Framework

AWS Well-Architected Framework

AWS Well-Architected Framework: Framework

Table of Contents

AWS Well-Architected Framework

Publication date: **October 3, 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 in and out 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

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

The following are design principles for operational excellence in the cloud:

- **Perform operations as code:** In the cloud, you can apply the same engineering discipline that you use for application code to your entire environment. You can define your entire workload (applications, infrastructure, etc.) as code and update it with code. You can script your operations procedures and automate their process by launching them in response to events. By performing operations as code, you limit human error and create consistent responses to events.
- **Make frequent, small, reversible changes:** Design workloads that are scalable and loosely coupled to permit components to be updated regularly. Automated deployment techniques together with smaller, incremental changes reduces the blast radius and allows for faster reversal when failures occur. This increases confidence to deliver beneficial changes to your workload while maintaining quality and adapting quickly to changes in market conditions.
- **Refine operations procedures frequently:** As you evolve your workloads, evolve your operations appropriately. As you use operations procedures, look for opportunities to improve them. Hold regular reviews and validate that all procedures are effective and that teams are familiar with them. Where gaps are identified, update procedures accordingly. Communicate procedural updates to all stakeholders and teams. Gamify your operations to share best practices and educate teams.
- **Anticipate failure:** Perform “pre-mortem” exercises to identify potential sources of failure so that they can be removed or mitigated. Test your failure scenarios and validate your understanding of their impact. Test your response procedures to ensure they are effective and that teams are familiar with their process. Set up regular game days to test workload 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.
- **Use managed services:** Reduce operational burden by using AWS managed services where possible. Build operational procedures around interactions with those services.
- **Implement observability for actionable insights:** Gain a comprehensive understanding of workload behavior, performance, reliability, cost, and health. Establish key performance indicators (KPIs) and leverage observability telemetry to make informed decisions and take prompt action when business outcomes are at risk. Proactively improve performance, reliability, and cost based on actionable observability data.

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

> **Note**
>
> All operational excellence questions have the OPS prefix as a shorthand for the pillar.

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 understan d their roles in the success of other teams, the role of other teams in their success, and have

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

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. Observability goes beyond simple monitoring, providing a comprehensive understanding of a system's internal workings based on its external outputs. Rooted in metrics, logs, and traces, observability offers profound insights into system behavior and dynamics. With effective observability, teams can discern patterns, anomalies, and trends, allowing them to proactively address potential issues and maintain optimal system health. Identifying key performance indicators (KPIs) is pivotal to ensure alignment between monitoring activities and business objectives. This alignment ensures that teams are making data-driven decisions using metrics that genuinely matter, optimizing both system performance and business outcomes. Furthermore, observability empowers businesses to be proactive rather than reactive. Teams can understand the cause-and-effect relationships within their systems, predicting and preventing issues rather than just reacting to them. As workloads evolve, it's essential to revisit and refine the observability strategy, ensuring it remains relevant and effective.

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 AWS 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 implement observability in your workload?

Implement observability in your workload so that you can understand its state and make data-driven decisions based on business requirements.

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

Observability allows you to focus on meaningful data and understand your workload's interactions and output. By concentrating on essential insights and eliminating unnecessary data, you maintain a straightforward approach to understanding workload performance. It's essential not only to collect data but also to interpret it correctly. Define clear baselines, set appropriate alert thresholds, and actively monitor for any deviations. A shift in a key metric, especially when correlated with other data, can pinpoint specific problem areas. With observability, you're better equipped to foresee and address potential challenges, ensuring that your workload operates smoothly and meets business needs.

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 utilize workload observability in your organization?

Ensure optimal workload health by leveraging observability. Utilize relevant metrics, logs, and traces to gain a comprehensive view of your workload's performance and address issues efficient ly.

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 QuickSight, 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

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 intellige nce 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**
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 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	Following the instructions in Using temporary credentia

Which user needs programmatic access?	To	By
	to the AWS CLI, AWS SDKs, or AWS APIs.	ls 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*.

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

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

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.

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

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 five best practice areas for performance efficiency in the cloud:

- **Architecture selection**
- **Compute and hardware**

- **Data management**
- **Networking and content delivery**
- **Process and culture**

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

Architecture selection

The optimal solution for a particular workload varies, and solutions often combine multiple approaches. Well-Architected workloads use multiple solutions and allow different features to improve performance.

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

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

PERF 1: How do you select appropriate cloud resources and architecture patterns for your workload?

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

Compute and hardware

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

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

- **Instances** are virtualized servers, 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 and use compute resources in your workload?

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

PERF 2: How do you select and use compute resources in your workload?

various components and turn on different features to improve performance. Selecting the wrong compute solution for an architecture can lead to lower performance efficiency.

Data management

The optimal data management solution for a particular system varies based on the kind of data type (block, file, or object), access patterns (random or sequential), required throughput, frequency of access (online, offline, archival), frequency of update (WORM, dynamic), and availability and durability constraints. Well-Architected workloads use purpose-built data stores which allow different features to improve performance.

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 store, manage, and access data in your workload?

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.

Networking and content delivery

The optimal networking 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.

On AWS, networking is virtualized and is available in a number of different types and configurations. This makes it easier to match your networking needs. AWS offers product features (for example, Enhanced Networking, Amazon EC2 networking 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 4: How do you select and configure networking resources in your workload?

This question includes guidance and best practices to design, configure, and operate efficient networking and content delivery solutions in the cloud.

Process and culture

When architecting workloads, there are principles and practices that you can adopt to help you better run efficient high-performing cloud workloads. To adopt a culture that fosters performance efficiency of cloud workloads, consider these key principles and practices.

Consider these key principles to build this culture:

- **Infrastructure as code:** Define your infrastructure as code using approaches such as AWS CloudFormation templates. The use of templates allows you to place your infrastructure into source control alongside your application code and configurations. This allows you to apply the same practices you use to develop software in your infrastructure so you can iterate rapidly.
- **Deployment pipeline:** Use a continuous integration/continuous deployment (CI/CD) pipeline (for example, source code repository, build systems, deployment, and testing automation) to deploy your infrastructure. This allows you to deploy in a repeatable, consistent, and low-cost fashion as you iterate.
- **Well-defined metrics:** Set up and monitor metrics to capture key performance indicators (KPIs). We recommend that you use both technical and business metrics. For websites or mobile apps, key metrics are capturing time-to-first-byte or rendering. Other generally applicable metrics include thread count, garbage collection rate, and wait states. Business metrics, such as the aggregate cumulative cost per request, can alert you to ways to drive down costs. Carefully consider how you plan to interpret metrics. For example, you could choose the maximum or 99th percentile instead of the average.
- **Performance test automatically:** As part of your deployment process, automatically start performance tests after the quicker running tests have passed successfully. The automation should create a new environment, set up initial conditions such as test data, and then run a series of benchmarks and load tests. Results from these tests should be tied back to the build so you can track performance changes over time. For long-running tests, you can make this part of the pipeline asynchronous from the rest of the build. Alternatively, you could run performance tests overnight using Amazon EC2 Spot Instances.
- **Load generation:** You should create a series of test scripts that replicate synthetic or prerecorded user journeys. These scripts should be idempotent and not coupled, and you might need to include *pre-warming* scripts to yield valid results. As much as possible, your test scripts should replicate the behavior of usage in production. You can use software or software-as-a-service (SaaS) solutions to generate the load. Consider using AWS Marketplace solutions and Spot Instances — they can be cost-effective ways to generate the load.
- **Performance visibility:** Key metrics should be visible to your team, especially metrics against each build version. This allows you to see any significant positive or negative trend over time. You should also display metrics on the number of errors or exceptions to make sure you are testing a working system.
- **Visualization:** Use visualization techniques that make it clear where performance issues, hot spots, wait states, or low utilization is occurring. Overlay performance metrics over architecture diagrams — call graphs or code can help identify issues quickly.

- **Regular review process:** Architectures performing poorly is usually the result of a non-existent or broken performance review process. If your architecture is performing poorly, implementing a performance review process allows you to drive iterative improvement.
- **Continual optimization:** Adopt a culture to continually optimize the performance efficiency of your cloud workload.

The following question focuses on these considerations for performance efficiency.

PERF 5: What process do you use to support more performance efficiency for your workload?

When architecting workloads, there are principles and practices that you can adopt to help you better run efficient high-performing cloud workloads. To adopt a culture that fosters performan ce efficiency of cloud workloads, consider these key principles and practices.

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

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 QuickSight 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 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

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 question focuses 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 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
- Ben Mergen, Senior Cost Lead Solutions Architect, Amazon Web Services
- Chris Kozlowski, Senior Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Alex Livingstone, Principal Specialist Solutions Architect, Cloud Operations, Amazon Web Services
- Paul Moran, Principal Technologist, Enterprise Support, Amazon Web Services
- Peter Mullen, Advisory Consultant, Professional Services, Amazon Web Services
- Chris Pates, Senior Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Arvind Raghunathan, Principal Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Sam Mokhtari, Senior Efficiency Lead Solutions Architect, 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
Major update	Major performance pillar restructure to bring number of best practice areas to five. Large update to best practices and guidance in the security pillar in Incident response (SEC 10). Major content changes and consolidation in operational excellence areas OPS 04, 05, 06, 08, and 09. Guidance updates throughout the cost optimization and reliability pillars. Minor updates to sustainability pillar risk levels.	October 3, 2023
Updates for new Framework	Best practices updated with prescriptive guidance and new best practices added. New questions added to the Security and Cost Optimization 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-Architected Tool, links to AWS Well-Architected Labs, and AWS Well-Architected Partners, minor fixes to enable multiple language version of framework.	July 1, 2019
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 descriptive 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 excellence 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

Note

To subscribe to RSS updates, you must have an RSS plugin enabled for the browser that you are using.

Framework versions:

- 2023-10-03 (current)
- 2023-04-10
- 2022-03-31

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.
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

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. AWS Chatbot 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:

- AWS Chatbot
- 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 implement observability in your workload?
- 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 implement observability in your workload?

Implement observability in your workload so that you can understand its state and make data-driven decisions based on business requirements.

Best practices

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS04-BP03 Implement user experience telemetry
- OPS04-BP04 Implement dependency telemetry
- OPS04-BP05 Implement distributed tracing

OPS04-BP01 Identify key performance indicators

Implementing observability in your workload starts with understanding its state and making data-driven decisions based on business requirements. One of the most effective ways to ensure alignment between monitoring activities and business objectives is by defining and monitoring key performance indicators (KPIs).

Desired outcome: Efficient observability practices that are tightly aligned with business objectives, ensuring that monitoring efforts are always in service of tangible business outcomes.

Common anti-patterns:

- Undefined KPIs: Working without clear KPIs can lead to monitoring too much or too little, missing vital signals.
- Static KPIs: Not revisiting or refining KPIs as the workload or business objectives evolve.
- Misalignment: Focusing on technical metrics that don’t correlate directly with business outcomes or are harder to correlate with real-world issues.

Benefits of establishing this best practice:

- Ease of issue identification: Business KPIs often surface issues more clearly than technical metrics. A dip in a business KPI can pinpoint a problem more effectively than sifting through numerous technical metrics.
- Business alignment: Ensures that monitoring activities directly support business objectives.
- Efficiency: Prioritize monitoring resources and attention on metrics that matter.
- Proactivity: Recognize and address issues before they have broader business implications.

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

Implementation guidance

To effectively define workload KPIs:

1. **Start with business outcomes:** Before diving into metrics, understand the desired business outcomes. Is it increased sales, higher user engagement, or faster response times?
2. **Correlate technical metrics with business objectives:** Not all technical metrics have a direct impact on business outcomes. Identify those that do, but it's often more straightforward to identify an issue using a business KPI.
3. **Use Amazon CloudWatch:** Employ CloudWatch to define and monitor metrics that represent your KPIs.

4. **Regularly review and update KPIs:** As your workload and business evolve, keep your KPIs relevant.
5. **Involve stakeholders:** Involve both technical and business teams in defining and reviewing KPIs.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- the section called "OPS04-BP02 Implement application telemetry"
- the section called "OPS04-BP03 Implement user experience telemetry"
- the section called "OPS04-BP04 Implement dependency telemetry"
- the section called "OPS04-BP05 Implement distributed tracing"

Related documents:

- AWS Observability Best Practices
- CloudWatch User Guide
- AWS Observability Skill Builder Course

Related videos:

- Developing an observability strategy

Related examples:

- One Observability Workshop

OPS04-BP02 Implement application telemetry

Application telemetry serves as the foundation for observability of your workload. It's crucial to emit telemetry that offers actionable insights into the state of your application and the achievement of both technical and business outcomes. From troubleshooting to measuring the impact of a new feature or ensuring alignment with business key performance indicators (KPIs), application telemetry informs the way you build, operate, and evolve your workload.

Metrics, logs, and traces form the three primary pillars of observability. These serve as diagnostic tools that describe the state of your application. Over time, they assist in creating baselines and identifying anomalies. However, to ensure alignment between monitoring activities and business objectives, it's pivotal to define and monitor KPIs. Business KPIs often make it easier to identify issues compared to technical metrics alone.

Other telemetry types, like real user monitoring (RUM) and synthetic transactions, complement these primary data sources. RUM offers insights into real-time user interactions, whereas synthetic transactions simulate potential user behaviors, helping detect bottlenecks before real users encounter them.

Desired outcome: Derive actionable insights into the performance of your workload. These insights allow you to make proactive decisions about performance optimization, achieve increased workload stability, streamline CI/CD processes, and utilize resources effectively.

Common anti-patterns:

- Incomplete observability: Neglecting to incorporate observability at every layer of the workload, resulting in blind spots that can obscure vital system performance and behavior insights.
- Fragmented data view: When data is scattered across multiple tools and systems, it becomes challenging to maintain a holistic view of your workload's health and performance.
- User-reported issues: A sign that proactive issue detection through telemetry and business KPI monitoring is lacking.

Benefits of establishing this best practice:

- Informed decision-making: With insights from telemetry and business KPIs, you can make data-driven decisions.
- Improved operational efficiency: Data-driven resource utilization leads to cost-effectiveness.
- Enhanced workload stability: Faster detection and resolution of issues leading to improved uptime.
- Streamlined CI/CD processes: Insights from telemetry data facilitate refinement of processes and reliable code delivery.

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

Implementation guidance

To implement application telemetry for your workload, use AWS services like Amazon CloudWatch and AWS X-Ray. Amazon CloudWatch provides a comprehensive suite of monitoring tools, allowing you to observe your resources and applications in AWS and on-premises environments. It collects, tracks, and analyzes metrics, consolidates and monitors log data, and responds to changes in your resources, enhancing your understanding of how your workload operates. In tandem, AWS X-Ray lets you trace, analyze, and debug your applications, giving you a deep understanding of your workload's behavior. With features like service maps, latency distributions, and trace timelines, X-Ray provides insights into your workload's performance and the bottlenecks affecting it.

Implementation steps

1. **Identify what data to collect:** Ascertain the essential metrics, logs, and traces that would offer substantial insights into your workload's health, performance, and behavior.
2. **Deploy the CloudWatch agent:** The CloudWatch agent is instrumental in procuring system and application metrics and logs from your workload and its underlying infrastructure. The CloudWatch agent can also be used to collect OpenTelemetry or X-Ray traces and send them to X-Ray.
3. **Define and monitor business KPIs:** Establish custom metrics that align with your business outcomes.
4. **Instrument your application with AWS X-Ray:** In addition to deploying the CloudWatch agent, it's crucial to instrument your application to emit trace data. This process can provide further insights into your workload's behavior and performance.
5. **Standardize data collection across your application:** Standardize data collection practices across your entire application. Uniformity aids in correlating and analyzing data, providing a comprehensive view of your application's behavior.
6. **Analyze and act on the data:** Once data collection and normalization are in place, use Amazon CloudWatch for metrics and logs analysis, and AWS X-Ray for trace analysis. Such analysis can yield crucial insights into your workload's health, performance, and behavior, guiding your decision-making process.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP03 Implement user experience telemetry
- OPS04-BP04 Implement dependency telemetry
- OPS04-BP05 Implement distributed tracing

Related documents:

- AWS Observability Best Practices
- CloudWatch User Guide
- AWS X-Ray Developer Guide
- Instrumenting distributed systems for operational visibility
- AWS Observability Skill Builder Course
- What's New with Amazon CloudWatch
- What's New with AWS X-Ray

Related videos:

- AWS re:Invent 2022 - Observability best practices at Amazon
- AWS re:Invent 2022 - Developing an observability strategy

Related examples:

- One Observability Workshop
- AWS Solutions Library: Application Monitoring with Amazon CloudWatch

OPS04-BP03 Implement user experience telemetry

Gaining deep insights into customer experiences and interactions with your application is crucial. Real user monitoring (RUM) and synthetic transactions serve as powerful tools for this purpose. RUM provides data about real user interactions granting an unfiltered perspective of user satisfaction, while synthetic transactions simulate user interactions, helping in detecting potential issues even before they impact real users.

Desired outcome: A holistic view of the customer experience, proactive detection of issues, and optimization of user interactions to deliver seamless digital experiences.

Common anti-patterns:

- Applications without real user monitoring (RUM):
  - Delayed issue detection: Without RUM, you might not become aware of performance bottlenecks or issues until users complain. This reactive approach can lead to customer dissatisfaction.
  - Lack of user experience insights: Not using RUM means you lose out on crucial data that shows how real users interact with your application, limiting your ability to optimize the user experience.
- Applications without synthetic transactions:
  - Missed edge cases: Synthetic transactions help you test paths and functions that might not be frequently used by typical users but are critical to certain business functions. Without them, these paths could malfunction and go unnoticed.
  - Checking for issues when the application is not being used: Regular synthetic testing can simulate times when real users aren't actively interacting with your application, ensuring the system always functions correctly.

Benefits of establishing this best practice:

- Proactive issue detection: Identify and address potential issues before they impact real users.
- Optimized user experience: Continuous feedback from RUM aids in refining and enhancing the overall user experience.
- Insights on device and browser performance: Understand how your application performs across various devices and browsers, enabling further optimization.
- Validated business workflows: Regular synthetic transactions ensure that core functionalities and critical paths remain operational and efficient.
- Enhanced application performance: Leverage insights gathered from real user data to improve application responsiveness and reliability.

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

Implementation guidance

To leverage RUM and synthetic transactions for user activity telemetry, AWS offers services like Amazon CloudWatch RUM and Amazon CloudWatch Synthetics. Metrics, logs, and traces, coupled with user activity data, provide a comprehensive view of both the application's operational state and the user experience.

Implementation steps

1. **Deploy Amazon CloudWatch RUM:** Integrate your application with CloudWatch RUM to collect, analyze, and present real user data.
   a. Use the CloudWatch RUM JavaScript library to integrate RUM with your application.
   b. Set up dashboards to visualize and monitor real user data.
2. **Configure CloudWatch Synthetics:** Create canaries, or scripted routines, that simulate user interactions with your application.
   a. Define critical application workflows and paths.
   b. Design canaries using CloudWatch Synthetics scripts to simulate user interactions for these paths.
   c. Schedule and monitor canaries to run at specified intervals, ensuring consistent performance checks.
3. **Analyze and act on data:** Utilize data from RUM and synthetic transactions to gain insights and take corrective measures when anomalies are detected. Use CloudWatch dashboards and alarms to stay informed.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS04-BP04 Implement dependency telemetry
- OPS04-BP05 Implement distributed tracing

Related documents:

- Amazon CloudWatch RUM Guide
- Amazon CloudWatch Synthetics Guide

Related videos:

- Optimize applications through end user insights with Amazon CloudWatch RUM
- AWS on Air ft. Real-User Monitoring for Amazon CloudWatch

Related examples:

- One Observability Workshop
- Git Repository for Amazon CloudWatch RUM Web Client
- Using Amazon CloudWatch Synthetics to measure page load time

OPS04-BP04 Implement dependency telemetry

Dependency telemetry is essential for monitoring the health and performance of the external services and components your workload relies on. It provides valuable insights into reachability, timeouts, and other critical events related to dependencies such as DNS, databases, or third-party APIs. By instrumenting your application to emit metrics, logs and traces about these dependencies, you gain a clearer understanding of potential bottlenecks, performance issues, or failures that might impact your workload.

Desired outcome: The dependencies your workload relies on are performing as expected, allowing you to proactively address issues and ensure optimal workload performance.

Common anti-patterns:

- Overlooking external dependencies: Focusing only on internal application metrics while neglecting metrics related to external dependencies.
- Lack of proactive monitoring: Waiting for issues to arise instead of continuously monitoring dependency health and performance.
- Siloed monitoring: Using multiple, disparate monitoring tools which can result in fragmented and inconsistent views of dependency health.

Benefits of establishing this best practice:

- Improved workload reliability: By ensuring that external dependencies are consistently available and performing optimally.

- Faster issue detection and resolution: Proactively identifying and addressing issues with dependencies before they impact the workload.
- Comprehensive view: Gaining a holistic view of both internal and external components that influence workload health.
- Enhanced workload scalability: By understanding the scalability limits and performance characteristics of external dependencies.

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

Implementation guidance

Implement dependency telemetry by starting with identifying the services, infrastructure, and processes that your workload depends on. Quantify what good conditions look like when those dependencies are functioning as expected, and then determine what data is needed to measure those. With that information, you can craft dashboards and alerts that provide insights to your operations teams on the state of those dependencies. Use AWS tools to discover and quantify the impacts when dependencies cannot deliver as needed. Continually revisit your strategy to account for changes in priorities, goals, and gained insights.

Implementation steps

To implement dependency telemetry effectively:

1. **Identify external dependencies:** Collaborate with stakeholders to pinpoint the external dependencies your workload relies on. External dependencies can encompass services like external databases, third-party APIs, network connectivity routes to other environments, and DNS services. The first step towards effective dependency telemetry is being comprehensive in understanding what those dependencies are.
2. **Develop a monitoring strategy:** Once you have a clear picture of your external dependencies, architect a monitoring strategy tailored to them. This involves understanding the criticality of each dependency, its expected behavior, and any associated service-level agreements or targets (SLA or SLTs). Set up proactive alerts to notify you of status changes or performance deviations.
3. **Leverage Amazon CloudWatch Internet Monitor:** It offers insights into the global internet, helping to understand outages or disruptions that might impact your external dependencies.
4. **Stay informed with AWS Health Dashboard:** It provides alerts and remediation guidance when AWS is experiencing events that may impact your services.

5. **Instrument your application with AWS X-Ray:** AWS X-Ray provides insights into how applications and their underlying dependencies are performing. By tracing requests from start to end, you can identify bottlenecks or failures in the external services or components your application relies on.
6. **Use Amazon DevOps Guru:** This machine learning-driven service identifies operational issues, predicts when critical issues might occur, and recommends specific actions to take. It's invaluable for gaining insights into dependencies and determining that they're not the source of operational problems.
7. **Monitor regularly:** Continually monitor metrics and logs related to external dependencies. Set up alerts for unexpected behavior or degraded performance.
8. **Validate after changes:** Whenever there's an update or change in any of the external dependencies, validate their performance and check their alignment with your application's requirements.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS04-BP03 Implement user experience telemetry
- OPS04-BP05 Implement distributed tracing

Related documents:

- What is AWS Health?
- Using Amazon CloudWatch Internet Monitor
- AWS X-Ray Developer Guide
- Amazon DevOps Guru User Guide

Related videos:

- Visibility into how internet issues impact app performance

- Introduction to Amazon DevOps Guru

Related examples:

- Gaining operational insights with AIOps using Amazon DevOps Guru
- AWS Health Aware

OPS04-BP05 Implement distributed tracing

Distributed tracing offers a way to monitor and visualize requests as they traverse through various components of a distributed system. By capturing trace data from multiple sources and analyzing it in a unified view, teams can better understand how requests flow, where bottlenecks exist, and where optimization efforts should focus.

Desired outcome: Achieve a holistic view of requests flowing through your distributed system, allowing for precise debugging, optimized performance, and improved user experiences.

Common anti-patterns:

- Inconsistent instrumentation: Not all services in a distributed system are instrumented for tracing.
- Ignoring latency: Only focusing on errors and not considering the latency or gradual performance degradations.

Benefits of establishing this best practice:

- Comprehensive system overview: Visualizing the entire path of requests, from entry to exit.
- Enhanced debugging: Quickly identifying where failures or performance issues occur.
- Improved user experience: Monitoring and optimizing based on actual user data, ensuring the system meets real-world demands.

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

Implementation guidance

Begin by identifying all of the elements of your workload that require instrumentation. Once all components are accounted for, leverage tools such as AWS X-Ray and OpenTelemetry to gather trace data for analysis with tools like X-Ray and Amazon CloudWatch ServiceLens Map. Engage in regular reviews with developers, and supplement these discussions with tools like Amazon DevOps Guru, X-Ray Analytics and X-Ray Insights to help uncover deeper findings. Establish alerts from trace data to notify when outcomes, as defined in the workload monitoring plan, are at risk.

Implementation steps

To implement distributed tracing effectively:

1. **Adopt AWS X-Ray:** Integrate X-Ray into your application to gain insights into its behavior, understand its performance, and pinpoint bottlenecks. Utilize X-Ray Insights for automatic trace analysis.
2. **Instrument your services:** Verify that every service, from an AWS Lambda function to an EC2 instance, sends trace data. The more services you instrument, the clearer the end-to-end view.
3. **Incorporate CloudWatch Real User Monitoring and synthetic monitoring:** Integrate Real User Monitoring (RUM) and synthetic monitoring with X-Ray. This allows for capturing real-world user experiences and simulating user interactions to identify potential issues.
4. **Use the CloudWatch agent:** The agent can send traces from either X-Ray or OpenTelemetry, enhancing the depth of insights obtained.
5. **Use Amazon DevOps Guru:** DevOps Guru uses data from X-Ray, CloudWatch, AWS Config, and AWS CloudTrail to provide actionable recommendations.
6. **Analyze traces:** Regularly review the trace data to discern patterns, anomalies, or bottlenecks that might impact your application's performance.
7. **Set up alerts:** Configure alarms in CloudWatch for unusual patterns or extended latencies, allowing proactive issue addressing.
8. **Continuous improvement:** Revisit your tracing strategy as services are added or modified to capture all relevant data points.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS04-BP03 Implement user experience telemetry
- OPS04-BP04 Implement dependency telemetry

Related documents:

- AWS X-Ray Developer Guide
- Amazon CloudWatch agent User Guide
- Amazon DevOps Guru User Guide

Related videos:

- Use AWS X-Ray Insights
- AWS on Air ft. Observability: Amazon CloudWatch and AWS X-Ray

Related examples:

- Instrumenting your Application with 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.

Desired outcome: Your teams collaborate on code. When merged, the code is consistent and no changes are lost. Errors are easily reverted through correct versioning.

Common anti-patterns:

- You have been developing and storing your code on your workstation. You have had an unrecoverable storage failure on the workstation and 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 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 shapes your future work. Someone has accidentally saved their shopping list over the final report. You are unable to revert the change and have to recreate the report.

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

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

Implementation guidance

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.

Resources

Related best practices:

- OPS05-BP04 Use build and deployment management systems

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: Your software changes are tested before they are delivered. Developers have access to test results and validations. 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 AWS 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: Change fail rate of software deployments are 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
- OPS05-BP06 Share design standards
- OPS05-BP10 Fully automate integration and deployment

Related documents:

- Adopt a test-driven development approach
- Automated AWS 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. 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.

Configurations should be deployed in a known and consistent state. You should use automated inspection to continually monitor resource configurations across environments and regions. These controls should be defined as code and management automated to ensure rules are consistently appplied across environments. Changes to configurations should be updated through agreed change control procedures and applied consistently, honoring version control. Application configuration should be managed independantly of application and infrastructure code. This allows for consistent deployment across multiple environments. Configuration changes do not result in rebuilding or redeploying the application.

Desired outcome: You configure, validate, and deploy as part of your continuous integration, continuous delivery (CI/CD) pipeline. You monitor to validate configurations are correct. This minimizes any impact to end users and customers.

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.
- You push a pre-production configuration into production through CI/CD without validation. You expose users and customers to incorrect data and services.

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. Configuration management systems provide assurances with regards to governance, compliance, and regulatory requirements.

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

Implementation guidance

Configuration management systems are used to track and implement changes to application and environment configurations. Configuration management systems are also used to reduce errors caused by manual processes, make configuration changes repeatable and auditable, and reduce the level of effort.

On AWS, you can use AWS Config to continually 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.

For dynamic configurations in your applications running on Amazon EC2 instances, AWS Lambda, containers, mobile applications, or IoT devices, you can use AWS AppConfig to configure, validate, deploy, and monitor them across your environments.

Implementation steps

1. Identify configuration owners.
   a. Make configurations owners aware of any compliance, governance, or regulatory needs.
2. Identify configuration items and deliverables.
   a. Configuration items are all application and environmental configurations affected by a deployment within your CI/CD pipeline.
   b. Deliverables include success criteria, validation, and what to monitor.
3. Select tools for configuration management based on your business requirements and delivery pipeline.
4. Consider a weighted deployments such are canary deployments for significant configuration changes to minimise the impact of incorrect configurations.
5. Integrate your configuration management into your CI/CD pipeline.
6. Validate all changes pushed.

Resources

Related best practices:

- OPS06-BP01 Plan for unsuccessful changes
- OPS06-BP02 Test deployments
- OPS06-BP03 Employ safe deployment strategies
- OPS06-BP04 Automate testing and rollback

Related documents:

- AWS Control Tower
- AWS Landing Zone Accelerator
- AWS Config
- What is AWS Config?
- AWS AppConfig
- What is AWS CloudFormation?
- AWS Developer Tools

Related videos:

- AWS re:Invent 2022 - Proactive governance and compliance for AWS workloads
- AWS re:Invent 2020: Achieve compliance as code using AWS Config
- Manage and Deploy Application Configurations with AWS AppConfig

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).

Desired outcome: Your build and deployment management systems support your organization's continuous integration continuous delivery (CI/CD) system that provide capabilities for automating safe rollouts with the correct configurations.

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

Build and deployment management systems are used to track and implement change, reduce errors caused by manual processes, and reduce the level of effort required for safe deployments. Fully automate the integration and deployment pipeline from code check-in through build, testing, deployment, and validation. This reduces lead time, decreases cost, encourages increased frequency of change, reduces the level of effort, and increases collaboration.

Implementation steps

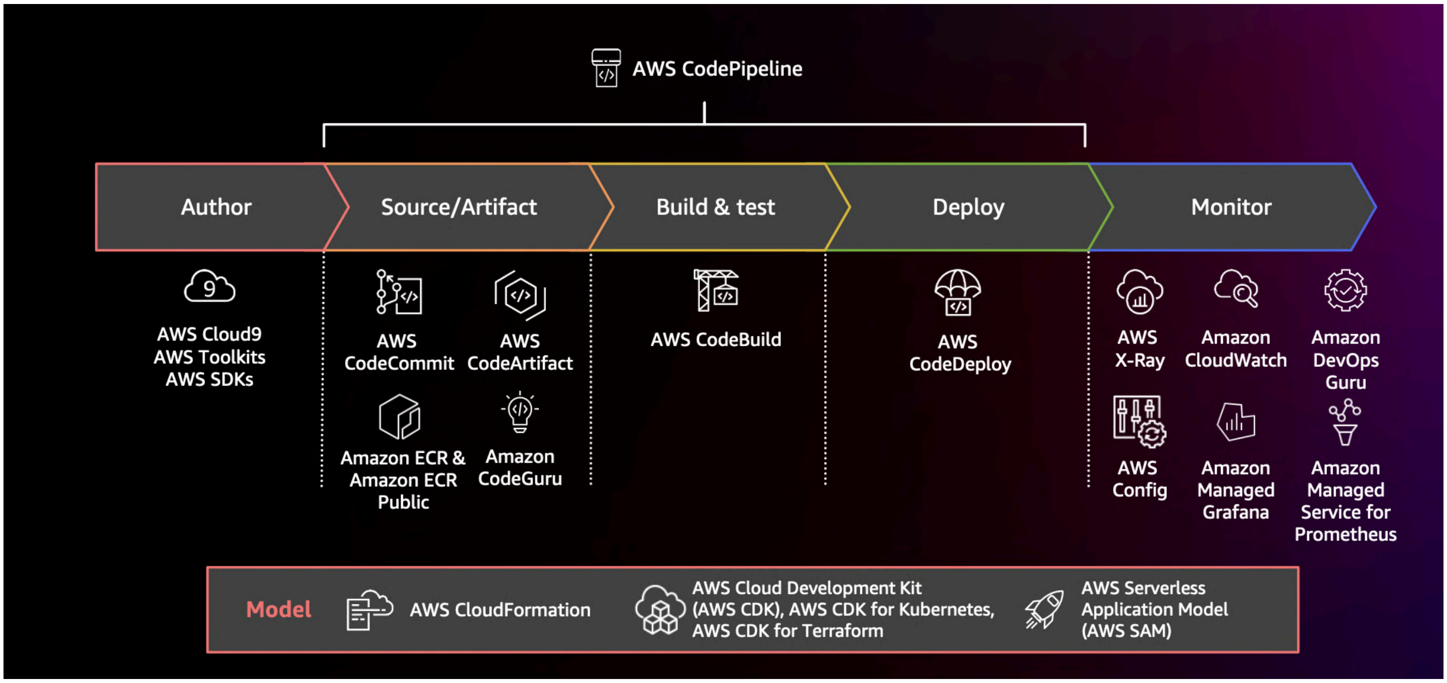


Diagram showing a CI/CD pipeline using AWS CodePipeline and related services

1. Use AWS CodeCommit to version control, store, and manage assets (such as documents, source code, and binary files).
2. Use CodeBuild to compile your source code, runs unit tests, and produces artifacts that are ready to deploy.
3. Use CodeDeploy as a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, serverless AWS Lambda functions, or Amazon ECS.
4. Monitor your deployments.

Resources

Related best practices:

- OPS06-BP04 Automate testing and rollback

Related documents:

- AWS Developer Tools
- What is AWS CodeCommit?

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

Related videos:

- AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps 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, scale, 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.

Amazon EC2 Image Builder provides pipelines to update machine images. As a part of patch management, consider Amazon Machine Images (AMIs) using an AMI image pipeline or container images with a Docker image pipeline, while AWS Lambda provides patterns for custom runtimes and additional libraries to remove vulnerabilities.

You should manage updates to Amazon Machine Images for Linux or Windows Server images using Amazon EC2 Image Builder. You can use Amazon Elastic Container Registry (Amazon ECR) with your existing pipeline to manage Amazon ECS images and manage Amazon EKS images. 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 Systems Manager Maintenance Windows.

Desired outcome: Your AMI and container images are patched, up-to-date, and ready for launch. You are able to track the status of all deployed images and know patch compliance. You are able to report on current status and have a process to meet your compliance needs.

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-shelf 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 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.
- You are notified of a zero-day vulnerability requiring an emergency fix and you have to patch all your environments manually.

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 can scale and report on patch levels. This provides assurances around security patching and ensure clear visibility on the status of known fixes being in place. This encourages adoption of desired features and capabilities, the rapid 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 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 avoid errors caused by manual processes, and lower the level of effort to patch.

Implementation steps

For Amazon EC2 Image Builder:

1. Using Amazon EC2 Image Builder, specify pipeline details:
    a. Create an image pipeline and name it
    b. Define pipeline schedule and time zone
    c. Configure any dependencies
2. Choose a recipe:
    a. Select existing recipe or create a new one
    b. Select image type
    c. Name and version your recipe
    d. Select your base image
    e. Add build components and add to target registry
3. Optional - define your infrastructure configuration.
4. Optional - define configuration settings.
5. Review settings.
6. Maintain recipe hygiene regularly.

For Systems Manager Patch Manager:

1. Create a patch baseline.
2. Select a pathing operations method.
3. Enable compliance reporting and scanning.

Resources

Related best practices:

- OPS06-BP04 Automate testing and rollback

Related documents:

- What is Amazon EC2 Image Builder
- Create an image pipeline using the Amazon EC2 Image Builder
- Create a container image pipeline
- AWS Systems Manager Patch Manager

- Working with Patch Manager
- Working with patch compliance reports
- AWS Developer Tools

Related videos:

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

Related examples:

- Well-Architected Labs - Inventory and Patch Management
- AWS Systems Manager Patch Manager tutorials

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 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 owns developing and updating design standards. This team should 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 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 operates as intended when deployed.

Desired outcome: You have multiple environments that reflect your compliance and governance needs. You test and promote code through environments on your path to production.

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, you test in your production environment.
- 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: 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 and 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.

Resources

Related documents:

- Instance Scheduler on AWS
- What is AWS CloudFormation?

OPS05-BP09 Make frequent, small, reversible changes

Frequent, small, and reversible changes reduce the scope and impact of a change. When used in conjunction with change management systems, configuration management systems, and build and delivery systems frequent, small, and reversible changes reduce the scope and impact of a change. This results in more effective troubleshooting and faster remediation with the option to roll back changes.

Common anti-patterns:

- You deploy a new version of your application quarterly with a change window that means a core service is turned off.
- You frequently make changes to your database schema without tracking changes in your management systems.
- You perform manual in-place updates, overwriting existing installations and configurations, and have no clear roll-back plan.

Benefits of establishing this best practice: Development efforts are faster by deploying small changes frequently. When the changes are small, it is much easier to identify if they have unintended consequences, and they are easier to reverse. When the changes are reversible, there is less risk to implementing the change, as recovery is simplified. The change process has a reduced risk and the impact of a failed change is reduced.

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

Implementation guidance

Use frequent, small, and reversible changes to 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.

Resources

Related best practices:

- OPS05-BP03 Use configuration management systems
- OPS05-BP04 Use build and deployment management systems
- OPS06-BP04 Automate testing and rollback

Related documents:

- Implementing Microservices on AWS
- Microservices - Observability

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.

Desired outcome: Developers use tools to deliver code and promote through to production. Developers do not have to log into the AWS Management Console to deliver updates. There is a full audit trail of change and configuration, meeting the needs of governance and compliance. Processes are repeatable and are standardized across teams. Developers are free to focus on development and code pushes, increasing productivity.

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 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.
- As a developer, you log into the AWS Management Console to create a new development environment using non-standard methods and systems.

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. You increase the speed of delivery as you promote through to production.

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

Implementation guidance

You 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, reduces the level of effort, increases the speed to market, results in increased productivity, and increases the security of your code as you promote through to production.

Resources

Related best practices:

- OPS05-BP03 Use configuration management systems
- OPS05-BP04 Use build and deployment management systems

Related documents:

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

Related videos:

- AWS re\:Invent 2022 - AWS Well-Architected best practices for DevOps 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 deployments
- OPS06-BP03 Employ safe deployment strategies
- OPS06-BP04 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 the deployment causes an undesired outcome. Having a policy to establish such a plan helps all teams develop strategies to recover from failed changes. Some example strategies are deployment and rollback steps, change policies, feature flags, traffic isolation, and traffic shifting. A single release may include multiple related component changes. The strategy should provide the ability to withstand or recover from a failure of any component change.

Desired outcome: You have prepared a detailed recovery plan for your change in the event it is unsuccessful. In addition, you have reduced the size of your release to minimize the potential impact on other workload components. As a result, you have reduced your business impact by shortening the potential downtime caused by a failed change and increased the flexibility and efficiency of recovery times.

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 rollback the change and impact the active users or wait to rollback 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 rollback everything or try to fix the inaccessible subnet. While you are making that determination, the subnet remains unreachable.
- Your systems are not architected in a way that allows them to be updated with smaller releases. As a result, you have difficulty in reversing those bulk changes during a failed deployment.
- You do not use infrastructure as code (IaC) and you made manual updates to your infrastructure that resulted in an undesired configuration. You are unable to effectively track and revert the manual changes.
- Because you have not measured increased frequency of your deployments, your team is not incentivized to reduce the size of their changes and improve their rollback plans for each change, leading to more risk and increased failure rates.
- You do not measure the total duration of an outage caused by unsuccessful changes. Your team is unable to prioritize and improve its deployment process and recovery plan effectiveness.

Benefits of establishing this best practice: Having a plan to recover from unsuccessful changes minimizes the mean time to recover (MTTR) and reduces your business impact.

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

Implementation guidance

A consistent, documented policy and practice adopted by release teams allows an organization to plan what should happen if unsuccessful changes occur. The policy should allow for fixing forward in specific circumstances. In either situation, a fix forward or rollback plan should be well documented and tested before deployment to live production so that the time it takes to revert a change is minimized.

Implementation steps

1. Document the policies that require teams to have effective plans to reverse changes within a specified period.
   a. Policies should specify when a fix-forward situation is allowed.
   b. Require a documented rollback plan to be accessible by all involved.
   c. Specify the requirements to rollback (for example, when it is found that unauthorized changes have been deployed).
2. Analyze the level of impact of all changes related to each component of a workload.

   a. Allow repeatable changes to be standardized, templated, and preauthorized if they follow a consistent workflow that enforces change policies.

   b. Reduce the potential impact of any change by making the size of the change smaller so recovery takes less time and causes less business impact.

   c. Ensure rollback procedures revert code to the known good state to avoid incidents where possible.

3. Integrate tools and workflows to enforce your policies programatically.

4. Make data about changes visible to other workload owners to improve the speed of diagnosis of any failed change that cannot be rolled back.

   a. Measure success of this practice using visible change data and identify iterative improvements.

5. Use monitoring tools to verify the success or failure of a deployment to speed up decision-making on rolling back.

6. Measure your duration of outage during an unsuccessful change to continually improve your recovery plans.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS06-BP04 Automate testing and rollback

Related documents:

- AWS Builders Library | Ensuring Rollback Safety During Deployments
- AWS Whitepaper | Change Management in the Cloud

Related videos:

- re:Invent 2019 | Amazon’s approach to high-availability deployment

OPS06-BP02 Test deployments

Test release procedures in pre-production by using the same deployment configuration, security controls, steps, and procedures as in production. Validate that all deployed steps are completed as expected, such as inspecting files, configurations, and services. Further test all changes with functional, integration, and load tests, along with any monitoring such as health checks. By doing these tests, you can identify deployment issues early with an opportunity to plan and mitigate them prior to production.

You can create temporary parallel environments for testing every change. Automate the deployment of the test environments using infrastructure as code (IaC) to help reduce amount of work involved and ensure stability, consistency, and faster feature delivery.

Desired outcome: Your organization adopts a test-driven development culture that includes testing deployments. This ensures teams are focused on delivering business value rather than managing releases. Teams are engaged early upon identification of deployment risks to determine the appropriate course of mitigation.

Common anti-patterns:

- During production releases, untested deployments cause frequent issues that require troubleshooting and escalation.
- Your release contains infrastructure as code (IaC) that updates existing resources. You are unsure if the IaC runs successfully or causes impact to the resources.
- You deploy a new feature to your application. It doesn't work as intended and there is no visibility until it gets reported by impacted users.
- You update your certificates. You accidentally install the certificates to the wrong components, which goes undetected and impacts website visitors because a secure connection to the website can't be established.

Benefits of establishing this best practice: Extensive testing in pre-production of deployment procedures, and the changes introduced by them, minimizes the potential impact to production caused by the deployments steps. This increases confidence during production release and minimizes operational support without slowing down velocity of the changes being delivered.

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

Implementation guidance

Testing your deployment process is as important as testing the changes that result from your deployment. This can be achieved by testing your deployment steps in a pre-production environment that mirrors production as closely as possible. Common issues, such as incomplete or incorrect deployment steps, or misconfigurations, can be caught as a result before going to production. In addition, you can test your recovery steps.

Customer example

As part of their continuous integration and continuous delivery (CI/CD) pipeline, AnyCompany Retail performs the defined steps needed to release infrastructure and software updates for its customers in a production-like environment. The pipeline is comprised of pre-checks to detect drift (detecting changes to resources performed outside of your IaC) in resources prior to deployment, as well as validate actions that the IaC takes upon its initiation. It validates deployment steps, like verifying that certain files and configurations are in place and services are in running states and are responding correctly to health checks on local host before re-registering with the load balancer. Additionally, all changes flag a number of automated tests, such as functional, security, regression, integration, and load tests.

Implementation steps

1. Perform pre-install checks to mirror the pre-production environment to production.
   a. Use drift detection to detect when resources have been changed outside of AWS CloudFormation.
   b. Use change sets to validate that the intent of a stack update matches the actions that AWS CloudFormation takes when the change set is initiated.
2. This triggers a manual approval step in AWS CodePipeline to authorize the deployment to the pre-production environment.
3. Use deployment configurations such as AWS CodeDeploy AppSpec files to define deployment and validation steps.
4. Where applicable, integrate AWS CodeDeploy with other AWS services or integrate AWS CodeDeploy with partner product and services.
5. Monitor deployments using Amazon CloudWatch, AWS CloudTrail, and Amazon SNS event notifications.
6. Perform post-deployment automated testing, including functional, security, regression, integration, and load testing.

7. Troubleshoot deployment issues.
8. Successful validation of preceding steps should initiate a manual approval workflow to authorize deployment to production.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS05-BP02 Test and validate changes

Related documents:

- AWS Builders' Library | Automating safe, hands-off deployments | Test Deployments
- AWS Whitepaper | Practicing Continuous Integration and Continuous Delivery on AWS
- The Story of Apollo - Amazon's Deployment Engine
- How to test and debug AWS CodeDeploy locally before you ship your code
- Integrating Network Connectivity Testing with Infrastructure Deployment

Related videos:

- re:Invent 2020 | Testing software and systems at Amazon

Related examples:

- Tutorial | Deploy and Amazon ECS service with a validation test

OPS06-BP03 Employ safe deployment strategies

Safe production roll-outs control the flow of beneficial changes with an aim to minimize any perceived impact for customers from those changes. The safety controls provide inspection mechanisms to validate desired outcomes and limit the scope of impact from any defects introduced by the changes or from deployment failures. Safe roll-outs may include strategies such as feature-flags, one-box, rolling (canary releases), immutable, traffic splitting, and blue/green deployments.

Desired outcome: Your organization uses a continuous integration continuous delivery (CI/CD) system that provides capabilities for automating safe rollouts. Teams are required to use appropriate safe roll-out strategies.

Common anti-patterns:

- You deploy an unsuccessful change to all of production all at once. As a result, all customers are impacted simultaneously.
- A defect introduced in a simultaneous deployment to all systems requires an emergency release. Correcting it for all customers takes several days.
- Managing production release requires planning and participation of several teams. This puts constraints on your ability to frequently update features for your customers.
- 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.

Benefits of establishing this best practice: Automated deployments balance speed of roll-outs against delivering beneficial changes consistently to customers. Limiting impact prevents costly deployment failures and maximizes teams ability to efficiently respond to failures.

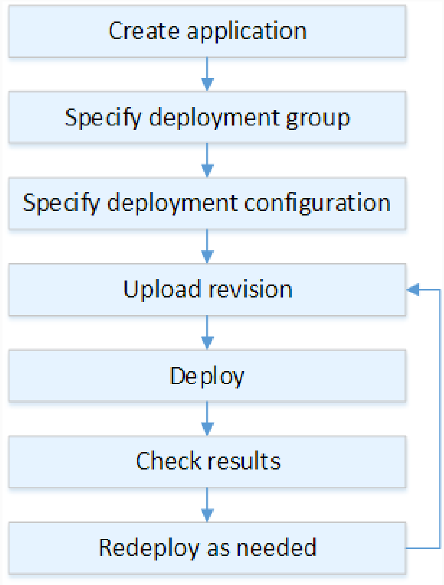
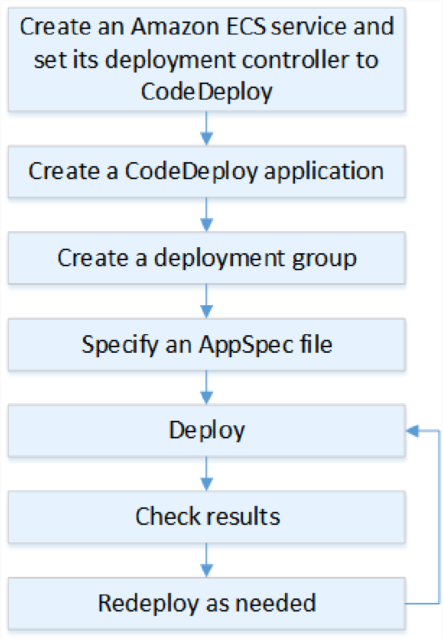
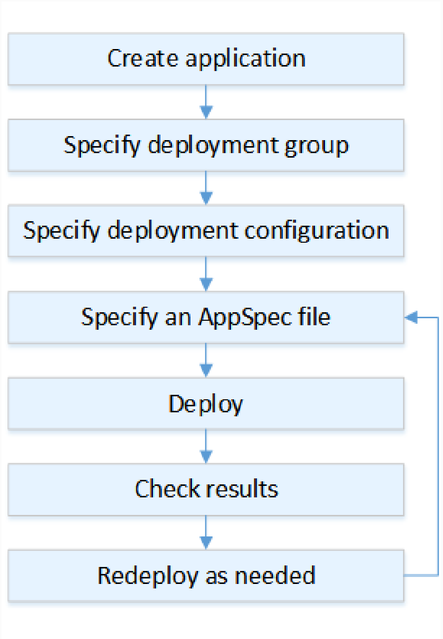
Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Continuous-delivery failures can lead to reduced service availability and bad customer experiences. To maximize the rate of successful deployments, implement safety controls in the end-to-end release process to minimize deployment errors, with a goal of achieving zero deployment failures.

Customer example

AnyCompany Retail is on a mission to achieve minimal to zero downtime deployments, meaning that there's no perceivable impact to its users during deployments. To accomplish this, the company has established deployment patterns (see the following workflow diagram), such as rolling and blue/green deployments. All teams adopt one or more of these patterns in their CI/CD pipeline.

CodeDeploy workflow for Amazon EC2	CodeDeploy workflow for Amazon ECS	CodeDeploy workflow for Lambda
		

Create an Amazon ECS service and set its deployment controller to CodeDeploy
Create a CodeDeploy application
Create a deployment group
Specify an AppSpec file
Deploy
Check results
Redeploy as needed

Implementation steps

1. Use an approval workflow to initiate the sequence of production roll-out steps upon promotion to production .
2. Use an automated deployment system such as AWS CodeDeploy. AWS CodeDeploy deployment options include in-place deployments for EC2/On-Premises and blue/green deployments for EC2/On-Premises, AWS Lambda, and Amazon ECS (see the preceding workflow diagram).
    a. Where applicable, integrate AWS CodeDeploy with other AWS services or integrate AWS CodeDeploy with partner product and services.
3. Use blue/green deployments for databases such as Amazon Aurora and Amazon RDS.
4. Monitor deployments using Amazon CloudWatch, AWS CloudTrail, and Amazon SNS event notifications.
5. Perform post-deployment automated testing including functional, security, regression, integration, and any load tests.
6. Troubleshoot deployment issues.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS05-BP02 Test and validate changes
- OPS05-BP09 Make frequent, small, reversible changes
- OPS05-BP10 Fully automate integration and deployment

Related documents:

- AWS Builders Library | Automating safe, hands-off deployments | Production deployments
- AWS Builders Library | My CI/CD pipeline is my release captain | Safe, automatic production releases
- AWS Whitepaper | Practicing Continuous Integration and Continuous Delivery on AWS | Deployment methods
- AWS CodeDeploy User Guide
- Working with deployment configurations in AWS CodeDeploy
- Set up an API Gateway canary release deployment
- Amazon ECS Deployment Types
- Fully Managed Blue/Green Deployments in Amazon Aurora and Amazon RDS
- Blue/Green deployments with AWS Elastic Beanstalk

Related videos:

- re:Invent 2020 | Hands-off: Automating continuous delivery pipelines at Amazon
- re:Invent 2019 | Amazon's Approach to high-availability deployment

Related examples:

- Try a Sample Blue/Green Deployment in AWS CodeDeploy
- Worlshop | Buiding CI/CD pipelines for Lambda canary deployments using AWS CDK
- Workshop | Blue/Green and Canary Deployment for EKS and ECS
- Workshop | Building a Cross-account CI/CD Pipeline

OPS06-BP04 Automate testing and rollback

To increase the speed, reliability, and confidence of your deployment process, have a strategy for automated testing and rollback capabilities in pre-production and production environments. Automate testing when deploying to production to simulate human and system interactions that verify the changes being deployed. Automate rollback to revert back to a previous known good state quickly. The rollback should be initiated automatically on pre-defined conditions such as when the desired outcome of your change is not achieved or when the automated test fails. Automating these two activities improves your success rate for your deployments, minimizes recovery time, and reduces the potential impact to the business.

Desired outcome: Your automated tests and rollback strategies are integrated into your continuous integration, continuous delivery (CI/CD) pipeline. Your monitoring is able to validate against your success criteria and initiate automatic rollback upon failure. This minimizes any impact to end users and customers. For example, when all testing outcomes have been satisfied, you promote your code into the production environment where automated regression testing is initiated, leveraging the same test cases. If regression test results do not match expectations, then automated rollback is initiated in the pipeline workflow.

Common anti-patterns:

- Your systems are not architected in a way that allows them to be updated with smaller releases. As a result, you have difficulty in reversing those bulk changes during a failed deployment.
- Your deployment process consists of a series of manual steps. After you deploy changes to your workload, you start post-deployment testing. After testing, you realize that your workload is inoperable and customers are disconnected. You then begin rolling back to the previous version. All of these manual steps delay overall system recovery and cause a prolonged impact to your customers.
- You spent time developing automated test cases for functionality that is not frequently used in your application, minimizing the return on investment in your automated testing capability.
- Your release is comprised of application, infrastructure, patches and configuration updates that are independent from one another. However, you have a single CI/CD pipeline that delivers all changes at once. A failure in one component forces you to revert all changes, making your rollback complex and inefficient.
- Your team completes the coding work in sprint one and begins sprint two work, but your plan did not include testing until sprint three. As a result, automated tests revealed defects from

sprint one that had to be resolved before testing of sprint two deliverables could be started and the entire release is delayed, devaluing your automated testing.

- Your automated regression test cases for the production release are complete, but you are not monitoring workload health. Since you have no visibility into whether or not the service has restarted, you are not sure if rollback is needed or if it has already occurred.

Benefits of establishing this best practice: Automated testing increases the transparency of your testing process and your ability to cover more features in a shorter time period. By testing and validating changes in production, you are able to identify issues immediately. Improvement in consistency with automated testing tools allows for better detection of defects. By automatically rolling back to the previous version, the impact on your customers is minimized. Automated rollback ultimately inspires more confidence in your deployment capabilities by reducing business impact. Overall, these capabilities reduce time-to-delivery while ensuring quality.

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

Implementation guidance

Automate testing of deployed environments to confirm desired outcomes more quickly. Automate rollback to a previous known good state when pre-defined outcomes are not achieved to minimize recovery time and reduce errors caused by manual processes. Integrate testing tools with your pipeline workflow to consistently test and minimize manual inputs. Prioritize automating test cases, such as those that mitigate the greatest risks and need to be tested frequently with every change. Additionally, automate rollback based on specific conditions that are pre-defined in your test plan.

Implementation steps

1. Establish a testing lifecycle for your development lifecycle that defines each stage of the testing process from requirements planning to test case development, tool configuration, automated testing, and test case closure.
   a. Create a workload-specific testing approach from your overall test strategy.
   b. Consider a continuous testing strategy where appropriate throughout the development lifecycle.
2. Select automated tools for testing and rollback based on your business requirements and pipeline investments.

3. Decide which test cases you wish to automate and which should be performed manually. These can be defined based on business value priority of the feature being tested. Align all team members to this plan and verify accountability for performing manual tests.
    a. Apply automated testing capabilities to specific test cases that make sense for automation, such as repeatable or frequently run cases, those that require repetitive tasks, or those that are required across multiple configurations.
    b. Define test automation scripts as well as the success criteria in the automation tool so continued workflow automation can be initiated when specific cases fail.
    c. Define specific failure criteria for automated rollback.
4. Prioritize test automation to drive consistent results with thorough test case development where complexity and human interaction have a higher risk of failure.
5. Integrate your automated testing and rollback tools into your CI/CD pipeline.
    a. Develop clear success criteria for your changes.
    b. Monitor and observe to detect these criteria and automatically reverse changes when specific rollback criteria are met.
6. Perform different types of automated production testing, such as:
    a. A/B testing to show results in comparison to the current version between two user testing groups.
    b. Canary testing that allows you to roll out your change to a subset of users before releasing it to all.
    c. Feature-flag testing which allows a single feature of the new version at a time to be flagged on and off from outside the application so that each new feature can be validated one at a time.
    d. Regression testing to verify new functionality with existing interrelated components.
7. Monitor the operational aspects of the application, transactions, and interactions with other applications and components. Develop reports to show success of changes by workload so that you can identify what parts of the automation and workflow can be further optimized.
    a. Develop test result reports that help you make quick decisions on whether or not rollback procedures should be invoked.
    b. Implement a strategy that allows for automated rollback based upon pre-defined failure conditions that result from one or more of your test methods.
8. Develop your automated test cases to allow for reusability across future repeatable changes.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS06-BP01 Plan for unsuccessful changes
- OPS06-BP02 Test deployments

Related documents:

- AWS Builders Library | Ensuring rollback safety during deployments
- Redeploy and rollback a deployment with AWS CodeDeploy
- 8 best practices when automating your deployments with AWS CloudFormation

Related examples:

- Serverless UI testing using Selenium, AWS Lambda, AWS Fargate (Fargate), and AWS Developer Tools

Related videos:

- re:Invent 2020 | Hands-off: Automating continuous delivery pipelines at Amazon
- re:Invent 2019 | Amazon's Approach to high-availability deployment

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, 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.

AWS 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
- 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 deployments - 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 utilize workload observability in your organization?
- 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 utilize workload observability in your organization?

Ensure optimal workload health by leveraging observability. Utilize relevant metrics, logs, and traces to gain a comprehensive view of your workload's performance and address issues efficiently.

Best practices

- OPS08-BP01 Analyze workload metrics
- OPS08-BP02 Analyze workload logs
- OPS08-BP03 Analyze workload traces
- OPS08-BP04 Create actionable alerts
- OPS08-BP05 Create dashboards

OPS08-BP01 Analyze workload metrics

After implementing application telemetry, regularly analyze the collected metrics. While latency, requests, errors, and capacity (or quotas) provide insights into system performance, it's vital to prioritize the review of business outcome metrics. This ensures you're making data-driven decisions aligned with your business objectives.

Desired outcome: Accurate insights into workload performance that drive data-informed decisions, ensuring alignment with business objectives.

Common anti-patterns:

- Analyzing metrics in isolation without considering their impact on business outcomes.
- Over-reliance on technical metrics while sidelining business metrics.
- Infrequent review of metrics, missing out on real-time decision-making opportunities.

Benefits of establishing this best practice:

- Enhanced understanding of the correlation between technical performance and business outcomes.

- Improved decision-making process informed by real-time data.
- Proactive identification and mitigation of issues before they affect business outcomes.

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

Implementation guidance

Leverage tools like Amazon CloudWatch to perform metric analysis. AWS services such as CloudWatch anomaly detection and Amazon DevOps Guru can be used to detect anomalies, especially when static thresholds are unknown or when patterns of behavior are more suited for anomaly detection.

Implementation steps

1. **Analyze and review:** Regularly review and interpret your workload metrics.
   a. Prioritize business outcome metrics over purely technical metrics.
   b. Understand the significance of spikes, drops, or patterns in your data.
2. **Utilize Amazon CloudWatch:** Use Amazon CloudWatch for a centralized view and deep-dive analysis.
   a. Configure CloudWatch dashboards to visualize your metrics and compare them over time.
   b. Use percentiles in CloudWatch to get a clear view of metric distribution, which can help in defining SLAs and understanding outliers.
   c. Set up CloudWatch anomaly detection to identify unusual patterns without relying on static thresholds.
   d. Implement CloudWatch cross-account observability to monitor and troubleshoot applications that span multiple accounts within a Region.
   e. Use CloudWatch Metric Insights to query and analyze metric data across accounts and Regions, identifying trends and anomalies.
   f. Apply CloudWatch Metric Math to transform, aggregate, or perform calculations on your metrics for deeper insights.
3. **Employ Amazon DevOps Guru:** Incorporate Amazon DevOps Guru for its machine learning-enhanced anomaly detection to identify early signs of operational issues for your serverless applications and remediate them before they impact your customers.
4. **Optimize based on insights:** Make informed decisions based on your metric analysis to adjust and improve your workloads.

Level of effort for the Implementation Plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry

Related documents:

- The Wheel Blog - Emphasizing the importance of continually reviewing metrics
- Percentile are important
- Using AWS Cost Anomaly Detection
- CloudWatch cross-account observability
- Query your metrics with CloudWatch Metrics Insights

Related videos:

- Enable Cross-Account Observability in Amazon CloudWatch
- Introduction to Amazon DevOps Guru
- Continuously Analyze Metrics using AWS Cost Anomaly Detection

Related examples:

- One Observability Workshop
- Gaining operation insights with AIOps using Amazon DevOps Guru

OPS08-BP02 Analyze workload logs

Regularly analyzing workload logs is essential for gaining a deeper understanding of the operational aspects of your application. By efficiently sifting through, visualizing, and interpreting log data, you can continually optimize application performance and security.

Desired outcome: Rich insights into application behavior and operations derived from thorough log analysis, ensuring proactive issue detection and mitigation.

Common anti-patterns:

- Neglecting the analysis of logs until a critical issue arises.
- Not using the full suite of tools available for log analysis, missing out on critical insights.
- Solely relying on manual review of logs without leveraging automation and querying capabilities.

Benefits of establishing this best practice:

- Proactive identification of operational bottlenecks, security threats, and other potential issues.
- Efficient utilization of log data for continuous application optimization.
- Enhanced understanding of application behavior, aiding in debugging and troubleshooting.

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

Implementation guidance

Amazon CloudWatch Logs is a powerful tool for log analysis. Integrated features like CloudWatch Logs Insights and Contributor Insights make the process of deriving meaningful information from logs intuitive and efficient.

Implementation steps

1. **Set up CloudWatch Logs:** Configure applications and services to send logs to CloudWatch Logs.
2. **Set up CloudWatch Logs Insights:** Use CloudWatch Logs Insights to interactively search and analyze your log data.
    a. Craft queries to extract patterns, visualize log data, and derive actionable insights.
3. **Leverage Contributor Insights:** Use CloudWatch Contributor Insights to identify top talkers in high cardinality dimensions like IP addresses or user-agents.
4. **Implement CloudWatch Logs metric filters:** Configure CloudWatch log metric filters to convert log data into actionable metrics. This allows you to set alarms or further analyze patterns.
5. **Regular review and refinement:** Periodically review your log analysis strategies to capture all relevant information and continually optimize application performance.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS08-BP01 Analyze workload metrics

Related documents:

- Analyzing Log Data with CloudWatch Logs Insights
- Using CloudWatch Contributor Insights
- Creating and Managing CloudWatch Logs Log Metric Filters

Related videos:

- Analyze Log Data with CloudWatch Logs Insights
- Use CloudWatch Contributor Insights to Analyze High-Cardinality Data

Related examples:

- CloudWatch Logs Sample Queries
- One Observability Workshop

OPS08-BP03 Analyze workload traces

Analyzing trace data is crucial for achieving a comprehensive view of an application's operational journey. By visualizing and understanding the interactions between various components, performance can be fine-tuned, bottlenecks identified, and user experiences enhanced.

Desired outcome: Achieve clear visibility into your application's distributed operations, enabling quicker issue resolution and an enhanced user experience.

Common anti-patterns:

- Overlooking trace data, relying solely on logs and metrics.
- Not correlating trace data with associated logs.

- Ignoring the metrics derived from traces, such as latency and fault rates.

Benefits of establishing this best practice:

- Improve troubleshooting and reduce mean time to resolution (MTTR).
- Gain insights into dependencies and their impact.
- Swift identification and rectification of performance issues.
- Leveraging trace-derived metrics for informed decision-making.
- Improved user experiences through optimized component interactions.

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

Implementation guidance

AWS X-Ray offers a comprehensive suite for trace data analysis, providing a holistic view of service interactions, monitoring user activities, and detecting performance issues. Features like ServiceLens, X-Ray Insights, X-Ray Analytics, and Amazon DevOps Guru enhance the depth of actionable insights derived from trace data.

Implementation steps

The following steps offer a structured approach to effectively implementing trace data analysis using AWS services:

1. **Integrate AWS X-Ray:** Ensure X-Ray is integrated with your applications to capture trace data.
2. **Analyze X-Ray metrics:** Delve into metrics derived from X-Ray traces such as latency, request rates, fault rates, and response time distributions using the service map to monitor application health.
3. **Use ServiceLens:** Leverage the ServiceLens map for enhanced observability of your services and applications. This allows for integrated viewing of traces, metrics, logs, alarms, and other health information.
4. **Enable X-Ray Insights:**
   a. Turn on X-Ray Insights for automated anomaly detection in traces.
   b. Examine insights to pinpoint patterns and ascertain root causes, such as increased fault rates or latencies.
   c. Consult the insights timeline for a chronological analysis of detected issues.

5. **Use X-Ray Analytics:** X-Ray Analytics allows you to thoroughly explore trace data, pinpoint patterns, and extract insights.
6. **Use groups in X-Ray:** Create groups in X-Ray to filter traces based on criteria such as high latency, allowing for more targeted analysis.
7. **Incorporate Amazon DevOps Guru:** Engage Amazon DevOps Guru to benefit from machine learning models pinpointing operational anomalies in traces.
8. **Use CloudWatch Synthetics:** Use CloudWatch Synthetics to create canaries for continually monitoring your endpoints and workflows. These canaries can integrate with X-Ray to provide trace data for in-depth analysis of the applications being tested.
9. **Use Real User Monitoring (RUM):** With AWS X-Ray and CloudWatch RUM, you can analyze and debug the request path starting from end users of your application through downstream AWS managed services. This helps you identify latency trends and errors that impact your users.
10. **Correlate with logs:** Correlate trace data with related logs within the X-Ray trace view for a granular perspective on application behavior. This allows you to view log events directly associated with traced transactions.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS08-BP01 Analyze workload metrics
- OPS08-BP02 Analyze workload logs

Related documents:

- Using ServiceLens to Monitor Application Health
- Exploring Trace Data with X-Ray Analytics
- Detecting Anomalies in Traces with X-Ray Insights
- Continuous Monitoring with CloudWatch Synthetics

Related videos:

- Analyze and Debug Applications Using Amazon CloudWatch Synthetics and AWS X-Ray

- Use AWS X-Ray Insights

Related examples:

- One Observability Workshop
- Implementing X-Ray with AWS Lambda
- CloudWatch Synthetics Canary Templates

OPS08-BP04 Create actionable alerts

Promptly detecting and responding to deviations in your application's behavior is crucial. Especially vital is recognizing when outcomes based on key performance indicators (KPIs) are at risk or when unexpected anomalies arise. Basing alerts on KPIs ensures that the signals you receive are directly tied to business or operational impact. This approach to actionable alerts promotes proactive responses and helps maintain system performance and reliability.

Desired outcome: Receive timely, relevant, and actionable alerts for rapid identification and mitigation of potential issues, especially when KPI outcomes are at risk.

Common anti-patterns:

- Setting up too many non-critical alerts, leading to alert fatigue.
- Not prioritizing alerts based on KPIs, making it hard to understand the business impact of issues.
- Neglecting to address root causes, leading to repetitive alerts for the same issue.

Benefits of establishing this best practice:

- Reduced alert fatigue by focusing on actionable and relevant alerts.
- Improved system uptime and reliability through proactive issue detection and mitigation.
- Enhanced team collaboration and quicker issue resolution by integrating with popular alerting and communication tools.

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

Implementation guidance

To create an effective alerting mechanism, it's vital to use metrics, logs, and trace data that flag when outcomes based on KPIs are at risk or anomalies are detected.

Implementation steps

1. **Determine key performance indicators (KPIs):** Identify your application's KPIs. Alerts should be tied to these KPIs to reflect the business impact accurately.
2. **Implement anomaly detection:**
   - **Use CloudWatch anomaly detection:** Set up CloudWatch anomaly detection to automatically detect unusual patterns, ensuring alerts are only generated for genuine anomalies.
   - **Use X-Ray Insights:**
     a. Set up X-Ray Insights to detect anomalies in trace data.
     b. Configure notifications for X-Ray Insights to be alerted on detected issues.
   - **Integrate with DevOps Guru:**
     a. Leverage Amazon DevOps Guru for its machine learning capabilities in detecting operational anomalies with existing data.
     b. Navigate to the notification settings in DevOps Guru to set up anomaly alerts.
3. **Implement actionable alerts:** Design alerts that provide adequate information for immediate action.
4. **Reduce alarm fatigue:** Minimize non-critical alerts. Overwhelming teams with numerous insignificant alerts can lead to oversight of critical issues and diminish the overall effectiveness of the alerting mechanism.
5. **Set up composite alarms:** Use Amazon CloudWatch composite alarms to consolidate multiple alarms.
6. **Integrate with alerting tools:** Incorporate tools like Ops Genie and PagerDuty.
7. **Engage AWS Chatbot** Integrate AWS Chatbotto relay alerts to Chime, Microsoft Teams, and Slack.
8. **Alert based on logs:** Use log metric filters in CloudWatch to create alarms based on specific log events.
9. **Review and iterate:** Regularly revisit and refine alert configurations.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS04-BP03 Implement user experience telemetry
- OPS04-BP04 Implement dependency telemetry
- OPS04-BP05 Implement distributed tracing
- OPS08-BP01 Analyze workload metrics
- OPS08-BP02 Analyze workload logs
- OPS08-BP03 Analyze workload traces

Related documents:

- Using Amazon CloudWatch Alarms
- Create a composite alarm
- Create a CloudWatch alarm based on anomaly detection
- DevOps Guru Notifications
- X-Ray Insights notifications
- OMonitor, operate, and troubleshoot your AWS resources with interactive ChatOps
- Amazon CloudWatch Integration Guide | PagerDuty
- Integrate OpsGenie with Amazon CloudWatch

Related videos:

- Create Composite Alarms in Amazon CloudWatch
- AWS Chatbot Overview
- AWS on Air ft. Mutative Commands in AWS Chatbot

Related examples:

- Alarms, incident management, and remediation in the cloud with Amazon CloudWatch
- Tutorial: Creating an Amazon EventBridge rule that sends notifications to AWS Chatbot

- One Observability Workshop

OPS08-BP05 Create dashboards

Dashboards are the human-centric view into the telemetry data of your workloads. While they provide a vital visual interface, they should not replace alerting mechanisms, but complement them. When crafted with care, not only can they offer rapid insights into system health and performance, but they can also present stakeholders with real-time information on business outcomes and the impact of issues.

Desired outcome: Clear, actionable insights into system and business health using visual representations.

Common anti-patterns:

- Overcomplicating dashboards with too many metrics.
- Relying on dashboards without alerts for anomaly detection.
- Not updating dashboards as workloads evolve.

Benefits of establishing this best practice:

- Immediate visibility into critical system metrics and KPIs.
- Enhanced stakeholder communication and understanding.
- Rapid insight into the impact of operational issues.

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

Implementation guidance

Business-centric dashboards

Dashboards tailored to business KPIs engage a wider array of stakeholders. While these individuals might not be interested in system metrics, they are keen on understanding the business implications of these numbers. A business-centric dashboard ensures that all technical and operational metrics being monitored and analyzed are in sync with overarching business goals. This alignment provides clarity, ensuring everyone is on the same page regarding what's essential and what's not. Additionally, dashboards that highlight business KPIs tend to be more actionable.

Stakeholders can quickly understand the health of operations, areas that need attention, and the potential impact on business outcomes.

With this in mind, when creating your dashboards, ensure that there's a balance between technical metrics and business KPIs. Both are vital, but they cater to different audiences. Ideally, you should have dashboards that provide a holistic view of the system's health and performance while also emphasizing key business outcomes and their implications.

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 AWS Regions and accounts.

Implementation steps

1. **Create a basic dashboard:** Create a new dashboard in CloudWatch, giving it a descriptive name.
2. **Use Markdown widgets:** Before diving into metrics, use Markdown widgets to add textual context at the top of your dashboard. This should explain what the dashboard covers, the significance of the represented metrics, and can also contain links to other dashboards and troubleshooting tools.
3. **Create dashboard variables:** Incorporate dashboard variables where appropriate to allow for dynamic and flexible dashboard views.
4. **Create metrics widgets:** Add metric widgets to visualize various metrics your application emits, tailoring these widgets to effectively represent system health and business outcomes.
5. **Log Insights queries:** Utilize CloudWatch Logs Insights to derive actionable metrics from your logs and display these insights on your dashboard.
6. **Set up alarms:** Integrate CloudWatch alarms into your dashboard for a quick view of any metrics breaching their thresholds.
7. **Use Contributor Insights:** Incorporate CloudWatch Contributor Insights to analyze high-cardinality fields and get a clearer understanding of your resource's top contributors.
8. **Design custom widgets:** For specific needs not met by standard widgets, consider creating custom widgets. These can pull from various data sources or represent data in unique ways.
9. **Iterate and refine:** As your application evolves, regularly revisit your dashboard to ensure its relevance.

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS08-BP01 Analyze workload metrics
- OPS08-BP02 Analyze workload logs
- OPS08-BP03 Analyze workload traces
- OPS08-BP04 Create actionable alerts

Related documents:

- Building Dashboards for Operational Visibility
- Using Amazon CloudWatch Dashboards

Related videos:

- Create Cross Account & Cross Region CloudWatch Dashboards
- AWS re:Invent 2021 - Gain enterprise visibility with AWS Cloud operation dashboards

Related examples:

- One Observability Workshop
- Application Monitoring with Amazon CloudWatch

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 Measure operations goals and KPIs with metrics
- OPS09-BP02 Communicate status and trends to ensure visibility into operation
- OPS09-BP03 Review operations metrics and prioritize improvement

OPS09-BP01 Measure operations goals and KPIs with metrics

Obtain goals and KPIs that define operations success from your organization and determine that metrics reflect these. Set baselines as a point of reference and reevaluate regularly. Develop mechanisms to collect these metrics from teams for evaluation.

Desired outcome:

- The goals and KPIs for the organization's operations teams have been published and shared.
- Metrics that reflect these KPIs are established. Examples may include:
  - Ticket queue depth or average age of ticket
  - Ticket count grouped by type of issue
  - Time spent working issues with or without a standardized operating procedure (SOP)
  - Amount of time spent recovering from a failed code push
  - Call volume

Common anti-patterns:

- Deployment deadlines are missed because developers are pulled away to perform troubleshooting tasks. Development teams argue for more personnel, but cannot quantify how many they need because the time taken away cannot be measured.
- A Tier 1 desk was set up to handle user calls. Over time, more workloads were added, but no headcount was allocated to the Tier 1 desk. Customer satisfaction suffers as call times increase and issues go longer without resolution, but management sees no indicators of such, preventing any action.
- A problematic workload has been handed off to a separate operations team for upkeep. Unlike other workloads, this new one was not supplied with proper documentation and runbooks. As such, teams spend longer troubleshooting and addressing failures. However, there are no metrics documenting this, which makes accountability difficult.

Benefits of establishing this best practice: Where workload monitoring shows the state of our applications and services, monitoring operations teams provide owners gain insight into changes among the consumers of those workloads, such as shifting business needs. Measure the effectiveness of these teams and evaluate them against business goals by creating metrics that can reflect the state of operations. Metrics can highlight support issues or identify when drifts occur away from a service level target.

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

Implementation guidance

Schedule time with business leaders and stakeholders to determine the overall goals of the service. Determine what the tasks of various operations teams should be and what challenges they could be approached with. Using these, brainstorm key performance indicators (KPIs) that might reflect these operations goals. These might be customer satisfaction, time from feature conception to deployment, average issue resolution time, and others.

Working from KPIs, identify the metrics and sources of data that might reflect these goals best. Customer satisfaction may be an combination of various metrics such as call wait or response times, satisfaction scores, and types of issues raised. Deployment times may be the sum of time needed for testing and deployment, plus any post-deployment fixes that needed to be added. Statistics showing the time spent on different types of issues (or the counts of those issues) can provide a window into where targeted effort is needed.

Resources

Related documents:

- Amazon QuickSight - Using KPIs
- Amazon CloudWatch - Using Metrics
- Building Dashboards
- How to track your cost optimization KPIs with KPI Dashboard

OPS09-BP02 Communicate status and trends to ensure visibility into operation

Knowing the state of your operations and its trending direction is necessary to identify when outcomes may be at risk, whether or not added work can be supported, or the effects that changes have had to your teams. During operations events, having status pages that users and operations teams can refer to for information can reduce pressure on communication channels and disseminate information proactively.

Desired outcome:

- Operations leaders have insight at a glance to see what sort of call volumes their teams are operating under and what efforts may be under way, such as deployments.

- Alerts are disseminated to stakeholders and user communities when impacts to normal operations occur.
- Organization leadership and stakeholders can check a status page in response to an alert or impact, and obtain information surrounding an operational event, such as points of contact, ticket information, and estimated recovery times.
- Reports are made available to leadership and other stakeholders to show operations statistics such as call volumes over a period of time, user satisfaction scores, numbers of outstanding tickets and their ages.

Common anti-patterns:

- A workload goes down, leaving a service unavailable. Call volumes spike as users request to know what's going on. Managers add to the volume requesting to know who's working an issue. Various operations teams duplicate efforts in trying to investigate.
- A desire for a new capability leads to several personnel being reassigned to an engineering effort. No backfill is provided, and issue resolution times spike. This information is not captured, and only after several weeks and dissatisfied user feedback does leadership become aware of the issue.

Benefits of establishing this best practice: During operational events where the business is impacted, much time and energy can be wasted querying information from various teams attempting to understand the situation. By establishing widely-disseminated status pages and dashboards, stakeholders can quickly obtain information such as whether or not an issue was detected, who has lead on the issue, or when a return to normal operations may be expected. This frees team members from spending too much time communicating status to others and more time addressing issues.

In addition, dashboards and reports can provide insights to decision-makers and stakeholders to see how operations teams are able to respond to business needs and how their resources are being allocated. This is crucial for determining if adequate resources are in place to support the business.

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

Implementation guidance

Build dashboards that show the current key metrics for your ops teams, and make them readily accessible both to operations leaders and management.

Build status pages that can be updated quickly to show when an incident or event is unfolding, who has ownership and who is coordinating the response. Share any steps or workarounds that users should consider on this page, and disseminate the location widely. Encourage users to check this location first when confronted with an unknown issue.

Collect and provide reports that show the health of operations over time, and distribute this to leaders and decision makers to illustrate the work of operations along with challenges and needs.

Share between teams these metrics and reports that best reflect goals and KPIs and where they have been influential in driving change. Dedicate time to these activities to elevate the importance of operations inside of and between teams.

Resources

Related documents:

- Measure Progress
- Building dashboards for operation visibility

Related solutions:

- Data Operations

OPS09-BP03 Review operations metrics and prioritize improvement

Setting aside dedicated time and resources for reviewing the state of operations ensures that serving the day-to-day line of business remains a priority. Pull together operations leaders and stakeholders to regularly review metrics, reaffirm or modify goals and objectives, and prioritize improvements.

Desired outcome:

- Operations leaders and staff regularly meet to review metrics over a given reporting period. Challenges are communicated, wins are celebrated, and lessons learned are shared.
- Stakeholders and business leaders are regularly briefed on the state of operations and solicited for input regarding goals, KPIs, and future initiatives. Tradeoffs between service delivery, operations, and maintenance are discussed and placed into context.

Common anti-patterns:

- A new product is launched, but the Tier 1 and Tier 2 operations teams are not adequately trained to support or given additional staff. Metrics that show the decrease in ticket resolution times and increase in incident volumes are not seen by leaders. Action is taken weeks later when subscription numbers start to fall as discontent users move off the platform.
- A manual process for performing maintenance on a workload has been in place for a long time. While a desire to automate has been present, this was a low priority given the low importance of the system. Over time however, the system has grown in importance and now these manual processes consume a majority of operations' time. No resources are scheduled for providing increased tooling to operations, leading to staff burnout as workloads increase. Leadership becomes aware once it's reported that staff are leaving for other competitors.

Benefits of establishing this best practice: In some organizations, it can become a challenge to allocate the same time and attention that is afforded to service delivery and new products or offerings. When this occurs, the line of business can suffer as the level of service expected slowly deteriorates. This is because operations does not change and evolve with the growing business, and can soon be left behind. Without regular review into the insights operations collects, the risk to the business may become visible only when it's too late. By allocating time to review metrics and procedures both among the operations staff and with leadership, the crucial role operations plays remains visible, and risks can be identified long before they reach critical levels. Operations teams get better insight into impending business changes and initiatives, allowing for proactive efforts to be undertaken. Leadership visibility into operations metrics showcases the role that these teams play in customer satisfaction, both internal and external, and let them better weigh choices for priorities, or ensure that operations has the time and resources to change and evolve with new business and workload initiatives.

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

Implementation guidance

Dedicate time to review operations metrics between stakeholders and operations teams and review report data. Place these reports in the contexts of the organizations goals and objectives to determine if they're being met. Identify sources of ambiguity where goals are not clear, or where there may be conflicts between what is asked for and what is given.

Identify where time, people, and tools can aid in operations outcomes. Determine which KPIs this would impact and what targets for success should be. Revisit regularly to ensure operations is resourced sufficiently to support the line of business.

Resources

Related documents:

- Amazon Athena
- Amazon CloudWatch metrics and dimensions reference
- Amazon QuickSight
- AWS Glue
- AWS Glue Data Catalog
- Collect metrics and logs from Amazon EC2 instances and on-premises servers with the Amazon CloudWatch Agent
- Using Amazon CloudWatch metrics

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.

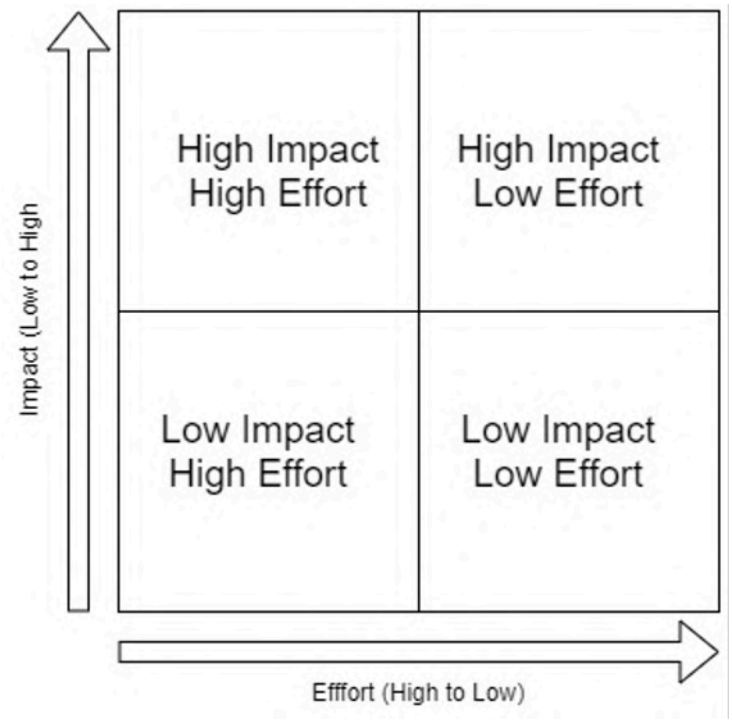
Implementation steps

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 AWS 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 Amazon QuickSight 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:

- Amazon QuickSight
- 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 Amazon QuickSight 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
- Amazon QuickSight
- 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: 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

This best practice was updated with new guidance on December 6, 2023.

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?

“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.

Threat Composer

To aid and guide you in performing threat modeling, consider using the Threat Composer tool, which aims to your reduce time-to-value when threat modeling. The tool helps you do the following:

- Write useful threat statements aligned to threat grammar that work in a natural non-linear workflow
- Generate a human-readable threat model
- Generate a machine-readable threat model to allow you treat threat models as code
- Help you to quickly identify areas of quality and coverage improvement using the Insights Dashboard

For further reference, visit Threat Composer and switch to the system-defined **Example Workspace**.

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

Related tools:

- Threat Composer

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 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

Credential type	Description	Suggested strategy
		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 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 (employees and contractors), 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 systems, because you are creating, assigning, managing, revoking, and auditing access from a single location.

Desired outcome: You have a centralized identity provider where you centrally manage workforce users, authentication policies (such as requiring multi-factor authentication (MFA)), and authorization to systems and applications (such as assigning access based on a user's group membership or attributes). Your workforce users sign in to the central identity provider and federate (single sign-on) to internal and external applications, removing the need for users to remember multiple credentials. Your identity provider is integrated with your human resources (HR) systems so that personnel changes are automatically synchronized to your identity provider. For example, if someone leaves your organization, you can automatically revoke access to federated applications and systems (including AWS). You have enabled detailed audit logging in your identity provider and are monitoring these logs for unusual user behavior.

Common anti-patterns:

- You do not use federation and single-sign on. Your workforce users create separate user accounts and credentials in multiple applications and systems.
- You have not automated the lifecycle of identities for workforce users, such as by integrating your identity provider with your HR systems. When a user leaves your organization or changes roles, you follow a manual process to delete or update their records in multiple applications and systems.

Benefits of establishing this best practice: By using a centralized identity provider, you have a single place to manage workforce user identities and policies, the ability to assign access to applications to users and groups, and the ability to monitor user sign-in activity. By integrating with your human resources (HR) systems, when a user changes roles, these changes are synchronized to the identity provider and automatically updates their assigned applications and permissions. When a user leaves your organization, their identity is automatically disabled in the identity provider, revoking their access to federated applications and systems.

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

Implementation guidance

Guidance for workforce users accessing AWS

Workforce users like employees and contractors in your organization may require access to AWS using the AWS Management Console or AWS Command Line Interface (AWS CLI) to perform their job functions. You can grant AWS access to your workforce users by federating from your centralized identity provider to AWS at two levels: direct federation to each AWS account or federating to multiple accounts in your AWS organization.

- To federate your workforce users directly with each AWS account, you can use a centralized identity provider to federate to AWS Identity and Access Management in that account. The flexibility of IAM allows you to enable a separate SAML 2.0 or an Open ID Connect (OIDC) Identity Provider for each AWS account and use federated user attributes for access control. Your workforce users will use their web browser to sign in to the identity provider by providing their credentials (such as passwords and MFA token codes). The identity provider issues a SAML assertion to their browser that is submitted to the AWS Management Console sign in URL to allow the user to single sign-on to the AWS Management Console by assuming an IAM Role. Your users can also obtain temporary AWS API credentials for use in the AWS CLI or AWS SDKs from AWS STS by assuming the IAM role using a SAML assertion from the identity provider.
- To federate your workforce users with multiple accounts in your AWS organization, you can use AWS IAM Identity Center to centrally manage access for your workforce users to AWS accounts and applications. You enable Identity Center for your organization and configure your identity source. IAM Identity Center provides a default identity source directory which you can use to manage your users and groups. Alternatively, you can choose an external identity source by connecting to your external identity provider using SAML 2.0 and automatically provisioning users and groups using SCIM, or connecting to your Microsoft AD Directory using AWS Directory Service. Once an identity source is configured, you can assign access to users and groups to AWS accounts by defining least-privilege policies in your permission sets. Your workforce users can authenticate through your central identity provider to sign in to the AWS access portal and single-sign on to the AWS accounts and cloud applications assigned to them. Your users can configure the AWS CLI v2 to authenticate with Identity Center and get credentials to run AWS CLI

commands. Identity Center also allows single-sign on access to AWS applications such as Amazon SageMaker Studio and AWS IoT Sitewise Monitor portals.

After you follow the preceding guidance, your workforce users will no longer need to use IAM users and groups for normal operations when managing workloads on AWS. Instead, your users and groups are managed outside of AWS and users are able to access AWS resources as a *federated identity*. Federated identities use the groups defined by your centralized identity provider. You should identify and remove IAM groups, IAM users, and long-lived user credentials (passwords and access keys) that are no longer needed in your AWS accounts. You can find unused credentials using IAM credential reports, delete the corresponding IAM users and delete IAM groups. You can apply a Service Control Policy (SCP) to your organization that helps prevent the creation of new IAM users and groups, enforcing that access to AWS is via federated identities.

Guidance for users of your applications

You can manage the identities of users of your applications, such as a mobile app, using Amazon Cognito as your centralized identity provider. Amazon Cognito enables authentication, authorization, and user management for your web and mobile apps. Amazon Cognito provides an identity store that scales to millions of users, supports social and enterprise identity federation, and offers advanced security features to help protect your users and business. You can integrate your custom web or mobile application with Amazon Cognito to add user authentication and access control to your applications in minutes. Built on open identity standards such as SAML and Open ID Connect (OIDC), Amazon Cognito supports various compliance regulations and integrates with frontend and backend development resources.

Implementation steps

Steps for workforce users accessing AWS

- Federate your workforce users to AWS using a centralized identity provider using one of the following approaches:
  - Use IAM Identity Center to enable single sign-on to multiple AWS accounts in your AWS organization by federating with your identity provider.
  - Use IAM to connect your identity provider directly to each AWS account, enabling federated fine-grained access.
- Identify and remove IAM users and groups that are replaced by federated identities.

Steps for users of your applications

- Use Amazon Cognito as a centralized identity provider towards your applications.
- Integrate your custom applications with Amazon Cognito using OpenID Connect and OAuth. You can develop your custom applications using the Amplify libraries that provide simple interfaces to integrate with a variety of AWS services, such as Amazon Cognito for authentication.

Resources

Related Well-Architected best practices:

- SEC02-BP06 Leverage user groups and attributes
- SEC03-BP02 Grant least privilege access
- SEC03-BP06 Manage access based on lifecycle

Related documents:

- Identity federation in AWS
- Security best practices in IAM
- AWS Identity and Access Management Best practices
- Getting started with IAM Identity Center delegated administration
- How to use customer managed policies in IAM Identity Center for advanced use cases
- AWS CLI v2: IAM Identity Center credential provider

Related videos:

- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive
- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Invent 2018: Mastering Identity at Every Layer of the Cake

Related examples:

- Workshop: Using AWS IAM Identity Center to achieve strong identity management
- Workshop: Serverless identity

Related tools:

- AWS Security Competency Partners: Identity and Access Management
- saml2aws

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: High

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 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
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*.

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

Create a process that allows for emergency access to your workloads in the unlikely event of an issue with your centralized identity provider.

You must design processes for different failure modes that may result in an emergency event. For example, under normal circumstances, your workforce users federate to the cloud using a centralized identity provider (SEC02-BP04) to manage their workloads. However, if your centralized identity provider fails, or the configuration for federation in the cloud is modified, then your workforce users may not be able to federate into the cloud. An emergency access process allows authorized administrators to access your cloud resources through alternate means (such as an alternate form of federation or direct user access) to fix issues with your federation configuration or your workloads. The emergency access process is used until the normal federation mechanism is restored.

Desired outcome:

- You have defined and documented the failure modes that count as an emergency: consider your normal circumstances and the systems your users depend on to manage their workloads. Consider how each of these dependencies can fail and cause an emergency situation. You may find the questions and best practices in the Reliability pillar useful to identify failure modes and architect more resilient systems to minimize the likelihood of failures.
- You have documented the steps that must be followed to confirm a failure as an emergency. For example, you can require your identity administrators to check the status of your primary and standby identity providers and, if both are unavailable, declare an emergency event for identity provider failure.
- You have defined an emergency access process specific to each type of emergency or failure mode. Being specific can reduce the temptation on the part of your users to overuse a general process for all types of emergencies. Your emergency access processes describe the circumstances under which each process should be used, and conversely situations where the process should not be used and points to alternate processes that may apply.
- Your processes are well-documented with detailed instructions and playbooks that can be followed quickly and efficiently. Remember that an emergency event can be a stressful time for your users and they may be under extreme time pressure, so design your process to be as simple as possible.

Common anti-patterns:

- You do not have well-documented and well-tested emergency access processes. Your users are unprepared for an emergency and follow improvised processes when an emergency event arises.
- Your emergency access processes depend on the same systems (such as a centralized identity provider) as your normal access mechanisms. This means that the failure of such a system may impact both your normal and emergency access mechanisms and impair your ability to recover from the failure.
- Your emergency access processes are used in non-emergency situations. For example, your users frequently misuse emergency access processes as they find it easier to make changes directly than submit changes through a pipeline.
- Your emergency access processes do not generate sufficient logs to audit the processes, or the logs are not monitored to alert for potential misuse of the processes.

Benefits of establishing this best practice:

- By having well-documented and well-tested emergency access processes, you can reduce the time taken by your users to respond to and resolve an emergency event. This can result in less downtime and higher availability of the services you provide to your customers.
- You can track each emergency access request and detect and alert on unauthorized attempts to misuse the process for non-emergency events.

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

Implementation guidance

This section provides guidance for creating emergency access processes for several failure modes related to workloads deployed on AWS, starting with common guidance that applies to all failure modes and followed by specific guidance based on the type of failure mode.

Common guidance for all failure modes

Consider the following as you design an emergency access process for a failure mode:

- Document the pre-conditions and assumptions of the process: when the process should be used and when it should not be used. It helps to detail the failure mode and document assumptions, such as the state of other related systems. For example, the process for the Failure Mode 2 assumes that the identity provider is available, but the configuration on AWS is modified or has expired.

- Pre-create resources needed by the emergency access process (SEC10-BP05). For example, pre-create the emergency access AWS account with IAM users and roles, and the cross-account IAM roles in all the workload accounts. This verifies that these resources are ready and available when an emergency event happens. By pre-creating resources, you do not have a dependency on AWS control plane APIs (used to create and modify AWS resources) that may be unavailable in an emergency. Further, by pre-creating IAM resources, you do not need to account for potential delays due to eventual consistency.
- Include emergency access processes as part of your incident management plans (SEC10-BP02). Document how emergency events are tracked and communicated to others in your organization such as peer teams, your leadership, and, when applicable, externally to your customers and business partners.
- Define the emergency access request process in your existing service request workflow system if you have one. Typically, such workflow systems allow you to create intake forms to collect information about the request, track the request through each stage of the workflow, and add both automated and manual approval steps. Relate each request with a corresponding emergency event tracked in your incident management system. Having a uniform system for emergency accesses allows you to track those requests in a single system, analyze usage trends, and improve your processes.
- Verify that your emergency access processes can only be initiated by authorized users and require approvals from the user's peers or management as appropriate. The approval process should operate effectively both inside and outside business hours. Define how requests for approval allow secondary approvers if the primary approvers are unavailable and are escalated up your management chain until approved.
- Verify that the process generates detailed audit logs and events for both successful and failed attempts to gain emergency access. Monitor both the request process and the emergency access mechanism to detect misuse or unauthorized accesses. Correlate activity with ongoing emergency events from your incident management system and alert when actions happen outside of expected time periods. For example, you should monitor and alert on activity in the emergency access AWS account, as it should never be used in normal operations.
- Test emergency access processes periodically to verify that the steps are clear and grant the correct level of access quickly and efficiently. Your emergency access processes should be tested as part of incident response simulations (SEC10-BP07) and disaster recovery tests (REL13-BP03).

Failure Mode 1: Identity provider used to federate to AWS is unavailable

As described in SEC02-BP04 Rely on a centralized identity provider, we recommend relying on a centralized identity provider to federate your workforce users to grant access to AWS accounts. You can federate to multiple AWS accounts in your AWS organization using IAM Identity Center, or you can federate to individual AWS accounts using IAM. In both cases, workforce users authenticate with your centralized identity provider before being redirected to an AWS sign-in endpoint to single sign-on.

In the unlikely event that your centralized identity provider is unavailable, your workforce users can't federate to AWS accounts or manage their workloads. In this emergency event, you can provide an emergency access process for a small set of administrators to access AWS accounts to perform critical tasks that cannot wait until your centralized identity providers are back online. For example, your identity provider is unavailable for 4 hours and during that period you need to modify the upper limits of an Amazon EC2 Auto Scaling group in a Production account to handle an unexpected spike in customer traffic. Your emergency administrators should follow the emergency access process to gain access to the specific production AWS account and make the necessary changes.

The emergency access process relies on a pre-created emergency access AWS account that is used solely for emergency access and has AWS resources (such as IAM roles and IAM users) to support the emergency access process. During normal operations, no one should access the emergency access account and you must monitor and alert on the misuse of this account (for more detail, see the preceding Common guidance section).

The emergency access account has emergency access IAM roles with permissions to assume cross-account roles in the AWS accounts that require emergency access. These IAM roles are pre-created and configured with trust policies that trust the emergency account's IAM roles.

The emergency access process can use one of the following approaches:

- You can pre-create a set of IAM users for your emergency administrators in the emergency access account with associated strong passwords and MFA tokens. These IAM users have permissions to assume the IAM roles that then allow cross-account access to the AWS account where emergency access is required. We recommend creating as few such users as possible and assigning each user to a single emergency administrator. During an emergency, an emergency administrator user signs into the emergency access account using their password and MFA token code, switches to the emergency access IAM role in the emergency account, and finally switches to the emergency access IAM role in the workload account to perform the emergency change action. The advantage of this approach is that each IAM user is assigned to one emergency administrator and you can know which user signed-in by reviewing CloudTrail events. The disadvantage is that you have to maintain multiple IAM users with their associated long-lived passwords and MFA tokens.

- You can use the emergency access AWS account root user to sign into the emergency access account, assume the IAM role for emergency access, and assume the cross-account role in the workload account. We recommend setting a strong password and multiple MFA tokens for the root user. We also recommend storing the password and the MFA tokens in a secure enterprise credential vault that enforces strong authentication and authorization. You should secure the password and MFA token reset factors: set the email address for the account to an email distribution list that is monitored by your cloud security administrators, and the phone number of the account to a shared phone number that is also monitored by security administrators. The advantage of this approach is that there is one set of root user credentials to manage. The disadvantage is that since this is a shared user, multiple administrators have ability to sign in as the root user. You must audit your enterprise vault log events to identify which administrator checked out the root user password.

Failure Mode 2: Identity provider configuration on AWS is modified or has expired

To allow your workforce users to federate to AWS accounts, you can configure the IAM Identity Center with an external identity provider or create an IAM Identity Provider (SEC02-BP04). Typically, you configure these by importing a SAML meta-data XML document provided by your identity provider. The meta-data XML document includes a X.509 certificate corresponding to a private key that the identity provider uses to sign its SAML assertions.

These configurations on the AWS-side may be modified or deleted by mistake by an administrator. In another scenario, the X.509 certificate imported into AWS may expire and a new meta-data XML with a new certificate has not yet been imported into AWS. Both scenarios can break federation to AWS for your workforce users, resulting in an emergency.

In such an emergency event, you can provide your identity administrators access to AWS to fix the federation issues. For example, your identity administrator uses the emergency access process to sign into the emergency access AWS account, switches to a role in the Identity Center administrator account, and updates the external identity provider configuration by importing the latest SAML meta-data XML document from your identity provider to re-enable federation. Once federation is fixed, your workforce users continue to use the normal operating process to federate into their workload accounts.

You can follow the approaches detailed in the previous Failure Mode 1 to create an emergency access process. You can grant least-privilege permissions to your identity administrators to access only the Identity Center administrator account and perform actions on Identity Center in that account.

Failure Mode 3: Identity Center disruption

In the unlikely event of an IAM Identity Center or AWS Region disruption, we recommend that you set up a configuration that you can use to provide temporary access to the AWS Management Console.

The emergency access process uses direct federation from your identity provider to IAM in an emergency account. For detail on the process and design considerations, see Set up emergency access to the AWS Management Console.

Implementation steps

Common steps for all failure modes

- Create an AWS account dedicated to emergency access processes. Pre-create the IAM resources needed in the account such as IAM roles or IAM users, and optionally IAM Identity Providers. Additionally, pre-create cross-account IAM roles in the workload AWS accounts with trust relationships with corresponding IAM roles in the emergency access account. You can use AWS CloudFormation StackSets with AWS Organizations to create such resources in the member accounts in your organization.

- Create AWS Organizations service control policies (SCPs) to deny the deletion and modification of the cross-account IAM roles in the member AWS accounts.
- Enable CloudTrail for the emergency access AWS account and send the trail events to a central S3 bucket in your log collection AWS account. If you are using AWS Control Tower to set up and govern your AWS multi-account environment, then every account you create using AWS Control Tower or enroll in AWS Control Tower has CloudTrail enabled by default and sent to an S3 bucket in a dedicated log archive AWS account.
- Monitor the emergency access account for activity by creating EventBridge rules that match on console login and API activity by the emergency IAM roles. Send notifications to your security operations center when activity happens outside of an ongoing emergency event tracked in your incident management system.

Additional steps for Failure Mode 1: Identity provider used to federate to AWS is unavailable and Failure Mode 2: Identity provider configuration on AWS is modified or has expired

- Pre-create resources depending on the mechanism you choose for emergency access:
  - **Using IAM users:** pre-create the IAM users with strong passwords and associated MFA devices.
  - **Using the emergency account root user:** configure the root user with a strong password and store the password in your enterprise credential vault. Associate multiple physical MFA devices with the root user and store the devices in locations that can be accessed quickly by members of your emergency administrator team.

Additional steps for Failure Mode 3: Identity center disruption

- As detailed in Set up emergency access to the AWS Management Console, in the emergency access AWS account, create an IAM Identity Provider to enable direct SAML federation from your identity provider.
- Create emergency operations groups in your IdP with no members.
- Create IAM roles corresponding to the emergency operations groups in the emergency access account.

Resources

Related Well-Architected best practices:

- SEC02-BP04 Rely on a centralized identity provider
- SEC03-BP02 Grant least privilege access
- SEC10-BP02 Develop incident management plans
- SEC10-BP07 Run game days

Related documents:

- Set up emergency access to the AWS Management Console
- Enabling SAML 2.0 federated users to access the AWS Management Console
- Break glass access

Related videos:

- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive

Related examples:

- AWS Break Glass Role
- AWS customer playbook framework
- AWS incident response playbook samples

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 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, 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 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 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 to send and aggregate findings for a resource from IAM Access Analyzer to Security Hub to help analyze the security posture of your environment. To integrate, turn on both IAM Access Analyzer and Security Hub in each Region in each account. You can also use AWS Config Rules to audit configuration and alert the appropriate party using AWS Chatbot with AWS Security Hub. 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 AWS 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 AWS 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 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 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 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 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 have an Administrator-Member model that can aggregate findings and insights across multiple accounts, and Security Hub 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
- 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 Findings
- Threat management in the cloud: Amazon GuardDuty and AWS Security Hub

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
- 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 Findings

- Threat management in the cloud: Amazon GuardDuty and AWS Security Hub

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 Findings
- Threat management in the cloud: Amazon GuardDuty and AWS Security Hub

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 (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, 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:** Security Hub 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

Secure key management includes the storage, rotation, access control, and monitoring of key material required to secure data at rest for your workload.

Desired outcome: A scalable, repeatable, and automated key management mechanism. The mechanism should provide the ability to enforce least privilege access to key material, provide the correct balance between key availability, confidentiality, and integrity. Access to keys should be monitored, and key material rotated through an automated process. Key material should never be accessible to human identities.

Common anti-patterns:

- Human access to unencrypted key material.
- Creating custom cryptographic algorithms.
- Overly broad permissions to access key material.

Benefits of establishing this best practice: By establishing a secure key management mechanism for your workload, you can help provide protection for your content against unauthorized access. Additionally, you may be subject to regulatory requirements to encrypt your data. An effective key management solution can provide technical mechanisms aligned to those regulations to protect key material.

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

Implementation guidance

Many regulatory requirements and best practices include encryption of data at rest as a fundamental security control. In order to comply with this control, your workload needs a mechanism to securely store and manage the key material used to encrypt your data at rest.

AWS offers AWS Key Management Service (AWS KMS) to provide durable, secure, and redundant storage for AWS KMS keys. Many AWS services integrate with AWS KMS to support encryption of your data. AWS KMS uses FIPS 140-2 Level 3 validated hardware security modules to protect your keys. There is no mechanism to export AWS KMS keys in plain text.

When deploying workloads using a multi-account strategy, it is considered best practice to keep AWS KMS keys in the same account as the workload that uses them. In this distributed model, responsibility for managing the AWS KMS keys resides with the application team. In other use cases, organizations may choose to store AWS KMS keys into a centralized account. This centralized structure requires additional policies to enable the cross-account access required for the workload account to access keys stored in the centralized account, but may be more applicable in use cases where a single key is shared across multiple AWS accounts.

Regardless of where the key material is stored, access to the key should be tightly controlled through the use of key policies and IAM policies. Key policies are the primary way to control access to a AWS KMS key. In addition, AWS KMS key grants can provide access to AWS services to encrypt and decrypt data on your behalf. Take time to review the best practices for access control to your AWS KMS keys.

It is best practice to monitor the use of encryption keys to detect unusual access patterns. Operations performed using AWS managed keys and customer managed keys stored in AWS KMS can be logged in AWS CloudTrail and should be reviewed periodically. Special attention should be placed on monitoring key destruction events. To mitigate accidental or malicious destruction of key material, key destruction events do not delete the key material immediately. Attempts to delete keys in AWS KMS are subject to a waiting period, which defaults to 30 days, providing administrators time to review these actions and roll back the request if necessary.

Most AWS services use AWS KMS in a way that is transparent to you - your only requirement is to decide whether to use an AWS managed or customer managed key. If your workload requires the direct use of AWS KMS to encrypt or decrypt data, the best practice is to use envelope encryption to protect your data. The AWS Encryption SDK can provide your applications client-side encryption primitives to implement envelope encryption and integrate with AWS KMS.

Implementation steps

1. Determine the appropriate key management options (AWS managed or customer managed) for the key.
   - For ease of use, AWS offers AWS owned and AWS managed keys for most services, which provide encryption-at-rest capability without the need to manage key material or key policies.
   - When using customer managed keys, consider the default key store to provide the best balance between agility, security, data sovereignty, and availability. Other use cases may require the use of custom key stores with AWS CloudHSM or the external key store.
2. Review the list of services that you are using for your workload to understand how AWS KMS integrates with the service. For example, EC2 instances can use encrypted EBS volumes, verifying that Amazon EBS snapshots created from those volumes are also encrypted using a customer managed key and mitigating accidental disclosure of unencrypted snapshot data.
   - How AWS services use AWS KMS
   - For detailed information about the encryption options that an AWS service offers, see the Encryption at Rest topic in the user guide or the developer guide for the service.
3. Implement AWS KMS: AWS KMS makes it simple for you to create and manage keys and control the use of encryption across a wide range of AWS services and in your applications.
   - Getting started: AWS Key Management Service (AWS KMS)
   - Review the best practices for access control to your AWS KMS keys.
4. 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
5. Enable IAM Access Analyzer to automatically review and notify if there are overly broad AWS KMS key policies.
6. Enable Security Hub to receive notifications if there are misconfigured key policies, keys scheduled for deletion, or keys without automated rotation enabled.
7. Determine the logging level appropriate for your AWS KMS keys. Since calls to AWS KMS, including read-only events, are logged, the CloudTrail logs associated with AWS KMS can become voluminous.
   - Some organizations prefer to segregate the AWS KMS logging activity into a separate trail. For more detail, see the Logging AWS KMS API calls with CloudTrail section of the AWS KMS developers guide.

Resources

Related documents:

- AWS Key Management Service
- AWS cryptographic services and tools
- Protecting Amazon S3 Data Using Encryption
- Envelope encryption
- Digital sovereignty pledge
- Demystifying AWS KMS key operations, bring your own key, custom key store, and ciphertext portability
- AWS Key Management Service cryptographic details

Related videos:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS
- AWS data protection: Using locks, keys, signatures, and certificates

Related examples:

- Implement advanced access control mechanisms using AWS KMS

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 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 S3 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 Amazon QuickSight, to display data to users instead of directly querying.
  - Amazon QuickSight
- 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

Transport Layer Security (TLS) certificates are used to secure network communications and establish the identity of websites, resources, and workloads over the internet, as well as private networks.

Desired outcome: A secure certificate management system that can provision, deploy, store, and renew certificates in a public key infrastructure (PKI). A secure key and certificate management mechanism prevents certificate private key material from disclosure and automatically renews the certificate on a periodic basis. It also integrates with other services to provide secure network communications and identity for machine resources inside of your workload. Key material should never be accessible to human identities.

Common anti-patterns:

- Performing manual steps during the certificate deployment or renewal processes.
- Paying insufficient attention to certificate authority (CA) hierarchy when designing a private CA.
- Using self-signed certificates for public resources.

Benefits of establishing this best practice:

- Simplify certificate management through automated deployment and renewal
- Encourage encryption of data in transit using TLS certificates
- Increased security and auditability of certificate actions taken by the certificate authority
- Organization of management duties at different layers of the CA hierarchy

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

Implementation guidance

Modern workloads make extensive use of encrypted network communications using PKI protocols such as TLS. PKI certificate management can be complex, but automated certificate provisioning, deployment, and renewal can reduce the friction associated with certificate management.

AWS provides two services to manage general-purpose PKI certificates: AWS Certificate Manager and AWS Private Certificate Authority (AWS Private CA). ACM is the primary service that customers use to provision, manage, and deploy certificates for use in both public-facing as well as private

AWS workloads. ACM issues certificates using AWS Private CA and integrates with many other AWS managed services to provide secure TLS certificates for workloads.

AWS Private CA allows you to establish your own root or subordinate certificate authority and issue TLS certificates through an API. You can use these kinds of certificates in scenarios where you control and manage the trust chain on the client side of the TLS connection. In addition to TLS use cases, AWS Private CA can be used to issue certificates to Kubernetes pods, Matter device product attestations, code signing, and other use cases with a custom template. You can also use IAM Roles Anywhere to provide temporary IAM credentials to on-premises workloads that have been issued X.509 certificates signed by your Private CA.

In addition to ACM and AWS Private CA, AWS IoT Core provides specialized support for provisioning, managing and deploying PKI certificates to IoT devices. AWS IoT Core provides specialized mechanisms for onboarding IoT devices into your public key infrastructure at scale.

Considerations for establishing a private CA hierarchy

When you need to establish a private CA, it's important to take special care to properly design the CA hierarchy upfront. It's a best practice to deploy each level of your CA hierarchy into separate AWS accounts when creating a private CA hierarchy. This intentional step reduces the surface area for each level in the CA hierarchy, making it simpler to discover anomalies in CloudTrail log data and reducing the scope of access or impact if there is unauthorized access to one of the accounts. The root CA should reside in its own separate account and should only be used to issue one or more intermediate CA certificates.

Then, create one or more intermediate CAs in accounts separate from the root CA's account to issue certificates for end users, devices, or other workloads. Finally, issue certificates from your root CA to the intermediate CAs, which will in turn issue certificates to your end users or devices. For more information on planning your CA deployment and designing your CA hierarchy, including planning for resiliency, cross-region replication, sharing CAs across your organization, and more, see Planning your AWS Private CA deployment.

Implementation steps

1. Determine the relevant AWS services required for your use case:
   - Many use cases can leverage the existing AWS public key infrastructure using AWS Certificate Manager. ACM can be used to deploy TLS certificates for web servers, load balancers, or other uses for publicly trusted certificates.

- Consider AWS Private CA when you need to establish your own private certificate authority hierarchy or need access to exportable certificates. ACM can then be used to issue many types of end-entity certificates using the AWS Private CA.
- For use cases where certificates must be provisioned at scale for embedded Internet of things (IoT) devices, consider AWS IoT Core.

2. Implement automated certificate renewal whenever possible:
   - Use ACM managed renewal for certificates issued by ACM along with integrated AWS managed services.
3. Establish logging and audit trails:
   - Enable CloudTrail logs to track access to the accounts holding certificate authorities. Consider configuring log file integrity validation in CloudTrail to verify the authenticity of the log data.
   - Periodically generate and review audit reports that list the certificates that your private CA has issued or revoked. These reports can be exported to an S3 bucket.
   - When deploying a private CA, you will also need to establish an S3 bucket to store the Certificate Revocation List (CRL). For guidance on configuring this S3 bucket based on your workload's requirements, see Planning a certificate revocation list (CRL).

Resources

Related best practices:

- SEC02-BP02 Use temporary credentials
- SEC08-BP01 Implement secure key management
- SEC09-BP04 Authenticate network communications

Related documents:

- How to host and manage an entire private certificate infrastructure in AWS
- How to secure an enterprise scale ACM Private CA hierarchy for automotive and manufacturing
- Private CA best practices
- How to use AWS RAM to share your ACM Private CA cross-account

Related videos:

- Activating AWS Certificate Manager Private CA (workshop)

Related examples:

- Private CA workshop
- IOT Device Management Workshop (including device provisioning)

Related tools:

- Plugin to Kubernetes cert-manager to use AWS Private CA

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

This best practice was updated with new guidance on December 6, 2023.

Verify the identity of communications by using protocols that support authentication, such as Transport Layer Security (TLS) or IPsec.

Design your workload to use secure, authenticated network protocols whenever communicating between services, applications, or to users. Using network protocols that support authentication and authorization provides stronger control over network flows and reduces the impact of unauthorized access.

Desired outcome: A workload with well-defined data plane and control plane traffic flows between services. The traffic flows use authenticated and encrypted network protocols where technically feasible.

Common anti-patterns:

- Unencrypted or unauthenticated traffic flows within your workload.
- Reusing authentication credentials across multiple users or entities.
- Relying solely on network controls as an access control mechanism.
- Creating a custom authentication mechanism rather than relying on industry-standard authentication mechanisms.
- Overly permissive traffic flows between service components or other resources in the VPC.

Benefits of establishing this best practice:

- Limits the scope of impact for unauthorized access to one part of the workload.
- Provides a higher level of assurance that actions are only performed by authenticated entities.

- Improves decoupling of services by clearly defining and enforcing intended data transfer interfaces.
- Enhances monitoring, logging, and incident response through request attribution and well-defined communication interfaces.
- Provides defense-in-depth for your workloads by combining network controls with authentication and authorization controls.

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

Implementation guidance

Your workload's network traffic patterns can be characterized into two categories:

- *East-west traffic* represents traffic flows between services that make up a workload.
- *North-south traffic* represents traffic flows between your workload and consumers.

While it is common practice to encrypt north-south traffic, securing east-west traffic using authenticated protocols is less common. Modern security practices recommend that network design alone does not grant a trusted relationship between two entities. When two services may reside within a common network boundary, it is still best practice to encrypt, authenticate, and authorize communications between those services.

As an example, AWS service APIs use the AWS Signature Version 4 (SigV4) signature protocol to authenticate the caller, no matter what network the request originates from. This authentication ensures that AWS APIs can verify the identity that requested the action, and that identity can then be combined with policies to make an authorization decision to determine whether the action should be allowed or not.

Services such as Amazon VPC Lattice and Amazon API Gateway allow you use the same SigV4 signature protocol to add authentication and authorization to east-west traffic in your own workloads. If resources outside of your AWS environment need to communicate with services that require SigV4-based authentication and authorization, you can use AWS Identity and Access Management (IAM) Roles Anywhere on the non-AWS resource to acquire temporary AWS credentials. These credentials can be used to sign requests to services using SigV4 to authorize access.

Another common mechanism for authenticating east-west traffic is TLS mutual authentication (mTLS). Many Internet of Things (IoT), business-to-business applications, and microservices use mTLS to validate the identity of both sides of a TLS communication through the use of both client and server-side X.509 certificates. These certificates can be issued by AWS Private Certificate Authority (AWS Private CA). You can use services such as Amazon API Gateway and AWS App Mesh to provide mTLS authentication for inter- or intra-workload communication. While mTLS provides authentication information for both sides of a TLS communication, it does not provide a mechanism for authorization.

Finally, OAuth 2.0 and OpenID Connect (OIDC) are two protocols typically used for controlling access to services by users, but are now becoming popular for service-to-service traffic as well. API Gateway provides a JSON Web Token (JWT) authorizer, allowing workloads to restrict access to API routes using JWTs issued from OIDC or OAuth 2.0 identity providers. OAuth2 scopes can be used as a source for basic authorization decisions, but the authorization checks still need to be implemented in the application layer, and OAuth2 scopes alone cannot support more complex authorization needs.

Implementation steps

- **Define and document your workload network flows:** The first step in implementing a defense-in-depth strategy is defining your workload's traffic flows.
  - Create a data flow diagram that clearly defines how data is transmitted between different services that comprise your workload. This diagram is the first step to enforcing those flows through authenticated network channels.
  - Instrument your workload in development and testing phases to validate that the data flow diagram accurately reflects the workload's behavior at runtime.
  - A data flow diagram can also be useful when performing a threat modeling exercise, as described in SEC01-BP07 Identify threats and prioritize mitigations using a threat model.
- **Establish network controls:** Consider AWS capabilities to establish network controls aligned to your data flows. While network boundaries should not be the only security control, they provide a layer in the defense-in-depth strategy to protect your workload.
  - Use security groups to establish define and restrict data flows between resources.
  - Consider using AWS PrivateLink to communicate with both AWS and third-party services that support AWS PrivateLink. Data sent through a AWS PrivateLink interface endpoint stays within the AWS network backbone and does not traverse the public Internet.
- **Implement authentication and authorization across services in your workload:** Choose the set of AWS services most appropriate to provide authenticated, encrypted traffic flows in your workload.

  - Consider Amazon VPC Lattice to secure service-to-service communication. VPC Lattice can use SigV4 authentication combined with auth policies to control service-to-service access.
  - For service-to-service communication using mTLS, consider API Gateway or App Mesh. AWS Private CA can be used to establish a private CA hierarchy capable of issuing certificates for use with mTLS.
  - When integrating with services using OAuth 2.0 or OIDC, consider API Gateway using the JWT authorizer.
  - For communication between your workload and IoT devices, consider AWS IoT Core, which provides several options for network traffic encryption and authentication.
- **Monitor for unauthorized access:** Continually monitor for unintended communication channels, unauthorized principals attempting to access protected resources, and other improper access patterns.
  - If using VPC Lattice to manage access to your services, consider enabling and monitoring VPC Lattice access logs. These access logs include information on the requesting entity, network information including source and destination VPC, and request metadata.
  - Consider enabling VPC flow logs to capture metadata on network flows and periodically review for anomalies.
  - Refer to the AWS Security Incident Response Guide and the Incident Response section of the AWS Well-Architected Framework security pillar for more guidance on planning, simulating, and responding to security incidents.

Resources

Related best practices:

- SEC03-BP07 Analyze public and cross-account access
- SEC02-BP02 Use temporary credentials
- SEC01-BP07 Identify threats and prioritize mitigations using a threat model

Related documents:

- Evaluating access control methods to secure Amazon API Gateway APIs
- Configuring mutual TLS authentication for a REST API
- How to secure API Gateway HTTP endpoints with JWT authorizer

- Authorizing direct calls to AWS services using AWS IoT Core credential provider
- AWS Security Incident Response Guide

Related videos:

- AWS re:invent 2022: Introducing VPC Lattice
- AWS re:invent 2020: Serverless API authentication for HTTP APIs on AWS

Related examples:

- Amazon VPC Lattice Workshop
- Zero-Trust Episode 1 – The Phantom Service Perimeter workshop

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?

Even with mature preventive and detective controls, your organization should implement mechanisms to respond to and mitigate the potential impact of security incidents. Your preparation strongly affects the ability of your teams to operate effectively during an incident, to isolate, contain and perform forensics on issues, 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, helps ensure that you can recover while minimizing business disruption.

Best practices

- SEC10-BP01 Identify key personnel and external resources
- SEC10-BP02 Develop incident management plans
- SEC10-BP03 Prepare forensic capabilities
- SEC10-BP04 Develop and test security incident response playbooks
- SEC10-BP05 Pre-provision access

- SEC10-BP06 Pre-deploy tools
- SEC10-BP07 Run simulations
- SEC10-BP08 Establish a framework for learning from incidents

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

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 can 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 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**
  - AWS 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 AWS Support is controlled by AWS Identity and Access Management. For more information about getting access to AWS Support features, see Getting started with AWS 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 AWS 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 AWS 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 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 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

Ahead of a security incident, consider developing forensics capabilities to support security event investigations.

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

Concepts from traditional on-premises forensics apply to AWS. For key information to start building forensics capabilities in the AWS Cloud, see Forensic investigation environment strategies in the AWS Cloud.

Once you have your environment and AWS account structure set up for forensics, define the technologies required to effectively perform forensically sound methodologies across the four phases:

- **Collection:** Collect relevant AWS logs, such as AWS CloudTrail, AWS Config, VPC Flow Logs, and host-level logs. Collect snapshots, backups, and memory dumps of impacted AWS resources where available.
- **Examination:** Examine the data collected by extracting and assessing the relevant information.
- **Analysis:** Analyze the data collected in order to understand the incident and draw conclusions from it.
- **Reporting:** Present the information resulting from the analysis phase.

Implementation steps

Prepare your forensics environment

AWS Organizations helps you centrally manage and govern an AWS environment as you grow and scale AWS resources. An AWS organization consolidates your AWS accounts so that you can administer them as a single unit. You can use organizational units (OUs) to group accounts together to administer as a single unit.

For incident response, it’s helpful to have an AWS account structure that supports the functions of incident response, which includes a *security OU* and a *forensics OU*. Within the security OU, you should have accounts for:

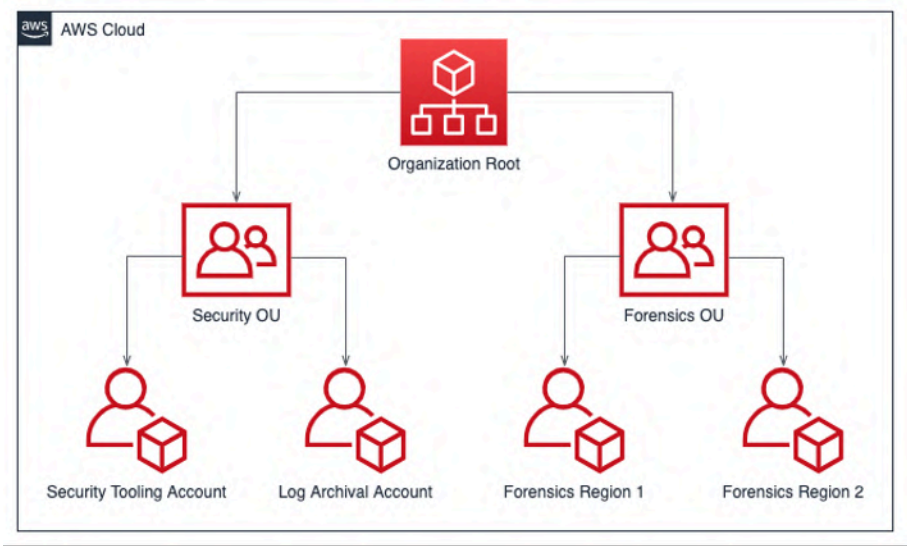
- **Log archival:** Aggregate logs in a log archival AWS account with limited permissions.
- **Security tools:** Centralize security services in a security tool AWS account. This account operates as the delegated administrator for security services.

Within the forensics OU, you have the option to implement a single forensics account or accounts for each Region that you operate in, depending on which works best for your business and operational model. If you create a forensics account per Region, you can block the creation of AWS resources outside of that Region and reduce the risk of resources being copied to an unintended region. For example, if you only operate in US East (N. Virginia) Region (`us-east-1`) and US West

(Oregon) (`us-west-2`), then you would have two accounts in the forensics OU: one for `us-east-1` and one for `us-west-2`.

You can create a forensics AWS account for multiple Regions. You should exercise caution in copying AWS resources to that account to verify you're aligning with your data sovereignty requirements. Because it takes time to provision new accounts, it is imperative to create and instrument the forensics accounts well ahead of an incident so that responders can be prepared to effectively use them for response.

The following diagram displays a sample account structure including a forensics OU with per-Region forensics accounts:



Per-Region account structure for incident response

Capture backups and snapshots

Setting up backups of key systems and databases are critical for recovering from a security incident and for forensics purposes. With backups in place, you can restore your systems to their previous safe state. On AWS, you can take snapshots of various resources. Snapshots provide you with point-in-time backups of those resources. There are many AWS services that can support you in backup and recovery. For detail on these services and approaches for backup and recovery, see Backup and Recovery Prescriptive Guidance and Use backups to recover from security incidents.

Especially when it comes to situations such as ransomware, it's critical for your backups to be well protected. For guidance on securing your backups, see Top 10 security best practices for securing backups in AWS. In addition to securing your backups, you should regularly test your backup and restore processes to verify that the technology and processes you have in place work as expected.

Automate forensics

During a security event, your incident response team must be able to collect and analyze evidence quickly while maintaining accuracy for the time period surrounding the event (such as capturing logs related to a specific event or resource or collecting memory dump of an Amazon EC2 instance). It's both challenging and time consuming for the incident response team to manually collect the relevant evidence, especially across a large number of instances and accounts. Additionally, manual collection can be prone to human error. For these reasons, you should develop and implement automation for forensics as much as possible.

AWS offers a number of automation resources for forensics, which are listed in the following Resources section. These resources are examples of forensics patterns that we have developed and customers have implemented. While they might be a useful reference architecture to start with, consider modifying them or creating new forensics automation patterns based on your environment, requirements, tools, and forensics processes.

Resources

Related documents:

- AWS Security Incident Response Guide - Develop Forensics Capabilities
- AWS Security Incident Response Guide - Forensics Resources
- Forensic investigation environment strategies in the AWS Cloud
- How to automate forensic disk collection in AWS
- AWS Prescriptive Guidance - Automate incident response and forensics

Related videos:

- Automating Incident Response and Forensics

Related examples:

- Automated Incident Response and Forensics Framework
- Automated Forensics Orchestrator for Amazon EC2

SEC10-BP04 Develop and test security incident response playbooks

A key part of preparing your incident response processes is developing playbooks. Incident response playbooks provide a series of prescriptive guidance and steps to follow when a security event occurs. Having clear structure and steps simplifies the response and reduces the likelihood for human error.

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

Implementation guidance

Playbooks should be created for incident scenarios such as:

- **Expected incidents**: Playbooks should be created for incidents you anticipate. This includes threats like denial of service (DoS), ransomware, and credential compromise.
- **Known security findings or alerts**: Playbooks should be created for your known security findings and alerts, such as GuardDuty findings. You might receive a GuardDuty finding and think, "Now what?" To prevent the mishandling or ignoring of a GuardDuty finding, create a playbook for each potential GuardDuty finding. Some remediation details and guidance can be found in the GuardDuty documentation. It’s worth noting that GuardDuty is not enabled by default and does incur a cost. For more detail on GuardDuty, see Appendix A: Cloud capability definitions - Visibility and alerting.

Playbooks should contain technical steps for a security analyst to complete in order to adequately investigate and respond to a potential security incident.

Implementation steps

Items to include in a playbook include:

- **Playbook overview**: What risk or incident scenario does this playbook address? What is the goal of the playbook?
- **Prerequisites**: What logs, detection mechanisms, and automated tools are required for this incident scenario? What is the expected notification?
- **Communication and escalation information**: Who is involved and what is their contact information? What are each of the stakeholders’ responsibilities?
- **Response steps**: Across phases of incident response, what tactical steps should be taken? What queries should an analyst run? What code should be run to achieve the desired outcome?
  - **Detect**: How will the incident be detected?

  - **Analyze**: How will the scope of impact be determined?
  - **Contain**: How will the incident be isolated to limit scope?
  - **Eradicate**: How will the threat be removed from the environment?
  - **Recover**: How will the affected system or resource be brought back into production?
- **Expected outcomes**: After queries and code are run, what is the expected result of the playbook?

Resources

Related Well-Architected best practices:

- SEC10-BP02 - Develop incident management plans

Related documents:

- Framework for Incident Response Playbooks
- Develop your own Incident Response Playbooks
- Incident Response Playbook Samples
- Building an AWS incident response runbook using Jupyter playbooks and CloudTrail Lake

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

Verify that security personnel have the right tools pre-deployed to reduce the time for investigation through to recovery.

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

Implementation guidance

To automate security response 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.

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.

During a security investigation, 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 certain actions of interest have happened. It is critical to select, enable, store, and set up querying and retrieval mechanisms, and set up alerting. Additionally, an effective way to provide tools to search log data is Amazon Detective.

AWS offers over 200 cloud services and thousands of features. We recommend that you review the services that can support and simplify your incident response strategy.

In addition to logging, you should develop and implement a tagging strategy. Tagging can help provide context around the purpose of an AWS resource. Tagging can also be used for automation.

Implementation steps

Select and set up logs for analysis and alerting

See the following documentation on configuring logging for incident response:

- Logging strategies for security incident response
- SEC04-BP01 Configure service and application logging

Enable security services to support detection and response

AWS provides native detective, preventative, and responsive capabilities, and other services can be used to architect custom security solutions. For a list of the most relevant services for security incident response, see Cloud capability definitions.

Develop and implement a tagging strategy

Obtaining contextual information on the business use case and relevant internal stakeholders surrounding an AWS resource can be difficult. One way to do this is in the form of tags, which assign metadata to your AWS resources and consist of a user-defined key and value. You can create tags to categorize resources by purpose, owner, environment, type of data processed, and other criteria of your choice.

Having a consistent tagging strategy can speed up response times and minimize time spent on organizational context by allowing you to quickly identify and discern contextual information about an AWS resource. Tags can also serve as a mechanism to initiate response automations. For more detail on what to tag, see Tagging your AWS resources. You'll want to first define the tags you want to implement across your organization. After that, you'll implement and enforce this tagging strategy. For more detail on implementation and enforcement, see Implement AWS resource tagging strategy using AWS Tag Policies and Service Control Policies (SCPs).

Resources

Related Well-Architected best practices:

- SEC04-BP01 Configure service and application logging
- SEC04-BP02 Analyze logs, findings, and metrics centrally

Related documents:

- Logging strategies for security incident response
- Incident response cloud capability definitions

Related examples:

- Threat Detection and Response with Amazon GuardDuty and Amazon Detective
- Security Hub Workshop
- Vulnerability Management with Amazon Inspector

SEC10-BP07 Run simulations

As organizations grow and evolve over time, so does the threat landscape, making it important to continually review your incident response capabilities. Running simulations (also known as game days) 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

SEC10-BP08 Establish a framework for learning from incidents

Implementing a *lessons learned* framework and root cause analysis capability can not only help improve incident response capabilities, but also help prevent the incident from recurring. By learning from each incident, you can help avoid repeating the same mistakes, exposures, or misconfigurations, not only improving your security posture, but also minimizing time lost to preventable situations.

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

Implementation guidance

It's important to implement a *lessons learned* framework that establishes and achieves, at a high level, the following points:

- When is a lessons learned held?
- What is involved in the lessons learned process?
- How is a lessons learned performed?
- Who is involved in the process and how?
- How will areas of improvement be identified?
- How will you ensure improvements are effectively tracked and implemented?

The framework should not focus on or blame individuals, but instead should focus on improving tools and processes.

Implementation steps

Aside from the preceding high-level outcomes listed, it’s important to make sure that you ask the right questions to derive the most value (information that leads to actionable improvements) from the process. Consider these questions to help get you started in fostering your lessons learned discussions:

- What was the incident?
- When was the incident first identified?

- How was it identified?
- What systems alerted on the activity?
- What systems, services, and data were involved?
- What specifically occurred?
- What worked well?
- What didn't work well?
- Which process or procedures failed or failed to scale to respond to the incident?
- What can be improved within the following areas:
  - **People**
    - Were the people who were needed to be contacted actually available and was the contact list up to date?
    - Were people missing training or capabilities needed to effectively respond and investigate the incident?
    - Were the appropriate resources ready and available?
  - **Process**
    - Were processes and procedures followed?
    - Were processes and procedures documented and available for this (type of) incident?
    - Were required processes and procedures missing?
    - Were the responders able to gain timely access to the required information to respond to the issue?
  - **Technology**
    - Did existing alerting systems effectively identify and alert on the activity?
    - How could we have reduced time-to-detection by 50%?
    - Do existing alerts need improvement or new alerts need to be built for this (type of) incident?
    - Did existing tools allow for effective investigation (search/analysis) of the incident?
    - What can be done to help identify this (type of) incident sooner?
    - What can be done to help prevent this (type of) incident from occurring again?
    - Who owns the improvement plan and how will you test that it has been implemented?
    - What is the timeline for the additional monitoring or preventative controls and processes to be implemented and tested?

This list isn't all-inclusive, but is intended to serve as a starting point for identifying what the organization and business needs are and how you can analyze them in order to most effectively learn from incidents and continuously improve your security posture. Most important is getting started by incorporating lessons learned as a standard part of your incident response process, documentation, and expectations across the stakeholders.

Resources

Related documents:

- AWS Security Incident Response Guide - Establish a framework for learning from incidents
- NCSC CAF guidance - Lessons learned

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

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 AWS 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 instances used in a Region. 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 AWS 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 approximately 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

This article explains how to maintain space between the resource quota and your usage, and how it can benefit your organization. After you finish using a resource, the usage quota may continue to account for that resource. This can result in a failing or inaccessible resource. Prevent resource failure by verifying that your quotas cover the overlap of inaccessible resources and their replacements. Consider cases like network failure, Availability Zone failure, or Region 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 service disruption events impact application availability, use the cloud to implement strategies to recover from these events. An example strategy is creating additional resources to replace inaccessible resources to accommodate failover conditions without exhausting your service limit.

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

Implementation guidance

When evaluating a quota limit, consider failover cases that might occur due to some degradation. Consider the following failover cases.

- A disrupted or inaccessible VPC.
- An inaccessible subnet.
- A degraded Availability Zone that impacts resource accessibility.
- Networking routes or ingress and egress points are blocked or changed.
- A degraded Region that impacts resource accessibility.
- A subset of resources affected by a failure in a Region or an Availability Zone.

The decision to failover is unique for each situation, as the business impact can vary. Address resource capacity planning in the failover location and the resources' quotas before deciding to failover an application or service.

Consider higher than normal peaks of activity when reviewing quotas for each service. These peaks might be related to resources that are inaccessible due to networking or permissions, but are still active. Unterminated active resources count against the service quota limit.

Implementation steps

- Maintain space between your service quota and your maximum usage to accommodate for a failover or loss of accessibility.
- Determine your service quotas. Account for typical deployment patterns, availability requirements, and consumption growth.
- Request quota increases if necessary. Anticipate a wait time for the quota increase request.
- Determine your reliability requirements (also known as your number of nines).
- Understand potential fault scenarios such as loss of a component, an Availability Zone, or a Region.
- Establish your deployment methodology (examples include canary, blue/green, red/black, and rolling).
- Include an appropriate buffer to the current quota limit. An example buffer could be 15%.
- Include calculations for static stability (Zonal and Regional) where appropriate.
- Plan consumption growth and monitor your consumption trends.
- Consider the static stability impact for your most critical workloads. Assess resources conforming to a statically stable system in all Regions and Availability Zones.
- Consider using On-Demand Capacity Reservations to schedule capacity ahead of any failover. This is a useful strategy to implement for 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 healthy 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 AWS 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 so that you can cover 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.

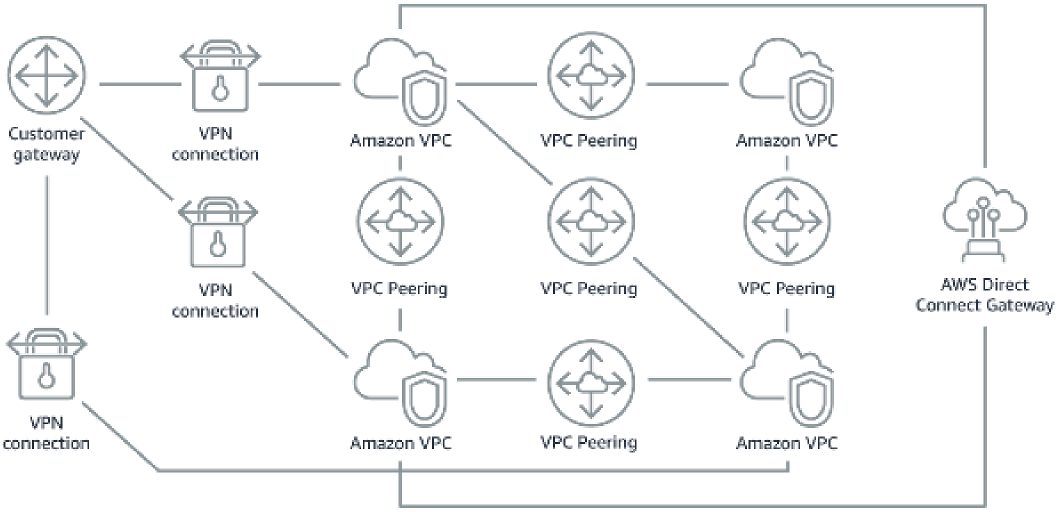


Figure 1: Without AWS Transit Gateway: You need to peer each Amazon VPC to each other and to each onsite location using a VPN connection, which can become complex as it scales.

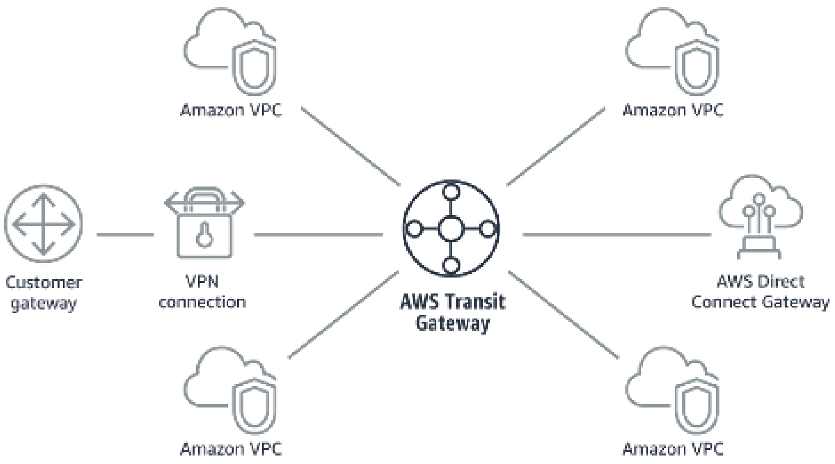


Figure 2: With AWS Transit Gateway: You simply connect each Amazon VPC or VPN to the AWS Transit Gateway and it routes traffic to and from each VPC or VPN.

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:

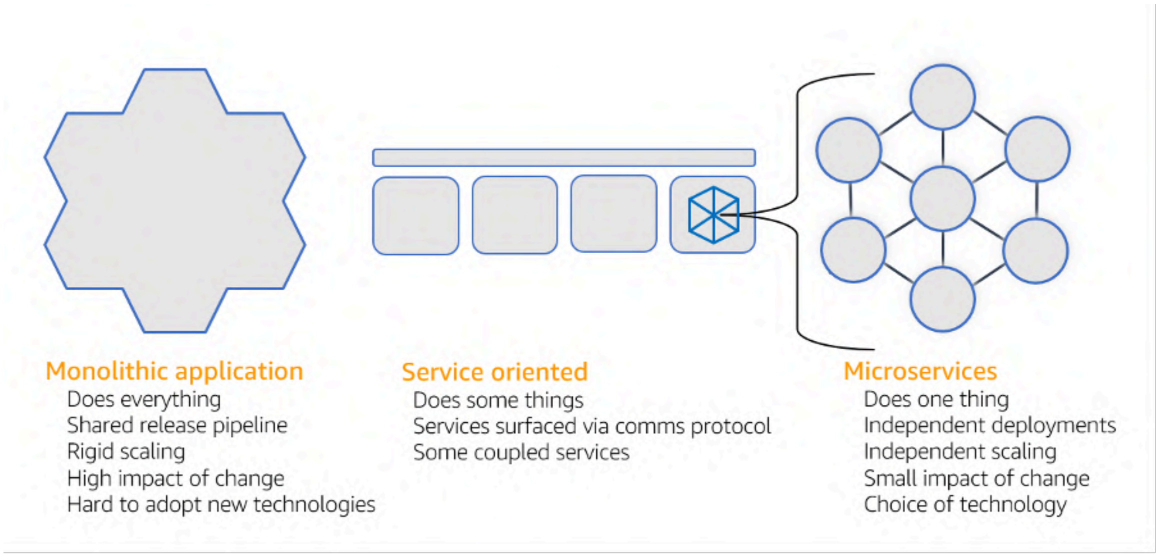
- More specific segments lead to greater agility, organizational flexibility, and scalability.
- Reduced impact of service interruptions.
- Application components may have different availability requirements, which can be supported by a more atomic segmentation.
- Well-defined responsibilities for teams 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

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:

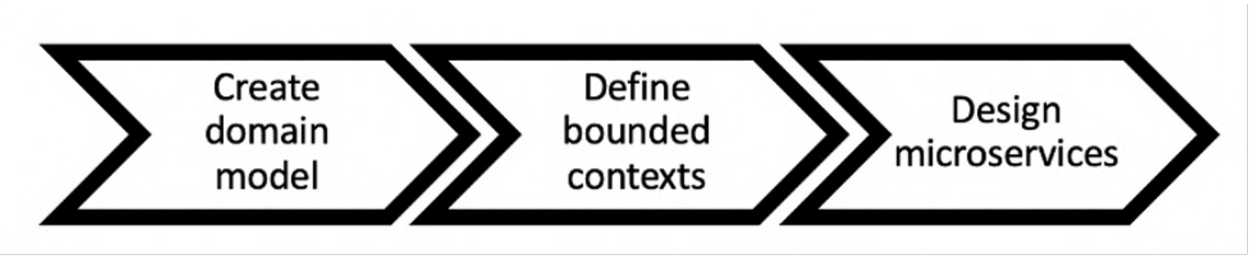
- Teams are formed around specific technical domains like UI and UX, middleware, or database instead of specific business domains.
- Applications span domain responsibilities. Services that span bounded contexts can be more difficult to maintain, require larger testing efforts, and require multiple domain 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 design (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 design

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:

- 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

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

This best practice was updated with new guidance on December 6, 2023.

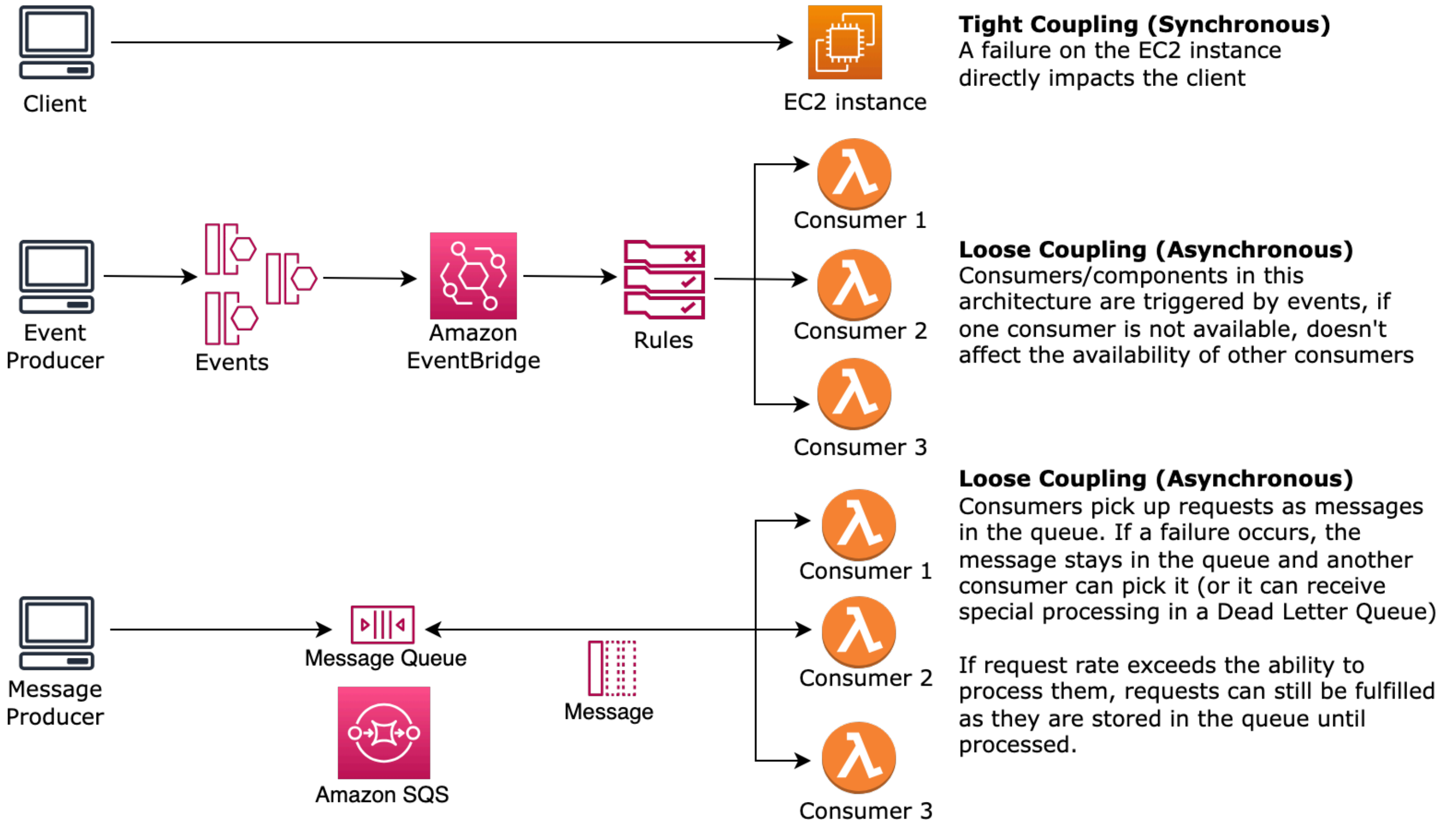
Dependencies such as queuing systems, streaming systems, workflows, and load balancers are loosely coupled. Loose coupling helps isolate behavior of a component from other components that depend on it, increasing resiliency and agility.

Decoupling dependencies, such as queuing systems, streaming systems, and workflows, help minimize the impact of changes or failure on a system. This separation isolates a component's behavior from affecting others that depend on it, improving resilience and agility.

In tightly coupled systems, changes to one component can necessitate changes in other components that rely on it, resulting in degraded performance across all components. *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 modify code or add features to a component while minimizing risk to other components that depend on it. It also allows for granular resilience at a component level where 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 Amazon SQS queue, a streaming data platform such as Amazon Kinesis, or AWS Step Functions.



Tight Coupling (Synchronous)
A failure on the EC2 instance directly impacts the client

Loose Coupling (Asynchronous)
Consumers/components in this architecture are triggered by events, if one consumer is not available, doesn't affect the availability of other consumers

Loose Coupling (Asynchronous)
Consumers pick up requests as messages in the queue. If a failure occurs, the message stays in the queue and another consumer can pick it (or it can receive special processing in a Dead Letter Queue)

If request rate exceeds the ability to process them, requests can still be fulfilled as they are stored in the queue until processed.

Figure 4: Dependencies such as queuing systems and load balancers are loosely coupled

Amazon SQS queues and AWS Step Functions 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.

Desired outcome: Implementing loosely coupled dependencies allows you to minimize the surface area for failure to a component level, which helps diagnose and resolve issues. It also simplifies development cycles, allowing teams to implement changes at a modular level without affecting the performance of other components that depend on it. This approach provides the capability to scale out at a component level based on resource needs, as well as utilization of a component contributing to cost-effectiveness.

Common anti-patterns:

- Deploying a monolithic workload.
- Directly invoking APIs between workload tiers with no capability of failover or asynchronous processing of the request.
- Tight coupling using shared data. Loosely coupled systems should avoid sharing data through shared databases or other forms of tightly coupled data storage, which can reintroduce tight coupling and hinder scalability.
- Ignoring back pressure. Your workload should have the ability to slow down or stop incoming data when a component can't process it at the same rate.

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. There are various solutions that allow you to build loosely coupled applications. These include services for implementing fully managed queues, automated workflows, react to events, and APIs among others which can help isolate behavior of components from other components, and as such increasing resilience and agility.

- **Build event-driven architectures:** Amazon EventBridge helps you build loosely coupled and distributed event-driven architectures.
- **Implement queues in distributed systems:** You can use Amazon Simple Queue Service (Amazon SQS) to integrate and decouple distributed systems.
- **Containerize components as microservices:** Microservices allow teams to build applications composed of small independent components which communicate over well-defined APIs. Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (Amazon EKS) can help you get started faster with containers.
- **Manage workflows with Step Functions:** Step Functions help you coordinate multiple AWS services into flexible workflows.
- **Leverage publish-subscribe (pub/sub) messaging architectures:** Amazon Simple Notification Service (Amazon SNS) provides message delivery from publishers to subscribers (also known as producers and consumers).

Implementation steps

- Components in an event-driven architecture are initiated by events. Events are actions that happen in a system, such as a user adding an item to a cart. When an action is successful, an event is generated that actuates the next component of the system.
  - Building Event-driven Applications with Amazon EventBridge
  - AWS re:Invent 2022 - Designing Event-Driven Integrations using Amazon EventBridge
- Distributed messaging systems have three main parts that need to be implemented for a queue based architecture. They include components of the distributed system, the queue that is used for decoupling (distributed on Amazon SQS servers), and the messages in the queue. A typical system has producers which initiate the message into the queue, and the consumer which receives the message from the queue. The queue stores messages across multiple Amazon SQS servers for redundancy.
  - Basic Amazon SQS architecture

  - Send Messages Between Distributed Applications with Amazon Simple Queue Service
- Microservices, when well-utilized, enhance maintainability and boost scalability, as loosely coupled components are managed by independent teams. It also allows for the isolation of behaviors to a single component in case of changes.
  - Implementing Microservices on AWS
  - Let's Architect! Architecting microservices with containers
- With AWS Step Functions you can build distributed applications, automate processes, orchestrate microservices, among other things. The orchestration of multiple components into an automated workflow allows you to decouple dependencies in your application.
  - Create a Serverless Workflow with AWS Step Functions and AWS Lambda
  - Getting Started with AWS Step Functions

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?
- Break up with your monolith
- Orchestrate Queue-based Microservices with AWS Step Functions and Amazon SQS
- Basic Amazon SQS architecture
- Queue-Based Architecture

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
- AWS re:Invent 2022 - Designing event-driven integrations using Amazon EventBridge
- AWS re:Invent 2017: Elastic Load Balancing Deep Dive and Best Practices

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

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