivity that spans Virtual Private Clouds (VPCs) spread across multiple AWS Regions.

Benefits of establishing this best practice: As the number of networks grows, the complexity of such meshed connections becomes untenable. AWS Transit Gateway provides an easy to maintain hub-and-spoke model, allowing routing of traffic among your multiple networks.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Prefer hub-and-spoke topologies over many-to-many mesh. If more than two network address spaces (VPCs, on-premises networks) are connected via VPC peering, AWS Direct Connect, or VPN, then use a hub-and-spoke model like that provided by AWS Transit Gateway.
  - For only two such networks, you can simply connect them to each other, but as the number of networks grows, the complexity of such meshed connections becomes untenable. AWS Transit Gateway provides an easy to maintain hub-and-spoke model, allowing routing of traffic across your multiple networks.
    - What Is a Transit Gateway?

Resources

Related documents:

- APN Partner: partners that can help plan your networking
- AWS Marketplace for Network Infrastructure
- Multiple data center HA network connectivity
- VPC Endpoints and VPC Endpoint Services (AWS PrivateLink)
- What Is Amazon VPC?
- What Is a Transit Gateway?

Related videos:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)

REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected

The IP address ranges of each of your VPCs must not overlap when peered or connected via VPN. You must similarly avoid IP address conflicts between a VPC and on-premises environments or with other cloud providers that you use. You must also have a way to allocate private IP address ranges when needed.

An IP address management (IPAM) system can help with this. Several IPAMs are available from the AWS Marketplace.

Common anti-patterns:

- Using the same IP range in your VPC as you have on premises or in your corporate network.
- Not tracking IP ranges of VPCs used to deploy your workloads.

Benefits of establishing this best practice: Active planning of your network will ensure that you do not have multiple occurrences of the same IP address in interconnected networks. This prevents routing problems from occurring in parts of the workload that are using the different applications.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Monitor and manage your CIDR use. Evaluate your potential usage on AWS, add CIDR ranges to existing VPCs, and create VPCs to allow planned growth in usage.
  - Capture current CIDR consumption (for example, VPCs, subnets)
    - Use service API operations to collect current CIDR consumption.
  - Capture your current subnet usage.
    - Use service API operations to collect subnets per VPC in each Region.
      - DescribeSubnets
    - Record the current usage.

- Determine if you created any overlapping IP ranges.
- Calculate the spare capacity.
- Identify overlapping IP ranges. You can either migrate to a new range of addresses or use Network and Port Translation (NAT) appliances from AWS Marketplace if you need to connect the overlapping ranges.

Resources

Related documents:

- APN Partner: partners that can help plan your networking
- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Multiple data center HA network connectivity
- What Is Amazon VPC?
- What is IPAM?

Related videos:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)

Workload architecture

Questions

- REL 3. How do you design your workload service architecture?
- REL 4. How do you design interactions in a distributed system to prevent failures?
- REL 5. How do you design interactions in a distributed system to mitigate or withstand failures?

REL 3. How do you design your workload service architecture?

Build highly scalable and reliable workloads using a service-oriented architecture (SOA) or a microservices architecture. Service-oriented architecture (SOA) is the practice of making software

components reusable via service interfaces. Microservices architecture goes further to make components smaller and simpler.

Best practices

- REL03-BP01 Choose how to segment your workload
- REL03-BP02 Build services focused on specific business domains and functionality
- REL03-BP03 Provide service contracts per API

REL03-BP01 Choose how to segment your workload

Workload segmentation is important when determining the resilience requirements of your application. Monolithic architecture should be avoided whenever possible. Instead, carefully consider which application components can be broken out into microservices. Depending on your application requirements, this may end up being a combination of a service-oriented architecture (SOA) with microservices where possible. Workloads that are capable of statelessness are more capable of being deployed as microservices.

Desired outcome: Workloads should be supportable, scalable, and as loosely coupled as possible.

When making choices about how to segment your workload, balance the benefits against the complexities. What is right for a new product racing to first launch is different than what a workload built to scale from the start needs. When refactoring an existing monolith, you will need to consider how well the application will support a decomposition towards statelessness. Breaking services into smaller pieces allows small, well-defined teams to develop and manage them. However, smaller services can introduce complexities which include possible increased latency, more complex debugging, and increased operational burden.

Common anti-patterns:

- The microservice *Death Star* is a situation in which the atomic components become so highly interdependent that a failure of one results in a much larger failure, making the components as rigid and fragile as a monolith.

Benefits of establishing this practice:

- More specific segments lead to greater agility, organizational flexibility, and scalability.
- Reduced impact of service interruptions.

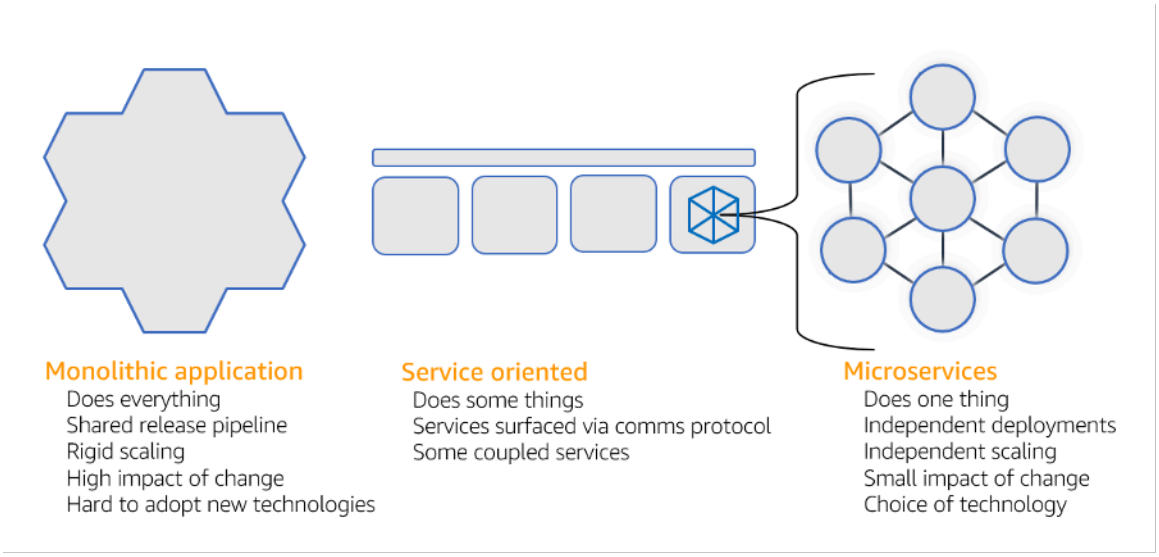
- Application components may have different availability requirements, which can be supported by a more atomic segmentation.
- Well-defined responsibilities for teams supporting the workload.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Choose your architecture type based on how you will segment your workload. Choose an SOA or microservices architecture (or in some rare cases, a monolithic architecture). Even if you choose to start with a monolith architecture, you must ensure that it's modular and can ultimately evolve to SOA or microservices as your product scales with user adoption. SOA and microservices offer respectively smaller segmentation, which is preferred as a modern scalable and reliable architecture, but there are trade-offs to consider, especially when deploying a microservice architecture.

One primary trade-off is that you now have a distributed compute architecture that can make it harder to achieve user latency requirements and there is additional complexity in the debugging and tracing of user interactions. You can use AWS X-Ray to assist you in solving this problem. Another effect to consider is increased operational complexity as you increase the number of applications that you are managing, which requires the deployment of multiple independency components.



Monolithic, service-oriented, and microservices architectures

Implementation steps

- Determine the appropriate architecture to refactor or build your application. SOA and microservices offer respectively smaller segmentation, which is preferred as a modern scalable and reliable architecture. SOA can be a good compromise for achieving smaller segmentation while avoiding some of the complexities of microservices. For more details, see Microservice Trade-Offs.
- If your workload is amenable to it, and your organization can support it, you should use a microservices architecture to achieve the best agility and reliability. For more details, see Implementing Microservices on AWS.
- Consider following the *Strangler Fig* pattern to refactor a monolith into smaller components. This involves gradually replacing specific application components with new applications and services. AWS Migration Hub Refactor Spaces acts as the starting point for incremental refactoring. For more details, see Seamlessly migrate on-premises legacy workloads using a strangler pattern.
- Implementing microservices may require a service discovery mechanism to allow these distributed services to communicate with each other. AWS App Mesh can be used with service-oriented architectures to provide reliable discovery and access of services. AWS Cloud Map can also be used for dynamic, DNS-based service discovery.
- If you're migrating from a monolith to SOA, Amazon MQ can help bridge the gap as a service bus when redesigning legacy applications in the cloud.
- For existing monoliths with a single, shared database, choose how to reorganize the data into smaller segments. This could be by business unit, access pattern, or data structure. At this point in the refactoring process, you should choose to move forward with a relational or non-relational (NoSQL) type of database. For more details, see From SQL to NoSQL.

Level of effort for the implementation plan: High

Resources

Related best practices:

- REL03-BP02 Build services focused on specific business domains and functionality

Related documents:

- Amazon API Gateway: Configuring a REST API Using OpenAPI

- What is Service-Oriented Architecture?
- Bounded Context (a central pattern in Domain-Driven Design)
- Implementing Microservices on AWS
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- Microservices on AWS
- What is AWS App Mesh?

Related examples:

- Iterative App Modernization Workshop

Related videos:

- Delivering Excellence with Microservices on AWS

REL03-BP02 Build services focused on specific business domains and functionality

This best practice was updated with new guidance on July 13th, 2023.

Service-oriented architectures (SOA) define services with well-delineated functions defined by business needs. Microservices use domain models and bounded context to draw service boundaries along business context boundaries. Focusing on business domains and functionality helps teams define independent reliability requirements for their services. Bounded contexts isolate and encapsulate business logic, allowing teams to better reason about how to handle failures.

Desired outcome: Engineers and business stakeholders jointly define bounded contexts and use them to design systems as services that fulfill specific business functions. These teams use established practices like event storming to define requirements. New applications are designed as services well-defined boundaries and loosely coupling. Existing monoliths are decomposed into bounded contexts and system designs move towards SOA or microservice architectures. When monoliths are refactored, established approaches like bubble contexts and monolith decomposition patterns are applied.

Domain-oriented services are executed as one or more processes that don't share state. They independently respond to fluctuations in demand and handle fault scenarios in light of domain specific requirements.

Common anti-patterns:

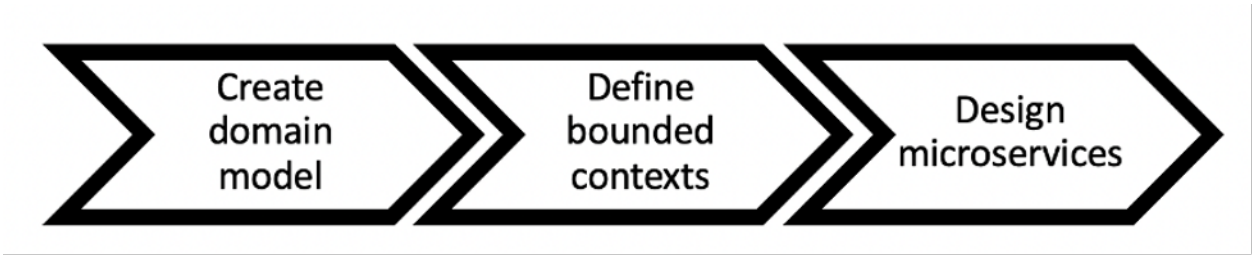
- Teams are formed around specific technical domains like UI and UX, middleware, or database instead of specific business domains.
- Applications span domain responsibilities. Services that span bounded contexts can be more difficult to maintain, require larger testing efforts, and require multiple domain teams to participate in software updates.
- Domain dependencies, like domain entity libraries, are shared across services such that changes for one service domain require changes to other service domains
- Service contracts and business logic don't express entities in a common and consistent domain language, resulting in translation layers that complicate systems and increase debugging efforts.

Benefits of establishing this best practice: Applications are designed as independent services bounded by business domains and use a common business language. Services are independently testable and deployable. Services meet domain specific resiliency requirements for the domain implemented.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Domain-driven decision (DDD) is the foundational approach of designing and building software around business domains. It's helpful to work with an existing framework when building services focused on business domains. When working with existing monolithic applications, you can take advantage of decomposition patterns that provide established techniques to modernize applications into services.



Domain-driven decision

Implementation steps

- Teams can hold event storming workshops to quickly identify events, commands, aggregates and domains in a lightweight sticky note format.
- Once domain entities and functions have been formed in a domain context, you can divide your domain into services using bounded context, where entities that share similar features and attributes are grouped together. With the model divided into contexts, a template for how to boundary microservices emerges.
  - For example, the Amazon.com website entities might include package, delivery, schedule, price, discount, and currency.
  - Package, delivery, and schedule are grouped into the shipping context, while price, discount, and currency are grouped into the pricing context.
- Decomposing monoliths into microservices outlines patterns for refactoring microservices. Using patterns for decomposition by business capability, subdomain, or transaction aligns well with domain-driven approaches.
- Tactical techniques such as the bubble context allow you to introduce DDD in existing or legacy applications without up-front rewrites and full commitments to DDD. In a bubble context approach, a small bounded context is established using a service mapping and coordination, or anti-corruption layer, which protects the newly defined domain model from external influences.

After teams have performed domain analysis and defined entities and service contracts, they can take advantage of AWS services to implement their domain-driven design as cloud-based services.

- Start your development by defining tests that exercise business rules of your domain. Test-driven development (TDD) and behavior-driven development (BDD) help teams keep services focused on solving business problems.
- Select the AWS services that best meet your business domain requirements and microservice architecture:
  - AWS Serverless allows your team focus on specific domain logic instead of managing servers and infrastructure.
  - Containers at AWS simplify the management of your infrastructure, so you can focus on your domain requirements.
  - Purpose built databases help you match your domain requirements to the best fit database type.

- Building hexagonal architectures on AWS outlines a framework to build business logic into services working backwards from a business domain to fulfill functional requirements and then attach integration adapters. Patterns that separate interface details from business logic with AWS services help teams focus on domain functionality and improve software quality.

Resources

Related best practices:

- REL03-BP01 Choose how to segment your workload
- REL03-BP03 Provide service contracts per API

Related documents:

- AWS Microservices
- Implementing Microservices on AWS
- How to break a Monolith into Microservices
- Getting Started with DDD when Surrounded by Legacy Systems
- Domain-Driven Design: Tackling Complexity in the Heart of Software
- Building hexagonal architectures on AWS
- Decomposing monoliths into microservices
- Event Storming
- Messages Between Bounded Contexts
- Microservices
- Test-driven development
- Behavior-driven development

Related examples:

- Enterprise Cloud Native Workshop
- Designing Cloud Native Microservices on AWS (from DDD/EventStormingWorkshop)

Related tools:

- AWS Cloud Databases
- Serverless on AWS
- Containers at AWS

REL03-BP03 Provide service contracts per API

This best practice was updated with new guidance on July 13th, 2023.

Service contracts are documented agreements between API producers and consumers defined in a machine-readable API definition. A contract versioning strategy allows consumers to continue using the existing API and migrate their applications to a newer API when they are ready. Producer deployment can happen any time as long as the contract is followed. Service teams can use the technology stack of their choice to satisfy the API contract.

Desired outcome:

Common anti-patterns: Applications built with service-oriented or microservice architectures are able to operate independently while having integrated runtime dependency. Changes deployed to an API consumer or producer do not interrupt the stability of the overall system when both sides follow a common API contract. Components that communicate over service APIs can perform independent functional releases, upgrades to runtime dependencies, or fail over to a disaster recovery (DR) site with little or no impact to each other. In addition, discrete services are able to independently scale absorbing resource demand without requiring other services to scale in unison.

- Creating service APIs without strongly typed schemas. This results in APIs that cannot be used to generate API bindings and payloads that can't be programmatically validated.
- Not adopting a versioning strategy, which forces API consumers to update and release or fail when service contracts evolve.
- Error messages that leak details of the underlying service implementation rather than describe integration failures in the domain context and language.
- Not using API contracts to develop test cases and mock API implementations to allow for independent testing of service components.

Benefits of establishing this best practice: Distributed systems composed of components that communicate over API service contracts can improve reliability. Developers can catch potential

issues early in the development process with type checking during compilation to verify that requests and responses follow the API contract and required fields are present. API contracts provide a clear self-documenting interface for APIs and provider better interoperability between different systems and programming languages.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Once you have identified business domains and determined your workload segmentation, you can develop your service APIs. First, define machine-readable service contracts for APIs, and then implement an API versioning strategy. When you are ready to integrate services over common protocols like REST, GraphQL, or asynchronous events, you can incorporate AWS services into your architecture to integrate your components with strongly-typed API contracts.

AWS services for service API contrats

Incorporate AWS services including Amazon API Gateway, AWS AppSync, and Amazon EventBridge into your architecture to use API service contracts in your application. Amazon API Gateway helps you integrate with directly native AWS services and other web services. API Gateway supports the OpenAPI specification and versioning. AWS AppSync is a managed GraphQL endpoint you configure by defining a GraphQL schema to define a service interface for queries, mutations and subscriptions. Amazon EventBridge uses event schemas to define events and generate code bindings for your events.

Implementation steps

- First, define a contract for your API. A contract will express the capabilities of an API as well as define strongly typed data objects and fields for the API input and output.
- When you configure APIs in API Gateway, you can import and export OpenAPI Specifications for your endpoints.
  - Importing an OpenAPI definition simplifies the creation of your API and can be integrated with AWS infrastructure as code tools like the AWS Serverless Application Model and AWS Cloud Development Kit (AWS CDK).
  - Exporting an API definition simplifies integrating with API testing tools and provides services consumer an integration specification.
- You can define and manage GraphQL APIs with AWS AppSync by defining a GraphQL schema file to generate your contract interface and simplify interaction with complex REST models, multiple database tables or legacy services.

- AWS Amplify projects that are integrated with AWS AppSync generate strongly typed JavaScript query files for use in your application as well as an AWS AppSync GraphQL client library for Amazon DynamoDB tables.
- When you consume service events from Amazon EventBridge, events adhere to schemas that already exist in the schema registry or that you define with the OpenAPI Spec. With a schema defined in the registry, you can also generate client bindings from the schema contract to integrate your code with events.
- Extending or version your API. Extending an API is a simpler option when adding fields that can be configured with optional fields or default values for required fields.
  - JSON based contracts for protocols like REST and GraphQL can be a good fit for contract extension.
  - XML based contracts for protocols like SOAP should be tested with service consumers to determine the feasibility of contract extension.
- When versioning an API, consider implementing proxy versioning where a facade is used to support versions so that logic can be maintained in a single codebase.
  - With API Gateway you can use request and response mappings to simplify absorbing contract changes by establishing a facade to provide default values for new fields or to strip removed fields from a request or response. With this approach the underlying service can maintain a single codebase.

Resources

Related best practices:

- REL03-BP01 Choose how to segment your workload
- REL03-BP02 Build services focused on specific business domains and functionality
- REL04-BP02 Implement loosely coupled dependencies
- REL05-BP03 Control and limit retry calls
- REL05-BP05 Set client timeouts

Related documents:

- What Is An API (Application Programming Interface)?
- Implementing Microservices on AWS
- Microservice Trade-Offs

- Microservices - a definition of this new architectural term
- Microservices on AWS
- Working with API Gateway extensions to OpenAPI
- OpenAPI-Specification
- GraphQL: Schemas and Types
- Amazon EventBridge code bindings

Related examples:

- Amazon API Gateway: Configuring a REST API Using OpenAPI
- Amazon API Gateway to Amazon DynamoDB CRUD application using OpenAPI
- Modern application integration patterns in a serverless age: API Gateway Service Integration
- Implementing header-based API Gateway versioning with Amazon CloudFront
- AWS AppSync: Building a client application

Related videos:

- Using OpenAPI in AWS SAM to manage API Gateway

Related tools:

- Amazon API Gateway
- AWS AppSync
- Amazon EventBridge

REL 4. How do you design interactions in a distributed system to prevent failures?

Distributed systems rely on communications networks to interconnect components, such as servers or services. Your workload must operate reliably despite data loss or latency in these networks. Components of the distributed system must operate in a way that does not negatively impact other components or the workload. These best practices prevent failures and improve mean time between failures (MTBF).

Best practices

- REL04-BP01 Identify which kind of distributed system is required
- REL04-BP02 Implement loosely coupled dependencies
- REL04-BP03 Do constant work
- REL04-BP04 Make all responses idempotent

REL04-BP01 Identify which kind of distributed system is required

Hard real-time distributed systems require responses to be given synchronously and rapidly, while soft real-time systems have a more generous time window of minutes or more for response. Offline systems handle responses through batch or asynchronous processing. Hard real-time distributed systems have the most stringent reliability requirements.

The most difficult challenges with distributed systems are for the hard real-time distributed systems, also known as request/reply services. What makes them difficult is that requests arrive unpredictably and responses must be given rapidly (for example, the customer is actively waiting for the response). Examples include front-end web servers, the order pipeline, credit card transactions, every AWS API, and telephony.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Identify which kind of distributed system is required. Challenges with distributed systems involved latency, scaling, understanding networking APIs, marshalling and unmarshalling data, and the complexity of algorithms such as Paxos. As the systems grow larger and more distributed, what had been theoretical edge cases turn into regular occurrences.
  - The Amazon Builders' Library: Challenges with distributed systems
    - Hard real-time distributed systems require responses to be given synchronously and rapidly.
    - Soft real-time systems have a more generous time window of minutes or greater for response.
    - Offline systems handle responses through batch or asynchronous processing.
    - Hard real-time distributed systems have the most stringent reliability requirements.

Resources

Related documents:

- Amazon EC2: Ensuring Idempotency
- The Amazon Builders' Library: Challenges with distributed systems
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
- What Is Amazon EventBridge?
- What Is Amazon Simple Queue Service?

Related videos:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL04-BP02 Implement loosely coupled dependencies

Dependencies such as queuing systems, streaming systems, workflows, and load balancers are loosely coupled. Loose coupling helps isolate behavior of a component from other components that depend on it, increasing resiliency and agility.

If changes to one component force other components that rely on it to also change, then they are *tightly* coupled. *Loose* coupling breaks this dependency so that dependent components only need to know the versioned and published interface. Implementing loose coupling between dependencies isolates a failure in one from impacting another.

Loose coupling allows you to add additional code or features to a component while minimizing risk to components that depend on it. Also, scalability is improved as you can scale out or even change underlying implementation of the dependency.

To further improve resiliency through loose coupling, make component interactions asynchronous where possible. This model is suitable for any interaction that does not need an immediate response and where an acknowledgment that a request has been registered will suffice. It involves one component that generates events and another that consumes them. The two components do not integrate through direct point-to-point interaction but usually through an intermediate durable storage layer, such as an SQS queue or a streaming data platform such as Amazon Kinesis, or AWS Step Functions.

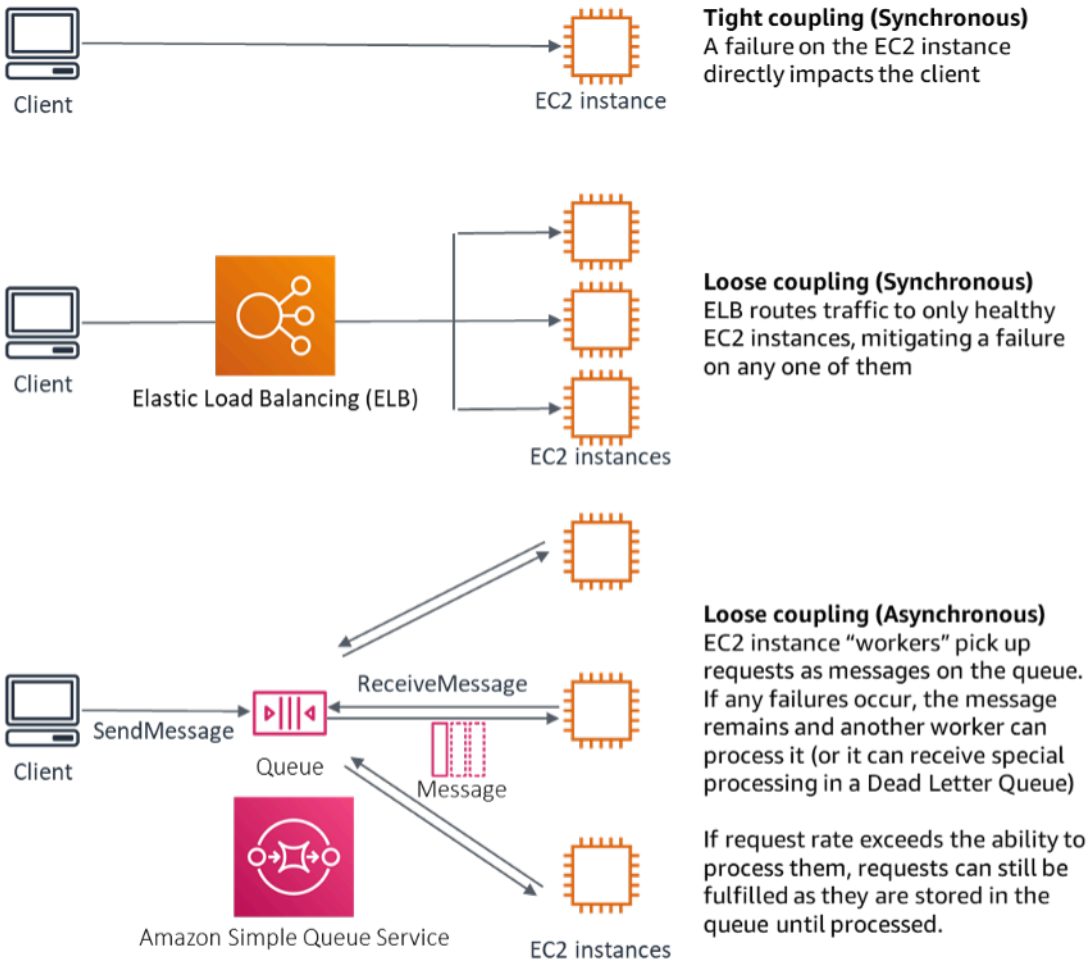


Figure 4: Dependencies such as queuing systems and load balancers are loosely coupled

Amazon SQS queues and Elastic Load Balancers are just two ways to add an intermediate layer for loose coupling. Event-driven architectures can also be built in the AWS Cloud using Amazon EventBridge, which can abstract clients (event producers) from the services they rely on (event consumers). Amazon Simple Notification Service (Amazon SNS) is an effective solution when you need high-throughput, push-based, many-to-many messaging. Using Amazon SNS topics, your publisher systems can fan out messages to a large number of subscriber endpoints for parallel processing.

While queues offer several advantages, in most hard real-time systems, requests older than a threshold time (often seconds) should be considered stale (the client has given up and is no longer waiting for a response), and not processed. This way more recent (and likely still valid requests) can be processed instead.

Common anti-patterns:

- Deploying a singleton as part of a workload.
- Directly invoking APIs between workload tiers with no capability of failover or asynchronous processing of the request.

Benefits of establishing this best practice: Loose coupling helps isolate behavior of a component from other components that depend on it, increasing resiliency and agility. Failure in one component is isolated from others.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Implement loosely coupled dependencies. Dependencies such as queuing systems, streaming systems, workflows, and load balancers are loosely coupled. Loose coupling helps isolate behavior of a component from other components that depend on it, increasing resiliency and agility.
  - AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
  - What Is Amazon EventBridge?
  - What Is Amazon Simple Queue Service?
    - Amazon EventBridge allows you to build event driven architectures, which are loosely coupled and distributed.
      - AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
    - If changes to one component force other components that rely on it to also change, then they are tightly coupled. Loose coupling breaks this dependency so that dependency components only need to know the versioned and published interface.
    - Make component interactions asynchronous where possible. This model is suitable for any interaction that does not need an immediate response and where an acknowledgement that a request has been registered will suffice.
      - AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda (API304)

Resources

Related documents:

- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
- Amazon EC2: Ensuring Idempotency
- The Amazon Builders' Library: Challenges with distributed systems
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
- What Is Amazon EventBridge?
- What Is Amazon Simple Queue Service?

Related videos:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda (API304)

REL04-BP03 Do constant work

Systems can fail when there are large, rapid changes in load. For example, if your workload is doing a health check that monitors the health of thousands of servers, it should send the same size payload (a full snapshot of the current state) each time. Whether no servers are failing, or all of them, the health check system is doing constant work with no large, rapid changes.

For example, if the health check system is monitoring 100,000 servers, the load on it is nominal under the normally light server failure rate. However, if a major event makes half of those servers unhealthy, then the health check system would be overwhelmed trying to update notification systems and communicate state to its clients. So instead the health check system should send the full snapshot of the current state each time. 100,000 server health states, each represented by a bit, would only be a 12.5-KB payload. Whether no servers are failing, or all of them are, the health check system is doing constant work, and large, rapid changes are not a threat to the system stability. This is actually how Amazon Route 53 handles health checks for endpoints (such as IP addresses) to determine how end users are routed to them.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Do constant work so that systems do not fail when there are large, rapid changes in load.
- Implement loosely coupled dependencies. Dependencies such as queuing systems, streaming systems, workflows, and load balancers are loosely coupled. Loose coupling helps isolate behavior of a component from other components that depend on it, increasing resiliency and agility.
  - The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
  - AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes constant work)
    - For the example of a health check system monitoring 100,000 servers, engineer workloads so that payload sizes remain constant regardless of number of successes or failures.

Resources

Related documents:

- Amazon EC2: Ensuring Idempotency
- The Amazon Builders' Library: Challenges with distributed systems
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee

Related videos:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes constant work)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL04-BP04 Make all responses idempotent

An idempotent service promises that each request is completed exactly once, such that making multiple identical requests has the same effect as making a single request. An idempotent service makes it easier for a client to implement retries without fear that a request will be erroneously

processed multiple times. To do this, clients can issue API requests with an idempotency token—the same token is used whenever the request is repeated. An idempotent service API uses the token to return a response identical to the response that was returned the first time that the request was completed.

In a distributed system, it's easy to perform an action at most once (client makes only one request), or at least once (keep requesting until client gets confirmation of success). But it's hard to guarantee an action is idempotent, which means it's performed *exactly* once, such that making multiple identical requests has the same effect as making a single request. Using idempotency tokens in APIs, services can receive a mutating request one or more times without creating duplicate records or side effects.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Make all responses idempotent. An idempotent service promises that each request is completed exactly once, such that making multiple identical requests has the same effect as making a single request.
  - Clients can issue API requests with an idempotency token—the same token is used whenever the request is repeated. An idempotent service API uses the token to return a response identical to the response that was returned the first time that the request was completed.
    - Amazon EC2: Ensuring Idempotency

Resources

Related documents:

- Amazon EC2: Ensuring Idempotency
- The Amazon Builders' Library: Challenges with distributed systems
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee

Related videos:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)

- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL 5. How do you design interactions in a distributed system to mitigate or withstand failures?

Distributed systems rely on communications networks to interconnect components (such as servers or services). Your workload must operate reliably despite data loss or latency over these networks. Components of the distributed system must operate in a way that does not negatively impact other components or the workload. These best practices permit workloads to withstand stresses or failures, more quickly recover from them, and mitigate the impact of such impairments. The result is improved mean time to recovery (MTTR).

Best practices

- REL05-BP01 Implement graceful degradation to transform applicable hard dependencies into soft dependencies
- REL05-BP02 Throttle requests
- REL05-BP03 Control and limit retry calls
- REL05-BP04 Fail fast and limit queues
- REL05-BP05 Set client timeouts
- REL05-BP06 Make services stateless where possible
- REL05-BP07 Implement emergency levers

REL05-BP01 Implement graceful degradation to transform applicable hard dependencies into soft dependencies

This best practice was updated with new guidance on July 13th, 2023.

Application components should continue to perform their core function even if dependencies become unavailable. They might be serving slightly stale data, alternate data, or even no data. This ensures overall system function is only minimally impeded by localized failures while delivering the central business value.

Desired outcome: When a component's dependencies are unhealthy, the component itself can still function, although in a degraded manner. Failure modes of components should be seen as normal

operation. Workflows should be designed in such a way that such failures do not lead to complete failure or at least to predictable and recoverable states.

Common anti-patterns:

- Not identifying the core business functionality needed. Not testing that components are functional even during dependency failures.
- Serving no data on errors or when only one out of multiple dependencies is unavailable and partial results can still be returned.
- Creating an inconsistent state when a transaction partially fails.
- Not having an alternative way to access a central parameter store.
- Invalidating or emptying local state as a result of a failed refresh without considering the consequences of doing so.

Benefits of establishing this best practice: Graceful degradation improves the availability of the system as a whole and maintains the functionality of the most important functions even during failures.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implementing graceful degradation helps minimize the impact of dependency failures on component function. Ideally, a component detects dependency failures and works around them in a way that minimally impacts other components or customers.

Architecting for graceful degradation means considering potential failure modes during dependency design. For each failure mode, have a way to deliver most or at least the most critical functionality of the component to callers or customers. These considerations can become additional requirements that can be tested and verified. Ideally, a component is able to perform its core function in an acceptable manner even when one or multiple dependencies fail.

This is as much a business discussion as a technical one. All business requirements are important and should be fulfilled if possible. However, it still makes sense to ask what should happen when not all of them can be fulfilled. A system can be designed to be available and consistent, but under circumstances where one requirement must be dropped, which one is more important? For payment processing, it might be consistency. For a real-time application, it might be availability. For a customer facing website, the answer may depend on customer expectations.

What this means depends on the requirements of the component and what should be considered its core function. For example:

- An ecommerce website might display data from multiple different systems like personalized recommendations, highest ranked products, and status of customer orders on the landing page. When one upstream system fails, it still makes sense to display everything else instead of showing an error page to a customer.
- A component performing batch writes can still continue processing a batch if one of the individual operations fails. It should be simple to implement a retry mechanism. This can be done by returning information on which operations succeeded, which failed, and why they failed to the caller, or putting failed requests into a dead letter queue to implement asynchronous retries. Information about failed operations should be logged as well.
- A system that processes transactions must verify that either all or no individual updates are executed. For distributed transactions, the saga pattern can be used to roll back previous operations in case a later operation of the same transaction fails. Here, the core function is maintaining consistency.
- Time critical systems should be able to deal with dependencies not responding in a timely manner. In these cases, the circuit breaker pattern can be used. When responses from a dependency start timing out, the system can switch to a closed state where no additional call are made.
- An application may read parameters from a parameter store. It can be useful to create container images with a default set of parameters and use these in case the parameter store is unavailable.

Note that the pathways taken in case of component failure need to be tested and should be significantly simpler than the primary pathway. Generally, fallback strategies should be avoided.

Implementation steps

Identify external and internal dependencies. Consider what kinds of failures can occur in them. Think about ways that minimize negative impact on upstream and downstream systems and customers during those failures.

The following is a list of dependencies and how to degrade gracefully when they fail:

1. **Partial failure of dependencies:** A component may make multiple requests to downstream systems, either as multiple requests to one system or one request to multiple systems each.

Depending on the business context, different ways of handling for this may be appropriate (for more detail, see previous examples in Implementation guidance).

2. **A downstream system is unable to process requests due to high load:** If requests to a downstream system are consistently failing, it does not make sense to continue retrying. This may create additional load on an already overloaded system and make recovery more difficult. The circuit breaker pattern can be utilized here, which monitors failing calls to a downstream system. If a high number of calls are failing, it will stop sending more requests to the downstream system and only occasionally let calls through to test whether the downstream system is available again.
3. **A parameter store is unavailable:** To transform a parameter store, soft dependency caching or sane defaults included in container or machine images may be used. Note that these defaults need to be kept up-to-date and included in test suites.
4. **A monitoring service or other non-functional dependncy is unavailable:** If a component is intermittently unable to send logs, metrics, or traces to a central monitoring service, it is often best to still execute business functions as usual. Silently not logging or pushing metrics for a long time is often not acceptable. Also, some use cases may require complete auditing entries to fulfill compliance requirements.
5. **A primary instance of a relational database may be unavailable:** Amazon Relational Database Service, like almost all relational databases, can only have one primary writer instance. This creates a single point of failure for write workloads and makes scaling more difficult. This can partially be mitigated by using a Multi-AZ configuration for high availability or Amazon Aurora Serverless for better scaling. For very high availability requirements, it can make sense to not rely on the primary writer at all. For queries that only read, read replicas can be used, which provide redundancy and the ability to scale out, not just up. Writes can be buffered, for example in an Amazon Simple Queue Service queue, so that write requests from customers can still be accepted even if the primary is temporarily unavailable.

Resources

Related documents:

- Amazon API Gateway: Throttle API Requests for Better Throughput
- CircuitBreaker (summarizes Circuit Breaker from “Release It!” book)
- Error Retries and Exponential Backoff in AWS
- Michael Nygard “Release It! Design and Deploy Production-Ready Software”

- The Amazon Builders' Library: Avoiding fallback in distributed systems
- The Amazon Builders' Library: Avoiding insurmountable queue backlogs
- The Amazon Builders' Library: Caching challenges and strategies
- The Amazon Builders' Library: Timeouts, retries, and backoff with jitter

Related videos:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

Related examples:

- Well-Architected lab: Level 300: Implementing Health Checks and Managing Dependencies to Improve Reliability

REL05-BP02 Throttle requests

> This best practice was updated with new guidance on July 13th, 2023.

Throttle requests to mitigate resource exhaustion due to unexpected increases in demand. Requests below throttling rates are processed while those over the defined limit are rejected with a return a message indicating the request was throttled.

Desired outcome: Large volume spikes either from sudden customer traffic increases, flooding attacks, or retry storms are mitigated by request throttling, allowing workloads to continue normal processing of supported request volume.

Common anti-patterns:

- API endpoint throttles are not implemented or are left at default values without considering expected volumes.
- API endpoints are not load tested or throttling limits are not tested.
- Throttling request rates without considering request size or complexity.
- Testing maximum request rates or maximum request size, but not testing both together.
- Resources are not provisioned to the same limits established in testing.

- Usage plans have not been configured or considered for application to application (A2A) API consumers.
- Queue consumers that horizontally scale do not have maximum concurrency settings configured.
- Rate limiting on a per IP address basis has not been implemented.

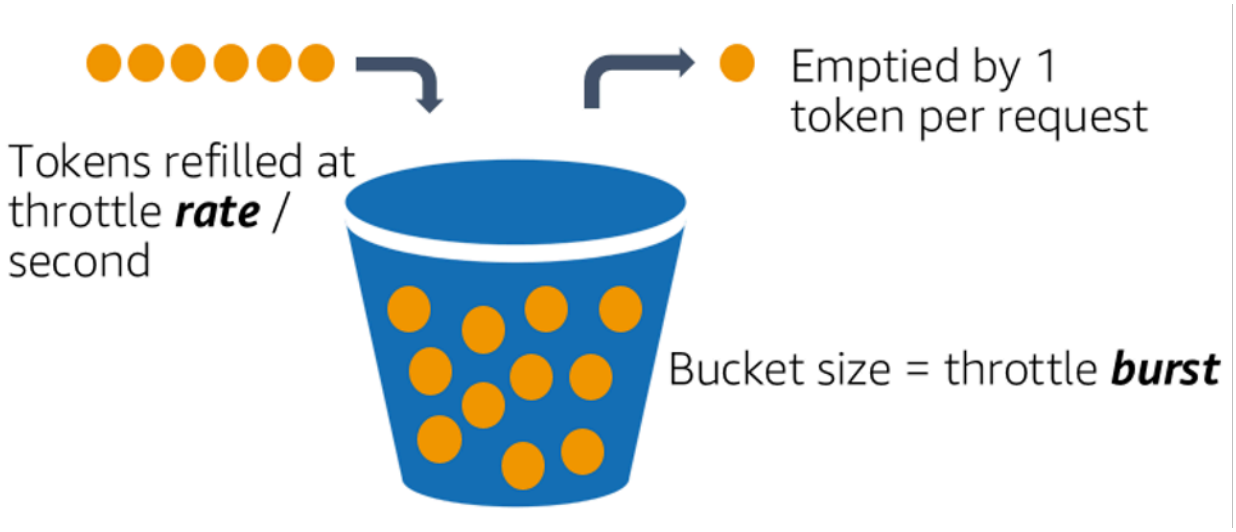
Benefits of establishing this best practice: Workloads that set throttle limits are able to operate normally and process accepted request load successfully under unexpected volume spikes. Sudden or sustained spikes of requests to APIs and queues are throttled and do not exhaust request processing resources. Rate limits throttle individual requestors so that high volumes of traffic from a single IP address or API consumer will not exhaust resources impact other consumers.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Services should be designed to process a known capacity of requests; this capacity can be established through load testing. If request arrival rates exceed limits, the appropriate response signals that a request has been throttled. This allows the consumer to handle the error and retry later.

When your service requires a throttling implementation, consider implementing the token bucket algorithm, where a token counts for a request. Tokens are refilled at a throttle rate per second and emptied asynchronously by one token per request.



The token bucket algorithm.

Amazon API Gateway implements the token bucket algorithm according to account and region limits and can be configured per-client with usage plans. Additionally, Amazon Simple Queue Service (Amazon SQS) and Amazon Kinesis can buffer requests to smooth out the request rate, and allow higher throttling rates for requests that can be addressed. Finally, you can implement rate limiting with AWS WAF to throttle specific API consumers that generate unusually high load.

Implementation steps

You can configure API Gateway with throttling limits for your APIs and return `429 Too Many Requests` errors when limits are exceeded. You can use AWS WAF with your AWS AppSync and API Gateway endpoints to enable rate limiting on a per IP address basis. Additionally, where your system can tolerate asynchronous processing, you can put messages into a queue or stream to speed up responses to service clients, which allows you to burst to higher throttle rates.

With asynchronous processing, when you've configured Amazon SQS as an event source for AWS Lambda, you can configure maximum concurrency to avoid high event rates from consuming available account concurrent execution quota needed for other services in your workload or account.

While API Gateway provides a managed implementation of the token bucket, in cases where you cannot use API Gateway, you can take advantage of language specific open-source implementations (see related examples in Resources) of the token bucket for your services.

- Understand and configure API Gateway throttling limits at the account level per region, API per stage, and API key per usage plan levels.
- Apply AWS WAF rate limiting rules to API Gateway and AWS AppSync endpoints to protect against floods and block malicious IPs. Rate limiting rules can also be configured on AWS AppSync API keys for A2A consumers.
- Consider whether you require more throttling control than rate limiting for AWS AppSync APIs, and if so, configure an API Gateway in front of your AWS AppSync endpoint.
- When Amazon SQS queues are set up as triggers for Lambda queue consumers, set maximum concurrency to a value that processes enough to meet your service level objectives but does not consume concurrency limits impacting other Lambda functions. Consider setting reserved concurrency on other Lambda functions in the same account and region when you consume queues with Lambda.
- Use API Gateway with native service integrations to Amazon SQS or Kinesis to buffer requests.

- If you cannot use API Gateway, look at language specific libraries to implement the token bucket algorithm for your workload. Check the examples section and do your own research to find a suitable library.
- Test limits that you plan to set, or that you plan to allow to be increased, and document the tested limits.
- Do not increase limits beyond what you establish in testing. When increasing a limit, verify that provisioned resources are already equivalent to or greater than those in test scenarios before applying the increase.

Resources

Related best practices:

- REL04-BP03 Do constant work
- REL05-BP03 Control and limit retry calls

Related documents:

- Amazon API Gateway: Throttle API Requests for Better Throughput
- AWS WAF: Rate-based rule statement
- Introducing maximum concurrency of AWS Lambda when using Amazon SQS as an event source
- AWS Lambda: Maximum Concurrency

Related examples:

- The three most important AWS WAF rate-based rules
- Java Bucket4j
- Python token-bucket
- Node token-bucket
- .NET System Threading Rate Limiting

Related videos:

- Implementing GraphQL API security best practices with AWS AppSync

Related tools:

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF

REL05-BP03 Control and limit retry calls

This best practice was updated with new guidance on July 13th, 2023.

Use exponential backoff to retry requests at progressively longer intervals between each retry. Introduce jitter between retries to randomize retry intervals. Limit the maximum number of retries.

Desired outcome: Typical components in a distributed software system include servers, load balancers, databases, and DNS servers. During normal operation, these components can respond to requests with errors that are temporary or limited, and also errors that would be persistent regardless of retries. When clients make requests to services, the requests consume resources including memory, threads, connections, ports, or any other limited resources. Controlling and limiting retries is a strategy to release and minimize consumption of resources so that system components under strain are not overwhelmed.

When client requests time out or receive error responses, they should determine whether or not to retry. If they do retry, they do so with exponential backoff with jitter and a maximum retry value. As a result, backend services and processes are given relief from load and time to self-heal, resulting in faster recovery and successful request servicing.

Common anti-patterns:

- Implementing retries without adding exponential backoff, jitter, and maximum retry values. Backoff and jitter help avoid artificial traffic spikes due to unintentionally coordinated retries at common intervals.
- Implementing retries without testing their effects or assuming retries are already built into an SDK without testing retry scenarios.

- Failing to understand published error codes from dependencies, leading to retrying all errors, including those with a clear cause that indicates lack of permission, configuration error, or another condition that predictably will not resolve without manual intervention.
- Not addressing observability practices, including monitoring and alerting on repeated service failures so that underlying issues are made known and can be addressed.
- Developing custom retry mechanisms when built-in or third-party retry capabilities suffice.
- Retrying at multiple layers of your application stack in a manner which compounds retry attempts further consuming resources in a retry storm. Be sure to understand how these errors affect your application the dependencies you rely on, then implement retries at only one level.
- Retrying service calls that are not idempotent, causing unexpected side effects like duplicated results.

Benefits of establishing this best practice: Retries help clients acquire desired results when requests fail but also consume more of a server's time to get the successful responses they want. When failures are rare or transient, retries work well. When failures are caused by resource overload, retries can make things worse. Adding exponential backoff with jitter to client retries allows servers to recover when failures are caused by resource overload. Jitter avoids alignment of requests into spikes, and backoff diminishes load escalation caused by adding retries to normal request load. Finally, it's important to configure a maximum number of retries or elapsed time to avoid creating backlogs that produce metastable failures.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Control and limit retry calls. Use exponential backoff to retry after progressively longer intervals. Introduce jitter to randomize retry intervals and limit the maximum number of retries.

Some AWS SDKs implement retries and exponential backoff by default. Use these built-in AWS implementations where applicable in your workload. Implement similar logic in your workload when calling services that are idempotent and where retries improve your client availability. Decide what the timeouts are and when to stop retrying based on your use case. Build and exercise testing scenarios for those retry use cases.

Implementation steps

- Determine the optimal layer in your application stack to implement retries for the services your application relies on.

- Be aware of existing SDKs that implement proven retry strategies with exponential backoff and jitter for your language of choice, and favor these over writing your own retry implementations.
- Verify that services are idempotent before implementing retries. Once retries are implemented, be sure they are both tested and regularly exercise in production.
- When calling AWS service APIs, use the AWS SDKs and AWS CLI and understand the retry configuration options. Determine if the defaults work for your use case, test, and adjust as needed.

Resources

Related best practices:

- REL04-BP04 Make all responses idempotent
- REL05-BP02 Throttle requests
- REL05-BP04 Fail fast and limit queues
- REL05-BP05 Set client timeouts
- REL11-BP01 Monitor all components of the workload to detect failures

Related documents:

- Error Retries and Exponential Backoff in AWS
- The Amazon Builders' Library: Timeouts, retries, and backoff with jitter
- Exponential Backoff and Jitter
- Making retries safe with idempotent APIs

Related examples:

- Spring Retry
- Resilience4j Retry

Related videos:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

Related tools:

- AWS SDKs and Tools: Retry behavior
- AWS Command Line Interface: AWS CLI retries

REL05-BP04 Fail fast and limit queues

This best practice was updated with new guidance on July 13th, 2023.

When a service is unable to respond successfully to a request, fail fast. This allows resources associated with a request to be released, and permits a service to recover if it's running out of resources. Failing fast is a well-established software design pattern that can be leveraged to build highly reliable workloads in the cloud. Queuing is also a well-established enterprise integration pattern that can smooth load and allow clients to release resources when asynchronous processing can be tolerated. When a service is able to respond successfully under normal conditions but fails when the rate of requests is too high, use a queue to buffer requests. However, do not allow a buildup of long queue backlogs that can result in processing stale requests that a client has already given up on.

Desired outcome: When systems experience resource contention, timeouts, exceptions, or grey failures that make service level objectives unachievable, fail fast strategies allow for faster system recovery. Systems that must absorb traffic spikes and can accommodate asynchronous processing can improve reliability by allowing clients to quickly release requests by using queues to buffer requests to backend services. When buffering requests to queues, queue management strategies are implemented to avoid insurmountable backlogs.

Common anti-patterns:

- Implementing message queues but not configuring dead letter queues (DLQ) or alarms on DLQ volumes to detect when a system is in failure.
- Not measuring the age of messages in a queue, a measurement of latency to understand when queue consumers are falling behind or erroring out causing retrying.
- Not clearing backlogged messages from a queue, when there is no value in processing these messages if the business need no longer exists.

- Configuring first in first out (FIFO) queues when last in first out (LIFO) queues would better serve client needs, for example when strict ordering is not required and backlog processing is delaying all new and time sensitive requests resulting in all clients experiencing breached service levels.
- Exposing internal queues to clients instead of exposing APIs that manage work intake and place requests into internal queues.
- Combining too many work request types into a single queue which can exacerbate backlog conditions by spreading resource demand across request types.
- Processing complex and simple requests in the same queue, despite needing different monitoring, timeouts and resource allocations.
- Not validating inputs or using assertions to implement fail fast mechanisms in software that bubble up exceptions to higher level components that can handle errors gracefully.
- Not removing faulty resources from request routing, especially when failures are grey emitting both successes and failures due to crashing and restarting, intermittent dependency failure, reduced capacity, or network packet loss.

Benefits of establishing this best practice: Systems that fail fast are easier to debug and fix, and often expose issues in coding and configuration before releases are published into production. Systems that incorporate effective queueing strategies provide greater resilience and reliability to traffic spikes and intermittent system fault conditions.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Fail fast strategies can be coded into software solutions as well as configured into infrastructure. In addition to failing fast, queues are a straightforward yet powerful architectural technique to decouple system components smooth load. Amazon CloudWatch provides capabilities to monitor for and alarm on failures. Once a system is known to be failing, mitigation strategies can be invoked, including failing away from impaired resources. When systems implement queues with Amazon SQS and other queue technologies to smooth load, they must consider how to manage queue backlogs, as well as message consumption failures.

Implementation steps

- Implement programmatic assertions or specific metrics in your software and use them to explicitly alert on system issues. Amazon CloudWatch helps you create metrics and alarms based on application log pattern and SDK instrumentation.

- Use CloudWatch metrics and alarms to fail away from impaired resources that are adding latency to processing or repeatedly failing to process requests.
- Use asynchronous processing by designing APIs to accept requests and append requests to internal queues using Amazon SQS and then respond to the message-producing client with a success message so the client can release resources and move on with other work while backend queue consumers process requests.
- Measure and monitor for queue processing latency by producing a CloudWatch metric each time you take a message off a queue by comparing now to message timestamp.
- When failures prevent successful message processing or traffic spikes in volumes that cannot be processed within service level agreements, sideline older or excess traffic to a spillover queue. This allows priority processing of new work, and older work when capacity is available. This technique is an approximation of LIFO processing and allows normal system processing for all new work.
- Use dead letter or redrive queues to move messages that can't be processed out of the backlog into a location that can be researched and resolved later
- Either retry or, when tolerable, drop old messages by comparing now to the message timestamp and discarding messages that are no longer relevant to the requesting client.

Resources

Related best practices:

- REL04-BP02 Implement loosely coupled dependencies
- REL05-BP02 Throttle requests
- REL05-BP03 Control and limit retry calls
- REL06-BP02 Define and calculate metrics (Aggregation)
- REL06-BP07 Monitor end-to-end tracing of requests through your system

Related documents:

- Avoiding insurmountable queue backlogs
- Fail Fast
- How can I prevent an increasing backlog of messages in my Amazon SQS queue?
- Elastic Load Balancing: Zonal Shift

- Amazon Route 53 Application Recovery Controller: Routing control for traffic failover

Related examples:

- Enterprise Integration Patterns: Dead Letter Channel

Related videos:

- AWS re:Invent 2022 - Operating highly available Multi-AZ applications

Related tools:

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- Amazon CloudWatch

REL05-BP05 Set client timeouts

> This best practice was updated with new guidance on July 13th, 2023.

Set timeouts appropriately on connections and requests, verify them systematically, and do not rely on default values as they are not aware of workload specifics.

Desired outcome: Client timeouts should consider the cost to the client, server, and workload associated with waiting for requests that take abnormal amounts of time to complete. Since it is not possible to know the exact cause of any timeout, clients must use knowledge of services to develop expectations of probable causes and appropriate timeouts

Client connections time out based on configured values. After encountering a timeout, clients make decisions to back off and retry or open a circuit breaker. These patterns avoid issuing requests that may exacerbate an underlying error condition.

Common anti-patterns:

- Not being aware of system timeouts or default timeouts.

- Not being aware of normal request completion timing.
- Not being aware of possible causes for requests to take abnormally long to complete, or the costs to client, service, or workload performance associated with waiting on these completions.
- Not being aware of the probability of impaired network causing a request to fail only once timeout is reached, and the costs to client and workload performance for not adopting a shorter timeout.
- Not testing timeout scenarios both for connections and requests.
- Setting timeouts too high, which can result in long wait times and increase resource utilization.
- Setting timeouts too low, resulting in artificial failures.
- Overlooking patterns to deal with timeout errors for remote calls like circuit breakers and retries.
- Not considering monitoring for service call error rates, service level objectives for latency, and latency outliers. These metrics can provide insight to aggressive or permissive timeouts

Benefits of establishing this best practice: Remote call timeouts are configured and systems are designed to handle timeouts gracefully so that resources are conserved when remote calls respond abnormally slow and timeout errors are handled gracefully by service clients.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Set both a connection timeout and a request timeout on any service dependency call and generally on any call across processes. Many frameworks offer built-in timeout capabilities, but be careful, as some have default values that are infinite or higher than acceptable for your service goals. A value that is too high reduces the usefulness of the timeout because resources continue to be consumed while the client waits for the timeout to occur. A value that is too low can generate increased traffic on the backend and increased latency because too many requests are retried. In some cases, this can lead to complete outages because all requests are being retried.

Consider the following when determining timeout strategies:

- Requests may take longer than normal to process because of their content, impairments in a target service, or a networking partition failure.
- Requests with abnormally expensive content could consume unnecessary server and client resources. In this case, timing out these requests and not retrying can preserve resources. Services should also protect themselves from abnormally expensive content with throttles and server-side timeouts.

- Requests that take abnormally long due to a service impairment can be timed out and retried. Consideration should be given to service costs for the request and retry, but if the cause is a localized impairment, a retry is not likely to be expensive and will reduce client resource consumption. The timeout may also release server resources depending on the nature of the impairment.
- Requests that take a long time to complete because the request or response has failed to be delivered by the network can be timed out and retried. Because the request or response was not delivered, failure would have been the outcome regardless of the length of timeout. Timing out in this case will not release server resources, but it will release client resources and improve workload performance.

Take advantage of well-established design patterns like retries and circuit breakers to handle timeouts gracefully and support fail-fast approaches. AWS SDKs and AWS CLI allow for configuration of both connection and request timeouts and for retries with exponential backoff and jitter. AWS Lambda functions support configuration of timeouts, and with AWS Step Functions, you can build low code circuit breakers that take advantage of pre-built integrations with AWS services and SDKs. AWS App Mesh Envoy provides timeout and circuit breaker capabilities.

Implementation steps

- Configure timeouts on remote service calls and take advantage of built-in language timeout features or open source timeout libraries.
- When your workload makes calls with an AWS SDK, review the documentation for language specific timeout configuration.
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go
  - Node.js
  - C++
- When using AWS SDKs or AWS CLI commands in your workload, configure default timeout values by setting the AWS configuration defaults for `connectTimeoutInMillis` and `tlsNegotiationTimeoutInMillis`.

- Apply command line options `cli-connect-timeout` and `cli-read-timeout` to control one-off AWS CLI commands to AWS services.
- Monitor remote service calls for timeouts, and set alarms on persistent errors so that you can proactively handle error scenarios.
- Implement CloudWatch Metrics and CloudWatch anomaly detection on call error rates, service level objectives for latency, and latency outliers to provide insight into managing overly aggressive or permissive timeouts.
- Configure timeouts on Lambda functions.
- API Gateway clients must implement their own retries when handling timeouts. API Gateway supports a 50 millisecond to 29 second integration timeout for downstream integrations and does not retry when integration requests timeout.
- Implement the circuit breaker pattern to avoid making remote calls when they are timing out. Open the circuit to avoid failing calls and close the circuit when calls are responding normally.
- For container based workloads, review App Mesh Envoy features to leverage built in timeouts and circuit breakers.
- Use AWS Step Functions to build low code circuit breakers for remote service calls, especially where calling AWS native SDKs and supported Step Functions integrations to simplify your workload.

Resources

Related best practices:

- REL05-BP03 Control and limit retry calls
- REL05-BP04 Fail fast and limit queues
- REL06-BP07 Monitor end-to-end tracing of requests through your system

Related documents:

- AWS SDK: Retries and Timeouts
- The Amazon Builders' Library: Timeouts, retries, and backoff with jitter
- Amazon API Gateway quotas and important notes
- AWS Command Line Interface: Command line options
- AWS SDK for Java 2.x: Configure API Timeouts

- AWS Botocore using the config object and Config Reference
- AWS SDK for .NET: Retries and Timeouts
- AWS Lambda: Configuring Lambda function options

Related examples:

- Using the circuit breaker pattern with AWS Step Functions and Amazon DynamoDB
- Martin Fowler: CircuitBreaker

Related tools:

- AWS SDKs
- AWS Lambda
- Amazon SQS
- AWS Step Functions
- AWS Command Line Interface

REL05-BP06 Make services stateless where possible

Services should either not require state, or should offload state such that between different client requests, there is no dependence on locally stored data on disk and in memory. This allows servers to be replaced at will without causing an availability impact. Amazon ElastiCache or Amazon DynamoDB are good destinations for offloaded state.

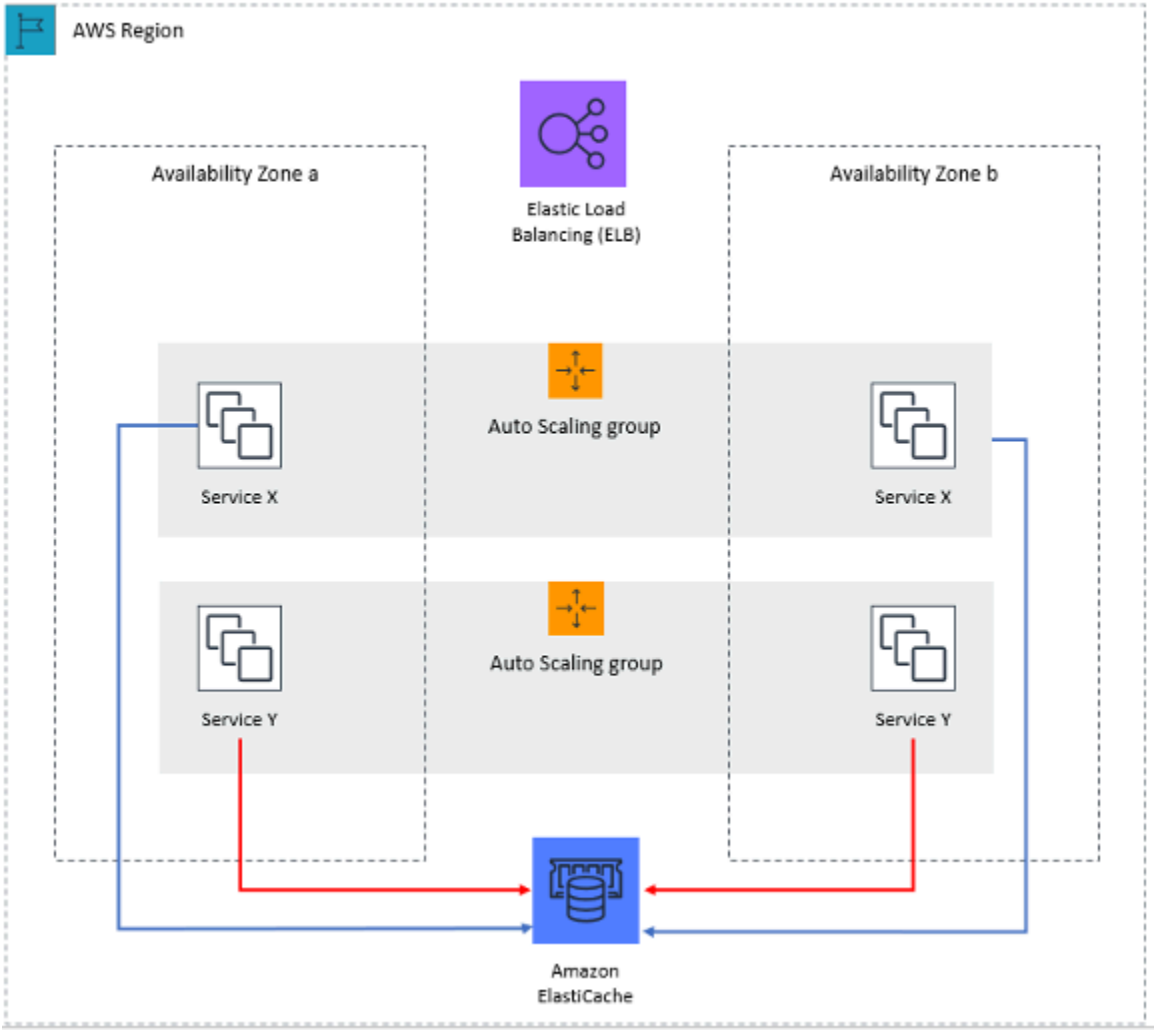


Figure 7: In this stateless web application, session state is offloaded to Amazon ElastiCache.

When users or services interact with an application, they often perform a series of interactions that form a session. A session is unique data for users that persists between requests while they use the application. A stateless application is an application that does not need knowledge of previous interactions and does not store session information.

Once designed to be stateless, you can then use serverless compute services, such as AWS Lambda or AWS Fargate.

In addition to server replacement, another benefit of stateless applications is that they can scale horizontally because any of the available compute resources (such as EC2 instances and AWS Lambda functions) can service any request.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Make your applications stateless. Stateless applications allow horizontal scaling and are tolerant to the failure of an individual node.
  - Remove state that could actually be stored in request parameters.
  - After examining whether the state is required, move any state tracking to a resilient multi-zone cache or data store like Amazon ElastiCache, Amazon RDS, Amazon DynamoDB, or a third-party distributed data solution. Store a state that could not be moved to resilient data stores.
    - Some data (like cookies) can be passed in headers or query parameters.
    - Refactor to remove state that can be quickly passed in requests.
    - Some data may not actually be needed per request and can be retrieved on demand.
    - Remove data that can be asynchronously retrieved.
    - Decide on a data store that meets the requirements for a required state.
    - Consider a NoSQL database for non-relational data.

Resources

Related documents:

- The Amazon Builders' Library: Avoiding fallback in distributed systems
- The Amazon Builders' Library: Avoiding insurmountable queue backlogs
- The Amazon Builders' Library: Caching challenges and strategies

REL05-BP07 Implement emergency levers

Emergency levers are rapid processes that can mitigate availability impact on your workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Implement emergency levers. These are rapid processes that may mitigate availability impact on your workload. They can be operated in the absence of a root cause. An ideal emergency lever reduces the cognitive burden on the resolvers to zero by providing fully deterministic activation and deactivation criteria. Levers are often manual, but they can also be automated

- Example levers include
  - Block all robot traffic
  - Serve static pages instead of dynamic ones
  - Reduce frequency of calls to a dependency
  - Throttle calls from dependencies
- Tips for implementing and using emergency levers
  - When levers are activated, do LESS, not more
  - Keep it simple, avoid bimodal behavior
  - Test your levers periodically
- These are examples of actions that are NOT emergency levers
  - Add capacity
  - Call up service owners of clients that depend on your service and ask them to reduce calls
  - Making a change to code and releasing it

Change management

Questions

- REL 6. How do you monitor workload resources?
- REL 7. How do you design your workload to adapt to changes in demand?
- REL 8. How do you implement change?

REL 6. How do you monitor workload resources?

Logs and metrics are powerful tools to gain insight into the health of your workload. You can configure your workload to monitor logs and metrics and send notifications when thresholds are crossed or significant events occur. Monitoring allows your workload to recognize when low-performance thresholds are crossed or failures occur, so it can recover automatically in response.

Best practices

- REL06-BP01 Monitor all components for the workload (Generation)
- REL06-BP02 Define and calculate metrics (Aggregation)
- REL06-BP03 Send notifications (Real-time processing and alarming)

- REL06-BP04 Automate responses (Real-time processing and alarming)
- REL06-BP05 Analytics
- REL06-BP06 Conduct reviews regularly
- REL06-BP07 Monitor end-to-end tracing of requests through your system

REL06-BP01 Monitor all components for the workload (Generation)

Monitor the components of the workload with Amazon CloudWatch or third-party tools. Monitor AWS services with AWS Health Dashboard.

All components of your workload should be monitored, including the front-end, business logic, and storage tiers. Define key metrics, describe how to extract them from logs (if necessary), and set thresholds for invoking corresponding alarm events. Ensure metrics are relevant to the key performance indicators (KPIs) of your workload, and use metrics and logs to identify early warning signs of service degradation. For example, a metric related to business outcomes such as the number of orders successfully processed per minute, can indicate workload issues faster than technical metric, such as CPU Utilization. Use AWS Health Dashboard for a personalized view into the performance and availability of the AWS services underlying your AWS resources.

Monitoring in the cloud offers new opportunities. Most cloud providers have developed customizable hooks and can deliver insights to help you monitor multiple layers of your workload. AWS services such as Amazon CloudWatch apply statistical and machine learning algorithms to continually analyze metrics of systems and applications, determine normal baselines, and surface anomalies with minimal user intervention. Anomaly detection algorithms account for the seasonality and trend changes of metrics.

AWS makes an abundance of monitoring and log information available for consumption that can be used to define workload-specific metrics, change-in-demand processes, and adopt machine learning techniques regardless of ML expertise.

In addition, monitor all of your external endpoints to ensure that they are independent of your base implementation. This active monitoring can be done with synthetic transactions (sometimes referred to as *user canaries*, but not to be confused with canary deployments) which periodically run a number of common tasks matching actions performed by clients of the workload. Keep these tasks short in duration and be sure not to overload your workload during testing. Amazon CloudWatch Synthetics allows you to create synthetic canaries to monitor your endpoints and APIs. You can also combine the synthetic canary client nodes with AWS X-Ray console to pinpoint which

synthetic canaries are experiencing issues with errors, faults, or throttling rates for the selected time frame.

Desired Outcome:

Collect and use critical metrics from all components of the workload to ensure workload reliability and optimal user experience. Detecting that a workload is not achieving business outcomes allows you to quickly declare a disaster and recover from an incident.

Common anti-patterns:

- Only monitoring external interfaces to your workload.
- Not generating any workload-specific metrics and only relying on metrics provided to you by the AWS services your workload uses.
- Only using technical metrics in your workload and not monitoring any metrics related to non-technical KPIs the workload contributes to.
- Relying on production traffic and simple health checks to monitor and evaluate workload state.

Benefits of establishing this best practice: Monitoring at all tiers in your workload allows you to more rapidly anticipate and resolve problems in the components that comprise the workload.

Level of risk exposed if this best practice is not established: High

Implementation guidance

1. **Turn on logging where available.** Monitoring data should be obtained from all components of the workloads. Turn on additional logging, such as S3 Access Logs, and permit your workload to log workload specific data. Collect metrics for CPU, network I/O, and disk I/O averages from services such as Amazon ECS, Amazon EKS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling, and Amazon EMR. See AWS Services That Publish CloudWatch Metrics for a list of AWS services that publish metrics to CloudWatch.
2. **Review all default metrics and explore any data collection gaps.** Every service generates default metrics. Collecting default metrics allows you to better understand the dependencies between workload components, and how component reliability and performance affect the workload. You can also create and publish your own metrics to CloudWatch using the AWS CLI or an API.
3. **Evaluate all the metrics to decide which ones to alert on for each AWS service in your workload.** You may choose to select a subset of metrics that have a major impact on workload

reliability. Focusing on critical metrics and threshold allows you to refine the number of alerts and can help minimize false-positives.

4. **Define alerts and the recovery process for your workload after the alert is invoked.** Defining alerts allows you to quickly notify, escalate, and follow steps necessary to recover from an incident and meet your prescribed Recovery Time Objective (RTO). You can use Amazon CloudWatch Alarms to invoke automated workflows and initiate recovery procedures based on defined thresholds.
5. **Explore use of synthetic transactions to collect relevant data about workloads state.** Synthetic monitoring follows the same routes and perform the same actions as a customer, which makes it possible for you to continually verify your customer experience even when you don't have any customer traffic on your workloads. By using synthetic transactions, you can discover issues before your customers do.

Resources

Related best practices:

- REL11-BP03 Automate healing on all layers

Related documents:

- Getting started with your AWS Health Dashboard – Your account health
- AWS Services That Publish CloudWatch Metrics
- Access Logs for Your Network Load Balancer
- Access logs for your application load balancer
- Accessing Amazon CloudWatch Logs for AWS Lambda
- Amazon S3 Server Access Logging
- Enable Access Logs for Your Classic Load Balancer
- Exporting log data to Amazon S3
- Install the CloudWatch agent on an Amazon EC2 instance
- Publishing Custom Metrics
- Using Amazon CloudWatch Dashboards
- Using Amazon CloudWatch Metrics

- Using Canaries (Amazon CloudWatch Synthetics)
- What are Amazon CloudWatch Logs?

User guides:

- Creating a trail
- Monitoring memory and disk metrics for Amazon EC2 Linux instances
- Using CloudWatch Logs with container instances
- VPC Flow Logs
- What is Amazon DevOps Guru?
- What is AWS X-Ray?

Related blogs:

- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray

Related examples and workshops:

- AWS Well-Architected Labs: Operational Excellence - Dependency Monitoring
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility
- Observability workshop

REL06-BP02 Define and calculate metrics (Aggregation)

Store log data and apply filters where necessary to calculate metrics, such as counts of a specific log event, or latency calculated from log event timestamps.

Amazon CloudWatch and Amazon S3 serve as the primary aggregation and storage layers. For some services, such as AWS Auto Scaling and Elastic Load Balancing, default metrics are provided by default for CPU load or average request latency across a cluster or instance. For streaming services, such as VPC Flow Logs and AWS CloudTrail, event data is forwarded to CloudWatch Logs and you need to define and apply metrics filters to extract metrics from the event data. This gives you time series data, which can serve as inputs to CloudWatch alarms that you define to invoke alerts.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Define and calculate metrics (Aggregation). Store log data and apply filters where necessary to calculate metrics, such as counts of a specific log event, or latency calculated from log event timestamps
  - Metric filters define the terms and patterns to look for in log data as it is sent to CloudWatch Logs. CloudWatch Logs uses these metric filters to turn log data into numerical CloudWatch metrics that you can graph or set an alarm on.
    - Searching and Filtering Log Data
  - Use a trusted third party to aggregate logs.
    - Follow the instructions of the third party. Most third-party products integrate with CloudWatch and Amazon S3.
  - Some AWS services can publish logs directly to Amazon S3. If your main requirement for logs is storage in Amazon S3, you can easily have the service producing the logs send them directly to Amazon S3 without setting up additional infrastructure.
    - Sending Logs Directly to Amazon S3

Resources

Related documents:

- Amazon CloudWatch Logs Insights Sample Queries
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- One Observability Workshop
- Searching and Filtering Log Data
- Sending Logs Directly to Amazon S3
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility

REL06-BP03 Send notifications (Real-time processing and alarming)

Organizations that need to know, receive notifications when significant events occur.

Alerts can be sent to Amazon Simple Notification Service (Amazon SNS) topics, and then pushed to any number of subscribers. For example, Amazon SNS can forward alerts to an email alias so that technical staff can respond.

Common anti-patterns:

- Configuring alarms at too low of threshold, causing too many notifications to be sent.
- Not archiving alarms for future exploration.

Benefits of establishing this best practice: Notifications on events (even those that can be responded to and automatically resolved) allow you to have a record of events and potentially address them in a different manner in the future.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Perform real-time processing and alarming. Organizations that need to know, receive notifications when significant events occur
  - Amazon CloudWatch dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those resources that are spread across different Regions.
    - Using Amazon CloudWatch Dashboards
  - Create an alarm when the metric surpasses a limit.
    - Using Amazon CloudWatch Alarms

Resources

Related documents:

- One Observability Workshop
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility
- Using Amazon CloudWatch Alarms
- Using Amazon CloudWatch Dashboards
- Using Amazon CloudWatch Metrics

REL06-BP04 Automate responses (Real-time processing and alarming)

Use automation to take action when an event is detected, for example, to replace failed components.

Alerts can invoke AWS Auto Scaling events, so that clusters react to changes in demand. Alerts can be sent to Amazon Simple Queue Service (Amazon SQS), which can serve as an integration point for third-party ticket systems. AWS Lambda can also subscribe to alerts, providing users an asynchronous serverless model that reacts to change dynamically. AWS Config continually monitors and records your AWS resource configurations, and can invoke AWS Systems Manager Automation to remediate issues.

Amazon DevOps Guru can automatically monitor application resources for anomalous behavior and deliver targeted recommendations to speed up problem identification and remediation times.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Use Amazon DevOps Guru to perform automated actions. Amazon DevOps Guru can automatically monitor application resources for anomalous behavior and deliver targeted recommendations to speed up problem identification and remediation times.
  - What is Amazon DevOps Guru?
- Use AWS Systems Manager to perform automated actions. AWS Config continually monitors and records your AWS resource configurations, and can invoke AWS Systems Manager Automation to remediate issues.
  - AWS Systems Manager Automation
    - Create and use Systems Manager Automation documents. These define the actions that Systems Manager performs on your managed instances and other AWS resources when an automation process runs.
    - Working with Automation Documents (Playbooks)
- Amazon CloudWatch sends alarm state change events to Amazon EventBridge. Create EventBridge rules to automate responses.
  - Creating an EventBridge Rule That Triggers on an Event from an AWS Resource
- Create and run a plan to automate responses.
  - Inventory all your alert response procedures. You must plan your alert responses before you rank the tasks.
  - Inventory all the tasks with specific actions that must be taken. Most of these actions are documented in runbooks. You must also have playbooks for alerts of unexpected events.
  - Examine the runbooks and playbooks for all automatable actions. In general, if an action can be defined, it most likely can be automated.

- Rank the error-prone or time-consuming activities first. It is most beneficial to remove sources of errors and reduce time to resolution.
- Establish a plan to complete automation. Maintain an active plan to automate and update the automation.
- Examine manual requirements for opportunities for automation. Challenge your manual process for opportunities to automate.

Resources

Related documents:

- AWS Systems Manager Automation
- Creating an EventBridge Rule That Triggers on an Event from an AWS Resource
- One Observability Workshop
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility
- What is Amazon DevOps Guru?
- Working with Automation Documents (Playbooks)

REL06-BP05 Analytics

Collect log files and metrics histories and analyze these for broader trends and workload insights.

Amazon CloudWatch Logs Insights supports a simple yet powerful query language that you can use to analyze log data. Amazon CloudWatch Logs also supports subscriptions that allow data to flow seamlessly to Amazon S3 where you can use or Amazon Athena to query the data. It also supports queries on a large array of formats. See Supported SerDes and Data Formats in the Amazon Athena User Guide for more information. For analysis of huge log file sets, you can run an Amazon EMR cluster to run petabyte-scale analyses.

There are a number of tools provided by AWS Partners and third parties that allow for aggregation, processing, storage, and analytics. These tools include New Relic, Splunk, Loggly, Logstash, CloudHealth, and Nagios. However, outside generation of system and application logs is unique to each cloud provider, and often unique to each service.

An often-overlooked part of the monitoring process is data management. You need to determine the retention requirements for monitoring data, and then apply lifecycle policies accordingly.

Amazon S3 supports lifecycle management at the S3 bucket level. This lifecycle management can be applied differently to different paths in the bucket. Toward the end of the lifecycle, you can transition data to Amazon Glacier for long-term storage, and then expiration after the end of the retention period is reached. The S3 Intelligent-Tiering storage class is designed to optimize costs by automatically moving data to the most cost-effective access tier, without performance impact or operational overhead.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- CloudWatch Logs Insights allows you to interactively search and analyze your log data in Amazon CloudWatch Logs.
  - Analyzing Log Data with CloudWatch Logs Insights
  - Amazon CloudWatch Logs Insights Sample Queries
- Use Amazon CloudWatch Logs send logs to Amazon S3 where you can use or Amazon Athena to query the data.
  - How do I analyze my Amazon S3 server access logs using Athena?
    - Create an S3 lifecycle policy for your server access logs bucket. Configure the lifecycle policy to periodically remove log files. Doing so reduces the amount of data that Athena analyzes for each query.
      - How Do I Create a Lifecycle Policy for an S3 Bucket?

Resources

Related documents:

- Amazon CloudWatch Logs Insights Sample Queries
- Analyzing Log Data with CloudWatch Logs Insights
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- How Do I Create a Lifecycle Policy for an S3 Bucket?
- How do I analyze my Amazon S3 server access logs using Athena?
- One Observability Workshop
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility

REL06-BP06 Conduct reviews regularly

Frequently review how workload monitoring is implemented and update it based on significant events and changes.

Effective monitoring is driven by key business metrics. Ensure these metrics are accommodated in your workload as business priorities change.

Auditing your monitoring helps ensure that you know when an application is meeting its availability goals. Root cause analysis requires the ability to discover what happened when failures occur. AWS provides services that allow you to track the state of your services during an incident:

- **Amazon CloudWatch Logs:** You can store your logs in this service and inspect their contents.
- **Amazon CloudWatch Logs Insights**: Is a fully managed service that allows you to analyze massive logs in seconds. It gives you fast, interactive queries and visualizations.
- **AWS Config:** You can see what AWS infrastructure was in use at different points in time.
- **AWS CloudTrail:** You can see which AWS APIs were invoked at what time and by what principal.

At AWS, we conduct a weekly meeting to review operational performance and to share learnings between teams. Because there are so many teams in AWS, we created The Wheel to randomly pick a workload to review. Establishing a regular cadence for operational performance reviews and knowledge sharing enhances your ability to achieve higher performance from your operational teams.

Common anti-patterns:

- Collecting only default metrics.
- Setting a monitoring strategy and never reviewing it.
- Not discussing monitoring when major changes are deployed.

Benefits of establishing this best practice: Regularly reviewing your monitoring allows for the anticipation of potential problems, instead of reacting to notifications when an anticipated problem actually occurs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Create multiple dashboards for the workload. You must have a top-level dashboard that contains the key business metrics, as well as the technical metrics you have identified to be the most relevant to the projected health of the workload as usage varies. You should also have dashboards for various application tiers and dependencies that can be inspected.
  - Using Amazon CloudWatch Dashboards
- Schedule and conduct regular reviews of the workload dashboards. Conduct regular inspection of the dashboards. You may have different cadences for the depth at which you inspect.
  - Inspect for trends in the metrics. Compare the metric values to historic values to see if there are trends that may indicate that something that needs investigation. Examples of this include: increasing latency, decreasing primary business function, and increasing failure responses.
  - Inspect for outliers/anomalies in your metrics. Averages or medians can mask outliers and anomalies. Look at the highest and lowest values during the time frame and investigate the causes of extreme scores. As you continue to eliminate these causes, lowering your definition of extreme allows you to continue to improve the consistency of your workload performance.
  - Look for sharp changes in behavior. An immediate change in quantity or direction of a metric may indicate that there has been a change in the application, or external factors that you may need to add additional metrics to track.

Resources

Related documents:

- Amazon CloudWatch Logs Insights Sample Queries
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- One Observability Workshop
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility
- Using Amazon CloudWatch Dashboards

REL06-BP07 Monitor end-to-end tracing of requests through your system

This best practice was updated with new guidance on July 13th, 2023.

Trace requests as they process through service components so product teams can more easily analyze and debug issues and improve performance.

Desired outcome: Workloads with comprehensive tracing across all components are easy to debug, improving mean time to resolution (MTTR) of errors and latency by simplifying root cause discovery. End-to-end tracing reduces the time it takes to discover impacted components and drill into the detailed root causes of errors or latency.

Common anti-patterns:

- Tracing is used for some components but not for all. For example, without tracing for AWS Lambda, teams might not clearly understand latency caused by cold starts in a spiky workload.
- Synthetic canaries or real-user monitoring (RUM) are not configured with tracing. Without canaries or RUM, client interaction telemetry is omitted from the trace analysis yielding an incomplete performance profile.
- Hybrid workloads include both cloud native and third party tracing tools, but steps have not been taken elect and fully integrate a single tracing solution. Based on the elected tracing solution, cloud native tracing SDKs should be used to instrument components that are not cloud native or third party tools should be configured to ingest cloud native trace telemetry.

Benefits of establishing this best practice: When development teams are alerted to issues, they can see a full picture of system component interactions, including component by component correlation to logging, performance, and failures. Because tracing makes it easy to visually identify root causes, less time is spent investigating root causes. Teams that understand component interactions in detail make better and faster decisions when resolving issues. Decisions like when to invoke disaster recovery (DR) failover or where to best implement self-healing strategies can be improved by analyzing systems traces, ultimately improving customer satisfaction with your services.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Teams that operate distributed applications can use tracing tools to establish a correlation identifier, collect traces of requests, and build service maps of connected components. All application components should be included in request traces including service clients, middleware gateways and event buses, compute components, and storage, including key value stores and databases. Include synthetic canaries and real-user monitoring in your end-to-end tracing

configuration to measure remote client interactions and latency so that you can accurately evaluate your systems performance against your service level agreements and objectives.

You can use AWS X-Ray and Amazon CloudWatch Application Monitoring instrumentation services to provide a complete view of requests as they travel through your application. X-Ray collects application telemetry and allows you to visualize and filter it across payloads, functions, traces, services, APIs, and can be turned on for system components with no-code or low-code. CloudWatch application monitoring includes ServiceLens to integrate your traces with metrics, logs, and alarms. CloudWatch application monitoring also includes synthetics to monitor your endpoints and APIs, as well as real-user monitoring to instrument your web application clients.

Implementation steps

- Use AWS X-Ray on all supported native services like Amazon S3, AWS Lambda, and Amazon API Gateway. These AWS services enable X-Ray with configuration toggles using infrastructure as code, AWS SDKs, or the AWS Management Console.
- Instrument applications AWS Distro for Open Telemetry and X-Ray or third-party collection agents.
- Review the AWS X-Ray Developer Guide for programming language specific implementation. These documentation sections detail how to instrument HTTP requests, SQL queries, and other processes specific to your application programming language.
- Use X-Ray tracing for Amazon CloudWatch Synthetic Canaries and Amazon CloudWatch RUM to analyze the request path from your end user client through your downstream AWS infrastructure.
- Configure CloudWatch metrics and alarms based on resource health and canary telemetry so that teams are alerted to issues quickly, and can then deep dive into traces and service maps with ServiceLens.
- Enable X-Ray integration for third party tracing tools like Datadog, New Relic, or Dynatrace if you are using third party tools for your primary tracing solution.

Resources

Related best practices:

- REL06-BP01 Monitor all components for the workload (Generation)
- REL11-BP01 Monitor all components of the workload to detect failures

Related documents:

- What is AWS X-Ray?
- Amazon CloudWatch: Application Monitoring
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility
- Integrating AWS X-Ray with other AWS services
- AWS Distro for OpenTelemetry and AWS X-Ray
- Amazon CloudWatch: Using synthetic monitoring
- Amazon CloudWatch: Use CloudWatch RUM
- Set up Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm
- Availability and Beyond: Understanding and Improving the Resilience of Distributed Systems on AWS

Related examples:

- One Observability Workshop

Related videos:

- AWS re:Invent 2022 - How to monitor applications across multiple accounts
- How to Monitor your AWS Applications

Related tools:

- AWS X-Ray
- Amazon CloudWatch
- Amazon Route 53

REL 7. How do you design your workload to adapt to changes in demand?

A scalable workload provides elasticity to add or remove resources automatically so that they closely match the current demand at any given point in time.

Best practices

- REL07-BP01 Use automation when obtaining or scaling resources
- REL07-BP02 Obtain resources upon detection of impairment to a workload
- REL07-BP03 Obtain resources upon detection that more resources are needed for a workload
- REL07-BP04 Load test your workload

REL07-BP01 Use automation when obtaining or scaling resources

When replacing impaired resources or scaling your workload, automate the process by using managed AWS services, such as Amazon S3 and AWS Auto Scaling. You can also use third-party tools and AWS SDKs to automate scaling.

Managed AWS services include Amazon S3, Amazon CloudFront, AWS Auto Scaling, AWS Lambda, Amazon DynamoDB, AWS Fargate, and Amazon Route 53.

AWS Auto Scaling lets you detect and replace impaired instances. It also lets you build scaling plans for resources including Amazon EC2 instances and Spot Fleets, Amazon ECS tasks, Amazon DynamoDB tables and indexes, and Amazon Aurora Replicas.

When scaling EC2 instances, ensure that you use multiple Availability Zones (preferably at least three) and add or remove capacity to maintain balance across these Availability Zones. ECS tasks or Kubernetes pods (when using Amazon Elastic Kubernetes Service) should also be distributed across multiple Availability Zones.

When using AWS Lambda, instances scale automatically. Every time an event notification is received for your function, AWS Lambda quickly locates free capacity within its compute fleet and runs your code up to the allocated concurrency. You need to ensure that the necessary concurrency is configured on the specific Lambda, and in your Service Quotas.

Amazon S3 automatically scales to handle high request rates. For example, your application can achieve at least 3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per prefix in a bucket. There are no limits to the number of prefixes in a bucket. You can increase your read or write performance by parallelizing reads. For example, if you create 10 prefixes in an Amazon S3 bucket to parallelize reads, you could scale your read performance to 55,000 read requests per second.

Configure and use Amazon CloudFront or a trusted content delivery network (CDN). A CDN can provide faster end-user response times and can serve requests for content from cache, therefore reducing the need to scale your workload.

Common anti-patterns:

- Implementing Auto Scaling groups for automated healing, but not implementing elasticity.
- Using automatic scaling to respond to large increases in traffic.
- Deploying highly stateful applications, eliminating the option of elasticity.

Benefits of establishing this best practice: Automation removes the potential for manual error in deploying and decommissioning resources. Automation removes the risk of cost overruns and denial of service due to slow response on needs for deployment or decommissioning.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Configure and use AWS Auto Scaling. This monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. Using AWS Auto Scaling, you can setup application scaling for multiple resources across multiple services.
  - What is AWS Auto Scaling?
    - Configure Auto Scaling on your Amazon EC2 instances and Spot Fleets, Amazon ECS tasks, Amazon DynamoDB tables and indexes, Amazon Aurora Replicas, and AWS Marketplace appliances as applicable.
      - Managing throughput capacity automatically with DynamoDB Auto Scaling
        - Use service API operations to specify the alarms, scaling policies, warm up times, and cool down times.
- Use Elastic Load Balancing. Load balancers can distribute load by path or by network connectivity.
  - What is Elastic Load Balancing?
    - Application Load Balancers can distribute load by path.
      - What is an Application Load Balancer?
        - Configure an Application Load Balancer to distribute traffic to different workloads based on the path under the domain name.
        - Application Load Balancers can be used to distribute loads in a manner that integrates with AWS Auto Scaling to manage demand.
          - Using a load balancer with an Auto Scaling group
    - Network Load Balancers can distribute load by connection.

    - What is a Network Load Balancer?
        - Configure a Network Load Balancer to distribute traffic to different workloads using TCP, or to have a constant set of IP addresses for your workload.
        - Network Load Balancers can be used to distribute loads in a manner that integrates with AWS Auto Scaling to manage demand.
- Use a highly available DNS provider. DNS names allow your users to enter names instead of IP addresses to access your workloads and distributes this information to a defined scope, usually globally for users of the workload.
    - Use Amazon Route 53 or a trusted DNS provider.
        - What is Amazon Route 53?
    - Use Route 53 to manage your CloudFront distributions and load balancers.
        - Determine the domains and subdomains you are going to manage.
        - Create appropriate record sets using ALIAS or CNAME records.
            - Working with records
- Use the AWS global network to optimize the path from your users to your applications. AWS Global Accelerator continually monitors the health of your application endpoints and redirects traffic to healthy endpoints in less than 30 seconds.
    - AWS Global Accelerator is a service that improves the availability and performance of your applications with local or global users. It provides static IP addresses that act as a fixed entry point to your application endpoints in a single or multiple AWS Regions, such as your Application Load Balancers, Network Load Balancers or Amazon EC2 instances.
        - What Is AWS Global Accelerator?
- Configure and use Amazon CloudFront or a trusted content delivery network (CDN). A content delivery network can provide faster end-user response times and can serve requests for content that may cause unnecessary scaling of your workloads.
    - What is Amazon CloudFront?
        - Configure Amazon CloudFront distributions for your workloads, or use a third-party CDN.
            - You can limit access to your workloads so that they are only accessible from CloudFront by using the IP ranges for CloudFront in your endpoint security groups or access policies.

Resources

Related documents:

- APN Partner: partners that can help you create automated compute solutions
- AWS Auto Scaling: How Scaling Plans Work
- AWS Marketplace: products that can be used with auto scaling
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- Using a load balancer with an Auto Scaling group
- What Is AWS Global Accelerator?
- What Is Amazon EC2 Auto Scaling?
- What is AWS Auto Scaling?
- What is Amazon CloudFront?
- What is Amazon Route 53?
- What is Elastic Load Balancing?
- What is a Network Load Balancer?
- What is an Application Load Balancer?
- Working with records

REL07-BP02 Obtain resources upon detection of impairment to a workload

Scale resources reactively when necessary if availability is impacted, to restore workload availability.

You first must configure health checks and the criteria on these checks to indicate when availability is impacted by lack of resources. Then either notify the appropriate personnel to manually scale the resource, or start automation to automatically scale it.

Scale can be manually adjusted for your workload, for example, changing the number of EC2 instances in an Auto Scaling group or modifying throughput of a DynamoDB table can be done through the AWS Management Console or AWS CLI. However automation should be used whenever possible (refer to **Use automation when obtaining or scaling resources**).

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Obtain resources upon detection of impairment to a workload. Scale resources reactively when necessary if availability is impacted, to restore workload availability.

- Use scaling plans, which are the core component of AWS Auto Scaling, to configure a set of instructions for scaling your resources. If you work with AWS CloudFormation or add tags to AWS resources, you can set up scaling plans for different sets of resources, per application. AWS Auto Scaling provides recommendations for scaling strategies customized to each resource. After you create your scaling plan, AWS Auto Scaling combines dynamic scaling and predictive scaling methods together to support your scaling strategy.
  - AWS Auto Scaling: How Scaling Plans Work
- Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes above this size.
  - What Is Amazon EC2 Auto Scaling?
- Amazon DynamoDB auto scaling uses the AWS Application Auto Scaling service to dynamically adjust provisioned throughput capacity on your behalf, in response to actual traffic patterns. This allows a table or a global secondary index to increase its provisioned read and write capacity to handle sudden increases in traffic, without throttling.
  - Managing Throughput Capacity Automatically with DynamoDB Auto Scaling

Resources

Related documents:

- APN Partner: partners that can help you create automated compute solutions
- AWS Auto Scaling: How Scaling Plans Work
- AWS Marketplace: products that can be used with auto scaling
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- What Is Amazon EC2 Auto Scaling?

REL07-BP03 Obtain resources upon detection that more resources are needed for a workload

Scale resources proactively to meet demand and avoid availability impact.

Many AWS services automatically scale to meet demand. If using Amazon EC2 instances or Amazon ECS clusters, you can configure automatic scaling of these to occur based on usage metrics that

correspond to demand for your workload. For Amazon EC2, average CPU utilization, load balancer request count, or network bandwidth can be used to scale out (or scale in) EC2 instances. For Amazon ECS, average CPU utilization, load balancer request count, and memory utilization can be used to scale out (or scale in) ECS tasks. Using Target Auto Scaling on AWS, the autoscaler acts like a household thermostat, adding or removing resources to maintain the target value (for example, 70% CPU utilization) that you specify.

AWS Auto Scaling can also do Predictive Auto Scaling, which uses machine learning to analyze each resource's historical workload and regularly forecasts the future load for the next two days.

Little's Law helps calculate how many instances of compute (EC2 instances, concurrent Lambda functions, etc.) that you need.

$L = \lambda W$

L = number of instances (or mean concurrency in the system)

λ = mean rate at which requests arrive (req/sec)

W = mean time that each request spends in the system (sec)

For example, at 100 rps, if each request takes 0.5 seconds to process, you will need 50 instances to keep up with demand.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Obtain resources upon detection that more resources are needed for a workload. Scale resources proactively to meet demand and avoid availability impact.
  - Calculate how many compute resources you will need (compute concurrency) to handle a given request rate.
    - Telling Stories About Little's Law
  - When you have a historical pattern for usage, set up scheduled scaling for Amazon EC2 auto scaling.
    - Scheduled Scaling for Amazon EC2 Auto Scaling
  - Use AWS predictive scaling.
    - Predictive Scaling for EC2, Powered by Machine Learning

Resources

Related documents:

- AWS Auto Scaling: How Scaling Plans Work
- AWS Marketplace: products that can be used with auto scaling
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- Predictive Scaling for EC2, Powered by Machine Learning
- Scheduled Scaling for Amazon EC2 Auto Scaling
- Telling Stories About Little's Law
- What Is Amazon EC2 Auto Scaling?

REL07-BP04 Load test your workload

Adopt a load testing methodology to measure if scaling activity meets workload requirements.

It's important to perform sustained load testing. Load tests should discover the breaking point and test the performance of your workload. AWS makes it easy to set up temporary testing environments that model the scale of your production workload. 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.

Load testing in production should also be considered as part of game days where the production system is stressed, during hours of lower customer usage, with all personnel on hand to interpret results and address any problems that arise.

Common anti-patterns:

- Performing load testing on deployments that are not the same configuration as your production.
- Performing load testing only on individual pieces of your workload, and not on the entire workload.
- Performing load testing with a subset of requests and not a representative set of actual requests.
- Performing load testing to a small safety factor above expected load.

Benefits of establishing this best practice: You know what components in your architecture fail under load and be able to identify what metrics to watch to indicate that you are approaching that load in time to address the problem, preventing the impact of that failure.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Perform load testing to identify which aspect of your workload indicates that you must add or remove capacity. Load testing should have representative traffic similar to what you receive in production. Increase the load while watching the metrics you have instrumented to determine which metric indicates when you must add or remove resources.
  - Distributed Load Testing on AWS: simulate thousands of connected users
    - Identify the mix of requests. You may have varied mixes of requests, so you should look at various time frames when identifying the mix of traffic.
    - Implement a load driver. You can use custom code, open source, or commercial software to implement a load driver.
    - Load test initially using small capacity. You see some immediate effects by driving load onto a lesser capacity, possibly as small as one instance or container.
    - Load test against larger capacity. The effects will be different on a distributed load, so you must test against as close to a product environment as possible.

Resources

Related documents:

- Distributed Load Testing on AWS: simulate thousands of connected users

REL 8. How do you implement change?

Controlled changes are necessary to deploy new functionality, and to verify that the workloads and the operating environment are running known software and can be patched or replaced in a predictable manner. If these changes are uncontrolled, then it makes it difficult to predict the effect of these changes, or to address issues that arise because of them.

Best practices

- REL08-BP01 Use runbooks for standard activities such as deployment

- REL08-BP02 Integrate functional testing as part of your deployment
- REL08-BP03 Integrate resiliency testing as part of your deployment
- REL08-BP04 Deploy using immutable infrastructure
- REL08-BP05 Deploy changes with automation

REL08-BP01 Use runbooks for standard activities such as deployment

Runbooks are the predefined procedures to achieve specific outcomes. Use runbooks to perform standard activities, whether done manually or automatically. Examples include deploying a workload, patching a workload, or making DNS modifications.

For example, put processes in place to ensure rollback safety during deployments. Ensuring that you can roll back a deployment without any disruption for your customers is critical in making a service reliable.

For runbook procedures, start with a valid effective manual process, implement it in code, and invoke it to automatically run where appropriate.

Even for sophisticated workloads that are highly automated, runbooks are still useful for running game days or meeting rigorous reporting and auditing requirements.

Note that playbooks are used in response to specific incidents, and runbooks are used to achieve specific outcomes. Often, runbooks are for routine activities, while playbooks are used for responding to non-routine events.

Common anti-patterns:

- Performing unplanned changes to configuration in production.
- Skipping steps in your plan to deploy faster, resulting in a failed deployment.
- Making changes without testing the reversal of the change.

Benefits of establishing this best practice: Effective change planning increases your ability to successfully run the change because you are aware of all the systems impacted. Validating your change in test environments increases your confidence.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Provide consistent and prompt responses to well-understood events by documenting procedures in runbooks.
  - AWS Well-Architected Framework: Concepts: Runbook
- Use the principle of infrastructure as code to define your infrastructure. By using AWS CloudFormation (or a trusted third party) to define your infrastructure, you can use version control software to version and track changes.
  - Use AWS CloudFormation (or a trusted third-party provider) to define your infrastructure.
    - What is AWS CloudFormation?
  - Create templates that are singular and decoupled, using good software design principles.
    - Determine the permissions, templates, and responsible parties for implementation.
      - Controlling access with AWS Identity and Access Management
    - Use source control, like AWS CodeCommit or a trusted third-party tool, for version control.
      - What is AWS CodeCommit?

Resources

Related documents:

- APN Partner: partners that can help you create automated deployment solutions
- AWS Marketplace: products that can be used to automate your deployments
- AWS Well-Architected Framework: Concepts: Runbook
- What is AWS CloudFormation?
- What is AWS CodeCommit?

Related examples:

- Automating operations with Playbooks and Runbooks

REL08-BP02 Integrate functional testing as part of your deployment

Functional tests are run as part of automated deployment. If success criteria are not met, the pipeline is halted or rolled back.

These tests are run in a pre-production environment, which is staged prior to production in the pipeline. Ideally, this is done as part of a deployment pipeline.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Integrate functional testing as part of your deployment. Functional tests are run as part of automated deployment. If success criteria are not met, the pipeline is halted or rolled back.
  - Invoke AWS CodeBuild during the 'Test Action' of your software release pipelines modeled in AWS CodePipeline. This capability allows you to easily run a variety of tests against your code, such as unit tests, static code analysis, and integration tests.
    - AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild
  - Use AWS Marketplace solutions for invoking automated tests as part of your software delivery pipeline.
    - Software test automation

Resources

Related documents:

- AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild
- Software test automation
- What Is AWS CodePipeline?

REL08-BP03 Integrate resiliency testing as part of your deployment

Resiliency tests (using the principles of chaos engineering) are run as part of the automated deployment pipeline in a pre-production environment.

These tests are staged and run in the pipeline in a pre-production environment. They should also be run in production as part of game days.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Integrate resiliency testing as part of your deployment. Use Chaos Engineering, the discipline of experimenting on a workload to build confidence in the workload's capability to withstand turbulent conditions in production.
  - Resiliency tests inject faults or resource degradation to assess that your workload responds with its designed resilience.
    - Well-Architected lab: Level 300: Testing for Resiliency of EC2 RDS and S3
  - These tests can be run regularly in pre-production environments in automated deployment pipelines.
  - They should also be run in production, as part of scheduled game days.
  - Using Chaos Engineering principles, propose hypotheses about how your workload will perform under various impairments, then test your hypotheses using resiliency testing.
    - Principles of Chaos Engineering

Resources

Related documents:

- Principles of Chaos Engineering
- What is AWS Fault Injection Simulator?

Related examples:

- Well-Architected lab: Level 300: Testing for Resiliency of EC2 RDS and S3

REL08-BP04 Deploy using immutable infrastructure

Immutable infrastructure is a model that mandates that no updates, security patches, or configuration changes happen in-place on production workloads. When a change is needed, the architecture is built onto new infrastructure and deployed into production.

The most common implementation of the immutable infrastructure paradigm is the ***immutable server***. This means that if a server needs an update or a fix, new servers are deployed instead of updating the ones already in use. So, instead of logging into the server via SSH and updating the software version, every change in the application starts with a software push to the code

repository, for example, git push. Since changes are not allowed in immutable infrastructure, you can be sure about the state of the deployed system. Immutable infrastructures are inherently more consistent, reliable, and predictable, and they simplify many aspects of software development and operations.

Use a canary or blue/green deployment when deploying applications in immutable infrastructures.

Canary deployment is the practice of directing a small number of your customers to the new version, usually running on a single service instance (the canary). You then deeply scrutinize any behavior changes or errors that are generated. You can remove traffic from the canary if you encounter critical problems and send the users back to the previous version. If the deployment is successful, you can continue to deploy at your desired velocity, while monitoring the changes for errors, until you are fully deployed. AWS CodeDeploy can be configured with a deployment configuration that will allow a canary deployment.

Blue/green deployment is similar to the canary deployment except that a full fleet of the application is deployed in parallel. You alternate your deployments across the two stacks (blue and green). Once again, you can send traffic to the new version, and fall back to the old version if you see problems with the deployment. Commonly all traffic is switched at once, however you can also use fractions of your traffic to each version to dial up the adoption of the new version using the weighted DNS routing capabilities of Amazon Route 53. AWS CodeDeploy and AWS Elastic Beanstalk can be configured with a deployment configuration that will allow a blue/green deployment.

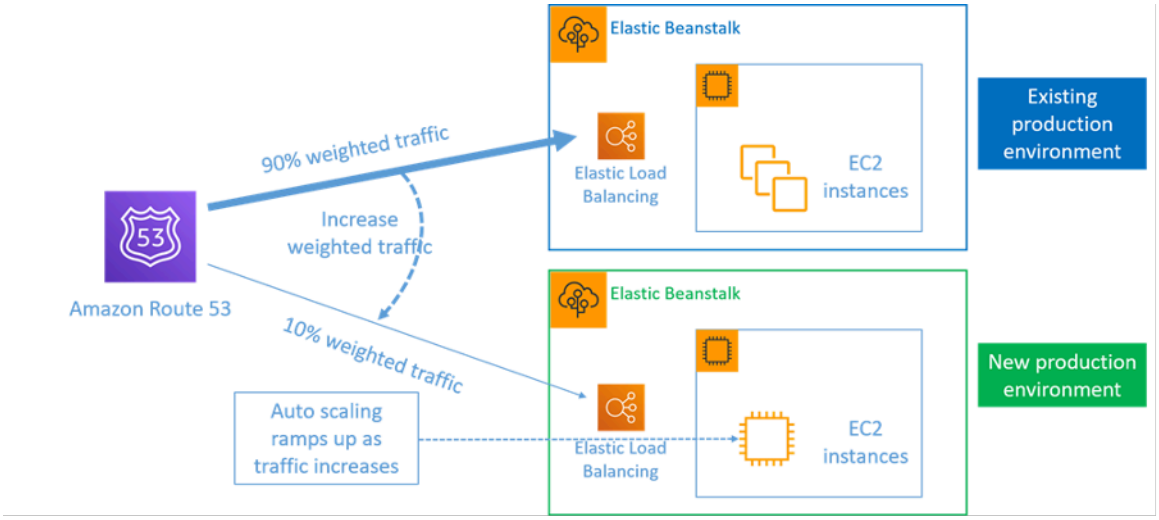


Figure 8: Blue/green deployment with AWS Elastic Beanstalk and Amazon Route 53

Benefits of immutable infrastructure:

- **Reduction in configuration drifts:** By frequently replacing servers from a base, known and version-controlled configuration, the infrastructure is **reset** to a known state, avoiding configuration drifts.
- **Simplified deployments**: Deployments are simplified because they don't need to support upgrades. Upgrades are just new deployments.
- **Reliable atomic deployments:** Deployments either complete successfully, or nothing changes. It gives more trust in the deployment process.
- **Safer deployments with fast rollback and recovery processes:** Deployments are safer because the previous working version is not changed. You can roll back to it if errors are detected.
- **Consistent testing and debugging environments:** Since all servers use the same image, there are no differences between environments. One build is deployed to multiple environments. It also prevents inconsistent environments and simplifies testing and debugging.
- **Increased scalability:** Since servers use a base image, are consistent, and repeatable, automatic scaling is trivial.
- **Simplified toolchain**: The toolchain is simplified since you can get rid of configuration management tools managing production software upgrades. No extra tools or agents are installed on servers. Changes are made to the base image, tested, and rolled-out.
- **Increased security:** By denying all changes to servers, you can disable SSH on instances and remove keys. This reduces the attack vector, improving your organization's security posture.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Deploy using immutable infrastructure. Immutable infrastructure is a model in which no updates, security patches, or configuration changes happen *in-place* on production systems. If any change is needed, a new version of the architecture is built and deployed into production.
  - Overview of a Blue/Green Deployment
  - Deploying Serverless Applications Gradually
  - Immutable Infrastructure: Reliability, consistency and confidence through immutability
  - CanaryRelease

Resources

Related documents:

- CanaryRelease
- Deploying Serverless Applications Gradually
- Immutable Infrastructure: Reliability, consistency and confidence through immutability
- Overview of a Blue/Green Deployment
- The Amazon Builders' Library: Ensuring rollback safety during deployments

REL08-BP05 Deploy changes with automation

Deployments and patching are automated to eliminate negative impact.

Making changes to production systems is one of the largest risk areas for many organizations. We consider deployments a first-class problem to be solved alongside the business problems that the software addresses. Today, this means the use of automation wherever practical in operations, including testing and deploying changes, adding or removing capacity, and migrating data. AWS CodePipeline lets you manage the steps required to release your workload. This includes a deployment state using AWS CodeDeploy to automate deployment of application code to Amazon EC2 instances, on-premises instances, serverless Lambda functions, or Amazon ECS services.

> **Recommendation**
>
> Although conventional wisdom suggests that you keep humans in the loop for the most difficult operational procedures, we suggest that you automate the most difficult procedures for that very reason.

Common anti-patterns:

- Manually performing changes.
- Skipping steps in your automation through emergency work flows.
- Not following your plans.

Benefits of establishing this best practice: Using automation to deploy all changes removes the potential for introduction of human error and provides the ability to test before changing production to ensure that your plans are complete.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Automate your deployment pipeline. Deployment pipelines allow you to invoke automated testing and detection of anomalies, and either halt the pipeline at a certain step before production deployment, or automatically roll back a change.
  - The Amazon Builders' Library: Ensuring rollback safety during deployments
  - The Amazon Builders' Library: Going faster with continuous delivery
    - Use AWS CodePipeline (or a trusted third-party product) to define and run your pipelines.
      - Configure the pipeline to start when a change is committed to your code repository.
        - What is AWS CodePipeline?
      - Use Amazon Simple Notification Service (Amazon SNS) and Amazon Simple Email Service (Amazon SES) to send notifications about problems in the pipeline or integrate with a team chat tool, like Amazon Chime.
        - What is Amazon Simple Notification Service?
        - What is Amazon SES?
        - What is Amazon Chime?
        - Automate chat messages with webhooks.

Resources

Related documents:

- APN Partner: partners that can help you create automated deployment solutions
- AWS Marketplace: products that can be used to automate your deployments
- Automate chat messages with webhooks.
- The Amazon Builders' Library: Ensuring rollback safety during deployments
- The Amazon Builders' Library: Going faster with continuous delivery
- What Is AWS CodePipeline?
- What Is CodeDeploy?
- AWS Systems Manager Patch Manager
- What is Amazon SES?
- What is Amazon Simple Notification Service?

Related videos:

- AWS Summit 2019: CI/CD on AWS

Failure management

Questions

- REL 9. How do you back up data?
- REL 10. How do you use fault isolation to protect your workload?
- REL 11. How do you design your workload to withstand component failures?
- REL 12. How do you test reliability?
- REL 13. How do you plan for disaster recovery (DR)?

REL 9. How do you back up data?

Back up data, applications, and configuration to meet your requirements for recovery time objectives (RTO) and recovery point objectives (RPO).

Best practices

- REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources
- REL09-BP02 Secure and encrypt backups
- REL09-BP03 Perform data backup automatically
- REL09-BP04 Perform periodic recovery of the data to verify backup integrity and processes

REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources

Understand and use the backup capabilities of the data services and resources used by the workload. Most services provide capabilities to back up workload data.

Desired outcome: Data sources have been identified and classified based on criticality. Then, establish a strategy for data recovery based on the RPO. This strategy involves either backing up these data sources, or having the ability to reproduce data from other sources. In the case of data loss, the strategy implemented allows recovery or the reproduction of data within the defined RPO and RTO.

Cloud maturity phase: Foundational

Common anti-patterns:

- Not aware of all data sources for the workload and their criticality.
- Not taking backups of critical data sources.
- Taking backups of only some data sources without using criticality as a criterion.
- No defined RPO, or backup frequency cannot meet RPO.
- Not evaluating if a backup is necessary or if data can be reproduced from other sources.

Benefits of establishing this best practice: Identifying the places where backups are necessary and implementing a mechanism to create backups, or being able to reproduce the data from an external source improves the ability to restore and recover data during an outage.

Level of risk exposed if this best practice is not established: High

Implementation guidance

All AWS data stores offer backup capabilities. Services such as Amazon RDS and Amazon DynamoDB additionally support automated backup that allows point-in-time recovery (PITR), which allows you to restore a backup to any time up to five minutes or less before the current time. Many AWS services offer the ability to copy backups to another AWS Region. AWS Backup is a tool that gives you the ability to centralize and automate data protection across AWS services. AWS Elastic Disaster Recovery allows you to copy full server workloads and maintain continuous data protection from on-premise, cross-AZ or cross-Region, with a Recovery Point Objective (RPO) measured in seconds.

Amazon S3 can be used as a backup destination for self-managed and AWS-managed data sources. AWS services such as Amazon EBS, Amazon RDS, and Amazon DynamoDB have built in capabilities to create backups. Third-party backup software can also be used.

On-premises data can be backed up to the AWS Cloud using AWS Storage Gateway or AWS DataSync. Amazon S3 buckets can be used to store this data on AWS. Amazon S3 offers multiple storage tiers such as Amazon Glacier or Amazon Glacier Deep Archive to reduce cost of data storage.

You might be able to meet data recovery needs by reproducing the data from other sources. For example, Amazon ElastiCache replica nodes or Amazon RDS read replicas could be used to

reproduce data if the primary is lost. In cases where sources like this can be used to meet your Recovery Point Objective (RPO) and Recovery Time Objective (RTO), you might not require a backup. Another example, if working with Amazon EMR, it might not be necessary to backup your HDFS data store, as long as you can reproduce the data into Amazon EMR from Amazon S3.

When selecting a backup strategy, consider the time it takes to recover data. The time needed to recover data depends on the type of backup (in the case of a backup strategy), or the complexity of the data reproduction mechanism. This time should fall within the RTO for the workload.

Implementation steps

1. **Identify all data sources for the workload**. Data can be stored on a number of resources such as databases, volumes, filesystems, logging systems, and object storage. Refer to the **Resources** section to find **Related documents** on different AWS services where data is stored, and the backup capability these services provide.
2. **Classify data sources based on criticality**. Different data sets will have different levels of criticality for a workload, and therefore different requirements for resiliency. For example, some data might be critical and require a RPO near zero, while other data might be less critical and can tolerate a higher RPO and some data loss. Similarly, different data sets might have different RTO requirements as well.
3. **Use AWS or third-party services to create backups of the data**. AWS Backup is a managed service that allows creating backups of various data sources on AWS. AWS Elastic Disaster Recovery handles automated sub-second data replication to an AWS Region. Most AWS services also have native capabilities to create backups. The AWS Marketplace has many solutions that provide these capabilites as well. Refer to the **Resources** listed below for information on how to create backups of data from various AWS services.
4. **For data that is not backed up, establish a data reproduction mechanism**. You might choose not to backup data that can be reproduced from other sources for various reasons. There might be a situation where it is cheaper to reproduce data from sources when needed rather than creating a backup as there may be a cost associated with storing backups. Another example is where restoring from a backup takes longer than reproducing the data from sources, resulting in a breach in RTO. In such situations, consider tradeoffs and establish a well-defined process for how data can be reproduced from these sources when data recovery is necessary. For example, if you have loaded data from Amazon S3 to a data warehouse (like Amazon Redshift), or MapReduce cluster (like Amazon EMR) to do analysis on that data, this may be an example of data that can be reproduced from other sources. As long as the results of these analyses are either stored somewhere or reproducible, you would not suffer a data loss from a failure in the

data warehouse or MapReduce cluster. Other examples that can be reproduced from sources include caches (like Amazon ElastiCache) or RDS read replicas.

5. **Establish a cadence for backing up data**. Creating backups of data sources is a periodic process and the frequency should depend on the RPO.

Level of effort for the Implementation Plan: Moderate

Resources

Related Best Practices:

REL13-BP01 Define recovery objectives for downtime and data loss

REL13-BP02 Use defined recovery strategies to meet the recovery objectives

Related documents:

- What Is AWS Backup?
- What is AWS DataSync?
- What is Volume Gateway?
- APN Partner: partners that can help with backup
- AWS Marketplace: products that can be used for backup
- Amazon EBS Snapshots
- Backing Up Amazon EFS
- Backing up Amazon FSx for Windows File Server
- Backup and Restore for ElastiCache for Redis
- Creating a DB Cluster Snapshot in Neptune
- Creating a DB Snapshot
- Creating an EventBridge Rule That Triggers on a Schedule
- Cross-Region Replication with Amazon S3
- EFS-to-EFS AWS Backup
- Exporting Log Data to Amazon S3
- Object lifecycle management
- On-Demand Backup and Restore for DynamoDB
- Point-in-time recovery for DynamoDB

- Working with Amazon OpenSearch Service Index Snapshots
- What is AWS Elastic Disaster Recovery?

Related videos:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS
- AWS Backup Demo: Cross-Account and Cross-Region Backup
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

Related examples:

- Well-Architected Lab - Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3
- Well-Architected Lab - Testing Backup and Restore of Data
- Well-Architected Lab - Backup and Restore with Failback for Analytics Workload
- Well-Architected Lab - Disaster Recovery - Backup and Restore

REL09-BP02 Secure and encrypt backups

Control and detect access to backups using authentication and authorization. Prevent and detect if data integrity of backups is compromised using encryption.

Common anti-patterns:

- Having the same access to the backups and restoration automation as you do to the data.
- Not encrypting your backups.

Benefits of establishing this best practice: Securing your backups prevents tampering with the data, and encryption of the data prevents access to that data if it is accidentally exposed.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Control and detect access to backups using authentication and authorization, such as AWS Identity and Access Management (IAM). Prevent and detect if data integrity of backups is compromised using encryption.

Amazon S3 supports several methods of encryption of your data at rest. Using server-side encryption, Amazon S3 accepts your objects as unencrypted data, and then encrypts them as they are stored. Using client-side encryption, your workload application is responsible for encrypting the data before it is sent to Amazon S3. Both methods allow you to use AWS Key Management Service (AWS KMS) to create and store the data key, or you can provide your own key, which you are then responsible for. Using AWS KMS, you can set policies using IAM on who can and cannot access your data keys and decrypted data.

For Amazon RDS, if you have chosen to encrypt your databases, then your backups are encrypted also. DynamoDB backups are always encrypted. When using AWS Elastic Disaster Recovery, all data in transit and at rest is encrypted. With Elastic Disaster Recovery, data at rest can be encrypted using either the default Amazon EBS encryption Volume Encryption Key or a custom customer-managed key.

Implementation steps

1. Use encryption on each of your data stores. If your source data is encrypted, then the backup will also be encrypted.
   - Use encryption in Amazon RDS.. You can configure encryption at rest using AWS Key Management Service when you create an RDS instance.
   - Use encryption on Amazon EBS volumes.. You can configure default encryption or specify a unique key upon volume creation.
   - Use the required Amazon DynamoDB encryption. DynamoDB encrypts all data at rest. You can either use an AWS owned AWS KMS key or an AWS managed KMS key, specifying a key that is stored in your account.
   - Encrypt your data stored in Amazon EFS. Configure the encryption when you create your file system.
   - Configure the encryption in the source and destination Regions. You can configure encryption at rest in Amazon S3 using keys stored in KMS, but the keys are Region-specific. You can specify the destination keys when you configure the replication.
   - Choose whether to use the default or custom Amazon EBS encryption for Elastic Disaster Recovery. This option will encrypt your replicated data at rest on the Staging Area Subnet disks and the replicated disks.
2. Implement least privilege permissions to access your backups. Follow best practices to limit the access to the backups, snapshots, and replicas in accordance with security best practices.

Resources

Related documents:

- AWS Marketplace: products that can be used for backup
- Amazon EBS Encryption
- Amazon S3: Protecting Data Using Encryption
- CRR Additional Configuration: Replicating Objects Created with Server-Side Encryption (SSE) Using Encryption Keys stored in AWS KMS
- DynamoDB Encryption at Rest
- Encrypting Amazon RDS Resources
- Encrypting Data and Metadata in Amazon EFS
- Encryption for Backups in AWS
- Managing Encrypted Tables
- Security Pillar - AWS Well-Architected Framework
- What is AWS Elastic Disaster Recovery?

Related examples:

- Well-Architected Lab - Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3

REL09-BP03 Perform data backup automatically

Configure backups to be taken automatically based on a periodic schedule informed by the Recovery Point Objective (RPO), or by changes in the dataset. Critical datasets with low data loss requirements need to be backed up automatically on a frequent basis, whereas less critical data where some loss is acceptable can be backed up less frequently.

Desired outcome: An automated process that creates backups of data sources at an established cadence.

Common anti-patterns:

- Performing backups manually.
- Using resources that have backup capability, but not including the backup in your automation.

Benefits of establishing this best practice: Automating backups verifies that they are taken regularly based on your RPO, and alerts you if they are not taken.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS Backup can be used to create automated data backups of various AWS data sources. Amazon RDS instances can be backed up almost continuously every five minutes and Amazon S3 objects can be backed up almost continuously every fifteen minutes, providing for point-in-time recovery (PITR) to a specific point in time within the backup history. For other AWS data sources, such as Amazon EBS volumes, Amazon DynamoDB tables, or Amazon FSx file systems, AWS Backup can run automated backup as frequently as every hour. These services also offer native backup capabilities. AWS services that offer automated backup with point-in-time recovery include Amazon DynamoDB, Amazon RDS, and Amazon Keyspaces (for Apache Cassandra) – these can be restored to a specific point in time within the backup history. Most other AWS data storage services offer the ability to schedule periodic backups, as frequently as every hour.

Amazon RDS and Amazon DynamoDB offer continuous backup with point-in-time recovery. Amazon S3 versioning, once turned on, is automatic. Amazon Data Lifecycle Manager can be used to automate the creation, copy and deletion of Amazon EBS snapshots. It can also automate the creation, copy, deprecation and deregistration of Amazon EBS-backed Amazon Machine Images (AMIs) and their underlying Amazon EBS snapshots.

AWS Elastic Disaster Recovery provides continuous block-level replication from the source environment (on-premises or AWS) to the target recovery region. Point-in-time Amazon EBS snapshots are automatically created and managed by the service.

For a centralized view of your backup automation and history, AWS Backup provides a fully managed, policy-based backup solution. It centralizes and automates the back up of data across multiple AWS services in the cloud as well as on premises using the AWS Storage Gateway.

In additional to versioning, Amazon S3 features replication. The entire S3 bucket can be automatically replicated to another bucket in the same, or a different AWS Region.

Implementation steps

1. **Identify data sources** that are currently being backed up manually. For more detail, see REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources.

2. **Determine the RPO** for the workload. For more detail, see REL13-BP01 Define recovery objectives for downtime and data loss.
3. **Use an automated backup solution or managed service**. AWS Backup is a fully-managed service that makes it easy to centralize and automate data protection across AWS services, in the cloud, and on-premises. Using backup plans in AWS Backup, create rules which define the resources to backup, and the frequency at which these backups should be created. This frequency should be informed by the RPO established in Step 2. For hands-on guidance on how to create automated backups using AWS Backup, see Testing Backup and Restore of Data. Native backup capabilities are offered by most AWS services that store data. For example, RDS can be leveraged for automated backups with point-in-time recovery (PITR).
4. **For data sources not supported** by an automated backup solution or managed service such as on-premises data sources or message queues, consider using a trusted third-party solution to create automated backups. Alternatively, you can create automation to do this using the AWS CLI or SDKs. You can use AWS Lambda Functions or AWS Step Functions to define the logic involved in creating a data backup, and use Amazon EventBridge to invoke it at a frequency based on your RPO.

Level of effort for the Implementation Plan: Low

Resources

Related documents:

- APN Partner: partners that can help with backup
- AWS Marketplace: products that can be used for backup
- Creating an EventBridge Rule That Triggers on a Schedule
- What Is AWS Backup?
- What Is AWS Step Functions?
- What is AWS Elastic Disaster Recovery?

Related videos:

- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

Related examples:

- Well-Architected Lab - Testing Backup and Restore of Data

REL09-BP04 Perform periodic recovery of the data to verify backup integrity and processes

Validate that your backup process implementation meets your Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO) by performing a recovery test.

Desired outcome: Data from backups is periodically recovered using well-defined mechanisms to verify that recovery is possible within the established recovery time objective (RTO) for the workload. Verify that restoration from a backup results in a resource that contains the original data without any of it being corrupted or inaccessible, and with data loss within the recovery point objective (RPO).

Common anti-patterns:

- Restoring a backup, but not querying or retrieving any data to check that the restoration is usable.
- Assuming that a backup exists.
- Assuming that the backup of a system is fully operational and that data can be recovered from it.
- Assuming that the time to restore or recover data from a backup falls within the RTO for the workload.
- Assuming that the data contained on the backup falls within the RPO for the workload
- Restoring when necessary, without using a runbook or outside of an established automated procedure.

Benefits of establishing this best practice: Testing the recovery of the backups verifies that data can be restored when needed without having any worry that data might be missing or corrupted, that the restoration and recovery is possible within the RTO for the workload, and any data loss falls within the RPO for the workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Testing backup and restore capability increases confidence in the ability to perform these actions during an outage. Periodically restore backups to a new location and run tests to verify the integrity of the data. Some common tests that should be performed are checking if all data is available, is not corrupted, is accessible, and that any data loss falls within the RPO for

the workload. Such tests can also help ascertain if recovery mechanisms are fast enough to accommodate the workload's RTO.

Using AWS, you can stand up a testing environment and restore your backups to assess RTO and RPO capabilities, and run tests on data content and integrity.

Additionally, Amazon RDS and Amazon DynamoDB allow point-in-time recovery (PITR). Using continuous backup, you can restore your dataset to the state it was in at a specified date and time.

If all the data is available, is not corrupted, is accessible, and any data loss falls within the RPO for the workload. Such tests can also help ascertain if recovery mechanisms are fast enough to accommodate the workload's RTO.

AWS Elastic Disaster Recovery offers continual point-in-time recovery snapshots of Amazon EBS volumes. As source servers are replicated, point-in-time states are chronicled over time based on the configured policy. Elastic Disaster Recovery helps you verify the integrity of these snapshots by launching instances for test and drill purposes without redirecting the traffic.

Implementation steps

1. **Identify data sources** that are currently being backed up and where these backups are being stored. For implementation guidance, see REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources.
2. **Establish criteria for data validation** for each data source. Different types of data will have different properties which might require different validation mechanisms. Consider how this data might be validated before you are confident to use it in production. Some common ways to validate data are using data and backup properties such as data type, format, checksum, size, or a combination of these with custom validation logic. For example, this might be a comparison of the checksum values between the restored resource and the data source at the time the backup was created.
3. **Establish RTO and RPO** for restoring the data based on data criticality. For implementation guidance, see REL13-BP01 Define recovery objectives for downtime and data loss.
4. **Assess your recovery capability**. Review your backup and restore strategy to understand if it can meet your RTO and RPO, and adjust the strategy as necessary. Using AWS Resilience Hub, you can run an assessment of your workload. The assessment evaluates your application configuration against the resiliency policy and reports if your RTO and RPO targets can be met.
5. **Do a test restore** using currently established processes used in production for data restoration. These processes depend on how the original data source was backed up, the format and storage

location of the backup itself, or if the data is reproduced from other sources. For example, if you are using a managed service such as AWS Backup, this might be as simple as restoring the backup into a new resource. If you used AWS Elastic Disaster Recovery you can launch a recovery drill.

6. **Validate data recovery** from the restored resource based on criteria you previously established for data validation. Does the restored and recovered data contain the most recent record or item at the time of backup? Does this data fall within the RPO for the workload?
7. **Measure time required** for restore and recovery and compare it to your established RTO. Does this process fall within the RTO for the workload? For example, compare the timestamps from when the restoration process started and when the recovery validation completed to calculate how long this process takes. All AWS API calls are timestamped and this information is available in AWS CloudTrail. While this information can provide details on when the restore process started, the end timestamp for when the validation was completed should be recorded by your validation logic. If using an automated process, then services like Amazon DynamoDB can be used to store this information. Additionally, many AWS services provide an event history which provides timestamped information when certain actions occurred. Within AWS Backup, backup and restore actions are referred to as *jobs*, and these jobs contain timestamp information as part of its metadata which can be used to measure time required for restoration and recovery.
8. **Notify stakeholders** if data validation fails, or if the time required for restoration and recovery exceeds the established RTO for the workload. When implementing automation to do this, such as in this lab, services like Amazon Simple Notification Service (Amazon SNS) can be used to send push notifications such as email or SMS to stakeholders. These messages can also be published to messaging applications such as Amazon Chime, Slack, or Microsoft Teams or used to create tasks as OpsItems using AWS Systems Manager OpsCenter.
9. **Automate this process to run periodically**. For example, services like AWS Lambda or a State Machine in AWS Step Functions can be used to automate the restore and recovery processes, and Amazon EventBridge can be used to invoke this automation workflow periodically as shown in the architecture diagram below. Learn how to Automate data recovery validation with AWS Backup. Additionally, this Well-Architected lab provides a hands-on experience on one way to do automation for several of the steps here.

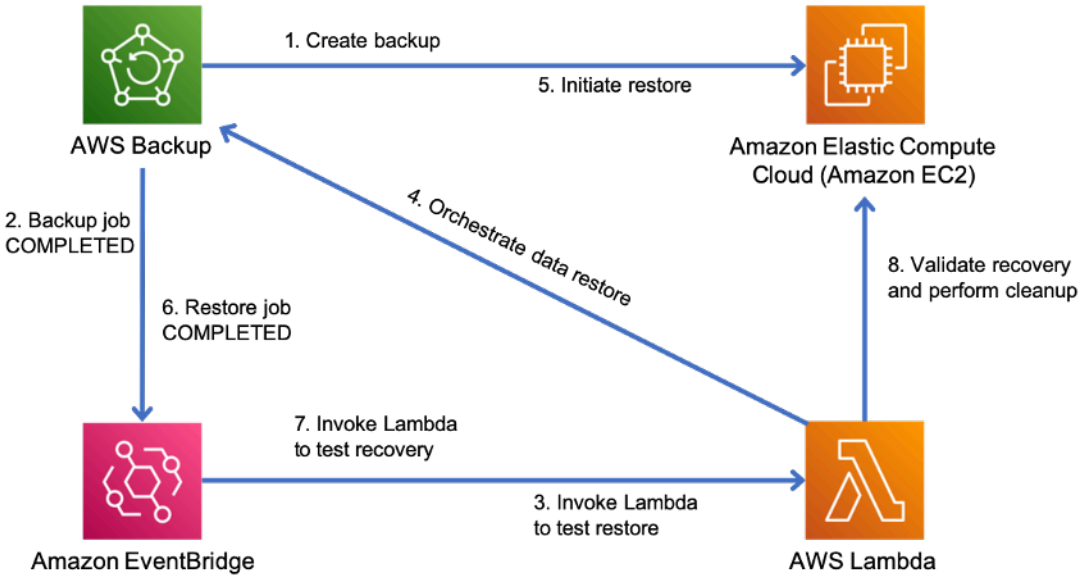


Figure 9. An automated backup and restore process

Level of effort for the Implementation Plan: Moderate to high depending on the complexity of the validation criteria.

Resources

Related documents:

- Automate data recovery validation with AWS Backup
- APN Partner: partners that can help with backup
- AWS Marketplace: products that can be used for backup
- Creating an EventBridge Rule That Triggers on a Schedule
- On-demand backup and restore for DynamoDB
- What Is AWS Backup?
- What Is AWS Step Functions?
- What is AWS Elastic Disaster Recovery
- AWS Elastic Disaster Recovery

Related examples:

- Well-Architected lab: Testing Backup and Restore of Data

REL 10. How do you use fault isolation to protect your workload?

Fault isolated boundaries limit the effect of a failure within a workload to a limited number of components. Components outside of the boundary are unaffected by the failure. Using multiple fault isolated boundaries, you can limit the impact on your workload.

Best practices

- REL10-BP01 Deploy the workload to multiple locations
- REL10-BP02 Select the appropriate locations for your multi-location deployment
- REL10-BP03 Automate recovery for components constrained to a single location
- REL10-BP04 Use bulkhead architectures to limit scope of impact

REL10-BP01 Deploy the workload to multiple locations

Distribute workload data and resources across multiple Availability Zones or, where necessary, across AWS Regions. These locations can be as diverse as required.

One of the bedrock principles for service design in AWS is the avoidance of single points of failure in underlying physical infrastructure. This motivates us to build software and systems that use multiple Availability Zones and are resilient to failure of a single zone. Similarly, systems are built to be resilient to failure of a single compute node, single storage volume, or single instance of a database. When building a system that relies on redundant components, it's important to ensure that the components operate independently, and in the case of AWS Regions, autonomously. The benefits achieved from theoretical availability calculations with redundant components are only valid if this holds true.

Availability Zones (AZs)

AWS Regions are composed of multiple Availability Zones that are designed to be independent of each other. Each Availability Zone is separated by a meaningful physical distance from other zones to avoid correlated failure scenarios due to environmental hazards like fires, floods, and tornadoes. Each Availability Zone also has independent physical infrastructure: dedicated connections to utility power, standalone backup power sources, independent mechanical services, and independent network connectivity within and beyond the Availability Zone. This design limits faults in any of these systems to just the one affected AZ. Despite being geographically separated, Availability Zones are located in the same regional area which allows high-throughput, low-latency networking. The entire AWS Region (across all Availability Zones, consisting of multiple physically

independent data centers) can be treated as a single logical deployment target for your workload, including the ability to synchronously replicate data (for example, between databases). This allows you to use Availability Zones in an active/active or active/standby configuration.

Availability Zones are independent, and therefore workload availability is increased when the workload is architected to use multiple zones. Some AWS services (including the Amazon EC2 instance data plane) are deployed as strictly zonal services where they have shared fate with the Availability Zone they are in. Amazon EC2 instances in the other AZs will however be unaffected and continue to function. Similarly, if a failure in an Availability Zone causes an Amazon Aurora database to fail, a read-replica Aurora instance in an unaffected AZ can be automatically promoted to primary. Regional AWS services, such as Amazon DynamoDB on the other hand internally use multiple Availability Zones in an active/active configuration to achieve the availability design goals for that service, without you needing to configure AZ placement.

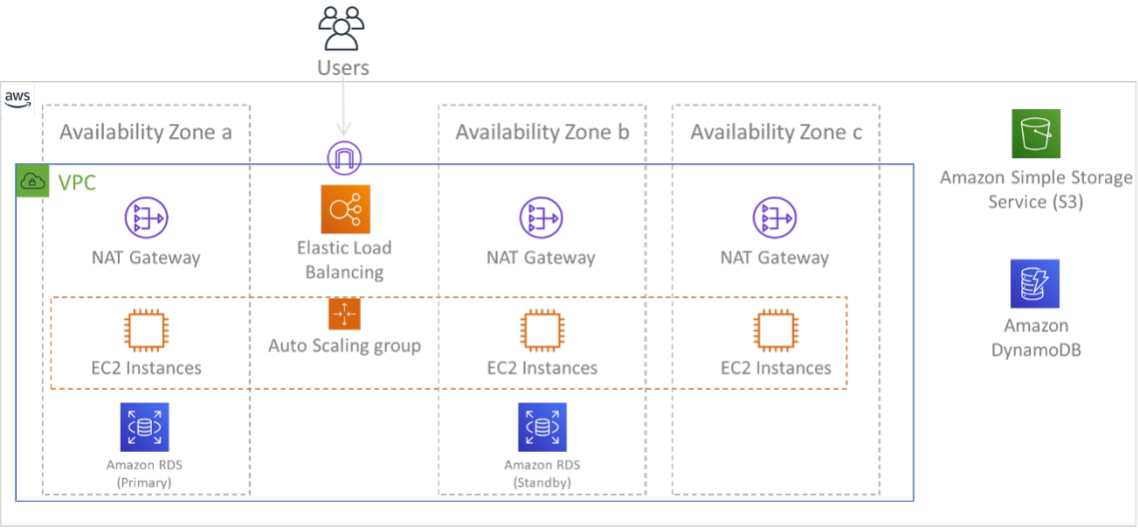


Figure 9: Multi-tier architecture deployed across three Availability Zones. Note that Amazon S3 and Amazon DynamoDB are always Multi-AZ automatically. The ELB also is deployed to all three zones.

While AWS control planes typically provide the ability to manage resources within the entire Region (multiple Availability Zones), certain control planes (including Amazon EC2 and Amazon EBS) have the ability to filter results to a single Availability Zone. When this is done, the request is processed only in the specified Availability Zone, reducing exposure to disruption in other Availability Zones. This AWS CLI example illustrates getting Amazon EC2 instance information from only the us-east-2c Availability Zone:

```
AWS ec2 describe-instances --filters Name=availability-zone,Values=us-east-2c
```

AWS Local Zones

AWS Local Zones act similarly to Availability Zones within their respective AWS Region in that they can be selected as a placement location for zonal AWS resources such as subnets and EC2 instances. What makes them special is that they are located not in the associated AWS Region, but near large population, industry, and IT centers where no AWS Region exists today. Yet they still retain high-bandwidth, secure connection between local workloads in the local zone and those running in the AWS Region. You should use AWS Local Zones to deploy workloads closer to your users for low-latency requirements.

Amazon Global Edge Network

Amazon Global Edge Network consists of edge locations in cities around the world. Amazon CloudFront uses this network to deliver content to end users with lower latency. AWS Global Accelerator allows you to create your workload endpoints in these edge locations to provide onboarding to the AWS global network close to your users. Amazon API Gateway allows edge-optimized API endpoints using a CloudFront distribution to facilitate client access through the closest edge location.

AWS Regions

AWS Regions are designed to be autonomous, therefore, to use a multi-Region approach you would deploy dedicated copies of services to each Region.

A multi-Region approach is common for *disaster recovery* strategies to meet recovery objectives when one-off large-scale events occur. See Plan for Disaster Recovery (DR) for more information on these strategies. Here however, we focus instead on *availability*, which seeks to deliver a mean uptime objective over time. For high-availability objectives, a multi-region architecture will generally be designed to be active/active, where each service copy (in their respective regions) is active (serving requests).

> **Recommendation**
>
> Availability goals for most workloads can be satisfied using a Multi-AZ strategy within a single AWS Region. Consider multi-Region architectures only when workloads have extreme availability requirements, or other business goals, that require a multi-Region architecture.

AWS provides you with the capabilities to operate services cross-region. For example, AWS provides continuous, asynchronous data replication of data using Amazon Simple Storage Service

(Amazon S3) Replication, Amazon RDS Read Replicas (including Aurora Read Replicas), and Amazon DynamoDB Global Tables. With continuous replication, versions of your data are available for near immediate use in each of your active Regions.

Using AWS CloudFormation, you can define your infrastructure and deploy it consistently across AWS accounts and across AWS Regions. And AWS CloudFormation StackSets extends this functionality by allowing you to create, update, or delete AWS CloudFormation stacks across multiple accounts and regions with a single operation. For Amazon EC2 instance deployments, an AMI (Amazon Machine Image) is used to supply information such as hardware configuration and installed software. You can implement an Amazon EC2 Image Builder pipeline that creates the AMIs you need and copy these to your active regions. This ensures that these *Golden AMIs* have everything you need to deploy and scale-out your workload in each new region.

To route traffic, both Amazon Route 53 and AWS Global Accelerator permit the definition of policies that determine which users go to which active regional endpoint. With Global Accelerator you set a traffic dial to control the percentage of traffic that is directed to each application endpoint. Route 53 supports this percentage approach, and also multiple other available policies including geoproximity and latency based ones. Global Accelerator automatically leverages the extensive network of AWS edge servers, to onboard traffic to the AWS network backbone as soon as possible, resulting in lower request latencies.

All of these capabilities operate so as to preserve each Region's autonomy. There are very few exceptions to this approach, including our services that provide global edge delivery (such as Amazon CloudFront and Amazon Route 53), along with the control plane for the AWS Identity and Access Management (IAM) service. Most services operate entirely within a single Region.

On-premises data center

For workloads that run in an on-premises data center, architect a hybrid experience when possible. AWS Direct Connect provides a dedicated network connection from your premises to AWS allowing you to run in both.

Another option is to run AWS infrastructure and services on premises using AWS Outposts. AWS Outposts is a fully managed service that extends AWS infrastructure, AWS services, APIs, and tools to your data center. The same hardware infrastructure used in the AWS Cloud is installed in your data center. AWS Outposts are then connected to the nearest AWS Region. You can then use AWS Outposts to support your workloads that have low latency or local data processing requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Use multiple Availability Zones and AWS Regions. Distribute workload data and resources across multiple Availability Zones or, where necessary, across AWS Regions. These locations can be as diverse as required.
  - Regional services are inherently deployed across Availability Zones.
    - This includes Amazon S3, Amazon DynamoDB, and AWS Lambda (when not connected to a VPC)
  - Deploy your container, instance, and function-based workloads into multiple Availability Zones. Use multi-zone datastores, including caches. Use the features of EC2 Auto Scaling, ECS task placement, AWS Lambda function configuration when running in your VPC, and ElastiCache clusters.
    - Use subnets that are in separate Availability Zones when you deploy Auto Scaling groups.
      - Example: Distributing instances across Availability Zones
      - Amazon ECS task placement strategies
      - Configuring an AWS Lambda function to access resources in an Amazon VPC
      - Choosing Regions and Availability Zones
    - Use subnets in separate Availability Zones when you deploy Auto Scaling groups.
      - Example: Distributing instances across Availability Zones
    - Use ECS task placement parameters, specifying DB subnet groups.
      - Amazon ECS task placement strategies
    - Use subnets in multiple Availability Zones when you configure a function to run in your VPC.
      - Configuring an AWS Lambda function to access resources in an Amazon VPC
    - Use multiple Availability Zones with ElastiCache clusters.
      - Choosing Regions and Availability Zones
- If your workload must be deployed to multiple Regions, choose a multi-Region strategy. Most reliability needs can be met within a single AWS Region using a multi-Availability Zone strategy. Use a multi-Region strategy when necessary to meet your business needs.
  - AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
    - Backup to another AWS Region can add another layer of assurance that data will be available when needed.
    - Some workloads have regulatory requirements that require use of a multi-Region strategy.

- Evaluate AWS Outposts for your workload. If your workload requires low latency to your on-premises data center or has local data processing requirements. Then run AWS infrastructure and services on premises using AWS Outposts
  - What is AWS Outposts?
- Determine if AWS Local Zones helps you provide service to your users. If you have low-latency requirements, see if AWS Local Zones is located near your users. If yes, then use it to deploy workloads closer to those users.
  - AWS Local Zones FAQ

Resources

Related documents:

- AWS Global Infrastructure
- AWS Local Zones FAQ
- Amazon ECS task placement strategies
- Choosing Regions and Availability Zones
- Example: Distributing instances across Availability Zones
- Global Tables: Multi-Region Replication with DynamoDB
- Using Amazon Aurora global databases
- Creating a Multi-Region Application with AWS Services blog series
- What is AWS Outposts?

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure (NET339)

REL10-BP02 Select the appropriate locations for your multi-location deployment

Desired Outcome

For high availability, always (when possible) deploy your workload components to multiple Availability Zones (AZs), as shown in Figure 10. For workloads with extreme resilience requirements, carefully evaluate the options for a multi-Region architecture.

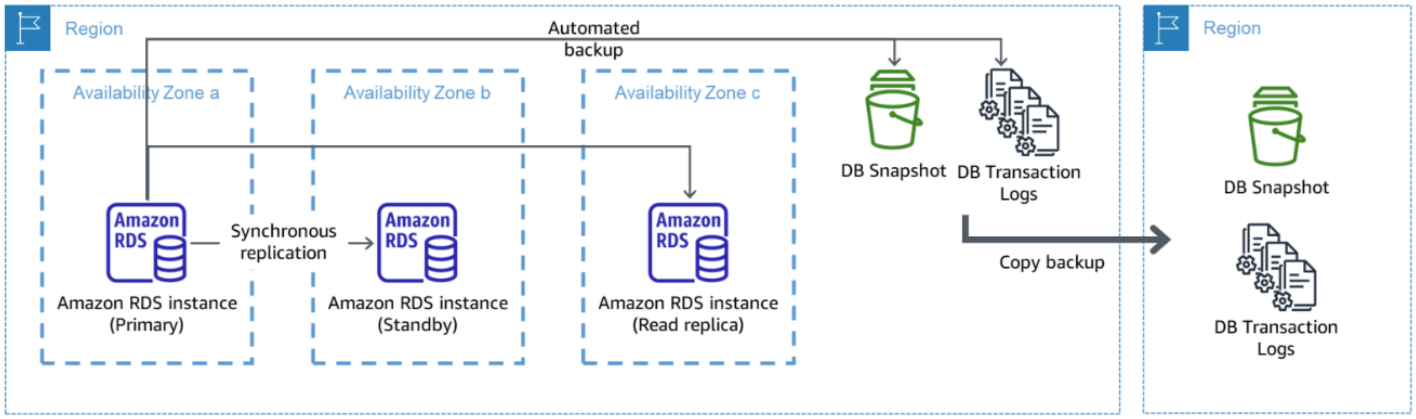


Figure 10: A resilient multi-AZ database deployment with backup to another AWS Region

Common anti-patterns

- Choosing to design a multi-Region architecture when a multi-AZ architecture would satisfy requirements.
- Not accounting for dependencies between application components if resilience and multi-location requirements differ between those components.

Benefits of establishing this best practice

For resilience, you should use an approach that builds layers of defense. One layer protects against smaller, more common, disruptions by building a highly available architecture using multiple AZs. Another layer of defense is meant to protect against rare events like widespread natural disasters and Region-level disruptions. This second layer involves architecting your application to span multiple AWS Regions.

- The difference between a 99.5% availability and 99.99% availability is over 3.5 hours per month. The expected availability of a workload can only reach “four nines” if it is in multiple AZs.
- By running your workload in multiple AZs, you can isolate faults in power, cooling, and networking, and most natural disasters like fire and flood.

- Implementing a multi-Region strategy for your workload helps protect it against widespread natural disasters that affect a large geographic region of a country, or technical failures of Region-wide scope. Be aware that implementing a multi-Region architecture can be significantly complex, and is usually not required for most workloads.

Level of risk exposed if this best practice is not established: High

Implementation guidance

For a disaster event based on disruption or partial loss of one Availability Zone, implementing a highly available workload in multiple Availability Zones within a single AWS Region helps mitigate against natural and technical disasters. Each AWS Region is comprised of multiple Availability Zones, each isolated from faults in the other zones and separated by a meaningful distance. However, for a disaster event that includes the risk of losing multiple Availability Zone components, which are a significant distance away from each other, you should implement disaster recovery options to mitigate against failures of a Region-wide scope. For workloads that require extreme resilience (critical infrastructure, health-related applications, financial system infrastructure, etc.), a multi-Region strategy may be required.

Implementation Steps

1. Evaluate your workload and determine whether the resilience needs can be met by a multi-AZ approach (single AWS Region), or if they require a multi-Region approach. Implementing a multi-Region architecture to satisfy these requirements will introduce additional complexity, therefore carefully consider your use case and its requirements. Resilience requirements can almost always be met using a single AWS Region. Consider the following possible requirements when determining whether you need to use multiple Regions:
    a. **Disaster recovery (DR)**: For a disaster event based on disruption or partial loss of one Availability Zone, implementing a highly available workload in multiple Availability Zones within a single AWS Region helps mitigate against natural and technical disasters. For a disaster event that includes the risk of losing multiple Availability Zone components, which are a significant distance away from each other, you should implement disaster recovery across multiple Regions to mitigate against natural disasters or technical failures of a Region-wide scope.
    b. **High availability (HA)**: A multi-Region architecture (using multiple AZs in each Region) can be used to achieve greater then four 9’s (> 99.99%) availability.

c. **Stack localization**: When deploying a workload to a global audience, you can deploy localized stacks in different AWS Regions to serve audiences in those Regions. Localization can include language, currency, and types of data stored.

d. **Proximity to users:** When deploying a workload to a global audience, you can reduce latency by deploying stacks in AWS Regions close to where the end users are.

e. **Data residency**: Some workloads are subject to data residency requirements, where data from certain users must remain within a specific country's borders. Based on the regulation in question, you can choose to deploy an entire stack, or just the data, to the AWS Region within those borders.

2. Here are some examples of multi-AZ functionality provided by AWS services:

   a. To protect workloads using EC2 or ECS, deploy an Elastic Load Balancer in front of the compute resources. Elastic Load Balancing then provides the solution to detect instances in unhealthy zones and route traffic to the healthy ones.

      i. Getting started with Application Load Balancers

      ii. Getting started with Network Load Balancers

   b. In the case of EC2 instances running commercial off-the-shelf software that do not support load balancing, you can achieve a form of fault tolerance by implementing a multi-AZ disaster recovery methodology.

      i. the section called "REL13-BP02 Use defined recovery strategies to meet the recovery objectives"

   c. For Amazon ECS tasks, deploy your service evenly across three AZs to achieve a balance of availability and cost.

      i. Amazon ECS availability best practices | Containers

   d. For non-Aurora Amazon RDS, you can choose Multi-AZ as a configuration option. Upon failure of the primary database instance, Amazon RDS automatically promotes a standby database to receive traffic in another availability zone. Multi-Region read-replicas can also be created to improve resilience.

      i. Amazon RDS Multi AZ Deployments

      ii. Creating a read replica in a different AWS Region

3. Here are some examples of multi-Region functionality provided by AWS services:

   a. For Amazon S3 workloads, where multi-AZ availability is provided automatically by the service, consider Multi-Region Access Points if a multi-Region deployment is needed.

      i. Multi-Region Access Points in Amazon S3

b. For DynamoDB tables, where multi-AZ availability is provided automatically by the service, you can easily convert existing tables to global tables to take advantage of multiple regions.
   i. Convert Your Single-Region Amazon DynamoDB Tables to Global Tables
c. If your workload is fronted by Application Load Balancers or Network Load Balancers, use AWS Global Accelerator to improve the availability of your application by directing traffic to multiple regions that contain healthy endpoints.
   i. Endpoints for standard accelerators in AWS Global Accelerator - AWS Global Accelerator (amazon.com)
d. For applications that leverage AWS EventBridge, consider cross-Region buses to forward events to other Regions you select.
   i. Sending and receiving Amazon EventBridge events between AWS Regions
e. For Amazon Aurora databases, consider Aurora global databases, which span multiple AWS regions. Existing clusters can be modified to add new Regions as well.
   i. Getting started with Amazon Aurora global databases
f. If your workload includes AWS Key Management Service (AWS KMS) encryption keys, consider whether multi-Region keys are appropriate for your application.
   i. Multi-Region keys in AWS KMS
g. For other AWS service features, see this blog series on Creating a Multi-Region Application with AWS Services series

Level of effort for the Implementation Plan: Moderate to High

Resources

Related documents:

- Creating a Multi-Region Application with AWS Services series
- Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active
- AWS Global Infrastructure
- AWS Local Zones FAQ
- Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud
- Disaster recovery is different in the cloud
- Global Tables: Multi-Region Replication with DynamoDB

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Auth0: Multi-Region High-Availability Architecture that Scales to 1.5B+ Logins a Month with automated failover

Related examples:

- Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud
- DTCC achieves resilience well beyond what they can do on premises
- Expedia Group uses a multi-Region, multi-Availability Zone architecture with a proprietary DNS service to add resilience to the applications
- Uber: Disaster Recovery for Multi-Region Kafka
- Netflix: Active-Active for Multi-Regional Resilience
- How we build Data Residency for Atlassian Cloud
- Intuit TurboTax runs across two Regions

REL10-BP03 Automate recovery for components constrained to a single location

If components of the workload can only run in a single Availability Zone or in an on-premises data center, implement the capability to do a complete rebuild of the workload within your defined recovery objectives.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

If the best practice to deploy the workload to multiple locations is not possible due to technological constraints, you must implement an alternate path to resiliency. You must automate the ability to recreate necessary infrastructure, redeploy applications, and recreate necessary data for these cases.

For example, Amazon EMR launches all nodes for a given cluster in the same Availability Zone because running a cluster in the same zone improves performance of the jobs flows as it provides a higher data access rate. If this component is required for workload resilience, then you must have a way to redeploy the cluster and its data. Also for Amazon EMR, you should provision redundancy

in ways other than using Multi-AZ. You can provision multiple nodes. Using EMR File System (EMRFS), data in EMR can be stored in Amazon S3, which in turn can be replicated across multiple Availability Zones or AWS Regions.

Similarly, for Amazon Redshift, by default it provisions your cluster in a randomly selected Availability Zone within the AWS Region that you select. All the cluster nodes are provisioned in the same zone.

For stateful server-based workloads deployed to an on-premise data center, you can use AWS Elastic Disaster Recovery to protect your workloads in AWS. If you are already hosted in AWS, you can use Elastic Disaster Recovery to protect your workload to an alternative Availability Zone or Region. Elastic Disaster Recovery uses continual block-level replication to a lightweight staging area to provide fast, reliable recovery of on-premises and cloud-based applications.

Implementation steps

1. Implement self-healing. Deploy your instances or containers using automatic scaling when possible. If you cannot use automatic scaling, use automatic recovery for EC2 instances or implement self-healing automation based on Amazon EC2 or ECS container lifecycle events.
    - Use Amazon EC2 Auto Scaling groups for instances and container workloads that have no requirements for a single instance IP address, private IP address, Elastic IP address, and instance metadata.
        - The launch template user data can be used to implement automation that can self-heal most workloads.
    - Use automatic recovery of Amazon EC2 instances for workloads that require a single instance ID address, private IP address, elastic IP address, and instance metadata.
        - Automatic Recovery will send recovery status alerts to a SNS topic as the instance failure is detected.
    - Use Amazon EC2 instance lifecycle events or Amazon ECS events to automate self-healing where automatic scaling or EC2 recovery cannot be used.
        - Use the events to invoke automation that will heal your component according to the process logic you require.
    - Protect stateful workloads that are limited to a single location using AWS Elastic Disaster Recovery.

Resources

Related documents:

- Amazon ECS events
- Amazon EC2 Auto Scaling lifecycle hooks
- Recover your instance.
- Service automatic scaling
- What Is Amazon EC2 Auto Scaling?
- AWS Elastic Disaster Recovery

REL10-BP04 Use bulkhead architectures to limit scope of impact

Implement bulkhead architectures (also known as cell-based architectures) to restrict the effect of failure within a workload to a limited number of components.

Desired outcome: A cell-based architecture uses multiple isolated instances of a workload, where each instance is known as a cell. Each cell is independent, does not share state with other cells, and handles a subset of the overall workload requests. This reduces the potential impact of a failure, such as a bad software update, to an individual cell and the requests it is processing. If a workload uses 10 cells to service 100 requests, when a failure occurs, 90% of the overall requests would be unaffected by the failure.

Common anti-patterns:

- Allowing cells to grow without bounds.
- Applying code updates or deployments to all cells at the same time.
- Sharing state or components between cells (with the exception of the router layer).
- Adding complex business or routing logic to the router layer.
- Not minimizing cross-cell interactions.

Benefits of establishing this best practice: With cell-based architectures, many common types of failure are contained within the cell itself, providing additional fault isolation. These fault boundaries can provide resilience against failure types that otherwise are hard to contain, such as unsuccessful code deployments or requests that are corrupted or invoke a specific failure mode (also known as *poison pill requests*).

Implementation guidance

On a ship, bulkheads ensure that a hull breach is contained within one section of the hull. In complex systems, this pattern is often replicated to allow fault isolation. Fault isolated boundaries restrict the effect of a failure within a workload to a limited number of components. Components outside of the boundary are unaffected by the failure. Using multiple fault isolated boundaries, you can limit the impact on your workload. On AWS, customers can use multiple Availability Zones and Regions to provide fault isolation, but the concept of fault isolation can be extended to your workload's architecture as well.

The overall workload is partitioned cells by a partition key. This key needs to align with the *grain* of the service, or the natural way that a service's workload can be subdivided with minimal cross-cell interactions. Examples of partition keys are customer ID, resource ID, or any other parameter easily accessible in most API calls. A cell routing layer distributes requests to individual cells based on the partition key and presents a single endpoint to clients.

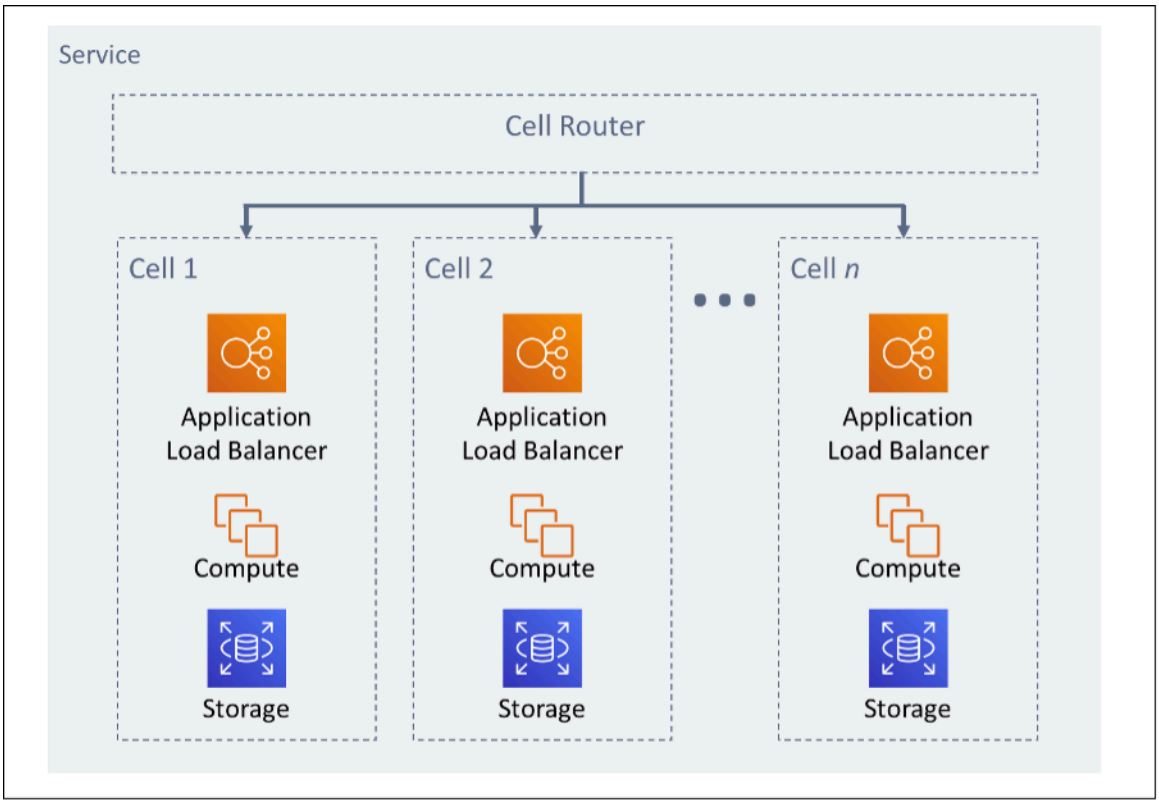


Figure 11: Cell-based architecture

Implementation steps

When designing a cell-based architecture, there are several design considerations to consider:

1. **Partition key**: Special consideration should be taken while choosing the partition key.
    - It should align with the grain of the service, or the natural way that a service's workload can be subdivided with minimal cross-cell interactions. Examples are `customer ID` or `resource ID`.
    - The partition key must be available in all requests, either directly or in a way that could be easily inferred deterministically by other parameters.
2. **Persistent cell mapping**: Upstream services should only interact with a single cell for the lifecycle of their resources.
    - Depending on the workload, a cell migration strategy may be needed to migrate data from one cell to another. A possible scenario when a cell migration may be needed is if a particular user or resource in your workload becomes too big and requires it to have a dedicated cell.
    - Cells should not share state or components between cells.
    - Consequently, cross-cell interactions should be avoided or kept to a minimum, as those interactions create dependencies between cells and therefore diminish the fault isolation improvements.
3. **Router layer**: The router layer is a shared component between cells, and therefore cannot follow the same compartmentalization strategy as with cells.
    - It is recommended for the router layer to distribute requests to individual cells using a partition mapping algorithm in a computationally efficient manner, such as combining cryptographic hash functions and modular arithmetic to map partition keys to cells.
    - To avoid multi-cell impacts, the routing layer must remain as simple and horizontally scalable as possible, which necessitates avoiding complex business logic within this layer. This has the added benefit of making it easy to understand its expected behavior at all times, allowing for thorough testability. As explained by Colm MacCárthaigh in Reliability, constant work, and a good cup of coffee, simple designs and constant work patterns produce reliable systems and reduce anti-fragility.
4. **Cell size**: Cells should have a maximum size and should not be allowed to grow beyond it.
    - The maximum size should be identified by performing thorough testing, until breaking points are reached and safe operating margins are established. For more detail on how to implement testing practices, see REL07-BP04 Load test your workload
    - The overall workload should grow by adding additional cells, allowing the workload to scale with increases in demand.
5. **Multi-AZ or Multi-Region strategies**: Multiple layers of resilience should be leveraged to protect against different failure domains.

- For resilience, you should use an approach that builds layers of defense. One layer protects against smaller, more common disruptions by building a highly available architecture using multiple AZs. Another layer of defense is meant to protect against rare events like widespread natural disasters and Region-level disruptions. This second layer involves architecting your application to span multiple AWS Regions. Implementing a multi-Region strategy for your workload helps protect it against widespread natural disasters that affect a large geographic region of a country, or technical failures of Region-wide scope. Be aware that implementing a multi-Region architecture can be significantly complex, and is usually not required for most workloads. For more detail, see REL10-BP02 Select the appropriate locations for your multi-location deployment.

6. **Code deployment**: A staggered code deployment strategy should be preferred over deploying code changes to all cells at the same time.
    - This will help minimize potential failure to multiple cells due to a bad deployment or human error. For more detail, see Automating safe, hands-off deployment.

Level of risk exposed if this best practice is not established: High

Resources

Related best practices:

- REL07-BP04 Load test your workload
- REL10-BP02 Select the appropriate locations for your multi-location deployment

Related documents:

- Reliability, constant work, and a good cup of coffee
- AWS and Compartmentalization
- Workload isolation using shuffle-sharding
- Automating safe, hands-off deployment

Related videos:

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small
- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338)

- Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)
- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience

Related examples:

- Well-Architected Lab - Fault isolation with shuffle sharding

REL 11. How do you design your workload to withstand component failures?

Workloads with a requirement for high availability and low mean time to recovery (MTTR) must be architected for resiliency.

Best practices

- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP02 Fail over to healthy resources
- REL11-BP03 Automate healing on all layers
- REL11-BP04 Rely on the data plane and not the control plane during recovery
- REL11-BP05 Use static stability to prevent bimodal behavior
- REL11-BP06 Send notifications when events impact availability
- REL11-BP07 Architect your product to meet availability targets and uptime service level agreements (SLAs)

REL11-BP01 Monitor all components of the workload to detect failures

Continuously monitor the health of your workload so that you and your automated systems are aware of degradation or failure as soon as they occur. Monitor for key performance indicators (KPIs) based on business value.

All recovery and healing mechanisms must start with the ability to detect problems quickly. Technical failures should be detected first so that they can be resolved. However, availability is based on the ability of your workload to deliver business value, so key performance indicators (KPIs) that measure this need to be a part of your detection and remediation strategy.

Common anti-patterns:

- No alarms have been configured, so outages occur without notification.

- Alarms exist, but at thresholds that don't provide adequate time to react.
- Metrics are not collected often enough to meet the recovery time objective (RTO).
- Only the customer facing tier of the workload is actively monitored.
- Only collecting technical metrics, no business function metrics.
- No metrics measuring the user experience of the workload.

Benefits of establishing this best practice: Having appropriate monitoring at all layers allows you to reduce recovery time by reducing time to detection.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Determine the collection interval for your components based on your recovery goals.
  - Your monitoring interval is dependent on how quickly you must recover. Your recovery time is driven by the time it takes to recover, so you must determine the frequency of collection by accounting for this time and your recovery time objective (RTO).
- Configure detailed monitoring for components.
  - Determine if detailed monitoring for EC2 instances and Auto Scaling is necessary. Detailed monitoring provides 1-min interval metrics, and default monitoring provides 5-minute interval metrics.
    - Enable or Disable Detailed Monitoring for Your Instance
    - Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch
  - Determine if enhanced monitoring for RDS is necessary. Enhanced monitoring uses an agent on the RDS instances to get useful information about different process or threads on an RDS instance.
    - Enhanced Monitoring
- Create custom metrics to measure business key performance indicators (KPIs). Workloads implement key business functions. These functions should be used as KPIs that help identify when an indirect problem happens.
  - Publishing Custom Metrics
- Monitor the user experience for failures using user canaries. Synthetic transaction testing (also known as canary testing, but not to be confused with canary deployments) that can run and simulate customer behavior is among the most important testing processes. Run these tests constantly against your workload endpoints from diverse remote locations.

  - Amazon CloudWatch Synthetics allows you to create user canaries
- Create custom metrics that track the user's experience. If you can instrument the experience of the customer, you can determine when the consumer experience degrades.
  - Publishing Custom Metrics
- Set alarms to detect when any part of your workload is not working properly, and to indicate when to Auto Scale resources. Alarms can be visually displayed on dashboards, send alerts via Amazon SNS or email, and work with Auto Scaling to scale up or down the resources for a workload.
  - Using Amazon CloudWatch Alarms
- Create dashboards to visualize your metrics. Dashboards can be used to visually see trends, outliers, and other indicators of potential problems, or to provide an indication of problems you may want to investigate.
  - Using CloudWatch Dashboards

Resources

Related documents:

- Amazon CloudWatch Synthetics enables you to create user canaries
- Enable or Disable Detailed Monitoring for Your Instance
- Enhanced Monitoring
- Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch
- Publishing Custom Metrics
- Using Amazon CloudWatch Alarms
- Using CloudWatch Dashboards

Related examples:

- Well-Architected lab: Level 300: Implementing Health Checks and Managing Dependencies to Improve Reliability

REL11-BP02 Fail over to healthy resources

Ensure that if a resource failure occurs, that healthy resources can continue to serve requests. For location failures (such as Availability Zone or AWS Region) ensure that you have systems in place to fail over to healthy resources in unimpaired locations.

AWS services, such as Elastic Load Balancing and Amazon EC2 Auto Scaling, help distribute load across resources and Availability Zones. Therefore, failure of an individual resource (such as an EC2 instance) or impairment of an Availability Zone can be mitigated by shifting traffic to remaining healthy resources. For multi-region workloads, this is more complicated. For example, cross-region read replicas allow you to deploy your data to multiple AWS Regions, but you still must promote the read replica to primary and point your traffic at it in the event of a failover. Amazon Route 53 and AWS Global Accelerator can help route traffic across AWS Regions.

If your workload is using AWS services, such as Amazon S3 or Amazon DynamoDB, then they are automatically deployed to multiple Availability Zones. In case of failure, the AWS control plane automatically routes traffic to healthy locations for you. Data is redundantly stored in multiple Availability Zones, and remains available. For Amazon RDS, you must choose Multi-AZ as a configuration option, and then on failure AWS automatically directs traffic to the healthy instance. For Amazon EC2 instances, Amazon ECS tasks, or Amazon EKS pods, you choose which Availability Zones to deploy to. Elastic Load Balancing then provides the solution to detect instances in unhealthy zones and route traffic to the healthy ones. Elastic Load Balancing can even route traffic to components in your on-premises data center.

For Multi-Region approaches (which might also include on-premises data centers), Amazon Route 53 provides a way to define internet domains, and assign routing policies that can include health checks to ensure that traffic is routed to healthy regions. Alternately, AWS Global Accelerator provides static IP addresses that act as a fixed entry point to your application, then routes to endpoints in AWS Regions of your choosing, using the AWS global network instead of the internet for better performance and reliability.

AWS approaches the design of our services with fault recovery in mind. We design services to minimize the time to recover from failures and impact on data. Our services primarily use data stores that acknowledge requests only after they are durably stored across multiple replicas within a Region. These services and resources include Amazon Aurora, Amazon Relational Database Service (Amazon RDS) Multi-AZ DB instances, Amazon S3, Amazon DynamoDB, Amazon Simple Queue Service (Amazon SQS), and Amazon Elastic File System (Amazon EFS). They are constructed to use cell-based isolation and use the fault isolation provided by Availability Zones. We use

automation extensively in our operational procedures. We also optimize our replace-and-restart functionality to recover quickly from interruptions.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Fail over to healthy resources. Ensure that if a resource failure occurs, that healthy resources can continue to serve requests. For location failures (such as Availability Zone or AWS Region) ensure you have systems in place to fail over to healthy resources in unimpaired locations.
  - If your workload is using AWS services, such as Amazon S3 or Amazon DynamoDB, then they are automatically deployed to multiple Availability Zones. In case of failure, the AWS control plane automatically routes traffic to healthy locations for you.
  - For Amazon RDS you must choose Multi-AZ as a configuration option, and then on failure AWS automatically directs traffic to the healthy instance.
    - High Availability (Multi-AZ) for Amazon RDS
  - For Amazon EC2 instances or Amazon ECS tasks, you choose which Availability Zones to deploy to. Elastic Load Balancing then provides the solution to detect instances in unhealthy zones and route traffic to the healthy ones. Elastic Load Balancing can even route traffic to components in your on-premises data center.
  - For multi-region approaches (which might also include on-premises data centers), ensure that data and resources from healthy locations can continue to serve requests
    - For example, cross-region read replicas allow you to deploy your data to multiple AWS Regions, but you still must promote the read replica to master and point your traffic at it in the event of a primary location failure.
      - Overview of Amazon RDS Read Replicas
    - Amazon Route 53 provides a way to define internet domains, and assign routing policies, which might include health checks, to ensure that traffic is routed to healthy Regions. Alternately, AWS Global Accelerator provides static IP addresses that act as a fixed entry point to your application, then routes to endpoints in AWS Regions of your choosing, using the AWS global network instead of the public internet for better performance and reliability.
      - Amazon Route 53: Choosing a Routing Policy
      - What Is AWS Global Accelerator?

Resources

Related documents:

- APN Partner: partners that can help with automation of your fault tolerance
- AWS Marketplace: products that can be used for fault tolerance
- AWS OpsWorks: Using Auto Healing to Replace Failed Instances
- Amazon Route 53: Choosing a Routing Policy
- High Availability (Multi-AZ) for Amazon RDS
- Overview of Amazon RDS Read Replicas
- Amazon ECS task placement strategies
- Creating Kubernetes Auto Scaling Groups for Multiple Availability Zones
- What is AWS Global Accelerator?

Related examples:

- Well-Architected lab: Level 300: Implementing Health Checks and Managing Dependencies to Improve Reliability

REL11-BP03 Automate healing on all layers

Upon detection of a failure, use automated capabilities to perform actions to remediate.

Ability to restart is an important tool to remediate failures. As discussed previously for distributed systems, a best practice is to make services stateless where possible. This prevents loss of data or availability on restart. In the cloud, you can (and generally should) replace the entire resource (for example, EC2 instance, or Lambda function) as part of the restart. The restart itself is a simple and reliable way to recover from failure. Many different types of failures occur in workloads. Failures can occur in hardware, software, communications, and operations. Rather than constructing novel mechanisms to trap, identify, and correct each of the different types of failures, map many different categories of failures to the same recovery strategy. An instance might fail due to hardware failure, an operating system bug, memory leak, or other causes. Rather than building custom remediation for each situation, treat any of them as an instance failure. Terminate the instance, and allow AWS Auto Scaling to replace it. Later, carry out the analysis on the failed resource out of band.

Another example is the ability to restart a network request. Apply the same recovery approach to both a network timeout and a dependency failure where the dependency returns an error. Both events have a similar effect on the system, so rather than attempting to make either event a "special case", apply a similar strategy of limited retry with exponential backoff and jitter.

Ability to restart is a recovery mechanism featured in Recovery Oriented Computing and high availability cluster architectures.

Amazon EventBridge can be used to monitor and filter for events such as CloudWatch Alarms or changes in state in other AWS services. Based on event information, it can then invoke AWS Lambda, AWS Systems Manager Automation, or other targets to run custom remediation logic on your workload.

Amazon EC2 Auto Scaling can be configured to check for EC2 instance health. If the instance is in any state other than running, or if the system status is impaired, Amazon EC2 Auto Scaling considers the instance to be unhealthy and launches a replacement instance. If using AWS OpsWorks, you can configure Auto Healing of EC2 instances at the OpsWorks layer level.

For large-scale replacements (such as the loss of an entire Availability Zone), static stability is preferred for high availability instead of trying to obtain multiple new resources at once.

Common anti-patterns:

- Deploying applications in instances or containers individually.
- Deploying applications that cannot be deployed into multiple locations without using automatic recovery.
- Manually healing applications that automatic scaling and automatic recovery fail to heal.

Benefits of establishing this best practice: Automated healing, even if the workload can only deployed into one location at a time will reduce your mean time to recovery, and ensure availability of the workload.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Use Auto Scaling groups to deploy tiers in an workload. Auto scaling can perform self-healing on stateless applications, and add and remove capacity.
  - How AWS Auto Scaling Works

- Implement automatic recovery on EC2 instances that have applications deployed that cannot be deployed in multiple locations, and can tolerate rebooting upon failures. Automatic recovery can be used to replace failed hardware and restart the instance when the application is not capable of being deployed in multiple locations. The instance metadata and associated IP addresses are kept, as well as the Amazon EBS volumes and mount points to Elastic File Systems or File Systems for Lustre and Windows.
  - Amazon EC2 Automatic Recovery
  - Amazon Elastic Block Store (Amazon EBS)
  - Amazon Elastic File System (Amazon EFS)
  - What is Amazon FSx for Lustre?
  - What is Amazon FSx for Windows File Server?
    - Using AWS OpsWorks, you can configure Auto Healing of EC2 instances at the layer level
      - AWS OpsWorks: Using Auto Healing to Replace Failed Instances
- Implement automated recovery using AWS Step Functions and AWS Lambda when you cannot use automatic scaling or automatic recovery, or when automatic recovery fails. When you cannot use automatic scaling, and either cannot use automatic recovery or automatic recovery fails, you can automate the healing using AWS Step Functions and AWS Lambda.
  - What is AWS Step Functions?
  - What is AWS Lambda?
    - Amazon EventBridge can be used to monitor and filter for events such as CloudWatch Alarms or changes in state in other AWS services. Based on event information, it can then invoke AWS Lambda (or other targets) to run custom remediation logic on your workload.
      - What Is Amazon EventBridge?
      - Using Amazon CloudWatch Alarms

Resources

Related documents:

- APN Partner: partners that can help with automation of your fault tolerance
- AWS Marketplace: products that can be used for fault tolerance
- AWS OpsWorks: Using Auto Healing to Replace Failed Instances
- Amazon EC2 Automatic Recovery
- Amazon Elastic Block Store (Amazon EBS)

- Amazon Elastic File System (Amazon EFS)
- How AWS Auto Scaling Works
- Using Amazon CloudWatch Alarms
- What Is Amazon EventBridge?
- What is AWS Lambda?
- AWS Systems Manager Automation
- What is AWS Step Functions?
- What is Amazon FSx for Lustre?
- What is Amazon FSx for Windows File Server?

Related videos:

- Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

Related examples:

- Well-Architected lab: Level 300: Implementing Health Checks and Managing Dependencies to Improve Reliability

REL11-BP04 Rely on the data plane and not the control plane during recovery

The control plane is used to configure resources, and the data plane delivers services. Data planes typically have higher availability design goals than control planes and are usually less complex. When implementing recovery or mitigation responses to potentially resiliency-impacting events, using control plane operations can lower the overall resiliency of your architecture. For example, you can rely on the Amazon Route 53 data plane to reliably route DNS queries based on health checks, but updating Route 53 routing policies uses the control plane, so do not rely on it for recovery.

The Route 53 data planes answer DNS queries, and perform and evaluate health checks. They are globally distributed and designed for a 100% availability service level agreement (SLA). The Route 53 management APIs and consoles where you create, update, and delete Route 53 resources run on control planes that are designed to prioritize the strong consistency and durability that you need when managing DNS. To achieve this, the control planes are located in a single Region, US East (N. Virginia). While both systems are built to be very reliable, the control planes are not

included in the SLA. There could be rare events in which the data plane's resilient design allows it to maintain availability while the control planes do not. For disaster recovery and failover mechanisms, use data plane functions to provide the best possible reliability.

For more information about data planes, control planes, and how AWS builds services to meet high availability targets, see the Static stability using Availability Zones paper and the Amazon Builders' Library.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Rely on the data plane and not the control plane when using Amazon Route 53 for disaster recovery. Route 53 Application Recovery Controller helps you manage and coordinate failover using readiness checks and routing controls. These features continually monitor your application's ability to recover from failures, and allows you to control your application recovery across multiple AWS Regions, Availability Zones, and on premises.
  - What is Route 53 Application Recovery Controller
  - Creating Disaster Recovery Mechanisms Using Amazon Route 53
  - Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack
  - Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 2: Multi-Region stack
- Understand which operations are on the data plane and which are on the control plane.
  - Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control
  - Amazon DynamoDB API (control plane and data plane)
  - AWS Lambda Executions (split into the control plane and the data plane)
  - AWS Lambda Executions (split into the control plane and the data plane)

Resources

Related documents:

- APN Partner: partners that can help with automation of your fault tolerance
- AWS Marketplace: products that can be used for fault tolerance

- Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control
- Amazon DynamoDB API (control plane and data plane)
- AWS Lambda Executions (split into the control plane and the data plane)
- AWS Elemental MediaStore Data Plane
- Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 2: Multi-Region stack
- Creating Disaster Recovery Mechanisms Using Amazon Route 53
- What is Route 53 Application Recovery Controller

Related examples:

- Introducing Amazon Route 53 Application Recovery Controller

REL11-BP05 Use static stability to prevent bimodal behavior

Bimodal behavior is when your workload exhibits different behavior under normal and failure modes, for example, relying on launching new instances if an Availability Zone fails. You should instead build workloads that are statically stable and operate in only one mode. In this case, provision enough instances in each Availability Zone to handle the workload load if one AZ were removed and then use Elastic Load Balancing or Amazon Route 53 health checks to shift load away from the impaired instances.

Static stability for compute deployment (such as EC2 instances or containers) will result in the highest reliability. This must be weighed against cost concerns. It's less expensive to provision less compute capacity and rely on launching new instances in the case of a failure. But for large-scale failures (such as an Availability Zone failure) this approach is less effective because it relies on reacting to impairments as they happen, rather than being prepared for those impairments before they happen. Your solution should weigh reliability versus the cost needs for your workload. By using more Availability Zones, the amount of additional compute you need for static stability decreases.

Figure 14: Static stability of EC2 instances across Availability Zones

After traffic has shifted, use AWS Auto Scaling to asynchronously replace instances from the failed zone and launch them in the healthy zones.

Another example of bimodal behavior would be a network timeout that could cause a system to attempt to refresh the configuration state of the entire system. This would add unexpected load to another component, and might cause it to fail, resulting in other unexpected consequences. This negative feedback loop impacts availability of your workload. Instead, you should build systems that are statically stable and operate in only one mode. A statically stable design would be to do constant work, and always refresh the configuration state on a fixed cadence. When a call fails, the workload uses the previously cached value, and initiates an alarm.

Another example of bimodal behavior is allowing clients to bypass your workload cache when failures occur. This might seem to be a solution that accommodates client needs, but should not be allowed because it significantly changes the demands on your workload and is likely to result in failures.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Use static stability to prevent bimodal behavior. Bimodal behavior is when your workload exhibits different behavior under normal and failure modes, for example, relying on launching new instances if an Availability Zone fails.
  - Minimizing Dependencies in a Disaster Recovery Plan

- The Amazon Builders' Library: Static stability using Availability Zones
- Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)
  - You should instead build systems that are statically stable and operate in only one mode. In this case, provision enough instances in each zone to handle workload load if one AZ were removed and then use Elastic Load Balancing or Amazon Route 53 health checks to shift load away from the impaired instances.
  - Another example of bimodal behavior is allowing clients to bypass your workload cache when failures occur. This might seem to be a solution to accommodate client needs, but should not be allowed since it significantly changes demands on your workload and is likely to result in failures.

Resources

Related documents:

- Minimizing Dependencies in a Disaster Recovery Plan
- The Amazon Builders' Library: Static stability using Availability Zones

Related videos:

- Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL11-BP06 Send notifications when events impact availability

Notifications are sent upon the detection of significant events, even if the issue caused by the event was automatically resolved.

Automated healing allows your workload to be reliable. However, it can also obscure underlying problems that need to be addressed. Implement appropriate monitoring and events so that you can detect patterns of problems, including those addressed by auto healing, so that you can resolve root cause issues. Amazon CloudWatch Alarms can be invoked based on failures that occur. They can also be invoked based on automated healing actions that run. CloudWatch Alarms can be configured to send emails, or to log incidents in third-party incident tracking systems using Amazon SNS integration.

Common anti-patterns:

- Sending alarms that no one acts upon.
- Performing auto healing automation, but not notifying that healing was needed.

Benefits of establishing this best practice: Notifications of recovery events will ensure that you don't ignore problems that occur infrequently.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Alarms on business Key Performance Indicators when they exceed a low threshold Having a low threshold alarm on your business KPIs help you know when your workload is unavailable or non-functional.
  - Creating a CloudWatch Alarm Based on a Static Threshold
- Alarm on events that invoke healing automation You can directly invoke an SNS API to send notifications with any automation that you create.
  - What is Amazon Simple Notification Service?

Resources

Related documents:

- Creating a CloudWatch Alarm Based on a Static Threshold
- What Is Amazon EventBridge?
- What is Amazon Simple Notification Service?

REL11-BP07 Architect your product to meet availability targets and uptime service level agreements (SLAs)

Architect your product to meet availability targets and uptime service level agreements (SLAs). If you publish or privately agree to availability targets or uptime SLAs, verify that your architecture and operational processes are designed to support them.

Desired outcome: Each application has a defined target for availability and SLA for performance metrics, which can be monitored and maintained in order to meet business outcomes.

Common anti-patterns:

- Designing and deploying workload's without setting any SLAs.
- SLA metrics are set to high without rationale or business requirements.
- Setting SLAs without taking into account for dependencies and their underlying SLA.
- Application designs are created without considering the Shared Responsibility Model for Resilience.

Benefits of establishing this best practice: Designing applications based on key resiliency targets helps you meet business objectives and customer expectations. These objectives help drive the application design process that evaluates different technologies and considers various tradeoffs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Application designs have to account for a diverse set of requirements that are derived from business, operational, and financial objectives. Within the operational requirements, workloads need to have specific resilience metric targets so they can be properly monitored and supported. Resilience metrics should not be set or derived after deploying the workload. They should be defined during the design phase and help guide various decisions and tradeoffs.

- Every workload should have its own set of resilience metrics. Those metrics may be different from other business applications.
- Reducing dependencies can have a positive impact on availability. Each workload should consider its dependencies and their SLAs. In general, select dependencies with availability goals equal to or greater than the goals of your workload.
- Consider loosely coupled designs so your workload can operate correctly despite dependency impairment, where possible.
- Reduce control plane dependencies, especially during recovery or a degradation. Evaluate designs that are statically stable for mission critical workloads. Use resource sparing to increase the availability of those dependencies in a workload.
- Observability and instrumentation are critical for achieving SLAs by reducing Mean Time to Detection (MTTD) and Mean Time to Repair (MTTR).
- Less frequent failure (longer MTBF), shorter failure detection times (shorter MTTD), and shorter repair times (shorter MTTR) are the three factors that are used to improve availability in distributed systems.

- Establishing and meeting resilience metrics for a workload is foundational to any effective design. Those designs must factor in tradeoffs of design complexity, service dependencies, performance, scaling, and costs.

Implementation steps

- Review and document the workload design considering the following questions:
  - Where are control planes used in the workload?
  - How does the workload implement fault tolerance?
  - What are the design patterns for scaling, automatic scaling, redundancy, and highly available components?
  - What are the requirements for data consistency and availability?
  - Are there considerations for resource sparing or resource static stability?
  - What are the service dependencies?
- Define SLA metrics based on the workload architecture while working with stakeholders. Consider the SLAs of all dependencies used by the workload.
- Once the SLA target has been set, optimize the architecture to meet the SLA.
- Once the design is set that will meet the SLA, implement operational changes, process automation, and runbooks that also will have focus on reducing MTTD and MTTR.
- Once deployed, monitor and report on the SLA.

Resources

Related best practices:

- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP05 Test resiliency using chaos engineering
- REL13-BP01 Define recovery objectives for downtime and data loss
- Understanding workload health

Related documents:

- Availability with redundancy
- Reliability pillar - Availability
- Measuring availability
- AWS Fault Isolation Boundaries
- Shared Responsibility Model for Resiliency
- Static stability using Availability Zones
- AWS Service Level Agreements (SLAs)
- Guidance for Cell-based Architecture on AWS
- AWS infrastructure
- Advanced Multi-AZ Resiliance Patterns whitepaper

Related services:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

REL 12. How do you test reliability?

After you have designed your workload to be resilient to the stresses of production, testing is the only way to verify that it will operate as designed, and deliver the resiliency you expect.

Best practices

- REL12-BP01 Use playbooks to investigate failures
- REL12-BP02 Perform post-incident analysis
- REL12-BP03 Test functional requirements
- REL12-BP04 Test scaling and performance requirements
- REL12-BP05 Test resiliency using chaos engineering
- REL12-BP06 Conduct game days regularly

REL12-BP01 Use playbooks to investigate failures

Permit consistent and prompt responses to failure scenarios that are not well understood, by documenting the investigation process in playbooks. Playbooks are the predefined steps performed to identify the factors contributing to a failure scenario. The results from any process step are used to determine the next steps to take until the issue is identified or escalated.

The playbook is proactive planning that you must do, to be able to take reactive actions effectively. When failure scenarios not covered by the playbook are encountered in production, first address the issue (put out the fire). Then go back and look at the steps you took to address the issue and use these to add a new entry in the playbook.

Note that playbooks are used in response to specific incidents, while runbooks are used to achieve specific outcomes. Often, runbooks are used for routine activities and playbooks are used to respond to non-routine events.

Common anti-patterns:

- Planning to deploy a workload without knowing the processes to diagnose issues or respond to incidents.
- Unplanned decisions about which systems to gather logs and metrics from when investigating an event.
- Not retaining metrics and events long enough to be able to retrieve the data.

Benefits of establishing this best practice: Capturing playbooks ensures that processes can be consistently followed. Codifying your playbooks limits the introduction of errors from manual activity. Automating playbooks shortens the time to respond to an event by eliminating the requirement for team member intervention or providing them additional information when their intervention begins.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Use playbooks to identify issues. Playbooks are documented processes to investigate issues. Allow consistent and prompt responses to failure scenarios by documenting processes in playbooks. Playbooks must contain the information and guidance necessary for an adequately skilled person to gather applicable information, identify potential sources of failure, isolate faults, and determine contributing factors (perform post-incident analysis).

- Implement playbooks as code. Perform your operations as code by scripting your playbooks to ensure consistency and limit reduce errors caused by manual processes. Playbooks can be composed of multiple scripts representing the different steps that might be necessary to identify the contributing factors to an issue. Runbook activities can be invoked or performed as part of playbook activities, or might prompt to run a playbook in response to identified events.
  - Automate your operational playbooks with AWS Systems Manager
  - AWS Systems Manager Run Command
  - AWS Systems Manager Automation
  - What is AWS Lambda?
  - What Is Amazon EventBridge?
  - Using Amazon CloudWatch Alarms

Resources

Related documents:

- AWS Systems Manager Automation
- AWS Systems Manager Run Command
- Automate your operational playbooks with AWS Systems Manager
- Using Amazon CloudWatch Alarms
- Using Canaries (Amazon CloudWatch Synthetics)
- What Is Amazon EventBridge?
- What is AWS Lambda?

Related examples:

- Automating operations with Playbooks and Runbooks

REL12-BP02 Perform post-incident analysis

Review customer-impacting events, and identify the contributing factors and preventative action items. Use this information to develop mitigations to limit or prevent recurrence. Develop procedures for prompt and effective responses. Communicate contributing factors and corrective

actions as appropriate, tailored to target audiences. Have a method to communicate these causes to others as needed.

Assess why existing testing did not find the issue. Add tests for this case if tests do not already exist.

Common anti-patterns:

- Finding contributing factors, but not continuing to look deeper for other potential problems and approaches to mitigate.
- Only identifying human error causes, and not providing any training or automation that could prevent human errors.

Benefits of establishing this best practice: Conducting post-incident analysis and sharing the results permits other workloads to mitigate the risk if they have implemented the same contributing factors, and allows them to implement the mitigation or automated recovery before an incident occurs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Establish a standard for your post-incident analysis. Good post-incident analysis provides opportunities to propose common solutions for problems with architecture patterns that are used in other places in your systems.
  - Ensure that the contributing factors are honest and blame free.
  - If you do not document your problems, you cannot correct them.
    - Ensure post-incident analysis is blame free so you can be dispassionate about the proposed corrective actions and promote honest self-assessment and collaboration on your application teams.
- Use a process to determine contributing factors. Have a process to identify and document the contributing factors of an event so that you can develop mitigations to limit or prevent recurrence and you can develop procedures for prompt and effective responses. Communicate contributing factors as appropriate, tailored to target audiences.
  - What is log analytics?

Resources

Related documents:

- What is log analytics?
- Why you should develop a correction of error (COE)

REL12-BP03 Test functional requirements

Use techniques such as unit tests and integration tests that validate required functionality.

You achieve the best outcomes when these tests are run automatically as part of build and deployment actions. For instance, using AWS CodePipeline, developers commit changes to a source repository where CodePipeline automatically detects the changes. Those changes are built, and tests are run. After the tests are complete, the built code is deployed to staging servers for testing. From the staging server, CodePipeline runs more tests, such as integration or load tests. Upon the successful completion of those tests, CodePipeline deploys the tested and approved code to production instances.

Additionally, experience shows that synthetic transaction testing (also known as *canary testing*, but not to be confused with canary deployments) that can run and simulate customer behavior is among the most important testing processes. Run these tests constantly against your workload endpoints from diverse remote locations. Amazon CloudWatch Synthetics allows you to create canaries to monitor your endpoints and APIs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Test functional requirements. These include unit tests and integration tests that validate required functionality.
  - Use CodePipeline with AWS CodeBuild to test code and run builds
  - AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild
  - Continuous Delivery and Continuous Integration
  - Using Canaries (Amazon CloudWatch Synthetics)
  - Software test automation

Resources

Related documents:

- APN Partner: partners that can help with implementation of a continuous integration pipeline
- AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild
- AWS Marketplace: products that can be used for continuous integration
- Continuous Delivery and Continuous Integration
- Software test automation
- Use CodePipeline with AWS CodeBuild to test code and run builds
- Using Canaries (Amazon CloudWatch Synthetics)

REL12-BP04 Test scaling and performance requirements

Use techniques such as load testing to validate that the workload meets scaling and performance requirements.

In the cloud, you can create a production-scale test environment on demand for your workload. If you run these tests on scaled down infrastructure, you must scale your observed results to what you think will happen in production. Load and performance testing can also be done in production if you are careful not to impact actual users, and tag your test data so it does not comingle with real user data and corrupt usage statistics or production reports.

With testing, ensure that your base resources, scaling settings, service quotas, and resiliency design operate as expected under load.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Test scaling and performance requirements. Perform load testing to validate that the workload meets scaling and performance requirements.
  - Distributed Load Testing on AWS: simulate thousands of connected users
  - Apache JMeter
    - Deploy your application in an environment identical to your production environment and run a load test.

- Use infrastructure as code concepts to create an environment as similar to your production environment as possible.

Resources

Related documents:

- Distributed Load Testing on AWS: simulate thousands of connected users
- Apache JMeter

REL12-BP05 Test resiliency using chaos engineering

Run chaos experiments regularly in environments that are in or as close to production as possible to understand how your system responds to adverse conditions.

Desired outcome:

The resilience of the workload is regularly verified by applying chaos engineering in the form of fault injection experiments or injection of unexpected load, in addition to resilience testing that validates known expected behavior of your workload during an event. Combine both chaos engineering and resilience testing to gain confidence that your workload can survive component failure and can recover from unexpected disruptions with minimal to no impact.

Common anti-patterns:

- Designing for resiliency, but not verifying how the workload functions as a whole when faults occur.
- Never experimenting under real-world conditions and expected load.
- Not treating your experiments as code or maintaining them through the development cycle.
- Not running chaos experiments both as part of your CI/CD pipeline, as well as outside of deployments.
- Neglecting to use past post-incident analyses when determining which faults to experiment with.

Benefits of establishing this best practice: Injecting faults to verify the resilience of your workload allows you to gain confidence that the recovery procedures of your resilient design will work in the case of a real fault.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Chaos engineering provides your teams with capabilities to continually inject real world disruptions (simulations) in a controlled way at the service provider, infrastructure, workload, and component level, with minimal to no impact to your customers. It allows your teams to learn from faults and observe, measure, and improve the resilience of your workloads, as well as validate that alerts fire and teams get notified in the case of an event.

When performed continually, chaos engineering can highlight deficiencies in your workloads that, if left unaddressed, could negatively affect availability and operation.

> **Note**
>
> Chaos engineering is the discipline of experimenting on a system in order to build confidence in the system's capability to withstand turbulent conditions in production. – Principles of Chaos Engineering

If a system is able to withstand these disruptions, the chaos experiment should be maintained as an automated regression test. In this way, chaos experiments should be performed as part of your systems development lifecycle (SDLC) and as part of your CI/CD pipeline.

To ensure that your workload can survive component failure, inject real world events as part of your experiments. For example, experiment with the loss of Amazon EC2 instances or failover of the primary Amazon RDS database instance, and verify that your workload is not impacted (or only minimally impacted). Use a combination of component faults to simulate events that may be caused by a disruption in an Availability Zone.

For application-level faults (such as crashes), you can start with stressors such as memory and CPU exhaustion.

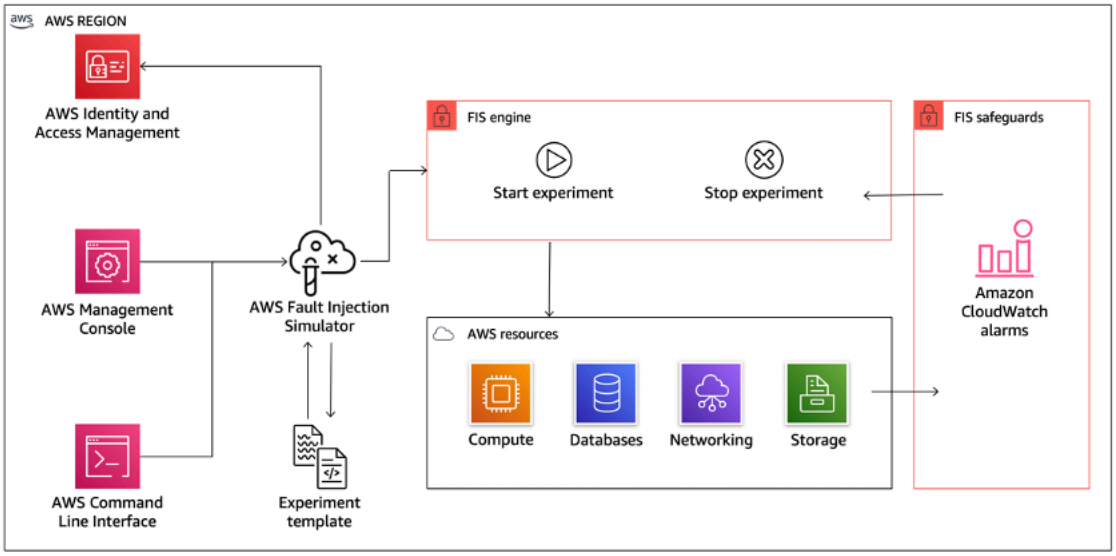
To validate fallback or failover mechanisms for external dependencies due to intermittent network disruptions, your components should simulate such an event by blocking access to the third-party providers for a specified duration that can last from seconds to hours.

Other modes of degradation might cause reduced functionality and slow responses, often resulting in a disruption of your services. Common sources of this degradation are increased latency on critical services and unreliable network communication (dropped packets). Experiments with

these faults, including networking effects such as latency, dropped messages, and DNS failures, could include the inability to resolve a name, reach the DNS service, or establish connections to dependent services.

Chaos engineering tools:

AWS Fault Injection Service (AWS FIS) is a fully managed service for running fault injection experiments that can be used as part of your CD pipeline, or outside of the pipeline. AWS FIS is a good choice to use during chaos engineering game days. It supports simultaneously introducing faults across different types of resources including Amazon EC2, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and Amazon RDS. These faults include termination of resources, forcing failovers, stressing CPU or memory, throttling, latency, and packet loss. Since it is integrated with Amazon CloudWatch Alarms, you can set up stop conditions as guardrails to rollback an experiment if it causes unexpected impact.



AWS Fault Injection Service integrates with AWS resources to allow you to run fault injection experiments for your workloads.

There are also several third-party options for fault injection experiments. These include open-source tools such as Chaos Toolkit, Chaos Mesh, and Litmus Chaos, as well as commercial options like Gremlin. To expand the scope of faults that can be injected on AWS, AWS FIS integrates with Chaos Mesh and Litmus Chaos, allowing you to coordinate fault injection workflows among multiple tools. For example, you can run a stress test on a pod's CPU using Chaos Mesh or Litmus faults while terminating a randomly selected percentage of cluster nodes using AWS FIS fault actions.

Implementation steps

1. Determine which faults to use for experiments.

   Assess the design of your workload for resiliency. Such designs (created using the best practices of the Well-Architected Framework) account for risks based on critical dependencies, past events, known issues, and compliance requirements. List each element of the design intended to maintain resilience and the faults it is designed to mitigate. For more information about creating such lists, see the Operational Readiness Review whitepaper which guides you on how to create a process to prevent reoccurrence of previous incidents. The Failure Modes and Effects Analysis (FMEA) process provides you with a framework for performing a component-level analysis of failures and how they impact your workload. FMEA is outlined in more detail by Adrian Cockcroft in Failure Modes and Continuous Resilience.

2. Assign a priority to each fault.

   Start with a coarse categorization such as high, medium, or low. To assess priority, consider frequency of the fault and impact of failure to the overall workload.

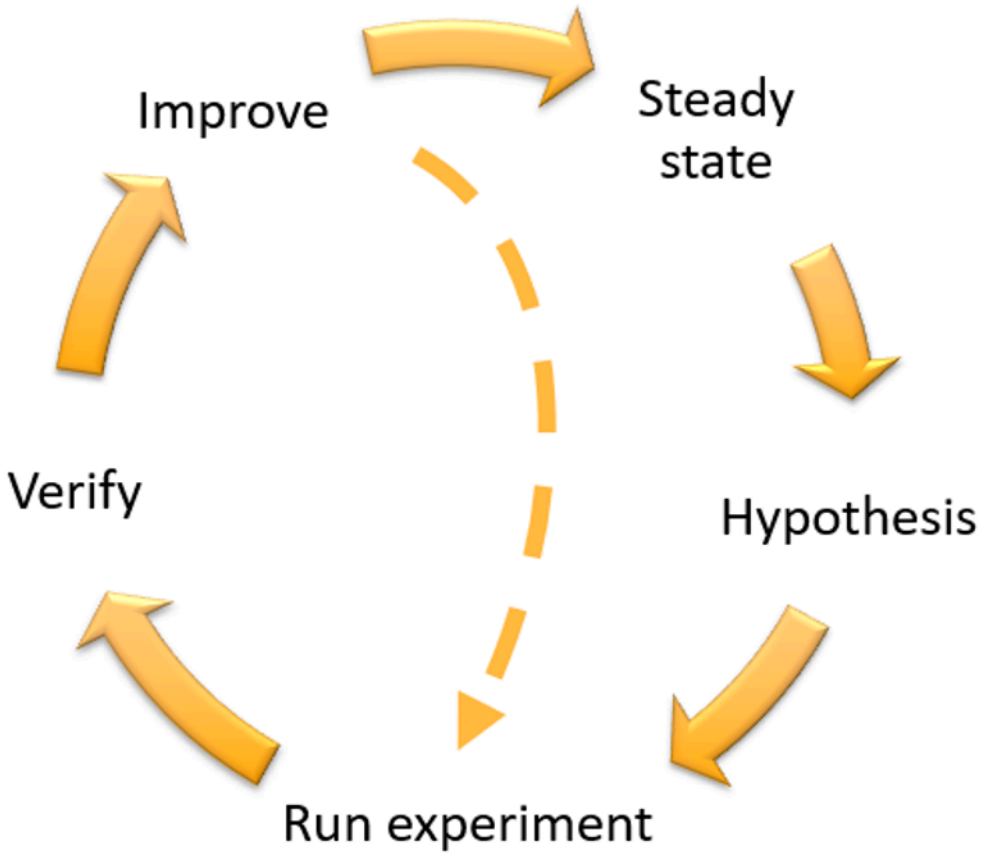
   When considering frequency of a given fault, analyze past data for this workload when available. If not available, use data from other workloads running in a similar environment.

   When considering impact of a given fault, the larger the scope of the fault, generally the larger the impact. Also consider the workload design and purpose. For example, the ability to access the source data stores is critical for a workload doing data transformation and analysis. In this case, you would prioritize experiments for access faults, as well as throttled access and latency insertion.

   Post-incident analyses are a good source of data to understand both frequency and impact of failure modes.

   Use the assigned priority to determine which faults to experiment with first and the order with which to develop new fault injection experiments.

3. For each experiment that you perform, follow the chaos engineering and continuous resilience flywheel in the following figure.



Chaos engineering and continuous resilience flywheel, using the scientific method by Adrian Hornsby.

a. Define steady state as some measurable output of a workload that indicates normal behavior.

   Your workload exhibits steady state if it is operating reliably and as expected. Therefore, validate that your workload is healthy before defining steady state. Steady state does not necessarily mean no impact to the workload when a fault occurs, as a certain percentage in faults could be within acceptable limits. The steady state is your baseline that you will observe during the experiment, which will highlight anomalies if your hypothesis defined in the next step does not turn out as expected.

   For example, a steady state of a payments system can be defined as the processing of 300 TPS with a success rate of 99% and round-trip time of 500 ms.

b. Form a hypothesis about how the workload will react to the fault.

A good hypothesis is based on how the workload is expected to mitigate the fault to maintain the steady state. The hypothesis states that given the fault of a specific type, the system or workload will continue steady state, because the workload was designed with specific mitigations. The specific type of fault and mitigations should be specified in the hypothesis.

The following template can be used for the hypothesis (but other wording is also acceptable):

> **Note**
>
> If *`specific fault`* occurs, the *`workload name`* workload will *`describe mitigating controls`* to maintain *`business or technical metric impact`*.

For example:

- If 20% of the nodes in the Amazon EKS node-group are taken down, the Transaction Create API continues to serve the 99th percentile of requests in under 100 ms (steady state). The Amazon EKS nodes will recover within five minutes, and pods will get scheduled and process traffic within eight minutes after the initiation of the experiment. Alerts will fire within three minutes.
- If a single Amazon EC2 instance failure occurs, the order system's Elastic Load Balancing health check will cause the Elastic Load Balancing to only send requests to the remaining healthy instances while the Amazon EC2 Auto Scaling replaces the failed instance, maintaining a less than 0.01% increase in server-side (5xx) errors (steady state).
- If the primary Amazon RDS database instance fails, the Supply Chain data collection workload will failover and connect to the standby Amazon RDS database instance to maintain less than 1 minute of database read or write errors (steady state).

c. Run the experiment by injecting the fault.

An experiment should by default be fail-safe and tolerated by the workload. If you know that the workload will fail, do not run the experiment. Chaos engineering should be used to find known-unknowns or unknown-unknowns. *Known-unknowns* are things you are aware of but don't fully understand, and *unknown-unknowns* are things you are neither aware of nor fully understand. Experimenting against a workload that you know is broken won't provide you with new insights. Your experiment should be carefully planned, have a clear scope of impact, and provide a rollback mechanism that can be applied in case of unexpected turbulence. If your due-diligence shows that your workload should survive the experiment, move forward

with the experiment. There are several options for injecting the faults. For workloads on AWS, AWS FIS provides many predefined fault simulations called actions. You can also define custom actions that run in AWS FIS using AWS Systems Manager documents.

We discourage the use of custom scripts for chaos experiments, unless the scripts have the capabilities to understand the current state of the workload, are able to emit logs, and provide mechanisms for rollbacks and stop conditions where possible.

An effective framework or toolset which supports chaos engineering should track the current state of an experiment, emit logs, and provide rollback mechanisms to support the controlled running of an experiment. Start with an established service like AWS FIS that allows you to perform experiments with a clearly defined scope and safety mechanisms that rollback the experiment if the experiment introduces unexpected turbulence. To learn about a wider variety of experiments using AWS FIS, also see the Resilient and Well-Architected Apps with Chaos Engineering lab. Also, AWS Resilience Hub will analyze your workload and create experiments that you can choose to implement and run in AWS FIS.

> **Note**
>
> For every experiment, clearly understand the scope and its impact. We recommend that faults should be simulated first on a non-production environment before being run in production.

Experiments should run in production under real-world load using canary deployments that spin up both a control and experimental system deployment, where feasible. Running experiments during off-peak times is a good practice to mitigate potential impact when first experimenting in production. Also, if using actual customer traffic poses too much risk, you can run experiments using synthetic traffic on production infrastructure against the control and experimental deployments. When using production is not possible, run experiments in pre-production environments that are as close to production as possible.

You must establish and monitor guardrails to ensure the experiment does not impact production traffic or other systems beyond acceptable limits. Establish stop conditions to stop an experiment if it reaches a threshold on a guardrail metric that you define. This should include the metrics for steady state for the workload, as well as the metric against the components into which you're injecting the fault. A synthetic monitor (also known as a user canary) is one metric you should usually include as a user proxy. Stop conditions for AWS FIS

are supported as part of the experiment template, allowing up to five stop-conditions per template.

One of the principles of chaos is minimize the scope of the experiment and its impact:

While there must be an allowance for some short-term negative impact, it is the responsibility and obligation of the Chaos Engineer to ensure the fallout from experiments are minimized and contained.

A method to verify the scope and potential impact is to perform the experiment in a non-production environment first, verifying that thresholds for stop conditions activate as expected during an experiment and observability is in place to catch an exception, instead of directly experimenting in production.

When running fault injection experiments, verify that all responsible parties are well-informed. Communicate with appropriate teams such as the operations teams, service reliability teams, and customer support to let them know when experiments will be run and what to expect. Give these teams communication tools to inform those running the experiment if they see any adverse effects.

You must restore the workload and its underlying systems back to the original known-good state. Often, the resilient design of the workload will self-heal. But some fault designs or failed experiments can leave your workload in an unexpected failed state. By the end of the experiment, you must be aware of this and restore the workload and systems. With AWS FIS you can set a rollback configuration (also called a post action) within the action parameters. A post action returns the target to the state that it was in before the action was run. Whether automated (such as using AWS FIS) or manual, these post actions should be part of a playbook that describes how to detect and handle failures.

d. Verify the hypothesis.

Principles of Chaos Engineering gives this guidance on how to verify steady state of your workload:

Focus on the measurable output of a system, rather than internal attributes of the system. Measurements of that output over a short period of time constitute a proxy for the system's steady state. The overall system's throughput, error rates, and latency percentiles could all be metrics of interest representing steady state behavior. By focusing on systemic behavior patterns during experiments, chaos engineering verifies that the system does work, rather than trying to validate how it works.

In our two previous examples, we include the steady state metrics of less than 0.01% increase in server-side (5xx) errors and less than one minute of database read and write errors.

The 5xx errors are a good metric because they are a consequence of the failure mode that a client of the workload will experience directly. The database errors measurement is good as a direct consequence of the fault, but should also be supplemented with a client impact measurement such as failed customer requests or errors surfaced to the client. Additionally, include a synthetic monitor (also known as a user canary) on any APIs or URIs directly accessed by the client of your workload.

e. Improve the workload design for resilience.

If steady state was not maintained, then investigate how the workload design can be improved to mitigate the fault, applying the best practices of the AWS Well-Architected Reliability pillar. Additional guidance and resources can be found in the AWS Builder's Library, which hosts articles about how to improve your health checks or employ retries with backoff in your application code, among others.

After these changes have been implemented, run the experiment again (shown by the dotted line in the chaos engineering flywheel) to determine their effectiveness. If the verify step indicates the hypothesis holds true, then the workload will be in steady state, and the cycle continues.

4. Run experiments regularly.

A chaos experiment is a cycle, and experiments should be run regularly as part of chaos engineering. After a workload meets the experiment's hypothesis, the experiment should be automated to run continually as a regression part of your CI/CD pipeline. To learn how to do this, see this blog on how to run AWS FIS experiments using AWS CodePipeline. This lab on recurrent AWS FIS experiments in a CI/CD pipeline allows you to work hands-on.

Fault injection experiments are also a part of game days (see REL12-BP06 Conduct game days regularly). Game days simulate a failure or event to verify systems, processes, and team responses. The purpose is to actually perform the actions the team would perform as if an exceptional event happened.

5. Capture and store experiment results.

Results for fault injection experiments must be captured and persisted. Include all necessary data (such as time, workload, and conditions) to be able to later analyze experiment results and

trends. Examples of results might include screenshots of dashboards, CSV dumps from your metric's database, or a hand-typed record of events and observations from the experiment. Experiment logging with AWS FIS can be part of this data capture.

Resources

Related best practices:

- REL08-BP03 Integrate resiliency testing as part of your deployment
- REL13-BP03 Test disaster recovery implementation to validate the implementation

Related documents:

- What is AWS Fault Injection Service?
- What is AWS Resilience Hub?
- Principles of Chaos Engineering
- Chaos Engineering: Planning your first experiment
- Resilience Engineering: Learning to Embrace Failure
- Chaos Engineering stories
- Avoiding fallback in distributed systems
- Canary Deployment for Chaos Experiments

Related videos:

- AWS re:Invent 2020: Testing resiliency using chaos engineering (ARC316)
- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)
- AWS re:Invent 2019: Performing chaos engineering in a serverless world (CMY301)

Related examples:

- Well-Architected lab: Level 300: Testing for Resiliency of Amazon EC2, Amazon RDS, and Amazon S3
- Chaos Engineering on AWS lab
- Resilient and Well-Architected Apps with Chaos Engineering lab

- Serverless Chaos lab
- Measure and Improve Your Application Resilience with AWS Resilience Hub lab

Related tools:

- AWS Fault Injection Service
- AWS Marketplace: Gremlin Chaos Engineering Platform
- Chaos Toolkit
- Chaos Mesh
- Litmus

REL12-BP06 Conduct game days regularly

Use game days to regularly exercise your procedures for responding to events and failures as close to production as possible (including in production environments) with the people who will be involved in actual failure scenarios. Game days enforce measures to ensure that production events do not impact users.

Game days simulate a failure or event to test systems, processes, and team responses. The purpose is to actually perform the actions the team would perform as if an exceptional event happened. This will help you understand where improvements can be made and can help develop organizational experience in dealing with events. These should be conducted regularly so that your team builds *muscle memory* on how to respond.

After your design for resiliency is in place and has been tested in non-production environments, a game day is the way to ensure that everything works as planned in production. A game day, especially the first one, is an “all hands on deck” activity where engineers and operations are all informed when it will happen, and what will occur. Runbooks are in place. Simulated events are run, including possible failure events, in the production systems in the prescribed manner, and impact is assessed. If all systems operate as designed, detection and self-healing will occur with little to no impact. However, if negative impact is observed, the test is rolled back and the workload issues are remedied, manually if necessary (using the runbook). Since game days often take place in production, all precautions should be taken to ensure that there is no impact on availability to your customers.

Common anti-patterns:

- Documenting your procedures, but never exercising them.
- Not including business decision makers in the test exercises.

Benefits of establishing this best practice: Conducting game days regularly ensures that all staff follows the policies and procedures when an actual incident occurs, and validates that those policies and procedures are appropriate.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Schedule game days to regularly exercise your runbooks and playbooks. Game days should involve everyone who would be involved in a production event: business owner, development staff, operational staff, and incident response teams.
  - Run your load or performance tests and then run your failure injection.
  - Look for anomalies in your runbooks and opportunities to exercise your playbooks.
    - If you deviate from your runbooks, refine the runbook or correct the behavior. If you exercise your playbook, identify the runbook that should have been used, or create a new one.

Resources

Related documents:

- What is AWS GameDay?

Related videos:

- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)

Related examples:

- AWS Well-Architected Labs - Testing Resiliency

REL 13. How do you plan for disaster recovery (DR)?

Having backups and redundant workload components in place is the start of your DR strategy. RTO and RPO are your objectives for restoration of your workload. Set these based on business needs.

Implement a strategy to meet these objectives, considering locations and function of workload resources and data. The probability of disruption and cost of recovery are also key factors that help to inform the business value of providing disaster recovery for a workload.

Best practices

- REL13-BP01 Define recovery objectives for downtime and data loss
- REL13-BP02 Use defined recovery strategies to meet the recovery objectives
- REL13-BP03 Test disaster recovery implementation to validate the implementation
- REL13-BP04 Manage configuration drift at the DR site or Region
- REL13-BP05 Automate recovery

REL13-BP01 Define recovery objectives for downtime and data loss

The workload has a recovery time objective (RTO) and recovery point objective (RPO).

Recovery Time Objective (RTO) is the maximum acceptable delay between the interruption of service and restoration of service. This determines what is considered an acceptable time window when service is unavailable.

Recovery Point Objective (RPO) is the maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable loss of data between the last recovery point and the interruption of service.

RTO and RPO values are important considerations when selecting an appropriate Disaster Recovery (DR) strategy for your workload. These objectives are determined by the business, and then used by technical teams to select and implement a DR strategy.

Desired Outcome:

Every workload has an assigned RTO and RPO, defined based on business impact. The workload is assigned to a predefined tier, defining service availability and acceptable loss of data, with an associated RTO and RPO. If such tiering is not possible then this can be assigned bespoke per workload, with the intent to create tiers later. RTO and RPO are used as one of the primary considerations for selection of a disaster recovery strategy implementation for the workload. Additional considerations in picking a DR strategy are cost constraints, workload dependencies, and operational requirements.

For RTO, understand impact based on duration of an outage. Is it linear, or are there nonlinear implications? (for example. after four hours, you shut down a manufacturing line until the start of the next shift).

A disaster recovery matrix, like the following, can help you understand how workload criticality relates to recovery objectives. (Note that the actual values for the X and Y axes should be customized to your organization needs).

Disaster Recovery Matrix						
		Recovery Point Objective				
		< 1 Minute	< 1 Hour	< 6 Hours	< 1 Day	+ 1 Day
Recovery Time Objective	< 10 Minutes	Critical	Critical	High	Medium	Medium
	< 2 Hours	Critical	High	Medium	Medium	Low
	< 8 Hours	High	Medium	Medium	Low	Low
	< 24 Hours	Medium	Medium	Low	Low	Low
	24 + Hours	Medium	Low	Low	Low	Low

Figure 16: Disaster recovery matrix

Common anti-patterns:

- No defined recovery objectives.
- Selecting arbitrary recovery objectives.
- Selecting recovery objectives that are too lenient and do not meet business objectives.
- Not understanding of the impact of downtime and data loss.
- Selecting unrealistic recovery objectives, such as zero time to recover and zero data loss, which may not be achievable for your workload configuration.
- Selecting recovery objectives more stringent than actual business objectives. This forces DR implementations that are costlier and more complicated than what the workload needs.
- Selecting recovery objectives incompatible with those of a dependent workload.
- Your recovery objectives do not consider regulatory compliance requirements.
- RTO and RPO defined for a workload, but never tested.

Benefits of establishing this best practice: Your recovery objectives for time and data loss are necessary to guide your DR implementation.

Level of risk exposed if this best practice is not established: High

Implementation guidance

For the given workload, you must understand the impact of downtime and lost data on your business. The impact generally grows larger with greater downtime or data loss, but the shape of this growth can differ based on the workload type. For example, you may be able to tolerate downtime for up to an hour with little impact, but after that impact quickly rises. Impact to business manifests in many forms including monetary cost (such as lost revenue), customer trust (and impact to reputation), operational issues (such as missing payroll or decreased productivity), and regulatory risk. Use the following steps to understand these impacts, and set RTO and RPO for your workload.

Implementation Steps

1. Determine your business stakeholders for this workload, and engage with them to implement these steps. Recovery objectives for a workload are a business decision. Technical teams then work with business stakeholders to use these objectives to select a DR strategy.

   > **Note**
   >
   > For steps 2 and 3, you can use the the section called “Implementation worksheet”.

2. Gather the necessary information to make a decision by answering the questions below.
3. Do you have categories or tiers of criticality for workload impact in your organization?
   a. If yes, assign this workload to a category
   b. If no, then establish these categories. Create five or fewer categories and refine the range of your recovery time objective for each one. Example categories include: critical, high, medium, low. To understand how workloads map to categories, consider whether the workload is mission critical, business important, or non-business driving.
   c. Set workload RTO and RPO based on category. Always choose a category more strict (lower RTO and RPO) than the raw values calculated entering this step. If this results in an unsuitably large change in value, then consider creating a new category.
4. Based on these answers, assign RTO and RPO values to the workload. This can be done directly, or by assigning the workload to a predefined tier of service.

5. Document the disaster recovery plan (DRP) for this workload, which is a part of your organization's business continuity plan (BCP), in a location accessible to the workload team and stakeholders
   a. Record the RTO and RPO, and the information used to determine these values. Include the strategy used for evaluating workload impact to the business
   b. Record other metrics besides RTO and RPO are you tracking or plan to track for disaster recovery objectives
   c. You will add details of your DR strategy and runbook to this plan when you create these.
6. By looking up the workload criticality in a matrix such as that in Figure 15, you can begin to establish predefined tiers of service defined for your organization.
7. After you have implemented a DR strategy (or a proof of concept for a DR strategy) as per the section called "REL13-BP02 Use defined recovery strategies to meet the recovery objectives", test this strategy to determine workload actual RTC (Recovery Time Capability) and RPC (Recovery Point Capability). If these do not meet the target recovery objectives, then either work with your business stakeholders to adjust those objectives, or make changes to the DR strategy is possible to meet target objectives.

Primary questions

1. What is the maximum time the workload can be down before severe impact to the business is incurred
   a. Determine the monetary cost (direct financial impact) to the business per minute if workload is disrupted.
   b. Consider that impact is not always linear. Impact can be limited at first, and then increase rapidly past a critical point in time.
2. What is the maximum amount of data that can be lost before severe impact to the business is incurred
   a. Consider this value for your most critical data store. Identify the respective criticality for other data stores.
   b. Can workload data be recreated if lost? If this is operationally easier than backup and restore, then choose RPO based on the criticality of the source data used to recreate the workload data.
3. What are the recovery objectives and availability expectations of workloads that this one depends on (downstream), or workloads that depend on this one (upstream)?

a. Choose recovery objectives that allow this workload to meet the requirements of upstream dependencies

b. Choose recovery objectives that are achievable given the recovery capabilities of downstream dependencies. Non-critical downstream dependencies (ones you can “work around”) can be excluded. Or, work with critical downstream dependencies to improve their recovery capabilities where necessary.

Additional questions

Consider these questions, and how they may apply to this workload:

4. Do you have different RTO and RPO depending on the type of outage (Region vs. AZ, etc.)?
5. Is there a specific time (seasonality, sales events, product launches) when your RTO/RPO may change? If so, what is the different measurement and time boundary?
6. How many customers will be impacted if workload is disrupted?
7. What is the impact to reputation if workload is disrupted?
8. What other operational impacts may occur if workload is disrupted? For example, impact to employee productivity if email systems are unavailable, or if Payroll systems are unable to submit transactions.
9. How does workload RTO and RPO align with Line of Business and Organizational DR Strategy?
10. Are there internal contractual obligations for providing a service? Are there penalties for not meeting them?
11. What are the regulatory or compliance constraints with the data?

Implementation worksheet

You can use this worksheet for implementation steps 2 and 3. You may adjust this worksheet to suit your specific needs, such as adding additional questions.

Step 2: Primary questions	Applies to workload?	workload RTO	workload RPO	RTO adjust.	RPO adjust.	Instructions
[1] maximum time the workload can be down						measured in time from start of outage to recovery
[2] maximum amount of data that can be lost						measured in time since last known good restorable dataset
[3a] upstream dependencies						enter the most strict upstream recovery objectives
[3b] downstream dependencies						enter the least strict downstream recovery objectives
[3a] reconciled upstream dependencies						If upstream value is less then current values and downstream value greater,
[3b] reconciled downstream dependencies						then work with dependencies to reconcile and enter reconciled values here
[3] dependencies						lower values to meet upstream dependencies or raise them based on downstream dependency capabilities
Step 2: Additional questions						Indicate if question applies. If it does not apply then skip it
Base RTO/RPO						Carry RTO and RPO values from above down to here
[4] type of outage	[ ]Y / [ ]N					Enter recovery objectives for event type with strictest requirements
[5] specific time-based objectives	[ ]Y / [ ]N					Enter recovery objectives for times with the strictest requirements
[6] customers disrupted	[ ]Y / [ ]N					Graph customers impacted as a function of time down or data lost. Use that to enter the maximum RTO and RPO permissible based on customer impact
[7] reputation impact	[ ]Y / [ ]N					Work with the business to determine maximum RTO and RPO based on impact to reputation
[8] operational impact	[ ]Y / [ ]N					Enter maximum RTO and RPO based on operational impact
[9] organizational alignment	[ ]Y / [ ]N					Enter maximum RTO and RPO for workloads of this type as per LOB and organizational requirements
[10] contractual obligations	[ ]Y / [ ]N					Enter maximum RTO and RPO based on contractual obligations
[11] regulatory compliance	[ ]Y / [ ]N					Enter maximum RTO and RPO based on applicable regulatory compliance
target based on additional questions						Take the minimum value (stricter value) from Q's 4-11 and enter it here
adjusted target						If the objectives on the above line cannot be accommodated, work with stakeholders to loosen constraints, and enter new minimum here
Adjusted RTO/RPO						Enter base RPO/RTO values, or adjusted target, whichever is lower
Step 3						
Map to predefined category or tier						Adjust both values to downward (more strict) to align to nearest defined tier

Worksheet

Level of effort for the Implementation Plan: Low

Resources

Related Best Practices:

- the section called “REL09-BP04 Perform periodic recovery of the data to verify backup integrity and processes”
- the section called “REL13-BP02 Use defined recovery strategies to meet the recovery objectives”
- the section called “REL13-BP03 Test disaster recovery implementation to validate the implementation”

Related documents:

- AWS Architecture Blog: Disaster Recovery Series
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)
- Managing resiliency policies with AWS Resilience Hub

- APN Partner: partners that can help with disaster recovery
- AWS Marketplace: products that can be used for disaster recovery

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Disaster Recovery of Workloads on AWS

REL13-BP02 Use defined recovery strategies to meet the recovery objectives

Define a disaster recovery (DR) strategy that meets your workload's recovery objectives. Choose a strategy such as backup and restore, standby (active/passive), or active/active.

Desired outcome: For each workload, there is a defined and implemented DR strategy that allows the workload to achieve DR objectives. DR strategies between workloads make use of reusable patterns (such as the strategies previously described),

Common anti-patterns:

- Implementing inconsistent recovery procedures for workloads with similar DR objectives.
- Leaving the DR strategy to be implemented ad-hoc when a disaster occurs.
- Having no plan for disaster recovery.
- Dependency on control plane operations during recovery.

Benefits of establishing this best practice:

- Using defined recovery strategies allows you to use common tooling and test procedures.
- Using defined recovery strategies improves knowledge sharing between teams and implementation of DR on the workloads they own.

Level of risk exposed if this best practice is not established: High. Without a planned, implemented, and tested DR strategy, you are unlikely to achieve recovery objectives in the event of a disaster.

Implementation guidance

A DR strategy relies on the ability to stand up your workload in a recovery site if your primary location becomes unable to run the workload. The most common recovery objectives are RTO and RPO, as discussed in REL13-BP01 Define recovery objectives for downtime and data loss.

A DR strategy across multiple Availability Zones (AZs) within a single AWS Region, can provide mitigation against disaster events like fires, floods, and major power outages. If it is a requirement to implement protection against an unlikely event that prevents your workload from being able to run in a given AWS Region, you can use a DR strategy that uses multiple Regions.

When architecting a DR strategy across multiple Regions, you should choose one of the following strategies. They are listed in increasing order of cost and complexity, and decreasing order of RTO and RPO. *Recovery Region* refers to an AWS Region other than the primary one used for your workload.

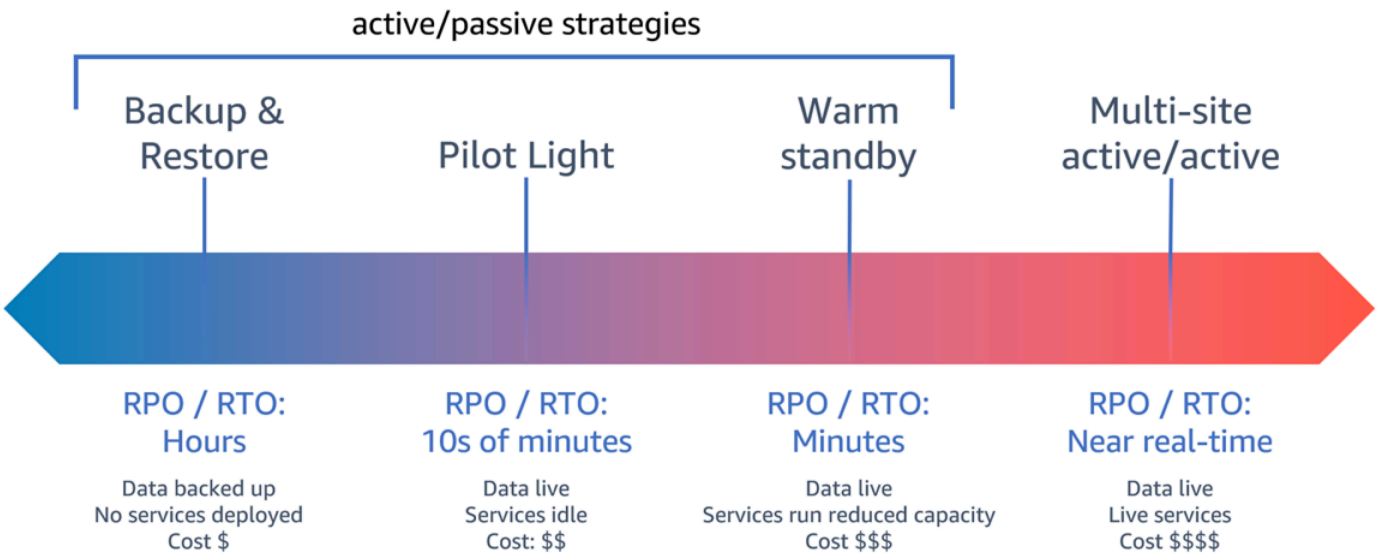


Figure 17: Disaster recovery (DR) strategies

- **Backup and restore** (RPO in hours, RTO in 24 hours or less): Back up your data and applications into the recovery Region. Using automated or continuous backups will permit point in time recovery (PITR), which can lower RPO to as low as 5 minutes in some cases. In the event of a disaster, you will deploy your infrastructure (using infrastructure as code to reduce RTO), deploy your code, and restore the backed-up data to recover from a disaster in the recovery Region.

- **Pilot light** (RPO in minutes, RTO in tens of minutes): Provision a copy of your core workload infrastructure in the recovery Region. Replicate your data into the recovery Region and create backups of it there. Resources required to support data replication and backup, such as databases and object storage, are always on. Other elements such as application servers or serverless compute are not deployed, but can be created when needed with the necessary configuration and application code.
- **Warm standby** (RPO in seconds, RTO in minutes): Maintain a scaled-down but fully functional version of your workload always running in the recovery Region. Business-critical systems are fully duplicated and are always on, but with a scaled down fleet. Data is replicated and live in the recovery Region. When the time comes for recovery, the system is scaled up quickly to handle the production load. The more scaled-up the warm standby is, the lower RTO and control plane reliance will be. When fully scales this is known as *hot standby*.
- **Multi-Region (multi-site) active-active** (RPO near zero, RTO potentially zero): Your workload is deployed to, and actively serving traffic from, multiple AWS Regions. This strategy requires you to synchronize data across Regions. Possible conflicts caused by writes to the same record in two different regional replicas must be avoided or handled, which can be complex. Data replication is useful for data synchronization and will protect you against some types of disaster, but it will not protect you against data corruption or destruction unless your solution also includes options for point-in-time recovery.

> **Note**
>
> The difference between pilot light and warm standby can sometimes be difficult to understand. Both include an environment in your recovery Region with copies of your primary region assets. The distinction is that pilot light cannot process requests without additional action taken first, while warm standby can handle traffic (at reduced capacity levels) immediately. Pilot light will require you to turn on servers, possibly deploy additional (non-core) infrastructure, and scale up, while warm standby only requires you to scale up (everything is already deployed and running). Choose between these based on your RTO and RPO needs.
>
> When cost is a concern, and you wish to achieve a similar RPO and RTO objectives as defined in the warm standby strategy, you could consider cloud native solutions, like AWS Elastic Disaster Recovery, that take the pilot light approach and offer improved RPO and RTO targets.

Implementation steps

1. **Determine a DR strategy that will satisfy recovery requirements for this workload.**

   Choosing a DR strategy is a trade-off between reducing downtime and data loss (RTO and RPO) and the cost and complexity of implementing the strategy. You should avoid implementing a strategy that is more stringent than it needs to be, as this incurs unnecessary costs.

   For example, in the following diagram, the business has determined their maximum permissible RTO as well as the limit of what they can spend on their service restoration strategy. Given the business' objectives, the DR strategies pilot light or warm standby will satisfy both the RTO and the cost criteria.

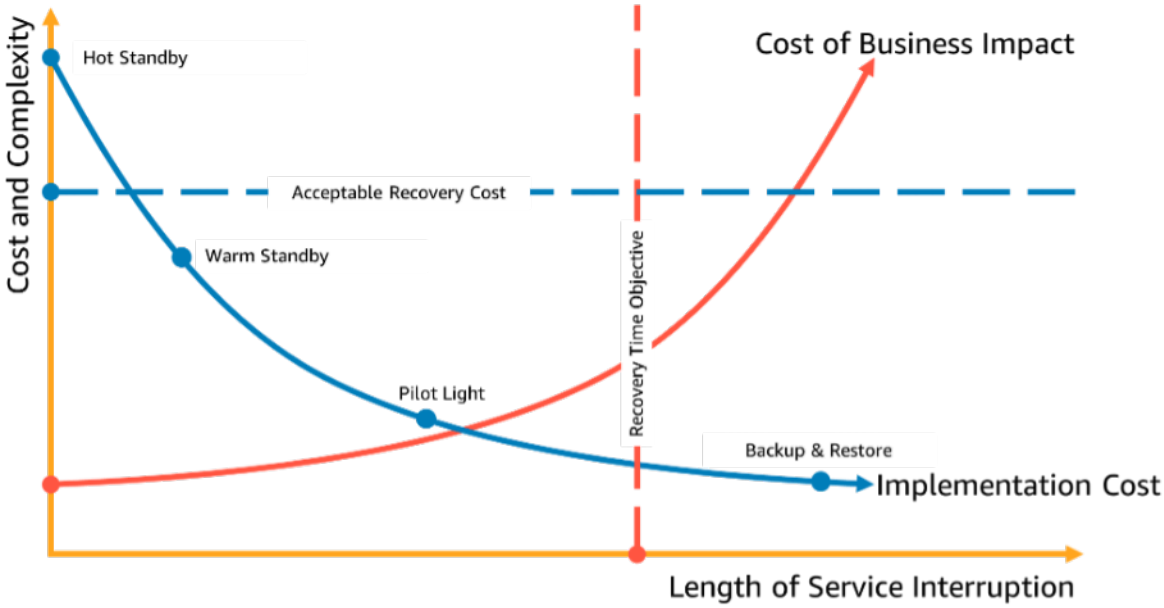


   Figure 18: Choosing a DR strategy based on RTO and cost

   To learn more, see Business Continuity Plan (BCP).

2. **Review the patterns for how the selected DR strategy can be implemented.**

   This step is to understand how you will implement the selected strategy. The strategies are explained using AWS Regions as the primary and recovery sites. However, you can also choose to use Availability Zones within a single Region as your DR strategy, which makes use of elements of multiple of these strategies.

   In the following steps, you can apply the strategy to your specific workload.

Backup and restore

Backup and restore is the least complex strategy to implement, but will require more time and effort to restore the workload, leading to higher RTO and RPO. It is a good practice to always make backups of your data, and copy these to another site (such as another AWS Region).

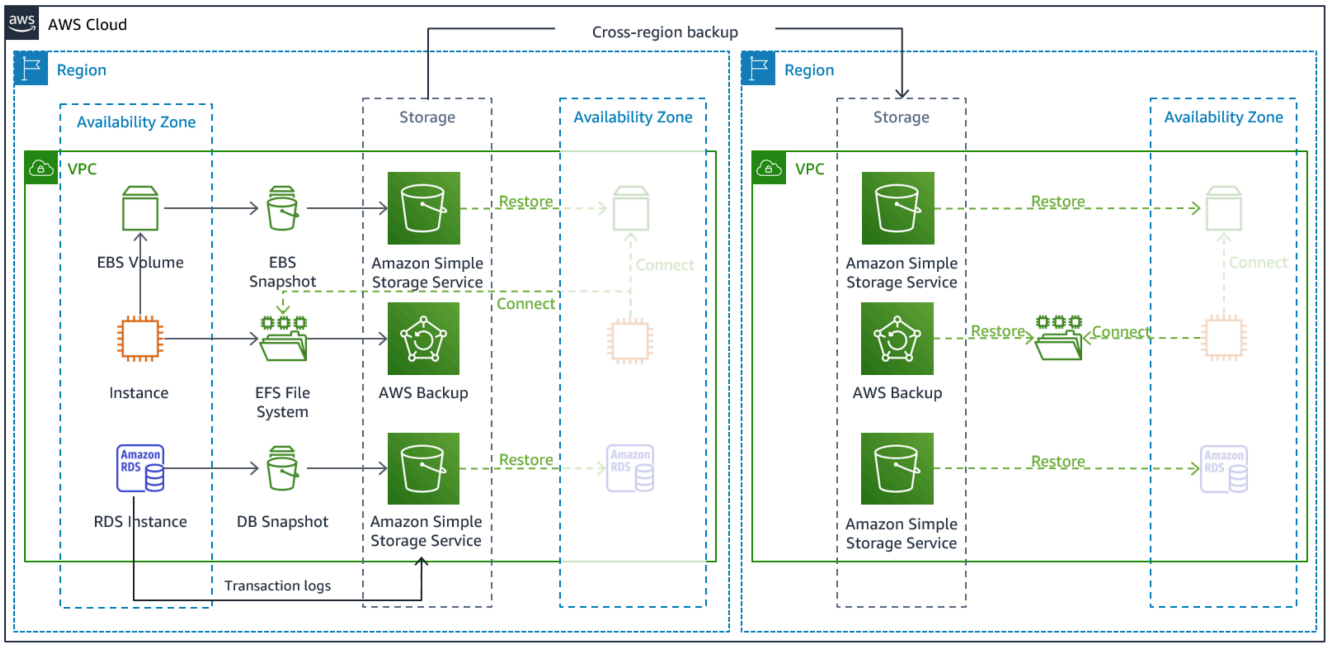


Figure 19: Backup and restore architecture

For more details on this strategy see Disaster Recovery (DR) Architecture on AWS, Part II: Backup and Restore with Rapid Recovery.

Pilot light

With the *pilot light* approach, you replicate your data from your primary Region to your recovery Region. Core resources used for the workload infrastructure are deployed in the recovery Region, however additional resources and any dependencies are still needed to make this a functional stack. For example, in Figure 20, no compute instances are deployed.

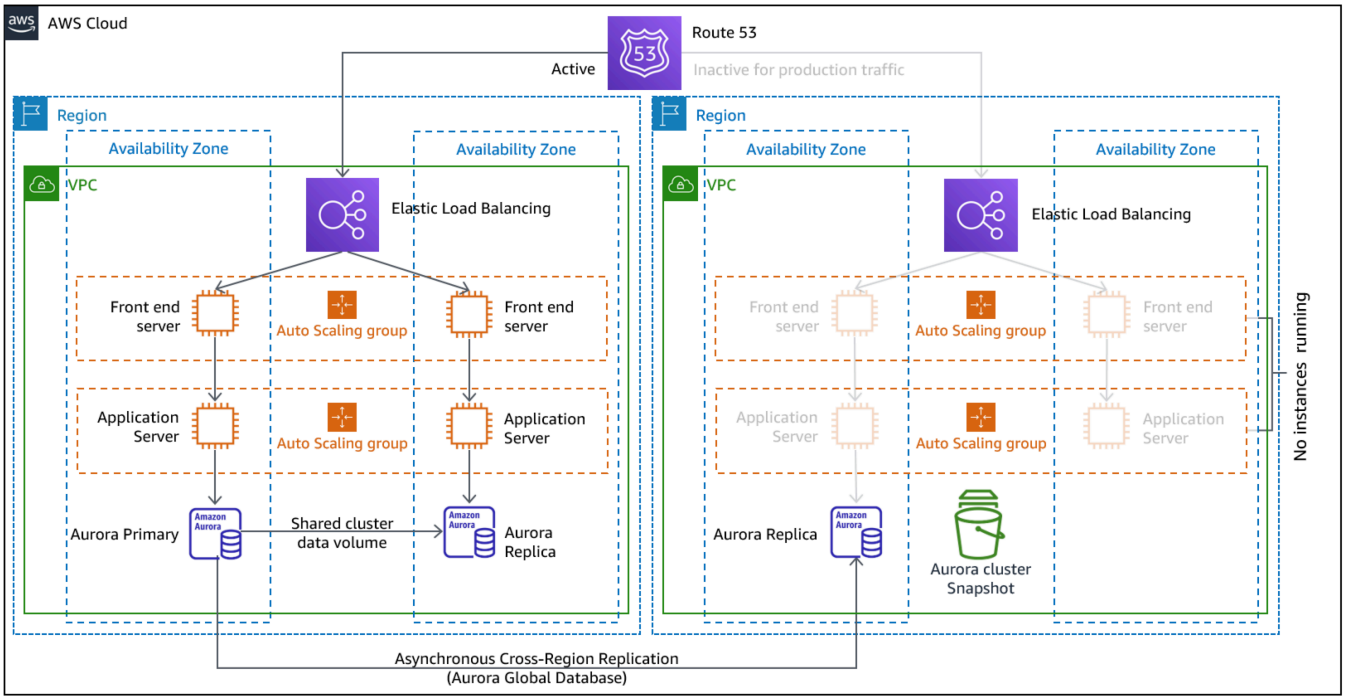


Figure 20: Pilot light architecture

For more details on this strategy, see Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby.

Warm standby

The *warm standby* approach involves ensuring that there is a scaled down, but fully functional, copy of your production environment in another Region. This approach extends the pilot light concept and decreases the time to recovery because your workload is always-on in another Region. If the recovery Region is deployed at full capacity, then this is known as *hot standby*.

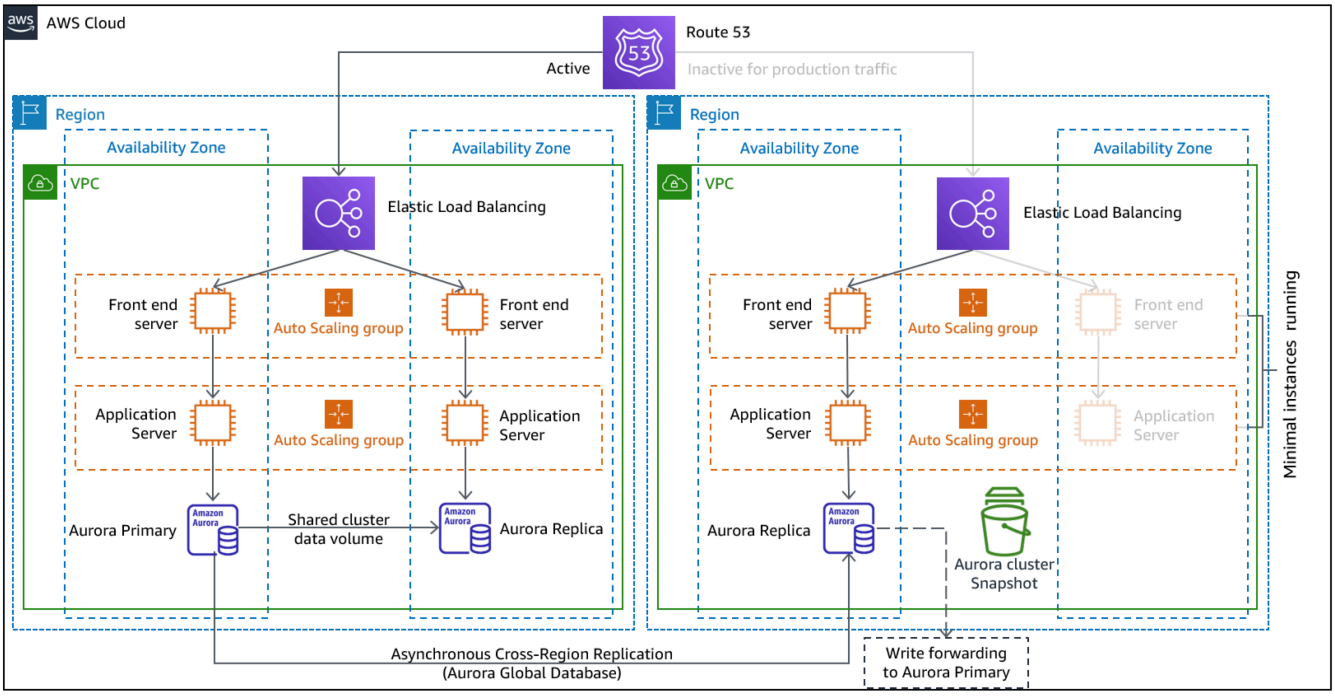


Figure 21: Warm standby architecture

Using warm standby or pilot light requires scaling up resources in the recovery Region. To verify capacity is available when needed, consider the use for capacity reservations for EC2 instances. If using AWS Lambda, then provisioned concurrency can provide runtime environments so that they are prepared to respond immediately to your function's invocations.

For more details on this strategy, see Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby.

Multi-site active/active

You can run your workload simultaneously in multiple Regions as part of a *multi-site active/active* strategy. Multi-site active/active serves traffic from all regions to which it is deployed. Customers may select this strategy for reasons other than DR. It can be used to increase availability, or when deploying a workload to a global audience (to put the endpoint closer to users and/or to deploy stacks localized to the audience in that region). As a DR strategy, if the workload cannot be supported in one of the AWS Regions to which it is deployed, then that Region is evacuated, and the remaining Regions are used to maintain availability. Multi-site active/active is the most operationally complex of the DR strategies, and should only be selected when business requirements necessitate it.

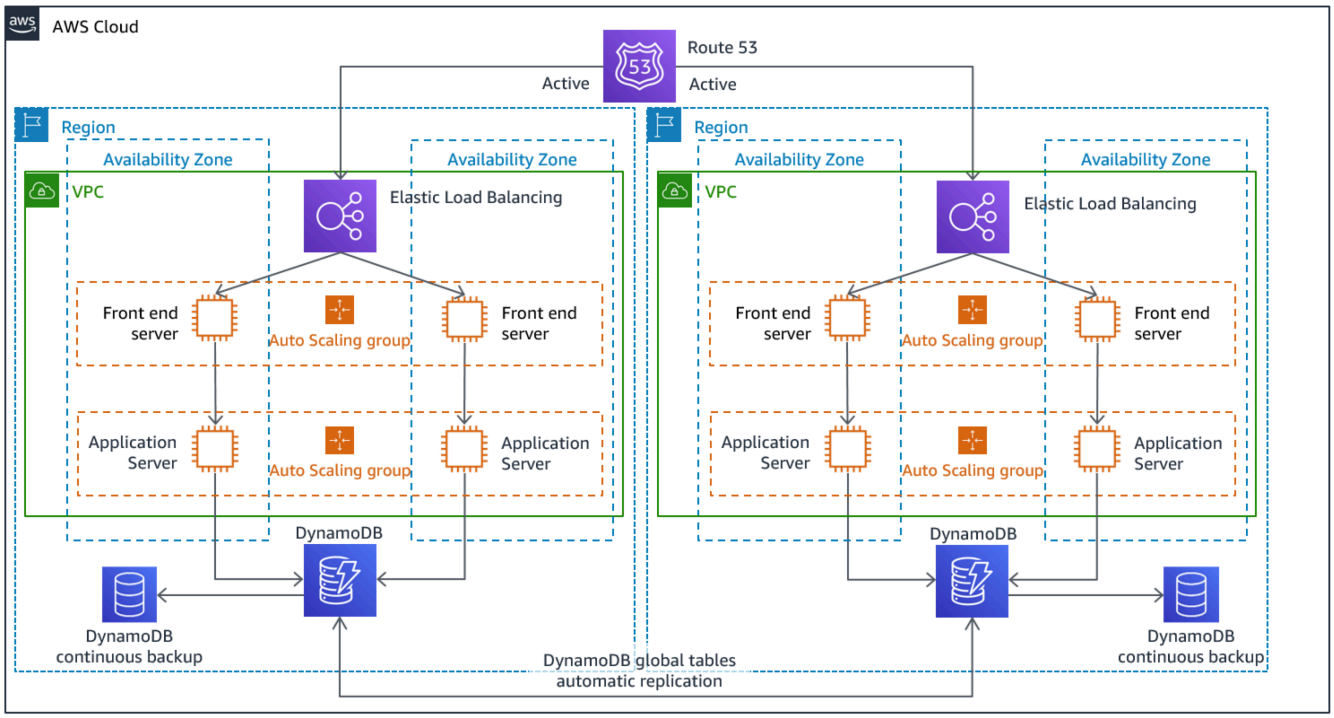


Figure 22: Multi-site active/active architecture

For more details on this strategy, see Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active.

AWS Elastic Disaster Recovery

If you are considering the pilot light or warm standby strategy for disaster recovery, AWS Elastic Disaster Recovery could provide an alternative approach with improved benefits. Elastic Disaster Recovery can offer an RPO and RTO target similar to warm standby, but maintain the low-cost approach of pilot light. Elastic Disaster Recovery replicates your data from your primary region to your recovery Region, using continual data protection to achieve an RPO measured in seconds and an RTO that can be measured in minutes. Only the resources required to replicate the data are deployed in the recovery region, which keeps costs down, similar to the pilot light strategy. When using Elastic Disaster Recovery, the service coordinates and orchestrates the recovery of compute resources when initiated as part of failover or drill.

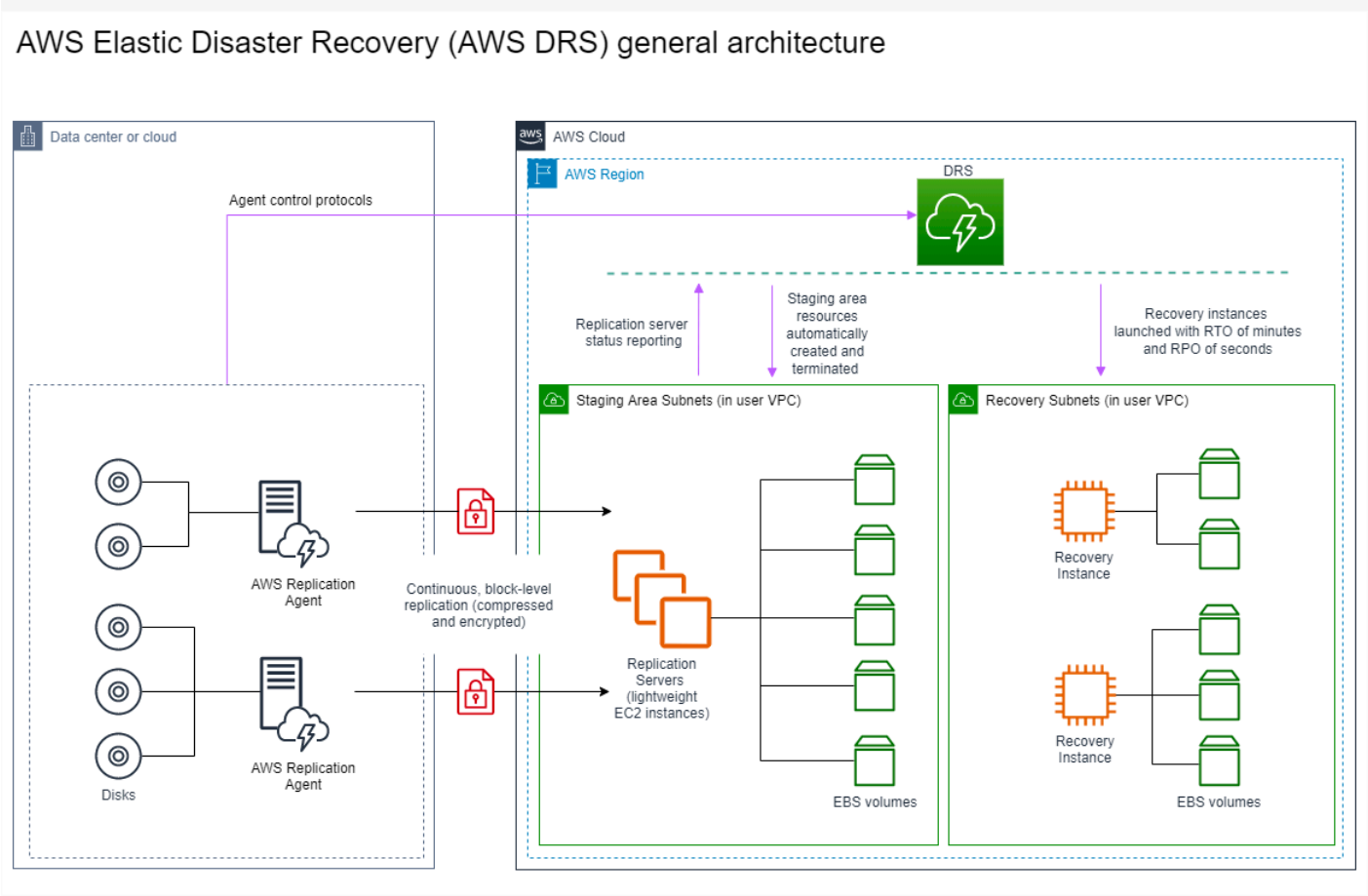


Figure 23: AWS Elastic Disaster Recovery architecture

Additional practices for protecting data

With all strategies, you must also mitigate against a data disaster. Continuous data replication protects you against some types of disaster, but it may not protect you against data corruption or destruction unless your strategy also includes versioning of stored data or options for point-in-time recovery. You must also back up the replicated data in the recovery site to create point-in-time backups in addition to the replicas.

Using multiple Availability Zones (AZs) within a single AWS Region

When using multiple AZs within a single Region, your DR implementation uses multiple elements of the above strategies. First you must create a high-availability (HA) architecture, using multiple AZs as shown in Figure 23. This architecture makes use of a multi-site active/active approach, as the Amazon EC2 instances and the Elastic Load Balancer have resources deployed in multiple AZs, actively handing requests. The architecture also demonstrates hot

standby, where if the primary Amazon RDS instance fails (or the AZ itself fails), then the standby instance is promoted to primary.

Figure 24: Multi-AZ architecture

In addition to this HA architecture, you need to add backups of all data required to run your workload. This is especially important for data that is constrained to a single zone such as Amazon EBS volumes or Amazon Redshift clusters. If an AZ fails, you will need to restore this data to another AZ. Where possible, you should also copy data backups to another AWS Region as an additional layer of protection.

An less common alternative approach to single Region, multi-AZ DR is illustrated in the blog post, Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack. Here, the strategy is to maintain as much isolation between the AZs as possible, like how Regions operate. Using this alternative strategy, you can choose an active/active or active/passive approach.

> **Note**
>
> Some workloads have regulatory data residency requirements. If this applies to your workload in a locality that currently has only one AWS Region, then multi-Region will not suit your business needs. Multi-AZ strategies provide good protection against most disasters.

3. **Assess the resources of your workload, and what their configuration will be in the recovery Region prior to failover (during normal operation).**

   For infrastructure and AWS resources use infrastructure as code such as AWS CloudFormation or third-party tools like Hashicorp Terraform. To deploy across multiple accounts and Regions with a single operation you can use AWS CloudFormation StackSets. For Multi-site active/active and Hot Standby strategies, the deployed infrastructure in your recovery Region has the same resources as your primary Region. For Pilot Light and Warm Standby strategies, the deployed infrastructure will require additional actions to become production ready. Using CloudFormation parameters and conditional logic, you can control whether a deployed stack is active or standby with a single template. When using Elastic Disaster Recovery, the service will replicate and orchestrate the restoration of application configurations and compute resources.

   All DR strategies require that data sources are backed up within the AWS Region, and then those backups are copied to the recovery Region. AWS Backup provides a centralized view where you can configure, schedule, and monitor backups for these resources. For Pilot Light, Warm Standby, and Multi-site active/active, you should also replicate data from the primary Region to data resources in the recovery Region, such as Amazon Relational Database Service (Amazon RDS) DB instances or Amazon DynamoDB tables. These data resources are therefore live and ready to serve requests in the recovery Region.

   To learn more about how AWS services operate across Regions, see this blog series on Creating a Multi-Region Application with AWS Services.

4. **Determine and implement how you will make your recovery Region ready for failover when needed (during a disaster event).**

   For multi-site active/active, failover means evacuating a Region, and relying on the remaining active Regions. In general, those Regions are ready to accept traffic. For Pilot Light and Warm Standby strategies, your recovery actions will need to deploy the missing resources, such as the EC2 instances in Figure 20, plus any other missing resources.

   For all of the above strategies you may need to promote read-only instances of databases to become the primary read/write instance.

   For backup and restore, restoring data from backup creates resources for that data such as EBS volumes, RDS DB instances, and DynamoDB tables. You also need to restore the infrastructure and deploy code. You can use AWS Backup to restore data in the recovery Region. See REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from

sources for more details. Rebuilding the infrastructure includes creating resources like EC2 instances in addition to the Amazon Virtual Private Cloud (Amazon VPC), subnets, and security groups needed. You can automate much of the restoration process. To learn how, see this blog post.

5. **Determine and implement how you will reroute traffic to failover when needed (during a disaster event).**

   This failover operation can be initiated either automatically or manually. Automatically initiated failover based on health checks or alarms should be used with caution since an unnecessary failover (false alarm) incurs costs such as non-availability and data loss. Manually initiated failover is therefore often used. In this case, you should still automate the steps for failover, so that the manual initiation is like the push of a button.

   There are several traffic management options to consider when using AWS services. One option is to use Amazon Route 53. Using Amazon Route 53, you can associate multiple IP endpoints in one or more AWS Regions with a Route 53 domain name. To implement manually initiated failover you can use Amazon Route 53 Application Recovery Controller, which provides a highly available data plane API to reroute traffic to the recovery Region. When implementing failover, use data plane operations and avoid control plane ones as described in REL11-BP04 Rely on the data plane and not the control plane during recovery.

   To learn more about this and other options see this section of the Disaster Recovery Whitepaper.

6. **Design a plan for how your workload will fail back.**

   Failback is when you return workload operation to the primary Region, after a disaster event has abated. Provisioning infrastructure and code to the primary Region generally follows the same steps as were initially used, relying on infrastructure as code and code deployment pipelines. The challenge with failback is restoring data stores, and ensuring their consistency with the recovery Region in operation.

   In the failed over state, the databases in the recovery Region are live and have the up-to-date data. The goal then is to re-synchronize from the recovery Region to the primary Region, ensuring it is up-to-date.

   Some AWS services will do this automatically. If using Amazon DynamoDB global tables, even if the table in the primary Region had become not available, when it comes back online, DynamoDB resumes propagating any pending writes. If using Amazon Aurora Global Database and using managed planned failover, then Aurora global database's existing replication topology

is maintained. Therefore, the former read/write instance in the primary Region will become a replica and receive updates from the recovery Region.

In cases where this is not automatic, you will need to re-establish the database in the primary Region as a replica of the database in the recovery Region. In many cases this will involve deleting the old primary database, and creating new replicas.

After a failover, if you can continue running in your recovery Region, consider making this the new primary Region. You would still do all the above steps to make the former primary Region into a recovery Region. Some organizations do a scheduled rotation, swapping their primary and recovery Regions periodically (for example every three months).

All of the steps required to fail over and fail back should be maintained in a playbook that is available to all members of the team, and is periodically reviewed.

When using Elastic Disaster Recovery, the service will assist in orchestrating and automating the failback process. For more details, see Performing a failback.

Level of effort for the Implementation Plan: High

Resources

Related best practices:

- the section called “REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources”
- the section called “REL11-BP04 Rely on the data plane and not the control plane during recovery”
- the section called “REL13-BP01 Define recovery objectives for downtime and data loss”

Related documents:

- AWS Architecture Blog: Disaster Recovery Series
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)
- Disaster recovery options in the cloud
- Build a serverless multi-region, active-active backend solution in an hour
- Multi-region serverless backend — reloaded

- RDS: Replicating a Read Replica Across Regions
- Route 53: Configuring DNS Failover
- S3: Cross-Region Replication
- What Is AWS Backup?
- What is Route 53 Application Recovery Controller?
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: Get Started - AWS
- APN Partner: partners that can help with disaster recovery
- AWS Marketplace: products that can be used for disaster recovery

Related videos:

- Disaster Recovery of Workloads on AWS
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Get Started with AWS Elastic Disaster Recovery | Amazon Web Services

Related examples:

- Well-Architected Lab - Disaster Recovery - Series of workshops illustrating DR strategies

REL13-BP03 Test disaster recovery implementation to validate the implementation

Regularly test failover to your recovery site to verify that it operates properly and that RTO and RPO are met.

Common anti-patterns:

- Never exercise failovers in production.

Benefits of establishing this best practice: Regularly testing you disaster recovery plan verifies that it will work when it needs to, and that your team knows how to perform the strategy.

Level of risk exposed if this best practice is not established: High

Implementation guidance

A pattern to avoid is developing recovery paths that are rarely exercised. For example, you might have a secondary data store that is used for read-only queries. When you write to a data store and the primary fails, you might want to fail over to the secondary data store. If you don't frequently test this failover, you might find that your assumptions about the capabilities of the secondary data store are incorrect. The capacity of the secondary, which might have been sufficient when you last tested, might be no longer be able to tolerate the load under this scenario. Our experience has shown that the only error recovery that works is the path you test frequently. This is why having a small number of recovery paths is best. You can establish recovery patterns and regularly test them. If you have a complex or critical recovery path, you still need to regularly exercise that failure in production to convince yourself that the recovery path works. In the example we just discussed, you should fail over to the standby regularly, regardless of need.

Implementation steps

1. Engineer your workloads for recovery. Regularly test your recovery paths. Recovery-oriented computing identifies the characteristics in systems that enhance recovery: isolation and redundancy, system-wide ability to roll back changes, ability to monitor and determine health, ability to provide diagnostics, automated recovery, modular design, and ability to restart. Exercise the recovery path to verify that you can accomplish the recovery in the specified time to the specified state. Use your runbooks during this recovery to document problems and find solutions for them before the next test.
2. For Amazon EC2-based workloads, use AWS Elastic Disaster Recovery to implement and launch drill instances for your DR strategy. AWS Elastic Disaster Recovery provides the ability to efficiently run drills, which helps you prepare for a failover event. You can also frequently launch of your instances using Elastic Disaster Recovery for test and drill purposes without redirecting the traffic.

Resources

Related documents:

- APN Partner: partners that can help with disaster recovery
- AWS Architecture Blog: Disaster Recovery Series
- AWS Marketplace: products that can be used for disaster recovery
- AWS Elastic Disaster Recovery

- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)
- AWS Elastic Disaster Recovery Preparing for Failover
- The Berkeley/Stanford recovery-oriented computing project
- What is AWS Fault Injection Simulator?

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS

Related examples:

- Well-Architected Lab - Testing for Resiliency

REL13-BP04 Manage configuration drift at the DR site or Region

Ensure that the infrastructure, data, and configuration are as needed at the DR site or Region. For example, check that AMIs and service quotas are up to date.

AWS Config continuously monitors and records your AWS resource configurations. It can detect drift and invoke AWS Systems Manager Automation to fix it and raise alarms. AWS CloudFormation can additionally detect drift in stacks you have deployed.

Common anti-patterns:

- Failing to make updates in your recovery locations, when you make configuration or infrastructure changes in your primary locations.
- Not considering potential limitations (like service differences) in your primary and recovery locations.

Benefits of establishing this best practice: Ensuring that your DR environment is consistent with your existing environment guarantees complete recovery.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Ensure that your delivery pipelines deliver to both your primary and backup sites. Delivery pipelines for deploying applications into production must distribute to all the specified disaster recovery strategy locations, including dev and test environments.
- Permit AWS Config to track potential drift locations. Use AWS Config rules to create systems that enforce your disaster recovery strategies and generate alerts when they detect drift.
  - Remediating Noncompliant AWS Resources by AWS Config Rules
  - AWS Systems Manager Automation
- Use AWS CloudFormation to deploy your infrastructure. AWS CloudFormation can detect drift between what your CloudFormation templates specify and what is actually deployed.
  - AWS CloudFormation: Detect Drift on an Entire CloudFormation Stack

Resources

Related documents:

- APN Partner: partners that can help with disaster recovery
- AWS Architecture Blog: Disaster Recovery Series
- AWS CloudFormation: Detect Drift on an Entire CloudFormation Stack
- AWS Marketplace: products that can be used for disaster recovery
- AWS Systems Manager Automation
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)
- How do I implement an Infrastructure Configuration Management solution on AWS?
- Remediating Noncompliant AWS Resources by AWS Config Rules

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

REL13-BP05 Automate recovery

Use AWS or third-party tools to automate system recovery and route traffic to the DR site or Region.

Based on configured health checks, AWS services, such as Elastic Load Balancing and AWS Auto Scaling, can distribute load to healthy Availability Zones while services, such as Amazon Route 53 and AWS Global Accelerator, can route load to healthy AWS Regions. Amazon Route 53 Application Recovery Controller helps you manage and coordinate failover using readiness check and routing control features. These features continually monitor your application's ability to recover from failures, so you can control application recovery across multiple AWS Regions, Availability Zones, and on premises.

For workloads on existing physical or virtual data centers or private clouds, AWS Elastic Disaster Recovery allows organizations to set up an automated disaster recovery strategy in AWS. Elastic Disaster Recovery also supports cross-Region and cross-Availability Zone disaster recovery in AWS.

Common anti-patterns:

- Implementing identical automated failover and failback can cause flapping when a failure occurs.

Benefits of establishing this best practice: Automated recovery reduces your recovery time by eliminating the opportunity for manual errors.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Automate recovery paths. For short recovery times, follow your disaster recovery plan to get your IT systems back online quickly in the case of a disruption.
  - Use Elastic Disaster Recovery for automated Failover and Failback. Elastic Disaster Recovery continuously replicates your machines (including operating system, system state configuration, databases, applications, and files) into a low-cost staging area in your target AWS account and preferred Region. In the case of a disaster, after choosing to recover using Elastic Disaster Recovery, Elastic Disaster Recovery automates the conversion of your replicated servers into fully provisioned workloads in your recovery Region on AWS.
    - Using Elastic Disaster Recovery for Failover and Failback
    - AWS Elastic Disaster Recovery resources

Resources

Related documents:

- APN Partner: partners that can help with disaster recovery

- AWS Architecture Blog: Disaster Recovery Series
- AWS Marketplace: products that can be used for disaster recovery
- AWS Systems Manager Automation
- AWS Elastic Disaster Recovery
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

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. You can find prescriptive guidance on implementation in the Performance Efficiency Pillar whitepaper.

Best practice areas

- Selection
- Review
- Monitoring
- Tradeoffs

Selection

Questions

- PERF 1. How do you select the best performing architecture?
- PERF 2. How do you select your compute solution?
- PERF 3. How do you select your storage solution?
- PERF 4. How do you select your database solution?
- PERF 5. How do you configure your networking solution?

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

Often, multiple approaches are required more efficient performance across a workload. Well-architected systems use multiple solutions and features to improve performance.

Best practices

- PERF01-BP01 Understand the available services and resources
- PERF01-BP02 Define a process for architectural choices
- PERF01-BP03 Factor cost requirements into decisions
- PERF01-BP04 Use policies or reference architectures
- PERF01-BP05 Use guidance from your cloud provider or an appropriate partner
- PERF01-BP06 Benchmark existing workloads
- PERF01-BP07 Load test your workload

PERF01-BP01 Understand the available services and resources

Learn about and understand the wide range of services and resources available in the cloud. Identify the relevant services and configuration options for your workload, and understand how to achieve optimal performance.

If you are evaluating an existing workload, you must generate an inventory of the various services resources it consumes. Your inventory helps you evaluate which components can be replaced with managed services and newer technologies.

Common anti-patterns:

- You use the cloud as a collocated data center.
- You use shared storage for all things that need persistent storage.
- You do not use automatic scaling.
- You use instance types that are closest matched, but larger where needed, to your current standards.
- You deploy and manage technologies that are available as managed services.

Benefits of establishing this best practice: By considering services you may be unfamiliar with, you may be able to greatly reduce the cost of infrastructure and the effort required to maintain

your services. You may be able to accelerate your time to market by deploying new services and features.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Inventory your workload software and architecture for related services: Gather an inventory of your workload and decide which category of products to learn more about. Identify workload components that can be replaced with managed services to increase performance and reduce operational complexity.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP02 Define a process for architectural choices

Use internal experience and knowledge of the cloud, or external resources such as published use cases, relevant documentation, or whitepapers, to define a process to choose resources and services. You should define a process that encourages experimentation and benchmarking with the services that could be used in your workload.

When you write critical user stories for your architecture, you should include performance requirements, such as specifying how quickly each critical story should run. For these critical stories, you should implement additional scripted user journeys to ensure that you have visibility into how these stories perform against your requirements.

Common anti-patterns:

- You assume your current architecture will become static and not be updated over time.
- You introduce architecture changes over time without justification.

Benefits of establishing this best practice: By having a defined process for making architectural changes, you permit using the gathered data to influence your workload design over time.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Select an architectural approach: Identify the kind of architecture that meets your performance requirements. Identify constraints, such as the media for delivery (desktop, web, mobile, IoT), legacy requirements, and integrations. Identify opportunities for reuse, including refactoring. Consult other teams, architecture diagrams, and resources such as AWS Solution Architects, AWS Reference Architectures, and AWS Partners to help you choose an architecture.

Define performance requirements: Use the customer experience to identify the most important metrics. For each metric, identify the target, measurement approach, and priority. Define the customer experience. Document the performance experience required by customers, including how customers will judge the performance of the workload. Prioritize experience concerns for critical user stories. Include performance requirements and implement scripted user journeys to ensure that you know how the stories perform against your requirements.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP03 Factor cost requirements into decisions

Workloads often have cost requirements for operation. Use internal cost controls to select resource types and sizes based on predicted resource need.

Determine which workload components could be replaced with fully managed services, such as managed databases, in-memory caches, and ETL services. Reducing your operational workload allows you to focus resources on business outcomes.

For cost requirement best practices, refer to the *Cost-Effective Resources* section of the Cost Optimization Pillar whitepaper.

Common anti-patterns:

- You only use one family of instances.
- You do not evaluate licensed solutions versus open-source solutions
- You only use block storage.
- You deploy common software on EC2 instances and Amazon EBS or ephemeral volumes that are available as a managed service.

Benefits of establishing this best practice: Considering cost when making your selections will allow you to allow other investments.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Optimize workload components to reduce cost: Right size workload components and allow elasticity to reduce cost and maximize component efficiency. Determine which workload

components can be replaced with managed services when appropriate, such as managed databases, in-memory caches, and reverse proxies.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- AWS Compute Optimizer

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture
- Optimize performance and cost for your AWS compute (CMP323-R1)

Related examples:

- AWS Samples
- AWS SDK Examples
- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

PERF01-BP04 Use policies or reference architectures

Maximize performance and efficiency by evaluating internal policies and existing reference architectures and using your analysis to select services and configurations for your workload.

Common anti-patterns:

- You allow wide use of technology selection that may impact the management overhead of your company.

Benefits of establishing this best practice: Establishing a policy for architecture, technology, and vendor choices will allow decisions to be made quickly.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Deploy your workload using existing policies or reference architectures: Integrate the services into your cloud deployment, then use your performance tests to ensure that you can continue to meet your performance requirements.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP05 Use guidance from your cloud provider or an appropriate partner

Use cloud company resources, such as solutions architects, professional services, or an appropriate partner to guide your decisions. These resources can help review and improve your architecture for optimal performance.

Reach out to AWS for assistance when you need additional guidance or product information. AWS Solutions Architects and AWS Professional Services provide guidance for solution implementation. AWS Partners provide AWS expertise to help you unlock agility and innovation for your business.

Common anti-patterns:

- You use AWS as a common data center provider.
- You use AWS services in a manner that they were not designed for.

Benefits of establishing this best practice: Consulting with your provider or a partner will give you confidence in your decisions.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Reach out to AWS resources for assistance: AWS Solutions Architects and Professional Services provide guidance for solution implementation. APN Partners provide AWS expertise to help you unlock agility and innovation for your business.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP06 Benchmark existing workloads

Benchmark the performance of an existing workload to understand how it performs on the cloud. Use the data collected from benchmarks to drive architectural decisions.

Use benchmarking with synthetic tests and real-user monitoring to generate data about how your workload's components perform. Benchmarking is generally quicker to set up than load testing and is used to evaluate the technology for a particular component. Benchmarking is often used at the start of a new project, when you lack a full solution to load test.

You can either build your own custom benchmark tests, or you can use an industry standard test, such as TPC-DS to benchmark your data warehousing workloads. Industry benchmarks are helpful when comparing environments. Custom benchmarks are useful for targeting specific types of operations that you expect to make in your architecture.

When benchmarking, it is important to pre-warm your test environment to ensure valid results. Run the same benchmark multiple times to ensure that you've captured any variance over time.

Because benchmarks are generally faster to run than load tests, they can be used earlier in the deployment pipeline and provide faster feedback on performance deviations. When you evaluate a significant change in a component or service, a benchmark can be a quick way to see if you can justify the effort to make the change. Using benchmarking in conjunction with load testing is important because load testing informs you about how your workload will perform in production.

Common anti-patterns:

- You rely on common benchmarks that are not indicative of your workload characteristics.
- You rely on customer feedback and perceptions as your only benchmark.

Benefits of establishing this best practice: Benchmarking your current implementation allows you to measure the improvement in performance.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Monitor performance during development: Implement processes that provide visibility into performance as your workload evolves.

Integrate into your delivery pipeline: Automatically run load tests in your delivery pipeline. Compare the test results against pre-defined key performance indicators (KPIs) and thresholds to ensure that you continue to meet performance requirements.

Test user journeys: Use synthetic or sanitized versions of production data (remove sensitive or identifying information) for load testing. Exercise your entire architecture by using replayed or pre-programmed user journeys through your application at scale.

Real-user monitoring: Use CloudWatch RUM to help you collect and view client-side data about your application performance. Use this data to help establish your real-user performance benchmarks.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- AWS Samples
- AWS SDK Examples
- Distributed Load Tests
- Measure page load time with Amazon CloudWatch Synthetics

- Amazon CloudWatch RUM Web Client

PERF01-BP07 Load test your workload

Deploy your latest workload architecture on the cloud using different resource types and sizes. Monitor the deployment to capture performance metrics that identify bottlenecks or excess capacity. Use this performance information to design or improve your architecture and resource selection.

Load testing uses your *actual* workload so that you can see how your solution performs in a production environment. Load tests must be run using synthetic or sanitized versions of production data (remove sensitive or identifying information). Use replayed or pre-programmed user journeys through your workload at scale that exercise your entire architecture. Automatically carry out load tests as part of your delivery pipeline, and compare the results against pre-defined KPIs and thresholds. This ensures that you continue to achieve required performance.

Common anti-patterns:

- You load test individual parts of your workload but not your entire workload.
- You load test on infrastructure that is not the same as your production environment.
- You only conduct load testing to your expected load and not beyond, to help foresee where you may have future problems.
- Performing load testing without informing AWS Support, and having your test defeated as it looks like a denial of service event.

Benefits of establishing this best practice: Measuring your performance under a load test will show you where you will be impacted as load increases. This can provide you with the capability of anticipating needed changes before they impact your workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Validate your approach with load testing: Load test a proof-of-concept to find out if you meet your performance requirements. You can use AWS services to run production-scale environments to test your architecture. Because you only pay for the test environment when it is needed, you can carry out full-scale testing at a fraction of the cost of using an on-premises environment.

Monitor metrics: Amazon CloudWatch can collect metrics across the resources in your architecture. You can also collect and publish custom metrics to surface business or derived metrics. Use CloudWatch or third-party solutions to set alarms that indicate when thresholds are breached.

Test at scale: Load testing uses your actual workload so you can see how your solution performs in a production environment. You can use AWS services to run production-scale environments to test your architecture. Because you only pay for the test environment when it is needed, you can run full-scale testing at a lower cost than using an on-premises environment. Take advantage of the AWS Cloud to test your workload to discover where it fails to scale, or if it scales in a non-linear way. For example, use Spot Instances to generate loads at low cost and discover bottlenecks before they are experienced in production.

Resources

Related documents:

- AWS CloudFormation
- Building AWS CloudFormation Templates using CloudFormer
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Distributed Load Testing on AWS

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Distributed Load Testing on AWS

PERF 2. How do you select your compute solution?

The most effective compute solution for a workload varies based on application design, usage patterns, and configuration settings. Architectures can use different compute solutions for various

components and activates different features to improve performance. Selecting the wrong compute solution for an architecture can lead to lower performance efficiency.

Best practices

- PERF02-BP01 Evaluate the available compute options
- PERF02-BP02 Understand the available compute configuration options
- PERF02-BP03 Collect compute-related metrics
- PERF02-BP04 Determine the required configuration by right-sizing
- PERF02-BP05 Use the available elasticity of resources
- PERF02-BP06 Continually evaluate compute needs based on metrics

PERF02-BP01 Evaluate the available compute options

Understand how your workload can benefit from the use of different compute options, such as instances, containers and functions.

Desired outcome: By understanding all of the compute options available, you will be aware of the opportunities to increase performance, reduce unnecessary infrastructure costs, and lower the operational effort required to maintain your workload. You can also accelerate your time to market when you deploy new services and features.

Common anti-patterns:

- In a post-migration workload, using the same compute solution that was being used on premises.
- Lacking awareness of the cloud compute solutions and how those solutions might improve your compute performance.
- Oversizing an existing compute solution to meet scaling or performance requirements, when an alternative compute solution would align to your workload characteristics more precisely.

Benefits of establishing this best practice: By identifying the compute requirements and evaluating the available compute solutions, business stakeholders and engineering teams will understand the benefits and limitations of using the selected compute solution. The selected compute solution should fit the workload performance criteria. Key criteria include processing needs, traffic patterns, data access patterns, scaling needs, and latency requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Understand the virtualization, containerization, and management solutions that can benefit your workload and meet your performance requirements. A workload can contain multiple types of compute solutions. Each compute solution has differing characteristics. Based on your workload scale and compute requirements, a compute solution can be selected and configured to meet your needs. The cloud architect should learn the advantages and disadvantages of instances, containers, and functions. The following steps will help you through how to select your compute solution to match your workload characteristics and performance requirements.

Type	Server	Containers	Function
AWS service	Amazon Elastic Compute Cloud (Amazon EC2)	Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS)	AWS Lambda
Key Characteristics	Has dedicated option for hardware license requirements, Placement Options, and a large selection of different instance families based on compute metrics	Easy deployment, consistent environments, runs on top of EC2 instances, Scalable	Short runtime (15 minutes or less), maximum memory and CPU are not as high as other services, Managed hardware layer, Scales to millions of concurrent requests
Common use-cases	Lift and shift migrations, monolithic application, hybrid environments, enterprise applications	Microservices, hybrid environments,	Microservices, event-driven applications

Implementation steps:

1. Select the location of where the compute solution must reside by evaluating the section called "PERF05-BP06 Choose your workload's location based on network requirements". This location will limit the types of compute solution available to you.
2. Identify the type of compute solution that works with the location requirement and application requirements
    a. Amazon Elastic Compute Cloud (Amazon EC2) virtual server instances come in a wide variety of different families and sizes. They offer a wide variety of capabilities, including solid state drives (SSDs) and graphics processing units (GPUs). EC2 instances offer the greatest flexibility on instance choice. When you launch an EC2 instance, the instance type that you specify determines the hardware of your instance. Each instance type offers different compute, memory, and storage capabilities. Instance types are grouped in instance families based on these capabilities. Typical use cases include: running enterprise applications, high performance computing (HPC), training and deploying machine learning applications and running cloud native applications.
    b. Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service that allows you to automatically run and manage containers on a cluster of EC2 instances or serverless instances using AWS Fargate. You can use Amazon ECS with other services such as Amazon Route 53, Secrets Manager, AWS Identity and Access Management (IAM), and Amazon CloudWatch. Amazon ECS is recommended if your application is containerized and your engineering team prefers Docker containers.
    c. Amazon Elastic Kubernetes Service (Amazon EKS) is a fully managed Kubernetes service. You can choose to run your EKS clusters using AWS Fargate, removing the need to provision and manage servers. Managing Amazon EKS is simplified due to integrations with AWS Services such as Amazon CloudWatch, Auto Scaling Groups, AWS Identity and Access Management (IAM), and Amazon Virtual Private Cloud (VPC). When using containers, you must use compute metrics to select the optimal type for your workload, similar to how you use compute metrics to select your EC2 or AWS Fargate instance types. Amazon EKS is recommended if your application is containerized and your engineering team prefers Kubernetes over Docker containers.
    d. You can use AWS Lambda to run code that supports the allowed runtime, memory, and CPU options. Simply upload your code, and AWS Lambda will manage everything required to run and scale that code. You can set up your code to automatically run from other AWS services

or call it directly. Lambda is recommended for short running, microservice architectures developed for the cloud.

3. After you have experimented with your new compute solution, plan your migration and validate your performance metrics. This is a continual process, see the section called "PERF02-BP04 Determine the required configuration by right-sizing".

Level of effort for the implementation plan: If a workload is moving from one compute solution to another, there could be a *moderate* level of effort involved in refactoring the application.

Resources

Related documents:

- Cloud Compute with AWS
- EC2 Instance Types
- Processor State Control for Your EC2 Instance
- EKS Containers: EKS Worker Nodes
- Amazon ECS Containers: Amazon ECS Container Instances
- Functions: Lambda Function Configuration
- Prescriptive Guidance for Containers
- Prescriptive Guidance for Serverless

Related videos:

- How to choose compute option for startups
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Amazon EC2 foundations (CMP211-R2)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Deliver high-performance ML inference with AWS Inferentia (CMP324-R1)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)

Related examples:

- Migrating the web application to containers

- Run a Serverless Hello World

PERF02-BP02 Understand the available compute configuration options

Each compute solution has options and configurations available to you to support your workload characteristics. Learn how various options complement your workload, and which configuration options are best for your application. Examples of these options include instance family, sizes, features (GPU, I/O), bursting, time-outs, function sizes, container instances, and concurrency.

Desired outcome: The workload characteristics including CPU, memory, network throughput, GPU, IOPS, traffic patterns, and data access patterns are documented and used to configure the compute solution to match the workload characteristics. Each of these metrics plus custom metrics specific to your workload are recorded, monitored, and then used to optimize the compute configuration to best meet the requirements.

Common anti-patterns:

- Using the same compute solution that was being used on premises.
- Not reviewing the compute options or instance family to match workload characteristics.
- Oversizing the compute to ensure bursting capability.
- You use multiple compute management platforms for the same workload.

Benefits of establishing this best practice: Be familiar with the AWS compute offerings so that you can determine the correct solution for each of your workloads. After you have selected the compute offerings for your workload, you can quickly experiment with those compute offerings to determine how well they meet your workload needs. A compute solution that is optimized to meet your workload characteristics will increase your performance, lower your cost and increase your reliability.

Level of risk exposed if this best practice is not established: High

Implementation guidance

If your workload has been using the same compute option for more than four weeks and you anticipate that the characteristics will remain the same in the future, you can use AWS Compute Optimizer to provide a recommendation to you based on your compute characteristics. If AWS Compute Optimizer is not an option due to lack of metrics, a non-supported instance type or

a foreseeable change in your characteristics then you must predict your metrics based on load testing and experimentation.

Implementation steps:

1. Are you running on EC2 instances or containers with the EC2 Launch Type?
   a. Can your workload use GPUs to increase performance?
      i. Accelerated Computing instances are GPU-based instances that provide the highest performance for machine learning training, inference and high performance computing.
   b. Does your workload run machine learning inference applications?
      i. AWS Inferentia (Inf1) — Inf1 instances are built to support machine learning inference applications. Using Inf1 instances, customers can run large-scale machine learning inference applications, such as image recognition, speech recognition, natural language processing, personalization, and fraud detection. You can build a model in one of the popular machine learning frameworks, such as TensorFlow, PyTorch, or MXNet and use GPU instances, to train your model. After your machine learning model is trained to meet your requirements, you can deploy your model on Inf1 instances by using AWS Neuron, a specialized software development kit (SDK) consisting of a compiler, runtime, and profiling tools that optimize the machine learning inference performance of Inferentia chips.
   c. Does your workload integrate with the low-level hardware to improve performance?
      i. Field Programmable Gate Arrays (FPGA) — Using FPGAs, you can optimize your workloads by having custom hardware-accelerated operation for your most demanding workloads. You can define your algorithms by leveraging supported general programming languages such as C or Go, or hardware-oriented languages such as Verilog or VHDL.
   d. Do you have at least four weeks of metrics and can predict that your traffic pattern and metrics will remain about the same in the future?
      i. Use Compute Optimizer to get a machine learning recommendation on which compute configuration best matches your compute characteristics.
   e. Is your workload performance constrained by the CPU metrics?
      i. Compute-optimized instances are ideal for the workloads that require high performing processors.
   f. Is your workload performance constrained by the memory metrics?
      i. Memory-optimized instances deliver large amounts of memory to support memory intensive workloads.
   g. Is your workload performance constrained by IOPS?

i. Storage-optimized instances are designed for workloads that require high, sequential read and write access (IOPS) to local storage.

h. Do your workload characteristics represent a balanced need across all metrics?

i. Does your workload CPU need to burst to handle spikes in traffic?

A. Burstable Performance instances are similar to Compute Optimized instances except they offer the ability to burst past the fixed CPU baseline identified in a compute-optimized instance.

ii. General Purpose instances provide a balance of all characteristics to support a variety of workloads.

i. Is your compute instance running on Linux and constrained by network throughput on the network interface card?

i. Review Performance Question 5, Best Practice 2: Evaluate available networking features to find the right instance type and family to meet your performance needs.

j. Does your workload need consistent and predictable instances in a specific Availability Zone that you can commit to for a year?

i. Reserved Instances confirms capacity reservations in a specific Availability Zone. Reserved Instances are ideal for required compute power in a specific Availability Zone.

k. Does your workload have licenses that require dedicated hardware?

i. Dedicated Hosts support existing software licenses and help you meet compliance requirements.

l. Does your compute solution burst and require synchronous processing?

i. On-Demand Instances let you use the compute capacity by the hour or second with no long-term commitment. These instances are good for bursting above performance baseline needs.

m. Is your compute solution stateless, fault-tolerant, and asynchronous?

i. Spot Instances let you take advantage of unused instance capacity for your stateless, fault-tolerant workloads.

2. Are you running containers on Fargate?

a. Is your task performance constrained by the memory or CPU?

i. Use the Task Size to adjust your memory or CPU.

b. Is your performance being affected by your traffic pattern bursts?

i. Use the Auto Scaling configuration to match your traffic patterns.

3. Is your compute solution on Lambda?
    a. Do you have at least four weeks of metrics and can predict that your traffic pattern and metrics will remain about the same in the future?
        i. Use Compute Optimizer to get a machine learning recommendation on which compute configuration best matches your compute characteristics.
    b. Do you not have enough metrics to use AWS Compute Optimizer?
        i. If you do not have metrics available to use Compute Optimizer, use AWS Lambda Power Tuning to help select the best configuration.
    c. Is your function performance constrained by the memory or CPU?
        i. Configure your Lambda memory to meet your performance needs metrics.
    d. Is your function timing out when running?
        i. Change the timeout settings
    e. Is your function performance constrained by bursts of activity and concurrency?
        i. Configure the concurrency settings to meet your performance requirements.
    f. Does your function run asynchronously and is failing on retries?
        i. Configure the maximum age of the event and the maximum retry limit in the asynchronous configuration settings.

Level of effort for the implementation plan:

To establish this best practice, you must be aware of your current compute characteristics and metrics. Gathering those metrics, establishing a baseline and then using those metrics to identify the ideal compute option is a *low* to *moderate* level of effort. This is best validated by load tests and experimentation.

Resources

Related documents:

- Cloud Compute with AWS
- AWS Compute Optimizer
- EC2 Instance Types
- Processor State Control for Your EC2 Instance
- EKS Containers: EKS Worker Nodes

- Amazon ECS Containers: Amazon ECS Container Instances
- Functions: Lambda Function Configuration

Related videos:

- Amazon EC2 foundations (CMP211-R2)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Optimize performance and cost for your AWS compute (CMP323-R1)

Related examples:

- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

PERF02-BP03 Collect compute-related metrics

To understand how your compute resources are performing, you must record and track the utilization of various systems. This data can be used to make more accurate determinations about resource requirements.

Workloads can generate large volumes of data such as metrics, logs, and events. Determine if your existing storage, monitoring, and observability service can manage the data generated. Identify which metrics reflect resource utilization and can be collected, aggregated, and correlated on a single platform across. Those metrics should represent all your workload resources, applications, and services, so you can easily gain system-wide visibility and quickly identify performance improvement opportunities and issues.

Desired outcome: All metrics related to the compute-related resources are identified, collected, aggregated, and correlated on a single platform with retention implemented to support cost and operational goals.

Common anti-patterns:

- You only use manual log file searching for metrics.
- You only publish metrics to internal tools.
- You only use the default metrics recorded by your selected monitoring software.
- You only review metrics when there is an issue.

Benefits of establishing this best practice: To monitor the performance of your workloads, you must record multiple performance metrics over a period of time. These metrics allow you to detect anomalies in performance. They will also help gauge performance against business metrics to ensure that you are meeting your workload needs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify, collect, aggregate, and correlate compute-related metrics. Using a service such as Amazon CloudWatch, can make the implementation quicker and easier to maintain. In addition to the default metrics recorded, identify and track additional system-level metrics within your workload. Record data such as CPU utilization, memory, disk I/O, and network inbound and outbound metrics to gain insight into utilization levels or bottlenecks. This data is crucial to understand how the workload is performing and how the compute solution is utilized. Use these metrics as part of a data-driven approach to actively tune and optimize your workload's resources.

Implementation steps:

1. Which compute solution metrics are important to track?
    a. EC2 default metrics
    b. Amazon ECS default metrics
    c. EKS default metrics
    d. Lambda default metrics
    e. EC2 memory and disk metrics
2. Do I currently have an approved logging and monitoring solution?
    a. Amazon CloudWatch
    b. AWS Distro for OpenTelemetry
    c. Amazon Managed Service for Prometheus
3. Have I identified and configured my data retention policies to match my security and operational goals?
    a. Default data retention for CloudWatch metrics
    b. Default data retention for CloudWatch Logs
4. How do you deploy your metric and log aggregation agents?
    a. AWS Systems Manager automation

b. OpenTelemetry Collector

Level of effort for the Implementation Plan: There is a *medium* level of effort to identify, track, collect, aggregate, and correlate metrics from all compute resources.

Resources

Related documents:

- Amazon CloudWatch documentation
- Collect metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch Agent
- Accessing Amazon CloudWatch Logs for AWS Lambda
- Using CloudWatch Logs with container instances
- Publish custom metrics
- AWS Answers: Centralized Logging
- AWS Services That Publish CloudWatch Metrics
- Monitoring Amazon EKS on AWS Fargate

Related videos:

- Application Performance Management on AWS
- Build a Monitoring Plan

Related examples:

- Level 100: Monitoring with CloudWatch Dashboards
- Level 100: Monitoring Windows EC2 instance with CloudWatch Dashboards
- Level 100: Monitoring an Amazon Linux EC2 instance with CloudWatch Dashboards

PERF02-BP04 Determine the required configuration by right-sizing

Analyze the various performance characteristics of your workload and how these characteristics relate to memory, network, I/O, and CPU usage. Use this data to choose resources that best match

your workload's profile. For example, a memory-intensive workload like a database may benefit from a higher memory per core ratio. However, a compute intensive workload may need a higher core count and frequency, but can be satisfied with a lower amount of memory per core.

Common anti-patterns:

- You choose an instance with the largest values across all performance characteristics available for all workloads.
- You standardize all instances types to one type for ease of management.
- You optimize against standard synthetic benchmarks without validating the actual requirements of a particular workload.
- You keep the same infrastructure for a long period of time without reevaluating and integrating new offerings.

Benefits of establishing this best practice: When you are familiar with the requirements of your workload, you can compare these needs with available compute offerings and quickly experiment to determine which ones meet the needs of your workload most efficiently. This allows for optimal performance without overpaying for resources not required.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Modify your workload configuration by right sizing. To optimize performance, overall efficiency, and cost effectiveness, determine first which resources your workload needs. Choose memory-optimized instances, such as the R-family of instances, for memory-intensive workloads like a database. For workloads that require higher compute capacity, choose the C-family of instances, or choose instances with higher core counts or higher core frequency. Choose I/O performance based on the needs of your workload instead of comparing against standard, synthetic benchmarks. For higher I/O performance, choose instances from the I-family of instances, select I/O optimized Amazon EBS volumes, or choose instances with instance store. For more detail on particular instance types, see Amazon EC2 instance types.

Right sizing verifies that your workloads perform as well as possible while not overpaying on resources not needed.

Implementation steps

- Know your workload or analyze its resource requirements.

- Evaluate workloads separately. The AWS Cloud gives you flexibility and agility to right-size each workload on its own without needing to compromise.
- Create test environments to find the best match of compute offerings against your workload.
- Continually reevaluate new compute offerings, and compare against your workload's needs.
- Routinely review new service offers for better price performance.
- Regularly conduct Well-Architected Framework Reviews.

Resources

Related best practices:

- PERF02-BP03 Collect compute-related metrics
- PERF02-BP06 Continually evaluate compute needs based on metrics

Related documents:

- AWS Compute Optimizer
- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Amazon ECS Containers: Amazon ECS Container Instances
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Functions: Lambda Function Configuration

Related videos:

- Amazon EC2 foundations (CMP211-R2)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- How to choose compute option for startups
- Optimize performance and cost for your AWS compute (CMP323-R1)

Related examples:

- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

PERF02-BP05 Use the available elasticity of resources

The cloud provides the flexibility to expand and reduce your resources dynamically through a variety of mechanisms to meet changes in demand. Combining this elasticity with compute-related metrics, a workload can automatically respond to changes to use the resources it needs and only the resources it needs.

Common anti-patterns:

- You overprovision to cover possible spikes.
- You react to alarms by manually increasing capacity.
- You increase capacity without considering provisioning time.
- You leave increased capacity after a scaling event instead of scaling back down.
- You monitor metrics that don't directly reflect your workloads true requirements.

Benefits of establishing this best practice: Demand can be fixed, variable, follow a pattern or be spiky. Matching supply to demand delivers the lowest cost for a workload. Monitoring, testing, and configuring workload elasticity will optimize performance, save money, and improve reliability as usage demands change. Although a manual approach to this is possible, it is impractical at larger scales. An automated and metrics-based approach assures resources meet demands and any given time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Metric based automation should be used to take advantage of elasticity with the goal that the supply of resources you have matches the demand of the resources your workload requires. For example, you can use Amazon CloudWatch metrics to monitor your resources, or use Amazon CloudWatch metrics for your Auto Scaling groups.

Combined with compute-related metrics, a workload can automatically respond to changes and use the optimal set of resources to achieve its goal. You also must plan for provisioning time and potential resource failures.

Instances, containers, and functions provide mechanisms for elasticity either as a feature of the service, in the form of Application Auto Scaling, or in combination with Amazon EC2 Auto Scaling. Use elasticity in your architecture to verify that you have sufficient capacity to meet performance requirements at a wide variety of scales of use.

Validate your metrics for scaling up or down elastic resources against the type of workload being deployed. As an example, if you are deploying a video transcoding application, 100% CPU utilization is expected and should not be your primary metric. Alternatively, you can measure against the queue depth of transcoding jobs waiting to scale your instance types.

Workload deployments need to handle both scale up and scale down events. Scaling down workload components safely is as critical as scaling up resources when demand dictates.

Create test scenarios for scaling events to verify that the workload behaves as expected.

Implementation steps

- Leverage historical data to analyze your workload's resource demands over time. Ask specific questions like:
  - Is your workload steady and increasing over time at a known rate?
  - Does your workload increase and decrease in seasonal, repeatable patterns?
  - Is your workload spiky? Can the spikes be anticipated or predicted?
- Leverage monitoring services and historical data as much as possible.
- Tagging resources can help with monitoring. When using tags, refer to tagging best practices. Additionally, tags can help you manage, identify, and organize resources.
- With AWS, you can use a number of different approaches to match supply with demand. The cost optimization pillar best practices (COST09-BP01 through COST09-03) describe how to use the following approaches to cost:
  - COST09-BP01 Perform an analysis on the workload demand
  - COST09-BP02 Implement a buffer or throttle to manage demand
  - COST09-BP03 Supply resources dynamically
- Create test scenarios for scale down events to verify that the workload behaves as expected.
- Most non-production instances should be stopped when they are not being used.

- For storage needs when using Amazon Elastic Block Store (Amazon EBS), take advantage of volume-based elasticity.
- For Amazon Elastic Compute Cloud (Amazon EC2), consider using Auto Scaling groups, which allow you to optimize performance and cost by automatically increasing the number of compute instances during demand spikes and decreasing capacity when demand decreases.

Resources

Related best practices:

- PERF02-BP03 Collect compute-related metrics
- PERF02-BP04 Determine the required configuration by right-sizing
- PERF02-BP06 Continually evaluate compute needs based on metrics

Related documents:

- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Amazon ECS Containers: Amazon ECS Container Instances
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Functions: Lambda Function Configuration

Related videos:

- Amazon EC2 foundations (CMP211-R2)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system

Related examples:

- Amazon EC2 Auto Scaling Group Examples
- Amazon EFS Tutorials

PERF02-BP06 Continually evaluate compute needs based on metrics

Use a data-driven approach to continually evaluate and optimize the compute resources for your workload over time.

Desired outcome: Use system-level metrics to actively monitor the behavior and requirements of your workload over time. Evaluate the demands of your workload against available resources based on the collected data, and make changes to your compute environment to best match your workload's profile. For example, a workload might be observed over time to be more memory-intensive than initially specified, so moving to a different instance family or size could improve both performance and efficiency.

Common anti-patterns:

- Monitoring system-level metrics to gain insight into your workload and not re-evaluating compute needs.
- Architecting your compute needs for peak workload requirements.
- Oversizing the existing compute solution to meet scaling or performance requirements when moving to an alternative compute solution would more efficiently match your workload characteristics.

Benefits of establishing this best practice: Optimized compute resources based on real-world data and your desired balance of cost and performance.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use a data-driven approach to optimize compute resources based on observed workload behavior. To achieve maximum performance and efficiency, use the data gathered over time from your workload to continually tune and optimize your resources. Look at the trends in your workload's usage of current resources and determine where you can make changes to better match your workload's needs. When resources are over-committed, system performance degrades, and when resources are not adequately used, the system is operating less efficiently and at a higher cost.

To optimize performance and resource utilization, you need a unified operational view, real-time granular data, and a historical reference. You can create automated dashboards to visualize this data and derive operational and utilization insights.

Implementation steps

1. Collect compute-related metrics over time.
2. Compare workload metrics against available resources in your selected compute solution.
3. Determine any required configuration changes by right-sizing the existing solution or evaluating alternative compute solutions.

Resources

Related best practices:

- PERF02-BP01 Evaluate the available compute options
- PERF02-BP02 Understand the available compute configuration options
- PERF02-BP03 Collect compute-related metrics
- PERF02-BP04 Determine the required configuration by right-sizing

Related documents:

- Cloud Compute with AWS
- AWS Compute Optimizer
- EC2 Instance Types
- Amazon ECS Containers: Amazon ECS Container Instances
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Best practices for working with AWS Lambda functions

Related videos:

- Amazon EC2 foundations (CMP211-R2)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Selecting and optimizing Amazon EC2 instances

Related examples:

- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

PERF 3. How do you select your storage solution?

The most effective storage solution for a system varies based on the kind of access operation (block, file, or object), patterns of access (random or sequential), required throughput, frequency of access (online, offline, archival), frequency of update (WORM, dynamic), and availability and durability constraints. Well-architected systems use multiple storage solutions and activates different features to improve performance and use resources efficiently.

Best practices

- PERF03-BP01 Understand storage characteristics and requirements
- PERF03-BP02 Evaluate available configuration options
- PERF03-BP03 Make decisions based on access patterns and metrics

PERF03-BP01 Understand storage characteristics and requirements

Identify and document the workload storage needs and define the storage characteristics of each location. Examples of storage characteristics include: shareable access, file size, growth rate, throughput, IOPS, latency, access patterns, and persistence of data. Use these characteristics to evaluate if block, file, object, or instance storage services are the most efficient solution for your storage needs.

Desired outcome: Identify and document the storage requirements per storage requirement and evaluate the available storage solutions. Based on the key storage characteristics, your team will understand how the selected storage services will benefit your workload performance. Key criteria include data access patterns, growth rate, scaling needs, and latency requirements.

Common anti-patterns:

- You only use one storage type, such as Amazon Elastic Block Store (Amazon EBS), for all workloads.
- You assume that all workloads have similar storage access performance requirements.

Benefits of establishing this best practice: Selecting the storage solution based on the identified and required characteristics will help improve your workloads performance, decrease costs and

lower your operational efforts in maintaining your workload. Your workload performance will benefit from the solution, configuration, and location of the storage service.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify your workload's most important storage performance metrics and implement improvements as part of a data-driven approach, using benchmarking or load testing. Use this data to identify where your storage solution is constrained, and examine configuration options to improve the solution. Determine the expected growth rate for your workload and choose a storage solution that will meet those rates. Research the AWS storage offerings to determine the correct storage solution for your various workload needs. Provisioning storage solutions in AWS increases the opportunity for you to test storage offerings and determine if they are appropriate for your workload needs.

AWS service	Key characteristics	Common use cases
Amazon S3	99.999999999% durability, unlimited growth, accessible from anywhere, several cost models based on access and resiliency	Cloud-native application data, data archiving, and backups, analytics, data lakes, static website hosting, IoT data
Amazon Glacier	Seconds to hours latency, unlimited growth, lowest cost, long-term storage	Data archiving, media archives, long-term backup retention.
Amazon EBS	Storage size requires management and monitorin g, low latency, persistent storage, 99.8% to 99.9% durability, most volume types are accessible only from one EC2 instance.	COTS applications, I/O intensive applications, relational and NoSQL databases, backup and recovery
EC2 Instance Store	Pre-determined storage size, lowest latency, not persisted,	COTS applications, I/O intensive applications, in-memory data store

AWS service	Key characteristics	Common use cases
	accessible only from one EC2 instance	
Amazon EFS	99.999999999% durability, unlimited growth, accessible by multiple compute services	Modernized applications sharing files across multiple compute services, file storage for scaling content management systems
Amazon FSx	Supports four file systems (NetApp, OpenZFS, Windows File Server, and Amazon FSx for Lustre), storage available different per file system, accessible by multiple compute services	Cloud native workloads , private cloud bursting, migrated workloads that require a specific file system, VMC, ERP systems, on-premises file storage and backups
Snow family	Portable devices, 256-bit encryption, NFS endpoint, on-board computing, TBs of storage	Migrating data to the cloud, storage, and computing in extreme on-premises conditions, disaster recovery, remote data collection
AWS Storage Gateway	Provides low-latency on-premises access to cloud-backed storage, fully managed on-premises cache	On-premises data to cloud migrations, populate cloud data lakes from on-premises sources, modernized file sharing.

Implementation steps:

1. Use benchmarking or load tests to collect the key characteristics of your storage needs. Key characteristics include:
   a. Shareable (what components access this storage)
   b. Growth rate

c. Throughput

d. Latency

e. I/O size

f. Durability

g. Access patterns (reads vs writes, frequency, spikey, or consistent)

2. Identify the type of storage solution that supports your storage characteristics.

   a. Amazon S3 is an object storage service with unlimited scalability, high availability, and multiple options for accessibility. Transferring and accessing objects in and out of Amazon S3 can use a service, such as Transfer Acceleration or Access Points to support your location, security needs, and access patterns. Use the Amazon S3 performance guidelines to help you optimize your Amazon S3 configuration to meet your workload performance needs.

   b. Amazon Glacier is a storage class of Amazon S3 built for data archiving. You can choose from three archiving solutions ranging from millisecond access to 5-12 hour access with different cost and security options. Amazon Glacier can help you meet performance requirements by implementing a data lifecycle that supports your business requirements and data characteristics.

   c. Amazon Elastic Block Store (Amazon EBS) is a high-performance block storage service designed for Amazon Elastic Compute Cloud (Amazon EC2). You can choose from SSD- or HDD-based solutions with different characteristics that prioritize IOPS or throughput. EBS volumes are well suited for high-performance workloads, primary storage for file systems, databases, or applications that can only access attached stage systems.

   d. Amazon EC2 Instance Store is similar to Amazon EBS as it attaches to an Amazon EC2 instance however, the Instance Store is only temporary storage that should ideally be used as a buffer, cache, or other temporary content. You cannot detach an Instance Store and all data is lost if the instance shuts down. Instance Stores can be used for high I/O performance and low latency use cases where data doesn't need to persist.

   e. Amazon Elastic File System (Amazon EFS) is a mountable file system that can be accessed by multiple types of compute solutions. Amazon EFS automatically grows and shrinks storage and is performance-optimized to deliver consistent low latencies. EFS has two performance configuration modes: General Purpose and Max I/O. General Purpose has a sub-millisecond read latency and a single-digit millisecond write latency. The Max I/O feature can support thousands of compute instance requiring a shared file system. Amazon EFS supports two throughput modes: Bursting and Provisioned. A workload that experiences a spikey access

pattern will benefit from the bursting throughput mode while a workload that is consistently high would be performant with a provisioned throughput mode.

f. Amazon FSx is built on the latest AWS compute solutions to support four commonly used file systems: NetApp ONTAP, OpenZFS, Windows File Server, and Lustre. Amazon FSx latency, throughput, and IOPS vary per file system and should be considered when selecting the right file system for your workload needs.

g. AWS Snow Family are storage and compute devices that support online and offline data migration to the cloud and data storage and computing on premises. AWS Snow devices support collecting large amounts of on-premises data, processing of that data and moving that data to the cloud. There are several documented performance best practices when it comes to the number of files, file sizes, and compression.

h. AWS Storage Gateway provides on-premises applications access to cloud-based storage. AWS Storage Gateway supports multiple cloud storage services including Amazon S3, Amazon Glacier, Amazon FSx, and Amazon EBS. It supports a number of protocols such as iSCSI, SMB, and NFS. It provides low-latency performance by caching frequently accessed data on premises and only sends changed data and compressed data to AWS.

3. After you have experimented with your new storage solution and identified the optimal configuration, plan your migration and validate your performance metrics. This is a continual process, and should be reevaluated when key characteristics change or available services or options change.

Level of effort for the implementation plan: If a workload is moving from one storage solution to another, there could be a *moderate* level of effort involved in refactoring the application.

Resources

Related documents:

- Amazon EBS Volume Types
- Amazon EC2 Storage
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon FSx for NetApp ONTAP performance
- Amazon FSx for OpenZFS performance

- Amazon Glacier: Amazon Glacier Documentation
- Amazon S3: Request Rate and Performance Considerations
- Cloud Storage with AWS
- AWS Snow Family
- EBS I/O Characteristics

Related videos:

- Deep dive on Amazon EBS (STG303-R1)
- Optimize your storage performance with Amazon S3 (STG343)

Related examples:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 Examples
- Amazon FSx for Lustre Container Storage Interface (CSI) Driver

PERF03-BP02 Evaluate available configuration options

Evaluate the various characteristics and configuration options and how they relate to storage. Understand where and how to use provisioned IOPS, SSDs, magnetic storage, object storage, archival storage, or ephemeral storage to optimize storage space and performance for your workload.

Amazon EBS provides a range of options that allow you to optimize storage performance and cost for your workload. These options are divided into two major categories: SSD-backed storage for transactional workloads, such as databases and boot volumes (performance depends primarily on IOPS), and HDD-backed storage for throughput-intensive workloads, such as MapReduce and log processing (performance depends primarily on MB/s).

SSD-backed volumes include the highest performance provisioned IOPS SSD for latency-sensitive transactional workloads and general-purpose SSD that balance price and performance for a wide variety of transactional data.

Amazon S3 transfer acceleration allows fast transfer of files over long distances between your client and your S3 bucket. Transfer acceleration leverages Amazon CloudFront globally distributed edge locations to route data over an optimized network path. For a workload in an S3 bucket that has intensive GET requests, use Amazon S3 with CloudFront. When uploading large files, use multi-part uploads with multiple parts uploading at the same time to help maximize network throughput.

Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. To support a wide variety of cloud storage workloads, Amazon EFS offers two performance modes: general purpose performance mode, and max I/O performance mode. There are also two throughput modes to choose from for your file system: Bursting Throughput, and Provisioned Throughput. To determine which settings to use for your workload, see the Amazon EFS User Guide.

Amazon FSx provides four file systems to choose from: Amazon FSx for Windows File Server for enterprise workloads, Amazon FSx for Lustre for high-performance workloads, Amazon FSx for NetApp ONTAP for NetApps popular ONTAP file system, and Amazon FSx for OpenZFS for Linux-based file servers. FSx is SSD-backed and is designed to deliver fast, predictable, scalable, and consistent performance. Amazon FSx file systems deliver sustained high read and write speeds and consistent low latency data access. You can choose the throughput level you need to match your workload's needs.

Common anti-patterns:

- You only use one storage type, such as Amazon EBS, for all workloads.
- You use Provisioned IOPS for all workloads without real-world testing against all storage tiers.
- You assume that all workloads have similar storage access performance requirements.

Benefits of establishing this best practice: Evaluating all storage service options can reduce the cost of infrastructure and the effort required to maintain your workloads. It can potentially accelerate your time to market for deploying new services and features.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Determine storage characteristics: When you evaluate a storage solution, determine which storage characteristics you require, such as ability to share, file size, cache size, latency, throughput, and persistence of data. Then match your requirements to the AWS service that best fits your needs.

Resources

Related documents:

- Cloud Storage with AWS
- Amazon EBS Volume Types
- Amazon EC2 Storage
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon Glacier: Amazon Glacier Documentation
- Amazon S3: Request Rate and Performance Considerations
- Cloud Storage with AWS
- Cloud Storage with AWS
- EBS I/O Characteristics

Related videos:

- Deep dive on Amazon EBS (STG303-R1)
- Optimize your storage performance with Amazon S3 (STG343)

Related examples:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 Examples

PERF03-BP03 Make decisions based on access patterns and metrics

Choose storage systems based on your workload's access patterns and configure them by determining how the workload accesses data. Increase storage efficiency by choosing object

storage over block storage. Configure the storage options you choose to match your data access patterns.

How you access data impacts how the storage solution performs. Select the storage solution that aligns best to your access patterns, or consider changing your access patterns to align with the storage solution to maximize performance.

Creating a RAID 0 array allows you to achieve a higher level of performance for a file system than what you can provision on a single volume. Consider using RAID 0 when I/O performance is more important than fault tolerance. For example, you could use it with a heavily used database where data replication is already set up separately.

Select appropriate storage metrics for your workload across all of the storage options consumed for the workload. When using filesystems that use burst credits, create alarms to let you know when you are approaching those credit limits. You must create storage dashboards to show the overall workload storage health.

For storage systems that are a fixed size, such as Amazon EBS or Amazon FSx, ensure that you are monitoring the amount of storage used versus the overall storage size and create automation if possible to increase the storage size when reaching a threshold

Common anti-patterns:

- You assume that storage performance is adequate if customers are not complaining.
- You only use one tier of storage, assuming all workloads fit within that tier.

Benefits of establishing this best practice: You need a unified operational view, real-time granular data, and historical reference to optimize performance and resource utilization. You can create automatic dashboards and data with one-second granularity to perform metric math on your data and derive operational and utilization insights for your storage needs.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Optimize your storage usage and access patterns: Choose storage systems based on your workload's access patterns and the characteristics of the available storage options. Determine the best place to store data that will allow you to meet your requirements while reducing overhead. Use performance optimizations and access patterns when configuring and interacting with data based on the characteristics of your storage (for example, striping volumes or partitioning data).

Select appropriate metrics for storage options: Ensure that you select the appropriate storage metrics for the workload. Each storage option offers various metrics to track how your workload performs over time. Ensure that you are measuring against any storage burst metrics (for example, monitoring burst credits for Amazon EFS). For storage systems that are fixed sized, such as Amazon Elastic Block Store or Amazon FSx, ensure that you are monitoring the amount of storage used versus the overall storage size. Create automation when possible to increase the storage size when reaching a threshold.

Monitor metrics: Amazon CloudWatch can collect metrics across the resources in your architecture. You can also collect and publish custom metrics to surface business or derived metrics. Use CloudWatch or third-party solutions to set alarms that indicate when thresholds are breached.

Resources

Related documents:

- Amazon EBS Volume Types
- Amazon EC2 Storage
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon Glacier: Amazon Glacier Documentation
- Amazon S3: Request Rate and Performance Considerations
- Cloud Storage with AWS
- EBS I/O Characteristics
- Monitoring and understanding Amazon EBS performance using Amazon CloudWatch

Related videos:

- Deep dive on Amazon EBS (STG303-R1)
- Optimize your storage performance with Amazon S3 (STG343)

Related examples:

- Amazon EFS CSI Driver

- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 Examples

PERF 4. How do you select your database solution?

The most effective database solution for a system varies based on requirements for availability, consistency, partition tolerance, latency, durability, scalability, and query capability. Many systems use different database solutions for various subsystems and activate different features to improve performance. Selecting the wrong database solution and features for a system can lead to lower performance efficiency.

Best practices

- PERF04-BP01 Understand data characteristics
- PERF04-BP02 Evaluate the available options
- PERF04-BP03 Collect and record database performance metrics
- PERF04-BP04 Choose data storage based on access patterns
- PERF04-BP05 Optimize data storage based on access patterns and metrics

PERF04-BP01 Understand data characteristics

Choose your data management solutions to optimally match the characteristics, access patterns, and requirements of your workload datasets. When selecting and implementing a data management solution, you must ensure that the querying, scaling, and storage characteristics support the workload data requirements. Learn how various database options match your data models, and which configuration options are best for your use-case.

AWS provides numerous database engines including relational, key-value, document, in-memory, graph, time series, and ledger databases. Each data management solution has options and configurations available to you to support your use-cases and data models. Your workload might be able to use several different database solutions, based on the data characteristics. By selecting the best database solutions to a specific problem, you can break away from monolithic databases, with the one-size-fits-all approach that is restrictive and focus on managing data to meet your customer's need.

Desired outcome: The workload data characteristics are documented with enough detail to facilitate selection and configuration of supporting database solutions, and provide insight into potential alternatives.

Common anti-patterns:

- Not considering ways to segment large datasets into smaller collections of data that have similar characteristics, resulting in missing opportunities to use more purpose-built databases that better match data and growth characteristics.
- Not identifying the data access patterns up front, which leads to costly and complex rework later.
- Limiting growth by using data storage strategies that don't scale as quickly as is needed
- Choosing one database type and vendor for all workloads.
- Sticking to one database solution because there is internal experience and knowledge of one particular type of database solution.
- Keeping a database solution because it worked well in an on-premises environment.

Benefits of establishing this best practice: Be familiar with all of the AWS database solutions so that you can determine the correct database solution for your various workloads. After you select the appropriate database solution for your workload, you can quickly experiment on each of those database offerings to determine if they continue to meet your workload needs.

Level of risk exposed if this best practice is not established: High

- Potential cost savings may not be identified.
- Data may not be secured to the level required.
- Data access and storage performance may not be optimal.

Implementation guidance

Define the data characteristics and access patterns of your workload. Review all available database solutions to identify which solution supports your data requirements. Within a given workload, multiple databases may be selected. Evaluate each service or group of services and assess them individually. If potential alternative data management solutions are identified for part or all of the data, experiment with alternative implementations that might unlock cost, security, performance, and reliability benefits. Update existing documentation, should a new data management approach be adopted.

Type	AWS Services	Key Characteristics	Common use-cases
Relational	Amazon RDS, Amazon Aurora	Referential integrity , ACID transactions, schema on write	ERP, CRM, Commercia l off-the-shelf software
Key Value	Amazon DynamoDB	High throughput, low latency, near-infinite scalability	Shopping carts (ecommerce), product catalogs, chat applications
Document	Amazon DocumentD B	Store JSON documents and query on any attribute	Content Managemen t (CMS), customer profiles, mobile applications
In Memory	Amazon ElastiCache, Amazon MemoryDB	Microsecond latency	Caching, game leaderboards
Graph	Amazon Neptune	Highly relational data where the relations hips between data have meaning	Social networks, personalization engines, fraud detection
Time Series	Amazon Timestream	Data where the primary dimension is time	DevOps, IoT, Monitoring
Wide column	Amazon Keyspaces	Cassandra workloads.	Industrial equipment maintenance, route optimization
Ledger	Amazon QLDB	Immutable and cryptographically verifiable ledger of changes	Systems of record, healthcare, supply chains, financial institutions

Implementation steps

1. How is the data structured? (for example, unstructured, key-value, semi-structured, relational)
    a. If the data is unstructured, consider an object-store such as Amazon S3 or a NoSQL database such as Amazon DocumentDB.
    b. For key-value data, consider DynamoDB, ElastiCache for Redis or MemoryDB.
    c. If the data has a relational structure, what level of referential integrity is required?
        i. For foreign key constraints, relational databases such as Amazon RDS and Aurora can provide this level of integrity.
        ii. Typically, within a NoSQL data-model, you would de-normalize your data into a single document or collection of documents to be retrieved in a single request rather than joining across documents or tables.
2. Is ACID (atomicity, consistency, isolation, durability) compliance required?
    a. If the ACID properties associated with relational databases are required, consider a relational database such as Amazon RDS and Aurora.
3. What consistency model is required?
    a. If your application can tolerate eventual consistency, consider a NoSQL implementation. Review the other characteristics to help choose which NoSQL database is most appropriate.
    b. If strong consistency is required, you can use strongly consistent reads with DynamoDB or a relational database such as Amazon RDS.
4. What query and result formats must be supported? (for example, SQL, CSV, Parque, Avro, JSON, etc.)
5. What data types, field sizes and overall quantities are present? (for example, text, numeric, spatial, time-series calculated, binary or blob, document)
6. How will the storage requirements change over time? How does this impact scalability?
    a. Serverless databases such as DynamoDB and Amazon Quantum Ledger Database will scale dynamically up to near-unlimited storage.
    b. Relational databases have upper bounds on provisioned storage, and often must be horizontally partitioned via mechanisms such as sharding once they reach these limits.
7. What is the proportion of read queries in relation to write queries? Would caching be likely to improve performance?
    a. Read-heavy workloads can benefit from a caching layer, this could be ElastiCache or DAX if the database is DynamoDB.
    b. Reads can also be offloaded to read replicas with relational databases such as Amazon RDS.

8. Does storage and modification (OLTP - Online Transaction Processing) or retrieval and reporting (OLAP - Online Analytical Processing) have a higher priority?
    a. For high-throughput transactional processing, consider a NoSQL database such as DynamoDB or Amazon DocumentDB.
    b. For analytical queries, consider a columnar database such as Amazon Redshift or exporting the data to Amazon S3 and performing analytics using Athena or QuickSight.
9. How sensitive is this data and what level of protection and encryption does it require?
    a. All Amazon RDS and Aurora engines support data encryption at rest using AWS KMS. Microsoft SQL Server and Oracle also support native Transparent Data Encryption (TDE) when using Amazon RDS.
    b. For DynamoDB, you can use fine-grained access control with IAM to control who has access to what data at the key level.
10. What level of durability does the data require?
    a. Aurora automatically replicates your data across three Availability Zones within a Region, meaning your data is highly durable with less chance of data loss.
    b. DynamoDB is automatically replicated across multiple Availability Zones, providing high availability and data durability.
    c. Amazon S3 provides 11 9s of durability. Many database services such as Amazon RDS and DynamoDB support exporting data to Amazon S3 for long-term retention and archival.
11. Do Recovery Time Objective (RTO) or Recovery Point Objectives (RPO) requirements influence the solution?
    a. Amazon RDS, Aurora, DynamoDB, Amazon DocumentDB, and Neptune all support point in time recovery and on-demand backup and restore.
    b. For high availability requirements, DynamoDB tables can be replicated globally using the Global Tables feature and Aurora clusters can be replicated across multiple Regions using the Global database feature. Additionally, S3 buckets can be replicated across AWS Regions using cross-region replication.
12. Is there a desire to move away from commercial database engines / licensing costs?
    a. Consider open-source engines such as PostgreSQL and MySQL on Amazon RDS or Aurora
    b. Leverage AWS DMS and AWS SCT to perform migrations from commercial database engines to open-source
13. What is the operational expectation for the database? Is moving to managed services a primary concern?

a. Leveraging Amazon RDS instead of Amazon EC2, and DynamoDB or Amazon DocumentDB instead of self-hosting a NoSQL database can reduce operational overhead.

14. How is the database currently accessed? Is it only application access, or are there Business Intelligence (BI) users and other connected off-the-shelf applications?

a. If you have dependencies on external tooling then you may have to maintain compatibility with the databases they support. Amazon RDS is fully compatible with the difference engine versions that it supports including Microsoft SQL Server, Oracle, MySQL, and PostgreSQL.

15. The following is a list of potential data management services, and where these can best be used:

a. Relational databases store data with predefined schemas and relationships between them. These databases are designed to support ACID (atomicity, consistency, isolation, durability) transactions, and maintain referential integrity and strong data consistency. Many traditional applications, enterprise resource planning (ERP), customer relationship management (CRM), and ecommerce use relational databases to store their data. You can run many of these database engines on Amazon EC2, or choose from one of the AWS-managed database services: Amazon Aurora, Amazon RDS, and Amazon Redshift.

b. Key-value databases are optimized for common access patterns, typically to store and retrieve large volumes of data. These databases deliver quick response times, even in extreme volumes of concurrent requests. High-traffic web apps, ecommerce systems, and gaming applications are typical use-cases for key-value databases. In AWS, you can utilize Amazon DynamoDB, a fully managed, multi-Region, multi-master, durable database with built-in security, backup and restore, and in-memory caching for internet-scale applications.

c. In-memory databases are used for applications that require real-time access to data, lowest latency and highest throughput. By storing data directly in memory, these databases deliver microsecond latency to applications where millisecond latency is not enough. You may use in-memory databases for application caching, session management, gaming leaderboards, and geospatial applications. Amazon ElastiCache is a fully managed in-memory data store, compatible with Redis or Memcached. In case the applications also higher durability requirements, Amazon MemoryDB for Redis offers this in combination being a durable, in-memory database service for ultra-fast performance.

d. A document database is designed to store semistructured data as JSON-like documents. These databases help developers build and update applications such as content management, catalogs, and user profiles quickly. Amazon DocumentDB is a fast, scalable, highly available, and fully managed document database service that supports MongoDB workloads.

e. A wide column store is a type of NoSQL database. It uses tables, rows, and columns, but unlike a relational database, the names and format of the columns can vary from row to

row in the same table. You typically see a wide column store in high scale industrial apps for equipment maintenance, fleet management, and route optimization. Amazon Keyspaces (for Apache Cassandra) is a wide column scalable, highly available, and managed Apache Cassandra–compatible database service.

f. Graph databases are for applications that must navigate and query millions of relationships between highly connected graph datasets with millisecond latency at large scale. Many companies use graph databases for fraud detection, social networking, and recommendation engines. Amazon Neptune is a fast, reliable, fully managed graph database service that makes it easy to build and run applications that work with highly connected datasets.

g. Time-series databases efficiently collect, synthesize, and derive insights from data that changes over time. IoT applications, DevOps, and industrial telemetry can utilize time-series databases. Amazon Timestream is a fast, scalable, fully managed time series database service for IoT and operational applications that makes it easy to store and analyze trillions of events per day.

h. Ledger databases provide a centralized and trusted authority to maintain a scalable, immutable, and cryptographically verifiable record of transactions for every application. We see ledger databases used for systems of record, supply chain, registrations, and even banking transactions. Amazon Quantum Ledger Database (Amazon QLDB) is a fully managed ledger database that provides a transparent, immutable, and cryptographically verifiable transaction log owned by a central trusted authority. Amazon QLDB tracks every application data change and maintains a complete and verifiable history of changes over time.

Level of effort for the implementation plan: If a workload is moving from one database solution to another, there could be a *high* level of effort involved in refactoring the data and application.

Resources

Related documents:

- Cloud Databases with AWS
- AWS Database Caching
- Amazon DynamoDB Accelerator
- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices

- Amazon DynamoDB best practices
- Choose between EC2 and Amazon RDS
- Best Practices for Implementing Amazon ElastiCache

Related videos:

- AWS purpose-built databases (DAT209-L)
- Amazon Aurora storage demystified: How it all works (DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

Related examples:

- Optimize Data Pattern using Amazon Redshift Data Sharing
- Database Migrations
- MS SQL Server - AWS Database Migration Service (DMS) Replication Demo
- Database Modernization Hands On Workshop
- Amazon Neptune Samples

PERF04-BP02 Evaluate the available options

Understand the available database options and how it can optimize your performance before you select your data management solution. Use load testing to identify database metrics that matter for your workload. While you explore the database options, take into consideration various aspects such as the parameter groups, storage options, memory, compute, read replica, eventual consistency, connection pooling, and caching options. Experiment with these various configuration options to improve the metrics.

Desired outcome: A workload could have one or more database solutions used based on data types. The database functionality and benefits optimally match the data characteristics, access patterns, and workload requirements. To optimize your database performance and cost, you must evaluate the data access patterns to determine the appropriate database options. Evaluate the acceptable query times to ensure that the selected database options can meet the requirements.

Common anti-patterns:

- Not identifying the data access patterns.

- Not being aware of the configuration options of your chosen data management solution.
- Relying solely on increasing the instance size without looking at other available configuration options.
- Not testing the scaling characteristics of the chosen solution.

Benefits of establishing this best practice: By exploring and experimenting with the database options you may be able to reduce the cost of infrastructure, improve performance and scalability and lower the effort required to maintain your workloads.

Level of risk exposed if this best practice is not established: High

- Having to optimize for a *one size fits all* database means making unnecessary compromises.
- Higher costs as a result of not configuring the database solution to match the traffic patterns.
- Operational issues may emerge from scaling issues.
- Data may not be secured to the level required.

Implementation guidance

Understand your workload data characteristics so that you can configure your database options. Run load tests to identify your key performance metrics and bottlenecks. Use these characteristics and metrics to evaluate database options and experiment with different configurations.

AWS Services	Amazon RDS, Amazon Aurora	Amazon DynamoD	Amazon Documen B	Amazon ElastiCac he	Amazon Neptune	Amazon Timestrea m	Amazon Keyspace	Amazon QLDB
Scaling Compute	Increase instance size, Aurora Serverles s instances autoscale	Automati read/ write scaling with on-demand capacity	Increase instance size	Increase instance size, add nodes to cluster	Increase instance size	Automati ally scales to adjust capacity	Automati read/ write scaling with on-demand capacity	Automatic ally scales to adjust capacity

AWS Services	Amazon RDS, Amazon Aurora	Amazon DynamoD	Amazon Documen B	Amazon ElastiCac he	Amazon Neptune	Amazon Timestrea m	Amazon Keyspace	Amazon QLDB
	in response to changes in load	mode or automatic scaling of provision ed read/ write capacity in provision ed capacity mode					mode or automatic scaling of provision ed read/ write capacity in provision ed capacity mode	
Scaling-out reads	All engines support read replicas. Aurora supports automatic scaling of read replica instances	Increase provision ed read capacity units	Read replicas	Read replicas	Read replicas. Supports automatic scaling of read replica instances	Automati ally scales	Increase provision ed read capacity units	Automatic ally scales up to documente d concurren cy limits

AWS Services	Amazon RDS, Amazon Aurora	Amazon DynamoD	Amazon Documen B	Amazon ElastiCac he	Amazon Neptune	Amazon Timestrea m	Amazon Keyspace	Amazon QLDB
Scaling-out writes	Increasin g instance size, batching writes in the applicati on or adding a queue in front of the database. Horizonta l scaling via applicati on-level sharding across multiple instances	Increase provision ed write capacity units. Ensuring optimal partition key to prevent partition level write throttlin g	Increasin g primary instance size	Using Redis in cluster mode to distribut e writes across shards	Increasin g instance size	Write requests may be throttled while scaling. If you encounte throttlin g exception s, continue to send data at the same (or higher) throughp t to automati ally scale. Batch writes to reduce concurren t write requests	Increase provision ed write capacity units. Ensuring optimal partition key to prevent partition level write throttlin g	Automatic ally scales up to documente d concurren cy limits

AWS Services	Amazon RDS, Amazon Aurora	Amazon DynamoD	Amazon Documen B	Amazon ElastiCac he	Amazon Neptune	Amazon Timestrea m	Amazon Keyspace	Amazon QLDB
Engine configura tion	Paramete groups	Not applicabl e	Paramete groups	Paramete groups	Paramete groups	Not applicabl e	Not applicabl e	Not applicabl e
Caching	In-memory caching, configura ble via paramete groups. Pair with a dedicated cache such as ElastiCac he for Redis to offload requests for commonl accessed items	DAX (DAX) fully managed cache available	In-memory caching. Optionall y, pair with a dedicated cache such as ElastiCac he for Redis to offload requests for commonl accessed items	Primary function is caching	Use the query results cache to cache the result of a read-only query	Timestrea m has two storage tiers; one of these is a high-perf ormance in-memory tier	Deploy a separate dedicated cache such as ElastiCac he for Redis to offload requests for commonl accessed items	Not applicabl e

AWS Services	Amazon RDS, Amazon Aurora	Amazon DynamoD	Amazon Documen B	Amazon ElastiCac he	Amazon Neptune	Amazon Timestrea m	Amazon Keyspace	Amazon QLDB
High availabil ity / disaster recovery	Recomme ed configura tion for productio n workload is to run a standby instance in a second Availabil ity Zone to provide resilienc y within a Region. For resilienc y across Regions, Aurora Global Database can be used	Highly available within a Region. Tables can be replicate d across Regions using DynamoD global tables	Create multiple instances across Availabil ity Zones for availabil ity. Snapshot can be shared across Regions and clusters can be replicate d using DMS to provide Cross-Region Replicati on / disaster recovery	Recomme ed configura tion for productio n clusters is to create at least one node in a secondary Availabil ity Zone. ElastiCac he Global Datastore can be used to replicate clusters across Regions.	Read replicas in other Availabil ity Zones serve as failover targets. Snapshot can be shared across Region and clusters can be replicate d using Neptune streams to replicate data between two clusters in two different Regions.	Highly available within a Region. cross-Region replicati on requires custom applicati on developm nt using the Timestrea m SDK	Highly available within a Region. Cross-Region Replicati on requires custom applicati on logic or third-party tools	Highly available within a Region. To replicate across Regions, export the contents of the Amazon QLDB journal to a S3 bucket and configure the bucket for Cross-Region Replicati on.

Implementation steps

1. What configuration options are available for the selected databases?
    a. Parameter Groups for Amazon RDS and Aurora allow you to adjust common database engine level settings such as the memory allocated for the cache or adjusting the time zone of the database
    b. For provisioned database services such as Amazon RDS, Aurora, Neptune, Amazon DocumentDB and those deployed on Amazon EC2 you can change the instance type, provisioned storage and add read replicas.
    c. DynamoDB allows you to specify two capacity modes: on-demand and provisioned. To account for differing workloads, you can change between these modes and increase the allocated capacity in provisioned mode at any time.
2. Is the workload read or write heavy?
    a. What solutions are available for offloading reads (read replicas, caching, etc.)?
        i. For DynamoDB tables, you can offload reads using DAX for caching.
        ii. For relational databases, you can create an ElastiCache for Redis cluster and configure your application to read from the cache first, falling back to the database if the requested item is not present.
        iii. Relational databases such as Amazon RDS and Aurora, and provisioned NoSQL databases such as Neptune and Amazon DocumentDB all support adding read replicas to offload the read portions of the workload.
        iv. Serverless databases such as DynamoDB will scale automatically. Ensure that you have enough read capacity units (RCU) provisioned to handle the workload.
    b. What solutions are available for scaling writes (partition key sharding, introducing a queue, etc.)?
        i. For relational databases, you can increase the size of the instance to accommodate an increased workload or increase the provisioned IOPs to allow for an increased throughput to the underlying storage.
            - You can also introduce a queue in front of your database rather than writing directly to the database. This pattern allows you to decouple the ingestion from the database and control the flow-rate so the database does not get overwhelmed.
            - Batching your write requests rather than creating many short-lived transactions can help improve throughput in high-write volume relational databases.

ii. Serverless databases like DynamoDB can scale the write throughput automatically or by adjusting the provisioned write capacity units (WCU) depending on the capacity mode.

- You can still run into issues with *hot* partitions though, when you reach the throughput limits for a given partition key. This can be mitigated by choosing a more evenly distributed partition key or by write-sharding the partition key.

3. What are the current or expected peak transactions per second (TPS)? Test using this volume of traffic and this volume +X% to understand the scaling characteristics.
   a. Native tools such as pg_bench for PostgreSQL can be used to stress-test the database and understand the bottlenecks and scaling characteristics.
   b. Production-like traffic should be captured so that it can be replayed to simulate real-world conditions in addition to synthetic workloads.
4. If using serverless or elastically scalable compute, test the impact of scaling this on the database. If appropriate, introduce connection management or pooling to lower impact on the database.
   a. RDS Proxy can be used with Amazon RDS and Aurora to manage connections to the database.
   b. Serverless databases such as DynamoDB do not have connections associated with them, but consider the provisioned capacity and automatic scaling policies to deal with spikes in load.
5. Is the load predictable, are there spikes in load and periods of inactivity?
   a. If there are periods of inactivity consider scaling down the provisioned capacity or instance size during these times. Aurora Serverless V2 will automatically scale up and down based on load.
   b. For non-production instances, consider pausing or stopping these during non-work hours.
6. Do you need to segment and break apart your data models based on access patterns and data characteristics?
   a. Consider using AWS DMS or AWS SCT to move your data to other services.

Level of effort for the implementation plan:

To establish this best practice, you must be aware of your current data characteristics and metrics. Gathering those metrics, establishing a baseline and then using those metrics to identify the ideal database configuration options is a *low* to *moderate* level of effort. This is best validated by load tests and experimentation.

Resources

Related documents:

- Cloud Databases with AWS
- AWS Database Caching
- Amazon DynamoDB Accelerator
- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices
- Amazon DynamoDB best practices

Related videos:

- AWS purpose-built databases (DAT209-L)
- Amazon Aurora storage demystified: How it all works (DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

Related examples:

- Amazon DynamoDB Examples
- AWS Database migration samples
- Database Modernization Workshop
- Working with parameters on your Amazon RDS for Postgress DB

PERF04-BP03 Collect and record database performance metrics

To understand how your data management systems are performing, it is important to track relevant metrics. These metrics will help you to optimize your data management resources, to ensure that your workload requirements are met, and that you have a clear overview on how the workload performs. Use tools, libraries, and systems that record performance measurements related to database performance.

There are metrics that are related to the system on which the database is being hosted (for example, CPU, storage, memory, IOPS), and there are metrics for accessing the data itself (for example, transactions per second, queries rates, response times, errors). These metrics should be readily accessible for any support or operational staff, and have sufficient historical record to be able to identify trends, anomalies, and bottlenecks.

Desired outcome: To monitor the performance of your database workloads, you must record multiple performance metrics over a period of time. This allows you to detect anomalies as well as measure performance against business metrics to ensure you are meeting your workload needs.

Common anti-patterns:

- You only use manual log file searching for metrics.
- You only publish metrics to internal tools used by your team and don't have a comprehensive picture of your workload.
- You only use the default metrics recorded by your selected monitoring software.
- You only review metrics when there is an issue.
- You only monitor system level metrics, not capturing data access or usage metrics.

Benefits of establishing this best practice: Establishing a performance baseline helps in understanding normal behavior and requirements of workloads. Abnormal patterns can be identified and debugged faster improving performance and reliability of the database. Database capacity can be configured to ensure optimal cost without compromising performance.

Level of risk exposed if this best practice is not established: High

- Inability to differentiate out of normal vs. normal performance level will create difficulties in issue identification, and decision making.
- Potential cost savings may not be identified.
- Growth patterns will not be identified which might result in reliability or performance degradation.

Implementation guidance

Identify, collect, aggregate, and correlate database-related metrics. Metrics should include both the underlying system that is supporting the database and the database metrics. The underlying system metrics might include CPU utilization, memory, available disk storage, disk I/O, and

network inbound and outbound metrics while the database metrics might include transactions per second, top queries, average queries rates, response times, index usage, table locks, query timeouts, and number of connections open. This data is crucial to understand how the workload is performing and how the database solution is used. Use these metrics as part of a data-driven approach to tune and optimize your workload's resources.

Implementation steps:

1. Which database metrics are important to track?
    a. Monitoring metrics for Amazon RDS
    b. Monitoring with Performance Insights
    c. Enhanced monitoring
    d. DynamoDB metrics
    e. Monitoring DynamoDB DAX
    f. Monitoring MemoryDB
    g. Monitoring Amazon Redshift
    h. Timeseries metrics and dimensions
    i. Cluster level metrics for Aurora
    j. Monitoring Amazon Keyspaces
    k. Monitoring Amazon Neptune
2. Would the database monitoring benefit from a machine learning solution that detects operational anomalies performance issues?
    a. Amazon DevOps Guru for Amazon RDS provides visibility into performance issues and makes recommendations for corrective actions.
3. Do you need application level details about SQL usage?
    a. AWS X-Ray can be instrumented into the application to gain insights and encapsulate all the data points for single query.
4. Do you currently have an approved logging and monitoring solution?
    a. Amazon CloudWatch can collect metrics across the resources in your architecture. You can also collect and publish custom metrics to surface business or derived metrics. Use CloudWatch or third-party solutions to set alarms that indicate when thresholds are breached.
5. You identified and configured your data retention policies to match my security and operational goals?

a. Default data retention for CloudWatch metrics

b. Default data retention for CloudWatch Logs

Level of effort for the implementation plan: There is a *medium* level of effort to identify, track, collect, aggregate, and correlate metrics from all database resources.

Resources

Related documents:

- AWS Database Caching
- Amazon Athena top 10 performance tips
- Amazon Aurora best practices
- Amazon DynamoDB Accelerator
- Amazon DynamoDB best practices
- Amazon Redshift Spectrum best practices
- Amazon Redshift performance
- Cloud Databases with AWS
- Amazon RDS Performance Insights

Related videos:

- AWS purpose-built databases (DAT209-L)
- Amazon Aurora storage demystified: How it all works (DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

Related examples:

- Level 100: Monitoring with CloudWatch Dashboards
- AWS Dataset Ingestion Metrics Collection Framework
- Amazon RDS Monitoring Workshop

PERF04-BP04 Choose data storage based on access patterns

Use the access patterns of the workload and requirements of the applications to decide on optimal data services and technologies to use.

Desired outcome: Data storage has been selected based on identified and documented data access patterns. This might include the most common read, write, and delete queries, the need for as necessary calculations and aggregations, complexity of the data, data interdependency, and the required consistency needs.

Common anti-patterns:

- You only select one database engine to simplify operations management.
- You assume that data access patterns will stay consistent over time.
- You implement complex transactions, rollback, and consistency logic in the application.
- The database is configured to support a potential high traffic burst, which results in the database resources remaining idle most of the time.
- Using a shared database for transactional and analytical uses.

Benefits of establishing this best practice: Selecting and optimizing your data storage based on access patterns will help decrease development complexity and optimize your performance opportunities. Understanding when to use read replicas, global tables, data partitioning, and caching will help you decrease operational overhead and scale based on your workload needs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Identify and evaluate your data access pattern to select the correct storage configuration. Each database solution has options to configure and optimize your storage solution. Use the collected metrics and logs and experiment with options to find the optimal configuration. Use the following table to review storage options per database service.

AWS Services	Amazon RDS	Amazon Aurora	Amazon Dynamo	Amazon Docume B	Amazon ElastiCa he	Amazon Neptune	Amazon Timestre m	Amazon Keyspac	Amazon QLDB
Scaling Storage	Storage can be	Storage scales	Storage automat	Storage scales	Storage is in-	Storage scales	Organize your	Scales table	Storage automatic

AWS Services	Amazon RDS	Amazon Aurora	Amazon Dynamo	Amazon Docume B	Amazon ElastiCa he	Amazon Neptune	Amazon Timestre m	Amazon Keyspac	Amazon QLDB
	scaled up manually or configur d to scale automat ally to a maximur of 64 TiB based on engine types. Provisior ed storage cannot be decrease .	automat ally up to maximur of 128 TiB and decrease when data is removed Maximur storage size also depends upon specific Aurora MySQL or Aurora Postgres engine versions.	ally scales. Tables are unconstr ined in terms of size.	automat ally up to maximur of 64 TiB. Starting Amazon Docume B 4.0 storage can decrease by compara e amounts for data removal through droppinc a collectio n or index. With Amazon Docume B 3.6 allocatec space remains	memory , tied to instance type or count.	automat ally can grow up to 128 TiB (or 64 TiB in few Regions) Upon data removal from, total allocatec space remains same and is reused in the future.	time series data to optimize query processir g and reduce storage costs. Retentio period can be configur d through in-memory and magneti tiers.	storage up and down automat ally as your applicati on writes, updates, and deletes data.	ally scales. Tables are unconstra ined in terms of size.

AWS Services	Amazon RDS	Amazon Aurora	Amazon Dynamo	Amazon Docume B	Amazon ElastiCa he	Amazon Neptune	Amazon Timestre m	Amazon Keyspac	Amazon QLDB
				same and free space is reused when data volume increases .					

Implementation steps:

1. Understand the requirement of transactions, atomicity, consistency, isolation, and durability (ACID) compliance, and consistent reads. Not every database supports these and most of the NoSQL databases provide an eventual consistency model.
2. Consider the traffic patterns, latency, and access requirements for a globally distributed application in order to identify the optimal storage solution.
3. Analyze query patterns, random access patterns and one-time queries. Considerations around highly specialized query functionality for text and natural language processing, time series, and graphs must also be taken into account.
4. Identify and document the anticipated growth of the data and traffic.
    a. Amazon RDS and Aurora support storage automatic scaling up to documented limits. Beyond this, consider transitioning older data to Amazon S3 for archival, aggregating historical data for analytics or scaling horizontally using sharding.
    b. DynamoDB and Amazon S3 will scale to near limitless storage volume automatically.
    c. Amazon RDS instances and databases running on EC2 can be manually resized and EC2 instances can have new EBS volumes added at a later date for additional storage.

   d. Instance types can be changed based on changes in activity. For example, you can start with a smaller instance while you are testing, then scale the instance as you begin to receive production traffic to the service. Aurora Serverless V2 automatically scales in response to changes in load.
5. Baseline requirements around normal and peak performance (transactions per second TPS and queries per second QPS) and consistency (ACID and eventual consistency).
6. Document solution deployment aspects and the database access requirements (like global replication, Multi-AZ, read replication, and multiple write nodes).

Level of effort for the implementation plan: Low. If you do not have logs or metrics for your data management solution, you will need to complete that before identifying and documenting your data access patterns. Once your data access pattern is understood, selecting and configuring your data storage is a low level of effort.

Resources

Related documents:

- Cloud Databases with AWS
- Working with storage for Amazon RDS DB instances
- Amazon DocumentDB Storage
- AWS Database Caching
- Amazon Timestream Storage
- Storage in Amazon Keyspaces
- Amazon ElastiCache FAQs
- Amazon Neptune storage, reliability, and availability
- Amazon Aurora best practices
- Amazon DynamoDB Accelerator
- Amazon DynamoDB best practices
- Amazon RDS Storage Types
- Hardware specifications for Amazon RDS instance classes
- Aurora Storage limits

Related videos:

- AWS purpose-built databases (DAT209-L)
- Amazon Aurora storage demystified: How it all works (DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

Related examples:

- Experiment and test with Distributed Load Testing on AWS

PERF04-BP05 Optimize data storage based on access patterns and metrics

Use performance characteristics and access patterns that optimize how data is stored or queried to achieve the best possible performance. Measure how optimizations such as indexing, key distribution, data warehouse design, or caching strategies impact system performance or overall efficiency.

Common anti-patterns:

- You only use manual log file searching for metrics.
- You only publish metrics to internal tools.

Benefits of establishing this best practice: In order to ensure you are meeting the metrics required for the workload, you must monitor database performance metrics related to both reads and writes. You can use this data to add new optimizations for both reads and writes to the data storage layer.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Optimize data storage based on metrics and patterns: Use reported metrics to identify any underperforming areas in your workload and optimize your database components. Each database system has different performance related characteristics to evaluate, such as how data is indexed, cached, or distributed among multiple systems. Measure the impact of your optimizations.

Resources

Related documents:

- AWS Database Caching

- Amazon Athena top 10 performance tips
- Amazon Aurora best practices
- Amazon DynamoDB Accelerator
- Amazon DynamoDB best practices
- Amazon Redshift Spectrum best practices
- Amazon Redshift performance
- Cloud Databases with AWS
- Analyzing performance anomalies with DevOps Guru for RDS
- Read/Write Capacity Mode for DynamoDB

Related videos:

- AWS purpose-built databases (DAT209-L)
- Amazon Aurora storage demystified: How it all works (DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

Related examples:

- Hands-on Labs for Amazon DynamoDB

PERF 5. How do you configure your networking solution?

The most effective network solution for a workload varies based on latency, throughput requirements, jitter, and bandwidth. Physical constraints, such as user or on-premises resources, determine location options. These constraints can be offset with edge locations or resource placement.

Best practices

- PERF05-BP01 Understand how networking impacts performance
- PERF05-BP02 Evaluate available networking features
- PERF05-BP03 Choose appropriately sized dedicated connectivity or VPN for hybrid workloads
- PERF05-BP04 Leverage load-balancing and encryption offloading
- PERF05-BP05 Choose network protocols to improve performance

- PERF05-BP06 Choose your workload's location based on network requirements
- PERF05-BP07 Optimize network configuration based on metrics

PERF05-BP01 Understand how networking impacts performance

Analyze and understand how network-related decisions impact workload performance. The network is responsible for the connectivity between application components, cloud services, edge networks and on-premises data and therefor it can highly impact workload performance. In addition to workload performance, user experience is also impacted by network latency, bandwidth, protocols, location, network congestion, jitter, throughput, and routing rules.

Desired outcome: Have a documented list of networking requirements from the workload including latency, packet size, routing rules, protocols, and supporting traffic patterns. Review the available networking solutions and identify which service meets your workload networking characteristics. Cloud-based networks can be quickly rebuilt, so evolving your network architecture over time is necessary to improve performance efficiency.

Common anti-patterns:

- All traffic flows through your existing data centers.
- You overbuild Direct Connect sessions without understanding the actual usage requirements.
- You don't consider workload characteristics and encryption overhead when defining your networking solutions.
- You use on-premises concepts and strategies for networking solutions in the cloud.

Benefits of establishing this best practice: Understanding how networking impacts workload performance will help you identify potential bottlenecks, improve user experience, increase reliability, and lower operational maintenance as the workload changes.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify important network performance metrics of your workload and capture its networking characteristics. Define and document requirements as part of a data-driven approach, using benchmarking or load testing. Use this data to identify where your network solution is constrained, and examine configuration options that could improve the workload. Understand the cloud-native

networking features and options available and how they can impact your workload performance based on the requirements. Each networking feature has advantages and disadvantages and can be configured to meet your workload characteristics and scale based on your needs.

Implementation steps:

1. Define and document networking performance requirements:
    a. Include metrics such as network latency, bandwidth, protocols, locations, traffic patterns (spikes and frequency), throughput, encryption, inspection, and routing rules
2. Capture your foundational networking characteristics:
    a. VPC Flow Logs
    b. AWS Transit Gateway metrics
    c. AWS PrivateLink metrics
3. Capture your application networking characteristics:
    a. Elastic Network Adaptor
    b. AWS App Mesh metrics
    c. Amazon API Gateway metrics
4. Capture your edge networking characteristics:
    a. Amazon CloudFront metrics
    b. Amazon Route 53 metrics
    c. AWS Global Accelerator metrics
5. Capture your hybrid networking characteristics:
    a. Direct Connect metrics
    b. AWS Site-to-Site VPN metrics
    c. AWS Client VPN metrics
    d. AWS Cloud WAN metrics
6. Capture your security networking characteristics:
    a. AWS Shield, WAF, and Network Firewall metrics
7. Capture end-to-end performance metrics with tracing tools:
    a. AWS X-Ray
    b. Amazon CloudWatch RUM
8. Benchmark and test network performance:

a. Benchmark network throughput: Some factors that can affect EC2 network performance when the instances are in the same VPC. Measure the network bandwidth between EC2 Linux instances in the same VPC.

b. Perform load tests to experiment with networking solutions and options

Level of effort for the implementation plan: There is a *medium* level of effort to document workload networking requirements, options, and available solutions.

Resources

Related documents:

- Application Load Balancer
- EC2 Enhanced Networking on Linux
- EC2 Enhanced Networking on Windows
- EC2 Placement Groups
- Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances
- Network Load Balancer
- Networking Products with AWS
- Transit Gateway
- Transitioning to latency-based routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs

Related videos:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- Improve Global Network Performance for Applications
- EC2 Instances and Performance Optimization Best Practices
- Optimizing Network Performance for Amazon EC2 Instances
- Networking best practices and tips with the Well-Architected Framework

- AWS networking best practices in large-scale migrations

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP02 Evaluate available networking features

Evaluate networking features in the cloud that may increase performance. Measure the impact of these features through testing, metrics, and analysis. For example, take advantage of network-level features that are available to reduce latency, packet loss, or jitter.

Desired outcome: You have documented the inventory of components within your workload and have identified which networking configurations per component will help you meet your performance requirements. After evaluating the networking features, you have experimented and measured the performance metrics to identify how to use the features available to you.

Common anti-patterns:

- You put all your workloads into an AWS Region closest to your headquarters instead of an AWS Region close to your users.
- You fail to benchmark your workload performance and do not continually evaluate your workload performance against that benchmark.
- You do not review service configurations for performance improving options.

Benefits of establishing this best practice: Evaluating all service features and options can increase your workload performance, reduce the cost of infrastructure, decrease the effort required to maintain your workload, and increase your overall security posture. You can use the global AWS backbone to ensure that you provide the optimal networking experience for your customers.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Review which network-related configuration options are available to you, and review how they could impact your workload. Performance optimization depends on understanding how these

options interact with your architecture and the impact that they will have on both measured performance and user experience.

Many services are created to improve performance and others commonly offer features to optimize network performance. Services such as AWS Global Accelerator and Amazon CloudFront exist to improve performance while most other services have product features to optimize network traffic. Review service features, such as Amazon EC2 instance network capability, enhanced networking instance types, Amazon EBS-optimized instances, Amazon S3 transfer acceleration, and CloudFront, to improve your workload performance.

Implementation steps:

1. Create a list of workload components.
    a. Build, manage and monitor your organizations network using AWS Cloud WAN when building a unified global network.
    b. Monitor your global and core networks with Amazon CloudWatch metrics. Leverage CloudWatch Real-User Monitoring (RUM), which provides insights to help to identify, understand, and enhance users' digital experience.
    c. View aggregate network latency between AWS Regions and Availability Zones, as well as within each Availability Zone, using AWS Network Manager to gain insight into how your application performance relates to the performance of the underlying AWS network.
    d. Use an existing configuration management database (CMDB) tool or a service such as AWS Config to create an inventory of your workload and how it's configured.
2. If this is an existing workload, identify and document the benchmark for your performance metrics, focusing on the bottlenecks and areas to improve. Performance-related networking metrics will differ per workload based on business requirements and workload characteristics. As a start, these metrics might be important to review for your workload: bandwidth, latency, packet loss, jitter, and retransmits.
3. If this is a new workload, perform load tests to identify performance bottlenecks.
4. For the performance bottlenecks you identify, review the configuration options for your solutions to identify performance improvement opportunities.
5. If you don't know your network path or routes, use Network Access Analyzer to identify them.
6. Review your network protocols to further reduce your latency (see PERF05-BP05 Choose network protocols to improve performance).
7. When connection from on-premises environments to AWS is required, review available configuration options for connectivity and estimate the bandwidth and latency requirements for

your hybrid workload (see PERF05-BP03 Choose appropriately sized dedicated connectivity or VPN for hybrid workloads).

- If you are using an AWS Site-to-Site VPN across multiple locations to connect to an AWS Region, then use an accelerated Site-to-Site VPN connection for the opportunity to improve network performance.
- If your hybrid network design consists of IPSec VPN connection over AWS Direct Connect, consider using Private IP VPN to improve security and achieve segmentation. AWS Site-to-Site Private IP VPN is deployed on top of Transit virtual Interface (VIF).
- AWS Direct Connect SiteLink allows creating low-latency and redundant connections between your data centers worldwide by sending data over the fastest path between AWS Direct Connect locations, bypassing AWS Regions.

8. When your workload traffic is spread across multiple accounts, evaluate your network topology and services to reduce latency.
   - Evaluate your operational and performance tradeoffs between VPC Peering and AWS Transit Gateway when connecting multiple accounts. AWS Transit Gateway supports Equal Cost Multipath (ECMP) across multiple AWS Site-to-Site VPN connections in order to deliver additional bandwidth. Traffic between an Amazon VPC and AWS Transit Gateway remains on the private AWS network and is not exposed to the internet. AWS Transit Gateway simplifies how you interconnect all of your VPCs, which can span across thousands of AWS accounts and into on-premises networks. Share your AWS Transit Gateway between multiple accounts using Resource Access Manager.
9. Review your user locations and minimize the distance between your users and the workload.
   a. AWS Global Accelerator is a networking service that improves the performance of your users' traffic by up to 60% using the AWS global network infrastructure. When the internet is congested, AWS Global Accelerator optimizes the path to your application to keep packet loss, jitter, and latency consistently low. It also provides static IP addresses that simplify moving endpoints between Availability Zones or AWS Regions without needing to update your DNS configuration or change client-facing applications. Add an accelerator when creating a load balancer to improve the performance and availability of your workload taking advantages of AWS backbone.
   b. Amazon CloudFront can improve the performance of your workload content delivery and latency globally. CloudFront has over 410 globally dispersed points of presence that can cache your content and lower the latency to the end user. Using Lambda@edge to run functions that customize the content that CloudFront delivers closer to the users, reduces latency and Improves performance.

c. Amazon Route 53 offers latency-based routing, geolocation routing, geoproximity routing, and IP-based routing options to help you improve your workload's performance for a global audience. Identify which routing option would optimize your workload performance by reviewing your workload traffic and user location when your workload is distributed globally.

10Evaluate additional Amazon S3 features to improve storage IOPs.

a. Amazon S3 Transfer acceleration is a feature that lets external users benefit from the networking optimizations of CloudFront to upload data to Amazon S3. This improves the ability to transfer large amounts of data from remote locations that don't have dedicated connectivity to the AWS Cloud.

b. Amazon S3 Multi-Region Access Points replicates content to multiple Regions and simplifies the workload by providing one access point. When a Multi-Region Access Point is used, you can request or write data to Amazon S3 with the service identifying the lowest latency bucket.

11Review your compute resource network bandwidth.

a. Elastic Network Interfaces (ENA) used by EC2 instances, containers, and Lambda functions are limited on a per-flow basis. Review your placement groups to optimize your EC2 networking throughput. To avoid the bottleneck at the per flow-basis, design your application to use multiple flows. To monitor and get visibility into your compute related networking metrics, use CloudWatch Metrics and `ethtool`. `ethtool` is included in the ENA driver and exposes additional network-related metrics that can be published as a custom metric to CloudWatch.

b. Newer Amazon EC2 instances can leverage enhanced networking. C7gn instances featuring new AWS Nitro Cards (Nitro V5) with enhanced networking offer the highest network bandwidth and packet rate performance across Amazon EC2 network-optimized instances.

c. Amazon Elastic Network Adapters (ENA) provide further optimization by delivering better throughput for your instances within a cluster placement group. Elastic Network Adapter Express (ENA-X) is a new ENA feature using scalable reliable datagram (SRD) protocol that improves single flow bandwidth and lower tail latency by increasing maximum single flow bandwidth of Amazon EC2 instances from 5 Gbps up to 25 Gbps, and it can provide up to 85% improvement in P99.9 latency for high throughput workloads. ENA-X is currently available for C6gn.16xl.

d. Elastic Fabric Adapter (EFA) is a network interface for Amazon EC2 instances that allows you to run workloads requiring high levels of internode communications at scale on AWS. With EFA, High Performance Computing (HPC) applications using the Message Passing Interface (MPI) and Machine Learning (ML) applications using NVIDIA Collective Communications Library (NCCL) can scale to thousands of CPUs or GPUs.

e. Amazon EBS-optimized instances use an optimized configuration stack and provide additional, dedicated capacity to increase the Amazon EBS I/O. This optimization provides the best performance for your EBS volumes by minimizing contention between Amazon EBS I/O and other traffic from your instance.

Level of effort for the implementation plan:

Low to Medium. To establish this best practice, you must be aware of your current workload component options that impact network performance. Gathering the components, evaluating network improvement options, experimenting, implementing, and documenting those improvements is a *low* to *moderate* level of effort.

Resources

Related documents:

- Amazon EBS - Optimized Instances
- Application Load Balancer
- Amazon EC2 instance network bandwidth
- EC2 Enhanced Networking on Linux
- EC2 Enhanced Networking on Windows
- EC2 Placement Groups
- Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances
- Network Load Balancer
- Networking Products with AWS
- AWS Transit Gateway
- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs
- Building a cloud CMDB
- Scaling VPN throughput using AWS Transit Gateway

Related videos:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- AWS Global Accelerator

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP03 Choose appropriately sized dedicated connectivity or VPN for hybrid workloads

When a common network is required to connect on-premises and cloud resources in AWS, verify that you have adequate bandwidth to meet your performance requirements. Estimate the bandwidth and latency requirements for your hybrid workload. These numbers will drive the sizing requirements for your connectivity options.

Desired outcome: When deploying a workload that will need hybrid networking, you have multiple configuration options for connectivity, such as a dedicated connection or virtual private network (VPN). Select the appropriate connection type for each workload while verifying that you have adequate bandwidth and encryption requirements between your location and the cloud.

Common anti-patterns:

- You fail to understand or identify all workload requirements (bandwidth, latency, jitter, encryption and traffic needs).
- You don't evaluate backup or parallel connectivity options.

Benefits of establishing this best practice: Selecting and configuring appropriately sized hybrid network solutions will increase the reliability of your workload and maximize performance opportunities. By identifying workload requirements, planning ahead, and evaluating hybrid solutions you will minimize expensive physical network changes and operational overhead while increasing your time to market.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Develop a hybrid networking architecture based on your bandwidth requirements. Estimate the bandwidth and latency requirements of your hybrid applications. Consider appropriate connectivity option between using a dedicated network connection or internet-based VPN.

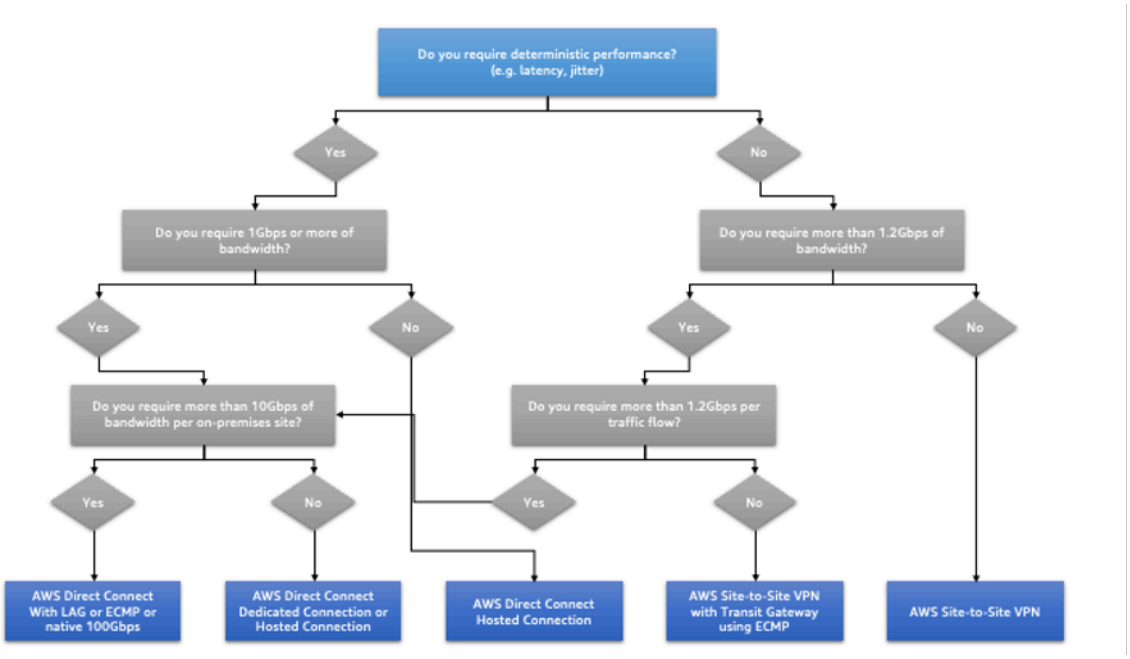
Dedicated connection establishes network connection over private lines. It is suitable when you need high-bandwidth, low-latency while achieving consistent performance. VPN connection establishes secure connection over the internet. It is suitable when you need encrypted connection using an existing internet connection.

Based on your bandwidth requirements, a single VPN or dedicated connection might not be enough, and you must architect a hybrid setup to permit traffic load balancing across multiple connections.

Implementation steps

1. Estimate the bandwidth and latency requirements of your hybrid applications.
    a. For existing apps that are moving to AWS, leverage the data from your internal network monitoring systems.
    b. For new apps or existing apps for which you don’t have monitoring data, consult with the product owners to derive adequate performance metrics and provide a good user experience.
2. Select dedicated connection or VPN as your connectivity option. Based on all workload requirements (encryption, bandwidth and traffic needs), you can either choose AWS Direct Connect or AWS Site-to-Site VPN (or both). The following diagram will help you choose the appropriate connection type.
    a. If you consider dedicated connection, AWS Direct Connect may be required, which offers more predictable and consistent performance due to its private network connectivity. AWS Direct Connect provides dedicated connectivity to the AWS environment, from 50 Mbps up to 100 Gbps, using either dedicated connection or hosted connection. This gives you managed and controlled latency and provisioned bandwidth so your workload can connect efficiently to other environments. Using an AWS Direct Connect partners, you can have end-to-end connectivity from multiple environments, providing an extended network with consistent performance. AWS offers scaling direct connect connection bandwidth using either native 100 Gbps, Link Aggregation Group (LAG), or BGP Equal-cost multipath (ECMP).
    b. If you consider VPN connection, an AWS managed VPN is the recommended option. The AWS Site-to-Site VPN provides a managed VPN service supporting Internet Protocol security

(IPsec) protocol. When a VPN connection is created, each VPN connection includes two tunnels for high availability. With AWS Transit Gateway, you can simplify the connectivity between multiple VPCs and also connect to any VPC attached to AWS Transit Gateway with a single VPN connection. AWS Transit Gateway also allows you to scale beyond the 1.25Gbps IPsec VPN throughput limit by allowing equal cost multi-path (ECMP) routing support over multiple VPN tunnels.



Deterministic performance flowchart.

Level of effort for the implementation plan: High. There is significant effort in evaluating workload needs for hybrid networks and implementing hybrid networking solutions.

Resources

Related documents:

- Network Load Balancer
- Networking Products with AWS
- AWS Transit Gateway
- Transitioning to latency-based Routing in Amazon Route 53
- VPC Endpoints

- VPC Flow Logs
- AWS Site-to-Site VPN
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure
- AWS Direct Connect
- Client VPN

Related videos:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- AWS Global Accelerator
- AWS Direct Connect
- Transit Gateway Connect
- VPN Solutions
- Security with VPN Solutions

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP04 Leverage load-balancing and encryption offloading

Use load balancers to achieve optimal performance efficiency of your target resources and improve the responsiveness of your system.

Desired outcome: Reduce the number of computing resources to serve your traffic. Avoid resource consumption imbalance in your targets. Offload compute-intensive tasks to the Load Balancer. Leverage cloud elasticity and flexibility to improve performance and optimize your architecture.

Common anti-patterns:

- You don't consider your workload requirements when choosing the load balancer type.
- You don't leverage the load balancer features for performance optimization.

- The workload is exposed directly to the Internet without a load balancer.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Load balancers act as the entry point for your workload and from there they distribute the traffic to your back-end targets, such as compute instances or containers. Choosing the right load balancer type is the first step to optimize your architecture.

Start by listing your workload characteristics, such as protocol (like TCP, HTTP, TLS, or WebSockets), target type (like instances, containers, or serverless), application requirements (like long running connections, user authentication, or stickiness), and placement (like Region, Local Zone, Outpost, or zonal isolation).

After choosing the right load balancer, you can start leveraging its features to reduce the amount of effort your back-end has to do to serve the traffic.

For example, using both Application Load Balancer (ALB) and Network Load Balancer (NLB), you can perform SSL/TLS encryption offloading, which is an opportunity to avoid the CPU-intensive TLS handshake from being completed by your targets and also to improve certificate management.

When you configure SSL/TLS offloading in your load balancer, it becomes responsible for the encryption of the traffic from and to clients while delivering the traffic unencrypted to your back-ends, freeing up your back-end resources and improving the response time for the clients.

Application Load Balancer can also serve HTTP2 traffic without needing to support it on your targets. This simple decision can improve your application response time, as HTTP2 uses TCP connections more efficiently.

Load balancers can also be used to make your architecture more flexible by distributing traffic across different back-end types such as containers and serverless. For example, Application Load Balancer can be configured with listener rules that forward traffic to different target groups based on the request parameters such as header, method or pattern.

Your workload latency requirements should also be considered when defining the architecture. As an example, if you have a latency-sensitive application, you may decide to use Network Load Balancer, which offers extremely low latencies. Alternatively, you may decide to bring your workload closer to your customers by leveraging Application Load Balancer in AWS Local Zones or even AWS Outposts.

Another consideration for latency-sensitive workloads is cross-zone load balancing. With cross-zone load balancing, each load balancer node distributes traffic across the registered targets in all allowed Availability Zones. This improves availability, although it can add a single digit millisecond to the roundtrip latency.

Lastly, both ALB and NLB offer monitoring resources such as logs and metrics. Properly setting up monitoring can help with gathering performance insights of your application. For example, you can use ALB access logs to find which requests are taking longer to be answered or which back-end targets are causing performance issues.

Implementation steps

1. Choose the right load balancer for your workload.
    a. Use Application Load Balancer for HTTP/HTTPS workloads.
    b. Use Network Load Balancer for non-HTTP workloads that run on TCP or UDP.
    c. Use a combination of both (ALB as a target of NLB) if you want to leverage features of both products. For example, you can do this if you want to use the static IPs of NLB together with HTTP header based routing from ALB, or if you want to expose your HTTP workload to an AWS PrivateLink.
    d. For a full comparison of load balancers, see ELB product comparison.
2. Use SSL/TLS offloading.
    a. Configure HTTPS/TLS listeners with both Application Load Balancer and Network Load Balancer integrated with AWS Certificate Manager.
    b. Note that some workloads may require end-to-end encryption for compliance reasons. In this case, it is a requirement to allow encryption at the targets.
    c. For security best practices, see SEC09-BP02 Enforce encryption in transit.
3. Select the right routing algorithm.
    a. The routing algorithm can make a difference in how well-used your back-end targets are and therefore how they impact performance. For example, ALB provides two options for routing algorithms:
    b. **Least outstanding requests:** Use to achieve a better load distribution to your back-end targets for cases when the requests for your application vary in complexity or your targets vary in processing capability.
    c. **Round robin:** Use when the requests and targets are similar, or if you need to distribute requests equally among targets.

4. Consider cross-zone or zonal isolation.
    a. Use cross-zone turned off (zonal isolation) for latency improvements and zonal failure domains. It is turned off by default in NLB and in ALB you can turn it off per target group.
    b. Use cross-zone turned on for increased availability and flexibility. By default, cross-zone is turned on for ALB and in NLB you can turn it on per target group.
5. Turn on HTTP keep-alives for your HTTP workloads.
    a. For HTTP workloads, turn on HTTP keep-alive in the web server settings for your back-end targets. With this feature, the load balancer can reuse backend connections until the keep-alive timeout expires, improving your HTTP request and response time and also reducing resource utilization on your back-end targets. For detail on how to do this for Apache and Nginx, see What are the optimal settings for using Apache or NGINX as a backend server for ELB?
6. Use Elastic Load Balancing integrations for better orchestration of compute resources.
    a. Use Auto Scaling integrated with your load balancer. One of the key aspects of a performance efficient system has to do with right-sizing your back-end resources. To do this, you can leverage load balancer integrations for back-end target resources. Using the load balancer integration with Auto Scaling groups, targets will be added or removed from the load balancer as required in response to incoming traffic.
    b. Load balancers can also integrate with Amazon ECS and Amazon EKS for containerised workloads.
        - Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group
        - Amazon ECS - Service load balancing
        - Application load balancing on Amazon EKS
        - Network load balancing on Amazon EKS
7. Monitor your load balancer to find performance bottlenecks.
    a. Turn on access logs for your Application Load Balancer and Network Load Balancer.
    b. The main fields to consider for ALB are `request_processing_time`, `request_processing_time`, and `response_processing_time`.
    c. The main fields to consider for NLB are `connection_time` and `tls_handshake_time`.
    d. Be ready to query the logs when you need them. You can use Amazon Athena to query both ALB logs and NLB Logs.
    e. Create alarms for performance related metrics such as `TargetResponseTime` for ALB.

Resources

Related best practices:

- SEC09-BP02 Enforce encryption in transit

Related documents:

- ELB product comparison
- AWS Global Infrastructure
- Improving Performance and Reducing Cost Using Availability Zone Affinity
- Step by step for Log Analysis with Amazon Athena
- Querying Application Load Balancer logs
- Monitor your Application Load Balancers
- Monitor your Network Load Balancers

Related videos:

- AWS re:Invent 2018: [REPEAT 1] Elastic Load Balancing: Deep Dive and Best Practices (NET404-R1)
- AWS re:Invent 2021 - How to choose the right load balancer for your AWS workloads
- AWS re:Inforce 2022 - How to use Elastic Load Balancing to enhance your security posture at scale (NIS203)
- AWS re:Invent 2019: Get the most from Elastic Load Balancing for different workloads (NET407-R2)

Related examples:

- CDK and CloudFormation samples for Log Analysis with Amazon Athena

PERF05-BP05 Choose network protocols to improve performance

Assess the performance requirements for your workload, and choose the network protocols that optimize your workload's overall performance.

There is a relationship between latency and bandwidth to achieve throughput. For instance, if your file transfer is using Transmission Control Protocol (TCP), higher latencies will reduce overall throughput. There are approaches to fix this with TCP tuning and optimized transfer protocols (some approaches use User Datagram Protocol (UDP)).

The scalable reliable datagram (SRD) protocol is a network transport protocol built by AWS for Elastic Fabric Adapters that provides reliable datagram delivery. Unlike the TCP protocol, SRD can reorder packets and deliver them out of order. This out of order delivery mechanism of SRD sends packets in parallel over alternate paths, increasing throughput.

Common anti-patterns:

- Using TCP for all workloads regardless of performance requirements.

Benefits of establishing this best practice:

- Selecting the proper protocol for communication between workload components ensures that you are getting the best performance for that workload.
- Verifying that an appropriate protocol is used for communication between users and workload components helps improve overall user experience for your applications. For instance, by using both TCP and UDP together, VDI workloads can take advantage of the reliability of TCP for critical data and the speed of UDP for real-time data.

Level of risk exposed if this best practice is not established: Medium (Using an inappropriate network protocol can lead to poor performance, such as slow response times, high latency and poor scalability)

Implementation guidance

A primary consideration for improving your workload's performance is to understand the latency and throughput requirements, and then choose network protocols that optimize performance.

When to consider using TCP

TCP provides reliable data delivery, and can be used for communication between workload components where reliability and guaranteed delivery of data is important. Many web-based applications rely on TCP-based protocols, such as HTTP and HTTPS, to open TCP sockets for communication with servers on AWS. Email and file data transfer are common applications that

also make use of TCP due to TCP's ability to control the rate of data exchange and network congestion. Using TLS with TCP can add some overhead to the communication, which can result in increased latency and reduced throughput. The overhead comes mainly from the added overhead of the handshake process, which can take several round-trips to complete. Once the handshake is complete, the overhead of encrypting and decrypting data is relatively small.

When to consider using UDP

UDP is a connectionless-oriented protocol and is therefore suitable for applications that need fast, efficient transmission, such as log, monitoring, and VoIP data. Also, consider using UDP if you have workload components that respond to small queries from large numbers of clients to ensure optimal performance of the workload. Datagram Transport Layer Security (DTLS) is the UDP equivalent of TLS. When using DTLS with UDP, the overhead comes from encrypting and decrypting the data, as the handshake process is simplified. DTLS also adds a small amount of overhead to the UDP packets, as it includes additional fields to indicate the security parameters and to detect tampering.

When to consider using SRD

Scalable reliable datagram (SRD) is a network transport protocol optimized for high-throughput workloads due to its ability to load-balancer traffic across multiple paths and quickly recover from packet drops or link failures. SRD is therefore best used for high performance computing (HPC) workloads that require high throughput and low latency communication between compute nodes. This might include parallel processing tasks such as simulation, modelling, and data analysis that involve a large amount of data transfer between nodes.

Implementation steps

1. Use the AWS Global Accelerator and AWS Transfer Family services to improve the throughput of your online file transfer applications. The AWS Global Accelerator service helps you achieve lower latency between your client devices and your workload on AWS. With AWS Transfer Family, you can use TCP-based protocols such as Secure Shell File Transfer Protocol (SFTP) and File Transfer Protocol over SSL (FTPS) to securely scale and manage your file transfers to AWS storage services.
2. Use network latency to determine if TCP is appropriate for communication between workload components. If the network latency between your client application and server is high, then the TCP three-way handshake can take some time, thereby impacting on the responsiveness of your application. Metrics such as Time to First Byte (TTFB) and Round-Trip Time (RTT) can be

used to measure network latency. If your workload serves dynamic content to users, consider using Amazon CloudFront, which establishes a persistent connection to each origin for dynamic content to eliminate the connection setup time that would otherwise slow down each client request.

3. Using TLS with TCP or UDP can result in increased latency and reduced throughput for your workload due to the impact of encryption and decryption. For such workloads, consider SSL/TLS offloading on Elastic Load Balancing to improve workload performance by allowing the load balancer to handle SSL/TLS encryption and decryption process instead of having backend instances do it. This can help reduce the CPU utilization on the backend instances, which can improve performance and increase capacity.
4. Use the Network Load Balancer (NLB) to deploy services that rely on the UDP protocol, such as authentication and authorization, logging, DNS, IoT, and streaming media, to improve the performance and reliability of your workload. The NLB distributes incoming UDP traffic across multiple targets, allowing you to scale your workload horizontally, increase capacity, and reduce the overhead of a single target.
5. For your High Performance Computing (HPC) workloads, consider using the Elastic Network Adapter (ENA) Express functionality that uses the SRD protocol to improve network performance by providing a higher single flow bandwidth (25Gbps) and lower tail latency (99.9 percentile) for network traffic between EC2 instances.
6. Use the Application Load Balancer (ALB) to route and load balance your gRPC (Remote Procedure Calls) traffic between workload components or between gRPC clients and services. gRPC uses the TCP-based HTTP/2 protocol for transport and it provides performance benefits such as lighter network footprint, compression, efficient binary serialization, support for numerous languages, and bi-directional streaming.

Resources

Related documents:

- Amazon EBS - Optimized Instances
- Application Load Balancer
- EC2 Enhanced Networking on Linux
- EC2 Enhanced Networking on Windows
- EC2 Placement Groups
- Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances

- Network Load Balancer
- Networking Products with AWS
- Transit Gateway
- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs

Related videos:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- Tuning Your Cloud: Improve Global Network Performance for Application
- Application Scaling with EFA and SRD

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP06 Choose your workload's location based on network requirements

Evaluate options for resource placement to reduce network latency and improve throughput, providing an optimal user experience by reducing page load and data transfer times.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Resources, such as Amazon EC2 instances, are placed into availability zones within AWS Regions, AWS Local Zones, AWS Outposts, or AWS Wavelength zones. Selection of this location influences network latency and throughput from a given user location. Edge services such as Amazon CloudFront and AWS Global Accelerator can also be used to improve network performance by either caching content at edge locations or providing users with an optimal path to the workload through the AWS global network.

Implementation steps

1. Choose the appropriate AWS Region or Regions for your deployment based on the following key elements:
   a. **Where your users are located:** choosing a Region close to your workload's users to ensure low latency when they use the workload.
   b. **Where your data is located:** for data-heavy applications, the major bottleneck in data transfer is latency. Application code should run as close to the data as possible.
   c. **Other constraints:** consider constraints such as security and compliance (for example, data residency requirements).
2. For a given workload, if a component consists of a group of interdependent Amazon EC2 instances requiring low-latency, consider using cluster placement groups to influence placement of those instances to meet the requirements of the workload. Instances in the same cluster placement group enjoy a higher per-flow throughput limit for TCP/IP traffic and are placed in the same high-bisection bandwidth segment of the network. Cluster placement groups are recommended for applications that benefit from low network latency, high network throughput, or both.
3. For a workload that is location-sensitive, for example with low-latency or data residency requirements, review AWS Local Zones or AWS Outposts.
   a. AWS Local Zones are a type of infrastructure deployment that places compute, storage, database, and other select AWS services close to large population and industry centers.
   b. AWS Outposts is a family of fully managed solutions delivering AWS infrastructure and services to virtually any on-premises or edge location for a truly consistent hybrid experience.
4. Applications such as high-resolution live video streaming, high-fidelity audio, and augmented reality/virtual reality (AR/VR) require ultra-low-latency for 5G devices. For such applications, consider AWS Wavelength. AWS Wavelength embeds AWS compute and storage services within 5G networks, providing mobile edge computing infrastructure for developing, deploying, and scaling ultra-low-latency applications.
5. If you have geographically distributed users, a content distribution network (CDN) may be used to accelerate distribution of static and dynamic web content by delivering data through globally dispersed points of presence (PoPs). CDNs typically also provide edge computing capabilities, performing latency sensitive operations such as HTTP header manipulations and URL rewrites and redirects at large scale at the edge. Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content. Use cases for CloudFront include accelerating static website content delivery and serving video on demand or live streaming video. CloudFront can also be used to customize the content and experience for viewers, at reduced latency.

6. Some applications require fixed entry points or higher performance by reducing first byte latency and jitter, and increasing throughput. These applications can benefit from networking services that provide static anycast IP addresses and TCP termination at edge locations. AWS Global Accelerator can improve performance for your applications by up to 60% and provide quick failover for multi-region architectures. AWS Global Accelerator provides you with static anycast IP addresses that serve as a fixed entry point for your applications hosted in one or more AWS Regions. These IP addresses permit traffic to ingress onto the AWS global network as close to your users as possible. AWS Global Accelerator reduces the initial connection setup time by establishing a TCP connection between the client and the AWS edge location closest to the client. Review the use of AWS Global Accelerator to improve the performance of your TCP/UDP workloads and provide quick failover for multi-region architectures.
7. If you have applications or users on-premises, you may benefit from having a dedication network connection between your network and the cloud. A dedicated network connection can reduce the chance of encountering congestion or unexpected increases in latency. AWS Direct Connect can improve application performance by connecting your network directly to AWS and bypassing the public internet. When creating a new connection, you can choose a hosted connection provided by an AWS Direct Connect Delivery Partner, or choose a dedicated connection from AWS and deploy at over 100 AWS Direct Connect locations around the globe. You can also reduce your networking costs with low data transfer rates out of AWS, and optionally configure a Site-to-Site VPN for failover.
8. If you configure a Site-to-Site VPN to connect to your resources within AWS, you can optionally turn on acceleration. An accelerated Site-to-Site VPN connection uses AWS Global Accelerator to route traffic from your on-premises network to an AWS edge location that is closest to your customer gateway device.
9. Identify which DNS routing option would optimize your workload performance by reviewing your workload traffic and user location. Amazon Route 53 offers latency-based routing, geolocation routing, geoproximity routing, and IP-based routing options to help you improve your workload's performance for a global audience.
    a. Route 53 also offers low query latency for your end users. Using a global anycast network of DNS servers around the world, Route 53 is designed to automatically answer queries from the optimal location depending on network conditions.

Resources

Related best practices:

- COST07-BP02 Implement Regions based on cost
- COST08-BP03 Implement services to reduce data transfer costs
- REL10-BP01 Deploy the workload to multiple locations
- REL10-BP02 Select the appropriate locations for your multi-location deployment
- SUS01-BP01 Choose Regions near Amazon renewable energy projects and Regions where the grid has a published carbon intensity that is lower than other locations (or Regions)
- SUS02-BP04 Optimize geographic placement of workloads for user locations
- SUS04-BP07 Minimize data movement across networks

Related documents:

- AWS Global Infrastructure
- AWS Local Zones and AWS Outposts, choosing the right technology for your edge workload
- Placement groups
- AWS Local Zones
- AWS Outposts
- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- Site-to-Site VPN
- Amazon Route 53

Related videos:

- AWS Local Zones Explainer Video
- AWS Outposts: Overview and How It Works
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises
- AWS re:Invent 2020: AWS Wavelength: Run apps with ultra-low latency at 5G edge
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator

- AWS re:Invent 2022 - Build your global wide area network using AWS
- AWS re:Invent 2020: Global traffic management with Amazon Route 53

Related examples:

- AWS Global Accelerator Workshop
- Handling Rewrites and Redirects using Edge Functions

PERF05-BP07 Optimize network configuration based on metrics

Improper network configuration often affects network performance, efficiency, and cost. In common network environments, in order to quickly complete the deployment in the early stage, the proper network configuration is not fully considered in terms of network performance. To optimize your network configuration, you must first have visibility and data about your network environment.

To understand how your network resources are performing, collect and analyze data to make informed decisions about optimizing your network configuration. Measure the impact of those changes and use the impact measurements to make future decisions.

Desired outcome: Use metrics and network monitoring tools to optimize network configuration as workloads evolve. Cloud-based networks can be optimized quickly, so evolving your network architecture over time is necessary to maintain performance efficiency.

Common anti-patterns:

- You assume that all performance-related issues are application-related.
- You only test your network performance from a location close to where you have deployed the workload.
- You use default configurations for all network services.
- You overprovision the network resource to provide sufficient capacity.

Benefits of establishing this best practice: Collecting necessary metrics of your AWS network and implementing network monitoring tools allows you to understand network performance and optimize network configurations.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Monitoring traffic to and from VPCs, subnets, or network interfaces is crucial to understanding how to utilize AWS network resources and how you can optimize network configurations. By using the following tools, you can further inspect information about the traffic usage, network access and logs.

Implementation steps

1. Use Amazon VPC IP Address Manager. You can use IPAM to plan, track, and monitor IP addresses for your AWS and on-premises workloads. This is the best practice for you for to optimize IP address usage and allocation.
2. Turn on VPC Flow logs. Use VPC Flow Logs to capture detailed information about traffic to and from network interfaces in your VPCs. With VPC Flow Logs, you can diagnose overly restrictive or permissive security group rules and determine the direction of the traffic to and from the network interfaces. Data ingestion and archival charges for vended logs apply when you publish flow logs.
3. Turn on DNS query logging. You can configure Amazon Route 53 to log information about public or private DNS queries Route 53 receives. With DNS logs, you can optimize DNS configurations by understanding the domain or subdomain that was requested or Route 53 EDGE locations that responded to DNS queries.
4. Use Reachability Analyzer to analyze and debug network reachability. Reachability Analyzer is a configuration analysis tool that allows you to perform connectivity testing between a source resource and a destination resource in your VPCs. This tool helps you verify that your network configuration matches your intended connectivity.
5. Use Network Access Analyzer to understand network access to your resources. You can use Network Access Analyzer to specify your network access requirements and identify potential network paths that do not meet your specified requirements. By optimizing your corresponding network configuration, you can understand and verify the state of your network and demonstrate if your network on AWS meets your compliance requirements.
6. Use Amazon CloudWatch and turn on the appropriate metrics for network options. Make sure to choose the right network metric for your workload. For example, you can turn on metrics for VPC Network Address Usage, VPC NAT Gateway, AWS Transit Gateway, VPN tunnel, AWS Network Firewall, Elastic Load Balancing, and AWS Direct Connect. Continually monitoring metrics is a good practice to observe and understand your network status and usage, and helps you optimize network configuration based on your observations.

Level of effort for the implementation plan: Medium

Resources

Related documents:

- VPC Flow Logs
- Public DNS query logging
- What is IPAM?
- What is Reachability Analyzer?
- What is Network Access Analyzer?
- CloudWatch metrics for your VPCs
- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format
- Monitoring your global and core networks with Amazon Cloudwatch metrics
- Continuously monitor network traffic and resources

Related videos:

- Networking best practices and tips with the Well-Architected Framework
- Monitoring and troubleshooting network traffic

Related examples:

- AWS Networking Workshops
- AWS Network Monitoring

Review

Question

- PERF 6. How do you evolve your workload to take advantage of new releases?

PERF 6. How do you evolve your workload to take advantage of new releases?

When architecting workloads, there are finite options that you can choose from. However, over time, new technologies and approaches become available that could improve the performance of your workload.

Best practices

- PERF06-BP01 Stay up-to-date on new resources and services
- PERF06-BP02 Define a process to improve workload performance
- PERF06-BP03 Evolve workload performance over time

PERF06-BP01 Stay up-to-date on new resources and services

Evaluate ways to improve performance as new services, design patterns, and product offerings become available. Determine which of these could improve performance or increase the efficiency of the workload through evaluation, internal discussion, or external analysis.

Define a process to evaluate updates, new features, and services relevant to your workload. For example, building a proof of concept that uses new technologies or consulting with an internal group. When trying new ideas or services, run performance tests to measure the impact that they have on the performance of the workload. Using infrastructure as code (IaC) and a DevOps culture to take advantage of the ability to test new ideas or technologies frequently with minimal cost or risk.

Desired outcome: You have documented the inventory of components, your design pattern, and your workload characteristics. You use that documentation to create a list of subscriptions to notify your team on service updates, features, and new products. You have identified component stakeholders that will evaluate the new releases and provide a recommendation for business impact and priority.

Common anti-patterns:

- You only review new options and services when your workload is not meeting performance requirements.
- You assume all new product offerings will not be useful to your workload.
- You always choose to build as opposed to buy when improving your workload.

Benefits of establishing this best practice: By considering new services or product offerings, you can improve the performance and efficiency of your workload, lower the cost of the infrastructure, and reduce the effort required to maintain your services.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Define a process to evaluate updates, new features, and services from AWS. For example, building proof-of-concepts that use new technologies. When trying new ideas or services, run performance tests to measure the impact on the efficiency or performance of the workload. Take advantage of the flexibilfity that you have in AWS to test new ideas or technologies frequently with minimal cost or risk.

Implementation steps

1. Document your workload solutions. Use your configuration management database (CMDB) solution to document your inventory and categorize your services and dependencies. Use tools like AWS Config to get a list of all services in AWS being used by your workload.
2. Use a tagging strategy to document owners for each workload component and category. For example, if you are currently using Amazon RDS as your database solution, have your database administrator (DBA) assigned and documented as the owner for evaluating and researching new services and updates.
3. Identify news and update sources related to your workload components. In the Amazon RDS example previously mentioned, the category owner should subscribe to the What's New at AWS blog for the products that match their workload component. You can subscribe to the RSS feed or manage your email subscriptions. Monitor upgrades to the Amazon RDS database you use, features introduced, instances released and new products like Amazon Aurora Serverless. Monitor industry blogs, products, and vendors that the component relies on.
4. Document your process for evaluating updates and new services. Provide your category owners the time and space needed to research, test, experiment, and validate updates and new services. Refer back to the documented business requirements and KPIs to help prioritize which update will make a positive business impact.

Level of effort for the implementation plan: To establish this best practice, you must be aware of your current workload components, identify category owners and identify sources for service updates. This is a low level of effort to start but is an ongoing process that could evolve and improve over time.

Resources

Related documents:

- AWS Blog
- What's New with AWS

Related videos:

- AWS Events YouTube Channel
- AWS Online Tech Talks YouTube Channel
- Amazon Web Services YouTube Channel

Related examples:

- AWS Github
- AWS Skill Builder

PERF06-BP02 Define a process to improve workload performance

Define a process to evaluate new services, design patterns, resource types, and configurations as they become available. For example, run existing performance tests on new instance offerings to determine their potential to improve your workload.

Your workload's performance has a few key constraints. Document these so that you know what kinds of innovation might improve the performance of your workload. Use this information when learning about new services or technology as it becomes available to identify ways to alleviate constraints or bottlenecks.

Common anti-patterns:

- You assume your current architecture will become static and never update over time.
- You introduce architecture changes over time with no metric justification.

Benefits of establishing this best practice: By defining your process for making architectural changes, you allow gathered data to influence your workload design over time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Identify the key performance constraints for your workload: Document your workload's performance constraints so that you know what kinds of innovation might improve the performance of your workload.

Resources

Related documents:

- AWS Blog
- What's New with AWS

Related videos:

- AWS Events YouTube Channel
- AWS Online Tech Talks YouTube Channel
- Amazon Web Services YouTube Channel

Related examples:

- AWS Github
- AWS Skill Builder

PERF06-BP03 Evolve workload performance over time

As an organization, use the information gathered through the evaluation process to actively drive adoption of new services or resources when they become available.

Use the information you gather when evaluating new services or technologies to drive change. As your business or workload changes, performance needs also change. Use data gathered from your workload metrics to evaluate areas where you can get the biggest gains in efficiency or performance, and proactively adopt new services and technologies to keep up with demand.

Common anti-patterns:

- You assume that your current architecture will become static and never update over time.
- You introduce architecture changes over time with no metric justification.

- You change architecture just because everyone else in the industry is using it.

Benefits of establishing this best practice: To optimize your workload performance and cost, you must evaluate all software and services available to determine the appropriate ones for your workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Evolve your workload over time: Use the information you gather when evaluating new services or technologies to drive change. As your business or workload changes, performance needs also change. Use data gathered from your workload metrics to evaluate areas where you can achieve the biggest gains in efficiency or performance, and proactively adopt new services and technologies to keep up with demand.

Resources

Related documents:

- AWS Blog
- What's New with AWS

Related videos:

- AWS Events YouTube Channel
- AWS Online Tech Talks YouTube Channel
- Amazon Web Services YouTube Channel

Related examples:

- AWS Github
- AWS Skill Builder

Monitoring

Question

- PERF 7. How do you monitor your resources to verify they are performing?

PERF 7. How do you monitor your resources to verify they are performing?

System performance can degrade over time. Monitor system performance to identify degradation and remediate internal or external factors, such as the operating system or application load.

Best practices

- PERF07-BP01 Record performance-related metrics
- PERF07-BP02 Analyze metrics when events or incidents occur
- PERF07-BP03 Establish key performance indicators (KPIs) to measure workload performance
- PERF07-BP04 Use monitoring to generate alarm-based notifications
- PERF07-BP05 Review metrics at regular intervals
- PERF07-BP06 Monitor and alarm proactively

PERF07-BP01 Record performance-related metrics

Use a monitoring and observability service to record performance-related metrics. Examples of metrics include record database transactions, slow queries, I/O latency, HTTP request throughput, service latency, or other key data.

Identify the performance metrics that matter for your workload and record them. This data is an important part of being able to identify which components are impacting overall performance or efficiency of the workload.

Working back from the customer experience, identify metrics that matter. For each metric, identify the target, measurement approach, and priority. Use these to build alarms and notifications to proactively address performance-related issues.

Common anti-patterns:

- You only monitor operating system level metrics to gain insight into your workload.
- You architect your compute needs for peak workload requirements.

Benefits of establishing this best practice: To optimize performance and resource utilization, you need a unified operational view of your key performance indicators. You can create dashboards and perform metric math on your data to derive operational and utilization insights.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify the relevant performance metrics for your workload and record them. This data helps identify which components are impacting overall performance or efficiency of your workload.

Identify performance metrics: Use the customer experience to identify the most important metrics. For each metric, identify the target, measurement approach, and priority. Use these data points to build alarms and notifications to proactively address performance-related issues.

Resources

Related documents:

- CloudWatch Documentation
- Collect metrics and logs from Amazon EC2 Instances and on-premises servers with the CloudWatch Agent
- Publish custom metrics
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation
- Amazon CloudWatch RUM

Related videos:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Application Performance Management on AWS
- Build a Monitoring Plan

Related examples:

- Level 100: Monitoring with CloudWatch Dashboards
- Level 100: Monitoring Windows EC2 instance with CloudWatch Dashboards
- Level 100: Monitoring an Amazon Linux EC2 instance with CloudWatch Dashboards

PERF07-BP02 Analyze metrics when events or incidents occur

In response to (or during) an event or incident, use monitoring dashboards or reports to understand and diagnose the impact. These views provide insight into which portions of the workload are not performing as expected.

When you write critical user stories for your architecture, include performance requirements, such as specifying how quickly each critical story should run. For these critical stories, implement additional scripted user journeys to ensure that you know how these stories perform against your requirement.

Common anti-patterns:

- You assume that performance events are one-time issues and only related to anomalies.
- You only evaluate existing performance metrics when responding to performance events.

Benefits of establishing this best practice: In determine whether your workload is operating at expected levels, you must respond to performance events by gathering additional metric data for analysis. This data is used to understand the impact of the performance event and suggest changes to improve workload performance.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Prioritize experience concerns for critical user stories: When you write critical user stories for your architecture, include performance requirements, such as specifying how quickly each critical story should run. For these critical stories, implement additional scripted user journeys to ensure that you know how the user stories perform against your requirements.

Resources

Related documents:

- CloudWatch Documentation
- Amazon CloudWatch Synthetics
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation

Related videos:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF07-BP03 Establish key performance indicators (KPIs) to measure workload performance

Identify the KPIs that quantitatively and qualitatively measures workload performance. KPIs help to measure the health of a workload as it relates to a business goal. KPIs allow business and engineering teams to align on the measurement of goals and strategies and how this combines to produce business outcomes. KPIs should be revisited when business goals, strategies, or end-user requirements change.

For example, a website workload might use the page load time as an indication of overall performance. This metric would be one of the multiple data points which measure an end user experience. In addition to identifying the page load time thresholds, you should document the expected outcome or business risk if the performance is not met. A long page load time would affect your end users directly, decrease their user experience rating and might lead to a loss of customers. When you define your KPI thresholds, combine both industry benchmarks and your end user expectations. For example, if the current industry benchmark is a webpage loading within a two second time period, but your end users expect a webpage to load within a one second time period, then you should take both of these data points into consideration when establishing the KPI. Another example of a KPI might focus on meeting internal performance needs. A KPI threshold might be established on generating sales reports within one business day after production data has been generated. These reports might directly affect daily decisions and business outcomes.

Desired outcome: Establishing KPIs involve different departments and stakeholders. Your team must evaluate your workload KPIs using real-time granular data and historical data for reference and create dashboards that perform metric math on your KPI data to derive operational and utilization insights. KPIs should be documented which explains the agreed upon KPIs and thresholds that support business goals and strategies as well as mapped to metrics being

monitored. The KPIs are identifying performance requirements, reviewed intentionally and are frequently shared and understood with all teams. Risks and tradeoffs are clearly identified and understood how business is impact within KPI thresholds are not met.

Common anti-patterns:

- You only monitor system level metrics to gain insight into your workload and don't understand business impacts to those metrics.
- You assume that your KPIs are already being published and shared as standard metric data.
- Defining KPIs but not sharing them with all the teams.
- Not defining a quantitative, measurable KPI.
- Not aligning KPIs with business goals or strategies.

Benefits of establishing this best practice: Identifying specific metrics which represent workload health help to align teams on their priorities and defining successful business outcomes. Sharing those metrics with all departments provides visibility and alignment on thresholds, expectations, and business impact.

Level of risk exposed if this best practice is not established: High

Implementation guidance

All departments and business teams impacted by the health of the workload should contribute to defining KPIs. A single person should drive the collaboration, timelines, documentation, and information related to an organization's KPIs. This single threaded owner will often share the business goals and strategies and assign business stakeholders tasks to create KPIs in their respective departments. Once KPIs are defined, the operations team will often help define the metrics that will support and inform the success of the different KPIs. KPIs are only effective if all team members supporting a workload are aware of the KPIs.

Implementation steps

1. Identify and document business stakeholders.
2. Identify company goals and strategies.
3. Review common industry KPIs that align with your company goals and strategies.
4. Review end user expectations of your workload.

5. Define and document KPIs that support company goals and strategies.
6. Identify and document approved tradeoff strategies to meet the KPIs.
7. Identify and document metrics that will inform the KPIs.
8. Identify and document KPI thresholds for severity or alarm level.
9. Identify and document the risk and impact if the KPI is not met.
10. Identify the frequency of review per KPI.
11. Communicate KPI documentation with all teams supporting the workload.

Level of effort for the implementation guidance: Defining and communicating the KPIs is a *low* amount of work. This can typically be done over a few weeks meeting with business stakeholders, reviewing goals, strategies, and workload metrics.

Resources

Related documents:

- CloudWatch documentation
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation
- Using Amazon CloudWatch dashboards
- Quick KPIs

Related videos:

- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R)
- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Build a Monitoring Plan

Related examples:

- Creating a dashboard with Quick

PERF07-BP04 Use monitoring to generate alarm-based notifications

Using the performance-related key performance indicators (KPIs) that you defined, use a monitoring system that generates alarms automatically when these measurements are outside expected boundaries.

Amazon CloudWatch can collect metrics across the resources in your architecture. You can also collect and publish custom metrics to surface business or derived metrics. Use CloudWatch or a third-party monitoring service to set alarms that indicate when thresholds are breached — alarms signal that a metric is outside of the expected boundaries.

Common anti-patterns:

- You rely on staff to watch metrics and react when they see an issue.
- You rely solely on operational runbooks, when serverless workflows could be started to accomplish the same task.

Benefits of establishing this best practice: You can set alarms and automate actions based on either predefined thresholds, or on machine learning algorithms that identify anomalous behavior in your metrics. These same alarms can also start serverless workflows, which can modify performance characteristics of your workload (for example, increasing compute capacity, altering database configuration).

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Monitor metrics: Amazon CloudWatch can collect metrics across the resources in your architecture. You can collect and publish custom metrics to surface business or derived metrics. Use CloudWatch or a third-party monitoring service to set alarms that indicate when thresholds are exceeded.

Resources

Related documents:

- CloudWatch Documentation
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation

- Using Alarms and Alarm Actions in CloudWatch

Related videos:

- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R)
- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Build a Monitoring Plan
- Using AWS Lambda with Amazon CloudWatch Events

Related examples:

- Cloudwatch Logs Customize Alarms

PERF07-BP05 Review metrics at regular intervals

As routine maintenance, or in response to events or incidents, review which metrics are collected. Use these reviews to identify which metrics were essential in addressing issues and which additional metrics, if they were being tracked, would help to identify, address, or prevent issues.

As part of responding to incidents or events, evaluate which metrics were helpful in addressing the issue and which metrics could have helped that are not currently being tracked. Use this to improve the quality of metrics you collect so that you can prevent or more quickly resolve future incidents.

Common anti-patterns:

- You allow metrics to stay in an alarm state for an extended period of time.
- You create alarms that are not actionable by an automation system.

Benefits of establishing this best practice: Continually review metrics that are being collected to ensure that they properly identify, address, or prevent issues. Metrics can also become stale if you let them stay in an alarm state for an extended period of time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Constantly improve metric collection and monitoring: As part of responding to incidents or events, evaluate which metrics were helpful in addressing the issue and which metrics could have helped

that are not currently being tracked. Use this method to improve the quality of metrics you collect so that you can prevent or more quickly resolve future incidents.

Resources

Related documents:

- CloudWatch Documentation
- Collect metrics and logs from Amazon EC2 Instances and on-premises servers with the CloudWatch Agent
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation

Related videos:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Application Performance Management on AWS
- Build a Monitoring Plan

Related examples:

- Creating a dashboard with Quick
- Level 100: Monitoring with CloudWatch Dashboards

PERF07-BP06 Monitor and alarm proactively

Use key performance indicators (KPIs), combined with monitoring and alerting systems, to proactively address performance-related issues. Use alarms to start automated actions to remediate issues where possible. Escalate the alarm to those able to respond if automated response is not possible. For example, you may have a system that can predict expected key performance indicators (KPI) values and alarm when they breach certain thresholds, or a tool that can automatically halt or roll back deployments if KPIs are outside of expected values.

Implement processes that provide visibility into performance as your workload is running. Build monitoring dashboards and establish baseline norms for performance expectations to determine if the workload is performing optimally.

Common anti-patterns:

- You only allow operations staff the ability to make operational changes to the workload.
- You let all alarms filter to the operations team with no proactive remediation.

Benefits of establishing this best practice: Proactive remediation of alarm actions allows support staff to concentrate on those items that are not automatically actionable. This ensures that operations staff are not overwhelmed by all alarms and instead focus only on critical alarms.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Monitor performance during operations: Implement processes that provide visibility into performance as your workload is running. Build monitoring dashboards and establish a baseline for performance expectations.

Resources

Related documents:

- CloudWatch Documentation
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation
- Using Alarms and Alarm Actions in CloudWatch

Related videos:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Application Performance Management on AWS
- Build a Monitoring Plan
- Using AWS Lambda with Amazon CloudWatch Events

Related examples:

- Cloudwatch Logs Customize Alarms

Tradeoffs

Question

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

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

When architecting solutions, determining tradeoffs allows you to select the more effective approach. Often you can improve performance by trading consistency, durability, and space for time and latency.

Best practices

- PERF08-BP01 Understand the areas where performance is most critical
- PERF08-BP02 Learn about design patterns and services
- PERF08-BP03 Identify how tradeoffs impact customers and efficiency
- PERF08-BP04 Measure the impact of performance improvements
- PERF08-BP05 Use various performance-related strategies

PERF08-BP01 Understand the areas where performance is most critical

Understand and identify areas where increasing the performance of your workload will have a positive impact on efficiency or customer experience. For example, a website that has a large amount of customer interaction can benefit from using edge services to move content delivery closer to customers.

Desired outcome: Increase performance efficiency by understanding your architecture, traffic patterns, and data access patterns, and identify your latency and processing times. Identify the potential bottlenecks that might affect the customer experience as the workload grows. When you identify those areas, look at which solution you could deploy to remove those performance concerns.

Common anti-patterns:

- You assume that standard compute metrics such as `CPUUtilization` or memory pressure are enough to catch performance issues.
- You only use the default metrics recorded by your selected monitoring software.

- You only review metrics when there is an issue.

Benefits of establishing this best practice: Understanding critical areas of performance helps workload owners monitor KPIs and prioritize high-impact improvements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Set up end-to-end tracing to identify traffic patterns, latency, and critical performance areas. Monitor your data access patterns for slow queries or poorly fragmented and partitioned data. Identify the constrained areas of the workload using load testing or monitoring.

Implementation steps

1. Set up end-to-end monitoring to capture all workload components and metrics.
   - Use Amazon CloudWatch Real-User Monitoring (RUM) to capture application performance metrics from real user client-side and frontend sessions.
   - Set up AWS X-Ray to trace traffic through the application layers and identify latency between components and dependencies. Use the X-Ray service maps to see relationships and latency between workload components.
   - Use Amazon Relational Database Service Performance Insights to view database performance metrics and identify performance improvements.
   - Use Amazon RDS Enhanced Monitoring to view database OS performance metrics.
   - Collect CloudWatch metrics per workload component and service and identify which metrics impact performance efficiency.
   - Set up Amazon DevOps Guru for additional performance insights and recommendations
2. Perform tests to generate metrics, identify traffic patterns, bottlenecks, and critical performance areas.
   - Set up CloudWatch Synthetic Canaries to mimic browser-based user activities programmatically using `cron` jobs or rate expressions to generate consistent metrics over time.
   - Use the AWS Distributed Load Testing solution to generate peak traffic or test the workload at the expected growth rate.
3. Evaluate the metrics and telemetry to identify your critical performance areas. Review these areas with your team to discuss monitoring and solutions to avoid bottlenecks.

4. Experiment with performance improvements and measure those changes with data.
    - Use CloudWatch Evidently to test new improvements and the performance impact to the workload.

Level of effort for the implementation plan: To establish this best practice, you must review your end-to-end metrics and be aware of your current workload performance. This is a moderate level of effort to set up end to end monitoring and identify your critical performance areas.

Resources

Related documents:

- Amazon Builders' Library
- X-Ray Documentation
- Amazon CloudWatch RUM
- Amazon DevOps Guru
- CloudWatch RUM and X-Ray

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- X-Ray SDK for Node.js
- X-Ray SDK for Python
- X-Ray SDK for Java
- X-Ray SDK for .Net
- X-Ray SDK for Ruby
- X-Ray Daemon
- Distributed Load Testing on AWS

PERF08-BP02 Learn about design patterns and services

Research and understand the various design patterns and services that help improve workload performance. As part of the analysis, identify what you could trade to achieve higher performance. For example, using a cache service can help to reduce the load placed on database systems. However, caching can introduce eventual consistency and requires engineering effort to implement within business requirements and customer expectations.

Desired outcome: Researching design patterns will lead you to choosing an architecture design that will support the best performing system. Learn which performance configuration options are available to you and how they could impact the workload. Optimizing the performance of your workload depends on understanding how these options interact with your architecture and the impact they will have on both measured performance and the performance perceived by end users.

Common anti-patterns:

- You assume that all traditional IT workload performance strategies are best suited for cloud workloads.
- You build and manage caching solutions instead of using managed services.
- You use the same design pattern for all your workloads without evaluating which pattern would improve the workload performance.

Benefits of establishing this best practice: By selecting the right design pattern and services for your workload you will be optimizing your performance, improving operational excellence and increasing reliability. The right design pattern will meet your current workload characteristics and help you scale for future growth or changes.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Learn which performance configuration options are available and how they could impact the workload. Optimizing the performance of your workload depends on understanding how these options interact with your architecture, and the impact they have on measured performance and user-perceived performance.

Implementation steps:

1. Evaluate and review design patterns that would improve your workload performance.

a. The Amazon Builders' Library provides you with a detailed description of how Amazon builds and operates technology. These articles are written by senior engineers at Amazon and cover topics across architecture, software delivery, and operations.

b. AWS Solutions Library is a collection of ready-to-deploy solutions that assemble services, code, and configurations. These solutions have been created by AWS and AWS Partners based on common use cases and design patterns grouped by industry or workload type. For example, you can set up a distributed load testing solution for your workload.

c. AWS Architecture Center provides reference architecture diagrams grouped by design pattern, content type, and technology.

d. AWS samples is a GitHub repository full of hands-on examples to help you explore common architecture patterns, solutions, and services. It is updated frequently with the newest services and examples.

2. Improve your workload to model the selected design patterns and use services and the service configuration options to improve your workload performance.

   a. Train your internal team with resources available at AWS Skills Guild.

   b. Use the AWS Partner Network to provide expertise quickly and to scale your ability to make improvements.

Level of effort for the implementation plan: To establish this best practice, you must be aware of the design patterns and services that could help improve your workload performance. After evaluating the design patterns, implementing the design patterns is a *high* level of effort.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- Amazon Builders' Library
- Using load shedding to avoid overload
- Caching challenges and strategies

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is My Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF08-BP03 Identify how tradeoffs impact customers and efficiency

When evaluating performance-related improvements, determine which choices will impact your customers and workload efficiency. For example, if using a key-value data store increases system performance, it is important to evaluate how the eventually consistent nature of it will impact customers.

Identify areas of poor performance in your system through metrics and monitoring. Determine how you can make improvements, what trade-offs those improvements bring, and how they impact the system and the user experience. For example, implementing caching data can help dramatically improve performance but requires a clear strategy for how and when to update or invalidate cached data to prevent incorrect system behavior.

Common anti-patterns:

- You assume that all performance gains should be implemented, even if there are tradeoffs for implementation such as eventual consistency.
- You only evaluate changes to workloads when a performance issue has reached a critical point.

Benefits of establishing this best practice: When you are evaluating potential performance-related improvements, you must decide if the tradeoffs for the changes are consistent with the workload requirements. In some cases, you may have to implement additional controls to compensate for the tradeoffs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify tradeoffs: Use metrics and monitoring to identify areas of poor performance in your system. Determine how to make improvements, and how tradeoffs will impact the system and the user experience. For example, implementing caching data can help dramatically improve performance, but it requires a clear strategy for how and when to update or invalidate cached data to prevent incorrect system behavior.

Resources

Related documents:

- Amazon Builders' Library
- Quick KPIs
- Amazon CloudWatch RUM
- X-Ray Documentation

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- Build a Monitoring Plan
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF08-BP04 Measure the impact of performance improvements

As changes are made to improve performance, evaluate the collected metrics and data. Use this information to determine impact that the performance improvement had on the workload, the workload's components, and your customers. This measurement helps you understand the improvements that result from the tradeoff, and helps you determine if any negative side-effects were introduced.

A well-architected system uses a combination of performance related strategies. Determine which strategy will have the largest positive impact on a given hotspot or bottleneck. For example, sharding data across multiple relational database systems could improve overall throughput while retaining support for transactions and, within each shard, caching can help to reduce the load.

Common anti-patterns:

- You deploy and manage technologies manually that are available as managed services.
- You focus on just one component, such as networking, when multiple components could be used to increase performance of the workload.
- You rely on customer feedback and perceptions as your only benchmark.

Benefits of establishing this best practice: For implementing performance strategies, you must select multiple services and features that, taken together, will allow you to meet your workload requirements for performance.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

A well-architected system uses a combination of performance-related strategies. Determine which strategy will have the largest positive impact on a given hotspot or bottleneck. For example, sharding data across multiple relational database systems could improve overall throughput while retaining support for transactions and, within each shard, caching can help to reduce the load.

Resources

Related documents:

- Amazon Builders' Library
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Distributed Load Testing on AWS

Related videos:

- Introducing The Amazon Builders' Library (DOP328)

- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- Distributed Load Testing on AWS

PERF08-BP05 Use various performance-related strategies

Where applicable, use multiple strategies to improve performance. For example, using strategies like caching data to prevent excessive network or database calls, using read-replicas for database engines to improve read rates, sharding or compressing data where possible to reduce data volumes, and buffering and streaming of results as they are available to avoid blocking.

As you make changes to the workload, collect and evaluate metrics to determine the impact of those changes. Measure the impacts to the system and to the end-user to understand how your trade-offs impact your workload. Use a systematic approach, such as load testing, to explore whether the tradeoff improves performance.

Common anti-patterns:

- You assume that workload performance is adequate if customers are not complaining.
- You only collect data on performance after you have made performance-related changes.

Benefits of establishing this best practice: To optimize performance and resource utilization, you need a unified operational view, real-time granular data, and historical reference. You can create dashboards and perform metric math on your data to derive operational and utilization insights for your workloads as they change over time.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use a data-driven approach to evolve your architecture: As you make changes to the workload, collect and evaluate metrics to determine the impact of those changes. Measure the impacts to

the system and to the end-user to understand how your tradeoffs impact your workload. Use a systematic approach, such as load testing, to explore whether the tradeoff improves performance.

Resources

Related documents:

- Amazon Builders' Library
- Best Practices for Implementing Amazon ElastiCache
- AWS Database Caching
- Amazon CloudWatch RUM
- Distributed Load Testing on AWS

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- AWS purpose-built databases (DAT209-L)
- Optimize applications through Amazon CloudWatch RUM

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- Distributed Load Testing on AWS

Cost optimization

The Cost Optimization pillar includes the ability to run systems to deliver business value at the lowest price point. You can find prescriptive guidance on implementation in the Cost Optimization Pillar whitepaper.

Best practice areas

- Practice Cloud Financial Management
- Expenditure and usage awareness
- Cost-effective resources

- Manage demand and supply resources
- Optimize over time

Practice Cloud Financial Management

Question

- COST 1. How do you implement cloud financial management?

COST 1. How do you implement cloud financial management?

Implementing Cloud Financial Management helps organizations realize business value and financial success as they optimize their cost and usage and scale on AWS.

Best practices

- COST01-BP01 Establish a cost optimization function
- COST01-BP02 Establish a partnership between finance and technology
- COST01-BP03 Establish cloud budgets and forecasts
- COST01-BP04 Implement cost awareness in your organizational processes
- COST01-BP05 Report and notify on cost optimization
- COST01-BP06 Monitor cost proactively
- COST01-BP07 Keep up-to-date with new service releases
- COST01-BP08 Create a cost-aware culture
- COST01-BP09 Quantify business value from cost optimization

COST01-BP01 Establish a cost optimization function

Create a team (Cloud Business Office or Cloud Center of Excellence) that is responsible for establishing and maintaining cost awareness across your organization. The team requires people from finance, technology, and business roles across the organization.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Establish a Cloud Business Office (CBO) or Cloud Center of Excellence (CCOE) team that is responsible for establishing and maintaining a culture of cost awareness in cloud computing.

It can be an existing individual, a team within your organization, or a new team of key finance, technology and organization stakeholders from across the organization.

The function (individual or team) prioritizes and spends the required percentage of their time on cost management and cost optimization activities. For a small organization, the function might spend a smaller percentage of time compared to a full-time function for a larger enterprise.

The function requires a multi-disciplined approach, with capabilities in project management, data science, financial analysis, and software or infrastructure development. The function can improve efficiencies of workloads by running cost optimizations within three different ownerships:

- **Centralized:** Through designated teams such as finance operations, cost optimization, CBO, or CCOE, customers can design and implement governance mechanisms and drive best practices company-wide.
- **Decentralized:** Influencing technology teams to run optimizations.
- **Hybrid:** A combination of both centralized and decentralized teams can work together to run cost optimizations.

The function may be measured against their ability to run and deliver against cost optimization goals (for example, workload efficiency metrics).

You must secure executive sponsorship for this function to make changes, which is a key success factor. The sponsor is regarded as champion for cost efficient cloud consumption, and provides escalation support for the function to ensure that cost optimization activities are treated with the level of priority defined by the organization. Otherwise, guidance will be ignored and cost-saving opportunities will not be prioritized. Together, the sponsor and function ensure that your organization consumes the cloud efficiently and continues to deliver business value.

If you have a Business, Enterprise-On-Ramp, or Enterprise Support plan, and need help to build this team or function, reach out to Cloud Finance Management (CFM) experts through your Account team.

Implementation steps

- **Define key members:** You need to ensure that all relevant parts of your organization contribute and have a stake in cost management. Common teams within organizations typically include: finance, application or product owners, management, and technical teams (DevOps). Some are engaged full time (finance, technical), others periodically as required. Individuals or teams performing CFM generally need the following set of skills:

  - Software development skills - in the case where scripts and automation are being built out.
  - Infrastructure engineering skills - to deploy scripts or automation, and understand how services or resources are provisioned.
  - Operations acumen - CFM is about operating on the cloud efficiently by measuring, monitoring, modifying, planning and scaling efficient use of the cloud.
- **Define goals and metrics:** The function needs to deliver value to the organization in different ways. These goals are defined and continually evolve as the organization evolves. Common activities include: creating and running education programs on cost optimization across the organization, developing organization-wide standards, such as monitoring and reporting for cost optimization, and setting workload goals on optimization. This function also needs to regularly report to the organization on the organization's cost optimization capability.

  You can define value-based key performance indicators (KPIs). KPIs can be cost-based or value-based. When you define the KPIs, you can calculate expected cost in terms of efficiency and expected business outcome. Value-based KPIs tie cost and usage metrics to business value drivers and help us rationalize changes in our AWS spend. The first step to deriving value-based KPIs is working together, cross-organizationally, to select and agree upon a standard set of KPIs.
- **Establish regular cadence:** The group (finance, technology, and business teams) should come together regularly to review their goals and metrics. A typical cadence involves reviewing the state of the organization, reviewing any programs currently running, and reviewing overall financial and optimization metrics. Then key workloads are reported on in greater detail.

  During these regular meetings, you can review workload efficiency (cost) and business outcome. For example, a 20% cost increase for a workload may align with increased customer usage. In this case, this 20% cost increase can be interpreted as an investment. These regular cadence calls can help teams to identify value-based KPIs that provide meaning to the entire organization.

Resources

Related documents:

- AWS CCOE Blog
- Creating Cloud Business Office
- CCOE - Cloud Center of Excellence

Related videos:

- Vanguard CCOE Success Story

Related examples:

- Using a Cloud Center of Excellence (CCOE) to Transform the Entire Enterprise
- Building a CCOE to transform the entire enterprise
- 7 Pitfalls to Avoid When Building CCOE

COST01-BP02 Establish a partnership between finance and technology

Involve finance and technology teams in cost and usage discussions at all stages of your cloud journey. Teams regularly meet and discuss topics such as organizational goals and targets, current state of cost and usage, and financial and accounting practices.

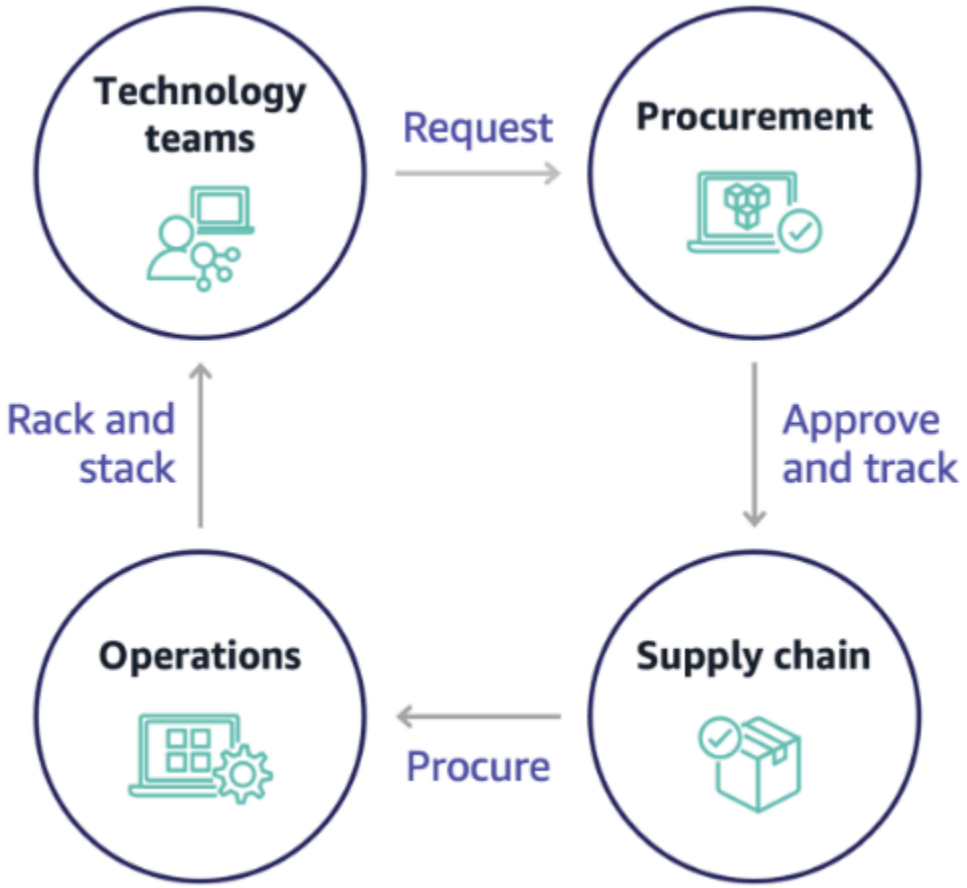
Level of risk exposed if this best practice is not established: High

Implementation guidance

Technology teams innovate faster in the cloud due to shortened approval, procurement, and infrastructure deployment cycles. This can be an adjustment for finance organizations previously used to running time-consuming and resource-intensive processes for procuring and deploying capital in data center and on-premises environments, and cost allocation only at project approval.

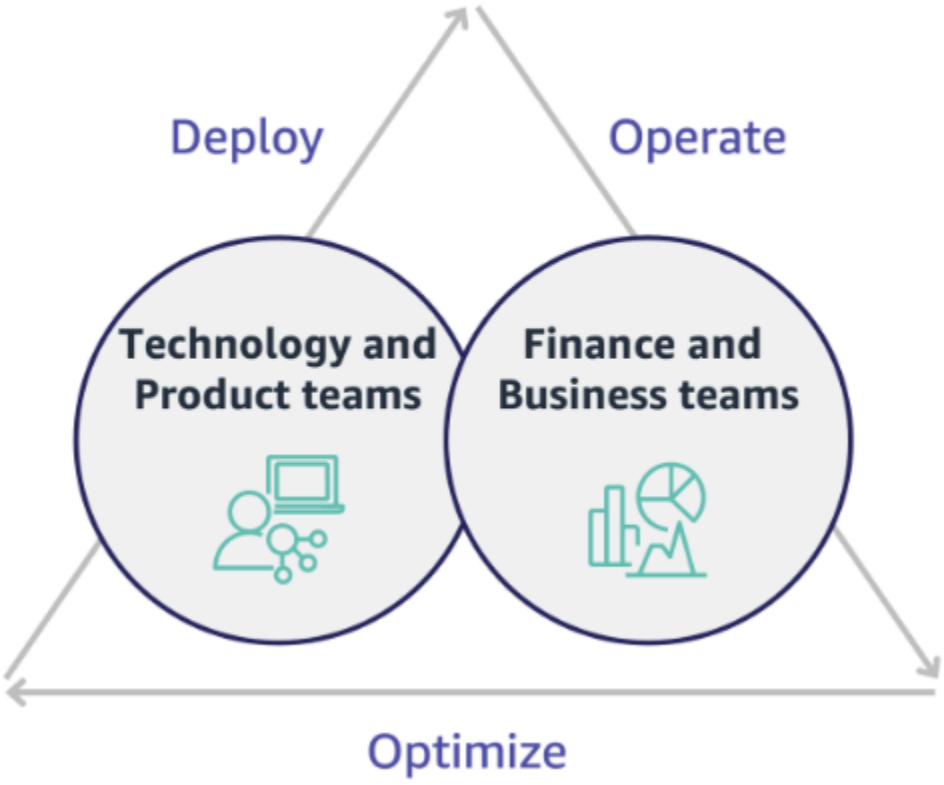
From a finance and procurement organization perspective, the process for capital budgeting, capital requests, approvals, procurement, and installing physical infrastructure is one that has been learned and standardized over decades:

- Engineering or IT teams are typically the requesters
- Various finance teams act as approvers and procurers
- Operations teams rack, stack, and hand off ready-to-use infrastructure



With the adoption of cloud, infrastructure procurement and consumption are no longer beholden to a chain of dependencies. In the cloud model, technology and product teams are no longer just builders, but operators and owners of their products, responsible for most of the activities historically associated with finance and operations teams, including procurement and deployment.

All it really takes to provision cloud resources is an account, and the right set of permissions. This is also what reduces IT and finance risk; which means teams are always a just few clicks or API calls away from terminating idle or unnecessary cloud resources. This is also what allows technology teams to innovate faster – the agility and ability to spin up and then tear down experiments. While the variable nature of cloud consumption may impact predictability from a capital budgeting and forecasting perspective, cloud provides organizations with the ability to reduce the cost of over-provisioning, as well as reduce the opportunity cost associated with conservative under-provisioning.



Establish a partnership between key finance and technology stakeholders to create a shared understanding of organizational goals and develop mechanisms to succeed financially in the variable spend model of cloud computing. Relevant teams within your organization must be involved in cost and usage discussions at all stages of your cloud journey, including:

- **Financial leads:** CFOs, financial controllers, financial planners, business analysts, procurement, sourcing, and accounts payable must understand the cloud model of consumption, purchasing options, and the monthly invoicing process. Finance needs to partner with technology teams to create and socialize an IT value story, helping business teams understand how technology spend is linked to business outcomes. This way, technology expenditures are viewed not as costs, but rather as investments. Due to the fundamental differences between the cloud (such as the rate of change in usage, pay as you go pricing, tiered pricing, pricing models, and detailed billing and usage information) compared to on-premises operation, it is essential that the finance organization understands how cloud usage can impact business aspects including procurement processes, incentive tracking, cost allocation and financial statements.
- **Technology leads:** Technology leads (including product and application owners) must be aware of the financial requirements (for example, budget constraints) as well as business requirements

(for example, service level agreements). This allows the workload to be implemented to achieve the desired goals of the organization.

The partnership of finance and technology provides the following benefits:

- Finance and technology teams have near real-time visibility into cost and usage.
- Finance and technology teams establish a standard operating procedure to handle cloud spend variance.
- Finance stakeholders act as strategic advisors with respect to how capital is used to purchase commitment discounts (for example, Reserved Instances or AWS Savings Plans), and how the cloud is used to grow the organization.
- Existing accounts payable and procurement processes are used with the cloud.
- Finance and technology teams collaborate on forecasting future AWS cost and usage to align and build organizational budgets.
- Better cross-organizational communication through a shared language, and common understanding of financial concepts.

Additional stakeholders within your organization that should be involved in cost and usage discussions include:

- **Business unit owners:** Business unit owners must understand the cloud business model so that they can provide direction to both the business units and the entire company. This cloud knowledge is critical when there is a need to forecast growth and workload usage, and when assessing longer-term purchasing options, such as Reserved Instances or Savings Plans.
- **Engineering team:** Establishing a partnership between finance and technology teams is essential for building a cost-aware culture that encourages engineers to take action on Cloud Financial Management (CFM). One of the common problems of CFM or finance operations practitioners and finance teams is getting engineers to understand the whole business on cloud, follow best practices, and take recommended actions.
- **Third parties:** If your organization uses third parties (for example, consultants or tools), ensure that they are aligned to your financial goals and can demonstrate both alignment through their engagement models and a return on investment (ROI). Typically, third parties will contribute to reporting and analysis of any workloads that they manage, and they will provide cost analysis of any workloads that they design.

Implementing CFM and achieving success requires collaboration across finance, technology, and business teams, and a shift in how cloud spend is communicated and evaluated across the organization. Include engineering teams so that they can be part of these cost and usage discussions at all stages, and encourage them to follow best practices and take agreed-upon actions accordingly.

Implementation steps

- **Define key members:** Verify that all relevant members of your finance and technology teams participate in the partnership. Relevant finance members will be those having interaction with the cloud bill. This will typically be CFOs, financial controllers, financial planners, business analysts, procurement, and sourcing. Technology members will typically be product and application owners, technical managers and representatives from all teams that build on the cloud. Other members may include business unit owners, such as marketing, that will influence usage of products, and third parties such as consultants, to achieve alignment to your goals and mechanisms, and to assist with reporting.
- **Define topics for discussion:** Define the topics that are common across the teams, or will need a shared understanding. Follow cost from that time it is created, until the bill is paid. Note any members involved, and organizational processes that are required to be applied. Understand each step or process it goes through and the associated information, such as pricing models available, tiered pricing, discount models, budgeting, and financial requirements.
- **Establish regular cadence:** To create a finance and technology partnership, establish a regular communication cadence to create and maintain alignment. The group needs to come together regularly against their goals and metrics. A typical cadence involves reviewing the state of the organization, reviewing any programs currently running, and reviewing overall financial and optimization metrics. Then key workloads are reported on in greater detail.

Resources

Related documents:

- AWS News Blog

COST01-BP03 Establish cloud budgets and forecasts

Adjust existing organizational budgeting and forecasting processes to be compatible with the highly variable nature of cloud costs and usage. Processes must be dynamic using trend-based or business driver-based algorithms, or a combination of both.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Customers use the cloud for efficiency, speed and agility, which creates a highly variable amount of cost and usage. Costs can decrease with increases in workload efficiency, or as new workloads and features are deployed. It is possible to see the cost increase when the workload efficiency increases, or as new workloads and features are deployed. Or, workloads will scale to serve more of your customers, which increases cloud usage and costs. Resources are now more readily accessible than ever before. With the elasticity of the cloud also brings an elasticity of costs and forecasts. Existing organizational budgeting processes must be modified to incorporate this variability.

Adjust existing budgeting and forecasting processes to become more dynamic using either a trend-based algorithm (using historical costs as inputs), or using business-driver-based algorithms (for example, new product launches or regional expansion), or a combination of both trend and business drivers.

Use AWS Budgets to set custom budgets at a granular level by specifying the time period, recurrence, or amount (fixed or variable), and adding filters such as service, AWS Region, and tags. To stay informed on the performance of your existing budgets you can create and schedule AWS Budgets Reports to be emailed to you and your stakeholders on a regular cadence. You can also create AWS Budgets Alerts based on actual costs, which is reactive in nature, or on forecasted costs, which provides time to implement mitigations against potential cost overruns. You will be alerted when your cost or usage exceeds, or if they are forecasted to exceed, your budgeted amount.

AWS gives you the flexibility to build dynamic forecasting and budgeting processes so you can stay informed on whether costs adhere to, or exceed, budgetary limits.

Use AWS Cost Explorer to forecast costs in a defined future time range based on your past spend. AWS Cost Explorer's forecasting engine segments your historical data based on charge types (for example, Reserved Instances) and uses a combination of machine learning and rule-based models to predict spend across all charge types individually. Use AWS Cost Explorer to forecast daily (up to three months) or monthly (up to 12 months) cloud costs based on machine learning algorithms applied to your historical costs (trend-based).

Once you've determined your trend-based forecast using Cost Explorer, use the AWS Pricing Calculator to estimate your AWS use case and future costs based on the expected usage (traffic, requests-per-second, required Amazon Elastic Compute Cloud (Amazon EC2) instance, and so forth). You can also use it to help you plan how you spend, find cost saving opportunities, and make informed decisions when using AWS.

Use AWS Cost Anomaly Detection to prevent or reduce cost surprises and enhance control without slowing innovation. AWS Cost Anomaly Detection leverages advanced machine learning technologies to identify anomalous spend and root causes, so you can quickly take action. With three simple steps, you can create your own contextualized monitor and receive alerts when any anomalous spend is detected. Let builders build, and let AWS Cost Anomaly Detection monitor your spend and reduce the risk of billing surprises.

As mentioned in the Well-Architected Cost Optimization Pillar's Finance and Technology Partnership section, it is important to have partnership and cadences between IT, Finance and other stakeholders to ensure that they are all using the same tooling or processes for consistency. In cases where budgets may need to change, increasing cadence touch points can help react to those changes more quickly.

Implementation steps

- **Update existing budget and forecasting processes:** Implement trend-based, business driver-based, or a combination of both in your budgeting and forecasting processes.
- **Configure alerts and notifications:** Use AWS Budgets Alerts and Cost Anomaly Detection.
- **Perform regular reviews with key stakeholders:** For example, stakeholders in IT, Finance, Platform, and other areas of the business, to align with changes in business direction and usage.

Resources

Related documents:

- AWS Cost Explorer
- AWS Budgets
- AWS Pricing Calculator
- AWS Cost Anomaly Detection
- AWS License Manager

Related examples:

- Launch: Usage-Based Forecasting now Available in AWS Cost Explorer
- AWS Well-Architected Labs - Cost and Usage Governance

COST01-BP04 Implement cost awareness in your organizational processes

Implement cost awareness, create transparency, and accountability of costs into new or existing processes that impact usage, and leverage existing processes for cost awareness. Implement cost awareness into employee training.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Cost awareness must be implemented in new and existing organizational processes. It is one of the foundational, prerequisite capabilities for other best practices. It is recommended to reuse and modify existing processes where possible — this minimizes the impact to agility and velocity. Report cloud costs to the technology teams and the decision makers in the business and finance teams to raise cost awareness, and establish efficiency key performance indicators (KPIs) for finance and business stakeholders. The following recommendations will help implement cost awareness in your workload:

- Verify that change management includes a cost measurement to quantify the financial impact of your changes. This helps proactively address cost-related concerns and highlight cost savings.
- Verify that cost optimization is a core component of your operating capabilities. For example, you can leverage existing incident management processes to investigate and identify root causes for cost and usage anomalies or cost overruns.
- Accelerate cost savings and business value realization through automation or tooling. When thinking about the cost of implementing, frame the conversation to include an return on investment (ROI) component to justify the investment of time or money.
- Allocate cloud costs by implementing showbacks or chargebacks for cloud spend, including spend on commitment-based purchase options, shared services and marketplace purchases to drive most cost-aware cloud consumption.
- Extend existing training and development programs to include cost-awareness training throughout your organization. It is recommended that this includes continuous training and certification. This will build an organization that is capable of self-managing cost and usage.

- Take advantage of free AWS native tools such as AWS Cost Anomaly Detection, AWS Budgets, and AWS Budgets Reports.

When organizations consistently adopt Cloud Financial Management (CFM) practices, those behaviours become ingrained in the way of working and decision-making. The result is a culture that is more cost-aware, from developers architecting a new born-in-the-cloud application, to finance managers analyzing the ROI on these new cloud investments.

Implementation steps

- **Identify relevant organizational processes:** Each organizational unit reviews their processes and identifies processes that impact cost and usage. Any processes that result in the creation or termination of a resource need to be included for review. Look for processes that can support cost awareness in your business, such as incident management and training.
- **Establish self-sustaining cost-aware culture:** Make sure all the relevant stakeholders align with cause-of-change and impact as a cost so that they understand cloud cost. This will allow your organization to establish a self-sustaining cost-aware culture of innovation.
- **Update processes with cost awareness:** Each process is modified to be made cost aware. The process may require additional pre-checks, such as assessing the impact of cost, or post-checks validating that the expected changes in cost and usage occurred. Supporting processes such as training and incident management can be extended to include items for cost and usage.

To get help, reach out to CFM experts through your Account team, or explore the resources and related documents below.

Resources

Related documents:

- AWS Cloud Financial Management

Related examples:

- Strategy for Efficient Cloud Cost Management
- Cost Control Blog Series #3: How to Handle Cost Shock
- A Beginner's Guide to AWS Cost Management

COST01-BP05 Report and notify on cost optimization

Configure AWS Budgets and AWS Cost Anomaly Detection to provide notifications on cost and usage against targets. Have regular meetings to analyze your workload's cost efficiency and to promote cost-aware culture.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

You must regularly report on cost and usage optimization within your organization. You can implement dedicated sessions to cost optimization, or include cost optimization in your regular operational reporting cycles for your workloads. Use services and tools to identify and implement cost savings opportunities. AWS Cost Explorer provides dashboards and reports. You can track your progress of cost and usage against configured budgets with AWS Budgets Reports.

Use AWS Budgets to set custom budgets to track your costs and usage, and respond quickly to alerts received from email or Amazon Simple Notification Service (Amazon SNS) notifications if you exceed your threshold. Set your preferred budget period to daily, monthly, quarterly, or annually, and create specific budget limits to stay informed on how actual or forecasted costs and usage progress toward your budget threshold. You can also set up alerts and actions against those alerts to run automatically, or through an approval process when a budget target is exceeded.

Implement notifications on cost and usage to ensure that changes in cost and usage can be acted upon quickly if they are unexpected. AWS Cost Anomaly Detection allows you to reduce cost surprises and enhance control without slowing innovation. AWS Cost Anomaly Detection identifies anomalous spend and root causes, which helps to reduce the risk of billing surprises. With three simple steps, you can create your own contextualized monitor and receive alerts when any anomalous spend is detected.

You can also use Amazon Quick with AWS Cost and Usage Report (CUR) data, to provide highly customized reporting with more granular data. Amazon Quick allows you to schedule reports and receive periodic Cost Report emails for historical cost and usage, or cost-saving opportunities.

Use AWS Trusted Advisor, which provides guidance to verify whether provisioned resources are aligned with AWS best practices for cost optimization.

Periodically create reports containing a highlight of Savings Plans, Reserved Instances and Amazon Elastic Compute Cloud (Amazon EC2) rightsizing recommendations from AWS Cost Explorer to start reducing the cost associated with steady-state workloads, idle, and underutilized resources.

Identify and recoup spend associated with cloud waste for resources that are deployed. Cloud waste occurs when incorrectly-sized resources are created, or different usage patterns are observed instead what is expected. Follow AWS best practices to reduce your waste and optimize and save your cloud costs.

Generate reports regularly for better purchasing options for your resources to drive down unit costs for your workloads. Purchasing options such as Savings Plans, Reserved Instances, or Amazon EC2 Spot Instances offer the deepest cost savings for fault-tolerant workloads and allow stakeholders (business owners, finance and tech teams) to be part of these commitment discussions.

Share the reports that contain opportunities or new release announcements that may help you to reduce total cost of ownership (TCO) of the cloud. Adopt new services, Regions, features, solutions, or new ways to achieve further cost reductions.

Implementation steps

- **Configure AWS Budgets:** Configure AWS Budgets on all accounts for your workload. Set a budget for the overall account spend, and a budget for the workload by using tags.
  - Well-Architected Labs: Cost and Governance Usage
- **Report on cost optimization:** Set up a regular cycle to discuss and analyze the efficiency of the workload. Using the metrics established, report on the metrics achieved and the cost of achieving them. Identify and fix any negative trends, and identify positive trends that you can promote across your organization. Reporting should involve representatives from the application teams and owners, finance, and management.
  - Well-Architected Labs: Visualization

Resources

Related documents:

- AWS Cost Explorer
- AWS Trusted Advisor
- AWS Budgets
- AWS Budgets Best Practices
- Amazon CloudWatch
- AWS CloudTrail

- Amazon S3 Analytics
- AWS Cost and Usage Report

Related examples:

- Well-Architected Labs: Cost and Governance Usage
- Well-Architected Labs: Visualization
- Key ways to start optimizing your AWS cloud costs

COST01-BP06 Monitor cost proactively

Implement tooling and dashboards to monitor cost proactively for the workload. Regularly review the costs with configured tools or out of the box tools, do not just look at costs and categories when you receive notifications. Monitoring and analyzing costs proactively helps to identify positive trends and allows you to promote them throughout your organization.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

It is recommended to monitor cost and usage proactively within your organization, not just when there are exceptions or anomalies. Highly visible dashboards throughout your office or work environment ensure that key people have access to the information they need, and indicate the organization's focus on cost optimization. Visible dashboards allow you to actively promote successful outcomes and implement them throughout your organization.

Create a daily or frequent routine to use AWS Cost Explorer or any other dashboard such as Amazon Quick to see the costs and analyze proactively. Analyze AWS service usage and costs at the AWS account-level, workload-level, or specific AWS service-level with grouping and filtering, and validate whether they are expected or not. Use the hourly- and resource-level granularity and tags to filter and identify incurring costs for the top resources. You can also build your own reports with the Cost Intelligence Dashboard, an Amazon Quick solution built by AWS Solutions Architects, and compare your budgets with the actual cost and usage.

Implementation steps

- **Report on cost optimization:** Set up a regular cycle to discuss and analyze the efficiency of the workload. Using the metrics established, report on the metrics achieved and the cost of

achieving them. Identify and fix any negative trends, and identify positive trends to promote across your organization. Reporting should involve representatives from the application teams and owners, finance, and management.

- **Create and activate daily granularity AWS Budgets for the cost and usage to take timely actions to prevent any potential cost overruns:** AWS Budgets allow you to configure alert notifications, so you stay informed if any of your budget types fall out of your pre-configured thresholds. The best way to leverage AWS Budgets is to set your expected cost and usage as your limits, so that anything above your budgets can be considered overspend.
- **Create AWS Cost Anomaly Detection for cost monitor:** AWS Cost Anomaly Detection uses advanced Machine Learning technology to identify anomalous spend and root causes, so you can quickly take action. It allows you to configure cost monitors that define spend segments you want to evaluate (for example, individual AWS services, member accounts, cost allocation tags, and cost categories), and lets you set when, where, and how you receive your alert notifications. For each monitor, attach multiple alert subscriptions for business owners and technology teams, including a name, a cost impact threshold, and alerting frequency (individual alerts, daily summary, weekly summary) for each subscription.
- **Use AWS Cost Explorer or integrate your AWS Cost and Usage Report (CUR) data with Amazon Quick dashboards to visualize your organization's costs:** AWS Cost Explorer has an easy-to-use interface that lets you visualize, understand, and manage your AWS costs and usage over time. The Cost Intelligence Dashboard is a customizable and accessible dashboard to help create the foundation of your own cost management and optimization tool.

Resources

Related documents:

- AWS Budgets
- AWS Cost Explorer
- Daily Cost and Usage Budgets
- AWS Cost Anomaly Detection

Related examples:

- Well-Architected Labs: Visualization
- Well-Architected Labs: Advanced Visualization

- Well-Architected Labs: Cloud Intelligence Dashboards
- Well-Architected Labs: Cost Visualization
- AWS Cost Anomaly Detection Alert with Slack

COST01-BP07 Keep up-to-date with new service releases

Consult regularly with experts or AWS Partners to consider which services and features provide lower cost. Review AWS blogs and other information sources.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS is constantly adding new capabilities so you can leverage the latest technologies to experiment and innovate more quickly. You may be able to implement new AWS services and features to increase cost efficiency in your workload. Regularly review AWS Cost Management, the AWS News Blog, the AWS Cost Management blog, and What's New with AWS for information on new service and feature releases. What's New posts provide a brief overview of all AWS service, feature, and Region expansion announcements as they are released.

Implementation steps

- **Subscribe to blogs:** Go to the AWS blogs pages and subscribe to the What's New Blog and other relevant blogs. You can sign up on the communication preference page with your email address.
- **Subscribe to AWS News:** Regularly review the AWS News Blog and What's New with AWS for information on new service and feature releases. Subscribe to the RSS feed, or with your email to follow announcements and releases.
- **Follow AWS Price Reductions:** Regular price cuts on all our services has been a standard way for AWS to pass on the economic efficiencies to our customers gained from our scale. As of April 2022, AWS has reduced prices 115 times since it was launched in 2006. If you have any pending business decisions due to price concerns, you can review them again after price reductions and new service integrations. You can learn about the previous price reductions efforts, including Amazon Elastic Compute Cloud (Amazon EC2) instances, in the price-reduction category of the AWS News Blog.
- **AWS events and meetups:** Attend your local AWS summit, and any local meetups with other organizations from your local area. If you cannot attend in person, try to attend virtual events to hear more from AWS experts and other customers' business cases.

- **Meet with your account team:** Schedule a regular cadence with your account team, meet with them and discuss industry trends and AWS services. Speak with your account manager, Solutions Architect, and support team.

Resources

Related documents:

- AWS Cost Management
- What's New with AWS
- AWS News Blog

Related examples:

- Amazon EC2 – 15 Years of Optimizing and Saving Your IT Costs
- AWS News Blog - Price Reduction

COST01-BP08 Create a cost-aware culture

Implement changes or programs across your organization to create a cost-aware culture. It is recommended to start small, then as your capabilities increase and your organization's use of the cloud increases, implement large and wide ranging programs.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

A cost-aware culture allows you to scale cost optimization and Cloud Financial Management (financial operations, cloud center of excellence, cloud operations teams, and so on) through best practices that are performed in an organic and decentralized manner across your organization. Cost awareness allows you to create high levels of capability across your organization with minimal effort, compared to a strict top-down, centralized approach.

Creating cost awareness in cloud computing, especially for primary cost drivers in cloud computing, allows teams to understand expected outcomes of any changes in cost perspective. Teams who access the cloud environments should be aware of pricing models and the difference between traditional on-premesis datacenters and cloud computing.

The main benefit of a cost-aware culture is that technology teams optimize costs proactively and continually (for example, they are considered a non-functional requirement when architecting new workloads, or making changes to existing workloads) rather than performing reactive cost optimizations as needed.

Small changes in culture can have large impacts on the efficiency of your current and future workloads. Examples of this include:

- Giving visibility and creating awareness in engineering teams to understand what they do, and what they impact in terms of cost.
- Gamifying cost and usage across your organization. This can be done through a publicly visible dashboard, or a report that compares normalized costs and usage across teams (for example, cost-per-workload and cost-per-transaction).
- Recognizing cost efficiency. Reward voluntary or unsolicited cost optimization accomplishments publicly or privately, and learn from mistakes to avoid repeating them in the future.
- Creating top-down organizational requirements for workloads to run at pre-defined budgets.
- Questioning business requirements of changes, and the cost impact of requested changes to the architecture infrastructure or workload configuration to make sure you pay only what you need.
- Making sure the change planner is aware of expected changes that have a cost impact, and that they are confirmed by the stakeholders to deliver business outcomes cost-effectively.

Implementation steps

- **Report cloud costs to technology teams:** To raise cost awareness, and establish efficiency KPIs for finance and business stakeholders.
- **Inform stakeholders or team members about planned changes:** Create an agenda item to discuss planned changes and the cost-benefit impact on the workload during weekly change meetings.
- **Meet with your account team:** Establish a regular meeting cadence with your account team, and discuss industry trends and AWS services. Speak with your account manager, architect, and support team.
- **Share success stories:** Share success stories about cost reduction for any workload, AWS account, or organization to create a positive attitude and encouragement around cost optimization.
- **Training:** Ensure technical teams or team members are trained for awareness of resource costs on AWS Cloud.

- **AWS events and meetups:** Attend local AWS summits, and any local meetups with other organizations from your local area.
- **Subscribe to blogs:** Go to the AWS blogs pages and subscribe to the What's New Blog and other relevant blogs to follow new releases, implementations, examples, and changes shared by AWS.

Resources

Related documents:

- AWS Blog
- AWS Cost Management
- AWS News Blog

Related examples:

- AWS Cloud Financial Management
- AWS Well-Architected Labs: Cloud Financial Management

COST01-BP09 Quantify business value from cost optimization

Quantifying business value from cost optimization allows you to understand the entire set of benefits to your organization. Because cost optimization is a necessary investment, quantifying business value allows you to explain the return on investment to stakeholders. Quantifying business value can help you gain more buy-in from stakeholders on future cost optimization investments, and provides a framework to measure the outcomes for your organization's cost optimization activities.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

In addition to reporting savings from cost optimization, it is recommended that you quantify the additional value delivered. Cost optimization benefits are typically quantified in terms of lower costs per business outcome. For example, you can quantify On-Demand Amazon Elastic Compute Cloud(Amazon EC2) cost savings when you purchase Savings Plans, which reduce cost and maintain workload output levels. You can quantify cost reductions in AWS spending when idle Amazon EC2 instances are terminated, or unattached Amazon Elastic Block Store (Amazon EBS) volumes are deleted.

The benefits from cost optimization, however, go above and beyond cost reduction or avoidance. Consider capturing additional data to measure efficiency improvements and business value.

Implementation steps

- **Executing cost optimization best practices:** For example, resource lifecycle management reduces infrastructure and operational costs and creates time and unexpected budget for experimentation. This increases organization agility and uncovers new opportunities for revenue generation.
- **Implementing automation:** For example, Auto Scaling, which ensures elasticity at minimal effort, and increases staff productivity by eliminating manual capacity planning work. For more details on operational resiliency, refer to the Well-Architected Reliability Pillar whitepaper.
- **Forecasting future AWS costs:** Forecasting helps finance stakeholders to set expectations with other internal and external organization stakeholders, and helps improve your organization's financial predictability. AWS Cost Explorer can be used to perform forecasting for your cost and usage.

Resources

Related documents:

- AWS Blog
- AWS Cost Management
- AWS News Blog
- Well-Architected Reliability Pillar whitepaper
- AWS Cost Explorer

Expenditure and usage awareness

Questions

- COST 2. How do you govern usage?
- COST 3. How do you monitor usage and cost?
- COST 4. How do you decommission resources?

COST 2. How do you govern usage?

Establish policies and mechanisms to verify that appropriate costs are incurred while objectives are achieved. By employing a checks-and-balances approach, you can innovate without overspending.

Best practices

- COST02-BP01 Develop policies based on your organization requirements
- COST02-BP02 Implement goals and targets
- COST02-BP03 Implement an account structure
- COST02-BP04 Implement groups and roles
- COST02-BP05 Implement cost controls
- COST02-BP06 Track project lifecycle

COST02-BP01 Develop policies based on your organization requirements

This best practice was updated with new guidance on July 13th, 2023.

Develop policies that define how resources are managed by your organization and inspect them periodically. Policies should cover the cost aspects of resources and workloads, including creation, modification, and decommissioning over a resource's lifetime.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Understanding your organization's costs and drivers is critical for managing your cost and usage effectively and identifying cost reduction opportunities. Organizations typically operate multiple workloads run by multiple teams. These teams can be in different organization units, each with its own revenue stream. The capability to attribute resource costs to the workloads, individual organization, or product owners drives efficient usage behaviour and helps reduce waste. Accurate cost and usage monitoring helps you understand how optimized a workload is, as well as how profitable organization units and products are. This knowledge allows for more informed decision making about where to allocate resources within your organization. Awareness of usage at all levels in the organization is key to driving change, as change in usage drives changes in cost. Consider taking a multi-faceted approach to becoming aware of your usage and expenditures.

The first step in performing governance is to use your organization's requirements to develop policies for your cloud usage. These policies define how your organization uses the cloud and how resources are managed. Policies should cover all aspects of resources and workloads that relate to cost or usage, including creation, modification, and decommissioning over a resource's lifetime. Verify that policies and procedures are followed and implemented for any change in a cloud environment. During your IT change management meetings, raise questions to find out the cost impact of planned changes, whether increasing or decreasing, the business justification, and the expected outcome.

Policies should be simple so that they are easily understood and can be implemented effectively throughout the organization. Policies also need to be easy to follow and interpret (so they are used) and specific (no misinterpretation between teams). Moreover, they need to be inspected periodically (like our mechanisms) and updated as customers business conditions or priorities change, which would make the policy outdated.

Start with broad, high-level policies, such as which geographic Region to use or times of the day that resources should be running. Gradually refine the policies for the various organizational units and workloads. Common policies include which services and features can be used (for example, lower performance storage in test and development environments), which types of resources can be used by different groups (for example, the largest size of resource in a development account is medium) and how long these resources will be in use (whether temporary, short term, or for a specific period of time).

Policy example

The following is a sample policy you can review to create your own cloud governance policies, which focus on cost optimization. Make sure you adjust policy based on your organization's requirements and your stakeholders' requests.

- **Policy name:** Define a clear policy name, such as Resource Optimization and Cost Reduction Policy.
- **Purpose:** Explain why this policy should be used and what is the expected outcome. The objective of this policy is to verify that there is a minimum cost required to deploy and run the desired workload to meet business requirements.
- **Scope:** Clearly define who should use this policy and when it should be used, such as DevOps X Team to use this policy in us-east customers for X environment (production or non-production).

Policy statement

1. Select us-east-1or multiple us-east regions based on your workload's environment and business requirement (development, user acceptance testing, pre-production, or production).
2. Schedule Amazon EC2 and Amazon RDS instances to run between six in the morning and eight at night (Eastern Standard Time (EST)).
3. Stop all unused Amazon EC2 instances after eight hours and unused Amazon RDS instances after 24 hours of inactivity.
4. Terminate all unused Amazon EC2 instances after 24 hours of inactivity in non-production environments. Remind Amazon EC2 instance owner (based on tags) to review their stopped Amazon EC2 instances in production and inform them that their Amazon EC2 instances will be terminated within 72 hours if they are not in use.
5. Use generic instance family and size such as m5.large and then resize the instance based on CPU and memory utilization using AWS Compute Optimizer.
6. Prioritize using auto scaling to dynamically adjust the number of running instances based on traffic.
7. Use spot instances for non-critical workloads.
8. Review capacity requirements to commit saving plans or reserved instances for predictable workloads and inform Cloud Financial Management Team.
9. Use Amazon S3 lifecycle policies to move infrequently accessed data to cheaper storage tiers. If no retention policy defined, use Amazon S3 Intelligent Tiering to move objects to archived tier automatically.
10. Monitor resource utilization and set alarms to trigger scaling events using Amazon CloudWatch.
11. For each AWS account, use AWS Budgets to set cost and usage budgets for your account based on cost center and business units.
12. Using AWS Budgets to set cost and usage budgets for your account can help you stay on top of your spending and avoid unexpected bills, allowing you to better control your costs.

Procedure: Provide detailed procedures for implementing this policy or refer to other documents that describe how to implement each policy statement. This section should provide step-by-step instructions for carrying out the policy requirements.

To implement this policy, you can use various third-party tools or AWS Config rules to check for compliance with the policy statement and trigger automated remediation actions using AWS Lambda functions. You can also use AWS Organizations to enforce the policy. Additionally, you

should regularly review your resource usage and adjust the policy as necessary to verify that it continues to meet your business needs.

Implementation steps

- **Meet with stakeholders:** To develop policies, ask stakeholders (cloud business office, engineers, or functional decision makers for policy enforcement) within your organization to specify their requirements and document them. Take an iterative approach by starting broadly and continually refine down to the smallest units at each step. Team members include those with direct interest in the workload, such as organization units or application owners, as well as supporting groups, such as security and finance teams.
- **Get confirmation:** Make sure teams agree on policies who can access and deploy to the AWS Cloud. Make sure they follow your organization's policies and confirm that their resource creations align with the agreed policies and procedures.
- **Create onboarding training sessions:** Ask new organization members to complete onboarding training courses to create cost awareness and organization requirements. They may assume different policies from their previous experience or not think of them at all.
- **Define locations for your workload:** Define where your workload operates, including the country and the area within the country. This information is used for mapping to AWS Regions and Availability Zones.
- **Define and group services and resources:** Define the services that the workloads require. For each service, specify the types, the size, and the number of resources required. Define groups for the resources by function, such as application servers or database storage. Resources can belong to multiple groups.
- **Define and group the users by function:** Define the users that interact with the workload, focusing on what they do and how they use the workload, not on who they are or their position in the organization. Group similar users or functions together. You can use the AWS managed policies as a guide.
- **Define the actions:** Using the locations, resources, and users identified previously, define the actions that are required by each to achieve the workload outcomes over its life time (development, operation, and decommission). Identify the actions based on the groups, not the individual elements in the groups, in each location. Start broadly with read or write, then refine down to specific actions to each service.
- **Define the review period:** Workloads and organizational requirements can change over time. Define the workload review schedule to ensure it remains aligned with organizational priorities.

- **Document the policies:** Verify the policies that have been defined are accessible as required by your organization. These policies are used to implement, maintain, and audit access of your environments.

Resources

Related documents:

- Change Management in the Cloud
- AWS Managed Policies for Job Functions
- AWS multiple account billing strategy
- Actions, Resources, and Condition Keys for AWS Services
- AWS Management and Governance
- Control access to AWS Regions using IAM policies
- Global Infrastructures Regions and AZs

Related videos:

- AWS Management and Governance at Scale

Related examples:

- VMware - What Are Cloud Policies?

COST02-BP02 Implement goals and targets

This best practice was updated with new guidance on July 13th, 2023.

Implement both cost and usage goals and targets for your workload. Goals provide direction to your organization on expected outcomes, and targets provide specific measurable outcomes to be achieved for your workloads.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Develop cost and usage goals and targets for your organization. As a growing organization on AWS, it is important to set and track goals for cost optimization. These goals or key performance indicators (KPIs) can include things like percent of spend on-demand, or adoption of certain optimized services such as AWS Graviton instances or gp3 EBS volume types. Setting measurable and achievable goals can help you to continue to measure efficiency improvements which is important to ongoing business operations. Goals provide guidance and direction to your organization on expected outcomes. Targets provide specific measurable outcomes to be achieved. In short, a goal is the direction you want to go and target is how far in that direction and when that goal should be achieved (using guidance of specific, measurable, assignable, realistic, and timely, or SMART). An example of a goal is that platform usage should increase significantly, with only a minor (non-linear) increase in cost. An example target is a 20% increase in platform usage, with less than a five percent increase in costs. Another common goal is that workloads need to be more efficient every six months. The accompanying target would be that the cost per business metrics needs to decrease by five percent every six months.

A goal for cost optimization is to increase workload efficiency, which means decreasing the cost per business outcome of the workload over time. It is recommended to implement this goal for all workloads, and also set a target such as a five percent increase in efficiency every six months to a year. This can be achieved in the cloud through building capability in cost optimization, and releasing new services and features.

It's important to have near real-time visibility over your KPIs and related savings opportunities and track your progress over time. To get started with defining and tracking KPI goals, we recommend the KPI dashboard from the Cloud Intelligence Dashboards (CID) framework. Based on the data from AWS Cost and Usage Report, the KPI dashboard provides a series of recommended cost optimization KPIs with the ability to set custom goals and track progress over time.

If you have another solution that allows you to set and track KPI goals, make sure it's adopted by all cloud financial management stakeholders in your organization.

Implementation steps

- **Define expected usage levels:** To begin, focus on usage levels. Engage with the application owners, marketing, and greater business teams to understand what the expected usage levels will be for the workload. How will customer demand change over time, and will there be any changes due to seasonal increases or marketing campaigns?

- **Define workload resourcing and costs:** With usage levels defined, quantify the changes in workload resources required to meet these usage levels. You may need to increase the size or number of resources for a workload component, increase data transfer, or change workload components to a different service at a specific level. Specify what the costs will be at each of these major points, and what the changes in cost will be when there are changes in usage.
- **Define business goals:** Taking the output from the expected changes in usage and cost, combine this with expected changes in technology, or any programs that you are running, and develop goals for the workload. Goals must address usage, cost and the relationship between the two. Goals must be simple, high level, and help people understand what the business expects in terms of outcomes (such as making sure unused resources are kept below certain cost level). You don't need to define goals for each unused resource type or define costs that cause losses for goals and targets. Verify that there are organizational programs (for example, capability building like training and education) if there are expected changes in cost without changes in usage.
- **Define targets:** For each of the defined goals specify a measurable target. If the goal is to increase efficiency in the workload, the target will quantify the amount of improvement (typically in business outputs for each dollar spent) and when it will be delivered. For example, if you set a goal of minimizing waste due to over-provisioning, then your target can be waste due to compute over-provisioning in the first tier of production workloads should not exceed 10% of tier compute cost, and waste due to compute over-provisioning in the second tier of production workloads should not exceed 5% of tier compute cost.

Resources

Related documents:

- AWS managed policies for job functions
- AWS multi-account strategy for your AWS Control Tower landing zone
- Control access to AWS Regions using IAM policies
- SMART Goals

Related videos:

- Well-Architected Labs: Goals and Targets (Level 100)

Related examples:

- Well-Architected Labs: Decommission resources (Goals and Targets)
- Well-Architected Labs: Resource Type, Size, and Number (Goals and Targets)

COST02-BP03 Implement an account structure

Implement a structure of accounts that maps to your organization. This assists in allocating and managing costs throughout your organization.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS Organizations allows you to create multiple AWS accounts which can help you centrally govern your environment as you scale your workloads on AWS. You can model your organizational hierarchy by grouping AWS accounts in organizational unit (OU) structure and creating multiple AWS accounts under each OU. To create an account structure, you need to decide which of your AWS accounts will be the management account first. After that, you can create new AWS accounts or select existing accounts as member accounts based on your designed account structure by following management account best practices and member account best practices.

It is advised to always have at least one management account with one member account linked to it, regardless of your organization size or usage. All workload resources should reside only within member accounts and no resource should be created in management account. There is no one size fits all answer for how many AWS accounts you should have. Assess your current and future operational and cost models to ensure that the structure of your AWS accounts reflects your organization's goals. Some companies create multiple AWS accounts for business reasons, for example:

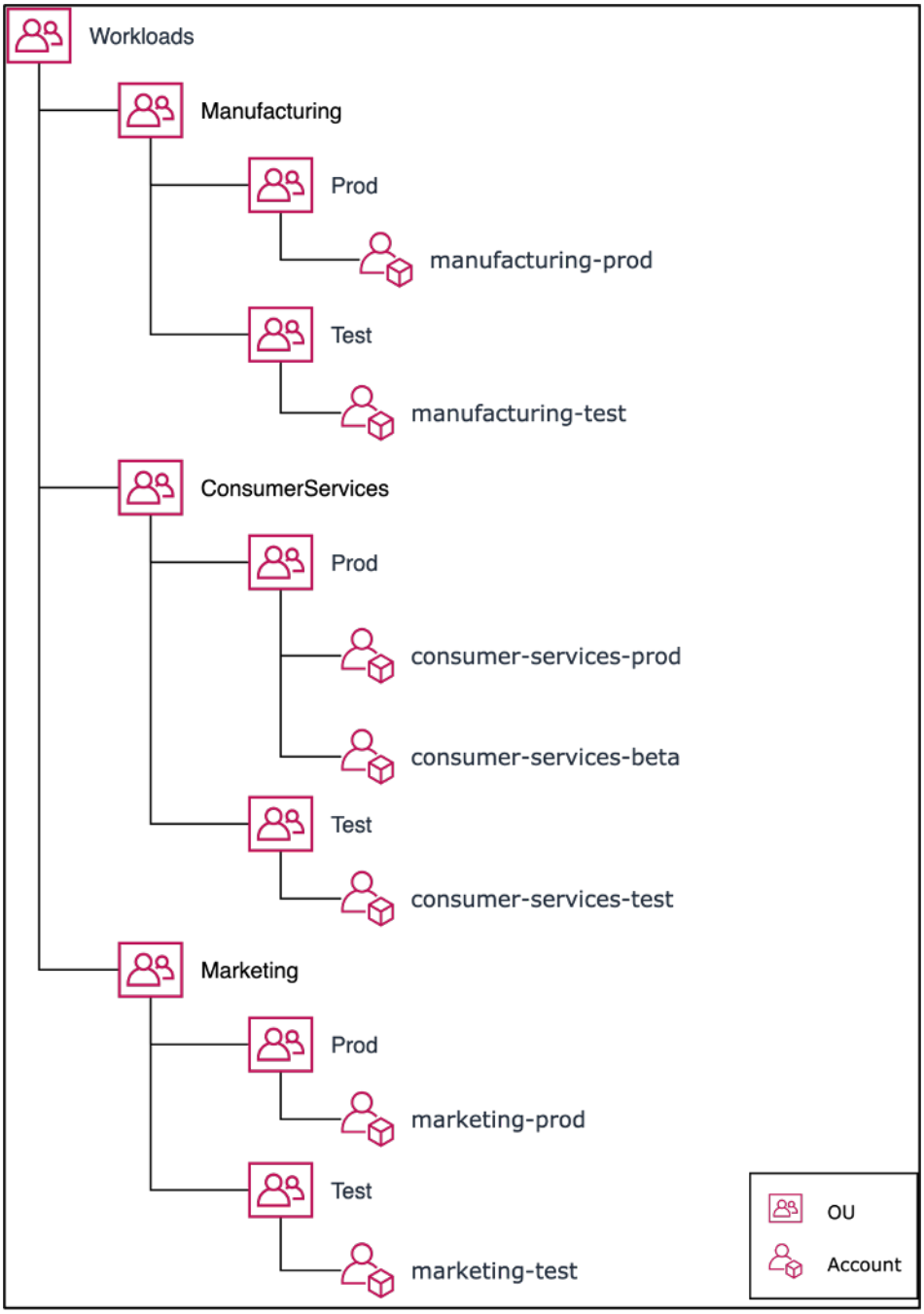
- Administrative or fiscal and billing isolation is required between organization units, cost centers, or specific workloads.
- AWS service limits are set to be specific to particular workloads.
- There is a requirement for isolation and separation between workloads and resources.

Within AWS Organizations, consolidated billing creates the construct between one or more member accounts and the management account. Member accounts allow you to isolate and distinguish your cost and usage by groups. A common practice is to have separate member accounts for each organization unit (such as finance, marketing, and sales), or for each environment

lifecycle (such as development, testing and production), or for each workload (workload a, b, and c), and then aggregate these linked accounts using consolidated billing.

Consolidated billing allows you to consolidate payment for multiple member AWS accounts under a single management account, while still providing visibility for each linked account's activity. As costs and usage are aggregated in the management account, this allows you to maximize your service volume discounts, and maximize the use of your commitment discounts (Savings Plans and Reserved Instances) to achieve the highest discounts.

The following diagram shows how you can use AWS Organizations with organizational units (OU) to group multiple accounts, and place multiple AWS accounts under each OU. It is recommended to use OUs for various use cases and workloads which provides patterns for organizing accounts.



Example of grouping multiple AWS accounts under organizational units.

AWS Control Tower can quickly set up and configure multiple AWS accounts, ensuring that governance is aligned with your organization's requirements.

Implementation steps

- **Define separation requirements:** Requirements for separation are a combination of multiple factors, including security, reliability, and financial constructs. Work through each factor in order

and specify whether the workload or workload environment should be separate from other workloads. Security promotes adhesion to access and data requirements. Reliability manages limits so that environments and workloads do not impact others. Review the security and reliability pillars of the Well-Architected Framework periodically and follow the provided best practices. Financial constructs create strict financial separation (different cost center, workload ownerships and accountability). Common examples of separation are production and test workloads being run in separate accounts, or using a separate account so that the invoice and billing data can be provided to the individual business units or departments in the organization or stakeholder who owns the account.

- **Define grouping requirements:** Requirements for grouping do not override the separation requirements, but are used to assist management. Group together similar environments or workloads that do not require separation. An example of this is grouping multiple test or development environments from one or more workloads together.
- **Define account structure:** Using these separations and groupings, specify an account for each group and maintain separation requirements. These accounts are your member or linked accounts. By grouping these member accounts under a single management or payer account, you combine usage, which allows for greater volume discounts across all accounts, which provides a single bill for all accounts. It's possible to separate billing data and provide each member account with an individual view of their billing data. If a member account must not have its usage or billing data visible to any other account, or if a separate bill from AWS is required, define multiple management or payer accounts. In this case, each member account has its own management or payer account. Resources should always be placed in member or linked accounts. The management or payer accounts should only be used for management.

Resources

Related documents:

- Using Cost Allocation Tags
- AWS managed policies for job functions
- AWS multiple account billing strategy
- Control access to AWS Regions using IAM policies
- AWS Control Tower
- AWS Organizations
- Best practices for management accounts and member accounts

- Organizing Your AWS Environment Using Multiple Accounts
- Turning on shared reserved instances and Savings Plans discounts
- Consolidated billing
- Consolidated billing

Related examples:

- Splitting the CUR and Sharing Access

Related videos:

- Introducing AWS Organizations
- Set Up a Multi-Account AWS Environment that Uses Best Practices for AWS Organizations

Related examples:

- Well-Architected Labs: Create an AWS Organization (Level 100)
- Splitting the AWS Cost and Usage Report and Sharing Access
- Defining an AWS Multi-Account Strategy for telecommunications companies
- Best Practices for Optimizing AWS accounts
- Best Practices for Organizational Units with AWS Organizations

COST02-BP04 Implement groups and roles

Implement groups and roles that align to your policies and control who can create, modify, or decommission instances and resources in each group. For example, implement development, test, and production groups. This applies to AWS services and third-party solutions.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

After you develop policies, you can create logical groups and roles of users within your organization. This allows you to assign permissions and control usage. Begin with high-level groupings of people. Typically this aligns with organizational units and job roles (for example, systems administrator in the IT Department, or financial controller). The groups join people that do

similar tasks and need similar access. Roles define what a group must do. For example, a systems administrator in IT requires access to create all resources, but an analytics team member only needs to create analytics resources.

Implementation steps

- **Implement groups:** Using the groups of users defined in your organizational policies, implement the corresponding groups, if necessary. Refer to the security pillar for best practices on users, groups, and authentication.
- **Implement roles and policies:** Using the actions defined in your organizational policies, create the required roles and access policies. Refer to the security pillar for best practices on roles and policies.

Resources

Related documents:

- AWS managed policies for job functions
- AWS multiple account billing strategy
- Control access to AWS Regions using IAM policies
- Well-Architected Security Pillar

Related examples:

- Well-Architected Lab Basic Identity and Access

COST02-BP05 Implement cost controls

Implement controls based on organization policies and defined groups and roles. These certify that costs are only incurred as defined by organization requirements such as control access to regions or resource types.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

A common first step in implementing cost controls is to set up notifications when cost or usage events occur outside of policies. You can act quickly and verify if corrective action is required

without restricting or negatively impacting workloads or new activity. After you know the workload and environment limits, you can enforce governance. AWS Budgets allows you to set notifications and define monthly budgets for your AWS costs, usage, and commitment discounts (Savings Plans and Reserved Instances). You can create budgets at an aggregate cost level (for example, all costs), or at a more granular level where you include only specific dimensions such as linked accounts, services, tags, or Availability Zones.

Once you set up your budget limits with AWS Budgets, use AWS Cost Anomaly Detection to reduce your unexpected cost. AWS Cost Anomaly Detection is a cost management services that uses machine learning to continually monitor your cost and usage to detect unusual spends. It helps you identify anomalous spend and root causes, so you can quickly take action. First, create a cost monitor in AWS Cost Anomaly Detection, then choose your alerting preference by setting up a dollar threshold (such as an alert on anomalies with impact greater than $1,000). Once you receive alerts, you can analyze the root cause behind the anomaly and impact on your costs. You can also monitor and perform your own anomaly analysis in AWS Cost Explorer.

Enforce governance policies in AWS through AWS Identity and Access Management and AWS Organizations Service Control Policies (SCP). IAM allows you to securely manage access to AWS services and resources. Using IAM, you can control who can create or manage AWS resources, the type of resources that can be created, and where they can be created. This minimizes the possibility of resources being created outside of the defined policy. Use the roles and groups created previously and assign IAM policies to enforce the correct usage. SCP offers central control over the maximum available permissions for all accounts in your organization, keeping your accounts stay within your access control guidelines. SCPs are available only in an organization that has all features turned on, and you can configure the SCPs to either deny or allow actions for member accounts by default. For more details on implementing access management, see the Well-Architected Security Pillar whitepaper.

Governance can also be implemented through management of AWS service quotas. By ensuring service quotas are set with minimum overhead and accurately maintained, you can minimize resource creation outside of your organization's requirements. To achieve this, you must understand how quickly your requirements can change, understand projects in progress (both creation and decommission of resources), and factor in how fast quota changes can be implemented. Service quotas can be used to increase your quotas when required.

Implementation steps

- **Implement notifications on spend:** Using your defined organization policies, create AWS Budgets to notify you when spending is outside of your policies. Configure multiple cost budgets,

one for each account, which notify you about overall account spending. Configure additional cost budgets within each account for smaller units within the account. These units vary depending on your account structure. Some common examples are AWS Regions, workloads (using tags), or AWS services. Configure an email distribution list as the recipient for notifications, and not an individual's email account. You can configure an actual budget for when an amount is exceeded, or use a forecasted budget for notifying on forecasted usage. You can also preconfigure AWS Budget Actions that can enforce specific IAM or SCP policies, or stop target Amazon EC2 or Amazon RDS instances. Budget Actions can be started automatically or require workflow approval.

- **Implement notifications on anomalous spend:** Use AWS Cost Anomaly Detection to reduce your surprise costs in your organization and analyze root cause of potential anomalous spend. Once you create cost monitor to identify unusual spend at your specified granularity and configure notifications in AWS Cost Anomaly Detection, it sends you alert when unusual spend is detected. This will allow you to analyze root case behind the anomaly and understand the impact on your cost. Use AWS Cost Categories while configuring AWS Cost Anomaly Detection to identify which project team or business unit team can analyze the root cause of the unexpected cost and take timely necessary actions.
- **Implement controls on usage:** Using your defined organization policies, implement IAM policies and roles to specify which actions users can perform and which actions they cannot. Multiple organizational policies may be included in an AWS policy. In the same way that you defined policies, start broadly and then apply more granular controls at each step. Service limits are also an effective control on usage. Implement the correct service limits on all your accounts.

Resources

Related documents:

- AWS managed policies for job functions
- AWS multiple account billing strategy
- Control access to AWS Regions using IAM policies
- AWS Budgets
- AWS Cost Anomaly Detection
- Control Your AWS Costs

Related videos:

- How can I use AWS Budgets to track my spending and usage

Related examples:

- Example IAM access management policies
- Example service control policies
- AWS Budgets Actions
- Create IAM Policy to control access to Amazon EC2 resources using Tags
- Restrict the access of IAM Identity to specific Amazon EC2 resources
- Create an IAM Policy to restrict Amazon EC2 usage by family
- Well-Architected Labs: Cost and Usage Governance (Level 100)
- Well-Architected Labs: Cost and Usage Governance (Level 200)
- Slack integrations for Cost Anomaly Detection using Amazon Q Developer in chat applications

COST02-BP06 Track project lifecycle

Track, measure, and audit the lifecycle of projects, teams, and environments to avoid using and paying for unnecessary resources.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Ensure that you track the entire lifecycle of the workload. This ensures that when workloads or workload components are no longer required, they can be decommissioned or modified. This is especially useful when you release new services or features. The existing workloads and components may appear to be in use, but should be decommissioned to redirect customers to the new service. Notice previous stages of workloads — after a workload is in production, previous environments can be decommissioned or greatly reduced in capacity until they are required again.

AWS provides a number of management and governance services you can use for entity lifecycle tracking. You can use AWS Config or AWS Systems Manager to provide a detailed inventory of your AWS resources and configuration. It is recommended that you integrate with your existing project or asset management systems to keep track of active projects and products within your organization. Combining your current system with the rich set of events and metrics provided by AWS allows you to build a view of significant lifecycle events and proactively manage resources to reduce unnecessary costs.

Refer to the Well-Architected Operational Excellence Pillar whitepaper for more details on implementing entity lifecycle tracking.

Implementation steps

- **Perform workload reviews:** As defined by your organizational policies, audit your existing projects. The amount of effort spent in the audit should be proportional to the approximate risk, value, or cost to the organization. Key areas to include in the audit would be risk to the organization of an incident or outage, value, or contribution to the organization (measured in revenue or brand reputation), cost of the workload (measured as total cost of resources and operational costs), and usage of the workload (measured in number of organization outcomes per unit of time). If these areas change over the lifecycle, adjustments to the workload are required, such as full or partial decommissioning.

Resources

Related documents:

- AWS Config
- AWS Systems Manager
- AWS managed policies for job functions
- AWS multiple account billing strategy
- Control access to AWS Regions using IAM policies

COST 3. How do you monitor usage and cost?

Establish policies and procedures to monitor and appropriately allocate your costs. This permits you to measure and improve the cost efficiency of this workload.

Best practices

- COST03-BP01 Configure detailed information sources
- COST03-BP02 Add organization information to cost and usage
- COST03-BP03 Identify cost attribution categories
- COST03-BP04 Establish organization metrics
- COST03-BP05 Configure billing and cost management tools

- COST03-BP06 Allocate costs based on workload metrics

COST03-BP01 Configure detailed information sources

Configure the AWS Cost and Usage Report, and Cost Explorer hourly granularity, to provide detailed cost and usage information. Configure your workload to have log entries for every delivered business outcome.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Enable hourly granularity in AWS Cost Explorer and create a AWS Cost and Usage Report (CUR). These data sources provide the most accurate view of cost and usage across your entire organization. The CUR provides daily or hourly usage granularity, rates, costs, and usage attributes for all chargeable AWS services. All possible dimensions are in the CUR including: tagging, location, resource attributes, and account IDs.

Configure your CUR with the following customizations:

- Include resource IDs
- Automatically refresh the CUR
- Hourly granularity
- **Versioning:** Overwrite existing report
- **Data integration:** Amazon Athena (Parquet format and compression)

Use AWS Glue to prepare the data for analysis, and use Amazon Athena to perform data analysis, using SQL to query the data. You can also use Amazon Quick to build custom and complex visualizations and distribute them throughout your organization.

Implementation steps

- **Configure the cost and usage report:** Using the billing console, configure at least one cost and usage report. Configure a report with hourly granularity that includes all identifiers and resource IDs. You can also create other reports with different granularities to provide higher-level summary information.
- **Configure hourly granularity in Cost Explorer:** Using the billing console, turn on Hourly and Resource Level Data.

> **Note**
>
> There will be associated costs with activating this feature. For details, refer to the pricing.

- **Configure application logging:** Verify that your application logs each business outcome that it delivers so it can be tracked and measured. Ensure that the granularity of this data is at least hourly so it matches with the cost and usage data. Refer to the Well-Architected Operational Excellence Pillar for more detail on logging and monitoring.

Resources

Related documents:

- AWS Account Setup
- AWS Cost and Usage Report (CUR)
- AWS Glue
- Amazon Quick
- AWS Cost Management Pricing
- Tagging AWS resources
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports
- Well-Architected Operational Excellence Pillar

Related examples:

- AWS Account Setup

COST03-BP02 Add organization information to cost and usage

Define a tagging schema based on your organization, workload attributes, and cost allocation categories so that you can filter and search for resources or monitor cost and usage in cost management tools. Implement consistent tagging across all resources where possible by purpose, team, environment, or other criteria relevant to your business.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implement tagging in AWS to add organization information to your resources, which will then be added to your cost and usage information. A tag is a key-value pair — the key is defined and must be unique across your organization, and the value is unique to a group of resources. An example of a key-value pair is the key is `Environment`, with a value of `Production`. All resources in the production environment will have this key-value pair. Tagging allows you categorize and track your costs with meaningful, relevant organization information. You can apply tags that represent organization categories (such as cost centers, application names, projects, or owners), and identify workloads and characteristics of workloads (such as test or production) to attribute your costs and usage throughout your organization.

When you apply tags to your AWS resources (such as Amazon Elastic Compute Cloud instances or Amazon Simple Storage Service buckets) and activate the tags, AWS adds this information to your Cost and Usage Reports. You can run reports and perform analysis on tagged and untagged resources to allow greater compliance with internal cost management policies and ensure accurate attribution.

Creating and implementing an AWS tagging standard across your organization's accounts helps you manage and govern your AWS environments in a consistent and uniform manner. Use Tag Policies in AWS Organizations to define rules for how tags can be used on AWS resources in your accounts in AWS Organizations. Tag Policies allow you to easily adopt a standardized approach for tagging AWS resources

AWS Tag Editor allows you to add, delete, and manage tags of multiple resources. With Tag Editor, you search for the resources that you want to tag, and then manage tags for the resources in your search results.

AWS Cost Categories allows you to assign organization meaning to your costs, without requiring tags on resources. You can map your cost and usage information to unique internal organization structures. You define category rules to map and categorize costs using billing dimensions, such as accounts and tags. This provides another level of management capability in addition to tagging. You can also map specific accounts and tags to multiple projects.

Implementation steps

- **Define a tagging schema:** Gather all stakeholders from across your business to define a schema. This typically includes people in technical, financial, and management roles. Define a list of tags

that all resources must have, as well as a list of tags that resources should have. Verify that the tag names and values are consistent across your organization.

- **Tag resources:** Using your defined cost attribution categories, place tags on all resources in your workloads according to the categories. Use tools such as the CLI, Tag Editor, or AWS Systems Manager to increase efficiency.
- **Implement AWS Cost Categories:** You can create Cost Categories without implementing tagging. Cost categories use the existing cost and usage dimensions. Create category rules from your schema and implement them into cost categories.
- **Automate tagging:** To verify that you maintain high levels of tagging across all resources, automate tagging so that resources are automatically tagged when they are created. Use services such as AWS CloudFormation to verify that resources are tagged when created. You can also create a custom solution to tag automatically using Lambda functions or use a microservice that scans the workload periodically and removes any resources that are not tagged, which is ideal for test and development environments.
- **Monitor and report on tagging:** To verify that you maintain high levels of tagging across your organization, report and monitor the tags across your workloads. You can use AWS Cost Explorer to view the cost of tagged and untagged resources, or use services such as Tag Editor. Regularly review the number of untagged resources and take action to add tags until you reach the desired level of tagging.

Resources

Related documents:

- Tagging Best Practices
- AWS CloudFormation Resource Tag
- AWS Cost Categories
- Tagging AWS resources
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports

Related videos:

- How can I tag my AWS resources to divide up my bill by cost center or project

- Tagging AWS Resources

Related examples:

- Automatically tag new AWS resources based on identity or role

COST03-BP03 Identify cost attribution categories

This best practice was updated with new guidance on July 13th, 2023.

Identify organization categories such as business units, departments, or projects that could be used to allocate cost within your organization to the internal consuming entities so that spend accountability can be enforced and consumption behaviors can be driven effectively.

Level of risk exposed if this best practice is not established: High

Implementation guidance

The process of categorizing costs is crucial in budgeting, accounting, financial reporting, decision making, benchmarking, and project management. By classifying and categorizing expenses, teams can gain a better understanding of the types of costs they will incur throughout their cloud journey, helping teams make informed decisions and manage budgets effectively.

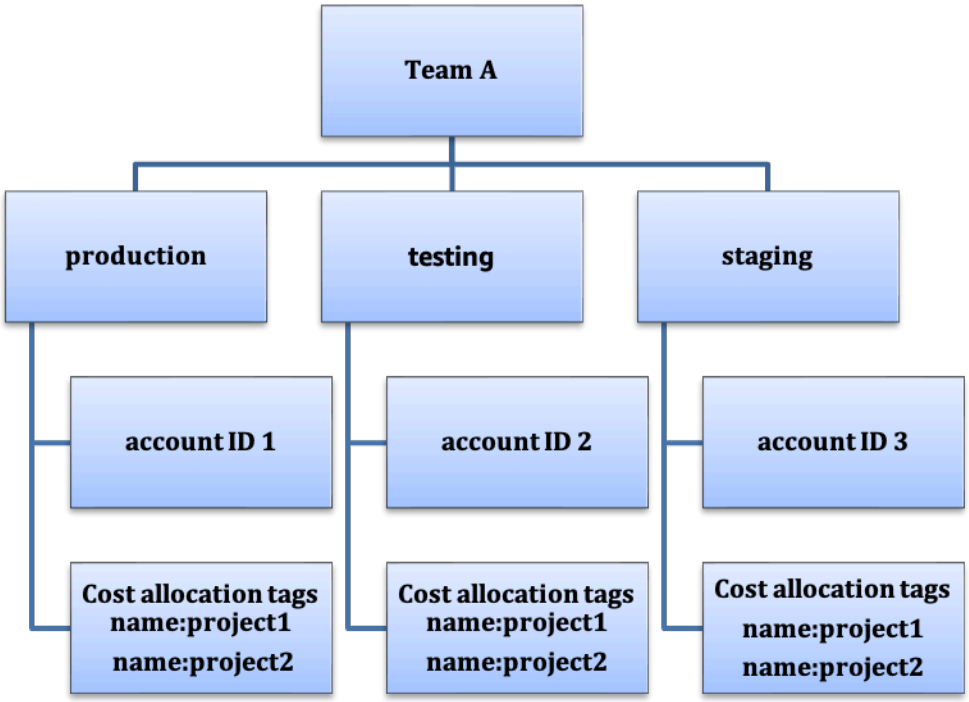
Cloud spend accountability establishes a strong incentive for disciplined demand and cost management. The result is significantly greater cloud cost savings for organizations that allocate most of their cloud spend to consuming business units or teams.

Work with your finance team and other relevant stakeholders to understand the requirements of how costs must be allocated within your organization. Workload costs must be allocated throughout the entire lifecycle, including development, testing, production, and decommissioning. Understand how the costs incurred for learning, staff development, and idea creation are attributed in the organization. This can be helpful to correctly allocate accounts used for this purpose to training and development budgets, instead of generic IT cost budgets.

After defining your cost attribution categories with your stakeholders in your organization, use AWS Cost Categories to group your cost and usage information into meaningful categories in the AWS Cloud such as cost for specific project or AWS accounts for departments or business units. You can create custom categories and map your cost and usage information into these categories

based on rules you define using various dimensions such as account, tag, service, charge type, and even other cost categories. Once cost categories are set up, you will be able to view your cost and usage information by these categories allowing your organization to make better strategic and purchasing decisions. These categories will be visible in AWS Cost Explorer, AWS Budgets, and AWS Cost and Usage Report as well.

As an example, the following diagram shows you how can you group your costs and usage information in your organization such as having multiple teams (cost category) having multiple environments (rules) and each environment having multiple resources or assets (dimensions).



Cost and usage organization chart.

Implementation steps

- **Define your organization categories:** Meet with stakeholders to define categories that reflect your organization's structure and requirements. These will directly map to the structure of existing financial categories, such as business unit, budget, cost center, or department. Look at the outcomes the cloud delivers for your business, such as training or education, as these are also organization categories. Multiple categories can be assigned to a resource, and a resource can be in multiple different categories, so define as many categories as needed.
- **Define your functional categories:** Meet with stakeholders to define categories that reflect the functions that you have within your business. This may be the workload or application names,

and the type of environment, such as production, testing, or development. Multiple categories can be assigned to a resource, and a resource can be in multiple different categories, so define as many categories as needed so that you can manage your costs within the categorized structure using AWS Cost Categories.

- **Define AWS Cost Categories:** You can create cost categories to organize your cost and usage information. Use AWS Cost Categories to map your AWS costs and usage into meaningful categories. With cost categories, you can organize your costs using a rule-based engine. The rules that you configure organize your costs into categories. Within these rules, you can filter with using multiple dimensions for each category such as specific AWS accounts, specific AWS services, or specific charge types. You can then use these categories across multiple products in the AWS Billing and Cost Management console. This includes AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report, and AWS Cost Anomaly Detection. You can create groupings of costs using cost categories as well. After you create the cost categories (allowing up to 24 hours after creating a cost category for your usage records to be updated with values), they appear in AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report, and AWS Cost Anomaly Detection. For example, create cost categories for your business units (DevOps Team), and under each category create multiple rules (rules for each sub category) with multiple dimensions (AWS accounts, cost allocation tags, services or charge type) based on your defined groupings. In AWS Cost Explorer and AWS Budgets, a cost category appears as an additional billing dimension. You can use this to filter for the specific cost category value, or group by the cost category.

Resources

Related documents:

- Tagging AWS resources
- Using Cost Allocation Tags
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports
- AWS Cost Categories
- Managing your costs with AWS Cost Categories
- Creating cost categories
- Tagging cost categories
- Splitting charges within cost categories

- AWS Cost Categories Features

Related examples:

- Organize your cost and usage data with AWS Cost Categories
- Managing your costs with AWS Cost Categories

COST03-BP04 Establish organization metrics

Establish the organization metrics that are required for this workload. Example metrics of a workload are customer reports produced, or web pages served to customers.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Understand how your workload's output is measured against business success. Each workload typically has a small set of major outputs that indicate performance. If you have a complex workload with many components, then you can prioritize the list, or define and track metrics for each component. Work with your teams to understand which metrics to use. This unit will be used to understand the efficiency of the workload, or the cost for each business output.

Implementation steps

- **Define workload outcomes:** Meet with the stakeholders in the business and define the outcomes for the workload. These are a primary measure of customer usage and must be business metrics and not technical metrics. There should be a small number of high-level metrics (less than five) per workload. If the workload produces multiple outcomes for different use cases, then group them into a single metric.
- **Define workload component outcomes:** Optionally, if you have a large and complex workload, or can easily break your workload into components (such as microservices) with well-defined inputs and outputs, define metrics for each component. The effort should reflect the value and cost of the component. Start with the largest components and work towards the smaller components.

Resources

Related documents:

- Tagging AWS resources
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports

COST03-BP05 Configure billing and cost management tools

This best practice was updated with new guidance on July 13th, 2023.

Configure cost management tools in line with your organization policies to manage and optimize cloud spend. This includes services, tools, and resources to organize and track cost and usage data, enhance control through consolidated billing and access permission, improve planning through budgeting and forecasts, receive notifications or alerts, and further lower cost with resources and pricing optimizations.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To establish strong accountability, your account strategy should be considered first as part of your cost allocation strategy. Get this right, and you may not need to go any further. Otherwise, there will be unawareness and further pain points.

To encourage accountability of cloud spend, users should have access to tools that provide visibility into their costs and usage. It is recommended that all workloads and teams have the tools configured for the following details and purposes:

- **Organize:** Establish your cost allocation and governance baseline with your own tagging strategy and taxonomy. Tag supported AWS resources and categorize them meaningfully based on your organization structure (business units, departments, or projects). Tag account names for specific cost centers and map them with AWS Cost Categories to group accounts for particular business units for their cost centers so that business unit owner can see multiple accounts' consumption in one place.
- **Access:** Track organization-wide billing information in consolidated billing and verify the right stakeholders and business owners have access.

- **Control:** Build effective governance mechanisms with the right guardrails to prevent unexpected scenarios when using SCP, tag policies, and budget alerts For example, with effective control mechanism, you can prevent teams from creating resources in unsupported Regions.
- **Current State:** Configure a dashboard showing current levels of cost and usage. The dashboard should be available in a highly visible place within the work environment similar to an operations dashboard. You can use Cloud Intelligence Dashboard (CID) or any other supported products to create this visibility.
- **Notifications:** Provide notifications when cost or usage is outside of defined limits and when anomalies occur with AWS Budgets or AWS Cost Anomaly Detection.
- **Reports:** Summarize all cost and usage information and raise awareness and accountability of your cloud spend with detailed, allocable cost data. Reports should be relevant to the team consuming them and ideally should contain recommendations.
- **Tracking:** Show the current cost and usage against configured goals or targets.
- **Analysis:** Allow team members to perform custom and deep analysis down to the hourly granularity, with all possible dimensions.
- **Inspect:** Stay up to date with your resource deployment and cost optimization opportunities. Get notifications (using Amazon CloudWatch, Amazon SNS, or Amazon SES) for resource deployments at organization level and review cost optimization recommendations (for example, AWS Compute Optimizer or AWS Trusted Advisor).
- **Trending:** Display the variability in cost and usage over the required period of time, with the required granularity.
- **Forecasts:** Show estimated future costs, estimate your resource usage, and spend with forecast dashboards that you create.

You can use AWS tools like AWS Cost Explorer, AWS Billing, or AWS Budgets for essentials, or you can integrate CUR data with Amazon Athena and Quick to provide this capability for more detailed views. If you don't have essential skills or bandwidth in your organization, you can work with AWS ProServ, AWS Managed Services (AMS), or AWS Partners and use their tools. You can also use third-party tools, but verify first that the cost provides value to your organization.

Implementation steps

- **Allow team-based access to tools:** Configure your accounts and create groups that have access to the required cost and usage reports for their consumptions and use AWS Identity and Access Management to control access to the tools such as AWS Cost Explorer. These groups must

include representatives from all teams that own or manage an application. This certifies that every team has access to their cost and usage information to track their consumption.

- **Configure AWS Budgets:** Configure AWS Budgets on all accounts for your workload. Set budgets for the overall account spend, and budgets for the workloads by using tags. Configure notifications in AWS Budgets to receive alerts for when you exceed your budgeted amounts, or when your estimated costs exceed your budgets.
- **Configure AWS Cost Explorer:** Configure AWS Cost Explorer for your workload and accounts to visualize your cost data for further analysis. Create a dashboard for the workload that tracks overall spend, key usage metrics for the workload, and forecast of future costs based on your historical cost data.
- **Configure AWS Cost Anomaly Detection:** Use AWS Cost Anomaly Detection for your accounts, core services, or Cost Categories you created to monitor your cost and usage and detect unusual spends. You can receive alerts individually in aggregated reports, and receive alerts in an email or an Amazon Simple Notification Service topic which allows you to analyze and determine the root cause of the anomaly, and identify the factor that is driving the cost increase.
- **Configure advanced tools:** Optionally, you can create custom tools for your organization that provide additional detail and granularity. You can implement advanced analysis capability using Amazon Athena, and dashboards using Quick. Consider using Cloud Intelligence Dashboards (CID) for pre-configured, advanced dashboards. There are also AWS Partners you can work with and adopt their cloud management solutions to activate cloud bill monitoring and optimization in one convenient location.

Resources

Related documents:

- AWS Cost Management
- Tagging AWS resources
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports
- AWS Cost Categories
- Cloud Financial Management with AWS
- AWS APN Partners - Cost Management

Related videos:

- Deploying Cloud Intelligence Dashboards
- Get Alerts on any FinOps or Cost Optimization Metric or KPI

Related examples:

- Well-Architected Labs - AWS Account Setup
- Well-Architected Labs: Billing Visualization
- Well-Architected Labs: Cost and Governance Usage
- Well-Architected Labs: Cost and Usage Analysis
- Well-Architected Labs: Cost and Usage Visualization
- Well-Architected Labs: Cloud Intelligence Dashboards

COST03-BP06 Allocate costs based on workload metrics

> This best practice was updated with new guidance on July 13th, 2023.

Allocate the workload's costs by usage metrics or business outcomes to measure workload cost efficiency. Implement a process to analyze the cost and usage data with analytics services, which can provide insight and charge back capability.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Cost optimization is delivering business outcomes at the lowest price point, which can only be achieved by allocating workload costs by workload metrics (measured by workload efficiency). Monitor the defined workload metrics through log files or other application monitoring. Combine this data with the workload costs, which can be obtained by looking at costs with a specific tag value or account ID. It is recommended to perform this analysis at the hourly level. Your efficiency will typically change if you have some static cost components (for example, a backend database running permanently) with a varying request rate (for example, usage peaks at nine in the morning to five in the evening, with few requests at night). Understanding the relationship between the static and variable costs will help you to focus your optimization activities.

Creating workload metrics for shared resources may be challenging compared to resources like containerized applications on Amazon Elastic Container Service (Amazon ECS) and Amazon API Gateway. However, there are certain ways you can categorize usage and track cost. If you need to track Amazon ECS and AWS Batch shared resources, you can enable split cost allocation data in AWS Cost Explorer. With split cost allocation data, you can understand and optimize the cost and usage of your containerized applications and allocate application costs back to individual business entities based on how shared compute and memory resources are consumed. If you have shared API Gateway and AWS Lambda function usage, then you can use AWS Application Cost Profiler to categorize their consumption based on their `Tenant ID` or `Customer ID`.

Implementation steps

- **Allocate costs to workload metrics:** Using the defined metrics and configured tags, create a metric that combines the workload output and workload cost. Use analytics services such as Amazon Athena and Amazon Quick to create an efficiency dashboard for the overall workload and any components.

Resources

Related documents:

- Tagging AWS resources
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports

Related examples:

- Improve cost visibility of Amazon ECS and AWS Batch with AWS Split Cost Allocation Data

COST 4. How do you decommission resources?

Implement change control and resource management from project inception to end-of-life. This ensures you shut down or terminates unused resources to reduce waste.

Best practices

- COST04-BP01 Track resources over their lifetime

- COST04-BP02 Implement a decommissioning process
- COST04-BP03 Decommission resources
- COST04-BP04 Decommission resources automatically
- COST04-BP05 Enforce data retention policies

COST04-BP01 Track resources over their lifetime

Define and implement a method to track resources and their associations with systems over their lifetime. You can use tagging to identify the workload or function of the resource.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Decommission workload resources that are no longer required. A common example is resources used for testing: after testing has been completed, the resources can be removed. Tracking resources with tags (and running reports on those tags) can help you identify assets for decommission, as they will not be in use or the license on them will expire. Using tags is an effective way to track resources, by labeling the resource with its function, or a known date when it can be decommissioned. Reporting can then be run on these tags. Example values for feature tagging are `feature-X testing` to identify the purpose of the resource in terms of the workload lifecycle. Another example is using `LifeSpan` or TTL for the resources, such as to-be-deleted tag key name and value to define the time period or specific time for decommissioning.

Implementation steps

- **Implement a tagging scheme:** Implement a tagging scheme that identifies the workload the resource belongs to, verifying that all resources within the workload are tagged accordingly. Tagging helps you categorize resources by purpose, team, environment, or other criteria relevant to your business. For more detail on tagging uses cases, strategies, and techniques, see AWS Tagging Best Practices.
- **Implement workload throughput or output monitoring:** Implement workload throughput monitoring or alarming, initiating on either input requests or output completions. Configure it to provide notifications when workload requests or outputs drop to zero, indicating the workload resources are no longer used. Incorporate a time factor if the workload periodically drops to zero under normal conditions. For more detail on unused or underutilized resources, see AWS Trusted Advisor Cost Optimization checks.

- **Group AWS resources:** Create groups for AWS resources. You can use AWS Resource Groups to organize and manage your AWS resources that are in the same AWS Region. You can add tags to most of your resources to help identify and sort your resources within your organization. Use Tag Editor add tags to supported resources in bulk. Consider using AWS Service Catalog to create, manage, and distribute portfolios of approved products to end users and manage the product lifecycle.

Resources

Related documents:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS Trusted Advisor Cost Optimization Checks
- Tagging AWS resources
- Publishing Custom Metrics

Related videos:

- How to optimize costs using AWS Trusted Advisor

Related examples:

- Organize AWS resources
- Optimize cost using AWS Trusted Advisor

COST04-BP02 Implement a decommissioning process

Implement a process to identify and decommission unused resources.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implement a standardized process across your organization to identify and remove unused resources. The process should define the frequency searches are performed and the processes to remove the resource to verify that all organization requirements are met.

Implementation steps

- **Create and implement a decommissioning process:** Work with the workload developers and owners to build a decommissioning process for the workload and its resources. The process should cover the method to verify if the workload is in use, and also if each of the workload resources are in use. Detail the steps necessary to decommission the resource, removing them from service while ensuring compliance with any regulatory requirements. Any associated resources should be included, such as licenses or attached storage. Notify the workload owners that the decommissioning process has been started.

  Use the following decommission steps to guide you on what should be checked as part of your process:

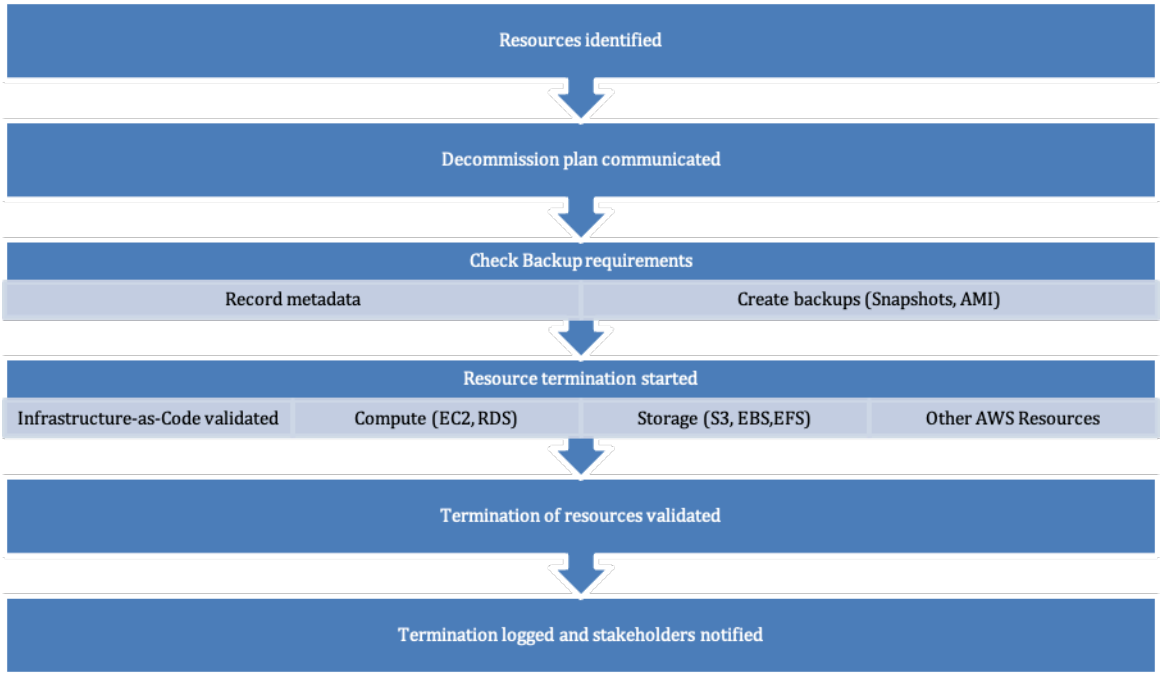
  - **Identify resources to be decommissioned:** Identify resources that are eligible for decommissioning in your AWS Cloud. Record all necessary information and schedule the decommission. In your timeline, be sure to account for if (and when) unexpected issues arise during the process.
  - **Coordinate and communicate:** Work with workload owners to confirm the resource to be decommissioned
  - **Record metadata and create backups:** Record metadata (such as public IPs, Region, AZ, VPC, Subnet, and Security Groups) and create backups (such as Amazon Elastic Block Store snapshots or taking AMI, keys export, and Certificate export) if it is required for the resources in the production environment or if they are critical resources.
  - **Validate infrastructure-as-code:** Determine whether resources were deployed with CloudFormation, Terraform, AWS Cloud Development Kit (AWS CDK), or any other infrastructure-as-code deployment tool so they can be re-deployed if necessary.
  - **Prevent access:** Apply restrictive controls for a period of time, to prevent the use of resources while you determine if the resource is required. Verify that the resource environment can be reverted to its original state if required.
  - **Follow your internal decommissioning process:** Follow the administrative tasks and decommissioning process of your organization, like removing the resource from your organization domain, removing the DNS record, and removing the resource from your configuration management tool, monitoring tool, automation tool and security tools.

  If the resource is an Amazon EC2 instance, consult the following list. For more detail, see How do I delete or terminate my Amazon EC2 resources?

- Stop or terminate all your Amazon EC2 instances and load balancers. Amazon EC2 instances are visible in the console for a short time after they're terminated. You aren't billed for any instances that aren't in the running state
- Delete your Auto Scaling infrastructure.
- Release all Dedicated Hosts.
- Delete all Amazon EBS volumes and Amazon EBS snapshots.
- Release all Elastic IP addresses.
- Deregister all Amazon Machine Images (AMIs).
- Terminate all AWS Elastic Beanstalk environments.

If the resource is an object in Amazon Glacier storage and if you delete an archive before meeting the minimum storage duration, you will be charged a prorated early deletion fee. Amazon Glacier minimum storage duration depends on the storage class used. For a summary of minimum storage duration for each storage class, see Performance across the Amazon S3 storage classes. For detail on how early deletion fees are calculated, see Amazon S3 pricing.

The following simple decommissioning process flowchart outlines the decommissioning steps. Before decommissioning resources, verify that resources you have identified for decommissioning are not being used by the organization.



Resource decommissioning flow.

Resources

Related documents:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS CloudTrail

Related videos:

- Delete CloudFormation stack but retain some resources
- Find out which user launched Amazon EC2 instance

Related examples:

- Delete or terminate Amazon EC2 resources
- Find out which user launched an Amazon EC2 instance

COST04-BP03 Decommission resources

Decommission resources initiated by events such as periodic audits, or changes in usage. Decommissioning is typically performed periodically and can be manual or automated.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The frequency and effort to search for unused resources should reflect the potential savings, so an account with a small cost should be analyzed less frequently than an account with larger costs. Searches and decommission events can be initiated by state changes in the workload, such as a product going end of life or being replaced. Searches and decommission events may also be initiated by external events, such as changes in market conditions or product termination.

Implementation steps

- **Decommission resources:** This is the depreciation stage of AWS resources that are no longer needed or ending of a licensing agreement. Complete all final checks completed before moving to the disposal stage and decommissioning resources to prevent any unwanted disruptions like

taking snapshots or backups. Using the decommissioning process, decommission each of the resources that have been identified as unused.

Resources

Related documents:

- AWS Auto Scaling
- AWS Trusted Advisor

Related examples:

- Well-Architected Labs: Decommission resources (Level 100)

COST04-BP04 Decommission resources automatically

Design your workload to gracefully handle resource termination as you identify and decommission non-critical resources, resources that are not required, or resources with low utilization.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use automation to reduce or remove the associated costs of the decommissioning process. Designing your workload to perform automated decommissioning will reduce the overall workload costs during its lifetime. You can use AWS Auto Scaling to perform the decommissioning process. You can also implement custom code using the API or SDK to decommission workload resources automatically.

Modern applications are built serverless-first, a strategy that prioritizes the adoption of serverless services. AWS developed serverless services for all three layers of your stack: compute, integration, and data stores. Using serverless architecture will allow you to save costs during low-traffic periods with scaling up and down automatically.

Implementation steps

- **Implement AWS Auto Scaling:** For resources that are supported, configure them with AWS Auto Scaling. AWS Auto Scaling can help you optimize your utilization and cost efficiencies when consuming AWS services. When demand drops, AWS Auto Scaling will automatically remove any excess resource capacity so you avoid overspending.

- **Configure CloudWatch to terminate instances:** Instances can be configured to terminate using CloudWatch alarms. Using the metrics from the decommissioning process, implement an alarm with an Amazon Elastic Compute Cloud action. Verify the operation in a non-production environment before rolling out.
- **Implement code within the workload:** You can use the AWS SDK or AWS CLI to decommission workload resources. Implement code within the application that integrates with AWS and terminates or removes resources that are no longer used.
- **Use serverless services:** Prioritize building serverless architectures and event-driven architecture on AWS to build and run your applications. AWS offers multiple serverless technology services that inherently provide automatically optimized resource utilization and automated decommissioning (scale in and scale out). With serverless applications, resource utilization is automatically optimized and you never pay for over-provisioning.

Resources

Related documents:

- AWS Auto Scaling
- AWS Trusted Advisor
- Serverless on AWS
- Create Alarms to Stop, Terminate, Reboot, or Recover an Instance
- Getting Started with Amazon EC2 Auto Scaling
- Adding terminate actions to Amazon CloudWatch alarms

Related examples:

- Scheduling automatic deletion of AWS CloudFormation stacks
- Well-Architected Labs – Decommission resources automatically (Level 100)
- Servian AWS Auto Cleanup

COST04-BP05 Enforce data retention policies

Define data retention policies on supported resources to handle object deletion per your organizations' requirements. Identify and delete unnecessary or orphaned resources and objects that are no longer required.

Level of risk exposed if this best practice is not established: Medium

Use data retention policies and lifecycle policies to reduce the associated costs of the decommissioning process and storage costs for the identified resources. Defining your data retention policies and lifecycle policies to perform automated storage class migration and deletion will reduce the overall storage costs during its lifetime. You can use Amazon Data Lifecycle Manager to automate the creation and deletion of Amazon Elastic Block Store snapshots and Amazon EBS-backed Amazon Machine Images (AMIs), and use Amazon S3 Intelligent-Tiering or an Amazon S3 lifecycle configuration to manage the lifecycle of your Amazon S3 objects. You can also implement custom code using the API or SDK to create lifecycle policies and policy rules for objects to be deleted automatically.

Implementation steps

- **Use Amazon Data Lifecycle Manager:** Use lifecycle policies on Amazon Data Lifecycle Manager to automate deletion of Amazon EBS snapshots and Amazon EBS-backed AMIs.
- **Set up lifecycle configuration on a bucket:** Use Amazon S3 lifecycle configuration on a bucket to define actions for Amazon S3 to take during an object's lifecycle, as well as deletion at the end of the object's lifecycle, based on your business requirements.

Resources

Related documents:

- AWS Trusted Advisor
- Amazon Data Lifecycle Manager
- How to set lifecycle configuration on Amazon S3 bucket

Related videos:

- Automate Amazon EBS Snapshots with Amazon Data Lifecycle Manager
- Empty an Amazon S3 bucket using a lifecycle configuration rule

Related examples:

- Empty an Amazon S3 bucket using a lifecycle configuration rule
- Well-Architected Lab: Decommission resources automatically (Level 100)

Cost-effective resources

Questions

- COST 5. How do you evaluate cost when you select services?
- COST 6. How do you meet cost targets when you select resource type, size and number?
- COST 7. How do you use pricing models to reduce cost?
- COST 8. How do you plan for data transfer charges?

COST 5. How do you evaluate cost when you select services?

Amazon EC2, Amazon EBS, and Amazon S3 are building-block AWS services. Managed services, such as Amazon RDS and Amazon DynamoDB, are higher level, or application level, AWS services. By selecting the appropriate building blocks and managed services, you can optimize this workload 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

- COST05-BP01 Identify organization requirements for cost
- COST05-BP02 Analyze all components of the workload
- COST05-BP03 Perform a thorough analysis of each component
- COST05-BP04 Select software with cost-effective licensing
- COST05-BP05 Select components of this workload to optimize cost in line with organization priorities
- COST05-BP06 Perform cost analysis for different usage over time

COST05-BP01 Identify organization requirements for cost

Work with team members to define the balance between cost optimization and other pillars, such as performance and reliability, for this workload.

Level of risk exposed if this best practice is not established: High

Implementation guidance

When selecting services for your workload, it is key that you understand your organization priorities. Ensure that you have a balance between cost and other Well-Architected pillars, such as performance and reliability. A fully cost-optimized workload is the solution that is most aligned to your organization's requirements, not necessarily the lowest cost. Meet with all teams within your organization to collect information, such as product, business, technical, and finance.

Implementation steps

- **Identify organization requirements for cost:** Meet with team members from your organization, including those in product management, application owners, development and operational teams, management, and financial roles. Prioritize the Well-Architected pillars for this workload and its components, the output is a list of the pillars in order. You can also add a weighting to each, which can indicate how much additional focus a pillar has, or how similar the focus is between two pillars.

Resources

Related documents:

- AWS Total Cost of Ownership (TCO) Calculator
- Amazon S3 storage classes
- Cloud products

COST05-BP02 Analyze all components of the workload

Verify every workload component is analyzed, regardless of current size or current costs. The review effort should reflect the potential benefit, such as current and projected costs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Perform a thorough analysis on all components in your workload. Ensure that balance between the cost of analysis and the potential savings in the workload over its lifecycle. You must find the current impact, and potential future impact, of the component. For example, if the cost of the proposed resource is $10 a month, and under forecasted loads would not exceed $15 a month, spending a day of effort to reduce costs by 50% ($5 a month) could exceed the potential benefit

over the life of the system. Using a faster and more efficient data-based estimation will create the best overall outcome for this component.

Workloads can change over time, and the right set of services may not be optimal if the workload architecture or usage changes. Analysis for selection of services must incorporate current and future workload states and usage levels. Implementing a service for future workload state or usage may reduce overall costs by reducing or removing the effort required to make future changes.

AWS Cost Explorer and the AWS Cost and Usage Report (CUR) can analyze the cost of a Proof of Concept (PoC) or running environment. You can also use AWS Pricing Calculator to estimate workload costs.

Implementation steps

- **List the workload components:** Build the list of all the workload components. This is used as verification to check that each component was analyzed. The effort spent should reflect the criticality to the workload as defined by your organization's priorities. Grouping together resources functionally improves efficiency, for example production database storage, if there are multiple databases.
- **Prioritize component list:** Take the component list and prioritize it in order of effort. This is typically in order of the cost of the component from most expensive to least expensive, or the criticality as defined by your organization's priorities.
- **Perform the analysis:** For each component on the list, review the options and services available and chose the option that aligns best with your organizational priorities.

Resources

Related documents:

- AWS Pricing Calculator
- AWS Cost Explorer
- Amazon S3 storage classes
- Cloud products

COST05-BP03 Perform a thorough analysis of each component

Look at overall cost to the organization of each component. Calculate the total cost of ownership by factoring in cost of operations and management, especially when using managed services

by cloud provider. The review effort should reflect potential benefit (for example, time spent analyzing is proportional to component cost).

Level of risk exposed if this best practice is not established: High

Implementation guidance

Consider the time savings that will allow your team to focus on retiring technical debt, innovation, value-adding features and building what differentiates the business. For example, you might need to lift and shift (also known as rehost) your databases from your on-premises environment to the cloud as rapidly as possible and optimize later. It is worth exploring the possible savings attained by using managed services on AWS that may remove or reduce license costs. Managed services on AWS remove the operational and administrative burden of maintaining a service, such as patching or upgrading the OS, and allow you to focus on innovation and business.

Since managed services operate at cloud scale, they can offer a lower cost per transaction or service. You can make potential optimizations in order to achieve some tangible benefit, without changing the core architecture of the application. For example, you may be looking to reduce the amount of time you spend managing database instances by migrating to a database-as-a-service platform like Amazon Relational Database Service (Amazon RDS) or migrating your application to a fully managed platform like AWS Elastic Beanstalk.

Usually, managed services have attributes that you can set to ensure sufficient capacity. You must set and monitor these attributes so that your excess capacity is kept to a minimum and performance is maximized. You can modify the attributes of AWS Managed Services using the AWS Management Console or AWS APIs and SDKs to align resource needs with changing demand. For example, you can increase or decrease the number of nodes on an Amazon EMR cluster (or an Amazon Redshift cluster) to scale out or in.

You can also pack multiple instances on an AWS resource to activate higher density usage. For example, you can provision multiple small databases on a single Amazon Relational Database Service (Amazon RDS) database instance. As usage grows, you can migrate one of the databases to a dedicated Amazon RDS database instance using a snapshot and restore process.

When provisioning workloads on managed services, you must understand the requirements of adjusting the service capacity. These requirements are typically time, effort, and any impact to normal workload operation. The provisioned resource must allow time for any changes to occur, provision the required overhead to allow this. The ongoing effort required to modify services can be reduced to virtually zero by using APIs and SDKs that are integrated with system and monitoring tools, such as Amazon CloudWatch.

Amazon RDS, Amazon Redshift, and Amazon ElastiCache provide a managed database service. Amazon Athena, Amazon EMR, and Amazon OpenSearch Service provide a managed analytics service.

AMS is a service that operates AWS infrastructure on behalf of enterprise customers and partners. It provides a secure and compliant environment that you can deploy your workloads onto. AMS uses enterprise cloud operating models with automation to allow you to meet your organization requirements, move into the cloud faster, and reduce your on-going management costs.

Implementation steps

- **Perform a thorough analysis:** Using the component list, work through each component from the highest priority to the lowest priority. For the higher priority and more costly components, perform additional analysis and assess all available options and their long term impact. For lower priority components, assess if changes in usage would change the priority of the component, and then perform an analysis of appropriate effort.
- **Compare managed and unmanaged resources:** Consider the operational cost for the resources you manage and compare them with AWS managed resources. For example, review your databases running on Amazon EC2 instances and compare with Amazon RDS options (an AWS managed service) or Amazon EMR compared to running Apache Spark on Amazon EC2. When moving from a self-managed workload to a AWS fully managed workload, research your options carefully. The three most important factors to consider are the type of managed service you want to use, the process you will use to migrate your data and understand the AWS shared responsibility model.

Resources

Related documents:

- AWS Total Cost of Ownership (TCO) Calculator
- Amazon S3 storage classes
- AWS Cloud products
- AWS Shared Responsibility Model

Related videos:

- Why move to a managed database?

- What is Amazon EMR and how can I use it for processing data?

Related examples:

- Why to move to a managed database
- Consolidate data from identical SQL Server databases into a single Amazon RDS for SQL Server database using AWS DMS
- Deliver data at scale to Amazon Managed Streaming for Apache Kafka (Amazon MSK)
- Migrate an ASP.NET web application to AWS Elastic Beanstalk

COST05-BP04 Select software with cost-effective licensing

Open-source software eliminates software licensing costs, which can contribute significant costs to workloads. Where licensed software is required, avoid licenses bound to arbitrary attributes such as CPUs, look for licenses that are bound to output or outcomes. The cost of these licenses scales more closely to the benefit they provide.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

The cost of software licenses can be eliminated through the use of open-source software. This can have significant impact on workload costs as the size of the workload scales. Measure the benefits of licensed software against the total cost to ensure that you have the most optimized workload. Model any changes in licensing and how they would impact your workload costs. If a vendor changes the cost of your database license, investigate how that impacts the overall efficiency of your workload. Consider historical pricing announcements from your vendors for trends of licensing changes across their products. Licensing costs may also scale independently of throughput or usage, such as licenses that scale by hardware (CPU-bound licenses). These licenses should be avoided because costs can rapidly increase without corresponding outcomes.

Implementation steps

- **Analyze license options:** Review the licensing terms of available software. Look for open-source versions that have the required functionality, and whether the benefits of licensed software outweigh the cost. Favorable terms will align the cost of the software to the benefit it provides.
- **Analyze the software provider:** Review any historical pricing or licensing changes from the vendor. Look for any changes that do not align to outcomes, such as punitive terms for running

on specific vendors hardware or platforms. Additionally look for how they run audits, and penalties that could be imposed.

Resources

Related documents:

- AWS Total Cost of Ownership (TCO) Calculator
- Amazon S3 storage classes
- Cloud products

COST05-BP05 Select components of this workload to optimize cost in line with organization priorities

Factor in cost when selecting all components for your workload. This includes using application level and managed services or serverless, containers, or event-driven architecture to reduce overall cost. Minimize license costs by using open-source software, software that does not have license fees, or alternatives to reduce the cost.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Consider the cost of services and options when selecting all components. This includes using application level and managed services, such as Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB, Amazon Simple Notification Service (Amazon SNS), and Amazon Simple Email Service (Amazon SES) to reduce overall organization cost. Use serverless and containers for compute, such as AWS Lambda and Amazon Simple Storage Service (Amazon S3) for static websites. Containerize your application if possible and use AWS Managed Container Services such as Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS). Minimize license costs by using open-source software, or software that does not have license fees (for example, Amazon Linux for compute workloads or migrate databases to Amazon Aurora).

You can use serverless or application-level services such as AWS Lambda, Amazon Simple Queue Service (Amazon SQS), Amazon SNS, and Amazon SES. These services remove the need for you to manage a resource, and provide the function of running code, queuing services, and message delivery. The other benefit is that they scale in performance and cost in line with usage, allowing efficient cost allocation and attribution.

Using event-driven architecture (EDA) is also possible with serverless services. Event-driven architectures are push-based, so everything happens on demand as the event presents itself in the router. This way, you're not paying for continuous polling to check for an event. This means less network bandwidth consumption, less CPU utilization, less idle fleet capacity, and fewer SSL/TLS handshakes.

For more information on Serverless, refer to the Well-Architected Serverless Application Lens whitepaper.

Implementation steps

- **Select each service to optimize cost:** Using your prioritized list and analysis, select each option that provides the best match with your organizational priorities. Instead of increasing the capacity to meet the demand, consider other options which may give you better performance with lower cost. For example, you need to review expected traffic for your databases on AWS and consider either increasing the instance size or using Amazon ElastiCache services (Redis or Memcached) to provide cached mechanisms for your databases.
- **Evaluate event-driven architecture:** Using serverless architecture also allows you to build event-driven architecture for distributed microservice-based applications, which helps you build scalable, resilient, agile and cost-effective solutions.

Resources

Related documents:

- AWS Total Cost of Ownership (TCO) Calculator
- AWS Serverless
- What is Event-Driven Architecture
- Amazon S3 storage classes
- Cloud products
- Amazon ElastiCache (Redis OSS)

Related examples:

- Getting started with event-driven architecture
- What is an Event-Driven Architecture?

- How Statsig runs 100x more cost-effectively using Amazon ElastiCache (Redis OSS)
- Best practices for working with AWS Lambda functions

COST05-BP06 Perform cost analysis for different usage over time

Workloads can change over time. Some services or features are more cost effective at different usage levels. By performing the analysis on each component over time and at projected usage, the workload remains cost-effective over its lifetime.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

As AWS releases new services and features, the optimal services for your workload may change. Effort required should reflect potential benefits. Workload review frequency depends on your organization requirements. If it is a workload of significant cost, implementing new services sooner will maximize cost savings, so more frequent review can be advantageous. Another initiation for review is change in usage patterns. Significant changes in usage can indicate that alternate services would be more optimal.

If you need to move data into AWS Cloud, you can select any wide variety of services AWS offers and partner tools to help you migrate your data sets, whether they are files, databases, machine images, block volumes, or even tape backups. For example, to move a large amount of data to and from AWS or process data at the edge, you can use one of the AWS purpose-built devices to cost effectively move petabytes of data offline. Another example is for higher data transfer rates, a direct connect service may be cheaper than a VPN which provides the required consistent connectivity for your business.

Based on the cost analysis for different usage over time, review your scaling activity. Analyze the result to see if the scaling policy can be tuned to add instances with multiple instance types and purchase options. Review your settings to see if the minimum can be reduced to serve user requests but with a smaller fleet size, and add more resources to meet the expected high demand.

Perform cost analysis for different usage over time by discussing with stakeholders in your organization and use AWS Cost Explorer's forecast feature to predict the potential impact of service changes. Monitor usage level launches using AWS Budgets, CloudWatch billing alarms and AWS Cost Anomaly Detection to identify and implement the most cost-effective services sooner.

Implementation steps

- **Define predicted usage patterns:** Working with your organization, such as marketing and product owners, document what the expected and predicted usage patterns will be for the workload. Discuss with business stakeholders about both historical and forecasted cost and usage increases and make sure increases align with business requirements. Identify calendar days, weeks, or months where you expect more users to use your AWS resources, which indicate that you should increase the capacity of the existing resources or adopt additional services to reduce the cost and increase performance.
- **Perform cost analysis at predicted usage:** Using the usage patterns defined, perform analysis at each of these points. The analysis effort should reflect the potential outcome. For example, if the change in usage is large, a thorough analysis should be performed to verify any costs and changes. In other words, when cost increases, usage should increase for business as well.

Resources

Related documents:

- AWS Total Cost of Ownership (TCO) Calculator
- Amazon S3 storage classes
- Cloud products
- Amazon EC2 Auto Scaling
- Cloud Data Migration
- AWS Snow Family

Related videos:

- AWS OpsHub for Snow Family

COST 6. How do you meet cost targets when you select resource type, size and number?

Verify that you choose the appropriate resource size and number of resources for the task at hand. You minimize waste by selecting the most cost effective type, size, and number.

Best practices

- COST06-BP01 Perform cost modeling

- COST06-BP02 Select resource type, size, and number based on data
- COST06-BP03 Select resource type, size, and number automatically based on metrics

COST06-BP01 Perform cost modeling

Identify organization requirements (such as business needs and existing commitments) and perform cost modeling (overall costs) of the workload and each of its components. Perform benchmark activities for the workload under different predicted loads and compare the costs. The modeling effort should reflect the potential benefit. For example, time spent is proportional to component cost.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Perform cost modelling for your workload and each of its components to understand the balance between resources, and find the correct size for each resource in the workload, given a specific level of performance. Understanding cost considerations can inform your organizational business case and decision-making process when evaluating the value realization outcomes for planned workload deployment.

Perform benchmark activities for the workload under different predicted loads and compare the costs. The modelling effort should reflect potential benefit; for example, time spent is proportional to component cost or predicted saving. For best practices, refer to the Review section of the Performance Efficiency Pillar of the AWS Well-Architected Framework.

As an example, to create cost modeling for a workload consisting of compute resources, AWS Compute Optimizer can assist with cost modelling for running workloads. It provides right-sizing recommendations for compute resources based on historical usage. Make sure CloudWatch Agents are deployed to the Amazon EC2 instances to collect memory metrics which help you with more accurate recommendations within AWS Compute Optimizer. This is the ideal data source for compute resources because it is a free service that uses machine learning to make multiple recommendations depending on levels of risk.

There are multiple services you can use with custom logs as data sources for rightsizing operations for other services and workload components, such as AWS Trusted Advisor, Amazon CloudWatch and Amazon CloudWatch Logs. AWS Trusted Advisor checks resources and flags resources with low utilization which can help you right size your resources and create cost modelling.

The following are recommendations for cost modelling data and metrics:

- The monitoring must accurately reflect the user experience. Select the correct granularity for the time period and thoughtfully choose the maximum or 99th percentile instead of the average.
- Select the correct granularity for the time period of analysis that is required to cover any workload cycles. For example, if a two-week analysis is performed, you might be overlooking a monthly cycle of high utilization, which could lead to under-provisioning.
- Choose the right AWS services for your planned workload by considering your existing commitments, selected pricing models for other workloads, and ability to innovate faster and focus on your core business value.

Implementation steps

- **Perform cost modeling for resources:** Deploy the workload or a proof of concept into a separate account with the specific resource types and sizes to test. Run the workload with the test data and record the output results, along with the cost data for the time the test was run. Afterwards, redeploy the workload or change the resource types and sizes and run the test again. Include license fees of any products you may use with these resources and estimated operations (labor or engineer) costs for deploying and managing these resources while creating cost modeling. Consider cost modeling for a period (hourly, daily, monthly, yearly or three years).

Resources

Related documents:

- AWS Auto Scaling
- Identifying Opportunities to Right Size
- Amazon CloudWatch features
- Cost Optimization: Amazon EC2 Right Sizing
- AWS Compute Optimizer
- AWS Pricing Calculator

Related examples:

- Perform a Data-Driven Cost Modelling
- Estimate the cost of planned AWS resource configurations
- Choose the right AWS tools

COST06-BP02 Select resource type, size, and number based on data

Select resource size or type based on data about the workload and resource characteristics. For example, compute, memory, throughput, or write intensive. This selection is typically made using a previous (on-premises) version of the workload, using documentation, or using other sources of information about the workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Select resource size or type based on workload and resource characteristics, for example, compute, memory, throughput, or write intensive. This selection is typically made using cost modelling, a previous version of the workload (such as an on-premises version), using documentation, or using other sources of information about the workload (whitepapers, published solutions).

Implementation steps

- **Select resources based on data:** Using your cost modeling data, select the expected workload usage level, then select the specified resource type and size.

Resources

Related documents:

- AWS Auto Scaling
- Amazon CloudWatch features
- Cost Optimization: EC2 Right Sizing

COST06-BP03 Select resource type, size, and number automatically based on metrics

Use metrics from the currently running workload to select the right size and type to optimize for cost. Appropriately provision throughput, sizing, and storage for compute, storage, data, and networking services. This can be done with a feedback loop such as automatic scaling or by custom code in the workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Create a feedback loop within the workload that uses active metrics from the running workload to make changes to that workload. You can use a managed service, such as AWS Auto Scaling, which you configure to perform the right sizing operations for you. AWS also provides APIs, SDKs, and features that allow resources to be modified with minimal effort. You can program a workload to stop-and-start an Amazon EC2 instance to allow a change of instance size or instance type. This provides the benefits of right-sizing while removing almost all the operational cost required to make the change.

Some AWS services have built in automatic type or size selection, such as Amazon Simple Storage Service Intelligent-Tiering. Amazon S3 Intelligent-Tiering automatically moves your data between two access tiers, frequent access and infrequent access, based on your usage patterns.

Implementation steps

- **Increase your observability by configuring workload metrics:** Capture key metrics for the workload. These metrics provide an indication of the customer experience, such as workload output, and align to the differences between resource types and sizes, such as CPU and memory usage. For compute resource, analyze performance data to right size your Amazon EC2 instances. Identify idle instances and ones that are underutilized. Key metrics to look for are CPU usage and memory utilization (for example, 40% CPU utilization at 90% of the time as explained in Rightsizing with AWS Compute Optimizer and Memory Utilization Enabled). Identify instances with a maximum CPU usage and memory utilization of less than 40% over a four-week period. These are the instances to right size to reduce costs. For storage resources such as Amazon S3, you can use Amazon S3 Storage Lens, which allows you to see 28 metrics across various categories at the bucket level, and 14 days of historical data in the dashboard by default. You can filter your Amazon S3 Storage Lens dashboard by summary and cost optimization or events to analyze specific metrics.
- **View rightsizing recommendations:** Use the rightsizing recommendations in AWS Compute Optimizer and the Amazon EC2 rightsizing tool in the Cost Management console, or review AWS Trusted Advisor right-sizing your resources to make adjustments on your workload. It is important to use the right tools when right-sizing different resources and follow right-sizing guidelines whether it is an Amazon EC2 instance, AWS storage classes, or Amazon RDS instance types. For storage resources, you can use Amazon S3 Storage Lens, which gives you visibility into object storage usage, activity trends, and makes actionable recommendations to optimize costs and apply data protection best practices. Using the contextual recommendations that

Amazon S3 Storage Lens derives from analysis of metrics across your organization, you can take immediate steps to optimize your storage.

- **Select resource type and size automatically based on metrics:** Using the workload metrics, manually or automatically select your workload resources. For compute resources, configuring AWS Auto Scaling or implementing code within your application can reduce the effort required if frequent changes are needed, and it can potentially implement changes sooner than a manual process. You can launch and automatically scale a fleet of On-Demand Instances and Spot Instances within a single Auto Scaling group. In addition to receiving discounts for using Spot Instances, you can use Reserved Instances or a Savings Plan to receive discounted rates of the regular On-Demand Instance pricing. All of these factors combined help you optimize your cost savings for Amazon EC2 instances and determine the desired scale and performance for your application. You can also use an attribute-based instance type selection (ABS) strategy in Auto Scaling Groups (ASG), which lets you express your instance requirements as a set of attributes, such as vCPU, memory, and storage. You can automatically use newer generation instance types when they are released and access a broader range of capacity with Amazon EC2 Spot Instances. Amazon EC2 Fleet and Amazon EC2 Auto Scaling select and launch instances that fit the specified attributes, removing the need to manually pick instance types. For storage resources, you can use the Amazon S3 Intelligent Tiering and Amazon EFS Infrequent Access features, which allow you to select storage classes automatically that deliver automatic storage cost savings when data access patterns change, without performance impact or operational overhead.

Resources

Related documents:

- AWS Auto Scaling
- AWS Right-Sizing
- AWS Compute Optimizer
- Amazon CloudWatch features
- CloudWatch Getting Set Up
- CloudWatch Publishing Custom Metrics
- Getting Started with Amazon EC2 Auto Scaling
- Amazon S3 Storage Lens
- Amazon S3 Intelligent-Tiering

- Amazon EFS Infrequent Access
- Launch an Amazon EC2 Instance Using the SDK

Related videos:

- Right Size Your Services

Related examples:

- Attribute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet
- Optimizing Amazon Elastic Container Service for cost using scheduled scaling
- Predictive scaling with Amazon EC2 Auto Scaling
- Optimize Costs and Gain Visibility into Usage with Amazon S3 Storage Lens
- Well-Architected Labs: Rightsizing Recommendations (Level 100)
- Well-Architected Labs: Rightsizing with AWS Compute Optimizer and Memory Utilization Enabled (Level 200)

COST 7. How do you use pricing models to reduce cost?

Use the pricing model that is most appropriate for your resources to minimize expense.

Best practices

- COST07-BP01 Perform pricing model analysis
- COST07-BP02 Implement Regions based on cost
- COST07-BP03 Select third-party agreements with cost-efficient terms
- COST07-BP04 Implement pricing models for all components of this workload
- COST07-BP05 Perform pricing model analysis at the management account level

COST07-BP01 Perform pricing model analysis

Analyze each component of the workload. Determine if the component and resources will be running for extended periods (for commitment discounts) or dynamic and short-running (for spot or on-demand). Perform an analysis on the workload using the recommendations in cost management tools and apply business rules to those recommendations to achieve high returns.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS has multiple pricing models that allow you to pay for your resources in the most cost-effective way that suits your organization's needs and depending on product. Work with your teams to determine the most appropriate pricing model. Often your pricing model consists of a combination of multiple options, as determined by your availability

On-Demand Instances allow you pay for compute or database capacity by the hour or by the second (60 seconds minimum) depending on which instances you run, without long-term commitments or upfront payments.

Savings Plans are a flexible pricing model that offers low prices on Amazon EC2, Lambda, and AWS Fargate usage, in exchange for a commitment to a consistent amount of usage (measured in dollars per hour) over one year or three years terms.

Spot Instances are an Amazon EC2 pricing mechanism that allows you request spare compute capacity at discounted hourly rate (up to 90% off the on-demand price) without upfront commitment.

Reserved Instances allow you up to 75 percent discount by prepaying for capacity. For more details, see Optimizing costs with reservations.

You might choose to include a Savings Plans for the resources associated with the production, quality, and development environments. Alternatively, because sandbox resources are only powered on when needed, you might choose an on-demand model for the resources in that environment. Use Amazon Spot Instances to reduce Amazon EC2 costs or use Compute Savings Plans to reduce Amazon EC2, Fargate, and Lambda cost. The AWS Cost Explorer recommendations tool provides opportunities for commitment discounts with Saving plans.

If you have been purchasing Reserved Instances for Amazon EC2 in the past or have established cost allocation practices inside your organization, you can continue using Amazon EC2 Reserved Instances for the time being. However, we recommend working on a strategy to use Savings Plans in the future as a more flexible cost savings mechanism. You can refresh Savings Plans (SP) Recommendations in AWS Cost Management to generate new Savings Plans Recommendations at any time. Use Reserved Instances (RI) to reduce Amazon RDS, Amazon Redshift, Amazon ElastiCache, and Amazon OpenSearch Service costs. Saving Plans and Reserved Instances are available in three options: all upfront, partial upfront and no upfront payments. Use the recommendations provided in AWS Cost Explorer RI and SP purchase recommendations.

To find opportunities for Spot workloads, use an hourly view of your overall usage, and look for regular periods of changing usage or elasticity. You can use Spot Instances for various fault-tolerant and flexible applications. Examples include stateless web servers, API endpoints, big data and analytics applications, containerized workloads, CI/CD, and other flexible workloads.

Analyze your Amazon EC2 and Amazon RDS instances whether they can be turned off when you don't use (after hours and weekends). This approach will allow you to reduce costs by 70% or more compared to using them 24/7. If you have Amazon Redshift clusters that only need to be available at specific times, you can pause the cluster and later resume it. When the Amazon Redshift cluster or Amazon EC2 and Amazon RDS Instance is stopped, the compute billing halts and only the storage charge applies.

Note that On-Demand Capacity reservations (ODCR) are not a pricing discount. Capacity Reservations are charged at the equivalent On-Demand rate, whether you run instances in reserved capacity or not. They should be considered when you need to provide enough capacity for the resources you plan to run. ODCRs don't have to be tied to long-term commitments, as they can be cancelled when you no longer need them, but they can also benefit from the discounts that Savings Plans or Reserved Instances provide.

Implementation steps

- **Analyze workload elasticity:** Using the hourly granularity in Cost Explorer or a custom dashboard, analyze your workload's elasticity. Look for regular changes in the number of instances that are running. Short duration instances are candidates for Spot Instances or Spot Fleet.
  - Well-Architected Lab: Cost Explorer
  - Well-Architected Lab: Cost Visualization
- **Review existing pricing contracts:** Review current contracts or commitments for long term needs. Analyze what you currently have and how much those commitments are in use. Leverage pre-existing contractual discounts or enterprise agreements. Enterprise Agreements give customers the option to tailor agreements that best suit their needs. For long term commitments, consider reserved pricing discounts, Reserved Instances or Savings Plans for the specific instance type, instance family, AWS Region, and Availability Zones.
- **Perform a commitment discount analysis:** Using Cost Explorer in your account, review the Savings Plans and Reserved Instance recommendations. To verify that you implement the correct recommendations with the required discounts and risk, follow the Well-Architected labs.

Resources

Related documents:

- Accessing Reserved Instance recommendations
- Instance purchasing options
- AWS Enterprise

Related videos:

- Save up to 90% and run production workloads on Spot

Related examples:

- Well-Architected Lab: Cost Explorer
- Well-Architected Lab: Cost Visualization
- Well-Architected Lab: Pricing Models

COST07-BP02 Implement Regions based on cost

> This best practice was updated with new guidance on July 13th, 2023.

Resource pricing may be different in each Region. Identify Regional cost differences and only deploy in Regions with higher costs to meet latency, data residency and data sovereignty requirements. Factoring in Region cost helps you pay the lowest overall price for this workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The AWS Cloud Infrastructure is global, hosted in multiple locations world-wide, and built around AWS Regions, Availability Zones, Local Zones, AWS Outposts, and Wavelength Zones. A Region is a physical location in the world and each Region is a separate geographic area where AWS has multiple Availability Zones. Availability Zones which are multiple isolated locations within each Region consist of one or more discrete data centers, each with redundant power, networking, and connectivity.

Each AWS Region operates within local market conditions, and resource pricing is different in each Region due to differences in the cost of land, fiber, electricity, and taxes, for example. Choose a specific Region to operate a component of or your entire solution so that you can run at the lowest possible price globally. Use AWS Calculator to estimate the costs of your workload in various Regions by searching services by location type (Region, wave length zone and local zone) and Region.

When you architect your solutions, a best practice is to seek to place computing resources closer to users to provide lower latency and strong data sovereignty. Select the geographic location based on your business, data privacy, performance, and security requirements. For applications with global end users, use multiple locations.

Use Regions which provide lower prices for AWS services to deploy your workloads if you have no obligations in data privacy, security and business requirements. For example, if your default Region is ap-southeasth-2 (Sydney), and if there are no restrictions (data privacy, security, for example) to use other Regions, deploying non-critical (development and test) Amazon EC2 instances in north-east-1 (N. Virginia) Region will cost you less.

	Compliance	*Latency*	*Cost*	*Services / Features*
Region 1	✓	15 ms	$$	✓
Region 2	✓	20 ms	$$$	X
Region 3	✓	80 ms	$	✓
Region 4	✓	15 ms	$$	✓
Region 5	✓	20 ms	$$$	X
Region 6	✓	15 ms	$	✓
Region 7	✓	80 ms	$	✓
Region 8	✓	15 ms	$	X

Region feature matrix table

The preceding matrix table shows us that Region 4 is the best option for this given scenario because latency is low compared to other regions, service is available, and it is the least expensive Region.

Implementation steps

- **Review AWS Region pricing:** Analyze the workload costs in the current Region. Starting with the highest costs by service and usage type, calculate the costs in other Regions that are available. If the forecasted saving outweighs the cost of moving the component or workload, migrate to the new Region.
- **Review requirements for multi-Region deployments:** Analyze your business requirements and obligations (data privacy, security, or performance) to find out if there are any restrictions for you to not to use multiple Regions. If there are no obligations to restrict you to use single Region, then use multiple Regions.
- **Analyze required data transfer:** Consider data transfer costs when selecting Regions. Keep your data close to your customer and close to the resources. Select less costly AWS Regions where data flows and where there is minimal data transfer. Depending on your business requirements for data transfer, you can use Amazon CloudFront, AWS PrivateLink, AWS Direct Connect, and AWS Virtual Private Network to reduce your networking costs, improve performance, and enhance security.

Resources

Related documents:

- Accessing Reserved Instance recommendations
- Amazon EC2 pricing
- Instance purchasing options
- Region Table

Related videos:

- Save up to 90% and run production workloads on Spot

Related examples:

- Overview of Data Transfer Costs for Common Architectures
- Cost Considerations for Global Deployments
- What to Consider when Selecting a Region for your Workloads
- Well-Architected Labs: Restrict service usage by Region (Level 200)

COST07-BP03 Select third-party agreements with cost-efficient terms

Cost efficient agreements and terms ensure the cost of these services scales with the benefits they provide. Select agreements and pricing that scale when they provide additional benefits to your organization.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

When you utilize third-party solutions or services in the cloud, it is important that the pricing structures are aligned to Cost Optimization outcomes. Pricing should scale with the outcomes and value it provides. An example of this is software that takes a percentage of savings it provides, the more you save (outcome) the more it charges. Agreements that scale with your bill are typically not aligned to Cost Optimization, unless they provide outcomes for every part of your specific bill. For example, a solution that provides recommendations for Amazon Elastic Compute Cloud(Amazon EC2) and charges a percentage of your entire bill will increase if you use other services for which it provides no benefit. Another example is a managed service that is charged at a percentage of the cost of resources that are managed. A larger instance size may not necessarily require more management effort, but will be charged more. Ensure that these service pricing arrangements include a cost optimization program or features in their service to drive efficiency.

Implementation steps

- **Analyze third-party agreements and terms:** Review the pricing in third party agreements. Perform modeling for different levels of your usage, and factor in new costs such as new service usage, or increases in current services due to workload growth. Decide if the additional costs provide the required benefits to your business.

Resources

Related documents:

- Accessing Reserved Instance recommendations
- Instance purchasing options

Related videos:

- Save up to 90% and run production workloads on Spot

COST07-BP04 Implement pricing models for all components of this workload

Permanently running resources should utilize reserved capacity such as Savings Plans or Reserved Instances. Short-term capacity is configured to use Spot Instances, or Spot Fleet. On-Demand Instances are only used for short-term workloads that cannot be interrupted and do not run long enough for reserved capacity, between 25% to 75% of the period, depending on the resource type.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Consider the requirements of the workload components and understand the potential pricing models. Define the availability requirement of the component. Determine if there are multiple independent resources that perform the function in the workload, and what the workload requirements are over time. Compare the cost of the resources using the default On-Demand pricing model and other applicable models. Factor in any potential changes in resources or workload components.

Implementation steps

- **Implement pricing models:** Using your analysis results, purchase Savings Plans (SPs), Reserved Instances (RIs) or implement Spot Instances. If it is your first RI purchase then choose the top 5 or 10 recommendations in the list, then monitor and analyze the results over the next month or two. Purchase small numbers of commitment discounts regular cycles, for example every two weeks or monthly. Implement Spot Instances for workloads that can be interrupted or are stateless.
- **Workload review cycle:** Implement a review cycle for the workload that specifically analyzes pricing model coverage. Once the workload has the required coverage, purchase additional commitment discounts every two to four weeks, or as your organization usage changes.

Resources

Related documents:

- Accessing Reserved Instance recommendations
- EC2 Fleet
- How to Purchase Reserved Instances
- Instance purchasing options
- Spot Instances

Related videos:

- Save up to 90% and run production workloads on Spot

COST07-BP05 Perform pricing model analysis at the management account level

This best practice was updated with new guidance on July 13th, 2023.

Check billing and cost management tools and see recommended discounts with commitments and reservations to perform regular analysis at the management account level.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Performing regular cost modeling helps you implement opportunities to optimize across multiple workloads. For example, if multiple workloads use On-Demand Instances, at an aggregate level, the risk of change is lower, and implementing a commitment-based discount will achieve a lower overall cost. It is recommended to perform analysis in regular cycles of two weeks to one month. This allows you to make small adjustment purchases, so the coverage of your pricing models continues to evolve with your changing workloads and their components.

Use the AWS Cost Explorer recommendations tool to find opportunities for commitment discounts in your management account. Recommendations at the management account level are calculated considering usage across all of the accounts in your AWS organization that have Reserve Instances or Savings Plans (SP) discount sharing activated to recommend a commitment that maximizes savings across accounts.

While purchasing at the management account level optimizes for max savings in many cases, there may be situations where you might consider purchasing SPs at the linked account level, like when you want the discounts to apply first to usage in that particular linked account. Member account recommendations are calculated at the individual account level, to maximize savings for each isolated account. If your account owns both RI and SP commitments, they will be applied in this order:

Zonal RI > Standard RI > Convertible RI > Instance Savings Plan > Compute Savings Plan

If you purchase an SP at the management account level, the savings will be applied based on highest to lowest discount percentage. SPs at the management account level look across all linked

accounts and apply the savings wherever the discount will be the highest. If you wish to restrict where the savings are applied, you can purchase a Savings Plan at the linked account level and any time that account is running eligible compute services, the discount will be applied there first. When the account is not running eligible compute services, the discount will be shared across the other linked accounts under the same management account. Discount sharing is turned on by default, but can be turned off if needed.

In a Consolidated Billing Family, Savings Plans are applied first to the owner account's usage, and then to other accounts' usage. This occurs only if you have sharing enabled. Your Savings Plans are applied to your highest savings percentage first. If there are multiple usages with equal savings percentages, Savings Plans are applied to the first usage with the lowest Savings Plans rate. Savings Plans continue to apply until there are no more remaining uses or your commitment is exhausted. Any remaining usage is charged at the On-Demand rates. You can refresh Savings Plans Recommendations in AWS Cost Management to generate new Savings Plans Recommendations at any time.

After analysing flexibility of instances, you can commit by following recommendations. Create cost modelling with analysing the workload’s short-term costs with potential different resource options, analysing AWS pricing models and aligning them with your business requirements to find out total cost of ownership and cost optimization opportunities.

Implementation steps

- **Perform a commitment discount analysis:** Using Cost Explorer in your account review the Savings Plans and Reserved Instance recommendations. Make sure you understand Saving Plan recommendations, estimate your monthly spend and estimate your monthly savings. Review recommendations at the management account level which are calculated considering usage across all of the member accounts in your AWS organization that have Reserve Instances or Savings Plans discount sharing activated for maximum savings across accounts. You can ensure you implemented the correct recommendations with the required discounts and risk by following the Well-Architected labs.

Resources

Related documents:

- How does AWS pricing work?
- Instance purchasing options

- Saving Plan Overview
- Saving Plan recommendations
- Accessing Reserved Instance recommendations
- How Savings Plans apply to your AWS usage
- Savings Plans with Consolidated Billing
- Turning on shared reserved instances and Savings Plans discounts

Related videos:

- Save up to 90% and run production workloads on Spot

Related examples:

- Well-Architected Lab: Pricing Models (Level 200)
- Well-Architected Labs: Pricing Model Analysis (Level 200)
- What should I consider before purchasing a Savings Plans?
- How can I use rolling Savings Plans to reduce commitment risk>
- When to Use Spot Instances

COST 8. How do you plan for data transfer charges?

Verify that you plan and monitor data transfer charges so that you can make architectural decisions to minimize costs. A small yet effective architectural change can drastically reduce your operational costs over time.

Best practices

- COST08-BP01 Perform data transfer modeling
- COST08-BP02 Select components to optimize data transfer cost
- COST08-BP03 Implement services to reduce data transfer costs

COST08-BP01 Perform data transfer modeling

Gather organization requirements and perform data transfer modeling of the workload and each of its components. This identifies the lowest cost point for its current data transfer requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Understand where the data transfer occurs in your workload, the cost of the transfer, and its associated benefit. This allows you to make an informed decision to modify or accept the architectural decision. For example, you may have a Multi-Availability Zone configuration where you replicate data between the Availability Zones. You model the cost of structure and decide that this is an acceptable cost (similar to paying for compute and storage in both Availability Zone) to achieve the required reliability and resilience.

Model the costs over different usage levels. Workload usage can change over time, and different services may be more cost effective at different levels.

Use AWS Cost Explorer or the AWS Cost and Usage Report (CUR) to understand and model your data transfer costs. Configure a proof of concept (PoC) or test your workload, and run a test with a realistic simulated load. You can model your costs at different workload demands.

Implementation steps

- **Calculate data transfer costs:** Use the AWS pricing pages and calculate the data transfer costs for the workload. Calculate the data transfer costs at different usage levels, for both increases and reductions in workload usage. Where there are multiple options for the workload architecture, calculate the cost for each option for comparison.
- **Link costs to outcomes:** For each data transfer cost incurred, specify the outcome that it achieves for the workload. If it is transfer between components, it may be for decoupling, if it is between Availability Zones it may be for redundancy.

Resources

Related documents:

- AWS caching solutions
- AWS Pricing
- Amazon EC2 Pricing
- Amazon VPC pricing
- Deliver content faster with Amazon CloudFront

COST08-BP02 Select components to optimize data transfer cost

All components are selected, and architecture is designed to reduce data transfer costs. This includes using components such as wide-area-network (WAN) optimization and Multi-Availability Zone (AZ) configurations

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Architecting for data transfer ensures that you minimize data transfer costs. This may involve using content delivery networks to locate data closer to users, or using dedicated network links from your premises to AWS. You can also use WAN optimization and application optimization to reduce the amount of data that is transferred between components.

Implementation steps

- **Select components for data transfer:** Using the data transfer modeling, focus on where the largest data transfer costs are or where they would be if the workload usage changes. Look for alternative architectures, or additional components that remove or reduce the need for data transfer, or lower its cost.

Resources

Related documents:

- AWS caching solutions
- Deliver content faster with Amazon CloudFront

COST08-BP03 Implement services to reduce data transfer costs

Implement services to reduce data transfer. For example, using a content delivery network (CDN) such as Amazon CloudFront to deliver content to end users, caching layers using Amazon ElastiCache, or using AWS Direct Connect instead of VPN for connectivity to AWS.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Amazon CloudFront is a global content delivery network that delivers data with low latency and high transfer speeds. It caches data at edge locations across the world, which reduces the load on

your resources. By using CloudFront, you can reduce the administrative effort in delivering content to large numbers of users globally, with minimum latency.

Direct Connect allows you to establish a dedicated network connection to AWS. This can reduce network costs, increase bandwidth, and provide a more consistent network experience than internet-based connections.

Site-to-Site VPN allows you to establish a secure and private connection between your private network and the AWS global network. It is ideal for small offices or business partners because it provides quick and easy connectivity, and it is a fully managed and elastic service.

VPC Endpoints allow connectivity between AWS services over private networking and can be used to reduce public data transfer and NAT gateways costs. Gateway VPC endpoints have no hourly charges, and support Amazon Simple Storage Service(Amazon S3) and Amazon DynamoDB. Interface VPC endpoints are provided by AWS PrivateLink and have an hourly fee and per GB usage cost.

Implementation steps

- **Implement services:** Using the data transfer modeling, look at where the largest costs and highest volume flows are. Review the AWS services and assess whether there is a service that reduces or removes the transfer, specifically networking and content delivery. Also look for caching services where there is repeated access to data, or large amounts of data.

Resources

Related documents:

- AWS Direct Connect
- AWS Explore Our Products
- AWS caching solutions
- Amazon CloudFront
- Deliver content faster with Amazon CloudFront

Manage demand and supply resources

Question

- COST 9. How do you manage demand, and supply resources?

COST 9. How do you manage demand, and supply resources?

For a workload that has balanced spend and performance, verify that everything you pay for is used and avoid significantly underutilizing instances. A skewed utilization metric in either direction has an adverse impact on your organization, in either operational costs (degraded performance due to over-utilization), or wasted AWS expenditures (due to over-provisioning).

Best practices

- COST09-BP01 Perform an analysis on the workload demand
- COST09-BP02 Implement a buffer or throttle to manage demand
- COST09-BP03 Supply resources dynamically

COST09-BP01 Perform an analysis on the workload demand

Analyze the demand of the workload over time. Verify that the analysis covers seasonal trends and accurately represents operating conditions over the full workload lifetime. Analysis effort should reflect the potential benefit, for example, time spent is proportional to the workload cost.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Know the requirements of the workload. The organization requirements should indicate the workload response times for requests. The response time can be used to determine if the demand is managed, or if the supply of resources will change to meet the demand.

The analysis should include the predictability and repeatability of the demand, the rate of change in demand, and the amount of change in demand. Ensure that the analysis is performed over a long enough period to incorporate any seasonal variance, such as end-of- month processing or holiday peaks.

Ensure that the analysis effort reflects the potential benefits of implementing scaling. Look at the expected total cost of the component, and any increases or decreases in usage and cost over the workload lifetime.

You can use AWS Cost Explorer or Amazon Quick with the AWS Cost and Usage Report (CUR) or your application logs to perform a visual analysis of workload demand.

Implementation steps

- **Analyze existing workload data:** Analyze data from the existing workload, previous versions of the workload, or predicted usage patterns. Use log files and monitoring data to gain insight on how customers use the workload. Typical metrics are the actual demand in requests per second, the times when the rate of demand changes or when it is at different levels, and the rate of change of demand. Ensure you analyze a full cycle of the workload, ensuring you collect data for any seasonal changes such as end of month or end of year events. The effort reflected in the analysis should reflect the workload characteristics. The largest effort should be placed on high-value workloads that have the largest changes in demand. The least effort should be placed on low-value workloads that have minimal changes in demand. Common metrics for value are risk, brand awareness, revenue or workload cost.
- **Forecast outside influence:** Meet with team members from across the organization that can influence or change the demand in the workload. Common teams would be sales, marketing, or business development. Work with them to know the cycles they operate within, and if there are any events that would change the demand of the workload. Forecast the workload demand with this data.

Resources

Related documents:

- AWS Auto Scaling
- AWS Instance Scheduler
- Getting started with Amazon SQS
- AWS Cost Explorer
- Amazon Quick

COST09-BP02 Implement a buffer or throttle to manage demand

Buffering and throttling modify the demand on your workload, smoothing out any peaks. Implement throttling when your clients perform retries. Implement buffering to store the request and defer processing until a later time. Verify that your throttles and buffers are designed so clients receive a response in the required time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Throttling: If the source of the demand has retry capability, then you can implement throttling. Throttling tells the source that if it cannot service the request at the current time it should try again later. The source will wait for a period of time and then re-try the request. Implementing throttling has the advantage of limiting the maximum amount of resources and costs of the workload. In AWS, you can use Amazon API Gateway to implement throttling. Refer to the Well-Architected Reliability pillar whitepaper for more details on implementing throttling.

Buffer based: Similar to throttling, a buffer defers request processing, allowing applications that run at different rates to communicate effectively. A buffer-based approach uses a queue to accept messages (units of work) from producers. Messages are read by consumers and processed, allowing the messages to run at the rate that meets the consumers' business requirements. You don't have to worry about producers having to deal with throttling issues, such as data durability and backpressure (where producers slow down because their consumer is running slowly).

In AWS, you can choose from multiple services to implement a buffering approach. Amazon Simple Queue Service(Amazon SQS) is a managed service that provides queues that allow a single consumer to read individual messages. Amazon Kinesis provides a stream that allows many consumers to read the same messages.

When architecting with a buffer-based approach, ensure that you architect your workload to service the request in the required time, and that you are able to handle duplicate requests for work.

Implementation steps

- **Analyze the client requirements:** Analyze the client requests to determine if they are capable of performing retries. For clients that cannot perform retries, buffers will need to be implemented. Analyze the overall demand, rate of change, and required response time to determine the size of throttle or buffer required.
- **Implement a buffer or throttle:** Implement a buffer or throttle in the workload. A queue such as Amazon Simple Queue Service (Amazon SQS) can provide a buffer to your workload components. Amazon API Gateway can provide throttling for your workload components.

Resources

Related documents:

- AWS Auto Scaling
- AWS Instance Scheduler
- Amazon API Gateway
- Amazon Simple Queue Service
- Getting started with Amazon SQS
- Amazon Kinesis

COST09-BP03 Supply resources dynamically

This best practice was updated with new guidance on July 13th, 2023.

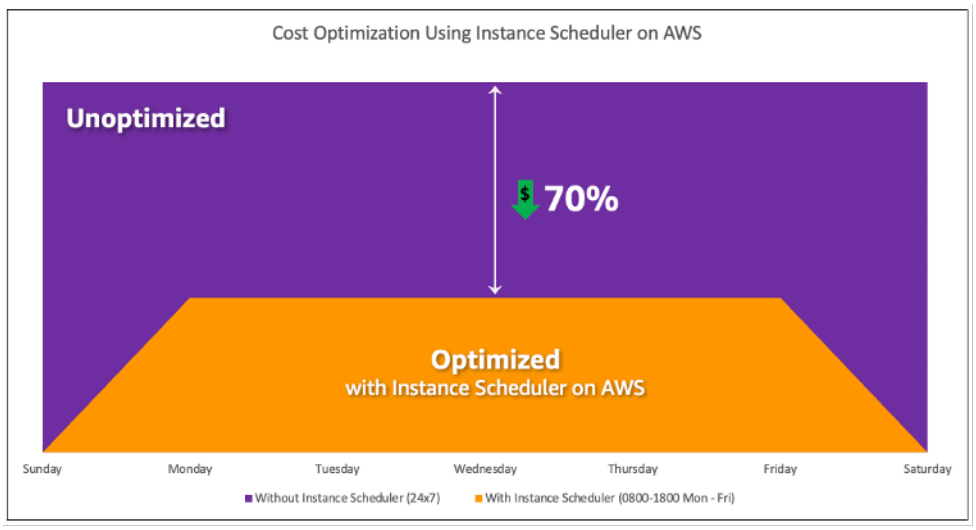
Resources are provisioned in a planned manner. This can be demand-based, such as through automatic scaling, or time-based, where demand is predictable and resources are provided based on time. These methods result in the least amount of over-provisioning or under-provisioning.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

There are several ways for AWS customers to increase the resources available to their applications and supply resources to meet the demand. One of these options is to use AWS Instance Scheduler, which automates the starting and stopping of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Relational Database Service (Amazon RDS) instances. The other option is to use AWS Auto Scaling, which allows you to automatically scale your computing resources based on the demand of your application or service. Supplying resources based on demand will allow you to pay for the resources you use only, reduce cost by launching resources when they are needed, and terminate them when they aren't.

AWS Instance Scheduler allows you to configure the stop and start of your Amazon EC2 and Amazon RDS instances at defined times so that you can meet the demand for the same resources within a consistent time pattern such as every day user access Amazon EC2 instances at eight in the morning that they don’t need after six at night. This solution helps reduce operational cost by stopping resources that are not in use and starting them when they are needed.



Cost optimization with AWS Instance Scheduler.

You can also easily configure schedules for your Amazon EC2 instances across your accounts and Regions with a simple user interface (UI) using AWS Systems Manager Quick Setup. You can schedule Amazon EC2 or Amazon RDS instances with AWS Instance Scheduler and you can stop and start existing instances. However, you cannot stop and start instances which are part of your Auto Scaling group (ASG) or that manage services such as Amazon Redshift or Amazon OpenSearch Service. Auto Scaling groups have their own scheduling for the instances in the group and these instances are created.

AWS Auto Scaling helps you adjust your capacity to maintain steady, predictable performance at the lowest possible cost to meet changing demand. It is a fully managed and free service to scale the capacity of your application that integrates with Amazon EC2 instances and Spot Fleets, Amazon ECS, Amazon DynamoDB, and Amazon Aurora. Auto Scaling provides automatic resource discovery to help find resources in your workload that can be configured, it has built-in scaling strategies to optimize performance, costs, or a balance between the two, and provides predictive scaling to assist with regularly occurring spikes.

There are multiple scaling options available to scale your Auto Scaling group:

- Maintain current instance levels at all times
- Scale manually
- Scale based on a schedule
- Scale based on demand
- Use predictive scaling

Auto Scaling policies differ and can be categorized as dynamic and scheduled scaling policies. Dynamic policies are manual or dynamic scaling which, scheduled or predictive scaling. You can use scaling policies for dynamic, scheduled, and predictive scaling. You can also use metrics and alarms from Amazon CloudWatch to trigger scaling events for your workload. We recommend you use launch templates, which allow you to access the latest features and improvements. Not all Auto Scaling features are available when you use launch configurations. For example, you cannot create an Auto Scaling group that launches both Spot and On-Demand Instances or that specifies multiple instance types. You must use a launch template to configure these features. When using launch templates, we recommended you version each one. With versioning of launch templates, you can create a subset of the full set of parameters. Then, you can reuse it to create other versions of the same launch template.

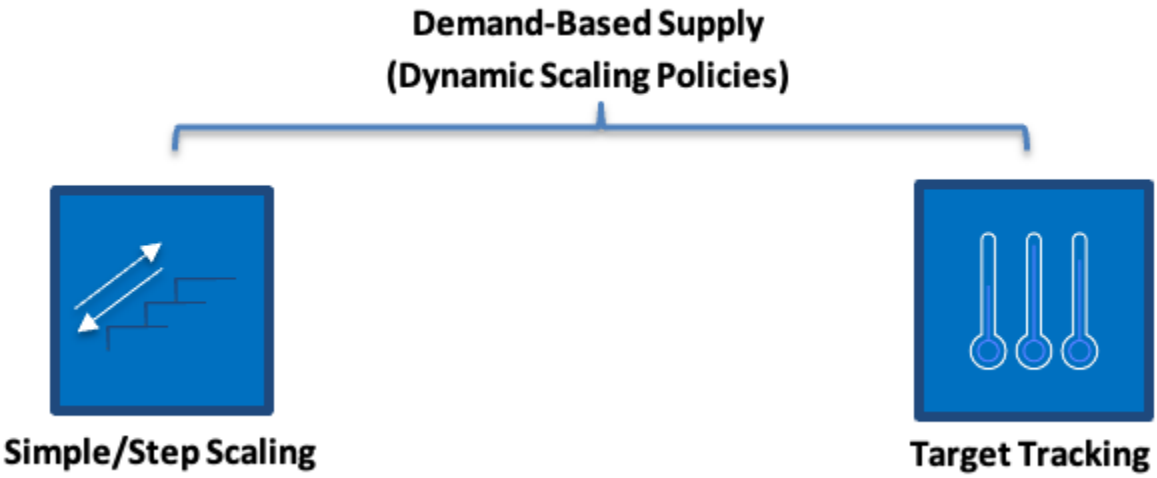
You can use AWS Auto Scaling or incorporate scaling in your code with AWS APIs or SDKs. This reduces your overall workload costs by removing the operational cost from manually making changes to your environment, and changes can be performed much faster. This also matches your workload resourcing to your demand at any time. In order to follow this best practice and supply resources dynamically for your organization, you should understand horizontal and vertical scaling in the AWS Cloud, as well as the nature of the applications running on Amazon EC2 instances. It is better for your Cloud Financial Management team to work with technical teams to follow this best practice.

Elastic Load Balancing (Elastic Load Balancing) helps you scale by distributing demand across multiple resources. With using ASG and Elastic Load Balancing, you can manage incoming requests by optimally routing traffic so that no one instance is overwhelmed in an Auto Scaling group. The requests would be distributed among all the targets of a target group in a round-robin fashion without consideration for capacity or utilization.

Typical metrics can be standard Amazon EC2 metrics, such as CPU utilization, network throughput, and Elastic Load Balancing observed request and response latency. When possible, you should use a metric that is indicative of customer experience, typically a custom metric that might originate from application code within your workload. To elaborate how to meet the demand dynamically in this document, we will group Auto Scaling into two categories as demand-based and time-based supply models and deep dive into each.

Demand-based supply: Take advantage of elasticity of the cloud to supply resources to meet changing demand by relying on near real-time demand state. For demand-based supply, use APIs or service features to programmatically vary the amount of cloud resources in your architecture. This allows you to scale components in your architecture and increase the number of resources

during demand spikes to maintain performance and decrease capacity when demand subsides to reduce costs.

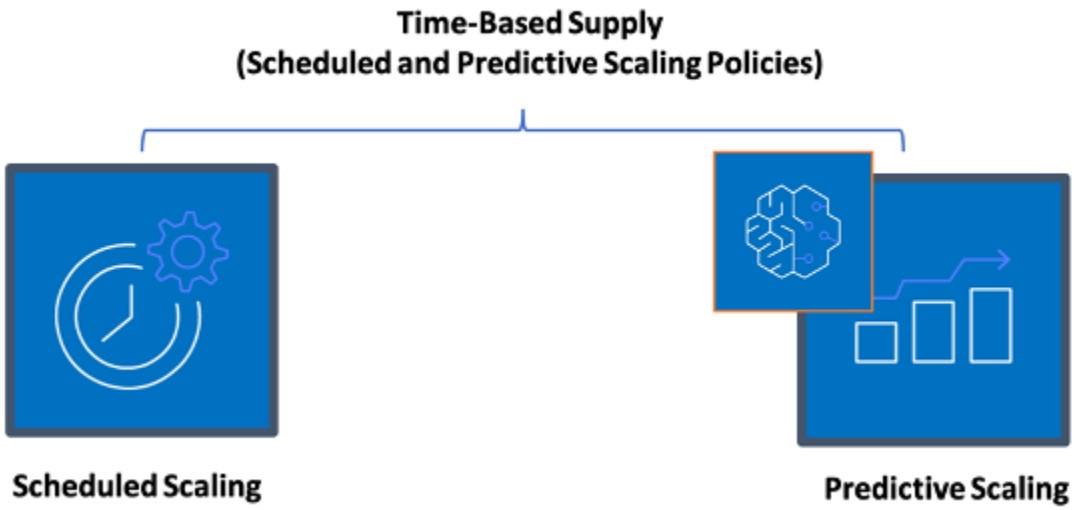


Demand-based dynamic scaling policies

- **Simple/Step Scaling:** Monitors metrics and adds/removes instances as per steps defined by the customers manually.
- **Target Tracking:** Thermostat-like control mechanism that automatically adds or removes instances to maintain metrics at a customer defined target.

When architecting with a demand-based approach keep in mind two key considerations. First, understand how quickly you must provision new resources. Second, understand that the size of margin between supply and demand will shift. You must be ready to cope with the rate of change in demand and also be ready for resource failures.

Time-based supply: A time-based approach aligns resource capacity to demand that is predictable or well-defined by time. This approach is typically not dependent upon utilization levels of the resources. A time-based approach ensures that resources are available at the specific time they are required and can be provided without any delays due to start-up procedures and system or consistency checks. Using a time-based approach, you can provide additional resources or increase capacity during busy periods.



Time-based scaling policies

You can use scheduled or predictive auto scaling to implement a time-based approach. Workloads can be scheduled to scale out or in at defined times (for example, the start of business hours), making resources available when users arrive or demand increases. Predictive scaling uses patterns to scale out while scheduled scaling uses pre-defined times to scale out. You can also use attribute-based instance type selection (ABS) strategy in Auto Scaling groups, which lets you express your instance requirements as a set of attributes, such as vCPU, memory, and storage. This also allows you to automatically use newer generation instance types when they are released and access a broader range of capacity with Amazon EC2 Spot Instances. Amazon EC2 Fleet and Amazon EC2 Auto Scaling select and launch instances that fit the specified attributes, removing the need to manually pick instance types.

You can also leverage the AWS APIs and SDKs and AWS CloudFormation to automatically provision and decommission entire environments as you need them. This approach is well suited for development or test environments that run only in defined business hours or periods of time. You can use APIs to scale the size of resources within an environment (vertical scaling). For example, you could scale up a production workload by changing the instance size or class. This can be achieved by stopping and starting the instance and selecting the different instance size or class. This technique can also be applied to other resources, such as Amazon EBS Elastic Volumes, which can be modified to increase size, adjust performance (IOPS) or change the volume type while in use.

When architecting with a time-based approach keep in mind two key considerations. First, how consistent is the usage pattern? Second, what is the impact if the pattern changes? You can increase the accuracy of predictions by monitoring your workloads and by using business intelligence. If you see significant changes in the usage pattern, you can adjust the times to ensure that coverage is provided.

Implementation steps

- **Configure scheduled scaling:** For predictable changes in demand, time-based scaling can provide the correct number of resources in a timely manner. It is also useful if resource creation and configuration is not fast enough to respond to changes on demand. Using the workload analysis configure scheduled scaling using AWS Auto Scaling. To configure time-based scheduling, you can use predictive scaling of scheduled scaling to increase the number of Amazon EC2 instances in your Auto Scaling groups in advance according to expected or predictable load changes.
- **Configure predictive scaling:** Predictive scaling allows you to increase the number of Amazon EC2 instances in your Auto Scaling group in advance of daily and weekly patterns in traffic flows. If you have regular traffic spikes and applications that take a long time to start, you should consider using predictive scaling. Predictive scaling can help you scale faster by initializing capacity before projected load compared to dynamic scaling alone, which is reactive in nature. For example, if users start using your workload with the start of the business hours and don't use after hours, then predictive scaling can add capacity before the business hours which eliminates delay of dynamic scaling to react to changing traffic.
- **Configure dynamic automatic scaling:** To configure scaling based on active workload metrics, use Auto Scaling. Use the analysis and configure Auto Scaling to launch on the correct resource levels, and verify that the workload scales in the required time. You can launch and automatically scale a fleet of On-Demand Instances and Spot Instances within a single Auto Scaling group. In addition to receiving discounts for using Spot Instances, you can use Reserved Instances or a Savings Plan to receive discounted rates of the regular On-Demand Instance pricing. All of these factors combined help you to optimize your cost savings for Amazon EC2 instances and help you get the desired scale and performance for your application.

Resources

Related documents:

- AWS Auto Scaling

- AWS Instance Scheduler
- Scale the size of your Auto Scaling group
- Getting Started with Amazon EC2 Auto Scaling
- Getting started with Amazon SQS
- Scheduled Scaling for Amazon EC2 Auto Scaling
- Predictive scaling for Amazon EC2 Auto Scaling

Related videos:

- Target Tracking Scaling Policies for Auto Scaling
- AWS Instance Scheduler

Related examples:

- Attribute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet
- Optimizing Amazon Elastic Container Service for cost using scheduled scaling
- Predictive Scaling with Amazon EC2 Auto Scaling
- How do I use Instance Scheduler with CloudFormation to schedule Amazon EC2 instances?

Optimize over time

Questions

- COST 10. How do you evaluate new services?
- COST 11. How do you evaluate the cost of effort?

COST 10. How do you evaluate new services?

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

Best practices

- COST10-BP01 Develop a workload review process
- COST10-BP02 Review and analyze this workload regularly

COST10-BP01 Develop a workload review process

Develop a process that defines the criteria and process for workload review. The review effort should reflect potential benefit. For example, core workloads or workloads with a value of over ten percent of the bill are reviewed quarterly or every six months, while workloads below ten percent are reviewed annually.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To have the most cost-efficient workload, you must regularly review the workload to know if there are opportunities to implement new services, features, and components. To achieve overall lower costs the process must be proportional to the potential amount of savings. For example, workloads that are 50% of your overall spend should be reviewed more regularly, and more thoroughly, than workloads that are five percent of your overall spend. Factor in any external factors or volatility. If the workload services a specific geography or market segment, and change in that area is predicted, more frequent reviews could lead to cost savings. Another factor in review is the effort to implement changes. If there are significant costs in testing and validating changes, reviews should be less frequent.

Factor in the long-term cost of maintaining outdated and legacy, components and resources and the inability to implement new features into them. The current cost of testing and validation may exceed the proposed benefit. However, over time, the cost of making the change may significantly increase as the gap between the workload and the current technologies increases, resulting in even larger costs. For example, the cost of moving to a new programming language may not currently be cost effective. However, in five years time, the cost of people skilled in that language may increase, and due to workload growth, you would be moving an even larger system to the new language, requiring even more effort than previously.

Break down your workload into components, assign the cost of the component (an estimate is sufficient), and then list the factors (for example, effort and external markets) next to each component. Use these indicators to determine a review frequency for each workload. For example, you may have webservers as a high cost, low change effort, and high external factors, resulting in high frequency of review. A central database may be medium cost, high change effort, and low external factors, resulting in a medium frequency of review.

Define a process to evaluate new services, design patterns, resource types, and configurations to optimize your workload cost as they become available. Similar to performance pillar review and

reliability pillar review processes, identify, validate, and prioritize optimization and improvement activities and issue remediation and incorporate this into your backlog.

Implementation steps

- **Define review frequency:** Define how frequently the workload and its components should be reviewed. Allocate time and resources to continual improvement and review frequency to improve the efficiency and optimization of your workload. This is a combination of factors and may differ from workload to workload within your organization and between components in the workload. Common factors include the importance to the organization measured in terms of revenue or brand, the total cost of running the workload (including operation and resource costs), the complexity of the workload, how easy is it to implement a change, any software licensing agreements, and if a change would incur significant increases in licensing costs due to punitive licensing. Components can be defined functionally or technically, such as web servers and databases, or compute and storage resources. Balance the factors accordingly and develop a period for the workload and its components. You may decide to review the full workload every 18 months, review the web servers every six months, the database every 12 months, compute and short-term storage every six months, and long-term storage every 12 months.
- **Define review thoroughness:** Define how much effort is spent on the review of the workload or workload components. Similar to the review frequency, this is a balance of multiple factors. Evaluate and prioritize opportunities for improvement to focus efforts where they provide the greatest benefits while estimating how much effort is required for these activities. If the expected outcomes do not satisfy the goals, and required effort costs more, then iterate using alternative courses of action. Your review processes should include dedicated time and resources to make continuous incremental improvements possible. As an example, you may decide to spend one week of analysis on the database component, one week of analysis for compute resources, and four hours for storage reviews.

Resources

Related documents:

- AWS News Blog
- Types of Cloud Computing
- What's New with AWS

Related examples:

- AWS Support Proactive Services
- Regular workload reviews for SAP workloads

COST10-BP02 Review and analyze this workload regularly

Existing workloads are regularly reviewed based on each defined process to find out if new services can be adopted, existing services can be replaced, or workloads can be re-architected.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS is constantly adding new features so you can experiment and innovate faster with the latest technology. AWS What's New details how AWS is doing this and provides a quick overview of AWS services, features, and Regional expansion announcements as they are released. You can dive deeper into the launches that have been announced and use them for your review and analyze of your existing workloads. To realize the benefits of new AWS services and features, you review on your workloads and implement new services and features as required. This means you may need to replace existing services you use for your workload, or modernize your workload to adopt these new AWS services. For example, you might review your workloads and replace the messaging component with Amazon Simple Email Service. This removes the cost of operating and maintaining a fleet of instances, while providing all the functionality at a reduced cost.

To analyze your workload and highlight potential opportunities, you should consider not only new services but also new ways of building solutions. Review the This is My Architecture videos on AWS to learn about other customers' architecture designs, their challenges and their solutions. Check the All-In series to find out real world applications of AWS services and customer stories. You can also watch the Back to Basics video series that explains, examines, and breaks down basic cloud architecture pattern best practices. Another source is How to Build This videos, which are designed to assist people with big ideas on how to bring their minimum viable product (MVP) to life using AWS services. It is a way for builders from all over the world who have a strong idea to gain architectural guidance from experienced AWS Solutions Architects. Finally, you can review the Getting Started resource materials, which has step by step tutorials.

Before starting your review process, follow your business' requirements for the workload, security and data privacy requirements in order to use specific service or Region and performance requirements while following your agreed review process.

Implementation steps

- **Regularly review the workload:** Using your defined process, perform reviews with the frequency specified. Verify that you spend the correct amount of effort on each component. This process would be similar to the initial design process where you selected services for cost optimization. Analyze the services and the benefits they would bring, this time factor in the cost of making the change, not just the long-term benefits.
- **Implement new services:** If the outcome of the analysis is to implement changes, first perform a baseline of the workload to know the current cost for each output. Implement the changes, then perform an analysis to confirm the new cost for each output.

Resources

Related documents:

- AWS News Blog
- What's New with AWS
- AWS Documentation
- AWS Getting Started
- AWS General Resources

Related videos:

- AWS - This is My Architecture
- AWS - Back to Basics
- AWS - All-In series
- How to Build This

COST 11. How do you evaluate the cost of effort?

Best practices

- COST11-BP01 Perform automations for operations

COST11-BP01 Perform automations for operations

Evaluate cost of effort for operations on cloud. Quantify reduction in time and effort for admin tasks, deployment and other operations using automation. Evaluate the required time and cost for the effort of operations and automate admin tasks to reduce the human effort where possible.

Level of risk exposed if this best practice is not established: Low

Automating operations improves consistency and scalability, provides more visibility, reliability, and flexibility, reduces costs, and accelerates innovation by freeing up human resources and improving metrics. It reduces the frequency of manual tasks, improves efficiency, and benefits enterprises by delivering a consistent and reliable experience when deploying, administering, or operating workloads. You can free up infrastructure resources from manual operational tasks and use them for higher value tasks and innovations, thereby improving business outcomes. Enterprises require a proven, tested way to manage their workloads in the cloud. That solution must be secure, fast, and cost effective, with minimum risk and maximum reliability.

Start by prioritizing your operations based on required effort by looking at overall operations cost in the cloud. For example, how long does it take to deploy new resources in the cloud, make optimization changes to existing ones, or implement necessary configurations? Look at the total cost of human actions by factoring in cost of operations and management. Prioritize automations for admin tasks to reduce the human effort. Review effort should reflect the potential benefit. For example, time spent performing tasks manually as opposed to automatically. Prioritize automating repetitive, high value activities. Activities that pose a higher risk of human error are typically the better place to start automating as the risk often poses an unwanted additional operational cost (like operations team working extra hours).

Using AWS services, tools, or third-party products, you can choose which AWS automations to implement and customize for your specific requirements. The following table shows some of the core operation functions and capabilities you can achieve with AWS services to automate administration and operation:

- AWS Audit Manager: Continually audit your AWS usage to simplify risk and compliance assessment
- AWS Backup: Centrally manage and automate data protection.
- AWS Config: Configure compute resources, assess, audit, and evaluate configurations and resource inventory.
- AWS CloudFormation: Launch highly available resources with infrastructure as code.

- AWS CloudTrail: IT change management, compliance, and control.
- Amazon EventBridge: Schedule events and launch AWS Lambda to take action.
- AWS Lambda: Automate repetitive processes by initiating them with events or by running them on a fixed schedule with Amazon EventBridge.
- AWS Systems Manager: Start and stop workloads, patch operating systems,a automate configuration, and ongoing management.
- AWS Step Functions: Schedule jobs and automate workflows.
- AWS Service Catalog: Template consumption and infrastructure as code with compliance and control.

Consider the time savings that will allow your team to focus on retiring technical debt, innovation, and value-adding features. For example, you might need to lift and shift your on-premises environment into the cloud as rapidly as possible and optimize later. It is worth exploring the savings you could realize by using fully managed services by AWS that remove or reduce license costs such as Amazon Relational Database Service, Amazon EMR, Amazon WorkSpaces, and Amazon SageMaker AI. Managed services remove the operational and administrative burden of maintaining a service, which allows you to focus on innovation. Additionally, because managed services operate at cloud scale, they can offer a lower cost per transaction or service.

If you would like to adopt automations immediately with using AWS products and service and if don't have skills in your organization, reach out to AWS Managed Services (AMS), AWS Professional Services, or AWS Partners to increase adoption of automation and improve your operational excellence in the cloud.

AWS Managed Services (AMS) is a service that operates AWS infrastructure on behalf of enterprise customers and partners. It provides a secure and compliant environment that you can deploy your workloads onto. AMS uses enterprise cloud operating models with automation to allow you to meet your organization requirements, move into the cloud faster, and reduce your on-going management costs.

AWS Professional Services can also help you achieve your desired business outcomes and automate operations with AWS. AWS Professional Services provides global specialty practices to support your efforts in focused areas of enterprise cloud computing. Specialty practices deliver targeted guidance through best practices, frameworks, tools, and services across solution, technology, and industry subject areas. They help customers to deploy automated, robust, agile IT operations, and governance capabilities optimized for the cloud center.

Implementation steps

- **Build once and deploy many:** Use infrastructure-as-code such as AWS CloudFormation, AWS SDK, or AWS Command Line Interface (AWS CLI) to deploy once and use many times for same environment or for disaster recovery scenarios. Tag while deploying to track your consumption as defined in other best practices. Use AWS Launch Wizard to reduce the time to deploy many popular enterprise workloads. AWS Launch Wizard guides you through the sizing, configuration, and deployment of enterprise workloads following AWS best practices. You can also use the AWS Service Catalog, which helps you create and manage infrastructure-as-code approved templates for use on AWS so anyone can discover approved, self-service cloud resources.
- **Automate operations:** Run routine operations automatically without human intervention. Using AWS services and tools, you can choose which AWS automations to implement and customize for your specific requirements. For example, use EC2 Image Builder for building, testing, and deployment of virtual machine and container images for use on AWS or on-premises. If your desired action cannot be done with AWS services or you need more complex actions with filtering resources, then automate your operations with using AWS CLI or AWS SDK tools. AWS CLI provides the ability to automate the entire process of controlling and managing AWS services via scripts without using the AWS Console. Select your preferred AWS SDKs to interact with AWS services. For other code examples, see AWS SDK Code examples repository.

Resources

Related documents:

- Modernizing operations in the AWS Cloud
- AWS Services for Automation
- AWS Systems Manager Automation
- AWS automations for SAP administration and operations
- AWS Managed Services
- AWS Professional Services
- Infrastructure and automation

Related examples:

- Reinventing automated operations (Part I)
- Reinventing automated operations (Part II)

- AWS automations for SAP administration and operations
- IT Automations with AWS Lambda
- AWS Code Examples Repository
- AWS Samples

Sustainability

The Sustainability pillar includes understanding the impacts of the services used, quantifying impacts through the entire workload lifecycle, and applying design principles and best practices to reduce these impacts when building cloud workloads. You can find prescriptive guidance on implementation in the Sustainability Pillar whitepaper.

Best practice areas

- Region selection
- Alignment to demand
- Software and architecture
- Data
- Hardware and services
- Process and culture

Region selection

Question

- SUS 1 How do you select Regions for your workload?

SUS 1 How do you select Regions for your workload?

The choice of Region for your workload significantly affects its KPIs, including performance, cost, and carbon footprint. To effectively improve these KPIs, you should choose Regions for your workloads based on both business requirements and sustainability goals.

Best practices

- SUS01-BP01 Choose Region based on both business requirements and sustainability goals

SUS01-BP01 Choose Region based on both business requirements and sustainability goals

Choose a Region for your workload based on both your business requirements and sustainability goals to optimize its KPIs, including performance, cost, and carbon footprint.

Common anti-patterns:

- You select the workload's Region based on your own location.
- You consolidate all workload resources into one geographic location.

Benefits of establishing this best practice: Placing a workload close to Amazon renewable energy projects or Regions with low published carbon intensity can help to lower the carbon footprint of a cloud workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The AWS Cloud is a constantly expanding network of Regions and points of presence (PoP), with a global network infrastructure linking them together. The choice of Region for your workload significantly affects its KPIs, including performance, cost, and carbon footprint. To effectively improve these KPIs, you should choose Regions for your workload based on both your business requirements and sustainability goals.

Implementation steps

- Follow these steps to assess and shortlist potential Regions for your workload based on your business requirements including compliance, available features, cost, and latency:
  - Confirm that these Regions are compliant, based on your required local regulations.
  - Use the AWS Regional Services Lists to check if the Regions have the services and features you need to run your workload.
  - Calculate the cost of the workload on each Region using the AWS Pricing Calculator.
  - Test the network latency between your end user locations and each AWS Region.
- Choose Regions near Amazon renewable energy projects and Regions where the grid has a published carbon intensity that is lower than other locations (or Regions).
  - Identify your relevant sustainability guidelines to track and compare year-to-year carbon emissions based on Greenhouse Gas Protocol (market-based and location based methods).

- Choose region based on method you use to track carbon emissions. For more detail on choosing a Region based on your sustainability guidelines, see How to select a Region for your workload based on sustainability goals.

Resources

Related documents:

- Understanding your carbon emission estimations
- Amazon Around the Globe
- Renewable Energy Methodology
- What to Consider when Selecting a Region for your Workloads

Related videos:

- Architecting sustainably and reducing your AWS carbon footprint

Alignment to demand

Question

- SUS 2 How do you align cloud resources to your demand?

SUS 2 How do you align cloud resources to your demand?

The way users and applications consume your workloads and other resources can help you identify improvements to meet sustainability goals. Scale infrastructure to continually match demand and verify that you use only the minimum resources required to support your users. Align service levels to customer needs. Position resources to limit the network required for users and applications to consume them. Remove unused assets. Provide your team members with devices that support their needs and minimize their sustainability impact.

Best practices

- SUS02-BP01 Scale workload infrastructure dynamically
- SUS02-BP02 Align SLAs with sustainability goals
- SUS02-BP03 Stop the creation and maintenance of unused assets

- SUS02-BP04 Optimize geographic placement of workloads based on their networking requirements
- SUS02-BP05 Optimize team member resources for activities performed
- SUS02-BP06 Implement buffering or throttling to flatten the demand curve

SUS02-BP01 Scale workload infrastructure dynamically

Use elasticity of the cloud and scale your infrastructure dynamically to match supply of cloud resources to demand and avoid overprovisioned capacity in your workload.

Common anti-patterns:

- You do not scale your infrastructure with user load.
- You manually scale your infrastructure all the time.
- You leave increased capacity after a scaling event instead of scaling back down.

Benefits of establishing this best practice: Configuring and testing workload elasticity help to efficiently match supply of cloud resources to demand and avoid overprovisioned capacity. You can take advantage of elasticity in the cloud to automatically scale capacity during and after demand spikes to make sure you are only using the right number of resources needed to meet your business requirements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The cloud provides the flexibility to expand or reduce your resources dynamically through a variety of mechanisms to meet changes in demand. Optimally matching supply to demand delivers the lowest environmental impact for a workload.

Demand can be fixed or variable, requiring metrics and automation to make sure that management does not become burdensome. Applications can scale vertically (up or down) by modifying the instance size, horizontally (in or out) by modifying the number of instances, or a combination of both.

You can use a number of different approaches to match supply of resources with demand.

- **Target-tracking approach:** Monitor your scaling metric and automatically increase or decrease capacity as you need it.

- **Predictive scaling:** Scale in anticipation of daily and weekly trends.
- **Schedule-based approach:** Set your own scaling schedule according to predictable load changes.
- **Service scaling:** Pick services (like serverless) that are natively scaling by design or provide auto scaling as a feature.

Identify periods of low or no utilization and scale resources to remove excess capacity and improve efficiency.

Implementation steps

- Elasticity matches the supply of resources you have against the demand for those resources. Instances, containers, and functions provide mechanisms for elasticity, either in combination with automatic scaling or as a feature of the service. AWS provides a range of auto scaling mechanisms to ensure that workloads can scale down quickly and easily during periods of low user load. Here are some examples of auto scaling mechanisms:

Auto scaling mechanism	Where to use
Amazon EC2 Auto Scaling	Use to verify you have the correct number of Amazon EC2 instances available to handle the user load for your application.
Application Auto Scaling	Use to automatically scale the resources for individual AWS services beyond Amazon EC2, such as Lambda functions or Amazon Elastic Container Service (Amazon ECS) services.
Kubernetes Cluster Autoscaler	Use to automatically scale Kubernetes clusters on AWS.

- Scaling is often discussed related to compute services like Amazon EC2 instances or AWS Lambda functions. Consider the configuration of non-compute services like Amazon DynamoDB read and write capacity units or Amazon Kinesis Data Streams shards to match the demand.
- Verify that the metrics for scaling up or down are validated against the type of workload being deployed. If you are deploying a video transcoding application, 100% CPU utilization is expected and should not be your primary metric. You can use a customized metric (such as memory

utilization) for your scaling policy if required. To choose the right metrics, consider the following guidance for Amazon EC2:

  - The metric should be a valid utilization metric and describe how busy an instance is.
  - The metric value must increase or decrease proportionally to the number of instances in the Auto Scaling group.
- Use dynamic scaling instead of manual scaling for your Auto Scaling group. We also recommend that you use target tracking scaling policies in your dynamic scaling.
- Verify that workload deployments can handle both scale-out and scale-in events. Create test scenarios for scale-in events to verify that the workload behaves as expected and does not affect the user experience (like losing sticky sessions). You can use Activity history to verify a scaling activity for an Auto Scaling group.
- Evaluate your workload for predictable patterns and proactively scale as you anticipate predicted and planned changes in demand. With predictive scaling, you can eliminate the need to overprovision capacity. For more detail, see Predictive Scaling with Amazon EC2 Auto Scaling.

Resources

Related documents:

- Getting Started with Amazon EC2 Auto Scaling
- Predictive Scaling for EC2, Powered by Machine Learning
- Analyze user behavior using Amazon OpenSearch Service, Amazon Data Firehose and Kibana
- What is Amazon CloudWatch?
- Monitoring DB load with Performance Insights on Amazon RDS
- Introducing Native Support for Predictive Scaling with Amazon EC2 Auto Scaling
- Introducing Karpenter - An Open-Source, High-Performance Kubernetes Cluster Autoscaler
- Deep Dive on Amazon ECS Cluster Auto Scaling

Related videos:

- Build a cost-, energy-, and resource-efficient compute environment
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)

Related examples:

- Lab: Amazon EC2 Auto Scaling Group Examples
- Lab: Implement Autoscaling with Karpenter

SUS02-BP02 Align SLAs with sustainability goals

Review and optimize workload service-level agreements (SLA) based on your sustainability goals to minimize the resources required to support your workload while continuing to meet business needs.

Common anti-patterns:

- Workload SLAs are unknown or ambiguous.
- You define your SLA just for availability and performance.
- You use the same design pattern (like Multi-AZ architecture) for all your workloads.

Benefits of establishing this best practice: Aligning SLAs with sustainability goals leads to optimal resource usage while meeting business needs.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

SLAs define the level of service expected from a cloud workload, such as response time, availability, and data retention. They influence the architecture, resource usage, and environmental impact of a cloud workload. At a regular cadence, review SLAs and make trade-offs that significantly reduce resource usage in exchange for acceptable decreases in service levels.

Implementation steps

- Define or redesign SLAs that support your sustainability goals while meeting your business requirements, not exceeding them.
- Make trade-offs that significantly reduce sustainability impacts in exchange for acceptable decreases in service levels.
  - **Sustainability and reliability:** Highly available workloads tend to consume more resources.
  - **Sustainability and performance:** Using more resources to boost performance could have a higher environmental impact.
  - **Sustainability and security:** Overly secure workloads could have a higher environmental impact.

- Use design patterns such as microservices on AWS that prioritize business-critical functions and allow lower service levels (such as response time or recovery time objectives) for non-critical functions.

Resources

Related documents:

- AWS Service Level Agreements (SLAs)
- Importance of Service Level Agreement for SaaS Providers

Related videos:

- Delivering sustainable, high-performing architectures
- Build a cost-, energy-, and resource-efficient compute environment

SUS02-BP03 Stop the creation and maintenance of unused assets

Decommission unused assets in your workload to reduce the number of cloud resources required to support your demand and minimize waste.

Common anti-patterns:

- You do not analyze your application for assets that are redundant or no longer required.
- You do not remove assets that are redundant or no longer required.

Benefits of establishing this best practice: Removing unused assets frees resources and improves the overall efficiency of the workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Unused assets consume cloud resources like storage space and compute power. By identifying and eliminating these assets, you can free up these resources, resulting in a more efficient cloud architecture. Perform regular analysis on application assets such as pre-compiled reports, datasets, static images, and asset access patterns to identify redundancy, underutilization, and potential

decommission targets. Remove those redundant assets to reduce the resource waste in your workload.

Implementation steps

- Use monitoring tools to identify static assets that are no longer required.
- Before removing any asset, evaluate the impact of removing it on the architecture.
- Develop a plan and remove assets that are no longer required.
- Consolidate overlapping generated assets to remove redundant processing.
- Update your applications to no longer produce and store assets that are not required.
- Instruct third parties to stop producing and storing assets managed on your behalf that are no longer required.
- Instruct third parties to consolidate redundant assets produced on your behalf.
- Regularly review your workload to identify and remove unused assets.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part II: Storage
- How do I terminate active resources that I no longer need on my AWS account?

Related videos:

- How do I check for and then remove active resources that I no longer need on my AWS account?

SUS02-BP04 Optimize geographic placement of workloads based on their networking requirements

This best practice was updated with new guidance on July 13th, 2023.

Select cloud location and services for your workload that reduce the distance network traffic must travel and decrease the total network resources required to support your workload.

Common anti-patterns:

- You select the workload's Region based on your own location.
- You consolidate all workload resources into one geographic location.
- All traffic flows through your existing data centers.

Benefits of establishing this best practice: Placing a workload close to its users provides the lowest latency while decreasing data movement across the network and reducing environmental impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The AWS Cloud infrastructure is built around location options such as Regions, Availability Zones, placement groups, and edge locations such as AWS Outposts and AWS Local Zones. These location options are responsible for maintaining connectivity between application components, cloud services, edge networks and on-premises data centers.

Analyze the network access patterns in your workload to identify how to use these cloud location options and reduce the distance network traffic must travel.

Implementation steps

- Analyze network access patterns in your workload to identify how users use your application.
  - Use monitoring tools, such as Amazon CloudWatch and AWS CloudTrail, to gather data on network activities.
  - Analyze the data to identify the network access pattern.
- Select the Regions for your workload deployment based on the following key elements:
  - **Your Sustainability goal:** as explained in Region selection.
  - **Where your data is located:** For data-heavy applications (such as big data and machine learning), application code should run as close to the data as possible.
  - **Where your users are located:** For user-facing applications, choose a Region (or Regions) close to your workload's users.
  - **Other constraints:** Consider constraints such as cost and compliance as explained in What to Consider when Selecting a Region for your Workloads.
- Use local caching or AWS Caching Solutions for frequently used assets to improve performance, reduce data movement, and lower environmental impact.

Service	When to use
Amazon CloudFront	Use to cache static content such as images, scripts, and videos, as well as dynamic content such as API responses or web applications.
Amazon ElastiCache	Use to cache content for web applications.
DynamoDB Accelerator	Use to add in-memory acceleration to your DynamoDB tables.

- Use services that can help you run code closer to users of your workload:

Service	When to use
Lambda@Edge	Use for compute-heavy operations that are initiated when objects are not in the cache.
Amazon CloudFront Functions	Use for simple use cases like HTTP(s) request or response manipulations that can be initiated by short-lived functions.
AWS IoT Greengrass	Use to run local compute, messaging, and data caching for connected devices.

- Use connection pooling to allow for connection reuse and reduce required resources.
- Use distributed data stores that don't rely on persistent connections and synchronous updates for consistency to serve regional populations.
- Replace pre-provisioned static network capacity with shared dynamic capacity, and share the sustainability impact of network capacity with other subscribers.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking

- Amazon ElastiCache Documentation
- What is Amazon CloudFront?
- Amazon CloudFront Key Features

Related videos:

- Demystifying data transfer on AWS
- Scaling network performance on next-gen Amazon EC2 instances

Related examples:

- AWS Networking Workshops
- Architecting for sustainability - Minimize data movement across networks

SUS02-BP05 Optimize team member resources for activities performed

Optimize resources provided to team members to minimize the environmental sustainability impact while supporting their needs.

Common anti-patterns:

- You ignore the impact of devices used by your team members on the overall efficiency of your cloud application.
- You manually manage and update resources used by team members.

Benefits of establishing this best practice: Optimizing team member resources improves the overall efficiency of cloud-enabled applications.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Understand the resources your team members use to consume your services, their expected lifecycle, and the financial and sustainability impact. Implement strategies to optimize these resources. For example, perform complex operations, such as rendering and compilation, on highly utilized scalable infrastructure instead of on underutilized high-powered single-user systems.

Implementation steps

- Provision workstations and other devices to align with how they're used.
- Use virtual desktops and application streaming to limit upgrade and device requirements.
- Move processor or memory-intensive tasks to the cloud to use its elasticity.
- Evaluate the impact of processes and systems on your device lifecycle, and select solutions that minimize the requirement for device replacement while satisfying business requirements.
- Implement remote management for devices to reduce required business travel.
  - AWS Systems Manager Fleet Manager is a unified user interface (UI) experience that helps you remotely manage your nodes running on AWS or on premises.

Resources

Related documents:

- What is Amazon WorkSpaces?
- Cost Optimizer for Amazon WorkSpaces
- Amazon AppStream 2.0 Documentation
- NICE DCV

Related videos:

- Managing cost for Amazon WorkSpaces on AWS

SUS02-BP06 Implement buffering or throttling to flatten the demand curve

Buffering and throttling flatten the demand curve and reduce the provisioned capacity required for your workload.

Common anti-patterns:

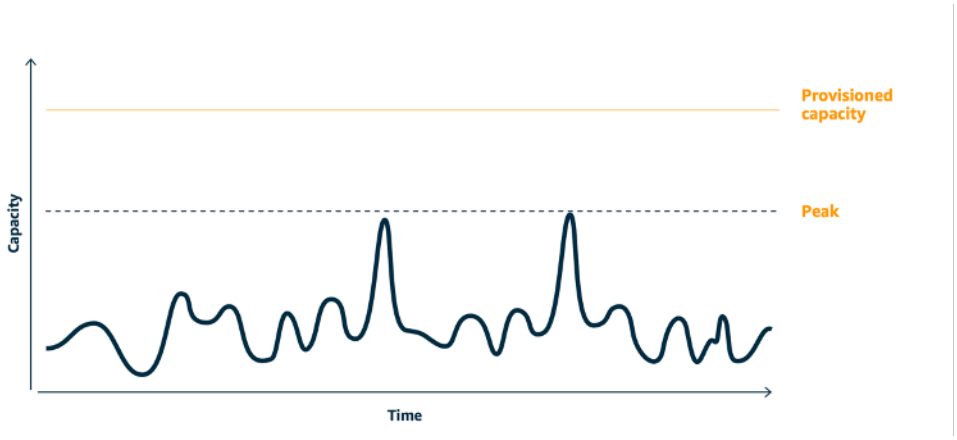
- You process the client requests immediately while it is not needed.
- You do not analyze the requirements for client requests.

Benefits of establishing this best practice: Flattening the demand curve reduce the required provisioned capacity for the workload. Reducing the provisioned capacity means less energy consumption and less environmental impact.

Level of risk exposed if this best practice is not established: Low

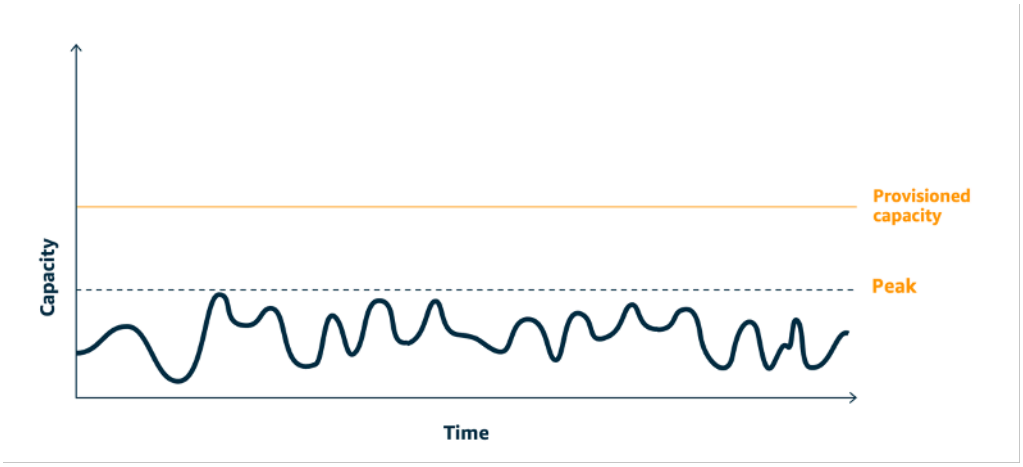
Implementation guidance

Flattening the workload demand curve can help you to reduce the provisioned capacity for a workload and reduce its environmental impact. Assume a workload with the demand curve shown in below figure. This workload has two peaks, and to handle those peaks, the resource capacity as shown by orange line is provisioned. The resources and energy used for this workload is not indicated by the area under the demand curve, but the area under the provisioned capacity line, as provisioned capacity is needed to handle those two peaks.



Demand curve with two distinct peaks that require high provisioned capacity.

You can use buffering or throttling to modify the demand curve and smooth out the peaks, which means less provisioned capacity and less energy consumed. Implement throttling when your clients can perform retries. Implement buffering to store the request and defer processing until a later time.



Throttling's effect on the demand curve and provisioned capacity.

Implementation steps

- Analyze the client requests to determine how to respond to them. Questions to consider include:
  - Can this request be processed asynchronously?
  - Does the client have retry capability?
- If the client has retry capability, then you can implement throttling, which tells the source that if it cannot service the request at the current time, it should try again later.
  - You can use Amazon API Gateway to implement throttling.
- For clients that cannot perform retries, a buffer needs to be implemented to flatten the demand curve. A buffer defers request processing, allowing applications that run at different rates to communicate effectively. A buffer-based approach uses a queue or a stream to accept messages from producers. Messages are read by consumers and processed, allowing the messages to run at the rate that meets the consumers' business requirements.
  - Amazon Simple Queue Service (Amazon SQS) is a managed service that provides queues that allow a single consumer to read individual messages.
  - Amazon Kinesis provides a stream that allows many consumers to read the same messages.
- Analyze the overall demand, rate of change, and required response time to right size the throttle or buffer required.

Resources

Related documents:

- Getting started with Amazon SQS
- Application integration Using Queues and Messages

Related videos:

- Choosing the Right Messaging Service for Your Distributed App

Software and architecture

Question

- SUS 3 How do you take advantage of software and architecture patterns to support your sustainability goals?

SUS 3 How do you take advantage of software and architecture patterns to support your sustainability goals?

Implement patterns for performing load smoothing and maintaining consistent high utilization of deployed resources to minimize the resources consumed. Components might become idle from lack of use because of changes in user behavior over time. Revise patterns and architecture to consolidate under-utilized components to increase overall utilization. Retire components that are no longer required. Understand the performance of your workload components, and optimize the components that consume the most resources. Be aware of the devices that your customers use to access your services, and implement patterns to minimize the need for device upgrades.

Best practices

- SUS03-BP01 Optimize software and architecture for asynchronous and scheduled jobs
- SUS03-BP02 Remove or refactor workload components with low or no use
- SUS03-BP03 Optimize areas of code that consume the most time or resources
- SUS03-BP04 Optimize impact on devices and equipment
- SUS03-BP05 Use software patterns and architectures that best support data access and storage patterns

SUS03-BP01 Optimize software and architecture for asynchronous and scheduled jobs

Use efficient software and architecture patterns such as queue-driven to maintain consistent high utilization of deployed resources.

Common anti-patterns:

- You overprovision the resources in your cloud workload to meet unforeseen spikes in demand.
- Your architecture does not decouple senders and receivers of asynchronous messages by a messaging component.

Benefits of establishing this best practice:

- Efficient software and architecture patterns minimize the unused resources in your workload and improve the overall efficiency.
- You can scale the processing independently of the receiving of asynchronous messages.
- Through a messaging component, you have relaxed availability requirements that you can meet with fewer resources.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Use efficient architecture patterns such as event-driven architecture that result in even utilization of components and minimize overprovisioning in your workload. Using efficient architecture patterns minimizes idle resources from lack of use due to changes in demand over time.

Understand the requirements of your workload components and adopt architecture patterns that increase overall utilization of resources. Retire components that are no longer required.

Implementation steps

- Analyze the demand for your workload to determine how to respond to those.
- For requests or jobs that don't require synchronous responses, use queue-driven architectures and auto scaling workers to maximize utilization. Here are some examples of when you might consider queue-driven architecture:

Queuing mechanism	Description
AWS Batch job queues	AWS Batch jobs are submitted to a job queue where they reside until they can be scheduled to run in a compute environment.
Amazon Simple Queue Service and Amazon EC2 Spot Instances	Pairing Amazon SQS and Spot Instances to build fault tolerant and efficient architecture.

- For requests or jobs that can be processed anytime, use scheduling mechanisms to process jobs in batch for more efficiency. Here are some examples of scheduling mechanisms on AWS:

Scheduling mechanism	Description
Amazon EventBridge Scheduler	A capability from Amazon EventBridge that allows you to create, run, and manage scheduled tasks at scale.
AWS Glue time-based schedule	Define a time-based schedule for your crawlers and jobs in AWS Glue.
Amazon Elastic Container Service (Amazon ECS) scheduled tasks	Amazon ECS supports creating scheduled tasks. Scheduled tasks use Amazon EventBridge rules to run tasks either on a schedule or in a response to an EventBridge event.
Instance Scheduler	Configure start and stop schedules for your Amazon EC2 and Amazon Relational Database Service instances.

- If you use polling and webhooks mechanisms in your architecture, replace those with events. Use event-driven architectures to build highly efficient workloads.
- Leverage serverless on AWS to eliminate over-provisioned infrastructure.
- Right size individual components of your architecture to prevent idling resources waiting for input.

Resources

Related documents:

- What is Amazon Simple Queue Service?
- What is Amazon MQ?
- Scaling based on Amazon SQS
- What is AWS Step Functions?
- What is AWS Lambda?
- Using AWS Lambda with Amazon SQS
- What is Amazon EventBridge?

Related videos:

- Moving to event-driven architectures

SUS03-BP02 Remove or refactor workload components with low or no use

Remove components that are unused and no longer required, and refactor components with little utilization to minimize waste in your workload.

Common anti-patterns:

- You do not regularly check the utilization level of individual components of your workload.
- You do not check and analyze recommendations from AWS rightsizing tools such as AWS Compute Optimizer.

Benefits of establishing this best practice: Removing unused components minimizes waste and improves the overall efficiency of your cloud workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Review your workload to identify idle or unused components. This is an iterative improvement process which can be initiated by changes in demand or the release of a new cloud service. For example, a significant drop in AWS Lambda function run time can be an indicator of a need to lower the memory size. Also, as AWS releases new services and features, the optimal services and architecture for your workload may change.

Continually monitor workload activity and look for opportunities to improve the utilization level of individual components. By removing idle components and performing rightsizing activities, you meet your business requirements with the fewest cloud resources.

Implementation steps

- Monitor and capture the utilization metrics for critical components of your workload (like CPU utilization, memory utilization, or network throughput in Amazon CloudWatch metrics).
- For stable workloads, check AWS rightsizing tools such as AWS Compute Optimizer at regular intervals to identify idle, unused, or underutilized components.

- For ephemeral workloads, evaluate utilization metrics to identify idle, unused, or underutilized components.
- Retire components and associated assets (like Amazon ECR images) that are no longer needed.
- Refactor or consolidate underutilized components with other resources to improve utilization efficiency. For example, you can provision multiple small databases on a single Amazon RDS database instance instead of running databases on individual under-utilized instances.
- Understand the resources provisioned by your workload to complete a unit of work.

Resources

Related documents:

- AWS Trusted Advisor
- What is Amazon CloudWatch?
- Automated Cleanup of Unused Images in Amazon ECR

Related examples:

- Well-Architected Lab - Rightsizing with AWS Compute Optimizer
- Well-Architected Lab - Optimize Hardware Patterns and Observe Sustainability KPIs

SUS03-BP03 Optimize areas of code that consume the most time or resources

This best practice was updated with new guidance on July 13th, 2023.

Optimize your code that runs within different components of your architecture to minimize resource usage while maximizing performance.

Common anti-patterns:

- You ignore optimizing your code for resource usage.
- You usually respond to performance issues by increasing the resources.
- Your code review and development process does not track performance changes.

Benefits of establishing this best practice: Using efficient code minimizes resource usage and improves performance.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

It is crucial to examine every functional area, including the code for a cloud architected application, to optimize its resource usage and performance. Continually monitor your workload's performance in build environments and production and identify opportunities to improve code snippets that have particularly high resource usage. Adopt a regular review process to identify bugs or anti-patterns within your code that use resources inefficiently. Leverage simple and efficient algorithms that produce the same results for your use case.

Implementation steps

- While developing your workloads, adopt an automated code review process to improve quality and identify bugs and anti-patterns.
  - Automate code reviews with Amazon CodeGuru Reviewer
  - Detecting concurrency bugs with Amazon CodeGuru
  - Raising code quality for Python applications using Amazon CodeGuru
- As you run your workloads, monitor resources to identify components with high resource requirements per unit of work as targets for code reviews.
- For code reviews, use a code profiler to identify the areas of code that use the most time or resources as targets for optimization.
  - Reducing your organization's carbon footprint with Amazon CodeGuru Profiler
  - Understanding memory usage in your Java application with Amazon CodeGuru Profiler
  - Improving customer experience and reducing cost with Amazon CodeGuru Profiler
- Use the most efficient operating system and programming language for the workload. For details on energy efficient programming languages (including Rust), see Sustainability with Rust.
- Replace computationally intensive algorithms with simpler and more efficient version that produce the same result.
- Remove unnecessary code such as sorting and formatting.

Resources

Related documents:

- What is Amazon CodeGuru Profiler?
- FPGA instances
- The AWS SDKs on Tools to Build on AWS

Related videos:

- Improve Code Efficiency Using Amazon CodeGuru Profiler
- Automate Code Reviews and Application Performance Recommendations with Amazon CodeGuru

SUS03-BP04 Optimize impact on devices and equipment

Understand the devices and equipment used in your architecture and use strategies to reduce their usage. This can minimize the overall environmental impact of your cloud workload.

Common anti-patterns:

- You ignore the environmental impact of devices used by your customers.
- You manually manage and update resources used by customers.

Benefits of establishing this best practice: Implementing software patterns and features that are optimized for customer device can reduce the overall environmental impact of cloud workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implementing software patterns and features that are optimized for customer devices can reduce the environmental impact in several ways:

- Implementing new features that are backward compatible can reduce the number of hardware replacements.
- Optimizing an application to run efficiently on devices can help to reduce their energy consumption and extend their battery life (if they are powered by battery).
- Optimizing an application for devices can also reduce the data transfer over the network.

Understand the devices and equipment used in your architecture, their expected lifecycle, and the impact of replacing those components. Implement software patterns and features that can help to minimize the device energy consumption, the need for customers to replace the device and also upgrade it manually.

Implementation steps

- Inventory the devices used in your architecture. Devices can be mobile, tablet, IOT devices, smart light, or even smart devices in a factory.
- Optimize the application running on the devices:
  - Use strategies such as running tasks in the background to reduce their energy consumption.
  - Account for network bandwidth and latency when building payloads, and implement capabilities that help your applications work well on low bandwidth, high latency links.
  - Convert payloads and files into optimized formats required by devices. For example, you can use Amazon Elastic Transcoder or AWS Elemental MediaConvert to convert large, high quality digital media files into formats that users can play back on mobile devices, tablets, web browsers, and connected televisions.
  - Perform computationally intense activities server-side (such as image rendering), or use application streaming to improve the user experience on older devices.
  - Segment and paginate output, especially for interactive sessions, to manage payloads and limit local storage requirements.
- Use automated over-the-air (OTA) mechanism to deploy updates to one or more devices.
  - You can use a CI/CD pipeline to update mobile applications.
  - You can use AWS IoT Device Management to remotely manage connected devices at scale.
- To test new features and updates, use managed device farms with representative sets of hardware and iterate development to maximize the devices supported. For more details, see SUS06-BP04 Use managed device farms for testing.

Resources

Related documents:

- What is AWS Device Farm?
- Amazon AppStream 2.0 Documentation
- NICE DCV

- OTA tutorial for updating firmware on devices running FreeRTOS

Related videos:

- Introduction to AWS Device Farm

SUS03-BP05 Use software patterns and architectures that best support data access and storage patterns

Understand how data is used within your workload, consumed by your users, transferred, and stored. Use software patterns and architectures that best support data access and storage to minimize the compute, networking, and storage resources required to support the workload.

Common anti-patterns:

- You assume that all workloads have similar data storage and access patterns.
- You only use one tier of storage, assuming all workloads fit within that tier.
- You assume that data access patterns will stay consistent over time.
- Your architecture supports a potential high data access burst, which results in the resources remaining idle most of the time.

Benefits of establishing this best practice: Selecting and optimizing your architecture based on data access and storage patterns will help decrease development complexity and increase overall utilization. Understanding when to use global tables, data partitioning, and caching will help you decrease operational overhead and scale based on your workload needs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Use software and architecture patterns that aligns best to your data characteristics and access patterns. For example, use modern data architecture on AWS that allows you to use purpose-built services optimized for your unique analytics use cases. These architecture patterns allow for efficient data processing and reduce the resource usage.

Implementation steps

- Analyze your data characteristics and access patterns to identify the correct configuration for your cloud resources. Key characteristics to consider include:

  - **Data type:** structured, semi-structured, unstructured
  - **Data growth:** bounded, unbounded
  - **Data durability:** persistent, ephemeral, transient
  - **Access patterns** reads or writes, update frequency, spiky, or consistent
- Use architecture patterns that best support data access and storage patterns.
  - Let's Architect! Modern data architectures
  - Databases on AWS: The Right Tool for the Right Job
- Use technologies that work natively with compressed data.
- Use purpose-built analytics services for data processing in your architecture.
- Use the database engine that best supports your dominant query pattern. Manage your database indexes to ensure efficient querying. For further details, see AWS Databases.
- Select network protocols that reduce the amount of network capacity consumed in your architecture.

Resources

Related documents:

- Athena Compression Support file formats
- COPY from columnar data formats with Amazon Redshift
- Converting Your Input Record Format in Firehose
- Format Options for ETL Inputs and Outputs in AWS Glue
- Improve query performance on Amazon Athena by Converting to Columnar Formats
- Loading compressed data files from Amazon S3 with Amazon Redshift
- Monitoring DB load with Performance Insights on Amazon Aurora
- Monitoring DB load with Performance Insights on Amazon RDS
- Amazon S3 Intelligent-Tiering storage class

Related videos:

- Building modern data architectures on AWS

Data

Question

- SUS 4 How do you take advantage of data management policies and patterns to support your sustainability goals?

SUS 4 How do you take advantage of data management policies and patterns to support your sustainability goals?

Implement data management practices to reduce the provisioned storage required to support your workload, and the resources required to use it. Understand your data, and use storage technologies and configurations that more effectively support the business value of the data and how it's used. Lifecycle data to more efficient, less performant storage when requirements decrease, and delete data that's no longer required.

Best practices

- SUS04-BP01 Implement a data classification policy
- SUS04-BP02 Use technologies that support data access and storage patterns
- SUS04-BP03 Use policies to manage the lifecycle of your datasets
- SUS04-BP04 Use elasticity and automation to expand block storage or file system
- SUS04-BP05 Remove unneeded or redundant data
- SUS04-BP06 Use shared file systems or storage to access common data
- SUS04-BP07 Minimize data movement across networks
- SUS04-BP08 Back up data only when difficult to recreate

SUS04-BP01 Implement a data classification policy

Classify data to understand its criticality to business outcomes and choose the right energy-efficient storage tier to store the data.

Common anti-patterns:

- You do not identify data assets with similar characteristics (such as sensitivity, business criticality, or regulatory requirements) that are being processed or stored.
- You have not implemented a data catalog to inventory your data assets.

Benefits of establishing this best practice: Implementing a data classification policy allows you to determine the most energy-efficient storage tier for data.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Data classification involves identifying the types of data that are being processed and stored in an information system owned or operated by an organization. It also involves making a determination on the criticality of the data and the likely impact of a data compromise, loss, or misuse.

Implement data classification policy by working backwards from the contextual use of the data and creating a categorization scheme that takes into account the level of criticality of a given dataset to an organization's operations.

Implementation steps

- Conduct an inventory of the various data types that exist for your workload.
  - For more detail on data classification categories, see Data Classification whitepaper.
- Determine criticality, confidentiality, integrity, and availability of data based on risk to the organization. Use these requirements to group data into one of the data classification tiers that you adopt.
  - As an example, see Four simple steps to classify your data and secure your startup.
- Periodically audit your environment for untagged and unclassified data, and classify and tag the data appropriately.
  - As an example, see Data Catalog and crawlers in AWS Glue.
- Establish a data catalog that provides audit and governance capabilities.
- Determine and document the handling procedures for each data class.
- Use automation to continually audit your environment to identify untagged and unclassified data, and classify and tag the data appropriately.

Resources

Related documents:

- Leveraging AWS Cloud to Support Data Classification
- Tag policies from AWS Organizations

Related videos:

- Enabling agility with data governance on AWS

SUS04-BP02 Use technologies that support data access and storage patterns

This best practice was updated with new guidance on July 13th, 2023.

Use storage technologies that best support how your data is accessed and stored to minimize the resources provisioned while supporting your workload.

Common anti-patterns:

- You assume that all workloads have similar data storage and access patterns.
- You only use one tier of storage, assuming all workloads fit within that tier.
- You assume that data access patterns will stay consistent over time.

Benefits of establishing this best practice: Selecting and optimizing your storage technologies based on data access and storage patterns will help you reduce the required cloud resources to meet your business needs and improve the overall efficiency of cloud workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Select the storage solution that aligns best to your access patterns, or consider changing your access patterns to align with the storage solution to maximize performance efficiency.

- Evaluate your data characteristics and access pattern to collect the key characteristics of your storage needs. Key characteristics to consider include:
  - **Data type:** structured, semi-structured, unstructured
  - **Data growth:** bounded, unbounded
  - **Data durability:** persistent, ephemeral, transient
  - **Access patterns:** reads or writes, frequency, spiky, or consistent

- Migrate data to the appropriate storage technology that supports your data characteristics and access pattern. Here are some examples of AWS storage technologies and their key characteristics:

Type	Technology	Key characteristics
Object storage	Amazon S3	An object storage service with unlimited scalability, high availability, and multiple options for accessibility. Transferring and accessing objects in and out of Amazon S3 can use a service, such as Transfer Acceleration or Access Points, to support your location, security needs, and access patterns.
Archiving storage	Amazon Glacier	Storage class of Amazon S3 built for data-archiving.
Shared file system	Amazon Elastic File System (Amazon EFS)	Mountable file system that can be accessed by multiple types of compute solutions. Amazon EFS automatically grows and shrinks storage and is performance-optimized to deliver consistent low latencies.

Type	Technology	Key characteristics
Shared file system	Amazon FSx	Built on the latest AWS compute solutions to support four commonly used file systems: NetApp ONTAP, OpenZFS, Windows File Server, and Lustre. Amazon FSx latency, throughput, and IOPS vary per file system and should be considered when selecting the right file system for your workload needs.
Block storage	Amazon Elastic Block Store (Amazon EBS)	Scalable, high-performance block-storage service designed for Amazon Elastic Compute Cloud (Amazon EC2). Amazon EBS includes SSD-backed storage for transactional, IOPS-intensive workloads and HDD-backe d storage for throughput-intensive workloads.

Type	Technology	Key characteristics
Relational database	Amazon Aurora, Amazon RDS, Amazon Redshift	Designed to support ACID (atomicity, consistency, isolation, durability) transactions and maintain referential integrity and strong data consistency. Many traditional applications, enterprise resource planning (ERP), customer relationship management (CRM), and ecommerce systems use relational databases to store their data.
Key-value database	Amazon DynamoDB	Optimized for common access patterns, typically to store and retrieve large volumes of data. High-traffic web apps, ecommerce systems, and gaming applications are typical use-cases for key-value databases.

- For storage systems that are a fixed size, such as Amazon EBS or Amazon FSx, monitor the available storage space and automate storage allocation on reaching a threshold. You can leverage Amazon CloudWatch to collect and analyze different metrics for Amazon EBS and Amazon FSx.
- Amazon S3 Storage Classes can be configured at the object level and a single bucket can contain objects stored across all of the storage classes.
- You can also use Amazon S3 Lifecycle policies to automatically transition objects between storage classes or remove data without any application changes. In general, you have to make a trade-off between resource efficiency, access latency, and reliability when considering these storage mechanisms.

Resources

Related documents:

- Amazon EBS volume types
- Amazon EC2 instance store
- Amazon S3 Intelligent-Tiering
- Amazon EBS I/O Characteristics
- Using Amazon S3 storage classes
- What is Amazon Glacier?

Related videos:

- Architectural Patterns for Data Lakes on AWS
- Deep dive on Amazon EBS (STG303-R1)
- Optimize your storage performance with Amazon S3 (STG343)
- Building modern data architectures on AWS

Related examples:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 Examples

SUS04-BP03 Use policies to manage the lifecycle of your datasets

Manage the lifecycle of all of your data and automatically enforce deletion to minimize the total storage required for your workload.

Common anti-patterns:

- You manually delete data.
- You do not delete any of your workload data.

- You do not move data to more energy-efficient storage tiers based on its retention and access requirements.

Benefits of establishing this best practice: Using data lifecycle policies ensures efficient data access and retention in a workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Datasets usually have different retention and access requirements during their lifecycle. For example, your application may need frequent access to some datasets for a limited period of time. After that, those datasets are infrequently accessed.

To efficiently manage your datasets throughout their lifecycle, configure lifecycle policies, which are rules that define how to handle datasets.

With Lifecycle configuration rules, you can tell the specific storage service to transition a dataset to more energy-efficient storage tiers, archive it, or delete it.

Implementation steps

- Classify datasets in your workload.
- Define handling procedures for each data class.
- Set automated lifecycle policies to enforce lifecycle rules. Here are some examples of how to set up automated lifecycle policies for different AWS storage services:

Storage service	How to set automated lifecycle policies
Amazon S3	You can use Amazon S3 Lifecycle to manage your objects throughout their lifecycle. If your access patterns are unknown, changing, or unpredictable, you can use Amazon S3 Intelligent-Tiering, which monitors access patterns and automatically moves objects that have not been accessed to lower-cos t access tiers. You can leverage Amazon S3 Storage Lens metrics to identify optimizat

Storage service	How to set automated lifecycle policies
	ion opportunities and gaps in lifecycle management.
Amazon Elastic Block Store	You can use Amazon Data Lifecycle Manager to automate the creation, retention, and deletion of Amazon EBS snapshots and Amazon EBS-backed AMIs.
Amazon Elastic File System	Amazon EFS lifecycle management automatic ally manages file storage for your file systems.
Amazon Elastic Container Registry	Amazon ECR lifecycle policies automate the cleanup of your container images by expiring images based on age or count.
AWS Elemental MediaStore	You can use an object lifecycle policy that governs how long objects should be stored in the MediaStore container.

- Delete unused volumes, snapshots, and data that is out of its retention period. Leverage native service features like Amazon DynamoDB Time To Live or Amazon CloudWatch log retention for deletion.
- Aggregate and compress data where applicable based on lifecycle rules.

Resources

Related documents:

- Optimize your Amazon S3 Lifecycle rules with Amazon S3 Storage Class Analysis
- Evaluating Resources with AWS Config Rules

Related videos:

- Simplify Your Data Lifecycle and Optimize Storage Costs With Amazon S3 Lifecycle
- Reduce Your Storage Costs Using Amazon S3 Storage Lens

SUS04-BP04 Use elasticity and automation to expand block storage or file system

Use elasticity and automation to expand block storage or file system as data grows to minimize the total provisioned storage.

Common anti-patterns:

- You procure large block storage or file system for future need.
- You overprovision the input and output operations per second (IOPS) of your file system.
- You do not monitor the utilization of your data volumes.

Benefits of establishing this best practice: Minimizing over-provisioning for storage system reduces the idle resources and improves the overall efficiency of your workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Create block storage and file systems with size allocation, throughput, and latency that are appropriate for your workload. Use elasticity and automation to expand block storage or file system as data grows without having to over-provision these storage services.

Implementation steps

- For fixed size storage like Amazon EBS, verify that you are monitoring the amount of storage used versus the overall storage size and create automation, if possible, to increase the storage size when reaching a threshold.
- Use elastic volumes and managed block data services to automate allocation of additional storage as your persistent data grows. As an example, you can use Amazon EBS Elastic Volumes to change volume size, volume type, or adjust the performance of your Amazon EBS volumes.
- Choose the right storage class, performance mode, and throughput mode for your file system to address your business need, not exceeding that.
  - Amazon EFS performance
  - Amazon EBS volume performance on Linux instances
- Set target levels of utilization for your data volumes, and resize volumes outside of expected ranges.
- Right size read-only volumes to fit the data.

- Migrate data to object stores to avoid provisioning the excess capacity from fixed volume sizes on block storage.
- Regularly review elastic volumes and file systems to terminate idle volumes and shrink over-provisioned resources to fit the current data size.

Resources

Related documents:

- Amazon FSx Documentation
- What is Amazon Elastic File System?

Related videos:

- Deep Dive on Amazon EBS Elastic Volumes
- Amazon EBS and Snapshot Optimization Strategies for Better Performance and Cost Savings
- Optimizing Amazon EFS for cost and performance, using best practices

SUS04-BP05 Remove unneeded or redundant data

Remove unneeded or redundant data to minimize the storage resources required to store your datasets.

Common anti-patterns:

- You duplicate data that can be easily obtained or recreated.
- You back up all data without considering its criticality.
- You only delete data irregularly, on operational events, or not at all.
- You store data redundantly irrespective of the storage service's durability.
- You turn on Amazon S3 versioning without any business justification.

Benefits of establishing this best practice: Removing unneeded data reduces the storage size required for your workload and the workload environmental impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Do not store data that you do not need. Automate the deletion of unneeded data. Use technologies that deduplicate data at the file and block level. Leverage native data replication and redundancy features of services.

Implementation steps

- Evaluate if you can avoid storing data by using existing publicly available datasets in AWS Data Exchange and Open Data on AWS.
- Use mechanisms that can deduplicate data at the block and object level. Here are some examples of how to deduplicate data on AWS:

Storage service	Deduplication mechanism
Amazon S3	Use AWS Lake Formation FindMatches to find matching records across a dataset (includin g ones without identifiers) by using the new FindMatches ML Transform.
Amazon FSx	Use data deduplication on Amazon FSx for Windows.
Amazon Elastic Block Store snapshots	Snapshots are incremental backups, which means that only the blocks on the device that have changed after your most recent snapshot are saved.

- Analyze the data access to identify unneeded data. Automate lifecycle policies. Leverage native service features like Amazon DynamoDB Time To Live, Amazon S3 Lifecycle, or Amazon CloudWatch log retention for deletion.
- Use data virtualization capabilities on AWS to maintain data at its source and avoid data duplication.
  - Cloud Native Data Virtualization on AWS
  - Lab: Optimize Data Pattern Using Amazon Redshift Data Sharing
- Use backup technology that can make incremental backups.

- Leverage the durability of Amazon S3 and replication of Amazon EBS to meet your durability goals instead of self-managed technologies (such as a redundant array of independent disks (RAID)).
- Centralize log and trace data, deduplicate identical log entries, and establish mechanisms to tune verbosity when needed.
- Pre-populate caches only where justified.
- Establish cache monitoring and automation to resize the cache accordingly.
- Remove out-of-date deployments and assets from object stores and edge caches when pushing new versions of your workload.

Resources

Related documents:

- Change log data retention in CloudWatch Logs
- Data deduplication on Amazon FSx for Windows File Server
- Features of Amazon FSx for ONTAP including data deduplication
- Invalidating Files on Amazon CloudFront
- Using AWS Backup to back up and restore Amazon EFS file systems
- What is Amazon CloudWatch Logs?
- Working with backups on Amazon RDS

Related videos:

- Fuzzy Matching and Deduplicating Data with ML Transforms for AWS Lake Formation

Related examples:

- How do I analyze my Amazon S3 server access logs using Amazon Athena?

SUS04-BP06 Use shared file systems or storage to access common data

Adopt shared file systems or storage to avoid data duplication and allow for more efficient infrastructure for your workload.

Common anti-patterns:

- You provision storage for each individual client.
- You do not detach data volume from inactive clients.
- You do not provide access to storage across platforms and systems.

Benefits of establishing this best practice: Using shared file systems or storage allows for sharing data to one or more consumers without having to copy the data. This helps to reduce the storage resources required for the workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

If you have multiple users or applications accessing the same datasets, using shared storage technology is crucial to use efficient infrastructure for your workload. Shared storage technology provides a central location to store and manage datasets and avoid data duplication. It also enforces consistency of the data across different systems. Moreover, shared storage technology allows for more efficient use of compute power, as multiple compute resources can access and process data at the same time in parallel.

Fetch data from these shared storage services only as needed and detach unused volumes to free up resources.

Implementation steps

- Migrate data to shared storage when the data has multiple consumers. Here are some examples of shared storage technology on AWS:

Storage option	When to use
Amazon EBS Multi-Attach	Amazon EBS Multi-Attach allows you to attach a single Provisioned IOPS SSD (io1 or io2) volume to multiple instances that are in the same Availability Zone.
Amazon EFS	See When to Choose Amazon EFS.
Amazon FSx	See Choosing an Amazon FSx File System.

Storage option	When to use
Amazon S3	Applications that do not require a file system structure and are designed to work with object storage can use Amazon S3 as a massively scalable, durable, low-cost object storage solution.

- Copy data to or fetch data from shared file systems only as needed. As an example, you can create an Amazon FSx for Lustre file system backed by Amazon S3 and only load the subset of data required for processing jobs to Amazon FSx.
- Delete data as appropriate for your usage patterns as outlined in SUS04-BP03 Use policies to manage the lifecycle of your datasets.
- Detach volumes from clients that are not actively using them.

Resources

Related documents:

- Linking your file system to an Amazon S3 bucket
- Using Amazon EFS for AWS Lambda in your serverless applications
- Amazon EFS Intelligent-Tiering Optimizes Costs for Workloads with Changing Access Patterns
- Using Amazon FSx with your on-premises data repository

related videos:

- Storage cost optimization with Amazon EFS

SUS04-BP07 Minimize data movement across networks

> This best practice was updated with new guidance on July 13th, 2023.

Use shared file systems or object storage to access common data and minimize the total networking resources required to support data movement for your workload.

Common anti-patterns:

- You store all data in the same AWS Region independent of where the data users are.
- You do not optimize data size and format before moving it over the network.

Benefits of establishing this best practice: Optimizing data movement across the network reduces the total networking resources required for the workload and lowers its environmental impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Moving data around your organization requires compute, networking, and storage resources. Use techniques to minimize data movement and improve the overall efficiency of your workload.

Implementation steps

- Consider proximity to the data or users as a decision factor when selecting a Region for your workload.
- Partition Regionally consumed services so that their Region-specific data is stored within the Region where it is consumed.
- Use efficient file formats (such as Parquet or ORC) and compress data before moving it over the network.
- Don't move unused data. Some examples that can help you avoid moving unused data:
  - Reduce API responses to only relevant data.
  - Aggregate data where detailed (record-level information is not required).
  - See Well-Architected Lab - Optimize Data Pattern Using Amazon Redshift Data Sharing.
  - Consider Cross-account data sharing in AWS Lake Formation.
- Use services that can help you run code closer to users of your workload.

Service	When to use
Lambda@Edge	Use for compute-heavy operations that are run when objects are not in the cache.
CloudFront Functions	Use for simple use cases such as HTTP(s) request/response manipulations that can be initiated by short-lived functions.

Service	When to use
AWS IoT Greengrass	Run local compute, messaging, and data caching for connected devices.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking
- AWS Global Infrastructure
- Amazon CloudFront Key Features including the CloudFront Global Edge Network
- Compressing HTTP requests in Amazon OpenSearch Service
- Intermediate data compression with Amazon EMR
- Loading compressed data files from Amazon S3 into Amazon Redshift
- Serving compressed files with Amazon CloudFront

Related videos:

- Demystifying data transfer on AWS

Related examples:

- Architecting for sustainability - Minimize data movement across networks

SUS04-BP08 Back up data only when difficult to recreate

Avoid backing up data that has no business value to minimize storage resources requirements for your workload.

Common anti-patterns:

- You do not have a backup strategy for your data.
- You back up data that can be easily recreated.

Benefits of establishing this best practice: Avoiding back-up of non-critical data reduces the required storage resources for the workload and lowers its environmental impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Avoiding the back up of unnecessary data can help lower cost and reduce the storage resources used by the workload. Only back up data that has business value or is needed to satisfy compliance requirements. Examine backup policies and exclude ephemeral storage that doesn't provide value in a recovery scenario.

Implementation steps

- Implement data classification policy as outlined in SUS04-BP01 Implement a data classification policy.
- Use the criticality of your data classification and design backup strategy based on your recovery time objective (RTO) and recovery point objective (RPO). Avoid backing up non-critical data.
  - Exclude data that can be easily recreated.
  - Exclude ephemeral data from your backups.
  - Exclude local copies of data, unless the time required to restore that data from a common location exceeds your service-level agreements (SLAs).
- Use an automated solution or managed service to back up business-critical data.
  - AWS Backup is a fully-managed service that makes it easy to centralize and automate data protection across AWS services, in the cloud, and on premises. For hands-on guidance on how to create automated backups using AWS Backup, see Well-Architected Labs - Testing Backup and Restore of Data.
  - Automate backups and optimize backup costs for Amazon EFS using AWS Backup.

Resources

Related best practices:

- REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources
- REL09-BP03 Perform data backup automatically
- REL13-BP02 Use defined recovery strategies to meet the recovery objectives

Related documents:

- Using AWS Backup to back up and restore Amazon EFS file systems
- Amazon EBS snapshots
- Working with backups on Amazon Relational Database Service
- APN Partner: partners that can help with backup
- AWS Marketplace: products that can be used for backup
- Backing Up Amazon EFS
- Backing Up Amazon FSx for Windows File Server
- Backup and Restore for Amazon ElastiCache (Redis OSS)

Related videos:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS
- AWS Backup Demo: Cross-Account and Cross-Region Backup
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

Related examples:

- Well-Architected Lab - Testing Backup and Restore of Data
- Well-Architected Lab - Backup and Restore with Failback for Analytics Workload
- Well-Architected Lab - Disaster Recovery - Backup and Restore

Hardware and services

Question

- SUS 5 How do you select and use cloud hardware and services in your architecture to support your sustainability goals?

SUS 5 How do you select and use cloud hardware and services in your architecture to support your sustainability goals?

Look for opportunities to reduce workload sustainability impacts by making changes to your hardware management practices. Minimize the amount of hardware needed to provision and deploy, and select the most efficient hardware and services for your individual workload.

Best practices

- SUS05-BP01 Use the minimum amount of hardware to meet your needs
- SUS05-BP02 Use instance types with the least impact
- SUS05-BP03 Use managed services
- SUS05-BP04 Optimize your use of hardware-based compute accelerators

SUS05-BP01 Use the minimum amount of hardware to meet your needs

Use the minimum amount of hardware for your workload to efficiently meet your business needs.

Common anti-patterns:

- You do not monitor resource utilization.
- You have resources with a low utilization level in your architecture.
- You do not review the utilization of static hardware to determine if it should be resized.
- You do not set hardware utilization goals for your compute infrastructure based on business KPIs.

Benefits of establishing this best practice: Rightsizing your cloud resources helps to reduce a workload's environmental impact, save money, and maintain performance benchmarks.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Optimally select the total number of hardware required for your workload to improve its overall efficiency. The AWS Cloud provides the flexibility to expand or reduce the number of resources dynamically through a variety of mechanisms, such as AWS Auto Scaling, and meet changes in demand. It also provides APIs and SDKs that allow resources to be modified with minimal effort. Use these capabilities to make frequent changes to your workload implementations. Additionally,

use rightsizing guidelines from AWS tools to efficiently operate your cloud resource and meet your business needs.

Implementation steps

- Choose the instances type to best fit your needs.
  - How do I choose the appropriate Amazon EC2 instance type for my workload?
  - Attribute-based instance type selection for Amazon EC2 Fleet.
  - Create an Auto Scaling group using attribute-based instance type selection.
- Scale using small increments for variable workloads.
- Use multiple compute purchase options in order to balance instance flexibility, scalability, and cost savings.
  - On-Demand Instances are best suited for new, stateful, and spiky workloads which can't be instance type, location, or time flexible.
  - Spot Instances are a great way to supplement the other options for applications that are fault tolerant and flexible.
  - Leverage Compute Savings Plans for steady state workloads that allow flexibility if your needs (like AZ, Region, instance families, or instance types) change.
- Use instance and availability zone diversity to maximize application availability and take advantage of excess capacity when possible.
- Use the rightsizing recommendations from AWS tools to make adjustments on your workload.
  - AWS Compute Optimizer
  - AWS Trusted Advisor
- Negotiate service-level agreements (SLAs) that allow for a temporary reduction in capacity while automation deploys replacement resources.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute
- Attirbute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet
- AWS Compute Optimizer Documentation
- Operating Lambda: Performance optimization

- Auto Scaling Documentation

Related videos:

- Build a cost-, energy-, and resource-efficient compute environment

Related examples:

- Well-Architected Lab - Rightsizing with AWS Compute Optimizer and Memory Utilization Enabled (Level 200)

SUS05-BP02 Use instance types with the least impact

> This best practice was updated with new guidance on July 13th, 2023.

Continually monitor and use new instance types to take advantage of energy efficiency improvements.

Common anti-patterns:

- You are only using one family of instances.
- You are only using x86 instances.
- You specify one instance type in your Amazon EC2 Auto Scaling configuration.
- You use AWS instances in a manner that they were not designed for (for example, you use compute-optimized instances for a memory-intensive workload).
- You do not evaluate new instance types regularly.
- You do not check recommendations from AWS rightsizing tools such as AWS Compute Optimizer.

Benefits of establishing this best practice: By using energy-efficient and right-sized instances, you are able to greatly reduce the environmental impact and cost of your workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Using efficient instances in cloud workload is crucial for lower resource usage and cost-effectiveness. Continually monitor the release of new instance types and take advantage of energy efficiency improvements, including those instance types designed to support specific workloads such as machine learning training and inference, and video transcoding.

Implementation steps

- Learn and explore instance types which can lower your workload environmental impact.
  - Subscribe to What's New with AWS to be up-to-date with the latest AWS technologies and instances.
  - Learn about different AWS instance types.
  - Learn about AWS Graviton-based instances which offer the best performance per watt of energy use in Amazon EC2 by watching re:Invent 2020 - Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances and Deep dive into AWS Graviton3 and Amazon EC2 C7g instances.
- Plan and transition your workload to instance types with the least impact.
  - Define a process to evaluate new features or instances for your workload. Take advantage of agility in the cloud to quickly test how new instance types can improve your workload environmental sustainability. Use proxy metrics to measure how many resources it takes you to complete a unit of work.
  - If possible, modify your workload to work with different numbers of vCPUs and different amounts of memory to maximize your choice of instance type.
  - Consider transitioning your workload to Graviton-based instances to improve the performance efficiency of your workload.
    - AWS Graviton Fast Start
    - Considerations when transitioning workloads to AWS Graviton-based Amazon Elastic Compute Cloud instances
    - AWS Graviton2 for ISVs
  - Consider selecting the AWS Graviton option in your usage of AWS managed services.
  - Migrate your workload to Regions that offer instances with the least sustainability impact and still meet your business requirements.
  - For machine learning workloads, take advantage of purpose-built hardware that is specific to your workload such as AWS Trainium, AWS Inferentia, and Amazon EC2 DL1. AWS Inferentia

instances such as Inf2 instances offer up to 50% better performance per watt over comparable Amazon EC2 instances.

- Use Amazon SageMaker AI Inference Recommender to right size ML inference endpoint.
- For spiky workloads (workloads with infrequent requirements for additional capacity), use burstable performance instances.
- For stateless and fault-tolerant workloads, use Amazon EC2 Spot Instances to increase overall utilization of the cloud, and reduce the sustainability impact of unused resources.

- Operate and optimize your workload instance.
  - For ephemeral workloads, evaluate instance Amazon CloudWatch metrics such as `CPUUtilization` to identify if the instance is idle or under-utilized.
  - For stable workloads, check AWS rightsizing tools such as AWS Compute Optimizer at regular intervals to identify opportunities to optimize and right-size the instances.
    - Well-Architected Lab - Rightsizing Recommendations
    - Well-Architected Lab - Rightsizing with Compute Optimizer
    - Well-Architected Lab - Optimize Hardware Patterns and Observice Sustainability KPIs

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute
- AWS Graviton
- Amazon EC2 DL1
- Amazon EC2 Capacity Reservation Fleets
- Amazon EC2 Spot Fleet
- Functions: Lambda Function Configuration
- Attribute-based instance type selection for Amazon EC2 Fleet
- Building Sustainable, Efficient, and Cost-Optimized Applications on AWS
- How the Contino Sustainability Dashboard Helps Customers Optimize Their Carbon Footprint

Related videos:

- Deep dive on AWS Graviton2 processer-powered Amazon EC2 instances

- Deep dive into AWS Graviton3 and Amazon EC2 C7g instances
- Build a cost-, energy-, and resource-efficient compute environment

Related examples:

- Solution: Guidance for Optimizing Deep Learning Workloads for Sustainability on AWS
- Well-Architected Lab - Rightsizing Recommendations
- Well-Architected Lab - Rightsizing with Compute Optimizer
- Well-Architected Lab - Optimize Hardware Patterns and Observe Sustainability KPIs
- Well-Architected Lab - Migrating Services to Graviton

SUS05-BP03 Use managed services

Use managed services to operate more efficiently in the cloud.

Common anti-patterns:

- You use Amazon EC2 instances with low utilization to run your applications.
- Your in-house team only manages the workload, without time to focus on innovation or simplifications.
- You deploy and maintain technologies for tasks that can run more efficiently on managed services.

Benefits of establishing this best practice:

- Using managed services shifts the responsibility to AWS, which has insights across millions of customers that can help drive new innovations and efficiencies.
- Managed service distributes the environmental impact of the service across many users because of the multi-tenet control planes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Managed services shift responsibility to AWS for maintaining high utilization and sustainability optimization of the deployed hardware. Managed services also remove the operational and

administrative burden of maintaining a service, which allows your team to have more time and focus on innovation.

Review your workload to identify the components that can be replaced by AWS managed services. For example, Amazon RDS, Amazon Redshift, and Amazon ElastiCache provide a managed database service. Amazon Athena, Amazon EMR, and Amazon OpenSearch Service provide a managed analytics service.

Implementation steps

1. Inventory your workload for services and components.
2. Assess and identify components that can be replaced by managed services. Here are some examples of when you might consider using a managed service:

Task	What to use on AWS
Hosting a database	Use managed Amazon Relational Database Service (Amazon RDS) instances instead of maintaining your own Amazon RDS instances on Amazon Elastic Compute Cloud (Amazon EC2).
Hosting a container workload	Use AWS Fargate, instead of implementing your own container infrastructure.
Hosting web apps	Use AWS Amplify Hosting as fully managed CI/CD and hosting service for static websites and server-side rendered web apps.

3. Identify dependencies and create a migrations plan. Update runbooks and playbooks accordingly.
   - The AWS Application Discovery Service automatically collects and presents detailed information about application dependencies and utilization to help you make more informed decisions as you plan your migration
4. Test the service before migrating to the managed service.
5. Use the migration plan to replace self-hosted services with managed service.
6. Continually monitor the service after the migration is complete to make adjustments as required and optimize the service.

Resources

Related documents:

- AWS Cloud Products
- AWS Total Cost of Ownership (TCO) Calculator
- Amazon DocumentDB
- Amazon Elastic Kubernetes Service (EKS)
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Related videos:

- Cloud operations at scale with AWS Managed Services

SUS05-BP04 Optimize your use of hardware-based compute accelerators

Optimize your use of accelerated computing instances to reduce the physical infrastructure demands of your workload.

Common anti-patterns:

- You are not monitoring GPU usage.
- You are using a general-purpose instance for workload while a purpose-built instance can deliver higher performance, lower cost, and better performance per watt.
- You are using hardware-based compute accelerators for tasks where they're more efficient using CPU-based alternatives.

Benefits of establishing this best practice: By optimizing the use of hardware-based accelerators, you can reduce the physical-infrastructure demands of your workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

If you require high processing capability, you can benefit from using accelerated computing instances, which provide access to hardware-based compute accelerators such as graphics processing units (GPUs) and field programmable gate arrays (FPGAs). These hardware accelerators perform certain functions like graphic processing or data pattern matching more efficiently

than CPU-based alternatives. Many accelerated workloads, such as rendering, transcoding, and machine learning, are highly variable in terms of resource usage. Only run this hardware for the time needed, and decommission them with automation when not required to minimize resources consumed.

Implementation steps

- Identify which accelerated computing instances can address your requirements.
- For machine learning workloads, take advantage of purpose-built hardware that is specific to your workload, such as AWS Trainium, AWS Inferentia, and Amazon EC2 DL1. AWS Inferentia instances such as Inf2 instances offer up to 50% better performance per watt over comparable Amazon EC2 instances.
- Collect usage metric for your accelerated computing instances. For example, you can use CloudWatch agent to collect metrics such as `utilization_gpu` and `utilization_memory` for your GPUs as shown in Collect NVIDIA GPU metrics with Amazon CloudWatch.
- Optimize the code, network operation, and settings of hardware accelerators to make sure that underlying hardware is fully utilized.
  - Optimize GPU settings
  - GPU Monitoring and Optimization in the Deep Learning AMI
  - Optimizing I/O for GPU performance tuning of deep learning training in Amazon SageMaker AI
- Use the latest high performant libraries and GPU drivers.
- Use automation to release GPU instances when not in use.

Resources

Related documents:

- Accelerated Computing
- Let's Architect! Architecting with custom chips and accelerators
- How do I choose the appropriate Amazon EC2 instance type for my workload?
- Amazon EC2 VT1 Instances
- Amazon Elastic Graphics
- Choose the best AI accelerator and model compilation for computer vision inference with Amazon SageMaker AI

Related videos:

- How to select Amazon EC2 GPU instances for deep learning
- Deep Dive on Amazon EC2 Elastic GPUs
- Deploying Cost-Effective Deep Learning Inference

Process and culture

Question

- SUS 6 How do your organizational processes support your sustainability goals?

SUS 6 How do your organizational processes support your sustainability goals?

Look for opportunities to reduce your sustainability impact by making changes to your development, test, and deployment practices.

Best practices

- SUS06-BP01 Adopt methods that can rapidly introduce sustainability improvements
- SUS06-BP02 Keep your workload up-to-date
- SUS06-BP03 Increase utilization of build environments
- SUS06-BP04 Use managed device farms for testing

SUS06-BP01 Adopt methods that can rapidly introduce sustainability improvements

Adopt methods and processes to validate potential improvements, minimize testing costs, and deliver small improvements.

Common anti-patterns:

- Reviewing your application for sustainability is a task done only once at the beginning of a project.
- Your workload has become stale, as the release process is too cumbersome to introduce minor changes for resource efficiency.
- You do not have mechanisms to improve your workload for sustainability.

Benefits of establishing this best practice: By establishing a process to introduce and track sustainability improvements, you will be able to continually adopt new features and capabilities, remove issues, and improve workload efficiency.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Test and validate potential sustainability improvements before deploying them to production. Account for the cost of testing when calculating potential future benefit of an improvement. Develop low cost testing methods to deliver small improvements.

Implementation steps

- Add requirements for sustainability improvement to your development backlog.
- Use an iterative improvement process to identify, evaluate, prioritize, test, and deploy these improvements.
- Continually improve and streamline your development processes. As an example, Automate your software delivery process using continuous integration and delivery (CI/CD) pipelines to test and deploy potential improvements to reduce the level of effort and limit errors caused by manual processes.
- Develop and test potential improvements using the minimum viable representative components to reduce the cost of testing.
- Continually assess the impact of improvements and make adjustment as needed.

Resources

Related documents:

- AWS enables sustainability solutions
- Scalable agile development practices based on AWS CodeCommit

Related videos:

- Delivering sustainable, high-performing architectures

Related examples:

- Well-Architected Lab - Turning cost & usage reports into efficiency reports

SUS06-BP02 Keep your workload up-to-date

Keep your workload up-to-date to adopt efficient features, remove issues, and improve the overall efficiency of your workload.

Common anti-patterns:

- You assume your current architecture is static and will not be updated over time.
- You do not have any systems or a regular cadence to evaluate if updated software and packages are compatible with your workload.

Benefits of establishing this best practice: By establishing a process to keep your workload up to date, you can adopt new features and capabilities, resolve issues, and improve workload efficiency.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Up to date operating systems, runtimes, middlewares, libraries, and applications can improve workload efficiency and make it easier to adopt more efficient technologies. Up to date software might also include features to measure the sustainability impact of your workload more accurately, as vendors deliver features to meet their own sustainability goals. Adopt a regular cadence to keep your workload up to date with the latest features and releases.

Implementation steps

- Define a process and a schedule to evaluate new features or instances for your workload. Take advantage of agility in the cloud to quickly test how new features can improve your workload to:
  - Reduce sustainability impacts.
  - Gain performance efficiencies.
  - Remove barriers for a planned improvement.
  - Improve your ability to measure and manage sustainability impacts.
- Inventory your workload software and architecture and identify components that need to be updated.
  - You can use AWS Systems Manager Inventory to collect operating system (OS), application, and instance metadata from your Amazon EC2 instances and quickly understand which

instances are running the software and configurations required by your software policy and which instances need to be updated.

- Understand how to update the components of your workload.

Workload component	How to update
Machine images	Use EC2 Image Builder to manage updates to Amazon Machine Images (AMIs) for Linux or Windows server images.
Container images	Use Amazon Elastic Container Registry (Amazon ECR) with your existing pipeline to manage Amazon Elastic Container Service (Amazon ECS) images.
AWS Lambda	AWS Lambda includes version management features.

- Use automation for the update process to reduce the level of effort to deploy new features and limit errors caused by manual processes.
  - You can use CI/CD to automatically update AMIs, container images, and other artifacts related to your cloud application.
  - You can use tools such as AWS Systems Manager Patch Manager to automate the process of system updates, and schedule the activity using AWS Systems Manager Maintenance Windows.

Resources

Related documents:

- AWS Architecture Center
- What's New with AWS
- AWS Developer Tools

Related examples:

- Well-Architected Labs - Inventory and Patch Management
- Lab: AWS Systems Manager

SUS06-BP03 Increase utilization of build environments

Increase the utilization of resources to develop, test, and build your workloads.

Common anti-patterns:

- You manually provision or terminate your build environments.
- You keep your build environments running independent of test, build, or release activities (for example, running an environment outside of the working hours of your development team members).
- You over-provision resources for your build environments.

Benefits of establishing this best practice: By increasing the utilization of build environments, you can improve the overall efficiency of your cloud workload while allocating the resources to builders to develop, test, and build efficiently.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use automation and infrastructure-as-code to bring build environments up when needed and take them down when not used. A common pattern is to schedule periods of availability that coincide with the working hours of your development team members. Your test environments should closely resemble the production configuration. However, look for opportunities to use instance types with burst capacity, Amazon EC2 Spot Instances, automatic scaling database services, containers, and serverless technologies to align development and test capacity with use. Limit data volume to just meet the test requirements. If using production data in test, explore possibilities of sharing data from production and not moving data across.

Implementation steps

- Use infrastructure-as-code to provision your build environments.
- Use automation to manage the lifecycle of your development and test environments and maximize the efficiency of your build resources.
- Use strategies to maximize the utilization of development and test environments.
  - Use minimum viable representative environments to develop and test potential improvements.
  - Use serverless technologies if possible.

- Use On-Demand Instances to supplement your developer devices.
- Use instance types with burst capacity, Spot Instances, and other technologies to align build capacity with use.
- Adopt native cloud services for secure instance shell access rather than deploying fleets of bastion hosts.
- Automatically scale your build resources depending on your build jobs.

Resources

Related documents:

- AWS Systems Manager Session Manager
- Amazon EC2 Burstable performance instances
- What is AWS CloudFormation?
- What is AWS CodeBuild?
- Instance Scheduler on AWS

Related videos:

- Continuous Integration Best Practices

SUS06-BP04 Use managed device farms for testing

Use managed device farms to efficiently test a new feature on a representative set of hardware.

Common anti-patterns:

- You manually test and deploy your application on individual physical devices.
- You do not use app testing service to test and interact with your apps (for example, Android, iOS, and web apps) on real, physical devices.

Benefits of establishing this best practice: Using managed device farms for testing cloud-enabled applications provides a number of benefits:

- They include more efficient features to test application on wide range of devices.
- They eliminate the need for in-house infrastructure for testing.

- They offer diverse device types, including older and less popular hardware, which eliminates the need for unnecessary device upgrades.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Using Managed device farms can help you to streamline the testing process for new features on a representative set of hardware. Managed device farms offer diverse device types including older, less popular hardware, and avoid customer sustainability impact from unnecessary device upgrades.

Implementation steps

- Define your testing requirements and plan (like test type, operating systems, and test schedule).
  - You can use Amazon CloudWatch RUM to collect and analyze client-side data and shape your testing plan.
- Select the managed device farm that can support your testing requirements. For example, you can use AWS Device Farm to test and understand the impact of your changes on a representative set of hardware.
- Use continuous integration/continuous deployment (CI/CD) to schedule and run your tests.
  - Integrating AWS Device Farm with your CI/CD pipeline to run cross-browser Selenium tests
  - Building and testing iOS and iPadOS apps with AWS DevOps and mobile services
- Continually review your testing results and make necessary improvements.

Resources

Related documents:

- AWS Device Farm device list
- Viewing the CloudWatch RUM dashboard

Related examples:

- AWS Device Farm Sample App for Android
- AWS Device Farm Sample App for iOS

- Appium Web tests for AWS Device Farm

Related videos:

- Optimize applications through end user insights with Amazon CloudWatch RUM

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Copyright © 2023 Amazon Web Services, Inc. or its affiliates.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.