Sustainability Pillar

AWS Well-Architected Framework

Sustainability Pillar: AWS Well-Architected Framework

Table of Contents

Sustainability Pillar - AWS Well-Architected Framework

Publication date: **July 13th, 2023** (*Document history* (p. 58))

This whitepaper focuses on the sustainability pillar of the Amazon Web Services (AWS) Well-Architected Framework. It provides design principles, operational guidance, best-practices, potential trade-offs, and improvement plans you can use to meet sustainability targets for your AWS workloads.

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 secure, reliable, efficient, cost-effective, and sustainable workloads in the AWS Cloud. The Framework provides a way for you to consistently measure your architectures against best practices and identify areas for improvement. Having well-architected workloads greatly increases your ability to support your business outcomes.

The Framework is based on six pillars:

- Operational excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization
- Sustainability

This document focuses on the sustainability pillar, and within the scope of sustainability, it focuses on environmental sustainability. It's intended for those in technology roles, such as chief technology officers (CTOs), architects, developers, and operations team members.

After reading this document, you will understand current AWS recommendations and strategies to use when designing cloud architectures with sustainability in mind. By adopting the practices in this paper, you can build architectures that maximize efficiency and reduce waste.

Cloud sustainability

The discipline of sustainability addresses the long-term environmental, economic, and societal impact of your business activities. The United Nations World Commission on Environment and Development defines sustainable development as "development that meets the needs of the present without compromising the ability of future generations to meet their own needs." Your business or organization can have negative environmental impacts like direct or indirect carbon emissions, unrecyclable waste, and damage to shared resources like clean water.

When building cloud workloads, the practice of sustainability is understanding the impacts of the services used, quantifying impacts through the entire workload lifecycle, and applying design principles and best practices to reduce these impacts. This document focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct action to reduce resource usage.

When focusing on environmental impacts, you should understand how these impacts are typically accounted for and the follow-on impacts to your organization's own emissions accounting. The Greenhouse Gas Protocol organizes carbon emissions into the following scopes, along with relevant emission examples within each scope for a cloud provider such as AWS:

- **Scope 1:** All direct emissions from the activities of an organization or under its control. For example, fuel combustion by data center backup generators.
- **Scope 2:** Indirect emissions from electricity purchased and used to power data centers and other facilities. For example, emissions from commercial power generation.
- **Scope 3:** All other indirect emissions from activities of an organization from sources it doesn't control. AWS examples include emissions related to data center construction, and the manufacture and transportation of IT hardware deployed in data centers.

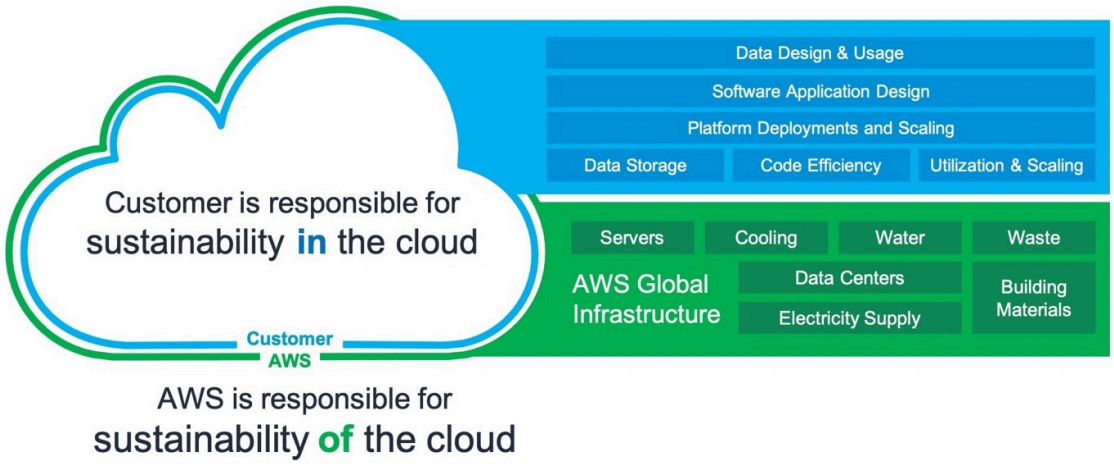
From an AWS customer perspective, emissions from your workloads running on AWS are accounted for as indirect emissions, and part of your Scope 3 emissions. Each workload deployed generates a fraction of the total AWS emissions from each of the previous scopes. The actual amount varies per workload and depends on several factors including the AWS services used, the energy consumed by those services, the carbon intensity of the electric grids serving the AWS data centers where they run, and the AWS procurement of renewable energy.

This document first describes a shared responsibility model for environmental sustainability, and then provides architectural best practices so you can minimize the impact of your workloads by reducing the total resources required for them to run in AWS data centers.

The shared responsibility model

Environmental sustainability is a shared responsibility between customers and AWS.

- AWS is responsible for optimizing the sustainability *of* the cloud – delivering efficient, shared infrastructure, water stewardship, and sourcing renewable power.
- Customers are responsible for sustainability *in* the cloud – optimizing workloads and resource utilization, and minimizing the total resources required to be deployed for your workloads.



Shared responsibility model

Sustainability of the cloud

Cloud providers have a lower carbon footprint and are more energy efficient than typical on-premises alternatives because they invest in efficient power and cooling technology, operate energy efficient server populations, and achieve high server utilization rates. Cloud workloads reduce impact by taking advantage of shared resources, such as networking, power, cooling, and physical facilities. You can migrate your cloud workloads to more efficient technologies as they become available and use cloud-based services to transform your workloads for better sustainability.

Resources

- The Carbon Reduction Opportunity of Moving to Amazon Web Services
- AWS enables sustainability solutions

Sustainability in the cloud

Sustainability in the cloud is a continuous effort focused primarily on energy reduction and efficiency across all components of a workload by achieving the maximum benefit from the resources provisioned and minimizing the total resources required. This effort can range from the initial selection of an efficient programming language, adoption of modern algorithms, use of efficient data storage techniques, deploying to correctly sized and efficient compute infrastructure, and minimizing requirements for high-powered end-user hardware.

Sustainability through the cloud

In addition to minimizing the impact of workloads that you've deployed, you can use the AWS Cloud to run workloads designed to support your wider sustainability challenges. Examples of these challenges include reducing carbon emissions, lowering energy consumption, recycling water, or reducing waste in other areas of your business or organization.

Sustainability *through* the cloud is when you use AWS technology to solve a broader sustainability challenge. For example, you can use a machine learning service like Amazon Monitron to detect abnormal behavior in industrial machinery. Using this detection data, you can conduct preventative

maintenance to reduce the risk of environmental incidents caused by unexpected equipment failures and ensure that the machinery continues to operate at peak efficiency.

Design principles for sustainability in the cloud

Apply these design principles when architecting your cloud workloads to maximize sustainability and minimize impact.

- **Understand your impact:** Measure the impact of your cloud workload and model the future impact of your workload. Include all sources of impact, including impacts resulting from customer use of your products, and impacts resulting from their eventual decommissioning and retirement. Compare the productive output with the total impact of your cloud workloads by reviewing the resources and emissions required per unit of work. Use this data to establish key performance indicators (KPIs), evaluate ways to improve productivity while reducing impact, and estimate the impact of proposed changes over time.
- **Establish sustainability goals:** For each cloud workload, establish long-term sustainability goals such as reducing the compute and storage resources required per transaction. Model the return on investment of sustainability improvements for existing workloads, and give owners the resources they need to invest in sustainability goals. Plan for growth, and architect your workloads so that growth results in reduced impact intensity measured against an appropriate unit, such as per user or per transaction. Goals help you support the wider sustainability goals of your business or organization, identify regressions, and prioritize areas of potential improvement.
- **Maximize utilization:** Right-size workloads and implement efficient design to ensure high utilization and maximize the energy efficiency of the underlying hardware. Two hosts running at 30% utilization are less efficient than one host running at 60% due to baseline power consumption per host. At the same time, eliminate or minimize idle resources, processing, and storage to reduce the total energy required to power your workload.
- **Anticipate and adopt new, more efficient hardware and software offerings:** Support the upstream improvements your partners and suppliers make to help you reduce the impact of your cloud workloads. Continually monitor and evaluate new, more efficient hardware and software offerings. Design for flexibility to allow for the rapid adoption of new efficient technologies.
- **Use managed services:** Sharing services across a broad customer base helps maximize resource utilization, which reduces the amount of infrastructure needed to support cloud workloads. For example, customers can share the impact of common data center components like power and networking by migrating workloads to the AWS Cloud and adopting managed services, such as AWS Fargate for serverless containers, where AWS operates at scale and is responsible for their efficient operation. Use managed services that can help minimize your impact, such as automatically moving infrequently accessed data to cold storage with Amazon S3 Lifecycle configurations or Amazon EC2 Auto Scaling to adjust capacity to meet demand.
- **Reduce the downstream impact of your cloud workloads:** Reduce the amount of energy or resources required to use your services. Reduce or eliminate the need for customers to upgrade their devices to use your services. Test using device farms to understand expected impact and test with customers to understand the actual impact from using your services.

Improvement process

The architectural improvement process includes understanding what you have and what you can do to improve, selecting targets for improvement, testing improvements, adopting successful improvements, quantifying your success and sharing what you have learned so that it can be replicated elsewhere, and then repeating the cycle.

The goals of your improvements can be:

- To eliminate waste, low utilization, and idle or unused resources
- To maximize the value from resources you consume

> **Note**
> Use all the resources you provision, and complete the same work with the minimum resources possible.

In early stages of optimization, first eliminate areas with waste or low utilization, and then move toward more targeted optimizations that fit your specific workload.

Monitor changes to the consumption of resources over time. Identify where accumulated changes result in inefficient or significant increases in resource consumption. Determine the need for improvements to address the changes in consumption and implement the improvements you prioritize.

The following steps are designed to be an iterative process that evaluates, prioritizes, tests, and deploys sustainability-focused improvements for cloud workloads.

1. **Identify targets for improvement:** Review your workloads against best practices for sustainability that are identified in this document, and identify targets for improvement.
2. **Evaluate specific improvements:** Evaluate specific changes for potential improvement, projected cost, and business risk.
3. **Prioritize and plan improvements:** Prioritize changes that offer the largest improvements at the least cost and risk, and establish a plan for testing and implementation.
4. **Test and validate improvements:** Implement changes in testing environments to validate their potential for improvement.
5. **Deploy changes to production:** Implement changes across production environments.
6. **Measure results and replicate successes:** Look for opportunities to replicate successes across workloads, and revert changes with unacceptable outcomes.

Example scenario

The following example scenario is referenced later in this document to illustrate each step of the improvement process.

Your company has a workload that performs complex image manipulations on Amazon EC2 instances and stores the modified and original files for user access. The processing activities are CPU intensive, and the output files are extremely large.

Identify targets for improvement

Understand the best practices that can help you achieve your sustainability goals. You can find detailed descriptions of these best practices (p. 13) and recommendations for improvement later in this document.

Review your workloads and the resources used. Identify *hot spots* such as large deployments and frequently used resources. Evaluate these hot spots for opportunities to improve the effective utilization of your resources and to reduce the total resources required to achieve your business outcomes.

Review your workload against best practices, and identify candidates for improvement.

Applying this step to the Example scenario (p. 5), you identify the following best practices as likely targets for improvement:

- Use the minimum amount of hardware to meet your needs
- Use technologies that best support your data access and storage patterns

Resources

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute
- Optimizing your AWS Infrastructure for Sustainability, Part II: Storage
- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking

Evaluate specific improvements

Understand the resources provisioned by your workload to complete a unit of work. Evaluate potential improvements, and estimate their potential impact, the cost to implement, and the associated risks.

To measure improvements over time, first understand what you have provisioned in AWS and how those resources are being consumed.

Start with a full overview of your AWS usage, and use AWS Cost and Usage Reports to help identify hot spots. Use this AWS sample code to help you review and analyze your report with the help of Amazon Athena.

Proxy metrics

When you evaluate specific changes, you must also evaluate which metrics best quantify the effect of that change on the associated resource. These metrics are called *proxy metrics*. Select proxy metrics that best reflect the type of improvement you are evaluating and the resources targeted by improvement. These metrics might evolve over time.

The resources provisioned to support your workload include compute, storage, and network resources. Evaluate the resources provisioned using your proxy metrics to see how those resources are consumed.

Use your proxy metrics to measure the resources provisioned to achieve business outcomes.

Resource	Example proxy metrics	Improvement goals
Compute	vCPU minutes	Maximize utilization of provisioned resources

Resource	Example proxy metrics	Improvement goals
Storage	GB provisioned	Reduce total provisioned
Network	GB transferred or packets transferred	Reduce total transferred and transferred distance

Business metrics

Select business metrics to quantify the achievement of business outcomes. Your business metrics should reflect the value provided by your workload, for example, the number of simultaneous active users, API calls served, or the number of transactions completed. These metrics may evolve over time. Be cautious when evaluating financial-based business metrics, since inconsistency in the value of transactions invalidates comparisons.

Key performance indicators

Using the following formula, divide the provisioned resources by the business outcomes achieved to determine the provisioned resources per unit of work.

$$Resources\ provisioned\ per\ unit\ of\ work = \frac{Proxy\ metric\ for\ provisioned\ resource}{Business\ metric\ for\ outcome}$$

KPI formula

Use your resources per unit of work as your KPIs. Establish baselines based on provisioned resources as the basis for comparisons.

Resource	Example KPIs	Improvement goals
Compute	vCPU minutes per transaction	Maximize utilization of provisioned resources
Storage	GB per transaction	Reduce total provisioned
Network	GB transferred per transaction or packets transferred per transaction	Reduce total transferred and transferred distance

Estimate improvement

Estimate improvement as both the quantitative reduction in resources provisioned (as indicated by your proxy metrics) and the percentage change from your baseline resources provisioned per unit of work.

Resource	Example KPIs	Improvement goals
Compute	% reduction of vCPUs minutes per transaction	Maximize utilization
Storage	% reduction GB per transaction	Reduce total provisioned

Resource	Example KPIs	Improvement goals
Network	% reduction of GB transferred per transaction or packets transferred per transaction	Reduce total transferred and transferred distance

Evaluate improvements

Evaluate potential improvements against the anticipated net benefit. Evaluate the time, cost, and level of effort to implement and maintain, and business risks such as unanticipated impacts.

Targeted improvements often represent trade-offs between the types of resources consumed. For example, to reduce compute consumption, you can store a result, or to limit data transferred, you can process data before sending the result to a client. These trade-offs (p. 12) are discussed in additional detail later.

Include non-functional requirements when evaluating the risks for your workload, including security, reliability, performance efficiency, cost optimization, and the impact of improvements on your ability to operate your workload.

Applying this step to the Example scenario (p. 5), you evaluate the target improvements with the following results:

Best practice	Targeted improvement	Potential	Cost	Risk
Use the minimum amount of hardware to meet your needs	Implement predictive scaling to reduce low utilization periods	Medium	Low	Low
Use technologies that best support your data access and storage patterns	Implement more effective compression mechanisms to reduce total storage and the time to achieve it	High	Low	Low

Implementing predictive scaling reduces the vCPU hours consumed by under-utilized or unused instances providing moderate benefits over existing scaling mechanisms with an estimated 11% reduction in resources consumed. The costs involved are low and include the configuration of the cloud resources and the operation of predictive scaling for Amazon EC2 Auto Scaling. The risk is constrained performance when scale-out is performed reactively in response to demand exceeding predictions.

Implementing more effective compression can have a significant impact with large reductions in file size across all of your original and manipulated images, with an estimated 25% reduction in storage requirements in production. Implementing the new algorithm is a low-effort substitution with little risk involved.

Prioritize and plan improvements

Prioritize your identified improvements based on the greatest anticipated impact with the lowest costs and acceptable risk.

Decide which improvements to focus on initially, and include them in your resource planning and development roadmap.

Applying this step to the Example scenario (p. 5), you prioritize the target improvements as follows:

Priority	Improvement	Potential	Cost	Risk
1	Implement more effective compression mechanisms	High	Low	Low
2	Implement predictive scaling	Medium	Low	Low

The high potential, low cost, and risk of updating file compression make it a high-value target for your company and a priority over implementing predictive scaling. You determine that implementing predictive scaling with its medium potential impact, low cost, and low risk should be the priority improvement after file compression is complete.

You assign a team member to implement improved file compression and add predictive scaling to your backlog.

Test and validate improvements

Perform small tests with minimized investment to reduce the risk of a large-scale effort.

Implement a representative copy of your workload in your testing environment to limit the cost and risk to perform testing and validation. Perform a predefined set of test transactions, measure the provisioned resources, and determine the resources used per unit of work to establish a testing baseline.

Implement your target improvement in the testing environment and repeat the test using the same methodology under the same conditions. Then measure the provisioned resources and resources used per unit of work with your improvement in place.

Calculate the percentage change from your baseline of the resources provisioned per unit of work, and determine the expected quantitative reduction in resources provisioned in your production environment. Compare these values against the anticipated values. Determine if the result is an acceptable level of improvement. Evaluate if any trade-offs in additional resources consumed make the net benefit from the improvement unacceptable.

Determine if the improvement is a success and if resources should be invested in implementing the change in production. If the change is evaluated as unsuccessful at this time, redirect your resources to test and validate your next target and continue your improvement cycle.

% Reduction in provisioned resources per unit of work	Quantitative reduction in provisioned resources	Action
Met expectations	Met expectations	Proceed with improvement
Did not meet expectations	Met expectations	Proceed with improvement
Met expectations	Did not meet expectations	Pursue alternative improvement
Did not meet expectations	Did not meet expectations	Pursue alternative improvement

Applying this step to the Example scenario (p. 5), you perform tests to validate success.

After you perform the tests on the improved compression algorithm, the percentage reduction in resources provisioned per unit of work (the storage required for both the original image and the modified image) met expectations with an average 30% reduction in provisioned storage and negligible increased compute load.

You determine that the additional compute resources required to apply the improved compression algorithm to existing files in production is insignificant compared to the reduction in storage achieved. You confirmed success with the quantitative reduction in resources required (TBs of storage), and the improvement is approved for production deployment.

Deploy changes to production

Implement tested, validated, and approved improvements to production. Implement using limited deployments, confirm the functionality of your workload, test the actual reduction in provisioned resources and resources consumed per unit of work within the limited deployment, and check for unintended consequences of the change. Proceed with full deployments after successful testing.

Revert changes if tests fail or you encounter unacceptable unintended consequences of your change.

Applying this step to the Example scenario (p. 5), you take the following actions.

You implement the changes in production using a limited deployment through a blue-green deployment methodology. Functionality tests against the newly deployed instances are successful. You see a 26% average reduction in provisioned storage for original and manipulated image files. You don't see any evidence of an increase in compute load compressing new files.

You notice an unanticipated decrease in the elapsed time to compress image files, and you attribute this to the highly optimized code for the new compression algorithm.

You proceed with full deployment of the new version.

Measure results and replicate successes

Measure results and replicate successes in the following ways:

- Measure the initial improvement to provisioned resources per unit of work and the quantitative decrease in resources provisioned.
- Compare initial estimates and testing results to your production measurements. Identify factors that might have contributed to differences, and update your estimation and testing methodologies where appropriate.
- Determine success, and degree of success, and share results with stakeholders.
- If you had to revert changes due to failed tests or unintended negative consequences from the change, identify the contributing factors. Iterate where viable, or evaluate new approaches to achieve the goals of the change.
- Take what you have learned, establish standards, and apply successful improvements to other systems that can similarly benefit. Capture and share your methodology, related artifacts, and net benefits, across teams and organizations so that others can adopt your standard and replicate your success.
- Monitor provisioned resources per unit of work and track changes and total impact over time. Changes to your workload, or how your customers consume your workload, can have an impact on the effectiveness of your improvement. Re-evaluate improvement opportunities if you notice significant short-term decreases in the effectiveness of your improvement or an accumulated reduction in effectiveness over time.

- Quantify the net benefit from your improvement over time (including the benefits received by other teams who applied your improvement if available) to show the return on investment from your improvement activities.

Applying this step to the Example scenario (p. 5), you measure the following results.

Your workload shows an initial improvement of 23% reduction in storage requirements after deploying and applying the new compression algorithm to existing image files.

The measured value is largely in agreement with initial estimates (25%), and the significant difference compared to testing (30%) is determined to be the result of the image files used in testing not being representative of image files present in production. You modify the testing image set to more appropriately reflect the images in production.

The improvement is considered a complete success. The total reduction in provisioned storage is 2% less than the estimated 25%, but 23% is still a huge improvement in sustainability impact, and is accompanied by an equivalent cost savings.

The only unintended consequences of the change are the beneficial reduction in elapsed time to perform the compression and an equivalent reduction vCPU consumed. These improvements are attributed to the highly optimized code.

You establish an internal open-source project where you share your code, associated artifacts, guidance on how to implement the change, and the results of your implementation. The internal open-source project makes it easy for your teams to adopt the code for all their persistent file storage use cases. Your teams adopt the improvement as a standard. Secondary benefits of the internal open-source project are that everyone who adopts the solution benefits from improvements to the solution, and anyone can contribute improvements to the project.

You publish your success and share the open-source project across your organization. Every team that adopts the solution replicates the benefit with minimum investment and adds to the net benefit received from your investment. You publish this data as a continuing success story.

You continue to monitor the impact of the improvement over time and will make changes to the internal open-source project as required.

Sustainability as a non-functional requirement

Adding sustainability to your list of business requirements can result in more cost-effective solutions. Focusing on getting more value from the resources you use and using fewer of them directly translates to cost savings on AWS as you pay only for what you use.

Meeting sustainability targets might not require equivalent trade-offs in one or more other traditional metrics such as uptime, availability, or response time. You can achieve significant gains in sustainability with no measurable impact on service levels. Where minor trade-offs are required, the sustainability improvements gained by these trade-offs can outweigh the change in quality of service.

Encourage your team members to continually experiment with sustainability improvements as they develop functional requirements. Teams should also embed proxy metrics when setting goals to ensure that they evaluate resource intensity when developing their workloads.

The following are example trade-offs that can reduce the cloud resources you consume:

Adjust quality of result: You can trade Quality of Results (QoR) for a reduction in workload intensity with approximate computing. The practice of approximate computing looks for opportunities to exploit the gap between what customers need and what you actually produce. For example, if you place your data in a *set* data structure, you can drop the ORDER BY operator in SQL to remove unnecessary processing, saving resources while still providing an acceptable answer.

Adjust response time: An answer with a slower response time can reduce carbon by minimizing shared overhead. Processing ad hoc, ephemeral tasks can incur startup overhead. Group and process tasks in batches instead of paying for overhead each time a task arrives. Batch processing trades increased response time for a reduction in the shared overhead of spinning up an instance, downloading the source code, and running the process.

Adjust availability: With AWS, you can add redundancy and meet high-availability targets with just a few clicks. You can increase redundancy through techniques like static stability by provisioning idle resources that always result in decreased utilization. Evaluate the needs of the business when setting targets. Relatively minor trade-offs in availability can result in much larger improvements in utilization. For example, the static stability architecture pattern involves provisioning idle failover capacity to immediately take on load after a component fault. Relaxing the availability requirement can remove the need for idle online capacity by allowing time for automation to deploy replacement resources. Adding failover capacity on-demand drives higher overall utilization with no impact to the business during normal operations and has the secondary benefit of reducing costs.

Best practices for sustainability in the cloud

Optimize workload placement, and optimize your architecture for demand, software, data, hardware, and process to increase energy efficiency. Each of these areas represents opportunities to employ best practices to reduce the sustainability impact of your cloud workload by maximizing utilization, and minimizing waste and the total resources deployed and powered to support your workload.

Topics

Region selection

The choice of Region for your workload significantly affects its KPIs, including performance, cost, and carbon footprint. To effectively improve these KPIs, you should choose Regions for your workloads based on both business requirements and sustainability goals.

Best practices

SUS01-BP01 Choose Region based on both business requirements and sustainability goals

Choose a Region for your workload based on both your business requirements and sustainability goals to optimize its KPIs, including performance, cost, and carbon footprint.

Common anti-patterns:

- You select the workload's Region based on your own location.
- You consolidate all workload resources into one geographic location.

Benefits of establishing this best practice: Placing a workload close to Amazon renewable energy projects or Regions with low published carbon intensity can help to lower the carbon footprint of a cloud workload.

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

Implementation guidance

The AWS Cloud is a constantly expanding network of Regions and points of presence (PoP), with a global network infrastructure linking them together. The choice of Region for your workload significantly affects its KPIs, including performance, cost, and carbon footprint. To effectively improve these KPIs, you should choose Regions for your workload based on both your business requirements and sustainability goals.

Implementation steps

- Follow these steps to assess and shortlist potential Regions for your workload based on your business requirements including compliance, available features, cost, and latency:
  - Confirm that these Regions are compliant, based on your required local regulations.
  - Use the AWS Regional Services Lists to check if the Regions have the services and features you need to run your workload.
  - Calculate the cost of the workload on each Region using the AWS Pricing Calculator.
  - Test the network latency between your end user locations and each AWS Region.
- Choose Regions near Amazon renewable energy projects and Regions where the grid has a published carbon intensity that is lower than other locations (or Regions).
  - Identify your relevant sustainability guidelines to track and compare year-to-year carbon emissions based on Greenhouse Gas Protocol (market-based and location based methods).
  - Choose region based on method you use to track carbon emissions. For more detail on choosing a Region based on your sustainability guidelines, see How to select a Region for your workload based on sustainability goals.

Resources

Related documents:

- Understanding your carbon emission estimations
- Amazon Around the Globe
- Renewable Energy Methodology
- What to Consider when Selecting a Region for your Workloads

Related videos:

- Architecting sustainably and reducing your AWS carbon footprint

Alignment to demand

The way users and applications consume your workloads and other resources can help you identify improvements to meet sustainability goals. Scale infrastructure to continually match demand and verify that you use only the minimum resources required to support your users. Align service levels to customer needs. Position resources to limit the network required for users and applications to consume them. Remove unused assets. Provide your team members with devices that support their needs and minimize their sustainability impact.

Best practices

- SUS02-BP01 Scale workload infrastructure dynamically (p. 15)
- SUS02-BP02 Align SLAs with sustainability goals (p. 17)
- SUS02-BP03 Stop the creation and maintenance of unused assets (p. 18)

- SUS02-BP04 Optimize geographic placement of workloads based on their networking requirements (p. 19)
- SUS02-BP05 Optimize team member resources for activities performed (p. 21)
- SUS02-BP06 Implement buffering or throttling to flatten the demand curve (p. 22)

SUS02-BP01 Scale workload infrastructure dynamically

Use elasticity of the cloud and scale your infrastructure dynamically to match supply of cloud resources to demand and avoid overprovisioned capacity in your workload.

Common anti-patterns:

- You do not scale your infrastructure with user load.
- You manually scale your infrastructure all the time.
- You leave increased capacity after a scaling event instead of scaling back down.

Benefits of establishing this best practice: Configuring and testing workload elasticity help to efficiently match supply of cloud resources to demand and avoid overprovisioned capacity. You can take advantage of elasticity in the cloud to automatically scale capacity during and after demand spikes to make sure you are only using the right number of resources needed to meet your business requirements.

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

Implementation guidance

The cloud provides the flexibility to expand or reduce your resources dynamically through a variety of mechanisms to meet changes in demand. Optimally matching supply to demand delivers the lowest environmental impact for a workload.

Demand can be fixed or variable, requiring metrics and automation to make sure that management does not become burdensome. Applications can scale vertically (up or down) by modifying the instance size, horizontally (in or out) by modifying the number of instances, or a combination of both.

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:** Pick services (like serverless) that are natively scaling by design or provide auto scaling as a feature.

Identify periods of low or no utilization and scale resources to remove excess capacity and improve efficiency.

Implementation steps

- Elasticity matches the supply of resources you have against the demand for those resources. Instances, containers, and functions provide mechanisms for elasticity, either in combination with automatic scaling or as a feature of the service. AWS provides a range of auto scaling mechanisms to ensure that workloads can scale down quickly and easily during periods of low user load. Here are some examples of auto scaling mechanisms:

Auto scaling mechanism	Where to use
Amazon EC2 Auto Scaling	Use to verify you have the correct number of Amazon EC2 instances available to handle the user load for your application.
Application Auto Scaling	Use to automatically scale the resources for individual AWS services beyond Amazon EC2, such as Lambda functions or Amazon Elastic Container Service (Amazon ECS) services.
Kubernetes Cluster Autoscaler	Use to automatically scale Kubernetes clusters on AWS.

- Scaling is often discussed related to compute services like Amazon EC2 instances or AWS Lambda functions. Consider the configuration of non-compute services like Amazon DynamoDB read and write capacity units or Amazon Kinesis Data Streams shards to match the demand.
- Verify that the metrics for scaling up or down are validated against the type of workload being deployed. If you are deploying a video transcoding application, 100% CPU utilization is expected and should not be your primary metric. You can use a customized metric (such as memory utilization) 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.
- 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 scale-out and scale-in events. Create test scenarios for scale-in events to verify that the workload behaves as expected and does not affect the user experience (like losing sticky sessions). 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:

- Getting Started with Amazon EC2 Auto Scaling
- Predictive Scaling for EC2, Powered by Machine Learning
- Analyze user behavior using Amazon OpenSearch Service, Amazon Kinesis Data Firehose and Kibana
- What is Amazon CloudWatch?
- Monitoring DB load with Performance Insights on Amazon RDS
- Introducing Native Support for Predictive Scaling with Amazon EC2 Auto Scaling
- Introducing Karpenter - An Open-Source, High-Performance Kubernetes Cluster Autoscaler
- Deep Dive on Amazon ECS Cluster Auto Scaling

Related videos:

- Build a cost-, energy-, and resource-efficient compute environment

- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)

Related examples:

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

SUS02-BP02 Align SLAs with sustainability goals

Review and optimize workload service-level agreements (SLA) based on your sustainability goals to minimize the resources required to support your workload while continuing to meet business needs.

Common anti-patterns:

- Workload SLAs are unknown or ambiguous.
- You define your SLA just for availability and performance.
- You use the same design pattern (like Multi-AZ architecture) for all your workloads.

Benefits of establishing this best practice: Aligning SLAs with sustainability goals leads to optimal resource usage while meeting business needs.

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

Implementation guidance

SLAs define the level of service expected from a cloud workload, such as response time, availability, and data retention. They influence the architecture, resource usage, and environmental impact of a cloud workload. At a regular cadence, review SLAs and make trade-offs that significantly reduce resource usage in exchange for acceptable decreases in service levels.

Implementation steps

- Define or redesign SLAs that support your sustainability goals while meeting your business requirements, not exceeding them.
- Make trade-offs that significantly reduce sustainability impacts in exchange for acceptable decreases in service levels.
  - **Sustainability and reliability:** Highly available workloads tend to consume more resources.
  - **Sustainability and performance:** Using more resources to boost performance could have a higher environmental impact.
  - **Sustainability and security:** Overly secure workloads could have a higher environmental impact.
- Use design patterns such as microservices on AWS that prioritize business-critical functions and allow lower service levels (such as response time or recovery time objectives) for non-critical functions.

Resources

Related documents:

- AWS Service Level Agreements (SLAs)
- Importance of Service Level Agreement for SaaS Providers

Related videos:

- Delivering sustainable, high-performing architectures
- Build a cost-, energy-, and resource-efficient compute environment

SUS02-BP03 Stop the creation and maintenance of unused assets

Decommission unused assets in your workload to reduce the number of cloud resources required to support your demand and minimize waste.

Common anti-patterns:

- You do not analyze your application for assets that are redundant or no longer required.
- You do not remove assets that are redundant or no longer required.

Benefits of establishing this best practice: Removing unused assets frees resources and improves the overall efficiency of the workload.

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

Implementation guidance

Unused assets consume cloud resources like storage space and compute power. By identifying and eliminating these assets, you can free up these resources, resulting in a more efficient cloud architecture. Perform regular analysis on application assets such as pre-compiled reports, datasets, static images, and asset access patterns to identify redundancy, underutilization, and potential decommission targets. Remove those redundant assets to reduce the resource waste in your workload.

Implementation steps

- Use monitoring tools to identify static assets that are no longer required.
- Before removing any asset, evaluate the impact of removing it on the architecture.
- Develop a plan and remove assets that are no longer required.
- Consolidate overlapping generated assets to remove redundant processing.
- Update your applications to no longer produce and store assets that are not required.
- Instruct third parties to stop producing and storing assets managed on your behalf that are no longer required.
- Instruct third parties to consolidate redundant assets produced on your behalf.
- Regularly review your workload to identify and remove unused assets.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part II: Storage
- How do I terminate active resources that I no longer need on my AWS account?

Related videos:

- How do I check for and then remove active resources that I no longer need on my AWS account?

SUS02-BP04 Optimize geographic placement of workloads based on their networking requirements

This best practice was updated with new guidance on July 13th, 2023.

Select cloud location and services for your workload that reduce the distance network traffic must travel and decrease the total network resources required to support your workload.

Common anti-patterns:

- You select the workload's Region based on your own location.
- You consolidate all workload resources into one geographic location.
- All traffic flows through your existing data centers.

Benefits of establishing this best practice: Placing a workload close to its users provides the lowest latency while decreasing data movement across the network and reducing environmental impact.

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

Implementation guidance

The AWS Cloud infrastructure is built around location options such as Regions, Availability Zones, placement groups, and edge locations such as AWS Outposts and AWS Local Zones. These location options are responsible for maintaining connectivity between application components, cloud services, edge networks and on-premises data centers.

Analyze the network access patterns in your workload to identify how to use these cloud location options and reduce the distance network traffic must travel.

Implementation steps

- 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 the Regions for your workload deployment based on the following key elements:
  - **Your Sustainability goal:** as explained in Region selection.
  - **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 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:

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) request 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.

- Use connection pooling to allow for connection reuse and reduce required resources.
- Use distributed data stores that don't rely on persistent connections and synchronous updates for consistency to serve regional populations.
- Replace pre-provisioned static network capacity with shared dynamic capacity, and share the sustainability impact of network capacity with other subscribers.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking
- Amazon ElastiCache Documentation
- What is Amazon CloudFront?
- Amazon CloudFront Key Features

Related videos:

- Demystifying data transfer on AWS
- Scaling network performance on next-gen Amazon EC2 instances

Related examples:

- AWS Networking Workshops
- Architecting for sustainability - Minimize data movement across networks

SUS02-BP05 Optimize team member resources for activities performed

Optimize resources provided to team members to minimize the environmental sustainability impact while supporting their needs.

Common anti-patterns:

- You ignore the impact of devices used by your team members on the overall efficiency of your cloud application.
- You manually manage and update resources used by team members.

Benefits of establishing this best practice: Optimizing team member resources improves the overall efficiency of cloud-enabled applications.

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

Implementation guidance

Understand the resources your team members use to consume your services, their expected lifecycle, and the financial and sustainability impact. Implement strategies to optimize these resources. For example, perform complex operations, such as rendering and compilation, on highly utilized scalable infrastructure instead of on underutilized high-powered single-user systems.

Implementation steps

- Provision workstations and other devices to align with how they're used.
- Use virtual desktops and application streaming to limit upgrade and device requirements.
- Move processor or memory-intensive tasks to the cloud to use its elasticity.
- Evaluate the impact of processes and systems on your device lifecycle, and select solutions that minimize the requirement for device replacement while satisfying business requirements.
- Implement remote management for devices to reduce required business travel.
  - AWS Systems Manager Fleet Manager is a unified user interface (UI) experience that helps you remotely manage your nodes running on AWS or on premises.

Resources

Related documents:

- What is Amazon WorkSpaces?
- Cost Optimizer for Amazon WorkSpaces
- Amazon AppStream 2.0 Documentation
- NICE DCV

Related videos:

- Managing cost for Amazon WorkSpaces on AWS

SUS02-BP06 Implement buffering or throttling to flatten the demand curve

Buffering and throttling flatten the demand curve and reduce the provisioned capacity required for your workload.

Common anti-patterns:

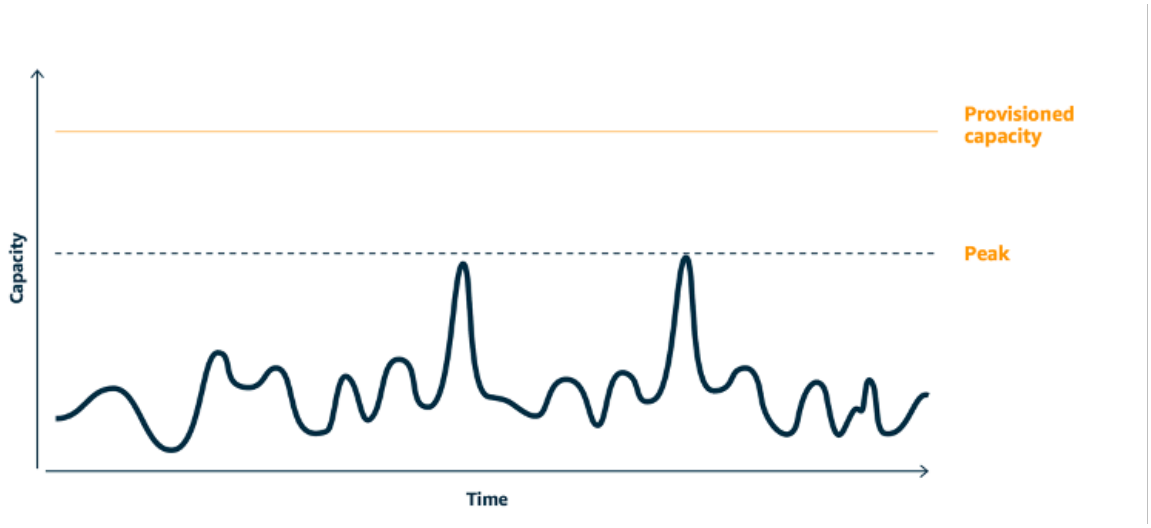
- You process the client requests immediately while it is not needed.
- You do not analyze the requirements for client requests.

Benefits of establishing this best practice: Flattening the demand curve reduce the required provisioned capacity for the workload. Reducing the provisioned capacity means less energy consumption and less environmental impact.

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

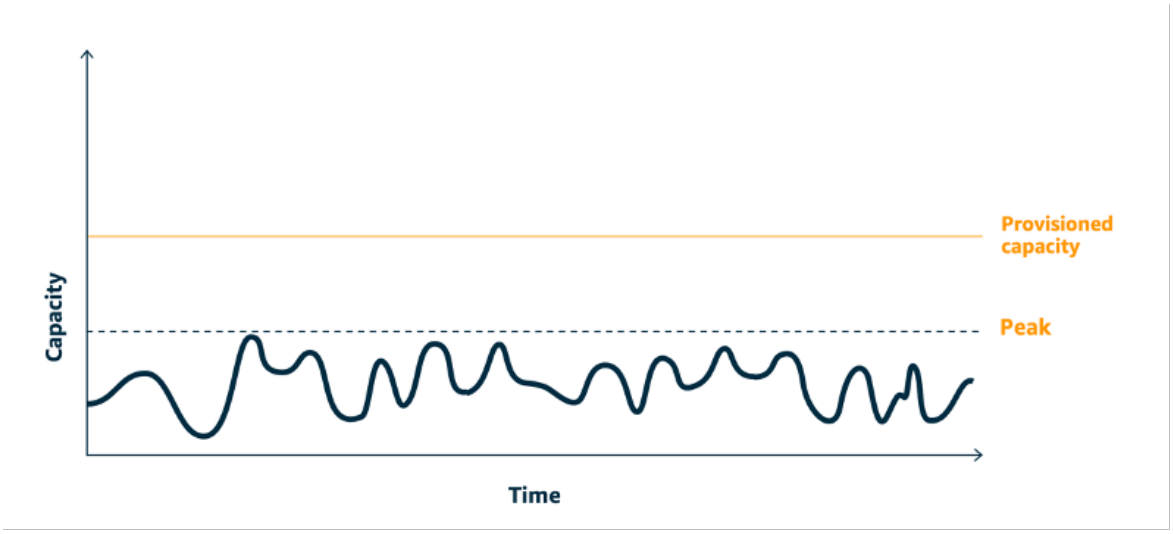
Implementation guidance

Flattening the workload demand curve can help you to reduce the provisioned capacity for a workload and reduce its environmental impact. Assume a workload with the demand curve shown in below figure. This workload has two peaks, and to handle those peaks, the resource capacity as shown by orange line is provisioned. The resources and energy used for this workload is not indicated by the area under the demand curve, but the area under the provisioned capacity line, as provisioned capacity is needed to handle those two peaks.



Demand curve with two distinct peaks that require high provisioned capacity.

You can use buffering or throttling to modify the demand curve and smooth out the peaks, which means less provisioned capacity and less energy consumed. Implement throttling when your clients can perform retries. Implement buffering to store the request and defer processing until a later time.



Throttling's effect on the demand curve and provisioned capacity.

Implementation steps

- Analyze the client requests to determine how to respond to them. Questions to consider include:
  - Can this request be processed asynchronously?
  - Does the client have retry capability?
- If the client has retry capability, then you can implement throttling, which tells the source that if it cannot service the request at the current time, it should try again later.
  - You can use Amazon API Gateway to implement throttling.
- For clients that cannot perform retries, a buffer needs to be implemented to flatten the demand curve. A buffer defers request processing, allowing applications that run at different rates to communicate effectively. A buffer-based approach uses a queue or a stream to accept messages from producers. Messages are read by consumers and processed, allowing the messages to run at the rate that meets the consumers' business requirements.
  - Amazon Simple Queue Service (Amazon SQS) is a managed service that provides queues that allow a single consumer to read individual messages.
  - Amazon Kinesis provides a stream that allows many consumers to read the same messages.
- Analyze the overall demand, rate of change, and required response time to right size the throttle or buffer required.

Resources

Related documents:

- Getting started with Amazon SQS
- Application integration Using Queues and Messages

Related videos:

- Choosing the Right Messaging Service for Your Distributed App

Software and architecture

Implement patterns for performing load smoothing and maintaining consistent high utilization of deployed resources to minimize the resources consumed. Components might become idle from lack of use because of changes in user behavior over time. Revise patterns and architecture to consolidate under-utilized components to increase overall utilization. Retire components that are no longer required. Understand the performance of your workload components, and optimize the components that consume the most resources. Be aware of the devices your customers use to access your services, and implement patterns to minimize the need for device upgrades.

Best practices

SUS03-BP01 Optimize software and architecture for asynchronous and scheduled jobs

Use efficient software and architecture patterns such as queue-driven to maintain consistent high utilization of deployed resources.

Common anti-patterns:

- You overprovision the resources in your cloud workload to meet unforeseen spikes in demand.
- Your architecture does not decouple senders and receivers of asynchronous messages by a messaging component.

Benefits of establishing this best practice:

- Efficient software and architecture patterns minimize the unused resources in your workload and improve the overall efficiency.
- You can scale the processing independently of the receiving of asynchronous messages.
- Through a messaging component, you have relaxed availability requirements that you can meet with fewer resources.

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

Implementation guidance

Use efficient architecture patterns such as event-driven architecture that result in even utilization of components and minimize overprovisioning in your workload. Using efficient architecture patterns minimizes idle resources from lack of use due to changes in demand over time.

Understand the requirements of your workload components and adopt architecture patterns that increase overall utilization of resources. Retire components that are no longer required.

Implementation steps

- Analyze the demand for your workload to determine how to respond to those.

- For requests or jobs that don't require synchronous responses, use queue-driven architectures and auto scaling workers to maximize utilization. Here are some examples of when you might consider queue-driven architecture:

Queuing mechanism	Description
AWS Batch job queues	AWS Batch jobs are submitted to a job queue where they reside until they can be scheduled to run in a compute environment.
Amazon Simple Queue Service and Amazon EC2 Spot Instances	Pairing Amazon SQS and Spot Instances to build fault tolerant and efficient architecture.

- For requests or jobs that can be processed anytime, use scheduling mechanisms to process jobs in batch for more efficiency. Here are some examples of scheduling mechanisms on AWS:

Scheduling mechanism	Description
Amazon EventBridge Scheduler	A capability from Amazon EventBridge that allows you to create, run, and manage scheduled tasks at scale.
AWS Glue time-based schedule	Define a time-based schedule for your crawlers and jobs in AWS Glue.
Amazon Elastic Container Service (Amazon ECS) scheduled tasks	Amazon ECS supports creating scheduled tasks. Scheduled tasks use Amazon EventBridge rules to run tasks either on a schedule or in a response to an EventBridge event.
Instance Scheduler	Configure start and stop schedules for your Amazon EC2 and Amazon Relational Database Service instances.

- If you use polling and webhooks mechanisms in your architecture, replace those with events. Use event-driven architectures to build highly efficient workloads.
- Leverage serverless on AWS to eliminate over-provisioned infrastructure.
- Right size individual components of your architecture to prevent idling resources waiting for input.

Resources

Related documents:

- What is Amazon Simple Queue Service?
- What is Amazon MQ?
- Scaling based on Amazon SQS
- What is AWS Step Functions?
- What is AWS Lambda?
- Using AWS Lambda with Amazon SQS
- What is Amazon EventBridge?

Related videos:

- Moving to event-driven architectures

SUS03-BP02 Remove or refactor workload components with low or no use

Remove components that are unused and no longer required, and refactor components with little utilization to minimize waste in your workload.

Common anti-patterns:

- You do not regularly check the utilization level of individual components of your workload.
- You do not check and analyze recommendations from AWS rightsizing tools such as AWS Compute Optimizer.

Benefits of establishing this best practice: Removing unused components minimizes waste and improves the overall efficiency of your cloud workload.

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

Implementation guidance

Review your workload to identify idle or unused components. This is an iterative improvement process which can be initiated by changes in demand or the release of a new cloud service. For example, a significant drop in AWS Lambda function run time can be an indicator of a need to lower the memory size. Also, as AWS releases new services and features, the optimal services and architecture for your workload may change.

Continually monitor workload activity and look for opportunities to improve the utilization level of individual components. By removing idle components and performing rightsizing activities, you meet your business requirements with the fewest cloud resources.

Implementation steps

- Monitor and capture the utilization metrics for critical components of your workload (like CPU utilization, memory utilization, or network throughput in Amazon CloudWatch metrics).
- For stable workloads, check AWS rightsizing tools such as AWS Compute Optimizer at regular intervals to identify idle, unused, or underutilized components.
- For ephemeral workloads, evaluate utilization metrics to identify idle, unused, or underutilized components.
- Retire components and associated assets (like Amazon ECR images) that are no longer needed.
- Refactor or consolidate underutilized components with other resources to improve utilization efficiency. For example, you can provision multiple small databases on a single Amazon RDS database instance instead of running databases on individual under-utilized instances.
- Understand the resources provisioned by your workload to complete a unit of work.

Resources

Related documents:

- AWS Trusted Advisor
- What is Amazon CloudWatch?
- Automated Cleanup of Unused Images in Amazon ECR

Related examples:

- Well-Architected Lab - Rightsizing with AWS Compute Optimizer
- Well-Architected Lab - Optimize Hardware Patterns and Observe Sustainability KPIs

SUS03-BP03 Optimize areas of code that consume the most time or resources

This best practice was updated with new guidance on July 13th, 2023.

Optimize your code that runs within different components of your architecture to minimize resource usage while maximizing performance.

Common anti-patterns:

- You ignore optimizing your code for resource usage.
- You usually respond to performance issues by increasing the resources.
- Your code review and development process does not track performance changes.

Benefits of establishing this best practice: Using efficient code minimizes resource usage and improves performance.

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

Implementation guidance

It is crucial to examine every functional area, including the code for a cloud architected application, to optimize its resource usage and performance. Continually monitor your workload's performance in build environments and production and identify opportunities to improve code snippets that have particularly high resource usage. Adopt a regular review process to identify bugs or anti-patterns within your code that use resources inefficiently. Leverage simple and efficient algorithms that produce the same results for your use case.

Implementation steps

- While developing your workloads, adopt an automated code review process to improve quality and identify bugs and anti-patterns.
  - Automate code reviews with Amazon CodeGuru Reviewer
  - Detecting concurrency bugs with Amazon CodeGuru
  - Raising code quality for Python applications using Amazon CodeGuru
- As you run your workloads, monitor resources to identify components with high resource requirements per unit of work as targets for code reviews.
- For code reviews, use a code profiler to identify the areas of code that use the most time or resources as targets for optimization.
  - Reducing your organization's carbon footprint with Amazon CodeGuru Profiler
  - Understanding memory usage in your Java application with Amazon CodeGuru Profiler
  - Improving customer experience and reducing cost with Amazon CodeGuru Profiler
- Use the most efficient operating system and programming language for the workload. For details on energy efficient programming languages (including Rust), see Sustainability with Rust.
- Replace computationally intensive algorithms with simpler and more efficient version that produce the same result.

- Remove unnecessary code such as sorting and formatting.

Resources

Related documents:

- What is Amazon CodeGuru Profiler?
- FPGA instances
- The AWS SDKs on Tools to Build on AWS

Related videos:

- Improve Code Efficiency Using Amazon CodeGuru Profiler
- Automate Code Reviews and Application Performance Recommendations with Amazon CodeGuru

SUS03-BP04 Optimize impact on devices and equipment

Understand the devices and equipment used in your architecture and use strategies to reduce their usage. This can minimize the overall environmental impact of your cloud workload.

Common anti-patterns:

- You ignore the environmental impact of devices used by your customers.
- You manually manage and update resources used by customers.

Benefits of establishing this best practice: Implementing software patterns and features that are optimized for customer device can reduce the overall environmental impact of cloud workload.

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

Implementation guidance

Implementing software patterns and features that are optimized for customer devices can reduce the environmental impact in several ways:

- Implementing new features that are backward compatible can reduce the number of hardware replacements.
- Optimizing an application to run efficiently on devices can help to reduce their energy consumption and extend their battery life (if they are powered by battery).
- Optimizing an application for devices can also reduce the data transfer over the network.

Understand the devices and equipment used in your architecture, their expected lifecycle, and the impact of replacing those components. Implement software patterns and features that can help to minimize the device energy consumption, the need for customers to replace the device and also upgrade it manually.

Implementation steps

- Inventory the devices used in your architecture. Devices can be mobile, tablet, IOT devices, smart light, or even smart devices in a factory.
- Optimize the application running on the devices:
  - Use strategies such as running tasks in the background to reduce their energy consumption.

  - Account for network bandwidth and latency when building payloads, and implement capabilities that help your applications work well on low bandwidth, high latency links.
  - Convert payloads and files into optimized formats required by devices. For example, you can use Amazon Elastic Transcoder or AWS Elemental MediaConvert to convert large, high quality digital media files into formats that users can play back on mobile devices, tablets, web browsers, and connected televisions.
  - Perform computationally intense activities server-side (such as image rendering), or use application streaming to improve the user experience on older devices.
  - Segment and paginate output, especially for interactive sessions, to manage payloads and limit local storage requirements.
- Use automated over-the-air (OTA) mechanism to deploy updates to one or more devices.
  - You can use a CI/CD pipeline to update mobile applications.
  - You can use AWS IoT Device Management to remotely manage connected devices at scale.
- To test new features and updates, use managed device farms with representative sets of hardware and iterate development to maximize the devices supported. For more details, see SUS06-BP04 Use managed device farms for testing (p. 53).

Resources

Related documents:

- What is AWS Device Farm?
- Amazon AppStream 2.0 Documentation
- NICE DCV
- OTA tutorial for updating firmware on devices running FreeRTOS

Related videos:

- Introduction to AWS Device Farm

SUS03-BP05 Use software patterns and architectures that best support data access and storage patterns

Understand how data is used within your workload, consumed by your users, transferred, and stored. Use software patterns and architectures that best support data access and storage to minimize the compute, networking, and storage resources required to support the workload.

Common anti-patterns:

- You assume that all workloads have similar data storage and access patterns.
- You only use one tier of storage, assuming all workloads fit within that tier.
- You assume that data access patterns will stay consistent over time.
- Your architecture supports a potential high data access burst, which results in the resources remaining idle most of the time.

Benefits of establishing this best practice: Selecting and optimizing your architecture based on data access and storage patterns will help decrease development complexity and increase overall utilization. Understanding when to use global tables, data partitioning, and caching will help you decrease operational overhead and scale based on your workload needs.

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

Implementation guidance

Use software and architecture patterns that aligns best to your data characteristics and access patterns. For example, use modern data architecture on AWS that allows you to use purpose-built services optimized for your unique analytics use cases. These architecture patterns allow for efficient data processing and reduce the resource usage.

Implementation steps

- Analyze your data characteristics and access patterns to identify the correct configuration for your cloud resources. Key characteristics to consider include:
  - **Data type:** structured, semi-structured, unstructured
  - **Data growth:** bounded, unbounded
  - **Data durability:** persistent, ephemeral, transient
  - **Access patterns** reads or writes, update frequency, spiky, or consistent
- Use architecture patterns that best support data access and storage patterns.
  - Let's Architect! Modern data architectures
  - Databases on AWS: The Right Tool for the Right Job
- Use technologies that work natively with compressed data.
- Use purpose-built analytics services for data processing in your architecture.
- Use the database engine that best supports your dominant query pattern. Manage your database indexes to ensure efficient querying. For further details, see AWS Databases.
- Select network protocols that reduce the amount of network capacity consumed in your architecture.

Resources

Related documents:

- Athena Compression Support file formats
- COPY from columnar data formats with Amazon Redshift
- Converting Your Input Record Format in Kinesis Data Firehose
- Format Options for ETL Inputs and Outputs in AWS Glue
- Improve query performance on Amazon Athena by Converting to Columnar Formats
- Loading compressed data files from Amazon S3 with Amazon Redshift
- Monitoring DB load with Performance Insights on Amazon Aurora
- Monitoring DB load with Performance Insights on Amazon RDS
- Amazon S3 Intelligent-Tiering storage class

Related videos:

- Building modern data architectures on AWS

Data

Implement data management practices to reduce the provisioned storage required to support your workload, and the resources required to use it. Understand your data, and use storage technologies

and configurations that best support the business value of the data and how it's used. Lifecycle data to more efficient, less performant storage when requirements decrease, and delete data that's no longer required.

Best practices

SUS04-BP01 Implement a data classification policy

Classify data to understand its criticality to business outcomes and choose the right energy-efficient storage tier to store the data.

Common anti-patterns:

- You do not identify data assets with similar characteristics (such as sensitivity, business criticality, or regulatory requirements) that are being processed or stored.
- You have not implemented a data catalog to inventory your data assets.

Benefits of establishing this best practice: Implementing a data classification policy allows you to determine the most energy-efficient storage tier for data.

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

Implementation guidance

Data classification involves identifying the types of data that are being processed and stored in an information system owned or operated by an organization. It also involves making a determination on the criticality of the data and the likely impact of a data compromise, loss, or misuse.

Implement data classification policy by working backwards from the contextual use of the data and creating a categorization scheme that takes into account the level of criticality of a given dataset to an organization's operations.

Implementation steps

- Conduct an inventory of the various data types that exist for your workload.
  - For more detail on data classification categories, see Data Classification whitepaper.
- Determine criticality, confidentiality, integrity, and availability of data based on risk to the organization. Use these requirements to group data into one of the data classification tiers that you adopt.
  - As an example, see Four simple steps to classify your data and secure your startup.
- Periodically audit your environment for untagged and unclassified data, and classify and tag the data appropriately.
  - As an example, see Data Catalog and crawlers in AWS Glue.
- Establish a data catalog that provides audit and governance capabilities.

- Determine and document the handling procedures for each data class.
- Use automation to continually audit your environment to identify untagged and unclassified data, and classify and tag the data appropriately.

Resources

Related documents:

- Leveraging AWS Cloud to Support Data Classification
- Tag policies from AWS Organizations

Related videos:

- Enabling agility with data governance on AWS

SUS04-BP02 Use technologies that support data access and storage patterns

This best practice was updated with new guidance on July 13th, 2023.

Use storage technologies that best support how your data is accessed and stored to minimize the resources provisioned while supporting your workload.

Common anti-patterns:

- You assume that all workloads have similar data storage and access patterns.
- You only use one tier of storage, assuming all workloads fit within that tier.
- You assume that data access patterns will stay consistent over time.

Benefits of establishing this best practice: Selecting and optimizing your storage technologies based on data access and storage patterns will help you reduce the required cloud resources to meet your business needs and improve the overall efficiency of cloud workload.

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

Implementation guidance

Select the storage solution that aligns best to your access patterns, or consider changing your access patterns to align with the storage solution to maximize performance efficiency.

- Evaluate your data characteristics and access pattern to collect the key characteristics of your storage needs. Key characteristics to consider include:
  - **Data type:** structured, semi-structured, unstructured
  - **Data growth:** bounded, unbounded
  - **Data durability:** persistent, ephemeral, transient
  - **Access patterns:** reads or writes, frequency, spiky, or consistent
- Migrate data to the appropriate storage technology that supports your data characteristics and access pattern. Here are some examples of AWS storage technologies and their key characteristics:

Type	Technology	Key characteristics
Object storage	Amazon S3	An object storage service with 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	Storage class of Amazon S3 built for data-archiving.
Shared file system	Amazon Elastic File System (Amazon EFS)	Mountable file system that can be accessed by multiple types of compute solutions. Amazon EFS automatically grows and shrinks storage and is performance-optimized to deliver consistent low latencies.
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-backed storage for throughput-intensive workloads.

Type	Technology	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 systems 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.

- For storage systems that are a fixed size, such as Amazon EBS or Amazon FSx, monitor the available storage space and automate storage allocation on reaching a threshold. You can leverage Amazon CloudWatch to collect and analyze different metrics for Amazon EBS and Amazon FSx.
- Amazon S3 Storage Classes can be configured at the object level and a single bucket can contain objects stored across all of the storage classes.
- You can also use Amazon S3 Lifecycle policies to automatically transition objects between storage classes or remove data without any application changes. In general, you have to make a trade-off between resource efficiency, access latency, and reliability when considering these storage mechanisms.

Resources

Related documents:

- Amazon EBS volume types
- Amazon EC2 instance store
- Amazon S3 Intelligent-Tiering
- Amazon EBS I/O Characteristics
- Using Amazon S3 storage classes
- What is Amazon S3 Glacier?

Related videos:

- Architectural Patterns for Data Lakes on AWS
- Deep dive on Amazon EBS (STG303-R1)
- Optimize your storage performance with Amazon S3 (STG343)
- Building modern data architectures on AWS

Related examples:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 Examples

SUS04-BP03 Use policies to manage the lifecycle of your datasets

Manage the lifecycle of all of your data and automatically enforce deletion to minimize the total storage required for your workload.

Common anti-patterns:

- You manually delete data.
- You do not delete any of your workload data.
- You do not move data to more energy-efficient storage tiers based on its retention and access requirements.

Benefits of establishing this best practice: Using data lifecycle policies ensures efficient data access and retention in a workload.

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

Implementation guidance

Datasets usually have different retention and access requirements during their lifecycle. For example, your application may need frequent access to some datasets for a limited period of time. After that, those datasets are infrequently accessed.

To efficiently manage your datasets throughout their lifecycle, configure lifecycle policies, which are rules that define how to handle datasets.

With Lifecycle configuration rules, you can tell the specific storage service to transition a dataset to more energy-efficient storage tiers, archive it, or delete it.

Implementation steps

- Classify datasets in your workload.
- Define handling procedures for each data class.
- Set automated lifecycle policies to enforce lifecycle rules. Here are some examples of how to set up automated lifecycle policies for different AWS storage services:

Storage service	How to set automated lifecycle policies
Amazon S3	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

Storage service	How to set automated lifecycle policies
	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 Elastic Block Store	You can use Amazon Data Lifecycle Manager to automate the creation, retention, and deletion of Amazon EBS snapshots and Amazon EBS-backed AMIs.
Amazon Elastic File System	Amazon EFS lifecycle management automatically manages file storage for your file systems.
Amazon Elastic Container Registry	Amazon ECR lifecycle policies automate the cleanup of your container images by expiring images based on age or count.
AWS Elemental MediaStore	You can use an object lifecycle policy that governs how long objects should be stored in the MediaStore container.

- Delete unused volumes, snapshots, and data that is out of its retention period. Leverage native service features like Amazon DynamoDB Time To Live or Amazon CloudWatch log retention for deletion.
- Aggregate and compress data where applicable based on lifecycle rules.

Resources

Related documents:

- Optimize your Amazon S3 Lifecycle rules with Amazon S3 Storage Class Analysis
- Evaluating Resources with AWS Config Rules

Related videos:

- Simplify Your Data Lifecycle and Optimize Storage Costs With Amazon S3 Lifecycle
- Reduce Your Storage Costs Using Amazon S3 Storage Lens

SUS04-BP04 Use elasticity and automation to expand block storage or file system

Use elasticity and automation to expand block storage or file system as data grows to minimize the total provisioned storage.

Common anti-patterns:

- You procure large block storage or file system for future need.
- You overprovision the input and output operations per second (IOPS) of your file system.
- You do not monitor the utilization of your data volumes.

Benefits of establishing this best practice: Minimizing over-provisioning for storage system reduces the idle resources and improves the overall efficiency of your workload.

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

Implementation guidance

Create block storage and file systems with size allocation, throughput, and latency that are appropriate for your workload. Use elasticity and automation to expand block storage or file system as data grows without having to over-provision these storage services.

Implementation steps

- For fixed size storage like Amazon EBS, verify that you are monitoring the amount of storage used versus the overall storage size and create automation, if possible, to increase the storage size when reaching a threshold.
- Use elastic volumes and managed block data services to automate allocation of additional storage as your persistent data grows. As an example, you can use Amazon EBS Elastic Volumes to change volume size, volume type, or adjust the performance of your Amazon EBS volumes.
- Choose the right storage class, performance mode, and throughput mode for your file system to address your business need, not exceeding that.
  - Amazon EFS performance
  - Amazon EBS volume performance on Linux instances
- Set target levels of utilization for your data volumes, and resize volumes outside of expected ranges.
- Right size read-only volumes to fit the data.
- Migrate data to object stores to avoid provisioning the excess capacity from fixed volume sizes on block storage.
- Regularly review elastic volumes and file systems to terminate idle volumes and shrink over-provisioned resources to fit the current data size.

Resources

Related documents:

- Amazon FSx Documentation
- What is Amazon Elastic File System?

Related videos:

- Deep Dive on Amazon EBS Elastic Volumes
- Amazon EBS and Snapshot Optimization Strategies for Better Performance and Cost Savings
- Optimizing Amazon EFS for cost and performance, using best practices

SUS04-BP05 Remove unneeded or redundant data

Remove unneeded or redundant data to minimize the storage resources required to store your datasets.

Common anti-patterns:

- You duplicate data that can be easily obtained or recreated.
- You back up all data without considering its criticality.
- You only delete data irregularly, on operational events, or not at all.
- You store data redundantly irrespective of the storage service's durability.
- You turn on Amazon S3 versioning without any business justification.

Benefits of establishing this best practice: Removing unneeded data reduces the storage size required for your workload and the workload environmental impact.

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

Implementation guidance

Do not store data that you do not need. Automate the deletion of unneeded data. Use technologies that deduplicate data at the file and block level. Leverage native data replication and redundancy features of services.

Implementation steps

- Evaluate if you can avoid storing data by using existing publicly available datasets in AWS Data Exchange and Open Data on AWS.
- Use mechanisms that can deduplicate data at the block and object level. Here are some examples of how to deduplicate data on AWS:

Storage service	Deduplication mechanism
Amazon S3	Use AWS Lake Formation FindMatches to find matching records across a dataset (including ones without identifiers) by using the new FindMatches ML Transform.
Amazon FSx	Use data deduplication on Amazon FSx for Windows.
Amazon Elastic Block Store snapshots	Snapshots are incremental backups, which means that only the blocks on the device that have changed after your most recent snapshot are saved.

- Analyze the data access to identify unneeded data. Automate lifecycle policies. Leverage native service features like Amazon DynamoDB Time To Live, Amazon S3 Lifecycle, or Amazon CloudWatch log retention for deletion.
- Use data virtualization capabilities on AWS to maintain data at its source and avoid data duplication.
  - Cloud Native Data Virtualization on AWS
  - Lab: Optimize Data Pattern Using Amazon Redshift Data Sharing
- Use backup technology that can make incremental backups.
- Leverage the durability of Amazon S3 and replication of Amazon EBS to meet your durability goals instead of self-managed technologies (such as a redundant array of independent disks (RAID)).
- Centralize log and trace data, deduplicate identical log entries, and establish mechanisms to tune verbosity when needed.
- Pre-populate caches only where justified.
- Establish cache monitoring and automation to resize the cache accordingly.
- Remove out-of-date deployments and assets from object stores and edge caches when pushing new versions of your workload.

Resources

Related documents:

- Change log data retention in CloudWatch Logs

- Data deduplication on Amazon FSx for Windows File Server
- Features of Amazon FSx for ONTAP including data deduplication
- Invalidating Files on Amazon CloudFront
- Using AWS Backup to back up and restore Amazon EFS file systems
- What is Amazon CloudWatch Logs?
- Working with backups on Amazon RDS

Related videos:

- Fuzzy Matching and Deduplicating Data with ML Transforms for AWS Lake Formation

Related examples:

- How do I analyze my Amazon S3 server access logs using Amazon Athena?

SUS04-BP06 Use shared file systems or storage to access common data

Adopt shared file systems or storage to avoid data duplication and allow for more efficient infrastructure for your workload.

Common anti-patterns:

- You provision storage for each individual client.
- You do not detach data volume from inactive clients.
- You do not provide access to storage across platforms and systems.

Benefits of establishing this best practice: Using shared file systems or storage allows for sharing data to one or more consumers without having to copy the data. This helps to reduce the storage resources required for the workload.

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

Implementation guidance

If you have multiple users or applications accessing the same datasets, using shared storage technology is crucial to use efficient infrastructure for your workload. Shared storage technology provides a central location to store and manage datasets and avoid data duplication. It also enforces consistency of the data across different systems. Moreover, shared storage technology allows for more efficient use of compute power, as multiple compute resources can access and process data at the same time in parallel.

Fetch data from these shared storage services only as needed and detach unused volumes to free up resources.

Implementation steps

- Migrate data to shared storage when the data has multiple consumers. Here are some examples of shared storage technology on AWS:

Storage option	When to use
Amazon EBS Multi-Attach	Amazon EBS Multi-Attach allows you to attach a single Provisioned IOPS SSD (io1 or io2) volume to multiple instances that are in the same Availability Zone.
Amazon EFS	See When to Choose Amazon EFS.
Amazon FSx	See Choosing an Amazon FSx File System.
Amazon S3	Applications that do not require a file system structure and are designed to work with object storage can use Amazon S3 as a massively scalable, durable, low-cost object storage solution.

- Copy data to or fetch data from shared file systems only as needed. As an example, you can create an Amazon FSx for Lustre file system backed by Amazon S3 and only load the subset of data required for processing jobs to Amazon FSx.
- Delete data as appropriate for your usage patterns as outlined in SUS04-BP03 Use policies to manage the lifecycle of your datasets (p. 35).
- Detach volumes from clients that are not actively using them.

Resources

Related documents:

- Linking your file system to an Amazon S3 bucket
- Using Amazon EFS for AWS Lambda in your serverless applications
- Amazon EFS Intelligent-Tiering Optimizes Costs for Workloads with Changing Access Patterns
- Using Amazon FSx with your on-premises data repository

related videos:

- Storage cost optimization with Amazon EFS

SUS04-BP07 Minimize data movement across networks

This best practice was updated with new guidance on July 13th, 2023.

Use shared file systems or object storage to access common data and minimize the total networking resources required to support data movement for your workload.

Common anti-patterns:

- You store all data in the same AWS Region independent of where the data users are.
- You do not optimize data size and format before moving it over the network.

Benefits of establishing this best practice: Optimizing data movement across the network reduces the total networking resources required for the workload and lowers its environmental impact.

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

Implementation guidance

Moving data around your organization requires compute, networking, and storage resources. Use techniques to minimize data movement and improve the overall efficiency of your workload.

Implementation steps

- Consider proximity to the data or users as a decision factor when selecting a Region for your workload.
- Partition Regionally consumed services so that their Region-specific data is stored within the Region where it is consumed.
- Use efficient file formats (such as Parquet or ORC) and compress data before moving it over the network.
- Don't move unused data. Some examples that can help you avoid moving unused data:
  - Reduce API responses to only relevant data.
  - Aggregate data where detailed (record-level information is not required).
  - See Well-Architected Lab - Optimize Data Pattern Using Amazon Redshift Data Sharing.
  - Consider Cross-account data sharing in AWS Lake Formation.
- Use services that can help you run code closer to users of your workload.

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

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking
- AWS Global Infrastructure
- Amazon CloudFront Key Features including the CloudFront Global Edge Network
- Compressing HTTP requests in Amazon OpenSearch Service
- Intermediate data compression with Amazon EMR
- Loading compressed data files from Amazon S3 into Amazon Redshift
- Serving compressed files with Amazon CloudFront

Related videos:

- Demystifying data transfer on AWS

Related examples:

- Architecting for sustainability - Minimize data movement across networks

SUS04-BP08 Back up data only when difficult to recreate

Avoid backing up data that has no business value to minimize storage resources requirements for your workload.

Common anti-patterns:

- You do not have a backup strategy for your data.
- You back up data that can be easily recreated.

Benefits of establishing this best practice: Avoiding back-up of non-critical data reduces the required storage resources for the workload and lowers its environmental impact.

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

Implementation guidance

Avoiding the back up of unnecessary data can help lower cost and reduce the storage resources used by the workload. Only back up data that has business value or is needed to satisfy compliance requirements. Examine backup policies and exclude ephemeral storage that doesn't provide value in a recovery scenario.

Implementation steps

- Implement data classification policy as outlined in SUS04-BP01 Implement a data classification policy (p. 31).
- Use the criticality of your data classification and design backup strategy based on your recovery time objective (RTO) and recovery point objective (RPO). Avoid backing up non-critical data.
  - Exclude data that can be easily recreated.
  - Exclude ephemeral data from your backups.
  - Exclude local copies of data, unless the time required to restore that data from a common location exceeds your service-level agreements (SLAs).
- Use an automated solution or managed service to back up business-critical data.
  - AWS Backup is a fully-managed service that makes it easy to centralize and automate data protection across AWS services, in the cloud, and on premises. For hands-on guidance on how to create automated backups using AWS Backup, see Well-Architected Labs - Testing Backup and Restore of Data.
  - Automate backups and optimize backup costs for Amazon EFS using AWS Backup.

Resources

Related best practices:

- REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources
- REL09-BP03 Perform data backup automatically
- REL13-BP02 Use defined recovery strategies to meet the recovery objectives

Related documents:

- Using AWS Backup to back up and restore Amazon EFS file systems
- Amazon EBS snapshots
- Working with backups on Amazon Relational Database Service
- APN Partner: partners that can help with backup
- AWS Marketplace: products that can be used for backup
- Backing Up Amazon EFS
- Backing Up Amazon FSx for Windows File Server
- Backup and Restore for Amazon ElastiCache for Redis

Related videos:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS
- AWS Backup Demo: Cross-Account and Cross-Region Backup
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

Related examples:

- Well-Architected Lab - Testing Backup and Restore of Data
- Well-Architected Lab - Backup and Restore with Failback for Analytics Workload
- Well-Architected Lab - Disaster Recovery - Backup and Restore

Hardware and services

Look for opportunities to reduce workload sustainability impacts by making changes to your hardware management practices. Minimize the amount of hardware needed to provision and deploy, and select the most efficient hardware and services for your individual workload.

Best practices

- SUS05-BP01 Use the minimum amount of hardware to meet your needs (p. 43)
- SUS05-BP02 Use instance types with the least impact (p. 45)
- SUS05-BP03 Use managed services (p. 47)
- SUS05-BP04 Optimize your use of hardware-based compute accelerators (p. 48)

SUS05-BP01 Use the minimum amount of hardware to meet your needs

Use the minimum amount of hardware for your workload to efficiently meet your business needs.

Common anti-patterns:

- You do not monitor resource utilization.
- You have resources with a low utilization level in your architecture.
- You do not review the utilization of static hardware to determine if it should be resized.
- You do not set hardware utilization goals for your compute infrastructure based on business KPIs.

Benefits of establishing this best practice: Rightsizing your cloud resources helps to reduce a workload's environmental impact, save money, and maintain performance benchmarks.

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

Implementation guidance

Optimally select the total number of hardware required for your workload to improve its overall efficiency. The AWS Cloud provides the flexibility to expand or reduce the number of resources dynamically through a variety of mechanisms, such as AWS Auto Scaling, and meet changes in demand. It also provides APIs and SDKs that allow resources to be modified with minimal effort. Use these capabilities to make frequent changes to your workload implementations. Additionally, use rightsizing guidelines from AWS tools to efficiently operate your cloud resource and meet your business needs.

Implementation steps

- Choose the instances 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.
- Scale using small increments for variable workloads.
- Use multiple compute purchase options in order to balance instance flexibility, scalability, and cost savings.
  - On-Demand Instances are best suited for new, stateful, and spiky workloads which can't be instance type, location, or time flexible.
  - Spot Instances are a great way to supplement the other options for applications that are fault tolerant and flexible.
  - Leverage Compute Savings Plans for steady state workloads that allow flexibility if your needs (like AZ, Region, instance families, or instance types) change.
- Use instance and availability zone diversity to maximize application availability and take advantage of excess capacity when possible.
- Use the rightsizing recommendations from AWS tools to make adjustments on your workload.
  - AWS Compute Optimizer
  - AWS Trusted Advisor
- Negotiate service-level agreements (SLAs) that allow for a temporary reduction in capacity while automation deploys replacement resources.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute
- Attirbute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet
- AWS Compute Optimizer Documentation
- Operating Lambda: Performance optimization
- Auto Scaling Documentation

Related videos:

- Build a cost-, energy-, and resource-efficient compute environment

Related examples:

- Well-Architected Lab - Rightsizing with AWS Compute Optimizer and Memory Utilization Enabled (Level 200)

SUS05-BP02 Use instance types with the least impact

This best practice was updated with new guidance on July 13th, 2023.

Continually monitor and use new instance types to take advantage of energy efficiency improvements.

Common anti-patterns:

- You are only using one family of instances.
- You are only using x86 instances.
- You specify one instance type in your Amazon EC2 Auto Scaling configuration.
- You use AWS instances in a manner that they were not designed for (for example, you use compute-optimized instances for a memory-intensive workload).
- You do not evaluate new instance types regularly.
- You do not check recommendations from AWS rightsizing tools such as AWS Compute Optimizer.

Benefits of establishing this best practice: By using energy-efficient and right-sized instances, you are able to greatly reduce the environmental impact and cost of your workload.

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

Implementation guidance

Using efficient instances in cloud workload is crucial for lower resource usage and cost-effectiveness. Continually monitor the release of new instance types and take advantage of energy efficiency improvements, including those instance types designed to support specific workloads such as machine learning training and inference, and video transcoding.

Implementation steps

- Learn and explore instance types which can lower your workload environmental impact.
  - Subscribe to What's New with AWS to be up-to-date with the latest AWS technologies and instances.
  - Learn about different AWS instance types.
  - Learn about AWS Graviton-based instances which offer the best performance per watt of energy use in Amazon EC2 by watching re:Invent 2020 - Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances and Deep dive into AWS Graviton3 and Amazon EC2 C7g instances.
- Plan and transition your workload to instance types with the least impact.
  - Define a process to evaluate new features or instances for your workload. Take advantage of agility in the cloud to quickly test how new instance types can improve your workload environmental sustainability. Use proxy metrics to measure how many resources it takes you to complete a unit of work.
  - If possible, modify your workload to work with different numbers of vCPUs and different amounts of memory to maximize your choice of instance type.
  - Consider transitioning your workload to Graviton-based instances to improve the performance efficiency of your workload.

    - AWS Graviton Fast Start
    - Considerations when transitioning workloads to AWS Graviton-based Amazon Elastic Compute Cloud instances
    - AWS Graviton2 for ISVs
  - Consider selecting the AWS Graviton option in your usage of AWS managed services.
  - Migrate your workload to Regions that offer instances with the least sustainability impact and still meet your business 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 per watt over comparable Amazon EC2 instances.
  - Use Amazon SageMaker Inference Recommender to right size ML inference endpoint.
  - For spiky workloads (workloads with infrequent requirements for additional capacity), use burstable performance instances.
  - For stateless and fault-tolerant workloads, use Amazon EC2 Spot Instances to increase overall utilization of the cloud, and reduce the sustainability impact of unused resources.
- Operate and optimize your workload instance.
  - For ephemeral workloads, evaluate instance Amazon CloudWatch metrics such as `CPUUtilization` to identify if the instance is idle or under-utilized.
  - For stable workloads, check AWS rightsizing tools such as AWS Compute Optimizer at regular intervals to identify opportunities to optimize and right-size the instances.
    - Well-Architected Lab - Rightsizing Recommendations
    - Well-Architected Lab - Rightsizing with Compute Optimizer
    - Well-Architected Lab - Optimize Hardware Patterns and Observice Sustainability KPIs

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute
- AWS Graviton
- Amazon EC2 DL1
- Amazon EC2 Capacity Reservation Fleets
- Amazon EC2 Spot Fleet
- Functions: Lambda Function Configuration
- Attribute-based instance type selection for Amazon EC2 Fleet
- Building Sustainable, Efficient, and Cost-Optimized Applications on AWS
- How the Contino Sustainability Dashboard Helps Customers Optimize Their Carbon Footprint

Related videos:

- Deep dive on AWS Graviton2 processer-powered Amazon EC2 instances
- Deep dive into AWS Graviton3 and Amazon EC2 C7g instances
- Build a cost-, energy-, and resource-efficient compute environment

Related examples:

- Solution: Guidance for Optimizing Deep Learning Workloads for Sustainability on AWS
- Well-Architected Lab - Rightsizing Recommendations

- Well-Architected Lab - Rightsizing with Compute Optimizer
- Well-Architected Lab - Optimize Hardware Patterns and Observe Sustainability KPIs
- Well-Architected Lab - Migrating Services to Graviton

SUS05-BP03 Use managed services

Use managed services to operate more efficiently in the cloud.

Common anti-patterns:

- You use Amazon EC2 instances with low utilization to run your applications.
- Your in-house team only manages the workload, without time to focus on innovation or simplifications.
- You deploy and maintain technologies for tasks that can run more efficiently on managed services.

Benefits of establishing this best practice:

- Using managed services shifts the responsibility to AWS, which has insights across millions of customers that can help drive new innovations and efficiencies.
- Managed service distributes the environmental impact of the service across many users because of the multi-tenet control planes.

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

Implementation guidance

Managed services shift responsibility to AWS for maintaining high utilization and sustainability optimization of the deployed hardware. Managed services also remove the operational and administrative burden of maintaining a service, which allows your team to have more time and focus on innovation.

Review your workload to identify the components that can be replaced by AWS managed services. For example, Amazon RDS, Amazon Redshift, and Amazon ElastiCache provide a managed database service. Amazon Athena, Amazon EMR, and Amazon OpenSearch Service provide a managed analytics service.

Implementation steps

1. Inventory your workload for services and components.
2. Assess and identify components that can be replaced by managed services. Here are some examples of when you might consider using a managed service:

Task	What to use on AWS
Hosting a database	Use managed Amazon Relational Database Service (Amazon RDS) instances instead of maintaining your own Amazon RDS instances on Amazon Elastic Compute Cloud (Amazon EC2).
Hosting a container workload	Use AWS Fargate, instead of implementing your own container infrastructure.
Hosting web apps	Use AWS Amplify Hosting as fully managed CI/CD and hosting service for static websites and server-side rendered web apps.

3. Identify dependencies and create a migrations plan. Update runbooks and playbooks accordingly.
    - The AWS Application Discovery Service automatically collects and presents detailed information about application dependencies and utilization to help you make more informed decisions as you plan your migration
4. Test the service before migrating to the managed service.
5. Use the migration plan to replace self-hosted services with managed service.
6. Continually monitor the service after the migration is complete to make adjustments as required and optimize the service.

Resources

Related documents:

- AWS Cloud Products
- AWS Total Cost of Ownership (TCO) Calculator
- Amazon DocumentDB
- Amazon Elastic Kubernetes Service (EKS)
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Related videos:

- Cloud operations at scale with AWS Managed Services

SUS05-BP04 Optimize your use of hardware-based compute accelerators

Optimize your use of accelerated computing instances to reduce the physical infrastructure demands of your workload.

Common anti-patterns:

- You are not monitoring GPU usage.
- You are using a general-purpose instance for workload while a purpose-built instance can deliver higher performance, lower cost, and better performance per watt.
- You are using hardware-based compute accelerators for tasks where they're more efficient using CPU-based alternatives.

Benefits of establishing this best practice: By optimizing the use of hardware-based accelerators, you can reduce the physical-infrastructure demands of your workload.

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

Implementation guidance

If you require high processing capability, you can benefit from using accelerated computing instances, which provide access to hardware-based compute accelerators such as graphics processing units (GPUs) and field programmable gate arrays (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 minimize resources consumed.

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 per watt over comparable Amazon EC2 instances.
- Collect usage metric 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:

- Accelerated Computing
- Let's Architect! Architecting with custom chips and accelerators
- How do I choose the appropriate Amazon EC2 instance type for my workload?
- Amazon EC2 VT1 Instances
- Amazon Elastic Graphics
- 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
- Deep Dive on Amazon EC2 Elastic GPUs
- Deploying Cost-Effective Deep Learning Inference

Process and culture

Look for opportunities to reduce your sustainability impact by making changes to your development, test, and deployment practices.

Best practices

- SUS06-BP01 Adopt methods that can rapidly introduce sustainability improvements (p. 50)
- SUS06-BP02 Keep your workload up-to-date (p. 51)
- SUS06-BP03 Increase utilization of build environments (p. 52)
- SUS06-BP04 Use managed device farms for testing (p. 53)

SUS06-BP01 Adopt methods that can rapidly introduce sustainability improvements

Adopt methods and processes to validate potential improvements, minimize testing costs, and deliver small improvements.

Common anti-patterns:

- Reviewing your application for sustainability is a task done only once at the beginning of a project.
- Your workload has become stale, as the release process is too cumbersome to introduce minor changes for resource efficiency.
- You do not have mechanisms to improve your workload for sustainability.

Benefits of establishing this best practice: By establishing a process to introduce and track sustainability improvements, you will be able to continually adopt new features and capabilities, remove issues, and improve workload efficiency.

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

Implementation guidance

Test and validate potential sustainability improvements before deploying them to production. Account for the cost of testing when calculating potential future benefit of an improvement. Develop low cost testing methods to deliver small improvements.

Implementation steps

- Add requirements for sustainability improvement to your development backlog.
- Use an iterative improvement process to identify, evaluate, prioritize, test, and deploy these improvements.
- Continually improve and streamline your development processes. As an example, Automate your software delivery process using continuous integration and delivery (CI/CD) pipelines to test and deploy potential improvements to reduce the level of effort and limit errors caused by manual processes.
- Develop and test potential improvements using the minimum viable representative components to reduce the cost of testing.
- Continually assess the impact of improvements and make adjustment as needed.

Resources

Related documents:

- AWS enables sustainability solutions
- Scalable agile development practices based on AWS CodeCommit

Related videos:

- Delivering sustainable, high-performing architectures

Related examples:

- Well-Architected Lab - Turning cost & usage reports into efficiency reports

SUS06-BP02 Keep your workload up-to-date

Keep your workload up-to-date to adopt efficient features, remove issues, and improve the overall 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 keep your workload up to date, you can adopt new features and capabilities, resolve issues, and improve workload efficiency.

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

Implementation guidance

Up to date operating systems, runtimes, middlewares, libraries, and applications can improve workload efficiency and make it easier to adopt more efficient technologies. Up to date software might also include features to measure the sustainability impact of your workload more accurately, as vendors deliver features to meet their own sustainability goals. Adopt a regular cadence to keep your workload up to date with the latest features and releases.

Implementation steps

- Define a process and a schedule to evaluate new features or instances for your workload. Take advantage of agility in the cloud to quickly test how new features can improve your workload to:
  - Reduce sustainability impacts.
  - Gain performance efficiencies.
  - Remove barriers for a planned improvement.
  - Improve your ability to measure and manage sustainability impacts.
- Inventory your workload software and architecture and identify components that need to be updated.
  - 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.

Workload component	How to update
Machine images	Use EC2 Image Builder to manage updates to Amazon Machine Images (AMIs) for Linux or Windows server images.
Container images	Use Amazon Elastic Container Registry (Amazon ECR) with your existing pipeline to manage Amazon Elastic Container Service (Amazon ECS) images.
AWS Lambda	AWS Lambda includes version management features.

- 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.

Resources

Related documents:

- AWS Architecture Center
- What's New with AWS
- AWS Developer Tools

Related examples:

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

SUS06-BP03 Increase utilization of build environments

Increase the utilization of resources to develop, test, and build your workloads.

Common anti-patterns:

- You manually provision or terminate your build environments.
- You keep your build environments running independent of test, build, or release activities (for example, running an environment outside of the working hours of your development team members).
- You over-provision resources for your build environments.

Benefits of establishing this best practice: By increasing the utilization of build environments, you can improve the overall efficiency of your cloud workload while allocating the resources to builders to develop, test, and build efficiently.

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

Implementation guidance

Use automation and infrastructure-as-code to bring build environments up when needed and take them down when not used. A common pattern is to schedule periods of availability that coincide with the working hours of your development team members. Your test environments should closely resemble the production configuration. However, look for opportunities to use instance types with burst capacity, Amazon EC2 Spot Instances, automatic scaling database services, containers, and serverless technologies to align development and test capacity with use. Limit data volume to just meet the test requirements. If using production data in test, explore possibilities of sharing data from production and not moving data across.

Implementation steps

- Use infrastructure-as-code to provision your build environments.
- Use automation to manage the lifecycle of your development and test environments and maximize the efficiency of your build resources.

- Use strategies to maximize the utilization of development and test environments.
  - Use minimum viable representative environments to develop and test potential improvements.
  - Use serverless technologies if possible.
  - Use On-Demand Instances to supplement your developer devices.
  - Use instance types with burst capacity, Spot Instances, and other technologies to align build capacity with use.
  - Adopt native cloud services for secure instance shell access rather than deploying fleets of bastion hosts.
  - Automatically scale your build resources depending on your build jobs.

Resources

Related documents:

- AWS Systems Manager Session Manager
- Amazon EC2 Burstable performance instances
- What is AWS CloudFormation?
- What is AWS CodeBuild?
- Instance Scheduler on AWS

Related videos:

- Continuous Integration Best Practices

SUS06-BP04 Use managed device farms for testing

Use managed device farms to efficiently test a new feature on a representative set of hardware.

Common anti-patterns:

- You manually test and deploy your application on individual physical devices.
- You do not use app testing service to test and interact with your apps (for example, Android, iOS, and web apps) on real, physical devices.

Benefits of establishing this best practice: Using managed device farms for testing cloud-enabled applications provides a number of benefits:

- They include more efficient features to test application on wide range of devices.
- They eliminate the need for in-house infrastructure for testing.
- They offer diverse device types, including older and less popular hardware, which eliminates the need for unnecessary device upgrades.

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

Implementation guidance

Using Managed device farms can help you to streamline the testing process for new features on a representative set of hardware. Managed device farms offer diverse device types including older, less popular hardware, and avoid customer sustainability impact from unnecessary device upgrades.

Implementation steps

- Define your testing requirements and plan (like test type, operating systems, and test schedule).
  - You can use Amazon CloudWatch RUM to collect and analyze client-side data and shape your testing plan.
- Select the managed device farm that can support your testing requirements. For example, you can use AWS Device Farm to test and understand the impact of your changes on a representative set of hardware.
- Use continuous integration/continuous deployment (CI/CD) to schedule and run your tests.
  - Integrating AWS Device Farm with your CI/CD pipeline to run cross-browser Selenium tests
  - Building and testing iOS and iPadOS apps with AWS DevOps and mobile services
- Continually review your testing results and make necessary improvements.

Resources

Related documents:

- AWS Device Farm device list
- Viewing the CloudWatch RUM dashboard

Related examples:

- AWS Device Farm Sample App for Android
- AWS Device Farm Sample App for iOS
- Appium Web tests for AWS Device Farm

Related videos:

- Optimize applications through end user insights with Amazon CloudWatch RUM

Conclusion

An increasing number of organizations are setting sustainability targets in response to changes in government regulation, competitive advantage, and customer, employee, and investor demand. CTOs, architects, developers, and operations team members are seeking ways that they can directly contribute to their organization's sustainability goals. By using these design principles and best practices supported by AWS services, you can make informed decisions balancing security, cost, performance, reliability, and operational excellence with sustainability outcomes for your AWS Cloud workloads. Every action you take to reduce resource usage and increase efficiency across your workloads contributes to a reduction in environmental impact and contributes to your organizations' broader sustainability goals.

Contributors

Contributors to this document include:

- Sam Mokhtari, Sustainability Pillar Lead, Amazon Web Services
- Brendan Sisson, Principal Sustainability Solutions Architect, Amazon Web Services
- Margaret O'Toole, Sustainability Tech Leader, Amazon Web Services
- Steffen Grunwald, Principal Sustainability Solutions Architect, Amazon Web Services
- Ryan Eccles, Principal Engineer, Sustainability, Amazon
- Rodney Lester, Principal Architect, Amazon Web Services
- Adrian Cockcroft, VP Sustainability Architecture, Amazon Web Services
- Ian Meyers, Director of Technology, Solutions Architecture, Amazon Web Services
- Brian Carlson, Operational Excellence Lead, Amazon Web Services

Further reading

For additional information, refer to:

- AWS Well-Architected
- AWS Architecture Center
- Sustainability in the Cloud
- AWS enables sustainability solutions
- The Climate Pledge
- United Nations Sustainable Development Goals
- Greenhouse Gas Protocol

Document history

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

Change	Description	Date
Updated best practice guidance (p. 58)	Best practices were updated with new guidance in the following areas: Alignment to demand, Software and architecture, Data, and Hardware and services.	July 13, 2023
Updated for new Framework (p. 58)	Best practices updated with prescriptive guidance and new best practices added.	April 10, 2023
Whitepaper updated (p. 58)	Best practices updated with new implementation guidance.	December 15, 2022
Whitepaper updated (p. 58)	Best practices expanded and improvement plans added.	October 20, 2022
Initial publication (p. 58)	Sustainability Pillar - AWS Well-Architected Framework published.	December 2, 2021

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*.