Operational Excellence Pillar

AWS Well-Architected Framework

Operational Excellence Pillar: AWS Well-Architected Framework

Table of Contents

Operational Excellence Pillar - AWS Well-Architected Framework

Publication date: **February 2, 2022** (*Document Revisions* (p. 38))

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, and cost-effective 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 enabled 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

The operational excellence pillar includes how your organization supports your business objectives, your ability to run workloads effectively, gain insight into their operations, and to continuously improve supporting processes and procedures to deliver business value.

Topics

Design Principles

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

- **Perform operations as code:** In the cloud, you can apply the same engineering discipline that you use for application code to your entire environment. You can define your entire workload (applications, infrastructure, etc.) as code and update it with code. You can script your operations procedures and automate their execution by triggering them in response to events. By performing operations as code, you limit human error and enable consistent responses to events.
- **Make frequent, small, reversible changes:** Design workloads to allow components to be updated regularly to increase the flow of beneficial changes into your workload. Make changes in small increments that can be reversed if they fail to aid in the identification and resolution of issues introduced to your environment (without affecting customers when possible).
- **Refine operations procedures frequently:** As you use operations procedures, look for opportunities to improve them. As you evolve your workload, evolve your procedures appropriately. Set up regular game days to review and validate that all procedures are effective and that teams are familiar with them.
- **Anticipate failure:** Perform “pre-mortem” exercises to identify potential sources of failure so that they can be removed or mitigated. Test your failure scenarios and validate your understanding of their impact. Test your response procedures to ensure they are effective and that teams are familiar with their execution. Set up regular game days to test workload and team responses to simulated events.
- **Learn from all operational failures:** Drive improvement through lessons learned from all operational events and failures. *Share what is learned* across teams and through the entire organization.

Definition

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

- Organization
- Prepare
- Operate
- Evolve

Your organization’s leadership defines business objectives. Your organization must understand requirements and priorities and use these to organize and conduct work to support the achievement of business outcomes. Your workload must emit the information necessary to support it. Implementing

services to enable integration, deployment, and delivery of your workload will enable 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 enable you to understand the health of your workload, your operations activities, and respond to incidents. Your priorities will change as your business needs and business environment changes. Use these as a feedback loop to continually drive improvement for your organization and the operation of your workload.

Organization

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

To enable operational excellence, you must understand the following:

Topics

Organization Priorities

Your teams need to have a shared understanding of your entire workload, their role in it, and shared business goals to set the priorities that will enable 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.

Evaluate external customer needs: Involve key stakeholders, including business, development, and operations teams, to determine where to focus efforts on external customer needs.

Evaluate internal customer needs: Involve key stakeholders, including business, development, and operations teams, to determine where to focus efforts on internal customer needs.

Evaluating customer needs will ensure that you have a thorough understanding of the support that is required to achieve business outcomes.

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

Evaluate governance requirements: Ensure that you are aware of guidelines or obligations defined by your organization that may mandate or emphasize specific focus. Evaluate internal factors, such as organization policy, standards, and requirements. Validate that you have mechanisms to identify changes to governance. If no governance requirements are identified, ensure that you have applied due diligence to this determination.

Evaluate external compliance requirements: Ensure that you are aware of guidelines or obligations that may mandate or emphasize specific focus. Evaluate external factors, such as regulatory compliance requirements and industry standards. Validate that you have mechanisms to identify changes to compliance requirements. If no compliance requirements are identified, ensure that you have applied due diligence to this determination.

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.

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.

Enterprise Support customers are eligible for a guided Well-Architected Review of their mission-critical workloads to measure their architectures against AWS best practices.

They are also 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.

Evaluate tradeoffs: Evaluate the impact of tradeoffs between competing interests or alternative approaches, to help make informed decisions when determining where to focus operations efforts or choosing a course of action. For example, accelerating speed to market for new features may be emphasized over cost optimization, or you may choose a relational database for non-relational data to simplify the effort to migrate a system, rather than migrating to a database optimized for your data type and updating your application.

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

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

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

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

Resources

Refer to the following resources to learn more about AWS best practices for organizational priorities.

Documentation

- AWS Trusted Advisor
- AWS Cloud Compliance

- AWS Well-Architected Framework
- AWS Business Support
- AWS Enterprise Support
- AWS Enterprise Support Entitlements
- AWS Support Cloud Operations Reviews
- AWS Cloud Adoption Framework

Operating Model

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

The needs of a team will be shaped by their industry, their organization, the makeup of the team, and the characteristics of their workload. It is unreasonable to expect a single operating model to be able to support all teams and their workloads.

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

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

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

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

Operating Model 2 by 2 Representations

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

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

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

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

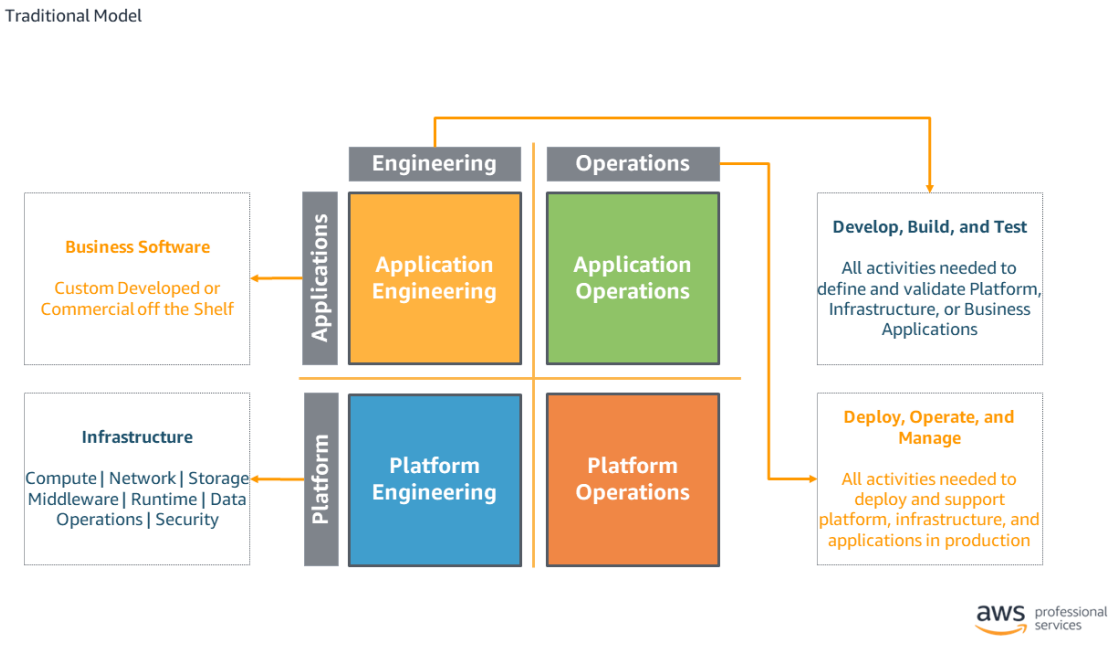
You can be successful using each of the following four operating models. Some models are more appropriate for specific use cases or at specific points in your development. Some of these models may provide advantages over the ones in use in your environment.

Topics

Fully Separated Operating model

In the following diagram, on the vertical axis we have applications and infrastructure. Applications refer to the workload serving a business outcome and can be custom developed or purchased software. Infrastructure 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.



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

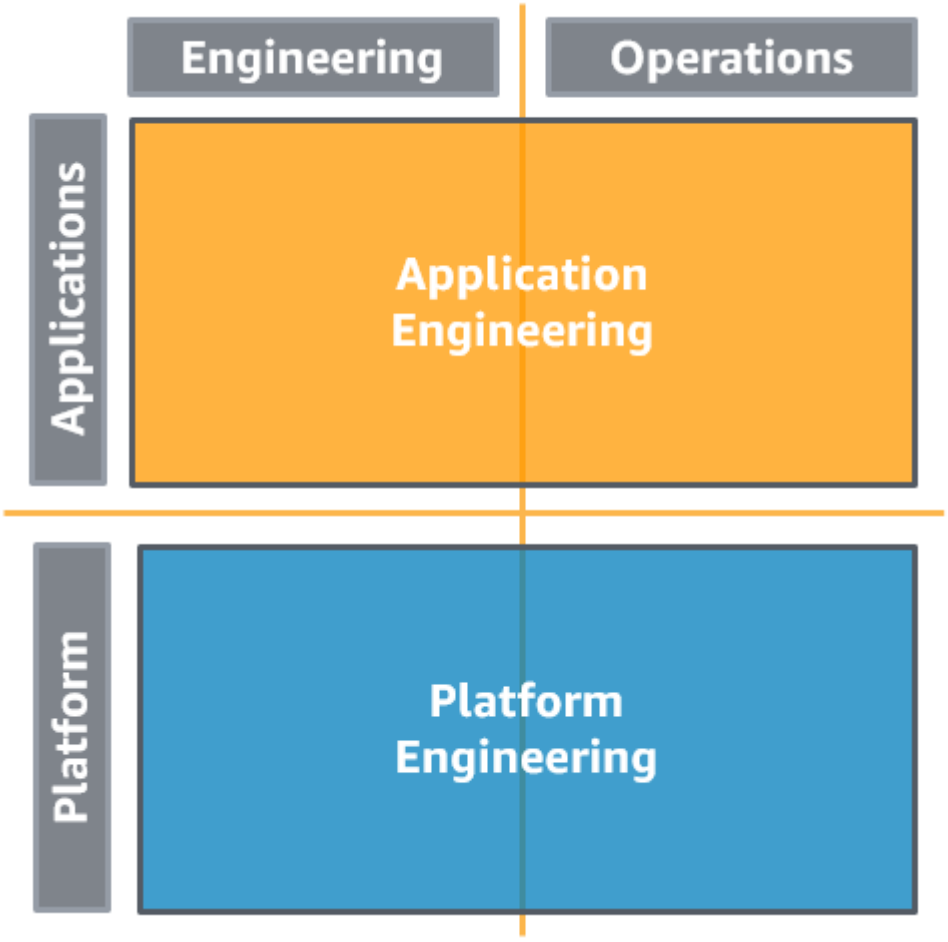
The transition of tasks to or between teams increases complexity, and creates bottlenecks and delays. Requests may be delayed until they are a priority. Defects identified late may require significant rework and may have to pass through the same teams and their functions once again. If there are incidents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

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

Separated Application Engineering and Operations (AEO) and Infrastructure Engineering and Operations (IEO) with Centralized Governance

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

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



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

You should use tools or services that enable you to centrally govern your environments across accounts, such as AWS Organizations. Services like AWS Control Tower expand this management capability enabling you to define blueprints (supporting your operating models) for the setup of accounts, apply ongoing governance using AWS Organizations, and automate provisioning of new accounts.

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

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

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

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

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

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

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

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

Separated AEO and IEO with Centralized Governance and a Service Provider

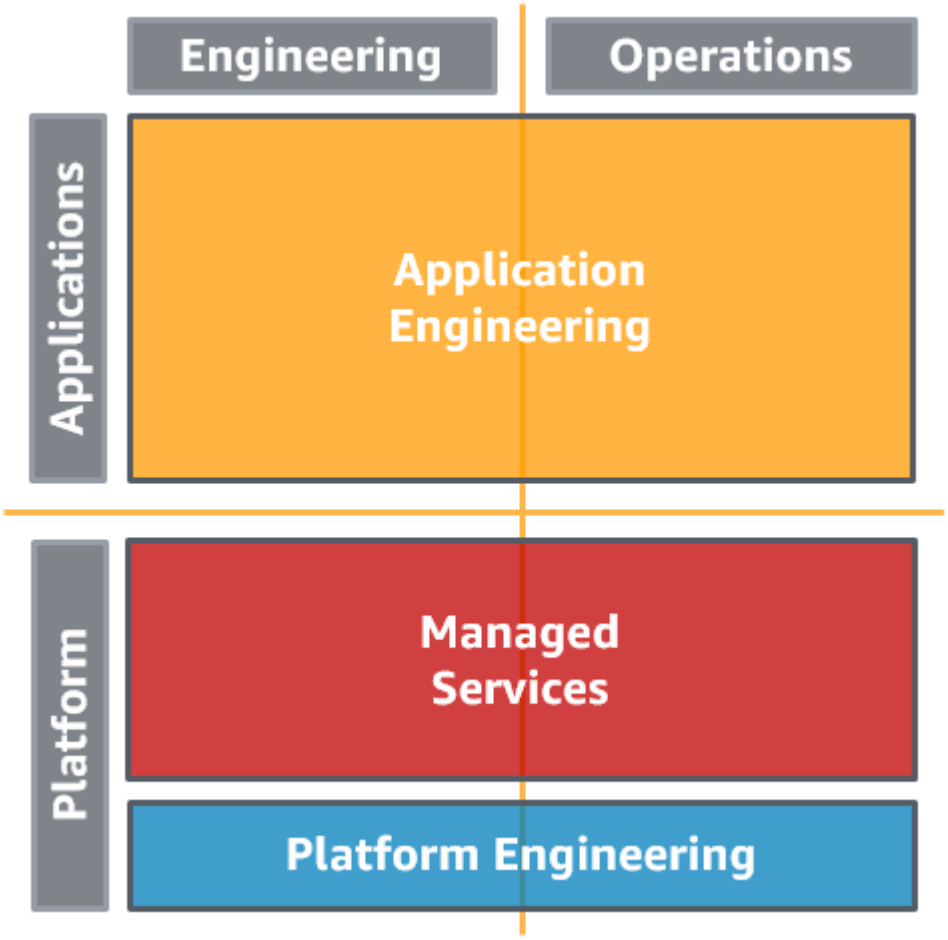
This "Separated AEO and IEO" model follows a "you build it you run it" methodology.

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

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

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

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



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

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

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

Adding Managed Services to your operating model can save you time and resources, and lets you keep your internal teams lean and focused on strategic outcomes that will differentiate your business, rather than developing new skills and capabilities.

Separated AEO and IEO with Centralized Governance and an Internal Service Provider Consulting Partner

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

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

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

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

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

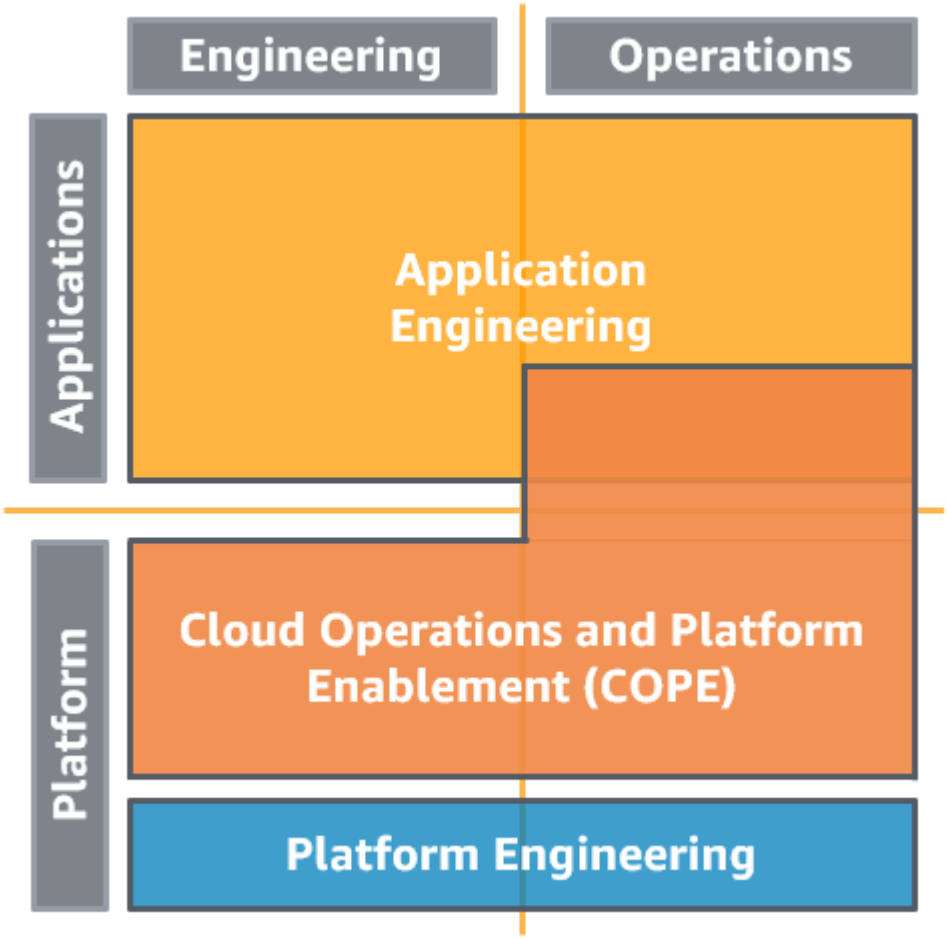
Your platform engineering team builds the core shared platform capabilities based on those standards for application teams to adopt. They codify the enterprise reference architectures and patterns that are provided to the application teams through a self-service mechanism. Using a service such as AWS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

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

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

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

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



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

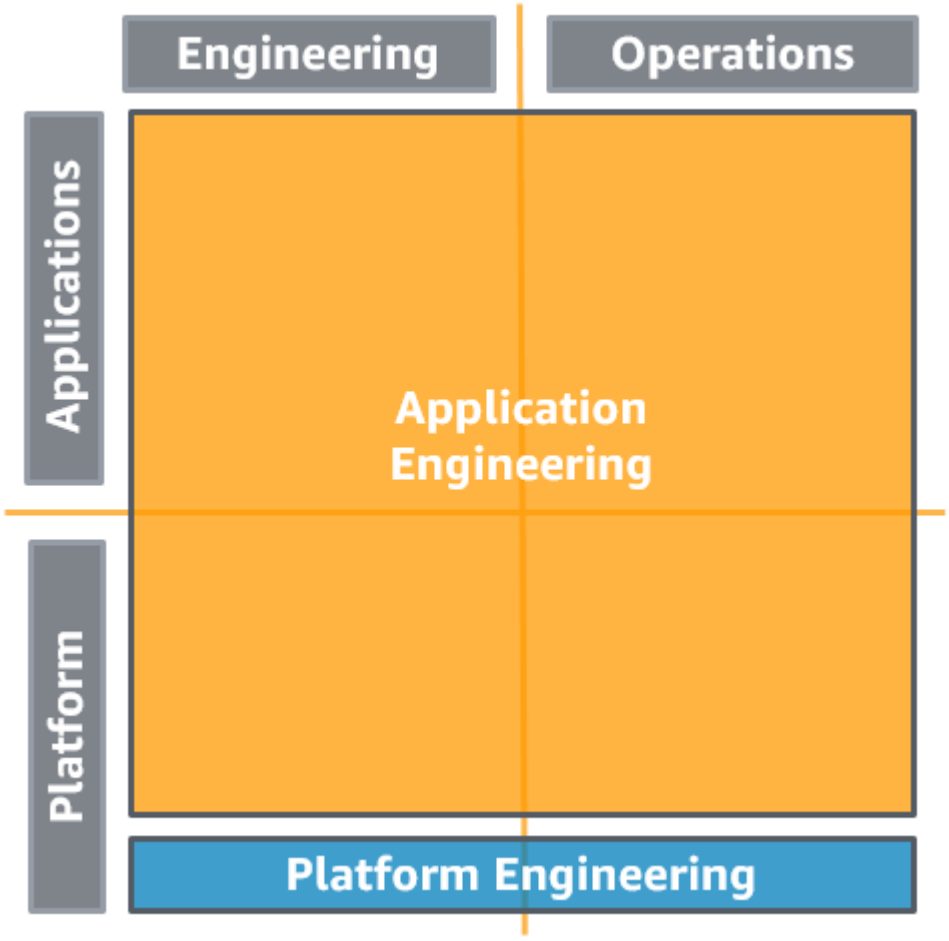
You gain the advantage of your organization's standards, best practices, processes, and expertise derived from the successes of your teams. You establish a mechanism to replicate these successful patterns for new teams adopting or modernizing on the cloud. This model places emphasis on the COPE team's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

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

Separated AEO and IEO with Decentralized Governance

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

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



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

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

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

You should enforce policies that are not specifically delegated to application teams. Use tools or services that enable you to centrally govern your environments across accounts, such as AWS Organizations. Services like AWS Control Tower expand this management capability enabling you to define blueprints (supporting your operating models) for the setup of accounts, apply ongoing governance using AWS Organizations, and automate provisioning of new accounts.

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

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

Relationships and Ownership

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

Resources have identified owners: Understand who has ownership of each application, workload, platform, and infrastructure component, what business value is provided by that component, and why that ownership exists. Understanding the business value of these individual components and how they support business outcomes informs the processes and procedures applied against them.

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 enables identification of improvement opportunities.

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 responsibility for performance of operations activities informs who will perform the action, validate the result, and provide feedback to the owner of the activity.

Team members know what they are responsible for: Understanding your role informs the prioritization of your tasks. This enables team members to recognize needs and respond appropriately.

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

Mechanisms exist to request additions, changes, and exceptions: You are able to make requests to owners of processes, procedures, and resources. Make informed decisions to approve requests where viable and determined to be appropriate after an evaluation of benefits and risks.

Responsibilities between teams are predefined or negotiated: There are 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). 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 enables them to 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.

Resources

Refer to the following resources to learn more about AWS best practices for operations design.

Videos

- AWS re:Invent 2019: [REPEAT 1] How to ensure configuration compliance (MGT303-R1)
- AWS re:Invent 2019: Automate everything: Options and best practices (MGT304)

Documentation

- AWS Managed Services
- AWS Organizations Features
- AWS Control Tower Features

Organizational Culture

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

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

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

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

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

Planned events can be recorded in a change calendar or maintenance schedule so that team members can identify what activities are pending.

On AWS, AWS Systems Manager Change Calendar can be used to record these details. It supports programmatic checks of calendar status to determine if the calendar is open or closed to activity at a particular point of time. Operations activities may be planned around specific “approved” windows of time that are reserved for potentially disruptive activities. AWS Systems Manager Maintenance Windows allows you to schedule activities against instances and other supported resources to automate the activities and make those activities discoverable.

Experimentation is encouraged: Experimentation accelerates learning and keeps team members interested and engaged. An undesired result is a successful experiment that has identified a path that will not lead to success. Team members are not punished for successful experiments with undesired results. Experimentation is required for innovation to happen and turn ideas into outcomes.

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

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

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

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

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

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

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

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

Resources

Refer to the following resources to learn more about AWS best practices for operations design.

Videos

- AWS re:Invent 2019: [REPEAT 1] How to ensure configuration compliance (MGT303-R1)
- AWS re:Invent 2019: Automate everything: Options and best practices (MGT304)

Documentation

- AWS Managed Services
- AWS Organizations Features
- AWS Control Tower Features

Prepare

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

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

Topics

Design Telemetry

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

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

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

Implement application telemetry: Instrument your application code to emit information about its internal state, status, and achievement of business outcomes, for example, queue depth, error messages, and response times. Use this information to determine when a response is required.

Use centralized and structured logging: Standardize your application logging to emit operational information about transactions, correlation identifiers, request identifiers across components, and business outcomes. Use this information to answer arbitrary questions about the state of your workload.

The following is an example of structured logging using JSON as the output.

```
{
"TIMESTAMP":"2019-11-26 18:17:33,774",
"LEVEL":"INFO",
"LOCATION":"CANCEL.CANCEL_BOOKING:45",
"SERVICE":"BOOKING",
"LAMBDA_FUNCTION_NAME":"TEST",
"LAMBDA_FUNCTION_MEMORY_SIZE":"128",
"LAMBDA_FUNCTION_ARN":"ARN:AWS:LAMBDA:EU-WEST-1: 12345678910:FUNCTION:TEST",
"COLD_START": "TRUE",
"MESSAGE":{
```

```
  "OPERATION":"UPDATE_ITEM",
  "DETAILS:":{
   "ATTRIBUTES":{
    "STATUS":"CANCELLED"
   },
   "RESPONSEMETADATA":{
    "REQUESTID":"G7S3SCFDEMEINPG6AOC6CL5IDNVV4KQNSO5AEMVJF66Q9ASUAAJG",
    "HTTPSTATUSCODE":200,
    "HTTPHEADERS":{
     "SERVER":"SERVER",
     "DATE":"THU, 26 NOV 2019 18:17:33 GMT",
     "CONTENT-TYPE":"APPLICATION/X-AMZ-JSON-1.0",
     "CONTENT-LENGTH":"43",
     "CONNECTION":"KEEP-ALIVE",
     "X-AMZN-REQUESTID":"G7S3SCFDEMEINPG6AOC6CL5IDNVV4KQNSO5AEMVJF66Q9ASUAAJG",
     "X-AMZ-CRC32":"1848747586"
    "RETRYATTEMPTS":0
   }
  }
 }
}
```

Centralized logging helps you search and analyze your serverless application logs. Structured logging makes it easier to derive queries to answer arbitrary questions about the health of your application. As your system grows and more logging is ingested, consider using appropriate logging levels and a sampling mechanism to log a small percentage of logs in DEBUG mode.

You should install and configure the Unified Amazon CloudWatch Agent to send system level application logs and advanced metrics from your EC2 instances and physical servers to Amazon CloudWatch.

Generate and publish custom metrics using the AWS CLI or the CloudWatch API. Ensure that you publish insightful business metrics as well as technical metrics to help you understand your customers' behaviors.

You can send logs directly from your application to CloudWatch using the CloudWatch Logs API, or send events using the AWS SDK and Amazon EventBridge. Insert logging statements into your AWS Lambda code to automatically store them in CloudWatch Logs.

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

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

Implement user activity telemetry: Instrument your application code to emit information about user activity, for example, click streams, or started, abandoned, and completed transactions. Use this information to help understand how the application is used, patterns of usage, and to determine when a response is required.

Implement dependency telemetry: Design and configure your workload to emit information about the status (for example, reachability or response time) of resources it depends on. Examples of external dependencies can include external databases, DNS, and network connectivity. Use this information to determine when a response is required.

Implement transaction traceability: Implement your application code and configure your workload components to emit information about the flow of transactions across the workload. Use this information to determine when a response is required and to assist you in identifying the factors contributing to an issue.

On AWS, you can use distributed tracing services, such as AWS X-Ray, to collect and record traces as transactions travel through your workload, generate maps to see how transactions flow across your workload and services, gain insight to the relationships between components, and identify and analyze issues in real time.

Iterate and develop telemetry as workloads evolve to ensure that you continue to receive the information necessary to gain insight to the health of your workload.

Resources

Refer to the following resources to learn more about AWS best practices for operations design.

Videos

AWS re:Invent 2016: Infrastructure Continuous Delivery Using AWS CloudFormation (DEV313)

AWS re:Invent 2016: DevOps on AWS: Accelerating Software Delivery with AWS Developer Tools (DEV201)

AWS CodeStar: The Central Experience to Quickly Start Developing Applications on AWS

Documents

- Accessing Amazon CloudWatch Logs for AWS Lambda
- Monitoring CloudTrail Log Files with Amazon CloudWatch Logs
- Publishing Flow Logs to CloudWatch Logs

Documentation

- Enhancing workload observability using Amazon CloudWatch Embedded Metric Format
- Getting Started With Amazon CloudWatch
- Store and Monitor OS & Application Log Files with Amazon CloudWatch
- High-Resolution Custom Metrics and Alarms for Amazon CloudWatch
- Monitoring AWS Health Events with Amazon CloudWatch Events
- AWS CloudFormation Documentation
- AWS Developer Tools
- Set Up a CI/CD Pipeline on AWS
- AWS X-Ray
- Enhancing workload observability using Amazon CloudWatch Embedded Metric Format

Design for Operations

Adopt approaches that improve the flow of changes into production and that enable refactoring, fast feedback on quality, and bug fixing. These accelerate beneficial changes entering production, limit issues deployed, and enable 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.

Use version control: Use version control to enable tracking of changes and releases.

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

Test and validate changes: Test and validate changes to help limit and detect errors. Automate testing to reduce errors caused by manual processes, and reduce the level of effort to test.

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

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.

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 like the AWS Developer Tools (for example, AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeDeploy, and AWS CodeStar).

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

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

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

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

Share design standards: Share best practices across teams to increase awareness and maximize the benefits of development efforts.

On AWS, application, compute, infrastructure, and operations can be defined and managed using code methodologies. This allows for easy release, sharing, and adoption.

Many AWS services and resources are designed to be shared across accounts, enabling you to share created assets and learnings across your teams. For example, you can share CodeCommit repositories, Lambda functions, Amazon S3 buckets, and AMIs to specific accounts.

When you publish new resources or updates, use Amazon SNS to provide cross account notifications. Subscribers can use Lambda to get new versions.

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

Implement practices to improve code quality: Implement practices to improve code quality and minimize defects, such as test-driven development, code reviews, and standards adoption. Automate the performance of quality checks in response to pull requests where possible to ensure consistent reviews.

On AWS, you can integrate services such as Amazon CodeGuru with your pipeline to automatically identify potential code and security issues using program analysis and machine learning. CodeGuru provides recommendations on how to implement the AWS best practices to address these issues.

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

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

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 enable identification of your resources. Tag your resources for organization, cost accounting, access controls, and targeting the execution of automated operations activities.

Resources

Refer to the following resources to learn more about AWS best practices for operations design.

Videos

- AWS re:Invent 2016: Infrastructure Continuous Delivery Using AWS CloudFormation (DEV313)
- AWS re:Invent 2016: DevOps on AWS: Accelerating Software Delivery with AWS Developer Tools (DEV201)
- AWS CodeStar: The Central Experience to Quickly Start Developing Applications on AWS

Documentation

- What Is AWS Resource Groups
- AWS CloudFormation Documentation
- AWS Developer Tools
- Set Up a CI/CD Pipeline on AWS
- Find Your Most Expensive Lines of Code with Amazon CodeGuru

Mitigate Deployment Risks

Adopt approaches that provide fast feedback on quality and enable 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

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

Resources

Refer to the following resources to learn more about AWS best practices for operations design.

Videos

- AWS re:Invent 2016: Infrastructure Continuous Delivery Using AWS CloudFormation (DEV313)
- AWS re:Invent 2016: DevOps on AWS: Accelerating Software Delivery with AWS Developer Tools (DEV201)
- AWS CodeStar: The Central Experience to Quickly Start Developing Applications on AWS

Documentation

- AWS CloudFormation Documentation
- AWS Developer Tools
- Decision-making framework for configuration with AWS AppConfig
- Set Up a CI/CD Pipeline on AWS

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 enable 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.

Ensure personnel capability: Have a mechanism to validate that you have the appropriate number of trained personnel to provide support for operational needs. Train personnel and adjust personnel capacity as necessary to maintain effective support.

You will need to have enough team members to cover all activities (including on-call). Ensure that your teams have the necessary skills to be successful with training on your workload, your operations tools, and AWS.

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

Ensure consistent review of operational readiness: Ensure you have a consistent review of your readiness to operate a workload. Reviews must include, at a minimum, the operational readiness of the teams and the workload, and security requirements. Implement review activities in code and initiate automated review in response to events where appropriate, to ensure consistency, speed of execution, and reduce errors caused by manual processes.

You should automate workload configuration testing by making baselines using AWS Config and checking your configurations using AWS Config Rules. You can evaluate security requirements and compliance using the services and features of AWS Security Hub. These services will aid in determining if your workloads are aligned with best practices and standards.

Manage the flow of changes in your environments: Manage change to support delivery of value to your customers. By managing changes you can remove conflicts from activities to prevent unintentional impacts, mitigate the impact of failed changes, avoid unauthorized changes, and ensure appropriate testing, validation, and auditability of changes. Automate your change process where possible to support increased velocity of change.

The participants and mechanisms used to approve changes can vary by type of change, level of risk, or the specific change scenario, and the needs of your organization. We will explore these concepts further and look at examples below.

Your organization may treat all changes the same way and follow a consistent evaluation and approval mechanism. If your organization wants or is required to differentiate change types, potentially with different approval mechanisms, common categories include emergency, normal, and standard. Emergency changes are performed as part of incident response or when there is an imminent threat to a production workload. Normal changes follow the full set of tests, validation, and oversight you have defined as appropriate to the change. Standard changes are former “normal” changes that have been approved to follow a reduced approval mechanism, or have received pre-approval, based on a proven track record of successful implementation with acceptable risk.

For example, you may have an extremely robust software development pipeline, including automated testing and standards enforcement, that satisfies all the testing and validation requirements for your application. After establishing a history of consistent successful workload deployments using your pipeline, verified as compliant with your change control standards, you choose to pre-approve all changes to the application code as “standard” changes. Doing so enables continuous deployment for changes of those type and can enable improved velocity of beneficial changes entering production.

In contrast, you have multiple business critical systems that are dependent upon a single common resource. You may choose to apply additional controls and oversight to changes to that resource because of the significant potential for customer impact. For example, you decide to not allow changes to dependent systems when a change to the dependency is being made, in order to make it easier to identify unintended consequences from that change. You configure your pipelines to check for an active conflicting deployment and to wait until it is complete before proceeding.

You can choose to define specific approval mechanisms and approvers for changes based on the amount of risk inherent in the change (such as, high, medium, or low), or you can choose to define specific approval mechanisms based on the scenario of the change. A risk based approach might be simpler to implement with only three approval mechanisms (for example, high, medium, and low) and a predefined set of approvers for each. A scenario-based approach has an approval mechanism for each scenario, so there can be many types of changes, but it streamlines the participants to those needed for the scenario. It ensures additional fidelity by specifically selecting and engaging subject matter experts appropriate to evaluate and approve the change.

On AWS, you can manage and track your changes within a single AWS account, or across your AWS Organizations managed accounts (from a single delegated administrator account), using AWS Systems Manager Change Manager. Using Systems Manager Automation Runbooks, changes can be preapproved or request approval from specific defined reviewers. You can remove conflicts from changes against scheduled activities and business events using integrations with Systems Manager Change Calendar.

Use metadata to identify your resources: Apply metadata using Resource Tags and AWS Resource Groups following a consistent tagging strategy to enable identification of your persistent and ephemeral resources. Use tags to enable organization, cost accounting, access controls, and targeting the execution of automated operations activities.

Use runbooks to perform procedures: Runbooks are documented procedures to achieve specific outcomes. Enable consistent and prompt responses to well-understood events by documenting

procedures in runbooks. Implement runbooks as code and initiate the running of runbooks in response to events where appropriate, to ensure consistency, speed responses, and reduce errors caused by manual processes.

Use playbooks to identify issues: Playbooks are documented processes to investigate issues. Enable consistent and prompt responses to failure scenarios by documenting investigation processes in playbooks. Implement playbooks as code and initiate playbook execution in response to events where appropriate, to ensure consistency, speed responses, and reduce errors caused by manual processes.

AWS allows you to treat your operations as code, scripting your runbook and playbook activities to reduce the risk of human error. You can use Resource Tags or Resource Groups with your scripts to selectively execute based on criteria you have defined (for example, environment, owner, role, or version).

You can use scripted procedures to enable automation by starting the scripts in response to events. By treating both your operations and workloads as code, you can also script and automate the evaluation of your environments.

You should script procedures on your instances using AWS Systems Manager (SSM) Run Command, use AWS Systems Manager Automation to script actions and create workflows on instances and other resources, or use AWS Lambda serverless compute functions to script responses to events across AWS service APIs and your own custom interfaces. You can also use AWS Step Functions to coordinate multiple AWS services scripted into serverless workflows. Automate your responses by initiating these scripts using CloudWatch Events and route desired events to additional operations support systems using Amazon EventBridge.

You should test your procedures, failure scenarios, and the success of your responses (for example, by holding game days and testing prior to going live) to identify areas you need to plan to address.

On AWS, you can create temporary parallel environments to lower the risk, effort, and cost of experimentation and testing. Automate the deployment of these environments using AWS CloudFormation to ensure consistent implementations of your temporary environments. Perform failure injection testing in safe environments where there will be acceptable or no customer impact, and develop or revise appropriate responses.

Make informed decisions to deploy systems and changes: Evaluate the capabilities of the team to support the workload and the workload's compliance with governance. Evaluate these against the benefits of deployment when determining whether to transition a system or change into production. Understand the benefits and risks to make informed decisions.

Use “pre-mortems” to anticipate failure and create procedures where appropriate. 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.

Resources

Refer to the following resources to learn more about AWS best practices for operational readiness.

Documentation

- AWS Systems Manager
- AWS Config Rules – Dynamic Compliance Checking for Cloud Resources
- How to track configuration changes to AWS CloudFormation stacks using AWS Config
- Amazon Inspector Update blog post
- AWS Systems Manager Change Manager
- Using AWS Systems Manager Change Calendar to prevent changes during critical events

- AWS Lambda
- AWS Events and Webinars
- AWS Training
- AWS Well-Architected Labs
- AWS launches Tag Policies

Operate

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

To be successful, you must be able to:

Topics

Understanding Workload Health

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

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

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

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

Define workload metrics: Define workload metrics to measure the achievement of KPIs (for example, abandoned shopping carts, orders placed, cost, price, and allocated workload expense). Define workload metrics to measure the health of the workload (for example, interface response time, error rate, requests made, requests completed, and utilization). Evaluate metrics to determine if the workload is achieving desired outcomes, and to understand the health of the workload.

You should send log data to a service like CloudWatch Logs, and generate metrics from observations of necessary log content.

CloudWatch has specialized features like Amazon CloudWatch Insights for .NET and SQL Server and Container Insights that can assist you by identifying and setting up key metrics, logs, and alarms across your specifically supported application resources and technology stack.

Collect and analyze workload metrics: Perform regular proactive reviews of metrics to identify trends and determine where appropriate responses are needed.

You should aggregate log data from your application, workload components, services, and API calls to a service like CloudWatch Logs. Generate metrics from observations of necessary log content to enable insight into the performance of operations activities.

On AWS, you can analyze workload metrics and identify operational issues using the machine learning capabilities of Amazon DevOps Guru. AWS DevOps Guru provides notification of operational issues with targeted and proactive recommendations to resolve issues and maintain application health.

In the AWS shared responsibility model, portions of monitoring are delivered to you through the AWS Personal Health Dashboard. This dashboard provides alerts and remediation guidance when AWS is experiencing events that might affect you. Customers with Business and Enterprise Support subscriptions also get access to the AWS Health API, enabling integration to their event management systems.

On AWS, you can export your log data to Amazon S3 or send logs directly to Amazon S3 for long-term storage. Using AWS Glue, you can discover and prepare your log data in Amazon S3 for analytics, storing associated metadata in the AWS Glue Data Catalog. Amazon Athena, through its native integration with 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.

An alternative solution would be to use the Amazon OpenSearch Service and OpenSearch Dashboards to collect, analyze, and display logs on AWS across multiple accounts and AWS Regions.

Establish workload metrics baselines: Establish baselines for metrics to provide expected values as the basis for comparison and identification of under and over performing components. Identify thresholds for improvement, investigation, and intervention.

Learn expected patterns of activity for workload: Establish patterns of workload activity to identify anomalous behavior so that you can respond appropriately if necessary.

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

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

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

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

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

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

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

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

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

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

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

Resources

Refer to the following resources to learn more about AWS best practices for understanding workload health.

Videos

- AWS re:Invent 2015: Log, Monitor, and Analyze your IT with Amazon CloudWatch (DVO315)
- AWS re:Invent 2016: Amazon CloudWatch Logs and AWS Lambda: A Match Made in Heaven (DEV301)

Documentation

- What Is CloudWatch Application Insights for .NET and SQL Server?
- Store and Monitor OS & Application Log Files with CloudWatch
- How to set up CloudWatch Anomaly Detection to set dynamic alarms, automate actions, and drive online sales
- Gaining operational insights with AIOps using Amazon DevOps Guru
- API & CloudFormation Support for CloudWatch Dashboards
- AWS Answers: Centralized Logging

Understanding Operational Health

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

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

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

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

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

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

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

On AWS, you can export your log data to Amazon S3 or send logs directly to Amazon S3 for long-term storage. Using AWS Glue, you can discover and prepare your log data in Amazon S3 for analytics, storing

associated metadata in the 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.

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

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

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

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

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

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

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

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

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

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

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

Resources

Refer to the following resources to learn more about AWS best practices for understanding operational health.

Videos

- AWS re:Invent 2015: Log, Monitor, and Analyze your IT with Amazon CloudWatch (DVO315)
- AWS re:Invent 2016: Amazon CloudWatch Logs and AWS Lambda: A Match Made in Heaven (DEV301)

Documentation

- Store and Monitor OS & Application Log Files with Amazon CloudWatch
- API & CloudFormation Support for Amazon CloudWatch Dashboards

- How to set up CloudWatch Anomaly Detection to set dynamic alarms, automate actions, and drive online sales
- Gaining operational insights with AIOps using Amazon DevOps Guru
- AWS Answers: Centralized Logging

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 trigger their execution 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.

Use processes for event, incident, and problem management: Have processes to address observed events, events that require intervention (incidents), and events that require intervention and either recur or cannot currently be resolved (problems). Use these processes to mitigate the impact of these events on the business and your customers by ensuring timely and appropriate responses.

On AWS, you can use AWS Systems Manager OpsCenter as a central location to view, investigate, and resolve operational issues related to any AWS resource. It aggregates operational issues and provides contextually relevant data to assist in incident response.

Have a process per alert: Have a well-defined response (runbook or playbook), with a specifically identified owner, for any event for which you raise an alert. This ensures effective and prompt responses to operations events and prevents actionable events from being obscured by less valuable notifications.

Prioritize operational events based on business impact: Ensure that when multiple events require intervention, those that are most significant to the business are addressed first. For example, impacts can include loss of life or injury, financial loss, or damage to reputation or trust.

Define escalation paths: Define escalation paths in your runbooks and playbooks, including what triggers escalation, and procedures for escalation. Specifically identify owners for each action to ensure effective and prompt responses to operations events.

Identify when a human decision is required before an action is taken. Work with decision makers to have that decision made in advance, and the action preapproved, so that MTTR is not extended waiting for a response.

Enable push notifications: Communicate directly with your users (for example, with email or SMS) when the services they use are impacted, and again when the services return to normal operating conditions, to enable users to take appropriate action.

Communicate status through dashboards: Provide dashboards tailored to their target audiences (for example, internal technical teams, leadership, and customers) to communicate the current operating status of the business and provide metrics of interest.

You can create dashboards using Amazon CloudWatch Dashboards on customizable home pages in the CloudWatch console. Using business intelligence services like Amazon QuickSight you can create

and publish interactive dashboards of your workload and operational health (for example, order rates, connected users, and transaction times). Create Dashboards that present system and business-level views of your metrics.

Automate responses to events: Automate responses to events to reduce errors caused by manual processes, and to ensure prompt and consistent responses.

There are multiple ways to automate the execution of runbook and playbook actions on AWS. To respond to an event from a state change in your AWS resources, or from your own custom events, you should create CloudWatch Events rules to trigger responses through CloudWatch targets (for example, Lambda functions, Amazon Simple Notification Service (Amazon SNS) topics, Amazon ECS tasks, and AWS Systems Manager Automation).

To respond to a metric that crosses a threshold for a resource (for example, wait time), you should create CloudWatch alarms to perform one or more actions using Amazon EC2 actions, Auto Scaling actions, or to send a notification to an Amazon SNS topic. If you need to perform custom actions, in response to an alarm, invoke Lambda through Amazon SNS notification. Use Amazon SNS to publish event notifications and escalation messages to keep people informed.

AWS also supports third-party systems through the AWS service APIs and SDKs. There are a number of monitoring tools provided by APN Partners and third parties that allow for monitoring, notifications, and responses. Some of these tools include New Relic, Splunk, Loggly, SumoLogic, and Datadog.

You should keep critical manual procedures available for use when automated procedures fail.

Resources

Refer to the following resources to learn more about AWS best practices for responding to events.

Video

- AWS re:Invent 2016: Automating Security Event Response, from Idea to Code to Execution (SEC313)

Documentation

- What is Amazon CloudWatch Events?
- How to Automatically Tag Amazon EC2 Resources in Response to API Events
- Amazon EC2 Systems Manager Automation is now an Amazon CloudWatch Events Target
- EC2 Run Command is Now a Amazon CloudWatch Events Target
- Automate remediation actions for Amazon EC2 notifications and beyond using EC2 Systems Manager Automation and AWS Health
- High-Resolution Custom Metrics and Alarms for Amazon CloudWatch

Evolve

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

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

Topics

Learn, Share, and Improve

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

Have a process for continuous improvement: Regularly evaluate and prioritize opportunities for improvement to focus efforts where they can provide the greatest benefits.

Perform post-incident analysis: Review customer-impacting events, and identify the contributing causes and preventative action items. Use this information to develop mitigations to limit or prevent recurrence. Develop procedures for prompt and effective responses. Communicate contributing factors and corrective actions as appropriate, tailored to target audiences.

Implement feedback loops: Include feedback loops in your procedures and workloads to help you identify issues and areas that need improvement.

Perform Knowledge Management: Mechanisms exist for your team members to discover the information that they are looking for in a timely manner, access it, and identify that it's current and complete. Mechanisms are present to identify needed content, content in need of refresh, and content that should be archived so that it's no longer referenced.

Define drivers for improvement: Identify drivers for improvement to help you evaluate and prioritize opportunities.

On AWS, you can aggregate the logs of all your operations activities, workloads, and infrastructure to create a detailed activity history. You can then use AWS tools to analyze your operations and workload health over time (for example, identify trends, correlate events and activities to outcomes, and compare and contrast between environments and across systems) to reveal opportunities for improvement based on your drivers.

You should use CloudTrail to track API activity (through the AWS Management Console, CLI, SDKs, and APIs) to know what is happening across your accounts. Track your AWS Developer Tools deployment activities with CloudTrail and CloudWatch. This will add a detailed activity history of your deployments and their outcomes to your CloudWatch Logs log data.

Export your log data to Amazon S3 for long-term storage. Using AWS Glue, you discover and prepare your log data in Amazon S3 for analytics. Use Amazon Athena, through its native integration with AWS Glue, to analyze your log data. Use a business intelligence tool like Amazon QuickSight to visualize, explore, and analyze your data.

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.

Perform operations metrics reviews: Regularly perform retrospective analysis of incidents and operations metrics with cross-team participants, including leadership, 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).

Document and share lessons learned: Document and share lessons learned from the execution of operations activities so that you can use them internally and across teams.

You should share what your teams learn to increase the benefit across your organization. You will want to share information and resources to prevent avoidable errors and ease development efforts. This will allow you to focus on delivering desired features.

Use AWS Identity and Access Management (IAM) to define permissions enabling controlled access to the resources you wish to share within and across accounts. You should then use version-controlled AWS CodeCommit repositories to share application libraries, scripted procedures, procedure documentation, and other system documentation. Share your compute standards by sharing access to your AMIs and by authorizing the use of your Lambda functions across accounts. You should also share your infrastructure standards as AWS CloudFormation templates.

Through the AWS APIs and SDKs, you can integrate external and third-party tools and repositories (for example, GitHub, BitBucket, and SourceForge).

When sharing what you have learned and developed, be careful to structure permissions to ensure the integrity of shared repositories.

Allocate time to make improvements: Dedicate time and resources within your processes to make continuous incremental improvements possible.

On AWS, you can create temporary duplicates of environments, lowering the risk, effort, and cost of experimentation and testing. These duplicated environments can be used to test the conclusions from your analysis, experiment, and develop and test planned improvements.

Resources

Refer to the following resources to learn more about AWS best practices for learning from experience.

Documentation

- Querying Amazon VPC Flog Logs
- Monitoring Deployments with Amazon CloudWatch Tools
- Analyzing VPC Flow Logs with Amazon Kinesis Data Firehose, Amazon Athena, and Amazon QuickSight
- Share an AWS CodeCommit Repository
- Use resource-based policies to give other accounts and AWS services permission to use your Lambda resources
- Sharing an AMI with Specific AWS Accounts
- Using AWS Lambda with Amazon SNS

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 enable 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

- Brian Carlson, Operations 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, Sr. Innovation Tech Consultant, Amazon Web Services
- Aden Leirer, Well-Architected Program Manager, Content, 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.

update-history-change	update-history-description	update-history-date
Whitepaper updated (p. 38)	Updates to reflect new AWS services and features, and latest best practices.	February 2, 2022
Minor update (p. 1)	Added Sustainability Pillar to introduction.	December 2, 2021
Updates for new Framework (p. 38)	Updates to reflect new AWS services and features, and latest best practices.	July 8, 2020
Whitepaper updated (p. 38)	Updates to reflect new AWS services and features, and updated references.	July 1, 2018
Initial publication (p. 38)	Operational Excellence Pillar - AWS Well-Architected Framework published.	November 1, 2017