AWS Well-Architected Framework

Performance Efficiency Pillar

Performance Efficiency Pillar: AWS Well-Architected Framework

Table of Contents

Performance Efficiency Pillar - AWS Well-Architected Framework

Publication date: **October 3, 2023** (*Document revisions*)

Abstract

This whitepaper focuses on the performance efficiency pillar of the AWS Well-Architected Framework. The scope of this document is to provide guidance that helps customers use cloud resources efficiently to meet their business requirements, and maintain that efficiency as demand changes and technologies evolve.

Introduction

The AWS Well-Architected Framework helps you understand the pros and cons of decisions you make while building workloads on AWS. Using the Framework helps you learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable workloads in the cloud. The Framework provides a way for you to consistently measure your architectures against best practices and identify areas for improvement. We believe that having well-architected workloads greatly increases the likelihood of business success.

The framework is based on six pillars:

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

This paper focuses on applying the principles of the performance efficiency pillar to your workloads. In traditional, on-premises environments, achieving high and lasting performance is challenging. Using the principles in this paper will help you build architectures on AWS that efficiently deliver sustained performance over time. The guidance and best practices in this

document are spread across five key focus areas that serve as guiding principles for building performance efficient cloud solutions on AWS. These focus areas are:

- Architecture selection
- Compute and hardware
- Data management
- Networking and content delivery
- Process and culture

This paper is intended for those in technology roles, such as chief technology officers (CTOs), architects, developers, and operations team members. After reading this paper, you'll understand AWS best practices and strategies to use when designing a performant cloud architecture.

Performance efficiency

The performance efficiency pillar focuses on the efficient use of computing resources to meet requirements, and how to maintain efficiency as demand changes and technologies evolve.

Topics

Design principles

The following design principles can help you achieve and maintain efficient workloads in the cloud.

- **Democratize advanced technologies:** Make advanced technology implementation easier for your team by delegating complex tasks to your cloud vendor. Rather than asking your IT team to learn about hosting and running a new technology, consider consuming the technology as a service. For example, NoSQL databases, media transcoding, and machine learning are all technologies that require specialized expertise. In the cloud, these technologies become services that your team can consume, allowing your team to focus on product development rather than resource provisioning and management.
- **Go global in minutes:** Deploying your workload in multiple AWS Regions around the world allows you to provide lower latency and a better experience for your customers at minimal cost.
- **Use serverless architectures:** Serverless architectures remove the need for you to run and maintain physical servers for traditional compute activities. For example, serverless storage services can act as static websites (removing the need for web servers) and event services can host code. This removes the operational burden of managing physical servers, and can lower transactional costs because managed services operate at cloud scale.
- **Experiment more often:** With virtual and automatable resources, you can quickly carry out comparative testing using different types of instances, storage, or configurations.
- **Consider mechanical sympathy:** Use the technology approach that aligns best with your goals. For example, consider data access patterns when you select database or storage for your workload.

Definition

Focus on the following areas to achieve performance efficiency in the cloud:

- Architecture selection
- Compute and hardware
- Data management
- Networking and content delivery
- Process and culture

Take a data-driven approach to build a high-performing architecture. Gather data on all aspects of the architecture, from the high-level design to the selection and configuration of resource types.

Reviewing your choices on a regular basis, ensures that you are taking advantage of the continually evolving AWS Cloud. Monitoring ensures that you are aware of any deviance from expected performance. Make trade-offs in your architecture to improve performance, such as using compression or caching, or relaxing consistency requirements.

Architecture selection

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

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

This focus area shares guidance and best practices on how to select efficient, high-performing cloud resources and architecture patterns.

Best practices

- PERF01-BP01 Learn about and understand available cloud services and features
- PERF01-BP02 Use guidance from your cloud provider or an appropriate partner to learn about architecture patterns and best practices
- PERF01-BP03 Factor cost into architectural decisions
- PERF01-BP04 Evaluate how trade-offs impact customers and architecture efficiency
- PERF01-BP05 Use policies and reference architectures
- PERF01-BP06 Use benchmarking to drive architectural decisions
- PERF01-BP07 Use a data-driven approach for architectural choices

PERF01-BP01 Learn about and understand available cloud services and features

Continually learn about and discover available services and configurations that help you make better architectural decisions and improve performance efficiency in your workload architecture.

Common anti-patterns:

- You use the cloud as a collocated data center.
- You do not modernize your application after migration to the cloud.

- You only use one storage type for all things that need to be persisted.
- You use instance types that are closest matched to your current standards, but are larger where needed.
- You deploy and manage technologies that are available as managed services.

Benefits of establishing this best practice: By considering new services and configurations, you may be able to greatly improve performance, reduce cost, and optimize the effort required to maintain your workload. It can also help you accelerate the time-to-value for cloud-enabled products.

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

Implementation guidance

AWS continually releases new services and features that can improve performance and reduce the cost of cloud workloads. Staying up-to-date with these new services and features is crucial for maintaining performance efficacy in the cloud. Modernizing your workload architecture also helps you accelerate productivity, drive innovation, and unlock more growth opportunities.

Implementation steps

- Inventory your workload software and architecture for related services. Decide which category of products to learn more about.
- Explore AWS offerings to identify and learn about the relevant services and configuration options that can help you improve performance and reduce cost and operational complexity.
  - What's New with AWS?
  - AWS Blog
  - AWS Skill Builder
  - AWS Events and Webinars
  - AWS Training and Certifications
  - AWS Youtube Channel
  - AWS Workshops
  - AWS Communities
- Use sandbox (non-production) environments to learn and experiment with new services without incurring extra cost.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- Build modern applications on AWS

Related videos:

- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP02 Use guidance from your cloud provider or an appropriate partner to learn about architecture patterns and best practices

Use cloud company resources such as documentation, solutions architects, professional services, or appropriate partners to guide your architectural decisions. These resources help you review and improve your architecture for optimal performance.

Common anti-patterns:

- You use AWS as a common cloud provider.
- You use AWS services in a manner that they were not designed for.
- You follow all guidance without considering your business context.

Benefits of establishing this best practice: Using guidance from a cloud provider or an appropriate partner can help you to make the right architectural choices for your workload and give you confidence in your decisions.

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

Implementation guidance

AWS offers a wide range of guidance, documentation, and resources that can help you build and manage efficient cloud workloads. AWS documentation provides code samples, tutorials, and detailed service explanations. In addition to documentation, AWS provides training and certification programs, solutions architects, and professional services that can help customers explore different aspects of cloud services and implement efficient cloud architecture on AWS.

Leverage these resources to gain insights into valuable knowledge and best practices, save time, and achieve better outcomes in the AWS Cloud.

Implementation steps

- Review AWS documentation and guidance and follow the best practices. These resources can help you effectively choose and configure services and achieve better performance.
  - AWS documentation (like user guides and whitepapers)
  - AWS Blog
  - AWS Training and Certifications
  - AWS Youtube Channel
- Join AWS partner events (like AWS Global Summits, AWS re:Invent, user groups, and workshops) to learn from AWS experts about best practices for using AWS services.
  - AWS Events and Webinars
  - AWS Workshops
  - AWS Communities
- Reach out to AWS for assistance when you need additional guidance or product information. AWS Solutions Architects and AWS Professional Services provide guidance for solution implementation. AWS Partners provide AWS expertise to help you unlock agility and innovation for your business.
- Use AWS Support if you need technical support to use a service effectively. Our Support plans are designed to give you the right mix of tools and access to expertise so that you can be

successful with AWS while optimizing performance, managing risk, and keeping costs under control.

Resources

Related documents:

- AWS Architecture Center
- AWS Solutions Library
- AWS Knowledge Center
- AWS Enterprise Support

Related videos:

- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP03 Factor cost into architectural decisions

Factor cost into your architectural decisions to improve resource utilization and performance efficiency of your cloud workload. When you are aware of the cost implications of your cloud workload, you are more likely to leverage efficient resources and reduce wasteful practices.

Common anti-patterns:

- You only use one family of instances.
- You do not evaluate licensed solutions against open-source solutions.
- You do not define storage lifecycle policies.
- You do not review new services and features of the AWS Cloud.
- You only use block storage.

Benefits of establishing this best practice: Factoring cost into your decision making allows you to use more efficient resources and explore other investments.

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

Implementation guidance

Optimizing workloads for cost can improve resource utilization and avoid waste in a cloud workload. Factoring cost into architectural decisions usually includes right-sizing workload components and enabling elasticity, which results in improved cloud workload performance efficiency.

Implementation steps

- Establish cost objectives like budget limits for your cloud workload.

- Identify the key components (like instances and storage) that drive cost of your workload. You can use AWS Pricing Calculator and AWS Cost Explorer to identify key cost drivers in your workload.

- Use Well-Architected cost optimization best practices to optimize these key components for cost.
- Continually monitor and analyze cost to identify cost optimization opportunities in your workload.
  - Use AWS Budgets to get alerts for unacceptable costs.
  - Use AWS Compute Optimizer or AWS Trusted Advisor to get cost optimization recommendations.
  - Use AWS Cost Anomaly Detection to get automated cost anomaly detection and root cause analysis.

Resources

Related documents:

- A Detailed Overview of the Cost Intelligence Dashboard
- AWS Architecture Center
- AWS Partner Network

- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- This is my Architecture
- Optimize performance and cost for your AWS compute

Related examples:

- AWS Samples
- AWS SDK Examples
- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

PERF01-BP04 Evaluate how trade-offs impact customers and architecture efficiency

When evaluating performance-related improvements, determine which choices impact your customers and workload efficiency. For example, if using a key-value data store increases system performance, it is important to evaluate how the eventually consistent nature of this change will impact customers.

Common anti-patterns:

- You assume that all performance gains should be implemented, even if there are tradeoffs for implementation.
- You only evaluate changes to workloads when a performance issue has reached a critical point.

Benefits of establishing this best practice: When you are evaluating potential performance-related improvements, you must decide if the tradeoffs for the changes are acceptable with the workload requirements. In some cases, you may have to implement additional controls to compensate for the tradeoffs.

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

Implementation guidance

Identify critical areas in your architecture in terms of performance and customer impact. Determine how you can make improvements, what trade-offs those improvements bring, and how they impact the system and the user experience. For example, implementing caching data can help dramatically improve performance but requires a clear strategy for how and when to update or invalidate cached data to prevent incorrect system behavior.

Implementation steps

- Understand your workload requirements and SLAs.
- Clearly define evaluation factors. Factors may relate to cost, reliability, security, and performance of your workload.
- Select architecture and services that can address your requirements.
- Conduct experimentation and proof of concepts (POCs) to evaluate trade-off factors and impact on customers and architecture efficiency. Usually, highly-available, performant, and secure workloads consume more cloud resources while providing better customer experience.

Resources

Related documents:

- Amazon Builders' Library
- Amazon QuickSight KPIs
- Amazon CloudWatch RUM
- X-Ray Documentation
- Understand resiliency patterns and trade-offs to architect efficiently in the cloud

Related videos:

- Build a Monitoring Plan
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP05 Use policies and reference architectures

Use internal policies and existing reference architectures when selecting services and configurations to be more efficient when designing and implementing your workload.

Common anti-patterns:

- You allow a wide variety of technology that may impact the management overhead of your company.

Benefits of establishing this best practice: Establishing a policy for architecture, technology, and vendor choices allows decisions to be made quickly.

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

Implementation guidance

Having internal policies in selecting resources and architecture provides standards and guidelines to follow when making architectural choices. Those guidelines streamline the decision-making process when choosing the right cloud service and can help improve performance efficiency. Deploy your workload using policies or reference architectures. Integrate the services into your cloud deployment, then use your performance tests to verify that you can continue to meet your performance requirements.

Implementation steps

- Clearly understand the requirements of your cloud workload.
- Review internal and external policies to identify the most relevant ones.
- Use the appropriate reference architectures provided by AWS or your industry best practices.
- Create a continuum consisting of policies, standards, reference architectures, and prescriptive guidelines for common situations. Doing so allows your teams to move faster. Tailor the assets for your vertical if applicable.
- Validate these policies and reference architectures for your workload in sandbox environments.

- Stay up-to-date with industry standards and AWS updates to make sure your policies and reference architectures help optimize your cloud workload.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP06 Use benchmarking to drive architectural decisions

Benchmark the performance of an existing workload to understand how it performs on the cloud and drive architectural decisions based on that data.

Common anti-patterns:

- You rely on common benchmarks that are not indicative of your workload's characteristics.
- You rely on customer feedback and perceptions as your only benchmark.

Benefits of establishing this best practice: Benchmarking your current implementation allows you to measure performance improvements.

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

Implementation guidance

Use benchmarking with synthetic tests to assess how your workload's components perform. Benchmarking is generally quicker to set up than load testing and is used to evaluate the technology for a particular component. Benchmarking is often used at the start of a new project, when you lack a full solution to load test.

You can either build your own custom benchmark tests or use an industry standard test, such as TPC-DS, to benchmark your workloads. Industry benchmarks are helpful when comparing environments. Custom benchmarks are useful for targeting specific types of operations that you expect to make in your architecture.

When benchmarking, it is important to pre-warm your test environment to get valid results. Run the same benchmark multiple times to verify that you've captured any variance over time.

Because benchmarks are generally faster to run than load tests, they can be used earlier in the deployment pipeline and provide faster feedback on performance deviations. When you evaluate a significant change in a component or service, a benchmark can be a quick way to see if you can justify the effort to make the change. Using benchmarking in conjunction with load testing is important because load testing informs you about how your workload performs in production.

Implementation steps

- Define the metrics (like CPU utilization, latency, or throughput) to evaluate your workload performance.
- Identify and set up a benchmarking tool that is suitable for your workload. You can use AWS services (like Amazon CloudWatch) or a third-party tool that is compatible with your workload.
- Perform your benchmark tests and monitor the metrics during the test.
- Analyze and document the benchmarking results to identify any bottlenecks and issues.
- Use the test results to make architectural decisions and adjust your workload. This may include changing services or adopting new features.
- Retest your workload after the adjustment.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics

Related videos:

- This is my Architecture
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- AWS Samples
- AWS SDK Examples
- Distributed Load Tests
- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP07 Use a data-driven approach for architectural choices

Define a clear, data-driven approach for architectural choices to verify that the right cloud services and configurations are used to meet your specific business needs.

Common anti-patterns:

- You assume your current architecture is static and should not be updated over time.
- Your architectural choices are based upon guesses and assumptions.
- You introduce architecture changes over time without justification.

Benefits of establishing this best practice: By having a well-defined approach for making architectural choices, you use data to influence your workload design and make informed decisions over time.

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

Implementation guidance

Use internal experience and knowledge of the cloud or external resources such as published use cases or whitepapers to choose resources and services in your architecture. You should have a well-defined process that encourages experimentation and benchmarking with the services that could be used in your workload.

Backlogs for critical workloads should consist of not just user stories which deliver functionality relevant to business and users, but also technical stories which form an architecture runway for the workload. This runway is informed by new advancements in technology and new services and adopts them based on data and proper justification. This verifies that the architecture remains future-proof and does not stagnate.

Implementation steps

- Engage with key stakeholders to define workload requirements, including performance, availability, and cost considerations. Consider factors such as the number of users and usage pattern for your workload.
- Create an architecture runway or a technology backlog which is prioritized along with the functional backlog.
- Evaluate and assess different cloud services (for more detail, see PERF01-BP01 Learn about and understand available cloud services and features).
- Explore different architectural patterns, like microservices or serverless, that meet your performance requirements (for more detail, see PERF01-BP02 Use guidance from your cloud provider or an appropriate partner to learn about architecture patterns and best practices).
- Consult other teams, architecture diagrams, and resources, such as AWS Solution Architects, AWS Architecture Center, and AWS Partner Network, to help you choose the right architecture for your workload.

- Define performance metrics like throughput and response time that can help you evaluate the performance of your workload.
- Experiment and use defined metrics to validate the performance of the selected architecture.
- Continually monitor and make adjustments as needed to maintain the optimal performance of your architecture.
- Document your selected architecture and decisions as a reference for future updates and learnings.
- Continually review and update the architecture selection approach based on learnings, new technologies, and metrics that indicate a needed change or problem in the current approach.

Resources

Related documents:

- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

Compute and hardware

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

This focus area shares guidance and best practices on how to identify and optimize compute options for performance efficiency in the cloud.

Best practices

- PERF02-BP01 Select the best compute options for your workload
- PERF02-BP02 Understand the available compute configuration and features
- PERF02-BP03 Collect compute-related metrics
- PERF02-BP04 Configure and right-size compute resources
- PERF02-BP05 Scale your compute resources dynamically
- PERF02-BP06 Use optimized hardware-based compute accelerators

PERF02-BP01 Select the best compute options for your workload

Selecting the most appropriate compute option for your workload allows you to improve performance, reduce unnecessary infrastructure costs, and lower the operational efforts required to maintain your workload.

Common anti-patterns:

- You use the same compute option that was used on premises.
- You lack awareness of the cloud compute options, features, and solutions, and how those solutions might improve your compute performance.
- You over-provision an existing compute option to meet scaling or performance requirements when an alternative compute option would align to your workload characteristics more precisely.

Benefits of establishing this best practice: By identifying the compute requirements and evaluating against the options available, you can make your workload more resource efficient.

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

Implementation guidance

To optimize your cloud workloads for performance efficiency, it is important to select the most appropriate compute options for your use case and performance requirements. AWS provides a variety of compute options that cater to different workloads in the cloud. For instance, you can use Amazon EC2 to launch and manage virtual servers, AWS Lambda to run code without having to provision or manage servers, Amazon ECS or Amazon EKS to run and manage containers, or AWS Batch to process large volumes of data in parallel. Based on your scale and compute needs, you should choose and configure the optimal compute solution for your situation. You can also consider using multiple types of compute solutions in a single workload, as each one has its own advantages and drawbacks.

The following steps guide you through selecting the right compute options to match your workload characteristics and performance requirements.

Implementation steps

1. Understand your workload compute requirements. Key requirements to consider include processing needs, traffic patterns, data access patterns, scaling needs, and latency requirements.
2. Learn about different compute options available for your workload on AWS (as outlined in PERF01-BP01 Learn about and understand available cloud services and features. Here are some key AWS compute options, their characteristics, and common use cases:

AWS service	Key characteristics	Common use cases
Amazon Elastic Compute Cloud (Amazon EC2)	Has dedicated option for hardware, license requirements, large selection of different instance families, processor types and compute accelerators	Lift and shift migrations, monolithic application, hybrid environments, enterprise applications
Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS)	Easy deployment, consistent environments, scalable	Microservices, hybrid environments

AWS service	Key characteristics	Common use cases
AWS Lambda	Serverless compute service that runs code in response to events and automatically manages the underlying compute resources.	Microservices, event-driven applications
AWS Batch	Efficiently and dynamically provisions and scales Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and AWS Fargate compute resources, with an option to use On-Demand or Spot Instances based on your job requirements	HPC, train ML models
Amazon Lightsail	Preconfigured Linux and Windows application for running small workloads	Simple web applications, custom website

3. Evaluate cost (like hourly charge or data transfer) and management overhead (like patching and scaling) associated to each compute option.
4. Perform experiments and benchmarking in a non-production environment to identify which compute option can best address your workload requirements.
5. Once you have experimented and identified your new compute solution, plan your migration and validate your performance metrics.
6. Use AWS monitoring tools like Amazon CloudWatch and optimization services like AWS Compute Optimizer to continually optimize your compute resources based on real-world usage patterns.

Resources

Related documents:

- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Amazon ECS Containers: Amazon ECS Container Instances
- Functions: Lambda Function Configuration
- Prescriptive Guidance for Containers
- Prescriptive Guidance for Serverless

Related videos:

- How to choose compute option for startups
- Optimize performance and cost for your AWS compute
- Amazon EC2 foundations
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Deploy ML models for inference at high performance and low cost
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2

Related examples:

- Migrating the Web application to containers
- Run a Serverless Hello World

PERF02-BP02 Understand the available compute configuration and features

Understand the available configuration options and features for your compute service to help you provision the right amount of resources and improve performance efficiency.

Common anti-patterns:

- You do not evaluate compute options or available instance families against workload characteristics.
- You over-provision compute resources to meet peak-demand requirements.

Benefits of establishing this best practice: Be familiar with AWS compute features and configurations so that you can use a compute solution optimized to meet your workload characteristics and needs.

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

Implementation guidance

Each compute solution has unique configurations and features available to support different workload characteristics and requirements. Learn how these options complement your workload, and determine which configuration options are best for your application. Examples of these options include instance family, sizes, features (GPU, I/O), bursting, time-outs, function sizes, container instances, and concurrency. If your workload has been using the same compute option for more than four weeks and you anticipate that the characteristics will remain the same in the future, you can use AWS Compute Optimizer to find out if your current compute option is suitable for the workloads from CPU and memory perspective.

Implementation steps

1. Understand workload requirements (like CPU need, memory, and latency).
2. Review AWS documentation and best practices to learn about recommended configuration options that can help improve compute performance. Here are some key configuration options to consider:

Configuration option	Examples
Instance type	• Compute-optimized instances are ideal for the workloads that require high higher vCPU to memory ratio. • Memory-optimized instances deliver large amounts of memory to support memory intensive workloads.

Configuration option	Examples
	• Storage-optimized instances are designed for workloads that require high, sequentia l read and write access (IOPS) to local storage.
Pricing model	• On-Demand Instances let you use the compute capacity by the hour or second with no long-term commitment. These instances are good for bursting above performance baseline needs. • Savings Plans offer significant savings over On-Demand Instances in exchange for a commitment to use a specific amount of compute power for a one or three-year period. • Spot Instances let you take advantage of unused instance capacity at a discount for your stateless, fault-tolerant workloads.
Auto Scaling	Use Auto Scaling configuration to match compute resources to traffic patterns.
Sizing	• Use Compute Optimizer to get a machine-learning powered recommendations on which compute configuration best matches your compute characteristics. • Use AWS Lambda Power Tuning to select the best configuration for your Lambda function.

Configuration option	Examples
Hardware-based compute accelerators	• Accelerated computing instances perform functions like graphics processing or data pattern matching more efficiently than CPU-based alternatives. • For machine learning workloads, take advantage of purpose-built hardware that is specific to your workload, such as AWS Trainium, AWS Inferentia, and Amazon EC2 DL1

Resources

Related documents:

- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Processor State Control for Your Amazon EC2 Instance
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Amazon ECS Containers: Amazon ECS Container Instances
- Functions: Lambda Function Configuration

Related videos:

- Amazon EC2 foundations
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Optimize performance and cost for your AWS compute

Related examples:

- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

PERF02-BP03 Collect compute-related metrics

Record and track compute-related metrics to better understand how your compute resources are performing and improve their performance and their utilization.

Common anti-patterns:

- You only use manual log file searching for metrics.
- You only use the default metrics recorded by your monitoring software.
- You only review metrics when there is an issue.

Benefits of establishing this best practice: Collecting performance-related metrics will help you align application performance with business requirements to ensure that you are meeting your workload needs. It can also help you continually improve the resource performance and utilization in your workload.

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

Implementation guidance

Cloud workloads can generate large volumes of data such as metrics, logs, and events. In the AWS Cloud, collecting metrics is a crucial step to improve security, cost efficiency, performance, and sustainability. AWS provides a wide range of performance-related metrics using monitoring services such as Amazon CloudWatch to provide you with valuable insights. Metrics such as CPU utilization, memory utilization, disk I/O, and network inbound and outbound can provide insight into utilization levels or performance bottlenecks. Use these metrics as part of a data-driven approach to actively tune and optimize your workload's resources. In an ideal case, you should collect all metrics related to your compute resources in a single platform with retention policies implemented to support cost and operational goals.

Implementation steps

1. Identify which performance-related metrics are relevant to your workload. You should collect metrics around resource utilization and the way your cloud workload is operating (like response time and throughput).
   a. Amazon EC2 default metrics
   b. Amazon ECS default metrics

   c. Amazon EKS default metrics
   d. Lambda default metrics
   e. Amazon EC2 memory and disk metrics
2. Choose and set up the right logging and monitoring solution for your workload.
   a. AWS native Observability
   b. AWS Distro for OpenTelemetry
   c. Amazon Managed Service for Prometheus
3. Define the required filter and aggregation for the metrics based on your workload requirements.
   a. Quantify custom application metrics with Amazon CloudWatch Logs and metric filters
   b. Collect custom metrics with Amazon CloudWatch strategic tagging
4. Configure data retention policies for your metrics to match your security and operational goals.
   a. Default data retention for CloudWatch metrics
   b. Default data retention for CloudWatch Logs
5. If required, create alarms and notifications for your metrics to help you proactively respond to performance-related issues.
   a. Create alarms for custom metrics using Amazon CloudWatch anomaly detection
   b. Create metrics and alarms for specific web pages with Amazon CloudWatch RUM
6. Use automation to deploy your metric and log aggregation agents.
   a. AWS Systems Manager automation
   b. OpenTelemetry Collector

Resources

Related documents:

- Amazon CloudWatch documentation
- Collect metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch Agent
- Accessing Amazon CloudWatch Logs for AWS Lambda
- Using CloudWatch Logs with container instances
- Publish custom metrics
- AWS Answers: Centralized Logging

- AWS Services That Publish CloudWatch Metrics
- Monitoring Amazon EKS on AWS Fargate

Related videos:

- Application Performance Management on AWS

Related examples:

- Level 100: Monitoring with CloudWatch Dashboards
- Level 100: Monitoring Windows EC2 instance with CloudWatch Dashboards
- Level 100: Monitoring an Amazon Linux EC2 instance with CloudWatch Dashboards

PERF02-BP04 Configure and right-size compute resources

Configure and right-size compute resources to match your workload's performance requirements and avoid under- or over-utilized resources.

Common anti-patterns:

- You ignore your workload performance requirements resulting in over-provisioned or under-provisioned compute resources.
- You only choose the largest or smallest instance available for all workloads.
- You only use one instance family for ease of management.
- You ignore recommendations from AWS Cost Explorer or Compute Optimizer for right-sizing.
- You do not re-evaluate the workload for suitability of new instance types.
- You certify only a small number of instance configurations for your organization.

Benefits of establishing this best practice: Right-sizing compute resources ensures optimal operation in the cloud by avoiding over-provisioning and under-provisioning resources. Properly sizing compute resources typically results in better performance and enhanced customer experience, while also lowering cost.

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

Implementation guidance

Right-sizing allows organizations to operate their cloud infrastructure in an efficient and cost-effective manner while addressing their business needs. Over-provisioning cloud resources can lead to extra costs, while under-provisioning can result in poor performance and a negative customer experience. AWS provides tools such as AWS Compute Optimizer and AWS Trusted Advisor that use historical data to provide recommendations to right-size your compute resources.

Implementation steps

- Choose an instance type to best fit your needs:
  - How do I choose the appropriate Amazon EC2 instance type for my workload?
  - Attribute-based instance type selection for Amazon EC2 Fleet
  - Create an Auto Scaling group using attribute-based instance type selection
  - Optimizing your Kubernetes compute costs with Karpenter consolidation
- Analyze the various performance characteristics of your workload and how these characteristics relate to memory, network, and CPU usage. Use this data to choose resources that best match your workload's profile and performance goals.
- Monitor your resource usage using AWS monitoring tools such as Amazon CloudWatch.
- Select the right configuration for compute resources.
  - For ephemeral workloads, evaluate instance Amazon CloudWatch metrics such as `CPUUtilization` to identify if the instance is under-utilized or over-utilized.
  - For stable workloads, check AWS rightsizing tools such as AWS Compute Optimizer and AWS Trusted Advisor at regular intervals to identify opportunities to optimize and right-size the compute resource.
    - Well-Architected Lab - Rightsizing Recommendations
    - Well-Architected Lab - Rightsizing with Compute Optimizer
- Test configuration changes in a non-production environment before implementing in a live environment.
- Continually re-evaluate new compute offerings and compare against your workload’s needs.

Resources

Related documents:

- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Amazon ECS Containers: Amazon ECS Container Instances
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Functions: Lambda Function Configuration
- Processor State Control for Your Amazon EC2 Instance

Related videos:

- Amazon EC2 foundations
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2
- Deploy ML models for inference at high performance and low cost
- Optimize performance and cost for your AWS compute
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Simplifying Data Processing to Enhance Innovation with Serverless Tools

Related examples:

- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

PERF02-BP05 Scale your compute resources dynamically

Use the elasticity of the cloud to scale your compute resources up or down dynamically to match your needs and avoid over- or under-provisioning capacity for your workload.

Common anti-patterns:

- You react to alarms by manually increasing capacity.
- You use the same sizing guidelines (generally static infrastructure) as in on-premises.
- You leave increased capacity after a scaling event instead of scaling back down.

Benefits of establishing this best practice: Configuring and testing the elasticity of compute resources can help you save money, maintain performance benchmarks, and improve reliability as traffic changes.

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

Implementation guidance

AWS provides the flexibility to scale your resources up or down dynamically through a variety of scaling mechanisms in order to meet changes in demand. Combined with compute-related metrics, a dynamic scaling allows workloads to automatically respond to changes and use the optimal set of compute resources to achieve its goal.

You can use a number of different approaches to match supply of resources with demand.

- **Target-tracking approach**: Monitor your scaling metric and automatically increase or decrease capacity as you need it.
- **Predictive scaling**: Scale in anticipation of daily and weekly trends.
- **Schedule-based approach**: Set your own scaling schedule according to predictable load changes.
- **Service scaling**: Choose services (like serverless) that that automatically scale by design.

You must ensure that workload deployments can handle both scale-up and scale-down events.

Implementation steps

- Compute instances, containers, and functions provide mechanisms for elasticity, either in combination with autoscaling or as a feature of the service. Here are some examples of automatic scaling mechanisms:

Autoscaling Mechanism	Where to use
Amazon EC2 Auto Scaling	To ensure you have the correct number of Amazon EC2 instances available to handle the user load for your application.
Application Auto Scaling	To automatically scale the resources for individual AWS services beyond Amazon EC2 such as AWS Lambda functions or Amazon

Autoscaling Mechanism	Where to use
	Elastic Container Service (Amazon ECS) services.
Kubernetes Cluster Autoscaler/Karpenter	To automatically scale Kubernetes clusters.

- Scaling is often discussed related to compute services like Amazon EC2 Instances or AWS Lambda functions. Be sure to also consider the configuration of non-compute services like AWS Glue to match the demand.
- Verify that the metrics for scaling match the characteristics of the workload being deployed. If you are deploying a video transcoding application, 100% CPU utilization is expected and should not be your primary metric. Use the depth of the transcoding job queue instead. You can use a customized metric for your scaling policy if required. To choose the right metrics, consider the following guidance for Amazon EC2:
  - The metric should be a valid utilization metric and describe how busy an instance is.
  - The metric value must increase or decrease proportionally to the number of instances in the Auto Scaling group.
- Make sure that you use dynamic scaling instead of manual scaling for your Auto Scaling group. We also recommend that you use target tracking scaling policies in your dynamic scaling.
- Verify that workload deployments can handle both scaling events (up and down). As an example, you can use Activity history to verify a scaling activity for an Auto Scaling group.
- Evaluate your workload for predictable patterns and proactively scale as you anticipate predicted and planned changes in demand. With predictive scaling, you can eliminate the need to overprovision capacity. For more detail, see Predictive Scaling with Amazon EC2 Auto Scaling.

Resources

Related documents:

- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Amazon ECS Containers: Amazon ECS Container Instances
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Functions: Lambda Function Configuration

- Processor State Control for Your Amazon EC2 Instance
- Deep Dive on Amazon ECS Cluster Auto Scaling
- Introducing Karpenter – An Open-Source High-Performance Kubernetes Cluster Autoscaler

Related videos:

- Amazon EC2 foundations
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2
- Optimize performance and cost for your AWS compute
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Build a cost-, energy-, and resource-efficient compute environment

Related examples:

- Amazon EC2 Auto Scaling Group Examples
- Implement Autoscaling with Karpenter

PERF02-BP06 Use optimized hardware-based compute accelerators

Use hardware accelerators to perform certain functions more efficiently than CPU-based alternatives.

Common anti-patterns:

- In your workload, you haven't benchmarked a general-purpose instance against a purpose-built instance that can deliver higher performance and lower cost.
- You are using hardware-based compute accelerators for tasks that can be more efficient using CPU-based alternatives.
- You are not monitoring GPU usage.

Benefits of establishing this best practice: By using hardware-based accelerators, such as graphics processing units (GPUs) and field programmable gate arrays (FPGAs), you can perform certain processing functions more efficiently.

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

Implementation guidance

Accelerated computing instances provide access to hardware-based compute accelerators such as GPUs and FPGAs. These hardware accelerators perform certain functions like graphic processing or data pattern matching more efficiently than CPU-based alternatives. Many accelerated workloads, such as rendering, transcoding, and machine learning, are highly variable in terms of resource usage. Only run this hardware for the time needed, and decommission them with automation when not required to improve overall performance efficiency.

Implementation steps

- Identify which accelerated computing instances can address your requirements.
- For machine learning workloads, take advantage of purpose-built hardware that is specific to your workload, such as AWS Trainium, AWS Inferentia, and Amazon EC2 DL1. AWS Inferentia instances such as Inf2 instances offer up to 50% better performance/watt over comparable Amazon EC2 instances.
- Collect usage metrics for your accelerated computing instances. For example, you can use CloudWatch agent to collect metrics such as `utilization_gpu` and `utilization_memory` for your GPUs as shown in Collect NVIDIA GPU metrics with Amazon CloudWatch.
- Optimize the code, network operation, and settings of hardware accelerators to make sure that underlying hardware is fully utilized.
  - Optimize GPU settings
  - GPU Monitoring and Optimization in the Deep Learning AMI
  - Optimizing I/O for GPU performance tuning of deep learning training in Amazon SageMaker
- Use the latest high performant libraries and GPU drivers.
- Use automation to release GPU instances when not in use.

Resources

Related documents:

- GPU instances
- Instances with AWS Trainium
- Instances with AWS Inferentia

- Let's Architect! Architecting with custom chips and accelerators

- Accelerated Computing
- Amazon EC2 VT1 Instances
- How do I choose the appropriate Amazon EC2 instance type for my workload?
- Choose the best AI accelerator and model compilation for computer vision inference with Amazon SageMaker

Related videos:

- How to select Amazon EC2 GPU instances for deep learning
- Deploying Cost-Effective Deep Learning Inference

Data management

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

This focus area shares guidance and best practices for optimizing data storage, movement and access patterns, and performance efficiency of data stores.

Best practices

- PERF03-BP01 Use a purpose-built data store that best supports your data access and storage requirements
- PERF03-BP02 Evaluate available configuration options for data store
- PERF03-BP03 Collect and record data store performance metrics
- PERF03-BP04 Implement strategies to improve query performance in data store
- PERF03-BP05 Implement data access patterns that utilize caching

PERF03-BP01 Use a purpose-built data store that best supports your data access and storage requirements

Understand data characteristics (like shareable, size, cache size, access patterns, latency, throughput, and persistence of data) to select the right purpose-built data stores (storage or database) for your workload.

Common anti-patterns:

- You stick to one data store because there is internal experience and knowledge of one particular type of database solution.
- You assume that all workloads have similar data storage and access requirements.
- You have not implemented a data catalog to inventory your data assets.

Benefits of establishing this best practice: Understanding data characteristics and requirements allows you to determine the most efficient and performant storage technology appropriate for your workload needs.

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

Implementation guidance

When selecting and implementing data storage, make sure that the querying, scaling, and storage characteristics support the workload data requirements. AWS provides numerous data storage and database technologies including block storage, object storage, streaming storage, file system, relational, key-value, document, in-memory, graph, time series, and ledger databases. Each data management solution has options and configurations available to you to support your use-cases and data models. By understanding data characteristics and requirements, you can break away from monolithic storage technology and restrictive, one-size-fits-all approaches to focus on managing data appropriately.

Implementation steps

- Conduct an inventory of the various data types that exist in your workload.
- Understand and document data characteristics and requirements, including:
  - Data type (unstructured, semi-structured, relational)
  - Data volume and growth
  - Data durability: persistent, ephemeral, transient
  - ACID (atomicity, consistency, isolation, durability) requirements
  - Data access patterns (read-heavy or write-heavy)
  - Latency
  - Throughput
  - IOPS (input/output operations per second)
  - Data retention period
- Learn about different data stores available for your workload on AWS that can meet your data characteristics (as outlined in PERF01-BP01 Learn about and understand available cloud services and features). Some examples of AWS storage technologies and their key characteristics include:

Type	AWS Services	Key characteristics
Object storage	Amazon S3	Unlimited scalability, high availability, and multiple options for accessibility. Transferring and accessing objects in and out of Amazon S3 can use a service, such as Transfer Acceleration or Access Points, to support your location, security needs, and access patterns.
Archiving storage	Amazon S3 Glacier	Built for data archiving.
Streaming storage	Amazon Kinesis Amazon Managed Streaming for Apache Kafka (Amazon MSK)	Efficient ingestion and storage of streaming data.
Shared file system	Amazon Elastic File System (Amazon EFS)	Mountable file system that can be accessed by multiple types of compute solutions.

Type	AWS Services	Key characteristics
Shared file system	Amazon FSx	Built on the latest AWS compute solutions to support four commonly used file systems: NetApp ONTAP, OpenZFS, Windows File Server, and Lustre. Amazon FSx latency, throughput, and IOPS vary per file system and should be considered when selecting the right file system for your workload needs.
Block storage	Amazon Elastic Block Store (Amazon EBS)	Scalable, high-performance block-storage service designed for Amazon Elastic Compute Cloud (Amazon EC2). Amazon EBS includes SSD-backed storage for transactional, IOPS-intensive workloads and HDD-backe d storage for throughput-intensive workloads.

Type	AWS Services	Key characteristics
Relational database	Amazon Aurora, Amazon RDS, Amazon Redshift.	Designed to support ACID (atomicity, consistency, isolation, durability) transactions, and maintain referential integrity and strong data consistency. Many traditional applications, enterprise resource planning (ERP), customer relationship management (CRM), and ecommerce use relational databases to store their data.
Key-value database	Amazon DynamoDB	Optimized for common access patterns, typically to store and retrieve large volumes of data. High-traffic web apps, ecommerce systems, and gaming applications are typical use-cases for key-value databases.
Document database	Amazon DocumentDB	Designed to store semi-structured data as JSON-like documents. These databases help developers build and update applications such as content management, catalogs, and user profiles quickly.

Type	AWS Services	Key characteristics
In-memory database	Amazon ElastiCache , Amazon MemoryDB for Redis	Used for applications that require real-time access to data, lowest latency and highest throughput. You may use in-memory databases for application caching, session management, gaming leaderboards, low latency ML feature store, microserv ices messaging system, and a high-throughput streaming mechanism
Graph database	Amazon Neptune	Used for applications that must navigate and query millions of relationships between highly connected graph datasets with milliseco nd latency at large scale. Many companies use graph databases for fraud detection, social networkin g, and recommendation engines.
Time Series database	Amazon Timestream	Used to efficiently collect, synthesize, and derive insights from data that changes over time. IoT applications, DevOps, and industrial telemetry can utilize time-series databases.

Type	AWS Services	Key characteristics
Wide column	Amazon Keyspaces (for Apache Cassandra)	Uses tables, rows, and columns, but unlike a relational database, the names and format of the columns can vary from row to row in the same table. You typically see a wide column store in high scale industrial apps for equipment maintenance, fleet management, and route optimization.
Ledger	Amazon Quantum Ledger Database (Amazon QLDB)	Provides a centralized and trusted authority to maintain a scalable, immutable, and cryptographically verifiabl e record of transactions for every application. We see ledger databases used for systems of record, supply chain, registrations, and even banking transactions.

- If you are building a data platform, leverage modern data architecture on AWS to integrate your data lake, data warehouse, and purpose-built data stores.
- The key questions that you need to consider when choosing a data store for your workload are as follows:

Question	Things to consider
How is the data structured?	• If the data is unstructured, consider an object-store such as Amazon S3 or a NoSQL database such as Amazon DocumentDB

Question	Things to consider
	• For key-value data, consider DynamoDB, Amazon ElastiCache for Redis or Amazon MemoryDB for Redis
What level of referential integrity is required?	• For foreign key constraints, relational databases such as Amazon RDS and Aurora can provide this level of integrity. • Typically, within a NoSQL data-model, you would de-normalize your data into a single document or collection of documents to be retrieved in a single request rather than joining across documents or tables.
Is ACID (atomicity, consistency, isolation, durability) compliance required?	• If the ACID properties associated with relational databases are required, consider a relational database such as Amazon RDS and Aurora. • If strong consistency is required for NoSQL database, you can use strongly consistent reads with DynamoDB.
How will the storage requirements change over time? How does this impact scalability?	• Serverless databases such as DynamoDB and Amazon Quantum Ledger Database (Amazon QLDB) will scale dynamically. • Relational databases have upper bounds on provisioned storage, and often must be horizontally partitioned using mechanism s such as sharding once they reach these limits.

Question	Things to consider
What is the proportion of read queries in relation to write queries? Would caching be likely to improve performance?	• Read-heavy workloads can benefit from a caching layer, like ElastiCache or DAX if the database is DynamoDB. • Reads can also be offloaded to read replicas with relational databases such as Amazon RDS.
Does storage and modification (OLTP - Online Transaction Processing) or retrieval and reporting (OLAP - Online Analytical Processing) have a higher priority?	• For high-throughput read as-is transacti onal processing, consider a NoSQL database such as DynamoDB. • For high-throughput and complex read patterns (like join) with consistency use Amazon RDS. • For analytical queries, consider a columnar database such as Amazon Redshift or exporting the data to Amazon S3 and performing analytics using Athena or Amazon QuickSight.
What level of durability does the data require?	• Aurora automatically replicates your data across three Availability Zones within a Region, meaning your data is highly durable with less chance of data loss. • DynamoDB is automatically replicate d across multiple Availability Zones, providing high availability and data durability. • Amazon S3 provides 11 nines of durabilit y. Many database services, such as Amazon RDS and DynamoDB, support exporting data to Amazon S3 for long-term retention and archival.

Question	Things to consider
Is there a desire to move away from commercial database engines or licensing costs?	• Consider open-source engines such as PostgreSQL and MySQL on Amazon RDS or Aurora. • Leverage AWS Database Migration Service and AWS Schema Conversion Tool to perform migrations from commercial database engines to open-source
What is the operational expectation for the database? Is moving to managed services a primary concern?	• Leveraging Amazon RDS instead of Amazon EC2, and DynamoDB or Amazon DocumentDB instead of self-hosting a NoSQL database can reduce operational overhead.
How is the database currently accessed? Is it only application access, or are there business intelligence (BI) users and other connected off-the-shelf applications?	• If you have dependencies on external tooling then you may have to maintain compatibility with the databases they support. Amazon RDS is fully compatible with the difference engine versions that it supports including Microsoft SQL Server, Oracle, MySQL, and PostgreSQL.

- Perform experiments and benchmarking in a non-production environment to identify which data store can address your workload requirements.

Resources

Related documents:

- Amazon EBS Volume Types
- Amazon EC2 Storage
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance

- Amazon S3 Glacier: S3 Glacier Documentation
- Amazon S3: Request Rate and Performance Considerations
- Cloud Storage with AWS
- Amazon EBS I/O Characteristics
- Cloud Databases with AWS
- AWS Database Caching
- DynamoDB Accelerator
- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices
- Amazon DynamoDB best practices
- Choose between Amazon EC2 and Amazon RDS
- Best Practices for Implementing Amazon ElastiCache

Related videos:

- Deep dive on Amazon EBS
- Optimize your storage performance with Amazon S3
- Modernize apps with purpose-built databases
- Amazon Aurora storage demystified: How it all works
- Amazon DynamoDB deep dive: Advanced design patterns

Related examples:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 Examples
- Optimize Data Pattern using Amazon Redshift Data Sharing

- Database Migrations
- MS SQL Server - AWS Database Migration Service (AWS DMS) Replication Demo
- Database Modernization Hands On Workshop
- Amazon Neptune Samples

PERF03-BP02 Evaluate available configuration options for data store

Understand and evaluate the various features and configuration options available for your data stores to optimize storage space and performance for your workload.

Common anti-patterns:

- You only use one storage type, such as Amazon EBS, for all workloads.
- You use provisioned IOPS for all workloads without real-world testing against all storage tiers.
- You are not aware of the configuration options of your chosen data management solution.
- You rely solely on increasing instance size without looking at other available configuration options.
- You are not testing the scaling characteristics of your data store.

Benefits of establishing this best practice: By exploring and experimenting with the data store configurations, you may be able to reduce the cost of infrastructure, improve performance, and lower the effort required to maintain your workloads.

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

Implementation guidance

A workload could have one or more data stores used based on data storage and access requirements. To optimize your performance efficiency and cost, you must evaluate data access patterns to determine the appropriate data store configurations. While you explore data store options, take into consideration various aspects such as the storage options, memory, compute, read replica, consistency requirements, connection pooling, and caching options. Experiment with these various configuration options to improve performance efficiency metrics.

Implementation steps

- Understand the current configurations (like instance type, storage size, or database engine version) of your data store.
- Review AWS documentation and best practices to learn about recommended configuration options that can help improve the performance of your data store. Key data store options to consider are the following:

Configuration option	Examples
Offloading reads (like read replicas and caching)	• For DynamoDB tables, you can offload reads using DAX for caching. • You can create an Amazon ElastiCache for Redis cluster and configure your applicati on to read from the cache first, falling back to the database if the requested item is not present. • Relational databases such as Amazon RDS and Aurora, and provisioned NoSQL databases such as Neptune and Amazon DocumentDB all support adding read replicas to offload the read portions of the workload. • Serverless databases such as DynamoDB will scale automatically. Ensure that you have enough read capacity units (RCU) provisioned to handle the workload.
Scaling writes (like partition key sharding or introducing a queue)	• For relational databases, you can increase the size of the instance to accommoda te an increased workload or increase the provisioned IOPs to allow for an increased throughput to the underlying storage. • You can also introduce a queue in front of your database rather than writing directly to the database. This pattern allows you to

Configuration option	Examples
	decouple the ingestion from the database and control the flow-rate so the database does not get overwhelmed. • Batching your write requests rather than creating many short-lived transactions can help improve throughput in high-write volume relational databases. • Serverless databases like DynamoDB can scale the write throughput automatically or by adjusting the provisioned write capacity units (WCU) depending on the capacity mode. • You can still run into issues with hot partitions when you reach the throughput limits for a given partition key. This can be mitigated by choosing a more evenly distributed partition key or by write-sharding the partition key.
Policies to manage the lifecycle of your datasets	• You can use Amazon S3 Lifecycle to manage your objects throughout their lifecycle. If your access patterns are unknown, changing, or unpredictable, you can use Amazon S3 Intelligent-Tiering, which monitors access patterns and automatically moves objects that have not been accessed to lower-cost access tiers. You can leverage Amazon S3 Storage Lens metrics to identify optimization opportunities and gaps in lifecycle management. • Amazon EFS lifecycle management automatically manages file storage for your file systems.

Configuration option	Examples
Connection management and pooling	• Amazon RDS Proxy can be used with Amazon RDS and Aurora to manage connections to the database. • Serverless databases such as DynamoDB do not have connections associated with them, but consider the provisioned capacity and automatic scaling policies to deal with spikes in load.

- Perform experiments and benchmarking in non-production environment to identify which configuration option can address your workload requirements.
- Once you have experimented, plan your migration and validate your performance metrics.
- Use AWS monitoring (like Amazon CloudWatch) and optimization (like Amazon S3 Storage Lens) tools to continuously optimize your data store using real-world usage pattern.

Resources

Related documents:

- Cloud Storage with AWS
- Amazon EBS Volume Types
- Amazon EC2 Storage
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon S3 Glacier: S3 Glacier Documentation
- Amazon S3: Request Rate and Performance Considerations
- Cloud Storage with AWS
- Cloud Storage with AWS
- Amazon EBS I/O Characteristics
- Cloud Databases with AWS
- AWS Database Caching

- DynamoDB Accelerator
- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices
- Amazon DynamoDB best practices

Related videos:

- Deep dive on Amazon EBS
- Optimize your storage performance with Amazon S3
- Modernize apps with purpose-built databases
- Amazon Aurora storage demystified: How it all works
- Amazon DynamoDB deep dive: Advanced design patterns

Related examples:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 Examples
- Amazon DynamoDB Examples
- AWS Database migration samples
- Database Modernization Workshop
- Working with parameters on your Amazon RDS for Postgress DB

PERF03-BP03 Collect and record data store performance metrics

Track and record relevant performance metrics for your data store to understand how your data management solutions are performing. These metrics can help you optimize your data store, verify

that your workload requirements are met, and provide a clear overview on how the workload performs.

Common anti-patterns:

- You only use manual log file searching for metrics.
- You only publish metrics to internal tools used by your team and don't have a comprehensive picture of your workload.
- You only use the default metrics recorded by your selected monitoring software.
- You only review metrics when there is an issue.
- You only monitor system-level metrics and do not capture data access or usage metrics.

Benefits of establishing this best practice: Establishing a performance baseline helps you understand the normal behavior and requirements of workloads. Abnormal patterns can be identified and debugged faster, improving the performance and reliability of the data store.

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

Implementation guidance

To monitor the performance of your data stores, you must record multiple performance metrics over a period of time. This allows you to detect anomalies, as well as measure performance against business metrics to verify you are meeting your workload needs.

Metrics should include both the underlying system that is supporting the data store and the database metrics. The underlying system metrics might include CPU utilization, memory, available disk storage, disk I/O, cache hit ratio, and network inbound and outbound metrics, while the data store metrics might include transactions per second, top queries, average queries rates, response times, index usage, table locks, query timeouts, and number of connections open. This data is crucial to understand how the workload is performing and how the data management solution is used. Use these metrics as part of a data-driven approach to tune and optimize your workload's resources.

Use tools, libraries, and systems that record performance measurements related to database performance.

Implementation steps

1. Identify the key performance metrics for your data store to track.

   a. Amazon S3 Metrics and dimensions
   b. Monitoring metrics for in an Amazon RDS instance
   c. Monitoring DB load with Performance Insights on Amazon RDS
   d. Overview of Enhanced Monitoring
   e. DynamoDB Metrics and dimensions
   f. Monitoring DynamoDB Accelerator
   g. Monitoring Amazon MemoryDB for Redis with Amazon CloudWatch
   h. Which Metrics Should I Monitor?
   i. Monitoring Amazon Redshift cluster performance
   j. Timestream metrics and dimensions
   k. Amazon CloudWatch metrics for Amazon Aurora
   l. Logging and monitoring in Amazon Keyspaces (for Apache Cassandra)
   m. Monitoring Amazon Neptune Resources
2. Use an approved logging and monitoring solution to collect these metrics. Amazon CloudWatch can collect metrics across the resources in your architecture. You can also collect and publish custom metrics to surface business or derived metrics. Use CloudWatch or third-party solutions to set alarms that indicate when thresholds are breached.
3. Check if data store monitoring can benefit from a machine learning solution that detects performance anomalies.
   a. Amazon DevOps Guru for Amazon RDS provides visibility into performance issues and makes recommendations for corrective actions.
4. Configure data retention in your monitoring and logging solution to match your security and operational goals.
   a. Default data retention for CloudWatch metrics
   b. Default data retention for CloudWatch Logs

Resources

Related documents:

- AWS Database Caching
- Amazon Athena top 10 performance tips

- Amazon Aurora best practices
- DynamoDB Accelerator
- Amazon DynamoDB best practices
- Amazon Redshift Spectrum best practices
- Amazon Redshift performance
- Cloud Databases with AWS
- Amazon RDS Performance Insights

Related videos:

- AWS purpose-built databases
- Amazon Aurora storage demystified: How it all works
- Amazon DynamoDB deep dive: Advanced design patterns
- Best Practices for Monitoring Redis Workloads on Amazon ElastiCache

Related examples:

- Level 100: Monitoring with CloudWatch Dashboards
- AWS Dataset Ingestion Metrics Collection Framework
- Amazon RDS Monitoring Workshop

PERF03-BP04 Implement strategies to improve query performance in data store

Implement strategies to optimize data and improve data query to enable more scalability and efficient performance for your workload.

Common anti-patterns:

- You do not partition data in your data store.
- You store data in only one file format in your data store.
- You do not use indexes in your data store.

Benefits of establishing this best practice: Optimizing data and query performance results in more efficiency, lower cost, and improved user experience.

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

Implementation guidance

Data optimization and query tuning are critical aspects of performance efficiency in a data store, as they impact the performance and responsiveness of the entire cloud workload. Unoptimized queries can result in greater resource usage and bottlenecks, which reduce the overall efficiency of a data store.

Data optimization includes several techniques to ensure efficient data storage and access. This also help to improve the query performance in a data store. Key strategies include data partitioning, data compression, and data denormalization, which help data to be optimized for both storage and access.

Implementation steps

- Understand and analyze the critical data queries which are performed in your data store.
- Identify the slow-running queries in your data store and use query plans to understand their current state.
  - Analyzing the query plan in Amazon Redshift
  - Using EXPLAIN and EXPLAIN ANALYZE in Athena
- Implement strategies to improve the query performance. Some of the key strategies include:
  - Using a columnar file format (like Parquet or ORC).
  - Compressing data in the data store to reduce storage space and I/O operation.
  - Data partitioning to split data into smaller parts and reduce data scanning time.
    - Partitioning data in Athena
    - Partitions and data distribution
  - Data indexing on the common columns in the query.
  - Choose the right join operation for the query. When you join two tables, specify the larger table on the left side of join and the smaller table on the right side of the join.
  - Distributed caching solution to improve latency and reduce the number of database I/O operation.
  - Regular maintenance such as executing statistics.

- Experiment and test strategies in a non-production environment.

Resources

Related documents:

- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- AWS Database Caching
- Best Practices for Implementing Amazon ElastiCache
- Partitioning data in Athena

Related videos:

- Optimize Data Pattern using Amazon Redshift Data Sharing
- Optimize Amazon Athena Queries with New Query Analysis Tools

Related examples:

- Amazon EFS CSI Driver

PERF03-BP05 Implement data access patterns that utilize caching

Implement access patterns that can benefit from caching data for fast retrieval of frequently accessed data.

Common anti-patterns:

- You cache data that changes frequently.
- You rely on cached data as if it is durably stored and always available.
- You don't consider the consistency of your cached data.
- You don't monitor the efficiency of your caching implementation.

Benefits of establishing this best practice: Storing data in a cache can improve read latency, read throughput, user experience, and overall efficiency, as well as reduce costs.

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

Implementation guidance

A cache is a software or hardware component aimed at storing data so that future requests for the same data can be served faster or more efficiently. The data stored in a cache can be reconstructed if lost by repeating an earlier calculation or fetching it from another data store.

Data caching can be one of the most effective strategies to improve your overall application performance and reduce burden on your underlying primary data sources. Data can be cached at multiple levels in the application, such as within the application making remote calls, known as *client-side caching*, or by using a fast secondary service for storing the data, known as *remote caching*.

Client-side caching

With client-side caching, each client (an application or service that queries the backend datastore) can store the results of their unique queries locally for a specified amount of time. This can reduce the number of requests across the network to a datastore by checking the local client cache first. If the results are not present, the application can then query the datastore and store those results locally. This pattern allows each client to store data in the closest location possible (the client itself), resulting in the lowest possible latency. Clients can also continue to serve some queries when the backend datastore is unavailable, increasing the availability of the overall system.

One disadvantage of this approach is that when multiple clients are involved, they may store the same cached data locally. This results in both duplicate storage usage and data inconsistency between those clients. One client might cache the results of a query, and one minute later another client can run the same query and get a different result.

Remote caching

To solve the issue of duplicate data between clients, a fast external service, or *remote cache*, can be used to store the queried data. Instead of checking a local data store, each client will check the remote cache before querying the backend datastore. This strategy allows for more consistent responses between clients, better efficiency in stored data, and a higher volume of cached data because the storage space scales independently of clients.

The disadvantage of a remote cache is that the overall system may see a higher latency, as an additional network hop is required to check the remote cache. Client-side caching can be used alongside remote caching for multi-level caching to improve the latency.

Implementation steps

1. Identify databases, APIs and network services that could benefit from caching. Services that have heavy read workloads, have a high read-to-write ratio, or are expensive to scale are candidates for caching.
   - Database Caching
   - Enabling API caching to enhance responsiveness
2. Identify the appropriate type of caching strategy that best fits your access pattern.
   - Caching strategies
   - AWS Caching Solutions
3. Follow Caching Best Practices for your data store.
4. Configure a cache invalidation strategy, such as a time-to-live (TTL), for all data that balances freshness of data and reducing pressure on backend datastore.
5. Enable features such as automatic connection retries, exponential backoff, client-side timeouts, and connection pooling in the client, if available, as they can improve performance and reliability.
   - Best practices: Redis clients and Amazon ElastiCache for Redis
6. Monitor cache hit rate with a goal of 80% or higher. Lower values may indicate insufficient cache size or an access pattern that does not benefit from caching.
   - Which metrics should I monitor?
   - Best practices for monitoring Redis workloads on Amazon ElastiCache
   - Monitoring best practices with Amazon ElastiCache for Redis using Amazon CloudWatch
7. Implement data replication to offload reads to multiple instances and improve data read performance and availability.

Resources

Related documents:

- Using the Amazon ElastiCache Well-Architected Lens

- Monitoring best practices with Amazon ElastiCache for Redis using Amazon CloudWatch
- Which Metrics Should I Monitor?
- Performance at Scale with Amazon ElastiCache whitepaper
- Caching challenges and strategies

Related videos:

- Amazon ElastiCache Learning Path
- Design for success with Amazon ElastiCache best practices

Related examples:

- Boosting MySQL database performance with Amazon ElastiCache for Redis

Networking and content delivery

The optimal networking solution for a workload varies based on latency, throughput requirements, jitter, and bandwidth. Physical constraints, such as user or on-premises resources, determine location options. These constraints can be offset with edge locations or resource placement.

On AWS, networking is virtualized and is available in a number of different types and configurations. This makes it easier to match your networking needs. AWS offers product features (for example, Enhanced Networking, Amazon EC2 networking optimized instances, Amazon S3 transfer acceleration, and dynamic Amazon CloudFront) to optimize network traffic. AWS also offers networking features (for example, Amazon Route 53 latency routing, Amazon VPC endpoints, AWS Direct Connect, and AWS Global Accelerator) to reduce network distance or jitter.

This focus area shares guidance and best practices to design, configure, and operate efficient networking and content delivery solutions in the cloud.

Best practices

- PERF04-BP01 Understand how networking impacts performance
- PERF04-BP02 Evaluate available networking features
- PERF04-BP03 Choose appropriate dedicated connectivity or VPN for your workload
- PERF04-BP04 Use load balancing to distribute traffic across multiple resources
- PERF04-BP05 Choose network protocols to improve performance
- PERF04-BP06 Choose your workload's location based on network requirements
- PERF04-BP07 Optimize network configuration based on metrics

PERF04-BP01 Understand how networking impacts performance

Analyze and understand how network-related decisions impact your workload to provide efficient performance and improved user experience.

Common anti-patterns:

- All traffic flows through your existing data centers.

- You route all traffic through central firewalls instead of using cloud-native network security tools.
- You provision AWS Direct Connect connections without understanding actual usage requirements.
- You don't consider workload characteristics and encryption overhead when defining your networking solutions.
- You use on-premises concepts and strategies for networking solutions in the cloud.

Benefits of establishing this best practice: Understanding how networking impacts workload performance helps you identify potential bottlenecks, improve user experience, increase reliability, and lower operational maintenance as the workload changes.

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

Implementation guidance

The network is responsible for the connectivity between application components, cloud services, edge networks, and on-premises data, and therefore it can heavily impact workload performance. In addition to workload performance, user experience can be also impacted by network latency, bandwidth, protocols, location, network congestion, jitter, throughput, and routing rules.

Have a documented list of networking requirements from the workload including latency, packet size, routing rules, protocols, and supporting traffic patterns. Review the available networking solutions and identify which service meets your workload networking characteristics. Cloud-based networks can be quickly rebuilt, so evolving your network architecture over time is necessary to improve performance efficiency.

Implementation steps:

1. Define and document networking performance requirements, including metrics such as network latency, bandwidth, protocols, locations, traffic patterns (spikes and frequency), throughput, encryption, inspection, and routing rules.
2. Learn about key AWS networking services like VPCs, AWS Direct Connect, Elastic Load Balancing (ELB), and Amazon Route 53.
3. Capture the following key networking characteristics:

Characteristics	Tools and metrics
Foundational networking characteristics	• VPC Flow Logs • AWS Transit Gateway Flow Logs • AWS Transit Gateway metrics • AWS PrivateLink metrics
Application networking characteristics	• Elastic Fabric Adapter • AWS App Mesh metrics • Amazon API Gateway metrics
Edge networking characteristics	• Amazon CloudFront metrics • Amazon Route 53 metrics • AWS Global Accelerator metrics
Hybrid networking characteristics	• AWS Direct Connect metrics • AWS Site-to-Site VPN metrics • AWS Client VPN metrics • AWS Cloud WAN metrics
Security networking characteristics	• AWS Shield, AWS WAF, and AWS Network Firewall metrics
Tracing characteristics	• AWS X-Ray • VPC Reachability Analyzer • Network Access Analyzer • Amazon Inspector • Amazon CloudWatch RUM

4. Benchmark and test network performance:
    a. Benchmark network throughput, as some factors can affect Amazon EC2 network performance when instances are in the same VPC. Measure the network bandwidth between Amazon EC2 Linux instances in the same VPC.
    b. Perform load tests to experiment with networking solutions and options.

Resources

Related documents:

- Application Load Balancer
- EC2 Enhanced Networking on Linux
- EC2 Enhanced Networking on Windows
- EC2 Placement Groups
- Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances
- Network Load Balancer
- Networking Products with AWS
- Transit Gateway
- Transitioning to latency-based routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs

Related videos:

- Connectivity to AWS and hybrid AWS network architectures
- Optimizing Network Performance for Amazon EC2 Instances
- Improve Global Network Performance for Applications
- EC2 Instances and Performance Optimization Best Practices
- Optimizing Network Performance for Amazon EC2 Instances
- Networking best practices and tips with the Well-Architected Framework
- AWS networking best practices in large-scale migrations

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF04-BP02 Evaluate available networking features

Evaluate networking features in the cloud that may increase performance. Measure the impact of these features through testing, metrics, and analysis. For example, take advantage of network-level features that are available to reduce latency, network distance, or jitter.

Common anti-patterns:

- You stay within one Region because that is where your headquarters is physically located.
- You use firewalls instead of security groups for filtering traffic.
- You break TLS for traffic inspection rather than relying on security groups, endpoint policies, and other cloud-native functionality.
- You only use subnet-based segmentation instead of security groups.

Benefits of establishing this best practice: Evaluating all service features and options can increase your workload performance, reduce the cost of infrastructure, decrease the effort required to maintain your workload, and increase your overall security posture. You can use the global AWS backbone to provide the optimal networking experience for your customers.

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

Implementation guidance

AWS offers services like AWS Global Accelerator and Amazon CloudFront that can help improve network performance, while most AWS services have product features (such as the Amazon S3 Transfer Acceleration feature) to optimize network traffic.

Review which network-related configuration options are available to you and how they could impact your workload. Performance optimization depends on understanding how these options interact with your architecture and the impact that they will have on both measured performance and user experience.

Implementation steps

- Create a list of workload components.
  - Consider using AWS Cloud WAN to build, manage and monitor your organization's network when building a unified global network.

  - Monitor your global and core networks with Amazon CloudWatch Logs metrics. Leverage Amazon CloudWatch RUM, which provides insights to help to identify, understand, and enhance users' digital experience.
  - View aggregate network latency between AWS Regions and Availability Zones, as well as within each Availability Zone, using AWS Network Manager to gain insight into how your application performance relates to the performance of the underlying AWS network.
  - Use an existing configuration management database (CMDB) tool or a service such as AWS Config to create an inventory of your workload and how it's configured.
- If this is an existing workload, identify and document the benchmark for your performance metrics, focusing on the bottlenecks and areas to improve. Performance-related networking metrics will differ per workload based on business requirements and workload characteristics. As a start, these metrics might be important to review for your workload: bandwidth, latency, packet loss, jitter, and retransmits.
- If this is a new workload, perform load tests to identify performance bottlenecks.
- For the performance bottlenecks you identify, review the configuration options for your solutions to identify performance improvement opportunities. Check out the following key networking options and features:

Improvement opportunity	Solution
Network path or routes	Use Network Access Analyzer to identify paths or routes.
Network protocols	See PERF04-BP05 Choose network protocols to improve performance
Network topology	Evaluate your operational and performan ce tradeoffs between VPC Peering and AWS Transit Gateway when connecting multiple accounts. AWS Transit Gateway simplifie s how you interconnect all of your VPCs, which can span across thousands of AWS accounts and into on-premises networks. Share your AWS Transit Gateway between multiple accounts using AWS Resource Access Manager.

Improvement opportunity	Solution
	See PERF04-BP03 Choose appropriate dedicated connectivity or VPN for your workload
Network services	AWS Global Accelerator is a networking service that improves the performance of your users' traffic by up to 60% using the AWS global network infrastructure. Amazon CloudFront can improve the performance of your workload content delivery and latency globally. Use Lambda@edge to run functions that customize the content that CloudFront delivers closer to the users, reduce latency, and improve performance. Amazon Route 53 offers latency-based routing, geolocation routing, geoproximity routing, and IP-based routing options to help you improve your workload's performan ce for a global audience. Identify which routing option would optimize your workload performance by reviewing your workload traffic and user location when your workload is distributed globally.

Improvement opportunity	Solution
Storage resource features	Amazon S3 Transfer Acceleration is a feature that lets external users benefit from the networking optimizations of CloudFront to upload data to Amazon S3. This improves the ability to transfer large amounts of data from remote locations that don't have dedicated connectivity to the AWS Cloud. Amazon S3 Multi-Region Access Points replicates content to multiple Regions and simplifies the workload by providing one access point. When a Multi-Region Access Point is used, you can request or write data to Amazon S3 with the service identifying the lowest latency bucket.

Improvement opportunity	Solution
Compute resource features	Elastic Network Interfaces (ENA) used by Amazon EC2 instances, containers, and Lambda functions are limited on a per-flow basis. Review your placement groups to optimize your EC2 networking throughput. To avoid a bottleneck on a per flow-basis, design your application to use multiple flows. To monitor and get visibility into your compute related networking metrics, use CloudWatch Metrics and ethtool. The `ethtool` command is included in the ENA driver and exposes additional network-related metrics that can be published as a custom metric to CloudWatch. Amazon Elastic Network Adapters (ENA) provide further optimization by delivering better throughput for your instances within a cluster placement group. Elastic Fabric Adapter (EFA) is a network interface for Amazon EC2 instances that allows you to run workloads requiring high levels of internode communications at scale on AWS. Amazon EBS-optimized instances use an optimized configuration stack and provide additional, dedicated capacity to increase the Amazon EBS I/O.

Resources

Related documents:

- Amazon EBS - Optimized Instances
- Application Load Balancer
- EC2 Enhanced Networking on Linux
- EC2 Enhanced Networking on Windows
- EC2 Placement Groups
- Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances
- Network Load Balancer
- Networking Products with AWS
- AWS Transit Gateway
- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs

Related videos:

- Connectivity to AWS and hybrid AWS network architectures
- Optimizing Network Performance for Amazon EC2 Instances
- AWS Global Accelerator

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF04-BP03 Choose appropriate dedicated connectivity or VPN for your workload

When hybrid connectivity is required to connect on-premises and cloud resources, provision adequate bandwidth to meet your performance requirements. Estimate the bandwidth and latency requirements for your hybrid workload. These numbers will drive your sizing requirements.

Common anti-patterns:

- You only evaluate VPN solutions for your network encryption requirements.
- You do not evaluate backup or redundant connectivity options.
- You do not identify all workload requirements (encryption, protocol, bandwidth, and traffic needs).

Benefits of establishing this best practice: Selecting and configuring appropriate connectivity solutions will increase the reliability of your workload and maximize performance. By identifying workload requirements, planning ahead, and evaluating hybrid solutions, you can minimize expensive physical network changes and operational overhead while increasing your time-to-value.

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

Implementation guidance

Develop a hybrid networking architecture based on your bandwidth requirements. AWS Direct Connect allows you to connect your on-premises network privately with AWS. It is suitable when you need high-bandwidth and low-latency while achieving consistent performance. A VPN connection establishes secure connection over the internet. It is used when only a temporary connection is required, when cost is a factor, or as a contingency while waiting for resilient physical network connectivity to be established when using AWS Direct Connect.

If your bandwidth requirements are high, you might consider multiple AWS Direct Connect or VPN services. Traffic can be load balanced across services, although we don't recommend load balancing between AWS Direct Connect and VPN because of the latency and bandwidth differences.

Implementation steps

1. Estimate the bandwidth and latency requirements of your existing applications.
   a. For existing workloads that are moving to AWS, leverage the data from your internal network monitoring systems.
   b. For new or existing workloads for which you don’t have monitoring data, consult with the product owners to determine adequate performance metrics and provide a good user experience.
2. Select dedicated connection or VPN as your connectivity option. Based on all workload requirements (encryption, bandwidth, and traffic needs), you can either choose AWS Direct Connect or AWS VPN (or both). The following diagram can help you choose the appropriate connection type.

   a. AWS Direct Connect provides dedicated connectivity to the AWS environment, from 50 Mbps up to 100 Gbps, using either dedicated connections or hosted connections. This gives you managed and controlled latency and provisioned bandwidth so your workload can connect efficiently to other environments. Using AWS Direct Connect partners, you can have end-to-end connectivity from multiple environments, providing an extended network with consistent performance. AWS offers scaling direct connect connection bandwidth using either native 100 Gbps, link aggregation group (LAG), or BGP equal-cost multipath (ECMP).

   b. The AWS Site-to-Site VPN provides a managed VPN service supporting internet protocol security (IPsec). When a VPN connection is created, each VPN connection includes two tunnels for high availability.

3. Follow AWS documentation to choose an appropriate connectivity option:

   a. If you decide to use AWS Direct Connect, select the appropriate bandwidth for your connectivity.

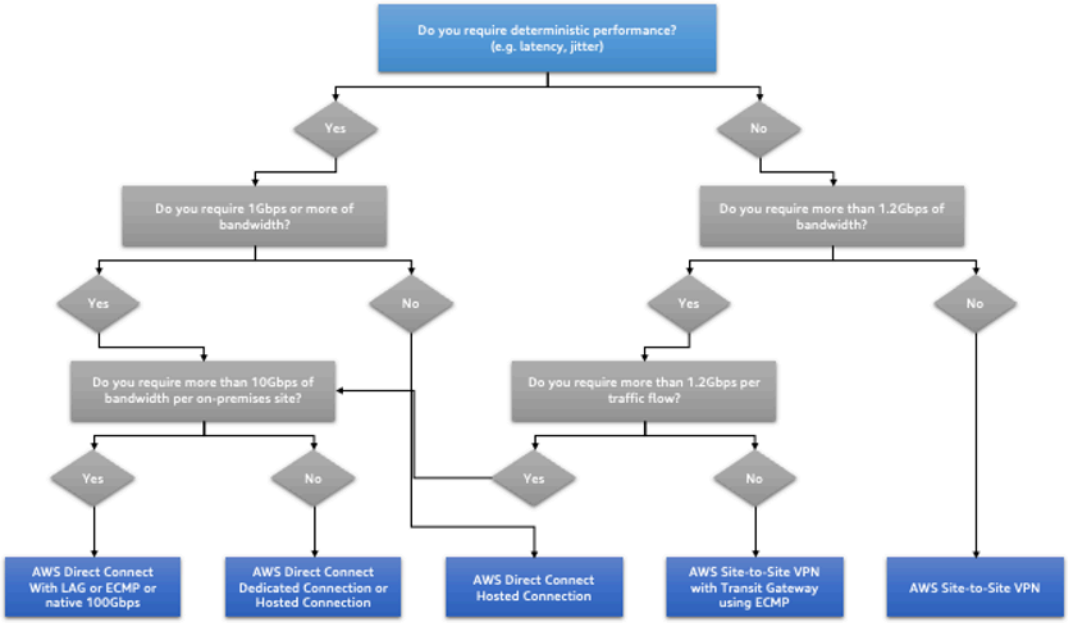
   b. If you are using an AWS Site-to-Site VPN across multiple locations to connect to an AWS Region, use an accelerated Site-to-Site VPN connection for the opportunity to improve network performance.

   c. If your network design consists of IPSec VPN connection over AWS Direct Connect, consider using Private IP VPN to improve security and achieve segmentation. AWS Site-to-Site Private IP VPN is deployed on top of transit virtual interface (VIF).

   d. AWS Direct Connect SiteLink allows creating low-latency and redundant connections between your data centers worldwide by sending data over the fastest path between AWS Direct Connect locations, bypassing AWS Regions.

4. Validate your connectivity setup before deploying to production. Perform security and performance testing to assure it meets your bandwidth, reliability, latency, and compliance requirements.

5. Regularly monitor your connectivity performance and usage and optimize if required.



Deterministic performance flowchart

Resources

Related documents:

- Network Load Balancer
- Networking Products with AWS
- AWS Transit Gateway
- Transitioning to latency-based Routing in Amazon Route 53
- VPC Endpoints
- Site-to-Site VPN
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure
- AWS Direct Connect
- Client VPN

Related videos:

- Connectivity to AWS and hybrid AWS network architectures
- Optimizing Network Performance for Amazon EC2 Instances

- AWS Global Accelerator
- AWS Direct Connect
- AWS Transit Gateway Connect
- VPN Solutions
- Security with VPN Solutions

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF04-BP04 Use load balancing to distribute traffic across multiple resources

Distribute traffic across multiple resources or services to allow your workload to take advantage of the elasticity that the cloud provides. You can also use load balancing for offloading encryption termination to improve performance, reliability and manage and route traffic effectively.

Common anti-patterns:

- You don't consider your workload requirements when choosing the load balancer type.
- You don't leverage the load balancer features for performance optimization.
- The workload is exposed directly to the internet without a load balancer.
- You route all internet traffic through existing load balancers.
- You use generic TCP load balancing and making each compute node handle SSL encryption.

Benefits of establishing this best practice: A load balancer handles the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones and enables high availability, automatic scaling, and better utilization for your workload.

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

Implementation guidance

Load balancers act as the entry point for your workload, from which point they distribute the traffic to your backend targets, such as compute instances or containers, to improve utilization.

Choosing the right load balancer type is the first step to optimize your architecture. Start by listing your workload characteristics, such as protocol (like TCP, HTTP, TLS, or WebSockets), target type (like instances, containers, or serverless), application requirements (like long running connections, user authentication, or stickiness), and placement (like Region, Local Zone, Outpost, or zonal isolation).

AWS provides multiple models for your applications to use load balancing. Application Load Balancer is best suited for load balancing of HTTP and HTTPS traffic and provides advanced request routing targeted at the delivery of modern application architectures, including microservices and containers.

Network Load Balancer is best suited for load balancing of TCP traffic where extreme performance is required. It is capable of handling millions of requests per second while maintaining ultra-low latencies, and it is optimized to handle sudden and volatile traffic patterns.

Elastic Load Balancing provides integrated certificate management and SSL/TLS decryption, allowing you the flexibility to centrally manage the SSL settings of the load balancer and offload CPU intensive work from your workload.

After choosing the right load balancer, you can start leveraging its features to reduce the amount of effort your backend has to do to serve the traffic.

For example, using both Application Load Balancer (ALB) and Network Load Balancer (NLB), you can perform SSL/TLS encryption offloading, which is an opportunity to avoid the CPU-intensive TLS handshake from being completed by your targets and also to improve certificate management.

When you configure SSL/TLS offloading in your load balancer, it becomes responsible for the encryption of the traffic from and to clients while delivering the traffic unencrypted to your backends, freeing up your backend resources and improving the response time for the clients.

Application Load Balancer can also serve HTTP/2 traffic without needing to support it on your targets. This simple decision can improve your application response time, as HTTP/2 uses TCP connections more efficiently.

Your workload latency requirements should be considered when defining the architecture. As an example, if you have a latency-sensitive application, you may decide to use Network Load Balancer,

which offers extremely low latencies. Alternatively, you may decide to bring your workload closer to your customers by leveraging Application Load Balancer in AWS Local Zones or even AWS Outposts.

Another consideration for latency-sensitive workloads is cross-zone load balancing. With cross-zone load balancing, each load balancer node distributes traffic across the registered targets in all allowed Availability Zones.

Use Auto Scaling integrated with your load balancer. One of the key aspects of a performance efficient system has to do with right-sizing your backend resources. To do this, you can leverage load balancer integrations for backend target resources. Using the load balancer integration with Auto Scaling groups, targets will be added or removed from the load balancer as required in response to incoming traffic. Load balancers can also integrate with Amazon ECS and Amazon EKS for containerized workloads.

- Amazon ECS - Service load balancing
- Application load balancing on Amazon EKS
- Network load balancing on Amazon EKS

Implementation steps

- Define your load balancing requirements including traffic volume, availability and application scalability.
- Choose the right load balancer type for your application.
  - Use Application Load Balancer for HTTP/HTTPS workloads.
  - Use Network Load Balancer for non-HTTP workloads that run on TCP or UDP.
  - Use a combination of both (ALB as a target of NLB) if you want to leverage features of both products. For example, you can do this if you want to use the static IPs of NLB together with HTTP header based routing from ALB, or if you want to expose your HTTP workload to an AWS PrivateLink.
  - For a full comparison of load balancers, see ELB product comparison.
- Use SSL/TLS offloading if possible.
  - Configure HTTPS/TLS listeners with both Application Load Balancer and Network Load Balancer integrated with AWS Certificate Manager.
  - Note that some workloads may require end-to-end encryption for compliance reasons. In this case, it is a requirement to allow encryption at the targets.

  - For security best practices, see SEC09-BP02 Enforce encryption in transit.
- Select the right routing algorithm (only ALB).
  - The routing algorithm can make a difference in how well-used your backend targets are and therefore how they impact performance. For example, ALB provides two options for routing algorithms:
  - **Least outstanding requests:** Use to achieve a better load distribution to your backend targets for cases when the requests for your application vary in complexity or your targets vary in processing capability.
  - **Round robin:** Use when the requests and targets are similar, or if you need to distribute requests equally among targets.
- Consider cross-zone or zonal isolation.
  - Use cross-zone turned off (zonal isolation) for latency improvements and zonal failure domains. It is turned off by default in NLB and in ALB you can turn it off per target group.
  - Use cross-zone turned on for increased availability and flexibility. By default, cross-zone is turned on for ALB and in NLB you can turn it on per target group.
- Turn on HTTP keep-alives for your HTTP workloads (only ALB). With this feature, the load balancer can reuse backend connections until the keep-alive timeout expires, improving your HTTP request and response time and also reducing resource utilization on your backend targets. For detail on how to do this for Apache and Nginx, see What are the optimal settings for using Apache or NGINX as a backend server for ELB?
- Turn on monitoring for your load balancer.
  - Turn on access logs for your Application Load Balancer and Network Load Balancer.
  - The main fields to consider for ALB are `request_processing_time`, `request_processing_time`, and `response_processing_time`.
  - The main fields to consider for NLB are `connection_time` and `tls_handshake_time`.
  - Be ready to query the logs when you need them. You can use Amazon Athena to query both ALB logs and NLB logs.
  - Create alarms for performance related metrics such as `TargetResponseTime` for ALB.

Resources

Related documents:

- ELB product comparison
- AWS Global Infrastructure
- Improving Performance and Reducing Cost Using Availability Zone Affinity
- Step by step for Log Analysis with Amazon Athena
- Querying Application Load Balancer logs
- Monitor your Application Load Balancers
- Monitor your Network Load Balancer
- Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group

Related videos:

- AWS re:Invent 2018: Elastic Load Balancing: Deep Dive and Best Practices
- AWS re:Invent 2021 - How to choose the right load balancer for your AWS workloads
- AWS re:Inforce 2022 - How to use Elastic Load Balancing to enhance your security posture at scale
- AWS re:Invent 2019: Get the most from Elastic Load Balancing for different workloads

Related examples:

- CDK and AWS CloudFormation samples for Log Analysis with Amazon Athena

PERF04-BP05 Choose network protocols to improve performance

Make decisions about protocols for communication between systems and networks based on the impact to the workload's performance.

There is a relationship between latency and bandwidth to achieve throughput. If your file transfer is using Transmission Control Protocol (TCP), higher latencies will most likely reduce overall throughput. There are approaches to fix this with TCP tuning and optimized transfer protocols, but one solution is to use User Datagram Protocol (UDP).

Common anti-patterns:

- You use TCP for all workloads regardless of performance requirements.

Benefits of establishing this best practice: Verifying that an appropriate protocol is used for communication between users and workload components helps improve overall user experience for your applications. For instance, connection-less UDP allows for high speed, but it doesn't offer retransmission or high reliability. TCP is a full featured protocol, but it requires greater overhead for processing the packets.

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

Implementation guidance

If you have the ability to choose different protocols for your application and you have expertise in this area, optimize your application and end-user experience by using a different protocol. Note that this approach comes with significant difficulty and should only be attempted if you have optimized your application in other ways first.

A primary consideration for improving your workload's performance is to understand the latency and throughput requirements, and then choose network protocols that optimize performance.

When to consider using TCP

TCP provides reliable data delivery, and can be used for communication between workload components where reliability and guaranteed delivery of data is important. Many web-based applications rely on TCP-based protocols, such as HTTP and HTTPS, to open TCP sockets for communication between application components. Email and file data transfer are common applications that also make use of TCP, as it is a simple and reliable transfer mechanism between application components. Using TLS with TCP can add some overhead to the communication, which can result in increased latency and reduced throughput, but it comes with the advantage of security. The overhead comes mainly from the added overhead of the handshake process, which can take several round-trips to complete. Once the handshake is complete, the overhead of encrypting and decrypting data is relatively small.

When to consider using UDP

UDP is a connection-less-oriented protocol and is therefore suitable for applications that need fast, efficient transmission, such as log, monitoring, and VoIP data. Also, consider using UDP if you have workload components that respond to small queries from large numbers of clients to ensure optimal performance of the workload. Datagram Transport Layer Security (DTLS) is the UDP equivalent of Transport Layer Security (TLS). When using DTLS with UDP, the overhead comes from

encrypting and decrypting the data, as the handshake process is simplified. DTLS also adds a small amount of overhead to the UDP packets, as it includes additional fields to indicate the security parameters and to detect tampering.

When to consider using SRD

Scalable reliable datagram (SRD) is a network transport protocol optimized for high-throughput workloads due to its ability to load-balancer traffic across multiple paths and quickly recover from packet drops or link failures. SRD is therefore best used for high performance computing (HPC) workloads that require high throughput and low latency communication between compute nodes. This might include parallel processing tasks such as simulation, modelling, and data analysis that involve a large amount of data transfer between nodes.

Implementation steps

1. Use the AWS Global Accelerator and AWS Transfer Family services to improve the throughput of your online file transfer applications. The AWS Global Accelerator service helps you achieve lower latency between your client devices and your workload on AWS. With AWS Transfer Family, you can use TCP-based protocols such as Secure Shell File Transfer Protocol (SFTP) and File Transfer Protocol over SSL (FTPS) to securely scale and manage your file transfers to AWS storage services.
2. Use network latency to determine if TCP is appropriate for communication between workload components. If the network latency between your client application and server is high, then the TCP three-way handshake can take some time, thereby impacting on the responsiveness of your application. Metrics such as time to first byte (TTFB) and round-trip time (RTT) can be used to measure network latency. If your workload serves dynamic content to users, consider using Amazon CloudFront, which establishes a persistent connection to each origin for dynamic content to remove the connection setup time that would otherwise slow down each client request.
3. Using TLS with TCP or UDP can result in increased latency and reduced throughput for your workload due to the impact of encryption and decryption. For such workloads, consider SSL/TLS offloading on Elastic Load Balancing to improve workload performance by allowing the load balancer to handle SSL/TLS encryption and decryption process instead of having backend instances do it. This can help reduce the CPU utilization on the backend instances, which can improve performance and increase capacity.
4. Use the Network Load Balancer (NLB) to deploy services that rely on the UDP protocol, such as authentication and authorization, logging, DNS, IoT, and streaming media, to improve the

performance and reliability of your workload. The NLB distributes incoming UDP traffic across multiple targets, allowing you to scale your workload horizontally, increase capacity, and reduce the overhead of a single target.

5. For your High Performance Computing (HPC) workloads, consider using the Elastic Network Adapter (ENA) Express functionality that uses the SRD protocol to improve network performance by providing a higher single flow bandwidth (25 Gbps) and lower tail latency (99.9 percentile) for network traffic between EC2 instances.
6. Use the Application Load Balancer (ALB) to route and load balance your gRPC (Remote Procedure Calls) traffic between workload components or between gRPC clients and services. gRPC uses the TCP-based HTTP/2 protocol for transport and it provides performance benefits such as lighter network footprint, compression, efficient binary serialization, support for numerous languages, and bi-directional streaming.

Resources

Related documents:

- Amazon EBS - Optimized Instances
- Application Load Balancer
- EC2 Enhanced Networking on Linux
- EC2 Enhanced Networking on Windows
- EC2 Placement Groups
- Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances
- Network Load Balancer
- Networking Products with AWS
- AWS Transit Gateway
- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs

Related videos:

- Connectivity to AWS and hybrid AWS network architectures

- Optimizing Network Performance for Amazon EC2 Instances

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF04-BP06 Choose your workload's location based on network requirements

Evaluate options for resource placement to reduce network latency and improve throughput, providing an optimal user experience by reducing page load and data transfer times.

Common anti-patterns:

- You consolidate all workload resources into one geographic location.
- You chose the closest Region to your location but not to the workload end user.

Benefits of establishing this best practice: User experience is greatly affected by the latency between the user and your application. By using appropriate AWS Regions and the AWS private global network, you can reduce latency and deliver a better experience to remote users.

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

Implementation guidance

Resources, such as Amazon EC2 instances, are placed into Availability Zones within AWS Regions, AWS Local Zones, AWS Outposts, or AWS Wavelength zones. Selection of this location influences network latency and throughput from a given user location. Edge services like Amazon CloudFront and AWS Global Accelerator can also be used to improve network performance by either caching content at edge locations or providing users with an optimal path to the workload through the AWS global network.

Amazon EC2 provides placement groups for networking. A placement group is a logical grouping of instances to decrease latency. Using placement groups with supported instance types and an Elastic Network Adapter (ENA) enables workloads to participate in a low-latency, reduced jitter 25

Gbps network. Placement groups are recommended for workloads that benefit from low network latency, high network throughput, or both.

Latency-sensitive services are delivered at edge locations using AWS global network, such as Amazon CloudFront. These edge locations commonly provide services like content delivery network (CDN) and domain name system (DNS). By having these services at the edge, workloads can respond with low latency to requests for content or DNS resolution. These services also provide geographic services, such as geotargeting of content (providing different content based on the end users' location) or latency-based routing to direct end users to the nearest Region (minimum latency).

Use edge services to reduce latency and to enable content caching. Configure cache control correctly for both DNS and HTTP/HTTPS to gain the most benefit from these approaches.

Implementation steps

- Capture information about the IP traffic going to and from network interfaces.
  - Logging IP traffic using VPC Flow Logs
  - How the client IP address is preserved in AWS Global Accelerator
- Analyze network access patterns in your workload to identify how users use your application.
  - Use monitoring tools, such as Amazon CloudWatch and AWS CloudTrail, to gather data on network activities.
  - Analyze the data to identify the network access pattern.
- Select Regions for your workload deployment based on the following key elements:
  - **Where your data is located:** For data-heavy applications (such as big data and machine learning), application code should run as close to the data as possible.
  - **Where your users are located**: For user-facing applications, choose a Region (or Regions) close to your workload's users.
  - **Other constraints**: Consider constraints such as cost and compliance as explained in What to Consider when Selecting a Region for your Workloads.
- Use AWS Local Zones to run workloads like video rendering. Local Zones allow you to benefit from having compute and storage resources closer to end users.
- Use AWS Outposts for workloads that need to remain on-premises and where you want that workload to run seamlessly with the rest of your other workloads in AWS.
- Applications like high-resolution live video streaming, high-fidelity audio, and augmented reality or virtual reality (AR/VR) require ultra-low-latency for 5G devices. For such applications,

consider AWS Wavelength. AWS Wavelength embeds AWS compute and storage services within 5G networks, providing mobile edge computing infrastructure for developing, deploying, and scaling ultra-low-latency applications.

- Use local caching or AWS Caching Solutions for frequently used assets to improve performance, reduce data movement, and lower environmental impact.

Service	When to use
Amazon CloudFront	Use to cache static content such as images, scripts, and videos, as well as dynamic content such as API responses or web applications.
Amazon ElastiCache	Use to cache content for web applications.
DynamoDB Accelerator	Use to add in-memory acceleration to your DynamoDB tables.

- Use services that can help you run code closer to users of your workload like the following:

Service	When to use
Lambda@edge	Use for compute-heavy operations that are initiated when objects are not in the cache.
Amazon CloudFront Functions	Use for simple use cases like HTTP(s) requests or response manipulations that can be initiated by short-lived functions.
AWS IoT Greengrass	Use to run local compute, messaging, and data caching for connected devices.

- Some applications require fixed entry points or higher performance by reducing first byte latency and jitter, and increasing throughput. These applications can benefit from networking services that provide static anycast IP addresses and TCP termination at edge locations. AWS Global Accelerator can improve performance for your applications by up to 60% and provide quick failover for multi-region architectures. AWS Global Accelerator provides you with static anycast IP addresses that serve as a fixed entry point for your applications hosted in one or more AWS

Regions. These IP addresses permit traffic to ingress onto the AWS global network as close to your users as possible. AWS Global Accelerator reduces the initial connection setup time by establishing a TCP connection between the client and the AWS edge location closest to the client. Review the use of AWS Global Accelerator to improve the performance of your TCP/UDP workloads and provide quick failover for multi-Region architectures.

Resources

Related best practices:

- COST07-BP02 Implement Regions based on cost
- COST08-BP03 Implement services to reduce data transfer costs
- REL10-BP01 Deploy the workload to multiple locations
- REL10-BP02 Select the appropriate locations for your multi-location deployment
- SUS01-BP01 Choose Region based on both business requirements and sustainability goals
- SUS02-BP04 Optimize geographic placement of workloads based on their networking requirements
- SUS04-BP07 Minimize data movement across networks

Related documents:

- AWS Global Infrastructure
- AWS Local Zones and AWS Outposts, choosing the right technology for your edge workload
- Placement groups
- AWS Local Zones
- AWS Outposts
- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- AWS Site-to-Site VPN
- Amazon Route 53

Related videos:

- AWS Local Zones Explainer Video
- AWS Outposts: Overview and How it Works
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises
- AWS re:Invent 2020: AWS Wavelength: Run apps with ultra-low latency at 5G edge
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2022 - Build your global wide area network using AWS
- AWS re:Invent 2020: Global traffic management with Amazon Route 53

Related examples:

- AWS Global Accelerator Workshop
- Handling Rewrites and Redirects using Edge Functions

PERF04-BP07 Optimize network configuration based on metrics

Use collected and analyzed data to make informed decisions about optimizing your network configuration.

Common anti-patterns:

- You assume that all performance-related issues are application-related.
- You only test your network performance from a location close to where you have deployed the workload.
- You use default configurations for all network services.
- You overprovision the network resource to provide sufficient capacity.

Benefits of establishing this best practice: Collecting necessary metrics of your AWS network and implementing network monitoring tools allows you to understand network performance and optimize network configurations.

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

Implementation guidance

Monitoring traffic to and from VPCs, subnets, or network interfaces is crucial to understand how to utilize AWS network resources and optimize network configurations. By using the following AWS networking tools, you can further inspect information about the traffic usage, network access and logs.

Implementation steps

- Identify the key performance metrics such as latency or packet loss to collect. AWS provides several tools that can help you to collect these metrics. By using the following tools, you can further inspect information about the traffic usage, network access, and logs:

AWS tool	Where to use
Amazon VPC IP Address Manager.	Use IPAM to plan, track, and monitor IP addresses for your AWS and on-premises workloads. This is a best practice to optimize IP address usage and allocation.
VPC Flow logs	Use VPC Flow Logs to capture detailed information about traffic to and from network interfaces in your VPCs. With VPC Flow Logs, you can diagnose overly restricti ve or permissive security group rules and determine the direction of the traffic to and from the network interfaces.
AWS Transit Gateway Flow Logs	Use AWS Transit Gateway Flow Logs to capture information about the IP traffic going to and from your transit gateways.
DNS query logging	Log information about public or private DNS queries Route 53 receives. With DNS logs, you can optimize DNS configurations by understanding the domain or subdomain that

AWS tool	Where to use
	was requested or Route 53 EDGE locations that responded to DNS queries.
Reachability Analyzer	Reachability Analyzer helps you analyze and debug network reachability. Reachability Analyzer is a configuration analysis tool that allows you to perform connectivity testing between a source resource and a destination resource in your VPCs. This tool helps you verify that your network configuration matches your intended connectivity.
Network Access Analyzer	Network Access Analyzer helps you understand network access to your resources. You can use Network Access Analyzer to specify your network access requirements and identify potential network paths that do not meet your specified requirements. By optimizing your corresponding network configuration, you can understand and verify the state of your network and demonstrate if your network on AWS meets your compliance requirements.

AWS tool	Where to use
Amazon CloudWatch	Use Amazon CloudWatch and turn on the appropriate metrics for network options. Make sure to choose the right network metric for your workload. For example, you can turn on metrics for VPC Network Address Usage, VPC NAT Gateway, AWS Transit Gateway, VPN tunnel, AWS Network Firewall, Elastic Load Balancing, and AWS Direct Connect. Continually monitoring metrics is a good practice to observe and understand your network status and usage, which helps you optimize network configuration based on your observations.
AWS Network Manager	Using AWS Network Manager, you can monitor the real-time and historical performance of the AWS Global Network for operational and planning purposes. Network Manager provides aggregate network latency between AWS Regions and Availability Zones and within each Availability Zone, allowing you to better understand how your applicati on performance relates to the performance of the underlying AWS network.
Amazon CloudWatch RUM	Use Amazon CloudWatch RUM to collect the metrics that give you the insights that help you identify, understand, and improve user experience.

- Identify top talkers and application traffic patterns using VPC and AWS Transit Gateway Flow Logs.
- Assess and optimize your current network architecture including VPCs, subnets, and routing. As an example, you can evaluate how different VPC peering or AWS Transit Gateway can help you improve the networking in your architecture.

- Assess the routing paths in your network to verify that the shortest path between destinations is always used. Network Access Analyzer can help you do this.

Resources

Related documents:

- VPC Flow Logs
- Public DNS query logging
- What is IPAM?
- What is Reachability Analyzer?
- What is Network Access Analyzer?
- CloudWatch metrics for your VPCs
- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format
- Monitoring your global and core networks with Amazon CloudWatch metrics
- Continuously monitor network traffic and resources

Related videos:

- Networking best practices and tips with the AWS Well-Architected Framework
- Monitoring and troubleshooting network traffic

Related examples:

- AWS Networking Workshops
- AWS Network Monitoring

Process and culture

When architecting workloads, there are principles and practices that you can adopt to help you better run efficient high-performing cloud workloads. This focus area offers best practices to help adopt a culture that fosters performance efficiency of cloud workloads.

Consider these key principles to build this culture:

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

- **Performance visibility:** Key metrics should be visible to your team, especially metrics against each build version. This allows you to see any significant positive or negative trend over time. You should also display metrics on the number of errors or exceptions to make sure you are testing a working system.
- **Visualization:** Use visualization techniques that make it clear where performance issues, hot spots, wait states, or low utilization is occurring. Overlay performance metrics over architecture diagrams — call graphs or code can help identify issues quickly.
- **Regular review process:** Architectures performing poorly is usually the result of a non-existent or broken performance review process. If your architecture is performing poorly, implementing a performance review process allows you to drive iterative improvement.
- **Continual optimization:** Adopt a culture to continually optimize the performance efficiency of your cloud workload.

Best practices

- PERF05-BP01 Establish key performance indicators (KPIs) to measure workload health and performance
- PERF05-BP02 Use monitoring solutions to understand the areas where performance is most critical
- PERF05-BP03 Define a process to improve workload performance
- PERF05-BP04 Load test your workload
- PERF05-BP05 Use automation to proactively remediate performance-related issues
- PERF05-BP06 Keep your workload and services up-to-date
- PERF05-BP07 Review metrics at regular intervals

PERF05-BP01 Establish key performance indicators (KPIs) to measure workload health and performance

Identify the KPIs that quantitatively and qualitatively measure workload performance. KPIs help you measure the health and performance of a workload related to a business goal.

Common anti-patterns:

- You only monitor system-level metrics to gain insight into your workload and don't understand business impacts to those metrics.

- You assume that your KPIs are already being published and shared as standard metric data.
- You do not define a quantitative, measurable KPI.
- You do not align KPIs with business goals or strategies.

Benefits of establishing this best practice: Identifying specific KPIs that represent workload health and performance helps align teams on their priorities and define successful business outcomes. Sharing those metrics with all departments provides visibility and alignment on thresholds, expectations, and business impact.

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

Implementation guidance

KPIs allow business and engineering teams to align on the measurement of goals and strategies and how these factors combine to produce business outcomes. For example, a website workload might use page load time as an indication of overall performance. This metric would be one of multiple data points that measures user experience. In addition to identifying the page load time thresholds, you should document the expected outcome or business risk if ideal performance is not met. A long page load time affects your end users directly, decreases their user experience rating, and can lead to a loss of customers. When you define your KPI thresholds, combine both industry benchmarks and your end user expectations. For example, if the current industry benchmark is a webpage loading within a two-second time period, but your end users expect a webpage to load within a one-second time period, then you should take both of these data points into consideration when establishing the KPI.

Your team must evaluate your workload KPIs using real-time granular data and historical data for reference and create dashboards that perform metric math on your KPI data to derive operational and utilization insights. KPIs should be documented and include thresholds that support business goals and strategies, and should be mapped to metrics being monitored. KPIs should be revisited when business goals, strategies, or end user requirements change.

Implementation steps

1. Identify and document key business stakeholders.
2. Work with these stakeholders to define and document objectives of your workload.
3. Review industry best practices to identify relevant KPIs aligned with your workload objectives.

4. Use industry best practices and your workload objectives to set targets for your workload KPI. Use this information to set KPI thresholds for severity or alarm level.
5. Identify and document the risk and impact if the KPI is not met.
6. Identify and document metrics that can help you to establish the KPIs.
7. Use monitoring tools such as Amazon CloudWatch or AWS Config to collect metrics and measure KPIs.
8. Use dashboards to visualize and communicate KPIs with stakeholders.
9. Regularly review and analyze metrics to identify areas of workload that need to be improved.
10. Revisit KPIs when business goals or workload performance changes.

Resources

Related documents:

- CloudWatch documentation
- Monitoring, Logging, and Performance AWS Partners
- X-Ray Documentation
- Using Amazon CloudWatch dashboards
- Amazon QuickSight KPIs

Related videos:

- AWS re:Invent 2019: Scaling up to your first 10 million users
- Cut through the chaos: Gain operational visibility and insight
- Build a Monitoring Plan

Related examples:

- Creating a dashboard with Amazon QuickSight

PERF05-BP02 Use monitoring solutions to understand the areas where performance is most critical

Understand and identify areas where increasing the performance of your workload will have a positive impact on efficiency or customer experience. For example, a website that has a large amount of customer interaction can benefit from using edge services to move content delivery closer to customers.

Common anti-patterns:

- You assume that standard compute metrics such as CPU utilization or memory pressure are enough to catch performance issues.
- You only use the default metrics recorded by your selected monitoring software.
- You only review metrics when there is an issue.

Benefits of establishing this best practice: Understanding critical areas of performance helps workload owners monitor KPIs and prioritize high-impact improvements.

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

Implementation guidance

Set up end-to-end tracing to identify traffic patterns, latency, and critical performance areas. Monitor your data access patterns for slow queries or poorly fragmented and partitioned data. Identify the constrained areas of the workload using load testing or monitoring.

Increase performance efficiency by understanding your architecture, traffic patterns, and data access patterns, and identify your latency and processing times. Identify the potential bottlenecks that might affect the customer experience as the workload grows. After investigating these areas, look at which solution you could deploy to remove those performance concerns.

Implementation steps

1. Set up end-to-end monitoring to capture all workload components and metrics. Here are examples of monitoring solutions on AWS.

Service	Where to use
Amazon CloudWatch Real-User Monitoring (RUM)	To capture application performance metrics from real user client-side and frontend sessions.
AWS X-Ray	To trace traffic through the application layers and identify latency between components and dependencies. Use X-Ray service maps to see relationships and latency between workload components.
Amazon Relational Database Service Performance Insights	To view database performance metrics and identify performance improvements.
Amazon RDS Enhanced Monitoring	To view database OS performance metrics.
Amazon DevOps Guru	To detect abnormal operating patterns so you can identify operational issues before they impact your customers.

2. Perform tests to generate metrics, identify traffic patterns, bottlenecks, and critical performance areas. Here are some examples of how to perform testing:
   - Set up CloudWatch Synthetic Canaries to mimic browser-based user activities programmatically using Linux cron jobs or rate expressions to generate consistent metrics over time.
   - Use the AWS Distributed Load Testing solution to generate peak traffic or test the workload at the expected growth rate.
3. Evaluate the metrics and telemetry to identify your critical performance areas. Review these areas with your team to discuss monitoring and solutions to avoid bottlenecks.
4. Experiment with performance improvements and measure those changes with data. As an example, you can use CloudWatch Evidently to test new improvements and performance impacts to your workload.

Resources

Related documents:

- Amazon Builders' Library
- X-Ray Documentation
- Amazon CloudWatch RUM
- Amazon DevOps Guru

Related videos:

- The Amazon Builders' Library: 25 years of Amazon operational excellence
- Visual Monitoring of Applications with Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- X-Ray SDK for Node.js
- X-Ray SDK for Python
- X-Ray SDK for Java
- X-Ray SDK for .Net
- X-Ray SDK for Ruby
- X-Ray Daemon
- Distributed Load Testing on AWS

PERF05-BP03 Define a process to improve workload performance

Define a process to evaluate new services, design patterns, resource types, and configurations as they become available. For example, run existing performance tests on new instance offerings to determine their potential to improve your workload.

Common anti-patterns:

- You assume your current architecture is static and won't be updated over time.
- You introduce architecture changes over time with no metric justification.

Benefits of establishing this best practice: By defining your process for making architectural changes, you can use gathered data to influence your workload design over time.

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

Implementation guidance

Your workload's performance has a few key constraints. Document these so that you know what kinds of innovation might improve the performance of your workload. Use this information when learning about new services or technology as it becomes available to identify ways to alleviate constraints or bottlenecks.

Identify the key performance constraints for your workload. Document your workload's performance constraints so that you know what kinds of innovation might improve the performance of your workload.

Implementation steps

- Identify your workload performance KPIs as outlined in PERF05-BP01 Establish key performance indicators (KPIs) to measure workload health and performance to baseline your workload.
- Use AWS observability tools to collect performance metrics and measure KPIs.
- Conduct in-depth analysis to identify the areas (like configuration and application code) in your workload that is under-performing as outlined in PERF05-BP02 Use monitoring solutions to understand the areas where performance is most critical.
- Use your analysis and performance tools to identify the performance optimization strategy.
- Use sandbox or pre-production environments to validate effectiveness of the strategy.
- Implement the changes in production and continually monitor the workload's performance.
- Document the improvements and communicate that to stakeholders.

Resources

Related documents:

- AWS Blog
- What's New with AWS

Related videos:

- AWS Events YouTube Channel
- AWS Online Tech Talks YouTube Channel
- Amazon Web Services YouTube Channel

Related examples:

- AWS Github
- AWS Skill Builder

PERF05-BP04 Load test your workload

Load test your workload to verify it can handle production load and identify any performance bottleneck.

Common anti-patterns:

- You load test individual parts of your workload but not your entire workload.
- You load test on infrastructure that is not the same as your production environment.
- You only conduct load testing to your expected load and not beyond, to help foresee where you may have future problems.
- You perform load testing without consulting the Amazon EC2 Testing Policy and submitting a Simulated Event Submissions Form. This results in your test failing to run, as it looks like a denial-of-service event.

Benefits of establishing this best practice: Measuring your performance under a load test will show you where you will be impacted as load increases. This can provide you with the capability of anticipating needed changes before they impact your workload.

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

Implementation guidance

Load testing in the cloud is a process to measure the performance of cloud workload under realistic conditions with expected user load. This process involves provisioning a production-like cloud environment, using load testing tools to generate load, and analyzing metrics to assess the ability of your workload handling a realistic load. Load tests must be run using synthetic or sanitized versions of production data (remove sensitive or identifying information). Automatically carry out load tests as part of your delivery pipeline, and compare the results against pre-defined KPIs and thresholds. This process helps you continue to achieve required performance.

Implementation steps

- Set up the test environment based on your production environment. You can use AWS services to run production-scale environments to test your architecture.
- Choose and configure the load testing tool that suits your workload.
- Define the load testing scenarios and parameters (like test duration and number of users).
- Perform test scenarios at scale. Take advantage of the AWS Cloud to test your workload to discover where it fails to scale, or if it scales in a non-linear way. For example, use Spot Instances to generate loads at low cost and discover bottlenecks before they are experienced in production.
- Monitor and record performance metrics (like throughput and response time). Amazon CloudWatch can collect metrics across the resources in your architecture. You can also collect and publish custom metrics to surface business or derived metrics.
- Analyze the results to identify performance bottlenecks and areas for improvements.
- Document and report on load testing process and results.

Resources

Related documents:

- AWS CloudFormation
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Distributed Load Testing on AWS

Related videos:

- Solving with AWS Solutions: Distributed Load Testing

- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Distributed Load Testing on AWS

PERF05-BP05 Use automation to proactively remediate performance-related issues

Use key performance indicators (KPIs), combined with monitoring and alerting systems, to proactively address performance-related issues.

Common anti-patterns:

- You only allow operations staff the ability to make operational changes to the workload.
- You let all alarms filter to the operations team with no proactive remediation.

Benefits of establishing this best practice: Proactive remediation of alarm actions allows support staff to concentrate on those items that are not automatically actionable. This helps operations staff handle all alarms without being overwhelmed and instead focus only on critical alarms.

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

Implementation guidance

Use alarms to trigger automated actions to remediate issues where possible. Escalate the alarm to those able to respond if automated response is not possible. For example, you may have a system that can predict expected key performance indicator (KPI) values and alarm when they breach certain thresholds, or a tool that can automatically halt or roll back deployments if KPIs are outside of expected values.

Implement processes that provide visibility into performance as your workload is running. Build monitoring dashboards and establish baseline norms for performance expectations to determine if the workload is performing optimally.

Implementation steps

- Identify and understand the performance issue that can be remediated automatically. Use AWS monitoring solutions such as Amazon CloudWatch or AWS X-Ray to help you better understand the root cause of the issue.
- Create a step-by-step remediation plan and process that can be used to automatically fix the issue.
- Configure the trigger to automatically initiate the remediation process. For example, you can define a trigger to automatically restart an instance when it reaches a certain threshold of CPU utilization.
- Use AWS services and technologies to automate the remediation process. For example, AWS Systems Manager Automation provides a secure and scalable way to automate the remediation process.
- Test the automated remediation process in a pre-production environment.
- After testing, implement the remediation process in production environment and continually monitor to identify areas for improvement.

Resources

Related documents:

- CloudWatch Documentation
- Monitoring, Logging, and Performance AWS Partner Network Partners
- X-Ray Documentation
- Using Alarms and Alarm Actions in CloudWatch

Related videos:

- Intelligently automating cloud operations
- Setting up controls at scale in your AWS environment
- Automating patch management and compliance using AWS

- How Amazon uses better metrics for improved website performance

Related examples:

- CloudWatch Logs Customize Alarms

PERF05-BP06 Keep your workload and services up-to-date

Stay up-to-date on new cloud services and features to adopt efficient features, remove issues, and improve the overall performance efficiency of your workload.

Common anti-patterns:

- You assume your current architecture is static and will not be updated over time.
- You do not have any systems or a regular cadence to evaluate if updated software and packages are compatible with your workload.

Benefits of establishing this best practice: By establishing a process to stay up-to-date on new services and offerings, you can adopt new features and capabilities, resolve issues, and improve workload performance.

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

Implementation guidance

Evaluate ways to improve performance as new services, design patterns, and product features become available. Determine which of these could improve performance or increase the efficiency of the workload through evaluation, internal discussion, or external analysis. Define a process to evaluate updates, new features, and services relevant to your workload. For example, build a proof of concept that uses new technologies or consult with an internal group. When trying new ideas or services, run performance tests to measure the impact that they have on the performance of the workload.

Implementation steps

- Inventory your workload software and architecture and identify components that need to be updated.

- Identify news and update sources related to your workload components. As an example, you can subscribe to the What's New at AWS blog for the products that match your workload component. You can subscribe to the RSS feed or manage your email subscriptions.
- Define a schedule to evaluate new services and features for your workload.
  - You can use AWS Systems Manager Inventory to collect operating system (OS), application, and instance metadata from your Amazon EC2 instances and quickly understand which instances are running the software and configurations required by your software policy and which instances need to be updated.
- Understand how to update the components of your workload. Take advantage of agility in the cloud to quickly test how new features can improve your workload to gain performance efficiencies.
- Use automation for the update process to reduce the level of effort to deploy new features and limit errors caused by manual processes.
  - You can use CI/CD to automatically update AMIs, container images, and other artifacts related to your cloud application.
  - You can use tools such as AWS Systems Manager Patch Manager to automate the process of system updates, and schedule the activity using AWS Systems Manager Maintenance Windows.
- Document your process for evaluating updates and new services. Provide your owners the time and space needed to research, test, experiment, and validate updates and new services. Refer back to the documented business requirements and KPIs to help prioritize which update will make a positive business impact.

Resources

Related documents:

- AWS Blog
- What's New with AWS

Related videos:

- AWS Events YouTube Channel
- AWS Online Tech Talks YouTube Channel
- Amazon Web Services YouTube Channel

Related examples:

- Well-Architected Labs - Inventory and Patch Management
- Lab: AWS Systems Manager

PERF05-BP07 Review metrics at regular intervals

As part of routine maintenance or in response to events or incidents, review which metrics are collected. Use these reviews to identify which metrics were essential in addressing issues and which additional metrics, if they were being tracked, could help identify, address, or prevent issues.

Common anti-patterns:

- You allow metrics to stay in an alarm state for an extended period of time.
- You create alarms that are not actionable by an automation system.

Benefits of establishing this best practice: Continually review metrics that are being collected to verify that they properly identify, address, or prevent issues. Metrics can also become stale if you let them stay in an alarm state for an extended period of time.

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

Implementation guidance

Constantly improve metric collection and monitoring. As part of responding to incidents or events, evaluate which metrics were helpful in addressing the issue and which metrics could have helped that are not currently being tracked. Use this method to improve the quality of metrics you collect so that you can prevent, or more quickly resolve future incidents.

As part of responding to incidents or events, evaluate which metrics were helpful in addressing the issue and which metrics could have helped that are not currently being tracked. Use this to improve the quality of metrics you collect so that you can prevent or more quickly resolve future incidents.

Implementation steps

1. Define critical performance metrics to monitor that are aligned to your workload objective.
2. Set a baseline and desirable value for each metric.
3. Set a cadence (like weekly or monthly) to review critical metrics.

4. During each review, assess trends and deviation from the baseline values. Look for any performance bottlenecks or anomalies.
5. For identified issues, conduct in-depth root cause analysis to understand the main reason behind the issue.
6. Document your findings and use strategies to deal with identified issues and bottlenecks.
7. Continually assess and improve the metrics review process.

Resources

Related documents:

- CloudWatch Documentation
- Collect metrics and logs from Amazon EC2 Instances and on-premises servers with the CloudWatch Agent
- Monitoring, Logging, and Performance AWS Partner Network Partners
- X-Ray Documentation

Related videos:

- Setting up controls at scale in your AWS environment
- How Amazon uses better metrics for improved website performance

Related examples:

- Creating a dashboard with Amazon QuickSight
- Level 100: Monitoring with CloudWatch Dashboards

Conclusion

Achieving and maintaining performance efficiency requires a data-driven approach. You should actively consider access patterns and trade-offs that will allow you to optimize for higher performance. Using a review process based on benchmarks and load tests allows you to select the appropriate resource types and configurations. Treating your infrastructure as code helps you to rapidly and safely evolve your architecture, while you use data to make fact-based decisions about your architecture. Putting in place a combination of active and passive monitoring ensures that the performance of your architecture does not degrade over time.

AWS strives to help you build architectures that perform efficiently while delivering business value. Use the tools and techniques discussed in this paper to ensure success.

Contributors

The following individuals and organizations contributed to this document:

- Sam Mokhtari, Senior Efficiency Lead Solutions Architect, Amazon Web Services
- Josh Hart, Solutions Architect, Amazon Web Services
- Richard Trabing, Solutions Architect, Amazon Web Services
- Brett Looney, Principal Solutions Architect, Amazon Web Services
- Nina Vogl, Principal Solutions Architect, Amazon Web Services
- Eric Pullen, Solutions Architect, Amazon Web Services
- Julien Lépine, Specialist SA Manager, Amazon Web Services
- Ronnen Slasky, Solutions Architect, Amazon Web Services

Further reading

For additional help, consult the following sources:

- AWS Well-Architected Framework
- AWS Architecture Center

Document revisions

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

Change	Description	Date
Major update and restructure	Pillar restructured to have five best practice areas (down from eight). Content has been consolidated into the five areas and updated. New best practice areas are Architecture selection, Compute and hardware, Data management, Networking and content delivery, and Process and culture.	October 3, 2023
Minor update	Remove non-inclusive language.	April 13, 2023
Updates for new Framework	Best practices updated with prescriptive guidance and new best practices added.	April 10, 2023
Whitepaper updated	Best practices updated with new implementation guidance.	December 15, 2022
Whitepaper updated	Best practices expanded and improvement plans added.	October 20, 2022
Minor update	Removed non-inclusive language.	April 22, 2022
Minor update	Added Sustainability Pillar to introduction.	December 2, 2021

Minor updates	Updated links.	March 10, 2021
Minor updates	Changed AWS Lambda timeout to 900 seconds and corrected name of Amazon Keyspaces (for Apache Cassandra).	October 5, 2020
Minor update	Fixed broken link.	July 15, 2020
Updates for new Framework	Major review and update of content	July 8, 2020
Whitepaper updated	Minor update for grammatical issues	July 1, 2018
Whitepaper updated	Refreshed the whitepaper to reflect changes in AWS	November 1, 2017
Initial publication	Performance Efficiency Pillar - AWS Well-Architected Framework published.	November 1, 2016

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*.