Sustainability Pillar

AWS Well-Architected Framework

Sustainability Pillar: AWS Well-Architected Framework

Table of Contents

Sustainability Pillar - AWS Well-Architected Framework

Publication date: **October 20, 2022** (*Document history* (p. 35))

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.

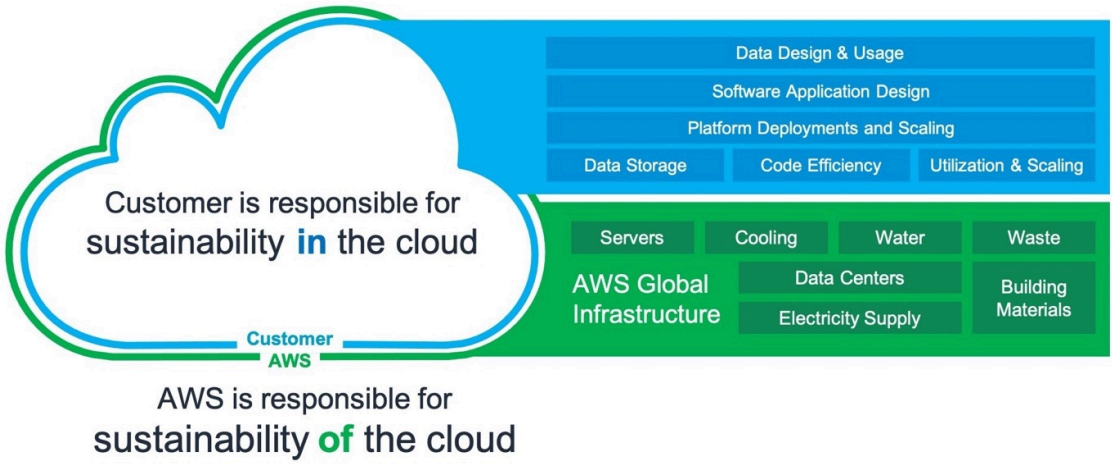
From an AWS customer perspective, emissions from your workloads running on AWS are accounted for as indirect emissions, and part of your Scope 3 emissions. Each workload deployed generates a fraction of the total AWS emissions from each of the previous scopes. The actual amount varies per workload and 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 scheduling 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 will 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 user, software, data, hardware, and development and deployment patterns 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

Choose Regions where you will implement your workloads based on both your business requirements and sustainability goals.

Best practices

SUS01-BP01 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)

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).

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

Implementation guidance

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).

Resources

Related documents:

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

User behavior patterns

The way users consume your workloads and other resources can help you identify improvements to meet sustainability goals. Scale infrastructure to continually match user load and ensure that only the minimum resources required to support users are deployed. Align service levels to customer needs. Position resources to limit the network required for users to consume them. Remove existing, unused assets. Identify created assets that are unused and stop generating them. Provide your team members with devices that support their needs with minimized sustainability impact.

Best practices

- SUS02-BP01 Scale infrastructure with user load (p. 14)
- SUS02-BP02 Align SLAs with sustainability goals (p. 15)
- SUS02-BP03 Stop the creation and maintenance of unused assets (p. 15)
- SUS02-BP04 Optimize geographic placement of workloads for user locations (p. 16)
- SUS02-BP05 Optimize team member resources for activities performed (p. 17)

SUS02-BP01 Scale infrastructure with user load

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

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

Implementation guidance

- Analyze the effect of users on load and capacity utilization over time and respond to changes in demand by scaling down the resources during periods of low utilization.
- Evaluate your workload for predictable patterns and proactively scale as you anticipate predicted and planned changes in demand.

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?
- What is AWS X-Ray?
- VPC Flow Logs

- Monitoring DB load with Performance Insights on Amazon RDS

Related videos:

- Building Sustainably on AWS
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)

Related examples:

- Amazon EC2 Auto Scaling Group Examples

SUS02-BP02 Align SLAs with sustainability goals

Define and update Service Level Agreements (SLAs) such as availability or data retention periods to minimize the number of resources required to support your workload while continuing to meet business requirements.

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

Implementation guidance

- Define SLAs that support your sustainability goals while meeting your business requirements.
- Redefine SLAs to meet business requirements, not exceed them.
- Make trade-offs that significantly reduce sustainability impacts in exchange for acceptable decreases in service levels.
- Use design patterns 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:

- Building Sustainably on AWS

SUS02-BP03 Stop the creation and maintenance of unused assets

Analyze application assets (such as pre-compiled reports, datasets, and static images) and asset access patterns to identify redundancy, underutilization, and potential decommission targets. Consolidate generated assets with redundant content (for example, monthly reports with overlapping or common datasets and outputs) to remove the resources consumed when duplicating outputs. Decommission unused assets (for example, images of products that are no longer sold) to free consumed resources and reduce the number of resources used to support the workload.

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

Implementation guidance

- Manage static assets and remove assets that are no longer required.
- Manage generated assets and stop generating and remove assets that are no longer required.
- Consolidate overlapping generated assets to remove redundant processing.
- 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.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part II: Storage

Related videos:

- Building Sustainably on AWS

SUS02-BP04 Optimize geographic placement of workloads for user locations

Analyze network access patterns to identify where your customers are connecting from geographically. Select Regions and services that reduce the distance network traffic must travel to decrease the total network resources required to support your workload.

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

Implementation guidance

- Use local caching for frequently-used resources.
- Use connection pooling to enable connection reuse and reduce required resources.
- Use edge caching to reduce the amount of data traversing the network from your origin server.
- 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:

- Building Sustainably on AWS

SUS02-BP05 Optimize team member resources for activities performed

Optimize resources provided to team members to minimize the sustainability impact while supporting their needs. For example, perform complex operations, such as rendering and compilation, on highly utilized shared cloud desktops instead of on underutilized high-powered single-user systems.

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

Implementation guidance

- 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.
- 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.

Resources

Related documents:

- What is Amazon WorkSpaces?
- Amazon AppStream 2.0 Documentation
- NICE DCV
- AWS Systems Manager Fleet Manager

Related videos:

- Building Sustainably on AWS

Software and architecture patterns

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 (p. 18)
- SUS03-BP02 Remove or refactor workload components with low or no use (p. 18)
- SUS03-BP03 Optimize areas of code that consume the most time or resources (p. 19)
- SUS03-BP04 Optimize impact on customer devices and equipment (p. 20)

- SUS03-BP05 Use software patterns and architectures that best support data access and storage patterns (p. 21)

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

Use efficient software designs and architectures to minimize the average resources required per unit of work. Implement mechanisms that result in even utilization of components to reduce resources that are idle between tasks and minimize the impact of load spikes.

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

Implementation guidance

- Queue requests that don't require immediate processing.
- Increase serialization to flatten utilization across your pipeline.
- Modify the capacity of individual components to prevent idling resources waiting for input.
- Create buffers and establish rate limiting to smooth the consumption of external services.
- Use the most efficient available hardware for your software optimizations.
- Use queue-driven architectures, pipeline management, and On-Demand Instance workers to maximize utilization for batch processing.
- Schedule tasks to avoid load spikes and resource contention from simultaneous execution.
- Schedule jobs during times of day where carbon intensity for power is lowest.

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:

- Building Sustainably on AWS
- Moving to event-driven architectures

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

Monitor workload activity to identify changes in utilization of individual components over time. Remove components that are unused and no longer required, and refactor components with little utilization to limit wasted resources.

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

Implementation guidance

- Analyze load (using indicators such as transaction flow and API calls) on functional components to identify unused and underutilized components.
- Retire components that are no longer needed.
- Refactor underutilized components.
- Consolidate underutilized components with other resources to improve utilization efficiency.

Resources

Related documents:

- What is AWS X-Ray?
- What is Amazon CloudWatch?
- Using ServiceLens to monitor the health of your applications
- Automated Cleanup of Unused Images in Amazon ECR

Related videos:

- Building Sustainably on AWS

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

Monitor workload activity to identify application components that consume the most resources. Optimize the code that runs within these components to minimize resource usage while maximizing performance.

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

Implementation guidance

- Monitor performance as a function of resource usage to identify components with high resource requirements per unit of work as targets for optimization.
- Use a code profiler to identify the areas of code that use the most time or resources as targets for optimization.
- Replace algorithms with more efficient versions that produce the same result.
- Use hardware acceleration to improve the efficiency of blocks of code with long execution times.
- Use the most efficient operating system and programming language for the workload.
- Remove unnecessary sorting and formatting.
- Use data transfer patterns that minimize the resources used based on how frequently the data changes and how it is consumed. For example, push state change information to a client instead of having it consume resources to poll and receive valueless ‘no change’ messages.

Resources

Related documents:

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

Related videos:

- Building Sustainably on AWS

SUS03-BP04 Optimize impact on customer devices and equipment

Understand the devices and equipment your customers use to consume your services, their expected lifecycle, and the financial and sustainability impact of replacing those components. Implement software patterns and architectures to minimize the need for customers to replace devices and upgrade equipment. For example, implement new features using code that is backward compatible with older hardware and operating system versions, or manage the size of payloads so they don't exceed the storage capacity of the target device.

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

Implementation guidance

- Inventory the devices your customers use.
- Test using managed device farms with representative sets of hardware to understand the impact of your changes, and iterate development to maximize the devices supported.
- Account for network bandwidth and latency when building payloads, and implement capabilities that help your applications work well on low-bandwidth, high-latency links.
- Pre-process data payloads to reduce local processing requirements and limit data transfer requirements.
- 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.

Resources

Related documents:

- What is AWS Device Farm?
- Amazon AppStream 2.0 Documentation
- NICE DCV
- Amazon Elastic Transcoder Documentation

Related videos:

- Building Sustainably on AWS

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. Select technologies to minimize data processing and storage requirements.

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

Implementation guidance

- Analyze your data access and storage patterns.
- Store data files in efficient file formats such as Parquet to prevent unnecessary processing (for example, when running analytics) and to reduce the total storage provisioned.
- Use technologies that work natively with compressed data.
- Use the database engine that best supports your dominant query pattern.
- Manage your database indexes to ensure index designs support efficient query execution.
- Select network protocols that reduce the amount of network capacity consumed.

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
- AWS IoT FleetWise

Related videos:

- Building Sustainably on AWS

Data patterns

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 (p. 22)
- SUS04-BP02 Use technologies that support data access and storage patterns (p. 22)
- SUS04-BP03 Use lifecycle policies to delete unnecessary data (p. 23)

SUS04-BP01 Implement a data classification policy

Classify data to understand its significance to business outcomes. Use this information to determine when you can move data to more energy-efficient storage or safely delete it.

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

Implementation guidance

- Determine requirements for the distribution, retention, and deletion of your data.
- Use tagging on volumes and objects to record the metadata that's used to determine how it's managed, including data classification.
- Periodically audit your environment for untagged and unclassified data, and classify and tag the data appropriately.

Resources

Related documents:

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

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

Use storage that best supports how your data is accessed and stored to minimize the resources provisioned while supporting your workload. For example, Solid State Devices (SSDs) are more energy intensive than magnetic drives and should be used only for active data use cases. Use energy-efficient, archival-class storage for infrequently accessed data.

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

Implementation guidance

- Monitor your data access patterns.
- Migrate data to the appropriate technology based on access pattern.
- Migrate archival data to storage designed for that purpose.

Resources

Related documents:

- Amazon EBS volume types
- Amazon EC2 instance store
- Amazon S3 Intelligent-Tiering
- Using Amazon S3 storage classes
- What is Amazon CloudWatch?
- What is Amazon S3 Glacier?

Related videos:

- Architectural Patterns for Data Lakes on AWS

SUS04-BP03 Use lifecycle policies to delete unnecessary data

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

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

Implementation guidance

- Define lifecycle policies for all your data classification types.
- Set automated lifecycle policies to enforce lifecycle rules.
- Delete unused volumes and snapshots.
- Aggregate data where applicable based on lifecycle rules.

Resources

Related documents:

- Amazon ECR Lifecycle policies
- Amazon EFS lifecycle management
- Amazon S3 Intelligent-Tiering
- Evaluating Resources with AWS Config Rules
- Managing your storage lifecycle on Amazon S3
- Object lifecycle policies in AWS Elemental MediaStore

Related videos:

- Amazon S3 Lifecycle

SUS04-BP04 Minimize over-provisioning in block storage

To minimize total provisioned storage, create block storage with size allocations that are appropriate for the workload. Use elastic volumes to expand storage as data grows without having to resize storage

attached to compute resources. Regularly review elastic volumes and shrink over-provisioned volumes to fit the current data size.

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

Implementation guidance

- Monitor the utilization of your data volumes.
- Use elastic volumes and managed block data services to automate allocation of additional storage as your persistent data grows.
- Set target levels of utilization for your data volumes, and resize volumes outside of expected ranges.
- 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.

Resources

Related documents:

- Amazon EBS Elastic Volumes
- Amazon FSx Documentation
- What is Amazon CloudWatch?
- What is Amazon Elastic File System?

SUS04-BP05 Remove unneeded or redundant data

Duplicate data only when necessary to minimize total storage consumed. Use backup technologies that deduplicate data at the file and block level. Limit the use of Redundant Array of Independent Drives (RAID) configurations except where required to meet Service Level Agreements (SLAs).

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

Implementation guidance

- Use mechanisms that can deduplicate data at the block and object level.
- Use backup technology that can make incremental backups and deduplicate data at the block, file, and object level.
- Use RAID only when required to meet your SLAs.
- 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 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:

- Amazon EBS snapshots
- 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 examples:

- Lab: Optimize Data Pattern Using Amazon Redshift Data Sharing

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

Adopt shared storage and single sources of truth to avoid data duplication and reduce the total storage requirements of your workload. Fetch data from shared storage only as needed. Detach unused volumes to make more resources available.

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

Implementation guidance

- Migrate data to shared storage when the data has multiple consumers.
- Fetch data from shared storage only as needed.
- Delete data as appropriate for your usage patterns, and implement time-to-live (TTL) functionality to manage cached data.
- Detach volumes from clients that are not actively using them.

Resources

Related documents:

- Amazon FSx
- Caching strategies
- What is Amazon Elastic File System?
- What is Amazon S3?

SUS04-BP07 Minimize data movement across networks

Use shared storage and access data from regional data stores to minimize the total networking resources required to support data movement for your workload.

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

Implementation guidance

- Store data as close to the consumer as possible.

- Partition regionally consumed services so that their Region-specific data is stored within the Region where it is consumed.
- Use block-level duplication instead of file or object-level duplication when copying changes across the network.
- Compress data before moving it over the network.

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

SUS04-BP08 Back up data only when difficult to recreate

To minimize storage consumption, 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.

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

Implementation guidance

- Use your data classification to establish what data needs to be backed up.
- Exclude data that you can easily recreate.
- 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).

Resources

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

Hardware patterns

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 for your individual workload.

Best practices

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

Using the capabilities of the cloud, you can make frequent changes to your workload implementations. Update deployed components as your needs change.

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

Implementation guidance

- Enable horizontal scaling, and use automation to scale out as loads increase and to scale in as loads decrease.
- Scale using small increments for variable workloads.
- Align scaling with cyclical utilization patterns (for example, a payroll system with intense bi-weekly processing activities) as load varies over days, weeks, months, or years.
- Negotiate service level Agreements (SLAs) that allow for a temporary reduction in capacity while automation deploys replacement resources.

Resources

Related documents:

- AWS Compute Optimizer Documentation
- Operating Lambda: Performance optimization
- Auto Scaling Documentation

SUS05-BP02 Use instance types with the least impact

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.

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

Implementation guidance

- Use the most efficient instance type compatible with your workload.
- Modify your workload to work with different numbers of CPUs and different amounts of memory to maximize your choice of instance type.
- Migrate your workload to Regions that offer instances with the least sustainability impact and that meet your service level agreements (SLAs).
- Use burstable instances to support workloads with infrequent requirements for additional capacity.

- Use Spot instances for stateless and fault-tolerant workloads to increase overall utilization of the cloud, and reduce the sustainability impact of unused resources.

Resources

Related documents:

- AWS Graviton Processor
- AWS Inferentia
- AWS Trainium
- Amazon EC2 Burstable performance instances
- Amazon EC2 Capacity Reservation Fleets
- Amazon EC2 Spot Fleet
- Amazon EC2 Spot Instances
- Amazon EC2 VT1 Instances
- Amazon EC2 instance types

Related videos:

- Deep dive into AWS Graviton3 and Amazon EC2 C7g instances

Related examples:

- Lab: Rightsizing Recommendations

SUS05-BP03 Use managed services

Managed services shift responsibility for maintaining high-average utilization, and sustainability optimization of the deployed hardware to AWS. Use managed services to distribute the sustainability impact of the service across all tenants of the service, reducing your individual contribution.

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

Implementation guidance

- Migrate from self-hosted services to managed services. For example, use managed Amazon Relational Database Service (Amazon RDS) instances instead of maintaining your own Amazon RDS instances on Amazon Elastic Compute Cloud (Amazon EC2), or use managed container services, such as AWS Fargate, instead of implementing your own container infrastructure.

Resources

Related documents:

- AWS Fargate
- Amazon DocumentDB
- Amazon Elastic Kubernetes Service (EKS)
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)
- Amazon Redshift

- Amazon Relational Database Service (RDS)

SUS05-BP04 Optimize your use of GPUs

Graphics Processing Units (GPUs) can be a source of high-power consumption, and many GPU workloads are highly variable, such as rendering, transcoding, and machine learning training and modeling. Only run GPU instances for the time needed, and decommission them with automation when not required to minimize resources consumed.

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

Implementation guidance

- Use GPUs only for tasks where they're more efficient than CPU-based alternatives.
- Use automation to release GPU instances when not in use.
- Use flexible graphics acceleration rather than dedicated GPU instances.
- Take advantage of custom-purpose hardware that is specific to your workload.

Resources

Related documents:

- Accelerated Computing
- AWS Inferentia
- AWS Trainium
- Accelerated Computing for EC2 Instances
- Amazon EC2 VT1 Instances
- Amazon Elastic Graphics

Development and deployment process

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. 29)
- SUS06-BP02 Keep your workload up-to-date (p. 30)
- SUS06-BP03 Increase utilization of build environments (p. 30)
- SUS06-BP04 Use managed device farms for testing (p. 31)

SUS06-BP01 Adopt methods that can rapidly introduce sustainability improvements

Test and validate potential 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 enable delivery of small improvements.

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

Implementation guidance

- Add requirements for sustainability to your development process.
- Allow resources to work in parallel to develop, test, and deploy sustainability improvements.
- Test and validate potential sustainability impact improvements before deploying into production.
- Test potential improvements using the minimum viable representative components.
- Deploy tested sustainability improvements to production as they become available.

Resources

Related documents:

- AWS enables sustainability solutions

Related examples:

- Lab: Turning cost & usage reports into efficiency reports

SUS06-BP02 Keep your workload up-to-date

Up-to-date operating systems, libraries, and applications can improve workload efficiency and enable easier adoption of 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.

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

Implementation guidance

- Update systems to gain performance efficiencies.
- Update systems to remove barriers for a planned improvement.
- Update systems to acquire software features to reduce sustainability impacts.
- Update systems to improve your ability to measure and manage sustainability impacts.

Resources

Related documents:

- AWS Architecture Center
- What's New with AWS

SUS06-BP03 Increase utilization of build environments

Use automation and infrastructure-as-code to bring pre-production 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. Hibernation is a useful tool to preserve the state and rapidly bring instances online only when needed. Use instance types with burst capacity, Spot

Instances, elastic database services, containers, and other technologies to align development and test capacity with use.

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

Implementation guidance

- Use automation to maximize utilization of your development and test environments.
- Use automation to manage the lifecycle of your development and test environments.
- Use minimum viable representative environments to develop and test potential improvements.
- Use On-Demand Instances to supplement your developer devices.
- Use automation to maximize the efficiency of your build resources.
- 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.

Resources

Related documents:

- AWS Systems Manager Session Manager
- Amazon EC2 Burstable performance instances
- What is AWS CloudFormation?

SUS06-BP04 Use managed device farms for testing

Managed device farms spread the sustainability impact of hardware manufacturing and resource usage across multiple tenants. Managed device farms offer diverse device types so you can support older, less popular hardware, and avoid customer sustainability impact from unnecessary device upgrades.

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

Implementation guidance

Test using managed device farms with representative sets of hardware to understand the impact of your changes, and iterate development to maximize the devices supported.

Resources

Related documents:

- What is AWS Device Farm?

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
Whitepaper updated (p. 35)	Best practices expanded and improvement plans added.	October 20, 2022
Initial publication (p. 35)	Sustainability Pillar - AWS Well-Architected Framework published.	December 2, 2021

Notices

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.