AWS Well-Architected Framework

Microsoft Workloads Lens - AWS Well-Architected Framework

Microsoft Workloads Lens - AWS Well-Architected Framework: AWS Well-Architected Framework

Table of Contents

Microsoft Workloads Lens - AWS Well-Architected Framework

Publication date: **February 2, 2026** (???)

This whitepaper describes the Microsoft Workloads Lens for the AWS Well-Architected Framework. It provides AWS customers with a set of Well-Architected best practices and guidance on cloud-based architectures and better understand the impact of design decisions. The document provides general design principles and specific best practices aligned to the six pillars of the Well-Architected Framework.

The AWS Well-Architected Microsoft Workloads Lens helps organizations understand and assess the advantages and challenges of decisions made while building Microsoft-based systems on AWS. By using this specialized lens, you will learn architectural best practices specifically tailored for designing and operating reliable, secure, efficient, and cost-effective Microsoft workloads in the cloud. It provides a systematic approach to consistently measure your Microsoft architectures against proven best practices and identify areas for improvement. This process involves constructive conversations about architectural decisions related to Microsoft technologies, rather than serving as an audit mechanism. AWS believes that well-architected workloads significantly increase the likelihood of business success.

AWS Solutions Architects have extensive experience architecting Microsoft solutions across diverse business verticals, technology stacks, and customer use cases. Through designing and reviewing thousands of customers' Microsoft architectures running on AWS, we have identified comprehensive best practices and core strategies for architecting Microsoft systems in the cloud. This expertise encompasses Windows Server deployments, .NET applications, SQL Server databases, Active Directory implementations, and various other Microsoft technologies and services.

The Microsoft Workloads Lens documents a set of foundational questions that help you understand if your Microsoft-specific architecture aligns with cloud best practices. It provides a consistent approach to evaluating Microsoft systems against the qualities expected from modern cloud-based solutions, along with detailed remediation guidance. The lens includes specialized insights for optimizing Microsoft workload performance, maintaining security, managing licensing, and maintaining operational excellence on AWS.

The Microsoft Workloads Lens extends the core Well-Architected Framework by incorporating best practices specific to Microsoft technologies and scenarios. As an extension of the Framework, it complements rather than overlaps with the Framework's fundamental best practices. Organizations should conduct a Well-Architected Framework Review before applying this specialized lens to their Microsoft workloads. This approach provides for a comprehensive evaluation of both general cloud architecture principles and Microsoft-specific considerations.

This lens is designed for technology professionals, including architects, developers, and operations teams working with Microsoft technologies on AWS. It provides detailed AWS best practices and strategies for designing and operating Microsoft cloud workloads, with extensive links to implementation details and architectural patterns. Additionally, the lens can be accessed through the AWS Well-Architected Tool, allowing organizations to systematically review and measure their Microsoft architectures. To support practical implementation, AWS offers Well-Architected Labs with hands-on examples and collaborates with AWS Partner Network (APN) Partners who specialize in Microsoft workloads. These partners can provide expert guidance in reviewing and optimizing Microsoft-based architectures on AWS.

Lens availability

Custom lenses extend the best practice guidance provided by AWS Well-Architected Tool. AWS WA Tool allows you to create your own custom lenses, or to use lenses created by others that have been shared with you.

To begin reviewing your Microsoft workload, download and import the Microsoft Workloads Lens into AWS Well-Architected Tool from the public AWS Well-Architected custom lens GitHub repository.

Definitions

Defining key terms and concepts is crucial when architecting, migrating and operating Microsoft workloads in the AWS Cloud. This section outlines the important AWS and Microsoft workload definitions, deployment models, licensing considerations, and benefits of running Microsoft applications and services on AWS.

Microsoft Workloads definitions

- **Windows Server:** Operating system for hosting Microsoft application servers, web servers, and domain controllers.
- **SQL Server:** Relational database management system for hosting business-critical applications.
- **Active Directory:** Directory services for identity and access management of users and resources.
- **Internet Information Services (IIS):** Microsoft's web server for hosting web applications.
- **.NET Framework:** Software framework for building and running Windows applications.

Deployment models

- **Migration:** Reconfiguring existing on-premises Microsoft workloads to run on AWS infrastructure with minimal changes.
- **Hybrid:** Running part of the Microsoft workload on-premises and part in the AWS Cloud, often connected using VPN or AWS Direct Connect.
- **Cloud-based:** Architecting new Microsoft workloads to use AWS cloud-based services and capabilities.

Licensing definitions

- **License included (LI):** Using license included instances allows you access to fully compliant Microsoft software licenses bundled with Amazon EC2 or Amazon RDS instances and pay for them as you go with no upfront costs or long-term investments.
- **Bring your own license (BYOL):** Using existing Microsoft licenses, such as Windows Server and SQL Server, when running on AWS.

- **License mobility:** Allows customers to bring eligible Microsoft licenses to the AWS Cloud under Software Assurance.
- **Software Assurance:** Microsoft's program that provides license mobility and other benefits when running Microsoft products on AWS.

Design principles

The Well-Architected Framework identifies a set of general design principles to facilitate good design in the cloud. In addition, the following design principles should also be considered for designing and operating Microsoft workloads:

- **Assess your Microsoft environment holistically:** Before designing or migrating, assess your current Microsoft workloads with tools like AWS Optimization and Licensing Assessment (OLA), Migration Evaluator, and Application Discovery Service. This foundational step helps define the business case, understand resource utilization, licensing, and technical dependencies.
- **Define clear business and technical goals:** Clarify the expected outcomes of moving or running Microsoft workloads on AWS. Whether you're aiming to reduce licensing costs, improve availability, increase agility, or enhance security, these goals should shape your architecture choices, migration strategy, and operational model.
- **Use AWS services and managed offerings:** Reduce operational burden and complexity by using managed services tailored for Microsoft workloads—such as Amazon RDS for SQL Server, Amazon FSx for Windows File Server, AWS Managed Microsoft AD, and EC2 Image Builder. This improves performance, reliability, and maintainability.
- **Optimize licensing and cost models:** Right-size instances and storage, automate start and stop schedules, and consider BYOL strategies when applicable. Evaluate SQL Server edition needs and instance consolidation opportunities. Use services like AWS Compute Optimizer and Trusted Advisor to continuously refine your resource usage.
- **Prioritize security aligned with workload criticality:** Apply Microsoft and AWS-aligned security controls. Enforce the principle of least privilege, encrypt data at rest, in transit, and in use (for example, using Always Encrypted), and align with your organizational identity strategy. Use centralized logging and SIEM integrations to detect and respond to anomalies.
- **Design for availability and resilience:** Define business-driven availability targets and choose architecture patterns that meet those needs—such as multi-AZ deployments, SQL Server Always On, or Amazon FSx Multi-AZ. Regularly test for failover and disaster recovery capabilities.
- **Enable operational excellence through automation and observability:** Automate infrastructure and patch management with AWS Systems Manager, and use infrastructure as code with AWS CloudFormation or Terraform. Implement robust observability using Amazon CloudWatch, Performance Counters, Application Insights, and AWS X-Ray for deep visibility into system health.

- **Embrace modernization incrementally:** Where feasible, modernize traditional workloads using containers (with ECS, EKS) or serverless solutions (like .NET on AWS Lambda). This unlocks agility, scalability, and cost optimization, while maintaining alignment with existing Microsoft technologies.
- **Govern through consistency and compliance:** Use AWS Control Tower, AWS Organizations, and tagging strategies to enforce adherence, manage accounts, and apply consistent policies across environments. Integrate security baselines and operational controls from day one.
- **Plan for lifecycle, sustainability, and continuous improvement:** Continuously monitor workload performance and cost. Reassess choices as AWS services evolve, license agreements change, or application usage shifts. Define update strategies for OS, SQL Server, and application components for sustainability.

Scenarios

This section explores six critical scenarios that businesses commonly encounter: Microsoft SQL Server migration, .NET application modernization, Active Directory integration, Windows Server and file services migration, Windows-based infrastructure modernization, and implementing high availability and disaster recovery for Windows workloads on AWS. Each scenario presents unique opportunities for improving scalability, reducing costs, enhancing security, and streamlining operations by leveraging AWS services and best practices. Whether you're looking to lift-and-shift existing workloads or completely transform your architecture, these scenarios provide valuable insights and practical solutions to help you navigate your cloud journey successfully.

Scenarios

Scenario 1: Microsoft SQL Server Migration to AWS

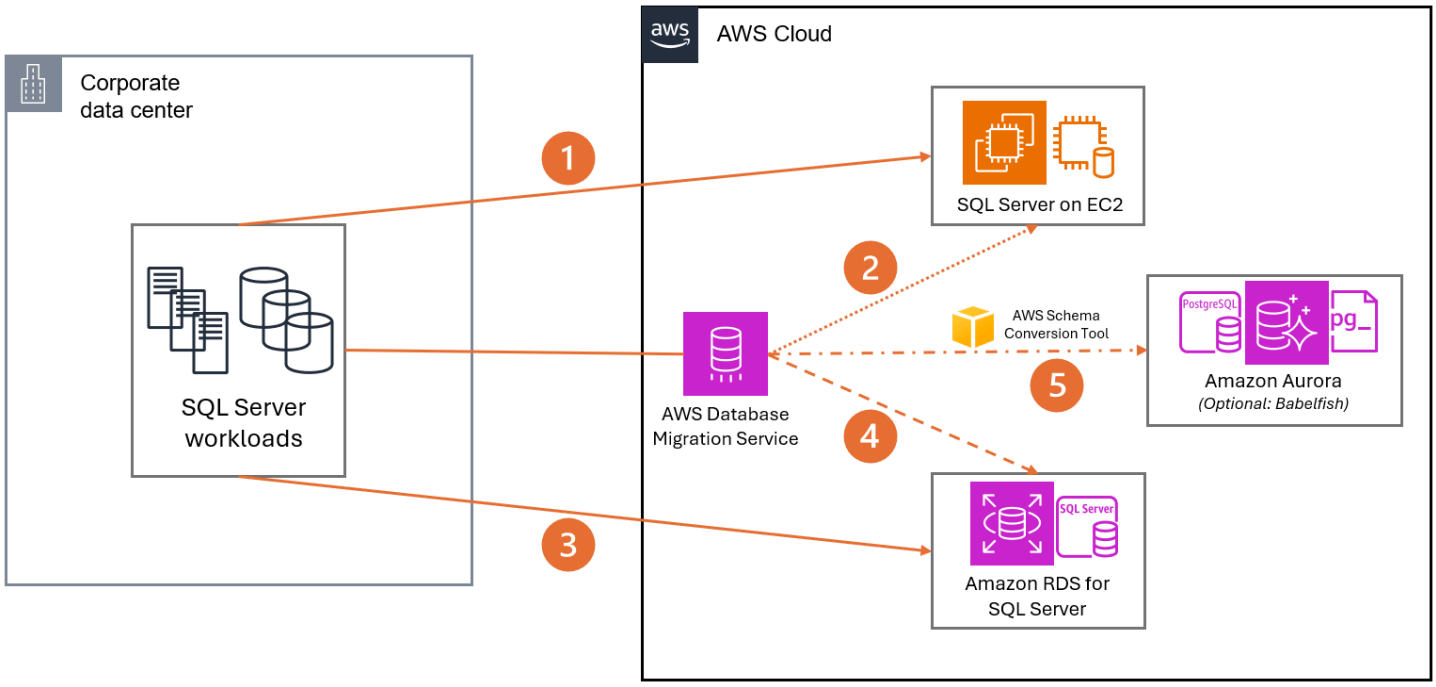
Organizations running business-critical SQL Server workloads often face challenges with hardware lifecycle management, licensing costs, and operational overhead. Migrating SQL Server to AWS offers opportunities for improved scalability, reduced maintenance burden, and potential cost savings through various deployment options. This scenario covers approaches from lift-and-shift to complete modernization, helping organizations choose the right path based on their specific requirements.

Characteristics

- Organizations are dealing with SQL Server workloads that are essential for business operations.
- Current challenges include managing hardware lifecycles, SQL Server licensing expenses, and operational maintenance.
- The scenario spans different migration approaches, from basic lift-and-shift to full modernization.

- The focus is on helping organizations select migration strategies based on their unique needs.

Reference architecture



1. **Self-Managed SQL Server on EC2:** This approach transfers SQL Server workloads from on-premises servers to self-managed EC2 instances using native SQL Server tools or full server replication methods. It's ideal for organizations needing detailed control or preserving complex configurations. Migration options include database-level methods (for example, backup and restore, log shipping) or server-level replication using tools like AWS Application Migration Service.
2. **Self-Managed SQL Server on EC2 (AWS DMS):** This approach migrates SQL Server workloads from on-premises servers to self-managed EC2 instances using AWS Database Migration Service. DMS facilitates data migration and synchronization, offering a streamlined process for transferring database contents while minimizing downtime. It's suitable for organizations seeking a managed migration tool with replication capabilities.
3. **Amazon RDS for SQL Server (Windows Methods):** This strategy replatforms SQL Server workloads from on-premises servers to Amazon RDS using native SQL Server migration tools. Methods like backup/restore or log shipping are employed to transfer databases to the managed RDS environment, balancing familiar SQL Server native capabilities with the benefits of a managed service.

4. **Amazon RDS for SQL Server (AWS DMS):** This approach replatforms SQL Server workloads to Amazon RDS using AWS Database Migration Service. DMS manages data migration and synchronization from on-premises servers to RDS, combining the advantages of a managed migration process with the operational benefits of RDS for ongoing database management.
5. **Modernize to Amazon Aurora:** This path transforms SQL Server workloads to Amazon Aurora, leveraging AWS Schema Conversion Tool for schema conversion to PostgreSQL and AWS DMS for data migration. For applicable use cases, Babelfish for Aurora PostgreSQL can be employed to minimize application code changes, offering a balance between modernization and compatibility.

Configuration notes

- Migrating to self-managed SQL Server on Amazon EC2 is often an effective strategy for projects with tight deadlines. This approach allows for a quick lift-and-shift migration while maintaining full control over the database environment. Depending on their existing licensing agreements, organizations can potentially reduce costs by using a bring your own license (BYOL) option, subject to Microsoft's licensing terms and conditions.
- Replatforming to Amazon RDS for SQL Server offers the benefits of a managed service without requiring changes to existing schema or application code. This option reduces operational overhead while maintaining SQL Server compatibility. For specific use cases requiring more control, Amazon RDS Custom for SQL Server allows a hybrid approach between fully-managed and self-managed services. Additionally, the Bring Your Own Media (BYOM) feature in RDS Custom can provide potential license cost savings for eligible customers.
- Modernizing to Amazon Aurora (or another purpose-built database service) typically yields the greatest long-term benefits. This approach frees organizations from SQL Server licensing constraints and allows them to fully use cloud-based, managed database services. While it may require more upfront effort for schema conversion and potential application changes, it often results in improved performance, scalability, and cost-effectiveness over time. The choice of target database (for example, Aurora PostgreSQL with Babelfish for SQL Server compatibility, or a completely different database type) should align with the organization's specific requirements and future technology strategy.

Scenario 2: .NET Application Modernization on AWS

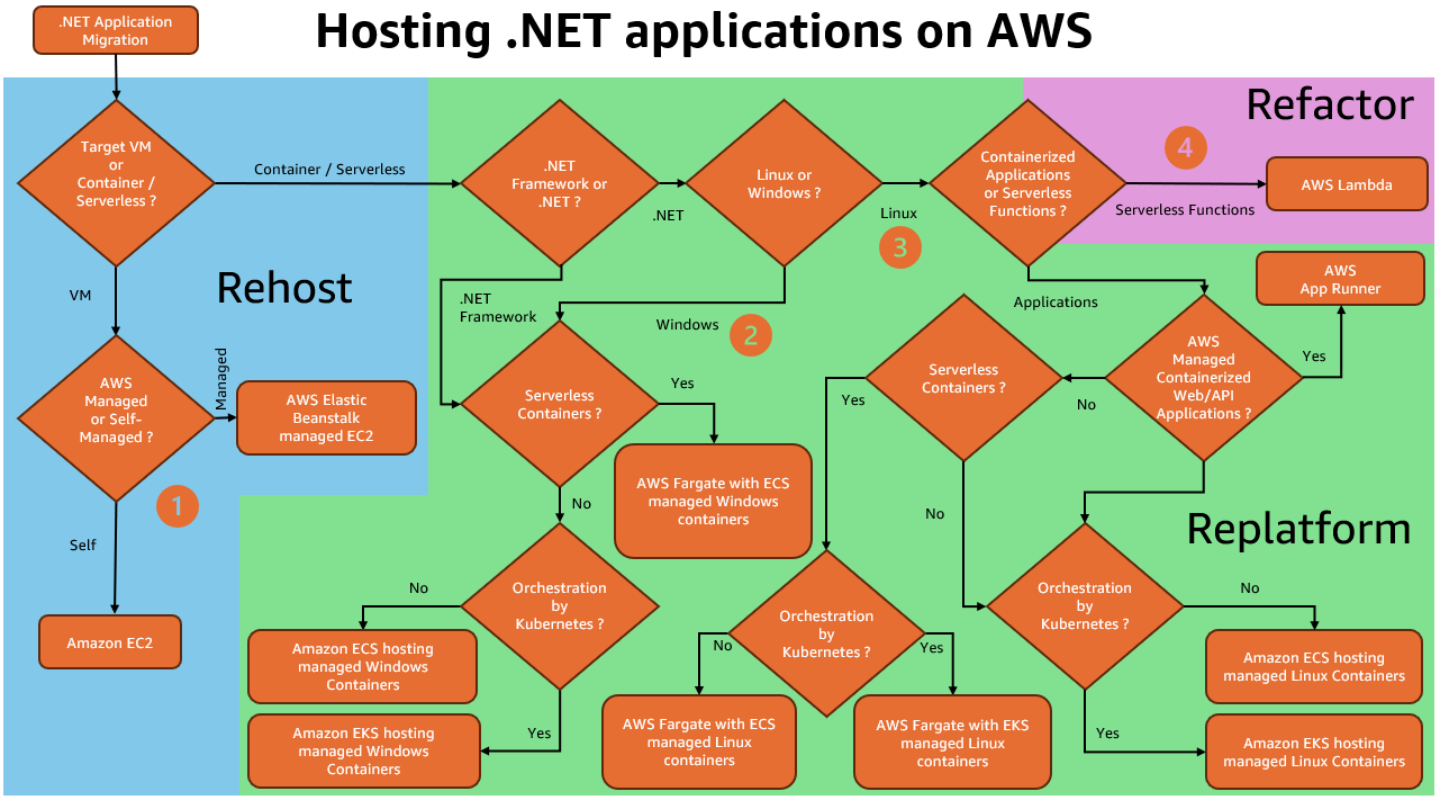
As organizations seek to modernize their legacy .NET applications, AWS provides a comprehensive platform for transformation. This scenario addresses the journey from traditional monolithic .NET

applications to cloud-based architectures, enabling organizations to use modern development practices, improve scalability, and reduce operational costs while maintaining familiarity with Microsoft technologies.

Characteristics

- **Development evolution:** Evolution from traditional .NET Framework monoliths to cloud-based .NET Core/6+ microservices, maintaining familiar Microsoft tooling while enabling modern development practices and improved team productivity.
- **Architectural transformation:** Shift from tightly-coupled monolithic architectures to loosely-coupled microservices, adopting API-first approaches and enabling independent service scaling and maintenance.
- **Infrastructure modernization:** Migration from on-premises Windows servers to container-based deployments on AWS, using infrastructure as code and automated deployment pipelines for improved operational efficiency.
- **Technology stack update:** Comprehensive modernization of the technology stack, including runtime (.NET Framework to .NET Core/6+), databases, authentication, storage, and monitoring solutions to align with cloud-based capabilities.
- **Business benefits:** Achievement of tangible business outcomes including reduced operational costs, faster deployment cycles, improved scalability, enhanced innovation capabilities, and simplified maintenance processes.

Reference architecture



This .NET hosting decision guide helps customers choose the best AWS service for their applications based on their specific requirements, whether they need full control, simplified management, or zero infrastructure overhead. This flowchart considers factors such as development team expertise, operational preferences, and application characteristics to recommend the most suitable hosting option on AWS:

1. Rehost .NET Framework or cross-platform .NET to AWS (Windows or Linux-based)
   - **Managed service:** AWS Elastic Beanstalk
   - **Self-managed:** Amazon Elastic Compute Cloud (EC2)
2. Replatform .NET Framework or cross-platform .NET to AWS (Windows-based)
   - **Serverless containers:** AWS Fargate with Amazon Elastic Container Service (ECS)
   - **AWS containers orchestration:** Amazon Elastic Container Service (ECS)
   - **Kubernetes orchestration:** Amazon Elastic Kubernetes Service (EKS)
3. Replatform cross-platform .NET to AWS (Linux-based)
   - **AWS serverless containers:** AWS Fargate with Amazon Elastic Container Service (ECS)
   - **Kubernetes serverless containers:** AWS Fargate with Amazon Elastic Kubernetes Service (EKS)
   - **AWS containers orchestration:** Amazon Elastic Container Service (ECS)
   - **Kubernetes orchestration:** Amazon Elastic Kubernetes Service (EKS)

- **AWS fully-managed containers:** AWS App Runner

4. Refactor cross-platform .NET to AWS (Linux-based)
   - **Serverless functions:** AWS Lambda

Configuration notes

- **EC2 Windows Server configuration:** Best suited for legacy applications requiring full control or specific Windows Server features. Implement across multiple Availability Zones using Auto Scaling groups, configure Systems Manager for automated patching and management, and use Amazon CloudWatch Application Insights for .NET-specific monitoring. Common challenges include licensing optimization and Windows authentication - address through proper instance sizing and Active Directory integration. Security involves network ACLs, security groups, and regular vulnerability assessments.
- **Elastic Beanstalk implementation strategy:** A managed service ideal for traditional .NET applications requiring minimal infrastructure management. Best suited for organizations starting their cloud journey. Implement with blue-green deployments for zero-downtime updates, configure enhanced health monitoring, and use environment variables for configuration management. Common challenges include deployment timeouts and Windows updates, which can be addressed through appropriate capacity planning and maintenance windows. Security is managed through security groups and IAM roles.
- **Container services approach (Amazon ECS or Amazon EKS):** Recommended for microservices architectures and modern .NET applications. Amazon ECS offers simpler management while Amazon EKS provides broader orchestration capabilities. Implement service mesh for inter-service communication, use AWS Secrets Manager for sensitive data, and configure auto scaling based on application metrics. Key considerations include container image security scanning, network segmentation, and proper task sizing. Troubleshoot using CloudWatch Container Insights and AWS X-Ray.
- **Serverless architecture pattern:** Optimal for event-driven workloads and APIs. Lambda with .NET requires careful attention to cold starts and memory allocation. Implement API Gateway with custom authorizers, use Step Functions for complex workflows, and configure dead-letter queues for error handling. Security focuses on IAM roles and API authentication. Monitor execution times and memory usage to optimize cost and performance.

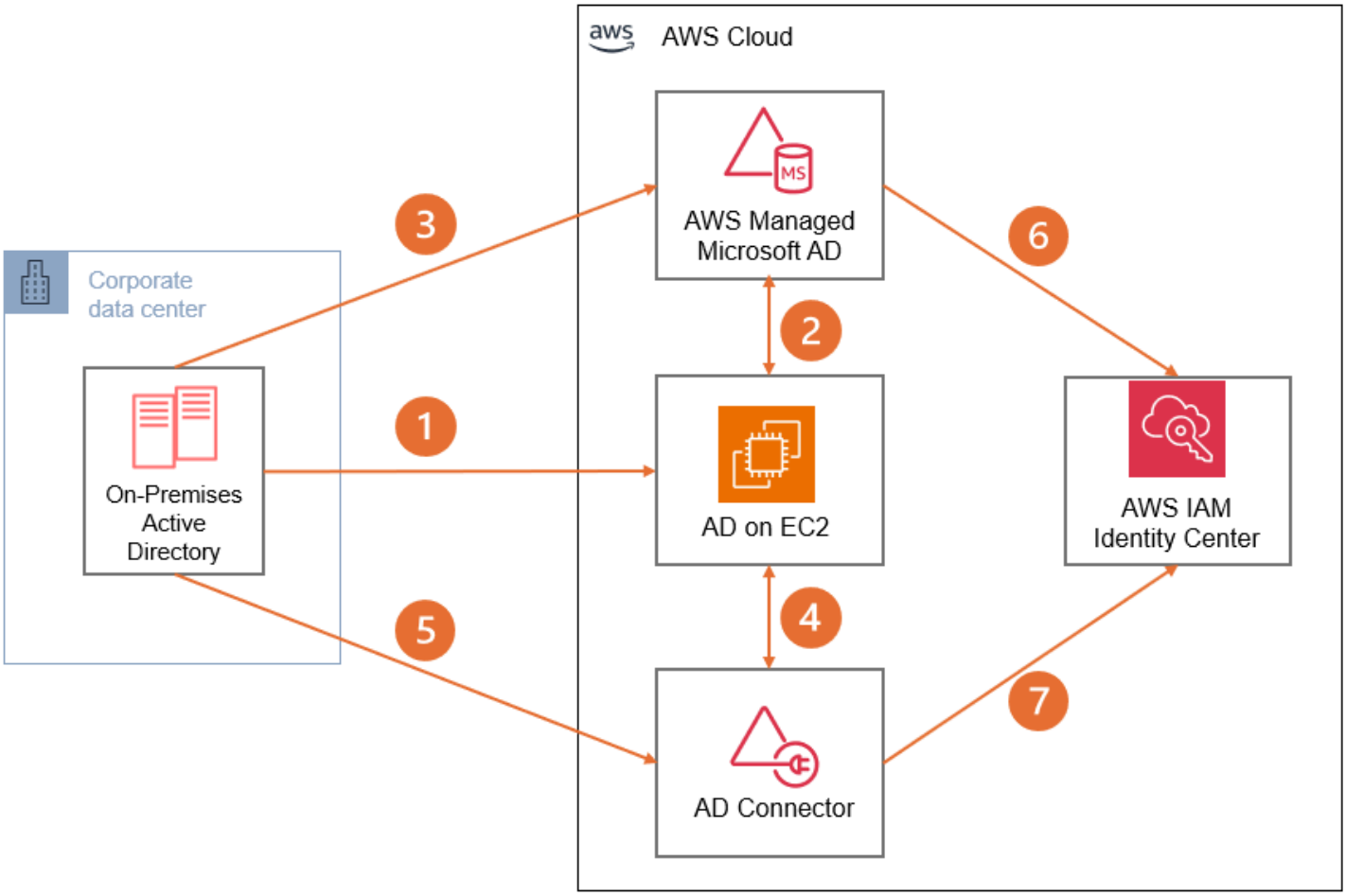
Scenario 3: Active Directory integration with AWS

For organizations heavily invested in Microsoft Active Directory, extending their identity and access management to AWS is crucial for maintaining security and operational consistency. This scenario outlines how to create a holistic identity management experience across on-premises and AWS environments, enabling single sign-on and unified access controls.

Characteristics

- **Identity integration:** Extension of on-premises Active Directory to AWS through Directory Service options (AWS Managed Microsoft AD or AD Connector), enabling consistent identity management across hybrid environments.
- **Authentication framework:** Implementation of unified authentication mechanisms allowing users to access both on-premises and AWS resources with existing AD credentials, supporting single sign-on (SSO) capabilities through AWS IAM Identity Center.
- **Access control strategy:** Centralized management of security policies and access permissions, using existing AD groups and security policies while integrating with AWS IAM roles and policies for cloud resource access.

Reference architecture



1. Self-managed Active Directory on EC2 is a common approach for organizations needing full control over their AD infrastructure. This brings authentication and authorization closer to AWS workloads, addressing network and security requirements effectively.
2. Extending self-managed AD on EC2 to Directory Services can be achieved by creating a trust with AWS Managed Microsoft Active Directory. This enables AWS services reliant on Managed AD to use the existing self-managed AD identity source.
3. Alternatively, a trust between on-premises AD and AWS Managed AD can be established using VPN or Direct Connect, meeting specific networking and security needs. This also opens the possibility of migrating the entire forest to AWS Managed AD, potentially decommissioning self-managed AD.
4. AD Connector serves as a simple gateway solution for extending self-managed AD running on EC2 to AWS services, offering an alternative integration approach.
5. For on-premises scenarios, AD Connector can also be utilized over a secure VPN or Direct Connect connection, providing flexibility in network design.

6. AWS IAM Identity Center can integrate with AWS Managed AD as an identity source, with the capability to expand to multiple self-managed AD forests or domains if required.
7. Alternatively, AD Connector can be used to connect self-managed AD as the identity source for AWS IAM Identity Center, though this is limited to one forest or domain.

Configuration notes

- **Self-managed AD infrastructure:** When implementing AD on EC2, deploy Domain Controllers across multiple Availability Zones for high availability. Configure security groups to allow only necessary AD ports, implement appropriate site-to-site replication, and use AWS Backup for automated system state backups. Common challenges include DNS configuration and replication latency, which are addressed through proper subnet design and monitoring. Set up CloudWatch alerts for DC health and performance metrics.
- **AWS managed Microsoft AD integration:** Establish two-way trust relationships with appropriate security filtering. Configure DNS conditional forwarders between environments, implement selective authentication if needed, and use AWS Managed Microsoft AD delegated administration. Key considerations include password policies synchronization and Group Policy Object strategy. Monitor trust health through Directory Service console and CloudWatch metrics.
- **Network connectivity configuration:** Design redundant connectivity using both AWS Direct Connect and VPN as backup. Implement appropriate route tables and network ACLs, maintain proper DNS resolution between networks, and configure bandwidth allocation for AD traffic. Common issues include asymmetric routing and DNS resolution failures, which can be resolved through proper route configuration and DNS forwarding setup.
- **Identity Center and SSO setup:** Configure AWS IAM Identity Center with appropriate directory integration (either Managed AD or AD Connector). Implement attribute-based access control (ABAC) using AD group membership, set up permission sets aligned with security policies, and establish user portal access. Monitor authentication failures and implement appropriate remediation processes for access issues.

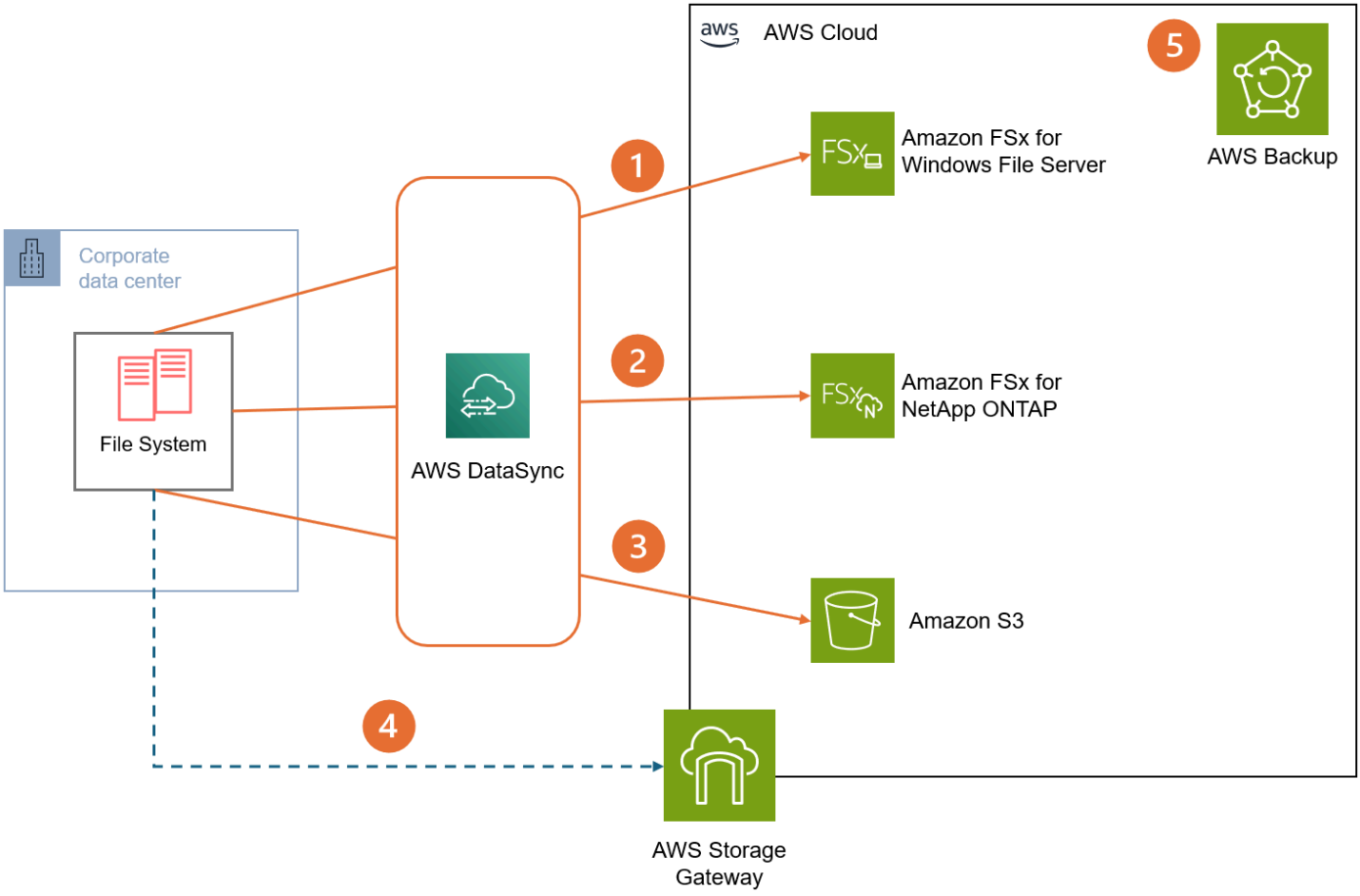
Scenario 4: Windows server and file services migration

Many organizations struggle with growing file storage needs, backup complications, and the management overhead of traditional Windows file servers. This scenario demonstrates how to modernize Windows-based file services using AWS services, providing better scalability, availability, and reduced operational complexity while maintaining compatibility with existing Windows-based workflows.

Characteristics

- **Storage architecture evolution:** Transformation from traditional Windows file servers to cloud-based solutions like Amazon FSx for Windows File Server or Amazon FSx for NetApp ONTAP, providing Windows compatibility while using AWS's scalability and availability.
- **Backup modernization:** Replacement of traditional backup systems with AWS backup solutions, enabling automated, consistent backup policies, simplified recovery processes, and integration with existing backup retention requirements.
- **Access control integration:** Preservation of existing NTFS permissions and Active Directory integration while implementing AWS security features, improving the user experience and maintaining security requirements.
- **Operational efficiency:** Reduction in management overhead through AWS managed services, automated maintenance, patching, and monitoring, while maintaining familiar Windows administrative tools and processes.

Reference architecture



1. **Amazon FSx for Windows File Server:** This managed service offers a Windows file system experience in AWS. Configure it to join your Active Directory for authentication and access control. Implement appropriate file share permissions and NTFS ACLs. Use AWS DataSync for efficient data migration from on-premises systems.
2. **Amazon FSx for NetApp ONTAP:** Use this versatile managed file system for both SMB and NFS access, with added iSCSI support for Windows environments. Configure AD integration for authentication and implement appropriate export policies and NTFS permissions.
3. **Amazon S3 for cold storage:** While usually not suitable for primary Windows file systems, Amazon S3 is excellent for cold storage in Windows environments. Implement S3 Lifecycle policies to automatically transition data to lower-cost storage classes.
4. **AWS Storage Gateway (Volume Gateway):** Use Volume Gateway for hybrid cloud storage with local caching. Configure in cache mode for frequently accessed data or stored mode for full local copies. Implement appropriate iSCSI initiator configurations on Windows servers and use Chap authentication for enhanced security.
5. **Backup and recovery strategy:** Use the default daily backups for FSx services and extend with AWS Backup for a comprehensive strategy. Configure backup plans with appropriate retention policies and implement cross-region backup copies for disaster recovery.

Configuration notes

- **FSx implementation strategy:** Configure FSx for Windows File Server with Multi-AZ deployment for high availability. Size storage and throughput based on performance analysis of existing file servers. Implement DFS namespaces for transparent client access and potential migration phases. Common challenges include DNS resolution and permission migration, addressed through proper DNS configuration and data replication. Monitor performance through CloudWatch metrics, focusing on storage capacity, IOPS, and throughput. Security implementation should include encryption at rest using KMS keys, in-transit encryption, and regular security audits of share permissions.
- **Migration and data transfer framework:** Establish a structured migration approach using AWS DataSync with appropriate bandwidth throttling. Create detailed inventory of shares, permissions, and DFS configurations before migration. Implement staged migration starting with smaller, non-critical shares. Common issues include network throttling and permission mapping, addressed through appropriate scheduling and thorough testing. Monitor migration progress using DataSync metrics and implement verification procedures.
- **Backup and recovery architecture:** Design comprehensive backup strategy using AWS Backup with appropriate retention policies. Implement daily backups with custom schedules for critical

shares. Configure cross-region backup copies for disaster recovery scenarios. Common challenges include backup window management and restoration testing, addressed through appropriate scheduling and regular DR exercises. Monitor backup success rates and storage costs through AWS Backup reports. Implement automated cleanup of old backups based on retention policies.

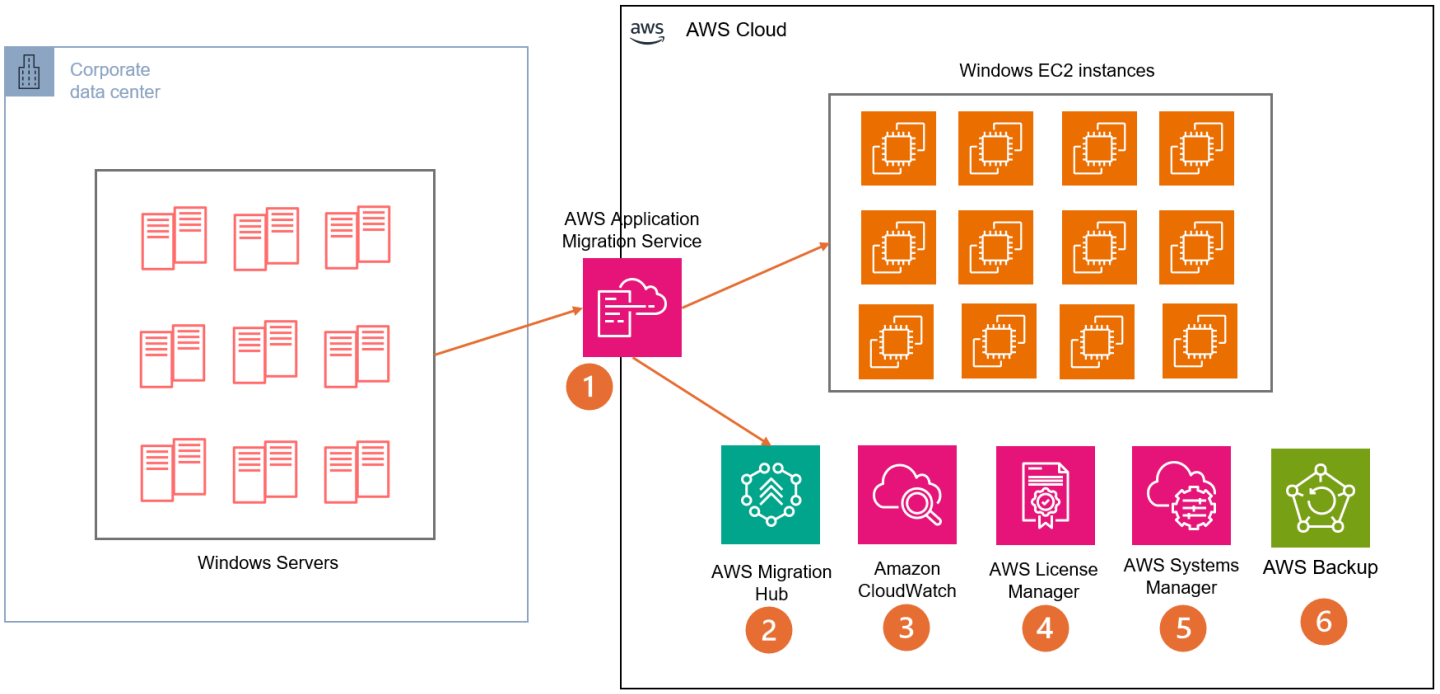
Scenario 5: Windows-based infrastructure modernization

As Windows Server workloads approach end-of-support or hardware refresh cycles, organizations need a strategic approach to infrastructure modernization. This scenario provides a framework for assessing, migrating, and optimizing Windows workloads on AWS, enabling automated management, improved security, and operational efficiency while maximizing existing Microsoft investments.

Characteristics

- **Lifecycle management:** Strategic approach to Windows Server modernization focusing on end-of-support deadlines and hardware refresh cycles, incorporating assessment tools and migration planning to optimize timing and resource allocation.
- **Migration strategy:** Systematic workload evaluation and migration approach using tools like AWS Application Migration Service (MGN) and AWS Migration Hub, enabling both migration and modernization paths based on workload requirements.
- **Infrastructure optimization:** Implementation of AWS automation for Windows Server management, including patching (AWS Systems Manager), monitoring (Amazon CloudWatch), and infrastructure provisioning (AWS CloudFormation), reducing operational overhead.
- **Security enhancement:** Integration of AWS security services with Windows Server security features, implementing improved access controls, network security, and compliance monitoring while maintaining existing security policies.
- **Cost management:** Optimization of Windows Server licensing and infrastructure costs through right-sizing, automation, and AWS pricing models (Reserve Instances, Savings Plans), while using existing Microsoft licensing benefits like BYOL.

Reference architecture



1. AWS Application Migration Service (MGN) facilitates block-level migration to Amazon EC2. It allows synchronization and transition of servers to EC2, regardless of their original location (on-premises or elsewhere).
2. AWS Migration Hub serves as a centralized platform for efficiently planning and managing your migration process, offering a comprehensive overview of your migration projects.
3. Amazon CloudWatch provides robust monitoring capabilities for Windows-based servers and workloads, enabling real-time tracking and analysis of performance metrics.
4. AWS License Manager streamlines the management of third-party licenses, helping organizations maintain adherence and optimize license utilization across their AWS environment.
5. AWS Systems Manager automates various aspects of operating system management, including but not limited to inventory tracking, patching, and other essential maintenance tasks, enhancing operational efficiency.
6. AWS Backup provides centralized protection for Windows-based workloads by automating backup processes, enabling consistent data protection, and offering secure storage of backups across multiple AWS services and resources.

Configuration notes

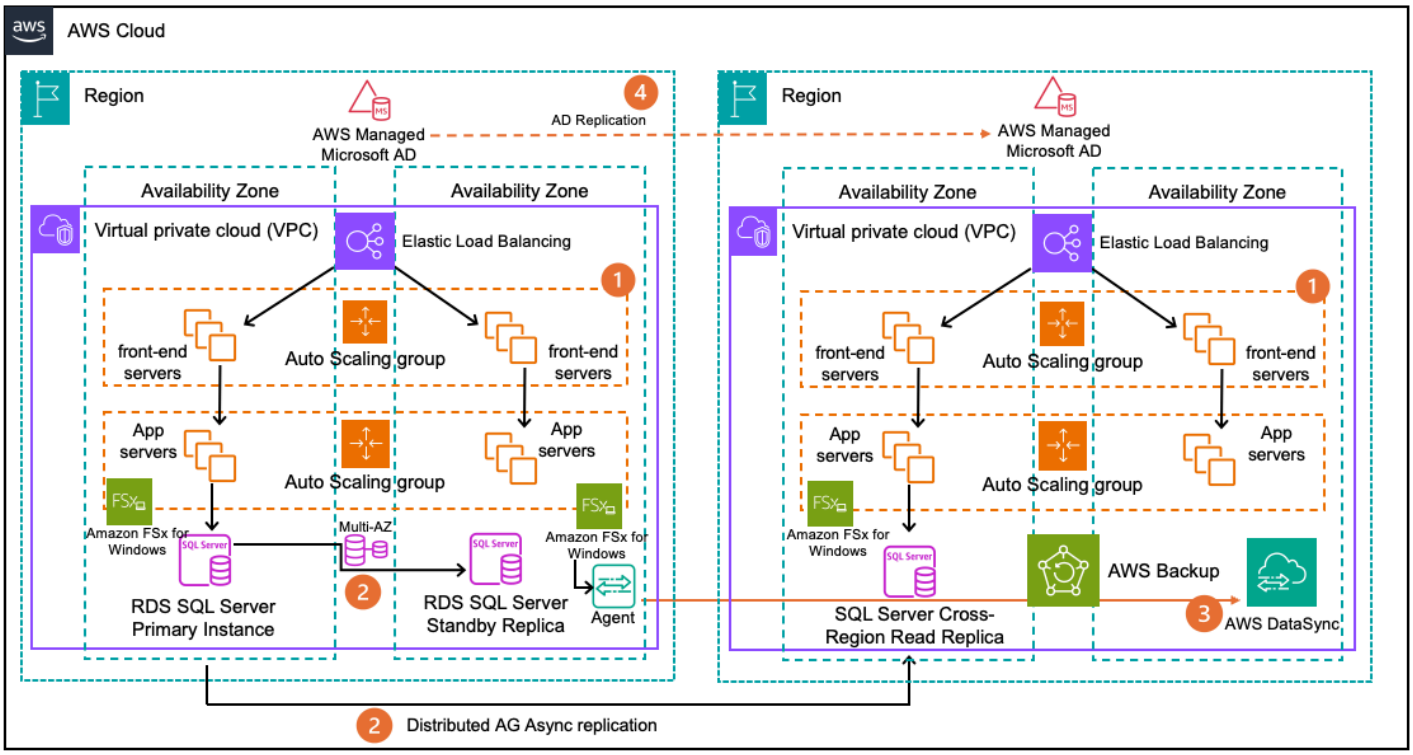
- **Initial assessment and planning:** Establish migration readiness using AWS Migration Evaluator, Migration Hub, and Application Discovery Service. Implement security best practices and define migration waves based on deadlines, dependencies, and compliance. Troubleshoot by monitoring agent health and reviewing logs.
- **Migration setup and execution:** Use AWS Application Migration Service for workload migration. Configure replication settings and prepare the target environment. Test with non-production workloads first. Monitor replication lag and network throughput, and verify connectivity for troubleshooting.
- **Monitoring and management configuration:** Set up CloudWatch for performance metrics, Systems Manager for patch management, and License Manager for tracking. Implement security controls, use parameter store for configurations, and enable audit logging. Configure backup and recovery procedures for comprehensive system management.
- **Cost optimization and compliance:** Optimize costs and maintain adherence using License Manager, Cost Explorer, and budget alerts. Implement auto scaling and scheduled resource management. Regularly review instance sizing and conduct compliance assessments. Establish procedures for license consolidation and non-compliance responses.
- Configure AWS Backup to provide centralized data protection for Windows workloads across AWS services. Set up backup policies with appropriate retention periods and scheduling. Implement automated backup monitoring through CloudWatch and establish recovery time objectives (RTOs). Regular backup testing and validation maintains business continuity and disaster recovery readiness.

Scenario 6: High availability and disaster recovery for Windows workloads on AWS

Characteristics

- Need to maintain critical Windows-based services highly available with minimal downtime.
- Requirements for rapid recovery in the event of disasters (like natural disasters or infrastructure failures).
- Regulatory needs around Recovery Point Objective (RPO) and Recovery Time Objective (RTO).

Reference architecture



1. Multi-AZ deployment of Windows Servers using Amazon EC2:
   - Auto Scaling groups for automatic scaling and replacement.
   - Use of Amazon FSx for Windows File Server or Amazon EFS for shared storage.
   - Implement Windows Server Failover Clustering for application-level high availability (HA).
2. Database high availability with Amazon RDS for SQL Server:
   - Multi-AZ deployment for automatic failover.
   - Use RDS SQL Server cross-Region read replica hosted in a different AWS Region.
   - Use of Amazon S3 and AWS Backup for offsite backups.
3. Cross-Region data replication using AWS DataSync:
   - Replicate file shares, databases, and other critical data to secondary AWS Region.
   - Use RPO-based scheduling and throttling for optimal performance.
4. Multi-Region Active Directory replication:
   - Enables AD Multi-Region replicas to keep AuthN and AuthZ during failovers.
5. Automated failover and recovery using AWS Systems Manager and AWS Lambda:
   - Define recovery plans and runbooks for easier failover.
   - Implement CloudWatch alarms and automated remediation workflows.
   - Regularly test failover procedures and validate RTO and RPO.

Configuration notes

- **EC2 instance design:** Size instances appropriately based on workload requirements. Use Instance Metadata Service and Systems Manager for automatic configuration. Implement security best practices like JIT access, Network ACLs, and Security Groups.
- **FSx and EFS configuration:** Enable Multi-AZ deployment and configure replication settings. Implement NTFS permissions and access control integration with AD. Monitor capacity, throughput, and latency using Amazon CloudWatch.
- **SQL Server HA on RDS:** Configure backup retention, point-in-time recovery, and automated backups. Implement cross-Region read replicas for geographic redundancy. Use AWS DMS for efficient data migration and replication.
- **Disaster recovery orchestration:** Define recovery time and point objectives (RTO and RPO) based on business requirements. Automate failover processes using AWS Systems Manager Runbooks and AWS Lambda. Implement alerting, monitoring, and regular DR testing procedures.

Operational excellence

The operational excellence pillar focuses on running and monitoring Microsoft workloads to deliver business value and continuously improve supporting processes and procedures. This includes implementing comprehensive monitoring and observability solutions that leverage Microsoft Performance Counters, Windows Event Logs, and Application Performance Monitoring (APM) for .NET applications. The pillar emphasizes operational automation through infrastructure as code (IaC), management and governance solutions, and the strategic use of AWS managed services to reduce operational overhead. By establishing robust monitoring, logging, and automation practices, organizations can verify that their Microsoft workloads operate efficiently, maintain high availability, and enable rapid response to operational issues.

Focus areas

- Definitions
- Design principles
- Monitoring and observability
- Operational automation

Definitions

- **Performance counters:** Windows-native monitoring mechanism that provides real-time metrics about system performance, application behavior, and resource utilization across Microsoft products.
- **Application performance monitoring (APM):** Tools and practices for monitoring and managing the performance and availability of software applications, particularly important for .NET applications and SQL Server environments.

Design principles

- **Implement comprehensive observability:** Establish monitoring and logging across all layers of your Microsoft workload, including infrastructure, applications, and databases, using both Microsoft-native tools and AWS services for complete visibility.

- **Automate operational tasks:** Leverage infrastructure as code (IaC), AWS Systems Manager, and other automation tools to reduce manual intervention, minimize human error, and perform consistent deployment and management of Microsoft workloads.
- **Embrace managed services:** Utilize AWS managed services like AWS Managed Microsoft AD, Amazon RDS for SQL Server, and Amazon FSx to reduce operational complexity and benefit from AWS's operational expertise.
- **Establish standardized processes:** Create and maintain standardized AMIs, deployment templates, and operational procedures to ensure consistency across environments and reduce operational overhead.
- **Enable rapid response:** Implement automated alerting, centralized logging, and well-defined incident response procedures to quickly identify and resolve operational issues affecting Microsoft workloads.

Monitoring and observability

Implementing comprehensive monitoring and observability for Microsoft workloads is essential for maintaining operational excellence. This includes using Microsoft Performance Counters for infrastructure monitoring, collecting Windows Event Logs and application logs, and implementing Application Performance Monitoring (APM) for .NET applications. By establishing proper observability across all layers of the Microsoft stack, organizations can proactively identify issues, optimize performance, and ensure reliable operation of their workloads on AWS.

MSFTOPS01: How do you implement monitoring and observability for your Microsoft workload?

Monitor your Microsoft workload so that you have visibility over its state. You should consider mechanisms across all components to provide real-time monitoring and logging to allow predictability and remediation.

Best practices

- MSFTOPS01-BP01 Implement infrastructure monitoring for your Microsoft workload
- MSFTOPS01-BP02 Implement and collect logging for your Microsoft workload
- MSFTOPS01-BP03 Implement Application Performance Monitoring (APM) for your Microsoft workload

MSFTOPS01-BP01 Implement infrastructure monitoring for your Microsoft workload

The implementation of infrastructure monitoring for Microsoft workloads on AWS will provide comprehensive visibility into system performance, resource utilization, and application health. This monitoring solution will detect anomalies in real time, generate actionable alerts, and enable rapid troubleshooting of issues before they impact end users. Consider leveraging Microsoft Performance Counters to cover the basic infrastructure monitoring for your Microsoft workload servers. Besides operating system and performance metrics, the counters will be expanded according to the Microsoft product deployed, such as SQL Server, Internet Information Services (IIS), Active Directory Federation Services, and others. The Performance Counters can also be integrated with monitoring solutions, like Amazon CloudWatch, Amazon Managed Service for Prometheus, and Amazon Managed Grafana.

Desired outcome: Establish comprehensive infrastructure monitoring that provides real-time visibility into the health and performance of your Microsoft workload components, enabling proactive issue identification and resolution while leveraging both Microsoft-native monitoring capabilities and AWS monitoring services for optimal observability.

Common anti-patterns:

- Relying solely on basic system monitoring without leveraging Microsoft Performance Counters, missing critical application-specific metrics that could indicate performance issues or potential failures before they impact users.
- Implementing monitoring in silos without integrating Microsoft Performance Counters with centralized monitoring solutions, leading to fragmented visibility and delayed incident response across the Microsoft workload infrastructure.
- Monitoring only during business hours or reactive monitoring after issues occur, rather than establishing continuous, proactive monitoring that can predict and prevent problems before they affect workload availability.

Benefits of establishing this best practice:

- Enhanced visibility and proactive issue detection through comprehensive monitoring of both operating system metrics and Microsoft product-specific performance counters, enabling early identification of potential problems before they impact business operations.

- Improved operational efficiency by integrating Microsoft Performance Counters with AWS monitoring services like Amazon CloudWatch, providing centralized dashboards, automated alerting, and streamlined incident response processes.
- Better capacity planning and performance optimization through detailed metrics collection across all Microsoft workload components, enabling data-driven decisions for resource allocation and performance tuning.

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

Implementation guidance

Implementing comprehensive infrastructure monitoring for Microsoft workloads requires a strategic approach that combines Microsoft-native monitoring capabilities with AWS services. Start by identifying the Microsoft products in your environment and their specific Performance Counters, then configure collection and integration with AWS monitoring services. This approach ensures you capture both standard system metrics and application-specific indicators that are crucial for maintaining optimal performance and availability of your Microsoft workloads.

Implementation steps

1. Inventory your Microsoft workload components and identify relevant Performance Counters for each product (Windows Server, SQL Server, IIS, and Active Directory).
2. Install and configure the Amazon CloudWatch Agent on Windows instances to collect Performance Counters and system metrics.
3. Configure custom Performance Counter collection for Microsoft-specific applications and services running in your environment.
4. Set up Amazon CloudWatch dashboards to visualize key performance metrics and create a centralized monitoring view.
5. Establish Amazon CloudWatch alarms and notifications for critical performance thresholds and anomaly detection.
6. Integrate with Amazon Managed Service for Prometheus and Amazon Managed Grafana for advanced monitoring and visualization capabilities.
7. Implement automated response mechanisms using AWS Systems Manager Automation for common performance issues.
8. Establish regular review processes to evaluate monitoring effectiveness and adjust thresholds based on workload behavior.

Resources

Related documents:

- Recommended metrics
- Monitoring Windows pods with Prometheus and Grafana
- Use AWS Systems Manager to enable CloudWatch memory metrics for Windows Server Amazon EC2 instances

Related tools:

- Collect metrics, logs, and traces using the CloudWatch agent
- Using Amazon CloudWatch Dashboard
- Amazon Managed Grafana - Grafana dashboards
- Prometheus Windows Exporter

MSFTOPS01-BP02 Implement and collect logging for your Microsoft workload

Set up logs for your Microsoft workload infrastructure and applications. Windows Event Logs are natively generated by the Windows operating system and usually by the applications deployed. Products such as SQL Server and Internet Information Services (IIS) also provide text logs that can bring insights to observability. Both Windows Event Logs and custom logs can be collected by Amazon CloudWatch Agent and have them centralized in the Amazon CloudWatch console. For enhanced security monitoring and analysis, these logs can be forwarded to Security Information and Event Management (SIEM) solutions through built-in connectors or AWS service integrations, enabling real-time security event monitoring, automated threat detection, compliance reporting, and advanced security analytics.

Desired outcome: Establish comprehensive logging collection and centralization for your Microsoft workload, providing complete visibility into system events, application behavior, and security activities while enabling efficient log analysis, troubleshooting, and compliance reporting through integrated AWS services and SIEM solutions.

Common anti-patterns:

- Relying only on local Windows Event Logs without centralized collection, making it difficult to correlate events across multiple systems and delaying incident response during critical situations.

- Collecting logs without proper retention policies or analysis capabilities, leading to storage inefficiencies and missed opportunities to identify patterns or security threats in the log data.
- Ignoring application-specific logs from Microsoft products like SQL Server and IIS, missing valuable insights into application performance, errors, and security events that could indicate potential issues.

Benefits of establishing this best practice:

- Centralized visibility and faster troubleshooting through consolidated log collection from all Microsoft workload components, enabling rapid identification of issues across the entire infrastructure stack.
- Enhanced security posture by forwarding logs to SIEM solutions for real-time threat detection, automated security analysis, and compliance reporting, improving overall security monitoring capabilities.
- Improved operational efficiency through automated log analysis, pattern recognition, and alerting capabilities that help identify recurring issues and optimize system performance proactively.

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

Implementation guidance

Implementing comprehensive logging for Microsoft workloads requires a systematic approach to collect, centralize, and analyze logs from multiple sources. Begin by identifying all log sources in your Microsoft environment, configure the Amazon CloudWatch Agent for log collection, and establish proper log retention and analysis processes. This approach ensures you capture critical events and application behaviors while maintaining efficient log management and enabling effective troubleshooting and security monitoring.

Implementation steps

1. Identify all log sources in your Microsoft workload including Windows Event Logs, SQL Server logs, IIS logs, and custom application logs.
2. Install and configure the Amazon CloudWatch Agent on Windows instances to collect and forward logs to Amazon CloudWatch Logs.
3. Configure log groups and streams in Amazon CloudWatch Logs with appropriate retention policies based on compliance and operational requirements.

4. Set up log filtering and metric filters to identify critical events and create automated alerts for important log patterns.
5. Implement log forwarding to SIEM solutions using AWS services like Amazon Data Firehose or AWS Lambda for enhanced security analysis.
6. Create Amazon CloudWatch Insights queries for efficient log analysis and troubleshooting across your Microsoft workload components.
7. Establish log monitoring dashboards and automated alerting for critical events and security incidents.
8. Implement log archiving strategies using Amazon S3 for long-term retention and compliance requirements.

Resources

Related documents:

- How do I upload my Windows logs to CloudWatch?
- Amazon EKS: Monitoring
- Centralized Logging for Windows Containers on Amazon EKS using Fluent Bit

Related tools:

- Amazon CloudWatch Logs
- SIEM on Amazon OpenSearch Service

MSFTOPS01-BP03 Implement Application Performance Monitoring (APM) for your Microsoft workload

Microsoft workloads developed with .NET and SQL technologies should also have Application Performance Monitoring (APM) implemented. Amazon CloudWatch Application Insights for .NET and SQL Server can be used for that purpose. AWS X-Ray can be used as well to improve traceability over the workload.

Desired outcome: Establish comprehensive Application Performance Monitoring (APM) for your Microsoft workloads, providing deep visibility into application behavior, performance bottlenecks, and user experience while enabling proactive optimization and rapid troubleshooting of .NET applications and SQL Server databases.

Common anti-patterns:

- Monitoring only infrastructure metrics without implementing application-level monitoring, missing critical insights into application performance, user experience, and business transaction flows that could indicate problems before they affect users.
- Implementing APM tools without proper configuration for Microsoft-specific technologies, failing to capture important .NET application metrics, SQL Server performance indicators, and transaction traces that are essential for effective troubleshooting.
- Using APM reactively only during incidents rather than proactively monitoring application performance trends, missing opportunities to optimize performance and prevent issues before they impact business operations.

Benefits of establishing this best practice:

- Enhanced application visibility and faster issue resolution through detailed monitoring of .NET application performance, SQL Server operations, and end-to-end transaction tracing, enabling rapid identification and resolution of performance bottlenecks.
- Improved user experience and business outcomes by monitoring application performance from the user perspective, identifying slow transactions, and optimizing critical business processes before they impact customer satisfaction.
- Proactive performance optimization through continuous monitoring of application metrics, enabling data-driven decisions for code optimization, database tuning, and infrastructure scaling to maintain optimal performance.

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

Implementation guidance

Implementing Application Performance Monitoring for Microsoft workloads requires a comprehensive approach that covers both .NET applications and SQL Server databases. Start by configuring Amazon CloudWatch Application Insights to automatically discover and monitor your Microsoft applications, then enhance visibility with AWS X-Ray for distributed tracing. This approach provides end-to-end visibility into application performance and enables proactive optimization of your Microsoft workloads.

Implementation steps

1. Enable Amazon CloudWatch Application Insights for your .NET applications and SQL Server instances to automatically discover and monitor application components.
2. Configure Application Insights to collect custom metrics specific to your Microsoft workload business logic and critical transactions.
3. Implement AWS X-Ray daemon in your .NET applications to track requests across distributed components and identify performance bottlenecks.
4. Set up custom dashboards in Amazon CloudWatch to visualize application performance metrics, error rates, and response times for critical business transactions.
5. Configure automated alerts for application performance thresholds, error rates, and anomaly detection to enable proactive issue identification.
6. Implement synthetic monitoring using Amazon CloudWatch Synthetics to continuously test critical application workflows and user journeys.
7. Establish performance baselines and regularly review APM data to identify optimization opportunities and performance trends.
8. Integrate APM data with your incident response processes to enable faster troubleshooting and root cause analysis.

Resources

Related documents:

- Monitoring .NET applications on AWS
- Amazon CloudWatch Synthetic monitoring (canaries)
- Monitor .NET and SQL Server applications using CloudWatch Application Insights
- .NET observability with Amazon CloudWatch and AWS X-Ray: Part 1 — Metrics
- .NET observability with Amazon CloudWatch and AWS X-Ray: Part 2 — Logging
- .NET Observability with Amazon CloudWatch and AWS X-Ray: Part 3 – Distributed Trace

Related tools:

- What is APM (Application Performance Monitoring)?
- Detect common application problems with CloudWatch Application Insights
- AWS X-Ray
- Amazon CloudWatch Synthetics

Operational automation

Operational automation is crucial for managing Microsoft workloads efficiently and consistently. This encompasses implementing management and governance solutions through AWS Systems Manager, establishing infrastructure as code (IaC) practices, maintaining standardized operating system images, and leveraging AWS managed services. By automating operational tasks, organizations can reduce human error, ensure consistent deployments, and minimize the operational overhead associated with managing Microsoft workloads on AWS.

MSFTOPS02: How do you implement operational automation for your Microsoft workload?

Safely automate your Microsoft workload from the infrastructure to the application layer, where possible, to limit human error and apply consistent engineering patterns.

Best practices

- MSFTOPS02-BP01 Implement Management and Governance solutions
- MSFTOPS02-BP02 Implement infrastructure deployment and update automation for your Microsoft workload
- MSFTOPS02-BP03 Implement operating system image control
- MSFTOPS02-BP04 Leverage managed services for your Microsoft workload

MSFTOPS02-BP01 Implement Management and Governance solutions

Set up Management and Governance solutions to ensure your Microsoft workload is patched and compliant with your security requirements. AWS Systems Manager functions as an operations hub for your workload, addressing fleet management, compliance, inventory, admin session management, state management, patch management, and running remote commands or scripts. Additionally, leverage AWS Systems Manager OpsCenter to provide a central location for viewing, investigating, and resolving operational issues related to your Microsoft workloads. OpsCenter aggregates and standardizes operations items across services while providing contextual investigation data about each operations item, related items, and related resources.

Desired outcome: Establish comprehensive management and governance capabilities for your Microsoft workloads through AWS Systems Manager, ensuring consistent patch management,

compliance monitoring, and centralized operational issue resolution while maintaining security standards and operational efficiency across your Windows-based infrastructure.

Common anti-patterns:

- Managing Microsoft workloads manually without centralized management tools, leading to inconsistent patch levels, security vulnerabilities, and increased operational overhead across the Windows infrastructure.
- Implementing patch management without proper testing and rollback procedures, risking system stability and application availability when updates are applied to production Microsoft workloads.
- Operating without centralized visibility into operational issues and compliance status, making it difficult to identify and resolve problems quickly across distributed Microsoft workload environments.

Benefits of establishing this best practice:

- Enhanced security posture and compliance through automated patch management, configuration compliance monitoring, and centralized governance of Microsoft workloads, reducing security vulnerabilities and ensuring adherence to organizational policies.
- Improved operational efficiency through centralized management capabilities, automated administrative tasks, and streamlined incident resolution processes that reduce manual effort and human error.
- Better visibility and control over Microsoft workload operations through centralized dashboards, automated reporting, and integrated operational issue management that enables faster problem resolution and improved system reliability.

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

Implementation guidance

Implementing comprehensive management and governance for Microsoft workloads requires a systematic approach using AWS Systems Manager capabilities. Begin by setting up the Systems Manager Agent on all Windows instances, configure patch management policies, and establish compliance monitoring. This approach ensures consistent management across your Microsoft workload infrastructure while maintaining security and operational standards.

Implementation steps

1. Install and configure the AWS Systems Manager Agent (SSM Agent) on all Windows instances in your Microsoft workload environment.
2. Set up AWS Systems Manager Patch Manager with maintenance windows and patch baselines appropriate for your Microsoft workload requirements.
3. Configure AWS Systems Manager Compliance to monitor configuration compliance and security standards across your Windows infrastructure.
4. Implement AWS Systems Manager Inventory to maintain an up-to-date inventory of software, configurations, and system information.
5. Set up AWS Systems Manager Session Manager for secure administrative access to Windows instances without requiring RDP or VPN connections.
6. Configure AWS Systems Manager State Manager to maintain consistent configuration states across your Microsoft workload components.
7. Implement AWS Systems Manager OpsCenter to centralize operational issue management and incident response for your Microsoft workloads.
8. Establish automated workflows using AWS Systems Manager Automation for common administrative tasks and incident response procedures.

Resources

Related documents:

- What is AWS Systems Manager?
- Patch Manager requirements and WSUS

Related tools:

- AWS Systems Manager Patch Manager
- AWS Systems Manager OpsCenter

MSFTOPS02-BP02 Implement infrastructure deployment and update automation for your Microsoft workload

Set up Infrastructure as Code (IaC) to apply patterns to the infrastructure of your Microsoft workload. You can use AWS CloudFormation to help model and deploy the required AWS resources based on templates. Third-party solutions, such as Terraform, are also useful for the case.

Desired outcome: Establish automated, repeatable, and version-controlled infrastructure deployment processes for your Microsoft workloads using Infrastructure as Code (IaC) practices, ensuring consistent environments, reducing deployment errors, and enabling rapid scaling and recovery of your Windows-based infrastructure.

Common anti-patterns:

- Deploying Microsoft workload infrastructure manually through the AWS console or CLI without using Infrastructure as Code, leading to configuration drift, inconsistent environments, and difficulty in reproducing deployments across different stages.
- Creating IaC templates without proper version control, testing, or documentation, making it difficult to track changes, rollback deployments, or collaborate effectively on infrastructure modifications.
- Implementing IaC without considering Microsoft workload-specific requirements such as Windows licensing, Active Directory integration, or SQL Server configuration, resulting in incomplete or non-functional deployments.

Benefits of establishing this best practice:

- Consistent and reliable deployments through standardized Infrastructure as Code templates that ensure all Microsoft workload components are deployed with the same configuration across development, testing, and production environments.
- Improved operational efficiency and reduced deployment time through automated infrastructure provisioning, enabling rapid scaling, disaster recovery, and environment replication for Microsoft workloads.
- Enhanced change management and auditability through version-controlled infrastructure templates that provide clear documentation of infrastructure changes and enable easy rollback when issues occur.
- IaC can help enforce security best practices by defining secure configurations within templates.
- By automating resource provisioning and de-provisioning, IaC can help optimize resource utilization and reduce costs.

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

Implementation guidance

Implementing Infrastructure as Code for Microsoft workloads requires careful consideration of Windows-specific requirements and AWS services. Start by identifying your Microsoft workload components and their dependencies, then create modular IaC templates that can be reused across environments. This approach ensures consistent deployments while accommodating the specific needs of Windows-based applications and services.

Implementation steps

1. Analyze your Microsoft workload architecture and identify all AWS resources, dependencies, and configuration requirements.
2. Choose an appropriate IaC tool (AWS CloudFormation, AWS CDK, or Terraform) based on your team's expertise and organizational requirements.
3. Create modular IaC templates for common Microsoft workload components such as Windows EC2 instances, SQL Server databases, and Active Directory services.
4. Implement version control for your IaC templates using Git repositories with proper branching strategies and code review processes.
5. Set up automated testing and validation for your IaC templates using tools like AWS CloudFormation Guard or Terraform validation.
6. Establish CI/CD pipelines for infrastructure deployment using AWS CodePipeline, GitHub Actions, or similar tools to automate template deployment.
7. Create environment-specific parameter files and configuration management to support deployment across development, testing, and production environments.
8. Implement infrastructure monitoring and drift detection to ensure deployed resources remain consistent with your IaC templates.

Resources

Related documents:

- AWS CloudFormation and Windows

Related videos:

- AWS re:Invent 2024 - Use generative AI to optimize cloud operations for Microsoft workloads (XNT312)

Related examples:

- Use Terraform to Build Microsoft Infrastructure on AWS

Related tools:

- AWS CloudFormation
- Terraform AWS Windows Workloads

MSFTOPS02-BP03 Implement operating system image control

AWS has developed a set of Amazon Machine Images (AMIs) for popular Microsoft solutions with License Included software, enabling standardized and automated deployments of Windows Server instances in Amazon EC2. You can either use the latest images that are built by AWS or create your own. You can subscribe to AWS Windows AMI notifications or create custom AMIs of your own to apply the standards required by your environment, such as regional settings, agents, base patches, and general tools. EC2 Image Builder is a fully managed AWS service that helps you to automate the creation, management, and deployment of customized, secure, and up-to-date server images.

Desired outcome: Establish standardized, secure, and consistently configured Windows Server images for your Microsoft workloads through automated AMI management, ensuring rapid deployment capabilities, security compliance, and operational consistency across all Windows instances in your environment.

Common anti-patterns:

- Using outdated or unpatched base AMIs without regular updates, leading to security vulnerabilities and inconsistent configurations across Windows instances in your Microsoft workload environment.
- Creating custom AMIs manually without automation or version control, resulting in configuration drift, difficulty in reproducing images, and challenges in maintaining security and compliance standards.
- Deploying Windows instances without standardized configurations, leading to operational inconsistencies, increased troubleshooting time, and potential security gaps across your Microsoft workload infrastructure.

Benefits of establishing this best practice:

- Faster deployment and improved consistency through standardized Windows Server images that include all necessary configurations, patches, and tools, reducing instance launch time and ensuring uniform environments.
- Enhanced security posture through automated image updates and patch management, ensuring all Windows instances are deployed with the latest security updates and compliance configurations.
- Reduced operational overhead through automated AMI creation and management processes that eliminate manual image preparation tasks and ensure consistent, repeatable deployments across environments.

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

Implementation guidance

Implementing operating system image control for Microsoft workloads requires a systematic approach to AMI management and automation. Begin by establishing your image requirements and standards, then implement EC2 Image Builder to automate the creation and maintenance of custom Windows AMIs. This approach ensures consistent, secure, and up-to-date images for your Microsoft workload deployments.

Implementation steps

1. Define your Windows Server image requirements including base configurations, security settings, required software, and organizational standards.
2. Set up EC2 Image Builder with appropriate IAM roles and permissions to automate AMI creation and management processes.
3. Create Image Builder recipes that include your required Windows configurations, software installations, and security hardening steps.
4. Configure automated testing pipelines to validate AMI functionality and security compliance before distribution.
5. Implement automated AMI distribution to multiple AWS regions and accounts as needed for your Microsoft workload deployment strategy.
6. Set up AMI lifecycle management policies to automatically deprecate and delete outdated images while maintaining required retention periods.
7. Subscribe to AWS Windows AMI notifications to stay informed about security updates and new releases for base Windows Server images.

8. Establish regular AMI update schedules to incorporate security patches, software updates, and configuration changes into your custom images.

Resources

Related documents:

- What is EC2 Image Builder?
- How Amazon creates AWS Windows AMIs

Related tools:

- Amazon EC2 AMIs
- EC2 Image Builder

MSFTOPS02-BP04 Leverage managed services for your Microsoft workload

To reduce operational overhead, implement the use of AWS managed services to address your Microsoft workload requirements. Consider AWS Managed Microsoft Active Directory, Amazon Relational Database Service for SQL Server, Amazon FSx for Windows File Server, Amazon FSx for NetApp ONTAP, AWS Elastic Beanstalk, and others.

Desired outcome: Reduce operational complexity and overhead for your Microsoft workloads by strategically adopting AWS managed services that handle infrastructure management, patching, backups, and scaling automatically, allowing your team to focus on application development and business value rather than infrastructure maintenance.

Common anti-patterns:

- Managing Microsoft infrastructure components manually when equivalent AWS managed services are available, leading to increased operational overhead, higher maintenance costs, and potential security vulnerabilities from delayed patching.
- Choosing self-managed solutions without evaluating the total cost of ownership, including operational effort, expertise requirements, and ongoing maintenance compared to AWS managed service alternatives.

- Implementing managed services without proper integration planning, resulting in architectural complexity, security gaps, or performance issues that could have been avoided with better design considerations.

Benefits of establishing this best practice:

- Significantly reduced operational overhead through AWS-managed infrastructure components that handle patching, backups, monitoring, and scaling automatically, freeing up resources for higher-value activities.
- Improved reliability and availability through AWS-managed services that provide built-in high availability, disaster recovery, and automated failover capabilities designed and tested by AWS experts.
- Enhanced security posture through managed services that include automatic security updates, encryption capabilities, and compliance features that are maintained and updated by AWS according to industry best practices.

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

Implementation guidance

Implementing AWS managed services for Microsoft workloads requires careful evaluation of your current architecture and identification of components that can be replaced or enhanced with managed alternatives. Begin by assessing your Microsoft workload components and their operational requirements, then systematically migrate to appropriate AWS managed services while ensuring proper integration and security.

Implementation steps

1. Conduct a comprehensive assessment of your current Microsoft workload architecture to identify components suitable for managed service replacement.
2. Evaluate AWS managed service options including AWS Managed Microsoft AD, Amazon RDS for SQL Server, Amazon FSx for Windows File Server, and AWS Elastic Beanstalk.
3. Develop a migration strategy that prioritizes high-maintenance components and considers dependencies between services and applications.
4. Implement pilot migrations with non-critical workloads to validate managed service configurations and integration patterns.

5. Configure managed services with appropriate security settings, backup policies, and monitoring to meet your operational requirements.
6. Establish connectivity and integration between managed services and existing Microsoft workload components using VPC networking and security groups.
7. Migrate production workloads systematically, ensuring proper testing and rollback procedures are in place for each migration phase.
8. Update operational procedures and documentation to reflect the new managed service architecture and reduced maintenance requirements.

Resources

Related documents:

- AWS Managed Services for Microsoft Workloads

Related tools:

- AWS Managed Microsoft AD
- Amazon RDS for SQL Server
- Amazon FSx for Windows File Server

Security

The security pillar within the Microsoft Workloads Lens extends the foundational security principles of the AWS Well-Architected Framework with specialized guidance tailored for Microsoft-centric environments. Rather than replacing the core Well-Architected security practices, this lens amplifies them by addressing the unique security considerations that arise when running Microsoft technologies on AWS.

This extension provides targeted security strategies for protecting Windows Server infrastructures, Active Directory implementations, SQL Server databases, Exchange Server deployments, SharePoint environments, and .NET applications. The lens integrates Microsoft security capabilities with AWS security services—including AWS IAM for identity management, AWS KMS for encryption, Amazon GuardDuty for threat detection, AWS Security Hub CSPM for centralized security posture management, and NitroTPM for hardware-based security—creating a comprehensive defense-in-depth approach.

By building upon the Well-Architected Framework's security foundation, this lens assists organizations in maintaining Microsoft workload security best practices while fully using AWS Cloud security capabilities, resulting in a hybrid security model that addresses both traditional Microsoft security requirements and modern cloud security paradigms. This includes implementing defense in depth strategies for Windows Server environments, Active Directory services, SQL Server databases, Exchange Server, SharePoint, and .NET applications, while using with AWS security services (including IAM, KMS, GuardDuty, Security Hub CSPM, and NitroTPM).

Establish proper identity and access management through AWS IAM integration with Active Directory, implement encryption at rest and in transit using AWS KMS and Microsoft TDE, and deploy comprehensive monitoring capabilities through AWS CloudTrail, Amazon CloudWatch, and Microsoft security logging. This strategy maintains strong security postures that protect against advanced persistent threats, insider threats, and compliance violations while adhering to regulatory requirements such as HIPAA, PCI DSS, SOX, and GDPR.

The pillar emphasizes the AWS shared responsibility model, where AWS secures the underlying cloud infrastructure, hypervisor, and physical facilities while customers are responsible for securing their Microsoft workloads within the cloud, including operating system hardening, application security, data classification and protection, network configuration, and identity management. This includes implementing security best practices such as least privilege access, regular security assessments, automated patch management through AWS Systems Manager, and incident response procedures tailored to Microsoft environments.

Key security considerations for Microsoft workloads include securing hybrid connectivity between on-premises Active Directory and AWS through AWS Direct Connect or Site-to-Site VPN, implementing proper network segmentation using VPCs and security groups, enabling comprehensive logging and monitoring for Windows events and Microsoft application logs, and establishing disaster recovery procedures that maintain security controls across Regions while maintaining business continuity for mission-critical Microsoft applications.

Focus areas

Definitions

- **Transparent Data Encryption (TDE):** A Microsoft SQL Server feature that performs real-time I/O encryption and decryption of data and log files to protect data at rest without requiring changes to applications.
- **Always Encrypted:** A SQL server feature that provides client-side encryption with separation between data owners and data managers, encrypting sensitive data in the database, during transit, and while being processed.
- **Windows event logs:** Windows logging system that records system, security, and application events, providing crucial audit trails and security monitoring capabilities for Microsoft workloads.
- **Performance counters:** Windows-based metrics that provide detailed information about system and application performance, including security-related metrics for monitoring and alerting purposes.

Design principles

Refer to design principles in the Well-Architected Framework Security Pillar for core security concepts. Additionally, consider these Microsoft-specific security aspects:

- **Microsoft-specific security configurations:** Use Microsoft security baselines, Active Directory Group Policies, and Windows-specific security features like Windows Defender, AppLocker, and BitLocker for enhanced workload protection.
- **Identity integration patterns:** Implement proper integration between AWS IAM and Microsoft Active Directory services (either AWS Managed Microsoft AD or self-managed AD) for secure authentication across hybrid environments.

Workload security

Securing Microsoft workloads requires a comprehensive approach that addresses the operating system, applications, databases, and underlying infrastructure. This focus area explores how to implement security hardening, patch management, and protective measures across each component of your Microsoft environment on AWS, verifying that each layer contributes to an overall robust security posture.

MSFTSEC01: How do you secure your Microsoft workload, including its database, operating system, and underlying infrastructure?

It's important to harden Microsoft applications and their underlying components, including operating systems, databases, cloud services, and infrastructure. This hardening should be applied to Microsoft environments, both production and non-production, at levels appropriate to your organization's needs.

Best practices

- MSFTSEC01-BP01 Protect the operating system
- MSFTSEC01-BP02 Secure the Microsoft application and database
- MSFTSEC01-BP03 Develop a comprehensive software update strategy

MSFTSEC01-BP01 Protect the operating system

Securing the operating system that hosts your Microsoft application is crucial in blocking unauthorized access, securing software availability, and maintaining the stability of your critical systems. By following Microsoft and AWS recommendations for Windows Server environments, you can reduce the risk of malicious attacks.

This is particularly important when using unmanaged or non-serverless services. Implementing these best practices creates a robust foundation for your Microsoft workload, enhancing overall security and resilience. Remember that a well-protected operating system forms a critical layer in your defense strategy, safeguarding your application and data from potential threats.

Desired outcome: Establish a hardened Windows Server environment that follows security best practices, implements proper access controls, and maintains up-to-date security configurations to protect your Microsoft workload from operating system-level threats and vulnerabilities.

Common anti-patterns:

- Using default Windows Server configurations without implementing security hardening measures, leaving systems vulnerable to common attack vectors and exploitation techniques.
- Failing to implement proper user account management and access controls, allowing excessive privileges or shared accounts that increase security risks and make it difficult to track user activities.
- Neglecting to configure Windows Firewall and network security settings appropriately, potentially exposing unnecessary services or ports to network-based attacks.

Benefits of establishing this best practice:

- Reduced attack surface through proper system hardening, disabling unnecessary services, and implementing security configurations that minimize potential entry points for malicious actors.
- Enhanced access control and accountability through proper user account management, role-based access controls, and audit logging that tracks system access and changes.
- Improved regulatory posture by implementing security baselines and configurations that align with industry standards and regulatory requirements for Windows Server environments.

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

Implementation guidance

Protecting the Windows Server operating system requires a systematic approach to security hardening that addresses user accounts, services, network configuration, and system monitoring. Start by implementing Microsoft security baselines and AWS-recommended configurations, then customize these settings based on your specific workload requirements. This comprehensive approach verifies that your Windows Server instances provide a secure foundation for your Microsoft applications while maintaining operational functionality.

Implementation steps

1. Apply Microsoft Security Baselines for Windows Server using Group Policy or local security policies to establish fundamental security configurations.
2. Configure Windows Firewall with appropriate rules that allow only necessary network traffic and block potentially malicious connections.
3. Implement proper user account management with strong password policies, account lockout settings, and regular review of user privileges.
4. Disable unnecessary Windows services and features that are not required for your Microsoft workload to reduce the attack surface.
5. Configure Windows Event Logging to capture security events, system changes, and access attempts for monitoring and audit purposes.
6. Enable Microsoft Windows Defender on your EC2 instances running Windows Server for local anti-malware protection.
7. Implement regular security assessments using AWS Systems Manager Compliance or third-party security scanning tools.
8. Establish automated patch management processes using AWS Systems Manager Patch Manager to keep the operating system current with security updates.

Resources

Related documents:

- Security best practices for Windows instances
- Add optional Windows Server components to Amazon EC2 Windows instances
- Security baselines

Related tools:

- AWS Systems Manager Patch Manager
- AWS Systems Manager Compliance
- Windows Security Baseline

MSFTSEC01-BP02 Secure the Microsoft application and database

Maintaining strong security at both the database and application layers is crucial, as even read-only access by malicious actors could compromise critical business data. To protect your Microsoft

environment, implement security best practices including the least access principle, least privilege model, encryption at rest, and encryption in transit. These measures help safeguard your Microsoft application and database against unauthorized access and potential data breaches, maintaining the confidentiality and integrity of your business-critical information.

Desired outcome: Establish comprehensive security controls for Microsoft applications and databases that implement defense in depth strategies, proper access controls, and encryption mechanisms to protect sensitive data and block unauthorized access at the application and database layers.

Common anti-patterns:

- Using default database and application configurations without implementing security hardening, leaving systems vulnerable to common attack vectors such as SQL injection, privilege escalation, and unauthorized data access.
- Implementing overly permissive database and application access controls that grant excessive privileges to users or applications, violating the principle of least privilege and increasing the risk of data breaches.
- Storing sensitive data in plaintext or using weak encryption methods, making it vulnerable to exposure if the database or application is compromised or if data is intercepted during transmission.

Benefits of establishing this best practice:

- Enhanced data protection through comprehensive encryption strategies that secure sensitive information both at rest and in transit, reducing the risk of data exposure even if systems are compromised.
- Improved access control and audit capabilities through implementation of least privilege principles and detailed logging, enabling better monitoring of data access patterns and potential security incidents.
- Reduced attack surface through application and database hardening measures that eliminate common vulnerabilities and implement security best practices specific to Microsoft technologies.

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

Implementation guidance

Securing Microsoft applications and databases requires a multi-layered approach that addresses authentication, authorization, encryption, and monitoring. Focus on implementing Microsoft SQL Server security features alongside AWS security services to create a comprehensive protection strategy. This includes configuring proper access controls, enabling encryption mechanisms, and establishing monitoring capabilities that provide visibility into application and database activities.

Implementation steps

1. Configure SQL Server authentication using Windows Authentication mode or mixed mode with strong password policies and account management practices.
2. Implement database-level security through proper user roles, schema permissions, and row-level security where appropriate for your Microsoft SQL Server environment.
3. Enable SQL Server audit logging to track database access, data modifications, and administrative activities for compliance and security monitoring.
4. Configure application-level security controls including input validation, output encoding, and secure session management for .NET applications.
5. Implement database connection security using encrypted connections (SSL/TLS) and connection string protection mechanisms.
6. Enable SQL Server security features such as dynamic data masking for sensitive data protection and Always Encrypted for client-side encryption.
7. Configure network security controls including database firewall rules and network segmentation to limit database access to authorized sources.
8. Establish regular security assessments and vulnerability scanning for both applications and databases using AWS and Microsoft security tools.

Resources

Related documents:

- Security best practices for Microsoft SQL Server on AWS
- Security Best Practices for Modernizing .NET Framework Applications on AWS

Related tools:

- SQL Server Management Studio
- AWS Database Migration Service

- Amazon RDS for SQL Server

MSFTSEC01-BP03 Develop a comprehensive software update strategy

Microsoft regularly releases scheduled security updates and emergency patches to address vulnerabilities. Stay informed about the latest security advisories from relevant vendors. It's crucial to keep your Microsoft application and its underlying components up-to-date with the latest security fixes on a regular schedule to avoid security gaps. Develop a plan for applying critical emergency patches when released. Consider implementing automated processes to streamline this update process, which keeps your Microsoft environment secure and current with minimal manual intervention.

Desired outcome: Establish an automated and systematic approach to patch management that verifies Microsoft workloads receive timely security updates while maintaining system stability and minimizing downtime through proper testing and deployment procedures.

Common anti-patterns:

- Applying patches without proper testing or staging procedures, potentially introducing system instability or breaking critical applications during production deployments.
- Delaying security updates for extended periods due to fear of system disruption, leaving systems vulnerable to known exploits and security threats that could have been avoided.
- Managing patches manually across multiple systems without automation, leading to inconsistent patch levels, missed updates, and increased administrative overhead that doesn't scale effectively.

Benefits of establishing this best practice:

- Reduced security vulnerabilities through timely application of security patches and updates, minimizing the window of exposure to known threats and exploits targeting Microsoft technologies.
- Improved operational efficiency through automated patch management processes that reduce manual effort, maintain consistency across environments, and provide better visibility into patch compliance status.
- Enhanced system stability and reliability through structured testing and deployment procedures that validate patches before production deployment, reducing the risk of patch-related outages or application failures.

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

Implementation guidance

Developing a comprehensive software update strategy requires balancing security needs with operational stability. Implement a structured approach that includes automated patch management, testing procedures, and emergency response capabilities.

Use AWS Systems Manager Patch Manager to automate routine updates while maintaining control over critical systems through approval processes and maintenance windows. For Windows instances in private subnets without internet connectivity, consider implementing Windows Server Update Services (WSUS) in public subnets to provide local patch distribution while still using Systems Manager for orchestration and compliance monitoring.

Implementation steps

1. Establish patch management policies that define update schedules, testing requirements, and approval processes for different types of systems and environments.
2. Configure AWS Systems Manager Patch Manager to automate patch deployment with appropriate maintenance windows and approval workflows.
3. Create staging environments that mirror production systems for testing patches before deployment to critical workloads.
4. Set up patch groups and baselines in Systems Manager to manage different update requirements for various system types and criticality levels.
5. Implement monitoring and alerting for patch compliance status using AWS Systems Manager Compliance and Amazon CloudWatch dashboards.
6. Establish emergency patch procedures for critical security vulnerabilities that require immediate attention outside of normal maintenance windows.
7. Configure automated rollback procedures and system snapshots before patch deployment to enable quick recovery if issues occur.
8. Create regular reporting mechanisms to track patch adherence, update success rates, and identify systems that require attention.

Resources

Related documents:

- AWS Systems Manager Patch Manager
- Microsoft Security Update Guide

Related tools:

- AWS Systems Manager
- Windows Server Update Services (WSUS)
- Microsoft Update Catalog

Access management

Controlling access to Microsoft workloads involves implementing proper authentication, authorization, and auditing mechanisms that integrate with both Microsoft and AWS identity services. This focus area addresses how to establish centralized identity management, implement least privilege principles, and maintain comprehensive access monitoring across your Microsoft environment.

MSFTSEC02: How do you manage and regulate user access to your Microsoft workload environment?

To control access to your Microsoft workload, utilize the authentication and authorization tools provided by AWS, Microsoft, and trusted third-party vendors. Implement a least-privilege approach, verifying that users and systems have only the permissions necessary for their roles.

Best practices

- MSFTSEC02-BP01 Align your Microsoft workload access with organizational identity strategy
- MSFTSEC02-BP02 Implement logging to track access and authorization changes

MSFTSEC02-BP01 Align your Microsoft workload access with organizational identity strategy

Microsoft workloads, including .NET applications and SQL Server environments, typically integrate with Active Directory (AD) or similar identity management systems. A centralized approach to user management, regardless of the specific solution, can significantly reduce security risks and operational complexity. This comprehensive approach enhances security, streamlines user access, and provides a consistent identity management experience across your Microsoft workloads in the cloud.

Desired outcome: Establish a unified identity management strategy that integrates Microsoft workloads with organizational identity systems, providing centralized authentication, authorization, and user lifecycle management while maintaining security and operational efficiency across cloud and on-premises environments.

Common anti-patterns:

- Creating isolated identity silos for different Microsoft workloads without integration with centralized identity systems, leading to inconsistent access controls and increased administrative overhead.
- Implementing local user accounts on individual systems instead of using centralized identity management, making it difficult to maintain consistent security policies and user lifecycle management.
- Failing to implement multi-factor authentication (MFA) and single sign-on (SSO) capabilities, resulting in weaker security posture and poor user experience across Microsoft applications.

Benefits of establishing this best practice:

- Enhanced security through centralized identity management that enables consistent policy enforcement, better access control, and improved visibility into user activities across your Microsoft workloads.
- Reduced operational complexity through unified user lifecycle management, automated provisioning and deprovisioning, and streamlined access request and approval processes.
- Improved user experience through single sign-on capabilities that reduce password fatigue while maintaining strong authentication requirements including multi-factor authentication.

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

Implementation guidance

When designing your identity strategy for Microsoft environments in AWS, consider using AWS services and compatible third-party tools to implement centralized user management, single sign-on (SSO) capabilities, and multi-factor authentication (MFA). Aligning Microsoft workload access with organizational identity strategy requires careful planning and integration between AWS services, Microsoft identity technologies, and existing organizational systems. Focus on establishing trust relationships, implementing proper authentication mechanisms, and providing a simple-to-use user experience while maintaining security controls.

Note: While local service accounts or isolated identity management may be necessary in specific technical scenarios, these approaches should be considered a last resort when centralized identity integration is not feasible. Local accounts increase security risks, administrative overhead, and compliance challenges. Always prioritize integration with organizational identity providers and use temporary credentials whenever possible, as recommended by AWS security best practices.

Implementation steps

1. Assess current organizational identity infrastructure and determine integration requirements for Microsoft workloads on AWS:
   - Document existing Active Directory schema, domains, and trust relationships.
   - Map out current authentication flows and identify Microsoft applications requiring integration.
2. Choose appropriate Directory Service option (AWS Managed Microsoft AD, AD Connector, or Active Directory on EC2) based on your organizational needs and existing infrastructure:
   - Compare directory service options against requirements (scale, features, management overhead).
   - Evaluate cost implications and licensing requirements for each option.
   - Conduct proof-of-concept testing with selected directory service option.
3. Establish trust relationships between AWS-hosted Active Directory and on-premises Active Directory domains if hybrid connectivity is required:
   - Configure VPN or Direct Connect for secure hybrid connectivity.
   - Set up forest and domain trusts and verify DNS resolution between environments.
   - Test authentication flows across trusted domains.
4. Configure single sign-on (SSO) using AWS IAM Identity Center or compatible third-party solutions to provide unified access across Microsoft applications:
   - Set up AWS IAM Identity Center integration with chosen directory service.
   - Configure application-specific SSO connectors for Microsoft workloads.
   - Test SSO flows for required applications.
5. Implement multi-factor authentication (MFA) for user accounts accessing Microsoft workloads using AWS IAM Identity Center MFA capabilities, which support various authentication methods including TOTP, SMS, email, and passwordless options like Windows Hello for Business and FIDO2 security keys, or integrate with AWS Managed Microsoft AD using RADIUS-based MFA solutions:
   - Define MFA policies and enforcement rules.
   - Configure selected MFA methods in AWS IAM Identity Center or RADIUS.
   - Test MFA enrollment and authentication processes.

6. Set up automated user provisioning and deprovisioning processes that integrate with HR systems and organizational identity management workflows:
   - Design user lifecycle workflows and approval processes.
   - Implement SCIM or PowerShell-based provisioning scripts.
   - Configure integration between HR systems and AWS directory service.
7. Configure role-based access control (RBAC) that aligns with organizational roles and responsibilities while following least privilege principles:
   - Define role hierarchy and permission sets.
   - Create security groups and implement group-based assignments.
   - Document and implement approval workflows for role changes.
8. Establish monitoring and auditing capabilities to track identity-related activities and adhere to organizational security policies:
   - Configure CloudWatch logging for directory service events.
   - Set up alerts for suspicious authentication activities.
   - Implement regular compliance reporting and access reviews.

Resources

Related documents:

- Directory services options in AWS
- Security considerations for Active Directory Domain Services on AWS
- Directory Service

Related tools:

- AWS IAM Identity Center
- AWS Managed Microsoft AD
- Active Directory Federation Services (ADFS)

MSFTSEC02-BP02 Implement logging to track access and authorization changes

Regularly log, analyze, and audit user access and authorization events in your Microsoft systems. Consolidate security events from Microsoft applications and databases with other architectural components to enable comprehensive tracing during security incidents. Implement a centralized

security information and event management (SIEM) system to automate event analysis, allowing your operations team to identify unusual or suspicious activities.

This integrated approach to security monitoring enhances your ability to detect and respond to potential threats across your entire Microsoft environment, strengthening your overall security posture and enabling more effective incident management.

Desired outcome: Establish comprehensive logging and monitoring capabilities that capture, analyze, and correlate access and authorization events across Microsoft workloads, enabling rapid detection of security incidents and providing detailed audit trails for compliance and forensic analysis.

Common anti-patterns:

- Collecting logs from individual systems without centralized aggregation and correlation, making it difficult to identify patterns or conduct comprehensive security analysis across the Microsoft environment.
- Focusing only on successful authentication events while ignoring failed attempts, authorization changes, and privilege escalations that could indicate security threats or policy violations.
- Storing logs locally on individual systems without proper retention, backup, or protection mechanisms, risking log loss during security incidents when they are most needed for investigation.

Benefits of establishing this best practice:

- Enhanced threat detection through comprehensive logging that captures authentication attempts, authorization changes, and access patterns, enabling early identification of suspicious activities and potential security breaches.
- Improved incident response capabilities through centralized log aggregation and correlation that provides security teams with complete visibility into user activities and system events during security investigations.
- Strengthened regulatory posture through detailed audit trails that document user access, privilege changes, and administrative activities, supporting regulatory requirements and internal security policies.

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

Implementation guidance

Implementing comprehensive access and authorization logging requires a systematic approach to log collection, centralization, and analysis. Focus on capturing relevant security events from each Microsoft workload component while protecting, retaining, and documenting for analysis and compliance purposes.

Implementation steps

1. Configure Windows Event Logging on your Microsoft workload systems to capture security events, including authentication attempts, privilege changes, and administrative activities.
2. Enable SQL Server audit logging to track database access, permission changes, and data access patterns for comprehensive database security monitoring.
3. Set up Amazon CloudWatch Logs or AWS CloudTrail to collect and centralize logs from Microsoft workloads running on AWS infrastructure.
4. Configure log forwarding from Microsoft applications and databases to a centralized SIEM solution such as Amazon Security Lake or third-party security platforms.
5. Implement log parsing and normalization for consistent event formats, enabling effective correlation across different Microsoft technologies and AWS services.
6. Set up automated alerting and monitoring rules to detect suspicious access patterns, failed authentication attempts, and unauthorized privilege escalations.
7. Establish log retention policies that meet regulatory requirements while balancing storage costs and operational needs for security analysis.
8. Create regular reporting and analysis procedures to review access patterns, identify security trends, and validate the effectiveness of access controls.

Resources

Related documents:

- Security Best Practices for Modernizing .NET Framework Applications on AWS - Application Logging
- AWS Security Best Practices

Related tools:

- Amazon CloudWatch Logs
- AWS CloudTrail

- Amazon Security Lake
- SQL Server Audit

Data protection

Microsoft workloads often handle sensitive enterprise data that requires protection through encryption, access controls, and secure data handling practices. This focus area explores how to implement comprehensive data protection strategies that secure data at rest, in transit, and in use, while using both Microsoft and AWS security capabilities.

MSFTSEC03: How do you protect your Microsoft workload data?

Whether using Microsoft SQL Server or other solutions, it's best practice to encrypt data both at rest and in transit, employing multiple encryption mechanisms to meet internal and external security requirements.

Best practices

- MSFTSEC03-BP01 Encrypt data stored in Microsoft workloads
- MSFTSEC03-BP02 Enable Always Encrypted feature for SQL Server
- MSFTSEC03-BP03 Use Trusted Platform Module (TPM) technology for hardware-based security on your instances

MSFTSEC03-BP01 Encrypt data stored in Microsoft workloads

Data at rest encompasses the entirety of your digitally stored information. Encryption is crucial to verify that this data remains visible only to authorized users and stays protected, even if storage or database access is compromised independently of the application. For Microsoft SQL Server environments, consider implementing Transparent Data Encryption (TDE). This technology provides robust encryption at rest solution specifically designed for Microsoft databases, offering strong protection for sensitive data without significant changes to your application architecture.

By employing these encryption methods, you enhance the security of your stored data, mitigating risks associated with unauthorized access and potential data breaches in your Microsoft SQL Server deployments.

Desired outcome: Implement comprehensive encryption at rest for sensitive data stored in Microsoft workloads, protecting data even if underlying storage systems are compromised while maintaining application performance and operational efficiency.

Common anti-patterns:

- Storing sensitive data in plaintext without any encryption protection, leaving it vulnerable to unauthorized access if storage systems or database files are compromised.
- Implementing encryption inconsistently across different data stores or only encrypting some sensitive data while leaving other critical information unprotected.
- Using weak encryption algorithms or poor key management practices that could be compromised, effectively negating the security benefits of encryption.

Benefits of establishing this best practice:

- Enhanced data protection through strong encryption that renders data unreadable to unauthorized users even if they gain access to storage systems or database files.
- Improved regulatory posture by meeting regulatory requirements for data protection that mandate encryption of sensitive information at rest.
- Reduced impact of security incidents through encryption that limits the value of stolen data and reduces the scope of potential data breaches.

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

Implementation guidance

When encrypting Microsoft workloads at rest, start with Windows-based solutions like SQL Server TDE to protect databases and files. Establish a secure key management process and monitor performance metrics to maintain application responsiveness.

Implementation steps

1. Identify each data store containing sensitive information in your Microsoft workload, including SQL Server databases, file systems, and application data repositories.
2. Enable Transparent Data Encryption (TDE) on SQL Server databases to encrypt data files, log files, and backup files at the database level.
3. Configure AWS Key Management Service (KMS) or SQL Server key management to securely store and manage encryption keys with proper access controls.

4. Implement file system encryption using Amazon EBS encryption for EC2 instance storage volumes.
5. Enable encryption for backup files and verify that database backups maintain encryption protection during storage and transfer operations.
6. Configure application-level encryption for sensitive data fields that require additional protection beyond database-level encryption.
7. Establish key rotation policies and procedures to regularly update encryption keys while maintaining data accessibility and system availability.
8. Monitor encryption status and key usage through logging and alerting mechanisms to maintain continuous protection and detect any encryption failures.

Resources

Related documents:

- Best Practices for Deploying Microsoft SQL Server on Amazon EC2 - Encryption at Rest
- SQL Server Transparent Data Encryption

Related tools:

- AWS Key Management Service (KMS)
- Amazon EBS Encryption
- BitLocker Drive Encryption

MSFTSEC03-BP02 Enable Always Encrypted feature for SQL Server

Microsoft SQL Server's *Always Encrypted* feature provides robust data protection using client-side encryption with certificates. This technology creates a clear separation between data owners who can view the information and data managers who shouldn't have access. It effectively safeguards sensitive data by encrypting it in the database, during transit, and even while being processed. Always Encrypted is available not only in on-premises SQL Server deployments but also in cloud environments, including Amazon RDS for SQL Server and SQL Server instances running on Amazon EC2. By implementing Always Encrypted, organizations can enhance their data security posture, particularly when handling sensitive information in Microsoft SQL Server environments on AWS.

Desired outcome: Implement client-side encryption capabilities that protect sensitive data throughout its entire lifecycle, keeping data encrypted even during processing and is only accessible to authorized applications and users with proper decryption keys.

Common anti-patterns:

- Processing sensitive data in plaintext within applications or databases, exposing it to potential compromise during computation or memory access by unauthorized processes.
- Implementing encryption in use inconsistently across different data types or applications, leaving some sensitive information vulnerable during processing operations.
- Using client-side encryption without proper key management or secure key distribution mechanisms, potentially compromising the security benefits of the encryption implementation.

Benefits of establishing this best practice:

- Maximum data protection through encryption that maintains data confidentiality even during processing operations, verifying that sensitive information is never exposed in plaintext to unauthorized systems or users.
- Enhanced separation of duties between data owners and data managers, allowing database administrators to manage systems without accessing sensitive business data.
- Improved regulatory capabilities for highly regulated industries that require the highest levels of data protection, including protection during data processing operations.

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

Implementation guidance

When implementing Always Encrypted, first identify which sensitive data elements truly need this protection layer. Then map out where to apply based on your data flows and access patterns. Evaluate feature limitations and potential performance impacts before proceeding, as encryption/decryption operations can affect response times and resource usage.

Implementation steps

1. Identify highly sensitive data elements that require protection during processing, such as personally identifiable information (PII), financial data, or healthcare records.
2. Configure SQL Server Always Encrypted for identified sensitive columns, choosing appropriate encryption types (deterministic or randomized) based on query requirements.

3. Set up certificate-based key management for Always Encrypted, and properly store and distribute keys to authorized client applications.
4. Modify client applications to handle encrypted data operations, including proper connection string configuration and query modifications.
5. Implement secure key provisioning mechanisms that allow authorized applications to access encryption keys while blocking unauthorized access.
6. Configure column master keys and column encryption keys with appropriate permissions and access controls to maintain separation of duties.
7. Test application functionality with encrypted data for proper operation and performance while maintaining data protection.
8. Establish monitoring and auditing procedures to track key usage, encryption operations, and access to sensitive encrypted data.

Resources

Related documents:

- Best Practices for Deploying Microsoft SQL Server on Amazon EC2 - Encryption in Use
- SQL Server Always Encrypted

Related tools:

- SQL Server Management Studio
- Always Encrypted Wizard
- Azure Key Vault (for hybrid scenarios)

MSFTSEC03-BP03 Use Trusted Platform Module (TPM) technology for hardware-based security on your instances

The AWS Nitro Trusted Platform Module (NitroTPM) is a virtual TPM 2.0 device that's fully integrated with the AWS Nitro System, providing hardware-based security features for EC2 instances running Windows Server 2016 and later, as well as supported Linux distributions. It enables secure boot functionality, disk encryption, and enhanced protection of sensitive data and keys directly through the operating system.

When enabled on instance launch, NitroTPM allows Windows to use BitLocker disk encryption, measured boot capabilities, and Windows Hello for Business authentication. The TPM functionality

is implemented through the Nitro security chip, which processes cryptographic operations and sensitive key material in an isolated, hardware-protected environment separate from the instance's CPU and hypervisor. This hardware-based root of trust helps meet requirements for regulated workloads and supports Windows security features that require TPM 2.0, including Windows Defender System Guard, Credential Guard, and Device Guard.

NitroTPM integrates with Windows' built-in security tools and third-party applications that rely on TPM capabilities, making it particularly valuable for enterprises requiring enhanced security posture for their Windows workloads on AWS. Additionally, NitroTPM supports attestation capabilities, allowing applications to verify the integrity and authenticity of the platform, which is essential for zero-trust architectures and confidential computing scenarios.

Desired outcome: Implement hardware-based security capabilities through TPM technology that provides a secure foundation for cryptographic operations, secure boot processes, and enhanced protection of sensitive keys and credentials in Microsoft workloads on AWS.

Common anti-patterns:

- Deploying Windows workloads without enabling TPM capabilities, missing opportunities to use hardware-based security features that provide stronger protection than software-only solutions.
- Using software-based encryption and key storage without the additional security layer provided by hardware security modules, potentially exposing keys to compromise through software vulnerabilities.
- Failing to integrate TPM capabilities with Windows security features, limiting the effectiveness of built-in security technologies that depend on hardware-based trust anchors.

Benefits of establishing this best practice:

- Enhanced security through hardware-based cryptographic operations that provide stronger protection for encryption keys and sensitive data compared to software-only solutions.
- Improved compliance capabilities through hardware-based attestation and secure boot features that help meet regulatory requirements for high-security environments.
- Strengthened Windows security features through TPM integration that enables advanced capabilities like Credential Guard, Device Guard, and Windows Hello for Business authentication.

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

Implementation guidance

Implementing TPM technology for Microsoft workloads requires enabling NitroTPM during EC2 instance launch and configuring Windows security features to use the hardware-based capabilities. Focus on integrating TPM with existing security controls and Windows features to maximize security benefits.

Implementation steps

1. Enable NitroTPM when launching new EC2 instances running Windows Server 2016 or later, verifying that the instance type supports TPM functionality and is built on the AWS Nitro System.
2. Deploy EC2 Instances with NitroTPM Using AWS CloudFormation.
3. Integrate AWS KMS with NitroTPM for Enhanced Key Management for runtime attestation and system integrity protection against firmware and kernel-level attacks.
4. Configure AWS Systems Manager for TPM-enabled Microsoft workloads to secure credential storage and certificate-based authentication capabilities.
5. Implement Credential Guard to protect domain credentials and other sensitive authentication information using TPM-based virtualization security.
6. Set up secure boot functionality that uses TPM to verify the integrity of the boot process and block unauthorized code execution during startup.
7. Configure monitoring and logging using AWS CloudTrail, Amazon CloudWatch, and Windows event logs to track TPM usage, key operations, and security events related to hardware-based security features.
8. Establish procedures for TPM endorsement key management and disaster recovery, understanding that NitroTPM keys are instance-specific and require proper backup strategies for encrypted data.
9. Implement AWS IAM Roles Anywhere for certificate-based authentication.

Resources

Related documents:

- Amazon EC2 now supports NitroTPM and UEFI Secure Boot
- Credential Guard for Windows instances
- Windows TPM 2.0 Overview

Related tools:

- IAM Roles Anywhere Credential Helper - GitHub Repository
- IAM Roles Anywhere Credential Helper - Configuration Examples
- BitLocker Drive Encryption
- Windows Defender System Guard

Reliability

The reliability pillar for Microsoft workloads on AWS focuses on designing resilient systems, implementing robust monitoring, conducting post-incident analysis, managing capacity through auto scaling, and automating recovery procedures. It emphasizes defining clear availability goals, choosing appropriate architectural patterns, and regularly testing resilience. The pillar guides organizations in avoiding failures, responding effectively to disruptions, and continuously improving their systems to maintain high availability and efficient operation of Microsoft workloads in the AWS environment.

Focus areas

- Design principles
- Workload architecture and availability
- Monitoring and incident response
- Automation and recovery management

Design principles

- **Use Microsoft high availability patterns:** Implement SQL Server Always On Availability Groups across multiple Availability Zones, deploy Active Directory domain controllers with proper site topology, and use Exchange Database Availability Groups to maintain Microsoft workload resilience while benefiting from AWS infrastructure reliability.
- **Integrate with AWS managed services for Microsoft technologies:** Use Amazon RDS Multi-AZ for SQL Server, AWS Managed Microsoft AD for directory services, and Amazon FSx for Windows File Server to reduce operational overhead while maintaining Microsoft workload compatibility and using AWS managed reliability features.
- **Design for cross-AZ resilience with Microsoft-aware configurations:** Configure SQL Server listener endpoints across Availability Zones, establish Active Directory sites aligned with Availability Zones, implement SharePoint farm topology with cross-AZ redundancy, and verify that .NET applications handle Availability Zone failures gracefully through stateless design and proper load balancing.

For general reliability design principles that apply to all workloads, see Design principles for reliability.

Workload architecture and availability

Building resilient Microsoft workloads starts with clearly defined availability goals and appropriate architectural choices. This focus area verifies that your workload is designed to handle failures, protect critical data, and meet business requirements through proper infrastructure patterns and redundancy strategies.

MSFTREL01: How do you design your Microsoft workload to withstand failure?

To create a resilient Microsoft infrastructure on AWS, define business requirements and availability goals, assess failure scenarios, and determine recovery needs. Choose and evaluate an appropriate architecture pattern, then implement and test regularly. This approach provides for a robust, cost-effective solution that can adapt to evolving needs and new AWS capabilities.

MSFTREL02: What methods and systems do you have in place to identify and respond to disruptions affecting your Microsoft workload?

Design resilient Microsoft workloads through targeted solutions and monitoring systems that detect and stop failures. Assess infrastructure components, identify vulnerabilities, and implement protective measures. Regular testing and automation help maintain system health and adaptability to challenges.

Best practices

- MSFTREL01-BP01 Define availability goals for Microsoft workloads
- MSFTREL01-BP02 Align your architectural design with your availability needs and capacity demands
- MSFTREL01-BP03 Safeguard the continuous accessibility of essential data from your Microsoft workload
- MSFTREL02-BP01 Implement comprehensive monitoring for potential failures across the application, AWS infrastructure, and network connectivity
- MSFTREL02-BP02 Develop a strategic plan for quickly reinstating service availability during outages

MSFTREL01-BP01 Define availability goals for Microsoft workloads

Identifying your organization's specific availability objectives is crucial as it guides your focus towards critical factors. This clarity enables you to establish relevant criteria for assessing and selecting the most appropriate architectural patterns for your workload.

Desired outcome: Have clearly defined and documented availability requirements for Microsoft workloads on AWS that align with business needs, enabling informed architectural decisions and providing measurable performance targets that support the organization's operational goals.

Common anti-patterns:

- Applying the same availability requirements to each of your Microsoft workloads without considering their individual business impact or criticality, leading to overprovisioning for less critical services or underestimating the needs of mission-critical applications.
- Defining initial availability goals but failing to review and adjust them as business needs evolve, resulting in outdated architectural decisions that no longer align with current organizational priorities or growth.

Benefits of establishing this best practice:

- Clear availability requirements avoid over-engineering or under-engineering solutions, allocating resources efficiently based on actual business needs rather than assumptions and leading to better cost control and ROI.
- Well-defined availability goals enable precise architectural planning and implementation of appropriate resilience measures, resulting in fewer service disruptions and better alignment between IT capabilities and business expectations.

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

Implementation guidance

Begin with stakeholder workshops to identify and document availability requirements for each Microsoft workload. Create a standardized template that captures key metrics (like RTO, RPO, and SLAs) and business impact levels, then prioritize workloads based on criticality.

Work with business units to validate these requirements, verifying that they reflect actual operational needs rather than technical assumptions.

Document the approved requirements in a central repository, establish a regular review cycle (for example, quarterly), and use these specifications to guide architectural decisions in AWS, including the selection of appropriate Availability Zones, backup strategies, and failover mechanisms.

Implementation steps

1. Conduct stakeholder workshops to gather and document availability requirements for each Microsoft workload, using a standardized template to capture RTO, RPO, and SLA targets.
2. Create a prioritized matrix of workloads based on business criticality and required availability levels, validated by business unit leaders.
3. Establish a central documentation repository for storing and managing availability requirements, with clear version control and change management processes.
4. Develop architectural blueprints that map the documented availability requirements to specific AWS services and design patterns.
5. Implement a quarterly review cycle to reassess and update availability requirements, continually aligning with business needs and AWS best practices.

Resources

Related documents:

- Understanding availability need

MSFTREL01-BP02 Align your architectural design with your availability needs and capacity demands

There are architecture recommendations regarding Microsoft workloads, whether addressing Windows infrastructure, Active Directory, SQL Server databases, .NET applications, or other technologies. Review the vendor recommendations along with AWS documentation to verify that your application is running with best practices to improve availability and resiliency.

Desired outcome: A Microsoft workload optimized for high availability and scalability and aligned with vendor recommendations and AWS best practices. This strategy verifies that the system meets availability targets and capacity demands efficiently, while remaining resilient and maintainable.

Common anti-patterns:

- Directly migrating Microsoft workloads to AWS without redesigning for cloud capabilities, resulting in missed opportunities for improved availability and automated scaling.
- Ignoring Microsoft's recommended high-availability configurations (like Always On Availability Groups for SQL Server) in favor of simplified single-instance deployments.
- Treating AWS Regions as simple datacenter replacements rather than using multi-AZ and Regional resilience patterns specific to AWS infrastructure.

Benefits of establishing this best practice:

- Increased system reliability through proven architectural patterns, reducing downtime and improving customer satisfaction while lowering operational overhead.
- Optimized cost-efficiency by properly sizing resources and utilizing AWS's elastic scaling capabilities, avoiding over-provisioning while maintaining performance.
- Enhanced disaster recovery capabilities through AWS-specific resilience features combined with Microsoft's high-availability solutions, maintaining business continuity.

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

Implementation guidance

Document your current Microsoft workload architecture and availability needs, and consult AWS and Microsoft best practices for your specific technologies.

Develop a phased migration plan incorporating these recommendations, prioritizing critical components.

Use AWS managed services where possible and implement multi-AZ deployments.

Regularly test resilience and conduct Well-Architected reviews to align with best practices.

Implementation steps

1. Evaluate SQL Server Always On Availability Groups configuration, Active Directory domain controller placement, Exchange Database Availability Groups, SharePoint farm topology, and .NET application dependencies. Document current RTO/RPO requirements and identify single points of failure in your Microsoft workload stack.
2. Implement SQL Server Always On across multiple Availability Zones using Amazon EC2 or Amazon RDS Multi-AZ, deploy Active Directory domain controllers in separate Availability Zones

with AWS Managed Microsoft AD integration, establish Exchange DAG with cross-AZ mailbox databases, and design .NET applications for stateless operation with Application Load Balancer distribution. For more detail, see Microsoft workload architecture patterns.

3. Replace self-managed components with Amazon RDS for SQL Server, AWS Managed Microsoft AD, Amazon FSx for Windows File Server, and AWS Systems Manager for Windows patch management. This reduces operational overhead while improving availability through AWS-managed infrastructure resilience.
4. Configure SQL Server Always On listener endpoints across Availability Zones, establish Active Directory site topology aligned with Availability Zones, implement Exchange transport high availability, and verify that .NET applications handle Availability Zone failures gracefully. Reference Microsoft SQL Server on AWS best practices.
5. Test SQL Server failover scenarios, Active Directory replication across Availability Zones, Exchange mailbox database switchovers, and .NET application resilience to infrastructure failures. Establish automated testing procedures that validate both AWS infrastructure and Microsoft application layer availability.

Resources

Related documents:

- Design your workload service architecture

MSFTREL01-BP03 Safeguard the continuous accessibility of essential data from your Microsoft workload

Microsoft workloads encompass diverse technologies including SQL Server databases, Active Directory domain controllers, Exchange Server mailbox databases, SharePoint content databases, IIS web applications, and Windows file servers, each requiring specialized backup and recovery approaches. A comprehensive data accessibility strategy must address the unique backup requirements of each Microsoft technology while using AWS services to maintain business continuity and data protection.

Desired outcome: The organization maintains comprehensive data accessibility by implementing a robust strategy that maintains continuous access to Microsoft SQL Server databases, Active Directory System State and SYSVOL, Exchange mailbox databases and transaction logs, SharePoint content databases and search indexes, IIS configurations and application pools, and Windows system components including certificates and registry settings. This strategy aligns with AWS

best practices for Microsoft workload management, enabling reliable data recovery and business continuity while using AWS infrastructure and services.

Common anti-patterns:

- Focusing solely on SQL Server databases while ignoring critical Active Directory System State backups, Exchange transaction logs, SharePoint service applications, and IIS configuration files.
- Implementing generic backup strategies without considering Microsoft-specific requirements such as VSS integration, application-consistent snapshots, and service dependencies.
- Migrating to AWS without adapting backup strategies to utilize AWS-native tools for comprehensive Microsoft workload protection including AWS Backup, FSx for Windows File Server, and application-aware backup solutions.

Benefits of establishing this best practice:

- Verifies complete Microsoft workload restoration by protecting databases, system configurations, application states, and supporting components across Microsoft technologies, minimizing business disruption.
- Using AWS-specific tools and services for Microsoft workload management reduces operational costs, improves resource efficiency, and provides centralized backup management across diverse Microsoft technologies.
- Maintains consistent access to critical data while enabling quick recovery of complex Microsoft environments, supporting uninterrupted business operations and improved adherence across Microsoft services.

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

Implementation guidance

Develop a comprehensive inventory of Microsoft workload components, including: - SQL Server databases and transaction logs - Active Directory System State and SYSVOL folders - Exchange mailbox databases and transport queues - SharePoint content databases and service applications - IIS application pools and web configurations, Windows certificates and registry settings - FSx for Windows File Server shares

Implement AWS Backup with VSS integration for application-consistent snapshots, configure Amazon RDS for SQL Server automated backups, establish AWS Managed Microsoft AD backup procedures, and use AWS Systems Manager for Windows system configuration backup.

Regularly test recovery procedures across your Microsoft technologies, and train staff on AWS best practices for comprehensive Microsoft workload management.

Implementation steps

1. Create a complete inventory of Microsoft workload components requiring continuous accessibility, including:
    1. SQL Server databases
    2. Active Directory System State and SYSVOL,
    3. Exchange mailbox databases,
    4. SharePoint content databases,
    5. IIS configurations,
    6. Windows system settings,
    7. and FSx for Windows File Server data
2. Configure AWS Backup with VSS integration for application-consistent snapshots of Microsoft workloads, implement Amazon RDS automated backups for managed SQL Server instances, establish AWS Managed Microsoft AD backup procedures, and set up FSx for Windows File Server backup policies.
3. Establish and document comprehensive recovery procedures including restoration order and dependencies between Microsoft services, Active Directory domain controller recovery, SQL Server Always On Availability Group restoration, Exchange Database Availability Group recovery, and SharePoint farm restoration procedures.
4. Schedule regular recovery testing and validation exercises across your Microsoft technologies to check the effectiveness of the comprehensive data accessibility strategy, including cross-service dependency validation and disaster recovery scenario testing.

Resources

Related documents:

- Migrating Microsoft SQL Server databases to the AWS Cloud
- Optimize SQL Server backup strategies
- How to simplify Microsoft SQL Server backup using AWS Backup and VSS
- Automating SQL Server Point-in-Time Recovery Using EBS Snapshots

MSFTREL02-BP01 Implement comprehensive monitoring for potential failures across the application, AWS infrastructure, and network connectivity

Monitoring your Microsoft application, AWS resources, and network connectivity enables prompt responses to both actual and potential failures, enhancing overall system reliability.

Desired outcome: Comprehensive monitoring across the Microsoft application, AWS infrastructure, and network components will enable early detection of issues and prompt response to potential failures, optimizing system reliability and performance.

Common anti-patterns:

- Only responding to issues after they cause significant disruptions, rather than proactively monitoring for potential problems.
- Monitoring individual components in isolation without considering the interconnected nature of the application, infrastructure, and network.
- Focusing monitoring efforts solely on the application layer while neglecting AWS infrastructure and network connectivity monitoring.

Benefits of establishing this best practice:

- Comprehensive monitoring enables quick identification and resolution of issues, reducing downtime and enhancing overall system stability.
- Continuous monitoring provides valuable insights into system behavior, allowing for data-driven optimizations and resource allocation.
- Proactive monitoring reduces the time and resources spent on troubleshooting, allowing IT teams to focus on strategic initiatives rather than firefighting.

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

Implementation guidance

Begin by identifying critical monitoring metrics across each layer (application, infrastructure, and network) and implement a primary monitoring solution like Amazon CloudWatch. Set up custom metrics and dashboards for Microsoft application-specific monitoring, configure detailed AWS resource monitoring, and establish network connectivity checks.

Set up automated alerting and aggregate logs. Define clear thresholds and escalation procedures, and implement automated responses where appropriate.

Regularly review and refine your monitoring parameters to maintain the effectiveness of the monitoring strategy.

Implementation steps

1. Define and configure essential metrics and alarms for Microsoft workloads with thresholds appropriate to your specific environment and SLA requirements, including:
    1. SQL Server performance monitoring (CPU utilization, memory availability, deadlock detection, backup status)
    2. Active Directory health checks (authentication failures, replication status, SYSVOL synchronization)
    3. IIS or .NET application monitoring (application pool health, HTTP error rates, worker process status)
    4. Windows system alerts (disk space, memory utilization, critical service status)
    5. AWS infrastructure monitoring for underlying EC2, EBS, and network components
2. Create consolidated dashboards for unified visibility across your monitored components.
3. Set up topics and subscription endpoints for automated alerting based on predefined thresholds.
4. Implement centralized logging and configure log metric filters.
5. Establish automated remediation actions in response to specific alarm conditions.

Resources

Related documents:

- Monitor workload resources
- Designing and implementing logging and monitoring with Amazon CloudWatch

Related tools:

- Amazon CloudWatch
- Amazon Simple Notification Service
- AWS Lambda

MSFTREL02-BP02 Develop a strategic plan for quickly reinstating service availability during outages

Microsoft workloads require specialized disaster recovery strategies due to their unique architectural dependencies and state management requirements. SQL Server Always On Availability Groups, Active Directory domain controllers, Exchange Database Availability Groups, and Windows Failover Clusters each have specific recovery procedures that differ from generic cloud workload restoration, requiring tailored approaches for rapid service reinstatement.

Desired outcome: Implement a comprehensive service restoration strategy specifically designed for Microsoft workloads that includes automated recovery procedures for SQL Server Always On configurations, Active Directory hybrid scenarios, and Windows-based clustering services. This plan should use AWS Elastic Disaster Recovery (DRS) for block-level replication while addressing Microsoft-specific challenges such as cluster reformation, domain controller restoration, and application-consistent recovery.

Common anti-patterns:

- Treating Microsoft workloads as generic applications during DR planning, failing to account for Active Directory dependencies, SQL Server cluster requirements, or Windows-specific services that require specialized recovery procedures.
- Relying on AWS DRS replication alone without planning for Microsoft cluster reformation, resulting in standalone SQL Server instances instead of properly configured Always On Availability Groups at the DR site.

Benefits of establishing this best practice:

- Verifies that complex Microsoft services like SQL Server clusters and Active Directory maintain their intended architecture and functionality after disaster recovery events.
- Pre-planned procedures for reforming Windows Failover Clusters and reestablishing Always On Availability Groups minimize manual intervention and potential errors during critical recovery operations.
- Maintains integration between on-premises Active Directory and AWS-hosted domain controllers, preserving authentication services and trust relationships during outages.

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

Implementation guidance

Develop Microsoft workload-specific disaster recovery procedures that address SQL Server Always On Availability Groups restoration using AWS DRS, Active Directory domain controller recovery with proper site configuration, and Windows Failover Cluster reformation at DR sites.

Implement AWS Backup for Active Directory System State backups and configure cross-region replication for critical Microsoft services.

Establish automated procedures for reestablishing trust relationships, cluster quorum, and application dependencies specific to Microsoft environments.

Implementation steps

1. Configure AWS Elastic Disaster Recovery (DRS) for SQL Server Always On primary nodes with plans for cluster reformation and secondary replica establishment at the DR site.
2. Implement Active Directory disaster recovery using AWS DRS for domain controllers combined with AWS Backup for System State backups, providing a proper site and subnet configuration for hybrid scenarios.
3. Establish Windows Failover Cluster recovery procedures including shared storage configuration using Amazon FSx for Windows File Server and cluster quorum reestablishment.
4. Create automated runbooks for Microsoft-specific recovery tasks including SQL Server Always On listener reconfiguration, Active Directory trust relationship validation, and Exchange Database Availability Group restoration.
5. Configure cross-Region backup strategies for Microsoft workloads using AWS Backup with application-consistent snapshots and coordinate with AWS DRS replication schedules.
6. Implement regular testing procedures that validate Microsoft service dependencies, cluster functionality, and hybrid Active Directory connectivity after DR scenarios.

Resources

Related documents:

- Set up high availability for SQL Server at DR site using AWS Elastic Disaster Recovery
- Hybrid Active Directory disaster recovery solutions on AWS
- Choose a high availability and disaster recovery solution

Monitoring and incident response

Establishing comprehensive monitoring and response mechanisms enables quick detection and resolution of issues affecting your Microsoft workloads. This includes implementing robust monitoring systems, developing efficient restoration procedures, and maintaining a continuous improvement cycle through regular testing and post-incident analysis.

MSFTREL03: How do you address post-incident analysis and continuous improvement?

Post-incident analysis and continuous improvement are essential for managing Microsoft workloads on AWS. Organizations analyze incidents to extract insights and implement preventive measures, helping them avoid similar issues and strengthen their operational resilience.

Best practices

- MSFTREL03-BP01 Use Microsoft logs for incident analysis
- MSFTREL03-BP02 Establish a structured review process that combines insights from both AWS and Microsoft monitoring tools
- MSFTREL03-BP03 Implement automated feedback loops

MSFTREL03-BP01 Use Microsoft logs for incident analysis

Microsoft workloads generate rich diagnostic information through specialized logging systems that provide deeper visibility into application behavior, security events, and system performance than standard infrastructure monitoring alone. These Microsoft log sources offer unique insights into Active Directory authentication patterns, SQL Server performance bottlenecks, IIS application errors, and Windows system events that are essential for comprehensive incident analysis and root cause identification.

Desired outcome: The implementation of comprehensive incident analysis should result in a detailed incident timeline constructed from CloudWatch and diverse Microsoft log sources including categorized Windows Event Logs (System, Security, Application events), Active Directory authentication and directory service logs, SQL Server operational and audit logs, IIS web server logs, Exchange messaging logs, and SharePoint service logs, accompanied by thorough post-incident documentation. Infrastructure improvements, derived from root cause analysis, should be automated through infrastructure as code and systematically deployed through AWS Systems

Manager, providing for consistent security patch management and configuration updates across your Windows instances.

Common anti-patterns:

- Reactive manual patching and configuration changes without proper documentation or version control, leading to inconsistent Windows environments and untraceable security vulnerabilities.
- Neglecting to collect or analyze Windows Event Logs (or general logs) during incidents, resulting in incomplete incident understanding and recurring issues due to unidentified root causes.
- Implementing infrastructure changes directly through the AWS Management Console instead of infrastructure as code, causing configuration drift and making it impossible to reliably replicate or roll back environment changes.

Benefits of establishing this best practice:

- Systematic log analysis and documented post-mortems enable faster, more effective resolution of future incidents and reduce mean time to recovery (MTTR).
- Infrastructure as code (IaC) creates reproducible, version-controlled environment configurations, eliminating manual errors and configuration drift.
- Centralized patch management through AWS Systems Manager maintains security standards across Windows instances while reducing operational overhead.
- ßComprehensive documentation and standardized runbooks preserve institutional knowledge, enabling better team collaboration and reducing dependency on specific individuals.

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

Implementation guidance

Review Amazon CloudWatch and comprehensive Microsoft log sources to analyze incidents, including: - Windows event logs (system, security, application, setup, and forwarded events) - Active Directory logs (security events, directory service logs, DNS server logs) - SQL Server logs (error logs, agent logs, audit logs, transaction logs) - IIS or .NET logs (access logs, error logs, ASP.NET application logs) - Exchange Server logs (message tracking, protocol logs, connectivity logs) - SharePoint ULS (Unified Logging Service) logs.

Document findings in a post-incident report covering root cause, impact, and fixes. Update infrastructure code (using AWS CloudFormation or AWS CDK) with improvements and revise Windows configurations. Automate future security updates using AWS Systems Manager.

Begins with establishing a robust logging strategy using CloudWatch and Windows event logs. A standardized post-incident template should include sections for incident timeline, root cause analysis, impact assessment, and remediation steps.

Migrate infrastructure to infrastructure as code using AWS CloudFormation or AWS CDK, incorporating lessons learned from incidents. Configure AWS Systems Manager to automate regular security patching and configuration updates across Windows instances, maintaining consistent application of security policies. Runbooks require regular review and updates to reflect the latest best practices and incident response procedures.

Implementation steps

1. Configure comprehensive logging through CloudWatch and Windows Event Logs, establishing appropriate retention periods and log analysis workflows.
2. Create standardized templates for incident documentation, including post-incident analysis, root cause identification, and remediation tracking.
3. Develop infrastructure as code templates using CloudFormation or CDK to manage and version control your infrastructure components.
4. Set up AWS Systems Manager for automated patch management and configuration updates across Windows instances.
5. Establish regular review cycles for runbooks and documentation to maintain accuracy and incorporate new learnings from incidents.

Resources

Related documents:

- Run an automated operation powered by Systems Manager Automation
- Quick Start: Enable your Amazon EC2 instances running Windows Server 2016 to send logs to CloudWatch Logs using the CloudWatch Logs agent
- Monitoring Windows services with Amazon CloudWatch

Related tools:

- AWS Systems Manager
- Amazon CloudWatch

MSFTREL03-BP02 Establish a structured review process that combines insights from both AWS and Microsoft monitoring tools

Document lessons learned in a centralized knowledge base, update incident response playbooks, and conduct regular tabletop exercises to validate improvements. Configure automated reporting to track incident metrics and measure the effectiveness of implemented changes. Regular reviews of Windows security baselines and AWS Well-Architected Framework improve your alignment with best practices.

Desired outcome: The integration of AWS Security Hub CSPM and Microsoft monitoring tools should provide unified visibility while maintaining documented processes and automated reporting to maintain continuous security improvement and adherence to established standards.

Common anti-patterns:

- Siloed monitoring approaches where AWS and Microsoft tools are used independently, creating blind spots and delayed incident response.
- Unorganized documentation of incidents and lessons learned without a structured knowledge base, leading to repeated issues and inconsistent security practices.

Benefits of establishing this best practice:

- Enhanced visibility across hybrid environments, enabling faster threat detection and response.
- Improved operational efficiency through centralized knowledge management and automated reporting.
- Consistent alignment with industry best practices, reducing security risks and regulatory gaps.

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

Implementation guidance

Integrate AWS Security Hub CSPM with existing Microsoft monitoring tools through APIs and automated workflows. Define standardized documentation templates and establish a regular review cadence for updating the knowledge base.

Configure automated reporting dashboards that combine metrics from both platforms, and schedule quarterly tabletop exercises to validate the integrated monitoring approach.

Implementation steps

1. Configure AWS Security Hub CSPM and enable integration with Microsoft monitoring tools through APIs.
2. Establish a centralized knowledge base system for documenting incidents and lessons learned.
3. Set up automated reporting workflows to track cross-platform security metrics.
4. Create standardized templates for incident documentation and response procedures.
5. Implement regular review cycles for security baselines and Well-Architected alignment.
6. Schedule quarterly tabletop exercises to validate monitoring effectiveness and response procedures.

Resources

Related documents:

- Understanding security standards in Security Hub CSPM CSPM
- Govern Microsoft workloads using the myApplications dashboard on AWS

MSFTREL03-BP03 Implement automated feedback loops

Automated feedback loops are critical for Microsoft workloads due to their complex interdependencies and the need for continuous monitoring of performance patterns, security posture, and operational effectiveness. Microsoft environments require systematic approaches to identify trends, measure improvement initiatives, and translate monitoring investments into actionable insights for maintaining system reliability and security.

Desired outcome: The implementation of automated feedback loops establishes comprehensive monitoring and continuous improvement mechanisms through Amazon CloudWatch dashboards, automated vulnerability assessments, and metrics-driven compliance reporting, enabling data-driven decision-making and proactive risk management for Windows workloads on AWS.

Common anti-patterns:

- Relying solely on manual monitoring and reactive troubleshooting instead of implementing automated alerts and dashboards
- Conducting unplanned or inconsistent vulnerability assessments without a standardized schedule or automated scanning process

- Failing to maintain a prioritized improvement backlog, addressing issues randomly rather than based on quantifiable risk metrics and business impact

Benefits of establishing this best practice:

- Custom CloudWatch dashboards provide real-time insights into system performance and security patterns, enabling faster issue detection and resolution.
- Regular automated vulnerability assessments help identify and address potential threats before they can be exploited, improving overall security posture.
- A prioritized backlog based on risk and business impact allocates resources efficiently to address the most critical issues first.
- Automated compliance reporting using CloudWatch metrics and alarms streamlines the audit process and helps maintain ongoing regulatory adherence with less manual effort.

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

Implementation guidance

Implement automated feedback loops by creating custom CloudWatch dashboards to monitor patterns and resolution times. Set up regular vulnerability assessments for EC2 instances running Windows workloads, and maintain a continuous improvement backlog prioritized by risk and business impact. Use CloudWatch metrics and alarms to monitor the effectiveness of implemented controls and automate compliance reporting.

Begin by creating custom dashboards that display key metrics for Windows workloads. Next, establish automated vulnerability scanning for EC2 instances on a regular schedule. Develop a process for maintaining and prioritizing a continuous improvement backlog based on identified risks and business impact. Finally, configure CloudWatch metrics and alarms to monitor control effectiveness and automate compliance reporting, verifying that each component works together in a cohesive feedback loop.

Implementation steps

1. Set up specialized CloudWatch dashboards for Microsoft workloads including:
    1. SQL Server metrics (connection pools, deadlocks, wait statistics, backup status)
    2. IIS or .NET performance (request queues, application pool health, session state)
    3. Active Directory monitoring (authentication failures, replication status, LDAP queries)
    4. Windows system performance (memory usage, disk I/O, Event Logs)

5. Exchange or SharePoint service health dashboards.

2. Focus on key Windows Performance Counters to drive continual improvement, such as:
    1. \Memory\Available MBytes
    2. \Processor(_Total)\% Processor Time
    3. \PhysicalDisk(_Total)\Avg. Disk Queue Length
    4. Critical Windows Event Log patterns (Security Event ID 4625 for failed logons, System Event ID 6008 for unexpected shutdowns)
    5. Application-specific metrics like \SQLServer:General Statistics\User Connections and \Web Service(_Total)\Current Connections
3. Enable Amazon Inspector for automated vulnerability assessments on Windows EC2 instances, which automatically discovers supported Windows instances and performs continuous scanning every 6 hours by default. Configure custom scan schedules using SSM associations and use Inspector's integration with Systems Manager to assess operating system vulnerabilities, software packages, and network reachability for comprehensive security monitoring.
4. Create and maintain a centralized improvement backlog system with clear risk scoring and business impact criteria.
5. Implement alarms with appropriate thresholds for identified key performance and security metrics.
6. Establish automated compliance reporting workflows using metrics and AWS reporting tools.
7. Develop documentation and runbooks for responding to automated alerts and managing the continuous improvement process.

Resources

Related documents:

- What is Amazon CloudWatch?

Automation and recovery management

Implementing automated solutions for recovery, scaling, and maintenance reduces human error and improves response times. This encompasses automated disaster recovery, systematic backup procedures, self-healing mechanisms, and dynamic capacity management through auto scaling, maintaining consistent and reliable operation of Microsoft workloads on AWS.

MSFTREL04: How do you address capacity management and auto scaling?

Capacity management and auto scaling for Microsoft workloads on AWS optimize performance and costs by automatically adjusting resources based on demand, while considering Microsoft-specific requirements and AWS capabilities.

MSFTREL05: How do you implement recovery automation for your Microsoft workload?

Recovery automation for Microsoft workloads on AWS combines AWS capabilities with Microsoft-specific tools to maintain business continuity. This practice focuses on automated approaches to protect and recover Microsoft-based applications and infrastructure on AWS, minimizing downtime and human intervention while maintaining consistent recovery procedures for Windows Server, SQL Server, and .NET applications.

Best practices

- MSFTREL04-BP01 Use Amazon EC2 Auto Scaling in combination with Application Auto Scaling
- MSFTREL05-BP01 Implement disaster recovery automation
- MSFTREL05-BP02 Implement backup automation
- MSFTREL05-BP03 Implement self-healing procedures
- MSFTREL05-BP04 Implement testing automation

MSFTREL04-BP01 Use Amazon EC2 Auto Scaling in combination with Application Auto Scaling

Microsoft workloads often experience variable demand patterns that require dynamic resource scaling to maintain performance and cost efficiency. Applications like SharePoint farms, SQL Server databases, and .NET web applications face fluctuating user loads, batch processing requirements, and seasonal traffic variations that make static resource provisioning ineffective and costly.

Desired outcome: Implement automated scaling mechanisms that combine instance-level and application-level auto scaling capabilities to optimize resource utilization. Configure predictive scaling using historical data and machine learning to anticipate capacity needs, while maintaining

regulatory requirements through dedicated host allocation. Automate capacity management through system tooling that responds to monitoring metrics and scheduled events.

Common anti-patterns:

- Relying on human intervention to adjust resources in response to demand changes, leading to slow reactions and potential over- or under-provisioning.
- Maintaining a fixed resource capacity based on peak demand expectations, resulting in wasted resources during low-demand periods and potential shortages during unexpected spikes.
- Focusing solely on infrastructure scaling without considering the need for application-level scaling, potentially leading to bottlenecks in database connections, caching, or other application components.

Benefits of establishing this best practice:

- Dynamically adjusts resource capacity to actual demand, reducing waste from over-provisioning while reducing performance issues from under-provisioning.
- Automatically responds to changes in workload patterns and system failures, maintaining service availability without manual intervention.
- Uses historical data and machine learning to anticipate and prepare for demand spikes before they occur, reducing reactive scaling delays.
- Maintains licensing and regulatory requirements through controlled scaling of dedicated resources while still achieving operational efficiency.
- Reduces administrative overhead through automated capacity management, freeing up teams to focus on higher-value activities while providing consistent performance.

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

Implementation guidance

Configure auto scaling groups to dynamically adjust compute resources based on demand patterns. Enable predictive scaling using historical metrics to proactively manage capacity before peak loads occur. For applications with specific licensing requirements, use dedicated hosts with auto scaling policies to maintain adherence. Implement automation tools to handle routine capacity management tasks, using monitoring alerts and scheduled actions to control resource scaling.

Begin by establishing baseline metrics and performance thresholds for both infrastructure and application components. Configure scaling policies that combine predictive and dynamic

adjustments, using historical data to anticipate capacity needs while maintaining real-time responsiveness to unexpected changes. Set up monitoring and alerting to track scaling activities, and implement gradual scaling steps to avoid resource thrashing. Document scaling decisions and regularly review performance data to refine thresholds and policies, providing optimal resource utilization while maintaining service levels and regulatory requirements.

Implementation steps

1. Configure EC2 Auto Scaling groups with appropriate launch templates and scaling policies for Windows instances, including SQL Server and IIS workloads.
2. Enable predictive scaling using AWS Auto Scaling with machine learning forecasts, incorporating Microsoft workload patterns and licensing considerations for dedicated hosts.
3. Set up Amazon CloudWatch Application Insights to monitor .NET and SQL Server applications, automatically detecting anomalies and providing application-level metrics for scaling decisions across IIS, SQL Server, and application servers.
4. Create CloudWatch alarms and custom metrics for Microsoft applications including SQL Server connection counts, IIS request queues, and .NET application performance counters to trigger scaling actions, using Application Insights automated dashboards.
5. Implement application-level scaling automation for SQL Server Always On Availability Groups, IIS worker process scaling, and SharePoint farm expansion using Systems Manager automation documents triggered by Application Insights events.
6. Configure session state management and connection pooling strategies to support horizontal scaling of .NET applications and SQL Server workloads across multiple instances, using Application Insights correlation data for optimization.

Resources

Related documents:

- Predictive scaling for Amazon EC2 Auto Scaling
- What is Amazon EC2 Auto Scaling?
- Accelerate Amazon EC2 Auto Scaling for Microsoft Windows workloads
- Automatically scale your Amazon ECS service

Related tools:

- AWS Auto Scaling

- Amazon CloudWatch
- Auto Scaling launch templates
- Elastic Load Balancing

MSFTREL05-BP01 Implement disaster recovery automation

Disaster recovery (DR) automation is essential for Microsoft workloads due to their complex interdependencies and stateful nature. Microsoft environments often involve intricate relationships between Active Directory, SQL Server databases, file services, and application servers that require coordinated recovery procedures to maintain business continuity and data integrity.

Desired outcome: An effective disaster recovery implementation should automate the recovery of Microsoft workloads using appropriate tools, providing for consistent configuration restoration and automated traffic failover while meeting defined RTO and RPO objectives.

Common anti-patterns:

- Manual DR procedures and runbooks that rely on human intervention during critical recovery events, leading to extended downtimes and potential configuration errors during restoration.
- Maintaining inconsistent Windows Server and SQL Server configurations between primary and DR environments, resulting in failed recoveries or application compatibility issues post-failover.
- Using single-region deployments for critical Microsoft workloads without automated DNS failover mechanisms, creating a single point of failure and complicating the recovery process during regional outages.

Benefits of establishing this best practice:

- Reduced recovery time objective (RTO) through automated DR procedures, minimizing business disruption during outages.
- Improved consistency and reliability in recovering Microsoft workloads, eliminating human errors associated with manual recovery processes.
- Enhanced scalability and flexibility in managing DR across multiple regions, allowing for easier testing and updates to recovery plans.
- Cost optimization by using AWS services for DR, reducing the need for dedicated standby infrastructure and manual management overhead.

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

Implementation guidance

Implement automated disaster recovery for Microsoft workloads using Infrastructure as Code templates and automation runbooks. Use tools like AWS CloudFormation for infrastructure provisioning, and configuration management solutions to maintain consistent Windows Server and SQL Server setups.

Develop runbooks to automate recovery steps, including instance launch and configuration restoration. Configure DNS failover and load balancers for automatic traffic redistribution during DR events.

Regularly test failover procedures and store code and configurations in version control, providing robust security across environments.

Implement disaster recovery automation by creating infrastructure-as-code templates and automated runbooks for orchestrating recovery procedures. Integrate DNS failover and load balancing for traffic management. Develop scripts to maintain consistent configurations across environments.

Regularly test and refine DR processes, and provide documentation and version control to support ongoing improvements.

Implementation steps

1. Define DR requirements including RTO and RPO objectives, and create CloudFormation templates to codify Microsoft workload infrastructure, including AWS Managed Microsoft AD multi-region replication, SQL Server Always On Availability Groups, and FSx for Windows File Server cross-region backup strategies.
2. Develop Systems Manager Automation runbooks for orchestrating recovery procedures, including Windows instance restoration using AWS DRS (supporting Windows Server 2008 R2 through 2022), SQL Server Always On failover automation, and FSx file system recovery with DataSync for cross-region data replication.
3. Configure Amazon Route 53 DNS failover policies and Application Load Balancers for automated traffic routing between primary and DR regions, and integrate with Windows-based authentication systems and SQL Server connection string updates during failover events.
4. Create and test automated scripts for maintaining Windows Server and SQL Server configurations across environments, including Always On Distributed Availability Groups for cross-Region SQL replication and AWS Managed Microsoft AD trust relationships between Regions.

5. Implement monitoring and alerting to trigger automated DR processes when failure conditions are detected, including SQL Server Always On dashboard monitoring, FSx performance metrics, and AWS Managed Microsoft AD health checks across regions.
6. Establish regular DR testing schedule and documentation procedures to validate and improve recovery automation, including SQL Server Always On failover testing, FSx restore validation, and Microsoft workload recovery scenarios.

Resources

Related documents:

- Disaster recovery options in the cloud
- Choose a high availability and disaster recovery solution
- On-premises DR to AWS
- DR for cloud-native workloads
- Configuring DNS failover

Related tools:

- AWS CloudFormation
- AWS Elastic Disaster Recovery
- Amazon Route 53

MSFTREL05-BP02 Implement backup automation

Backup automation is critical for Microsoft workloads running on AWS, maintaining consistent data protection without manual intervention. Microsoft applications like SQL Server, Exchange, and SharePoint require application-aware backup strategies that maintain data integrity and support reliable recovery scenarios.

Desired outcome: Implement automated, application-consistent backups for Microsoft workloads on AWS to provide for reliable data protection, reduce manual effort, and enhance disaster recovery capabilities, ultimately improving system resilience and recoverability.

Common anti-patterns:

- Relying solely on manual, unplanned backups of Windows instances and databases, leading to inconsistent backup schedules, missed backups, and potential data loss during critical application states.
- Using basic snapshot mechanisms without VSS integration, resulting in crash-inconsistent backups that may not properly capture the state of running applications and could cause data corruption during restoration.

Benefits of establishing this best practice:

- Application-consistent backups captures data accurately, including in-memory and in-flight transactions, which reduces the risk of data corruption or loss.
- Automated backup processes reduce manual effort, minimize human errors, and free up IT staff to focus on more strategic tasks.
- Regular, automated backups with cross-region replication provide a robust foundation for quick and reliable recovery in case of system failures or disasters, minimizing downtime and data loss.

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

Implementation guidance

Configure automated backups for servers, databases, and storage volumes with defined schedules and retention policies. Enable application-consistent snapshots for data integrity, implement backup verification, and set up cross-Region copies for resilience. This strategy provides for comprehensive protection while maintaining application and file system consistency.

Configure AWS Backup with automated plans targeting Windows workloads and setting appropriate retention policies. Start with critical systems, establish backup windows during off-peak hours, and implement cross-region replication for essential workloads. Test backup integrity and recovery procedures regularly, and monitor backup success rates through AWS Backup reports.

Implementation steps

1. Define backup requirements and policies for SQL Server databases (EC2 and RDS), Active Directory domain controllers, Exchange Server mailbox databases, FSx for Windows File Server, SharePoint farms, and Windows file server volumes, considering RPO/RTO requirements for each service.
2. Configure AWS Backup plans with VSS-aware schedules for SQL Server transaction log backups, AD System State backups, Exchange VSS backups, FSx automatic backups, and SharePoint

farm-level backups with appropriate retention rules. For advanced Microsoft workload backup scenarios, consider AWS Partner solutions available in AWS Marketplace.
3. Set up cross-Region backup replication for critical SQL Server databases, AD domain controllers, Exchange databases, and FSx file systems to maintain disaster recovery capabilities. Additionally, configure cross-region replication for customer-managed S3 buckets containing backup data to enhance resilience.
4. Implement automated backup integrity checks using SQL Server CHECKDB, AD database verification, Exchange database consistency checks, FSx backup validation, and SharePoint content database consistency checks.
5. Establish and test recovery procedures for SQL Server point-in-time recovery, AD authoritative restore, Exchange mailbox recovery, FSx file system restoration, SharePoint farm recovery, and Windows file server volume recovery scenarios.

Resources

Related documents:

- Create a backup plan
- Create Windows VSS backups
- Application-consistent backup for Windows application on Amazon EC2 with AWS Backup
- How to simplify Microsoft SQL Server backup using AWS Backup and VSS
- Automating SQL Server Point-in-Time Recovery Using EBS Snapshots

Related tools:

- AWS Backup
- Amazon EBS snapshots

MSFTREL05-BP03 Implement self-healing procedures

Establish automated remediation capabilities that can detect, diagnose, and resolve common issues in Microsoft workloads without human intervention. Self-healing procedures reduce mean time to recovery (MTTR) and minimize the impact of transient failures on business operations.

Desired outcome: Implement automated self-healing procedures to proactively detect and remediate common issues in Microsoft workloads, providing for minimal downtime through automated instance recovery, health-based reboots, and configuration management.

Common anti-patterns:

- Waiting for human operators to detect and respond to system failures during off-hours.
- Implementing reactive fixes without addressing root causes through automation.
- Creating complex automation that requires extensive maintenance and troubleshooting.
- Over-automating without proper testing, leading to cascading failures.

Benefits of establishing this best practice:

- Significantly reduced mean time to recovery (MTTR) for common failure scenarios.
- Improved availability during off-hours when human operators may not be immediately available.
- Consistent and predictable response to system issues, reducing human error.
- Enhanced operational efficiency by freeing teams to focus on strategic initiatives rather than routine maintenance.
- Improved adherence to service level agreements (SLAs) through automated response capabilities.

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

Implementation guidance

When implementing self-healing procedures for Microsoft workloads, consider the balance between automation sophistication and operational complexity. Start with well-understood, low-risk scenarios before expanding to more complex remediation actions.

Key considerations for customers

- **Scope and prioritization:** Begin by identifying the most common and impactful failure scenarios in your Microsoft workloads. Focus on issues that occur frequently, have clear remediation steps, and pose minimal risk when automated. Examples include service restarts, disk space cleanup, and basic connectivity issues.
- **Testing and validation:** Thoroughly test automated remediation actions in non-production environments. Establish clear success criteria and rollback procedures. Consider implementing gradual rollouts and canary deployments for automation changes.
- **Monitoring and alerting strategy:** Design monitoring that can distinguish between symptoms and root causes. Avoid creating automation that treats symptoms without addressing underlying issues, as this can mask systemic problems.

- **Impact scope control:** Implement safeguards to avoid automated actions from causing widespread impact. Use circuit breakers, rate limiting, and approval workflows for high-risk remediation actions.
- **Documentation and knowledge transfer:** Maintain clear documentation of automated procedures, including trigger conditions, actions taken, and escalation paths. Verify that team members understand when and how automation will intervene.

Implementation steps

1. Analyze historical incidents to identify the most common and impactful issues. Prioritize scenarios with clear remediation steps and low automation risk.
2. Configure Amazon CloudWatch alarms and custom metrics to detect failure conditions. Verify that monitoring can differentiate between transient issues and persistent problems.
3. Create AWS Systems Manager Automation documents for each remediation scenario. Test thoroughly in non-production environments with various failure conditions.
4. Start with simple, low-risk actions like service restarts. Gradually expand to more complex scenarios like instance replacement or database failover as confidence builds.
5. Configure specific Microsoft workload automation:
   - Deploy auto-recovery for EC2 instances running Windows Server.
   - Set up automated instance reboots based on health checks for IIS and other Windows services.
   - Configure automatic failover for SQL Server Always On Availability Groups.
   - Implement automated patch management through AWS Systems Manager Patch Manager.
   - Use State Manager for configuration drift correction on Windows systems.
   - Active Directory and DNS automation:
     - Automatically restart Active Directory Domain Services when authentication failures exceed thresholds.
     - Reset DNS service when name resolution failures are detected.
     - Trigger domain controller health checks and automatic promotion of backup DCs.
   - IIS and web application remediation:
     - Restart application pools when memory usage exceeds defined limits.
     - Clear IIS logs when disk space is low.
     - Reset worker processes experiencing high CPU utilization.
     - Automatically recycle application pools based on request failure rates.
   - SQL Server specific automation:
     - Restart SQL Server services when connection timeouts increase.
     - Automatically shrink transaction logs when they exceed size thresholds.

    - Trigger index maintenance when fragmentation levels are high.
    - Reset SQL Server Agent jobs that fail due to transient issues.
  - Windows service and process management:
    - Restart Windows services that have stopped unexpectedly.
    - Kill and restart hung processes based on CPU or memory thresholds.
    - Clear Windows event logs when they reach capacity limits.
    - Reset network adapters when connectivity issues are detected.
  - File system and storage remediation:
    - Automatically clean temporary files when disk space is low.
    - Compress old log files to free up storage space.
    - Move archived data to lower-cost storage tiers.
    - Reset file permissions when access issues are detected.
  - Performance and resource optimization:
    - Automatically scale EC2 instances based on performance metrics.
    - Clear memory caches when system performance degrades.
    - Restart services consuming excessive resources.
    - Trigger garbage collection for .NET applications experiencing memory leaks.
6. Monitor automation effectiveness and adjust thresholds based on real-world performance. Implement logging and metrics to track automation success rates.

Resources

Related documents:

- AWS Systems Manager Automation
- Working with SSM Agent on EC2 instances for Windows Server
- Patching applications released by Microsoft on Windows Server
- Use AWS Systems Manager to enable CloudWatch memory metrics for Windows Server Amazon EC2 instances

Related tools:

- AWS Systems Manager
- Amazon CloudWatch

MSFTREL05-BP04 Implement testing automation

Establish comprehensive automated testing frameworks that continuously validate the reliability and recoverability of Microsoft workloads on AWS. Testing automation verifies that disaster recovery (DR) procedures, backup systems, and failover mechanisms work as expected when needed, reducing the risk of surprises during actual incidents.

Desired outcome: Implementing automated testing for disaster recovery processes maintains continuous validation of recovery mechanisms, backup integrity, and failover procedures. This approach enables regular verification of system resilience, enhancing overall reliability and minimizing manual intervention during critical recovery operations.

Common anti-patterns:

- Testing only during scheduled maintenance windows or annual DR exercises, missing critical issues that develop over time.
- Focusing solely on infrastructure testing while ignoring application-level validation and data integrity checks.
- Creating tests that don't reflect real-world failure scenarios or business-critical workflows.
- Implementing testing automation without proper validation of Microsoft-specific dependencies and licensing requirements.

Benefits of establishing this best practice:

- Continuous validation of Microsoft workload recovery capabilities, maintaining business continuity when failures occur.
- Early detection of configuration drift, licensing issues, and dependency problems that could impact recovery.
- Reduced recovery time objectives (RTO) through proven, tested procedures that work reliably under pressure.
- Enhanced confidence in disaster recovery capabilities, enabling better business decision-making around risk tolerance.
- Compliance validation for Microsoft licensing and regulatory requirements during recovery scenarios.

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

Implementation guidance

Microsoft workloads on AWS require specialized testing approaches that account for Windows-specific dependencies, Active Directory integration, SQL Server clustering, and Microsoft licensing considerations. Effective testing automation must validate not just infrastructure recovery, but also application functionality, data consistency, and regulatory requirements.

Key considerations for Microsoft workload testing

- **Windows-specific validation requirements:** Microsoft workloads have unique dependencies that must be tested, including Windows services startup order, registry configurations, Windows authentication, and domain trust relationships. Testing must verify that these components function correctly after recovery operations.
- **Active Directory and authentication testing:** Automated tests should validate domain controller functionality, DNS resolution, Kerberos authentication, and group policy application. This includes testing domain trust relationships, LDAP connectivity, and certificate services recovery.
- **SQL Server and database validation:** Database recovery testing must go beyond simple connectivity checks to include transaction log integrity, Always On Availability Group functionality, SQL Server Agent jobs, and linked server connections. Validate backup chain integrity and point-in-time recovery capabilities.
- **Microsoft licensing compliance:** Automated testing should verify that recovered systems maintain proper licensing adherence, including Windows Server activation, SQL Server licensing validation, and CAL (Client Access License) requirements.
- **Application-specific testing:** Test Microsoft applications like SharePoint, Exchange, or custom .NET applications to verify that they function correctly after recovery, including service dependencies, configuration settings, and data access patterns.

Implementation steps

1. Catalog your Windows services, Active Directory dependencies, SQL Server components, and Microsoft applications. Identify critical recovery scenarios specific to your Microsoft environment.
2. Create test cases that validate Windows-specific functionality, including:
   - Domain controller recovery and DNS functionality.
   - SQL Server Always On Availability Group failover.
   - Windows service startup and dependency chains.
   - Active Directory replication and authentication.
   - Microsoft application functionality and data access.

3. Implement automated testing frameworks:
   - Use AWS Systems Manager State Manager with PowerShell DSC to validate Windows configurations post-recovery.
   - Create automated scripts to validate database integrity, backup chain validation, and Always On cluster health.
   - Implement automated domain controller health validation, replication monitoring, and authentication testing.
   - Automate testing of critical Windows services, IIS application pools, and .NET application functionality.
4. Configure AWS testing tools:
   - Use AWS Systems Manager Automation with Windows-specific runbooks.
   - Use AWS Fault Injection Service to test Windows instance failures, network partitions, and storage issues.
   - Implement Amazon CloudWatch custom metrics for Microsoft-specific monitoring during tests.
5. Establish Microsoft workload-specific validation:
   - Automate checks for Windows activation status and SQL Server licensing compliance.
   - Validate that recovered systems meet performance expectations for Microsoft workloads.
   - Verify Windows security policies, firewall rules, and certificate validity post-recovery.
   - Implement automated database consistency checks and application data validation.
6. Create comprehensive reporting and alerting:
   - Generate detailed reports on Microsoft workload recovery test results.
   - Implement alerting for failed tests specific to Windows and SQL Server components.
   - Track compliance metrics for Microsoft licensing and configuration standards.

Resources

Related documents:

- AWS Systems Manager Automation
- What is AWS Fault Injection Service?
- Working with SSM Agent on EC2 instances for Windows Server
- PowerShell DSC and AWS Systems Manager State Manager
- SQL Server on Amazon EC2

Related tools:

- AWS Systems Manager
- AWS Fault Injection Service
- Amazon CloudWatch
- AWS Config

Performance efficiency

The performance efficiency pillar focuses on using computing resources efficiently to meet system requirements and maintaining that efficiency as demand changes and technologies evolve. For Microsoft workloads, this involves selecting appropriate cloud resources including traditional EC2 instances, managed container services, and serverless options like AWS Lambda for .NET applications. The pillar emphasizes choosing optimal compute resources and features for Windows servers, implementing appropriate storage solutions from EBS volumes to Amazon FSx services, and establishing comprehensive performance measurement and monitoring practices. By leveraging AWS's diverse service portfolio and Microsoft-specific optimizations, organizations can achieve optimal performance while maintaining cost efficiency and operational simplicity.

Focus areas

Definitions

- **EC2 fast launch:** AWS feature that accelerates Windows instance launch times by pre-provisioning snapshots and completing initialization steps like Sysprep and OOBE in advance.
- **Instance store:** Temporary block-level storage provided by disks physically attached to the EC2 host computer, offering high-performance local storage for specific use cases.

Design principles

- **Select appropriate cloud resources:** Evaluate and choose the most suitable AWS services for your Microsoft workload, considering traditional VMs, containers, and serverless options to optimize performance and operational efficiency.

- **Optimize compute resources:** Choose the right EC2 instance families and features for Windows workloads, leveraging capabilities like Fast Launch, EBS optimizations, and instance store storage to maximize performance.
- **Implement efficient storage solutions:** Select appropriate storage options from EBS volume types to managed services like Amazon FSx, matching storage performance characteristics to workload requirements.
- **Monitor and measure performance:** Establish comprehensive performance monitoring using historical data and baseline requirements to identify optimization opportunities and maintain optimal performance levels.
- **Embrace automation and managed services:** Leverage AWS managed services and automation capabilities to reduce operational overhead while maintaining high performance standards for Microsoft workloads.

Cloud resource selection

Selecting appropriate cloud resources is fundamental to achieving optimal performance for Microsoft workloads on AWS. This involves evaluating different deployment models including traditional Windows servers on EC2, managed container orchestration services, and serverless computing options. By understanding the performance characteristics and operational trade-offs of each approach, organizations can choose the most suitable architecture for their specific Microsoft workload requirements while optimizing for both performance and operational efficiency.

MSFTPERF01: How do you select the appropriate cloud resources for your Microsoft workload architecture?

It is not unusual to have multiple cloud solutions to address your Microsoft workload architecture needs. Evaluate the cloud resources that better fit your workload following the AWS Well-Architected recommendations.

Best practices

- MSFTPERF01-BP01 Consider AWS Elastic Beanstalk for running traditional Windows servers hosting your Microsoft application

- MSFTPERF01-BP02 Consider Amazon managed container orchestrator services to run containers on AWS
- MSFTPERF01-BP03 Run serverless Microsoft applications on AWS Lambda

MSFTPERF01-BP01 Consider AWS Elastic Beanstalk for running traditional Windows servers hosting your Microsoft application

In scenarios where the traditional virtual machine approach is a requirement, evaluate the EC2 instance families that better address your Microsoft workload needs. Using data-driven decisions for architectural considerations, you can also opt for Elastic Beanstalk to reduce the operational overhead on managing the EC2 instances and surrounding infrastructure resources, such as Elastic Load Balancing, and Auto Scaling Groups.

Desired outcome: Optimize performance efficiency for traditional Microsoft applications by leveraging AWS Elastic Beanstalk to reduce operational overhead while maintaining the familiar Windows Server environment, enabling faster deployments, automated scaling, and simplified infrastructure management without sacrificing application performance or functionality.

Common anti-patterns:

- Managing EC2 instances manually for Microsoft applications without leveraging platform services, leading to increased operational complexity, slower deployment cycles, and higher maintenance overhead for load balancing and scaling configurations.
- Choosing inappropriate EC2 instance families for Microsoft workloads without considering performance requirements, resulting in either over-provisioned resources that waste costs or under-provisioned instances that impact application performance.
- Implementing traditional deployment approaches without considering managed platform services, missing opportunities to improve deployment speed, reliability, and operational efficiency through automation and best practices.

Benefits of establishing this best practice:

- Reduced operational overhead through automated infrastructure management, including load balancing, auto scaling, and health monitoring, allowing teams to focus on application development rather than infrastructure maintenance.

- Improved deployment efficiency and reliability through AWS Elastic Beanstalk's automated deployment processes, version management, and rollback capabilities that streamline application updates and reduce deployment risks.
- Enhanced performance optimization through automatic scaling capabilities and integrated monitoring that ensures Microsoft applications maintain optimal performance during varying load conditions.

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

Implementation guidance

Implementing AWS Elastic Beanstalk for Microsoft applications requires understanding your application requirements and configuring the platform appropriately for Windows workloads. Begin by assessing your current application architecture and deployment processes, then configure Elastic Beanstalk with the appropriate instance types and platform settings to optimize performance while reducing operational complexity.

Implementation steps

1. Assess your Microsoft application requirements including runtime dependencies, performance needs, and scaling patterns to determine appropriate Elastic Beanstalk configuration.
2. Choose the appropriate AWS Elastic Beanstalk platform version that supports your .NET Framework or .NET Core application requirements.
3. Select optimal EC2 instance families based on your application's compute, memory, and I/O requirements, considering options like m7i, r7i, or c7i instances.
4. Configure AWS Elastic Beanstalk environment settings including auto scaling policies, load balancer configuration, and health check parameters.
5. Set up application deployment processes using AWS Elastic Beanstalk deployment methods such as rolling deployments or blue/green deployments.
6. Implement monitoring and logging integration with Amazon CloudWatch to track application performance and infrastructure metrics.
7. Configure environment variables and application settings through AWS Elastic Beanstalk configuration options to maintain environment-specific configurations.
8. Establish backup and disaster recovery procedures using AWS Elastic Beanstalk's configuration management and version control capabilities.

Resources

Related documents:

- Using Elastic Beanstalk with .NET
- Seamless Production Deployment with Elastic Beanstalk

Related tools:

- AWS Elastic Beanstalk

MSFTPERF01-BP02 Consider Amazon managed container orchestrator services to run containers on AWS

Amazon Elastic Kubernetes Service (EKS), Amazon Elastic Container Service (ECS), and AWS Fargate support both Linux and Windows containers. Either running cross-platform .NET on Linux or .NET Framework on Windows, you can run your Microsoft container-compatible workload taking advantage of the benefits of the managed service, improving performance efficiency.

Desired outcome: Achieve improved performance efficiency and operational simplicity for Microsoft workloads by leveraging managed container orchestration services that provide automated scaling, resource optimization, and reduced infrastructure management overhead while supporting both Windows and Linux container deployments.

Common anti-patterns:

- Running containerized Microsoft applications on self-managed container platforms without leveraging AWS managed services, increasing operational complexity and missing optimization opportunities.
- Choosing container orchestration without considering workload characteristics, leading to over-engineered solutions for simple applications or under-powered platforms for complex distributed systems.
- Implementing containers without proper resource allocation and scaling policies, resulting in performance issues or resource waste.
- Microsoft workloads on self-managed container orchestration platforms may limit the availability of AWS services and features that are designed to simplify the deployment and management of these workloads.

Benefits of establishing this best practice:

- Enhanced scalability and resource utilization through managed container orchestration that automatically optimizes resource allocation and scaling based on workload demands.
- Reduced operational overhead through AWS-managed control planes, automated updates, and integrated monitoring capabilities.
- Improved deployment flexibility supporting both Windows and Linux containers for different Microsoft workload components.

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

Implementation guidance

Implementing managed container orchestration for Microsoft workloads requires careful evaluation of your application architecture and container requirements. Choose the appropriate service based on your complexity needs and operational preferences, then configure for optimal performance and efficiency.

Implementation steps

1. Assess your Microsoft applications for containerization readiness and determine Windows versus Linux container requirements.
2. Choose between EKS, ECS, or Fargate based on complexity, control requirements, and operational preferences.
3. Configure container resource allocation, scaling policies, and networking for optimal performance.
4. Implement container image optimization and security scanning processes.
5. Set up monitoring and logging integration with Amazon CloudWatch Container Insights.
6. Establish CI/CD pipelines for automated container deployment and updates.

Resources

Related documents:

- Windows in Kubernetes
- Deploy Windows nodes on EKS clusters
- Launching an Amazon ECS Windows container instance
- Windows Containers Isolation Modes

- Windows containers on AWS

Related tools:

- Amazon Elastic Kubernetes Service Documentation
- Amazon Elastic Container Service Documentation
- Simplify compute management with AWS Fargate

MSFTPERF01-BP03 Run serverless Microsoft applications on AWS Lambda

Use cases such as event-driven, like message processing and API routes, application-based function, like packaging and entire cross-platform ASP.NET runtime, file processing, mobile backend, cloud automation, and similar are usually suitable for running on AWS Lambda.

Desired outcome: Achieve optimal performance efficiency and cost optimization for suitable Microsoft workloads by leveraging AWS Lambda's serverless architecture, eliminating infrastructure management overhead while providing automatic scaling, high availability, and pay-per-execution pricing for event-driven and function-based applications.

Common anti-patterns:

- Running long-running or stateful Microsoft applications on Lambda without considering execution time limits and stateless requirements, leading to performance issues or architectural mismatches.
- Implementing serverless solutions for workloads that require persistent connections or complex state management, missing the benefits of serverless while introducing unnecessary complexity.
- Choosing Lambda without evaluating cold start impacts on performance-sensitive applications, potentially affecting user experience for latency-critical workloads.

Benefits of establishing this best practice:

- Eliminated infrastructure management overhead through fully managed serverless execution environment that automatically handles scaling, patching, and availability.
- Optimized cost efficiency through pay-per-execution pricing model that eliminates costs for idle resources and automatically scales to zero when not in use.

- Enhanced performance for event-driven workloads through automatic scaling and optimized execution environment designed for short-lived, stateless functions.

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

Implementation guidance

Implementing serverless Microsoft applications on AWS Lambda requires careful evaluation of application architecture and suitability for serverless patterns. Focus on event-driven, stateless workloads and optimize for Lambda's execution model to achieve maximum performance efficiency.

Implementation steps

1. Identify Microsoft workload components suitable for serverless architecture including event-driven functions, API endpoints, and batch processing tasks.
2. Refactor applications to follow serverless patterns with stateless, event-driven design principles.
3. Configure Lambda functions with appropriate runtime, memory allocation, and timeout settings for optimal performance.
4. Implement efficient cold start optimization techniques including provisioned concurrency for latency-sensitive functions.
5. Set up event sources and triggers using services like API Gateway, S3, SQS, or EventBridge.
6. Configure monitoring and observability using CloudWatch, X-Ray, and Lambda Insights for performance tracking.

Resources

Related documents:

- Building .NET applications on AWS Lambda

Related tools:

- AWS Lambda
- AWS .NET Development Blog

Compute resources

Optimizing compute resources for Microsoft workloads running on Windows servers requires careful selection of EC2 instance families and features that align with workload requirements. This includes choosing appropriate instance types for different use cases, leveraging performance-enhancing features like EC2 Fast Launch and EBS optimizations, and utilizing instance store storage for high-performance scenarios. By understanding the performance characteristics of different compute options, organizations can maximize the efficiency of their Windows-based applications while maintaining cost-effectiveness.

MSFTPERF02: How do you select the appropriate compute resources and features for your Microsoft workloads running on Windows servers?

Amazon EC2 provides scalability, flexibility, and cost efficiency resources that can apply to Windows EC2 instances. Appropriate instance families and EC2 features can be used by your Windows machines to bring the performance efficiency needed by your workload.

Best practices

- MSFTPERF02-BP01 Choose the Amazon EC2 instance families that best fit the Microsoft workload
- MSFTPERF02-BP02 Consider the use for EC2 Fast Launch to accelerate launching your Microsoft workload instances
- MSFTPERF02-BP03 Consider using Amazon EBS fast snapshot restore
- MSFTPERF02-BP04 Consider using Amazon EBS Provisioned Rate for Volume Initialization

MSFTPERF02-BP01 Choose the Amazon EC2 instance families that best fit the Microsoft workload

Amazon EC2 provides different instance family types, addressing different purposes. For example, General purpose instances, such as m7i and m7a can be used for most production applications running on Windows Server. For non-production or less critical environments, t3 burstable instances may also be a fit. Memory optimized instances, such as r7i, r7a, and x2iedn provide greater ratio of memory to vCPU and are ideal for memory-intensive workloads, such as Microsoft SQL Server.

Desired outcome: Optimize performance and cost efficiency by selecting the most appropriate EC2 instance families that align with your Microsoft workload's specific compute, memory, and I/O requirements, ensuring optimal resource utilization while maintaining application performance and scalability.

Common anti-patterns:

- Choosing instance types based solely on cost without considering performance requirements, leading to under-provisioned resources that impact application performance and user experience.
- Using the same instance family for all workloads without evaluating specific requirements, missing opportunities to optimize performance for memory-intensive applications like SQL Server or compute-intensive .NET applications.
- Over-provisioning instances with excessive resources "just in case" without analyzing actual workload patterns, resulting in unnecessary costs and inefficient resource utilization.

Benefits of establishing this best practice:

- Optimized performance through instance families specifically designed for different workload characteristics, ensuring Microsoft applications receive appropriate compute, memory, and I/O resources.
- Improved cost efficiency by matching instance capabilities to actual workload requirements, avoiding over-provisioning while maintaining performance standards.
- Enhanced scalability and flexibility through understanding of instance family characteristics, enabling better architectural decisions for different Microsoft workload components.

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

Implementation guidance

Selecting appropriate EC2 instance families for Microsoft workloads requires understanding both your application requirements and the characteristics of different instance types. Begin by analyzing your workload patterns, then match them to instance families that provide optimal price-performance ratios for your specific use cases.

Implementation steps

1. Analyze your Microsoft workload requirements including CPU utilization patterns, memory requirements, storage I/O needs, and network performance requirements.
2. Evaluate different EC2 instance families based on your workload characteristics:
   - General purpose (m7i, m7a, m6i) for balanced workloads
   - Memory optimized (r7i, r7a, x2iedn) for SQL Server and memory-intensive applications
   - Compute optimized (c7i, c7a) for CPU-intensive .NET applications
   - Burstable (t3, t4g1) for variable or low-utilization workloads
3. Consider processor architecture options including Intel, AMD, and AWS Graviton processors based on application compatibility and performance requirements.
4. Evaluate instance sizes within families to match vCPU and memory requirements without over-provisioning resources.
5. Test different instance types in non-production environments to validate performance and cost characteristics.
6. Implement monitoring using Amazon CloudWatch and AWS Compute Optimizer to track instance utilization and receive rightsizing recommendations.
7. Establish regular review processes to evaluate instance performance and adjust selections based on changing workload patterns.
8. Document instance selection criteria and rationale for different Microsoft workload components to guide future decisions.

Windows Server OS does not support ARM based processors. Consider using AWS Graviton based instances can be used to run cross-platform .NET on Linux.

Resources

Related documents:

- Amazon EC2 Instance Types

Related tools:

- AWS Compute Optimizer

MSFTPERF02-BP02 Consider the use for EC2 Fast Launch to accelerate launching your Microsoft workload instances

EC2 Fast Launch will speed up the Windows EC2 instance launch process. When you configure a Windows Server AMI for EC2 Fast Launch, Amazon EC2 creates a set of pre-provisioned snapshots to use for faster launching. It completes steps such as Sysprep specialize, Windows Out of Box Experience (OOBE), and rebooting as required. Especially useful when you need to scale fast.

Desired outcome: Significantly reduce Windows instance launch times and improve scaling responsiveness for Microsoft workloads by leveraging EC2 Fast Launch to pre-provision snapshots and complete initialization steps, enabling rapid deployment and auto-scaling capabilities for time-sensitive applications.

Common anti-patterns:

- Accepting standard Windows instance launch times without evaluating Fast Launch benefits, missing opportunities to improve application availability and user experience during scaling events.
- Implementing Fast Launch without considering the additional costs of pre-provisioned snapshots and temporary instances, potentially increasing expenses without adequate benefit analysis.
- Using Fast Launch for infrequently launched instances where the preparation overhead exceeds the benefits, leading to unnecessary complexity and costs.

Benefits of establishing this best practice:

- Dramatically reduced instance launch times through pre-provisioned snapshots that eliminate Windows initialization steps like Sysprep and OOBE during actual instance launches.
- Improved application availability and scaling responsiveness during traffic spikes or auto-scaling events, enhancing user experience and system reliability.
- Enhanced disaster recovery capabilities through faster instance replacement and environment restoration when rapid recovery is critical for business continuity.

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

Implementation guidance

Implementing EC2 Fast Launch requires careful evaluation of your scaling patterns and cost-benefit analysis. Focus on AMIs that are frequently launched and where launch time significantly impacts application performance or user experience.

Implementation steps

1. Identify Windows AMIs that are frequently launched or require rapid scaling capabilities for your Microsoft workloads.
2. Analyze current instance launch times and scaling patterns to determine potential benefits of Fast Launch implementation.
3. Configure Fast Launch for selected AMIs through the EC2 console or AWS CLI, specifying the number of pre-provisioned snapshots to maintain.
4. Monitor Fast Launch metrics including launch time improvements and associated costs for pre-provisioned resources.
5. Evaluate cost-benefit ratio considering snapshot storage costs, temporary instance costs, and performance improvements.
6. Integrate Fast Launch-enabled AMIs into auto-scaling groups and deployment processes to maximize scaling responsiveness.
7. Establish monitoring and alerting for Fast Launch resource utilization to optimize the number of pre-provisioned snapshots.
8. Document Fast Launch configuration and regularly review effectiveness based on actual scaling patterns and requirements.

Resources

Related documents:

- Configuring your Windows AMI for faster launching
- Launch Microsoft Windows Server instances on Amazon EC2 up to 65% faster than before

Related tools:

- Launch Microsoft Windows Server instances on Amazon EC2 up to 65% faster than before

MSFTPERF02-BP03 Consider using Amazon EBS fast snapshot restore

Amazon EBS Fast Snapshot Restore (FSR) offers significant advantages for Microsoft workloads by eliminating the initialization latency typically associated with first-use EBS volumes created from snapshots. This is particularly beneficial for Windows Server instances and SQL Server deployments where quick recovery time objectives (RTOs) are crucial. When enabled on selected snapshots in specific Availability Zones, FSR ensures that EBS volumes created from these snapshots deliver their full performance immediately without the need for the traditional initialization process, which normally requires reading all blocks from S3. For Microsoft workloads that require rapid failover, disaster recovery, or test environment provisioning, FSR can dramatically reduce the time needed to bring systems online.

Desired outcome: Achieve immediate full performance for EBS volumes created from snapshots, eliminating initialization latency for Microsoft workloads and enabling rapid disaster recovery, failover scenarios, and test environment provisioning with predictable performance characteristics from the moment volumes are attached.

Common anti-patterns:

- Accepting standard EBS volume initialization performance without evaluating FSR benefits for time-critical Microsoft workloads, missing opportunities to improve recovery times and system availability.
- Implementing FSR on all snapshots without cost-benefit analysis, leading to unnecessary expenses for snapshots that don't require immediate full performance.
- Using FSR without proper planning for Availability Zone placement, limiting the effectiveness of the feature for disaster recovery and high availability scenarios.

Benefits of establishing this best practice:

- Eliminated initialization latency providing immediate full performance for EBS volumes, crucial for rapid disaster recovery and failover scenarios for Microsoft workloads.
- Improved predictability for recovery time objectives (RTOs) by removing variable initialization times that can impact business continuity planning.
- Enhanced operational efficiency for test environment provisioning and development workflows where rapid volume availability is essential for productivity.

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

Implementation guidance

Implementing EBS fast snapshot restore (FSR) requires strategic selection of snapshots and Availability Zones based on your Microsoft workload's recovery and performance requirements. Focus on critical snapshots used for disaster recovery, production failover, or frequently accessed test environments.

Implementation steps

1. Identify critical EBS snapshots used for Microsoft workload disaster recovery, production databases, or frequently provisioned test environments.
2. Analyze recovery time objectives (RTOs) and determine which workloads would benefit most from immediate volume performance.
3. Enable FSR for selected snapshots in appropriate Availability Zones based on your deployment architecture.
4. Monitor FSR usage and costs to ensure the feature provides adequate value for the additional expense incurred.
5. Integrate FSR-enabled snapshots into disaster recovery procedures and automated failover processes.
6. Test volume creation and performance validation procedures to confirm FSR effectiveness for your Microsoft workloads.
7. Establish policies for FSR lifecycle management including enabling or disabling based on snapshot age and usage patterns.
8. Document FSR configuration and include in operational runbooks for disaster recovery and environment provisioning procedures.

Resources

Related documents:

- Amazon EBS fast snapshot restore
- Instant performance on Amazon EBS volumes restored from snapshots using Fast Snapshot Restore

Related tools:

- Amazon EBS

MSFTPERF02-BP04 Consider using Amazon EBS Provisioned Rate for Volume Initialization

Amazon EBS provisioned rate for volume initialization (PRVI) offers significant advantages for Microsoft workloads by providing predictable and faster initialization times for new EBS volumes created from snapshots. This feature is particularly valuable for Windows Server deployments and SQL Server environments where consistent and reliable performance during volume initialization is crucial. By allowing you to specify the initialization rate up to 300 MiB/s, PRVI enables you to control and accelerate the background process of loading data from S3 to the EBS volume, ensuring your Microsoft applications can access their data more quickly and predictably.

Desired outcome: Achieve predictable and accelerated EBS volume initialization for Microsoft workloads through controlled initialization rates, ensuring consistent performance during volume creation and reducing the impact of initialization processes on application availability and user experience.

Common anti-patterns:

- Accepting variable and unpredictable volume initialization times without considering PRVI benefits, leading to inconsistent application performance and unpredictable recovery times.
- Implementing PRVI without cost-benefit analysis for specific workloads, potentially incurring additional costs without adequate performance improvements for the use case.
- Using PRVI without proper integration into disaster recovery and scaling procedures, missing opportunities to improve overall system reliability and predictability.

Benefits of establishing this best practice:

- Predictable initialization performance through controlled initialization rates that enable reliable capacity planning and recovery time estimation for Microsoft workloads.
- Improved application availability during scaling events and disaster recovery scenarios where consistent volume initialization performance is critical for meeting SLAs.
- Enhanced operational efficiency through reduced variability in volume provisioning times, enabling more reliable automation and orchestration of Microsoft workload deployments.

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

Implementation guidance

Implementing EBS Provisioned Rate for Volume Initialization requires careful evaluation of your Microsoft workload's initialization requirements and cost considerations. Focus on scenarios where predictable initialization performance is critical for meeting operational objectives.

Implementation steps

1. Identify Microsoft workloads that require predictable volume initialization performance, particularly for disaster recovery and scaling scenarios.
2. Analyze current volume initialization patterns and determine appropriate provisioned rates based on performance requirements and cost considerations.
3. Configure PRVI for relevant EBS volumes with initialization rates up to 300 MiB/s based on workload needs and budget constraints.
4. Monitor initialization performance and costs to validate the effectiveness of PRVI implementation for your specific use cases.
5. Integrate PRVI-configured volumes into automated deployment and disaster recovery procedures to maximize predictability benefits.
6. Establish monitoring and alerting for initialization performance to ensure PRVI is delivering expected results.
7. Document PRVI configuration decisions and include in operational procedures for volume management and disaster recovery.
8. Regularly review PRVI usage and costs to optimize configuration based on actual performance requirements and business value.

Resources

Related documents:

- Initialize Amazon EBS volumes
- Accelerate the transfer of data from an Amazon EBS snapshot to a new EBS volume

Related tools:

- Accelerate EBS snapshot data transfer

Storage solutions

Selecting appropriate storage solutions is critical for Microsoft workload performance, as storage often becomes a bottleneck for Windows applications and SQL Server databases. AWS offers various storage options from different EBS volume types to fully managed services like Amazon FSx for Windows File Server and FSx for ONTAP. Understanding the performance characteristics, use cases, and cost implications of each storage option enables organizations to optimize their Microsoft workloads for both performance and efficiency.

MSFTPERF03: How do you select the appropriate storage solutions for your Microsoft workloads running on Windows servers?

AWS offers different storage options that can address different needs for your Microsoft workloads. Appropriate storage options can be used by your Windows machines to bring the performance efficiency needed by your workload.

Best practices

- MSFTPERF03-BP01 Consider Amazon EBS gp3 volumes for general workloads
- MSFTPERF03-BP02 Consider Amazon EBS io2 Block Express volumes for high-intense I/O workloads
- MSFTPERF03-BP03 Consider Amazon FSx for Windows File Server
- MSFTPERF03-BP04 Consider Amazon FSx for NetApp ONTAP
- MSFTPERF03-BP05 Leverage instance store temporary block storage for EC2 instances

MSFTPERF03-BP01 Consider Amazon EBS gp3 volumes for general workloads

Amazon EBS's newest and most cost-effective SSD option, General Purpose SSD (gp3) volumes, strikes an optimal balance between price and performance for a wide range of applications. A key advantage of gp3 volumes is the ability to adjust performance independently of storage capacity, allowing users to meet specific performance requirements without unnecessarily increasing block storage. Moreover, gp3 volumes offer significant cost savings, with prices 20% lower per GiB compared to their predecessor, General Purpose SSD (gp2) volumes.

Desired outcome: Optimize storage performance and cost efficiency for Microsoft workloads by leveraging gp3 volumes that provide independent scaling of IOPS and throughput from storage capacity, enabling right-sized storage configurations that meet performance requirements while minimizing costs.

Common anti-patterns:

- Continuing to use gp2 volumes without evaluating gp3 benefits, missing opportunities for cost savings and performance optimization through independent IOPS and throughput scaling.
- Over-provisioning storage capacity to meet IOPS requirements when using gp2 volumes, leading to unnecessary storage costs that could be avoided with gp3's independent performance scaling.
- Choosing high-performance storage options like io1/io2 for workloads that could be adequately served by gp3 with appropriate IOPS configuration, resulting in unnecessary costs.

Benefits of establishing this best practice:

- Significant cost savings through 20% lower per-GiB pricing compared to gp2 volumes while maintaining or improving performance characteristics for Microsoft workloads.
- Enhanced flexibility through independent scaling of IOPS and throughput from storage capacity, enabling optimal resource allocation without over-provisioning storage.
- Improved performance predictability through consistent baseline performance and the ability to provision additional IOPS and throughput as needed for specific workload requirements.

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

Implementation guidance

Implementing gp3 volumes for Microsoft workloads requires understanding your storage performance requirements and migrating from existing volume types where appropriate. Focus on workloads that can benefit from independent IOPS and throughput scaling while achieving cost savings.

Implementation steps

1. Analyze current storage performance requirements for Microsoft workloads including IOPS, throughput, and capacity needs.
2. Identify existing gp2 volumes and other storage types that could benefit from migration to gp3 for cost and performance optimization.

3. Plan gp3 volume configurations with appropriate baseline performance and additional provisioned IOPS or throughput based on workload requirements.
4. Test gp3 performance in non-production environments to validate performance characteristics for your specific Microsoft applications.
5. Implement migration procedures for existing volumes using EBS volume modification or snapshot-based migration approaches.
6. Monitor storage performance and costs after migration to validate expected benefits and optimize configurations as needed.
7. Establish policies for new volume provisioning that default to gp3 unless specific requirements dictate alternative storage types.
8. Document gp3 configuration standards and include in storage provisioning procedures for consistent implementation across environments.

Resources

Related documents:

- General Purpose SSD (gp3) volumes

Related tools:

- Migrate Amazon EBS volumes from gp2 to gp3

MSFTPERF03-BP02 Consider Amazon EBS io2 Block Express volumes for high-intense I/O workloads

Amazon EBS io2 Block Express volumes are based on an updated storage server architecture. They are designed to handle high I/O requirements for applications running on Nitro System-based instances. These volumes offer improved durability and lower latency. As a result, they are suitable for resource-intensive applications that require consistent performance, such as certain database systems (For example, Oracle, SAP HANA, and Microsoft SQL Server) and SAS Analytics.

Desired outcome: Achieve maximum I/O performance and lowest latency for demanding Microsoft workloads, particularly SQL Server databases and other I/O-intensive applications, through io2 Block Express volumes that provide consistent high-performance storage with enhanced durability and reliability.

Common anti-patterns:

- Using general-purpose storage for high-performance Microsoft SQL Server databases without evaluating io2 Block Express benefits, potentially limiting application performance and user experience.
- Implementing io2 Block Express for workloads that don't require extreme I/O performance, leading to unnecessary costs without proportional performance benefits.
- Choosing io2 Block Express without ensuring compatibility with Nitro System-based instances, missing the full performance potential of the storage technology.

Benefits of establishing this best practice:

- Maximum I/O performance through io2 Block Express architecture designed specifically for high-intensity workloads, enabling optimal performance for demanding Microsoft applications.
- Enhanced reliability and durability through improved storage architecture that provides consistent performance and reduced latency for mission-critical workloads.
- Improved application responsiveness for I/O-intensive Microsoft workloads including SQL Server databases, analytics applications, and high-performance computing scenarios.

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

Implementation guidance

Implementing io2 Block Express volumes requires careful evaluation of I/O requirements and cost considerations. Focus on workloads that genuinely require extreme I/O performance and can justify the additional costs through improved application performance and business outcomes.

Implementation steps

1. Identify Microsoft workloads with high I/O requirements that would benefit from io2 Block Express performance characteristics, particularly SQL Server databases and analytics applications.
2. Analyze current I/O patterns including IOPS requirements, throughput needs, and latency sensitivity to determine if io2 Block Express is appropriate.
3. Ensure compatibility with Nitro System-based instances that can fully utilize io2 Block Express performance capabilities.

4. Configure io2 Block Express volumes with appropriate IOPS provisioning based on workload requirements and performance testing results.
5. Implement performance testing in non-production environments to validate expected performance improvements and cost justification.
6. Monitor storage performance metrics including IOPS utilization, throughput, and latency to ensure optimal configuration and utilization.
7. Establish cost monitoring and optimization procedures to ensure io2 Block Express usage remains cost-effective for the performance benefits provided.
8. Document io2 Block Express configuration standards and use cases for consistent implementation across high-performance Microsoft workloads.

Resources

Related documents:

- Provisioned IOPS SSD (io2 Block Express) volumes
- Best practices for Amazon RDS for SQL Server with Amazon EBS io2 Block Express volumes up to 64 TiB

Related tools:

- io2 Block Express considerations

MSFTPERF03-BP03 Consider Amazon FSx for Windows File Server

Amazon FSx for Windows File Server is a managed service that provides file storage using Microsoft Windows file system technology. It supports Windows file system features and uses the Server Message Block (SMB) protocol for network file access, making it compatible with various Windows-based enterprise workloads and applications. The service offers integration with other AWS services and performance optimized for enterprise applications, aiming to provide low-latency file storage. FSx for Windows File Server is designed for Windows workloads that require shared file storage, such as File Servers, Application Server configuration stores, and even Microsoft SQL Server databases.

Desired outcome: Achieve high-performance, fully managed Windows file storage that seamlessly integrates with Microsoft workloads, providing native Windows file system features, SMB protocol

support, and optimized performance while reducing operational overhead through AWS-managed infrastructure.

Common anti-patterns:

- Implementing self-managed Windows file servers on EC2 without evaluating FSx benefits, missing opportunities to reduce operational overhead and improve performance through managed services.
- Using general-purpose storage solutions for Windows workloads that require specific Windows file system features, potentially limiting functionality and performance.
- Choosing FSx configurations without proper performance analysis, leading to either over-provisioned resources that increase costs or under-provisioned storage that impacts application performance.

Benefits of establishing this best practice:

- Reduced operational overhead through fully managed Windows file storage that eliminates the need to manage file server infrastructure, patching, and maintenance tasks.
- Enhanced performance and reliability through AWS-managed infrastructure optimized for Windows workloads with built-in high availability and backup capabilities.
- Native Windows integration providing full compatibility with Windows file system features, Active Directory integration, and SMB protocol support for seamless application integration.

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

Implementation guidance

Implementing Amazon FSx for Windows File Server requires understanding your file storage requirements and migration planning from existing file server infrastructure. Focus on workloads that require Windows-native file system features and can benefit from managed service advantages.

Implementation steps

1. Assess current Windows file storage requirements including capacity, performance, and feature needs for your Microsoft workloads.
2. Evaluate existing file server infrastructure and identify workloads suitable for migration to FSx for Windows File Server.

3. Choose appropriate FSx deployment options including Single-AZ or Multi-AZ configurations based on availability and performance requirements.
4. Configure FSx file systems with appropriate storage capacity, throughput, and IOPS settings based on workload analysis and performance testing.
5. Plan migration procedures for existing file shares and data, including user access permissions and Active Directory integration.
6. Implement backup and disaster recovery strategies using FSx's built-in backup capabilities and cross-region replication options.
7. Monitor file system performance and utilization using CloudWatch metrics to optimize configuration and identify scaling needs.
8. Establish operational procedures for FSx management including access control, monitoring, and capacity planning for ongoing operations.

Resources

Related documents:

- What is Amazon FSx for Windows File Server?
- Optimizing Amazon FSx for Windows File Server performance with new metrics

Related tools:

- Amazon FSx performance

MSFTPERF03-BP04 Consider Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP is a fully managed AWS service that provides scalable, high-performance file storage based on the widely-used NetApp ONTAP file system. It combines the familiar features and capabilities of NetApp systems with the benefits of a cloud-managed service. This service offers fast, flexible shared file storage accessible from Linux, Windows, and macOS instances, both in AWS and on-premises. FSx for ONTAP provides high-performance SSD storage with very low latencies and also HDD storage. Amazon FSx for NetApp ONTAP offers robust file storage capabilities, including support for petabyte-scale datasets in a single namespace and high throughput of up to tens of GBps per file system.

Desired outcome: Achieve enterprise-grade, high-performance file storage for Microsoft workloads through FSx for ONTAP, providing multi-protocol access, advanced data management features, and

cost optimization capabilities while maintaining compatibility with existing NetApp environments and Microsoft applications.

Common anti-patterns:

- Using basic file storage solutions for enterprise Microsoft workloads without evaluating FSx for ONTAP's advanced features, missing opportunities for performance optimization and data management capabilities.
- Implementing FSx for ONTAP without leveraging its multi-protocol capabilities, limiting the potential for workload consolidation and simplified architecture.
- Choosing FSx for ONTAP configurations without considering data tiering and compression features, potentially missing significant cost optimization opportunities.

Benefits of establishing this best practice:

- Superior performance and scalability through NetApp ONTAP technology providing high throughput, low latency, and support for petabyte-scale datasets in a single namespace.
- Advanced data management capabilities including automatic tiering, compression, deduplication, and snapshot technologies that optimize both performance and costs.
- Multi-protocol flexibility supporting NFS, SMB, iSCSI, and NVMe protocols, enabling consolidation of diverse Microsoft workload storage requirements on a single platform.

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

Implementation guidance

Implementing Amazon FSx for NetApp ONTAP requires understanding your enterprise storage requirements and planning for advanced data management features. Focus on workloads that can benefit from multi-protocol access, advanced data services, and cost optimization through data efficiency features.

Implementation steps

1. Assess enterprise storage requirements for Microsoft workloads including performance, capacity, protocol needs, and data management requirements.
2. Evaluate existing NetApp environments and plan migration strategies to leverage familiar ONTAP features in the cloud.

3. Configure FSx for ONTAP file systems with appropriate performance tiers, capacity planning, and multi-protocol access based on workload requirements.
4. Implement data efficiency features including compression, deduplication, and automatic tiering to optimize storage costs and performance.
5. Configure multi-protocol access (SMB, NFS, iSCSI) to support diverse Microsoft workload requirements and enable workload consolidation.
6. Establish backup and disaster recovery procedures using ONTAP's snapshot and replication capabilities for data protection.
7. Monitor storage performance, utilization, and cost optimization through CloudWatch metrics and ONTAP management tools.
8. Implement ongoing data management policies including tiering, retention, and capacity planning to maintain optimal performance and costs.

Resources

Related documents:

- What is Amazon FSx for NetApp ONTAP?
- Managing storage on Windows servers with Amazon FSx for NetApp ONTAP
- Best practice configuration of Amazon FSx for NetApp ONTAP for Microsoft SQL Server workloads
- AWS Guidance: Best Practices for running MSSQL workloads on FSx for ONTAP

Related tools:

- Amazon FSx for NetApp ONTAP performance

MSFTPERF03-BP05 Leverage instance store temporary block storage for EC2 instances

An instance store is a form of temporary block-level storage for EC2 instances, provided by disks physically attached to the host computer. It is well-suited for storing frequently changing data such as buffers, caches, and scratch data, as well as temporary data replicated across multiple instances like in a load-balanced web server pool, and Microsoft SQL Server TempDB data. The capacity and number of instance store volumes available vary depending on the instance type and size, with some instance types not offering instance stores at all.

Desired outcome: Maximize I/O performance for temporary and cache data in Microsoft workloads by leveraging instance store volumes that provide the highest possible storage performance through direct attachment to the host computer, particularly beneficial for SQL Server TempDB, application caches, and high-performance computing scenarios.

Common anti-patterns:

- Using EBS volumes for temporary data and caches when instance store volumes are available, missing opportunities for maximum I/O performance and potentially increasing storage costs.
- Storing persistent or critical data on instance store volumes without understanding their temporary nature, risking data loss during instance stops or failures.
- Choosing instance types without instance store when workloads could benefit from high-performance temporary storage, limiting application performance potential.

Benefits of establishing this best practice:

- Maximum I/O performance through direct-attached storage that provides the highest possible throughput and lowest latency for temporary data operations.
- Cost optimization by using included instance store volumes for appropriate use cases instead of provisioning additional EBS volumes for temporary storage needs.
- Enhanced application performance for Microsoft workloads that heavily utilize temporary storage, such as SQL Server TempDB operations and application-level caching.

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

Implementation guidance

Implementing instance store volumes requires careful planning to ensure appropriate use cases and data management practices. Focus on temporary, cache, and scratch data that can benefit from maximum I/O performance while ensuring critical data remains on persistent storage.

Implementation steps

1. Identify Microsoft workload components that generate temporary data, caches, or scratch files that could benefit from high-performance instance store volumes.
2. Evaluate EC2 instance families that include instance store volumes and assess their capacity and performance characteristics for your workload requirements.

3. Plan data placement strategies to ensure only appropriate temporary data is stored on instance store volumes while maintaining persistent data on EBS.
4. Configure applications to utilize instance store volumes for SQL Server TempDB, application caches, temporary files, and other high-I/O temporary data.
5. Implement data management procedures that account for the temporary nature of instance store volumes, including startup initialization and data replication strategies.
6. Monitor instance store utilization and performance to validate expected benefits and optimize usage patterns for maximum efficiency.
7. Establish operational procedures for instance lifecycle management that properly handle instance store data during maintenance and scaling operations.
8. Document instance store usage patterns and include in application deployment and disaster recovery procedures.

Resources

Related documents:

- Amazon EC2 instance store
- Make instance store volume available for use on an EC2 instance

Related tools:

- Instance store volumes limits for EC2 instances

Performance measurement

Measuring and monitoring performance is essential for maintaining optimal efficiency of Microsoft workloads over time. This involves using historical data to establish performance baselines, defining performance requirements, and implementing monitoring systems that can detect anomalies and optimization opportunities. By establishing comprehensive performance measurement practices, organizations can proactively identify issues, optimize resource utilization, and ensure their Microsoft workloads continue to meet performance expectations as requirements evolve.

MSFTPERF04: How do you measure performance requirements changes and anomalies?

Determine your Microsoft workload's typical operational metrics by analyzing past performance data. Configure notification systems to alert IT personnel when significant variations from these norms occur. Create and document protocols for technical staff to rectify such anomalies, utilizing both human-driven and automated correction methods. This strategy facilitates preemptive oversight of the workload efficiency and contributes to overall system stability.

Best practices

- MSFTPERF04-BP01 Use historical data to evaluate performance
- MSFTPERF04-BP02 Define baseline performance requirements

MSFTPERF04-BP01 Use historical data to evaluate performance

Effective assessment of Microsoft workload performance requires comprehensive data collection across key system components: compute, memory, storage, and networking. This approach aligns with the Well-Architected Framework's Performance Excellence guidelines, specifically the best practice PERF02-BP03, which focuses on gathering compute-related metrics. By monitoring these critical areas, organizations can identify suboptimal performance and implement timely corrective measures. This holistic monitoring strategy enables proactive management of Microsoft workloads, ensuring they meet performance expectations and allowing for swift intervention when performance falls below desired thresholds.

Desired outcome: Establish comprehensive performance data collection and analysis capabilities for Microsoft workloads that enable data-driven optimization decisions, proactive issue identification, and continuous performance improvement through historical trend analysis and performance pattern recognition.

Common anti-patterns:

- Collecting performance data without systematic analysis or historical comparison, missing opportunities to identify performance trends and optimization opportunities over time.
- Monitoring only basic system metrics without collecting Microsoft-specific performance indicators, limiting visibility into application-level performance issues and optimization potential.
- Implementing reactive performance monitoring that only triggers during incidents, rather than proactive analysis that can prevent performance degradation before it impacts users.

Benefits of establishing this best practice:

- Data-driven optimization decisions through comprehensive historical performance analysis that identifies trends, patterns, and optimization opportunities across Microsoft workload components.
- Proactive issue identification and prevention through continuous monitoring and analysis that can detect performance degradation before it impacts business operations.
- Improved capacity planning and resource allocation through historical data analysis that enables accurate forecasting of future performance and scaling requirements.

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

Implementation guidance

Implementing comprehensive performance data collection and analysis requires establishing systematic monitoring across all Microsoft workload components and creating processes for regular performance evaluation and optimization.

Implementation steps

1. Identify key performance metrics for Microsoft workload components including compute, memory, storage, and network performance indicators.
2. Configure comprehensive monitoring using Amazon CloudWatch, Performance Counters, and application-specific monitoring tools to collect historical performance data.
3. Establish data retention policies that maintain sufficient historical data for trend analysis and performance comparison over time.
4. Implement automated data analysis and reporting processes that regularly evaluate performance trends and identify optimization opportunities.
5. Create performance dashboards and visualization tools that enable easy analysis of historical performance data and trend identification.
6. Establish regular performance review processes that analyze historical data to identify patterns, anomalies, and optimization opportunities.
7. Document performance baselines and thresholds based on historical analysis to enable effective anomaly detection and alerting.
8. Integrate historical performance analysis into capacity planning and architectural decision-making processes for continuous improvement.

Resources

Related documents:

- Monitoring Windows services with Amazon CloudWatch
- How do I use the CloudWatch agent on a Windows Server to view metrics for Performance Monitor?
- How to monitor Windows and Linux servers and get internal performance metrics
- Run ADOTCollector on AWS Windows Ec2 Host

Related tools:

- Amazon CloudWatch
- OpenTelemetry

MSFTPERF04-BP02 Define baseline performance requirements

Microsoft workloads vary in their performance needs, making historical data analysis crucial for establishing baseline performance metrics. This approach allows organizations to detect and quantify performance fluctuations effectively. By implementing targeted alerts, IT teams can quickly identify anomalies, such as unexpected CPU usage spikes, changes in storage throughput, increased memory consumption, or more intricate performance issues. The collected monitoring data serves a dual purpose: it not only helps in detecting problems, but also provides valuable insights for ongoing performance optimization.

Desired outcome: Establish clear, measurable performance baselines for Microsoft workloads that enable effective anomaly detection, performance optimization, and capacity planning while providing objective criteria for evaluating system health and performance improvements over time.

Common anti-patterns:

- Operating Microsoft workloads without defined performance baselines, making it difficult to identify when performance degrades or to measure the effectiveness of optimization efforts.
- Setting performance baselines based on assumptions rather than actual historical data analysis, leading to inappropriate thresholds that generate false alerts or miss genuine performance issues.

- Creating static baselines that does not account for normal performance variations or business cycles, resulting in alert fatigue or missed performance degradation during expected usage patterns.

Benefits of establishing this best practice:

- Effective anomaly detection through well-defined baselines that enable accurate identification of performance deviations and potential issues before they impact business operations.
- Improved performance optimization through objective measurement criteria that enable evaluation of optimization efforts and identification of areas requiring attention.
- Enhanced capacity planning and resource allocation through baseline-driven analysis that supports data-driven decisions about scaling and infrastructure investments.

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

Implementation guidance

Implementing performance baselines requires systematic analysis of historical performance data and establishment of meaningful thresholds that account for normal variations while detecting genuine performance issues.

Implementation steps

1. Collect sufficient historical performance data across all Microsoft workload components to establish statistically meaningful baselines.
2. Analyze performance patterns including daily, weekly, and seasonal variations to understand normal performance fluctuations.
3. Define performance baseline metrics for key indicators including CPU utilization, memory consumption, storage I/O, network throughput, and application response times.
4. Establish performance thresholds and alert criteria based on statistical analysis of historical data and business requirements.
5. Configure monitoring and alerting systems to detect deviations from established baselines and notify appropriate teams of performance anomalies.
6. Implement regular baseline review and adjustment processes to account for changing workload patterns and business requirements.
7. Document baseline definitions, measurement criteria, and alert thresholds for consistent application across environments and teams.

8. Integrate baseline monitoring into operational procedures and incident response processes to enable rapid performance issue identification and resolution.

Resources

Related documents:

- Using CloudWatch outlier detection
- Best practices for monitoring Microsoft SQL Server on Amazon EC2
- Windows Server - Power and performance tuning
- Select the right instance type for Windows workloads
- FSx for Windows File Server performance
- Windows container memory requirements

Related tools:

- Amazon CloudWatch
- AWS Systems Manager

Cost optimization

The cost optimization pillar focuses on maximizing value while minimizing expenses for Microsoft workloads on AWS. This includes comprehensive assessment tools, Windows EC2 and SQL Server cost optimization through right-sizing and licensing strategies, storage optimization, and modernization approaches. By implementing solutions like AWS Managed services, containerization, and serverless architectures where appropriate, organizations can reduce costs while maintaining performance and scalability. The pillar emphasizes both immediate cost-saving opportunities through resource optimization and long-term benefits through strategic modernization of Microsoft workloads.

Focus areas

- Design principles
- Assessment
- Operating system
- Databases
- Storage
- Active directory
- Containers
- .NET

Design principles

- **Implement right-sizing:** Continuously assess and adjust resource allocation to match actual workload requirements, avoiding over-provisioning.
- **Leverage flexible licensing models:** Utilize Bring Your Own License (BYOL) options and evaluate different editions to optimize software licensing costs.
- **Adopt managed services:** Embrace AWS-managed solutions to reduce operational overhead and benefit from built-in cost optimizations.
- **Modernize strategically:** Gradually refactor applications to leverage cloud-native and serverless architectures for improved cost-efficiency.
- **Automate cost management:** Implement automated tools and processes for ongoing cost monitoring, reporting, and optimization across your Microsoft workloads.

Assessment

Comprehensive assessment of your Microsoft workload environment is a critical first step in optimizing costs on AWS. By utilizing discovery and assessment tools like Migration Evaluator and AWS OLA, organizations can gain complete visibility into their workloads, understand resource utilization patterns, and identify opportunities for optimization. This evaluation process creates the foundation for effective cost management and resource rightsizing of Microsoft workloads.

MSFTCOST01: How comprehensively have you assessed your Microsoft workload environment?

It is important to have the complete picture of your workload. Taking this first step ensures your applications are using right-sized resources. Gathering licensing information may also be aided by this.

Best practices

- MSFTCOST01-BP01 Run discovery tools
- MSFTCOST01-BP02 Run assessment tools
- MSFTCOST01-BP03 Run platform-specific tools

MSFTCOST01-BP01 Run discovery tools

Migration Evaluator is a migration assessment service that helps you create a directional business case for AWS cloud planning and migration. The information that the AWS Migration Evaluator collects includes server profile information (for example, OS, number of CPUs, amount of RAM), SQL Server metadata (for example, version and edition), utilization metrics, and network connections. AWS Application Discovery Service helps you plan cloud migration projects, by gathering information about your on-premises data centers. It discovers the connections between applications and servers to uncover unknown servers, better understand dependencies, and establish move groups.

Desired outcome: Gain comprehensive visibility into your infrastructure environment by collecting detailed server profiles, SQL Server configurations, utilization patterns, and application dependencies to create an accurate business case for cloud migration and optimize resource planning.

Common anti-patterns:

- Relying on manual inventory tracking and documentation, leading to incomplete or outdated infrastructure information and missed optimization opportunities.
- Making migration decisions based solely on static server specifications without considering actual utilization patterns and application dependencies.
- Planning cloud migrations in isolation without understanding the full scope of application relationships, resulting in overlooked servers and disrupted service connections.

Benefits of establishing this best practice:

- Accurate cost projections and resource planning through automated discovery of server configurations, SQL Server metadata, and utilization metrics.
- Reduced migration risks by identifying hidden dependencies and establishing appropriate move groups based on discovered application connections.
- Optimized infrastructure spend by right-sizing resources based on actual utilization patterns rather than assumptions or outdated documentation.

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

Implementation guidance

Implementing comprehensive infrastructure discovery through tools like AWS Migration Evaluator and AWS Application Discovery Service is crucial for successful cloud migrations and cost optimization. These tools automatically collect detailed information about server configurations, SQL Server deployments, resource utilization, and application dependencies, replacing error-prone manual tracking methods. This automated approach not only provides accurate data for building business cases and planning migrations but also helps organizations avoid the common pitfalls of oversizing resources or missing critical application connections, ultimately leading to more successful and cost-effective cloud deployments.

Implementation steps

- Deploy AWS Application Discovery Agent on target servers or configure AWS Application Discovery Agentless Collector for VMware environments
- Enable data collection in AWS Migration Hub

- Monitor and analyze collected data, including server profiles, utilization patterns, and application dependencies
- Generate reports and recommendations for migration business case, infrastructure requirements, migration waves, and resource optimization strategies

Resources

Related documents:

- Discover on-premises resources using AWS Migration Hub discovery tools

Related tools:

- Migration Evaluator

MSFTCOST01-BP02 Run assessment tools

AWS Optimization and Licensing Assessment (OLA) is a complimentary program for new and existing customers to assess and optimize current on-premises and cloud environments, based on actual resource utilization, third-party licensing, and application dependencies.

Desired outcome: The desired outcome of an AWS OLA is to significantly reduce costs and improve efficiency in both on-premises and cloud environments by optimizing resource utilization, streamlining third-party licensing, and understanding application dependencies, resulting in a more cost-effective and agile IT infrastructure aligned with business needs.

Common anti-patterns:

- Overprovisioning resources based on peak usage estimates rather than actual utilization data, leading to unnecessary costs and wasted capacity across both on-premises and cloud environments.
- Maintaining duplicate or redundant software licenses without understanding application dependencies and actual usage patterns, resulting in excessive licensing costs and complicated compliance management.

Benefits of establishing this best practice:

- Immediate cost reduction through the elimination of underutilized resources and redundant licensing, delivering significant savings on cloud and software spending.
- Enhanced operational efficiency through data-driven decision making, enabling better capacity planning and resource allocation based on actual usage patterns rather than assumptions.

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

Implementation guidance

To implement an AWS OLA, form a cross-functional team and collect comprehensive utilization data across all environments. Document software licenses and map application dependencies. Develop a prioritized optimization plan addressing immediate opportunities like resource right-sizing and license consolidation, followed by long-term strategic initiatives. Establish regular review cycles to ensure continuous optimization and alignment with business goals.

Implementation steps

1. Contact your AWS account manager or AWS Sales
2. Schedule an initial discovery meeting
3. Sign the OLA agreement
4. Provide access to required systems and data
5. Participate in data collection and assessment workshops
6. Review preliminary findings with AWS team
7. Receive and analyze final OLA report
8. Develop action plan based on recommendations
9. Schedule follow-up meetings for implementation support
10. Implement optimization strategies with AWS guidance

Resources

Related documents:

- AWS Optimization and Licensing Assessment
- AWS Prescriptive Guidance - OLA
- How to optimize costs for Microsoft workloads on AWS
- Reduce software licensing costs with an AWS Optimization and Licensing Assessment

Related videos:

- AWS re:Invent 2022 - How to save costs and optimize Microsoft workloads on AWS (ENT205)

MSFTCOST01-BP03 Run platform-specific tools

Platform-specific tools, such as Azure Resource Discovery Tool, are useful for environment understanding. This is a PowerShell script provided by AWS that generates an inventory report including detailed metrics of an Azure environment to which you have read access for the previous 30 days. Especially useful for non-virtual machine(VM) resources.

Desired outcome: Generate a comprehensive 30-day inventory report using platform-specific tools to provide detailed resource metrics, asset visibility, and usage patterns across the Azure environment, enabling informed decisions for cloud resource management and optimization, while documenting key metrics that would be essential for planning a potential AWS migration.

Common anti-patterns:

- Manual Resource Tracking: Relying solely on manual methods or spreadsheets to track cloud resources and their usage, instead of leveraging automated platform-specific tools. This approach is error-prone, time-consuming, and often results in incomplete or outdated information about the environment.
- One-Size-Fits-All: Using generic assessmetn tools that are not tailored to the specific cloud platform (in this case, Azure). This can lead to missed insights, inability to capture platform-specific metrics, and incomplete understanding of resource utilization and costs, especially for non-VM resources that may have unique characteristics in Azure.

Benefits of establishing this best practice:

- Comprehensive Resource Visibility: Platform-specific tools provide detailed, accurate insights into all resources within the Azure environment, including often overlooked non-VM resources. This comprehensive view enables better resource management, cost optimization, and capacity planning.
- Time and Effort Efficiency: Automated platform-specific tools can quickly generate detailed reports that would take significantly longer to compile manually. This efficiency allows IT teams to focus on analyzing the data and making strategic decisions rather than spending time on data collection and organization.

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

Implementation guidance

To implement this best practice, start by identifying and selecting appropriate platform-specific tools for Azure, such as Resource Discovery. Ensure you have the necessary read permissions across your Azure environment. Schedule regular automated runs of these tools, ideally on a monthly basis, to capture a rolling 30-day window of resource utilization. Set up a process to review and analyze the generated reports, focusing on resource allocation, usage patterns, and potential optimization opportunities. Integrate these insights into your cloud management and decision-making processes, and use the data to inform capacity planning, cost optimization strategies, and potential migration assessments. Regularly update and refine your use of these tools as your Azure environment evolves and as new features become available.

Implementation steps

1. Verify the required access permissions across all target Azure subscriptions and resource groups
2. Install and configure the chosen platform-specific tool (for example, Azure Resource Discovery Tool)
3. Save the output data
4. Contact your AWS account team to help analyzing Azure resources in preparation for potential AWS migration scenarios

Resources

Related tools:

- Azure Resource Discovery Tool

Operating system

Optimizing Windows EC2 costs is essential for managing cloud expenses effectively. AWS provides multiple approaches to reduce costs while maintaining performance, including instance right-sizing through AWS Compute Optimizer, automated scheduling for non-production workloads, and flexible licensing options like BYOL. By implementing these strategies, organizations can significantly reduce their Windows workload operating costs while leveraging AWS's scalability and flexibility.

MSFTCOST02: How do you save on Windows EC2 costs?

Amazon EC2 provides scalability, flexibility, and cost efficiency resources that can apply to Windows EC2 instances.

Best practices

- MSFTCOST02-BP01 Right size Windows instances
- MSFTCOST02-BP02 Automate stop and start schedules
- MSFTCOST02-BP03 Bring Your Own Licenses (BYOL)

MSFTCOST02-BP01 Right size Windows instances

AWS Compute Optimizer uses machine learning to analyze the performance metrics and utilization patterns of Microsoft workloads running on AWS, including Windows Server instances and SQL Server deployments. By examining historical resource usage data across CPU, memory, and network dimensions, Compute Optimizer provides tailored recommendations for right-sizing EC2 instances running Microsoft applications, helping organizations optimize both performance and cost. The service can identify when Windows workloads are over-provisioned or under-provisioned, suggesting instance types that are aligned with actual resource requirements. This is valuable for Microsoft-heavy enterprises that have migrated to AWS, as Windows workloads often have different resource consumption patterns and proper sizing is crucial for managing the additional licensing costs associated with Windows Server and SQL Server instances.

Desired outcome: Optimize Microsoft workload deployments on AWS to substantially reduce compute costs while maintaining or improving application performance through right-sized instances, resulting in lower Windows licensing fees and improved resource utilization metrics across CPU, memory, and network resources as validated by AWS Compute Optimizer recommendations.

Common anti-patterns:

- Deploying Microsoft workloads on the largest available EC2 instance types as a precautionary measure regardless of actual resource requirements, leading to severe over-provisioning and unnecessary Windows licensing costs for unused capacity.

- Ignoring AWS Compute Optimizer's recommendations and maintaining static instance sizes based on initial deployment configurations, even when utilization metrics consistently show periods of low resource usage or performance bottlenecks that indicate the need for right-sizing.

Benefits of establishing this best practice:

- Cost Efficiency: Organizations can eliminate resource waste by precisely matching instance types to actual workload requirements, reducing both EC2 instance costs and associated Microsoft licensing fees which are typically tied to instance size and processor count.
- Performance Optimization: Workloads receive the right balance of compute resources, preventing both performance bottlenecks from under-provisioning and excess capacity from over-provisioning, leading to consistent and reliable application performance for end users.
- Data-Driven Decision Making: IT teams can make instance sizing decisions based on machine learning-analyzed historical performance data rather than guesswork, reducing the operational overhead of manual monitoring and enabling proactive capacity planning for Microsoft workloads.

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

Implementation guidance

This implementation guide provides a high-level approach to leveraging AWS Compute Optimizer for Microsoft workload optimization on AWS. By following these best practices, organizations can establish a systematic process for analyzing and right-sizing their Windows-based instances while ensuring optimal performance and cost efficiency. The guide covers essential steps from initial Compute Optimizer activation and baseline assessment through to ongoing monitoring and adjustment phases. Whether you are running Windows Server applications, SQL Server databases, or other Microsoft workloads, these recommendations will help you implement a data-driven optimization strategy that aligns with both AWS architectural principles and Microsoft licensing considerations.

Implementation steps

1. Enable AWS Compute Optimizer and verify data collection across accounts
2. Create inventory of Microsoft workloads and licenses
3. Define performance baselines and thresholds for each workload type
4. Review initial optimization recommendations after 14-day analysis period

5. Create prioritized migration schedule for instance right-sizing
6. Execute instance changes during maintenance windows
7. Set up automated monitoring and reporting

Resources

Related documents:

- Right size Windows workloads
- Reduce Microsoft SQL Server licensing costs with AWS Compute Optimizer
- Optimizing your cost with rightsizing recommendations

Related tools:

- AWS Compute Optimizer

MSFTCOST02-BP02 Automate stop and start schedules

Leverage the Instance Scheduler on AWS to reduce the use of Amazon EC2 and Amazon Relational Database Service instances that do not need to run continuously. The Instance Scheduler helps reduce operational costs by stopping and starting resources as needed.

Desired outcome: Achieve significant cost reduction in non-production environments by implementing automated start/stop schedules for EC2 instances and RDS databases that are only required during business hours (for example, 8 AM - 6 PM on weekdays), while ensuring zero impact to business operations and development activities during working hours.

Common anti-patterns:

- Always-On Resources: Keeping all development, testing, and staging environments running 24/7, even when they're not actively used, resulting in unnecessary costs and resource waste.
- Manual Start/Stop Management: Relying on developers or operations teams to manually start and stop instances based on their work schedules, leading to inconsistent resource management, potential delays in availability, and increased risk of human error.

Benefits of establishing this best practice:

- Cost Optimization: Significant reduction in operational costs by automatically shutting down non-essential resources during off-hours, weekends, and holidays, directly impacting the organization's cloud spending.
- Operational Efficiency: Elimination of manual intervention for resource management, allowing IT teams to focus on more strategic tasks while ensuring consistent and reliable resource availability when needed.
- Environmental Impact: Reduced energy consumption and carbon footprint by minimizing unnecessary compute resource usage, supporting organizational sustainability goals and responsible cloud computing practices.

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

Implementation guidance

Begin by identifying and tagging non-production resources suitable for automated scheduling. Configure AWS Instance Scheduler with appropriate start/stop periods aligned with business hours and team schedules. Implement a gradual rollout strategy, starting with a small subset of resources to validate functionality. Establish monitoring mechanisms to track schedule execution and create override procedures for exceptional situations.

Implementation steps

1. Identify and tag non-production resources for scheduling
2. Install and configure AWS Instance Scheduler
3. Define business hours and create scheduling periods
4. Test schedule on pilot group of resources
5. Monitor and validate functionality
6. Roll out to remaining resources

Resources

Related documents:

- Automate stop and start schedules

Related tools:

- Instance Scheduler on AWS

MSFTCOST02-BP03 Bring Your Own Licenses (BYOL)

If you have already invested in licenses for your Microsoft workload, such as having enterprise licensing agreements, you can choose to bring your own licenses to AWS and save costs on EC2. Depending on when the licenses were acquired and the version, Windows Server licenses can be brought to Amazon EC2 Dedicated Hosts. Other products covered by Software Assurance and License Mobility in the agreement, like SQL Server, can be brought to default (shared) tenancy.

AWS License Manager provides the flexibility to convert between Bring Your Own License (BYOL) and License Included configurations, allowing you to optimize licensing costs based on your needs and eligibility. This conversion capability enables you to switch between license models without having to rebuild instances, making it easier to adapt to changing licensing requirements or to take advantage of different cost models. For more information on licensing options, see the Microsoft FAQ on the AWS public page, or contact your account team to help you engage with a Microsoft expert on AWS to guide you through the options.

Desired outcome: Successfully optimize costs and maintain compliance by leveraging existing Microsoft licenses through BYOL implementation on AWS, while ensuring seamless license management and flexibility to convert between license models as needed, resulting in documented cost savings and efficient resource utilization without service disruption.

Common anti-patterns:

- Misaligned license deployment: Incorrectly deploying Windows Server licenses on shared tenancy instead of required Dedicated Hosts, or placing SQL Server with software assurance on Dedicated Hosts when it could run on shared tenancy, resulting in unnecessary costs and compliance violations.
- Missed conversion opportunities: Failing to utilize AWS License Manager's conversion capabilities between BYOL and license included configurations, leading to unnecessary instance rebuilds and downtime when licensing requirements change or cost optimization opportunities arise.
- Independent license decision-making: Making BYOL decisions without consulting AWS account teams or Microsoft licensing experts, resulting in missed opportunities for cost savings, improper license mobility implementation, and potential compliance issues with enterprise agreements.

Benefits of establishing this best practice:

- By leveraging existing Microsoft licenses through BYOL, organizations can significantly reduce EC2 instance costs compared to License Included options. This maximizes the value of existing enterprise licensing agreements and allows for more efficient allocation of IT budgets.
- AWS License Manager's ability to convert between BYOL and License Included configurations provides unprecedented flexibility. This allows organizations to adapt quickly to changing business needs, licensing requirements, or cost structures without service interruptions or time-consuming instance rebuilds.
- Properly implementing BYOL with guidance from AWS and Microsoft experts ensures compliance with complex licensing terms. This minimizes the risk of unexpected costs or penalties during audits, while also ensuring that licenses are correctly applied to the right types of instances (for example, Windows Server on Dedicated Hosts, SQL Server on shared tenancy when applicable).

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

Implementation guidance

Start with a comprehensive audit of existing Microsoft licenses, then engage AWS account teams for expert guidance on BYOL implementation. Deploy AWS License Manager to track and manage licenses, ensuring proper instance placement (Dedicated Hosts versus shared tenancy) based on license terms. Regularly review and optimize configurations, maintaining thorough documentation for compliance purposes.

Implementation steps

1. Audit existing Microsoft licenses and enterprise agreements
2. Consult AWS and Microsoft experts for BYOL eligibility and options
3. Set up AWS License Manager for tracking and conversion capabilities
4. Deploy licenses correctly (for example, Windows Server on Dedicated Hosts, SQL Server on shared tenancy)
5. Establish regular review process for ongoing optimization and compliance

Resources

Related documents:

- Amazon Web Services and Microsoft FAQs
- Bring licenses for Windows and SQL Server workloads

Related tools:

- What is AWS License Manager?

Databases

Managing SQL Server costs effectively is crucial for organizations running Microsoft workloads on AWS. AWS provides various optimization strategies, from flexible licensing options and instance consolidation to edition right-sizing and modernization paths. Whether leveraging BYOL, exploring SQL Server on Linux, or considering migration to cloud-native alternatives like Aurora PostgreSQL with Babelfish, organizations can significantly reduce database costs while maintaining performance and reliability.

MSFTCOST03: How do you save on SQL Server costs?

AWS offers different options to run Microsoft SQL Server and leverage existing licenses.

MSFTCOST04: How do you modernize your Microsoft SQL Server workloads?

When embarking on a journey to modernize legacy databases for improved scalability, performance, and cost optimization, organizations often face challenges with commercial databases like SQL Server, which can be expensive, create vendor lock-in, and impose restrictive licensing terms. Migrating to open-source databases offers a viable solution, with options like Amazon Aurora PostgreSQL providing an attractive alternative. This transition allows companies to save on Windows and SQL Server licensing costs while benefiting from the flexibility and cost-effectiveness of open-source solutions. Cloud-native modern databases like Aurora combine these advantages with enterprise-grade features typically associated with commercial databases, offering a compelling compromise. By refactoring SQL Server databases to such platforms, organizations can achieve significant cost savings and enhanced performance without sacrificing robustness or reliability.

Best practices

- MSFTCOST03-BP01 Understand Microsoft SQL Server licensing and BYOL availability

- MSFTCOST03-BP02 Consolidate Microsoft SQL Server instances
- MSFTCOST03-BP03 Check if your workload is running with the right SQL Server edition
- MSFTCOST03-BP04 Evaluate SQL Server Developer edition
- MSFTCOST03-BP05 Evaluate SQL Server on Linux
- MSFTCOST03-BP06 Evaluate Optimize CPU feature
- MSFTCOST04-BP01 Use caching to enhance SQL Server workloads
- MSFTCOST04-BP02 Consider Babelfish for Amazon Aurora PostgreSQL
- MSFTCOST04-BP03 Consider purpose-built databases

MSFTCOST03-BP01 Understand Microsoft SQL Server licensing and BYOL availability

AWS offers a range of flexible cost optimization choices for licensing. These licensing options are designed to help you reduce costs, maintain compliance, and meet your business needs. AWS offers the license include option, which you can launch Windows EC2 instances with SQL Server installed and licensed on-demand, paying only for what you use. With the right requirements, you can also bring your own licenses to AWS, either to Amazon EC2 Dedicated Hosts or default (shared) tenancy.

Desired outcome: Optimize costs by thoroughly evaluating Microsoft SQL Server licensing options on AWS, including both the license-included model for on-demand usage and the Bring Your Own License (BYOL) approach for either Amazon EC2 Dedicated Hosts or default shared tenancy, ensuring that the chosen licensing strategy aligns with compliance requirements, maximizes cost savings, and effectively supports business objectives through AWS's flexible licensing framework.

Common anti-patterns:

- Automatically defaulting to license-included instances without analyzing BYOL cost benefits, potentially missing out on significant savings from existing Microsoft Enterprise Agreements or Software Assurance benefits that could be leveraged on AWS.
- Failing to properly track and document SQL Server deployments across different AWS environments, leading to over-provisioned licenses or compliance risks from unintentionally running SQL Server workloads on shared tenancy when BYOL requires dedicated hosts.
- Choosing licensing models based solely on immediate costs without considering long-term implications, such as selecting on-demand licensing when workloads are actually stable and predictable, resulting in higher total cost of ownership compared to BYOL options.

Benefits of establishing this best practice:

- Significant Cost Optimization: By carefully evaluating and implementing the most appropriate licensing model (BYOL versus license-included), organizations can achieve substantial cost savings through efficient license utilization, maximizing existing investments in Microsoft agreements, and aligning licensing costs with actual usage patterns.
- Enhanced Compliance and Risk Management: Proper licensing practices ensure continuous compliance with Microsoft's licensing terms and AWS's infrastructure requirements, reducing the risk of audit findings, unexpected true-up costs, and potential penalties while maintaining clear documentation of license deployment and usage.
- Improved Operational Flexibility: Understanding and implementing the right licensing strategy enables organizations to scale their SQL Server workloads more effectively, choose the most cost-effective deployment options (dedicated hosts versus shared tenancy), and maintain the agility to adjust licensing approaches as business needs evolve.

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

Implementation guidance

To implement effective Microsoft SQL Server licensing on AWS, start by inventorying existing licenses and analyzing workload patterns to determine the most cost-effective option between BYOL and license-included models. Establish clear documentation and tracking processes using AWS License Manager, and implement regular reviews to optimize costs while maintaining compliance with both Microsoft and AWS requirements.

Implementation steps

1. Conduct a comprehensive inventory of existing SQL Server licenses and associated rights (AWS OLA can be useful as well).
2. Analyze workload characteristics and usage patterns to determine the most cost-effective licensing model (BYOL versus license-included).
3. Set up AWS License Manager to track and manage SQL Server deployments across your AWS environment.
4. Implement a tagging strategy to accurately monitor and allocate SQL Server licensing costs.
5. Establish a regular review process to optimize licensing strategy and ensure ongoing compliance with Microsoft and AWS requirements.

Resources

Related documents:

- Understand SQL Server licensing
- Amazon Web Services and Microsoft FAQs

Related tools:

- What is AWS License Manager?

MSFTCOST03-BP02 Consolidate Microsoft SQL Server instances

A SQL instance is part of the SQL Server Database Engine, that provides the SQL service to clients or applications. It is common to have SQL instances installed per server when it comes to large production environments, to avoid resources issues and to follow resource governance. Although, for smaller or non-critical workloads organizations can leverage shared resources and have multiple SQL instances installed on the same server or set of servers. This approach will help your workload save costs on SQL licensing (less cores running SQL) and often in compute resources as well.

Desired outcome: Optimize costs and resource utilization by strategically consolidating Microsoft SQL Server instances, particularly for smaller or non-critical workloads. This approach aims to reduce SQL licensing expenses by minimizing the number of cores running SQL Server, while also potentially decreasing overall compute resource consumption. By carefully assessing workload requirements and identifying consolidation opportunities, organizations can achieve significant cost savings without compromising performance for mission-critical applications.

Common anti-patterns:

- Over-isolation: Deploying separate SQL Server instances for every application or workload, regardless of size or criticality, leading to unnecessary licensing costs and underutilized resources.
- Indiscriminate consolidation: Merging SQL Server instances without proper assessment of workload characteristics, resource requirements, and potential conflicts, resulting in performance degradation and operational issues for critical applications.

Benefits of establishing this best practice:

- Reduced Licensing Costs: By consolidating multiple SQL Server instances, particularly those running on less than 4 vCPUs, organizations can significantly reduce licensing expenses since SQL Server requires licensing for a minimum of 4 vCPUs per instance regardless of actual usage.
- Optimized Resource Utilization: Consolidation enables more efficient use of compute resources by sharing infrastructure across multiple workloads, reducing the total number of servers required and decreasing overall infrastructure costs.
- Simplified Management Overhead: Fewer SQL Server instances mean reduced administrative effort for maintenance, patching, backup management, and monitoring, leading to operational efficiency and lower management costs.

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

Implementation guidance

To implement SQL Server instance consolidation, assess workload patterns and resource requirements to identify compatible instances for merging, prioritizing non-critical workloads and especially targeting instances with less than 4 vCPUs since SQL Server requires licensing for a minimum of 4 vCPUs regardless of actual usage. Use AWS migration tools for consolidation, implement resource governance for effective management, and maintain regular performance monitoring to ensure optimal efficiency while reducing licensing costs.

Implementation steps

1. Conduct a comprehensive assessment of existing SQL Server instances, identifying workloads using less than 4 vCPUs and evaluating resource usage patterns, performance requirements, and compatibility.
2. Create a consolidation plan that groups compatible workloads, prioritizing non-critical applications and instances that can share resources without performance impact.
3. Implement database migrations between instances as needed, leveraging AWS Database Migration Service to facilitate the process.
4. Establish monitoring and review processes to track performance metrics, resource utilization, and cost savings of the consolidated environment.

Resources

Related documents:

- Consolidate instances

Related tools:

- AWS Database Migration Service

MSFTCOST03-BP03 Check if your workload is running with the right SQL Server edition

Microsoft offers types of SQL Server editions, with a different set of features and license cost. The Enterprise edition provides data center capabilities with high performance, unlimited virtualization, and several business intelligence tools. The Standard edition provides basic data management and business intelligence for smaller organizations and departments. The Web edition is suitable for companies that are web hosts or web value added providers (VAPs) and it should only be used to support public and internet accessible webpages, websites, and web services; its license does not allow the use for line-of-business applications. The Developer edition includes all functionality of the Enterprise edition, but it is intended for development purposes only. And the Express edition is a free database that can be used for learning or for building desktop applications. Over the release of SQL versions, Microsoft has added more features to the Standard edition, and it is not so unusual to see customers evaluating the downgrade from the Enterprise edition to the Standard one.

Desired outcome: The workload should run on the most cost-effective SQL Server edition that meets its functional and performance requirements. After evaluating feature usage, performance needs, and licensing costs across available editions (Enterprise, Standard, Web, Developer, and Express), the organization can confirm they are using the optimal edition or identify opportunities to downgrade to a more cost-effective edition without compromising workload functionality or performance targets.

Common anti-patterns:

- Enterprise by Default: Automatically deploying SQL Server Enterprise edition for all database workloads without analyzing actual feature requirements, resulting in unnecessary licensing costs for workloads that could run effectively on Standard or Web editions.
- Feature Underutilization: Paying for Enterprise edition licenses but only using features available in lower editions, such as using Enterprise solely for basic OLTP workloads without leveraging advanced features like in-memory OLTP, partitioning, or advanced security features.

Benefits of establishing this best practice:

- Cost Optimization: Significant cost savings through appropriate edition selection, particularly when downgrading from Enterprise to Standard edition where feasible. This can result in significant reduction in licensing costs while maintaining necessary functionality for workloads that do not require Enterprise-specific features.
- Resource Efficiency: Better alignment of database capabilities with actual workload requirements, ensuring resources are allocated efficiently and preventing overprovisioning of features that aren't being utilized. This leads to more streamlined database management and reduced operational overhead.
- Compliance and Risk Management: Appropriate edition selection ensures compliance with licensing terms—particularly critical for Web edition restrictions—while maintaining suitable feature sets for different environments. This reduces both compliance risks and potential audit findings.

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

Implementation guidance

Conduct a thorough assessment of your SQL Server workloads using built-in monitoring tools to identify actual feature usage and performance requirements. Compare these against available edition features and costs, using AWS Prescriptive Guidance to evaluate potential downgrades. Test thoroughly in non-production environments before implementing any edition changes, and establish a regular review process to ensure ongoing optimization.

Implementation steps

1. Audit current SQL Server workloads for feature usage and performance requirements using proper tools or scripts.
2. Compare workload needs against features available in different SQL Server editions, using AWS Prescriptive Guidance for potential downgrade scenarios.
3. Test workload performance on proposed new editions in non-production environments to validate functionality and performance.
4. Implement edition changes in a phased approach, starting with non-critical workloads, and establish a regular review process for ongoing edition optimization.

Resources

Related documents:

- Compare SQL Server editions
- Evaluate downgrading Microsoft SQL Server from Enterprise edition to Standard edition on AWS

MSFTCOST03-BP04 Evaluate SQL Server Developer edition

SQL Server Developer edition includes all functionality of the Enterprise edition. It is a free edition that can be used in non-production environments. A production environment is defined as an environment that is accessed by the end users of an application (for example, a website) and is used for more than gathering feedback or acceptance testing of that application. The Developer edition can be leveraged for development and testing your workload.

Desired outcome: By evaluating and implementing SQL Server Developer edition in non-production environments, the organization aims to reduce licensing costs while maintaining full Enterprise edition functionality for development and testing purposes. This change will optimize costs without compromising the ability to develop and test workloads effectively, ensuring that production environments remain properly licensed while development environments leverage the free Developer edition.

Common anti-patterns:

- Using SQL Server Developer edition in production environments to save costs, exposing the organization to licensing compliance issues and violating Microsoft's terms of use while putting end-user applications at risk.
- Maintaining Enterprise edition licenses across all development and testing environments without evaluating Developer edition alternatives, resulting in unnecessary licensing costs and inefficient resource allocation for non-production workloads.

Benefits of establishing this best practice:

- Significant cost savings: By implementing SQL Server Developer edition in non-production environments, organizations can substantially reduce licensing costs, as Developer edition is free for use in development and testing scenarios.
- Full feature access for development: Teams gain access to all Enterprise edition features in their development and testing environments, ensuring that they can build and test applications using the full range of SQL Server capabilities without incurring additional costs.

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

Implementation guidance

Identify non-production SQL Server instances, create a migration plan for downgrading to Developer edition, and implement controls to ensure Developer edition is only used in development and testing environments while maintaining proper licensing compliance.

Implementation steps

1. Inventory all SQL Server instances, identifying non-production environments.
2. Develop a migration plan for downgrading eligible instances to Developer edition.
3. Implement the downgrade process following AWS documentation.
4. Test applications thoroughly in the downgraded environments.
5. Implement controls and monitoring to prevent Developer edition use in production.

Resources

Related documents:

- Evaluate SQL Server Developer edition
- How to manually downgrade SQL Server Enterprise edition to Developer edition on AWS and save on licensing costs
- Automate downgrading SQL Server to Developer edition on Amazon EC2

MSFTCOST03-BP05 Evaluate SQL Server on Linux

Beginning on SQL Server 2017, Microsoft offers the option to run SQL Server on Linux operating systems. SQL Server on Linux is enterprise ready and offers flexibility, high performance, security features, reduced TCO, HA/DR features, and a great user experience. You can switch from SQL Server on Windows Server to SQL Server on Linux to save on Windows Server licensing costs.

Desired outcome: Successfully migrate compatible SQL Server workloads from Windows Server to Linux, resulting in reduced Total Cost of Ownership (TCO) through elimination of Windows Server licensing costs. This migration would maintain enterprise-level performance, security, and high availability features while leveraging the flexibility of SQL Server on Linux, ultimately optimizing costs for Microsoft workloads in the organization's IT infrastructure.

Common anti-patterns:

- Automatic Migration Without Compatibility Assessment: Organizations hastily migrating SQL Server workloads to Linux without first evaluating compatibility, resulting in application failures, performance issues, and potential data loss due to unsupported features or incompatible dependencies.
- Ignoring Total Cost of Operation: Companies focusing solely on the potential licensing cost savings of moving to SQL Server on Linux, while overlooking other operational costs such as retraining staff, modifying existing scripts and tools, and potential performance tuning needed in the new environment. This narrow focus may lead to unexpected expenses and operational challenges that offset the intended cost savings.

Benefits of establishing this best practice:

- Cost Optimization: Elimination of Windows Server licensing fees significantly reduces Total Cost of Ownership (TCO), enabling better resource allocation across the organization.
- Simplified Cross-Platform Management: Standardization of database management across Windows and Linux platforms reduces complexity and streamlines operational processes.

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

Implementation guidance

To implement SQL Server on Linux evaluation, start with a comprehensive workload compatibility assessment. Create a detailed migration plan including testing and rollback procedures. Conduct a pilot migration on a non-critical workload. Train IT staff on Linux and SQL Server on Linux management. Implement the full migration in stages, closely monitoring performance and functionality throughout to ensure a smooth transition and achieve cost savings and management simplification.

Implementation steps

1. Conduct workload compatibility assessment to identify SQL Server instances suitable for Linux migration, reviewing feature requirements and dependencies
2. Develop migration plan with testing procedures, success metrics, and rollback strategy, including pilot test selection
3. Implement pilot migration on selected non-critical workload while providing Linux administration training to IT staff

4. Implement phased migration of remaining workloads following successful pilot, with continuous monitoring of performance and costs

Resources

Related documents:

- Evaluate SQL Server on Linux
- Editions and supported features of SQL Server 2022 on Linux

Related tools: Windows to Linux replatforming assistant for Microsoft SQL Server Databases

MSFTCOST03-BP06 Evaluate Optimize CPU feature

Optimize CPUs for Amazon EC2 Instances provides customers greater control of your EC2 instances on two fronts. First, you can specify a custom number of vCPUs to save on vCPU-based licensing costs (such as SQL Server). Second, you can disable multithreading for workloads that perform well with single-threaded CPUs, like certain high-performance computing (HPC) applications. Reducing the number of vCPUs or disabling multithreading offers fewer cores to your SQL Server workload on EC2 without affecting the other compute resources available to the machine (such as RAM and storage), and most of the time without compromising the workload performance. This feature is available for both Bring Your Own License (BYOL) and License Included (LI) deployments.

Desired outcome: By evaluating and implementing CPU optimization for Amazon EC2 instances, we aim to reduce vCPU-based licensing costs for SQL Server while maintaining workload performance. This will be achieved by specifying custom vCPU counts and, where appropriate, disabling multithreading. The outcome will be a more cost-effective utilization of resources, particularly for SQL Server workloads, without compromising on performance or available compute resources such as RAM and storage.

Common anti-patterns:

- Over-provisioning vCPUs: Organizations often provision EC2 instances with more vCPUs than necessary for their SQL Server workloads, believing that more is better. This leads to unnecessarily high licensing costs for vCPU-based software like SQL Server, without providing any tangible performance benefits. The excess vCPUs remain unused while still incurring licensing fees.

- Ignoring multithreading optimization: Many teams leave multithreading enabled by default for all workloads, including those that do not benefit from it (such as certain HPC applications or single-threaded workloads). This can result in suboptimal performance for these specific workloads and potentially higher licensing costs, as some software is licensed per logical processor rather than physical core.

Benefits of establishing this best practice:

- Reduced SQL Server licensing costs through optimized vCPU allocation and core usage, resulting in direct cost savings.
- Better workload performance by matching CPU configurations to specific needs, especially for single-threaded applications and HPC workloads.

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

Implementation guidance

Start by analyzing your SQL Server workload performance and resource utilization. Use AWS tools like CloudWatch to gather metrics. Identify instances where you can reduce vCPU count without impacting performance. Set instances with custom vCPU counts to match your licensing. Test disabling multithreading for workloads that benefit from single-threaded performance. Monitor performance closely after changes to ensure workload efficiency is maintained. Regularly review and adjust your configurations as workload demands evolve.

Implementation steps

1. Analyze current SQL Server workload performance and resource utilization using AWS CloudWatch and other monitoring tools.
2. Set EC2 instances with custom vCPU counts that align with your SQL Server licensing and workload requirements.
3. Test and implement multithreading disablement for workloads that perform better with single-threaded CPUs, monitoring performance before and after the change.

Resources

Related documents:

- CPU options for Amazon EC2 instances

- Optimize CPUs for License-Included instances
- Optimize CPU best practices for SQL Server workloads
- Optimize CPUs best practices for SQL Server workloads – continued

MSFTCOST04-BP01 Use caching to enhance SQL Server workloads

Caching in .NET applications reduces costs and improves performance by storing frequently accessed data, lowering the load on backend databases like SQL Server. While especially useful for read-heavy operations, choose a caching method that fits your needs, considering that local caching has scalability limitations. Evaluate the trade-off between performance gains and caching costs when implementing your strategy.

Desired outcome: Implement an effective caching strategy for our .NET applications with SQL Server backends to reduce database load, cut costs, and improve performance. This tailored approach will optimize workloads and initiate our application modernization efforts, balancing performance gains with implementation costs.

Common anti-patterns:

- Over-Caching: Caching all data indiscriminately without considering data volatility or access patterns. This leads to stale data issues, increased memory consumption, and potentially higher costs than direct database queries would incur.
- Local Cache Sprawl: Implementing isolated local caches across multiple application instances without a coherent invalidation strategy, resulting in data inconsistencies, increased maintenance overhead, and poor scalability in distributed environments.

Benefits of establishing this best practice:

- Reduced load on SQL Server instances leads to lower database sizing requirements and decreased infrastructure costs, as fewer resources are needed to handle the same workload volume.
- Faster application response times through immediate access to cached data, eliminating repetitive database queries and reducing network latency for frequently accessed information.

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

Implementation guidance

Identify frequently accessed, static data for caching. Use distributed caching solutions like Redis for scalability. Implement robust cache invalidation to maintain data freshness. Apply cache-aside pattern for seamless database fallback. Regularly monitor cache hit rates and performance metrics to optimize your strategy as your application evolves.

Implementation steps

1. Configure a distributed cache service (like Redis) and integrate it with your .NET application using appropriate client libraries and connection strings
2. Implement cache-aside pattern in your data access layer, wrapping database calls with cache checks and updates using appropriate timeouts and invalidation logic
3. Set up monitoring for cache performance metrics (hit rates, memory usage, latency) using Application Insights or similar tools to validate and optimize caching effectiveness

Resources

Related documents:

- Use caching to reduce database demand

MSFTCOST04-BP02 Consider Babelfish for Amazon Aurora PostgreSQL

Babelfish is an Amazon Aurora PostgreSQL feature that allows SQL Server client applications to connect directly to PostgreSQL databases. It works by understanding SQL Server's TDS protocol and common SQL statements, providing a dedicated endpoint for SQL Server connections. This functionality enables organizations to migrate from SQL Server to PostgreSQL while maintaining application compatibility and minimizing the need for code modifications.

Desired outcome: By implementing Babelfish for Amazon Aurora PostgreSQL, we aim to efficiently migrate our SQL Server-based applications while minimizing development effort, reducing migration costs, and maintaining application compatibility. This will enable a smoother transition to PostgreSQL without extensive code refactoring.

Common anti-patterns:

- Complete rewrite approach: Unnecessarily rewriting entire applications to migrate from SQL Server to PostgreSQL, instead of leveraging Babelfish's compatibility features. This approach

often leads to extended project timelines, increased costs, and potential introduction of new bugs.

- Ignoring dialect differences: Assuming full SQL Server compatibility and neglecting to test and adjust for specific T-SQL features not supported by Babelfish. This can result in unexpected behavior or errors in production after migration.

Benefits of establishing this best practice:

- Reduces costs and complexity by minimizing code changes when transitioning from SQL Server to PostgreSQL, accelerating the migration process.
- Enables quick transition to Amazon Aurora PostgreSQL while maintaining application compatibility, allowing organizations to leverage cloud benefits with minimal disruption.

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

Implementation guidance

Begin by identifying SQL Server applications suitable for Babelfish migration and assess their T-SQL compatibility using the Babelfish Compass tool. Create a test environment to validate application functionality with Babelfish-enabled Amazon Aurora PostgreSQL cluster, focusing on critical database operations and stored procedures. Implement the migration in phases, starting with non-critical applications, and maintain detailed documentation of any required code adjustments or workarounds for unsupported features.

Implementation steps

1. Assess application compatibility using the Babelfish Compass tool or the AWS Schema Convertion Tool.
2. Set up a Babelfish-enabled Amazon Aurora PostgreSQL cluster and configure the TDS listener port.
3. Modify application connection strings to point to the Babelfish endpoint instead of the SQL Server instance.
4. Test thoroughly, focusing on critical database operations, stored procedures, and application functionality before migrating production workloads.

Resources

Related documents:

- Using Babelfish for Aurora PostgreSQL
- Migrating a SQL Server database to Babelfish for Aurora PostgreSQL
- Prepare for Babelfish migration with the AWS SCT assessment report

Related tools:

- Babelfish Compass
- What is the AWS Schema Conversion Tool?

MSFTCOST04-BP03 Consider purpose-built databases

Purpose-built databases are gaining traction among businesses adopting modern architectures like microservices, as they can precisely accommodate specific data access patterns. These databases, whether SQL or NoSQL, offer application teams benefits like reduced costs, enhanced scalability, and improved resilience. By selecting the right database for each specific use case, teams can optimize their data management while leveraging cloud advantages for more efficient solutions.

Desired outcome: By adopting purpose-built databases tailored to specific workload requirements, application teams will optimize data management, reduce costs, and enhance scalability and resilience. This approach will lead to more efficient cloud-based solutions, whether using SQL or NoSQL databases, and minimize undifferentiated heavy lifting in database operations.

Common anti-patterns:

- Using a single database technology for all applications and workloads, regardless of their specific data access patterns or requirements. This can lead to suboptimal performance, scalability issues, and increased costs.
- Adopting multiple specialized databases for every minor variation in data needs, resulting in a fragmented and overly complex data architecture that increases management overhead and potentially negates cost savings.

Benefits of establishing this best practice:

- Purpose-built databases are designed to handle specific data access patterns, resulting in improved query performance and overall system efficiency tailored to each application's unique needs.
- By choosing databases that align closely with workload requirements, organizations can avoid over-provisioning resources and reduce unnecessary licensing costs associated with general-purpose database solutions.
- Purpose-built databases often come with built-in features for horizontal scaling and high availability, making it easier for applications to handle growth in data volume and user traffic while maintaining robust performance and reliability.

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

Implementation guidance

Begin by analyzing your application's data access patterns, query requirements, and scalability needs. Identify distinct workload characteristics across your application portfolio and map them to appropriate purpose-built database solutions. For new applications, design the data architecture with these specific requirements in mind from the start. For existing applications, consider a phased migration approach, starting with the components that would benefit most from purpose-built databases. Evaluate AWS purpose-built database options such as Amazon DynamoDB for high-performance NoSQL needs, Amazon RDS for traditional relational workloads, or Amazon Redshift for data warehousing, ensuring each choice aligns with both technical requirements and cost optimization goals.

Implementation steps

1. Analyze current data access patterns and requirements across your application portfolio
2. Identify suitable purpose-built database solutions for each distinct workload
3. Develop a migration strategy, prioritizing components that will benefit most from the change
4. Implement and test the new database solutions in a staging environment
5. Gradually migrate production workloads, monitoring performance and costs throughout the process

Resources

Related documents:

- Consider purpose-built databases

Related tools:

- What is the AWS Schema Conversion Tool?

Storage

Optimizing storage costs is essential for Microsoft workloads on AWS, as storage represents a significant portion of infrastructure expenses. AWS offers various cost-effective storage solutions, from newer generation EBS volumes (gp3) to fully managed services like Amazon FSx for Windows File Server and FSx for ONTAP. By implementing proper lifecycle management for volumes and snapshots, and choosing the right storage solutions for specific workload requirements, organizations can significantly reduce storage costs while maintaining or improving performance.

MSFTCOST05: How do you save on storage for your Microsoft workload?

The storage layer is a critical architecture component for most applications, including Microsoft workloads. Exploring the compatible Amazon storage offers can help you provide the required performance to your workloads and save costs. Constantly managing storage resources avoids unused resources, over-provisioning, and keeps the workloads performant.

Best practices

- MSFTCOST05-BP01 Migrate Amazon EBS volumes from gp2 to gp3
- MSFTCOST05-BP02 Control Amazon EBS volumes or snapshots lifecycle
- MSFTCOST05-BP03 Use Amazon FSx for NetApp ONTAP
- MSFTCOST05-BP04 Use Amazon FSx for Windows File Server

MSFTCOST05-BP01 Migrate Amazon EBS volumes from gp2 to gp3

General Purpose SSD (gp3) volumes are the latest generation of General Purpose SSD volumes, and the lowest cost SSD volume offered by Amazon EBS. This volume type helps to provide the right balance of price and performance for most applications. It also helps you to scale volume performance independently of volume size. This means that you can provision the required performance with no need to provision additional block storage capacity. Additionally, gp3 volumes offer a 20 percent lower price per GiB than General Purpose SSD (gp2) volumes.

Desired outcome: By migrating Amazon EBS volumes from gp2 to gp3, we aim to achieve a 20% reduction in storage costs while gaining the ability to independently scale volume performance without increasing capacity. This transition will optimize our storage expenses and provide better performance control for our workloads, ultimately resulting in improved cost efficiency without compromising performance requirements.

Common anti-patterns:

- Some organizations maintain large gp2 volumes to achieve higher IOPS, unnecessarily increasing costs. They fail to recognize that gp3 volumes allow separate scaling of IOPS and throughput, potentially leading to significant overspending on storage.
- Some teams might hastily migrate all gp2 volumes to gp3 without analyzing workload-specific performance needs. This can result in performance degradation for applications that require higher baseline performance than what's provided by the default gp3 configuration, leading to potential application issues and the need for retroactive adjustments.

Benefits of establishing this best practice:

- Immediate 20% reduction in per-GiB storage costs compared to gp2 volumes, leading to significant cost savings across the storage infrastructure, especially for large-scale deployments with multiple volumes.
- Independent scaling of IOPS and throughput without increasing volume size, allowing precise performance tuning based on application needs while avoiding unnecessary storage capacity expenses.
- The ability to maintain smaller volume sizes while still achieving desired performance levels eliminates the need to overprovision storage for performance reasons, resulting in more efficient resource utilization and easier capacity management.

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

Implementation guidance

Begin by identifying all existing gp2 volumes using AWS Cost Explorer or AWS Systems Manager. Create a phased migration plan, prioritizing non-production environments first to validate the performance impact. Use AWS CloudFormation templates or AWS CLI scripts to automate the modification process, ensuring each volume's baseline performance requirements are properly configured during the transition to gp3. Monitor performance metrics through Amazon

CloudWatch before and after migration to verify that application performance remains optimal. Include snapshot backups in your migration strategy as a rollback mechanism, and schedule migrations during low-traffic periods to minimize potential impact on business operations.

Implementation steps

1. Conduct an inventory analysis using AWS Cost Explorer and Systems Manager to identify all gp2 volumes, documenting their current size, IOPS requirements, and associated workloads.
2. Create automated scripts using AWS CLI or Infrastructure as Code (IaC) to modify volumes from gp2 to gp3, including proper configuration of baseline IOPS and throughput based on historical performance data.
3. Implement the migration in phases, starting with development and test environments, followed by non-critical production workloads, and finally business-critical applications, with performance validation at each stage.
4. Monitor post-migration performance using Amazon CloudWatch metrics and establish a feedback loop to adjust gp3 volume configurations as needed, ensuring optimal performance while maintaining cost savings.

Resources

Related documents:

- Amazon EBS General Purpose SSD volumes
- Migrate Amazon EBS volumes from gp2 to gp3

MSFTCOST05-BP02 Control Amazon EBS volumes or snapshots lifecycle

EBS snapshots are incremental backups stored in S3, saving only changed blocks since the last snapshot. They can backup unattached volumes before deletion. Two storage tiers available: Standard (higher storage cost and free retrieval) and Archive (lower storage cost and paid retrieval). Managing snapshot lifecycles and removing unused volumes helps optimize costs.

Desired outcome: Implement an effective EBS volume and snapshot management strategy that automatically identifies and removes unused volumes while maintaining cost-efficient snapshot lifecycles across appropriate storage tiers, resulting in optimized storage costs for Microsoft workloads on AWS.

Common anti-patterns:

- Neglecting to delete unattached EBS volumes: Keeping unused volumes active, leading to unnecessary ongoing storage costs for resources that are no longer needed.
- Inconsistent or manual snapshot management: Relying on manual processes or ad-hoc scripts for creating and managing snapshots, leading to inconsistent backup coverage, potential data loss, and inefficient use of storage resources.

Benefits of establishing this best practice:

- By systematically managing EBS volumes and snapshots, you can significantly reduce storage costs by removing unused resources and efficiently tiering snapshots based on access needs.
- Regular lifecycle management ensures that your backup strategy is consistent and up-to-date, reducing the risk of data loss and maintaining appropriate retention periods for compliance and disaster recovery purposes.

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

Implementation guidance

To effectively control EBS volume and snapshot lifecycles, start by implementing automated tools such as AWS Data Lifecycle Manager. Configure policies to regularly identify and delete unattached volumes, create consistent snapshot schedules, and manage snapshot retention across appropriate storage tiers. Use tags to categorize resources and enable granular control. Regularly review and adjust your policies to ensure they align with changing business needs and cost optimization goals. Implement monitoring and alerting to track resource usage and potential cost savings opportunities.

Implementation steps

1. Set up AWS Data Lifecycle Manager policies to automate snapshot creation and deletion based on defined schedules and retention rules.
2. Implement a tagging strategy to categorize EBS volumes and snapshots, enabling easier management and cost allocation.
3. Create an automated process to identify and alert on unattached EBS volumes, with an option to delete them after a specified period.
4. Establish a tiering policy to move infrequently accessed snapshots from Standard to Archive tier after a set duration to optimize storage costs.

Resources

Related documents:

- Modify Amazon EBS snapshots
- Delete unattached Amazon EBS volumes

MSFTCOST05-BP03 Use Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP offers a file system that supports SMB and iSCSI protocols. Useful for critical Microsoft SQL Server environments, as ONTAP volumes can be mapped to Windows Server instances as block storage devices using the iSCSI model, also providing shared storage for cluster-aware applications. FSx for ONTAP has two capacity settings (HDD and SSD), data deduplication, and cache layers. Smaller EC2 instances can leverage the FSx solution to achieve high performance storage levels.

Desired outcome: By implementing Amazon FSx for NetApp ONTAP, an organization can achieve a highly available and performant storage solution for Microsoft workloads. The implementation will leverage both SMB and iSCSI protocols, enabling efficient block storage access while benefiting from advanced features like data deduplication and multi-tiered caching. This will result in optimized storage costs, improved performance even with smaller EC2 instances, and reduced operational overhead for managing Microsoft workloads.

Common anti-patterns:

- Running Microsoft SQL Server or other Microsoft workloads with directly attached EBS volumes may limit high availability and scalability, making failover scenarios complex and time-consuming. This approach also lacks the advanced storage management features and efficiency benefits provided by FSx for ONTAP, potentially leading to higher costs and operational overhead.
- Compensating for storage performance requirements by using oversized EC2 instances with local storage or multiple EBS volumes, rather than leveraging FSx for ONTAP's efficient storage architecture. This results in unnecessary compute costs and doesn't address the underlying need for enterprise-grade storage features like deduplication and efficient snapshots.

Benefits of establishing this best practice:

- FSx for ONTAP provides high-performance storage, allowing even small EC2 instances to achieve excellent I/O capabilities.
- Easily scale storage capacity and performance independently of compute resources, adapting to changing workload demands.
- Reduce management overhead with built-in features like data deduplication, snapshots, and multi-protocol support.

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

Implementation guidance

To implement FSx for ONTAP for Microsoft workloads, create an FSx file system in your VPC. Configure SVMs and volumes for your applications, setting up SMB shares and iSCSI LUNs as needed. Connect Windows instances to these resources using native tools. For high availability, use Windows Server Failover Clustering with FSx as shared storage. Migrate your data, then update backup and recovery processes to leverage FSx features like snapshots and replication.

Implementation steps

1. Create FSx for ONTAP file system within your VPC, configuring the appropriate storage capacity and throughput based on workload requirements
2. Set up Storage Virtual Machines (SVMs) and configure storage volumes with proper protocols (SMB/iSCSI) based on your Microsoft application needs
3. Connect Windows Server instances to FSx storage using native tools (File Explorer for SMB, iSCSI Initiator for block storage)
4. Configure Windows Server Failover Clustering if high availability is required, using FSx for ONTAP as the shared storage
5. Migrate existing data to FSx storage and implement backup/recovery procedures using ONTAP's snapshot and replication capabilities

Resources

Related documents:

- What is Amazon FSx for NetApp ONTAP?
- Provisioning iSCSI for Windows

MSFTCOST05-BP04 Use Amazon FSx for Windows File Server

Amazon FSx for Windows File Server is a fully managed file storage service that's optimized for Microsoft workloads. It provides an SMB file system that can be accessed by applications, including Windows web servers and Microsoft SQL Server. FSx for Windows File Server is a scalable solution that offers Single AZ or Multi AZ availability, automatic data deduplication, different pricing options, and two capacity settings (HDD and SSD), being flexible to fit your Microsoft workloads. Fairly small EC2 instances can leverage the FSx solution to achieve high performance storage levels.

Desired outcome: Implement Amazon FSx for Windows File Server to optimize storage for Microsoft workloads, reducing operational overhead while enhancing scalability and performance. This change aims to improve efficiency, simplify management, and potentially reduce costs associated with file storage for Windows-based applications in AWS.

Common anti-patterns:

- Running Windows file servers on EC2 instances with attached EBS volumes, requiring manual management of storage capacity, backups, and Windows Server maintenance while incurring higher operational costs and complexity.
- Using non-Windows-optimized storage solutions for Windows workloads, resulting in compatibility issues, degraded performance, and the need for additional software or configurations to handle SMB protocol requirements.

Benefits of establishing this best practice:

- Eliminates the need for manual file server management, Windows patching, and backup administration through AWS's fully managed service.
- Provides high-performance storage with automatic capacity management and the ability to scale up or down based on workload demands, while supporting both Single-AZ and Multi-AZ deployments.
- Offers flexible storage options (HDD/SSD) and pricing models, allowing organizations to align costs with actual needs while eliminating the overhead of maintaining dedicated Windows file servers and associated licenses.

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

Implementation guidance

Begin by assessing your current Windows workload requirements, including storage capacity, performance needs, and availability requirements. Choose between Single-AZ or Multi-AZ deployment based on your reliability needs, and select the appropriate storage type (HDD for general purpose or SSD for performance-intensive workloads). Start with a pilot migration of a non-critical workload to validate the setup and performance. Configure your existing Windows applications and services to connect to the FSx file system using standard SMB protocol, and implement appropriate security groups and Active Directory integration. Once validated, proceed with a phased migration approach for remaining workloads while monitoring performance metrics through CloudWatch.

Implementation steps

1. Assess workload requirements and select appropriate FSx configuration (Single/Multi-AZ, HDD/SSD, and storage capacity) based on performance needs and budget constraints.
2. Configure network security by setting up VPC security groups, ensuring proper routing, and establishing Active Directory integration for authentication.
3. Migrate existing file data to FSx using AWS DataSync or standard file copy tools, validating data integrity and permissions post-migration.
4. Update application configurations to point to the new FSx file share endpoints and verify connectivity, performance, and functionality across all dependent services.

Resources

Related documents:

- Amazon FSx for Windows File Server

Related tools:

- AWS DataSync

Active directory

Active Directory is a critical component for many Microsoft workloads, and optimizing its deployment on AWS can lead to significant cost savings and operational efficiencies. AWS offers

multiple options to support Active Directory needs, including AWS Managed Microsoft AD, AD Connector, and self-managed Active Directory on EC2. Each option provides different benefits in terms of management overhead, scalability, and cost. By evaluating these options against specific workload requirements, organizations can choose the most cost-effective and operationally efficient Active Directory solution for their AWS environment.

MSFTCOST06: How do you save on Active Directory for your Microsoft workload?

Microsoft Active Directory has been a widely used identity management solution in Windows networks for decades. It delivers authentication and access protocols, such as LDAP and Kerberos. When deploying Active Directory on AWS, consider the options to make sure you are saving on direct or operational costs for your Microsoft workload

Best practices

- MSFTCOST06-BP01 Use AWS Managed Microsoft Active Directory
- MSFTCOST06-BP02 Use AD Connector
- MSFTCOST06-BP03 Use self-managed Active Directory on Amazon EC2

MSFTCOST06-BP01 Use AWS Managed Microsoft Active Directory

AWS Managed Microsoft AD provides redundant Windows Server domain controllers across two Availability Zones in your VPC. It handles all maintenance tasks automatically, including monitoring, replication, backups, and updates. The service supports directory-aware workloads like SharePoint and .NET applications, and can integrate with on-premises Active Directory through trust relationships. Based on size requirements, you can either choose the Standard or the Enterprise edition.

Desired outcome: By implementing AWS Managed Microsoft Active Directory, you aim to reduce operational overhead and costs associated with managing Active Directory in the AWS Cloud. This solution provides a highly available, fully managed directory service that automatically handles maintenance tasks, supports directory-aware workloads, and enables integration with on-premises Active Directory. The result is a more efficient, scalable, and cost-effective approach to directory services in your cloud environment.

Common anti-patterns:

- Managing your own domain controllers on EC2 instances, resulting in unnecessary operational overhead from manual patching, backups, monitoring, and high availability configuration.
- Creating multiple standalone Active Directory environments across different workloads instead of using a centralized managed service, leading to increased costs and management complexity.

Benefits of establishing this best practice:

- Reduced operational overhead: AWS handles maintenance tasks such as patching, backups, and monitoring, freeing up IT staff to focus on core business activities.
- Improved reliability and availability: The service automatically deploys domain controllers across multiple Availability Zones, ensuring high availability and disaster recovery capabilities.
- Seamless integration: Enables easy connection with on-premises Active Directory through trust relationships, facilitating hybrid cloud scenarios and simplifying user access management across environments.

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

Implementation guidance

To implement AWS Managed Microsoft AD effectively, start by assessing your directory service needs and selecting the appropriate edition (Standard or Enterprise) based on your organization's size and requirements. Plan your VPC and network configuration to ensure proper connectivity for your domain controllers. If you have an existing on-premises Active Directory, establish a trust relationship to enable seamless integration. Migrate your directory-aware workloads gradually, beginning with less critical applications to gain experience and confidence. Leverage AWS IAM Identity Center for unified access management across your AWS environment. Finally, regularly review and optimize your setup to ensure it continues to meet your evolving needs while maximizing cost efficiency.

Implementation steps

1. Prepare your environment by configuring VPC with appropriate subnets across multiple AZs and setting up required network connectivity for hybrid scenarios
2. Deploy AWS Managed Microsoft AD by selecting the appropriate edition, choosing target VPC and subnets, and configuring directory administrator credentials and DNS settings
3. Establish connectivity and access through security group configuration, trust relationship setup with on-premises AD if needed, and AWS IAM Identity Center integration

4. Migrate workloads gradually, starting with test applications, followed by directory-aware workloads, while updating DNS settings and application configurations
5. Monitor and maintain the environment using CloudWatch metrics, managing directory users and groups, and performing regular access permission audits

Resources

Related documents:

- AWS Managed Microsoft AD
- How to migrate your on-premises domain to AWS Managed Microsoft AD using ADMT

MSFTCOST06-BP02 Use AD Connector

AD Connector is a directory gateway with which you can redirect directory requests to your on-premises Microsoft Active Directory without caching any information in the cloud. AD Connector comes in two sizes, small and large. A small AD Connector is designed for smaller organizations and is intended to handle a low number of operations per second. A large AD Connector is designed for larger organizations and is intended to handle a moderate to high number of operations per second. You can spread application loads across multiple AD Connectors to scale to your performance needs. There are no enforced user or connection limits.

Desired outcome: By implementing AD Connector, the organization may achieve seamless integration between on-premises Microsoft Active Directory and AWS Cloud services without data replication or cloud caching. The solution will be appropriately sized (small or large) based on operational requirements, with the flexibility to add multiple AD Connectors for increased performance as needed, ensuring cost-effective directory services management while maintaining security and scalability.

Common anti-patterns:

- Replicating the entire on-premises Active Directory to AWS using AWS Managed Microsoft AD or EC2 instances running AD Domain Controllers, when only authentication and authorization services are required, leading to increased attack surface, higher costs, and unnecessary data duplication.
- Implementing AWS Managed Microsoft AD when only directory authentication is needed, resulting in higher operational costs and unnecessary complexity when AD Connector could

provide the same functionality through simple proxying of authentication requests to the existing on-premises Active Directory.

Benefits of establishing this best practice:

- AD Connector eliminates the need for complex directory synchronization solutions or maintaining separate directory infrastructures in the cloud, reducing both capital and operational expenses associated with directory services management.
- By not storing or caching any directory information in the cloud, AD Connector minimizes the risk of data breaches and maintains a smaller attack surface, while still allowing AWS resources to leverage existing on-premises Active Directory for authentication and authorization.

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

Implementation guidance

To implement AD Connector effectively, start by assessing your organization's directory service needs and sizing requirements. Choose between small and large AD Connector options based on your expected operation volume. Ensure your on-premises Active Directory is properly configured and accessible from your AWS VPC through a secure connection, such as AWS Direct Connect or a VPN. Set up the AD Connector in your VPC, configure the necessary security groups, and test the connection thoroughly. For high availability, consider deploying AD Connectors in multiple Availability Zones. Finally, integrate your AWS resources and applications with AD Connector for seamless authentication and authorization using your existing Active Directory credentials.

Implementation steps

1. Establish network connectivity between AWS and on-premises environment through either AWS Direct Connect or AWS Site-to-Site VPN, ensuring proper routing and security group configurations are in place.
2. Deploy AD Connector in your VPC, selecting the appropriate size (small or large) based on your organization's authentication operation volume and configuring it with your on-premises Active Directory service account credentials.
3. Test AD Connector functionality by verifying authentication flows and ensuring proper communication between AWS resources and your on-premises Active Directory.
4. Enable and configure AWS services and applications to use AD Connector for authentication, including AWS Management Console access and AWS Enterprise Applications that support SAML 2.0.

Resources

Related documents:

- AD Connector

MSFTCOST06-BP03 Use self-managed Active Directory on Amazon EC2

Depending on the requirements, your Microsoft workload may require the use of a self-managed Active Directory deployment (either a new forest or extending an existing one). Deploying Active Directory domains controllers to Amazon EC2 is fairly simple. Domain controllers are usually good candidates to run on Amazon EC2 burstable instance family, saving on compute costs. Make sure to evaluate the capacity planning recommendations provided by Microsoft to address your workload requirements.

Desired outcome: Deploy self-managed Active Directory domain controllers on Amazon EC2, leveraging burstable instance families where appropriate to optimize costs. The implementation will follow Microsoft's capacity planning guidelines and support either new or extended Active Directory forests based on organizational requirements, ensuring a robust and cost-effective directory service solution.

Common anti-patterns:

- Deploying Active Directory domain controllers on oversized EC2 instances, such as using high-performance compute-optimized instances when not required. This leads to unnecessary costs and underutilized resources, contradicting the goal of cost optimization.
- Implementing Active Directory on EC2 without properly evaluating Microsoft's capacity planning recommendations. This can result in performance issues, scalability problems, and potential service disruptions, ultimately affecting the reliability of the Microsoft workload and potentially increasing long-term costs due to necessary remediation efforts.

Benefits of establishing this best practice:

- Cost-effective directory services through optimized EC2 instance selection and burstable computing, reducing operational expenses while maintaining performance requirements.
- Enhanced control and flexibility in deploying and managing Active Directory infrastructure, supporting both new implementations and extensions of existing directory services.

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

Implementation guidance

To implement self-managed Active Directory on Amazon EC2, begin by assessing your workload requirements and consulting Microsoft's capacity planning recommendations. Choose appropriate EC2 instance types, favoring burstable instances where suitable. Deploy at least two domain controllers across different Availability Zones for high availability. Use Amazon EBS volumes for storage, ensuring proper backups and snapshots. Implement security best practices, including network segmentation with VPCs and security groups. Finally, establish monitoring and alerting using Amazon CloudWatch to maintain optimal performance and availability of your Active Directory infrastructure.

Implementation steps

1. Size and deploy EC2 instances based on Microsoft's capacity planning guidelines, utilizing burstable instance families where appropriate, and ensure distribution across multiple Availability Zones for high availability.
2. Configure networking components including VPC design, security groups, and DHCP options to support Active Directory requirements and establish secure communication paths.
3. Install and configure Active Directory Domain Services, establishing either a new forest or extending an existing one, following Microsoft's recommended configurations and security baselines.
4. Implement monitoring and backup solutions using Amazon CloudWatch and automated EBS snapshots to maintain service health and enable disaster recovery capabilities.

Resources

Related documents:

- Self-managed Active Directory on Amazon EC2

Containers

Containerizing Windows workloads represents a significant opportunity to optimize infrastructure costs and improve operational efficiency. AWS provides comprehensive tools and services to support this transformation, such as advanced optimization tools like AWS Compute Optimizer for Fargate tasks and Kubecost for EKS environments. By leveraging these solutions along with

modern scaling strategies like Karpenter, organizations can reduce their Windows Server footprint while gaining the benefits of containerized deployments, including improved resource utilization and automated scaling.

MSFTCOST07: How do you save on Windows Server footprint moving to Windows Containers?

Containers can help optimize the infrastructure and also promote agility for your workloads. If compatible, applications that require components from the Windows OS may leverage the infrastructure modernization of running on Windows containers.

Best practices

- MSFTCOST07-BP01 Optimize AWS Fargate tasks with AWS Compute Optimizer
- MSFTCOST07-BP02 Improve Amazon Elastic Kubernetes Service cost tracking with Kubecost
- MSFTCOST07-BP03 Change your scale strategy for Windows Containers on Kubernetes using Karpenter

MSFTCOST07-BP01 Optimize AWS Fargate tasks with AWS Compute Optimizer

AWS Compute Optimizer can be used to help right size and optimize your workloads running on AWS Fargate. If not reviewed, long running tasks can be overprovisioned and increase compute costs.

Desired outcome: Aim to achieve optimal resource allocation and cost efficiency. Right-size our Fargate tasks, avoiding overprovisioning and reducing unnecessary compute costs. Through regular review and implementation of Compute Optimizer's recommendations, we expect to maintain well-optimized workloads that balance performance and cost-effectiveness, ultimately leading to improved resource utilization and significant cost savings in our AWS environment.

Common anti-patterns:

- Deploying Fargate tasks with initial resource configurations and never reviewing or adjusting them over time, ignoring Compute Optimizer's recommendations and missing significant cost optimization opportunities.

- Deliberately configuring Fargate tasks with excessive CPU and memory as a precautionary measure, leading to consistent overprovisioning and unnecessary costs despite Compute Optimizer showing opportunities for right-sizing.

Benefits of establishing this best practice:

- Regular review and implementation of Compute Optimizer recommendations leads to right-sized Fargate tasks, eliminating waste from overprovisioning and reducing monthly compute costs while maintaining required performance levels.
- Leveraging Compute Optimizer's ML-powered analytics provides objective, metrics-based insights for resource allocation decisions, replacing guesswork with actual usage patterns and ensuring optimal task configurations across your workloads.

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

Implementation guidance

Enable AWS Compute Optimizer for your organization or account. Establish a monthly review process for Fargate task recommendations. Implement changes gradually, starting with non-production workloads. Monitor performance before and after optimizations. Create a feedback loop to inform future deployments, and consider automating recommendation implementation through Infrastructure as Code practices for consistency and efficiency.

Implementation steps

1. Enable AWS Compute Optimizer in your environment through the AWS Management Console and verify it begins collecting task utilization data for analysis.
2. Schedule monthly review meetings with relevant stakeholders to assess Compute Optimizer's Fargate task recommendations and prioritize which optimizations to implement.
3. Create a change management process that includes testing optimizations in non-production environments first, with clear rollback procedures if needed.
4. Implement approved recommendations through your existing Infrastructure as Code (IaC) framework, ensuring changes are tracked and reproducible.
5. Set up CloudWatch dashboards or alerts to monitor performance metrics post-optimization, ensuring the changes maintain desired service levels while achieving cost savings.

Resources

Related documents:

- Optimize costs for AWS Fargate tasks on Amazon ECS

Related tools:

- What is AWS Compute Optimizer?

MSFTCOST07-BP02 Improve Amazon Elastic Kubernetes Service cost tracking with Kubecost

Kubecost improves the cost tracking for your Windows containers. Kubecost helps right sizing cluster nodes, container requests, and manages underutilized infrastructure.

Desired outcome: Aim to achieve improved cost tracking for our Windows containers. The desired outcome is to optimize cluster resource utilization through right-sizing of nodes and container requests, while effectively managing underutilized infrastructure. This implementation may provide better visibility into EKS costs, enabling more informed decision-making and ultimately leading to cost savings in Kubernetes deployments.

Common anti-patterns:

- Lack of cost monitoring tools, leading to untracked spending and no visibility into workload-specific costs across Amazon Elastic Kubernetes Service (EKS) clusters.
- Blindly overprovisioning Windows container resources without usage data, resulting in unnecessary infrastructure costs and resource waste.

Benefits of establishing this best practice:

- Gain detailed insights into container-level expenses, enabling accurate cost allocation across teams, projects, and workloads.
- Identify and right-size underutilized resources, leading to significant cost savings and improved cluster efficiency for Windows containers.

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

Implementation guidance

Start by deploying Kubecost into your environment. Configure it to integrate with your EKS cluster and AWS Cost and Usage Reports. Set up proper tagging for resources to ensure accurate cost allocation. Regularly review Kubecost dashboards to identify cost-saving opportunities, such as right-sizing nodes and optimizing container requests. Use Kubecost's recommendations to adjust resource allocations and implement cost controls. Continuously monitor and refine your cost optimization strategy based on the insights provided by Kubecost.

Implementation steps

1. Deploy Kubecost to your EKS clusters, ensuring proper IAM roles and permissions are configured
2. Set up AWS Cost and Usage Report integration and configure Kubecost to access your billing data
3. Implement a comprehensive resource tagging strategy to accurately track costs across teams and applications
4. Configure alerts and thresholds for cost anomalies and resource utilization metrics
5. Review initial baseline metrics and identify immediate optimization opportunities for Windows containers
6. Establish regular review cycles to analyze Kubecost reports and implement recommended optimizations

Resources

Related documents:

- Gain visibility into your Amazon EKS costs
- Learn more about Kubecost

MSFTCOST07-BP03 Change your scale strategy for Windows Containers on Kubernetes using Karpenter

Karpenter is a Kubernetes cluster autoscaler that dynamically provisions EC2 instances based on your workload demands, automatically launching right-sized instances in response to pending pods and continuously evaluating the cluster to optimize costs by consolidating workloads onto more efficient instance types. The tool proactively replaces outdated nodes with newer ones to maintain

security compliance and supports diverse compute requirements by selecting from a broad range of instance types and purchasing options, including both On-Demand and Spot instances.

Desired outcome: Expect to achieve improved resource utilization, reduced operational overhead, and optimized cloud costs. EKS clusters will dynamically scale to meet application demands, maintain up-to-date and secure infrastructure, and efficiently manage diverse workloads without manual intervention, ultimately leading to a more responsive, cost-effective, and easily managed Kubernetes environment on AWS.

Common anti-patterns:

- Teams often configure Karpenter with unnecessarily specific instance type constraints or narrow capacity requirements, limiting its ability to efficiently provision nodes and potentially increasing costs by forcing the use of suboptimal instance types.
- Organizations frequently deploy Karpenter without properly configuring Pod Disruption Budgets (PDBs), leading to unexpected application downtime during node consolidation or replacement operations, as Karpenter may terminate nodes without ensuring proper workload migration.

Benefits of establishing this best practice:

- By allowing Karpenter to intelligently select from a broad range of instance types and automatically consolidate workloads, organizations can significantly reduce their AWS compute costs while maintaining optimal performance for their applications.
- Teams spend less time on manual cluster management and capacity planning, as Karpenter automates node provisioning, scaling, and replacement activities, enabling engineers to focus on higher-value development tasks.
- With Karpenter's automated node replacement feature, clusters maintain better security hygiene through regular updates and patches, reducing the risk of vulnerabilities while ensuring compliance with security standards without manual intervention.

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

Implementation guidance

To implement Karpenter effectively, define flexible provisioner configurations that accommodate both Linux and Windows workloads, ensuring appropriate instance types are available for each OS. Set up distinct provisioners with OS-specific requirements, configure Pod Disruption Budgets

for critical applications, and establish proper taints and tolerations to ensure workloads land on compatible nodes. Regularly monitor cluster behavior and costs to optimize your configuration.

Implementation steps

1. Install and configure Karpenter in your EKS cluster, ensuring proper IAM permissions and VPC settings
2. Create flexible provisioner configurations for both Linux and Windows workloads, specifying appropriate instance types and purchasing options
3. Set up Pod Disruption Budgets for critical applications to maintain availability during node consolidation
4. Configure monitoring and alerting to track Karpenter's performance and cluster resource utilization
5. Regularly review and adjust Karpenter settings based on observed cluster behavior and cost metrics

Resources

Related documents:

- Karpenter
- Getting Started with Karpenter

.NET

Optimizing .NET applications for cost-efficiency in the cloud involves leveraging the latest advancements in cross-platform .NET technologies. By migrating from traditional .NET Framework to modern, cross-platform .NET versions, organizations can take advantage of Linux-based deployments, including cost-effective ARM64 instances like AWS Graviton2. This shift not only reduces licensing costs but also opens up opportunities for serverless architectures and improved performance. AWS provides tools like AWS Transform to facilitate this modernization process, enabling .NET applications to fully exploit cloud benefits and achieve significant cost savings while enhancing scalability and maintainability.

MSFTCOST08: How do you save on .NET for your Microsoft workload?

While .NET Framework applications often utilize AWS virtual machines or containers, cross-platform .NET's introduction enables modern applications to fully exploit cloud benefits, including serverless environments. Cross-platform .NET further expands possibilities by offering performant hosting on ARM64 EC2 instances, like Graviton2 families. This advancement allows access to specialized compute options on Amazon EC2, optimizing performance for diverse workloads such as video encoding, web serving, and high-performance computing. Thus, .NET applications can now harness the latest processor technologies, adapting to specific task requirements and maximizing efficiency in the cloud environment.

Best practices

- MSFTCOST08-BP01 Refactor to cross-platform .NET and move to Linux
- MSFTCOST08-BP02 Consider serverless architecture for your Microsoft .NET applications

MSFTCOST08-BP01 Refactor to cross-platform .NET and move to Linux

Migrating .NET Framework applications to .NET 8 or later enables cross-platform deployment, improved security, and better performance. This modernization allows applications to run on Linux systems, leverage cloud-native features, and benefit from the latest optimizations, resulting in more efficient and maintainable systems.

Desired outcome: Aim to achieve reduced licensing costs, improved performance, and enhanced security. The modernized applications run efficiently on Linux environments, including AWS Graviton processors, while leveraging the latest .NET features and cloud-native capabilities. This transformation results in a more cost-effective, scalable, and maintainable application portfolio that aligns with modern cloud architecture principles.

Common anti-patterns:

- Continuing to rely on Windows-specific dependencies and COM components without evaluating modern alternatives, making the migration to Linux impossible and perpetuating technical debt and higher operational costs.
- Attempting to run portions of the application on Linux while keeping critical components on Windows servers, creating a complex hybrid architecture that increases operational overhead and negates the cost benefits of the migration while introducing potential compatibility issues.

Benefits of establishing this best practice:

- Moving to Linux eliminates Windows licensing costs while enabling the use of cost-effective infrastructure options like AWS Graviton processors, resulting in significant operational cost savings and improved performance metrics through modern .NET optimizations.
- Cross-platform .NET applications can be deployed across diverse environments using containerization technologies, enabling efficient CI/CD pipelines, simplified scaling strategies, and better resource utilization in cloud environments, leading to improved application reliability and reduced maintenance overhead.

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

Implementation guidance

Begin with an AWS Transform-assisted assessment to identify Windows dependencies, then implement an incremental migration strategy focusing on high-impact components first. Utilize automated refactoring tools for conversion to cross-platform .NET, containerize the application, and establish comprehensive testing protocols to ensure successful deployment on Linux environments while maintaining application performance and reliability.

Implementation steps

1. Conduct application assessment using AWS Transform to identify Windows dependencies and migration challenges
2. Develop a phased migration plan, prioritizing components for maximum cost-benefit impact
3. Refactor code to cross-platform .NET using AWS Transform's AI-powered tools and manual adjustments
4. Containerize the application and implement a robust testing strategy for Linux environments
5. Deploy the refactored application to Linux servers, including AWS Graviton instances, and monitor performance

Resources

Related documents:

- Refactor to modern .NET and move to Linux

Related tools:

- Modernizing .NET with AWS Transform

MSFTCOST08-BP02 Consider serverless architecture for your Microsoft .NET applications

AWS Lambda enables .NET developers to build serverless applications without managing servers, paying only for actual usage. Using modern .NET versions, developers can create scalable functions in C# or F# that run on-demand in the cloud, reducing costs and development time.

Desired outcome: By adopting serverless architecture with AWS Lambda for .NET applications, organizations may achieve improved scalability and cost efficiency, paying only for resources used while eliminating server management overhead. Using AWS tools like Microservice Extractor may further simplify the modernization of existing applications.

Common anti-patterns:

- Maintaining continuously operational and over-sized servers for .NET applications instead of leveraging serverless architecture, resulting in unnecessary costs and underutilized resources.
- Keeping large and monolithic .NET applications intact rather than breaking them down into microservices or serverless functions, leading to reduced flexibility and scalability.

Benefits of establishing this best practice:

- Organizations only pay for actual compute resources consumed during function execution, eliminating costs associated with idle server capacity and infrastructure management.
- Development teams can focus on code rather than server maintenance, reducing operational overhead and accelerating deployment cycles.
- Applications automatically scale based on demand without manual intervention, ensuring optimal performance during peak loads while maintaining cost efficiency during low-usage periods.

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

Implementation guidance

Begin by identifying suitable .NET applications or components that can benefit from serverless architecture. Start small by migrating discrete functions to AWS Lambda using modern .NET versions (Core or later). Utilize AWS Microservice Extractor for .NET to analyze and decompose existing monolithic applications into smaller, manageable services. Implement proper monitoring

and logging from the outset using AWS CloudWatch, and establish clear deployment pipelines using AWS CI/CD tools. Gradually, expand the serverless footprint as the team gains experience and confidence with the architecture.

Implementation steps

1. Install and configure AWS Microservice Extractor for .NET to analyze existing monolithic applications and identify potential microservice candidates
2. Assess existing .NET applications and identify functions suitable for serverless migration, prioritizing stateless operations and event-driven processes
3. Set up the AWS Lambda development environment with .NET SDK and necessary AWS tools (AWS Toolkit for Visual Studio and AWS CLI)
4. Create and test initial Lambda functions using modern .NET versions, implementing proper error handling and logging
5. Configure automated deployment pipelines using AWS CI/CD services (CodePipeline and CodeBuild) for consistent function updates
6. Monitor function performance and costs using AWS CloudWatch, and optimize resource allocation based on actual usage patterns

Resources

Related documents:

- Consider serverless .NET

Related tools:

*What Is AWS Microservice Extractor for .NET?

Sustainability

The sustainability pillar focuses on designing and operating Microsoft workloads on AWS in an environmentally responsible manner. It emphasizes minimizing environmental impact through efficient resource utilization while maintaining business effectiveness. This pillar addresses three key areas: aligning business requirements with sustainable architecture designs, optimizing Microsoft ecosystems for sustainability, and implementing effective monitoring of sustainability performance metrics

Focus areas

- Design principles
- Efficient infrastructure

Design principles

- **Efficient resource utilization:** Design Microsoft workloads on AWS to maximize resource efficiency. This involves running fewer, highly-utilized instances and implementing practices that reduce energy consumption and infrastructure footprint. Align these efforts with cost optimization strategies while being mindful of potential trade-offs with performance.
- **Sustainability-driven architecture:** Integrate sustainability considerations into the core of your Microsoft architecture design process. Establish clear, measurable sustainability objectives that align with broader business goals. Verify that stakeholders understand and agree on these objectives, recognizing that prioritizing sustainability may require adjustments to traditional performance or cost priorities.

Efficient infrastructure

Efficient infrastructure focuses on optimizing Microsoft workloads on AWS to maximize resource utilization while minimizing environmental impact. This involves designing architectures with fewer, highly-utilized instances, reducing energy consumption, and shrinking infrastructure footprint. It requires balancing performance needs with sustainability goals, implementing robust monitoring, and practicing continuous optimization to achieve both environmental and cost benefits.

MSFTSUS01: How do you design your Microsoft workload to minimize its environmental impact?

Microsoft-based cloud architectures on AWS generally become more environmentally sustainable as they reduce infrastructure footprints and energy consumption, which often aligns with cost optimization in cloud environments like AWS. This approach is illustrated by practices such as running fewer, highly-utilized instances, as described in the Well-Architected Framework sustainability pillar.

However, business stakeholders and Microsoft architecture teams should recognize that prioritizing sustainability may necessitate adjustments to performance goals. Furthermore, organizations should promote continuous optimization practices that focus on energy efficiency, fostering long-term improvements in both sustainability and operational effectiveness.

By carefully balancing these considerations, companies can develop cloud architectures that are not only cost-effective but also environmentally responsible, aligning their technological strategies with broader sustainability objectives while maintaining optimal performance.

Best practices

- MSFTSUS01-BP01 Align business requirements with sustainable Microsoft architecture designs
- MSFTSUS01-BP02 Monitor Microsoft workload sustainability performance

MSFTSUS01-BP01 Align business requirements with sustainable Microsoft architecture designs

When designing Microsoft environments in the cloud, business leaders and architecture teams must actively prioritize sustainability. This requires a clear understanding of the trade-offs involved and the ability to articulate them effectively. In production workloads, with reliability and performance traditionally taking precedence over costs, sustainable designs may not always result in overall cost savings. In fact, implementing more sustainable practices for Microsoft workloads on AWS could potentially increase expenses in areas such as software licensing, staffing, or other business operations. To successfully meet sustainability goals, these trade-offs must be carefully balanced against other business priorities. It's crucial that stakeholders agree on clearly defined, measurable sustainability objectives that align with broader business needs and performance

requirements. This approach integrates sustainability efforts thoughtfully into the overall business strategy, rather than being treated as an isolated initiative.

Desired outcome: Your Microsoft workload architecture design reflects clearly defined sustainability objectives that are agreed upon by your stakeholders and balanced effectively with business priorities, performance requirements, and cost considerations.

Common anti-patterns:

- Treating sustainability as a secondary concern or add-on, rather than integrating it into the core design process of Microsoft workloads on AWS. This often results in missed opportunities for environmental improvements and potential conflicts with established architectures.
- Setting sustainability goals for Microsoft environments without considering their impact on other business priorities, performance requirements, or cost structures. This can lead to unrealistic expectations, unintended consequences, and resistance from stakeholders.
- Implementing sustainability measures for Microsoft workloads without clear agreement and understanding among relevant stakeholders. This often results in conflicting priorities, inadequate support for sustainability initiatives, and difficulty in measuring or achieving environmental objectives.

Benefits of establishing this best practice:

- Clear prioritization of sustainability in Microsoft workload design enables better alignment between environmental goals and business objectives. This integrated approach verifies that sustainability initiatives support rather than conflict with other strategic priorities, leading to more successful and sustainable outcomes.
- By establishing clear understanding of trade-offs and getting stakeholder agreement on sustainability objectives, organizations can make better-informed decisions about resource allocation, performance requirements, and infrastructure investments. This leads to more balanced and effective implementation of sustainable practices.
- Having clearly defined, measurable sustainability objectives allows organizations to effectively track and demonstrate the environmental impact of their Microsoft workloads. This enables better reporting, validates sustainability investments, and supports continuous improvement while maintaining alignment with business performance requirements.

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

Implementation guidance

Organizations must integrate sustainability into their Microsoft workload designs on AWS by establishing clear environmental objectives and gaining stakeholder alignment on trade-offs. This approach balances environmental responsibility with operational effectiveness, helping meet growing expectations from customers, investors, and regulators while maintaining business performance.

Implementation steps

- Integrate sustainability as a core consideration in Microsoft workload design on AWS. Communicate clearly about trade-offs, set measurable sustainability objectives, and gain stakeholder consensus on balancing these goals with other business priorities.
- Emphasize the importance of environmental responsibility in the context of current business trends. Highlight how sustainability efforts can meet the expectations of customers, investors, and regulators.
- Align sustainability efforts with overall business strategy. Focus on reducing environmental impact while maintaining operational effectiveness. Demonstrate how this approach can lead to competitive advantages and contribute to broader corporate sustainability goals.

Resources

Related documents:

- AWS Well-Architected Framework - Sustainability pillar

MSFTSUS01-BP02 Monitor Microsoft workload sustainability performance

Monitoring and reporting on Microsoft workload sustainability in AWS provides essential feedback on the effectiveness of implemented changes. This data supports sustainability reporting for shareholders, regulators, and eco-conscious customers. Using metrics from the operational excellence pillar, these reports can showcase improvements in your Microsoft landscape's operational sustainability, demonstrating alignment with corporate sustainability goals. This process creates a continuous feedback loop, allowing for ongoing optimization and validation of sustainability efforts in your Microsoft environment.

Desired outcome: Regular monitoring and comprehensive reporting demonstrates measurable improvements in Microsoft workload sustainability metrics on AWS, enabling data-driven decisions, validating green initiatives, and meeting stakeholder sustainability reporting requirements while maintaining operational excellence.

Common anti-patterns:

- Organizations collect extensive sustainability metrics for Microsoft workloads on AWS but fail to translate data into actionable improvements. Despite sophisticated monitoring tools and regular reporting, the information remains unused, leading to stagnant sustainability initiatives and wasted resources in data collection while missing opportunities for meaningful environmental impact.
- Sustainability monitoring and reporting for Microsoft workloads operate in isolation from other operational metrics and business processes. This disconnected approach, where sustainability teams work separately from IT operations, results in missed opportunities for integrated improvements and limited understanding of sustainability metrics across departments, ultimately reducing the effectiveness of environmental initiatives.

Benefits of establishing this best practice:

- By monitoring and reporting on Microsoft workload sustainability in AWS, organizations gain valuable insights into their environmental impact. This data enables informed decision-making, allowing companies to identify areas of high resource consumption or inefficiency. With this knowledge, they can implement targeted improvements, optimize workloads, and reduce their carbon footprint more effectively. The result is a measurable decrease in energy use and emissions, contributing significantly to corporate sustainability goals.
- Regular sustainability reporting builds transparency and trust with shareholders, regulators, and environmentally conscious customers. By providing clear, quantifiable data on sustainability efforts and improvements in Microsoft workloads, companies demonstrate their commitment to environmental responsibility. This transparency not only aids in regulatory adherence but also strengthens the company's reputation, potentially attracting eco-minded investors and customers. It can lead to improved stakeholder relationships, better market positioning, and even competitive advantages in an increasingly sustainability-focused business landscape.

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

Implementation guidance

Create a structured approach for monitoring and reporting Microsoft workload sustainability in AWS. Identify metrics, set up AWS tools, integrate with operational processes, establish reporting cycles, implement improvement loops, and and align with stakeholder needs. This comprehensive system enables organizations to measure, improve, and communicate their environmental impact effectively.

Implementation steps

- Set up custom metrics and dashboards in Amazon CloudWatch to monitor key sustainability indicators for Microsoft workloads, such as CPU utilization, storage efficiency, and network traffic. Establish alarms for thresholds that indicate potential sustainability issues or opportunities for optimization.
- Enable and configure AWS Cost and Usage Reports to track resource consumption and associated costs for Microsoft workloads. Use these reports to identify patterns in usage, pinpoint areas of high energy consumption, and inform decisions on rightsizing and optimizing workloads for improved sustainability.

Resources

Related documents:

- What are AWS Cost and Usage Reports?
- Increasing sustainability for your Microsoft workloads on AWS

Related tools:

- Amazon CloudWatch

Conclusion

In conclusion, the AWS Well-Architected for Microsoft Workloads Lens is an invaluable resource for organizations seeking to optimize their Microsoft-based systems on the AWS cloud platform. This specialized lens builds upon the foundational AWS Well-Architected Framework, offering tailored guidance for the unique challenges and opportunities presented by Microsoft technologies in a cloud environment.

By using the extensive experience of AWS Solutions Architects and incorporating best practices gleaned from countless real-world implementations, the Microsoft Workloads Lens provides a comprehensive approach to designing, deploying, and maintaining robust Microsoft solutions on AWS. It addresses critical aspects such as performance optimization, security, licensing management, and operational excellence within the context of Microsoft-specific workloads.

The systematic evaluation process offered by this lens, coupled with its integration into the AWS Well-Architected Tool, empowers technology professionals to make informed decisions and continuously improve their Microsoft-based architectures. Furthermore, the availability of hands-on labs and support from specialized APN Partners provides organizations the practical resources to implement these best practices effectively.

The AWS Well-Architected Microsoft Workloads Lens serves as a crucial tool for businesses looking to harness the full potential of Microsoft technologies on the AWS Cloud. By adhering to the guidance provided in this lens, organizations can significantly enhance the reliability, security, efficiency, and cost-effectiveness of their Microsoft workloads, thereby increasing their chances of achieving business success in the cloud.

Contributors

The following individuals and groups contributed to this document:

- Bruno Lopes, Sr. Geo SSA Containers, Amazon Web Services
- Carlos Felicio, Sr TAM (Partner), Amazon Web Services
- Luciano Bernardes, Specialist Sr. SA, Amazon Web Services
- Vitor Euphrasio da Silva, Sr. Spec. SA, Infra Mig & Mod, Amazon Web Services
- Stewart Matzek, Sr. Technical Writer, Amazon Web Services
- Madhuri Srinivasan, Sr. Technical Writer, Amazon Web Services
- Matthew Wygant, Sr. TPM Guidance, Amazon Web Services

Document revisions

The following table describes the documentation releases for the Microsoft Workloads Lens.

Change	Description	Date
Initial release	Initial release of the Microsoft Workloads Lens.	February 2, 2026

Notices

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

AWS Glossary

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