AWS Well-Architected Framework

Operational Excellence Pillar

Operational Excellence Pillar: AWS Well-Architected Framework

Table of Contents

Abstract and introduction 1
Introduction 1
Operational excellence 3
Design principles 3
Definition 4
Organization 6
Organization priorities 9
OPS01-BP01 Evaluate external customer needs 9
OPS01-BP02 Evaluate internal customer needs 10
OPS01-BP03 Evaluate governance requirements 12
OPS01-BP04 Evaluate compliance requirements 14
OPS01-BP05 Evaluate threat landscape 17
OPS01-BP06 Evaluate tradeoffs while managing benefits and risks 19
Operating model 23
Operating model 2 by 2 representations 24
Relationships and ownership 33
Organizational culture 53
OPS03-BP01 Provide executive sponsorship 54
OPS03-BP02 Team members are empowered to take action when outcomes are at risk 57
OPS03-BP03 Escalation is encouraged 60
OPS03-BP04 Communications are timely, clear, and actionable 63
OPS03-BP05 Experimentation is encouraged 68
OPS03-BP06 Team members are encouraged to maintain and grow their skill sets 70
OPS03-BP07 Resource teams appropriately 73
Prepare 77
Implement observability 78
OPS04-BP01 Identify key performance indicators 79
OPS04-BP02 Implement application telemetry 81
OPS04-BP03 Implement user experience telemetry 84
OPS04-BP04 Implement dependency telemetry 87
OPS04-BP05 Implement distributed tracing 90
Design for operations 93
OPS05-BP01 Use version control 93
OPS05-BP02 Test and validate changes 94

OPS05-BP03 Use configuration management systems 98
OPS05-BP04 Use build and deployment management systems 101
OPS05-BP05 Perform patch management 103
OPS05-BP06 Share design standards 106
OPS05-BP07 Implement practices to improve code quality 109
OPS05-BP08 Use multiple environments 112
OPS05-BP09 Make frequent, small, reversible changes 114
OPS05-BP10 Fully automate integration and deployment 115
Mitigate deployment risks 117
OPS06-BP01 Plan for unsuccessful changes 117
OPS06-BP02 Test deployments 120
OPS06-BP03 Employ safe deployment strategies 123
OPS06-BP04 Automate testing and rollback 126
Operational readiness and change management 130
OPS07-BP01 Ensure personnel capability 130
OPS07-BP02: Ensure a consistent review of operational readiness 132
OPS07-BP03 Use runbooks to perform procedures 136
OPS07-BP04 Use playbooks to investigate issues 140
OPS07-BP05 Make informed decisions to deploy systems and changes 144
OPS07-BP06 Create support plans for production workloads 146
Operate 149
Utilizing workload observability 150
OPS08-BP01 Analyze workload metrics 151
OPS08-BP02 Analyze workload logs 153
OPS08-BP03 Analyze workload traces 155
OPS08-BP04 Create actionable alerts 158
OPS08-BP05 Create dashboards 161
Understanding operational health 164
OPS09-BP01 Measure operations goals and KPIs with metrics 164
OPS09-BP02 Communicate status and trends to ensure visibility into operation 167
OPS09-BP03 Review operations metrics and prioritize improvement 169
Responding to events 171
OPS10-BP01 Use a process for event, incident, and problem management 171
OPS10-BP02 Have a process per alert 176
OPS10-BP03 Prioritize operational events based on business impact 180
OPS10-BP04 Define escalation paths 183

OPS10-BP05 Define a customer communication plan for service-impacting events 185
OPS10-BP06 Communicate status through dashboards 188
OPS10-BP07 Automate responses to events 191
Evolve 194
Learn, share, and improve 195
OPS11-BP01 Have a process for continuous improvement 195
OPS11-BP02 Perform post-incident analysis 197
OPS11-BP03 Implement feedback loops 199
OPS11-BP04 Perform knowledge management 203
OPS11-BP05 Define drivers for improvement 205
OPS11-BP06 Validate insights 207
OPS11-BP07 Perform operations metrics reviews 209
OPS11-BP08 Document and share lessons learned 211
OPS11-BP09 Allocate time to make improvements 212
Conclusion 215
Contributors 216
Further reading 217
Document revisions 218
Notices 220
AWS Glossary 221

Operational Excellence Pillar - AWS Well-Architected Framework

Publication date: **November 6, 2024** (*Document revisions*)

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

Operational excellence (OE) is a commitment to build software correctly while consistently delivering a great customer experience. The operational excellence pillar contains best practices for organizing your team, designing your workload, operating it at scale, and evolving it over time.

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:

- **Organize teams around business outcomes:** The ability of a team to achieve business outcomes comes from leadership vision, effective operations, and a business-aligned operating model. Leadership should be fully invested and committed to a CloudOps transformation with a suitable cloud operating model that incentivizes teams to operate in the most efficient way and meet business outcomes. The right operating model uses people, process, and technology capabilities to scale, optimize for productivity, and differentiate through agility, responsiveness, and adaptation. The organization's long-term vision is translated into goals that are communicated across the enterprise to stakeholders and consumers of your cloud services. Goals and operational KPIs are aligned at all levels. This practice sustains the long-term value derived from implementing the following design principles.
- **Implement observability for actionable insights:** Gain a comprehensive understanding of workload behavior, performance, reliability, cost, and health. Establish key performance indicators (KPIs) and leverage observability telemetry to make informed decisions and take prompt action when business outcomes are at risk. Proactively improve performance, reliability, and cost based on actionable observability data.
- **Safely automate where possible:** 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 and its operations (applications, infrastructure, configuration, and procedures) as code, and update it. You can then automate your workload's operations by initiating them in response to events. In the cloud, you can employ automation safety by configuring guardrails, including

rate control, error thresholds, and approvals. Through effective automation, you can achieve consistent responses to events, limit human error, and reduce operator toil.

- **Make frequent, small, reversible changes:** Design workloads that are scalable and loosely coupled to permit components to be updated regularly. Automated deployment techniques together with smaller, incremental changes reduces the blast radius and allows for faster reversal when failures occur. This increases confidence to deliver beneficial changes to your workload while maintaining quality and adapting quickly to changes in market conditions.
- **Refine operations procedures frequently:** As you evolve your workloads, evolve your operations appropriately. As you use operations procedures, look for opportunities to improve them. Hold regular reviews and validate that all procedures are effective and that teams are familiar with them. Where gaps are identified, update procedures accordingly. Communicate procedural updates to all stakeholders and teams. Gamify your operations to share best practices and educate teams.
- **Anticipate failure:** Maximize operational success by driving failure scenarios to understand the workload's risk profile and its impact on your business outcomes. Test the effectiveness of your procedures and your team's response against these simulated failures. Make informed decisions to manage open risks that are identified by your testing.
- **Learn from all operational events and metrics:** Drive improvement through lessons learned from all operational events and failures. Share what is learned across teams and through the entire organization. Learnings should highlight data and anecdotes on how operations contribute to business outcomes.
- **Use managed services:** Reduce operational burden by using AWS managed services where possible. Build operational procedures around interactions with those services.

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

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

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

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

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

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

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

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

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

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

Use tools or services that permit you to centrally govern your environments across accounts, such as AWS Organizations, to help manage your operating models. Services like AWS Control Tower expand this management capability by allowing you to define blueprints (supporting your operating models) for the setup of accounts, apply ongoing governance using AWS Organizations, and automate provisioning of new accounts. Managed Services providers such as AWS Managed Services, AWS Managed Services Partners, or Managed Services Providers in the AWS Partner Network, provide expertise implementing cloud environments, and support your security and compliance requirements and business goals. Adding Managed Services to your operating model can save you time and resources, and lets you keep your internal teams lean and focused on strategic outcomes that will differentiate your business, rather than developing new skills and capabilities.

You might find that you want to emphasize a small subset of your priorities at some point in time. Use a balanced approach over the long term to verify the development of needed capabilities and management of risk. Review your priorities regularly and update them as needs change. When responsibility and ownership are undefined or unknown, you are at risk of both not performing necessary action in a timely fashion and of redundant and potentially conflicting efforts emerging to address those needs. Organizational culture has a direct impact on team member job satisfaction and retention. Activate the engagement and capabilities of your team members to achieve the success of your business. Experimentation is required for innovation to happen and turn ideas into outcomes. Recognize that an undesired result is a successful experiment that has identified a path that will not lead to success.

Topics

- Organization priorities
- Operating model

- Organizational culture

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
- OPS01-BP02 Evaluate internal customer needs
- OPS01-BP03 Evaluate governance requirements
- OPS01-BP04 Evaluate compliance requirements
- OPS01-BP05 Evaluate threat landscape
- OPS01-BP06 Evaluate tradeoffs while managing benefits and risks

OPS01-BP01 Evaluate external customer needs

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

Desired outcome:

- You work backwards from customer outcomes.
- You understand how your operational practices support business outcomes and objectives.
- You engage all relevant parties.
- You have mechanisms to capture external customer needs.

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 a shared understanding: Establish a 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 best practices:

- OPS11-BP03 Implement feedback loops

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.

Desired outcome:

- You 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).
- You 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 informs 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 best practices:

- OPS11-BP03 Implement feedback loops

OPS01-BP03 Evaluate governance requirements

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

1. Identify the stakeholders for your workload, including any centralized teams.
2. Work with stakeholders to identify governance requirements.
3. Once you've generated a list, prioritize the improvement items, and begin implementing them into your workload.
   a. Use services like AWS Config to create governance-as-code and validate that governance requirements are followed.
   b. If you use AWS Organizations, you can leverage Service Control Policies to implement governance requirements.
4. Provide documentation that validates the implementation.

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

Resources

Related best practices:

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

Related documents:

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

Related videos:

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

Related examples:

- AWS Config Conformance Pack Samples

Related services:

- AWS Config
- AWS Organizations - Service Control Policies

OPS01-BP04 Evaluate compliance requirements

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

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

- Evaluating and understanding the compliance requirements that apply to your workload will inform how you prioritize your efforts to deliver business value.
- You choose the right locations and technologies that are congruent with your compliance framework.
- Designing your workload for auditability helps you to prove you are adhering to your compliance framework.

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

Implementation guidance

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

Customer example

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

Implementation steps

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

Desired outcome:

- You regularly review and act on Well-Architected and Trusted Advisor outputs
- You are aware of the latest patch status of your services
- You understand the risk and impact of known threats and act accordingly
- You implement mitigations as necessary
- You communicate actions and context

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: You identify and understand the threats to your organization and workload, which 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 best practice:

- SEC01-BP07 Identify threats and prioritize mitigations using a threat model

Related documents:

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

Related videos:

- AWS re:Inforce 2023 - A tool to help improve your threat modeling

OPS01-BP06 Evaluate tradeoffs while managing benefits and risks

Competing interests from multiple parties can make it challenging to prioritize efforts, build capabilities, and deliver outcomes aligned with business strategies. For example, you may be asked to accelerate speed-to-market for new features over optimizing IT infrastructure costs. This can put two interested parties in conflict with one another. In these situations, decisions need to be brought to a higher authority to resolve conflict. Data is required to remove emotional attachment from the decision-making process.

The same challenge may occur at a tactical level. For example, the choice between using relational or non-relational database technologies can have a significant impact on the operation of an application. It's critical to understand the predictable results of various choices.

AWS can help you educate your teams about AWS and its services to increase their understanding of how their choices can have an impact on your workload. Use the resources provided by Support (AWS Knowledge Center, AWS Discussion Forums, and Support Center) and AWS Documentation to educate your teams. For further questions, reach out to Support.

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

Desired outcome: You have a clearly defined decision-making governance framework to facilitate important decisions at every level within your cloud delivery organization. This framework includes features like a risk register, defined roles that are authorized to make decisions, and a defined models for each level of decision that can be made. This framework defines in advance how conflicts are resolved, what data needs to be presented, and how options are prioritized, so that once decisions are made you can commit without delay. The decision-making framework includes a standardized approach to reviewing and weighing the benefits and risks of every decision to understand the tradeoffs. This may include external factors, such as adherence to regulatory compliance requirements.

Common anti-patterns:

- 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.
- You have decided to include a library 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.
- The original business justification for your migration was based upon the modernization of 60% of your application workloads. However, due to technical difficulties, a decision was made to modernize only 20%, leading to a reduction in planned benefits long-term, increased operator toil for infrastructure teams to manually support legacy systems, and greater reliance on developing new skillsets in your infrastructure teams that were not planning for this change.

Benefits of establishing this best practice: Fully aligning and supporting board-level business priorities, understanding the risks to achieving success, making informed decisions, and acting appropriately when risks impede chances for success. Understanding the implications and consequences of your decisions helps you to prioritize your options and bring leaders into agreement faster, leading to improved business outcomes. Identifying the available benefits of your choices and being aware of the risks to your organization helps you make data-driven decisions, rather than relying on anecdotes.

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

Implementation guidance

Managing benefits and risks should be defined by a governing body that drives the requirements for key decision-making. You want decisions to be made and prioritized based on how they benefit the organization, with an understanding of the risks involved. Accurate information is critical for making the organizational decisions. This should be based on solid measurements and defined by common industry practices of cost benefit analysis. To make these types of decisions, strike a balance between centralized and decentralized authority. There is always a tradeoff, and it's important to understand how each choice impacts defined strategies and desired business outcomes.

Implementation steps

1. Formalize benefits measurement practices within a holistic cloud governance framework.
    a. Balance central control of decision-making with decentralized authority for some decisions.
    b. Understand that burdensome decision-making processes imposed on every decision can slow you down.
    c. Incorporate external factors into your decision making process (like compliance requirements).
2. Establish an agreed-upon decision-making framework for various levels of decisions, which includes who is required to unblock decisions that are subject to conflicted interests.
    a. Centralize one-way door decisions that could be irreversible.
    b. Allow two-way door decisions to be made by lower level organizational leaders.
3. Understand and manage benefits and risks. Balance the benefits of decisions against the risks involved.
    a. **Identify benefits**: Identify benefits based on business goals, needs, and priorities. Examples include business case impact, time-to-market, security, reliability, performance, and cost.

   b. **Identify risks**: Identify risks based on business goals, needs, and priorities. Examples include time-to-market, security, reliability, performance, and cost.
   c. **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.
4. Programatically enforce key decisions that automate your adherence to compliance requirements.
5. Leverage known industry frameworks and capabilities, such as Value Stream Analysis and LEAN, to baseline current state performance, business metrics, and define iterations of progress towards improvements to these metrics.

Level of effort for the implementation plan: Medium-High

Resources

Related best practices:

- OPS01-BP05 Evaluate threat landscape

Related documents:

- Elements of Amazon's Day 1 Culture | Make high quality, high velocity decisions
- Cloud Governance
- Management & Governance Cloud Environment
- Governance in the Cloud and in the Digital Age: Parts One & Two

Related videos:

- Podcast | Jeff Bezos | On how to make decisions

Related examples:

- Make informed decisions using data (The DevOps Sagas)

- Using development value stream mapping to identify constraints to DevOps outcomes

Operating model

In this section, we provide a way to understand the operating model you work within, how that model can be visualized, and how, at a team level, you should evolve to extract maximum value from your investment in cloud services. By doing so, you can enhance your operational practices, build agile teams and workloads, and positively contribute to business outcomes.

It is common for your team to exist within multiple organizational layers, and those layers have existing ways of working. Participating with your team in achieving business outcomes means understanding where your teams are in those layers, the position of the teams you interact with, and how they work. Furthermore, teams need to understand their roles in the success of other teams, know the role of other teams in their success, and have shared goals.

These layers make up the overall operating model of the organization. How the organization functions to deliver business outcomes depends upon many factors, such as type, industry, geography, size, and level of autonomy. However, it likely falls into three broad categories:

- Centralized
- Decentralized
- Federated

These organization-level topologies are described in Organize for success.

Your team and workload exist within your organization's operating model. However, it is unreasonable to expect a single operating model to be able to support all teams and their workloads. Therefore, your team also needs its own operating model. This way of working is shaped by your organization, your department, the makeup of your team, and the characteristics of the workload itself.

Most organizations that move to the cloud do so as part of an enterprise transformation program that seeks to unlock new ways of working (the operating model) to support long-term strategic aims. This journey is not a point in time exercise, but a process that requires continual evolution and incremental progress towards the strategic goal. This allows workloads owners to adapt to the evolving operating model with minimal disruption.

Amazon is often used as an example of how a large organization is able to innovate at scale by empowering teams to stay close to customers, rapidly launch innovative products and services, and take advantage of technical architectures that support speed and agility. This required us to restructure how our teams are organized, now known as *two-pizza teams*. A two-pizza team has all the right resources embedded within it (engineering, testing, product and program management, and operations) to own and run a workload end-to-end.

We advise working towards this operating model as a proven way for workload teams to move quickly and contribute to overall business outcomes in the way that best serves their customers.

Organizations seeking to emulate this success may need to adapt their operating model throughout their transformation journey. At both organization and team level, this requires consideration, planning, and communication. The following section provides a way to visualize these team-level operating models and how they evolve to *you build it, you run it*.

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.

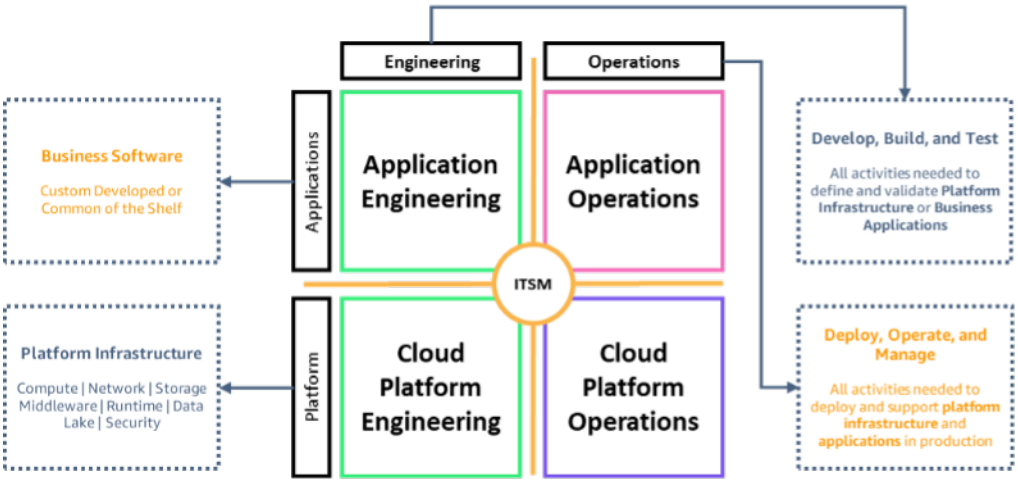
Topics

- Fully separated operating model
- DevOps with cloud-managed service provider
- Cloud operations and platform enablement (COPE)
- Distributed DevOps
- Decentralized DevOps
- Evolving your operating model

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

Historically, organizations embraced frameworks such as ITIL or standards like ISO and shaped their operational activties around them, which often resulted in a fully-separated topology. In this model, activities in each quadrant are performed by a separate team. Work is passed between teams through mechanisms such as work requests, 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.

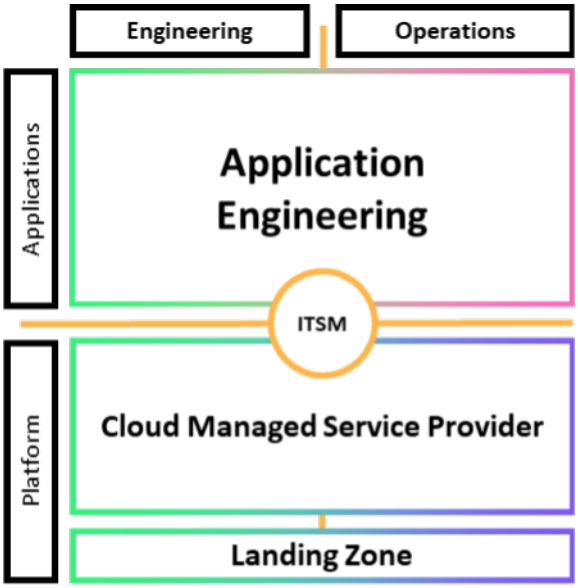
There is a higher risk of misalignment when business, development, and operations teams are organized around the activities or functions that are being performed. This can lead to teams focusing on their specific responsibilities instead of focusing on achieving business outcomes. Teams may be narrowly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

DevOps with cloud-managed service provider

The DevOps with cloud-managed service provider model follows a *you build it, you run it* methodology for application teams. However, your organization may not have the existing skills or team members to support a dedicated platform engineering and operations team, or you may not be in a position to make the time and effort investments to do so.

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

Managed services providers such as AWS Managed Services or providers in the AWS Partner Network provide expertise implementing cloud environments, and support your security and compliance requirements and business goals.



DevOps with cloud managed service provider

For this variation, we 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 requires 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 keeps your internal teams lean and focused on strategic outcomes that differentiate your business, rather than developing new skills and capabilities. It can also provide time for you to build and mature your own platform capabilities without slowing down your cloud migration programs.

Cloud operations and platform enablement (COPE)

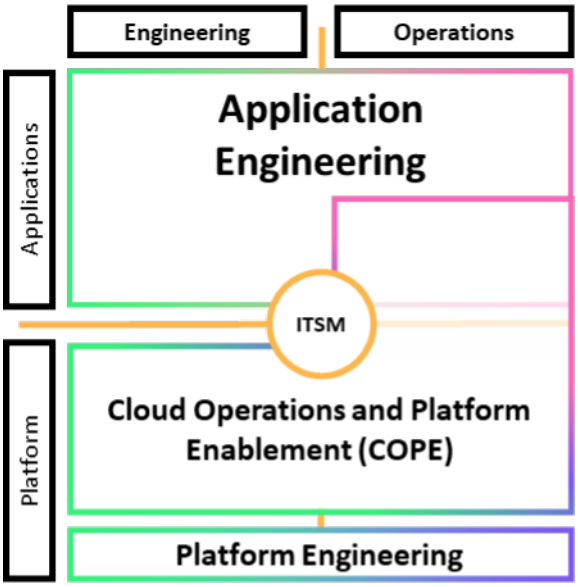
This cloud operations and platform enablement (COPE) model seeks to establish a *you build it, you run it* methodology by supporting application teams to perform the engineering and operations activities for their workloads, adopting a DevOps culture.

Your application teams may be tasked with migrating, adopting the cloud, or modernizing your workloads, but might not have the existing skills to adequately support cloud architecture and operations. This lack of application team capabilities and familiarity is likely to slow down your organization's agility and impact business outcomes.

To address this concern, use your existing operational expertise from within your organization to support application teams on their journey to cloud operations. This can be a dedicated team of experts or a virtual team with participants selected from across your organization. However, the goal remains the same, which is to provide operational support that builds the capability of the workload team, using cloud first principles of automation, removing undifferentiated heavy lifting, providing standardized patterns, and promoting autonomy. The aim is to build sufficient maturity across cloud capabilities and lower the barrier of operational responsibilities so that application teams no longer need additional support.

The COPE model focuses on the workload level. If this approach is needed across multiple teams at once, if you are performing a complex, large-scale, multi-year migration project, or if you are building a platform to support these initiatives, consider using a Cloud Center of Excellence (CCoE).

This is a mechanism that many have found successful when seeking to accelerate their migrations to the cloud and broadly transform their organization.



Cloud Operations and Platform Enablement (COPE)

Your platform engineering team builds a thin layer of core shared platform capabilities, which are based on predefined standards for application teams to adopt and are provided by the COPE team. The platform engineering team codifies 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, observability tools, backup and recovery tools, and networking) to the application teams.

The COPE team manages and supports the standardized services and provides assistance to application teams establishing their cloud presence based on the reference architectures and patterns. They work with the application teams to help them establish baseline operations. During this process, the application teams progressively take more responsibility for their systems and resources over time. The COPE team drives continual improvement together with the platform engineering team and acts as proponents for the application teams.

The application teams get assistance setting up environments, CI/CD pipelines, change management, observability and monitoring, and establishing incident and event management

processes, with the COPE team integrated 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 gains 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 may also retain the access necessary to support operations activities, provide an enterprise-operations view spanning application teams, and offer critical incident management support. The COPE team retains responsibility for activities considered as 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.

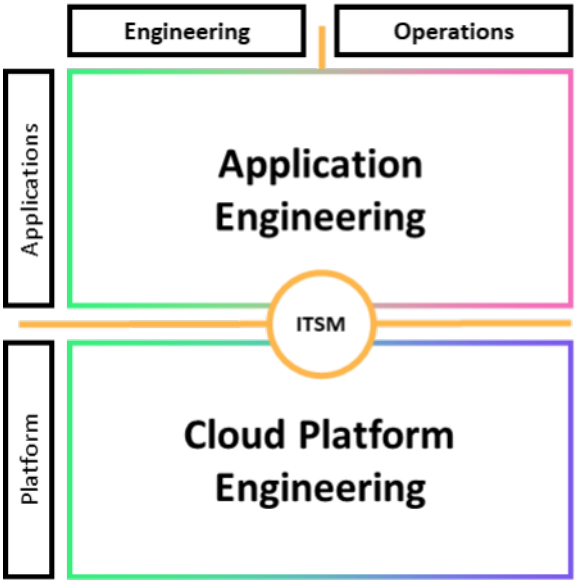
You gain the advantage of your organization's standards, best practices, processes, and expertise, derived from the successes of your teams. You establish a mechanism to replicate these successful patterns for new teams adopting or modernizing in the cloud. This model places emphasis on the COPE team's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

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

Distributed DevOps

The distributed DevOps model separates (or distributes) the application engineering operations and infrastructure engineering operations responsibilities across the engineering teams, following the COPE methodology.

Your application engineers 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.



Distributed DevOps

For this example, we treat governance as centralized elsewhere within the organization. Standards are distributed, provided, or shared to the application and platform teams.

Use tools or services that help you centrally govern your environments across accounts, such as AWS Organizations. Services like AWS Control Tower expand this management capability by helping you 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 networking) 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 an application team's operations.

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

The distributed DevOps model provides strong feedback loops to application teams. Day-to-day operations of a workload increase 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.

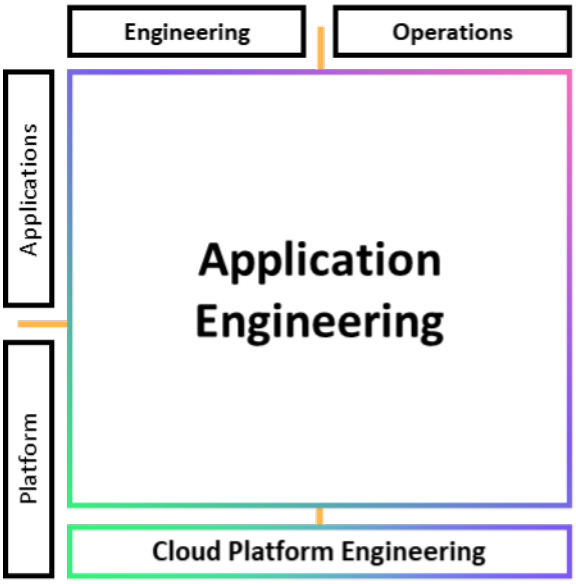
All of this is also true for the platform team supporting the application teams, as the platform team should view these application teams as their customers.

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 focus on differentiating their workloads.

Decentralized DevOps

The decentralized DevOps model is a variation of the *you build it, you run it* methodology where operations are primarily under the ownership of workload teams.

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



Decentralized DevOps

For this example, we 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 inadequate and defects are not recognized early.

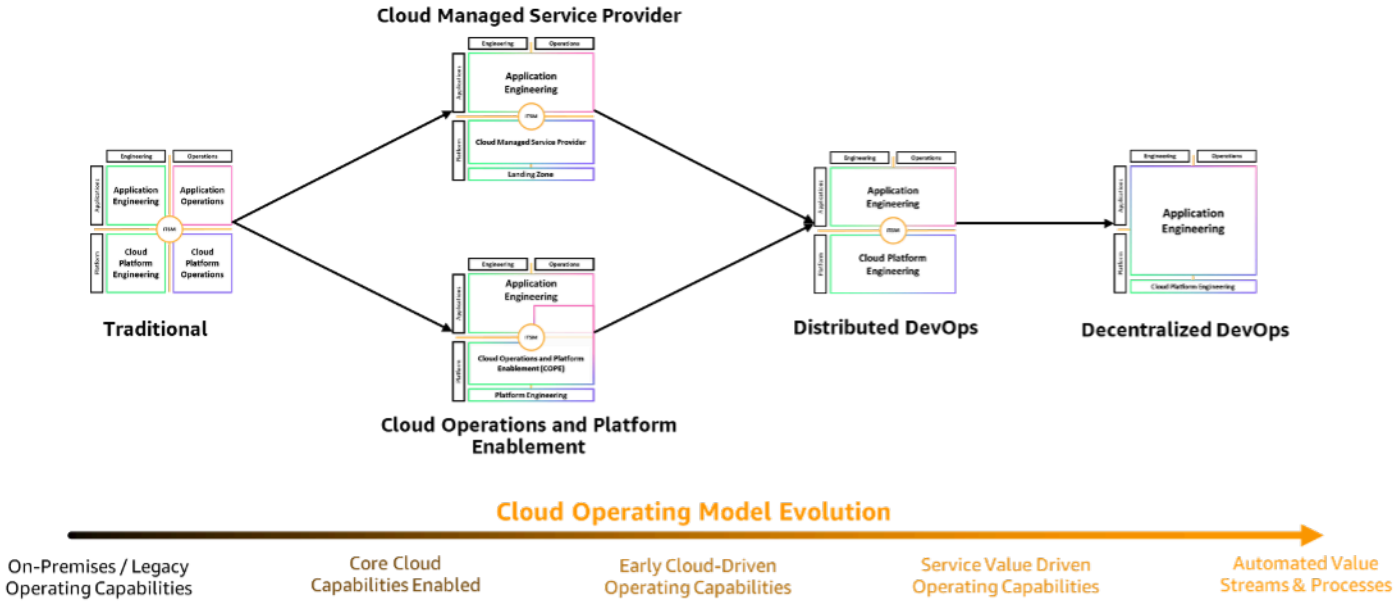
Enforce policies that are not specifically delegated to application teams. Use tools or services that help you to centrally govern your environments across accounts, such as AWS Organizations. Services like AWS Control Tower expand this management capability by helping you define blueprints (supporting 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 can 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.

Evolving your operating model

The models provided progressively move towards more autonomy at the workload level, matching the two-pizza team principle. It is important to understand that this journey from a traditional approach to decentralized DevOps (as a foundation for continued evolution to a two-pizza team model) is likely to take time and require building maturity across a number of capabilities. Therefore, we have provided an example of how you may transition between models as your team and organization move along the enterprise transformation journey. In each change or model, you are evolving towards a more autonomous, but still organizationally-aligned team.



Cloud operating model evolution

When evaluating how your team can support your organizations evolution, examine the trade-offs between models, where your individual teams exist within the models (as they transition and evolve), how your team's relationship and responsibilities could change, and if the benefits merit the impact on your organization. Keep in mind that change is never linear. Some models are more appropriate for specific use cases or points in the journey, and some of these models may provide advantages over the ones in your environment.

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
- OPS02-BP02 Processes and procedures have identified owners
- OPS02-BP03 Operations activities have identified owners responsible for their performance
- OPS02-BP04 Mechanisms exist to manage responsibilities and ownership
- OPS02-BP05 Mechanisms exist to request additions, changes, and exceptions
- OPS02-BP06 Responsibilities between teams are predefined or negotiated

OPS02-BP01 Resources have identified owners

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

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

4. Use Amazon Q Business, a conversational assistant that uses generative AI to enhance workforce productivity, answer questions, and complete tasks based on information in your enterprise systems.
    a. Connect Amazon Q Business to your company's data source. Amazon Q Business offers prebuilt connectors to over 40 supported data sources, including Amazon Simple Storage Service (Amazon S3), Microsoft SharePoint, Salesforce, and Atlassian Confluence. For more information, see Amazon Q Business connectors.
5. 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
- OPS02-BP04 Mechanisms exist to manage responsibilities and ownership

Related documents:

- AWS Account Management - Updating contact information
- AWS Organizations - Updating alternative contacts in your organization
- AWS Tagging Best Practices whitepaper
- Build private and secure enterprise generative AI apps with Amazon Q Business and AWS IAM Identity Center
- Amazon Q Business, now generally available, helps boost workforce productivity with generative AI
- AWS Cloud Operations & Migrations Blog - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations
- AWS Security Blog - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps Blog - Integrating AWS CloudFormation Guard into CI/CD pipelines

Related workshops:

- AWS Workshop - Tagging

Related examples:

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

Related services:

- AWS Config Rules - required-tags
- 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.

Desired outcome: Your organization has a well defined and maintained set of process and procedures for operational tasks. The process and procedures are stored in a central location and available to your team members. Process and procedures are updated frequently, by clearly assigned ownership. Where possible, scripts, templates, and automation documents are implemented as code.

Common anti-patterns:

- Processes are not documented. Fragmented scripts may exist on isolated operator workstations.
- Knowledge of how to use scripts is held by a few individuals or informally as team knowledge.
- A legacy process is due for an update, but ownership of the update is unclear, and the original author is no longer part of the organization.
- Processes and scripts are not discoverable, so they are not readily available when required (for example, in response to an incident).

Benefits of establishing this best practice:

- Processes and procedures boost your efforts to operate your workloads.
- New team members become effective more quickly.
- Reduced time to mitigate incidents.
- Different team members (and teams) can use the same processes and procedures in a consistent manner.
- Teams can scale their processes with repeatable processes.
- Standardized processes and procedures help mitigate the impact of transferring workload responsibilities between teams.

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

Implementation guidance

- Processes and procedures have identified owners who are responsible for their definition.
  - Identify the operations activities conducted in support of your workloads. Document these activities in a discoverable location.
  - Uniquely identify the individual or team responsible for the specification of an activity. They are responsible to verify that 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 for providing the detailed feedback necessary for the activity to be improved.
  - Capture ownership in the metadata of the activity artifact through services like AWS Systems Manager, through documents, and AWS Lambda. Capture resource ownership using tags or resource groups, specifying ownership and contact information. Use AWS Organizations to create tagging polices and capture ownership and contact information.
- Over time, these procedures should be evolved to be runnable as code, reducing the need for human intervention.
  - For example, consider AWS Lambda functions, CloudFormation templates, or AWS Systems Manager automation docs.
  - Perform version control in appropriate repositories.
  - Include suitable resource tagging so owners and documentation can readily be identified.

Customer example

AnyCompany Retail defines ownership as the team or individual that owns processes for an application or groups of applications (that share common architetural practices and technologies). Initially, the process and procedures are documented as step-by-step guides in the document management system, discoverable using tags on the AWS account that hosts the application and on specific groups of resources within the account. They leverage AWS Organizations to manage their AWS accounts. Over time, these processes are converted to code, and resources are defined using infrastructure as code (such as CloudFormation or AWS Cloud Development Kit (AWS CDK) templates). The operational processes become automation documents in AWS Systems Manager or AWS Lambda functions, which can be initiated as scheduled tasks, in response to events such as AWS CloudWatch alarms or AWS EventBridge events, or started by requests within an IT service management (ITSM) platform. All process have tags to identify ownership. Documentation for the automation and process is maintained within the wiki pages generated by the code repository for the process.

Implementation steps

1. Document the existing processes and procedures.
    a. Review and keep them up-to-date.
    b. Identify an owner for each process or procedure.
    c. Place them under version control.
    d. Where possible, share processes and procedures across workloads and environments that share architectural designs.
2. Establish mechanisms for feedback and improvement.
    a. Define policies for how frequently processes should be reviewed.
    b. Define processes for reviewers and approvers.
    c. Implement issues or a ticketing queue for feedback to be provided and tracked.
    d. Whereever possible, processes and procedures should have pre-approval and risk classification from a change approval board (CAB).
3. Verify that processes and procedures are accessible and discoverable by those who need to run them.
    a. Use tags to indicate where the process and procedures can be accessed for the workload.
    b. Use meaningful error and event messaging to indicate the appropriate processes or procedures to address an issue.
    c. Use wikis and document management, and make processes and procedures searchable consistently accross the organization.

4. Use Amazon Q Business, a conversational assistant that uses generative AI to enhance workforce productivity, answer questions, and complete tasks based on information in your enterprise systems.
    a. Connect Amazon Q Business to your company's data source. Amazon Q Business offers prebuilt connectors to over 40 supported data sources, including Amazon S3, Microsoft SharePoint, Salesforce, and Atlassian Confluence. For more information, see Amazon Q connectors.
5. Automate when appropriate.
    a. Automations should be developed when services and technologies provide an API.
    b. Educate adequately on processes. Develop the user stories and requirements to automate those processes.
    c. Measure the use of your processes and procedures successfully, and create issues or tickets to support iterative improvement.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS02-BP01 Resources have identified owners
- OPS02-BP04 Mechanisms exist to manage responsibilities and ownership
- OPS11-BP04 Perform knowledge management

Related documents:

- AWS Whitepaper - Introduction to DevOps on AWS
- AWS Whitepaper - Best Practices for Tagging AWS Resources
- AWS Whitepaper - Organizing Your AWS Environment Using Multiple Accounts
- AWS Cloud Operations and Migrations Blog - Using Amazon Q Business to streamline your operations
- AWS Cloud Operations & Migrations Blog - Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Cloud Operations & Migrations Blog - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations

- AWS Security Blog - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps Blog - Integrating AWS CloudFormation Guard into CI/CD pipelines

Related workshops:

- AWS Well-Architected Operational Excellence Workshop
- AWS Workshop - Tagging

Related videos:

- How to automate IT Operations on AWS
- AWS re:Invent 2020 - Automate anything with AWS Systems Manager
- AWS re:Inforce 2022 - Automating patch management and compliance using AWS (NIS306)
- Supports You - Diving Deep into AWS Systems Manager

Related services:

- AWS Systems Manager - Automation
- AWS Service Management Connector

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.

Desired outcome:

Your organization clearly defines responsibilities to perform specific activities on defined workloads and respond to events generated by the workload. The organization documents ownership of processes and fulfillment and makes this information discoverable. You review and update responsibilities when organizational changes take place, and teams track and measure the performance of defect and inefficiency identification activities. You implement feedback mechanisms to track defects and improvements and support iterative improvement.

Common anti-patterns:

- You do not document responsibilities.
- Fragmented scripts exist on isolated operator workstations. Only a few individuals know how to use them or informally refer to them as *team knowledge*.
- A legacy process is due for update, but no one knows who owns the process, and the original author is no longer part of the organization.
- Processes and scripts can't be discovered, and they are not readily available when required (for example, in response to an incident).

Benefits of establishing this best practice:

- You understand who is responsible to perform an activity, who to notify when action is needed, and who performs the action, validates the result, and provides feedback to the owner of the activity.
- Processes and procedures boost your efforts to operate your workloads.
- New team members become effective more quickly.
- You reduce the time it takes to mitigate incidents.
- Different teams use the same processes and procedures to perform tasks in a consistent manner.
- Teams can scale their processes with repeatable processes.
- Standardized processes and procedures help mitigate the impact of transferring workload responsibilties between teams.

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

Implementation guidance

To begin to define responsibilities, start with existing documentation, like responsibility matrices, processes and procedures, roles and responsibilities, and tools and automation. Review and host discussions on the responsibilities for documented processes. Review with teams to identify misalignments between document responsibilities and processes. Discuss services offered with internal customers of that team to identify expectations gaps between teams.

Analyze and address the discrepancies. Identify opportunities to improvement, and look for frequently requested, resource-intensive activities, which are typically strong candidates for

improvement. Explore best practices, patterns, and prescriptive guidance to simplify and standardize improvements. Record improvement opportunities, and track the improvements to completion.

Over time, these procedures should be evolved to be run as code, reducing the need for human intervention. For example, procedures can be initiated as AWS Lambda functions, AWS CloudFormation templates, or AWS Systems Manager Automation documents. Verify that these procedures are version-controlled in appropriate repositories, and include suitable resource tagging so that teams can readily identify owners and documentation. Document the responsibility for carrying out the activities, and then monitor the automations for successful initiation and operation, as well as performance of the desired outcomes.

Customer example

AnyCompany Retail defines ownership as the team or individual that owns processes for an application or groups of applications that share common architectural practices and technologies. Initially, the company documents the processes and procedures as step-by-step guides in the document management system. They make the procedures discoverable using tags on the AWS account that hosts the application and on specific groups of resources within the account, using AWS Organizations to manage their AWS accounts. Over time, AnyCompany Retail converts these processes to code and defines resources using infrastructure as code (through services like CloudFormation or AWS Cloud Development Kit (AWS CDK) templates). The operational processes become Automation documents in AWS Systems Manager or AWS Lambda functions, which can be initiated as scheduled tasks in response to events such as Amazon CloudWatch alarms or Amazon EventBridge events or by requests within an IT service management (ITSM) platform. All processes have tags to identify who owns them. Teams manage documentation for the automation and process within the wiki pages generated by the code repository for the process.

Implementation steps

1. Document the existing processes and procedures.
    a. Review and verify that they are up-to-date.
    b. Verify that each process or procedure has an owner.
    c. Place the procedures under version control.
    d. Where possible, share processes and procedures across workloads and environments that share architectural designs.
2. Establish mechanisms for feedback and improvement.

   a. Define policies for how frequently processes should be reviewed.
   b. Define processes for reviewers and approvers.
   c. Implement issues or a ticketing queue to provide and track feedback.
   d. Wherever possible, provide pre-approval and risk classification for processes and procedures from a change approval board (CAB).
3. Make process and procedures accessible and discoverable by users who need to run them.
   a. Use tags to indicate where the process and procedures can be accessed for the workload.
   b. Use meaningful error and event messaging to indicate the appropriate process or proceedure to address the issue.
   c. Use wikis or document management to make processes and procedures consistently searchable across the organization.
4. Automate when it is appropriate to do so.
   a. Where services and technologies provide an API, develop automations.
   b. Verify that processes are well-understood, and develop the user stories and requirements to automate those processes.
   c. Measure the successful use of processes and procedures, with issue tracking to support iterative improvement.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS02-BP01 Resources have identified owners
- OPS02-BP02 Processes and procedures have identified owners
- OPS02-BP04 Mechanisms exist to manage responsibilities and ownership
- OPS02-BP05 Mechanisms exist to identify responsibility and ownership
- OPS11-BP04 Perform knowledge management

Related documents:

- AWS Whitepaper | Introduction to DevOps on AWS
- AWS Whitepaper | Best Practices for Tagging AWS Resources

- AWS Whitepaper | Organizing Your AWS Environment Using Multiple Accounts
- AWS Cloud Operations & Migrations Blog | Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Workshop - Tagging
- AWS Service Management Connector

Related videos:

- AWS Knowledge Center Live | Tagging AWS Resources
- AWS re:Invent 2020 | Automate anything with AWS Systems Manager
- AWS re:Inforce 2022 | Automating patch management and compliance using AWS (NIS306)
- Supports You | Diving Deep into AWS Systems Manager

OPS02-BP04 Mechanisms exist to manage responsibilities and ownership

Understand the responsibilities of your role and how you contribute to business outcomes, as this understanding informs the prioritization of your tasks and why your role is important. This helps team members recognize needs and respond appropriately. When team members know their role, they can establish ownership, identify improvement opportunities, and understand how to influence or make appropriate changes.

Occasionally, a responsibility might not have a clear owner. In these situations, design a mechanism to resolve this gap. Create a well-defined escalation path to someone with the authority to assign ownership or plan to address the need.

Desired outcome: Teams within your organization have clearly-defined responsibilities that include how they are related to resources, actions to be performed, processes, and procedures. These responsibilities align to the team's responsibilities and goals, as well as the responsibilities of other teams. You document the routes of escalation in a consistent and discoverable manner and feed these decisions into documentation artifacts, such as responsibility matrices, team definitions, or wiki pages.

Common anti-patterns:

- The responsibilities of the team are ambiguous or poorly-defined.
- The team does not align roles with responsibilities.

- The team does not align its goals and objectives its responsibilities, which makes it difficult to measure success.
- Team member responsibilities do not align with the team and the wider organization.
- Your team does not keep responsibilities up-to-date, which makes them inconsistent with the tasks performed by the team.
- Escalation paths for determining responsibilities aren't defined or are unclear.
- Escalation paths have no single thread owner to ensure timely reponse.
- Roles, responsibilities, and escalation paths are not discoverable, and they are not readily available when required (for example, in response to an incident).

Benefits of establishing this best practice:

- When you understand who has responsibility or ownership, you can contact the proper team or team member to make a request or transition a task.
- To reduce the risk of inaction and unaddressed needs, you have identified a person who has the authority to assign responsibility or ownership.
- When you clearly define the scope of a responsibility, your team members gain autonomy and ownership.
- Your responsibilities inform the decisions you make, the actions you take, and your handoff activities to their proper owners.
- It's easy to identify abandoned responsibilities because you have a clear understanding of what falls outside of your team's responsibility, which helps you escalate for clarification.
- Teams avoid confusion and tension, and they can more adequately manage their workloads and resources.

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

Implementation guidance

Identify team members roles and responsibilities, and verify that they understand the expectations of their role. Make this information discoverable so that members of your organization can identify who they need to contact for specific needs, whether it's a team or individual. As organizations seek to capitalize on the opportunities to migrate and modernize on AWS, roles and responsibilities might also change. Keep your teams and their members aware of their responsibilities, and train them appropriately to carry out their tasks during this change.

Determine the role or team that should receive escalations to identify responsibility and ownership. This team can engage with various stakeholders to come to a decision. However, they should own the management of the decision making process.

Provide accessible mechanisms for members of your organization to discover and identify ownership and responsibility. These mechanisms teach them who to contact for specific needs.

Customer example

AnyCompany Retail recently completed a migration of workloads from an on-premises environment to their landing zone in AWS with a lift and shift approach. They performed an operations review to reflect on how they accomplish common operational tasks and verified that their existing responsibility matrix reflects operations in the new environment. When they migrated from on-premises to AWS, they reduced the infrastructure teams responsibilities relating to the hardware and physical infrastructure. This move also revealed new opportunities to evolve the operating model for their workloads.

While they identified, addressed, and documented the majority of responsibilities, they also defined escalation routes for any responsibilities that were missed or that may need to change as operations practices evolve. To explore new opportunities to standardize and improve efficiency across your workloads, provide access to operations tools like AWS Systems Manager and security tools like AWS Security Hub and Amazon GuardDuty. AnyCompany Retail puts together a review of responsibilities and strategy based on improvements they wants to address first. As the company adopts new ways of working and technology patterns, they update their responsibility matrix to match.

Implementation steps

1. Start with existing documentation. Some typical source documents might include:
    a. Responsibility or responsible, accountable, consulted, and informed (RACI) matrices
    b. Team definitions or wiki pages
    c. Service definitions and offerings
    d. Role or job descriptions
2. Review and host discussions on the documented responsibilities:
    a. Review with teams to identify misalignments between documented responsibilities and responsibilities the team typically performs.
    b. Discuss potential services offered by internal customers to identify gaps in expectations between teams.

3. Analysis and address the discrepancies.
4. Identify opportunities for improvement.
    a. Identify frequently-requested, resource-intensive requests, which are typically strong candidates for improvement.
    b. Look for best practices, understand patterns, follow prescriptive guidance, and simplify and standardize improvements.
    c. Record improvement opportunities, and track them to completion.
5. If a team doesn't already hold responsibility for managing and tracking the assignment of responsibilities, identify someone on the team to hold this responsibility.
6. Define a process for teams to request clarification of responsibility.
    a. Review the process, and verify that it is clear and simple to use.
    b. Make sure that someone owns and tracks escalations to their conclusion.
    c. Establish operational metrics to measure effectiveness.
    d. Create a feedback mechanisms to verify that teams can highlight improvement opportunities.
    e. Implement a mechanism for periodic review.
7. Document in a discoverable and accessible location.
    a. Wikis or documentation portal are common choices.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS01-BP06 Evaluate tradeoffs
- OPS03-BP02 Team members are empowered to take action when outcomes are at risk
- OPS03-BP03 Escalation is encouraged
- OPS03-BP07 Resource teams appropriately
- OPS09-BP01 Measure operations goals and KPIs with metrics
- OPS09-BP03 Review operations metrics and prioritize improvement
- OPS11-BP01 Have a process for continuous improvement

Related documents:

- AWS Whitepaper - Introduction to DevOps on AWS
- AWS Whitepaper - AWS Cloud Adoption Framework: Operations Perspective
- AWS Well-Architected Framework Operational Excellence - Workload level Operating model topologies
- AWS Prescriptive Guidance - Building your Cloud Operating Model
- AWS Prescriptive Guidance - Create a RACI or RASCI matrix for a cloud operating model
- AWS Cloud Operations & Migrations Blog - Delivering Business Value with Cloud Platform Teams
- AWS Cloud Operations & Migrations Blog - Why a Cloud Operating Model?
- AWS DevOps Blog - How organizations are modernizing for cloud operations

Related videos:

- AWS Summit Online - Cloud Operating Models for Accelerated Transformation
- AWS re:Invent 2023 - Future-proofing cloud security: A new operating model

OPS02-BP05 Mechanisms exist to request additions, changes, and exceptions

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

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

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

Resources

Related best practices:

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

Related documents:

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

Related services:

- AWS Systems Manager Change Manager

OPS02-BP06 Responsibilities between teams are predefined or negotiated

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

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

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

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

Resources

Related best practices:

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

Related documents:

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

Organizational culture

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

Best practices

- OPS03-BP01 Provide executive sponsorship
- OPS03-BP02 Team members are empowered to take action when outcomes are at risk
- OPS03-BP03 Escalation is encouraged
- OPS03-BP04 Communications are timely, clear, and actionable
- OPS03-BP05 Experimentation is encouraged
- OPS03-BP06 Team members are encouraged to maintain and grow their skill sets
- OPS03-BP07 Resource teams appropriately

OPS03-BP01 Provide executive sponsorship

At the highest level, senior leadership acts as the executive sponsor to clearly set expectations and direction for the organization's outcomes, including evaluating its success. The sponsor advocates and drives adoption of best practices and evolution of the organization.

Desired outcome: Organizations that endeavor to adopt, transform, and optimize their cloud operations establish clear lines of leadership and accountability for desired outcomes. The organization understands each capability required by the organization to accomplish a new outcome and assigns ownership to functional teams for development. Leadership actively sets this direction, assigns ownership, takes accountability, and defines the work. As a result, individuals across the organization can mobilize, feel inspired, and actively work towards the desired objectives.

Common anti-patterns:

- There is a mandate for workload owners to migrate workloads to AWS without a clear sponsor and plan for cloud operations. This results in teams not consciously collaborating to improve and mature their operational capabilities. Lack of operational best practice standards overwhelm teams (such as operator-toil, on-calls, and technical debt), which constrains innovation.
- A new organization-wide goal has been set to adopt an emerging technology without providing leadership sponsor and strategy. Teams interpret goals differently, which causes confusion on where to focus efforts, why they matter, and how to measure impact. Consequently, the organization loses momentum in adopting the technology.

Benefits of establishing this best practice: When executive sponsorship clearly communicates and shares vision, direction, and goals, team members know what is expected of them. Individuals and teams begin to intensely focus effort in the same direction to accomplish defined objectives when leaders are actively engaged. As a result, the organization maximizes the ability to succeed. When you evaluate success, you can better identify barriers to success so that they can be addressed through intervention by the executive sponsor.

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

Implementation guidance

- At every phase of the cloud journey (migration, adoption, or optimization), success requires active involvement at the highest level of leadership with a designated executive sponsor. The

executive sponsor aligns the team's mindset, skillsets, and ways of working to the defined strategy.

- **Explain the *why*:** Bring clarity and explain the reasoning behind the vision and strategy.
- **Set expectations:** Define and publish goals for your organizations, including how progress and success are measured.
- **Track achievement of goals:** Measure the incremental achievement of goals regularly (not just completion of tasks). Share the results so that appropriate action can be taken if outcomes are at risk.
- **Provide the resources necessary to achieve your goals:** Bring people and teams together to collaborate and build the right solutions that bring about the defined outcomes. This reduces or eliminates organizational friction.
- **Advocate for your teams:** Remain engaged with your teams so that you understand their perforamnce and whether there are external factors affecting them. Identify obstacles that are impeding your teams progress. Act on behalf of your teams to help address obstacles and remove unnecessary burdens. When your teams are impacted by external factors, reevaluate goals and adjust targets as appropriate.
- **Drive 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.
- **Encourage evolution of your teams:** Create a culture of continual improvement, and proactively learn from progress made as well as failures. Encourage both personal and organizational growth and development. Use data and anecdotes to evolve the vision and strategy.

Customer example

AnyCompany Retail is in the process of business transformation through rapid reinvention of customer experiences, enhancement of productivity, and acceleration of growth through generative AI.

Implementation steps

1. Establish single-threaded leadership, and assign a primary executive sponsor to lead and drive the transformation.

2. Define clear business outcomes of your transformation, and assign ownership and accountability. Empower the primary executive with the authority to lead and make critical decisions.
3. Verify that your transformational strategy is very clear and communicated widely by the executive sponsor to every level of the organization.
    a. Establish clearly defined business objectives for IT and cloud initiatives.
    b. Document key business metrics to drive IT and cloud transformation.
    c. Communicate the vision consistently to all teams and individuals responsible for parts of the strategy.
4. Develop communication planning matrices that specify what message needs to be delivered to specified leaders, managers, and individual contributors. Specify the person or team that should deliver this message.
    a. Fulfill communications plans consistently and reliably.
    b. Set and manage expectations through in-person events on a regular basis.
    c. Accept feedback on the effectiveness of communications, and adjust the communications and plan accordingly.
    d. Schedule communication events to proactively understand challenges from teams, and establish a consistent feedback loop that allows for correcting course where necessary.
5. Actively engage each initiative from a leadership perspective to verify that all impacted teams understand the outcomes they are accountable to achieve.
6. At every status meeting, executive sponsors should look for blockers, inspect established metrics, anecdotes, or feedback from the teams, and measure progress towards objectives.

Level of effort for the implementation plan Medium

Resources

Related best practices:

- OPS03-BP04 Communications are timely, clear, and actionable
- OP11-BP01 Have a process for continuous improvement
- OPS11-BP07 Perform operations metrics reviews

Related documents:

- Untangling Your Organisational Hairball: Highly Aligned
- The Living Transformation: Pragmatically approaching changes
- Becoming a Future-Ready Enterprise
- 7 Pitfalls to Avoid When Building a CCOE
- Navigating the Cloud: Key Performance Indicators for Success

Related videos:

- AWS re:Invent 2023: A leader's guide to generative AI: Using history to shape the future (SEG204)

Related examples:

- Prosci: Primary Sponsor's Role & Importance

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

A cultural behavior of ownership instilled by leadership results in any employee feeling empowered to act on behalf of the entire company beyond their defined scope of role and accountability. Employees can act to proactively identify risks as they emerge and take appropriate action. Such a culture allows employees to make high value decisions with situational awareness.

For example, Amazon uses Leadership Principles as the guidelines to drive desired behavior for employees to move forward in situations, solve problems, deal with conflict, and take action.

Desired outcome: Leadership has influenced a new culture that allows individuals and teams to make critical decisions, even at lower levels of the organization (as long as decisions are defined with auditable permissions and safety mechanisms). Failure is not discouraged, and teams iteratively learn to improve their decision-making and responses to tackle similar situations going forward. If someone's actions result in an improvement that can benefit other teams, they proactively share knowledge from such actions. Leadership measures operational improvements and incentivizes the individual and organization for adoption of such patterns.

Common anti-patterns:

- There isn't clear guidance or mechanisms in an organization for what to do when a risk is identified. For example, when an employee notices a phishing attack, they fail to report to the

security team, resulting in a large portion of the organization falling for the attack. This causes a data breach.

- Your customers complain about service unavailability, which primarily stems from failed deployments. Your SRE team is responsible for the deployment tool, and an automated rollback for deployments is in their long-term roadmap. In a recent application rollout, one of the engineers devised a solution to automate rolling back their application to a previous version. Though their solution can become the pattern for SRE teams, other teams do not adopt, as there is no process to track such improvements. The organization continues to be plagued with failed deployments impacting customers and causing further negative sentiment.
- In order to stay compliant, your infosec team oversees a long-established process to rotate shared SSH keys regularly on behalf of operators connecting to their Amazon EC2 Linux instances. It takes several days for the infosec teams to complete rotating keys, and you are blocked from connecting to those instances. No one inside or outside of infosec suggests using other options on AWS to achieve the same result.

Benefits of establishing this best practice: By decentralizing authority to make decisions and empowering your teams to decide key decisions, you are able to address issues more quickly with increasing success rates. In addition, teams start to realize a sense of ownership, and failures are acceptable. Experimentation becomes a cultural mainstay. Managers and directors do not feel as though they are micro-managed through every aspect of their work.

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

Implementation guidance

1. Develop a culture where it is expected that failures can occur.
2. Define clear ownership and accountability for various functional areas within the organization.
3. Communicate ownership and accountability to everyone so that individuals know who can help them facilitate decentralized decisions.
4. Define your one-way and two-way door decisions to help individuals know when they do need to escalate to higher levels of leadership.
5. Create organizational awareness that all employees are empowered to take action at various levels when outcomes are at risk. Provide your team members documentation of governance, permission-levels, tools, and opportunities to practice the skills necessary to respond effectively.

6. Give your team members the opportunity to practice the skills necessary to respond to various decisions. Once decision levels are defined, perform game days to verify that all individual contributors understand and can demonstrate the process.
   a. Provide alternative safe environments where processes and procedures can be tested and trained upon.
   b. Acknowledge and create awareness that team members have authority to take action when the outcome has a predefined level of risk.
   c. Define the authority of your team members to take action by assigning permissions and access to the workloads and components they support.
7. Provide ability for teams to share their learnings (operational successes and failures).
8. Empower teams to challenge the status quo, and provide mechanisms to track and measure improvements, as well as their impact to the organization.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS01-BP06 Evaluate tradeoffs while managing benefits and risks
- OPS02-BP05 Mechanisms exist to identify responsibility and ownership

Related documents:

- AWS Blog Post | The agile enterprise
- AWS Blog Post | Measuring success : A paradox and a plan
- AWS Blog Post | Letting go : Enabling autonomy in teams
- Centralize or Decentralize?

Related videos:

- re:Invent 2023 | How to not sabotage your transformation (SEG201)
- re:Invent 2021 | Amazon Builders' Library: Operational Excellence at Amazon
- Centralization vs. Decentralization

Related examples:

- Using architectural decision records to streamline technical decision-making for a software development project

OPS03-BP03 Escalation is encouraged

Team members are encouraged by leadership to escalate issues and concerns to higher-level decision makers and stakeholders if they believe desired outcomes are at risk and expected standards are not met. This is a feature of the organization's culture and is driven at all levels. Escalation should be done early and often so that risks can be identified and prevented from causing incidents. Leadership does not reprimand individuals for escalating an issue.

Desired outcome: Individuals throughout the organization are comfortable to escalate problems to their immediate and higher levels of leadership. Leadership has deliberately and consciously established expectations that their teams should feel safe to escalate any issue. A mechanism exists to escalate issues at each level within the organization. When employees escalate to their manager, they jointly decide the level of impact and whether the issue should be escalated. In order to initiate an escalation, employees are required to include a recommended work plan to address the issue. If direct management does not take timely action, employees are encouraged to take issues to the highest level of leadership if they feel strongly that the risks to the organization warrant the escalation.

Common anti-patterns:

- Executive leaders do not ask enough probing questions during your cloud transformation program status meeting to find where issues and blockers are occurring. Only good news is presented as status. The CIO has made it clear that she only likes to hear good news, as any challenges brought up make the CEO think that the program is failing.
- You are a cloud operations engineer and you notice that the new knowledge management system is not being widely adopted by application teams. The company invested one year and several million dollars to implement this new knowledge management system, but people are still authoring their runbooks locally and sharing them on an organizational cloud share, making it difficult to find knowledge pertinent to supported workloads. You try to bring this to leadership's attention, because consistent use of this system can enhance operational efficiency. When you bring this to the director who lead the implementation of the knowledge management system, she reprimands you because it calls the investment into question.

- The infosec team responsible for hardening compute resources has decided to put a process in place that requires performing the scans necessary to ensure that EC2 instances are fully secured before the compute team releases the resource for use. This has created a time delay of an additional week for resources to be deployed, which breaks their SLA. The compute team is afraid to escalate this to the VP over cloud because this makes the VP of information security look bad.

Benefits of establishing this best practice:

Complex or critical issues are addressed before they impact the business. Less time is wasted. Risks are minimized. Teams become more proactive and results focused when solving problems.

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

Implementation guidance

The willingness and ability to escalate freely at every level in the organization is an organizational and cultural foundation that should be consciously developed through emphasized training, leadership communications, expectation setting, and the deployment of mechanisms throughout the organization at every level.

Implementation steps

1. Define policies, standards, and expectations for your organization.
   a. Ensure wide adoption and understanding of policies, expectations, and standards.
2. Encourage, train, and empower workers for early and frequent escalation when standards are not met.
3. Organizationally acknowledge that early and frequent escalation is the best practice. 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.
   a. Build a mechanism for escalation (like an Andon cord system).
   b. Have documented procedures defining when and how escalation should occur.
   c. Define the series of people with increasing authority to take or approve action, as well as each stakeholder's contact information.
4. When escalation occurs, it should continue until the team member is satisfied that the risk has been mitigated through actions driven from leadership.
   a. Escalations should include:

   i. Description of the situation, and the nature of the risk
   ii. Criticality of the situation
   iii. Who or what is impacted
   iv. How great the impact is
   v. Urgency if impact occurs
   vi. Suggested remedies and plans to mitigate
   b. 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.
5. Encourage a culture of continuous improvement feedback loops in everything that the organization produces. Feedback loops act as minor escalations to individuals responsible, and they identify improvement opportunities, even when escalation is not needed. Continuous improvement cultures force everyone to be more proactive.
6. Leadership should periodically reemphasize the policies, standards, mechanisms, and the desire for open escalation and continuous feedback loops without retribution.

Level of effort for the Implementation Plan: Medium

Resources

Related best practices:

- OPS02-BP05 Mechanisms exist to request additions, changes, and exceptions

Related documents:

- How do you foster a culture of continuous improvement and learning from Andon and escalation systems?
- The Andon Cord (IT Revolution)
- AWS DevOps Guidance | Establish clear escalation paths and encourage constructive disagreement

Related videos:

- Jeff Bezos on how to make decisions (& increase velocity)

- Toyota Product System: Stopping Production, a Button, and an Andon Electric Board
- Andon Cord in LEAN Manufacturing

Related examples:

- Working with escalation plans in Incident Manager

OPS03-BP04 Communications are timely, clear, and actionable

Leadership is responsible for the creation of strong and effective communications, especially when the organization adopts new strategies, technologies, or ways of working. Leaders should set expectations for all staff to work towards the company objectives. Devise communication mechanisms that create and maintain awareness among the teams responsible for running plans that are funded and sponsored by leadership. Make use of cross-organizational diversity, and listen attentively to multiple unique perspectives. Use this perspective to increase innovation, challenge your assumptions, and reduce the risk of confirmation bias. Foster inclusion, diversity, and accessibility within your teams to gain beneficial perspectives.

Desired outcome: Your organization designs communication strategies to address the impact of change to the organization. Teams remain informed and motivated to continue working with one another rather than against each other. Individuals understand how important their role is to achieve the stated objectives. Email is only a passive mechanism for communications and used accordingly. Management spends time with their individual contributors to help them understand their responsibility, the tasks to complete, and how their work contributes to the overall mission. When necessary, leaders engage people directly in smaller venues to convey messages and verify that these messages are being delivered effectively. As a result of good communications strategies, the organization performs at or above the expectations of leadership. Leadership encourages and seeks diverse opinions within and across teams.

Common anti-patterns:

- Your organization has a five year plan to migrate all workloads to AWS. The business case for cloud includes the modernization of 25% of all workloads to take advantage of serverless technology. The CIO communicates this strategy to direct reports and expects each leader to cascade this presentation to managers, directors, and individual contributors without any in-person communication. The CIO steps back and expects his organization to perform the new strategy.

- Leadership does not provide or use a mechanism for feedback, and an expectation gap grows, which leads to stalled projects.
- You are asked to make a change to your security groups, but you are not given any details of what change needs to be made, what the impact of the change could be on all the workloads, and when it should happen. The manager forwards an email from the VP of InfoSec and adds the message "Make this happen."
- Changes were made to your migration strategy that reduce the planned modernization number from 25% to 10%. This has downstream effects on the operations organization. They were not informed of this strategic change and thus, they are not ready with enough skilled capacity to support a greater number of workloads lifted and shifted into AWS.

Benefits of establishing this best practice:

- Your organization is well-informed on new or changed strategies, and they act accordingly with strong motivation to help each other achieve the overall objectives and metrics set by leadership.
- Mechanisms exist and are used to provide timely notice to team members of known risks and planned events.
- New ways of working (including changes to people or the organization, processes, or technology), along with required skills, are more effectively adopted by the organization, and your organization realizes business benefits more quickly.
- Team members have the necessary context of the communications being received, and they can be more effective in their jobs.

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. For any significant IT transitions, an established planning team can more successfully manage the impact of change to its people than an organization that ignores this practice. Larger organizations can be more challenging when managing change because it's critical to establish strong buy-in on a new strategy with all individual contributors. In the absence of such a transition planning team, leadership holds 100% of the responsibility for effective communications. When establishing a transition planning team, assign team members to work with all organizational leadership to define and manage effective communications at every level.

Customer example

AnyCompany Retail signed up for AWS Enterprise Support and depends on other third-party providers for its cloud operations. The company uses chat and chatops as their main communication medium for operational activities. Alerts and other information populate specific channels. When someone must act, they clearly state the desired outcome, and in many cases, they receive a runbook or playbook to use. They schedule major changes to production systems with a change calendar.

Implementation steps

1. Establish a core team within the organization that has accountability to build and initiate communication plans for changes that happen at multiple levels within the organization.
2. Institute single-threaded ownership to achieve oversight. Give individual teams the ability to innovate independently, and balance the use of consistent mechanisms, which allows for the right level of inspection and directional vision.
3. Work with stakeholders across your organization to agree to communication standards, practices, and plans.
4. Verify that the core communications team collaborates with organizational and program leadership to craft messages to appropriate staff on behalf of leaders.
5. Build strategic communication mechanisms to manage change through announcements, shared calendars, all-hands meetings, and in-person or one-on-one methods so that team members have proper expectations on the actions they should take.
6. Provide necessary context, details, and time (when possible) to determine if action is necessary. When action is needed, provide the required action and its impact.
7. Implement tools that facilitate tactical communications, like internal chat, email, and knowledge management.
8. Implement mechanisms to measure and verify that all communications lead to desired outcomes.
9. Establish a feedback loop that measures the effectiveness of all communications, especially when communications are related to resistance to changes throughout the organization.
10. For all AWS accounts, establish alternate contacts for billing, security, and operations. Ideally, each contact should be an email distribution as opposed to a specific individual contact.
11. Establish an escalation and reverse escalation communication plan to engage with your internal and external teams, including AWS support and other third-party providers.

12. Initiate and perform communication strategies consistently throughout the life of each transformation program.
13. Prioritize actions that are repeatable where possible to safely automate at scale.
14. When communications are required in scenarios with automated actions, the communication's purpose should be to inform teams, for auditing, or a part of the change management process.
15. Analyze communications from your alert systems for false positives or alerts that are constantly created. Remove or change these alerts 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.
16. 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.
17. 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.
18. 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.
19. **Seek diverse opinions and perspectives:** Encourage contributions from everyone. Give communication opportunities to under-represented groups. Rotate roles and responsibilities in meetings.
    a. **Expand roles and responsibilities:** Provide opportunities for team members to take on roles that they might not otherwise. They can gain experience and perspective from the role and from interactions with new team members with whom they might not otherwise interact. They can also bring their experience and perspective to the new role and team members they interact with. As perspective increases, identify emergent business opportunities or new opportunities for improvement. Rotate common tasks between members within a team that others typically perform to understand the demands and impact of performing them.
    b. **Provide a safe and welcoming environment:** Establish policy and controls that protect the mental and physical safety of team members 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 are heard, they are more likely to share valuable insights (for example, marketing opportunities, accessibility needs, unserved market segments, and unacknowledged risks in your environment).

c. **Encourage 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 develop skills for working around them. These uniquely-developed skills can provide significant benefit to your organization. Support team members with necessary accommodations to increase the benefits you can receive from their contributions.

Resources

Related best practices:

- OPS03-BP01 Provide executive sponsorship
- OPS07-BP03 Use runbooks to perform procedures
- OPS07-BP04 Use playbooks to investigate issues

Related documents:

- AWS Blog post | Accountability and empowerment are key to high-performing agile organizations
- AWS Executive Insights | Learn to scale innovation, not complexity | Single-threaded Leaders
- AWS Security Bulletins
- Open CVE
- Support App in Slack to Manage Support Cases
- Manage AWS resources in your Slack channels with Amazon Q Developer in chat applications

Related services:

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

OPS03-BP05 Experimentation is encouraged

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

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

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

Resources

Related best practices:

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

Related documents:

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

- Organizing Your AWS Environment Using Multiple Accounts - Sandbox OU
- Using AWS AppConfig Feature Flags

Related videos:

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

Related examples:

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

Related services:

- Amazon CloudWatch Evidently
- AWS AppConfig
- AWS Control Tower

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

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

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

Resources such as Support, (AWS re:Post, Support Center), and AWS Documentation help remove technical roadblocks and improve operations. Reach out to Support through Support Center for help with your questions.

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.

AWS Training and Certification includes free training through self-paced digital courses, along with learning plans by role or domain. You can also register for instructor-led training to further support the development of your teams' AWS skills.

Desired outcome: Your organization constantly evaluates skill gaps and closes them with structured budget and investment. Teams encourage and incentivize their members with upskilling activities such as acquiring leading industry certifications. Teams take advantage of dedicated cross-sharing knowledge programs such as lunch-and-learns, immersion days, hackathons, and gamedays. Your organization's keeps its knowledge systems up-to-date and relevant to cross-train team members, including new-hire onboarding trainings.

Common anti-patterns:

- In the absence of a structured training program and budget, teams experience uncertainty as they try to keep pace with technology evolution, which results in increased attrition.
- As part of migrating to AWS, your organization demonstrates skill gaps and varying cloud fluency amongst teams. Without an effort to upskill, teams find themselves overtasked with legacy and inefficient management of the cloud environment, which causes increased operator toil. This burn out increases employee dissatisfaction.

Benefits of establishing this best practice: When your organization consciously invests in improving the skills of its teams, it also helps accelerate and scale cloud adoption and optimization. Targeted learning programs drive innovation and build operational ability for teams to be prepared to handle events. Teams consciously invest in the implementation and evolution of best practices. Team morale is high, and team members value their contribution to the business.

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

Implementation guidance

To adopt new technologies, fuel innovation, and keep pace with changes in demand and responsibilities to support your workloads, continually invest in the professional growth of your teams.

Implementation steps

1. **Use structured cloud advocacy programs:** AWS Skills Guild provides consultative training to increase cloud skill confidence and ignite a culture of continuous learning.
2. **Provide resources for education:** Provide dedicated, structured time and access to training materials and lab resources, and support participation in conferences and access to professional organizations that provide opportunities for learning from both educators and peers. Provide your junior team members with access to senior team members as mentors, or allow the junior team members to shadow their seniors' 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.
3. **Encourage use of expert technical resources:** Leverage resources such as AWS re:Post to get access to curated knowledge and vibrant community.
4. **Build and maintain an up-to-date knowledge repository:** Use knowledge sharing platforms such as wikis and runbooks. Create your own reusable expert knowledge source with AWS re:Post Private to streamline collaboration, improve productivity, and accelerate employee onboarding.
5. **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.
6. **Support pursuit and maintenance of industry certifications:** Support your team members in the acquisition and maintenance of industry certifications that validate what they have learned and acknowledge their accomplishments.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS03-BP01 Provide executive sponsorship

- OPS11-BP04 Perform knowledge management

Related documents:

- AWS Whitepaper | Cloud Adoption Framework: People Perspective
- Investing in continuous learning to grow your organization's future
- AWS Skills Guild
- AWS Training and Certification
- Support
- AWS re:Post
- AWS Getting Started Resource Center
- AWS Blogs
- AWS Cloud Compliance
- AWS Documentation
- The Official AWS Podcast.
- AWS Online Tech Talks
- AWS Events and Webinars
- AWS Well-Architected Labs
- The Amazon Builders' Library

Related videos:

- AWS re:Invent 2023 | Reskilling at the speed of cloud: Turning employees into entrepreneurs
- WS re:Invent 2023 | Building a culture of curiosity through gamification

OPS03-BP07 Resource teams appropriately

Provision the right amount of proficient team members, and provide tools and resources to support your workload needs. Overburdening team members increases the risk of human error. Investments in tools and resources, such as automation, can scale the effectiveness of your team and help them support a greater number of workloads without requiring additional capacity.

Desired outcome:

- You have appropriately staffed your team to gain the skillsets needed for them to operate workloads in AWS in accordance with your migration plan. As your team has scaled itself up during the course of your migration project, they have gained proficiency in the core AWS technologies that the business plans to use when migrating or modernizing their applications.
- You have carefully aligned your staffing plan to make efficient use of resources by leveraging automation and workflow. A smaller team can now manage more infrastructure on behalf of the application development teams.
- With shifting operational priorities, any resource staffing constraints are proactively identified to protect the success of business initiatives.
- Operational metrics that report operational toil (such as on-call fatigue or excessive paging) are reviewed to verify that staff are not overwhelmed.

Common anti-patterns:

- Your staff have not ramped up on AWS skills as you close in on your multi-year cloud migration plan, which risks support of the workloads and lowers employee morale.
- Your entire IT organization is shifting into agile ways of working. The business is prioritizing the product portfolio and setting metrics for what features need to be developed first. Your agile process does not require teams to assign story points to their work plans. As a result, it is impossible to know the level of capacity required for the next amount of work, or if you have the right skills assigned to the work.
- You are having an AWS partner migrate your workloads, and you don't have a support transition plan for your teams once the partner completes the migration project. Your teams struggle to efficiently and effectively support the workloads.

Benefits of establishing this best practice: You have appropriately-skilled team members available in your organization to support the workloads. Resource allocation can adapt to shifting priorities without impacting performance. The result is teams being proficient at supporting workloads while maximizing time to focus on innovating for customers, which in turn raises employee satisfaction.

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

Implementation guidance

Resource planning for your cloud migration should occur at an organizational level that aligns to your migration plan, as well as the desired operating model being implemented to support your new cloud environment. This should include understanding which cloud technologies are deployed for the business and application development teams. Infrastructure and operations leadership should plan for skills gap analysis, training, and role definition for engineers who are leading cloud adoption.

Implementation steps

1. Define success criteria for team's success with relevant operational metrics such as staff productivity (for example, cost to support a workload or operator hours spent during incidents).
2. Define resource capacity planning and inspection mechanisms to verify that the right balance of qualified capacity is available when needed and can be adjusted over time.
3. Create mechanisms (for example, sending a monthly survey to teams) to understand work-related challenges that impact teams (like increasing responsibilities, changes in technology, loss of personnel, or increase in customers supported).
4. Use these mechanisms to engage with teams and spot trends that may contribute to employee productivity challenges. When your teams are impacted by external factors, reevaluate goals and adjust targets as appropriate. Identify obstacles that are impeding your team's progress.
5. Regularly review if your currently-provisioned resources are still sufficient, of if additional resources are needed, and make appropriate adjustments to support teams.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS03-BP06 Team members are encouraged to maintain and grow their skill sets
- OPS09-BP03 Review operations metrics and prioritize improvement
- OPS10-BP01 Use a process for event, incident, and problem management
- OPS10-BP07 Automate responses to events

Related documents:

- AWS Cloud Adoption Framework: People Perspective
- Becoming a Future-Ready Enterprise
- Prioritize your Employees' Skills to Drive Business Growth
- High performing organization - the Amazon Two-Pizza team
- How Cloud-Mature Enterprises Succeed

Prepare

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

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

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

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

Evaluate the operational readiness of your workload, processes, procedures, and personnel to understand the operational risks related to your workload. Use a consistent process (including manual or automated checklists) to know when you are ready to go live with your workload or

a change. This will also help you to find any areas that you must make plans to address. Have runbooks that document your routine activities and playbooks that guide your processes for issue resolution. Understand the benefits and risks to make informed decisions to permit changes to enter production.

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

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

Topics

- Implement observability
- Design for operations
- Mitigate deployment risks
- Operational readiness and change management

Implement observability

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

Observability goes beyond simple monitoring, providing a comprehensive understanding of a system's internal workings based on its external outputs. Rooted in metrics, logs, and traces, observability offers profound insights into system behavior and dynamics. With effective

observability, teams can discern patterns, anomalies, and trends, allowing them to proactively address potential issues and maintain optimal system health.

Identifying key performance indicators (KPIs) is pivotal to ensure alignment between monitoring activities and business objectives. This alignment ensures that teams are making data-driven decisions using metrics that genuinely matter, optimizing both system performance and business outcomes.

Furthermore, observability empowers businesses to be proactive rather than reactive. Teams can understand the cause-and-effect relationships within their systems, predicting and preventing issues rather than just reacting to them. As workloads evolve, it's essential to revisit and refine the observability strategy, ensuring it remains relevant and effective.

Best practices

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

OPS04-BP01 Identify key performance indicators

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

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

To effectively define workload KPIs:

1. **Start with business outcomes:** Before diving into metrics, understand the desired business outcomes. Is it increased sales, higher user engagement, or faster response times?
2. **Correlate technical metrics with business objectives:** Not all technical metrics have a direct impact on business outcomes. Identify those that do, but it's often more straightforward to identify an issue using a business KPI.
3. **Use Amazon CloudWatch:** Employ CloudWatch to define and monitor metrics that represent your KPIs.
4. **Regularly review and update KPIs:** As your workload and business evolve, keep your KPIs relevant.
5. **Involve stakeholders:** Involve both technical and business teams in defining and reviewing KPIs.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

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

Related documents:

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

Related videos:

- Developing an observability strategy

Related examples:

- One Observability Workshop

OPS04-BP02 Implement application telemetry

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

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

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

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Implementation steps

1. **Identify what data to collect:** Ascertain the essential metrics, logs, and traces that would offer substantial insights into your workload's health, performance, and behavior.
2. **Deploy the CloudWatch agent:** The CloudWatch agent is instrumental in procuring system and application metrics and logs from your workload and its underlying infrastructure. The

CloudWatch agent can also be used to collect OpenTelemetry or X-Ray traces and send them to X-Ray.

3. **Implement anomaly detection for logs and metrics:** Use CloudWatch Logs anomaly detection and CloudWatch Metrics anomaly detection to automatically identify unusual activities in your application's operations. These tools use machine learning algorithms to detect and alert on anomalies, which enanhces your monitoring capabilities and speeds up response time to potential disruptions or security threats. Set up these features to proactively manage application health and security.
4. **Secure sensitive log data:** Use Amazon CloudWatch Logs data protection to mask sensitive information within your logs. This feature helps maintain privacy and compliance through automatic detection and masking of sensitive data before it is accessed. Implement data masking to securely handle and protect sensitive details such as personally identifiable information (PII).
5. **Define and monitor business KPIs:** Establish custom metrics that align with your business outcomes.
6. **Instrument your application with AWS X-Ray:** In addition to deploying the CloudWatch agent, it's crucial to instrument your application to emit trace data. This process can provide further insights into your workload's behavior and performance.
7. **Standardize data collection across your application:** Standardize data collection practices across your entire application. Uniformity aids in correlating and analyzing data, providing a comprehensive view of your application's behavior.
8. **Implement cross-account observability:** Enhance monitoring efficiency across multiple AWS accounts with Amazon CloudWatch cross-account observability. With this feature, you can consolidate metrics, logs, and alarms from different accounts into a single view, which simplifies management and improves response times for identified issues across your organization's AWS environment.
9. **Analyze and act on the data:** Once data collection and normalization are in place, use Amazon CloudWatch for metrics and logs analysis, and AWS X-Ray for trace analysis. Such analysis can yield crucial insights into your workload's health, performance, and behavior, guiding your decision-making process.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS04-BP01 Define workload KPIs
- OPS04-BP03 Implement user activity telemetry
- OPS04-BP04 Implement dependency telemetry
- OPS04-BP05 Implement transaction traceability

Related documents:

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

Related videos:

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

Related examples:

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

OPS04-BP03 Implement user experience telemetry

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

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

To leverage RUM and synthetic transactions for user activity telemetry, AWS offers services like Amazon CloudWatch RUM and Amazon CloudWatch Synthetics. Metrics, logs, and traces, coupled

with user activity data, provide a comprehensive view of both the application's operational state and the user experience.

Implementation steps

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

Level of effort for the implementation plan: Medium

Resources

Related best practices:

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

Related documents:

- Amazon CloudWatch RUM Guide
- Amazon CloudWatch Synthetics Guide

Related videos:

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

Related examples:

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

OPS04-BP04 Implement dependency telemetry

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

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

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

Implementation guidance

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

Implementation steps

To implement dependency telemetry effectively:

1. **Identify external dependencies:** Collaborate with stakeholders to pinpoint the external dependencies your workload relies on. External dependencies can encompass services like external databases, third-party APIs, network connectivity routes to other environments, and DNS services. The first step towards effective dependency telemetry is being comprehensive in understanding what those dependencies are.
2. **Develop a monitoring strategy:** Once you have a clear picture of your external dependencies, architect a monitoring strategy tailored to them. This involves understanding the criticality of each dependency, its expected behavior, and any associated service-level agreements or targets (SLA or SLTs). Set up proactive alerts to notify you of status changes or performance deviations.
3. **Use network monitoring:** Use Internet Monitor and Network Monitor, which provide comprehensive insights into global internet and network conditions. These tools help you understand and respond to outages, disruptions, or performance degradations that affect your external dependencies.

4. **Stay informed with AWS Health:** AWS Health is the authoritative source of information about the health of your AWS Cloud resources. Use AWS Health to visualize and receive notifications about any current service events and upcoming changes, such as planned lifecycle events, so you can take steps to mitigate impacts.
    a. Create purpose-fit AWS Health event notifications to e-mail and chat channels through AWS User Notifications, and integrate programatically with your monitoring and alerting tools through Amazon EventBridge or the AWS Health API.
    b. Plan and track progress on health events that require action by integrating with change management or ITSM tools (like Jira or ServiceNow) that you may already use through Amazon EventBridge or the AWS Health API.
    c. If you use AWS Organizations, enable organization view for AWS Health to aggregate AWS Health events across accounts.
5. **Instrument your application with AWS X-Ray:** AWS X-Ray provides insights into how applications and their underlying dependencies are performing. By tracing requests from start to end, you can identify bottlenecks or failures in the external services or components your application relies on.
6. **Use Amazon DevOps Guru:** This machine learning-driven service identifies operational issues, predicts when critical issues might occur, and recommends specific actions to take. It's invaluable for gaining insights into dependencies and ensuring they're not the source of operational problems.
7. **Monitor regularly:** Continually monitor metrics and logs related to external dependencies. Set up alerts for unexpected behavior or degraded performance.
8. **Validate after changes:** Whenever there's an update or change in any of the external dependencies, validate their performance and check their alignment with your application's requirements.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Define workload KPIs
- OPS04-BP02 Implement application telemetry
- OPS04-BP03 Implement user activity telemetry

- OPS04-BP05 Implement transaction traceability
- OP08-BP04 Create actionable alerts

Related documents:

- Amazon Personal AWS Health Dashboard User Guide
- AWS Internet Monitor User Guide
- AWS X-Ray Developer Guide
- AWS DevOps Guru User Guide

Related videos:

- Visibility into how internet issues impact app performance
- Introduction to Amazon DevOps Guru
- Manage resource lifecycle events at scale with AWS Health

Related examples:

- AWS Health Aware
- Using Tag-Based Filtering to Manage AWS Health Monitoring and Alerting at Scale

OPS04-BP05 Implement distributed tracing

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

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

Common anti-patterns:

- Inconsistent instrumentation: Not all services in a distributed system are instrumented for tracing.

- Ignoring latency: Only focusing on errors and not considering the latency or gradual performance degradations.

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Implementation steps

To implement distributed tracing effectively:

1. **Adopt AWS X-Ray:** Integrate X-Ray into your application to gain insights into its behavior, understand its performance, and pinpoint bottlenecks. Utilize X-Ray Insights for automatic trace analysis.
2. **Instrument your services:** Verify that every service, from an AWS Lambda function to an EC2 instance, sends trace data. The more services you instrument, the clearer the end-to-end view.
3. **Incorporate CloudWatch Real User Monitoring and synthetic monitoring:** Integrate Real User Monitoring (RUM) and synthetic monitoring with X-Ray. This allows for capturing real-world user experiences and simulating user interactions to identify potential issues.
4. **Use the CloudWatch agent:** The agent can send traces from either X-Ray or OpenTelemetry, enhancing the depth of insights obtained.
5. **Use Amazon DevOps Guru:** DevOps Guru uses data from X-Ray, CloudWatch, AWS Config, and AWS CloudTrail to provide actionable recommendations.

6. **Analyze traces:** Regularly review the trace data to discern patterns, anomalies, or bottlenecks that might impact your application's performance.
7. **Set up alerts:** Configure alarms in CloudWatch for unusual patterns or extended latencies, allowing proactive issue addressing.
8. **Continuous improvement:** Revisit your tracing strategy as services are added or modified to capture all relevant data points.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

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

Related documents:

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

Related videos:

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

Related examples:

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

OPS05-BP01 Use version control

Use version control to activate tracking of changes and releases.

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Resources

Related best practices:

- OPS05-BP04 Use build and deployment management systems

Related videos:

- AWS re:Invent 2023 - How Lockheed Martin builds software faster, powered by DevSecOps
- AWS re:Invent 2023 - How GitHub operationalizes AI for team collaboration and productivity

OPS05-BP02 Test and validate changes

Every change deployed must be tested to avoid errors in production. This best practice is focused on testing changes from version control to artifact build. Besides application code changes, testing

should include infrastructure, configuration, security controls, and operations procedures. Testing takes many forms, from unit tests to software component analysis (SCA). Move tests further to the left in the software integration and delivery process results in higher certainty of artifact quality.

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

Desired outcome: Your software changes are tested before they are delivered. Developers have access to test results and validations. Your organization has a testing standard that applies to all software changes.

Common anti-patterns:

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

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

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

Implementation guidance

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

Use the power of generative AI with Amazon Q Developer to improve developer productivity and code quality. Amazon Q Developer includes generation of code suggestions (based on large language models), production of unit tests (including boundary conditions), and code security enhancements through detection and remediation of security vulnerabilities.

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?
    1. 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. Use Amazon Q Developer, a generative AI tool that can help create unit test cases (including boundary conditions), generate functions using code and comments, and implement well-known algorithms.
    b. Use Amazon CodeGuru Reviewer to test your application code for defects.
    c. You can use AWS CodeBuild to conduct tests on software artifacts.
    d. AWS CodePipeline can orchestrate your software tests into a pipeline.

Resources

Related best practices:

- OPS05-BP01 Use version control
- OPS05-BP06 Share design standards
- OPS05-BP07 Implement practices to improve code quality
- OPS05-BP10 Fully automate integration and deployment

Related documents:

- Adopt a test-driven development approach
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q Developer Center
- 10 ways to build applications faster with Amazon CodeWhisperer
- Looking beyond code coverage with Amazon CodeWhisperer
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- 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:

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)
- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- 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
- 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 Q Developer
- Amazon CodeGuru Reviewer
- AWS CodeBuild
- AWS CodePipeline

OPS05-BP03 Use configuration management systems

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

Static configuration management sets values when initializing a resource that are expected to remain consistent throughout the resource's lifetime. Dynamic configuration management sets values at initialization that can or are expected to change during the lifetime of a resource. For example, you could set a feature toggle to activate functionality in your code through a configuration change, or change the level of log detail during an incident.

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

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

Common anti-patterns:

- You manually update the web server configuration across your fleet and a number of servers become unresponsive due to update errors.

- You manually update your application server fleet over the course of many hours. The inconsistency in configuration during the change causes unexpected behaviors.
- Someone has updated your security groups and your web servers are no longer accessible. Without knowledge of what was changed you spend significant time investigating the issue extending your time to recovery.
- You push a pre-production configuration into production through CI/CD without validation. You expose users and customers to incorrect data and services.

Benefits of establishing this best practice: Adopting configuration management systems reduces the level of effort to make and track changes, and the frequency of errors caused by manual procedures. Configuration management systems provide assurances with regards to governance, compliance, and regulatory requirements.

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

Implementation guidance

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

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

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

Implementation steps

1. Identify configuration owners.
   a. Make configurations owners aware of any compliance, governance, or regulatory needs.
2. Identify configuration items and deliverables.
   a. Configuration items are all application and environmental configurations affected by a deployment within your CI/CD pipeline.

   b. Deliverables include success criteria, validation, and what to monitor.
3. Select tools for configuration management based on your business requirements and delivery pipeline.
4. Consider weighted deployments such as canary deployments for significant configuration changes to minimize the impact of incorrect configurations.
5. Integrate your configuration management into your CI/CD pipeline.
6. Validate all changes pushed.

Resources

Related best practices:

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

Related documents:

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

Related videos:

- AWS re:Invent 2022 - Proactive governance and compliance for AWS workloads

- AWS re:Invent 2020: Achieve compliance as code using AWS Config
- Manage and Deploy Application Configurations with AWS AppConfig

OPS05-BP04 Use build and deployment management systems

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

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Implementation steps

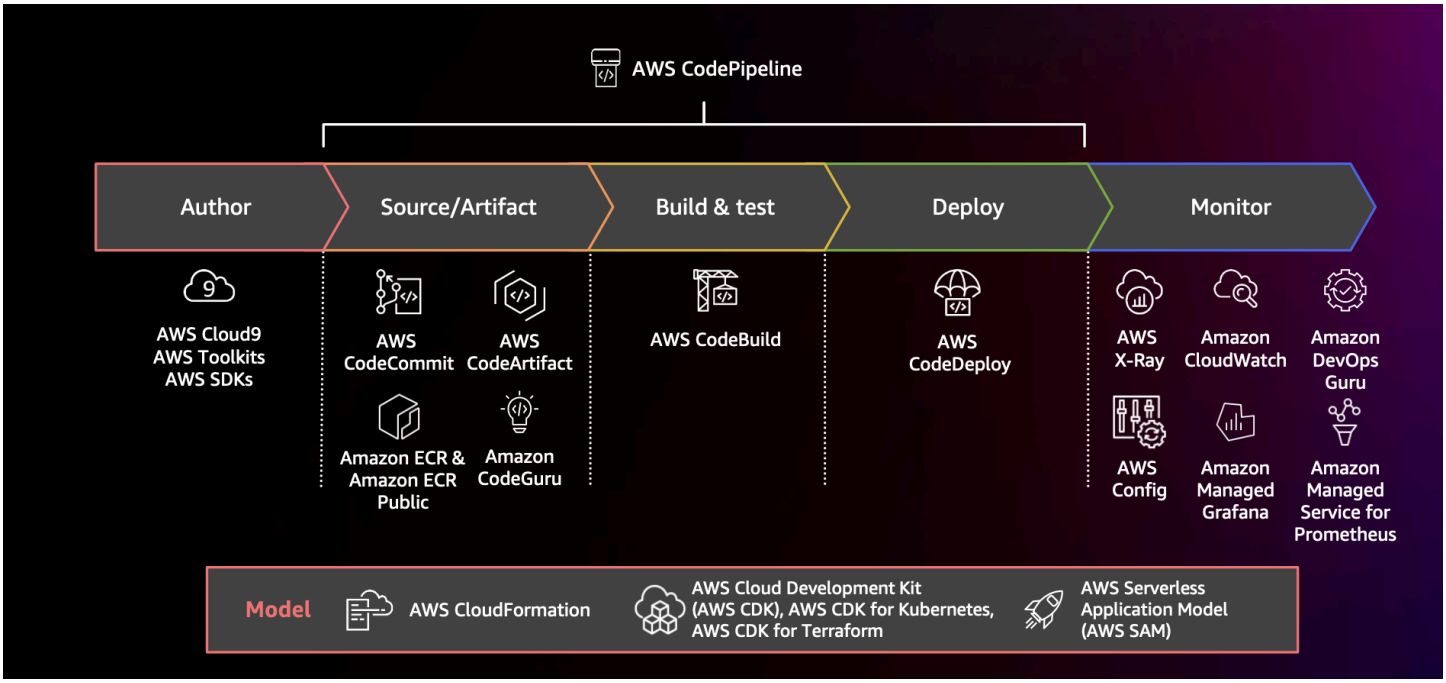


Diagram showing a CI/CD pipeline using AWS CodePipeline and related services

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

Resources

Related best practices:

- OPS06-BP04 Automate testing and rollback

Related documents:

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

Related videos:

- AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS

OPS05-BP05 Perform patch management

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

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

AWS Health is the authoritative source of information about planned lifecycle events and other action-required events that affect the health of your AWS Cloud resources. You should be aware of upcoming changes and updates that should be performed. Major planned lifecycle events are sent at least six months in advance.

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

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

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

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

Common anti-patterns:

- You are given a mandate to apply all new security patches within two hours resulting in multiple outages due to application incompatibility with patches.
- An unpatched library results in unintended consequences as unknown parties use vulnerabilities within it to access your workload.
- You patch the developer environments automatically without notifying the developers. You receive multiple complaints from the developers that their environment cease to operate as expected.
- You have not patched the commercial off-the-shelf software on a persistent instance. When you have an issue with the software and contact the vendor, they notify you that version is not supported and you have to patch to a specific level to receive any assistance.
- A recently released patch for the encryption software you used has significant performance improvements. Your unpatched system has performance issues that remain in place as a result of not patching.
- You are notified of a zero-day vulnerability requiring an emergency fix and you have to patch all your environments manually.
- You are not aware of critical actions needed to maintain your resources, such as mandatory version updates because you do not review upcoming planned lifecycle events and other information. You lose critical time for planning and execution, resulting in emergency changes for your teams and potential impact or unexpected downtime.

Benefits of establishing this best practice: By establishing a patch management process, including your criteria for patching and methodology for distribution across your environments, you can scale and report on patch levels. This provides assurances around security patching and ensure clear visibility on the status of known fixes being in place. This encourages adoption of desired features and capabilities, the rapid removal of issues, and sustained compliance with governance. Implement patch management systems and automation to reduce the level of effort to deploy patches and limit errors caused by manual processes.

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

Implementation guidance

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

Implementation steps

For Amazon EC2 Image Builder:

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

For Systems Manager Patch Manager:

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

Resources

Related best practices:

- OPS06-BP04 Automate testing and rollback

Related documents:

- What is Amazon EC2 Image Builder
- Create an image pipeline using the Amazon EC2 Image Builder
- Create a container image pipeline
- AWS Systems Manager Patch Manager
- Working with Patch Manager
- Working with patch compliance reports
- AWS Developer Tools

Related videos:

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

Related examples:

- AWS Systems Manager Patch Manager tutorials

OPS05-BP06 Share design standards

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

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Customer example

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

Implementation steps

1. Identify a cross-functional team that owns developing and updating design standards. This team should work with stakeholders across your organization to develop design standards, operating

procedures, checklists, guidance, and governance requirements. Document the design standards and share them within your organization.

   a. AWS Service Catalog can be used to create portfolios representing design standards using infrastructure as code. You can share portfolios across accounts.

2. Have a mechanism in place to keep design standards up-to-date as new best practices are identified.
3. If design standards are centrally enforced, have a process to request changes, updates, and exemptions.

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

Resources

Related best practices:

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

Related documents:

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

Related videos:

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

Related examples:

- AWS Service Catalog Reference Architecture
- AWS Service Catalog Workshop

Related services:

- AWS Service Catalog

OPS05-BP07 Implement practices to improve code quality

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

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

Common anti-patterns:

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

Benefits of establishing this best practice: By adopting practices to improve code quality, you can help minimize issues introduced to production. Code quality best practices include pair programming, code reviews, and implementation of AI productivity tools.

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.

Use the power of generative AI with Amazon Q Developer to improve developer productivity and code quality. Amazon Q Developer includes generation of code suggestions (based on large language models), production of unit tests (including boundary conditions), and code security enhancements through detection and remediation of security vulnerabilities.

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. Use Amazon Q Developer, a generative AI tool that can help create unit test cases (including boundary conditions), generate functions using code and comments, implement well-known algorithms, detect security policy violations and vulnerabilities in your code, detect secrets, scan infrastructure as code (IaC), document code, and learn third-party code libraries more quickly.
   b. Amazon CodeGuru Reviewer can provide programming recommendations for Java and Python code using machine learning.

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-BP02 Test and validate changes
- OPS05-BP06 Share design standards

Related documents:

- Adopt a test-driven development approach
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q Developer Center
- 10 ways to build applications faster with Amazon CodeWhisperer
- Looking beyond code coverage with Amazon CodeWhisperer
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- 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:

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)
- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- 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 Q Developer
- Amazon CodeGuru Reviewer
- Amazon CodeGuru Profiler

OPS05-BP08 Use multiple environments

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

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

1. Your organization does this through the establishment of a landing zone, which provides governance, controls, account automations, networking, security, and operational observability. Manage these landing zone capabilities by using multiple environments. A common example is a sandbox organization for developing and testing changes to an AWS Control Tower-based landing zone, which includes AWS IAM Identity Center and policies such as service control policies (SCPs). All of these elements can significantly impact the access to and operation of AWS accounts within the landing zone.
2. In addition to these services, your teams extend the landing zones capabilites with solutions published by AWS and AWS partners or as custom solutions developed within your organization. Examples of solutions published by AWS include Customizations for AWS Control Tower (CfCT) and AWS Control Tower Account Factory for Terraform (AFT).
3. Your organization applies the same principles of testing, promoting code, and policy changes for the landing zone through environments on your path to production. This strategy provides a stable and secure landing zone environment for your application and workload teams.

Common anti-patterns:

- You are performing development in a shared development environment and another developer overwrites your code changes.

- The restrictive security controls on your shared development environment are preventing you from experimenting with new services and features.
- You perform load testing on your production systems and cause an outage for your users.
- A critical error resulting in data loss has occurred in production. In your production environment, you attempt to recreate the conditions that lead to the data loss so that you can identify how it happened and prevent it from happening again. To prevent further data loss during testing, you are forced to make the application unavailable to your users.
- You are operating a multi-tenant service and are unable to support a customer request for a dedicated environment.
- You may not always test, but when you do, you test in your production environment.
- You believe that the simplicity of a single environment overrides the scope of impact of changes within the environment.
- You upgrade a key landing zone capability, but the change impairs your team's ability to vend accounts for either new projects or your existing workloads.
- You apply new controls to your AWS accounts, but the change impacts your workload team's ability to deploy changes within their AWS accounts.

Benefits of establishing this best practice: When you deploy multiple environments, you can support multiple simultaneous development, testing, and production environments without creating conflicts between developers or user communities. For complex capabilities such as landing zones, it significantly reduces the risk of changes, simplifies the improvement process, and reduces the risk of critical updates to the environment. Organizations that use landing zones naturally benefit from multi-accounts in their AWS environment, with account structure, governance, network, and security configurations. Over time, as your organization grows, the landing zone can evolve to secure and organize your workloads and resources.

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

Implementation guidance

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

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

Teams such as platform engineering, networking, and security operations often manage capabilies at the organization level with distinct requirements. A separation of accounts alone is insufficient to provide and maintain separate environments for experimentation, development, and testing. In such cases, create separate instances of AWS Organizations.

Resources

Related documents:

- Instance Scheduler on AWS
- What is AWS CloudFormation?
- Organizing Your AWS Environment Using Multiple Accounts - Multiple organizations - Test changes to your overall AWS environment
- AWS Control Tower Guide

OPS05-BP09 Make frequent, small, reversible changes

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Resources

Related best practices:

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

Related documents:

- Implementing Microservices on AWS
- Microservices - Observability

OPS05-BP10 Fully automate integration and deployment

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

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

Desired outcome: Developers use tools to deliver code and promote through to production. Developers do not have to log into the AWS Management Console to deliver updates. There is a full audit trail of change and configuration, meeting the needs of governance and compliance.

Processes are repeatable and are standardized across teams. Developers are free to focus on development and code pushes, increasing productivity.

Common anti-patterns:

- On Friday, you finish authoring the new code for your feature branch. On Monday, after running your code quality test scripts and each of your unit tests scripts, you check in your code for the next scheduled release.
- You are assigned to code a fix for a critical issue impacting a large number of customers in production. After testing the fix, you commit your code and email change management to request approval to deploy it to production.
- As a developer, you log into the AWS Management Console to create a new development environment using non-standard methods and systems.

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

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

Implementation guidance

You use build and deployment management systems to track and implement change, to reduce errors caused by manual processes, and reduce the level of effort. Fully automate the integration and deployment pipeline from code check-in through build, testing, deployment, and validation. This reduces lead time, encourages increased frequency of change, reduces the level of effort, increases the speed to market, results in increased productivity, and increases the security of your code as you promote through to production.

Resources

Related best practices:

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

Related documents:

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

Related videos:

- AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS

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
- OPS06-BP02 Test deployments
- OPS06-BP03 Employ safe deployment strategies
- OPS06-BP04 Automate testing and rollback

OPS06-BP01 Plan for unsuccessful changes

Plan to revert to a known good state, or remediate in the production environment if the deployment causes an undesired outcome. Having a policy to establish such a plan helps all teams develop strategies to recover from failed changes. Some example strategies are deployment and rollback steps, change policies, feature flags, traffic isolation, and traffic shifting. A single release may include multiple related component changes. The strategy should provide the ability to withstand or recover from a failure of any component change.

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

shortening the potential downtime caused by a failed change and increased the flexibility and efficiency of recovery times.

Common anti-patterns:

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

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

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

Implementation guidance

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

Implementation steps

1. Document the policies that require teams to have effective plans to reverse changes within a specified period.
   a. Policies should specify when a fix-forward situation is allowed.
   b. Require a documented rollback plan to be accessible by all involved.
   c. Specify the requirements to rollback (for example, when it is found that unauthorized changes have been deployed).
2. Analyze the level of impact of all changes related to each component of a workload.
   a. Allow repeatable changes to be standardized, templated, and preauthorized if they follow a consistent workflow that enforces change policies.
   b. Reduce the potential impact of any change by making the size of the change smaller so recovery takes less time and causes less business impact.
   c. Ensure rollback procedures revert code to the known good state to avoid incidents where possible.
3. Integrate tools and workflows to enforce your policies programatically.
4. Make data about changes visible to other workload owners to improve the speed of diagnosis of any failed change that cannot be rolled back.
   a. Measure success of this practice using visible change data and identify iterative improvements.
5. Use monitoring tools to verify the success or failure of a deployment to speed up decision-making on rolling back.
6. Measure your duration of outage during an unsuccessful change to continually improve your recovery plans.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS06-BP04 Automate testing and rollback

Related documents:

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

Related videos:

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

OPS06-BP02 Test deployments

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

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

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

Common anti-patterns:

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

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

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

Implementation guidance

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

Customer example

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

Implementation steps

1. Perform pre-install checks to mirror the pre-production environment to production.
   a. Use drift detection to detect when resources have been changed outside of AWS CloudFormation.
   b. Use change sets to validate that the intent of a stack update matches the actions that AWS CloudFormation takes when the change set is initiated.
2. This triggers a manual approval step in AWS CodePipeline to authorize the deployment to the pre-production environment.
3. Use deployment configurations such as AWS CodeDeploy AppSpec files to define deployment and validation steps.

4. Where applicable, integrate AWS CodeDeploy with other AWS services or integrate AWS CodeDeploy with partner product and services.
5. Monitor deployments using Amazon CloudWatch, AWS CloudTrail, and Amazon SNS event notifications.
6. Perform post-deployment automated testing, including functional, security, regression, integration, and load testing.
7. Troubleshoot deployment issues.
8. Successful validation of preceding steps should initiate a manual approval workflow to authorize deployment to production.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS05-BP02 Test and validate changes

Related documents:

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

Related videos:

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

Related examples:

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

OPS06-BP03 Employ safe deployment strategies

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

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

Common anti-patterns:

- You deploy an unsuccessful change to all of production all at once. As a result, all customers are impacted simultaneously.
- A defect introduced in a simultaneous deployment to all systems requires an emergency release. Correcting it for all customers takes several days.
- Managing production release requires planning and participation of several teams. This puts constraints on your ability to frequently update features for your customers.
- You perform a mutable deployment by modifying your existing systems. After discovering that the change was unsuccessful, you are forced to modify the systems again to restore the old version, extending your time to recovery.

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

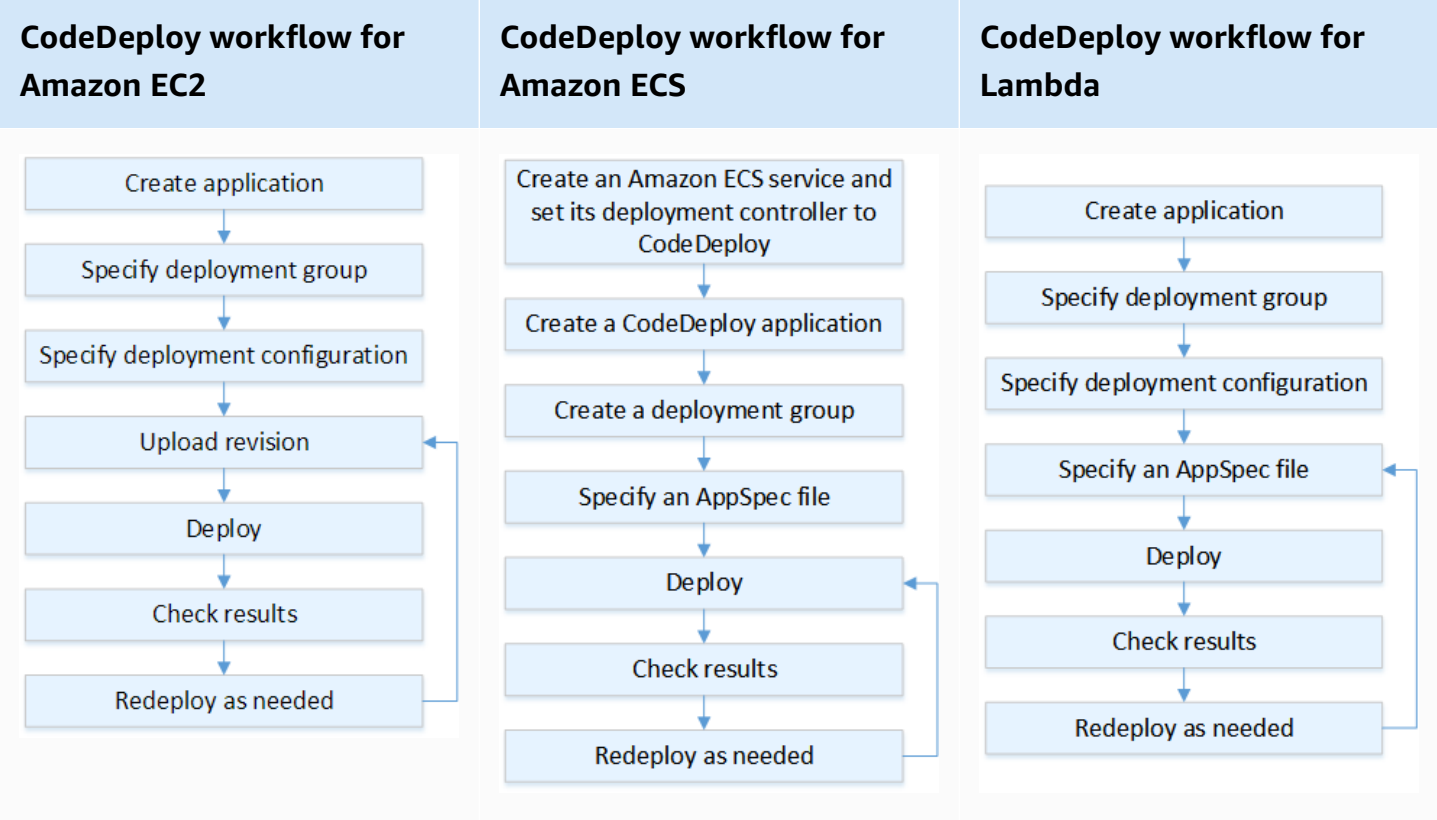
Level of risk exposed if this best practice is not established: Medium

Implementation guidance

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

Customer example

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



Implementation steps

1. Use an approval workflow to initiate the sequence of production roll-out steps upon promotion to production .
2. Use an automated deployment system such as AWS CodeDeploy. AWS CodeDeploy deployment options include in-place deployments for EC2/On-Premises and blue/green deployments for EC2/On-Premises, AWS Lambda, and Amazon ECS (see the preceding workflow diagram).
   a. Where applicable, integrate AWS CodeDeploy with other AWS services or integrate AWS CodeDeploy with partner product and services.
3. Use blue/green deployments for databases such as Amazon Aurora and Amazon RDS.
4. Monitor deployments using Amazon CloudWatch, AWS CloudTrail, and Amazon Simple Notification Service (Amazon SNS) event notifications.

5. Perform post-deployment automated testing including functional, security, regression, integration, and any load tests.
6. Troubleshoot deployment issues.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

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

Related documents:

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

Related videos:

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

Related examples:

- Try a Sample Blue/Green Deployment in AWS CodeDeploy
- Workshop | Building CI/CD pipelines for Lambda canary deployments using AWS CDK
- Workshop | Building your first DevOps Blue/Green pipeline with Amazon ECS
- Workshop | Building your first DevOps Blue/Green pipeline with Amazon EKS
- Workshop | EKS GitOps with ArgoCD
- Workshop | CI/CD on AWS Workshop
- Implementing cross-account CI/CD with AWS SAM for container-based Lambda functions

OPS06-BP04 Automate testing and rollback

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

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

Common anti-patterns:

- Your systems are not architected in a way that allows them to be updated with smaller releases. As a result, you have difficulty in reversing those bulk changes during a failed deployment.
- Your deployment process consists of a series of manual steps. After you deploy changes to your workload, you start post-deployment testing. After testing, you realize that your workload is inoperable and customers are disconnected. You then begin rolling back to the previous version. All of these manual steps delay overall system recovery and cause a prolonged impact to your customers.

- You spent time developing automated test cases for functionality that is not frequently used in your application, minimizing the return on investment in your automated testing capability.
- Your release is comprised of application, infrastructure, patches and configuration updates that are independent from one another. However, you have a single CI/CD pipeline that delivers all changes at once. A failure in one component forces you to revert all changes, making your rollback complex and inefficient.
- Your team completes the coding work in sprint one and begins sprint two work, but your plan did not include testing until sprint three. As a result, automated tests revealed defects from sprint one that had to be resolved before testing of sprint two deliverables could be started and the entire release is delayed, devaluing your automated testing.
- Your automated regression test cases for the production release are complete, but you are not monitoring workload health. Since you have no visibility into whether or not the service has restarted, you are not sure if rollback is needed or if it has already occurred.

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

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

Implementation guidance

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

Implementation steps

1. Establish a testing lifecycle for your development lifecycle that defines each stage of the testing process from requirements planning to test case development, tool configuration, automated testing, and test case closure.
   a. Create a workload-specific testing approach from your overall test strategy.
   b. Consider a continuous testing strategy where appropriate throughout the development lifecycle.
2. Select automated tools for testing and rollback based on your business requirements and pipeline investments.
3. Decide which test cases you wish to automate and which should be performed manually. These can be defined based on business value priority of the feature being tested. Align all team members to this plan and verify accountability for performing manual tests.
   a. Apply automated testing capabilities to specific test cases that make sense for automation, such as repeatable or frequently run cases, those that require repetitive tasks, or those that are required across multiple configurations.
   b. Define test automation scripts as well as the success criteria in the automation tool so continued workflow automation can be initiated when specific cases fail.
   c. Define specific failure criteria for automated rollback.
4. Prioritize test automation to drive consistent results with thorough test case development where complexity and human interaction have a higher risk of failure.
5. Integrate your automated testing and rollback tools into your CI/CD pipeline.
   a. Develop clear success criteria for your changes.
   b. Monitor and observe to detect these criteria and automatically reverse changes when specific rollback criteria are met.
6. Perform different types of automated production testing, such as:
   a. A/B testing to show results in comparison to the current version between two user testing groups.
   b. Canary testing that allows you to roll out your change to a subset of users before releasing it to all.
   c. Feature-flag testing which allows a single feature of the new version at a time to be flagged on and off from outside the application so that each new feature can be validated one at a time.
   d. Regression testing to verify new functionality with existing interrelated components.

7. Monitor the operational aspects of the application, transactions, and interactions with other applications and components. Develop reports to show success of changes by workload so that you can identify what parts of the automation and workflow can be further optimized.
   a. Develop test result reports that help you make quick decisions on whether or not rollback procedures should be invoked.
   b. Implement a strategy that allows for automated rollback based upon pre-defined failure conditions that result from one or more of your test methods.
8. Develop your automated test cases to allow for reusability across future repeatable changes.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

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

Related documents:

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

Related examples:

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

Related videos:

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

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

OPS07-BP01 Ensure personnel capability

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

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

Implementation steps

1. Assign an adequate number of personnel to operate and support your workload, including on-call duties, security issues, and lifecycle events, such as end of support and certificate rotation tasks.
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. Perform the following on a regular basis:

- Evaluate team size and skills as operating conditions and the workload change.
- Adjust team size and skills to match operational requirements.
- Verify ability and capacity to address planned lifecycle events, unplanned security, and operational notifications through AWS Health.

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

Resources

Related best practices:

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

Related documents:

- AWS Events and Webinars
- AWS Training and Certification

OPS07-BP02 Ensure a consistent review of operational readiness

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

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

Run ORRs before a workload launches to general availability and then throughout the software development lifecycle. Running the ORR before launch increases your ability to operate the

workload safely. Periodically re-run your ORR on the workload to catch any drift from best practices. You can have ORR checklists for new services launches and ORRs for periodic reviews. This helps keep you up to date on new best practices that arise and incorporate lessons learned from post-incident analysis. As your use of the cloud matures, you can build ORR requirements into your architecture as defaults.

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Customer example

After several production incidents, AnyCompany Retail decided to implement an ORR process. They built a checklist composed of best practices, governance and compliance requirements, and lessons learned from outages. New workloads conduct ORRs before they launch. Every workload conducts a yearly ORR with a subset of best practices to incorporate new best practices and requirements that are added to the ORR checklist. Over time, AnyCompany Retail used AWS Config to detect some best practices, speeding up the ORR process.

Implementation steps

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

1. Gather the key stakeholders together, including representatives from security, operations, and development.
2. Have each stakeholder provide at least one requirement. For the first iteration, try to limit the number of items to thirty or less.
    - Appendix B: Example ORR questions from the Operational Readiness Reviews (ORR) whitepaper contains sample questions that you can use to get started.
3. Collect your requirements into a spreadsheet.
    - You can use custom lenses in the AWS Well-Architected Tool to develop your ORR and share them across your accounts and AWS Organization.
4. Identify one workload to conduct the ORR on. A pre-launch workload or an internal workload is ideal.
5. Run through the ORR checklist and take note of any discoveries made. Discoveries might be acceptable if a mitigation is in place. For any discovery that lacks a mitigation, add those to your backlog of items and implement them before launch.
6. Continue to add best practices and requirements to your ORR checklist over time.

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

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

Resources

Related best practices:

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

Related documents:

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

Related videos:

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

Related examples:

- Sample Operational Readiness Review (ORR) Lens

Related services:

- AWS Config
- AWS Control Tower
- AWS Security Hub
- 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 (version control system) and updated frequently. For example, your runbooks provide capabilities for your teams to monitor, communicate, and respond to AWS Health events for critical accounts during application alarms, operational issues, and planned lifecycle events.

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. AWS Systems Manager Automation also provides a low-code visual design experience to create automation runbooks more easily.

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
- OPS07-BP04 Use playbooks to investigate issues
- OPS10-BP01 Use a process for event, incident, and problem management
- OPS10-BP02 Have a process per alert
- OPS11-BP04 Perform knowledge management

Related documents:

- AWS Well-Architected Framework: Concepts: Runbook development
- 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
- How to automate IT Operations on AWS | Amazon Web Services
- Integrate Scripts into AWS Systems Manager

Related examples:

- Well-Architected Labs: Automating operations with Playbooks and Runbooks
- AWS Blog Post: Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- 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

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
- OPS07-BP03 Use runbooks to perform procedures
- OPS10-BP01 Use a process for event, incident, and problem management
- OPS10-BP02 Have a process per alert
- OPS11-BP04 Perform knowledge management

Related documents:

- AWS Well-Architected Framework: Concepts: Playbook development
- 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

Related services:

- AWS Systems Manager Automation
- AWS Systems Manager Incident Manager

OPS07-BP05 Make informed decisions to deploy systems and changes

Have processes in place for successful and unsuccessful changes to your workload. A pre-mortem is an exercise where a team simulates a failure to develop mitigation strategies. Use pre-mortems to anticipate failure and create procedures where appropriate. Evaluate the benefits and risks of deploying changes to your workload. Verify that all changes comply with governance.

Desired outcome:

- You make informed decisions when deploying changes to your workload.
- Changes comply with governance.

Common anti-patterns:

- Deploying a change to our workload without a process to handle a failed deployment.
- Making changes to your production environment that are out of compliance with governance requirements.
- Deploying a new version of your workload without establishing a baseline for resource utilization.

Benefits of establishing this best practice:

- You are prepared for unsuccessful changes to your workload.
- Changes to your workload are compliant with governance policies.

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

Implementation guidance

Use pre-mortems to develop processes for unsuccessful changes. Document your processes for unsuccessful changes. Ensure that all changes comply with governance. Evaluate the benefits and risks to deploying changes to your workload.

Customer example

AnyCompany Retail regularly conducts pre-mortems to validate their processes for unsuccessful changes. They document their processes in a shared Wiki and update it frequently. All changes comply with governance requirements.

Implementation steps

1. Make informed decisions when deploying changes to your workload. Establish and review criteria for a successful deployment. Develop scenarios or criteria that would initiate a rollback of a change. Weigh the benefits of deploying changes against the risks of an unsuccessful change.
2. Verify that all changes comply with governance policies.
3. Use pre-mortems to plan for unsuccessful changes and document mitigation strategies. Run a table-top exercise to model an unsuccessful change and validate roll-back procedures.

Level of effort for the implementation plan: Moderate. Implementing a practice of pre-mortems requires coordination and effort from stakeholders across your organization

Resources

Related best practices:

- OPS01-BP03 Evaluate governance requirements - Governance requirements are a key factor in determining whether to deploy a change.
- OPS06-BP01 Plan for unsuccessful changes - Establish plans to mitigate a failed deployment and use pre-mortems to validate them.
- OPS06-BP02 Test deployments - Every software change should be properly tested before deployment in order to reduce defects in production.
- OPS07-BP01 Ensure personnel capability - Having enough trained personnel to support the workload is essential to making an informed decision to deploy a system change.

Related documents:

- Amazon Web Services: Risk and Compliance
- AWS Shared Responsibility Model
- Governance in the AWS Cloud: The Right Balance Between Agility and Safety

OPS07-BP06 Create support plans for production workloads

Enable support for any software and services that your production workload relies on. Select an appropriate support level to meet your production service-level needs. Support plans for these dependencies are necessary in case there is a service disruption or software issue. Document support plans and how to request support for all service and software vendors. Implement mechanisms that verify that support points of contacts are kept up to date.

Desired outcome:

- Implement support plans for software and services that production workloads rely on.
- Choose an appropriate support plan based on service-level needs.
- Document the support plans, support levels, and how to request support.

Common anti-patterns:

- You have no support plan for a critical software vendor. Your workload is impacted by them and you can do nothing to expedite a fix or get timely updates from the vendor.
- A developer that was the primary point of contact for a software vendor left the company. You are not able to reach the vendor support directly. You must spend time researching and navigating generic contact systems, increasing the time required to respond when needed.
- A production outage occurs with a software vendor. There is no documentation on how to file a support case.

Benefits of establishing this best practice:

- With the appropriate support level, you are able to get a response in the time frame necessary to meet service-level needs.
- As a supported customer you can escalate if there are production issues.
- Software and services vendors can assist in troubleshooting during an incident.

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

Implementation guidance

Enable support plans for any software and services vendors that your production workload relies on. Set up appropriate support plans to meet service-level needs. For AWS customers, this means activating AWS Business Support or greater on any accounts where you have production workloads. Meet with support vendors on a regular cadence to get updates about support offerings, processes, and contacts. Document how to request support from software and services vendors, including how to escalate if there is an outage. Implement mechanisms to keep support contacts up to date.

Customer example

At AnyCompany Retail, all commercial software and services dependencies have support plans. For example, they have AWS Enterprise Support activated on all accounts with production workloads. Any developer can raise a support case when there is an issue. There is a wiki page with information on how to request support, whom to notify, and best practices for expediting a case.

Implementation steps

1. Work with stakeholders in your organization to identify software and services vendors that your workload relies on. Document these dependencies.
2. Determine service-level needs for your workload. Select a support plan that aligns with them.
3. For commercial software and services, establish a support plan with the vendors.
    a. Subscribing to AWS Business Support or greater for all production accounts provides faster response time from AWS Support and strongly recommended. If you don't have premium support, you must have an action plan to handle issues, which require help from AWS Support. AWS Support provides a mix of tools and technology, people, and programs designed to proactively help you optimize performance, lower costs, and innovate faster. In addition, AWS Business Support provides additional benefits, including API access to AWS Trusted Advisor and AWS Health for programmatic integration with your systems, alongside other access methods like the AWS Management Console and Amazon EventBridge channels.
4. Document the support plan in your knowledge management tool. Include how to request support, who to notify if a support case is filed, and how to escalate during an incident. A wiki is a good mechanism to allow anyone to make necessary updates to documentation when they become aware of changes to support processes or contacts.

Level of effort for the implementation plan: Low. Most software and services vendors offer opt-in support plans. Documenting and sharing support best practices on your knowledge management system verifies that your team knows what to do when there is a production issue.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners

Related documents:

- AWS Support Plans

Related services:

- AWS Business Support
- AWS Enterprise Support

Operate

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

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

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

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

In AWS, you can generate dashboard views of your metrics collected from workloads and natively from AWS. You can leverage CloudWatch or third-party applications to aggregate and present

business, workload, and operations level views of operations activities. AWS provides workload insights through logging capabilities including AWS X-Ray, CloudWatch, CloudTrail, and VPC Flow Logs to identify workload issues in support of root cause analysis and remediation.

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

Topics

- Utilizing workload observability
- Understanding operational health
- Responding to events

Utilizing workload observability

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

Observability allows you to focus on meaningful data and understand your workload's interactions and output. By concentrating on essential insights and eliminating unnecessary data, you maintain a straightforward approach to understanding workload performance.

It's essential not only to collect data but also to interpret it correctly. Define clear baselines, set appropriate alert thresholds, and actively monitor for any deviations. A shift in a key metric, especially when correlated with other data, can pinpoint specific problem areas.

With observability, you're better equipped to foresee and address potential challenges, ensuring that your workload operates smoothly and meets business needs.

AWS offers specific tools like Amazon CloudWatch for monitoring and logging, and AWS X-Ray for distributed tracing. These services integrate effortlessly with various AWS resources, allowing for efficient data collection, setting up alerts based on predefined thresholds, and presenting data on dashboards for easy interpretation. By leveraging these insights, you can make well-informed, data-driven decisions that align with your operational goals.

Best practices

- OPS08-BP01 Analyze workload metrics

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

OPS08-BP01 Analyze workload metrics

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

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

Common anti-patterns:

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

Benefits of establishing this best practice:

- Enhanced understanding of the correlation between technical performance and business outcomes.
- Improved decision-making process informed by real-time data.
- Proactive identification and mitigation of issues before they affect business outcomes.

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

Implementation guidance

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

Implementation steps

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

Level of effort for the Implementation Plan: Medium

Resources

Related best practices:

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

Related documents:

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

Related videos:

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

Related examples:

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

OPS08-BP02 Analyze workload logs

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

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Implementation steps

1. **Set up CloudWatch Logs**: Configure applications and services to send logs to CloudWatch Logs.
2. **Use log anomaly detection:** Utilize Amazon CloudWatch Logs anomaly detection to automatically identify and alert on unusual log patterns. This tool helps you proactively manage anomalies in your logs and detect potential issues early.
3. **Set up CloudWatch Logs Insights**: Use CloudWatch Logs Insights to interactively search and analyze your log data.
   a. Craft queries to extract patterns, visualize log data, and derive actionable insights.
   b. Use CloudWatch Logs Insights pattern analysis to analyze and visualize frequent log patterns. This feature helps you understand common operational trends and potential outliers in your log data.
   c. Use CloudWatch Logs compare (diff) to perform differential analysis between different time periods or across different log groups. Use this capability to pinpoint changes and assess their impacts on your system's performance or behavior.
4. **Monitor logs in real-time with Live Tail:** Use Amazon CloudWatch Logs Live Tail to view log data in real-time. You can actively monitor your application's operational activities as they occur, which provides immediate visibility into system performance and potential issues.
5. **Leverage Contributor Insights**: Use CloudWatch Contributor Insights to identify top talkers in high cardinality dimensions like IP addresses or user-agents.
6. **Implement CloudWatch Logs metric filters**: Configure CloudWatch Logs metric filters to convert log data into actionable metrics. This allows you to set alarms or further analyze patterns.

7. **Implement CloudWatch cross-account observability:** Monitor and troubleshoot applications that span multiple accounts within a Region.
8. **Regular review and refinement**: Periodically review your log analysis strategies to capture all relevant information and continually optimize application performance.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

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

Related documents:

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

Related videos:

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

Related examples:

- CloudWatch Logs Sample Queries
- One Observability Workshop

OPS08-BP03 Analyze workload traces

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

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

Common anti-patterns:

- Overlooking trace data, relying solely on logs and metrics.
- Not correlating trace data with associated logs.
- Ignoring the metrics derived from traces, such as latency and fault rates.

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Implementation steps

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

1. **Integrate AWS X-Ray**: Ensure X-Ray is integrated with your applications to capture trace data.
2. **Analyze X-Ray metrics**: Delve into metrics derived from X-Ray traces, such as latency, request rates, fault rates, and response time distributions, using the service map to monitor application health.

3. **Use ServiceLens**: Leverage the ServiceLens map for enhanced observability of your services and applications. This allows for integrated viewing of traces, metrics, logs, alarms, and other health information.
4. **Enable X-Ray Insights**:
    a. Turn on X-Ray Insights for automated anomaly detection in traces.
    b. Examine insights to pinpoint patterns and ascertain root causes, such as increased fault rates or latencies.
    c. Consult the insights timeline for a chronological analysis of detected issues.
5. **Use X-Ray Analytics**: X-Ray Analytics allows you to thoroughly explore trace data, pinpoint patterns, and extract insights.
6. **Use groups in X-Ray**: Create groups in X-Ray to filter traces based on criteria such as high latency, allowing for more targeted analysis.
7. **Incorporate Amazon DevOps Guru**: Engage Amazon DevOps Guru to benefit from machine learning models pinpointing operational anomalies in traces.
8. **Use CloudWatch Synthetics**: Use CloudWatch Synthetics to create canaries for continually monitoring your endpoints and workflows. These canaries can integrate with X-Ray to provide trace data for in-depth analysis of the applications being tested.
9. **Use Real User Monitoring (RUM)**: With AWS X-Ray and CloudWatch RUM, you can analyze and debug the request path starting from end users of your application through downstream AWS managed services. This helps you identify latency trends and errors that impact your end users.
10. **Correlate with logs**: Correlate trace data with related logs within the X-Ray trace view for a granular perspective on application behavior. This allows you to view log events directly associated with traced transactions.
11. **Implement CloudWatch cross-account observability:** Monitor and troubleshoot applications that span multiple accounts within a Region.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

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

Related documents:

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

Related videos:

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

Related examples:

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

OPS08-BP04 Create actionable alerts

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

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

Common anti-patterns:

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

Benefits of establishing this best practice:

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

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

Implementation guidance

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

Implementation steps

1. **Determine key performance indicators (KPIs)**: Identify your application's KPIs. Alerts should be tied to these KPIs to reflect the business impact accurately.
2. **Implement anomaly detection**:
   - **Use Amazon CloudWatch anomaly detection**: Set up Amazon CloudWatch anomaly detection to automatically detect unusual patterns, which helps you only generate alerts for genuine anomalies.
   - **Use AWS X-Ray Insights**:
     a. Set up X-Ray Insights to detect anomalies in trace data.
     b. Configure notifications for X-Ray Insights to be alerted on detected issues.
   - **Integrate with Amazon DevOps Guru**:
     a. Leverage Amazon DevOps Guru for its machine learning capabilities in detecting operational anomalies with existing data.
     b. Navigate to the notification settings in DevOps Guru to set up anomaly alerts.
3. **Implement actionable alerts**: Design alerts that provide adequate information for immediate action.
   1. Monitor AWS Health events with Amazon EventBridge rules, or integrate programatically with the AWS Health API to automate actions when you receive AWS Health events. These can be general actions, such as sending all planned lifecycle event messages to a chat interface, or specific actions, such as the initiation of a workflow in an IT service management tool.

4. **Reduce alert fatigue**: Minimize non-critical alerts. When teams are overwhelmed with numerous insignificant alerts, they can lose oversight of critical issues, which diminishes the overall effectiveness of the alert mechanism.
5. **Set up composite alarms**: Use Amazon CloudWatch composite alarms to consolidate multiple alarms.
6. **Integrate with alert tools**: Incorporate tools like Ops Genie and PagerDuty.
7. **Engage Amazon Q Developer in chat applications**: Integrate Amazon Q Developer in chat applications to relay alerts to Amazon Chime, Microsoft Teams, and Slack.
8. **Alert based on logs**: Use log metric filters in CloudWatch to create alarms based on specific log events.
9. **Review and iterate**: Regularly revisit and refine alert configurations.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

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

Related documents:

- Using Amazon CloudWatch alarms
- Create a composite alarm
- Create a CloudWatch alarm based on anomaly detection
- DevOps Guru Notifications
- X-ray insights notifications

- Monitor, operate, and troubleshoot your AWS resources with interactive ChatOps
- Amazon CloudWatch Integration Guide | PagerDuty
- Integrate Opsgenie with Amazon CloudWatch

Related videos:

- Create Composite Alarms in Amazon CloudWatch
- Amazon Q Developer in chat applications Overview
- AWS On Air ft. Mutative Commands in Amazon Q Developer in chat applications

Related examples:

- Alarms, incident management, and remediation in the cloud with Amazon CloudWatch
- Tutorial: Creating an Amazon EventBridge rule that sends notifications to Amazon Q Developer in chat applications
- One Observability Workshop

OPS08-BP05 Create dashboards

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of this best practice:

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

Level of risk if this best practice isn't established: Medium

Implementation guidance

Business-centric dashboards

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

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

Amazon CloudWatch Dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those resources that are spread across different AWS Regions and accounts.

Implementation steps

1. **Create a basic dashboard:** Create a new dashboard in CloudWatch, giving it a descriptive name.
2. **Use Markdown widgets:** Before diving into the metrics, use Markdown widgets to add textual context at the top of your dashboard. This should explain what the dashboard covers, the significance of the represented metrics, and can also contain links to other dashboards and troubleshooting tools.
3. **Create dashboard variables:** Incorporate dashboard variables where appropriate to allow for dynamic and flexible dashboard views.

4. **Create metrics widgets:** Add metric widgets to visualize various metrics your application emits, tailoring these widgets to effectively represent system health and business outcomes.
5. **Log Insights queries:** Utilize CloudWatch Log Insights to derive actionable metrics from your logs and display these insights on your dashboard.
6. **Set up alarms:** Integrate CloudWatch Alarms into your dashboard for a quick view of any metrics breaching their thresholds.
7. **Use Contributor Insights:** Incorporate CloudWatch Contributor Insights to analyze high-cardinality fields and get a clearer understanding of your resource's top contributors.
8. **Design custom widgets:** For specific needs not met by standard widgets, consider creating custom widgets. These can pull from various data sources or represent data in unique ways.
9. **Use AWS Health:** AWS Health is the authoritative source of information about the health of your AWS Cloud resources. Use AWS Health Dashboard out of the box, or use AWS Health data in your own dashboards and tools so you have the right information available to make informed decisions.
10. **Iterate and refine:** As your application evolves, regularly revisit your dashboard to ensure its relevance.

Resources

Related best practices:

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

Related documents:

- Building Dashboards for Operational Visibility
- Using Amazon CloudWatch Dashboards

Related videos:

- Create Cross Account & Cross Region CloudWatch Dashboards

- AWS re:Invent 2021 - Gain enterprise visibility with AWS Cloud operation dashboards)

Related examples:

- One Observability Workshop
- Application Monitoring with Amazon CloudWatch
- AWS Health Events Intelligence Dashboards and Insights
- Visualize AWS Health events using Amazon Managed Grafana

Understanding operational health

Define, capture, and analyze operations metrics to gain visibility to the activities of operations teams so that you can take appropriate action.

Your organization should be able to understand the health of your operations easily. You will want to define the business goals of your operations teams, identify key performance indicators that reflect those goals. Afterwards, develop and use metrics based on operations outcomes to gain useful insights. You should use these metrics to implement dashboards and reports with business and technical viewpoints that will help leaders and stakeholders 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 Measure operations goals and KPIs with metrics
- OPS09-BP02 Communicate status and trends to ensure visibility into operation
- OPS09-BP03 Review operations metrics and prioritize improvement

OPS09-BP01 Measure operations goals and KPIs with metrics

Obtain goals and KPIs that define operations success from your organization and determine that metrics reflect these. Set baselines as a point of reference and reevaluate regularly. Develop mechanisms to collect these metrics from teams for evaluation. The DevOps Research and Assessment (DORA) metrics provide a popular method to measure progress towards DevOps practices of software delivery.

Desired outcome:

- The organization publishes and shares the goals and KPIs for the operations teams.
- You establish metrics that reflect these KPIs. Examples may include:
  - Ticket queue depth or average age of ticket
  - Ticket count grouped by type of issue
  - Time spent working issues with or without a standardized operating procedure (SOP)
  - Amount of time spent recovering from a failed code push
  - Call volume

Common anti-patterns:

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

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

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

Implementation guidance

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

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

Resources

Related documents:

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

Related examples:

- Monitor the performance of your software delivery using native AWS monitoring and observability tools
- Balance deployment speed and stability with DORA metrics
- Example MLOps operational metrics in the financial services industry
- How to track your cost optimization KPIs with the KPI Dashboard

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

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

Desired outcome:

- Operations leaders have insight at a glance to see what sort of call volumes their teams are operating under and what efforts may be under way, such as deployments.
- Alerts are disseminated to stakeholders and user communities when impacts to normal operations occur.
- Organization leadership and stakeholders can check a status page in response to an alert or impact, and obtain information surrounding an operational event, such as points of contact, ticket information, and estimated recovery times.
- Reports are made available to leadership and other stakeholders to show operations statistics such as call volumes over a period of time, user satisfaction scores, numbers of outstanding tickets and their ages.

Common anti-patterns:

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

Benefits of establishing this best practice: During operational events where the business is impacted, much time and energy can be wasted querying information from various teams attempting to understand the situation. By establishing widely-disseminated status pages and dashboards, stakeholders can quickly obtain information such as whether or not an issue was

detected, who has lead on the issue, or when a return to normal operations may be expected. This frees team members from spending too much time communicating status to others and more time addressing issues.

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

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

Implementation guidance

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

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

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

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

Use AWS Health alongside your own dashboards, or integrate AWS Health events into them, so that your teams can correlate application issues to AWS service status.

Resources

Related best practices:

- OPS09-BP01 Measure operations goals and KPIs with metrics

Related documents:

- Measure Progress
- Building dashboards for operation visibility

Related examples:

- Data Operations
- How to track your cost optimization KPIs with KPI Dashboard
- The Importance of Key Performance Indicators (KPIs) for Large-Scale Cloud Migrations

OPS09-BP03 Review operations metrics and prioritize improvement

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

Desired outcome:

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

Common anti-patterns:

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

Benefits of establishing this best practice: In some organizations, it can become a challenge to allocate the same time and attention that is afforded to service delivery and new products or

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

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

Implementation guidance

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

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

Resources

Related documents:

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

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
- OPS10-BP02 Have a process per alert
- OPS10-BP03 Prioritize operational events based on business impact
- OPS10-BP04 Define escalation paths
- OPS10-BP05 Define a customer communication plan for service-impacting events
- OPS10-BP06 Communicate status through dashboards
- OPS10-BP07 Automate responses to events

OPS10-BP01 Use a process for event, incident, and problem management

The ability to efficiently manage events, incidents, and problems is key to maintaining workload health and performance. It's crucial to recognize and understand the differences between these elements to develop an effective response and resolution strategy. Establishing and following a well-defined process for each aspect helps your team swiftly and effectively handle any operational challenges that arise.

Desired outcome: Your organization effectively manages operational events, incidents, and problems through well-documented and centrally stored processes. These processes are consistently updated to reflect changes, streamlining handling and maintaining high service reliability and workload performance.

Common anti-patterns:

- You reactively, rather than proactively, respond to events.
- Inconsistent approaches are taken to different types of events or incidents.
- Your organization does not analyze and learn from incidents to prevent future occurrences.

Benefits of establishing this best practice:

- Streamlined and standardized response processes.
- Reduced impact of incidents on services and customers.
- Expedited issue resolution.
- Continuous improvement in operational processes.

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.

Understanding events, incidents, and problems

- **Events:** An *event* is an observation of an action, occurrence, or change of state. Events can be planned or unplanned and they can originate internally or externally to the workload.
- **Incidents:** *Incidents* are events that require a response, like unplanned interruptions or degradations of service quality. They represent disruptions that need immediate attention to restore normal workload operation.
- **Problems:** *Problems* are the underlying causes of one or more incidents. Identifying and resolving problems involves digging deeper into the incidents to prevent future occurrences.

Implementation steps

Events

1. **Monitor events:**
   - Implement observability and utilize workload observability.
   - Monitor actions taken by a user, role, or an AWS service are recorded as events in AWS CloudTrail.
   - Respond to operational changes in your applications in real time with Amazon EventBridge.
   - Continually assess, monitor, and record resource configuration changes with AWS Config.
2. **Create processes:**
   - Develop a process to assess which events are significant and require monitoring. This involves setting thresholds and parameters for normal and abnormal activities.
   - Determine criteria escalating an event to an incident. This could be based on the severity, impact on users, or deviation from expected behavior.
   - Regularly review the event monitoring and response processes. This includes analyzing past incidents, adjusting thresholds, and refining alerting mechanisms.

Incidents

1. **Respond to incidents:**
   - Use insights from observability tools to quickly identify and respond to incidents.
   - Implement AWS Systems Manager Ops Center to aggregate, organize, and prioritize operational items and incidents.
   - Use services like Amazon CloudWatch and AWS X-Ray for deeper analysis and troubleshooting.
   - Consider AWS Managed Services (AMS) for enhanced incident management, leveraging its proactive, preventative, and detective capabilities. AMS extends operational support with services like monitoring, incident detection and response, and security management.
   - Enterprise Support customers can use AWS Incident Detection and Response, which provides continual proactive monitoring and incident management for production workloads.
2. **Create an incident management process:**
   - Establish a structured incident management process, including clear roles, communication protocols, and steps for resolution.

- Integrate incident management with tools like Amazon Q Developer in chat applications for efficient response and coordination.
- Categorize incidents by severity, with predefined incident response plans for each category.

3. **Learn and improve:**
   - Conduct post-incident analysis to understand root causes and resolution effectiveness.
   - Continually update and improve response plans based on reviews and evolving practices.
   - Document and share lessons learned across teams to enhance operational resilience.
   - 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.

Problems

1. **Identify problems:**
   - Use data from previous incidents to identify recurring patterns that may indicate deeper systemic issues.
   - Leverage tools like AWS CloudTrail and Amazon CloudWatch to analyze trends and uncover underlying problems.
   - Engage cross-functional teams, including operations, development, and business units, to gain diverse perspectives on the root causes.
2. **Create a problem management process:**
   - Develop a structured process for problem management, focusing on long-term solutions rather than quick fixes.
   - Incorporate root cause analysis (RCA) techniques to investigate and understand the underlying causes of incidents.
   - Update operational policies, procedures, and infrastructure based on findings to prevent recurrence.
3. **Continue to improve:**
   - Foster a culture of constant learning and improvement, encouraging teams to proactively identify and address potential problems.
   - Regularly review and revise problem management processes and tools to align with evolving business and technology landscapes.

- Share insights and best practices across the organization to build a more resilient and efficient operational environment.

4. **Engage AWS Support:**
   - Use AWS support resources, such as AWS Trusted Advisor, for proactive guidance and optimization recommendations.
   - Enterprise Support customers can access specialized programs like AWS Countdown for support during critical events.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS07-BP03 Use runbooks to perform procedures
- OPS07-BP04 Use playbooks to investigate issues
- OPS08-BP01 Analyze workload metrics
- OPS11-BP02 Perform post-incident analysis

Related documents:

- AWS Security Incident Response Guide
- AWS Incident Detection and Response
- AWS Cloud Adoption Framework: Operations Perspective - Incident and problem management
- Incident Management in the Age of DevOps and SRE
- PagerDuty - What is Incident Management?

Related videos:

- Top incident response tips from AWS
- AWS re:Invent 2022 - The Amazon Builders' Library: 25 yrs of Amazon operational excellence
- AWS re:Invent 2022 - AWS Incident Detection and Response (SUP201)

- Introducing Incident Manager from AWS Systems Manager

Related examples:

- AWS Proactive Services – Incident Management Workshop
- How to Automate Incident Response with PagerDuty and AWS Systems Manager Incident Manager
- Engage Incident Responders with the On-Call Schedules in AWS Systems Manager Incident Manager
- Improve the Visibility and Collaboration during Incident Handling in AWS Systems Manager Incident Manager
- Incident reports and service requests in AMS

Related services:

- Amazon EventBridge

OPS10-BP02 Have a process per alert

Establishing a clear and defined process for each alert in your system is essential for effective and efficient incident management. This practice ensures that every alert leads to a specific, actionable response, improving the reliability and responsiveness of your operations.

Desired outcome: Every alert initiates a specific, well-defined response plan. Where possible, responses are automated, with clear ownership and a defined escalation path. Alerts are linked to an up-to-date knowledge base so that any operator can respond consistently and effectively. Responses are quick and uniform across the board, enhancing operational efficiency and reliability.

Common anti-patterns:

- Alerts have no predefined response process, leading to makeshift and delayed resolutions.
- Alert overload causes important alerts to be overlooked.
- Alerts are inconsistently handled due to lack of clear ownership and responsibility.

Benefits of establishing this best practice:

- Reduced alert fatigue by only raising actionable alerts.
- Decreased mean time to resolution (MTTR) for operational issues.
- Decreased mean time to investigate (MTTI), which helps reduce MTTR.
- Enhanced ability to scale operational responses.
- Improved consistency and reliability in handling operational events.

For example, you have a defined process for AWS Health events for critical accounts, including application alarms, operational issues, and planned lifecycle events (like updating Amazon EKS versions before clusters are auto-updated), and you provide the capability for your teams to actively monitor, communicate, and respond to these events. These actions help you prevent service disruptions caused by AWS-side changes or mitigate them faster when unexpected issues occur.

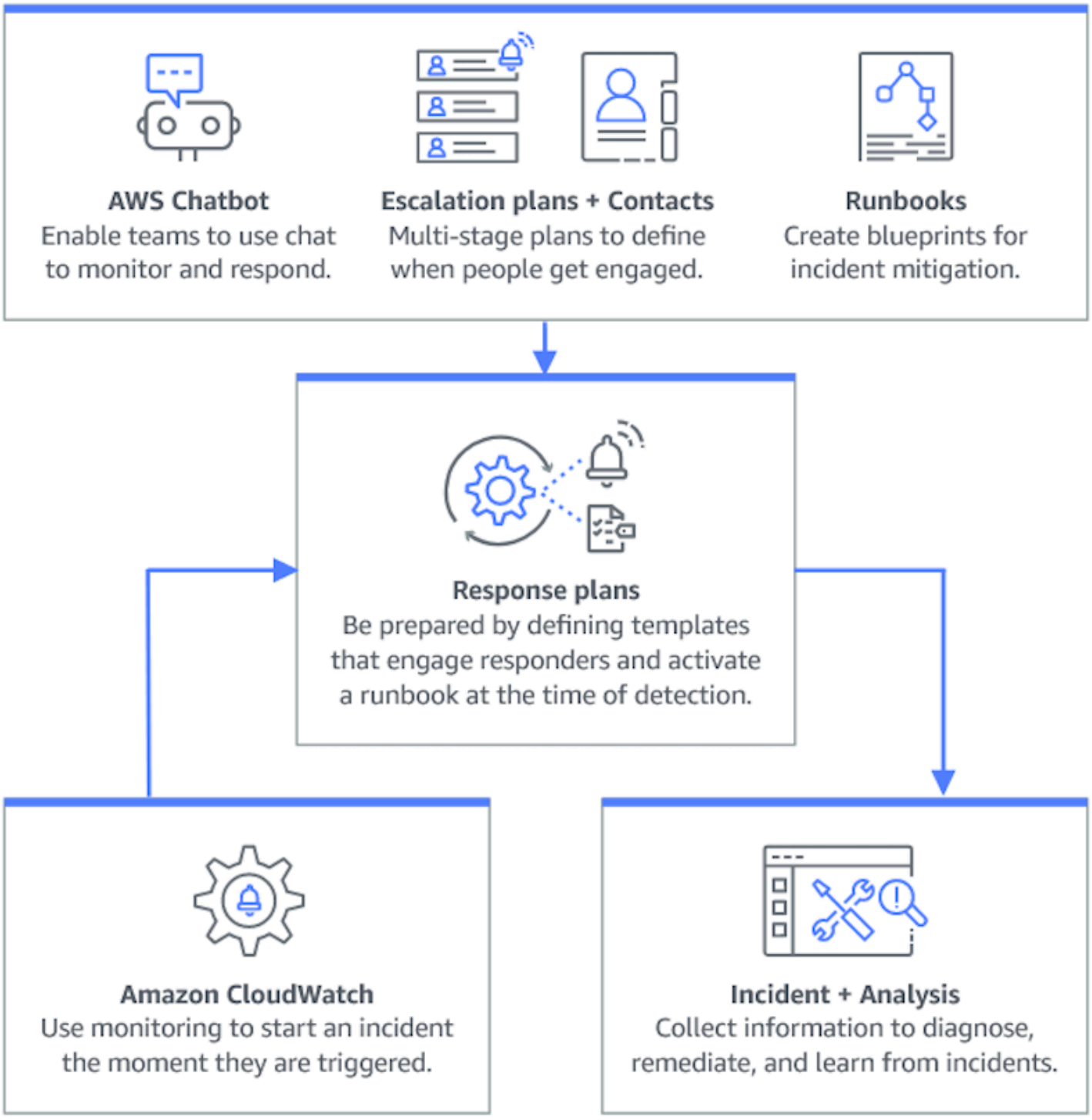
Level of risk exposed if this best practice is not established: High

Implementation guidance

Having a process per alert involves establishing a clear response plan for each alert, automating responses where possible, and continually refining these processes based on operational feedback and evolving requirements.

Implementation steps

The following diagram illustrates the incident management workflow within AWS Systems Manager Incident Manager. It is designed to respond swiftly to operational issues by automatically creating incidents in response to specific events from Amazon CloudWatch or Amazon EventBridge. When an incident is created, either automatically or manually, Incident Manager centralizes the management of the incident, organizes relevant AWS resource information, and initiates predefined response plans. This includes running Systems Manager Automation runbooks for immediate action, as well as creating a parent operational work item in OpsCenter to track related tasks and analyses. This streamlined process speeds up and coordinates incident response across your AWS environment.



1. **Use composite alarms:** Create composite alarms in CloudWatch to group related alarms, reducing noise and allowing for more meaningful responses.
2. **Stay informed with AWS Health:** AWS Health is the authoritative source of information about the health of your AWS Cloud resources. Use AWS Health to visualize and get notified of any

current service events and upcoming changes, such as planned lifecycle events, so you can take steps to mitigate impacts.

a. Create purpose-fit AWS Health event notifications to e-mail and chat channels through AWS User Notifications, and integrate programatically with your monitoring and alerting tools through Amazon EventBridge or the AWS Health API.

b. Plan and track progress on health events that require action by integrating with change management or ITSM tools (like Jira or ServiceNow) that you may already use through Amazon EventBridge or the AWS Health API.

c. If you use AWS Organizations, enable organization view for AWS Health to aggregate AWS Health events across accounts.

3. **Integrate Amazon CloudWatch alarms with Incident Manager:** Configure CloudWatch alarms to automatically create incidents in AWS Systems Manager Incident Manager.
4. **Integrate Amazon EventBridge with Incident Manager:** Create EventBridge rules to react to events and create incidents using defined response plans.
5. **Prepare for incidents in Incident Manager:**
   - Establish detailed response plans in Incident Manager for each type of alert.
   - Establish chat channels through Amazon Q Developer in chat applications connected to response plans in Incident Manager, facilitating real-time communication during incidents across platforms like Slack, Microsoft Teams, and Amazon Chime.
   - Incorporate Systems Manager Automation runbooks within Incident Manager to drive automated responses to incidents.

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS08-BP04 Create actionable alerts

Related documents:

- AWS Cloud Adoption Framework: Operations Perspective - Incident and problem management
- Using Amazon CloudWatch alarms
- Setting up AWS Systems Manager Incident Manager

- Preparing for incidents in Incident Manager

Related videos:

- Top incident response tips from AWS
- re:Invent 2023 | Manage resource lifecycle events at scale with AWS Health

Related examples:

- AWS Workshops - AWS Systems Manager Incident Manager - Automate incident response to security events

OPS10-BP03 Prioritize operational events based on business impact

Responding promptly to operational events is critical, but not all events are equal. When you prioritize based on business impact, you also prioritize addressing events with the potential for significant consequences, such as safety, financial loss, regulatory violations, or damage to reputation.

Desired outcome: Responses to operational events are prioritized based on potential impact to business operations and objectives. This makes the responses efficient and effective.

Common anti-patterns:

- Every event is treated with the same level of urgency, leading to confusion and delays in addressing critical issues.
- You fail to distinguish between high and low impact events, leading to misallocation of resources.
- Your organization lacks a clear prioritization framework, resulting in inconsistent responses to operational events.
- Events are prioritized based on the order they are reported, rather than their impact on business outcomes.

Benefits of establishing this best practice:

- Ensures critical business functions receive attention first, minimizing potential damage.
- Improves resource allocation during multiple concurrent events.

- Enhances the organization's ability to maintain trust and meet regulatory requirements.

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

Implementation guidance

When faced with multiple operational events, a structured approach to prioritization based on impact and urgency is essential. This approach helps you make informed decisions, direct efforts where they're needed most, and mitigate the risk to business continuity.

Implementation steps

1. **Assess impact:** Develop a classification system to evaluate the severity of events in terms of their potential impact on business operations and objectives. The following example shows impact categories:

Impact level	Description
High	Affects many staff or customers, high financial impact, high reputational damage, or injury.
Medium	Affects a groups of staff or customers, moderate financial impact, or moderate reputational damage.
Low	Affects individual staff or customers, low financial impact, or low reputational damage.

2. **Assess urgency:** Define urgency levels for how quickly an event needs a response, considering factors such as safety, financial implications, and service-level agreements (SLAs). The following example demonstrates urgency categories:

Urgency level	Description
High	Exponentially increasing damage, time-sens itive work impacted, imminent escalation, or VIP users or groups affected.

Urgency level	Description
Medium	Damage increases over time, or single VIP user or group affected.
Low	Marginal damage increase over time, or non-time-sensitive work impacted.

3. **Create a prioritization matrix:**
    - Use a matrix to cross-reference impact and urgency, assigning priority levels to different combinations.
    - Make the matrix accessible and understood by all team members responsible for operational event responses.
    - The following example matrix displays incident severity according to urgency and impact:

Urgency and impact	High	Medium	Low
High	Critical	Urgent	High
Medium	Urgent	High	Normal
Low	High	Normal	Low

4. **Train and communicate:** Train response teams on the prioritization matrix and the importance of following it during an event. Communicate the prioritization process to all stakeholders to set clear expectations.
5. **Integrate with incident response:**
    - Incorporate the prioritization matrix into your incident response plans and tools.
    - Automate the classification and prioritization of events where possible to speed up response times.
    - Enterprise Support customers can leverage AWS Incident Detection and Response, which provides 24x7 proactive monitoring and incident management for production workloads.
6. **Review and adapt:** Regularly review the effectiveness of the prioritization process and make adjustments based on feedback and changes in the business environment.

Resources

Related best practices:

- OPS03-BP03 Escalation is encouraged
- OPS08-BP04 Create actionable alerts
- OPS09-BP01 Measure operations goals and KPIs with metrics

Related documents:

- Atlassian - Understanding incident severity levels
- IT Process Map - Checklist Incident Priority

OPS10-BP04 Define escalation paths

Establish clear escalation paths within your incident response protocols to facilitate timely and effective action. This includes specifying prompts for escalation, detailing the escalation process, and pre-approving actions to expedite decision-making and reduce mean time to resolution (MTTR).

Desired outcome: A structured and efficient process that escalates incidents to the appropriate personnel, minimizing response times and impact.

Common anti-patterns:

- Lack of clarity on recovery procedures leads to makeshift responses during critical incidents.
- Absence of defined permissions and ownership results in delays when urgent action is needed.
- Stakeholders and customers are not informed in line with expectations.
- Important decisions are delayed.

Benefits of establishing this best practice:

- Streamlined incident response through predefined escalation procedures.
- Reduced downtime with pre-approved actions and clear ownership.
- Improved resource allocation and support-level adjustments according to incident severity.
- Improved communication to stakeholders and customers.

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

Implementation guidance

Properly defined escalation paths are crucial for rapid incident response. AWS Systems Manager Incident Manager supports the setup of structured escalation plans and on-call schedules, which alert the right personnel so that they are ready to act when incidents occur.

Implementation steps

1. **Set up escalation prompts:** Set up CloudWatch alarms to create an incident in AWS Systems Manager Incident Manager.
2. **Set up on-call schedules:** Create on-call schedules in Incident Manager that align with your escalation paths. Equip on-call personnel with the necessary permissions and tools to act swiftly.
3. **Detail escalation procedures:**
   - Determine specific conditions under which an incident should be escalated.
   - Create escalation plans in Incident Manager.
   - Escalation channels should consist of a contact or an on-call schedule.
   - Define the roles and responsibilities of the team at each escalation level.
4. **Pre-approve mitigation actions:** Collaborate with decision-makers to pre-approve actions for anticipated scenarios. Use Systems Manager Automation runbooks integrated with Incident Manager to speed up incident resolution.
5. **Specify ownership:** Clearly identify internal owners for each step of the escalation path.
6. **Detail third-party escalations:**
   - Document third-party service-level agreements (SLAs), and align them with internal goals.
   - Set clear protocols for vendor communication during incidents.
   - Integrate vendor contacts into incident management tools for direct access.
   - Conduct regular drills that include third-party response scenarios.
   - Keep vendor escalation information well-documented and easily accessible.
7. **Train and rehearse escalation plans:** Train your team on the escalation process and conduct regular incident response drills or game days. Enterprise Support customers can request an Incident Management Workshop.
8. **Continue to improve:** Review the effectiveness of your escalation paths regularly. Update your processes based on lessons learned from incident post-mortems and continuous feedback.

Level of effort for the implementation plan: Moderate

Resources

Related best practices:

- OPS08-BP04 Create actionable alerts
- OPS10-BP02 Have a process per alert
- OPS11-BP02 Perform post-incident analysis

Related documents:

- AWS Systems Manager Incident Manager Escalation Plans
- Working with on-call schedules in Incident Manager
- Creating and Managing Runbooks
- Temporary elevated access management with AWS IAM Identity Center
- Atlassian - Escalation policies for effective incident management

OPS10-BP05 Define a customer communication plan for service-impacting events

Effective communication during service impacting events is critical to maintain trust and transparency with customers. A well-defined communication plan helps your organization quickly and clearly share information, both internally and externally, during incidents.

Desired outcome:

- A robust communication plan that effectively informs customers and stakeholders during service impacting events.
- Transparency in communication to build trust and reduce customer anxiety.
- Minimizing the impact of service impacting events on customer experience and business operations.

Common anti-patterns:

- Inadequate or delayed communication leads to customer confusion and dissatisfaction.

- Overly technical or vague messaging fails to convey the actual impact on users.
- There is no predefined communication strategy, resulting in inconsistent and reactive messaging.

Benefits of establishing this best practice:

- Enhanced customer trust and satisfaction through proactive and clear communication.
- Reduced burden on support teams by preemptively addressing customer concerns.
- Improved ability to manage and recover from incidents effectively.

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

Implementation guidance

Creating a comprehensive communication plan for service impacting events involves multiple facets, from choosing the right channels to crafting the message and tone. The plan should be adaptable, scalable, and cater to different outage scenarios.

Implementation steps

1. **Define roles and responsibilities:**
   - Assign a major incident manager to oversee incident response activities.
   - Designate a communications manager responsible for coordinating all external and internal communications.
   - Include the support manager to provide consistent communication through support tickets.
2. **Identify communication channels:** Select channels like workplace chat, email, SMS, social media, in-app notifications, and status pages. These channels should be resilient and able to operate independently during service impacting events.
3. **Communicate quickly, clearly, and regularly to customers:**
   - Develop templates for various service impairment scenarios, emphasizing simplicity and essential details. Include information about the service impairment, expected resolution time, and impact.
   - Use Amazon Pinpoint to alert customers using push notifications, in-app notifications, emails, text messages, voice messages, and messages over custom channels.
   - Use Amazon Simple Notification Service (Amazon SNS) to alert subscribers programatically or through email, mobile push notifications, and text messages.

- Communicate status through dashboards by sharing an Amazon CloudWatch dashboard publicly.
- Encourage social media engagement:
  - Actively monitor social media to understand customer sentiment.
  - Post on social media platforms for public updates and community engagement.
  - Prepare templates for consistent and clear social media communication.

4. **Coordinate internal communication:** Implement internal protocols using tools like Amazon Q Developer in chat applications for team coordination and communication. Use CloudWatch dashboards to communicate status.
5. **Orchestrate communication with dedicated tools and services:**
   - Use AWS Systems Manager Incident Manager with Amazon Q Developer in chat applications to set up dedicated chat channels for real-time internal communication and coordination during incidents.
   - Use AWS Systems Manager Incident Manager runbooks to automate customer notifications through Amazon Pinpoint, Amazon SNS, or third-party tools like social media platforms during incidents.
   - Incorporate approval workflows within runbooks to optionally review and authorize all external communications before sending.
6. **Practice and improve:**
   - Conduct training on the use of communication tools and strategies. Empower teams to make timely decisions during incidents.
   - Test the communication plan through regular drills or gamedays. Use these tests to refine messaging and evaluate the effectiveness of channels.
   - Implement feedback mechanisms to assess communication effectiveness during incidents. Continually evolve the communication plan based on feedback and changing needs.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS07-BP03 Use runbooks to perform procedures
- OPS10-BP06 Communicate status through dashboards

- OPS11-BP02 Perform post-incident analysis

Related documents:

- Atlassian - Incident communication best practices
- Atlassian - How to write a good status update
- PagerDuty - A Guide to Incident Communications

Related videos:

- Atlassian - Create your own incident communication plan: Incident templates

Related examples:

- AWS Health Dashboard

OPS10-BP06 Communicate status through dashboards

Use dashboards as a strategic tool to convey real-time operational status and key metrics to different audiences, including internal technical teams, leadership, and customers. These dashboards offer a centralized, visual representation of system health and business performance, enhancing transparency and decision-making efficiency.

Desired outcome:

- Your dashboards provide a comprehensive view of the system and business metrics relevant to different stakeholders.
- Stakeholders can proactively access operational information, reducing the need for frequent status requests.
- Real-time decision-making is enhanced during normal operations and incidents.

Common anti-patterns:

- Engineers joining an incident management call require status updates to get up to speed.
- Relying on manual reporting for management, which leads to delays and potential inaccuracies.
- Operations teams are frequently interrupted for status updates during incidents.

Benefits of establishing this best practice:

- Empowers stakeholders with immediate access to critical information, promoting informed decision-making.
- Reduces operational inefficiencies by minimizing manual reporting and frequent status inquiries.
- Increases transparency and trust through real-time visibility into system performance and business metrics.

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

Implementation guidance

Dashboards effectively communicate the status of your system and business metrics and can be tailored to the needs of different audience groups. Tools like Amazon CloudWatch dashboards and Amazon QuickSight help you create interactive, real-time dashboards for system monitoring and business intelligence.

Implementation steps

1. **Identify stakeholder needs:** Determine the specific information needs of different audience groups, such as technical teams, leadership, and customers.
2. **Choose the right tools:** Select appropriate tools like Amazon CloudWatch dashboards for system monitoring and Amazon QuickSight for interactive business intelligence. AWS Health provides a ready-to-use experience in the AWS Health Dashboard, or you can use Health events in Amazon EventBridge or through the AWS Health API to augment your own dashboards.
3. **Design effective dashboards:**
   - Design dashboards to clearly present relevant metrics and KPIs, ensuring they are understandable and actionable.
   - Incorporate system-level and business-level views as needed.
   - Include both high-level (for broad overviews) and low-level (for detailed analysis) dashboards.
   - Integrate automated alarms within dashboards to highlight critical issues.
   - Annotate dashboards with important metrics thresholds and goals for immediate visibility.
4. **Integrate data sources:**
   - Use Amazon CloudWatch to aggregate and display metrics from various AWS services and query metrics from other data sources, creating a unified view of your system's health and business metrics.

- Use features like CloudWatch Logs Insights to query and visualize log data from different applications and services.
- Use AWS Health events to stay informed about the operational status and confirmed operational issues from AWS services through the AWS Health API or AWS Health events on Amazon EventBridge.

5. **Provide self-service access:**
   - Share CloudWatch dashboards with relevant stakeholders for self-service information access using dashboard sharing features.
   - Ensure that dashboards are easily accessible and provide real-time, up-to-date information.
6. **Regularly update and refine:**
   - Continually update and refine dashboards to align with evolving business needs and stakeholder feedback.
   - Regularly review the dashboards to keep them relevant and effective for conveying the necessary information.

Resources

Related best practices:

- OPS08-BP05 Create dashboards

Related documents:

- Building dashboards for operational visibility
- Using Amazon CloudWatch dashboards
- Create flexible dashboards with dashboard variables
- Sharing CloudWatch dashboards
- Query metrics from other data sources
- Add a custom widget to a CloudWatch dashboard

Related examples:

- One Observability Workshop - Dashboards

OPS10-BP07 Automate responses to events

Automating event responses is key for fast, consistent, and error-free operational handling. Create streamlined processes and use tools to automatically manage and respond to events, minimizing manual interventions and enhancing operational effectiveness.

Desired outcome:

- Reduced human errors and faster resolution times through automation.
- Consistent and reliable operational event handling.
- Enhanced operational efficiency and system reliability.

Common anti-patterns:

- Manual event handling leads to delays and errors.
- Automation is overlooked in repetitive, critical tasks.
- Repetitive, manual tasks lead to alert fatigue and missing critical issues.

Benefits of establishing this best practice:

- Accelerated event responses, reducing system downtime.
- Reliable operations with automated and consistent event handling.

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

Implementation guidance

Incorporate automation to create efficient operational workflows and minimize manual interventions.

Implementation steps

1. **Identify automation opportunites:** Determine repetitive tasks for automation, such as issue remediation, ticket enrichment, capacity management, scaling, deployments, and testing.
2. **Identify automation prompts:**
   - Assess and define specific conditions or metrics that initiate automated responses using Amazon CloudWatch alarm actions.

- Use Amazon EventBridge to respond to events in AWS services, custom workloads, and SaaS applications.
- Consider initiation events such as specific log entries, performance metrics thresholds, or state changes in AWS resources.

3. **Implement event-driven automation:**
   - Use AWS Systems Manager Automation runbooks to simplify maintenance, deployment, and remediation tasks.
   - Creating incidents in Incident Manager automatically gathers and adds details about the involved AWS resources to the incident.
   - Proactively monitor quotas using Quota Monitor for AWS.
   - Automatically adjust capacity with AWS Auto Scaling to maintain availability and performance.
   - Automate development pipelines with Amazon CodeCatalyst.
   - Smoke test or continually monitor endpoints and APIs using synthetic monitoring.
4. **Perform risk mitigation through automation:**
   - Implement automated security responses to swiftly address risks.
   - Use AWS Systems Manager State Manager to reduce configuration drift.
   - Remediate noncompliant resources with AWS Config Rules.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS08-BP04 Create actionable alerts
- OPS10-BP02 Have a process per alert

Related documents:

- Using Systems Manager Automation runbooks with Incident Manager
- Creating incidents in Incident Manager
- AWS service quotas
- Monitor resource usage and send notifications when approaching quotas

- AWS Auto Scaling
- What is Amazon CodeCatalyst?
- Using Amazon CloudWatch alarms
- Using Amazon CloudWatch alarm actions
- Remediating Noncompliant Resources with AWS Config Rules
- Creating metrics from log events using filters
- AWS Systems Manager State Manager

Related videos:

- Create Automation Runbooks with AWS Systems Manager
- How to automate IT Operations on AWS
- AWS Security Hub automation rules
- Start your software project fast with Amazon CodeCatalyst blueprints

Related examples:

- Amazon CodeCatalyst Tutorial: Creating a project with the Modern three-tier web application blueprint
- One Observability Workshop
- Respond to incidents using Incident Manager

Evolve

Learn, share, and continuously improve to sustain operational excellence. Dedicate work cycles to making nearly continuous incremental improvements. Perform post-incident analysis of all customer impacting events. Identify the contributing factors and preventative action to limit or prevent recurrence. Communicate contributing factors with affected communities as appropriate. Regularly evaluate and prioritize opportunities for improvement (for example, feature requests, issue remediation, and compliance requirements), including both the workload and operations procedures.

Include feedback loops within your procedures to rapidly identify areas for improvement and capture learnings from running operations.

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

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

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

Topics

- Learn, share, and improve

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
- OPS11-BP02 Perform post-incident analysis
- OPS11-BP03 Implement feedback loops
- OPS11-BP04 Perform knowledge management
- OPS11-BP05 Define drivers for improvement
- OPS11-BP06 Validate insights
- OPS11-BP07 Perform operations metrics reviews
- OPS11-BP08 Document and share lessons learned
- OPS11-BP09 Allocate time to make improvements

OPS11-BP01 Have a process for continuous improvement

Evaluate your workload against internal and external architecture best practices. Conduct frequent, intentional workload reviews. Prioritize improvement opportunities into your software development cadence.

Desired outcome:

- You analyze your workload against architecture best practices frequently.
- You give improvement opportunities equal priority to features 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.

- You give a lower priority to improvement opportunities. Compared to new features, these opportunities 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.
- You evolve your workload in an intentional manner.
- You can leverage organization best practices to improve all workloads.
- You make marginal gains that have a cumulative impact, which drives deeper efficiencies.

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

Implementation guidance

Frequently conduct an architectural review of your workload. Use internal and external best practices, evaluate your workload, and identify improvement opportunities. Prioritize improvement opportunities into your software development cadence.

Implementation steps

1. Conduct periodic architecture reviews of your production workload with an agreed-upon frequency. Use a documented architectural standard that includes AWS-specific best practices.
   a. Use your internally-defined standards for these reviews. If you do not have an internal standard, use the AWS Well-Architected Framework.
   b. Use the AWS Well-Architected Tool to create a custom lens of your internal best practices and conduct your architecture review.
   c. Contact your AWS Solution Architect or Technical Account Manager to conduct a guided Well-Architected Framework Review of your 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
- OPS11-BP08 Document and share lessons learned
- OPS04 Implement Observability

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:

- AWS re:Invent 2023 - Scaling AWS Well-Architected best practices across your organization

Related examples:

- 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.

Desired outcome:

- You have established incident management processes that include post-incident analysis.
- You have observability plans in place to collect data on events.
- With this data, you understand and collect metrics that support your post-incident analysis process.

- You learn from incidents to improve future outcomes.

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:

- You have a predefined process to determine the components, conditions, actions, and events that contributed to an incident, which helps you identify opportunities for improvement.
- You use data from post-incident analysis to make improvements.

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 incident root causes as appropriate, and tailor the communication to your target audience. Share learnings openly within your organization.

Implementation steps

1. Collect metrics such as deployment change, configuration change, incident start time, alarm time, time of engagement, mitigation start time, and incident resolved time.
2. Describe key time points on the timeline to understand the events of the incident.
3. Ask the following questions:

a. Could you improve time to detection?
b. Are there updates to metrics and alarms that would detect the incident sooner?
c. Can you improve the time to diagnosis?
d. Are there updates to your response plans or escalation plans that would engage the correct responders sooner?
e. Can you improve the time to mitigation?
f. Are there runbook or playbook steps that you could add or improve?
g. Can you prevent future incidents from occurring?

4. Create checklists and actions. Track and deliver all actions.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS11-BP01 Have a process for continuous improvement
- OPS 4 - Implement observability

Related documents:

- Performing a post-incident analysis in Incident Manager
- Operational Readiness Review

OPS11-BP03 Implement feedback loops

Feedback loops provide actionable insights that drive decision making. Build feedback loops into your procedures and workloads. This helps you identify issues and areas that need improvement. They also validate investments made in improvements. These feedback loops are the foundation for continuously improving your workload.

Feedback loops fall into two categories: *immediate feedback* and *retrospective analysis*. Immediate feedback is gathered through review of the performance and outcomes from operations activities. This feedback comes from team members, customers, or the automated output of the activity. Immediate feedback is received from things like A/B testing and shipping new features, and it is essential to failing fast.

Retrospective analysis is performed regularly to capture feedback from the review of operational outcomes and metrics over time. These retrospectives happen at the end of a sprint, on a cadence, or after major releases or events. This type of feedback loop validates investments in operations or your workload. It helps you measure success and validates your strategy.

Desired outcome: You use immediate feedback and retrospective analysis to drive improvements. There is a mechanism to capture user and team member feedback. Retrospective analysis is used to identify trends that drive improvements.

Common anti-patterns:

- You launch a new feature but have no way of receiving customer feedback on it.
- After investing in operations improvements, you don't conduct a retrospective to validate them.
- You collect customer feedback but don't regularly review it.
- Feedback loops lead to proposed action items but they aren't included in the software development process.
- Customers don't receive feedback on improvements they've proposed.

Benefits of establishing this best practice:

- You can work backwards from the customer to drive new features.
- Your organization culture can react to changes faster.
- Trends are used to identify improvement opportunities.
- Retrospectives validate investments made to your workload and operations.

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

Implementation guidance

Implementing this best practice means that you use both immediate feedback and retrospective analysis. These feedback loops drive improvements. There are many mechanisms for immediate feedback, including surveys, customer polls, or feedback forms. Your organization also uses retrospectives to identify improvement opportunities and validate initiatives.

Customer example

AnyCompany Retail created a web form where customers can give feedback or report issues. During the weekly scrum, user feedback is evaluated by the software development team. Feedback

is regularly used to steer the evolution of their platform. They conduct a retrospective at the end of each sprint to identify items they want to improve.

Implementation steps

1. Immediate feedback
   - You need a mechanism to receive feedback from customers and team members. Your operations activities can also be configured to deliver automated feedback.
   - Your organization needs a process to review this feedback, determine what to improve, and schedule the improvement.
   - Feedback must be added into your software development process.
   - As you make improvements, follow up with the feedback submitter.
     - You can use AWS Systems Manager OpsCenter to create and track these improvements as OpsItems.
2. Retrospective analysis
   - Conduct retrospectives at the end of a development cycle, on a set cadence, or after a major release.
   - Gather stakeholders involved in the workload for a retrospective meeting.
   - Create three columns on a whiteboard or spreadsheet: Stop, Start, and Keep.
     - *Stop* is for anything that you want your team to stop doing.
     - *Start* is for ideas that you want to start doing.
     - *Keep* is for items that you want to keep doing.
   - Go around the room and gather feedback from the stakeholders.
   - Prioritize the feedback. Assign actions and stakeholders to any Start or Keep items.
   - Add the actions to your software development process and communicate status updates to stakeholders as you make the improvements.

Level of effort for the implementation plan: Medium. To implement this best practice, you need a way to take in immediate feedback and analyze it. Also, you need to establish a retrospective analysis process.

Resources

Related best practices:

- OPS01-BP01 Evaluate external customer needs: Feedback loops are a mechanism to gather external customer needs.
- OPS01-BP02 Evaluate internal customer needs: Internal stakeholders can use feedback loops to communicate needs and requirements.
- OPS11-BP02 Perform post-incident analysis: Post-incident analyses are an important form of retrospective analysis conducted after incidents.
- OPS11-BP07 Perform operations metrics reviews: Operations metrics reviews identify trends and areas for improvement.

Related documents:

- 7 Pitfalls to Avoid When Building a CCOE
- Atlassian Team Playbook - Retrospectives
- Email Definitions: Feedback Loops
- Establishing Feedback Loops Based on the AWS Well-Architected Framework Review
- IBM Garage Methodology - Hold a retrospective
- Investopedia – The PDCS Cycle
- Maximizing Developer Effectiveness by Tim Cochran
- Operations Readiness Reviews (ORR) Whitepaper - Iteration
- ITIL CSI - Continual Service Improvement
- When Toyota met e-commerce: Lean at Amazon

Related videos:

- Building Effective Customer Feedback Loops

Related examples:

- Astuto - Open source customer feedback tool
- AWS Solutions - QnABot on AWS
- Fider - A platform to organize customer feedback

Related services:

- AWS Systems Manager OpsCenter

OPS11-BP04 Perform knowledge management

Knowledge management helps team members find the information to perform their job. In learning organizations, information is freely shared which empowers individuals. The information can be discovered or searched. Information is accurate and up to date. Mechanisms exist to create new information, update existing information, and archive outdated information. The most common example of a knowledge management platform is a content management system like a wiki.

Desired outcome:

- Team members have access to timely, accurate information.
- Information is searchable.
- Mechanisms exist to add, update, and archive information.

Common anti-patterns:

- There is no centralized knowledge storage. Team members manage their own notes on their local machines.
- You have a self-hosted wiki but no mechanisms to manage information, resulting in outdated information.
- Someone identifies missing information but there's no process to request adding it the team wiki. They add it themselves but they miss a key step, leading to an outage.

Benefits of establishing this best practice:

- Team members are empowered because information is shared freely.
- New team members are onboarded faster because documentation is up to date and searchable.
- Information is timely, accurate, and actionable.

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

Implementation guidance

Knowledge management is an important facet of learning organizations. To begin, you need a central repository to store your knowledge (as a common example, a self-hosted wiki). You must develop processes for adding, updating, and archiving knowledge. Develop standards for what should be documented and let everyone contribute.

Customer example

AnyCompany Retail hosts an internal Wiki where all knowledge is stored. Team members are encouraged to add to the knowledge base as they go about their daily duties. On a quarterly basis, a cross-functional team evaluates which pages are least updated and determines if they should be archived or updated.

Implementation steps

1. Start with identifying the content management system where knowledge will be stored. Get agreement from stakeholders across your organization.
   a. If you don't have an existing content management system, consider running a self-hosted wiki or using a version control repository as a starting point.
2. Develop runbooks for adding, updating, and archiving information. Educate your team on these processes.
3. Identify what knowledge should be stored in the content management system. Start with daily activities (runbooks and playbooks) that team members perform. Work with stakeholders to prioritize what knowledge is added.
4. On a periodic basis, work with stakeholders to identify out-of-date information and archive it or bring it up to date.

Level of effort for the implementation plan: Medium. If you don't have an existing content management system, you can set up a self-hosted wiki or a version-controlled document repository.

Resources

Related best practices:

- OPS11-BP08 Document and share lessons learned - Knowledge management facilitates information sharing about lessons learned.

Related documents:

- Atlassian - Knowledge Management

Related examples:

- DokuWiki
- Gollum
- MediaWiki
- Wiki.js

OPS11-BP05 Define drivers for improvement

Identify drivers for improvement to help you evaluate and prioritize opportunities based on data and feedback loops. Explore improvement opportunities in your systems and processes, and automate where appropriate.

Desired outcome:

- You track data from across your environment.
- You correlate events and activities to business outcomes.
- You can compare and contrast between environments and systems.
- You maintain a detailed activity history of your deployments and outcomes.
- You collect data to support your security posture.

Common anti-patterns:

- You collect data from across your environment but do not correlate events and activities.
- You collect detailed data from across your estate, and it drives high Amazon CloudWatch and AWS CloudTrail activity and cost. However, you do not use this data meaningfully.
- You do not account for business outcomes when defining drivers for improvement.
- You do not measure the effects of new features.

Benefits of establishing this best practice:

- You minimize the impact of event-based motivations or emotional investment by determining criteria for improvement.
- You respond to business events, not just technical ones.
- You measure your environment to identify areas of improvement.

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. Track rightsizing options, and seek optimization opportunities.
    - AWS Latest Security Bulletins
    - AWS Trusted Advisor
    - Cloud Intelligence Dashboards
  - 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 best practices:

- OPS01 Organization priorities
- OPS02 Relationships and Ownerships
- OPS04-BP01 Identify key performance indicators
- OPS08 Utilizing Workload Observability

- OPS09 Understanding Operational Health
- OPS11-BP03 Implement feedback loops

Related documents:

- Amazon Athena
- 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
- The Imperatives of Customer-Centric Innovation
- Digital Transformation: Hype or a Strategic Necessity?

Related Videos

- AWS re:Invent 2023 - Improve operational efficiency and resilience with Support (SUP310)

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.

Desired outcomes:

- You review insights with business owners on a regular basis. Business owners provide additional context to newly-gained insights.
- You review insights and request feedback from technical peers, and you share your learnings across teams.

- You publish data and insights for other technical and business teams to review. You factor in your learnings to new practices by other departments.
- Summarize and review new insights with senior leaders. Senior leaders use new insights to define strategy.

Common anti-patterns:

- You release a new feature. This feature changes some of your customer behaviors. Your observability does not take these changes into account. You do not quantify the benefits of these changes.
- You push a new update and neglect refreshing your CDN. The CDN cache is no longer compatible with the latest release. You measure the percentage of requests with errors. All of your users report HTTP 400 errors when communicating with backend servers. You investigate the client errors and find that because you measured the wrong dimension, your time was wasted.
- Your service-level agreement stipulates 99.9% uptime, and your recovery point objective is four hours. The service owner maintains that the system is zero downtime. You implement an expensive and complex replication solution, which wastes time and money.

Benefits of establishing this best practice:

- When you validate insights with business owners and subject matter experts, you establish common understanding and more effectively guide improvement.
- You discover hidden issues and factor them into future decisions.
- Your focus moves from technical outcomes to business outcomes.

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.

Resources

Related best practices:

- OPS01-BP06 Evaluate tradeoffs while managing benefits and risks
- OPS02-BP06 Responsibilities between teams are predefined or negotiated
- OPS11-BP03 Implement feedback loops

Related documents:

- Designing a Cloud Center of Excellence (CCOE)

Related videos:

- Building observability to increase resiliency

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).

Desired outcome:

- You frequently review business-affecting metrics
- You detect and review anomalies through your observability capabilities
- You use data to support business outcomes and goals

Common anti-patterns:

- Your maintenance window interrupts a significant retail promotion. 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 you commonly use an outdated library in your organization. You have since migrated to a supported library. The other teams in your organization do not know that they are at risk.
- You do not regularly review attainment of customer SLAs. You are trending to not meet your customer SLAs. There are financial penalties related to not meeting your customer SLAs.

Benefits of establishing this best practice:

- When you meet regularly to review operations metrics, events, and incidents, you maintain common understanding across teams.
- Your team meets routinely to review metrics and incidents, which positions you to take action on risks and recognize customer SLAs.
- You share lessons learned, which provides data for prioritization and targeted improvements for business outcomes.

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

Implementation guidance

- 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 share lessons learned.
- Use their insights to identify opportunities for improvement and potential courses of action.

Resources

Related best practices:

- OPS08-BP05 Create dashboards
- OPS09-BP03 Review operations metrics and prioritize improvement
- OPS10-BP01 Use a process for event, incident, and problem management

Related documents:

- Amazon CloudWatch
- Amazon CloudWatch metrics and dimensions reference
- Publish custom metrics
- Using Amazon CloudWatch metrics
- Dashboards and visualizations with CloudWatch

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. Share information and resources to prevent avoidable errors and ease development efforts, and focus on delivery of desired features.

Use AWS Identity and Access Management (IAM) to define permissions that permit controlled access to the resources you wish to share within and across accounts.

Desired outcome:

- You use version-controlled repositories to share application libraries, scripted procedures, procedure documentation, and other system documentation.
- You share your infrastructure standards as version-controlled AWS CloudFormation templates.
- You review lessons learned across teams.

Common anti-patterns:

- You suffered an extended outage because your organization commonly uses buggy library. You have since migrated to a reliable library. The other teams in your organization do not know they are at risk. No one documents and shares the experience with this library, and they are not 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.
- 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.

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.
  - Leverage AWS re:Post Private as a knowledge service to streamline collaboration and knowledge sharing within your organization.

Resources

Related best practices:

- OPS02-BP06 Responsibilities between teams are predefined or negotiated
- OPS05-BP01 Use version control
- OPS05-BP06 Share design standards
- OPS11-BP03 Implement feedback loops
- OPS11-BP07 Perform operations metrics reviews

Related documents:

- Increase collaboration and securely share cloud knowledge with AWS re:Post Private
- Reduce project delays with a docs-as-code solution

Related videos:

- AWS re:Invent 2023 - Collaborate within your company and with AWS using AWS re:Post Private
- Supports You | Exploring the Incident Management Tabletop Exercise

OPS11-BP09 Allocate time to make improvements

Dedicate time and resources within your processes to make continuous incremental improvements possible.

Desired outcome:

- You create temporary duplicates of environments, which lowers 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.
- You run gamedays, and you use Fault Injection Service (FIS) to provide the controls and guardrails that teams need to run experiments in a production-like environment.

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 would 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.
- Operational acceptance is complete, and you do not test operational practices again.

Benefits of establishing this best practice: By dedicating time and resources within your processes, you can 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.
- 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.
- Simulate production workloads through game days, and use learnings from these simulations to improve.

Resources

Related best practices:

- OPS05-BP08 Use multiple environments

Related videos:

- AWS re:Invent 2023 - Improve application resilience with AWS Fault Injection Service

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
- Chris Kozlowski, Senior Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Alex Livingstone, Principal Specialist Solutions Architect, Cloud Operations, Amazon Web Services
- Paul Moran, Principal Technologist, Enterprise Support, Amazon Web Services
- Peter Mullen, Advisory Consultant, Professional Services, Amazon Web Services
- Chris Pates, Senior Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Arvind Raghunathan, Principal Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Fatih (Ben) Mergen, Senior Cost Lead Solutions Architect, 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
Updated best practice guidance	Best practices were updated with new guidance in the following areas: OPS 2, OPS 5, OPS 9, and OPS 10. Guidance includes new recommendations on AWS services and generative AI.	November 6, 2024
Updated best practice guidance	Large-scale best practice updates were made throughout the pillar. Multiple consolidations of content in OPS 1, OPS 2, and OPS 3. Risk rating changes in OPS 10.	June 27, 2024
Major content update and consolidation	Content has been updated and consolidated in multiple best practice areas. Two best practice areas (OPS 4 and OPS 8) have been rewritten with new content and focus. Best practices have been updated and consolidated in the following areas: Design for operations, Mitigate deployment risks, and Understanding operation al health. Best practice area OPS 04 has been updated to Implement observability. Best	October 3, 2023

	practice area OPS 08 has been updated to Utilizing workload observability.	
Updates for new Framework	Best practices updated with prescriptive guidance and new best practices added.	April 10, 2023
Whitepaper updated	Best practices updated with new implementation guidance.	December 15, 2022
Whitepaper updated	Best practices expanded and improvement plans added.	October 20, 2022
Minor update	Small editorial update.	August 8, 2022
Whitepaper updated	Updates to reflect new AWS services and features, and latest best practices.	February 2, 2022
Updates for new Framework	Updates to reflect new AWS services and features, and latest best practices.	July 8, 2020
Whitepaper updated	Updates to reflect new AWS services and features, and updated references.	July 1, 2018
Initial publication	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*.