Operational Excellence Pillar

AWS Well-Architected Framework

Operational Excellence Pillar: AWS Well-Architected Framework

Table of Contents

Operational Excellence Pillar - AWS Well-Architected Framework

Publication date: **April 10, 2023** (*Document revisions* (p. 123))

The focus of this paper is the operational excellence pillar of the AWS Well-Architected Framework. It provides guidance to help you apply best practices in the design, delivery, and maintenance of AWS workloads.

Introduction

The AWS Well-Architected Framework helps you understand the benefits and risks of decisions you make while building workloads on AWS. By using the Framework you will learn operational and architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable workloads in the cloud. It provides a way to consistently measure your operations and architectures against best practices and identify areas for improvement. We believe that having Well-Architected workloads that are designed with operations in mind greatly increases the likelihood of business success.

The framework is based on six pillars:

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

This paper focuses on the operational excellence pillar and how to apply it as the foundation of your well-architected solutions. Operational excellence is challenging to achieve in environments where operations is perceived as a function isolated and distinct from the lines of business and development teams that it supports. By adopting the practices in this paper you can build architectures that provide insight to their status, are activated for effective and efficient operation and event response, and can continue to improve and support your business goals.

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

Operational excellence

At Amazon, we define operational excellence as a commitment to build software correctly while consistently delivering a great customer experience. It contains best practices for organizing your team, designing your workload, operating it at scale, and evolving it over time. Operational excellence helps your team to focus more of their time on building new features that benefit customers, and less time on maintenance and firefighting. To build correctly, we look to best practices that result in well-running systems, a balanced workload for you and your team, and most importantly, a great customer experience.

The goal of operational excellence is to get new features and bug fixes into customers' hands quickly and reliably. Organizations that invest in operational excellence consistently delight customers while building new features, making changes, and dealing with failures. Along the way, operational excellence drives towards continuous integration and continuous delivery (CI/CD) by helping developers achieve high quality results consistently.

Design principles

The following are the 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 to allow components to be updated regularly to increase the flow of beneficial changes into your workload. Make changes in small increments that can be reversed if they fail to aid in the identification and resolution of issues introduced to your environment (without affecting customers when possible).
- **Refine operations procedures frequently:** As you use operations procedures, look for opportunities to improve them. As you evolve your workload, evolve your procedures appropriately. Set up regular game days to review and validate that all procedures are effective and that teams are familiar with them.
- **Anticipate failure:** Perform "pre-mortem" exercises to identify potential sources of failure so that they can be removed or mitigated. Test your failure scenarios and validate your understanding of their impact. Test your response procedures to ensure 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.

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 activate 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. You must 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 help 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.

Organization

You need to understand your organization’s priorities, your organizational structure, and how your organization supports your team members, so that they can support your business outcomes.

To create operational excellence, you must understand the following:

Topics

Organization priorities

Your teams need to have a shared understanding of your entire workload, their role in it, and shared business goals to set the priorities that will create business success. Well-defined priorities will maximize the benefits of your efforts. Review your priorities regularly so that they can be updated as your organization's needs change.

Best practices

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 (p. 8) - 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.

Operating model

Your teams must understand their part in achieving business outcomes. Teams need to 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 their industry, their organization, the makeup of the team, and the characteristics of their workload. It is unreasonable to expect a single operating model to be able to support all teams and their workloads.

The number of operating models present in an organization is likely to increase with the number of development teams. You may need to use a combination of operating models.

Adopting standards and consuming services can simplify operations and limit the support burden in your operating model. The benefit of development efforts on shared standards is magnified by the number of teams who have adopted the standard and who will adopt new features.

It’s critical that mechanisms exist to request additions, changes, and exceptions to standards in support of the teams’ activities. Without this option, standards become a constraint on innovation. Requests should be approved where viable and determined to be appropriate after an evaluation of benefits and risks.

A well-defined set of responsibilities will reduce the frequency of conflicting and redundant efforts. Business outcomes are easier to achieve when there is strong alignment and relationships between business, development, and operations teams.

Operating model 2 by 2 representations

These operating model 2 by 2 representations are illustrations to help you understand the relationships between teams in your environment. These diagrams focus on who does what and the relationships between teams, but we will also discuss governance and decision making in context of these examples.

Your teams may have responsibilities in multiple parts of multiple models depending on the workloads they support. You may wish to break out more specialized discipline areas than the high-level ones described. There is the potential for endless variation on these models as you separate or aggregate activities, or overlay teams and provide more specific detail.

You may identify that you have overlapping or unrecognized capabilities across teams that can provide additional advantage, or lead to efficiencies. You may also identify unsatisfied needs in your organization that you can plan to address.

When evaluating organizational change, examine the trade-offs between models, where your individual teams exist within the models (now and after the change), how your teams’ relationship and responsibilities will change, and if the benefits merit the impact on your organization.

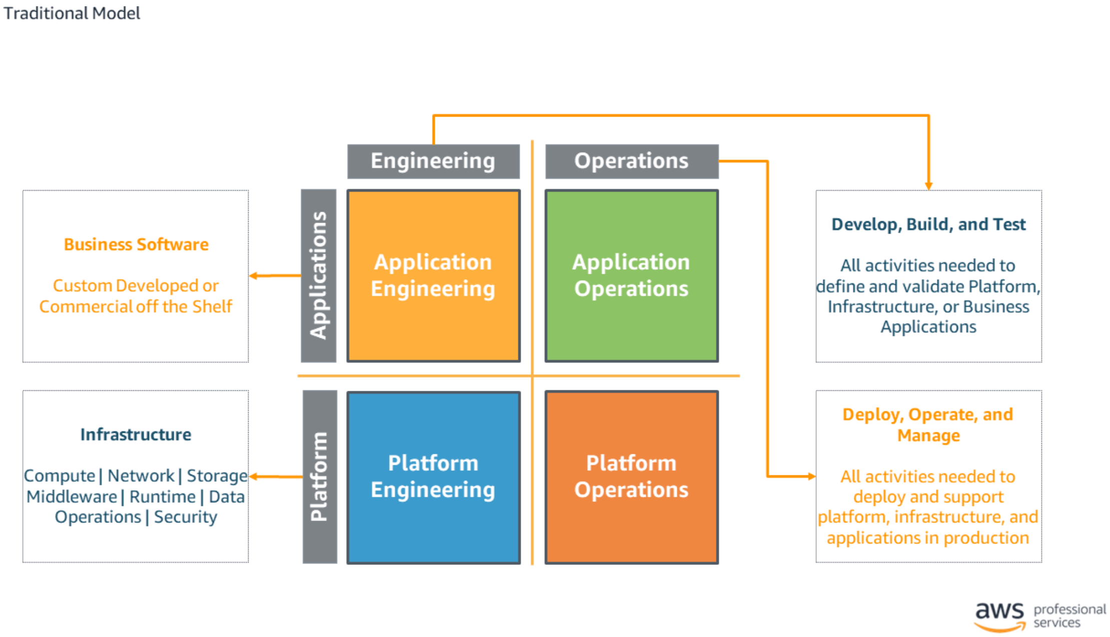
You can be successful using each of the following four operating models. Some models are more appropriate for specific use cases or at specific points in your development. Some of these models may provide advantages over the ones in use in your environment.

Topics

Fully separated operating model

In the following diagram, on the vertical axis we have Applications and Platform. Applications refer to the workload serving a business outcome and can be custom developed or purchased software. Platform refers to the physical and virtual infrastructure and other software that supports that workload.

On the horizontal axis, we have Engineering and Operations. Engineering refers to the development, building, and testing of applications and infrastructure. Operations is the deployment, update, and ongoing support of applications and infrastructure.


Traditional Model
Engineering
Operations
Applications
Platform
Application Engineering
Application Operations
Platform Engineering
Platform Operations
Business Software
Custom Developed or Commercial off the Shelf
Infrastructure
Compute | Network | Storage
Middleware | Runtime | Data
Operations | Security
Develop, Build, and Test
All activities needed to define and validate Platform, Infrastructure, or Business Applications
Deploy, Operate, and Manage
All activities needed to deploy and support platform, infrastructure, and applications in production
aws professional services

In many organizations, this "fully separated" model is present. The activities in each quadrant are performed by a separate team. Work is passed between teams through mechanisms such as work requests, work queues, tickets, or by using an IT service management (ITSM) system.

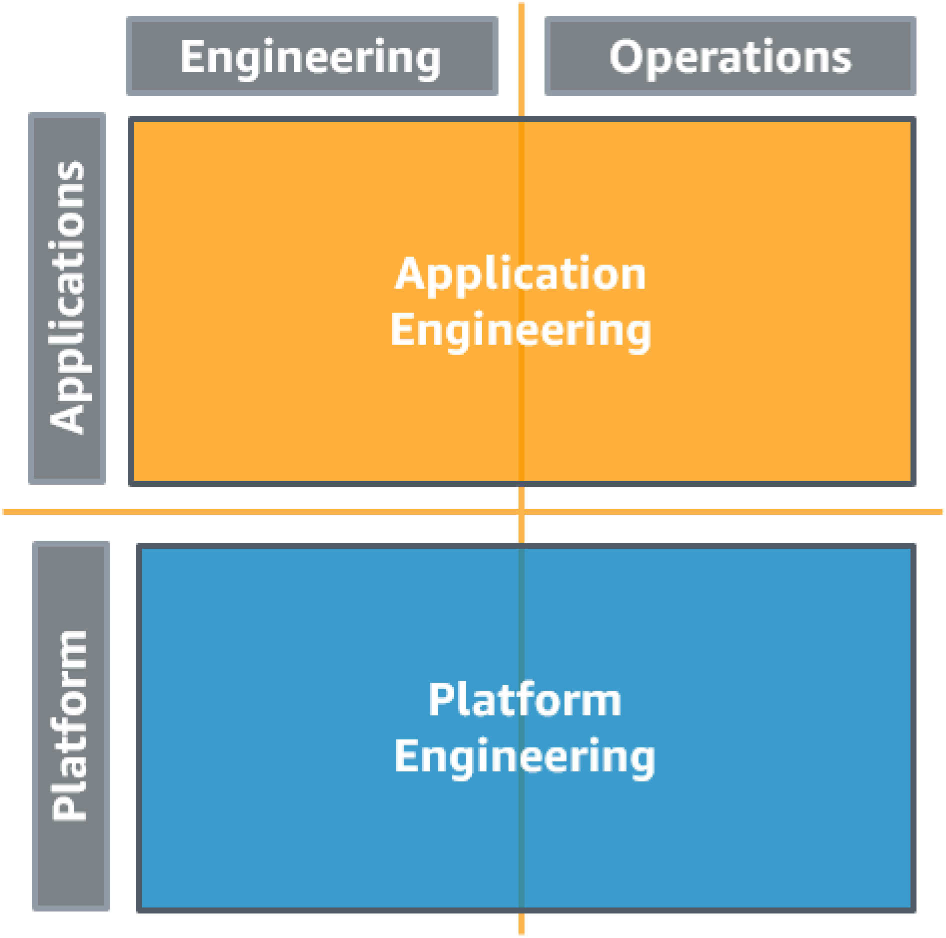
The transition of tasks to or between teams increases complexity, and creates bottlenecks and delays. Requests may be delayed until they are a priority. Defects identified late may require significant rework and may have to pass through the same teams and their functions once again. If there are incidents that require action by engineering teams, their responses are delayed by the hand off activity.

There is a higher risk of misalignment when business, development, and operations teams are organized around the activities or functions that are being performed. This can lead to teams focusing on their specific responsibilities instead of focusing on achieving business outcomes. Teams may be narrowly specialized, physically isolated, or logically isolated, hindering communication and collaboration.

Separated Application Engineering and Operations (AEO) and Infrastructure Engineering and Operations (IEO) with centralized governance

This “Separated AEO and IEO” model follows a “you build it you run it” methodology.

Your application engineers and developers perform both the engineering and the operation of their workloads. Similarly, your infrastructure engineers perform both the engineering and operation of the platforms they use to support application teams.



For this example, we are going to treat governance as centralized. Standards are distributed, provided, or shared to the application teams.

You should use tools or services that help you to centrally govern your environments across accounts, such as AWS Organizations. Services like AWS Control Tower expand this management capability helping 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.

“You build it you run it” does not mean that the application team is responsible for the full stack, tool chain, and platform.

The platform engineering team provides a standardized set of services (for example, development tools, monitoring tools, backup and recovery tools, and network) to the application team. The platform team may also provide the application team access to approved cloud provider services, specific configurations of the same, or both.

Mechanisms that provide a self-service capability for deploying approved services and configurations, such as Service Catalog, can help limit delays associated with fulfillment requests while enforcing governance.

The platform team activates full stack visibility so that application teams can differentiate between issues with their application components and the services and infrastructure components their applications consume. The platform team may also provide assistance configuring these services and guidance on how to improve the applications teams’ operations.

As discussed previously, it’s critical that mechanisms exist for the application team to request additions, changes, and exceptions to standards in support of teams’ activities and innovation of their application.

The Separated AEO IEO model provides strong feedback loops to application teams. Day to day operations of a workload increases contact with customers either through direct interaction or indirectly through support and feature requests. This heightened visibility allows application teams to address issues more quickly. The deeper engagement and closer relationship provides insight to customer needs and creates more rapid innovation.

All of this is also true for the platform team supporting the application teams.

Adopted standards may be pre-approved for use, reducing the amount of review necessary to enter production. Consuming supported and tested standards provided by the platform team may reduce the frequency of issues with those services. Adoption of standards helps application teams to focus on differentiating their workloads.

Separated AEO and IEO with centralized governance and a service provider

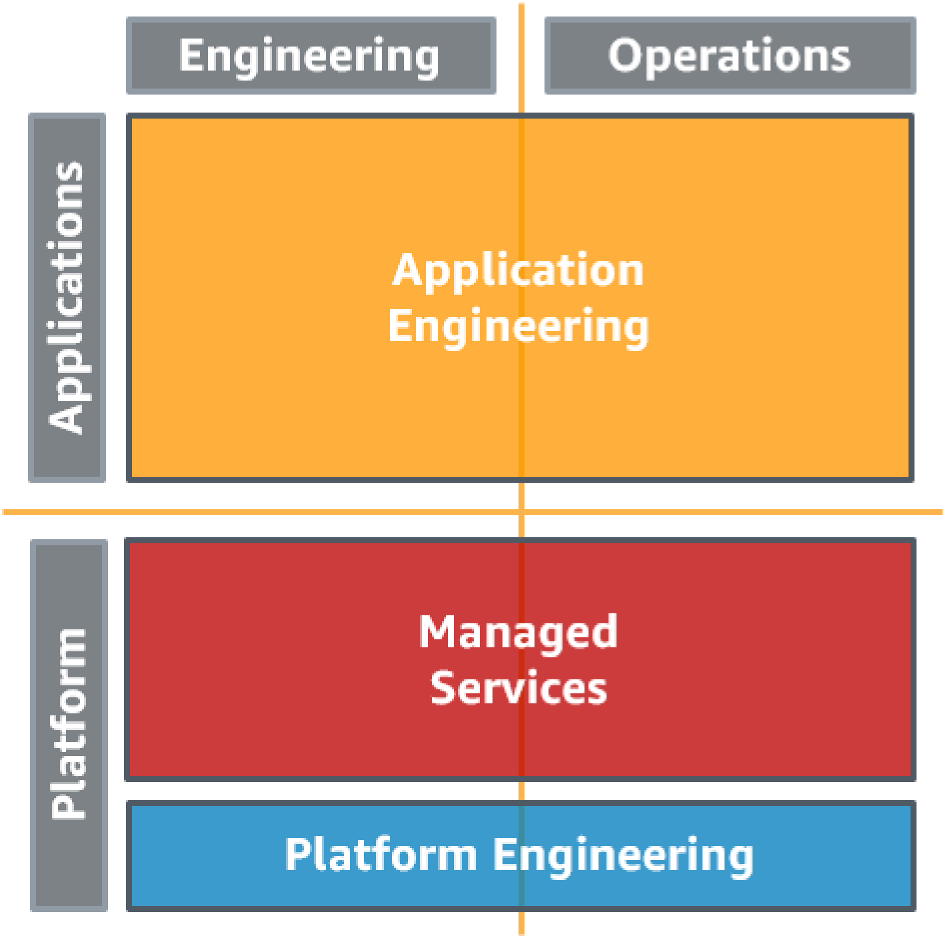
This “Separated AEO and IEO” model follows a “you build it you run it” methodology.

Your application engineers and developers perform both the engineering and the operation of their workloads.

Your organization may not have the existing skills, or team members, to support a dedicated platform engineering and operations team, or you may not want to make the investments of time and effort to do so.

Alternatively, you may wish to have a platform team that is focused on creating capabilities that will differentiate your business, but you want to offload the undifferentiated day to day operations to an outsourcer.

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.



For this variation, we are going to treat governance as centralized and managed by the platform team, with account creation and policies managed with AWS Organizations and AWS Control Tower.

This model does require you to modify your mechanisms to work with those of your service provider. It does not address the bottlenecks and delays created by transition of tasks between teams, including your service provider, or the potential rework related to the late identification of defects.

You gain the advantage of your providers' standards, best practices, processes, and expertise. You also gain the benefits of their ongoing development of their service offerings.

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.

Separated AEO and IEO with centralized governance and an internal service provider consulting partner

This “Separated AEO and IEO” model seeks to establish a “you build it you run it” methodology.

You want your application teams to perform the engineering and operations activities for their workloads, and to adopt a more DevOps like culture.

Your application teams may be in-progress migrating, adopting the cloud, or modernizing your workloads, and not have the existing skills to adequately support cloud and cloud operations. This lack of application team capabilities or familiarity may be barriers to your efforts.

To address this concern you establish a Cloud Center of Enablement team (CCoE) that provides a forum to ask questions, discuss needs, and identify solutions. Depending on the needs of your organization, the CCoE can be a dedicated team of experts or a virtual team with participants selected from across your organization. The CCoE provides cloud transformation for teams, establishes centralized cloud governance, and defines account and organization management standards. They also identify successful reference architectures and patterns for enterprise use.

We refer to CCoE as Cloud Center of Enablement, instead of the more common Cloud Center of Excellence, to place the emphasis on creating the success of the supported teams and the achievement of business outcomes.

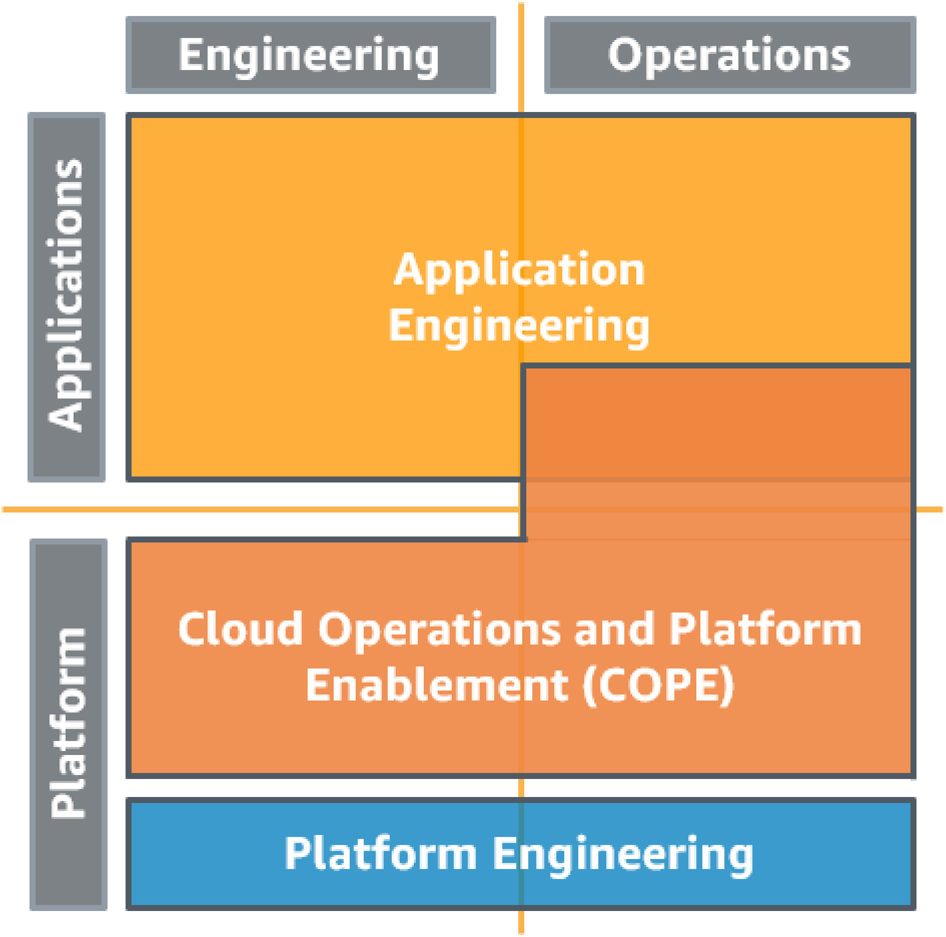
Your platform engineering team builds the core shared platform capabilities based on those standards for application teams to adopt. They codify the enterprise reference architectures and patterns that are provided to the application teams through a self-service mechanism. Using a service such as AWS Service Catalog the application teams can deploy approved reference architectures, patterns, services, and configurations, compliant by default with the centralized governance and security standards.

The platform engineering team also provides a standardized set of services (for example, development tools, monitoring tools, backup and recovery tools, and network) to the application teams.

Your organization has an “Internal MSP and Consulting Partner” that manages and supports the standardized services and provides assistance to application teams establishing their cloud presence based on the reference architectures and patterns. This “Cloud Operations and Platform Enablement (COPE)” team works with the applications teams to help them establish baseline operations with the application teams progressively taking more responsibility for their systems and resources over time. The COPE team drives continual improvement together with the CCoE and Platform Engineering teams, and acts as proponents for the application teams.

The application teams get assistance setting up environments, CICD pipelines, change management, observability and monitoring, and establishing incident and event management processes with the COPE team integrated with those of the enterprise as required. The COPE team participates with the application teams in the performance of these operations activities, phasing out the COPE team engagement over time as the application teams take ownership.

The application team gains the benefit of the skills of the COPE team and the lessons learned by the organization. They are protected by the guardrails established through centralized governance. The application team builds upon recognized successes and gain the benefit of continuing development of the organizational standards they have adopted. They gain greater insight to the operation of their workload through the process of establishing observability and monitoring, and are better able to understand the impact of changes they make to their workloads.



The COPE team retains the access necessary to support operations activities, provide an enterprise-operations view spanning application teams, and to provide critical incident management support. The COPE team retains responsibility for activities considered undifferentiated heavy lifting, which they satisfy through standard solutions supportable at scale. They also continue to manage well-understood programmatic and automated operations activities for the application teams so that they can focus on differentiating their applications.

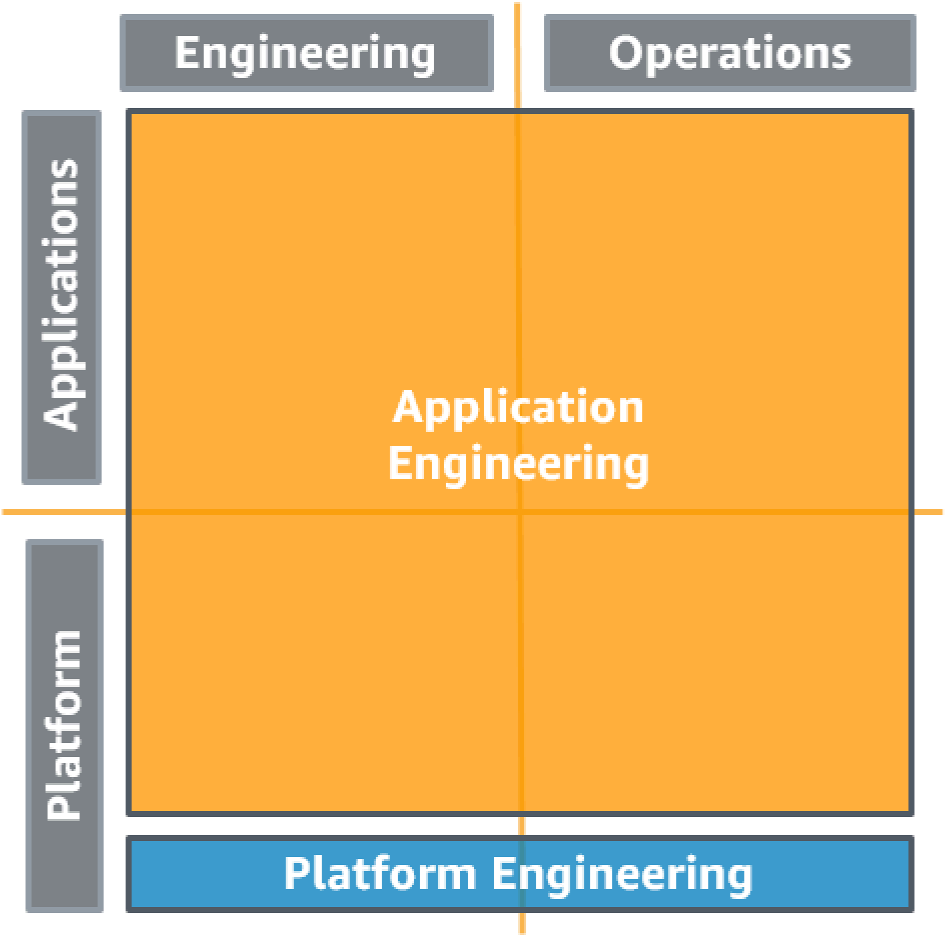
You gain the advantage of your organization's standards, best practices, processes, and expertise derived from the successes of your teams. You establish a mechanism to replicate these successful patterns for new teams adopting or modernizing on the cloud. This model places emphasis on the COPE team's ability to help application team get established, and transition knowledge and artifacts. It reduces the operational burdens of the application teams with the risk that application teams will fail to become largely independent. It establishes relationships between CCoE, COPE, and application teams creating a feedback loop to support further evolution and innovation.

Establishing your CCoE and COPE teams, while defining organization wide standards, can facilitate cloud adoption and support modernization efforts. By providing the additional supports of a COPE team acting as consultants and partners to your application teams you can remove barriers that slow application team adoption of beneficial cloud capabilities.

Separated AEO and IEO with decentralized governance

This “Separated AEO and IEO” model follows a “you build it you run it” methodology.

Your application engineers and developers perform both the engineering and the operation of their workloads. Similarly your infrastructure engineers perform both the engineering and operation of the platforms they use to support application teams.



For this example, we are going to treat governance as decentralized.

Standards are still distributed, provided, or shared to application teams by the platform team, but application teams are free to engineer and operate new platform capabilities in support of their workload.

In this model, there are fewer constraints on the application team, but that comes with a significant increase in responsibilities. Additional skills, and potentially team members, must be present to support the additional platform capabilities. The risk of significant rework is increased if skill sets are not adequate and defects are not recognized early.

You should enforce policies that are not specifically delegated to application teams. Use tools or services that help you to centrally govern your environments across accounts, such as AWS Organizations. Services like AWS Control Tower expand this management capability helping 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.

It’s beneficial to have mechanisms for the application team to request additions and changes to standards. They may be able to contribute new standards that can provide benefit to other application teams. The platform teams may decide that providing direct support for these additional capabilities is an effective support for business outcomes.

This model limits constraints on innovation with significant skill and team member requirements. It addresses many of the bottlenecks and delays created by transition of tasks between teams while still promoting the development of effective relationships between teams and customers.

Relationships and ownership

Your operating model defines the relationships between teams and supports identifiable ownership and responsibility.

Best practices

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 (p. 23) - The processes and procedures to support resources depends on resource ownership.
- OPS02-BP04 Team members know what they are responsible for (p. 24) - Team members should understand what resources they are owners of.
- OPS02-BP05 Mechanisms exist to identify responsibility and ownership (p. 24) - 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 (p. 21), OPS02-BP02 Processes and procedures have identified owners (p. 23), or OPS02-BP03 Operations activities have identified owners responsible for their performance (p. 23), 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 (p. 21) - Resources need identified owners before you build a change management process.
- OPS02-BP02 Processes and procedures have identified owners (p. 23) - Processes need identified owners before you build a change management process.
- OPS02-BP03 Operations activities have identified owners responsible for their performance (p. 23) - 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 (p. 23) - Process ownership must be identified before setting agreements between teams.
- OPS02-BP03 Operations activities have identified owners responsible for their performance (p. 23) - 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

Organizational culture

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

Best practices

- OPS03-BP05 Experimentation is encouraged (p. 31)
- OPS03-BP06 Team members are encouraged to maintain and grow their skill sets (p. 33)
- OPS03-BP07 Resource teams appropriately (p. 34)
- OPS03-BP08 Diverse opinions are encouraged and sought within and across teams (p. 35)

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 (p. 69) - Make communications actionable by supplying a runbook when the outcome is known.
- OPS07-BP04 Use playbooks to investigate issues (p. 72) - 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 (p. 110) - Learning from incidents is an important driver for innovation along with experimentation.
- OPS11-BP03 Implement feedback loops (p. 110) - 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

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.

To prepare for operational excellence, you need to perform the following:

Topics

Design telemetry

Design your workload so that it provides the information necessary for you to understand its internal state (for example, metrics, logs, events, and traces) across all components in support of observability and investigating issues. Iterate to develop the telemetry necessary to monitor the health of your workload, identify when outcomes are at risk, and create effective responses.

In AWS, you can emit and collect logs, metrics, and events from your applications and workloads components to help you to understand their internal state and health. You can integrate distributed tracing to track requests as they travel through your workload. Use this data to understand how your application and underlying components interact and to analyze issues and performance.

When instrumenting your workload, capture a broad set of information to provide situational awareness (for example, changes in state, user activity, privilege access, utilization counters), knowing that you can use filters to select the most useful information over time.

Best practices

OPS04-BP01 Implement application telemetry

Application telemetry is the foundation for observability of your workload. Your application should emit telemetry that provides insight into the state of the application and the achievement of business outcomes. From troubleshooting to measuring the impact of a new feature, application telemetry informs the way you build, operate, and evolve your workload.

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

Desired Outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

1. Identify a central location for telemetry storage for the applications in your workload. The location should support both collection of telemetry and analysis capabilities. Anomaly detection and automated insights are recommended features.
   a. Amazon CloudWatch provides telemetry collection, dashboards, analysis, and event generation capabilities.
2. To identify what telemetry you need, start by answering this question: what is the state of my application? Your application should emit logs and metrics that collectively answer this question. If you can’t answer the questions with the existing application telemetry, work with business and engineering stakeholders to create a list of telemetry requirements.

   a. You can request expert technical advice from your AWS account team as you identify and develop new application telemetry.

3. Once the additional application telemetry has been identified, work with your engineering stakeholders to instrument your application.
   a. The AWS Distro for Open Telemetry provides APIs, libraries, and agents that collect application telemetry. This example demonstrates how to instrument a JavaScript application with custom metrics.
   b. If you want to understand the observability services that AWS offers, work through the One Observability Workshop or request support from your AWS account team.
   c. For a deeper dive into application telemetry, read the Instrumenting distributed systems for operational visibility article in the Amazon Builder's Library, which explains how Amazon instruments applications and can serve as a guide for developing your own instrumentation guidelines.

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

Resources

Related best practices:

the section called "OPS04-BP02 Implement and configure workload telemetry" (p. 39) – Application telemetry is a component of workload telemetry. In order to understand the health of the overall workload you need to understand the health of individual applications that make up the workload.

the section called "OPS04-BP03 Implement user activity telemetry" (p. 40) – User activity telemetry is often a subset of application telemetry. User activity like add to cart events, click streams, or completed transactions provide insight into the user experience.

the section called "OPS04-BP04 Implement dependency telemetry" (p. 42) – Dependency checks are related to application telemetry and may be instrumented into your application. If your application relies on external dependencies like DNS or a database your application can emit metrics and logs on reachability, timeouts, and other events.

the section called "OPS04-BP05 Implement transaction traceability" (p. 44) – Tracing transactions across a workload requires each application to emit information about how they process shared events. The way individual applications handle these events is emitted through their application telemetry.

the section called "OPS08-BP02 Define workload metrics" (p. 80) – Workload metrics are the key health indicators for your workload. Key application metrics are a part of workload metrics.

Related documents:

- AWS Builders Library – Instrumenting Distributed Systems for Operational Visibility
- AWS Distro for OpenTelemetry
- AWS Well-Architected Operational Excellence Whitepaper – Design Telemetry
- Creating metrics from log events using filters
- Implementing Logging and Monitoring with Amazon CloudWatch
- Monitoring application health and performance with AWS Distro for OpenTelemetry
- New – How to better monitor your custom application metrics using Amazon CloudWatch Agent
- Observability at AWS
- Scenario – Publish metrics to CloudWatch

- Start Building – How to Monitor your Applications Effectively
- Using CloudWatch with an AWS SDK

Related videos:

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

Related examples:

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

Related services:

- Amazon CloudWatch

OPS04-BP02 Implement and configure workload telemetry

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

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

Common anti-patterns:

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

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

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

Implementation guidance

- Implement log and metric telemetry: Instrument your workload to emit information about its internal state, status, and the achievement of business outcomes. Use this information to determine when a response is required.

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

Resources

Related documents:

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

Related videos:

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

OPS04-BP03 Implement user activity telemetry

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

1. Instrument your application to emit telemetry (metrics, events, logs, and traces) about user activity. Once instrumented, front-end components emit telemetry automatically as the user interacts with the user interface. Backend applications emit telemetry on user events and transactions.
   a. Amazon CloudWatch RUM can provide insight into end user experience for front-end applications.
   b. You can use the AWS Distro for Open Telemetry to instrument and capture telemetry from your applications.
   c. Amazon Pinpoint can analyze user behavior through campaigns, providing insight on user engagement.
   d. Customers with Enterprise Support can request the Building a Monitoring Strategy Workshop from their Technical Account Manager. This workshop helps you build an observability strategy for your workload.
2. Establish synthetic user activity to monitor your application. Synthetic user activity simulates user actions to validate that your application is working properly.
   a. Amazon CloudWatch Synthetics can simulate user activity using canaries.

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

Resources

Related best practices:

- OPS04-BP01 Implement application telemetry (p. 36) - Application telemetry is required in order to build in user activity telemetry.
- OPS04-BP02 Implement and configure workload telemetry (p. 39) - Some user activity telemetry may also be considered workload telemetry.

Related documents:

- How to Monitor your Applications Effectively

Related videos:

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

Related examples:

- Amazon CloudWatch RUM Web Client
- AWS Distro for Open Telemetry
- Implementing Real User Monitoring of Amplify Application using Amazon CloudWatch RUM
- One Observability Workshop

Related services:

- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Amazon Pinpoint

OPS04-BP04 Implement dependency telemetry

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

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

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

Resources

Related best practices:

- OPS04-BP01 Implement application telemetry (p. 36) - You may build dependency monitoring into your application telemetry.

Related documents:

- Monitor your private internal endpoints 24x7 using CloudWatch Synthetics

Related videos:

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

- AWS re:Invent 2022 - Observability best practices at Amazon

Related examples:

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

Related services:

- Amazon CloudWatch Synthetics
- AWS Health

OPS04-BP05 Implement transaction traceability

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

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

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

Resources

Related best practices:

- OPS04-BP01 Implement application telemetry (p. 36) - Application telemetry covers transaction traceability and handling and needs to be implementing first.

Related documents:

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

Related videos:

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

Related examples:

- AWS X-Ray Multi API Gateway Tracing Example

Related services:

- AWS Distro for OpenTelemetry
- AWS X-Ray

Design for operations

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

In AWS, you can view your entire workload (applications, infrastructure, policy, governance, and operations) as code. It can all be defined in and updated using code. This means you can apply the same engineering discipline that you use for application code to every element of your stack.

Best practices

OPS05-BP01 Use version control

Use version control to activate tracking of changes and releases.

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

Common anti-patterns:

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

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

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

Implementation guidance

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

- Introduction to AWS CodeCommit
- What is AWS CodeCommit?

Resources

Related documents:

- What is AWS CodeCommit?

Related videos:

- Introduction to AWS CodeCommit

OPS05-BP02 Test and validate changes

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

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

Desired outcome:

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

Common anti-patterns:

- You deploy a new software change without any tests. It fails to run in production, which leads to an outage.
- New security groups are deployed with 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:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

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

Resources

Related best practices:

- OPS05-BP01 Use version control (p. 46) - All software artifacts must be backed by a version-controlled repository.
- OPS05-BP06 Share design standards (p. 53) - Your organizations software testing standards inform your design standards.
- OPS05-BP10 Fully automate integration and deployment (p. 58) - Software tests should be automatically run as part of your larger integration and deployment pipeline.

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. Some examples include setting the configuration for a web or application server on an instance, or defining the configuration of an AWS service within the AWS Management Console or through the AWS CLI.

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

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

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

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

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Resources

Related documents:

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

Related videos:

- Introduction to AWS CloudFormation

- Introduction to AWS Elastic Beanstalk

OPS05-BP04 Use build and deployment management systems

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Resources

Related documents:

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

Related videos:

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

OPS05-BP05 Perform patch management

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

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

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

- AWS Systems Manager Patch Manager

Resources

Related documents:

- AWS Developer Tools
- AWS Systems Manager Patch Manager

Related videos:

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

Related examples:

- Well-Architected Labs – Inventory and Patch Management

OPS05-BP06 Share design standards

Share best practices across teams to increase awareness and maximize the benefits of development efforts. Document them and keep them up to date as your architecture evolves. If shared standards are enforced in your organization, it’s critical that mechanisms exist to request additions, changes, and exceptions to standards. Without this option, standards become a constraint on innovation.

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

AnyCompany Retail has a cross-functional architecture team that creates software architecture patterns. This team builds the architecture with compliance and governance built in. Teams that adopt these shared standards get the benefits of having compliance and governance built in. They can quickly build on top of the design standard. The architecture team meets quarterly to evaluate architecture patterns and update them if necessary.

Implementation steps

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

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

Resources

Related best practices:

- OPS01-BP03 Evaluate governance requirements (p. 6) - Governance requirements influence design standards.
- OPS01-BP04 Evaluate compliance requirements (p. 8) - Compliance is a vital input in creating design standards.
- OPS07-BP02 Ensure a consistent review of operational readiness (p. 67) - Operational readiness checklists are a mechanism to implement design standards when designing your workload.
- OPS11-BP01 Have a process for continuous improvement (p. 108) - Updating design standards is a part of continuous improvement.
- OPS11-BP04 Perform knowledge management (p. 113) - As part of your knowledge management practice, document and share design standards.

Related documents:

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

Related videos:

- AWS Service Catalog – Getting Started

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

Related examples:

- AWS Service Catalog Reference Architecture
- AWS Service Catalog Workshop

Related services:

- AWS Service Catalog

OPS05-BP07 Implement practices to improve code quality

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

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

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

Resources

Related best practices:

- OPS05-BP06 Share design standards (p. 53) - You can share design standards as part of your code quality practice.

Related documents:

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

Related videos:

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

Related services:

- Amazon CodeGuru Reviewer
- Amazon CodeGuru Profiler
- AWS Cloud9

OPS05-BP08 Use multiple environments

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

Common anti-patterns:

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

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

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

Implementation guidance

- Use multiple environments: Provide developers sandbox environments with minimized controls to aid in experimentation. Provide individual development environments to help work in parallel, increasing development agility. Implement more rigorous controls in the environments approaching production to allow developers to innovate. Use infrastructure as code and configuration management systems to deploy environments that are configured consistent with the controls present in production to ensure systems operate as expected when deployed. When environments are not in use, turn them off to avoid costs associated with idle resources (for example, development systems on evenings and weekends). Deploy production equivalent environments when load testing to improve valid results.
  - What is AWS CloudFormation?
  - How do I stop and start Amazon EC2 instances at regular intervals using AWS Lambda?

Resources

Related documents:

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

OPS05-BP09 Make frequent, small, reversible changes

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

Common anti-patterns:

- You deploy a new version of your application quarterly.

- You frequently make changes to your database schema.
- You perform manual in-place updates, overwriting existing installations and configurations.

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

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

Implementation guidance

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

OPS05-BP10 Fully automate integration and deployment

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

- What is AWS CodeDeploy?

Resources

Related documents:

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

Related videos:

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

Mitigate deployment risks

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

The design of your workload should include how it will be deployed, updated, and operated. You will want to implement engineering practices that align with defect reduction and quick and safe fixes.

Best practices

OPS06-BP01 Plan for unsuccessful changes

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

Common anti-patterns:

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

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

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

Implementation guidance

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

OPS06-BP02 Test and validate changes

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Resources

Related documents:

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

OPS06-BP03 Use deployment management systems

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Resources

Related documents:

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

Related videos:

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

OPS06-BP04 Test using limited deployments

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Resources

Related documents:

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

OPS06-BP05 Deploy using parallel environments

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Resources

Related documents:

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

Related videos:

- Deep Dive on Advanced Continuous Delivery Techniques Using AWS

OPS06-BP06 Deploy frequent, small, reversible changes

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

Common anti-patterns:

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

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

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

Implementation guidance

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

OPS06-BP07 Fully automate integration and deployment

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Resources

Related documents:

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

Related videos:

- Continuous integration best practices for software development
- Deep Dive on Advanced Continuous Delivery Techniques Using AWS

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

OPS06-BP08 Automate testing and rollback

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Resources

Related documents:

- Redeploy and roll back a deployment with AWS CodeDeploy

Operational readiness and change management

Evaluate the operational readiness of your workload, processes, procedures, and personnel to understand the operational risks related to your workload. Manage the flow of change into your environments.

You should 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 need to make plans to address. You will have runbooks that document your routine activities and playbooks that guide your processes for issue resolution. Use a mechanism to manage changes that supports the delivery of business value and help mitigate risks associated to change.

Best practices

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 (p. 113) - 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 (p. 6) – Governance requirements are a natural fit for an ORR checklist.
- OPS01-BP04 Evaluate compliance requirements (p. 8) – Compliance requirements are sometimes included in an ORR checklist. Other times they are a separate process.
- OPS03-BP07 Resource teams appropriately (p. 34) – Team capability is a good candidate for an ORR requirement.
- OPS06-BP01 Plan for unsuccessful changes (p. 59) – A rollback or rollforward plan must be established before you launch your workload.
- OPS07-BP01 Ensure personnel capability (p. 66) – 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 (p. 23): Runbooks should have an owner in charge of maintaining them.
- OPS07-BP04 Use playbooks to investigate issues (p. 72): 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 (p. 97): Runbooks are a part of a good event, incident, and problem management practice.
- OPS10-BP02 Have a process per alert (p. 100): Runbooks and playbooks should be used to respond to alerts. Over time these reactions should be automated.
- OPS11-BP04 Perform knowledge management (p. 113): 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 (p. 23): Playbooks should have an owner in charge of maintaining them.
- OPS07-BP03 Use runbooks to perform procedures (p. 69): 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 (p. 97): Playbooks are a part of good event, incident, and problem management practice.
- OPS10-BP02 Have a process per alert (p. 100): Runbooks and playbooks should be used to respond to alerts. Over time, these reactions should be automated.
- OPS11-BP04 Perform knowledge management (p. 113): 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 (p. 6) - Governance requirements are a key factor in determining whether to deploy a change.
- OPS06-BP01 Plan for unsuccessful changes (p. 59) - Establish plans to mitigate a failed deployment and use pre-mortems to validate them.
- OPS06-BP02 Test and validate changes (p. 60) - Every software change should be properly tested before deployment in order to reduce defects in production.
- OPS07-BP01 Ensure personnel capability (p. 66) - 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 (p. 23)

Related documents:

- AWS Support Plans

Related services:

- AWS Business Support
- AWS Enterprise Support

Operate

Success is the achievement of business outcomes as measured by the metrics you define. By understanding the health of your workload and operations, you can identify when organizational and business outcomes may become at risk, or are at risk, and respond appropriately.

To be successful, you must be able to:

Topics

Understanding workload health

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

Your team should be able to understand the health of your workload easily. You will want to use metrics based on workload outcomes to gain useful insights. You should use these metrics to implement dashboards with business and technical viewpoints that will help team members make informed decisions.

AWS makes it easy to bring together and analyze your workload logs so that you can generate metrics, understand the health of your workload, and gain insight from operations over time.

Best practices

OPS08-BP01 Identify key performance indicators

Identify key performance indicators (KPIs) based on desired business outcomes (for example, order rate, customer retention rate, and profit versus operating expense) and customer outcomes (for example, customer satisfaction). Evaluate KPIs to determine workload success.

Common anti-patterns:

- You are asked by business leadership how successful a workload has been serving business needs but have no frame of reference to determine success.
- You are unable to determine if the commercial off-the-shelf application you operate for your organization is cost-effective.

Benefits of establishing this best practice: By identifying key performance indicators you help achieve business outcomes as the test of the health and success of your workload.

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

Implementation guidance

- Identify key performance indicators: Identify key performance indicators (KPIs) based on desired business and customer outcomes. Evaluate KPIs to determine workload success.

OPS08-BP02 Define workload metrics

Define metrics that measure the health of the workload. Workload health is measured by the achievement of business outcomes (KPIs) and the state of workload components and applications. Examples of KPIs are abandoned shopping carts, orders placed, cost, price, and allocated workload expense. While you may collect telemetry from multiple components, select a subset that provides insight into the overall workload health. Adjust workload metrics over time as business needs change.

Desired outcome:

- You have identified metrics that validate the achievement of KPIs that reflect business outcomes.
- You have metrics that show a consistent view of workload health.
- Workload metrics are evaluated periodically as business needs change.

Common anti-patterns:

- You are monitoring all the applications in your workload but are unable to determine if your workload is achieving business outcomes.
- You have defined workload metrics but they are not associated to any business KPIs.

Benefits of establishing this best practice:

- You can measure your workload against the achievement of business outcomes.
- You know if your workload is in a healthy state or needs intervention.

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

Implementation guidance

The goal of this best practice is that you can answer the following question: is my workload healthy? Workload health is determined by the achievement of business outcomes and the state of applications and components in the workload. Work backwards from business KPIs to identify metrics. Identify key metrics from components and applications. Periodically review workload metrics as business needs change.

Customer example

Workload health is determined at AnyCompany Retail by a collection of application and component metrics. Starting with business KPIs, they identify metrics like order rate that can show they are achieving business outcomes. They also include key application metrics like page response and component metrics like open database connections. On a quarterly basis, they re-evaluate workload metrics to make sure they are still valid in determining workload health.

Implementation steps

1. Starting with business KPIs, identify metrics that show you are achieving business outcomes. If there are KPIs that do not have metrics, instrument your workload with additional metrics for any missing business KPIs.
   a. You can publish custom metrics from your applications to Amazon CloudWatch.
   b. The AWS Distro for OpenTelemetry can collect metrics from existing applications and be used to add new metrics.
   c. Customers with Enterprise Support can request the Building a Monitoring Strategy Workshop from their Technical Account Manager. This workshop will help you build an observability strategy for your workload.
2. Identify metrics for applications and components in the workload. What are key metrics that show the health of individual components and applications? Applications and components may emit many different metrics, but choose one to three key metrics that show their overall health.
3. Implement a mechanism to evaluate workload metrics periodically. When business KPIs change, work with stakeholders to update workload metrics. As your workload components and applications evolve, adjust your workload metrics.

Level of effort for the implementation plan: Medium. Adding metrics for business KPIs to applications may require moderate effort.

Resources

Related best practices:

- OPS04-BP01 Implement application telemetry (p. 36) - Your application must emit telemetry that supports business outcomes.
- OPS04-BP02 Implement and configure workload telemetry (p. 39) - You must instrument your workload to emit telemetry before you can define workload metrics that support business outcomes.
- OPS08-BP01 Identify key performance indicators (p. 79) - You must identify key performance indicators first before selecting workload metrics.

Related documents:

- Adding metrics and traces to your application on Amazon EKS with AWS Distro for OpenTelemetry, AWS X-Ray, and Amazon CloudWatch
- Instrumenting distributed systems for operational visibility
- Implementing health checks
- How to Monitor your Applications Effectively
- How to better monitor your custom application metrics using Amazon CloudWatch Agent

Related videos:

- AWS re:Invent 2020: Monitoring production services at Amazon
- AWS re:Invent 2022 - Building observable applications with OpenTelemetry (BOA310)
- How to Easily Setup Application Monitoring for Your AWS Workloads - AWS Online Tech Talks
- Mastering Observability of Your Serverless Applications - AWS Online Tech Talks

Related examples:

- One Observability Workshop

Related services:

- Amazon CloudWatch
- AWS Distro for OpenTelemetry

OPS08-BP03 Collect and analyze workload metrics

Perform regular, proactive reviews of workload metrics to identify trends and determine if a response is necessary and validate the achievement of business outcomes. Aggregate metrics from your workload applications and components to a central location. Use dashboards and analytics tools to analyze telemetry and determine workload health. Implement a mechanism to conduct workload health reviews on periodic basis with stakeholders in your organization.

Desired outcome:

- Workload metrics are collected in a central location.
- Dashboards and analytics tools are used to analyze workload health trends.
- You conduct periodic workload metric reviews with your organization.

Common anti-patterns:

- Your organization collects metrics from the workload in two different observability platforms. You are unable to determine workload health because the platforms are incompatible.
- Error rates for a component of your workload are slowly increasing. You fail to notice this trend because your organization does not conduct periodic workload metric reviews. The component fails after a week, impairing your workload.

Benefits of establishing this best practice:

- You have increased awareness of workload health and the achievement of business outcomes.
- Workload health trends can be developed over time.

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

Implementation guidance

Collect workload metrics in a central location. Using dashboards and analytics tools, analyze workload metrics to gain insight into workload health, develop workload health trends, and validate the achievement of business outcomes. Implement a mechanism to conduct periodic reviews of workload metrics.

Customer example

AnyCompany Retail conducts workload metric reviews every week on Wednesday. They gather stakeholders from across the company and go through the previous week's metrics. During the meeting, they highlight trends and insights gleaned from analytics tools. Internal dashboards are published with key workload metrics that any employee can view and search.

Implementation steps

1. Identify the workload metrics that are tied to workload health. Starting with business KPIs, identify the metrics for applications, components, and platforms that provide an overall view of workload health.

   a. You can publish custom metrics to Amazon CloudWatch. You can leverage the Amazon CloudWatch agent to collect metrics and logs from Amazon EC2 instances and on-premises servers.
   b. The AWS Distro for OpenTelemetry can collect metrics from existing applications and be used to add new metrics.
   c. Customers with Enterprise Support can request the Building a Monitoring Strategy Workshop from their Technical Account Manager. This workshop helps you build an observability strategy for your workload.
2. Collect workload metrics in a central platform. If workload metrics are split between different platform, this can make it difficult to analyze and develop trends. The platform should have dashboards and analytic capabilities.
   a. Amazon CloudWatch can collect and store workload metrics. In multi-account topologies, it is recommended to have a central logging and monitoring account, referred to as a *log archive account*.
3. Build a consolidated dashboard of workload metrics. Use this view for metrics reviews and analysis of trends.
   a. You can create custom CloudWatch dashboards to collect your workload metrics in a consolidated view.
4. Implement a workload metric review process. On a weekly, bi-weekly, or monthly basis, review your workload metrics with stakeholders, including technical and non-technical personnel. Use these review sessions to identify trends and gain insight into workload health.

Level of effort for the implementation plan: High. If workload metrics are not centrally collected, it could require significant investment to consolidate them in one platform.

Resources

Related best practices:

- OPS08-BP01 Identify key performance indicators (p. 79) - You must identify key performance indicators first before selecting workload metrics.
- OPS08-BP02 Define workload metrics (p. 80) - You must define workload metrics before collecting and analyzing them.

Related documents:

- Power operational insights with Amazon QuickSight
- Using Amazon CloudWatch dashboards custom widgets

Related videos:

- Create Cross Account & Cross Region CloudWatch Dashboards
- Monitor AWS Resources Using Amazon CloudWatch Dashboards

Related examples:

- AWS Management and Governance Tools Workshop - CloudWatch Dashboards
- Well-Architected Labs - Level 100: Monitoring with CloudWatch Dashboards

Related services:

- Amazon CloudWatch

- AWS Distro for OpenTelemetry

OPS08-BP04 Establish workload metrics baselines

Establishing a baseline for workload metrics aids in understanding workload health and performance. Using baselines, you can identify under- and over-performing applications and components. A workload baseline adds to your ability to mitigate issues before they become incidents. Baselines are foundational in developing patterns of activity and implementing anomaly detection when metrics deviate from expected values.

Desired outcome:

- You have a baseline level of metrics for your workload under normal conditions.
- You can determine if your workload is functioning normally.

Common anti-patterns:

- After deploying a new feature, there is drop in request latency. A baseline was not established for a composite metric of incoming processed requests and overall latency. You are unable to determine if the change caused an improvement or caused a defect.
- A sudden spike in user activity occurs, but you have not established a metric baseline. The activity spike slowly leads to a memory leak in an application. Eventually this takes your workload offline.

Benefits of establishing this best practice:

- You understand the normal pattern of activity for your workload using metrics for key components and applications.
- You can determine if your workload, its applications, and components, are behaving normally or may require intervention.

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

Implementation guidance

Use historical data to establish a baseline of workload metrics for applications and components in your workload. Leverage the metric baseline in metric review meetings and troubleshooting. Periodically review workload performance and adjust the baseline as the architecture evolves.

Customer example

Baselines are established for all components and applications at AnyCompany Retail. Using historical data, AnyCompany Retail developed their workload metric baselines over a two-month metric window. Every two months they re-assess baselines and adjust them based on real-world data.

Implementation steps

1. Working backwards from your workload metrics, establish a metric baseline for key components and applications using historical data. Limit the number of metrics per component or application and avoid monitor fatigue.
   a. You can use Amazon CloudWatch Metrics Insights to query metrics at scale and identify trends and patterns.
   b. Amazon CloudWatch anomaly detection uses machine learning algorithms to identify patterns of behavior for metrics, determine baselines, and surfaces anomalies.

  c. Amazon DevOps Guru provides the ability to detect operational issues with your workload using machine learning.
  d. Customers with Enterprise Support can request the Building a Monitoring Strategy Workshop from their Technical Account Manager. This workshop will help you build an observability strategy for your workload.

2. Put in place a mechanism to periodically review workload metric baselines, especially before significant business events. At least once a quarter, evaluate your workload metric baseline using historical data. Use the baseline in your metric review meetings.

Level of effort for the implementation plan: Low. Having established workload metrics, establishing baselines may require you to collect enough data to identify normal patterns of behavior.

Resources

Related best practices:

- OPS08-BP02 Define workload metrics (p. 80) - Workload metrics must be established first before determining baselines.
- OPS08-BP03 Collect and analyze workload metrics (p. 82) - Collecting and analyzing workload metrics is necessary to have in place before establishing metric baselines.
- OPS08-BP05 Learn expected patterns of activity for workload (p. 86) - This best practice builds on top of the baseline to develop usage trends.
- OPS08-BP06 Alert when workload outcomes are at risk (p. 86) - Metric baselines are necessary to identifying thresholds and developing alerts.
- OPS08-BP07 Alert when workload anomalies are detected (p. 87) - Anomaly detection requires the establishment of metric baselines.

Related documents:

- AWS Observability Best Practices - Alarms
- How to Monitor your Applications Effectively
- How to set up CloudWatch Anomaly Detection to set dynamic alarms, automate actions, and drive online sales
- Operationalizing CloudWatch Anomaly Detection

Related videos:

- AWS re:Invent 2020: Monitoring production services at Amazon
- AWS re:Invent 2021- Get insights from operational metrics at scale with CloudWatch Metrics Insights
- AWS re:Invent 2022 - Developing an observability strategy (COP302)
- AWS Summit DC 2022 - Monitoring and observability for modern applications
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS (COP310)

Related examples:

- AWS CloudTrail and Amazon CloudWatch Integration Workshop

Related services:

- Amazon CloudWatch

- Amazon DevOps Guru

OPS08-BP05 Learn expected patterns of activity for workload

Establish patterns of workload activity to identify anomalous behavior so that you can respond appropriately if required.

CloudWatch through the CloudWatch Anomaly Detection feature applies statistical and machine learning algorithms to generate a range of expected values that represent normal metric behavior.

Amazon DevOps Guru can be used to identify anomalous behavior through event correlation, log analysis, and applying machine learning to analyze your workload telemetry. When unexpected behaviors are detected, it provides the related metrics and events with recommendations to address the behavior.

Common anti-patterns:

- You are reviewing network utilization logs and see that network utilization increased between 11:30am and 1:30pm and then again at 4:30pm through 6:00pm. You are unaware if this should be considered normal or not.
- Your web servers reboot every night at 3:00am. You are unaware if this is an expected behavior.

Benefits of establishing this best practice: By learning patterns of behavior you can recognize unexpected behavior and take action if necessary.

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

Implementation guidance

- Learn expected patterns of activity for workload: Establish patterns of workload activity to determine when behavior is outside of the expected values so that you can respond appropriately if required.

Resources

Related documents:

- Amazon DevOps Guru
- CloudWatch Anomaly Detection

OPS08-BP06 Alert when workload outcomes are at risk

Raise an alert when workload outcomes are at risk so that you can respond appropriately if necessary.

Ideally, you have previously identified a metric threshold that you are able to alarm upon or an event that you can use to initiate an automated response.

On AWS, you can use Amazon CloudWatch Synthetics to create canary scripts to monitor your endpoints and APIs by performing the same actions as your customers. The telemetry generated and the insight gained can help you to identify issues before your customers are impacted.

You can also use CloudWatch Logs Insights to interactively search and analyze your log data using a purpose-built query language. CloudWatch Logs Insights automatically discovers fields in logs from AWS services, and custom log events in JSON. It scales with your log volume and query complexity and gives you answers in seconds, helping you to search for the contributing factors of an incident.

Common anti-patterns:

- You have no network connectivity. No one is aware. No one is trying to identify why or taking action to restore connectivity.
- Following a patch, your persistent instances have become unavailable, disrupting users. Your users have opened support cases. No one has been notified. No one is taking action.

Benefits of establishing this best practice: By identifying that business outcomes are at risk and alerting for action to be taken you have the opportunity to prevent or mitigate the impact of an incident.

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

Implementation guidance

- Alert when workload outcomes are at risk: Raise an alert when workload outcomes are at risk so that you can respond appropriately if required.
  - What is Amazon CloudWatch Events?
  - Creating Amazon CloudWatch Alarms
  - Invoking Lambda functions using Amazon SNS notifications

Resources

Related documents:

- Amazon CloudWatch Synthetics
- CloudWatch Logs Insights
- Creating Amazon CloudWatch Alarms
- Invoking Lambda functions using Amazon SNS notifications
- What is Amazon CloudWatch Events?

OPS08-BP07 Alert when workload anomalies are detected

Raise an alert when workload anomalies are detected so that you can respond appropriately if necessary.

Your analysis of your workload metrics over time may establish patterns of behavior that you can quantify sufficiently to define an event or raise an alarm in response.

Once trained, the CloudWatch Anomaly Detection feature can be used to alarm on detected anomalies or can provide overlaid expected values onto a graph of metric data for ongoing comparison.

Common anti-patterns:

- Your retail website sales have increased suddenly and dramatically. No one is aware. No one is trying to identify what led to this surge. No one is taking action to ensure quality customer experiences under the additional load.

- Following the application of a patch, your persistent servers are rebooting frequently, disrupting users. Your servers typically reboot up to three times but not more. No one is aware. No one is trying to identify why this is happening.

Benefits of establishing this best practice: By understanding patterns of workload behavior, you can identify unexpected behavior and take action if necessary.

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

Implementation guidance

- Alert when workload anomalies are detected: Raise an alert when workload anomalies are detected so that you can respond appropriately if required.
  - What is Amazon CloudWatch Events?
  - Creating Amazon CloudWatch Alarms
  - Invoking Lambda functions using Amazon SNS notifications

Resources

Related documents:

- Creating Amazon CloudWatch Alarms
- CloudWatch Anomaly Detection
- Invoking Lambda functions using Amazon SNS notifications
- What is Amazon CloudWatch Events?

OPS08-BP08 Validate the achievement of outcomes and the effectiveness of KPIs and metrics

Create a business-level view of your workload operations to help you determine if you are satisfying needs and to identify areas that need improvement to reach business goals. Validate the effectiveness of KPIs and metrics and revise them if necessary.

AWS also has support for third-party log analysis systems and business intelligence tools through the AWS service APIs and SDKs (for example, Grafana, Kibana, and Logstash).

Common anti-patterns:

- Page response time has never been considered a contributor to customer satisfaction. You have never established a metric or threshold for page response time. Your customers are complaining about slowness.
- You have not been achieving your minimum response time goals. In an effort to improve response time, you have scaled up your application servers. You are now exceeding response time goals by a significant margin and also have significant unused capacity you are paying for.

Benefits of establishing this best practice: By reviewing and revising KPIs and metrics, you understand how your workload supports the achievement of your business outcomes and can identify where improvement is needed to reach business goals.

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

Implementation guidance

- Validate the achievement of outcomes and the effectiveness of KPIs and metrics: Create a business level view of your workload operations to help you determine if you are satisfying needs and to identify areas that need improvement to reach business goals. Validate the effectiveness of KPIs and metrics and revise them if necessary.
  - Using Amazon CloudWatch dashboards
  - What is log analytics?

Resources

Related documents:

- Using Amazon CloudWatch dashboards
- What is log analytics?

Understanding operational health

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

Your team should be able to understand the health of your operations easily. You will want to use metrics based on operations outcomes to gain useful insights. You should use these metrics to implement dashboards with business and technical viewpoints that will help team members make informed decisions.

AWS makes it easier to bring together and analyze your operations logs so that you can generate metrics, know the status of your operations, and gain insight from operations over time.

Best practices

- OPS09-BP01 Identify key performance indicators (p. 89)
- OPS09-BP02 Define operations metrics (p. 90)
- OPS09-BP03 Collect and analyze operations metrics (p. 91)
- OPS09-BP04 Establish operations metrics baselines (p. 92)
- OPS09-BP05 Learn the expected patterns of activity for operations (p. 92)
- OPS09-BP06 Alert when operations outcomes are at risk (p. 93)
- OPS09-BP07 Alert when operations anomalies are detected (p. 95)
- OPS09-BP08 Validate the achievement of outcomes and the effectiveness of KPIs and metrics (p. 96)

OPS09-BP01 Identify key performance indicators

Identify key performance indicators (KPIs) based on desired business outcomes (for example, new features delivered) and customer outcomes (for example, customer support cases). Evaluate KPIs to determine operations success.

Common anti-patterns:

- You are asked by business leadership how successful operations is at accomplishing business goals but have no frame of reference to determine success.
- You are unable to determine if your maintenance windows have an impact on business outcomes.

Benefits of establishing this best practice: By identifying key performance indicators you help achieve business outcomes as the test of the health and success of your operations.

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

Implementation guidance

- Identify key performance indicators: Identify key performance indicators (KPIs) based on desired business and customer outcomes. Evaluate KPIs to determine operations success.

OPS09-BP02 Define operations metrics

Define operations metrics to measure the achievement of KPIs (for example, successful deployments, and failed deployments). Define operations metrics to measure the health of operations activities (for example, mean time to detect an incident (MTTD), and mean time to recovery (MTTR) from an incident). Evaluate metrics to determine if operations are achieving desired outcomes, and to understand the health of your operations activities.

Common anti-patterns:

- Your operations metrics are based on what the team thinks is reasonable.
- You have errors in your metrics calculations that will yield incorrect results.
- You don't have any metrics defined for your operations activities.

Benefits of establishing this best practice: By defining and evaluating operations metrics you can determine the health of your operations activities and measure the achievement of business outcomes.

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

Implementation guidance

- Define operations metrics: Define operations metrics to measure the achievement of KPIs. Define operations metrics to measure the health of operations and its activities. Evaluate metrics to determine if operations are achieving desired outcomes, and to understand the health of the operations.
  - Publish custom metrics
  - Searching and filtering log data
  - Amazon CloudWatch metrics and dimensions reference

Resources

Related documents:

- AWS Answers: Centralized Logging
- Amazon CloudWatch metrics and dimensions reference
- Detect and React to Changes in Pipeline State with Amazon CloudWatch Events
- Publish custom metrics
- Searching and filtering log data

Related videos:

- Build a Monitoring Plan

OPS09-BP03 Collect and analyze operations metrics

Perform regular, proactive reviews of metrics to identify trends and determine where appropriate responses are needed.

You should aggregate log data from the processing of your operations activities and operations API calls, into a service such as CloudWatch Logs. Generate metrics from observations of necessary log content to gain insight into the performance of operations activities.

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

Common anti-patterns:

- Consistent delivery of new features is considered a key performance indicator. You have no method to measure how frequently deployments occur.
- You log deployments, rolled back deployments, patches, and rolled back patches to track you operations activities, but no one reviews the metrics.
- You have a recovery time objective to restore a lost database within fifteen minutes that was defined when the system was deployed and had no users. You now have ten thousand users and have been operating for two years. A recent restore took over two hours. This was not recorded and no one is aware.

Benefits of establishing this best practice: By collecting and analyzing your operations metrics, you gain understanding of the health of your operations and can gain insight to trends that have may an impact on your operations or the achievement of your business outcomes.

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

Implementation guidance

- Collect and analyze operations metrics: Perform regular proactive reviews of metrics to identify trends and determine where appropriate responses are needed.
  - Using Amazon CloudWatch metrics
  - Amazon CloudWatch metrics and dimensions reference
  - Collect metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch Agent

Resources

Related documents:

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

OPS09-BP04 Establish operations metrics baselines

Establish baselines for metrics to provide expected values as the basis for comparison and identification of under and over performing operations activities.

Common anti-patterns:

- You have been asked what the expected time to deploy is. You have not measured how long it takes to deploy and can not determine expected times.
- You have been asked what how long it takes to recover from an issue with the application servers. You have no information about time to recovery from first customer contact. You have no information about time to recovery from first identification of an issue through monitoring.
- You have been asked how many support personnel are required over the weekend. You have no idea how many support cases are typical over a weekend and can not provide an estimate.
- You have a recovery time objective to restore lost databases within fifteen minutes that was defined when the system was deployed and had no users. You now have ten thousand users and have been operating for two years. You have no information on how the time to restore has changed for your database.

Benefits of establishing this best practice: By defining baseline metric values you are able to evaluate current metric values, and metric trends, to determine if action is required.

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

Implementation guidance

- Learn expected patterns of activity for operations: Establish patterns of operations activity to determine when behavior is outside of the expected values so that you can respond appropriately if required.

OPS09-BP05 Learn the expected patterns of activity for operations

Establish patterns of operations activities to identify anomalous activity so that you can respond appropriately if necessary.

Common anti-patterns:

- Your deployment failure rate has increased substantially recently. You address each of the failures independently. You do not realize that the failures correspond to deployments by a new employee who is unfamiliar with the deployment management system.

Benefits of establishing this best practice: By learning patterns of behavior, you can recognize unexpected behavior and take action if necessary.

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

Implementation guidance

- Learn expected patterns of activity for operations: Establish patterns of operations activity to determine when behavior is outside of the expected values so that you can respond appropriately if required.

OPS09-BP06 Alert when operations outcomes are at risk

Whenever operations outcomes are at risk, an alert must be raised and acted upon. Operations outcomes are any activity that supports a workload in production. This includes everything from deploying new versions of applications to recovering from an outage. Operations outcomes must be treated with the same importance as business outcomes.

Software teams should identify key operations metrics and activities and build alerts for them. Alerts must be timely and actionable. If an alert is raised, a reference to a corresponding runbook or playbook should be included. Alerts without a corresponding action can lead to alert fatigue.

Desired outcome: When operations activities are at risk, alerts are sent to drive action. The alerts contain context on why an alert is being raised and point to a playbook to investigate or a runbook to mitigate. Where possible, runbooks are automated and notifications are sent.

Common anti-patterns:

- You are investigating an incident and support cases are being filed. The support cases are breaching the service level agreement (SLA) but no alerts are being raised.
- A deployment to production scheduled for midnight is delayed due to last-minute code changes. No alert is raised and the deployment hangs.
- A production outage occurs but no alerts are sent.
- Your deployment time consistently runs behind estimates. No action is taken to investigate.

Benefits of establishing this best practice:

- Alerting when operations outcomes are at risk boosts your ability to support your workload by staying ahead of issues.
- Business outcomes are improved due to healthy operations outcomes.
- Detection and remediation of operations issues are improved.
- Overall operational health is increased.

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

Implementation guidance

Operations outcomes must be defined before you can alert on them. Start by defining what operations activities are most important to your organization. Is it deploying to production in under two hours or responding to a support case within a set amount of time? Your organization must define key operations activities and how they are measured so that they can be monitored, improved, and alerted on. You need a central location where workload and operations telemetry is stored and analyzed. The same mechanism should be able to raise an alert when an operations outcome is at risk.

Customer example

A CloudWatch alarm was initiated during a routine deployment at AnyCompany Retail. The lead time for deployment was breached. Amazon EventBridge created an OpsItem in AWS Systems Manager OpsCenter. The Cloud Operations team used a playbook to investigate the issue and identified that a schema change was taking longer than expected. They alerted the on-call developer and continued monitoring the deployment. Once the deployment was complete, the Cloud Operations team resolved the OpsItem. The team will analyze the incident during a postmortem.

Implementation steps

1. If you have not identified operations KPIs, metrics, and activities, work on implementing the preceding best practices to this question (OPS09-BP01 to OPS09-BP05).
   - AWS Support customers with Enterprise Support can request the Operations KPI Workshop from their Technical Account Manager. This collaborative workshop helps you define operations KPIs and metrics aligned to business goals, provided at no additional cost. Contact your Technical Account Manager to learn more.
2. Once you have operations activities, KPIs, and metrics established, configure alerts in your observability platform. Alerts should have an action associated to them, like a playbook or runbook. Alerts without an action should be avoided.
3. Over time, you should evaluate your operations metrics, KPIs, and activities to identify areas of improvement. Capture feedback in runbooks and playbooks from operators to identify areas for improvement in responding to alerts.
4. Alerts should include a mechanism to flag them as a false-positive. This should lead to a review of the metric thresholds.

Level of effort for the implementation plan: Medium. There are several best practices that must be in place before implementing this best practice. Once operations activities have been identified and operations KPIs established, alerts should be established.

Resources

Related best practices:

- OPS02-BP03 Operations activities have identified owners responsible for their performance (p. 23): Every operation activity and outcome should have an identified owner that's responsible. This is who should be alerted when outcomes are at risk.
- OPS03-BP02 Team members are empowered to take action when outcomes are at risk (p. 28): When alerts are raised, your team should have agency to act to remedy the issue.
- OPS09-BP01 Identify key performance indicators (p. 89): Alerting on operations outcomes starts with identify operations KPIs.
- OPS09-BP02 Define operations metrics (p. 90): Establish this best practice before you start generating alerts.
- OPS09-BP03 Collect and analyze operations metrics (p. 91): Centrally collecting operations metrics is required to build alerts.
- OPS09-BP04 Establish operations metrics baselines (p. 92): Operations metrics baselines provide the ability to tune alerts and avoid alert fatigue.
- OPS09-BP05 Learn the expected patterns of activity for operations (p. 92): You can improve the accuracy of your alerts by understanding the activity patterns for operations events.
- OPS09-BP08 Validate the achievement of outcomes and the effectiveness of KPIs and metrics (p. 96): Evaluate the achievement of operations outcomes to ensure that your KPIs and metrics are valid.
- OPS10-BP02 Have a process per alert (p. 100): Every alert should have an associated runbook or playbook and provide context for the person being alerted.
- OPS11-BP02 Perform post-incident analysis (p. 110): Conduct a post-incident analysis after the alert to identify areas for improvement.

Related documents:

- AWS Deployment Pipelines Reference Architecture: Application Pipeline Architecture
- GitLab: Getting Started with Agile / DevOps Metrics

Related videos:

- Aggregate and Resolve Operational Issues Using AWS Systems Manager OpsCenter
- Integrate AWS Systems Manager OpsCenter with Amazon CloudWatch Alarms
- Integrate Your Data Sources into AWS Systems Manager OpsCenter Using Amazon EventBridge

Related examples:

- Automate remediation actions for Amazon EC2 notifications and beyond using Amazon EC2 Systems Manager Automation and AWS Health
- AWS Management and Governance Tools Workshop - Operations 2022
- Ingesting, analyzing, and visualizing metrics with DevOps Monitoring Dashboard on AWS

Related services:

- Amazon EventBridge
- AWS Support Proactive Services - Operations KPI Workshop
- AWS Systems Manager OpsCenter
- CloudWatch Events

OPS09-BP07 Alert when operations anomalies are detected

Raise an alert when operations anomalies are detected so that you can respond appropriately if necessary.

Your analysis of your operations metrics over time may established patterns of behavior that you can quantify sufficiently to define an event or raise an alarm in response.

Once trained, the CloudWatch Anomaly Detection feature can be used to alarm on detected anomalies or can provide overlaid expected values onto a graph of metric data for ongoing comparison.

Amazon DevOps Guru can be used to identify anomalous behavior through event correlation, log analysis, and applying machine learning to analyze your workload telemetry. The insights gained are presented with the relevant data and recommendations.

Common anti-patterns:

- You are applying a patch to your fleet of instances. You tested the patch successfully in the test environment. The patch is failing for a large percentage of instances in your fleet. You do nothing.
- You note that there are deployments starting Friday end of day. Your organization has predefined maintenance windows on Tuesdays and Thursdays. You do nothing.

Benefits of establishing this best practice: By understanding patterns of operations behavior you can identify unexpected behavior and take action if necessary.

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

Implementation guidance

- Alert when operations anomalies are detected: Raise an alert when operations anomalies are detected so that you can respond appropriately if required.

  - What is Amazon CloudWatch Events?
  - Creating Amazon CloudWatch alarms
  - Invoking Lambda functions using Amazon SNS notifications

Resources

Related documents:

- Amazon DevOps Guru
- CloudWatch Anomaly Detection
- Creating Amazon CloudWatch alarms
- Detect and React to Changes in Pipeline State with Amazon CloudWatch Events
- Invoking Lambda functions using Amazon SNS notifications
- What is Amazon CloudWatch Events?

OPS09-BP08 Validate the achievement of outcomes and the effectiveness of KPIs and metrics

Create a business-level view of your operations activities to help you determine if you are satisfying needs and to identify areas that need improvement to reach business goals. Validate the effectiveness of KPIs and metrics and revise them if necessary.

AWS also has support for third-party log analysis systems and business intelligence tools through the AWS service APIs and SDKs (for example, Grafana, Kibana, and Logstash).

Common anti-patterns:

- The frequency of your deployments has increased with the growth in number of development teams. Your defined expected number of deployments is once per week. You have been regularly deploying daily. When their is an issue with your deployment system, and deployments are not possible, it goes undetected for days.
- When your business previously provided support only during core business hours from Monday to Friday. You established a next business day response time goal for incidents. You have recently started offering 24x7 support coverage with a two hour response time goal. Your overnight staff are overwhelmed and customers are unhappy. There is no indication that there are issues with incident response times because you are reporting against a next business day target.

Benefits of establishing this best practice: By reviewing and revising KPIs and metrics, you understand how your workload supports the achievement of your business outcomes and can identify where improvement is needed to reach business goals.

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

Implementation guidance

- Validate the achievement of outcomes and the effectiveness of KPIs and metrics: Create a business level view of your operations activities to help you determine if you are satisfying needs and to identify areas that need improvement to reach business goals. Validate the effectiveness of KPIs and metrics and revise them if necessary.
  - Using Amazon CloudWatch dashboards

- What is log analytics?

Resources

Related documents:

- Using Amazon CloudWatch dashboards
- What is log analytics?

Responding to events

You should anticipate operational events, both planned (for example, sales promotions, deployments, and failure tests) and unplanned (for example, surges in utilization and component failures). You should use your existing runbooks and playbooks to deliver consistent results when you respond to alerts. Defined alerts should be owned by a role or a team that is accountable for the response and escalations. You will also want to know the business impact of your system components and use this to target efforts when needed. You should perform a root cause analysis (RCA) after events, and then prevent recurrence of failures or document workarounds.

AWS simplifies your event response by providing tools supporting all aspects of your workload and operations as code. These tools allow you to script responses to operations events and start their initiation in response to monitoring data.

In AWS, you can improve recovery time by replacing failed components with known good versions, rather than trying to repair them. You can then carry out analysis on the failed resource out of band.

Best practices

- OPS10-BP01 Use a process for event, incident, and problem management (p. 97)
- OPS10-BP02 Have a process per alert (p. 100)
- OPS10-BP03 Prioritize operational events based on business impact (p. 101)
- OPS10-BP04 Define escalation paths (p. 102)
- OPS10-BP05 Define a customer communication plan for outages (p. 102)
- OPS10-BP06 Communicate status through dashboards (p. 105)
- OPS10-BP07 Automate responses to events (p. 106)

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

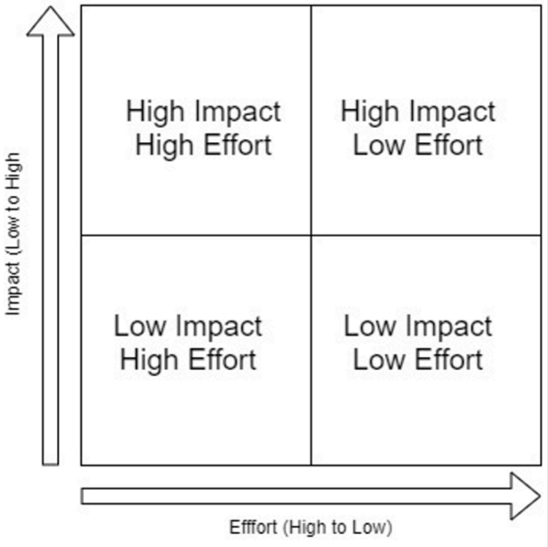
AnyCompany Retail has a portion of their internal wiki devoted to processes for event, incident, and problem management. All events are sent to Amazon EventBridge. Problems are identified as OpsItems in AWS Systems Manager OpsCenter and prioritized to fix, reducing undifferentiated labor. As processes 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 (p. 69): Known problems need an associated runbook so that mitigation efforts are consistent.
- OPS07-BP04 Use playbooks to investigate issues (p. 72): Incidents must be investigated using playbooks.
- OPS11-BP02 Perform post-incident analysis (p. 110): 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 (p. 69) - Your communication plan should have a runbook associated with it so that your personnel know how to respond.
- OPS11-BP02 Perform post-incident analysis (p. 110) - 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

Evolution is the continuous cycle of improvement over time. Implement frequent small incremental changes based on the lessons learned from your operations activities and evaluate their success at bringing about improvement.

To evolve your operations over time, you must be able to:

Topics

Learn, share, and improve

It's essential that you regularly provide time for analysis of operations activities, analysis of failures, experimentation, and making improvements. When things fail, you will want to ensure that your team, as well as your larger engineering community, learns from those failures. You should analyze failures to identify lessons learned and plan improvements. You will want to regularly review your lessons learned with other teams to validate your insights.

Best practices

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 (p. 110) - 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 (p. 117) - 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 (p. 4): Feedback loops are a mechanism to gather external customer needs.
- OPS01-BP02 Evaluate internal customer needs (p. 5): Internal stakeholders can use feedback loops to communicate needs and requirements.
- OPS11-BP02 Perform post-incident analysis (p. 110): Post-incident analyses are an important form of retrospective analysis conducted after incidents.
- OPS11-BP07 Perform operations metrics reviews (p. 116): 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 (p. 117) - 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.

Conclusion

Operational excellence is an ongoing and iterative effort.

Set up your organization for success by having shared goals. Ensure that everyone understands their part in achieving business outcomes and how they impact the ability of others to succeed. Provide support for your team members so that they can support your business outcomes.

Every operational event and failure should be treated as an opportunity to improve the operations of your architecture. By understanding the needs of your workloads, predefining runbooks for routine activities, and playbooks to guide issue resolution, using the operations as code features in AWS, and maintaining situational awareness, your operations will be better prepared and able to respond more effectively when incidents occur.

Through focusing on incremental improvement based on priorities as they change, and lessons learned from event response and retrospective analysis, you will help the success of your business by increasing the efficiency and effectiveness of your activities.

AWS strives to help you build and operate architectures that maximize efficiency while you build highly responsive and adaptive deployments. To increase the operational excellence of your workloads, you should use the best practices discussed in this paper.

Contributors

- Rich Boyd, Operational Excellence Pillar Lead, Well-Architected, Amazon Web Services
- Jon Steele, Solutions Architect Well-Architected, Amazon Web Services
- Ryan King, Sr. Technical Program Manager, Amazon Web Services
- Chris Kunselman, Advisory Consultant, Amazon Web Services
- Peter Mullen, Advisory Consultant, Amazon Web Services
- Brian Quinn, Sr. Advisory Consultant, Amazon Web Services
- David Stanley, Cloud Operating Model Lead, Amazon Web Services

Further reading

For additional guidance, consult the following sources:

- AWS Well-Architected Framework
- AWS Architecture Center

Document revisions

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

Change	Description	Date
Updates for new Framework (p. 123)	Best practices updated with prescriptive guidance and new best practices added.	April 10, 2023
Whitepaper updated (p. 123)	Best practices updated with new implementation guidance.	December 15, 2022
Whitepaper updated (p. 123)	Best practices expanded and improvement plans added.	October 20, 2022
Minor update (p. 123)	Small editorial update.	August 8, 2022
Whitepaper updated (p. 123)	Updates to reflect new AWS services and features, and latest best practices.	February 2, 2022
Minor update (p. 1)	Added Sustainability Pillar to introduction.	December 2, 2021
Updates for new Framework (p. 123)	Updates to reflect new AWS services and features, and latest best practices.	July 8, 2020
Whitepaper updated (p. 123)	Updates to reflect new AWS services and features, and updated references.	July 1, 2018
Initial publication (p. 123)	Operational Excellence Pillar - AWS Well-Architected Framework published.	November 1, 2017

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.