Optimize costs for Microsoft workloads on AWS

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Optimize costs for Microsoft workloads on AWS

Table of Contents

Optimize costs for Microsoft workloads on AWS

Bill Pfeiffer, Chase Lindeman, and Kevin Sookhan, Amazon Web Services (AWS)

October 2025 (document history)

Overview

This guide provides recommendations, best practices, and strategies to help optimize costs for your Microsoft workloads on AWS. The guide also includes foundational AWS knowledge, cost optimization techniques, and reference architectures to help you build and automate cost-effective, high-performing workloads that meet your business objectives. Collectively, this guidance is referred to as *Microsoft on AWS Cost Optimization (MACO)*. MACO guidance was developed by industry experts and is based on real-world scenarios.

This guide covers the following Microsoft workloads:

- Windows on Amazon Elastic Compute Cloud (Amazon EC2)
- SQL Server
- Containers
- Storage
- Active Directory
- .NET

Audience

This guide is intended for architects, engineers, administrators, directors, CTOs, technical decision makers, and AWS Partners. It's helpful but not necessary to have prior experience with and a basic understanding of AWS billing, Microsoft technologies, and AWS systems administration.

How to use this guide

You can use this guide to plan and implement your MACO journey to the cloud. We recommend that you read this guide from start to finish to get a comprehensive understanding of the options and approaches for optimizing the costs of your Microsoft workloads on AWS. You can review the following workload sections based on your organization's needs:

- Windows on Amazon EC2
- SQL Server
- Containers
- Storage
- Active Directory
- .NET

> **Important**
>
> The code samples provided in this guide are for demonstration purposes only. It's a best practice to test all code in a development environment prior to using it in a production environment. Before you implement any code, we recommend that you test your code in small batches and then review the cost changes that result from the code by using AWS Cost Explorer. This can help you troubleshoot edge cases and other issues that can become problematic later on.

> **Important**
>
> The pricing examples in this guide are based on prices at the time of publication. Prices are subject to change. Additionally, your costs may vary depending on your AWS Region, AWS service quotas, and other factors related to your cloud environment.

Targeted business outcomes

This guide can help you and your organization achieve the following business outcomes:

- Learn how to use an AWS Optimization and Licensing Assessment (AWS OLA) to assess and optimize your current on-premises and cloud environments, based on resource utilization, third-party licensing, and application dependencies.
- Develop a business case for cost optimization by using the AWS Modernization Calculator for Microsoft Workloads.
- Optimize costs for your specific Microsoft workloads, including workloads for Windows on Amazon Elastic Compute Cloud (Amazon EC2), SQL Server, containers, storage, Active Directory, and .NET.

Cost optimization journey

The scope, timing, and specific path of your cloud migration journey depends on your business objectives, technical requirements, and other factors. This section provides an example of a cloud migration journey that's focused on Cloud Financial Management with AWS and adheres to MACO recommendations and best practices. You can use this example to gain an understanding of how to design a cloud migration journey for Microsoft workloads.

The following high-level tasks illustrate the approach that an organization could take to implement MACO recommendations and best practices:

- Establish a tagging strategy and enable user-defined cost allocation tags. For more information, see the AWS Whitepaper Best Practices for Tagging AWS Resources.
- Define budgets based on applications, teams, or departments. For more information, see Managing your costs with AWS Budgets in the *AWS Billing and Cost Management User Guide*.
- Perform an AWS Optimization and Licensing Assessment (AWS OLA) to accelerate savings. For more information, see AWS Optimization and Licensing Assessment in the AWS documentation.
- Bring Your Own License (BYOL) for Windows and SQL Server workloads by using Amazon Elastic Compute Cloud Dedicated Hosts. For more information, see the Bring licenses for Windows and SQL Server workloads section of this guide.
- Optimize your SQL Server licensing on AWS. For more information, see the Understand SQL Server licensing section of this guide.
- Select the right instance type for Windows workloads. For more information, see the Select the right instance type for Windows workloads section of this guide.
- Select the right instance type for SQL workloads. For more information, see the Select the right EC2 instance for SQL Server workloads section of this guide.
- Migrate Amazon Elastic Block Store (Amazon EBS) from gp2 to gp3. For more information, see the Migrate Amazon EBS volumes from gp2 to gp3 section of this guide.
- Control workloads with EC2 Instance Scheduler on AWS. For more information, see the Automate stop and start schedules section of this guide.
- Remove SQL Server costs for non-production workloads by using SQL Server Developer Edition. For more information, see the Evaluate SQL Server Developer edition section of this guide.
- Use a single Availability Zone for Amazon FSx for Windows File Server for development and testing workloads. For more information, see the Use a single Availability Zone section of this guide.

- Rightsize your Windows workloads by using AWS Compute Optimizer. For more information, see the Right size Windows workloads section of this guide.
- Optimize spending on Windows on Amazon EC2 by using Savings Plans. For more information, see the Optimize spending for Windows on Amazon EC2 section of this guide.
- Enable data deduplication on FSx for Windows File Server. For more information, see the Enable data deduplication in Amazon FSx section of this guide.
- Use data sharding for file systems on FSx for Windows File Server. For more information, see the Understand data sharding in FSx for Windows File Server section of this guide.
- Optimize your SQL Server backup strategies. For more information, see the Optimize SQL Server backup strategies section of this guide.
- Make static .NET framework apps support dynamic scaling. For more information, see the Support dynamic scaling for static .NET Framework apps of this guide.
- Use serverless .NET microservices. For more information, see the Consider serverless .NET section of this guide.
- Move your Windows apps to containers. For more information, see the Containerize .NET apps section of this guide.
- Use AWS Compute Optimizer to rightsize Windows containers running on AWS Fargate for Amazon Elastic Container Service (Amazon ECS). For more information, see the Enable Compute Optimizer section of this guide.
- Refactor to modern .NET and move to Linux. For more information, see the Refactor to modern .NET and move to Linux section of this guide.
- Leverage Graviton instances and containers. For more information, see the Use Graviton instances and containers section of this guide.
- Modernize SQL Server databases. For more information, see the Modernize SQL Server databases section of this guide.
- Design Active Directory infrastructure. For more information, see the Active Directory section of this guide.

For more information about a customer journey focused on Cloud Financial Management with AWS, see the AWS Whitepaper Cloud Financial Management capability.

Top recommendations for optimizing costs

Overview

Cost optimization is one of the pillars of the AWS Well-Architected Framework and it plays a critical role in your cloud migration plans. You will find recommendations for cost optimizations throughout this guide, but this section calls out the highest-impact recommendations. You can implement these recommendations quickly and they will have a significant impact on your organization. These recommendations can help lay the groundwork for your entire cost optimization effort.

Top recommendations

The following table lists the top recommendations for the highest-impact cost optimizations. The "Difficultly to implement" column rates each optimization based on a scale of what's easiest to implement (1) to what's most difficult to implement (5). The "Estimated savings" column shows a percentage-based estimate of how much your organization can save for each recommended optimization.

Optimizations	Difficulty to implement	Estimated savings
Right size Windows workloads	3	25%
Bring licenses for Windows and SQL Server workloads	3	30%
Evaluate SQL Server Developer edition	2	20%
Understand SQL Server licensing	2	Up to 50%
Automate stop and start schedules	3	Up to 40%
Select the right instance type for Windows workloads	1	10–30%

Optimizations	Difficulty to implement	Estimated savings
Refactor to modern .NET and move to Linux	5	10–20%
Optimize spending for Windows on Amazon EC2	3	Up to 20–40%
Migrate Amazon EBS volumes from gp2 to gp3	4	Up to 20%

Important

The estimated savings in the preceding table apply to each individual technical domain, not overall AWS spend within an account. For example, you can implement the Instance Scheduler in a variety of environment types and sizes that can alter the potential savings. The estimates apply specifically to Amazon EC2 instance costs and don't imply any overall savings for other AWS services. These estimates are provided as a gauge, not a guarantee.

MACO experts are available to talk about cost optimizations in more depth. To set up a meeting for a deep dive into your use case, contact your account team or email optimize-microsoft@amazon.com.

AWS Optimization and Licensing Assessment

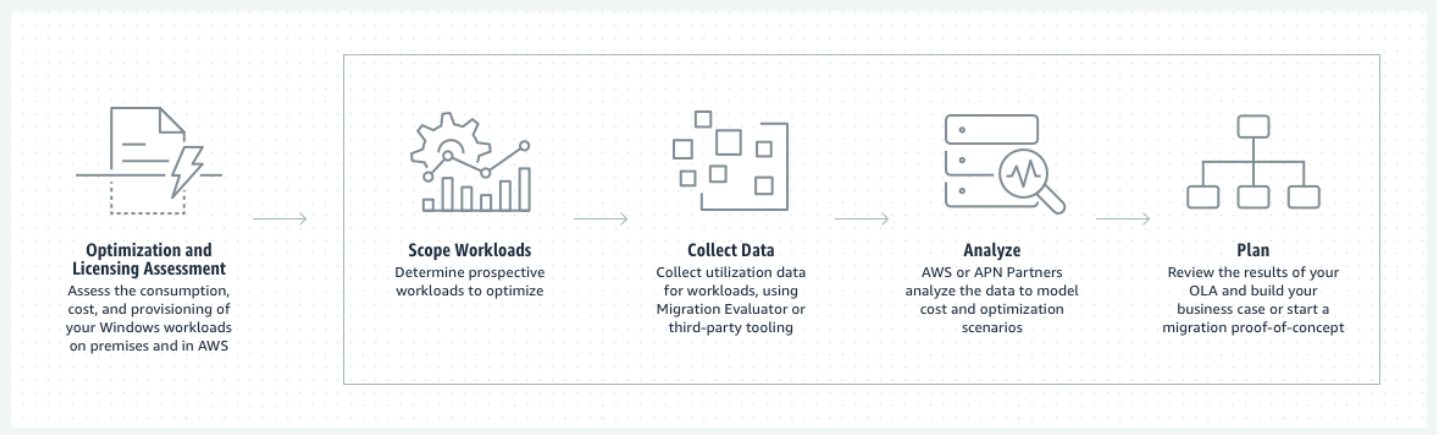
Overview

An AWS Optimization and Licensing Assessment (AWS OLA) can help you assess and optimize your current on-premises and existing cloud environments, based on resource utilization, third-party licensing, and application dependencies. You can use AWS OLA to help your organization build a migration and licensing strategy that unlocks cost savings as you migrate to AWS or assess existing Microsoft workloads on AWS. An AWS OLA can also help you achieve the following:

- Understand existing deployments, application performance, and contracts.
- Right size your resources.
- Develop a roadmap to the AWS Cloud.
- Reduce or eliminate costs by using existing investments and paying only for what you use.

We recommend that you make an AWS OLA the first step on your cost optimization journey. You can work with the AWS Partner Network to complete an AWS OLA. They will help you gather assessment data and provide you with recommendations for optimizing your licensing and instance costs.

The following diagram provides an overview of the assessment process.



Assessment options

You can choose from two AWS OLA options for your Microsoft workloads on AWS:

- **Lite version** – In this use case, all of your workloads are on VMware. You can provide AWS with an output from RVTools. Then, AWS can offer a turnaround time of 1–5 days. This approach uses point-in-time information pulled directly from VMware vCenter to develop sizing recommendations and offer on-demand pricing options.
- **Full version** – In this use case, you have a mixed environment running in different cloud providers, physical servers, and virtual servers. AWS uses operating system agents to collect usage data from 14 to 30 days. This allows AWS to make informed instance sizing decisions based on your application usage patterns. AWS uses several third-party tools, such as Cloudamize, to complete the analysis. AWS works with its AWS Partner Network to help deliver the final total cost of ownership (TCO) assessment with multiple pricing options that factor in pricing models and different architectures.

Full assessment

The full AWS OLA assessment is kicked off with a one-hour phone call. During this call, AWS helps you to determine the most optimal AWS infrastructure to support your migration, choose a data collection method, and establish a timeline for completion. Implementing discovery tooling in your organization depends on the data collection method, the size of your organization, and the tooling that your organization uses to manage its fleet of servers. It typically takes two weeks to collect usage data.

The full AWS OLA process takes between 30–45 days and consists of the following phases:

- Scope workloads
- Collect data
- Analyze data
- Plan next steps

Scope workloads

First, AWS works with you and your team to determine the scope of the assessment. This is usually broken down by environment type (for example, non-production and production). The scope includes the location of the workloads. This could be workloads that you're migrating to AWS, workloads that are already running on AWS (for example, AWS OLA for Amazon EC2), or workloads running in other cloud providers.

Collect data

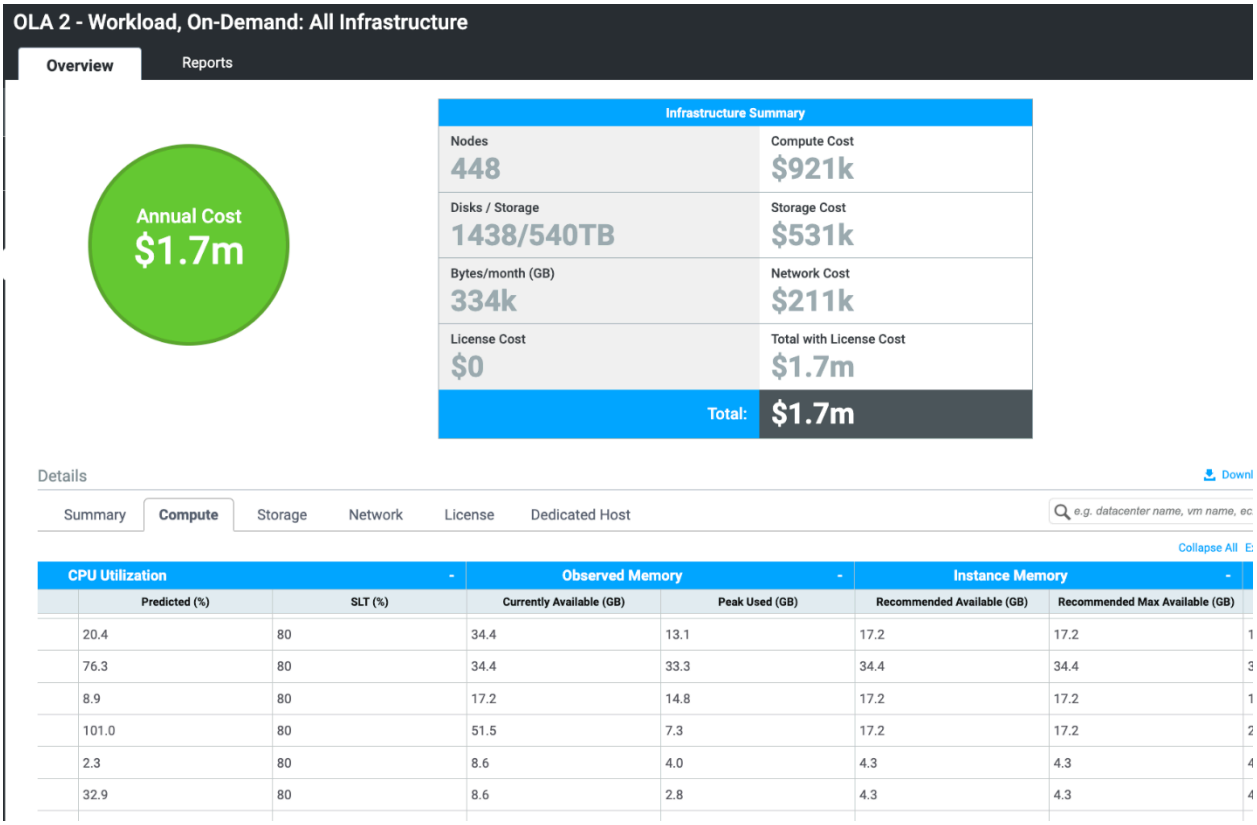
Next, AWS deploys tooling to help with resource discovery and collect performance data from your servers. This tooling comes in four deployment options:

- Tools that can query the hypervisor (requires only VMware vCenter or Hyper-V credentials)
- Agents that can be deployed on physical or virtual machines
- Agentless discovery by using SSH, Windows Remote Management (WinRM), or Windows Management Instrumentation (WMI) depending on your environment and operating system
- Flat file data collection and analysis

For your tooling deployment, you can mix and match each option and consolidate results. It's crucial to ensure that whatever option you choose doesn't strain your IT resources. AWS strives to make the assessment process as turnkey as possible. Beyond a brief phone call to assist with setup, the AWS OLA team and Microsoft specialist solutions architects will prep the total cost of ownership (TCO) analysis and recommendations for review.

Data collection usually takes two to three weeks when CPU utilization, RAM utilization, storage throughput, IOPS, and network throughput are analyzed. Ideally, this collection takes place during the peak times of your business month (for example, during end-of-month financial reporting). AWS wants to capture the peak usage because this gives good statistical samples for what the right-sized AWS instance should be, while still guaranteeing the performance can exceed what's available on premises. AWS merges utilization metrics with performance heuristics of various processor generations to target exactly how much CPU and RAM a given workload requires. These targets are usually less than what's allocated on premises. This not only reduces the compute cost for size of the instance but also optimizes licensing costs.

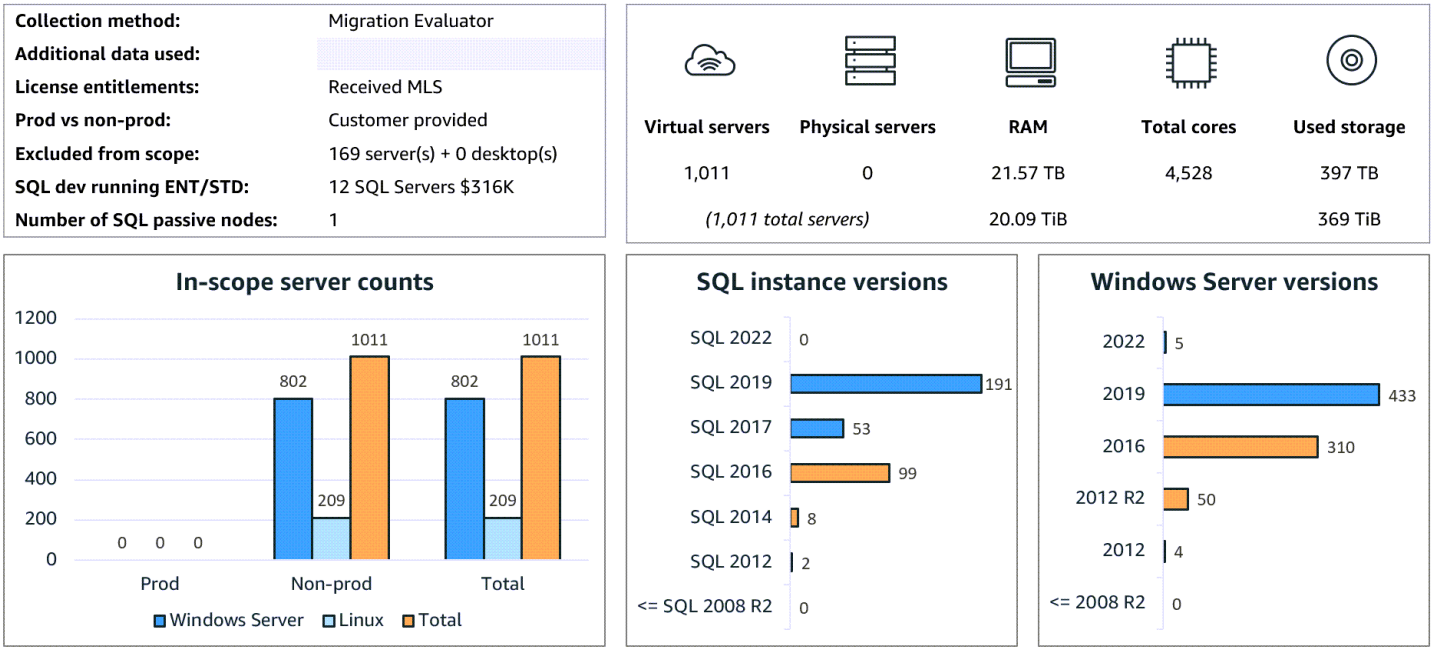
The following dashboard view shows an example of infrastructure costs that can be captured by an assessment.



Analyze data

AWS delivers a debrief presentation after the data collection is completed. AWS reviews the data, summarizes findings, and then makes recommendations for on-premises usage and cloud migration. You can reduce compute and licensing costs by examining consolidation opportunities, elasticity gains (where workloads can be turned off or seasonally adjusted), right-SKU opportunities (for example, SQL Server Enterprise edition is in use, but resource requirements and feature usage suggest SQL Server Standard edition is suitable). For products like SQL Server which are licensed by the core, it often makes financial sense to place workloads in a more expensive compute instance. That is, if the CPU profile and ratio of RAM to vCPU serves a net effect to reduce the number of licensed cores for both license-included and Bring Your Own License (BYOL) use cases.

The following shows an example analysis based on the data collected by the assessment.



Common optimization scenarios include identifying both AWS resource optimization opportunities and third-party license savings.

Examples of AWS resource optimization opportunities:

- Avoid overprovisioning for peak usage.
- Avoid over-specifying and underusing resources.
- Right size your instances and migrate to the newest generations of EC2 instances.
- Save on operations costs by moving to managed databases.

Examples of third-party license savings:

- Reduce required cores to run the same workload.
- Get rid of unnecessary SQL Server Enterprise edition and add-on packs.
- Remove zombie servers and replace outdated hardware.
- Use BYOL and license-included options to reduce future commercial agreements.
- Modernize to open-source and cloud-native solutions.

Plan next steps

Finally, AWS uses the collected performance data to estimate specific workload sizing and cost. AWS can also look in aggregate at your scoped environment and provide a quantitative analysis. This can help you determine if the best option is an on-premises refresh or a migration to AWS. You can build a cloud economic business case by using the TCO analysis summary (as shown in the following example) provided at the end of an AWS OLA.

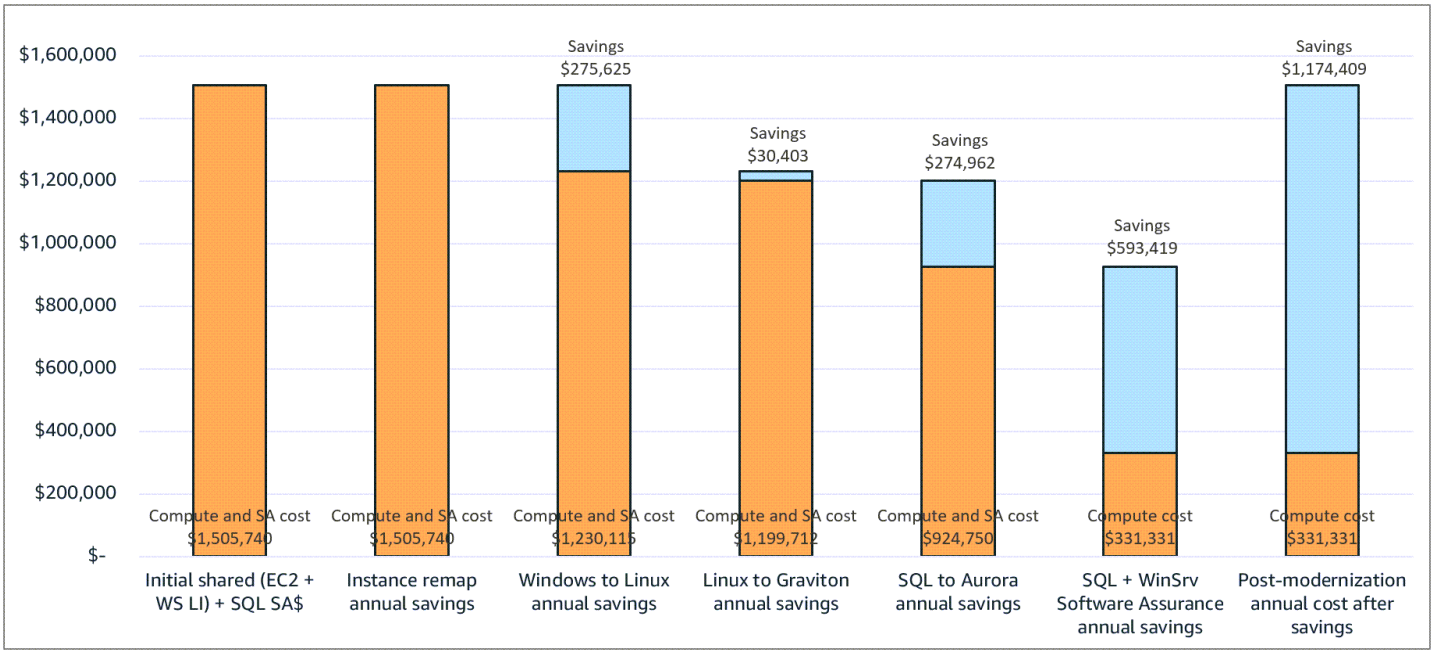
	Option 1: Amazon EC2 shared	**Option 1a: Amazon EC2 shared + power management**	**Option 2: Amazon EC2 mixed**	**Option 2a: Amazon EC2 mixed + power management**
Option details: compute	*100% Reserved Instances (RIs)*	*RIs + on-demand power management*	*100% Ris*	*RIs + on-demand power management*
Option details: Microsoft licenses	*WS LI and SQL BYOL*	*WS LI and SQL BYOL*	*WS BYOL or LI+SQL BYOL*	*WS BYOL or LI+SQL BYOL*
Compute costs[1]				
Year 1 compute cost	$414,546	$482,623	$504,019	$513,941
Year 1 vendor license included cost	$392,858	$244,415	$9,804	$4,783
	$807,404	**$727,038**	**$513,823**	**$518,724**
Total compute savings in year 1, compared to Option 1	—	*10% ($80,366)*	*36% ($293,581)*	*36% ($288,680)*
Storage and networking costs[2]				
Annual estimated storage cost	$336,494	$336,494	$336,494	$336,494
Annual estimated networking cost	$41,455	$41,455	$41,455	$41,455
	$377,949	**$377,949**	**$377,949**	**$377,949**
Microsoft license costs**				
WS/CIS annual Software Assurance (SA) + current SPLA/Subs cost	$0	$0	$0	$0
WS/CIS license + SA + SPLA/Subs true-up cost	$0	$0	$0	$0
SQL annual SA + current SPLA/Subs cost	$0	$0	$0	$0
SQL license SA + current SPLA/Subs true-up cost	$0	$0	$0	$0
	$0	**$0**	**$0**	**$0**
Total estimated costs	**$1,185,353**	**$1,104,987**	**$891,772**	**$896,673**
Annual TCO savings in year 1, compared to Option 1	—	*7% ($80,366)*	*25% ($293,581)*	*24% ($288,680)*

[1] Pricing model used: 3-year, no upfront RI
[2] Software Assurance and true-up costs provided by Microsoft

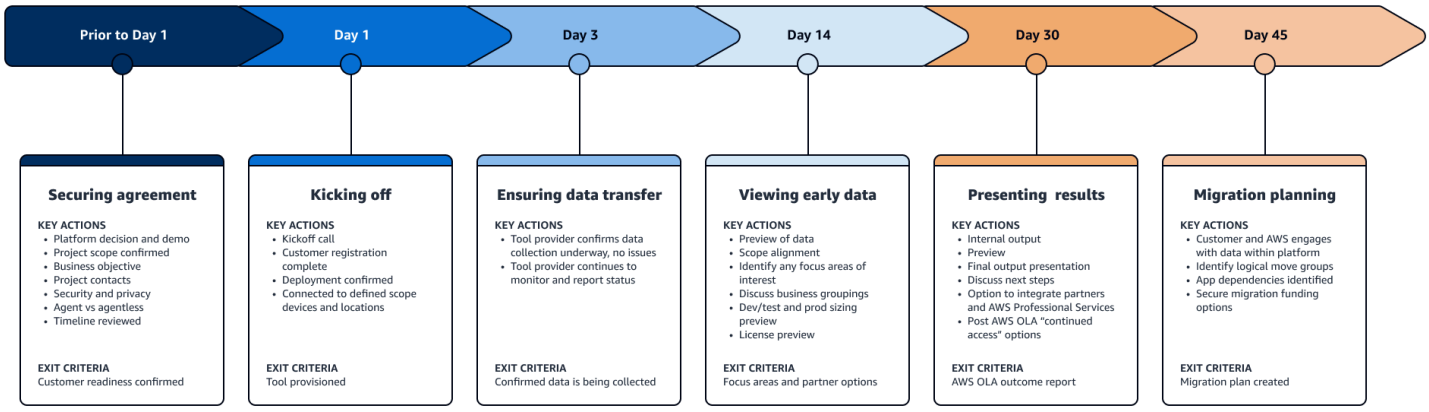
An AWS OLA also provides insight into the impact that modernization can have on your existing workloads by making suggestions such as the following:

- Move to a Linux operating system.
- Add application support for ARM processors (AWS Graviton).
- Move SQL Server workloads to Amazon Aurora.
- Remove software assurance by moving Windows and SQL Server workloads to open-source technologies.

The following diagram shows the cost savings that can be achieved through modernization techniques such as moving from Windows to Linux or from SQL Server to Aurora.



The full AWS OLA process takes roughly 45 days from start to finish. The following diagram shows an example timeline.



If you have a pure VMware environment and can provide output from RVTools, then you can reduce this timeline to one business week. Additionally, AWS can analyze a flat file that includes asset and utilization data, such as CPU average, CPU peak, RAM average, and RAM peak.

Assessment impact

The average customer typically experiences a 20–30 percent cost reduction from the right-sizing effort. Right sizing matches the source workload to the best sized AWS instances based on usage data. These right-sizing adjustments not only reduce the monthly cost of the AWS environment, but frequently result in savings elsewhere in the organization. For example, a 20–30 percent gain

of Windows or SQL Server licensing can reduce the next true-up with Microsoft, or free up licensing for additional line-of-business applications. Consolidation and right sizing of SQL Server workloads is commonly where the most dramatic financial gains are realized.

AWS can help you categorize systems into modernization buckets. Some systems are legacy and are not financially viable to touch, while others may be modernized into containers or serverless applications where the most significant savings are realized. The conversation with your AWS team moves from generalized topics of what the cloud enables to more specific discussions of how and why specific workloads should be modernized. AWS also helps you explore potential innovation opportunities.

Next steps

If you're beginning your cost optimization journey for Microsoft workloads that are running in on-premises environments or on AWS, engage with your AWS account team and request an AWS OLA. AWS team members can answer your questions and help you decide if an AWS OLA is ultimately the right choice for you and your organization. Alternatively, you can request an AWS OLA online.

Additional resources

- AWS Optimization and Licensing Assessment (AWS documentation)
- AWS re:Invent 2022 - How to save costs and optimize Microsoft workloads on AWS (ENT205) (YouTube)

Windows on Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a highly flexible and scalable cloud computing platform that's ideal for running your Windows workloads. You can use Amazon EC2 to deploy, manage, and scale your Windows Server workloads on the secure, reliable, highly available, and adaptable infrastructure of the AWS Cloud. Consider the following key benefits of running Windows workloads on Amazon EC2:

- **Scalability** – Amazon EC2 enables you to easily scale your Windows workloads to accommodate changing requirements. You can quickly create new EC2 instances to handle increased demand, and just as easily terminate the instances when they're no longer needed. You only pay for the resources that you actually use.
- **Flexibility** – Windows on Amazon EC2 supports a wide range of instance types that are designed to cater to various workload requirements, from general-purpose instances to memory or compute-optimized instances. This flexibility ensures that you can choose the best instance type for your specific Windows-based applications, maximizing performance and minimizing costs.
- **Security** – AWS provides multiple layers of security for your Windows workloads, including network firewalls, data encryption, and secure access controls. This means you can trust that your applications and data are protected, while still having complete control over your security settings and configurations.
- **Cost efficiency** – The pay-as-you-go pricing model enables you to pay only for the resources that you use—eliminating the need for upfront investments in hardware and software. This model also enables you to optimize your costs, reduce capital expenditures, and increase operational efficiency. It's an ideal pricing model for businesses of all sizes.

This section of the guide covers the following topics:

Automate stop and start schedules

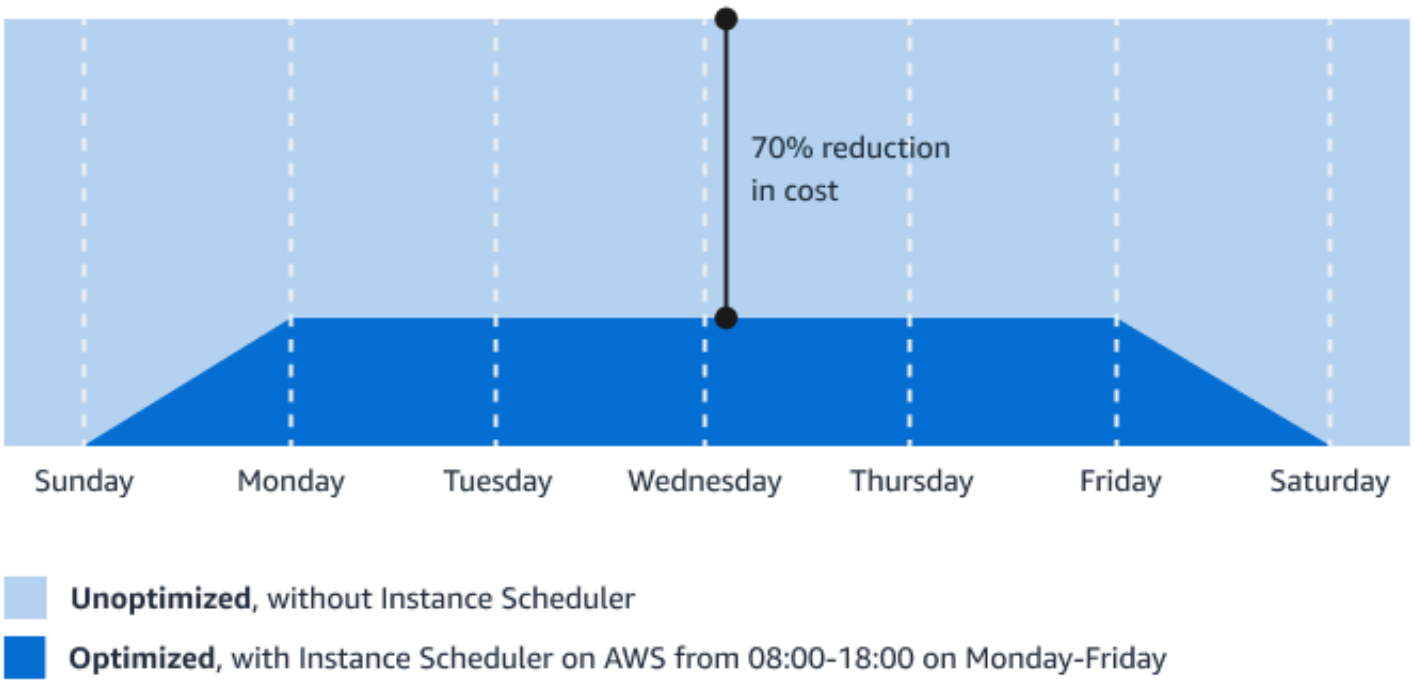
Overview

The Instance Scheduler on AWS can help you reduce operational costs by automating the starting and stopping of Amazon EC2 and Amazon Relational Database Service (Amazon RDS) instances. If you leave all your instances running at full utilization continuously, you could end up paying for resources that aren't being used. The Instance Scheduler on AWS enables you to turn off instances during times when they're not needed, such as during non-business hours, weekends, or other periods when usage is low. This can lead to significant cost savings over time.

The Instance Scheduler on AWS also offers cross-account instance scheduling, automated tagging, and the ability to configure schedules or periods by using a command-line interface or the AWS Systems Manager maintenance window. These features can help you manage your instances more effectively and accurately track and allocate costs across different projects or teams.

Case studies

Consider the example of a company that uses Instance Scheduler on AWS in a production environment to automatically stop instances outside of business hours every day. If this company leaves all of its instances running at full utilization, they can achieve up to 70 percent cost savings for those instances that are only necessary during regular business hours. The following chart shows how the weekly utilization is reduced from 168 hours to 50 hours.

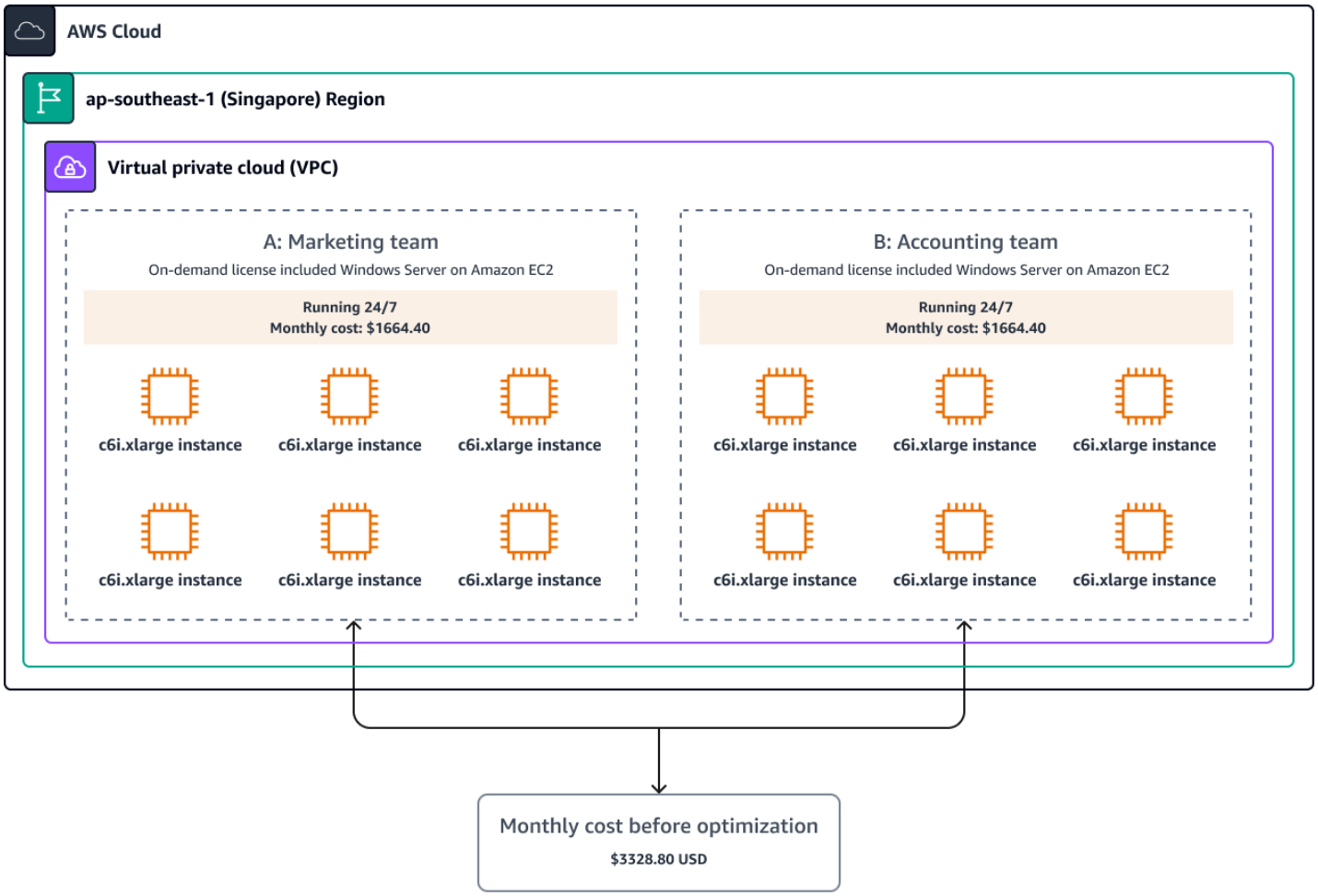


Consider another example. The electric utility company Jamaica Public Service Company Limited (JPS) migrated its database to Amazon RDS. Now, JPS uses Amazon EC2 to host API services and run other applications. For JPS, Instance Scheduler on AWS became the key tool for managing non-production environments. JPS used the Instance Scheduler on AWS to reduce development costs and manage EC2 instances based on team needs and work schedules. This helped JPS reduce costs by 40 percent. For more information, see the AWS case study Jamaica Public Service Migrates Efficiently to the Cloud, Reduces Costs by 40% Using AWS Instance Scheduler.

Cost optimization scenario

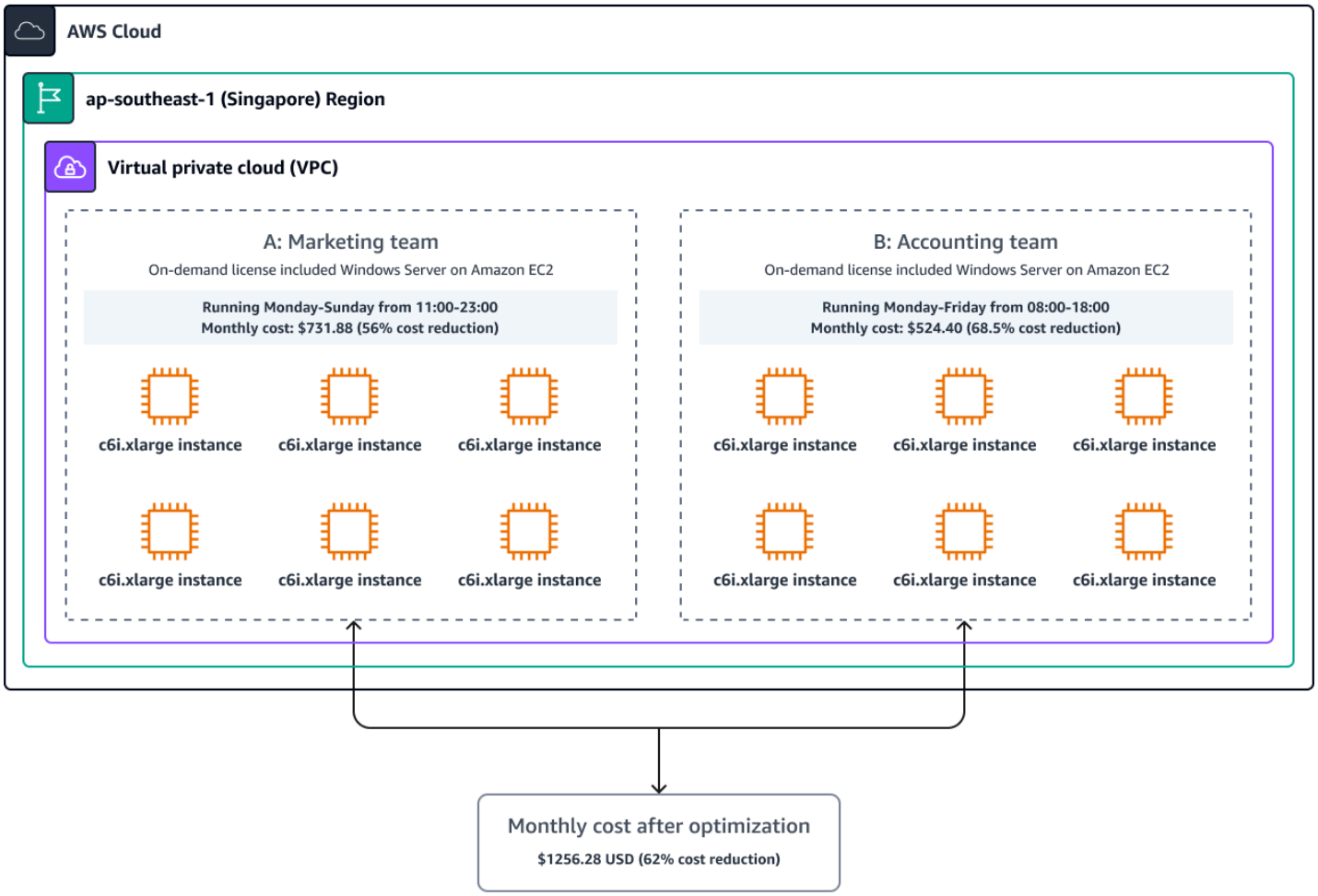
The following example scenario helps illustrate the cost advantages of using Instance Scheduler on AWS. In this scenario, a major retail company in Singapore deploys two Windows environments on Amazon EC2. The first environment, known as workload A, is utilized by the marketing team to analyze real-time in-store transactions while the stores are open. The second environment, known as workload B, is reserved for the accounting team, which works only during regular business hours. The current operating schedule of both environments (24/7) is not ideal given current usage patterns and requires optimization to reduce the company's operating costs.

The following diagram shows the monthly cost before optimization.



For example, there are 31 days in the month of March, out of which 23 are weekdays. If the marketing team uses Instance Scheduler on AWS and operates their instances only when needed (that is, for 321 hours per month instead of 730 hours per month), they could potentially save $932.52 each month. This amounts to a 56 percent reduction in operating costs. The accounting team can experience significant advantages as well, with their instance usage time dropping from 730 hours per month to 230 hours. This results in a reduction of $1,140, or 68.5 percent. The company could save a combined total of $2,072.52 per month (equal to a 62 percent reduction), or $24,870.24 annually.

The following diagram shows the monthly cost after optimization.



> **Note**
>
> The pricing for this example was determined by using the AWS Pricing Calculator in March 2023.

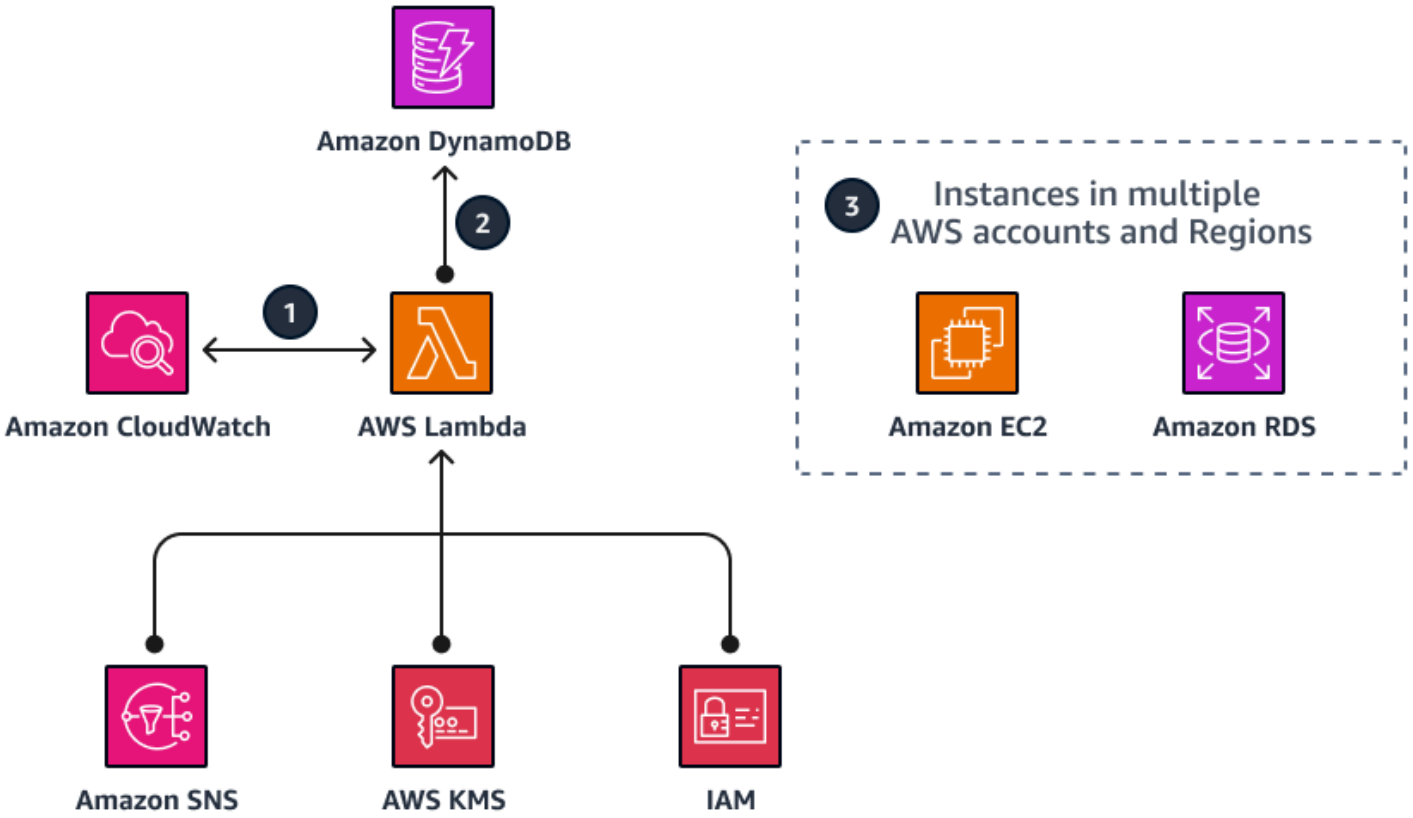
Cost optimization recommendations

This section explains how to deploy and configure the Instance Scheduler on AWS based on the example scenario covered in the previous *Cost optimization scenario* section. We recommend that you take the following next steps to optimize your costs by using the Instance Scheduler on AWS:

1. Launch the Instance Scheduler stack
2. Configure periods
3. Configure schedules

4. Tag instances

The following architecture diagram shows what's created in the AWS Cloud by the Instance Scheduler stack.



The diagram shows the following workflow steps:

1. An AWS CloudFormation template sets up an Amazon CloudWatch event at an interval that you define. This event invokes an AWS Lambda function. During configuration, you define the AWS Regions and accounts. You also define a custom tag that Instance Scheduler on AWS uses to associate schedules with applicable Amazon EC2 instances, Amazon RDS instances, and clusters.
2. The schedule configuration values are stored in Amazon DynamoDB, and the Lambda function retrieves them each time it runs. You can then apply the custom tag to applicable instances.
3. During initial configuration of the Instance Scheduler, you define a tag key for identifying applicable Amazon EC2 and Amazon RDS instances. When you create a schedule, the name you specify is used as the tag value that identifies the schedule that you want to apply to the tagged resource.

Launch the Instance Scheduler stack

This section shows you how to launch the CloudFormation stack for the Instance Scheduler on AWS.

> **Note**
>
> You're responsible for the cost of the AWS services used while running Instance Scheduler on AWS. As of January 2023, the cost for running this solution with default settings in the `us-east-1` Region is approximately $9.90 per month for Lambda charges, or less if you have a Lambda free tier monthly usage credit. For more information, see the *Cost* section of the Instance Scheduler on AWS Implementation Guide in the AWS Solutions Library.

To launch the instance scheduler stack, complete the following steps.

1. Sign in to the AWS Management Console and choose Launch solution (downloadable template) to launch `instance-scheduler-on-aws.template` CloudFormation template.

   > **Note**
   >
   > You can also download the template as a starting point for your own implementation.

2. The template launches in the US East (N. Virginia) Region by default. To launch Instance Scheduler in a different Region, use the Region selector in the console navigation bar.

   > **Note**
   >
   > This example uses the Asia Pacific (Singapore) Region.

3. On the **Create Stack** page, in the **Prerequisite - Prepare template** section, verify that the **Template is ready** option is selected. In the **Template source section**, verify that the **Amazon S3 URL** option is selected.
4. Verify that the correct template URL is in the **Amazon S3 URL** text box, and then choose **Next**.
5. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, see IAM and STS Limits in the AWS Identity and Access Management (IAM) documentation. The stack name for the example in this guide is called `MyInstanceScheduler`.

> **Note**
>
> The stack name can't contain more than 28 characters.

6. Under **Parameters**, review the parameters for the template and modify them as necessary.
7. Choose **Next**. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Select the box acknowledging that the template will create IAM resources.
9. Choose **Create** to deploy the stack.

Configure periods

After you deploy the CloudFormation template, the solution creates a DynamoDB table that contains sample period rules and schedules that you can use as a reference to create your own custom period rules and schedules. For an example period configuration, see Sample schedules in the Instance Scheduler on AWS documentation.

To complete the step for this scenario, you must generate periods that correspond to each workload and meet their specific needs. For example:

```
Period 1 (Workload A):
    Name: retail-hours
    Days: Monday to Sunday
    Hours: 1100 - 2300
Period 2 (Workload B):
    Name: office-hours
    Days: Monday to Friday
    Hours: 0800 - 1800
```

To configure periods, complete the following steps:

1. Sign in to the DynamoDB console and make sure you're in the same Region where you launched the CloudFormation template for the Instance Scheduler on AWS.
2. In the navigation pane, choose **Tables**, and then select the table named **ConfigTable**.
3. Choose **Explore table items**.
4. To create a period for office hours, select **period** for the **office-hours** item.

5. On the **Edit item** page, change the value of **begintime** to **0800** and **endtime** to **1800**. Leave the default value in place for weekdays.

> **Note**
>
> The **begintime** and **endtime** values determine when the instances should be started and stopped, while the **weekdays** value determines which days of the week this schedule applies to (Monday to Friday for this example).

6. Choose **Save changes**.
7. To duplicate the **office-hours** period and use it to create a new period for retail hours, select **period** for the **office-hours** item. Then, from the **Actions** menu, choose **Duplicate item**.
8. Modify the attributes to match your needs. The following attributes are used to meet the requirements of the example scenario:

```
type: period
name: retail-hours
begintime: 11:00
description: Retail hours
endtime: 23:00
weekdays: mon-sun
```

9. Choose **Create item**.
10. In the DynamoDB **ConfigTable**, identify the two periods that you just created listed in the item lists.

Configure schedules

In the context of Instance Scheduler on AWS, schedules refer to the application of one or more periods and the relevant time zone. These schedules are then assigned to your instances as tags. This section shows you how to create two schedules (shown below) to accommodate the varying time patterns of the two example workloads, and then associate the schedules with the periods that you created in the previous section.

```
Schedule 1:
    Name: singapore-office-hours
    Period: office-hours
    Timezone: Asia/Singapore
Schedule 2:
```

```
    Name: singapore-retail-hours
    Period: retail-hours
    Timezone: Asia/Singapore
```

To create and configure schedules, complete the following steps:

1. Sign in to the DynamoDB console and make sure you're in the same Region where you launched the CloudFormation template for the Instance Scheduler on AWS.
2. In the navigation pane, choose **Tables**, and then select the table named **ConfigTable**.
3. Choose **Explore table items**.
4. To duplicate the UK office hours schedule and use it to create a new schedule for your office hours (Singapore office hours, for this example), select **schedule** for the **uk-office-hours** item. Then, from the **Actions** menu, choose **Duplicate item**.
5. Modify the attributes to match your needs. The following attributes are used to meet the requirements of the example scenario:

```
type: schedule
name: singapore-office-hours
description: Office hours in Singapore
periods: office-hours
timezone: Asia/Singapore
```

6. Choose **Create item**.
7. Repeat steps 4–6 to create a schedule for Singapore retail hours using the following attribute values:

```
type: schedule
name: singapore-retail-hours
description: Retail hours in Singapore
periods: retail-hours
timezone: Asia/Singapore
```

8. In the DynamoDB **ConfigTable**, identify the two schedules and two periods that you created.

Tag instances

After you establish your schedules, you must uses tags to allocate the schedules to the specific instances that you want to use. You can use the tag editor within AWS Resource Groups to generate and assign tags to your Amazon EC2 instances.

1. Sign in to the AWS Management Console and make sure you're in the same Region where you previously launched the CloudFormation template.
2. Open the Resource Groups console. In the navigation pane, expand **Tagging**, and then choose **Tag Editor**.
3. In the **Find resources to tag** section, for **Regions**, choose your Regions. For **Resource types**, choose Amazon EC2 or Amazon RDS. This scenario focuses on the Amazon EC2 instances in workload A. The marketing team is using workload A in the Singapore Region. The resources for this workload are already tagged with a **Department** key and a **Marketing** value. You can use this tag to search for the instances.
4. Choose **Search resources**.
5. Select the instances that you want to include in the schedule from the list of search results, and then choose **Manage tags of selected resources**.
6. In the **Edit tags of all selected resources** section, choose **Add tag** to add the Instance Scheduler schedule tags to your EC2 instances. You can use the tag keys and values that match **schedulea** (previously created in DynamoDB).
7. For **Tag key**, add **Schedule**. For **Tag value**, enter **singapore-retail-hours**.
8. Choose **Review and apply tag changes**.
9. To apply the tag to all the EC2 instances that you selected, choose **Apply changes to all selected**.

10Repeat steps 3–9 for any additional schedules you wish to apply.

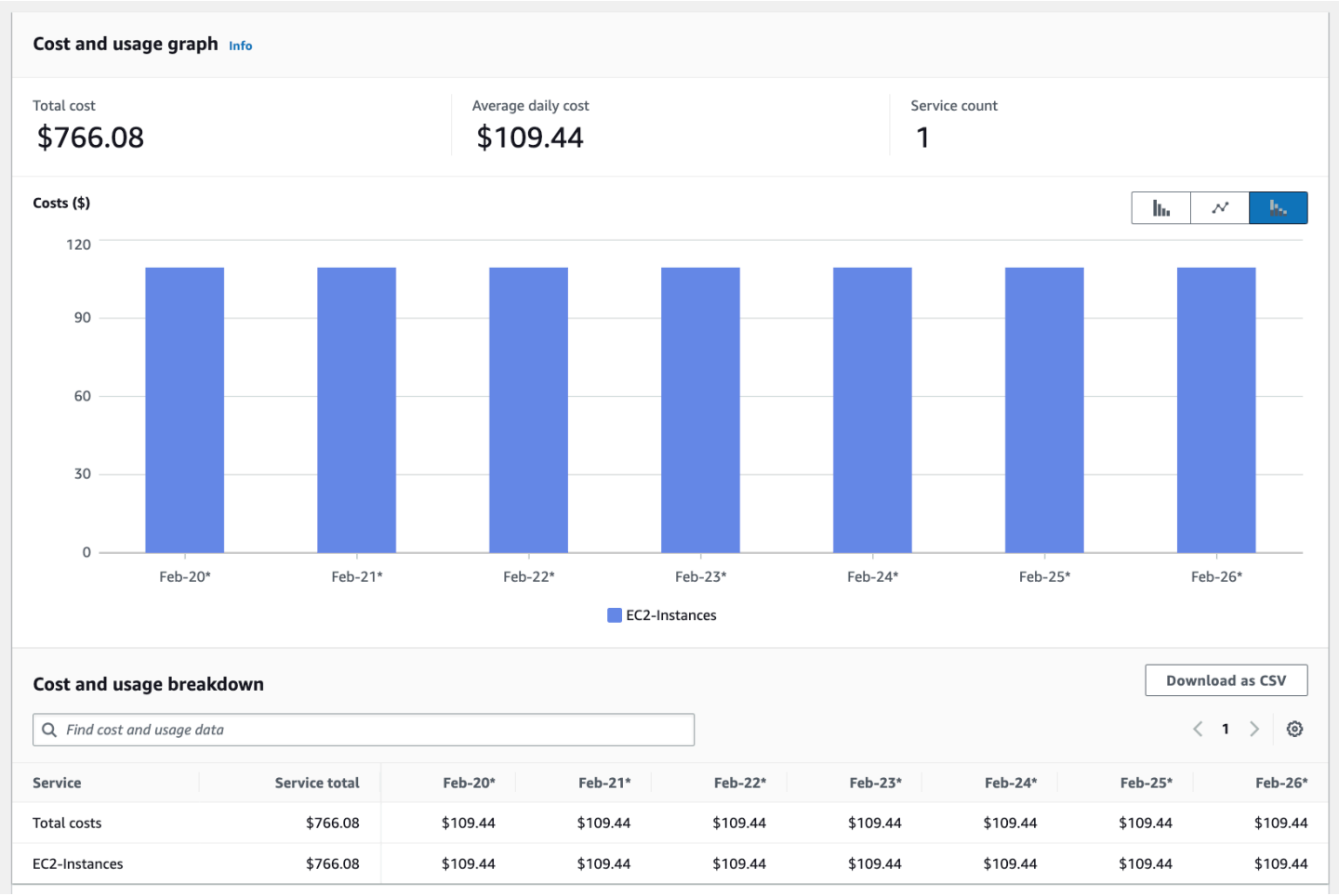
Validate results

We recommend that you use AWS Cost Explorer to measure the cost benefits of using Instance Scheduler on AWS. You can use Cost Explorer to do the following:

- View and analyze the costs associated with your EC2 instances, including instances managed by Instance Scheduler.
- Filter your Cost Explorer view by tags so that you can focus on specific workloads and get a granular view of the cost savings achieved by using Instance Scheduler.
- Gain insights into the financial impact of using Instance Scheduler.
- Identify opportunities for further cost optimization and make data-driven decisions to optimize your AWS spending.

The following charts illustrate the cost of operating workload A and workload B during a seven day period (Monday–Sunday) before optimization by using Instance Scheduler.

Combined total expenses of workloads A and B



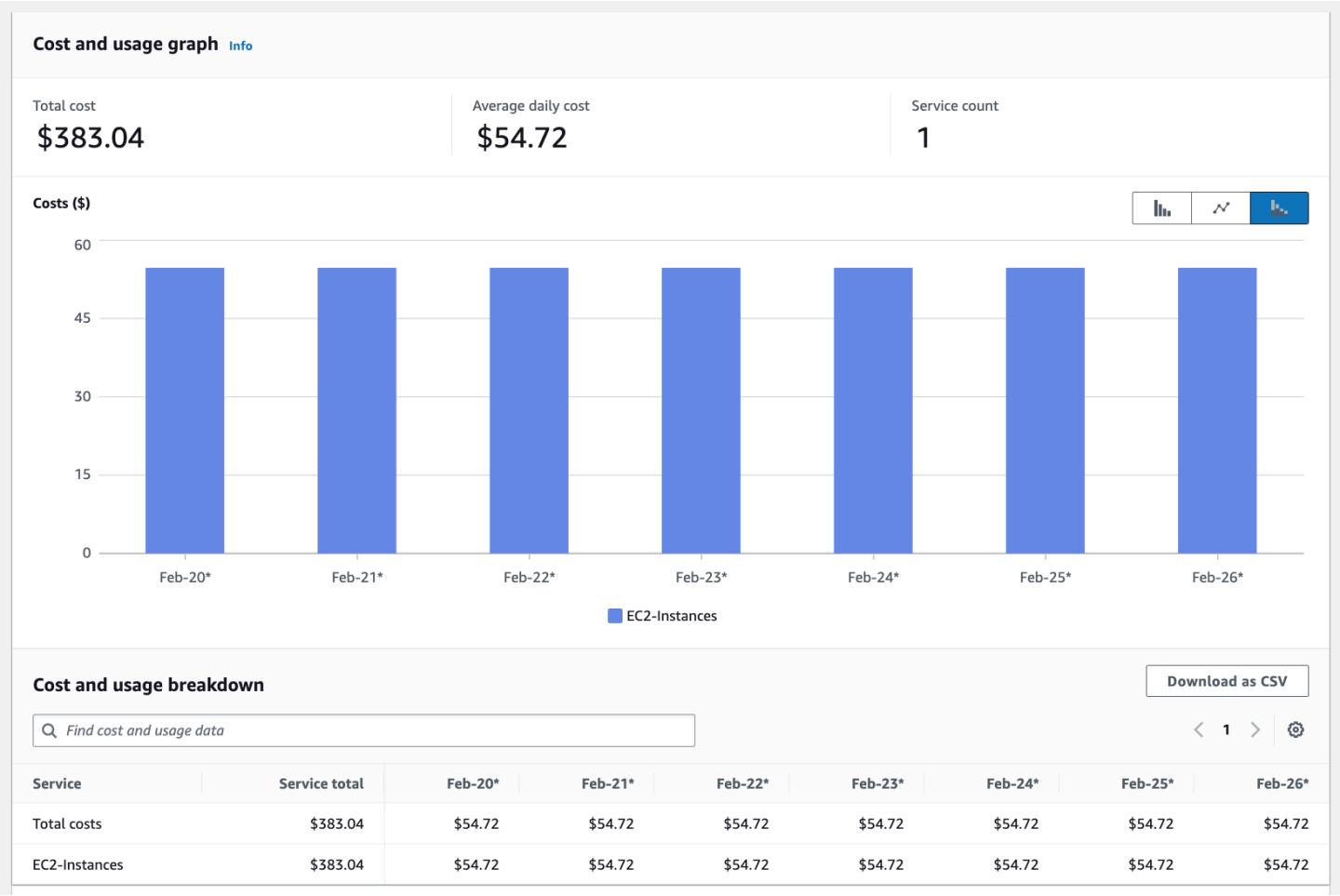
Cost and usage breakdown Download as CSV

Find cost and usage data

1

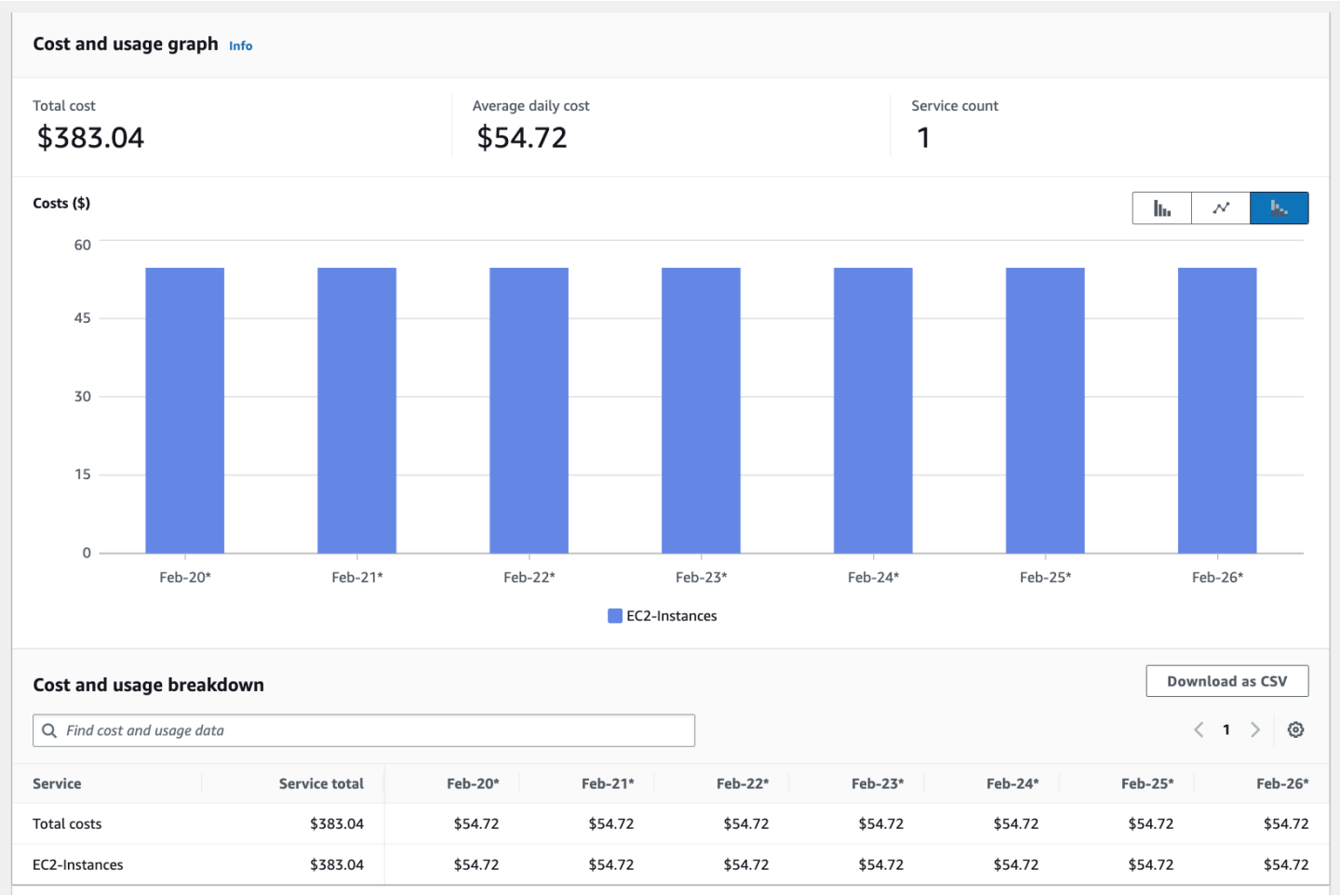
Service	Service total	Feb-20*	Feb-21*	Feb-22*	Feb-23*	Feb-24*	Feb-25*	Feb-26*
Total costs	$766.08	$109.44	$109.44	$109.44	$109.44	$109.44	$109.44	$109.44
EC2-Instances	$766.08	$109.44	$109.44	$109.44	$109.44	$109.44	$109.44	$109.44

Workload A expenses



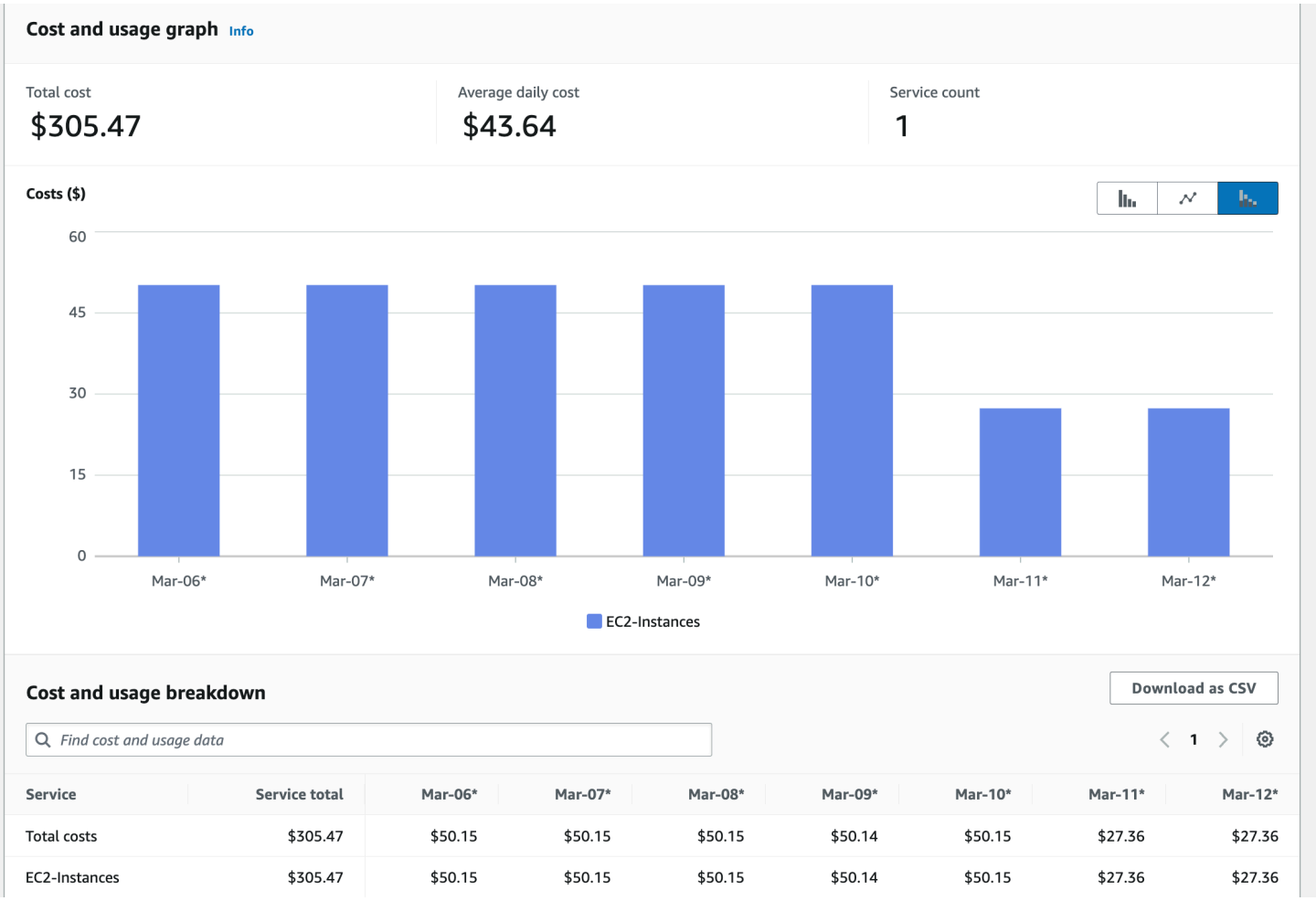
Service	Service total	Feb-20*	Feb-21*	Feb-22*	Feb-23*	Feb-24*	Feb-25*	Feb-26*
Total costs	$383.04	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72
EC2-Instances	$383.04	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72	$54.72

Workload B expenses



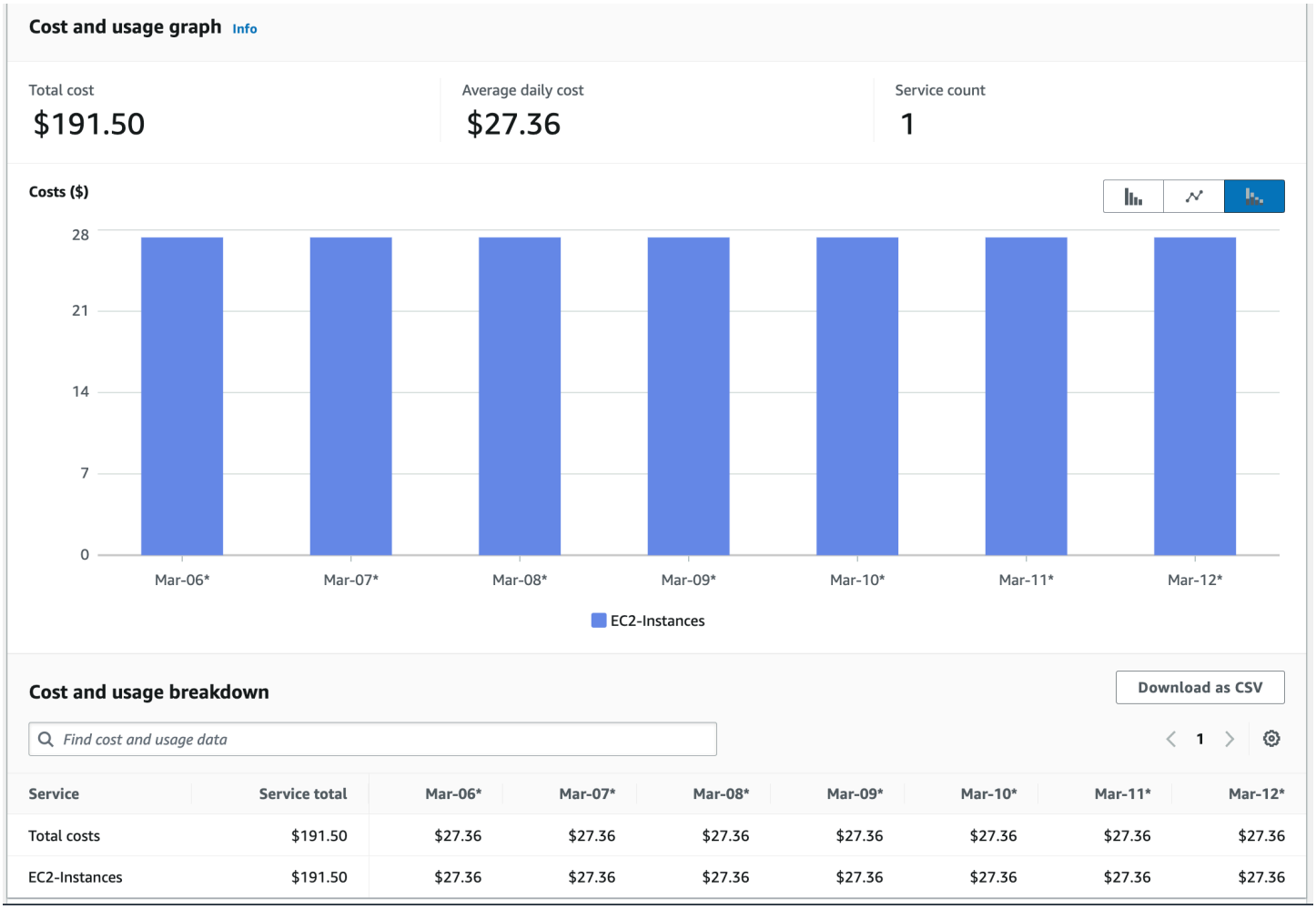
In this scenario, Cost Explorer shows the cost reductions that result from implementing Instance Scheduler on AWS. The following charts depict the operational costs of workload A and workload B for a period of seven days (Monday–Sunday) post-optimization.

Combined total expenses of workload A and B

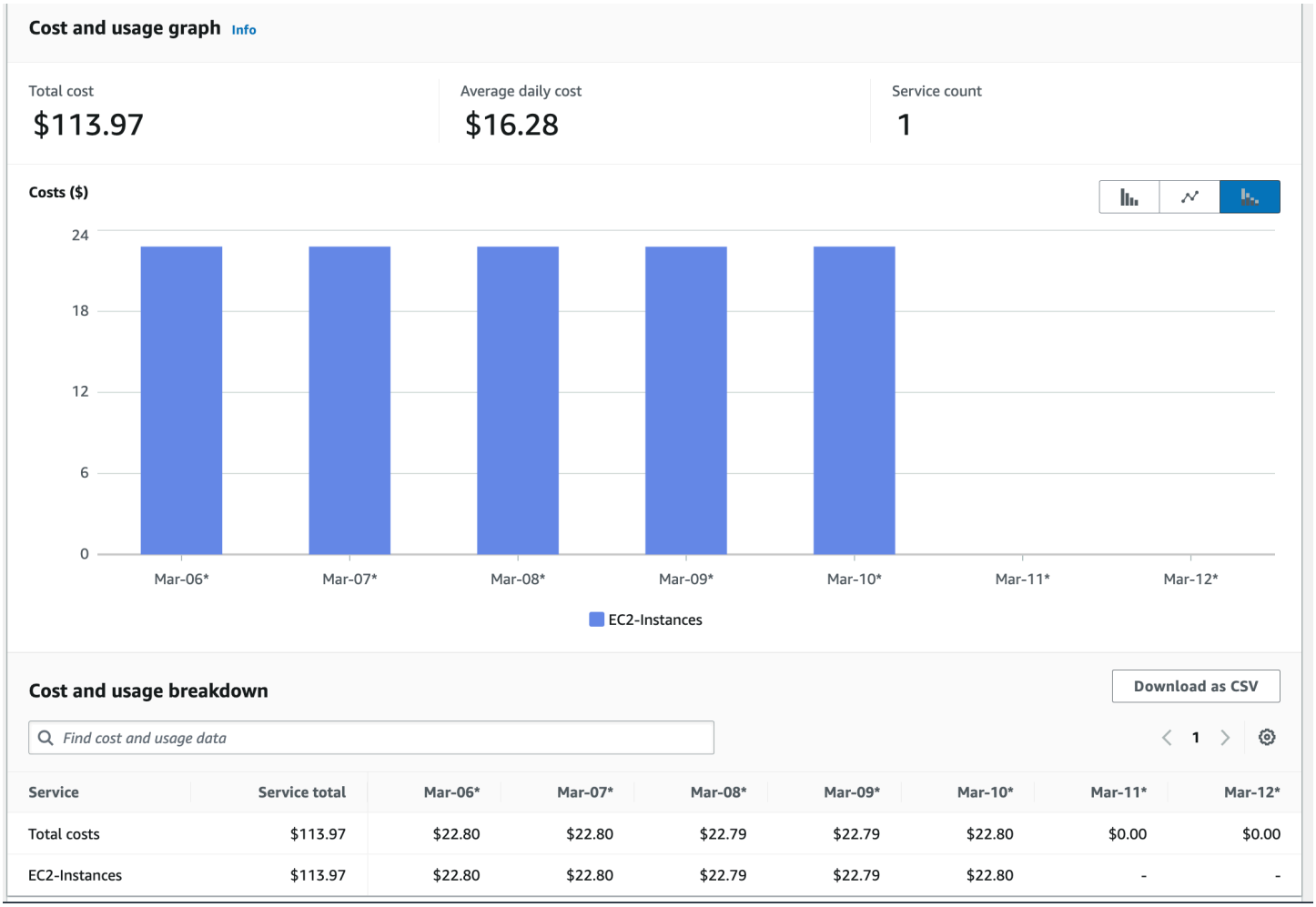


Service	Service total	Mar-06*	Mar-07*	Mar-08*	Mar-09*	Mar-10*	Mar-11*	Mar-12*
Total costs	$305.47	$50.15	$50.15	$50.15	$50.14	$50.15	$27.36	$27.36
EC2-Instances	$305.47	$50.15	$50.15	$50.15	$50.14	$50.15	$27.36	$27.36

Workload A expenses



Workload B expenses



Service	Service total	Mar-06*	Mar-07*	Mar-08*	Mar-09*	Mar-10*	Mar-11*	Mar-12*
Total costs	$113.97	$22.80	$22.80	$22.79	$22.79	$22.80	$0.00	$0.00
EC2-Instances	$113.97	$22.80	$22.80	$22.79	$22.79	$22.80	-	-

Additional resources

- Automate starting and stopping AWS instances (Instance Scheduler on AWS documentation)
- Back to Basics: Using an Instance Scheduler to Control Amazon EC2 and Amazon RDS Resource Costs (YouTube)
- Tagging your AWS resources (Tagging AWS Resources User Guide)
- Analyzing your costs with AWS Cost Explorer (AWS Billing and Cost Management documentation)

Right size Windows workloads

Overview

Right sizing is one of the most potent cost-saving tools. AWS offers various methods to collect right sizing information, from reviewing potential workloads by using an AWS Optimization and Licensing Assessment (AWS OLA) to reviewing existing workloads by using AWS Cost Explorer.

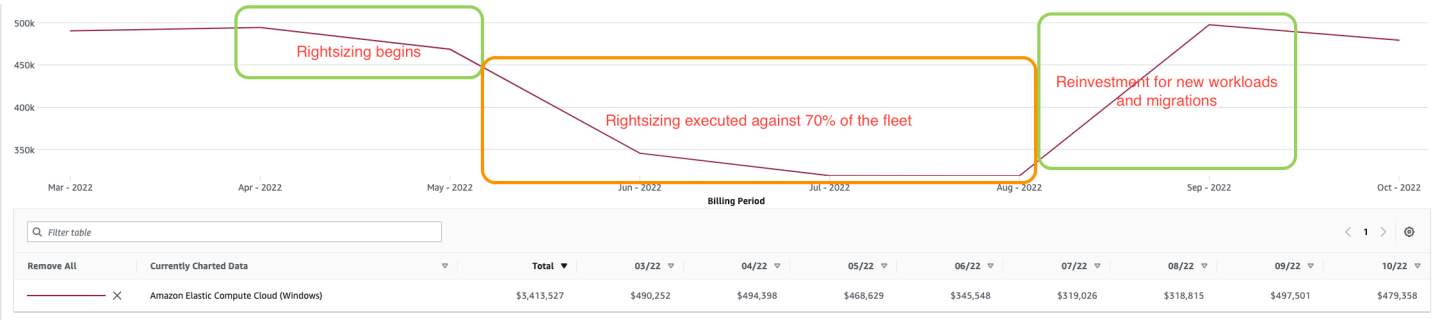
This section shows you how to use AWS Compute Optimizer to identify Amazon EC2 right sizing opportunities. Compute Optimizer helps prevent over-provisioning and under-provisioning for the following types of AWS resources:

- Amazon Elastic Compute Cloud (Amazon EC2) instance types
- Amazon Elastic Block Store (Amazon EBS) volumes
- Amazon Elastic Container Service (Amazon ECS) services on AWS Fargate
- AWS Lambda functions based on utilization data provided by Amazon CloudWatch

Cost optimization scenario

Measuring the effectiveness of right sizing can be challenging, because right sizing efforts can be directed toward a specific app, team, or the entire organization. For example, consider an organization that migrates several thousand instances to AWS, with 90 percent of their fleet comprising Windows workloads. The organization can employ Compute Optimizer to analyze their fleet and discover significant over-provisioning across their accounts and AWS Regions. Then, they can use AWS Systems Manager Automation to right size their fleet through multiple maintenance windows. As a result, the organization manages to adjust the right sized instance type for 70 percent of their fleet and achieves a cost savings of 35 percent.

The following dashboard illustrates the savings achieved over several months as this example organization strategically implemented the right sizing recommendations of Compute Optimizer. Their objective was to operate their existing workloads as efficiently as possible in order to resume a stalled migration from a colocation data center nearing the end of its contract.



Cost optimization recommendations

We recommend that you take the following next steps to optimize your costs by using Compute Optimizer:

- Enable Compute Optimizer
- Enable Memory metrics collection for Windows nodes
- Consume Compute Optimizer recommendations
- Tag instances for right sizing
- Enable the cost allocation tag to work with AWS billing tools
- Implement right sizing recommendations with AWS Systems Manager Automation
- Consider alternative resizing methods
- Review before and after costs in Cost Explorer

Enable Compute Optimizer

You can enable Compute Optimizer at the organization or single account level in AWS Organizations. The organization-wide configuration provides ongoing reports for new and existing instances across your entire fleet for all member accounts. This enables right sizing to be a recurring activity instead of a point-in-time activity.

Organization level

For most organizations, the most efficient way to use Compute Optimizer is at the organization level. This provides multi-account and multi-Region visibility into your organization and centralizes the data into one source for review. To enable this at the organization level, do the following:

1. Sign in to your Organizations management account with a role that has the required permissions and choose to opt in for all accounts within this organization. Your organization must have all features enabled.
2. After you enable the management account, you can sign in to the account, see all other member accounts, and browse their recommendations.

> **Note**
>
> It's a best practice to configure a delegated administrator account for Compute Optimizer. This enables you to exercise the principle of least privilege. That way, you can minimize access to the organization's management account while still providing access to the organization-wide service.

Single account level

If you're targeting an account with high costs but don't have access to AWS Organizations, you can still enable Compute Optimizer for that account and Region. To learn about the opt-in process, see Getting started with AWS Compute Optimizer in the Compute Optimizer documentation.

Enable memory metrics collection for Windows nodes

Memory metrics supply Compute Optimizer with the essential metrics required to make well-informed right sizing recommendations in your organization. This is due to the analysis of CPU, memory, network, and storage being conducted before offering a recommendation.

To pass memory metrics from Windows EC2 instances to Compute Optimizer, you must enable the CloudWatch agent and configure memory metrics to be collected every 60 seconds. There is no additional cost for using memory metrics with CloudWatch.

Enable the CloudWatch agent and configure memory metrics

Download the ComputeOptimize.yml file. You can use this file to enable memory collection for all instances in your account. The template file generates the following components:

- AWS Systems Manager Parameter Store – This stores the configuration for the CloudWatch agent that's required for collecting memory metrics.
- AWS Identity and Access Management (IAM) role with AWS managed policies for AWS Systems Manager attached – This is for the Systems Manager Automation document.

- AWS Systems Manager documents – This installs and configures the CloudWatch agent (replacing any existing CloudWatch configuration).
- AWS Systems Manager State Manager association – This enables Systems Manager documents to run on all instances in your account.

> **Important**
>
> Running this template overwrites any existing CloudWatch configuration on the instances.

Next, do the following:

1. Sign in to the AWS Management Console and open the CloudFormation console.
2. In the navigation pane, choose **Stacks**.
3. Choose **Create stack**, and then choose **With existing resources (import resources)**.
4. Choose **Next**.
5. For **Template resource**, select **Upload a template file**.
6. Choose **file**, and then upload the `ComputeOptimize.yml` file.
7. Choose **Next**.
8. On the **Specify stack details** page, for **Stack name**, enter a name for your stack, and then choose **Next**.
9. On the **Identify resources** page, enter the identifier values for the resources you're importing.
10. Choose **Import resources**.
11. After the stack is deployed, choose the **Outputs** tab to find the key, value, and description for your association.

Monitor the progress of the association

1. After the deployment of the CloudFormation stack is complete, open the Systems Manager console.
2. In the navigation pane, in the **Node Management** section, choose **State Manager**.
3. On the **Associations** page, choose the association ID of your association.
4. Choose the **Execution history** tab.

5. In the **Execution id** column, choose your association's execution ID. The status should be **Success**.

View the metrics in CloudWatch

We recommend that you wait for at least five minutes for the metrics to populate CloudWatch.

1. Open the CloudWatch console.
2. In the navigation pane, expand the **Metrics** section, and then choose **All metrics**.
3. Confirm that the metrics appear under the **CWAgent** namespace.

> **Note**
>
> To apply the settings to any new instances, rerun the association.

Consume Compute Optimizer recommendations

Consider an example that focuses on making right sizing changes within a single account and single Region. In this example, Compute Optimizer is enabled at the organization level across all accounts. Keep in mind that right sizing is a disruptive process that in most cases is carried out with precision by the application owners during a scheduled maintenance window over several weeks.

If you navigate to Compute Optimizer from within an organization's management account (as shown in the following steps), you can choose the account that you want to investigate. In this example, there are six instances running in a single account in the `us-east-1` Region. All six instances are over-provisioned. The goal is to resize the instances based on the recommendations from Compute Optimizer.

Identify over-provisioned instances and export recommendation details

1. Sign in to the AWS Management Console and open the Compute Optimizer console.
2. In the navigation pane, choose **Dashboard**.
3. In the search box on the **Dashboard** page, enter **Region=US East (N. Virginia)**. Then, enter **Findings=Over-provisioned**. These filters allow you to see all the over-provisioned instances in the `us-east-1` Region.

4. To review detailed recommendations for over-provisioned EC2 instances, scroll down to the **EC2 instances** card, and then choose **View recommendations**.
5. Choose **Export** and save the file for future use.
6. For **S3 bucket**, enter the name of the Amazon S3 bucket that you want to be the destination for the export file.

> **Note**
>
> To save recommendations for future review, you must have an S3 bucket available for Compute Optimizer to write to in each Region. For more information, see Amazon S3 bucket policy for AWS Compute Optimizer in the Compute Optimizer documentation.

7. In the **Export filters** section, select the **Include recommendations for all member accounts in the organization** check box.
8. For **Resource type**, choose **EC2 instances**.
9. In the **Columns to include** section, select the **Select all** check box.
10. Choose **Export**.

Choose instances based on recommendations

Instance recommendations are based on the performance metrics collected and analyzed by Compute Optimizer. It's essential to be aware of the workloads running on the instance to ensure that you choose the best instance. This example assumes that you can choose from the latest generation of Amazon EC2 R6i, R5, and T3 instances. T3 instances are burstable and have lower network bandwidth capabilities. R5 and R6 instances have the same cost per hour and are nearly identical. However, the R6 instance has a higher network bandwidth capacity, features the latest generation of Intel processors, and offers the same compute footprint as the R5. In this example, R6 is the best option to choose for resizing.

1. In the Compute Optimizer console, choose **Recommendations for EC2 instances** from the navigation bar. This page shows you a comparison of the current instance type with recommended options to replace it.
2. To get the ID of the instance that you want to right size, open the Amazon S3 console from the management account in AWS Organizations.
3. In the navigation pane, choose **Buckets**, and then choose the bucket that you're using to store your exported results.

4. On the **Objects** tab, select your export file from the objects list, and then choose **Download**.
5. To extract the instance information from the file, you can use the **Text to Columns** button on the **Data** tab in Microsoft Excel.

> **Note**
>
> Instance IDs are represented as Amazon Resource Names (ARNs). Be sure to set the delimiter to "/" and extract the instance ID. Alternatively, you can write a script or use an integrated development environment (IDE) to trim the ARN.

6. In Excel, filter the **finding** column to display only the **OVER_PROVISIONED** instances. These are the instances that you're targeting for right sizing.
7. Save the instance IDs in a text editor for easy access later.

Tag instances for right sizing

Tagging your workloads is a powerful tool for organizing your resources in AWS. Tags enable you to gain fine-grained visibility into costs and facilitate chargeback. For more information about strategies and methods for adding tags to AWS resources, see the AWS Whitepaper Best Practices for Tagging AWS Resources. For this example, you can use the AWS Tag Editor to make tagging adjustments across the over-provisioned instances that you wish to target for resizing during a maintenance window. You can also use this tag to view the costs before and after the change.

1. Sign in to the AWS Management Console and open the AWS Resource Groups console for the account containing the instances targeted for resizing.
2. On the navigation bar, in the **Tagging** section, choose **Tag Editor**.
3. For **Regions**, select your target Region.
4. For **Resources types**, choose **AWS::EC2::Instance**.
5. Choose **Search resources**.
6. On the **Resource search results** page, select all the instances that you want to right size, and then choose **Manage tags of selected resources**.
7. Choose **Add tag**.
8. For **Tag key**, enter **Rightsizing**. For **Tag value**, enter **enabled**. Then, choose **Review and apply tag changes**.

> **Note**
>
> You can include additional metadata like Team or Business Unit to help with filtering later on in Cost Explorer.

After creating and applying user-defined tags to your resources, it may take up to 24 hours for the tags to appear on your cost allocation tags page for activation. After you select your tags for activation, it could take another 24 hours for the tags to become active.

For advanced users, you can use AWS CloudShell within the target account and Region to tag multiple instances. For example:

```
bash
#!/bin/bash
# Set variables
TAG_KEY="rightsizing"
TAG_VALUE="type-m5"
# Get a list of instance IDs
INSTANCE_IDS=$(aws ec2 describe-instances –query
 "Reservations[].Instances[].InstanceId" –output text)
# Loop through each instance ID and add the tag
for INSTANCE_ID in $INSTANCE_IDS; do
 aws ec2 create-tags –resources $INSTANCE_ID –tags Key=$TAG_KEY,Value=$TAG_VALUE
done
```

Enable the cost allocation tag to work with AWS billing tools

We recommend activating the user-defined cost allocation tag. This enables the **Rightsizing** tag to be recognized and filterable in the AWS billing tools (for example, Cost Explorer and AWS Cost and Usage Report). If you don't enable this, the tag filtering option and data won't be available. For information about using cost allocation tags, see Activating user-defined cost allocation tags in the AWS Billing and Cost Management documentation.

1. Sign in to the AWS Management Console and open the AWS Billing console.
2. On the navigation pane, in the **Billing** section, choose **Cost allocation tags**.
3. On the **User-defined cost allocation tags** tab, enter **Rightsizing**.
4. Select the **Rightsizing** tag key, and then choose **Activate**.

After 24 hours, the tag should appear in Cost Explorer.

Implement right sizing recommendations with Systems Manager Automation

Resizing is a scenario that requires an instance to be stopped and started. In this scenario, you might need to handle this interruption in a maintenance window and need different teams to handle their own resizing. Before you change an instance type, review the Considerations for compatible instance types in the Amazon EC2 documentation.

The example steps in this section implement right sizing recommendations per account and Region by using a Systems Manager Automation document called AWS-ResizeInstance. This approach is typical for most organizations as most organizations require different instance types for different purposes. You can also use the same `AWS-ResizeInstance` automation document to target single and multi-account deployments.

1. Sign in to the AWS Management Console and open the Systems Manager console.
2. On the navigation pane, in the **Shared Resources** section, choose **Documents**.
3. In the search bar, enter **AWS-ResizeInstance**, and then choose the **AWS-ResizeInstance** from the search results.
4. Choose **Execute automation**.
5. On the **Execute automation runbook** page, choose **Simple execution**.
6. In the **Input parameters** section, enter **InstanceId** and **InstanceType**. Keep the rest of the default values.
7. Choose **Execute**, and then wait for the automation to go through the steps to change the instance type.

Consider alternative resizing methods

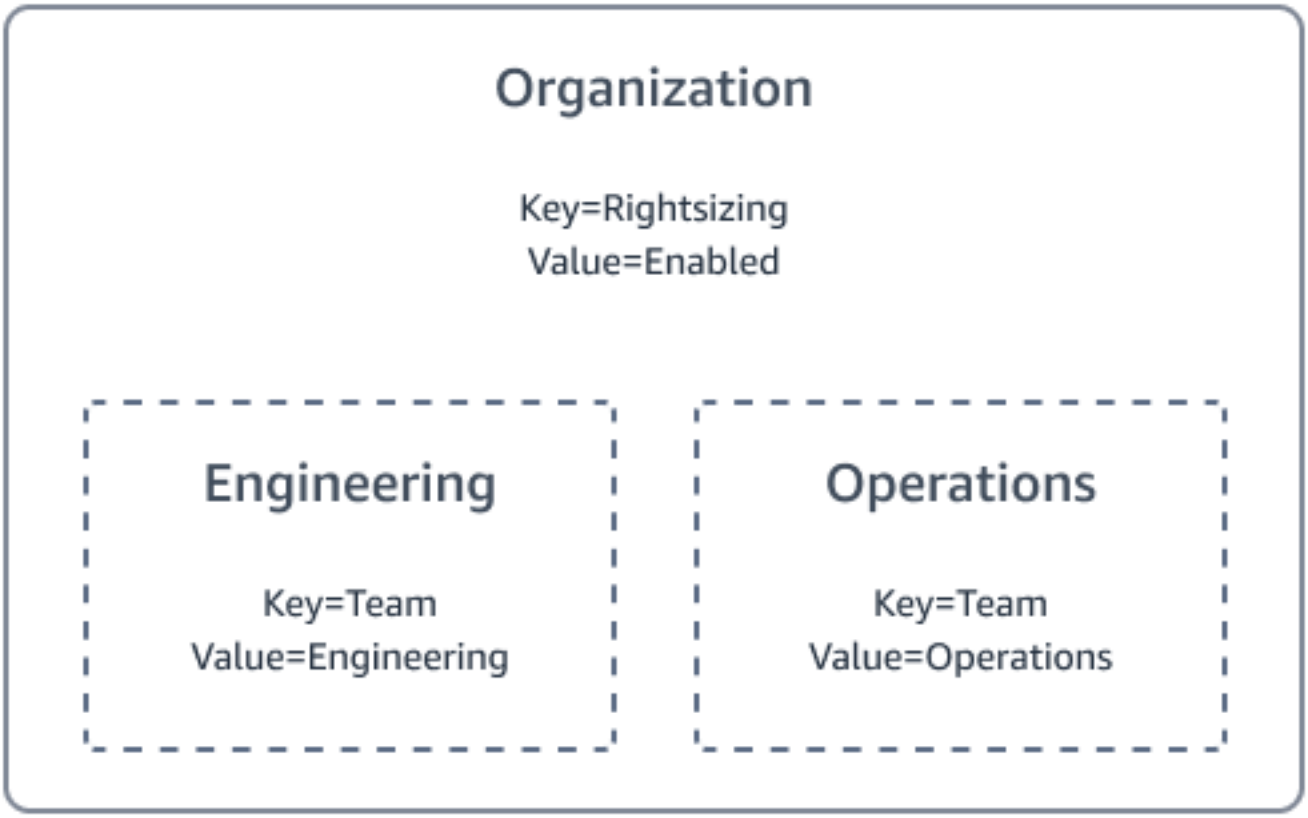
If you're using a launch template to deploy your instances, you can update the launch template with the right sized instance type, and then perform an instance refresh to replace the instances with the right-sized version.

If you plan to use the right sizing process across multiple accounts and Regions, you must create a custom Systems Manager Automation document. This document enables you to feed in multiple instances as a parameter and target instances moving to the same destination instance type (for example, all instances transitioning to t3a.medium, irrespective of the source instance type).

Review before and after costs in Cost Explorer

After you have right sized your resources, you can use Cost Explorer to show before and after costs by using the **Rightsizing** tag. Recall that you can use resource tags to track costs. By using several layers of tags, you can achieve granular visibility into your costs. In the example covered in this guide, the **Rightsizing** tag is used to apply a generic tag to all targeted instances. Then, a team tag is used to further organize resources. The next step is to introduce application tags to further show the cost impact for operating a specific application.

The following diagram shows the tag structure for an organization.



Consider the example of a business that right sizes the production web servers owned by the Operations team. In Cost Explorer, the **Rightsizing** tag is set to **enabled**, and the **Team** tag is set to **operations**. In this example, the right sizing effort reduces operating costs from 0.89 cents to 0.28 cents an hour. Assuming 744 hours per month, the annual cost before right sizing is $7,945.92. After right sizing, the annual cost drops to $2,499.84. This translates to a 68.5 percent decrease in annual workload costs. Imagine the impact of this across a large organization. Keep in mind, this is done in a sample environment and the instances are mostly idle. In a production environment, you can see savings between 10–35 percent.

Now, consider the impact of right sizing the production bastion host owned by the Engineering team. In Cost Explorer, the **Rightsizing** tag is set to **enabled**, and the **Team** tag is set to **engineering**. In this example, the right sizing effort reduces costs from 0.75 cents to 0.44 cents an hour. Assuming 744 hours per month, the annual cost before right sizing is $6,696.00. After right sizing, the annual cost drops to $3,928.32.

If you use multiple tags, you can filter the data down to granular cost details. In this example, the **Team** tag reduces the noise so that you can view the impact at a team level. Because the **Rightsizing** tag is enabled, you can also filter for any instance that has that tag with the value of **enabled** or no value present. This can provide a global view of your right sizing efforts, particularly when viewed in the management account (payer) at the Cost Explorer level. This view enables you to see all accounts and instances.

Consider an example at the single account level where the **Rightsizing** tag is set to **enabled**. The operating costs drop from $1.64 an hour to $.72 cents an hour. Assuming 744 hours per month, the annual cost before right sizing is $14,641.92. After right sizing, the annual cost drops to $6,428.16. This translates to a 56 percent decrease in compute costs for this account.

Before embarking on your right sizing journey, consider the following:

- AWS offers many options for cost reduction. This includes AWS OLA, where AWS reviews your on-premises instances prior to moving to AWS. The AWS OLA also provides you with right sizing recommendations and licensing guidance.
- Complete all of your right sizing before purchasing Savings Plans. This can help you avoid over purchases on your Savings Plans commitment.

Recommendations

We recommend the following next steps:

1. Review your existing landscape and consider converting Amazon EBS gp2 volumes to gp3 volumes.
2. Review Savings Plans.

Additional resources

- AWS Compute Optimizer (AWS documentation)

- Best Practices for Tagging AWS Resources (AWS Whitepapers)
- How to collect data from AWS Compute Optimizer and AWS Trusted Advisor across your AWS Organizations (YouTube)
- Optimizing performance and reducing licensing costs: Leveraging AWS Compute Optimizer for Amazon EC2 SQL Server instances (Microsoft Workloads on AWS blog)

Select the right instance type for Windows workloads

Overview

A significant distinction between workloads operating in the cloud compared to on-premises environments is the practice of over-provisioning. When purchasing physical hardware for on-premises use, you make a capital expenditure expected to last for a predetermined duration, typically 3–5 years. To accommodate anticipated growth during the hardware's lifespan, the hardware is acquired with more resources than your workload currently requires. Consequently, physical hardware is often over-provisioned well beyond the needs of your actual workload.

Virtual machine (VM) technology emerged as an effective means of utilizing surplus hardware resources. Administrators over-provisioned VMs with vCPUs and RAM, enabling the hypervisor to manage physical resource usage between busy and idle servers by allocating unused resources to each VM. When managing VMs, the vCPU and RAM resources allocated to each VM functioned more as resource governors rather than indicators of actual usage. VM resource over-allocation could easily exceed three times the available compute resources.

Amazon Elastic Compute Cloud (Amazon EC2) refrains from over-provisioning VMs on the underlying hardware, as it is unnecessary. Cloud computing is an operational expense, not a capital expense, and you only pay for what you use. If your workload requires more resources in the future, provision them when you actually need them, rather than doing so preemptively.

There are hundreds of options for choosing the right Amazon EC2 instance types. If you plan to migrate a Windows workload to the cloud, AWS offers an AWS OLA to help you better understand your current workload and provide an example of its performance on AWS. The AWS OLA analysis aims to match the suitable EC2 instance type and size to your actual on-premises usage.

If you already have workloads running on Amazon EC2 and seek cost optimization strategies, this section of the guide helps you identify differences between Amazon EC2 instances and their applicability to typical Windows workloads.

Cost optimization recommendations

To optimize the costs of your EC2 instance types, we recommend that you do the following:

- Choose the right instance family for your workload
- Understand price variations between processor architectures
- Understand price to performance differences across EC2 generations
- Migrate to newer instances
- Use burstable instances

Choose the right instance family for your workload

It's important to choose the right instance family for your workload.

Amazon EC2 instances are divided into these various groups:

- General purpose
- Compute optimized
- Memory optimized
- Accelerated computing
- Storage optimized
- HPC optimized

Most Windows workloads fit into the following categories:

- General purpose
- Compute optimized
- Memory optimized

To simplify this even further, consider a baseline EC2 instance in each category:

- Compute optimized – C6i
- General purpose – M6i
- Memory optimized – R6i

The previous generation of EC2 instances exhibited minor differences in processor types. For example, C5 compute optimized instances have faster processors than M5 general purpose instances or R5 memory optimized instances. The latest generation of EC2 instances (C6i, M6i, R6i, C6a, M6a, and R6a) all use the same processor across instance families. Since the processor is consistent among the latest generation of instances, the price difference between the instance families now depends more on the amount of RAM. The more RAM an instance has, the more expensive it is.

The following example illustrates the hourly pricing for an Intel-based 4 vCPU instance running in the `us-east-1` Region.

Instance	vCPUs	RAM	Hourly price
c6i.xlarge	4	8	$0.17
m6i.xlarge	4	16	$0.19
r6i.xlarge	4	32	$0.25

> **Note**
>
> Pricing is based on on-demand hourly pricing in the `us-east-1` Region.

Burstable instances

While it's a best practice in cloud computing to turn off unused compute resources to avoid charges, not all workloads can be switched off and on each time they're needed. Some workloads remain idle for extended periods but must be accessible 24 hours a day.

Burstable instances (T3) offer a way to maintain spiky or low-utilization workloads online all day while still keeping compute costs low. Burstable EC2 instances have a maximum amount of vCPU resources that the instance can use for brief periods. These instances employ a system based on burstable CPU credits. These credits are accumulated during idle periods throughout the day. Burstable instances offer varying vCPU-to-RAM ratios, making them alternatives to compute optimized instances in some cases and to other general purpose instances in others.

The following example illustrates the hourly pricing for a T3 instance (that is, burstable instance) running in the `us-east-1` Region.

Instance	vCPUs	RAM (GB)	Hourly price
t3.nano	2	0.5	$0.0052
t3.micro	2	1	$0.0104
t3.small	2	2	$0.0208
t3.medium	2	4	$0.0416
t3.large	2	8	$0.0832
t3.xlarge	4	16	$0.1664
t3.2xlarge	8	32	$0.3328

Note

Pricing is based on on-demand hourly pricing in the `us-east-1` Region.

Understand price variations between processor architectures

Intel processors have been the standard for EC2 instances since their inception. Earlier generations of EC2 instances, such as C5, M5, and R5, don't indicate Intel as the processor architecture (as it was the default). Newer generations of EC2 instances, like C6i, M6i, and R6i, include an "i" to denote the use of an Intel processor.

The change in processor architecture annotation is due to the introduction of additional processor options. The most comparable processor to Intel is AMD (denoted with an "a"). AMD EPYC processors use the same x86 architecture and offer performance similar to Intel processors but at a lower price. As demonstrated in the following pricing examples, AMD EC2 instances provide an approximately 10 percent discount on compute costs compared to their Intel counterparts.

Intel instance	Hourly price	AMD instance	Price	% difference
c6i.xlarge	$0.17	c6a.xlarge	$0.153	10%

Intel instance	Hourly price	AMD instance	Price	% difference
m6i.xlarge	$0.192	m6a.xlarge	$0.1728	10%
r6i.xlarge	$0.252	r6a.xlarge	$0.2268	10%

Note

Pricing is based on on-demand hourly pricing in the `us-east-1` Region.

The third major processor architecture option is AWS Graviton processors (denoted with a "g") on EC2 instances. Designed by AWS, Graviton processors offer the best price performance on Amazon EC2. Not only are current Graviton processors 20 percent cheaper than their Intel counterparts, but they also deliver a 20 percent or greater performance boost. The next generation of Graviton processors is expected to further extend this performance difference, with testing showing an additional 25 percent increase in performance.

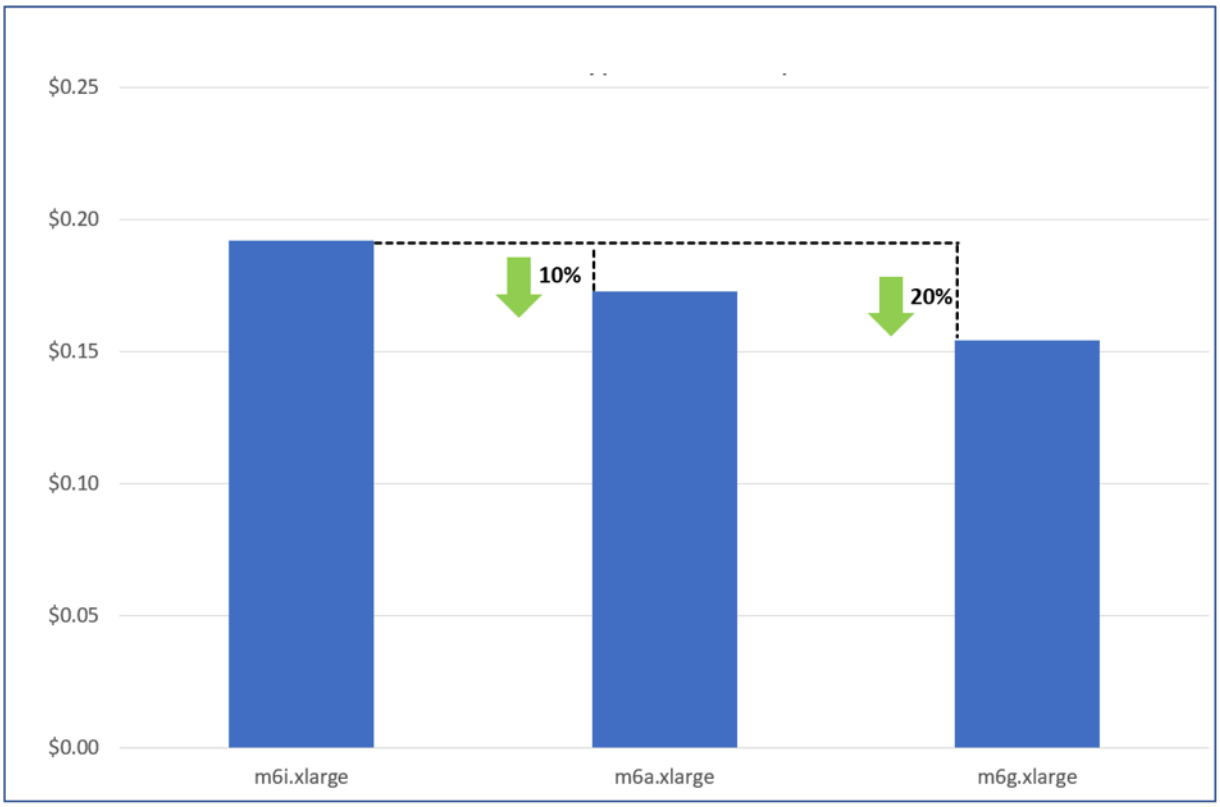
Windows Server can't run on Graviton processors, which are based on ARM architecture. In fact, Windows Server operates only on x86 processors. Although you can't achieve a 40 percent price performance boost by using Graviton-based instances for Windows Server, you can still use Graviton processors with specific Microsoft workloads. For example, newer versions of .NET can run on Linux. That means these workloads can use ARM processors and benefit from faster, more affordable Graviton EC2 instances.

The following example illustrates the hourly pricing for a Graviton instance running in the `us-east-1` Region.

Intel instance	Hourly price	Graviton instance	Hourly price	% difference
c6i.xlarge	$0.17	c6g.xlarge	$0.136	20%
m6i.xlarge	$0.192	m6g.xlarge	$0.154	20%
r6i.xlarge	$0.252	r6g.xlarge	$0.2016	20%

> **Note**
>
> Pricing is based on on-demand hourly pricing in the `us-east-1` Region.

The following chart compares the prices of M series instances.



Understand price performance differences across EC2 generations

One of the most consistent characteristics of Amazon EC2 is that each new generation offers better price performance than its predecessor. As the following table shows, the price of newer generation EC2 instances decreases with each subsequent release.

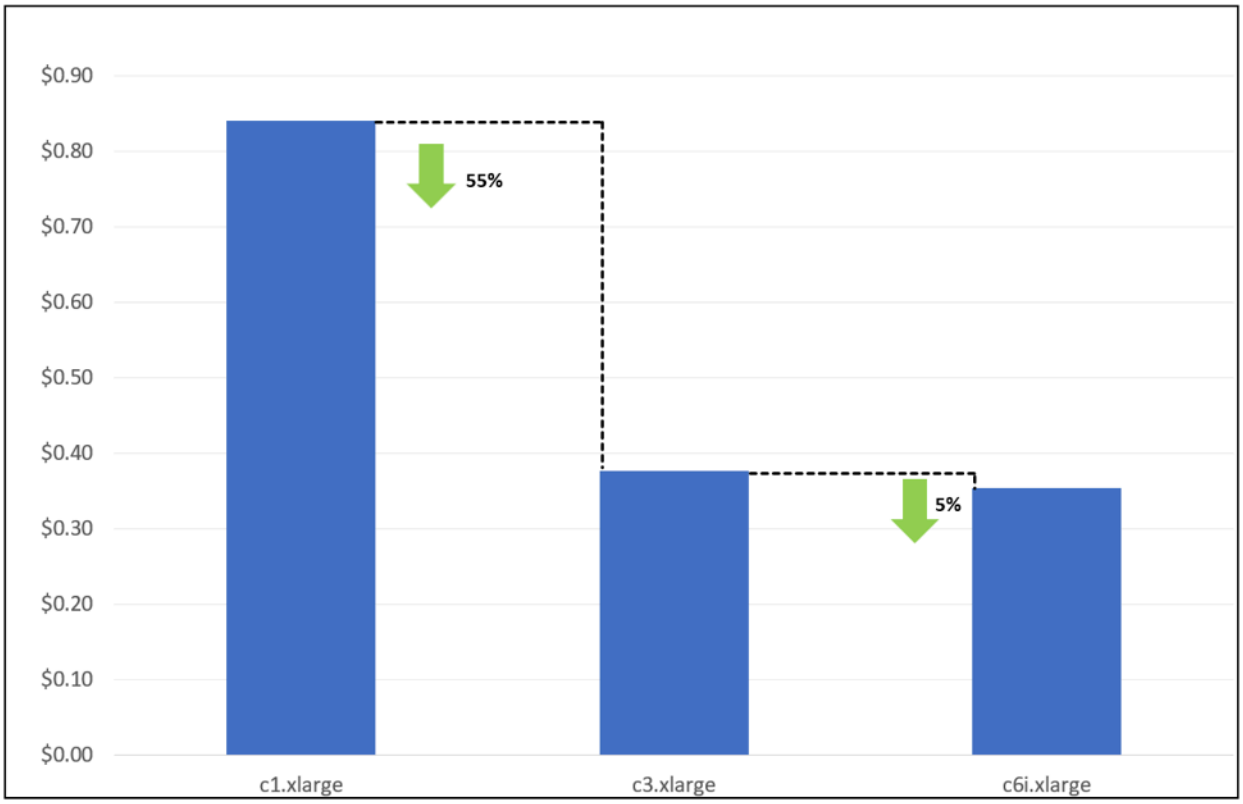
Compute optimized instance	Hourly price	General purpose instance	Hourly price	Memory optimized instance	Hourly price
C1.xlarge	$0.52	M1.xlarge	$0.35	r1.xlarge	n/a
C3.xlarge	$0.21	M3.xlarge	$0.266	r3.xlarge	$0.333

Compute optimized instance	Hourly price	General purpose instance	Hourly price	Memory optimized instance	Hourly price
C5.xlarge	$0.17	M5.xlarge	$0.192	r5.xlarge	$0.252

Note

Pricing is based on on-demand hourly pricing in the `us-east-1` Region.

The following chart compares the costs of the different generations of C series instances.



However, the 6th generation of instances are the same price as the 5th generation, as the following table shows.

Compute optimized instance	Hourly price	General purpose instance	Hourly price	Memory optimized instance	Hourly price
C5.xlarge	$0.17	M5.xlarge	$0.192	r5.xlarge	$0.252
C6i.xlarge	$0.17	M6i.xlarge	$0.192	r6i.xlarge	$0.252

Note

Pricing is based on on-demand hourly pricing in the `us-east-1` Region.

Despite having the same cost, the newer generation provides superior price performance due to faster processors, enhanced networking throughput, and increased Amazon Elastic Block Store (Amazon EBS) throughput and IOPS.

One of the most significant price performance improvements is the enhancement of the X2i instance. This generation of the instance offers up to 55 percent greater price performance than the previous generation. As the following table shows, the x2iedn demonstrates improvement in every performance aspect (all at the same price as the preceding generation).

Instance	Hourly price	vCPUs	RAM	Processor speed	Instance storage	Networking	Amazon EBS throughput	EBS IOPS
x1e.2xlarge	$1.66	8	244	2.3 GHz	237GB SSD	10 Gbps	125 MB/s	7400
x1iedn.2xlarge	$1.66	8	256	3.5 GHz	240GB NVMe SSD	25 Gbps	2500 MB/s	65000

> **Note**
>
> Pricing is based on on-demand hourly pricing in the `us-east-1` Region.

Example scenarios

Consider the example of an analytics company that tracks delivery vehicles and wishes to improve its SQL Server performance. After a MACO SME reviews this company's performance bottlenecks, the company transitions from x1e.2xlarge instances to x2iedn.xlarge instances. The new instance size is smaller, but the enhancements to the x2 instances enable increased SQL Server performance and optimization through the use of Buffer Pool Extensions. This enables the company to downgrade from SQL Server Enterprise edition to SQL Server Standard edition. It also enables the company to reduce its SQL Server licensing from 8 vCPUs to 4 vCPUs.

Before optimization:

Server	EC2 instance	SQL Server edition	Monthly cost
ProdDB1	x1e.2xlarge	Enterprise	$3,918.64
ProdDB2	x1e.2xlarge	Enterprise	$3,918.64
Total			**$7,837.28**

After optimization:

Server	EC2 instance	SQL Server edition	Monthly cost
ProdDB1	x2iedn.xlarge	Standard	$1,215.00
ProdDB2	x2iedn.xlarge	Standard	$1,215.00
Total			**$2,430.00**

All combined, the change from x1e.2xlarge instances to x2iedn.xlarge instances enables the company in the example scenario to save $5,407 per month on its production database servers. This reduces the total cost of the workload by 69 percent.

> **Note**
>
> Pricing is based on on-demand hourly pricing in the `us-east-1` Region.

Migrate to newer instances

Older generations of Amazon EC2 run on the Xen hypervisor, while newer generations operate on the AWS Nitro System. The Nitro System delivers almost all of the compute and memory resources of the host hardware to your instances. This results in improved overall performance. There are special considerations when migrating from Xen to Nitro-based instances. For example, AWS Windows AMIs are configured with default settings and customizations used by the Microsoft installation media. The customizations include drivers and configurations that support the latest generation instance types (instances built on the Nitro System).

If you're launching instances from custom Windows AMIs or from Windows AMIs provided by Amazon that were created before August 2018, we recommend that you complete the steps from Migrate to latest generation instance types in the Amazon EC2 documentation.

Use burstable instances

While burstable instances are a good way to save on compute costs, we recommend that you avoid them in the following scenarios:

- Minimum specs for Windows Server with the Desktop Experience require 2 GB of RAM. Avoid using t3.micro or t3.nano instances with Windows Server because they lack the minimum amount of RAM.
- If your workload is spiky but doesn't stay idle long enough to build burst credits, using normal EC2 instances is more efficient than using burstable instances. We recommend monitoring your CPU credits to verify this.
- We recommend that you avoid using burstable instances with SQL Server in most scenarios. The licensing for SQL Server is based on the number of vCPUs assigned to an instance. If SQL Server is idle the majority of the day, you would be paying for SQL licenses that you aren't fully utilizing. In these scenarios, we recommend that you consolidate multiple SQL Server instances onto a larger server.

Next steps

We recommend that you take the following next steps to optimize your costs for Amazon EC2 Windows instances:

- Use the latest-generation EC2 instance for the best price performance.
- Use EC2 instances with AMD processors for a ten percent reduction in compute costs.
- Maximize resource utilization by choosing an EC2 instance type that matches your workload.

The following table shows examples of typical starting points for Windows workloads. Additional options are available, such as instance storage volumes to enhance SQL Server workloads or EC2 instances with much larger vCPU-to-RAM ratios. We recommend that you test your workloads thoroughly and use monitoring tools like AWS Compute Optimizer to help make necessary adjustments.

Workload	Typical	Optional
Active Directory	T3, M6i	R6i
File servers	T3, M6i	C6i
Web servers	T3, C6i	M6i, R6i
SQL Server	R6i	x2iedn, X2iezn

If you must change your EC2 instance type, the process typically involves just a simple server reboot. For more information, see Change the instance type in the Amazon EC2 documentation.

Before you change your instance type, we recommend that you consider the following:

- You must stop your instances backed by Amazon EBS before you can change its instance type. Be sure to plan for downtime while your instance is stopped. Stopping the instance and changing its instance type might take a few minutes, and restarting your instance might take a variable amount of time depending on your application's startup scripts. For more information, see Stop and start your instance in the Amazon EC2 documentation.

- When you stop and start an instance, AWS moves the instance to new hardware. If your instance has a public IPv4 address, AWS releases the address and gives your instance a new public IPv4 address. If you require a public IPv4 address that doesn't change, use an Elastic IP address.
- You can't change the instance type if hibernation is enabled on the instance.
- You can't change the instance type of a Spot Instance.
- If your instance is in an Auto Scaling group, Amazon EC2 Auto Scaling marks the stopped instance as unhealthy, and may terminate it and launch a replacement instance. To prevent this, you can suspend the scaling processes for the group while you're changing the instance type. For more information, see Suspend and resume a process for an Auto Scaling group in the Amazon EC2 Auto Scaling documentation.
- When you change the instance type of an instance with NVMe instance store volumes, the updated instance might have additional instance store volumes, because all NVMe instance store volumes are available even if they aren't specified in the Amazon Machine Image (AMI) or instance block device mapping. Otherwise, the updated instance has the same number of instance store volumes that you specified when you launched the original instance.

Additional resources

- Amazon EC2 instance types (AWS documentation)
- AWS Optimization and Licensing Assessment (AWS documentation)

Bring licenses for Windows and SQL Server workloads

Overview

If you have significant investments in Microsoft workloads and existing enterprise licensing agreements, you can choose from several AWS options to support these workloads, including license included (provided by AWS) and Bring Your Own License (BYOL) options. You can use Amazon EC2 Dedicated Hosts to fully take advantage of existing Microsoft licensing agreements and bring Windows Server to AWS. This can save you up to 50 percent on Amazon EC2 instance costs. Since Windows licenses account for approximately half of the instance costs, bringing Windows Server to AWS on Dedicated Hosts can result in substantial cost savings. Because Windows Server can't be brought to default (shared) tenancy, Dedicated Hosts is the ideal choice if you want to use your existing licenses for Windows Server on AWS.

Dedicated Hosts are not just for Windows Server BYOL instances. They also offer you the flexibility to match your on-premises licensing for existing SQL Server workloads. Dedicated Hosts expose the physical cores of the underlying server, and enable you to license SQL Server at the physical core level. This isn't possible in default (shared) tenancy where SQL Server licensing is based on the number of virtual CPUs allocated to the instance. This feature enables you to license SQL Server workloads on AWS in a way that's consistent with your on-premises licensing strategy. Consequently, you can save up to 50 percent on SQL Server licensing costs compared to default (shared) tenancy, in addition to cost savings on instance costs, by using eligible Windows licenses. For more information about this scenario, see the Understand SQL Server licensing section of this guide.

Amazon EC2 Dedicated Hosts

An Amazon EC2 Dedicated Host is essentially the same EC2 host that AWS uses to run its EC2 compute offerings. The difference is that these hosts are fully dedicated to a single customer and provide exclusive access to the underlying physical infrastructure. You can use Dedicated Hosts to run your instances on hardware that's entirely dedicated to your use, instead of sharing resources with other AWS customers. This gives you greater control over cloud resources and enables you to reduce costs by bringing your own software licenses, such as Windows Server and SQL Server, to AWS.

Keep in mind the following:

- A Dedicated Host is a physical server fully dedicated to a single customer. You get visibility into the sockets and physical cores of the Dedicated Host so that you can address licensing compliance requirements, such as per-socket, per-core, or per-VM software licensing agreements.
- Dedicated Hosts that can support multiple instance sizes of the same instance family are known as heterogeneous Dedicated Hosts. These instance families include T3, A1, C5, M5, R5, C5n, R5n, and M5n. In contrast, other instance families support only one instance size on the same Dedicated Host. These are called homogeneous Dedicated Hosts.
- Dedicated Hosts are billed on a per-host basis. This means that you are charged per Dedicated Host regardless of how many instances are running on it. Dedicated Host pricing varies based on the instance family, Region, and payment option selected. You can choose the optimal configuration for your workload to achieve your desired performance and cost outcomes.

This diagram illustrates the differences between shared tenancy instances and Dedicated Hosts.

Homogenous Dedicated Hosts

Consider a scenario where an M6i Dedicated Host is used. M6i and R6i Dedicated Hosts have two sockets, 64 physical cores, and support instance types of the same size. These are called homogenous Dedicated Hosts. This means that the number of instances that you can launch on a single M6i Dedicated Host depends on the instance size.

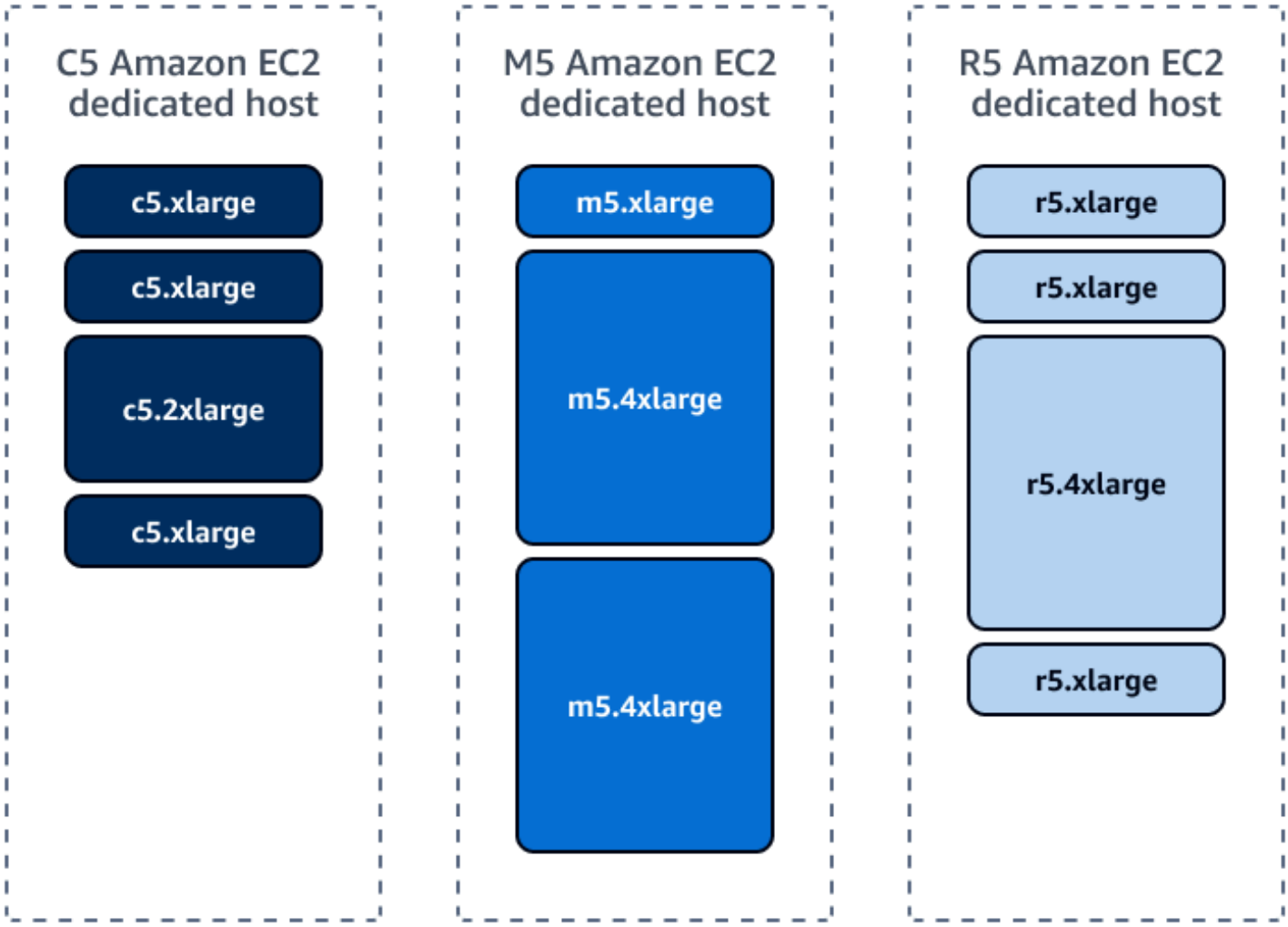
For example:

- In the case of xlarge (4 vCPUs), you can launch a maximum of 32 m6i.xlarge instances on this Dedicated Host.
- In the case of 8xlarge (32 vCPUs), you can launch a maximum of 4 m6i.8xlarge instances on this Dedicated Host.
- In the case of metal (128 vCPUs), you can launch a maximum of 1 m6i.metal instance on this Dedicated Host.

The following diagram shows the Dedicated Host options for M6 instances.

Heterogenous Dedicated Hosts

Dedicated Hosts that support multiple instance sizes on the same host are referred to as a heterogenous Amazon EC2 Dedicated Hosts. The following diagram shows an example of C5, M5, and R5 Dedicated Hosts with various instance sizes, such as 2xlarge, xlarge, and 4xlarge.



Dedicated Host management

We recommend that you consider the following in regard to managing Amazon EC2 Dedicated Hosts:

- To take full advantage of Dedicated Hosts, you can share a single host between multiple accounts within your organization. Host sharing enables resource optimization and can result in cost savings by using every available slot on the host. By sharing a Dedicated Host between business units, you can centralize your IT infrastructure and improve resource utilization, while still maintaining separation between workloads. If you're part of an organization in AWS Organizations and sharing is enabled within your organization, then consumers in your organization are automatically granted access to the shared Dedicated Host. Otherwise, consumers receive an invitation to join the resource share and are granted access to the shared Dedicated Host after accepting the invitation.

- You can run Windows Server 2022 on Dedicated Hosts under the license-included model, as Windows Server 2019 is the latest version where you can BYOL. If you want to use Windows Server 2022 on Dedicated Hosts, you must use Windows Server 2022 license-included instances.
- AWS License Manager is a comprehensive solution for managing software licenses from various vendors across AWS and on-premises environments. If you use License Manager, you can gain greater visibility and control over how software licenses are used, leading to cost savings and improved compliance. You can use License Manager to set rules to emulate your unique licensing terms. This enables you to enforce those rules and prevent license misuse. This can reduce the risk of noncompliance and improve license management processes.
- You can use License Manager to automate the placement, release, and recovery of hosts by using host resource groups. This can increase productivity and reduce management overhead. License Manager also provides a centralized view of license usage across AWS and on-premises environments based on licensing rules, making it easy to manage incremental licensing purchases, compliance, and vendor audits across your organization. Furthermore, License Manager integrates with AWS Organizations and AWS Resource Access Manager (AWS RAM) to share license configurations across accounts and Regions. This enables you to create reports for your entire environment based on a schedule and manage licensing rules centrally in one AWS account. Ultimately, this can improve governance and reduce complexity.
- When designing high availability for Dedicated Hosts within a single Region, make sure that you've allocated a minimum of two Dedicated Hosts in a minimum of two Availability Zones for production-critical workloads. For more information, see the Amazon EC2 Dedicated Hosts for Microsoft Windows on AWS reference deployment.
- For each Dedicated Host instance family, there is a limit on the number of instances that you can run for each instance size. For more information, see Dedicated Hosts Configuration Table in the Amazon EC2 documentation.

AWS licensing options

Licenses are classified into the following primary categories:

- **License included** – This licensing option enables you to purchase and use licenses on demand, paying solely for what you use. It's ideal for use cases where you're seeking flexibility in your licensing usage and wish to avoid upfront costs. You can choose from a variety of Windows Server, SQL Server, and other Microsoft products.
- **BYOL products with License Mobility** – If you already have existing licenses and want to use them in the cloud, this licensing option allows you to bring your own licenses to the cloud

through the Microsoft License Mobility program. Products with license mobility, such as SQL Server with Software Assurance (SA), can be brought to either shared or dedicated tenancy. This reduces AWS instance costs.

- **BYOL products without License Mobility** – For Microsoft products like Windows Server that lack License Mobility, AWS provides dedicated options for using these products in the cloud. Additionally, Dedicated Hosts enable licensing at the physical core level, making it possible to save 50 percent or more on the licenses needed to run your workloads. Dedicated Hosts are an excellent choice for stable and predictable workloads that run most of the time.

Bringing Windows Server licenses

Bringing your own Windows licenses is one of the most effective strategies for license optimization because it enables you to take advantage of existing investments and reduce your AWS expenses. Specific BYOL scenarios don't require SA or License Mobility benefits, but Amazon EC2 dedicated infrastructure is always necessary. To be eligible, you must have purchased perpetual licenses before October 1, 2019 or added them as a true-up under an active Enterprise Enrollment effective before October 1, 2019. In these specific BYOL scenarios, you can upgrade only licenses to versions available prior to October 1, 2019. For example, if you dropped SA in 2017, you have the rights to deploy only up to Windows Server 2016, not 2019. However, 2019 is the last version eligible for BYOL to AWS. For more information, see Licensing – Windows Server in the AWS documentation.

Bringing licenses can significantly impact the cost of running Microsoft workloads on AWS. When you bring your own licenses, you're not required to pay additional licensing costs for the instances running in the cloud, which can lead to considerable cost savings.

The following table show the on-demand monthly cost of running a single c5.xlarge instance 24/7 on various configurations.

Configuration	Monthly cost (USD)
Windows Server + SQL Server Enterprise edition	$1,353.00 (LI)
Windows Server + SQL Server Standard edition	$609.00 (LI)
Windows Server only	$259.00 (LI)

Configuration	Monthly cost (USD)
Compute only (Linux)	$127.00

You can use your existing licenses to reduce licensing costs and save money on your overall AWS bill.

To be eligible for BYOL on Amazon EC2 Dedicated Hosts, you must bring your own software licenses, such as for Windows Server and SQL Server. BYOL allows you to use your existing licenses on AWS and can result in cost savings. To bring your own licenses, you must have the license entitlements from the software vendor and must also provide the installation media or image for the software. The installation media or image can be used to launch instances on Dedicated Hosts. To learn more about creating a BYOL AMI, see How to create Windows Server Bring-Your-Own-License AMIs from on-premises with VM Import/Export in the Microsoft Workloads on AWS blog.

> **Note**
>
> A license type set to **Auto** is the equivalent of an AWS license-included option. This option can result in unwanted on-demand spending. You will need to switch licensing types.

Cost optimization scenarios

Right sizing and optimizing licenses is a key component of cost optimization on AWS. If you implement the right strategies, you can reduce licensing costs, maintain compliance, and achieve the best possible value from your licensing investment by using Amazon EC2 Dedicated Hosts and the BYOL option.

This section covers the following example scenarios:

- Cost savings with T3 Dedicated Hosts
- Comparing shared tenancy to Dedicated Hosts with SQL Server BYOL
- Highly available SQL Server deployments

Cost savings with T3 Dedicated Hosts

T3 Dedicated Hosts differ from other Amazon EC2 Dedicated Hosts that traditionally provide fixed CPU resources. T3 Dedicated Hosts, in contrast, support burstable instances that are capable of sharing CPU resources, providing baseline CPU performance, and bursting when needed. Sharing CPU resources, also known as oversubscription, is what enables a single T3 Dedicated Host to support up to four times more instances than comparable general-purpose Dedicated Hosts.

T3 Dedicated Hosts drive a lower TCO by delivering higher instance density than any other Amazon EC2 Dedicated Host. Burstable T3 instances enable you to consolidate a higher number of instances with low-to-moderate average CPU utilization on fewer hosts than ever before. T3 Dedicated Hosts also offer smaller instance sizes in a greater number of vCPU and memory combinations than other Amazon EC2 Dedicated Hosts. Smaller instance sizes can contribute to lower TCO and help deliver consolidation ratios equivalent to or greater than on-premises hosts.

T3 Dedicated Hosts are best suited for running BYOL software with low-to-moderate CPU utilization and eligible per-socket, per-core, or per-VM software licenses, including Microsoft Windows desktop, Windows Server, SQL Server, and Oracle Databases.

Use T3 Dedicated Hosts to reduce Windows Server Datacenter licenses (per core)

In on-premises environments, you're taking advantage of the fact that you can easily oversubscribe your physical CPUs on VMware hosts and achieve high levels of consolidation.

Consider the following example. You're currently using 10x36 core, 384 GB RAM VMware hosts in an on-premises environment. Additionally, each host is running 96x2 vCPU, 4 GB RAM Windows Server virtual machines with low average CPU utilization.

You can now achieve much higher levels of consolidation by moving your virtual machines to T3 Dedicated Hosts, which have twice the amount of RAM compared to your current on-premises VMware hosts. You can run the same number of servers on T3 Dedicated Hosts with 50 percent less host cost. This can help you reduce Windows Server licensing costs by 33 percent. The following table highlights the savings of using T3 Dedicated Hosts.

	On-premises VMware hosts	T3 Dedicated Hosts	Savings
Physical servers	10	5	

	On-premises VMware hosts	T3 Dedicated Hosts	Savings
Physical cores per host	36	48	
RAM per host (GB)	384	768	
2 vCPU, 4 GB RAM VMs per host	96	192	
Total number of VMs	960	960	
Total Windows Server Datacenter licenses (per core) = (Number of servers * Physical core count)	10 * 36 = 360	5 * 48 = 240	33%

Comparing shared tenancy to Dedicated Hosts with SQL Server BYOL

Consider a practical example to demonstrate the value of Amazon EC2 Dedicated Hosts. In this scenario, an organization runs a SQL Server workload in on-premises environment with 240 cores and wants to deploy the same workload cost-effectively on AWS. If this organization brings their own licenses (BYOL), they continue to pay for SA and reducing the number of cores directly affects their costs.

The following diagram compares AWS savings between Microsoft entitlements and SQL Server.

Microsoft entitlements (Enterprise Agreements)	
	Number of cores
SQL Server Enterprise edition	208
SQL Server Standard edition	32
Total SA cost	**$341,000**

SQL Server savings with AWS	
AWS shared vCPUs	**AWS BYOL/Dedicated Hosts cores**
120	96
20	-
$197,418	**$151,355**

By right sizing the instances on AWS shared tenancy, you can reduce SQL Server licenses down to 140 cores. This results in SA costs of $197,000.

Amazon EC2 Dedicated Hosts enable you to license SQL Server at the physical core level. This isn't possible in shared tenancy where SQL Server licensing is based on the number of vCPUs allocated to the instance. Consequently, by using two R5 Dedicated Hosts with 48 cores each, you only need to cover 96 cores instead of the 140 vCPUs required on shared tenancy. By deploying R5 Dedicated Hosts and licensing the workloads at the physical level, you can bring the required number of SQL Server Enterprise edition licenses down to 96 cores. This means that you can deploy as many as 192 cores (accounting for hyper-threading) of SQL Server workloads, while still meeting licensing requirements and achieving significant cost savings.

In this case, the organization pays approximately $341,000 annually in SA costs. After right sizing on shared tenancy, they reduce costs to $197,000 with 140 vCPUs. Amazon EC2 Dedicated Hosts further reduces costs to $151,000 (an approximately 56 percent decrease).

Highly available SQL Server deployments

This example analyzes how cost can influence a SQL Server deployment on AWS with various licensing considerations. Suppose an organization needs to deploy six SQL Server Enterprise servers on AWS to support three applications. These servers require high availability and have 16 vCPUs and 256 GB of RAM each. See the following scenario details:

- **Server** – SQL Server
- **Operating system edition** – Windows Server Datacenter 2019
- **SQL Server edition** – SQL Server Enterprise 2019
- **vCPU** – 16
- **Memory (GB)** – 256
- **Quantity** – 6

To optimize costs on AWS without sacrificing performance, we recommend that you right size instances based on CPU, memory, network, and disk (IOPS/BW) utilization. After right sizing the workloads, place them on the x2iedn.4xlarge instance type, which offers 16 vCPUs. However, this instance type also includes twice the necessary memory for the workloads. Further optimization is still possible.

Scenario 1

An organization deploys six SQL Server Enterprise servers on AWS shared tenancy by using the license-included option for both Windows and SQL Server. With this option, the cost of the

Windows and SQL Server licenses is incorporated into the instance price. See the following scenario details:

- **Shared tenancy (instance)** – x2iedn.4xlarge
- **Hourly cost (USD)** – $10.0705
- **Monthly cost per unit (USD)** – $7,351.47
- **Number of servers** – 6
- **CPU** – 16
- **Memory** – 512
- **Monthly cost for 6 servers** – $44,108

Scenario 2

An organization has SA and BYOL for SQL Server on shared tenancy. This means that the organization uses the license-included option for Windows, but provides their own SQL Server licenses based on the number of vCPUs allocated to the instance. Since the organization has six SQL Server Enterprise servers with 16 vCPUs each, a total of 96 vCPUs is required. See the following scenario details:

- **Shared tenancy (instance)** – x2iedn.4xlarge
- **Hourly cost (USD)** – $4.0705
- **Monthly cost per unit (USD)** – $2971.47
- **Number of servers** – 6
- **CPU** – 16
- **Memory** – 512
- **BYOL cores** – 96
- **Monthly cost for 6 servers** – $17,828

By bringing their own SQL Server licenses with SA, the organization in this scenario can achieve cost savings compared to using the license-included option for SQL Server. The precise cost savings depend on the specific licensing agreement's pricing and terms. In this scenario, AWS costs decrease by $26,280 per month when bringing SQL Server Enterprise licenses to AWS.

Scenario 3

An organization has BYOL for both Windows and SQL Server on Amazon EC2 Dedicated Hosts. This means that the organization will assign licenses at the physical core level, enabling them to license only the host's physical cores. Licensing at the physical core level allows you to deploy the maximum number of instances without affecting the required licenses. This licensing model is commonly used with Windows Server Datacenter and SQL Server Enterprise edition.

This scenario uses two X2iezn Amazon EC2 Dedicated Hosts. Each host has 24 physical cores and 48 vCPUs. This provides adequate capacity for the six SQL Server Enterprise servers with 16 vCPUs and 256 GB of RAM each. See the following scenario details:

- **Number of dedicated hosts** – 2
- **Instance family** – x2iezn
- **Hourly cost (USD)** – $11.009
- **Monthly cost per unit (USD)** – $8,036
- **Physical core** – 48
- **Available vCPU** – 96
- **Windows Server core licenses required** – 24
- **Licenses required for SQL Server Enterprise cores** – 24
- **Monthly cost** – 16,073

The total cost for two X2iezn family Amazon EC2 Dedicated Hosts is $16,073 per month. For more information about pricing, see the AWS Pricing Calculator estimate for this scenario. The organization in this scenario can save $1,755.65 per month by bringing their Windows licenses. If they use Amazon EC2 Dedicated Hosts, they can also reduce the number of required SQL Server licenses. In shared tenancy, they would need 96 SQL Server Enterprise licenses to cover the six SQL Server Enterprise servers with 16 vCPUs each. However, by using Amazon EC2 Dedicated Hosts and licensing at the physical core level, they can reduce the number of required licenses to 48 cores.

The following details compare the costs from example 3 and show how much you can save by deploying workloads on Amazon EC2 Dedicated Hosts with the BYOL option compared to other scenarios.

- **On-premises server** – SQL Server
- **vCPU** – 16

- **Memory** – 256
- **Number of servers** – 6
- **Monthly cost for scenario 1: Windows (LI) + SQL Server Enterprise (LI)** – $44,108
- **Monthly cost for scenario 2: Windows (LI) + SQL Server Enterprise (BYOL)** – $17,828
- **Monthly cost for scenario 3: Windows (LI) + SQL Server Enterprise (BYOL) on Amazon EC2 Dedicated Host** – $16,073

> **Note**
>
> Cost is based on on-demand pricing. You can further reduce costs by using Savings Plans or Dedicated Reserved Instances. These options offer a flexible pricing model with significant cost savings compared to on-demand pricing. With these plans, you can commit to a term of one or three years. For more information, see the Optimize spending for Windows on Amazon EC2 section of this guide.

Consider the following payment options for Amazon EC2 Dedicated Hosts:

- Dedicated Hosts (Amazon EC2 documentation)
- Dedicated Host Reservations (Amazon EC2 documentation)
- Savings Plans (Amazon EC2 documentation)

The AWS Pricing Calculator now supports Dedicated Host pricing. This can help you choose the appropriate underlying Dedicated Host.

Cost optimization recommendations

We recommend that you take the following next steps to optimize your costs by using the AWS Cost Explorer:

1. Enable Cost Explorer.
2. Use Cost Explorer to view and analyze the costs and usage of your Amazon EC2 Dedicated Host deployments.
3. Validate that you're running BYOL. You can display the following platform details and usage operation values on the instances or AMI pages in the Amazon EC2 console, or in the response that's returned by the `describe-images` or `describe-instances` command.

- **Platform details:** Windows , **Usage operation:** RunInstances:0002 (License included)
- **Platform details**: Windows BYOL, **Usage operation:** RunInstances:0800

Additional resources

- Eligible license types for license type conversion (AWS License Manager documentation)
- AWS License Manager & Dedicated Host workshop (AWS License Manager Workshop)
- Amazon EC2 Dedicated Hosts FAQs (AWS documentation)
- How to create Windows Server Bring-Your-Own-License AMIs from on-premises with VM Import/Export (Microsoft Workloads on AWS blog)
- VM Import/Export (AWS documentation)
- Amazon Web Services and Microsoft: Frequently Asked Questions (AWS documentation)
- License type conversions in License Manager (AWS License Manager documentation)
- Deploying highly-available SQL Server on Amazon EC2 Dedicated Hosts (AWS Cloud Operations & Migrations Blog)

Optimize spending for Windows on Amazon EC2

Overview

One of the top concerns about migrating servers to AWS is the infrastructure costs. It's true that one of the benefits of the cloud is paying for the resources on demand, but there are production workloads that need to be available 24/7/365. Savings Plans are designed to save money on your steady-state AWS usage across EC2 instances, AWS Lambda, and AWS Fargate.

Savings Plans offer a flexible pricing model and can help you reduce pricing on Amazon EC2, Fargate, Lambda, and Amazon SageMaker AI usage in exchange for a commitment to a consistent amount of usage (for example, $10/hour). You commit to a consistent amount of hourly compute spending over one or three years and, in exchange, you receive a discount for that usage.

You can choose from three different payment options with Savings Plans:

- The **No Upfront** option doesn't require any upfront payment, and your commitment is charged purely on a monthly basis.

- The **Partial Upfront** option offers lower prices on Savings Plans. You are charged at least half of your commitment upfront and the remaining is charged on a monthly basis.
- The **All Upfront** option offers the lowest prices and your entire commitment is charged in one payment.

You can track your Savings Plans expirations and upcoming queued Savings Plans in AWS Cost Explorer. You can use Savings Plans alerts to receive advance email alerts 1, 7, 30, or 60 days before your plan expiration date, or when a commitment is queued for purchase. These notifications also alert you on the expiration date. You can send notifications to up to 10 email recipients.

Understanding Savings Plans

Every type of compute usage has an on-demand rate and a Savings Plans rate. If you commit to $10/hour of compute usage, then you get Savings Plans prices on all usage up to $10 at the Savings Plans rate. Any usage beyond the compute spending commitment is charged at regular on-demand rates. You can get started with Savings Plans by using Cost Explorer in the AWS Management Console.

You can easily make a commitment to Savings Plans by using the recommendations provided in Cost Explorer to realize the biggest savings. The recommended hourly commitment is based on your historical on-demand usage and your choice of plan type, term length, and payment option. Savings Plans is first applied to the account that purchased the plan, and then it's shared to other accounts in the consolidated billing family.

> **Note**
>
> The Savings Plans sharing option in AWS Organizations is enabled by default. You can decline this option in the AWS Billing console of the payer account. You can visit your Recommendations page to see the Savings Plans that AWS recommends to help you save on eligible usage. These recommendations can be refreshed at any time to make it easy for you to purchase the optimal Savings Plans.

Compute Savings Plans

Compute Savings Plans provide the most flexibility and help reduce your costs. These plans automatically apply to EC2 instance usage regardless of instance family, size, Availability Zone,

Region, operating system, or tenancy. They also apply to Fargate or Lambda usage. For example, with Compute Savings Plans, you can change from C4 to M5 instances, shift a workload from EU (Ireland) to EU (London), or move a workload from EC2 to Fargate or Lambda at any time. You automatically continue to pay the Savings Plans price.

EC2 Instance Savings Plans

EC2 Instance Savings Plans provide the deepest discounts in exchange for a commitment to usage of individual instance families in a Region (for example, committing to a consistent level of M5 usage in N. Virginia). This automatically provides you with discounts on the on-demand price of the selected instance family in that Region regardless of Availability Zone, size, operating system, or tenancy. EC2 Instance Savings Plans allow you to change your usage between instances within a family in that Region. For example, you can move from c5.xlarge running Windows to c5.2xlarge running Linux, and automatically benefit from the Savings Plans prices.

Both Compute and EC2 Instance Savings Plans apply to EC2 instances that are a part of Amazon EMR, Amazon Elastic Kubernetes Service (Amazon EKS) and Amazon Elastic Container Service (Amazon ECS) clusters. Amazon EMR, Amazon EKS, and Amazon ECS charges aren't covered by Savings Plans, but the underlying EC2 instances are. EC2 Instance Savings Plans are applied before Compute Savings Plans because Compute Savings Plans have broader applicability.

> **Note**
>
> You can't change a Savings Plans easily after you've made a commitment. We recommend that you plan carefully before making a commitment to either Savings Plans option. Savings Plans offer lower prices compared to on-demand pricing in exchange for a commitment, and can't be cancelled during the term.

Hourly commitment example

If you purchase a Savings Plans, you make an hourly monetary commitment to the term of the plan. If you commit to $10/hour of compute usage, the Savings Plans pricing is automatically applied on all usage up to $10 dollars every hour. Any usage beyond the commitment is charged at the regular on-demand rates. You can use the Savings Plans purchase recommendations tool in Cost Explorer to get recommended commitments that can maximize your savings. The hourly financial commitment for a specific plan can't be modified for the term of the plan. If you want an increased commitment after analyzing usage, then you can purchase an additional Savings Plans to cover excess usage.

Benefits of Savings Plans

Compared to Reserved Instances, Savings Plans offer a more flexible pricing model that can save you money while you take advantage of the broader selection of compute options offered by Savings Plans. Savings Plans offer discounts, even as your compute needs change. This can help you keep up with your ever changing dynamic environment without incurring any additional management overhead. Here are some other benefits of using Savings Plans:

- **Easy to use** – Receive automatic discounts in exchange for monetary commitment.
- **Flexibility** – A single commitment that applies across multiple usage types.
- **Potential savings** – There are a variety of ways to save. Consider the following examples:
  - 60 percent savings on Windows Server workloads using Compute Savings Plans (d2.8xlarge, 3 years, all upfront, windows, shared tenancy, us-east-2)
  - 73 percent savings on Windows Server workloads using EC2 Instance Savings Plans (d2.8xlarge, 3 years, all upfront, windows, shared tenancy, us-east-2)
  - 28–41 percent savings on non-exotic instance types (t3 family, 3 years, all upfront, windows, shared tenancy, us-east-2)
  - 25–40 percent average savings for Windows Servers

> **Note**
>
> EC2 Instance Savings Plans offer a greater discount than Compute Savings Plans because of the reduced flexibility. You commit to usage for a discounted price.

Every type of compute usage has a Savings Plans rate and on-demand rate. The following table shows the Savings Plans and on-demand rates for every operating system type. You're charged the Savings Plans rates on the committed usage and any usage beyond the commitment is charged at regular on-demand rates.

Instance name	Savings Plans rate	On-demand savings	On-demand rate	Operating system	Region	Payment option	Term length
x2iedn.xlarge	$0.32	61%	$0.83	Linux	US East (N. Virginia)	No upfront	3
x2iedn.xlarge	$2.01	50%	$1.02	Windows	US East (N. Virginia)	No upfront	3
x2iedn.xlarge	$1.02	20%	$2.52	Windows license included + SQL Server Enterprise edition	US East (N. Virginia)	No upfront	3
x2iedn.xlarge	$0.32	61%	$0.83	BYOL	US East (N. Virginia)	No upfront	3

Savings Plans include the operating system, and they have a separate discount for BYOL. They are all broken down in the Compute Savings Plans calculator.

Reserved Instance pricing model

AWS has another pricing model based on commitment known as Reserved Instances. This model can be problematic if your compute changes after you already make a commitment, causing the Reserved Instances to go unused. Savings Plans are designed to offer similar cost reductions as Standard and Convertible Reserved Instances, but with much greater flexibility. Compute Savings Plans provide lower prices on EC2 instance usage regardless of instance family, size, operating system, tenancy, or Region. They also enable maximum flexibility.

The following table can help you choose between Savings Plans or Reserved Instances.

	Reserved Instance	EC2 Instance Savings Plans	Compute Savings Plans
Average 1-year discount	Up to 38%	Up to 29%	Up to 29%
Average 3-year discount	Up to 58%	Up to 73%	Up to 60%
Instance family	Fixed	Fixed	Flexible
Instance size	Fixed (not Linux)	Flexible	Flexible
Geography	1 Region	1 Region	Flexible
Operating system	Fixed	Flexible	Flexible
Service	Amazon EC2 or Amazon RDS	Amazon EC2	Amazon EC2, Fargate, Lambda
Payment options	All, partial, no upfront	All, partial, no upfront	All, partial, no upfront
Instance limits	20 per Availability Zone	No limit	No limit

Note

Savings Plans work by giving you a discount based on an hourly monetary commitment. The hourly financial commitment can't be canceled or changed during the term of your plan, but you can purchase additional Savings Plans to cover additional usage. This enables you to keep a consistent hourly commitment as your fleet grows.

You can use tools like AWS Cost Explorer or AWS Cloud Intelligence Dashboards to track your commitment. Cost Explorer provides a coverage target line that can help your organization plan its Savings Plans coverage strategy. If 75 percent of your workload is steady state, then 75 percent is a good target. This leaves 25 percent of the spending on-demand/variable based on dynamic

workloads. If you need to increase that to 85 percent coverage, you can buy another Savings Plans commitment to increase the hourly monetary commitment.

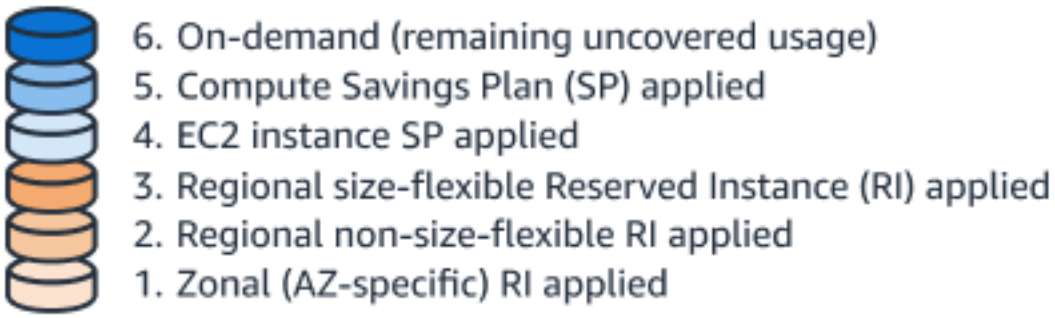
> **Note**
>
> We recommend that you purchase Savings Plans instead of Reserved Instances, but the two commitment models can work together if you already purchased Reserved Instances.

Consider an example where you purchased a Reserved Instance, but you want to start trying a Savings Plans option. There is logic for this combination to apply to your final billing. Here's a hierarchy that you can apply to your AWS accounts:

1. Zonal Reserved Instance applies to the account that owns it. If a Reserved Instance has hours left, it applies to the rest of the organization.
2. Non size-flexible Regional Reserved Instances for Windows apply to matching usage on the account that owns it. Anything that remains rolls out to the rest of the organization.
3. Size-flexible Regional Reserved Instances apply to the account that owns it (smallest instance within the family first and going up to larger instances), and then to the rest of organization.
4. Regional Reserved Instances apply to any unused on-demand capacity reservation.
5. EC2 Instance Savings Plans apply within the account that purchased it.
6. Compute Savings Plans apply within the account that purchased it.

> **Note**
>
> Discounts start with the usage that result in the highest discount and then down to the least discount. Windows instances traditionally have a lower discount potential then Linux for most common instance types (for example, T3, M6, and C5). This means that Linux instances benefit more than Windows instances in most cases.

The following graphic shows the price after dividing Reserved Instances from Savings Plans. Both Compute and EC2 Instance Savings Plans apply to running instances first and then to the unused On-Demand Capacity Reservations.



Cost optimization scenarios

This section covers cost optimization scenarios for Amazon EC2 Dedicated Hosts and Amazon EC2 instances that use a license-included billing model.

Amazon EC2 Dedicated Hosts

Consider a scenario where you're going to migrate your on-premises Windows workloads to AWS. You data center has the following servers:

- Two servers with 16 vCPU and 128 GB RAM
- Two servers with 32 vCPU and 164 GB RAM
- One server with 8 vCPU and 64 GB RAM
- 16 servers with vCPU and 32 GB RAM

Additionally, assume that you can bring your own license to AWS because you have enough licenses to bring over. The following table shows the server instances that you can use in AWS.

Instance type	CPU	RAM	Amount
r5.4xlarge	16	128	2
r5.8xlarge	32	256	2
r5.2xlarge	8	64	1
r5.xlarge	4	32	16
			21

An analysis shows that these 21 virtual machines can be distributed across two Dedicated Hosts with an R5 instance family host. The following table shows the cost of these two Dedicated Hosts.

Dedicated Host on-demand scenario	Upfront pay	1 month	1 year	3 years	AWS Pricing Calculator
On-demand	None	$10,123	$121,475	$364,392	AWS Pricing Calculator estimate
1-year Savings Plans	None	$7,447	$89,362	–	AWS Pricing Calculator estimate
3-year Savings Plans	None	$5,476	$65,712	$197,128	AWS Pricing Calculator estimate
3-year Savings Plans with upfront payment	$84,438	$2,755	$117,499	$183,618	AWS Pricing Calculator estimate

If you have the servers that you want to migrate to AWS, the final price for a 1-year Savings Plans is $89,362 instead of $121,475 for an on-demand price. This represents a 26.5 percent discount after one year. If you're considering staying in AWS for a longer period, then you can choose the 3-year Savings Plans for even deeper cost savings. At the end of three years, you pay $197,128 instead of $364,392. This results in a savings of 46 percent of the total amount after three years.

Amazon EC2 instances with licenses included

Consider a scenario where you're going to migrate a single three-tier application to AWS, and you want to use the licenses provided by AWS. Furthermore, assume that your application works with the following servers:

- Two web servers with two vCPUs and 4 GB RAM

- Two application servers with eight vCPUs and 16 GB RAM
- Two databases servers with 16 vCPUs and 64 GB RAM (using SQL Server Standard edition)

The following table shows the server instances that you can use in AWS.

Instance type	CPU	RAM	Amount
c5.large	2	4	2
c5.2xlarge	8	16	2
r5.2xlarge	8	64	2
			6 server

The following table shows the cost of these servers in AWS.

License included by AWS	Upfront pay	1 month	1 year	3 years	AWS Pricing Calculator
On-demand	None	$3,912	$46,950	$140,849	AWS Pricing Calculator estimate
1-year Savings Plans	None	$3,466	$41,952		AWS Pricing Calculator estimate
3-year Savings Plans with no upfront payment	None	$3,189	$38,264	$114,804	AWS Pricing Calculator estimate

License included by AWS	Upfront pay	1 month	1 year	3 years	AWS Pricing Calculator
3-year Savings Plans with upfront payment	$112,110	None	None	None	AWS Pricing Calculator estimate

If you want to run these servers for production environments (24/7) with on-demand pricing, you pay a monthly cost of $3,912. Paying this monthly cost equates to $46,950 after one year and a total of $140,849 after three years.

If you choose the 1-year Savings Plans with no upfront payment, the monthly cost decreases to $3,466. At the end of the first year, you pay $41,952. This is a total discount of 11 percent. If you choose the 3-year Savings Plans with no upfront payment, the monthly cost decreases to $3,189. At the end of three years, you pay $114,804. That gives you an 18.5 percent savings.

Cost optimization recommendations

Both scenarios help you save money when you plan and forecast your workloads in AWS. It's important to recognize that the discount in the second scenario is less compared with the first scenario. In the second scenario, the license price is included on the price of the cloud server. AWS doesn't offer a discount on the license price, but you can always bring your licenses (under specific scenarios) and AWS can always warranty the best compute/instance price.

We recommend that you do the following to control your AWS spending on compute and instance resources:

- Access recommendations
- Customize recommendations according to your needs
- Review the hourly commitment

Access recommendations

You can use the Amazon EC2 console to access recommendations for your Savings Plans. You can even download your recommendations to review later in CSV format. For more information, see Monitoring your Savings Plans in the Savings Plans documentation.

Customize recommendations according your needs

Open the Amazon EC2 console, expand the **Instances** section, and then choose **Savings Plans**. This page shows you instance and compute pricing before and after making a recommendation. You can also adjust the following factors for your recommendation:

- **Term** – For example, 1–3 years
- **Payment option** – For example, **Upfront**, **Partial Upfront**, or **No Upfront**
- **History** – For example, the last 7, 30 or 60 days

Review the hourly commitment

Using the same example, assume you have an instance that's running 24/7. The recommendation is to use a Savings Plans. According to the size, you have an on-demand price of $120/hour. You have the option to commit $90/hour, but this may vary depending on the Region, instance, and purchase option. In this example, you can save 25 percent compared with the on-demand cost. You can also track your utilization and coverage, if they are under the threshold that you defined, and configure an alert when the budget is going to end.

Review recommendations

We recommend that you review Savings Plans recommendations carefully. AWS won't change anything without your permission. These are recommendations only and it's up to you to apply them or not.

Purchase a plan

Open the Amazon EC2 console, expand the **Instances** section, and then choose **Savings Plans**. Then, choose **Purchase Savings Plans**. Based on your requirements, you can select the following options: term, Region, instance family, hourly commitment, payment option, and even start date. You can choose from Compute Savings Plans, EC2 Instance Savings Plans, and SageMaker AI Savings Plans. For more information, see Purchasing Savings Plans in the Savings Plans documentation.

Get a utilization report

After you purchase a Savings Plans, you can get a utilization report. The report helps you check your utilization, see if the purchased plan is enough to cover and maximize the discount, and cancel or add new discounts. This report can be exported to other formats like CSV. For more information, see Using the utilization report in the Savings Plans documentation.

Follow purchasing best practices

We recommend that you follow these best practices before purchasing Savings Plans:

- Use AWS Trusted Advisor to remove idle EC2 resources.
- Perform any right sizing ahead of Savings Plans purchases.
- Establish an hourly rate that you consistently keep for 30–60 days.
- Purchase a commitment to cover as much of the consistent hourly rate as your organization is comfortable with. Consider fluctuations in demand or season.
- Choose a quarterly review Savings Plans budget to maintain a consistent rate (for example, 70 percent coverage target for Savings Plans coverage). If the rate drops below desired coverage, purchase an additional Savings Plans as a true-up to meet your coverage goal.

Additional resources

- Savings Plans for Amazon EC2 Reserved Instances (AWS Whitepapers)
- Understanding how Savings Plans apply to your AWS usage (Savings Plans documentation)
- Announcing per second billing for EC2 Windows Server and SQL Server Instances (AWS documentation)
- AWS Cost Optimisation Series: Savings Plans Video | Amazon Web Services (YouTube)

Monitor costs using AWS tools

Overview

Cost visibility is a key factor in optimizing costs on AWS. AWS has a number of tools you can use to visualize costs and create alerts in reaction to those costs. These include tools, like AWS Budgets, that help you track and report your spending. This section covers specific ways to monitor your

Windows on AWS spending, so you can track and react accordingly to your budget requirements. This includes adding necessary tags to your Windows EC2 resources. These tags enable you to properly monitor Windows EC2 and other Microsoft services by using AWS Budgets.

By monitoring spending and creating alerts with AWS tools, you can be more informed about current spending, projected spending, and spending anomalies. If you use Savings Plans to help reduce your hourly EC2 instance pricing, we recommend that you view the overall utilization and coverage of the Savings Plans. This can help you ensure that you're continually realizing savings. You can use AWS Cost Explorer to view Savings Plans inventory and get recommendations for additional Savings Plans based on previous usage. You can also track specific spending by using AWS Budgets and setting up AWS Cost Anomaly Detection.

Cost optimization recommendations

We recommend that you take the following next steps to optimize your costs by using AWS Budgets, Cost Explorer, and anomaly detection:

- Tag Windows EC2 resources
- Set up alerts by using AWS Budgets
- Enable Cost Anomaly Detection
- Get a real-time spending analysis
- View license-included spending for Windows by using Cost Explorer

Tag Windows EC2 resources

To effectively monitor your AWS spending, you must establish a tagging strategy for the workloads that you want to monitor. This is important so that you can categorically group resources and get notified on specific spending, as opposed to general usage spending. You can use tagging resources that not only help with cost but can also be used for other purposes such as AWS Systems Manager automation. Additionally, we recommend that you implement some management for required tags.

To track your spending in AWS Budgets, Cost Explorer, and Cost Anomaly Detection, you must ensure that proper tags are in place. You can use tags to set up a specific budget for items matching those tags so that you are alerted when spending increases.

For example, you can use a simple tag like **Key=OS Value=Windows**. This puts all of your Windows instances together into one group that you can track spending for. You can also use tags for other

items, such as Systems Manager. After you create a tag, you must activate the tag for cost tracking. Consider adding an AWS Config rule that monitors for tags that are attached to certain resources. AWS Config can alert you if there are resources running that don't contain the appropriate tags, which provide you with an accurate representation of your Windows EC2 spending.

After you have your tags in place, you can create a custom budget in AWS Billing. This provides visibility into your Windows EC2 spending. You can set a daily budget or a monthly budget.

Set up alerts using AWS Budgets

In this example scenario, you create a daily budget for Windows EC2. It's a recurring budget that uses the auto-adjusting option to track your spending and adjust the budget accordingly. If you have a static environment, you can use a fixed budget instead. Make sure to choose a baseline time-range (for example, 30 days).

1. Sign in to the AWS Management Console and open the AWS Cost Management console.
2. In the navigation pane, choose **Budgets**.
3. At the top of the page, choose **Create budget**.
4. Under **Budget setup**, choose **Customize (advanced)**.
5. Under **Budget types**, choose **Cost budget**. Then, choose **Next**.
6. Under **Details**, for **Budget name**, enter the name of your budget. For example, **Windows EC2 spend**.
7. Under **Set budget amount**, for **Period**, choose **Daily**.
8. For **Budget renewal type**, choose **Recurring budget** for a budget that resets after the budget period.
9. For **Start date**, choose the start date or period to begin tracking against your budgeted amount.
10. For **Budgeting method**, choose **Auto-adjusting (New)**.
11. For **Baseline time range**, choose **Custom range**, and then enter 30 days.
12. Choose **Next**.
13. In the **Budget scope** section, select **Filter specific AWS cost dimensions**. This is where tags are used to create the proper dimensions. AWS Budgets doesn't support **Platform Type** as an option in its filters. For this reason, you must apply **OS** tags.
14. Choose **Add filter**, and then select the **Tag** option from **Dimensions**.
15. Choose the **OS** tag, and then choose the Windows value for this to create a budget for the tag.
16. Choose **Next**.

17. On the **Configure alerts** page, choose **Add an alert threshold**. Here you set up two alerts: one for a 50 percent threshold and one for a 100 percent threshold. If the 50 percent threshold alert is breached before the halfway point in the month, it will provide a warning. That way, you can check if your spending is more than expected and react before reaching the end of the month.
18. For **Threshold**, enter **50** and select **% of budgeted amount**.
19. For **Trigger**, choose **Actual**.
20. For **email recipients**, enter an email address. Add another alert for a threshold of **100**.

> **Note**
>
> This example uses an email notification for the alert, but you can also use other approaches, such as Slack.

Enable Cost Anomaly Detection

You can use your cost tags to set up spending alerts that are an anomaly. For example, you can use AWS Cost Anomaly Detection to create monitors for your spending and get alerted when the system detects abnormal spending in your account.

To set up a monitor and alerts for the **Key=OS and Value=Windows** tag that you created previously, do the following:

1. Sign in to the AWS Management Console and open the AWS Cost Management console.
2. In the navigation pane, choose **Cost Anomaly Detection**.
3. Choose the **Cost monitors** tab, and then choose **Create monitor**.
4. In Step 1, choose the **Cost Allocation Tag** as your monitor type.
5. For **Cost Allocation Tag key**, choose **Windows EC2 spend**.
6. For **Cost Allocation Tag value**, choose **Windows**.
7. For **Name your monitor**, enter **Windows EC2 spend**.
8. Choose **Next**.
9. To create a subscription for the alerts, select **Create a new subscription**. If you have existing subscriptions, select **Choose an existing subscription**.
10. For **Subscription name**, enter **Windows EC2 spend anomaly**.
11. For **Alerting frequency**, choose **Daily summaries**.

12. For **Alert recipients**, enter your email address.
13. Choose **Add threshold**. For **Threshold**, enter **10** and then select **percent above expected speed**.
14. Choose **Create monitor**.

Get a real-time view into spending

An alert is a useful tool for monitoring your Windows EC2 spending, but you must use Cost Explorer if you want a real-time view into spending. Watch this video to learn how Cost Explorer enables you to analyze and reduce your EC2 costs. For more information, watch the AWS Supports You | Understanding and Reducing Your EC2 Costs video on YouTube.

View license-included spending for Windows

You can view the EC2 Windows spending in your account by using Cost Explorer. To see license-included spending for Windows, you must set the following correct filters in Cost Explorer:

- For **Platform**, choose **Windows (Amazon VPC)**. For **API operation**, choose **RunInstance:0002**. This is the AWS Billing code for license-included Windows EC2 instances.
- If you want to view your BYOL instance spending, change **RunInstance:0002** to **RunInstance:0800**. This is the billing code for Windows EC2 BYOL.

With this visibility in Cost Explorer, you can quickly filter down your costs to exactly what you're spending on Windows EC2. If you want to dive even deeper into your AWS spending, you can use AWS Cost and Usage Report to filter down to spending at the individual instance level. You can also generate reports that can be visualized in Amazon Quick Suite and build customized dashboards.

For more information, watch the AWS Supports You - Visualizing Your Cost and Usage Reports video on YouTube.

Additional resources

- Setting up required tags with AWS Config (AWS Config documentation)
- AWS Budgets Tutorial - Setup Alerts for AWS Billing | Amazon Web Services (YouTube)
- AWS Cost and Usage Report Query Library (AWS Well-Architected Labs)

SQL Server

Customers have been running Microsoft workloads on AWS for over 15 years, longer than any other cloud provider. This is largely because AWS has the most experience with Microsoft applications in the cloud and offers the best platform for Windows Server and Microsoft SQL Server in the following areas:

- Higher performance and reliability
- Greater security and identity services
- More migration support
- The broadest and deepest capabilities
- Lower total cost of ownership (TCO)
- Flexible licensing options

AWS supports everything necessary to build and run Windows applications that rely on SQL Server, including Active Directory, .NET, SQL Server, Windows desktop as a service, and all supported versions of Windows Server. With proven expertise, AWS can help you easily lift and shift, refactor, or even modernize your Windows workloads.

This section of the guide covers the following topics:

- Choose a high availability and disaster recovery solution
- Understand SQL Server licensing
- Select the right EC2 instance for SQL Server workloads
- Consolidate instances
- Compare SQL Server editions
- Evaluate SQL Server Developer edition
- Evaluate SQL Server on Linux
- Optimize SQL Server backup strategies
- Modernize SQL Server databases
- Optimize storage for SQL Server
- Optimize SQL Server licensing by using Compute Optimizer
- Optimize SQL Server sizing by using Compute Optimizer

- Review Trusted Advisor recommendations for SQL Server workloads

Choose a high availability and disaster recovery solution

Overview

We recommend that you design an architecture for your SQL Server deployment on AWS that meets your business needs while also meeting your disaster recovery (DR) objectives, including your recovery time objective (RTO) and recovery point objective (RPO). The following solutions can help you design the right architecture for SQL Server on Amazon Elastic Compute Cloud (Amazon EC2) while also optimizing costs for your SQL Server workloads.

- **SQL Server Always On availability groups** – SQL Server Always On availability groups provide high availability and disaster recovery (HA/DR) solutions for SQL Server databases. An availability group consists of a set of user databases that fail over together. Always On availability groups also provide redundancy at the database level, but don't require shared storage—each replica has its own local storage. You can deploy this feature as an HA/DR solution. For more information, see What is an Always On availability group? in the Microsoft documentation.
- **SQL Server Always On failover cluster instances (FCI)** – SQL Server Always On FCIs use Windows Server Failover Clustering (WSFC) to provide HA at the SQL Server instance level. FCIs require shared storage to host databases. You can use either shared block storage or shared file storage. For example, you can use Amazon FSx for Windows File Server or Amazon FSx for NetApp ONTAP as a shared storage solution with multiple Availability Zones. For more information, see Always On Failover Cluster Instances (SQL Server) in the Microsoft documentation.
- **SIOS DataKeeper** – SIOS DataKeeper can help you meet both HA and DR requirements by enabling a SQL Server FCI that spans both Availability Zones and AWS Regions. SIOS DataKeeper creates a clustered virtual SAN by using local Amazon Elastic Block Store (Amazon EBS) volumes and uses synchronous replication between Availability Zones for HA, while using asynchronous replication between Regions and for disaster recovery. For more information, see High Availability Protection for Windows Applications in the SIOS documentation.
- **Distributed availability groups** – Distributed availability groups are a special type of availability group that spans across two separate Always On availability groups. An availability group can reside across two separate Regions (for example, `us-east-1` and `us-west-1`). You can think of a distributed availability group as an availability group of availability groups because

the underlying Always On availability groups are configured on two different WSFC clusters. SQL Server Enterprise edition is required to deploy distributed availability groups. For more information, see Distributed availability groups in the Microsoft documentation.

- **Log shipping** – You can implement log shipping to protect your databases across multiple Regions, in the rare event a Region is impacted and becomes unavailable. Depending on the transaction and log shipping frequency, you can achieve RPO and RTO within minutes. For more information, see About Log Shipping (SQL Server) in the Microsoft documentation.
- **AWS Elastic Disaster Recovery** – Elastic Disaster Recovery is a software as a service (SaaS) application that manages the replication of servers from any infrastructure to AWS for DR purposes. You can also use Elastic Disaster Recovery to replicate SQL Server across Regions. Elastic Disaster Recovery is an agent-based solution that replicates entire virtual machines, including the operating system, all installed applications, and all databases into a staging area. For more information, see What is Elastic Disaster Recovery? in the Elastic Disaster Recovery documentation.
- **AWS Database Migration Service (AWS DMS)** – AWS DMS supports live migration of data to and from AWS, including a different Region. You can use this feature to set up a separate SQL Server instance in a different Region to serve as a disaster recovery database. For more information, see What is AWS Database Migration Service? in the AWS DMS documentation.

SQL Server Always On availability groups

If you're using SQL Server Enterprise edition just for a high availability Always On availability group, then you can downgrade to SQL Server Standard edition by taking advantage of basic availability groups. You can reduce costs from 65–75 percent by using basic availability groups instead of Always On availability groups.

> **Note**
>
> For additional information on cost differences between different SQL Server editions, see the Compare SQL Server editions section of this guide.

Features

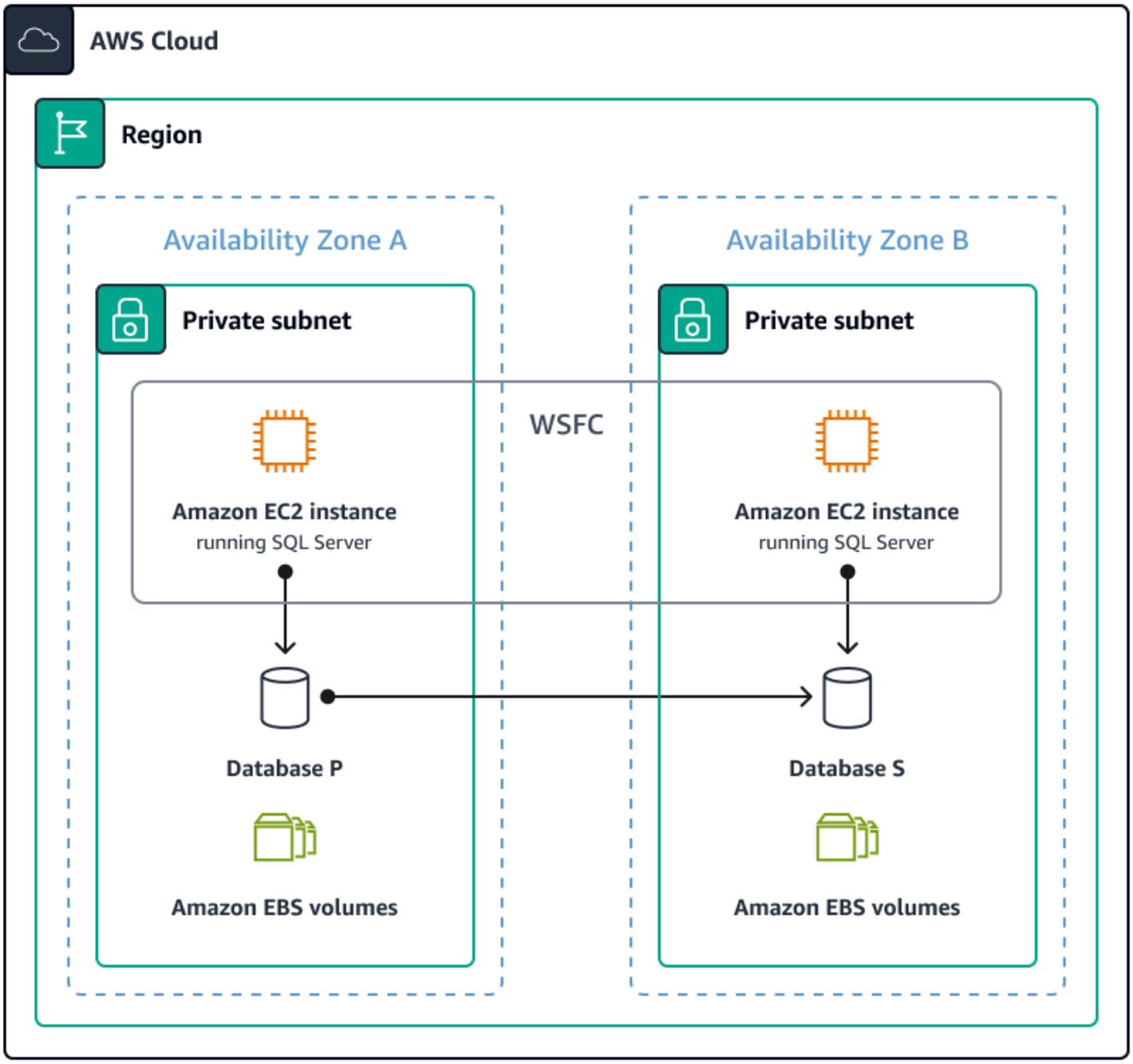
- Available in SQL Server Standard edition
- Limit of two replicas (primary and secondary)

- No read access on secondary replica
- No integrity checks on secondary replicas

Limitations

- Support for only one availability database per availability group
- Basic availability groups can't be part of a distributed availability group

The following diagram shows an example architecture for a Windows Server Failover Cluster solution.



SQL Server Always On failover cluster instances

You can use failover cluster instances (FCIs) to ensure continuous database operations while minimizing downtime and reducing the risk of data loss. FCIs offer a reliable solution if you're seeking high availability for your SQL Server database without a read replica configuration.

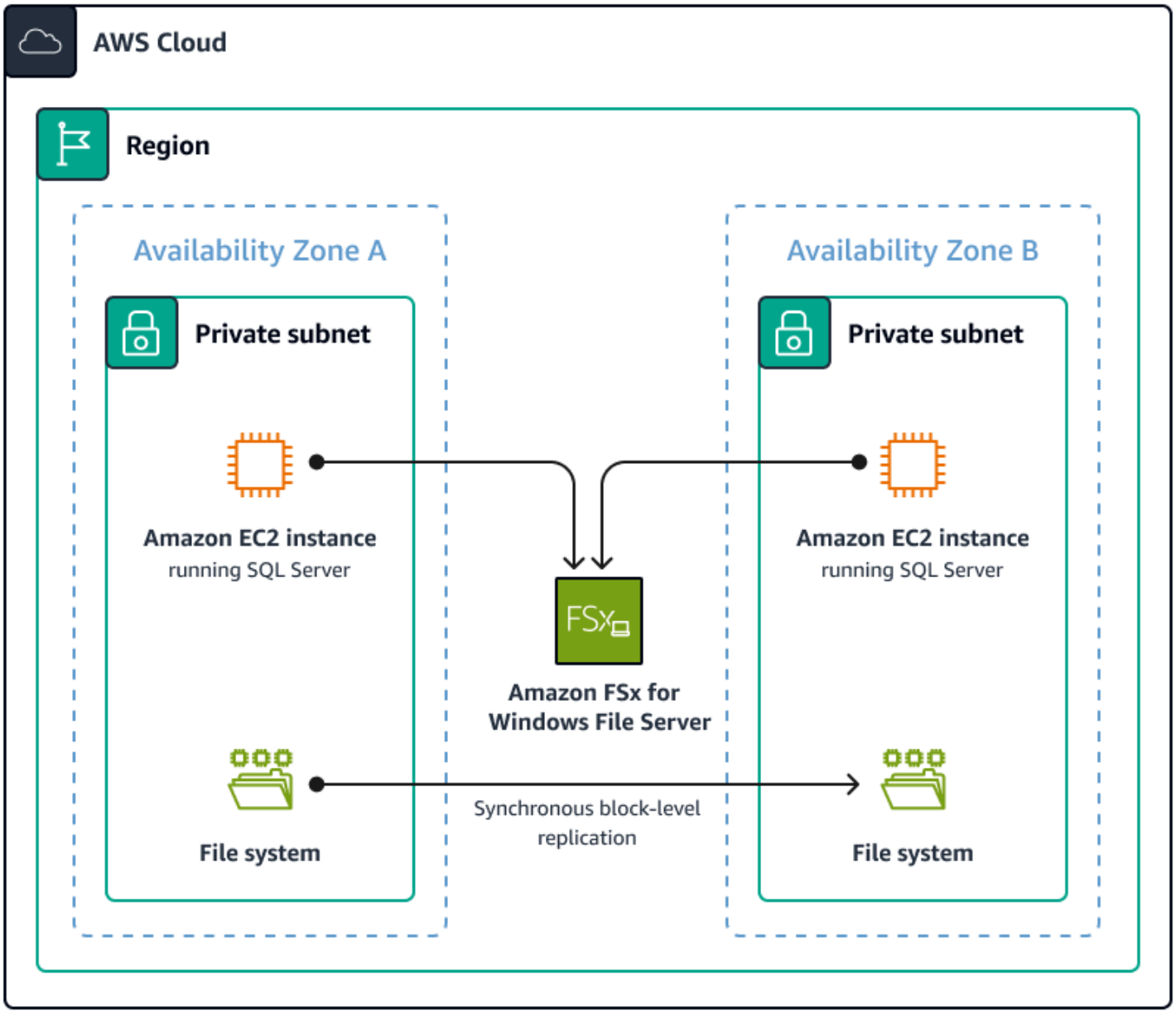
Unlike availability groups, FCIs can provide a dependable failover solution without requiring SQL Server Enterprise edition. Instead, FCIs require only SQL Server Standard edition licensing. You can use FCIs to reduce SQL Server licensing costs by 65–75 percent.

> **Note**
>
> For additional information on cost differences between SQL Server editions, see the Compare SQL Server editions section of this guide.

Consider the following:

- Amazon FSx for Windows File Server offers a powerful solution for meeting your SQL Server FCI shared storage requirements. You can use FSx for Windows File Server to avoid the need to purchase a license for a storage replication solution and manage shared storage on your own. This can result in significant cost savings of 30-40 percent. For more information, see the Simplify your Microsoft SQL Server high availability deployments using Amazon FSx for Windows File Server post on the AWS Storage Blog.
- With the Software Assurance benefits summary (downloadable PDF) and the Bring Your Own License (BYOL) model, you can take advantage of passive failover benefits, as long as the secondary server is passive. This results in cost savings for SQL licensing because you don't have to provide licenses to the passive node of the cluster.

The following diagram shows an example architecture for a SQL Server FCI by using FSx for Windows File Server.



SIOS DataKeeper

We recommend that you consider shared storage requirements if you're planning to deploy SQL Server FCIs on AWS. Traditional on-premises installations typically use a storage area network (SAN) to meet shared storage requirements, but this isn't a viable option on AWS. Amazon FSx for Windows File Server is the recommended storage solution for SQL Server FCI on AWS, but it has limitations that prevent adding cluster servers in different AWS Regions.

You can use SIOS DataKeeper to create a SQL Server FCI that covers both Availability Zones and Regions while reducing costs by 58–71 percent. SIOS DataKeeper can help you achieve the high

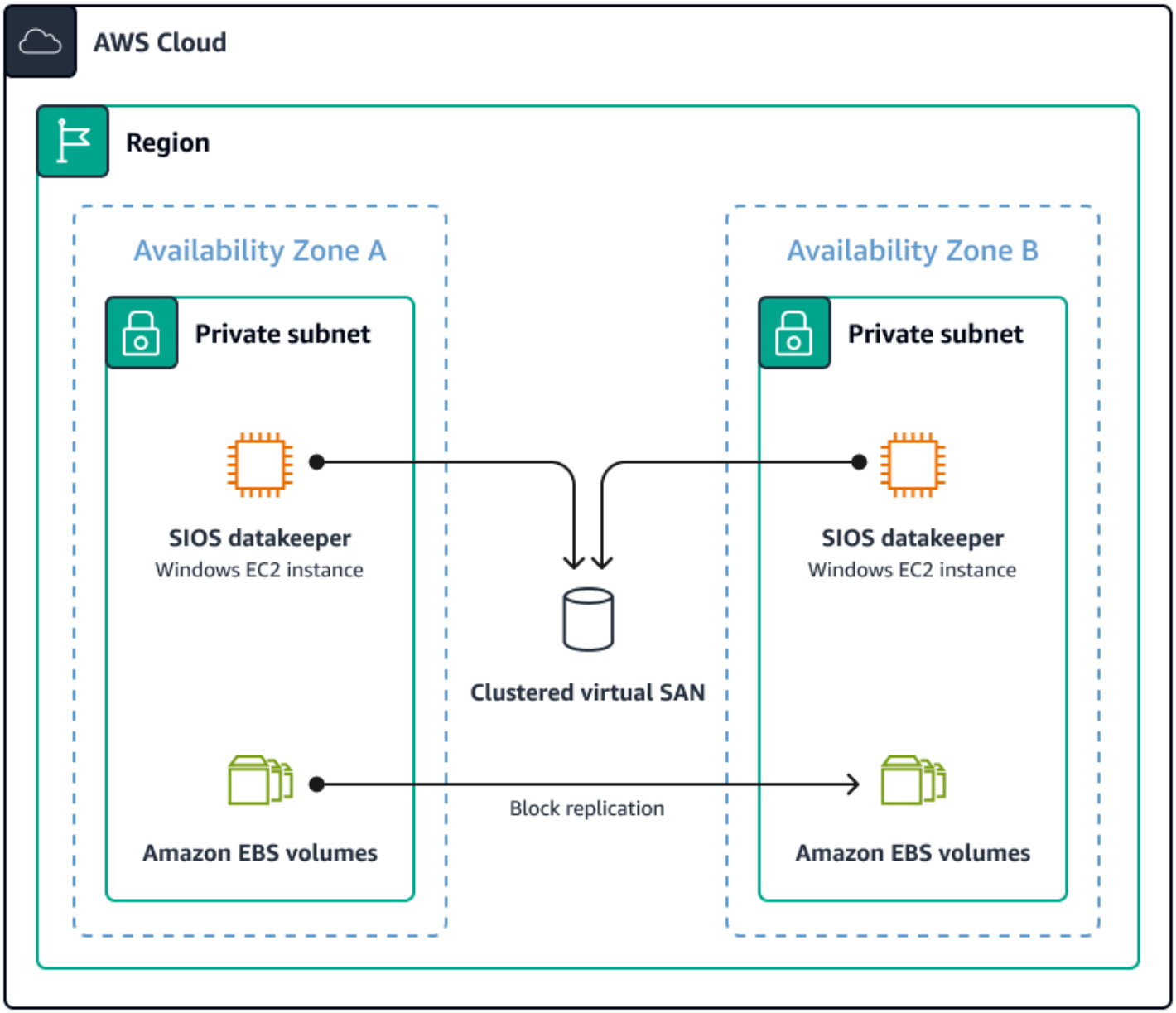
availability benefits of FCI. This makes SIOS DataKeeper a cost-effective and dependable solution for organizations.

Consider the following additional benefits of using SIOS DataKeeper:

- SIOS DataKeeper creates a clustered virtual SAN by using local EBS volumes and uses synchronous replication between Availability Zones for high availability. For disaster recovery, SIOS DataKeeper uses asynchronous replication between Regions.
- SIOS DataKeeper provides enterprise-class clustering features by using SQL Server Standard edition. This reduces SQL Server licensing costs between 65–75 percent compared to implementing high availability with SQL Server Always On availability groups that use SQL Server Enterprise edition. With SIOS DataKeeper, you can create a highly available, flexible, and cost-effective SQL Server environment that meets your organization's needs.

> **Note**
>
> For additional information on cost differences between SQL Server editions, see the Compare SQL Server editions section of this guide.

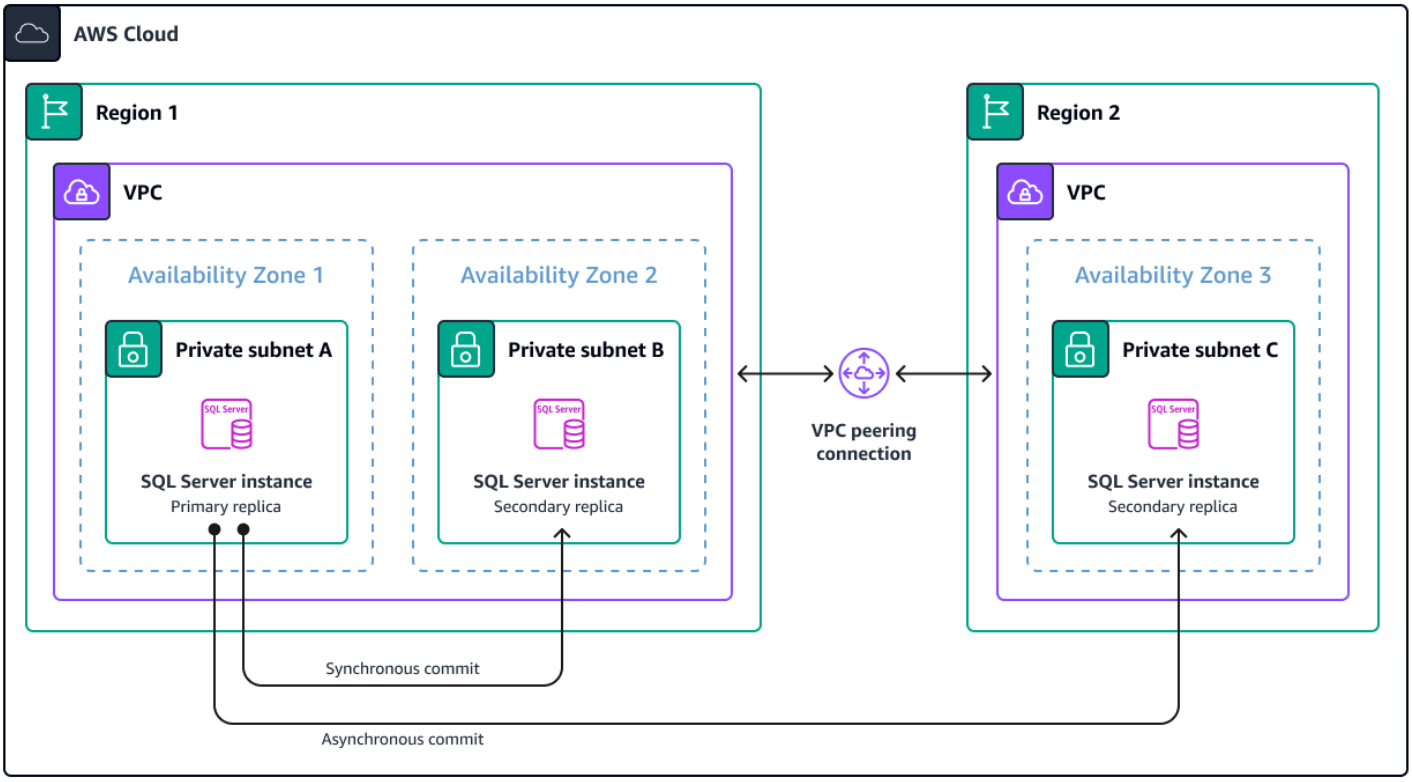
The following diagram shows an example architecture for a SQL Server FCI using a clustered virtual SAN solution.



Always On availability groups

You can use Always On availability groups for both high availability and disaster recovery purposes. You can achieve high availability by deploying SQL Server across two Availability Zones in one Region. You can achieve disaster recovery by extending availability groups across Regions.

The following diagram shows an example architecture for a solution based on Always On availability groups. The replicas in Region 1 of the diagram are using a Synchronous Commit, which provides an automatic failover of the availability group. The replica in Region 2 is using an Asynchronous Commit, which will require a manual failover of the availability group.

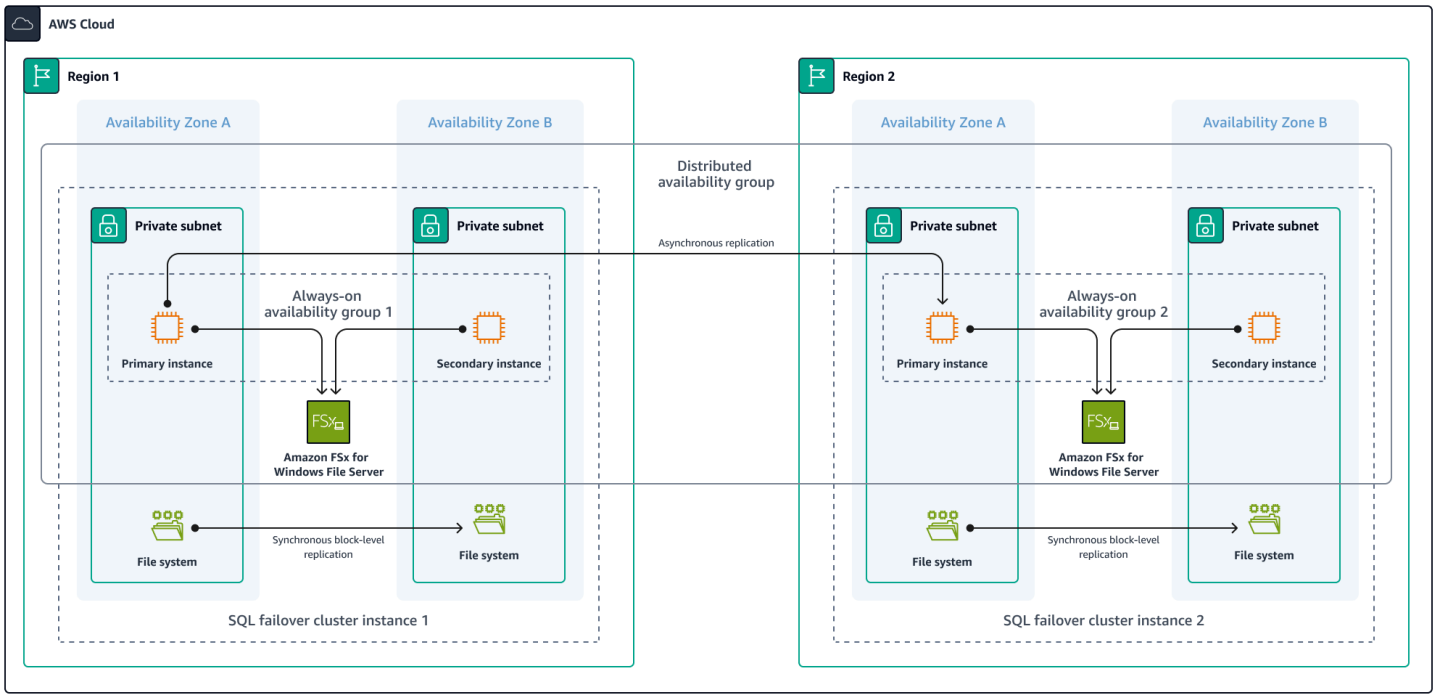


Distributed availability groups

For mission-critical SQL Server deployments where you can't compromise on reliability or disaster recovery, we recommend a multi-Region approach. Distributing your availability groups across multiple Regions is the most resilient solution for maintaining business continuity and minimizing downtime.

This architecture takes full advantage of the capabilities of Amazon FSx for Windows File Server, including shared storage, synchronous block-level replication, and SQL Server FCIs. These capabilities make it possible for you to create a highly available SQL Server environment that spans multiple Availability Zones. By replicating this setup in another Region, you get a fully redundant system that can handle even the most severe disruptions. What sets this solution apart is the level of flexibility and security it provides. The domain-independent architecture of distributed availability groups enables underlying Windows cluster servers to join different Active Directory domains, while certificate-based authentication ensures maximum protection for your SQL Server environments and provides high RTO and RPO requirements for a multi-Region DR strategy. For information about building a multi-Region architecture, see Field Notes: Building a Multi-Region Architecture for SQL Server using FCI and Distributed Availability Groups in the AWS Architecture Blog.

The following diagram shows an example architecture for a multi-Region solution using distributed availability groups.



Log shipping

Log shipping is a proven, reliable, and cost-effective method to safeguard your databases across Regions in the event of an unexpected outage. Organizations have been using log shipping to protect their data for decades.

If you implement log shipping on AWS, you can achieve RPO and RTO in minutes, depending on the frequency of transactions and log shipping jobs. In the unlikely event that a Region becomes inaccessible, log shipping keeps your data secure and recoverable.

Consider the following additional benefits of using log shipping:

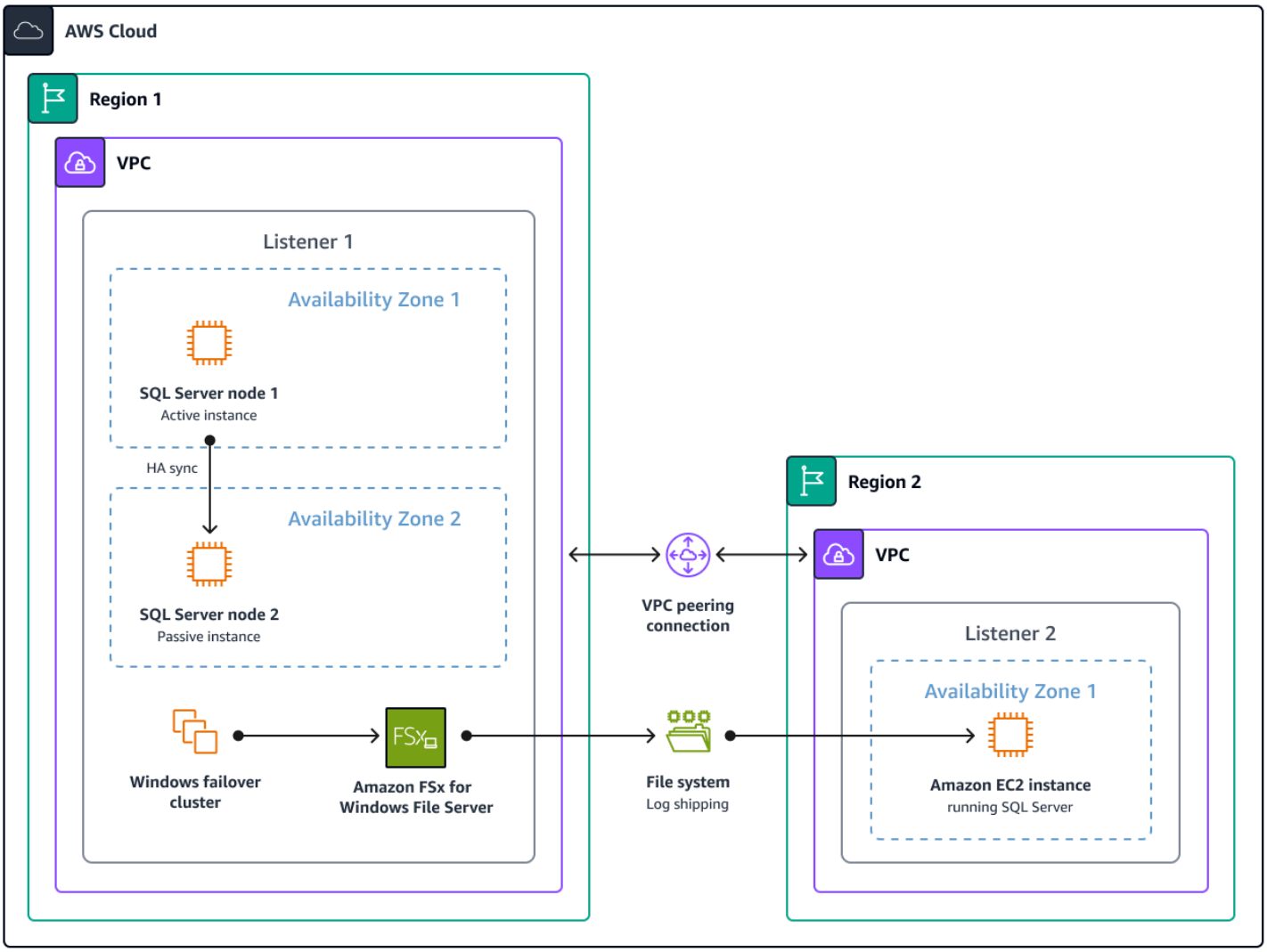
- Reduce costs and meet your business requirements by using log shipping for disaster recovery resilience across Regions. Log shipping reduces your TCO because you only need SQL Server Standard edition or SQL Server Web edition licenses.
- Remove licensing costs from a disaster recovery/passive server by using log shipping with active Software Assurance. Only the primary/active SQL Server needs to be licensed when you use log shipping with Software Assurance.

- Reduce SQL Server licensing costs by 65–75 percent by removing the need for SQL Server Enterprise edition to set up distributed availability groups between the Regions. You can do this by using SQL Server Standard edition and SQL Server FCIs combined with log shipping to meet your disaster recovery requirements.

> **Note**
>
> For additional information on cost differences between SQL Server editions, see the Compare SQL Server editions section of this guide.

For more information, see Extend SQL Server DR using log shipping for SQL Server FCI with Amazon FSx for Windows configuration in the AWS Architecture Blog.

The following diagram shows an example architecture for a log shipping solution.

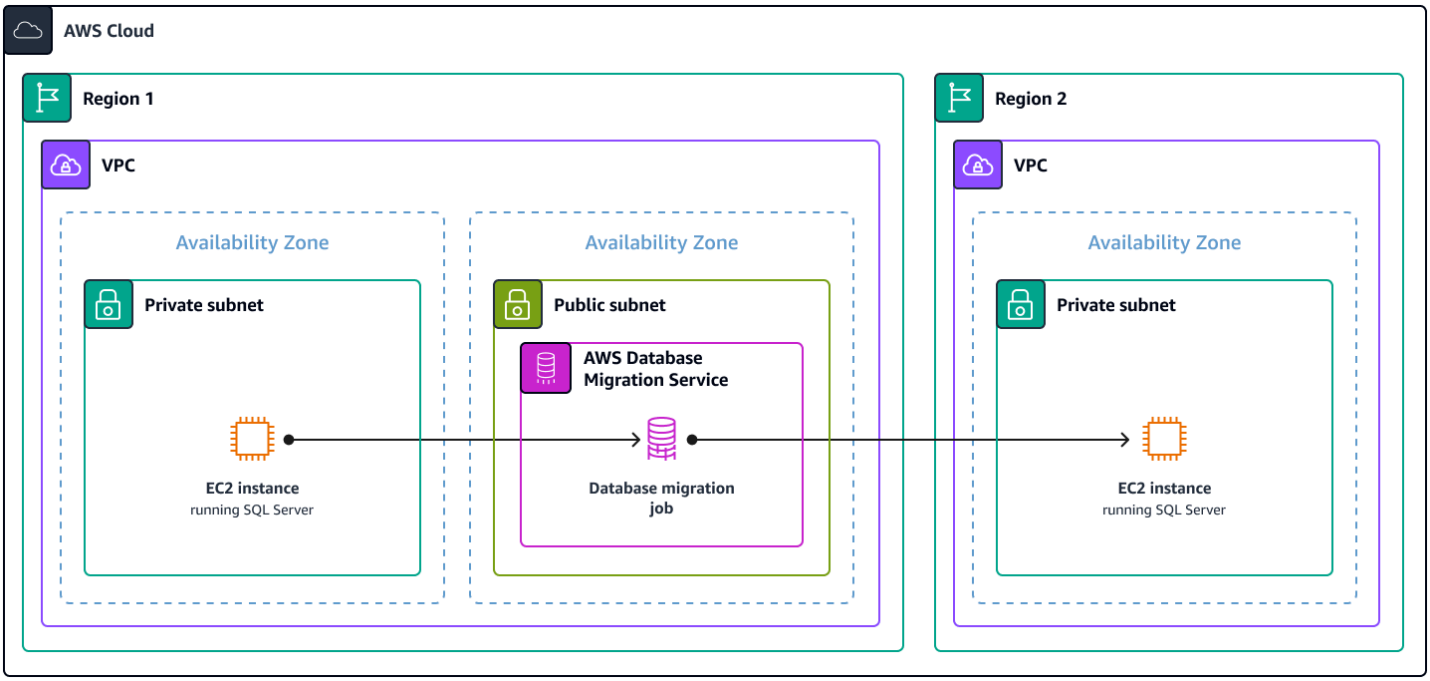


AWS Database Migration Service

You can use AWS Database Migration Service (AWS DMS) to design an HA/DR solution based on your application needs. AWS DMS enables you to easily copy data to a secondary SQL Server database in the same Region (HA) or across Regions (DR). This approach is technically sound, and allows you to maximize your investment in AWS infrastructure while optimizing your resource usage.

AWS DMS is a cost-effective service. You are charged only for the CPU resources used during the transfer process and any additional log storage. This means that you can benefit from this solution without incurring significant additional costs. You can use AWS DMS to ensure your data is available and accessible, while minimizing costs associated with licensing and resource usage.

The following diagram shows an example architecture for a solution based on AWS DMS.



AWS Elastic Disaster Recovery

Some organizations must ensure that all critical business applications have a disaster recovery plan in place. In the past, many of these organizations invested heavily in traditional disaster recovery solutions, which require you to pre-build and maintain an entire duplicate infrastructure. This approach is costly, time-consuming, and difficult to scale.

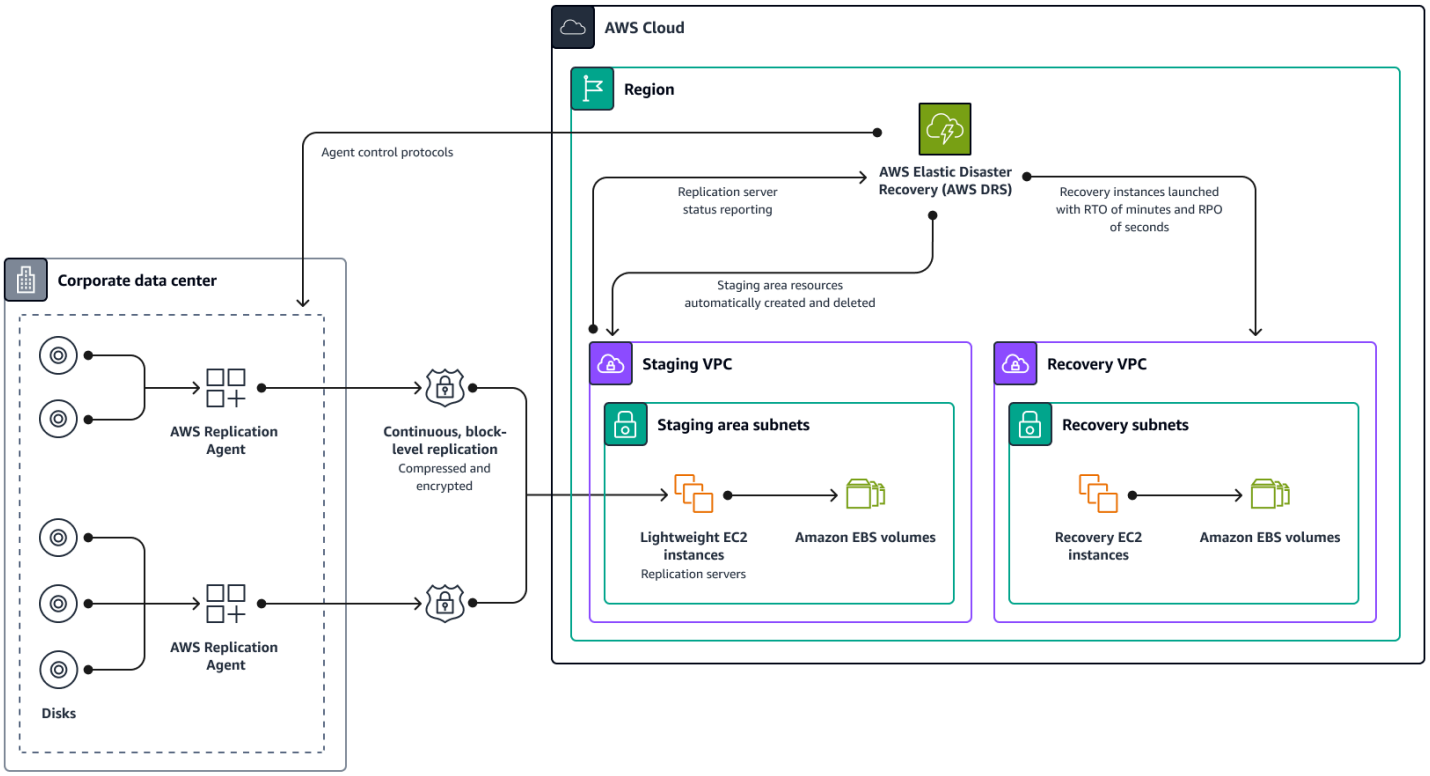
Now, you can use AWS Elastic Disaster Recovery to eliminate the need for pre-building a disaster recovery infrastructure. Disaster recovery machines are not started in Elastic Disaster Recovery until necessary, so you only pay for what you use when you need it. This means that you can significantly reduce your software licensing and high-performance compute costs.

Additionally, the staging area for the disaster recovery solution contains low-cost Amazon Elastic Block Store (Amazon EBS) volumes. EBS volumes further reduce the cost of provisioning duplicate resources. This allows you to reduce your overall disaster recovery costs while still maintaining a robust and reliable disaster recovery solution that meets your business requirements. You can use Elastic Disaster Recovery to focus on your core business activities, while AWS takes care of the underlying infrastructure for your disaster recovery solution.

For SQL Server, you can use Elastic Disaster Recovery as a cost-effective disaster recovery option. The licensing for the passive node in a fault-tolerant, highly-available SQL Server architecture is covered if you use active Software Assurance. However, you're still paying compute costs for the

passive server to be online. With Elastic Disaster Recovery, the primary server can replicate to the DR environment without the need to maintain active Software Assurance and without having to pay for disaster recovery compute costs. This combination of savings can reduce your SQL Server disaster recovery costs by 50 percent or more.

The following diagram shows an example architecture for a solution based on Elastic Disaster Recovery.



For more information, see How to set up high availability for SQL Server at DR site that was restored using AWS Elastic Disaster Recovery on the Microsoft Workloads on AWS Blog.

Cost comparison

The following table compares the costs of the HA/DR solutions covered in this section. The following assumptions are made for the purposes of this comparison:

- **Instance type** – r5d.xlarge
- **License type** – License included for both Windows and SQL Server
- **Region** – `us-east-1`

Solution	High availability	Disaster recovery	Enterprise edition	Standard edition	Cost
Log shipping	No	Yes	Yes	Yes	SQL Server Enterprise edition: $32,674.8 (2 nodes) SQL Server Standard edition: $14,804.4 (2 nodes)
Always On availability groups	Yes	Yes	Yes	Yes, but basic availability groups (2 nodes)	SQL Server Enterprise edition: $32,674.8 (2 nodes) SQL Server Standard edition: $14,804.4 (2 nodes)
Always On FCIs	Yes	No	Yes	Yes (2 nodes)	SQL Server Standard edition: $14,804.4
Distributed availability groups	Yes	Yes	Yes	No	SQL Server Enterprise edition: $65,349.6 (4 nodes)

Solution	High availability	Disaster recovery	Enterprise edition	Standard edition	Cost
Elastic Disaster Recovery	No	Yes	Yes	Yes	Approx. $107.48/ month for replication of 1 instance and 1 TB of storage **Note**: Elastic Disaster Recovery is billed hourly, per replicati ng server. The cost is the same, regardless of the number of disks, size of storage, number of drill or recovery launches, or the Region that you're replicating.

Solution	High availability	Disaster recovery	Enterprise edition	Standard edition	Cost
SIOS Data Keeper	Yes	Yes	Yes	Yes	Always On availability groups with Software Assurance (2 nodes, 24 cores): $213,480 2-node SQL Server cluster running on SQL Server Standard edition with SIOS DataKeeper and Software Assurance: $61,530 (2 nodes)
AWS DMS	No	Yes	Yes	Yes	$745.38/ month for r5.xlarge instance and 1 TB of storage

Cost optimization recommendations

We recommend that you take the following next steps to choose an HA/DR solution that meets your organization's requirements:

- Review the Select the right EC2 instance for SQL Server workloads section of this guide.
- Determine the IOPS and throughput requirements of your workloads by running performance counters during peak workloads:
  - IOPS = disk reads/sec + disk writes/sec
  - Throughput = disk read bytes/sec + disk write bytes/sec
- Use the following storage volume types for better performance and cost savings:
  - NVMe instance storage for `tempdb` and buffer pool extension
  - io2 volumes for database files
- Use AWS Trusted Advisor for recommendations on cost optimization for SQL Server on Amazon EC2. You don't need to install an agent for Trusted Advisor to do SQL Server optimization checks. Trusted Advisor inspects your Amazon EC2 SQL Server license-included instance configurations, such as virtual CPUs (vCPUs), version, and edition. Then, Trusted Advisor makes recommendations based on best practices.
- Use AWS Compute Optimizer for both Amazon EC2 instance and Amazon EBS right sizing recommendations.
- Use AWS Pricing Calculator to design your HA/DR strategy for cost estimations.
- To determine if downgrading from SQL Server Enterprise edition to SQL Server Standard edition is a possible option, use the sys dm_db_persisted_sku_features dynamic management view to identify edition-specific features that are active in the current database.

> **Note**
>
> Side-by-side migrations are necessary for SQL Server edition changes when using license-included EC2 instances.

- Perform semi-yearly or yearly disaster recovery drills to better architect a design that could recover the database with defined RTO and RPO. This can also help you identify any architecture weaknesses.

Additional resources

- Simplify your Microsoft SQL Server high availability deployments using Amazon FSx for Windows File Server (AWS Storage Blog)

- Field Notes: Building a Multi-Region Architecture for SQL Server using FCI and Distributed Availability Groups (AWS Architecture Blog)
- Architect a disaster recovery for SQL Server on AWS: Part 1 (AWS Database Blog)
- Microsoft SQL high availability with Amazon FSx for Windows (YouTube)
- Maximizing Microsoft SQL Server Performance with Amazon EBS (AWS Storage Blog)
- Comparing your on-premises storage patterns with AWS Storage services (AWS Storage Blog)
- Planning to replace a data center NAS with Amazon FSx File Gateway (AWS Storage Blog)
- Optimizing cost for your high availability SQL Server deployments on AWS (AWS Storage Blog)
- How to set up disaster recovery for SQL Server Always On Availability Groups using AWS Elastic Disaster Recovery (Microsoft Workloads on AWS)
- How to set up high availability for SQL Server at DR site that was restored using AWS Elastic Disaster Recovery (Microsoft Workloads on AWS)

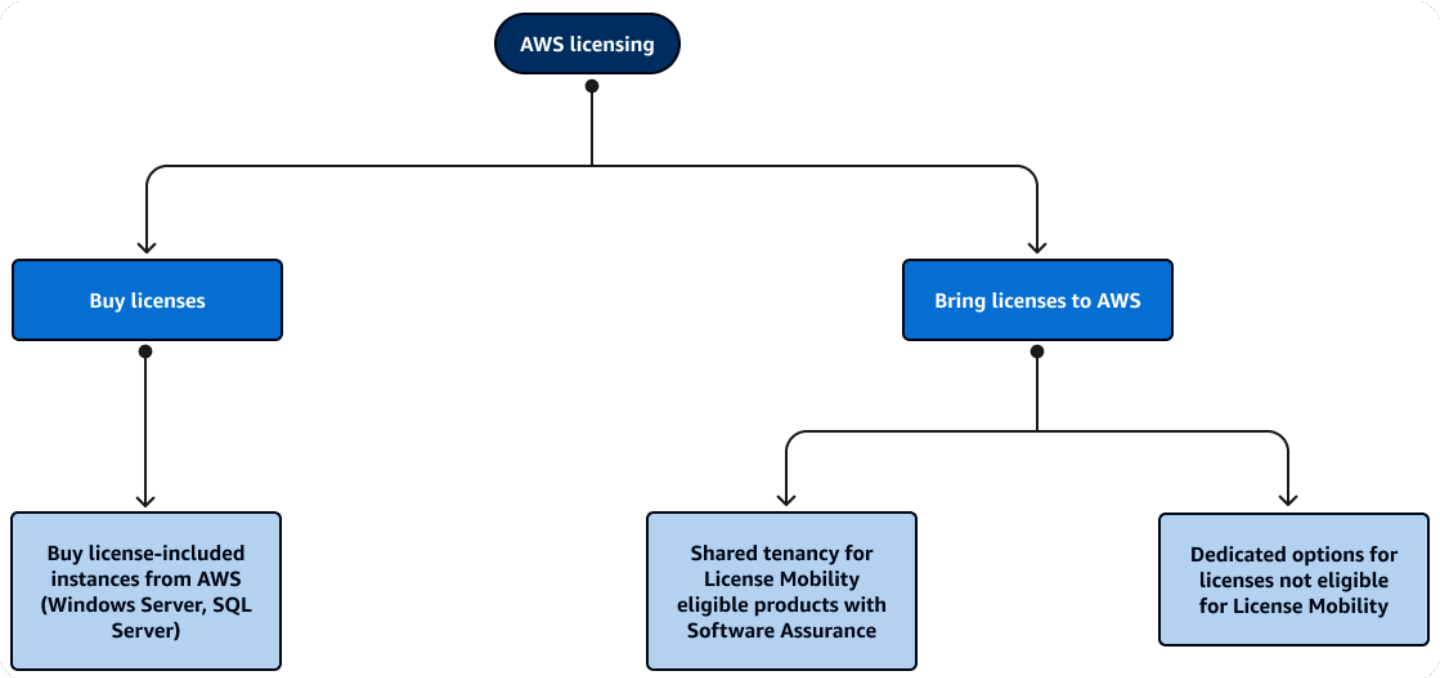
Understand SQL Server licensing

Overview

As more and more businesses move their workloads to the cloud, optimizing costs on cloud platforms has become a top priority. Licensing is one of the most significant costs associated with running Microsoft workloads on AWS. This section explains how to optimize costs on AWS by optimizing Microsoft licensing for SQL Server.

AWS licensing options

AWS offers a range of flexible cost optimization choices for licensing. These licensing options are designed to help you reduce costs, maintain compliance, and meet your business needs.



AWS categorizes licenses into three main types:

1. **License included** – This licensing option enables you to purchase and use licenses on demand, paying only for what you use. The license-included option is ideal for scenarios where you require flexibility in your licensing usage and want to avoid upfront costs. You can choose from a range of Windows Server, SQL Server, and other Microsoft products.
2. **Bring Your Own License (BYOL) products with license mobility** – This licensing option is designed for scenarios where you already have existing licenses and want to use them in the cloud. AWS allows customers to bring their own licenses to the cloud through Microsoft's License Mobility program. You can bring products that have License Mobility, such as SQL Server with Software Assurance (SA), to either shared or dedicated tenancy to reduce your AWS instance costs.
3. **BYOL products without license mobility** – For Microsoft products that don't have License Mobility, such as Windows Server, AWS offers dedicated options to use these products in the cloud. Additionally, dedicated hosts offer an opportunity to license at the physical core level. This can save you 50 percent or more on the licenses required to run your workloads. Dedicated hosts are a great option for stable and predictable workloads running most of the time.

Cost impact of bringing licenses

Bringing licenses can have a significant impact on the cost of running Microsoft workloads on AWS. If you bring your own licenses, you aren't required to pay additional licensing costs for the instances running in the cloud. This can lead to significant cost savings.

The following comparison shows the on-demand monthly cost of running a single c5.xlarge instance 24/7:

- Windows Server + SQL Server Enterprise edition: $1353/month (License included)
- Windows Server + SQL Server Standard edition: $609/month (License included)
- Windows Server only: $259/month (License included)
- Compute only (Linux): $127/month

Ultimately, bringing your own licenses can have a significant impact on the cost of running Microsoft workloads on AWS. If you use your existing licenses, you can reduce licensing costs and save money on your overall AWS bill.

License optimization

An AWS Optimization and Licensing Assessment (AWS OLA) can help you optimize your licensing by reducing compute and licensing costs. AWS OLA is designed to evaluate your licensing requirements for workloads running on AWS or for workloads that are planned for migration. AWS OLA provides recommendations on optimizing license usage.

One of the key strategies for optimizing licensing usage is right sizing instances. Right sizing involves selecting the right instance type for your workload based on its CPU, memory, and storage requirements. By choosing the appropriate instance size, you can ensure that you're using resources in a cost-efficient manner. This can lead to significant cost savings.

With Microsoft software licensing, the number of cores that the software runs on is a critical factor in determining licensing costs. For example, Windows Server and SQL Server licenses are typically licensed on the number of cores. By right sizing instances, you can lower the number of cores that the Microsoft software runs on and, in turn, reduce both the cost of the instance and the number of licenses required.

Cost optimization recommendations

Optimizing licenses is a key component of cost optimization on AWS. By implementing the right strategies, you can reduce licensing costs, maintain compliance, and achieve the best possible value from your licensing investment. This section outlines several strategies for license optimization.

Bring your eligible Windows Server licenses

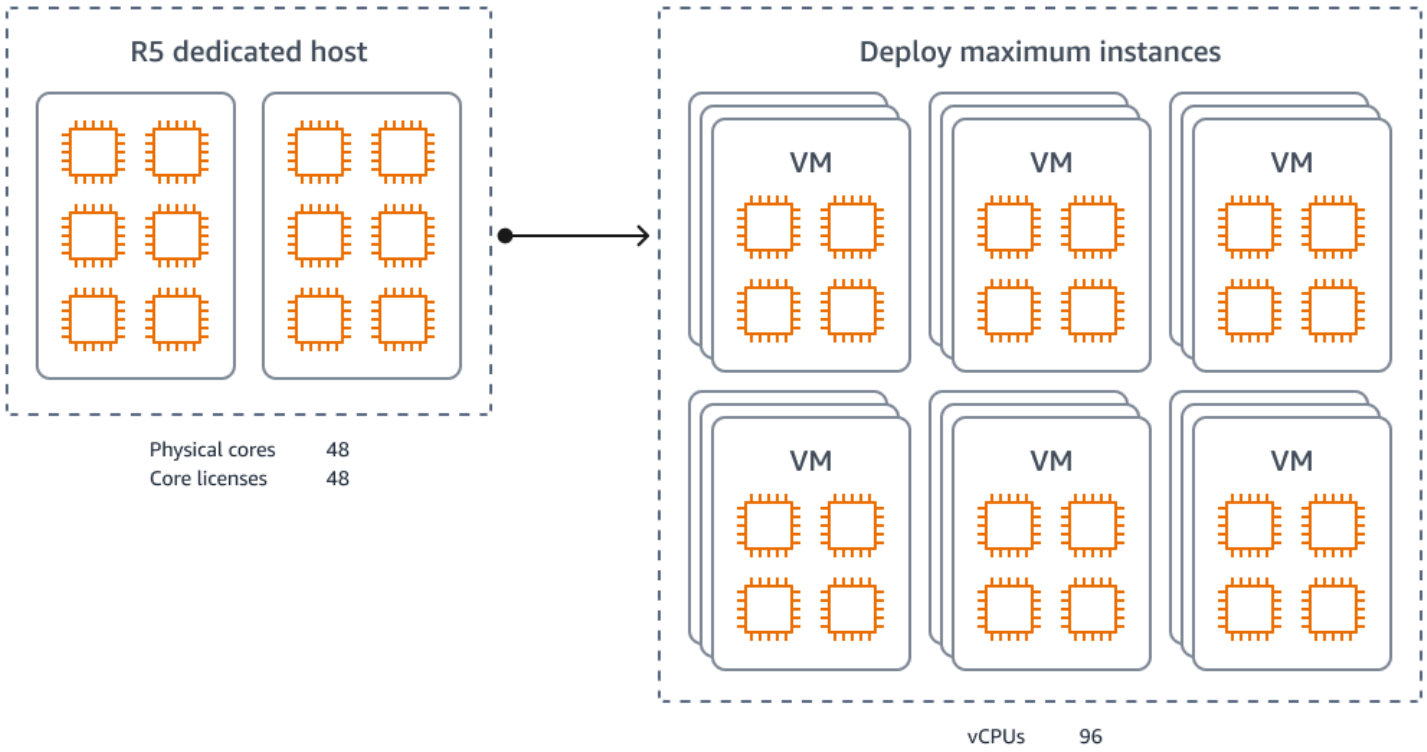
Bringing your own Windows Server licenses is one of the most effective strategies for license optimization. This strategy enables you to take advantage of your existing investments to reduce your AWS spending.

For example, you can deploy Windows Server 2019 and earlier versions on Amazon EC2 Dedicated Hosts if you purchased the licenses before 1/10/2019 or purchased the licenses as true-ups under active Enterprise Agreements signed before that date. This rule is based on a change that Microsoft made in 2019 to its licensing terms and conditions for products without License Mobility, such as Windows Server, when deployed on Listed Providers (for example, Alibaba, AWS, or Google Cloud). Under the new terms, you can't bring your own Windows Server licenses to AWS but must instead use license-included instances. However, if you purchased perpetual licenses prior to that date, then you can still deploy those Windows Server licenses on Amazon EC2 Dedicated Hosts.

Physical-level licenses

Licensing at the physical core level enables you to license just the physical cores of a host, so that you can then deploy a maximum number of instances without impacting the number of licenses required. This is typically done by using Windows Server Datacenter and SQL Server Enterprise edition.

As an example, consider an R5 dedicated host with 48 cores, which translates to 96 vCPUs. If you use Windows Server Datacenter edition, you only need 48 licenses. This enables you to deploy a combination of instances with up to 96 vCPUs, as the following diagram shows.



This approach can be especially cost-effective if you have enough workloads to maximize the number of instances that you can run on a host. By licensing at the physical core level, you can avoid additional licensing costs for each instance and achieve the best possible value for your licensing investment.

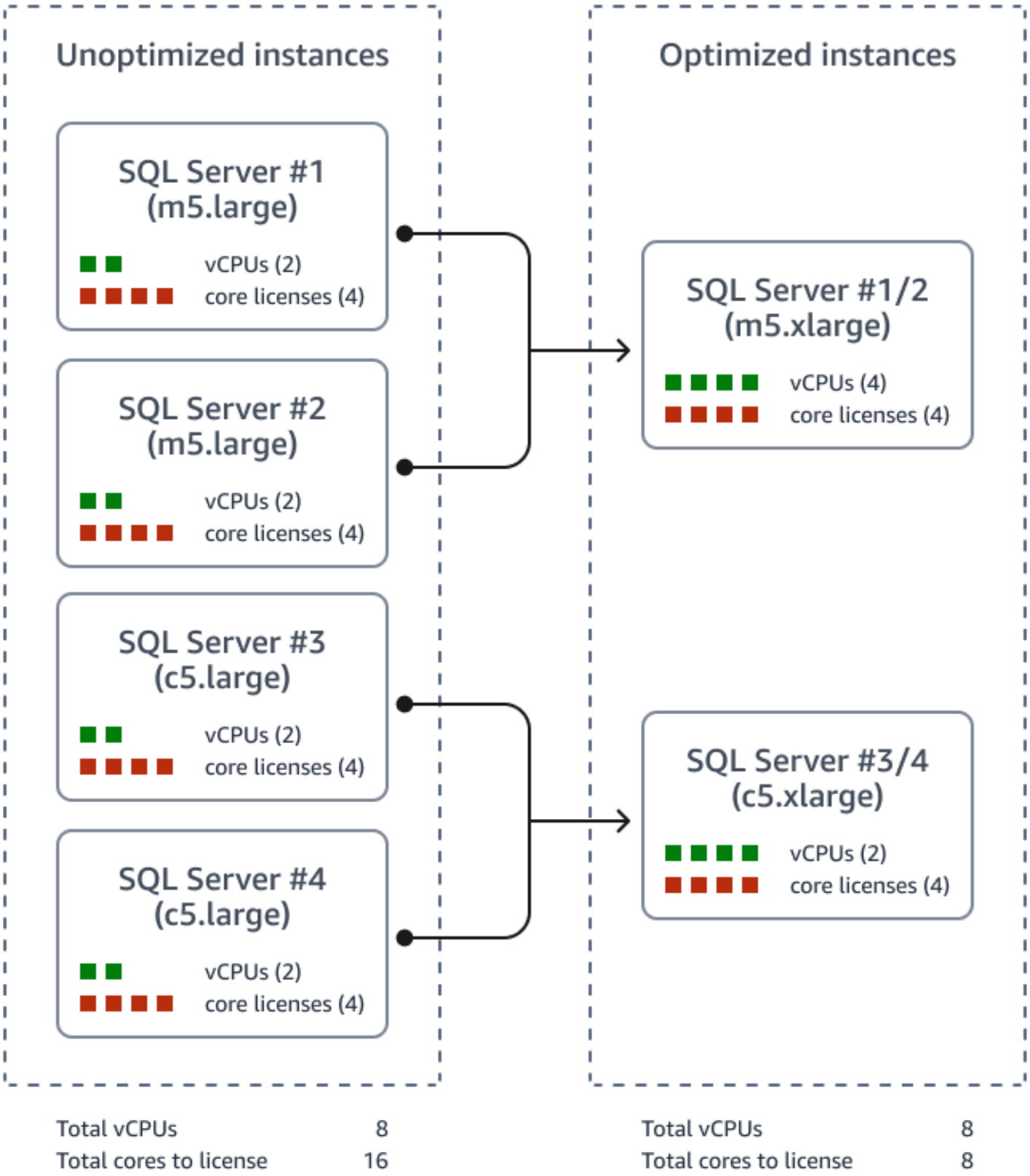
License at the physical core level of SQL Server

In shared tenancy, SQL Server licensing is based on the number of vCPUs allocated to the instance. In contrast, with dedicated hosts, you can license SQL Server Enterprise edition at the physical core level or at the vCPU level.

As with the previous example of the R5 dedicated host, if you license SQL Server Enterprise edition at the physical core level, then you only need 48 SQL Server Enterprise edition licenses to license the hosts. In contrast, in shared tenancy, where the only option is to license by vCPU, you must have 96 SQL Server Enterprise edition licenses for the same workload. Therefore, dedicated hosts can save you up to 50 percent on SQL Server licensing costs compared to shared tenancy. This is in addition to saving on instance costs by bringing eligible Windows licenses.

Consolidate SQL Server instances

SQL Server consolidation is the process of combining multiple SQL Server instances onto one server. SQL Server requires a minimum of four core licenses per instance, even if the instance only has two vCPUs. This means that running SQL Server on servers with less than four cores can cause you to over-license these instances and use more licenses than necessary.


Unoptimized instances
SQL Server #1 (m5.large)
vCPUs (2)
core licenses (4)
SQL Server #2 (m5.large)
vCPUs (2)
core licenses (4)
SQL Server #3 (c5.large)
vCPUs (2)
core licenses (4)
SQL Server #4 (c5.large)
vCPUs (2)
core licenses (4)
Optimized instances
SQL Server #1/2 (m5.xlarge)
vCPUs (4)
core licenses (4)
SQL Server #3/4 (c5.xlarge)
vCPUs (2)
core licenses (4)
Total vCPUs 8
Total cores to license 16
Total vCPUs 8
Total cores to license 8

For example, consolidating two instances with two vCPUs each into a single instance with four vCPUs can reduce the licensing requirement by 50 percent. This is because only four core licenses are required instead of eight.

For more information about consolidation, see the SQL Server consolidation section of this guide.

Downgrade SQL Server editions

Changing SQL Server editions can be a key strategy for optimizing licensing usage and reducing costs. The Enterprise edition of SQL Server is considerably more expensive than the Standard edition, so the downgrading can result in significant cost savings.

Transparent Data Encryption (TDE) and Always On availability groups are two popular features in SQL Server Enterprise edition. There are, however, cost-effective alternatives to these features that you can consider if you don't require the full feature set of SQL Server Enterprise edition. For example, you could get TDE in SQL Server Standard edition starting with SQL Server 2019. In place of Always On availability groups, you could use failover clustering with shared storage on FSx for Windows File Server for high availability with SQL Server Standard edition.

By downgrading from SQL Server Enterprise edition to SQL Server Standard edition, you can significantly reduce your licensing costs. For more information, see the Optimizing cost for your high availability SQL Server deployments on AWS post on the AWS Storage Blog.

In addition to reducing licensing costs, downgrading SQL Server editions can help reduce your Software Assurance spending and help you avoid future true-ups. If you return unused licenses to the shelf, you can avoid additional licensing costs and achieve the best possible value from your licensing investment.

It's important to evaluate your SQL Server workloads carefully and determine which features are critical for your business needs. For more information, see Assessing your environment in AWS Prescriptive Guidance, and determine whether your Microsoft SQL Server database uses SQL Server Enterprise edition-specific features.

If you choose the right edition of SQL Server and use alternatives to SQL Server Enterprise edition features, you can achieve significant cost savings while maintaining compliance and meeting your business needs. For more information about downgrading options, see the Compare SQL Server editions section of this guide.

Use SQL Server Developer edition in non-production environments

In non-production environments, you can deploy licensable editions of SQL Server, such as Enterprise or Standard edition, by using MSDN subscriptions in on-premises environments. However, the MSDN subscription doesn't have License Mobility. So, if you migrate to AWS, you can't bring those licenses over. You must use SQL Server Developer edition instead.

SQL Server Developer edition is a fully-featured edition of SQL Server that's available for free. This edition is available for SQL Server versions 2016 and later. You can download it from the Microsoft website. SQL Server Developer edition is intended to be used in all non-production environments, such as development, testing, and staging, as long as it's not connecting to live production data.

If you use SQL Server Developer edition in non-production environments, you can avoid additional licensing costs. For more information, see the Evaluate SQL Server Developer edition section of this guide.

Optimize CPU for SQL Server workloads

In some cases, you could be required to choose an instance type with more CPUs than are required for your workload due to other factors such as RAM or networking limits. However, AWS provides a solution to help you optimize your licensing costs in these situations.

You can, like most customers who bring SQL Server core licenses, disable hyperthreading or turn off CPUs on the EC2 instance to limit the number of available CPUs to the host. This option enables you to take advantage of other instance capabilities, such as RAM, while still saving on the cost of purchasing extra licenses.

For example, if you deploy an r5.4xlarge instance because your workload requires 128 GB of memory but you only need eight cores of SQL Server, then you can disable hyperthreading of an instance with just eight active CPUs. By doing this, you can save 50 percent on required SQL Server licenses, as you only need to license the eight cores that are actively being used.

Instance type	Total vCPUs	Active vCPU with Optimize CPUs feature	SQL Server license savings
r5.4xlarge	16	8	50%
r5.12xlarge	48	8	83%

The Optimize CPU feature can be configured during the Amazon EC2 launch configuration or by modifying an existing instance. It can also be applied to both BYOL and license-included Amazon EC2 instances. This flexibility helps you to rightsize your CPU to your workload's needs, while also reducing Windows Server and SQL Server licenses. For license-included Amazon EC2 instances, reducing CPUs provides an instant savings on licensing costs.

If you rightsize your instances, you can ensure that you're using the most cost-effective instance types for your workloads. As AWS introduces new instance types, it's important to evaluate whether these new instances can satisfy workload requirements with fewer cores.

Additional resources

- Amazon Web Services and Microsoft: Frequently Asked Questions (AWS documentation)

Select the right EC2 instance for SQL Server workloads

> **Important**
>
> Before you read this section, we recommend that you first read the Understand SQL Server licensing and Select the right instance type for Windows workloads sections of this guide.

Overview

Microsoft SQL Server has been running on Amazon Elastic Compute Cloud (Amazon EC2) instances for over 15 years. AWS has taken that experience and used it to help develop Amazon EC2 instances to fit SQL Server workloads running from minimal specifications all the way to high performance, multi-Region clusters.

Choosing the correct EC2 instance for SQL Server is largely dependent on your workload. Understanding how SQL Server is licensed, how it uses memory, and how SQL Server features align with Amazon EC2 offerings can help guide you to the best EC2 instance for your application.

This section addresses a variety of SQL Server workloads and how they can be paired with certain EC2 instances to keep your licensing and compute costs to a minimum.

Cost comparison

Amazon EC2 enables you to Bring Your Own License (BYOL) or pay as you go with Windows Server and SQL Server licensing. For pay-as-you-go licensing, the licensing costs for the Windows Server and SQL Server licenses are baked into the hourly cost of the EC2 instance. For example, you can have different AMIs with different prices. The price of the AMI is contingent on the SQL Server edition that the AMI runs on.

Windows Server and SQL Server pricing isn't itemized. You won't find itemized pricing on tools like the AWS Pricing Calculator. If you select different combinations of license-included offerings, the licensing costs can be deduced, as the following table shows.

EC2 instance	AMI	Compute price	Windows license price	SQL license price	Total price
r5.xlarge	Linux (compute pricing)	$183.96	-	-	$183.96
r5.xlarge	Linux + SQL Developer	$183.96	$0	$0	$183.96
r5.xlarge	Windows Server (LI)	$183.96	$134.32	-	$318.28
r5.xlarge	Windows + SQL Developer	$183.96	$134.32	$0	$318.28
r5.xlarge	Windows + SQL Web (LI)	$183.96	$134.32	$49.64	$367.92
r5.xlarge	Windows + SQL Standard (LI)	$183.96	$134.32	$350.4	$668.68
r5.xlarge	Windows + SQL	$183.96	$134.32	$1095	$1413.28

EC2 instance	AMI	Compute price	Windows license price	SQL license price	Total price
	Enterprise (LI)				

> **Note**
>
> Pricing in the preceding table is based on the on-demand pricing in the `us-east-1` Region.

The most cost-effective method for running SQL Server is to stay at a lower-level edition until you need a feature from a higher-level edition. For more information, see the Compare SQL Server editions section of this guide. Upgrading from SQL Server Web edition to SQL Server Standard edition is over seven times the SQL Server licensing cost and over three times the cost of moving from Standard edition to Enterprise edition. The disparity in licensing costs is a major factor to consider and is explored in the rest of this section.

Cost optimization scenario

Consider an example scenario where an analytics company tracking delivery vehicles is seeking to improve its SQL Server performance. After a MACO expert reviews the company's performance bottlenecks, the company transitions from x1e.2xlarge instances to x2iedn.xlarge instances. Although the instance size is smaller, the enhancements to the x2 instances improve SQL Server performance and optimization by using buffer pool extensions. This allowed the company to downgrade from SQL Server Enterprise edition to SQL Server Standard edition and reduce its SQL Server licensing from 8 vCPUs to 4 vCPUs.

Before optimization:

Server	EC2 instance	SQL Server edition	Monthly cost
ProdDB1	x1e.2xlarge	Enterprise	$3,918.64
ProdDB2	x1e.2xlarge	Enterprise	$3,918.64
Total			**$7,837.28**

After optimization:

Server	EC2 instance	SQL Server edition	Monthly cost
ProdDB1	x2iedn.xlarge	Standard	$1,215.00
ProdDB2	x2iedn.xlarge	Standard	$1,215.00
Total			**$2,430.00**

The combined changes from x1e.2xlarge instances to x2iedn.xlarge instances enabled the example customer to save $5,407 per month on their production database servers. This reduced the total cost of the workload by 69 percent.

> **Note**
>
> Pricing in the preceding table is based on the on-demand pricing in the `us-east-1` Region.

Cost optimization recommendations

Memory optimized instances

One of the most important aspects of SQL Server is understanding its reliance on memory. SQL Server attempts to use all available RAM not being used by the operating system (up to 2 TB for a default installation). It does this for performance reasons. Working with data in memory is much more performant than having to constantly pull data from disk, make changes, and then write it back to the disk. Instead, SQL Server attempts to load as much data from the attached databases as possible and keeps that data in RAM. Changes made to the data happen in memory and are hardened to disk at a later time.

> **Note**
>
> For a detailed explanation of how SQL Server writes changes, see Writing Pages in the Microsoft documentation.

Since SQL Server performs better with larger amounts of RAM, we typically recommend starting with Amazon EC2 memory optimized instance types. Memory optimized instances are versatile and offer a variety of different options. The R family has a 1-to-8 vCPU-to-RAM ratio and has options for Intel processors, AMD processors, enhanced networking, enhanced EBS performance, instance storage, and enhanced processor speed. For memory-heavy workloads, there's also an X family that combines many of the same options and extends the vCPU-to-RAM ratio to 1-to-32. Due to the versatility of memory optimized instances, you can apply them to SQL Server workloads of all shapes and sizes.

Workloads below minimal resources (less than 4 vCPUs)

Although some use cases work well with burstable (T3) instances, we recommend that you generally avoid using burstable instances for SQL Server workloads. The licensing for SQL Server is based on the number of vCPUs assigned to an instance. If SQL Server is idle the majority of the day and is acquiring burst credits, you pay for SQL licenses which you aren't fully utilizing. In addition, SQL Server has a minimum license requirement of 4 cores per server. This means if you have a SQL Server workload that doesn't require 4 vCPUs worth of compute power, you're paying SQL Server licensing which you aren't using. In these scenarios, it would be best to consolidate multiple SQL Server instances onto a larger server.

Workloads using minimal resources (less than 64 GB RAM)

Many SQL Server workloads under 64 GB RAM don't prioritize high performance or high availability. For these types of workloads, SQL Server Web edition might be a good fit if the application is covered under Microsoft's licensing restrictions.

> **Important**
>
> SQL Server Web edition has a restricted use case based on Microsoft's licensing terms. SQL Server Web edition may be used only to support public and internet accessible webpages, websites, web applications, and web services. It may not be used to support line-of-business applications (for example, customer relationship management, enterprise resource management, and other similar applications).

SQL Server Web edition scales up to 32 vCPUs and 64 GB RAM and is 86 percent less expensive than SQL Server Standard edition. For low resource workloads, using an AMD memory optimized instance like the r6a, which has a 10 percent less expensive compute price than its Intel counterpart, is also a good way to keep compute and SQL licensing costs to a minimum.

Workloads with average resources (less than 128 GB RAM)

SQL Server Standard edition is used on the majority of SQL Server workloads up to 128 GB RAM. SQL Server Standard edition is 65–75 percent less expensive than SQL Server Enterprise edition and can scale up to 48 vCPUs and 128 GB RAM. Since the 128 GB RAM limitation is typically hit before the 48 vCPU limitation, it's the focus of most customers who want to avoid upgrading to SQL Server Enterprise edition.

SQL Server has a feature called the buffer pool extension. This feature enables SQL Server to use a portion of a disk to act as an extension of RAM. The buffer pool extension works well when combined with ultra-fast storage, like the NVMe SSDs used in Amazon EC2 instance storage. Amazon EC2 instances containing instance storage are denoted with a "d" in the instance name (for example, r5d, r6id, and x2iedn).

Buffer pool extensions are not a replacement for normal RAM. However, if you require more than 128 GB of RAM, you can use buffer pool extensions with EC2 instances like the r6id.4xlarge and x2iedn.xlarge to delay an upgrade to Enterprise edition licensing.

High performance workloads (more than 128 GB RAM)

SQL Server workloads requiring high performance are challenging for cost optimization because of their reliance on a lot of resources. However, understanding the differences in EC2 instances can prevent you from making the wrong choice.

The following table shows a variety of memory optimized EC2 instances and their performance limits.

	r5b	r6idn	r7iz	x2iedn	x2iezn
Processor	3.1 GHz 2nd Generation Intel Xeon Processor	3.5 GHz 3rd Generation Intel Xeon Processor	3.9 GHz 4th Generation Intel Xeon Scalable Processor	3.5 GHz 3rd Generation Intel Xeon Processor	4.5 GHz 2nd Generation Intel Xeon Processor
CPU:RAM ratio	1:8	1:8	1:8	1:32	1:32

	r5b	r6idn	r7iz	x2iedn	x2iezn
Max vCPU	96	128	128	128	48
Max RAM	768 GB	1,024 GB	1,024 GB	4,096 GB	1,536 GB
Instance storage	–	NVMe SSD (4x 1900 GB)	–	NVMe SSD (2x 1900 GB)	–
io2 Block Express	Supported	Supported	Supported	Supported	–
Max EBS IOPS	260,000	350,000	160,000	260,000	80,000
Max EBS throughput	60 Gbps	80 Gbps	40 Gbps	80 Gbps	19 Gbps
Max network bandwidth	25 Gbps	200 Gbps	50 Gbps	100 Gbps	100 Gbps

Each instance is used for a different purpose. Understanding your SQL Server workload can help you choose the instance type that's best for you.

Details on attributes:

- **r5b** – The "b" attribute in r5b means this instance type is focused on high EBS performance. In the fifth generation of memory optimized instances, the r5b was the preferred choice. It was the first instance type to utilize io2 Block Express volumes and reach maximum storage IOPS of 260,000. The r5b instance type is still a cost-effective alternative for high EBS performance needs.
- **r6idn** – The sixth generation of memory optimized instances offered considerable improvements over the previous generation. The EBS performance enhancements from the r5b are taken a step further with the r6idn, bumping up the maximum IOPS to 350,000. The r6idn also has an instance store volume for tempdb and buffer pool extensions to further increase SQL Server performance.

- **x2iedn** – The x2iedn is similar to the r6idn. It offers similar levels of enhanced EBS, enhanced networking, and NVMe SSD instance storage, but with a 1:32 vCPU-to-RAM ratio for high memory workloads and low CPU quantity (lower SQL Server licensing costs).
- **x2iezn** – The "z" attribute in x2iezn indicates this instance type is focused on high processor performance. The Cascade Lake processor has an all-core turbo frequency up to 4.5 GHz. We recommend that you use this EC2 instance, coupled with a 1:32 vCPU-to-RAM ratio, in a scenario where you want to keep vCPU quantity low. This, in turn, can keep SQL Server licensing costs low.
- **r7iz** – The "z" attribute in r7iz indicates this instance type is focused on high processor performance. The Sapphire rapids processor has an all-core turbo frequency up to 3.9 GHz. Like the x2iezn instances, the r7iz prioritizes high frequency processor performance but with a 1:8 vCPU-to-RAM ratio.

Additional resources

- General purpose Amazon EC2 instances (AWS documentation)
- Comparison tool (Vantage)
- Licensing – SQL Server (AWS documentation)

Consolidate instances

This section focuses on the cost optimization technique of combining multiple SQL Server instances onto the same server to minimize licensing costs and maximize resource utilization.

Overview

Creating an instance is part of the process for installing the SQL Server Database Engine. The SQL Server instance is a complete install, containing its own server files, security logins, and system databases (master, model, msdb, and tempdb). Because an instance has all its own files and services, you can install multiple SQL Server instances on the same operating system without the instances interfering with each other. However, since the instances are all installed on the same server, they all share the same hardware resources, such as compute, memory, and networking.

It's typical to use only a single SQL Server instance per server in production environments so that a "busy" instance doesn't overuse the shared hardware resources. Giving each SQL Server instance its own operating system, with its own resources, is a better boundary than relying on resource

governance. This is especially true for high performance SQL Server workloads that require large amounts of RAM and CPU resources.

However, not all SQL Server workloads use a large amount of resources. For example, some organizations assign each of their customers their own dedicated SQL Server instance for compliance or security purposes. For smaller clients or clients that aren't typically active, that means running the SQL Server instances with minimal resources.

As noted in the Microsoft SQL Server 2019: Licensing guide, each server running SQL Server must account for a minimum of four CPU licenses. This means that even if you run a server with only two vCPUs, you must still license SQL Server for four vCPUs. Based on Microsoft's public SQL Server pricing that's a difference of $3,945 if you use SQL Server Standard edition. For organizations running multiple servers with single SQL Server instances using minimal resources, the combined cost of having to license unused resources can be substantial.

Cost optimization scenario

This section explores an example scenario that compares the difference between running four Windows Server servers, each with a single SQL Server instance, to a single larger Windows Server server running multiple SQL Server instances simultaneously.

If each SQL Server instance only needs two vCPUs and 8 GB RAM, the total cost per server is $7,890 for the SQL Server license in addition to an hourly compute cost of $0.096.

EC2 instance	vCPUs	RAM	Price	vCPUs to license	Total SQL Server licensing cost
m6i.large	2	8	0.096	4	$7,890

Expanding this out to four servers, the total cost is $31,560 for the SQL Server license with an hourly compute cost of $0.384.

EC2 instance	vCPUs	RAM	Price	vCPUs to license	Total SQL Server licensing cost
4x m6i.large	2	32	0.384	16	$31,560

If you combine all four SQL Server instances onto a single EC2 instance, the total amount of compute resources and compute stays the same. However, by removing unnecessary SQL Server licensing costs, you can reduce the total cost to run the workload by $15,780.

EC2 instance	vCPUs	RAM	Price	vCPUs to license	Total SQL Server licensing cost
m6i.2xlarge	8	32	0.384	8	$15,780

Note

In the preceding tables, compute costs show hourly on-demand pricing for Amazon EC2 servers running Windows Server in the `us-east-1` Region. The SQL Server Standard Edition licensing costs are referring to Microsoft's public SQL Server pricing.

Cost optimization recommendations

If you're considering consolidating SQL Server instances, the biggest concern is the resource consumption for each of the instances that you want to consolidate. It's important to get performance metrics over long periods to get a better understanding of the workload patterns on each server. Some common tools for resource consumption monitoring are Amazon CloudWatch, Windows Performance Monitor (perfmon), and the native monitoring tools of SQL Server.

We recommend that you consider the following questions when analyzing whether your SQL Server workloads could be combined to use the same server resources without them interfering with one another:

- What resources (CPU, memory, and network bandwidth) are consumed during your steady state?
- What resources (CPU, memory, and network bandwidth) are consumed during spikes?
- How often do spikes take place? Are spikes consistent?
- Do the resource spikes of one server coincide with the resource spikes of another server?
- What are the storage IOPS and throughput used by SQL Server?

If you wish to move forward with a plan to combine SQL Server instances, see the Run multiple instances of SQL Server on one Amazon EC2 instance post on the AWS Cloud Operations & Migrations Blog. This post provides instructions on how to make the configuration changes in SQL Server to add additional instances. Before you get started, consider the minor differences when multiple instances are installed on the same server:

- The default SQL Server database instance is named `MSSQLSERVER` and uses port 1433.
- Each additional instance installed on the same server is a "named" database instance.
- Each named instance has a unique instance name and a unique port.
- The SQL Server Browser must run to coordinate traffic to the named instances.
- Each instance can use separate locations for database data files and separate logins.
- The SQL Server max server memory settings must be configured according to the performance needs of each instance, with their combined total also leaving enough memory for the underlying operating system.
- You can use the SQL Server native backup and restore capabilities or AWS DMS for migration or consolidation.

Additional resources

- SQL Server Licensing Datasheet (AWS Cloud Operations & Migrations Blog)
- SQL Server Multiple instance setup blog post (AWS Cloud Operations & Migrations Blog)
- SQL Server Best practices guide (AWS Prescriptive Guidance documentation)

Compare SQL Server editions

Overview

Microsoft SQL Server licensing is one of the largest expenses for a Windows workload environment. Licensing costs for SQL Server can easily extend beyond the compute costs to run the workload. If you choose the wrong edition, you could pay for features that you aren't using or don't even need. This section compares the following SQL Server editions, including their features and relative costs:

- **Enterprise** – SQL Server Enterprise edition provides data center capabilities with high performance, unlimited virtualization, and several business intelligence (BI) tools.
- **Standard** – SQL Server Standard edition provides basic data management and business intelligence for smaller organizations and departments.
- **Web** – SQL Server Web edition is suitable for companies that are web hosters or web value added providers (VAPs). This edition offers a low total cost of ownership, and it provides scalability and manageability capabilities for small to large scale web properties.

> **Important**
>
> You can use SQL Server Web edition to support only public and internet accessible webpages, websites, web applications, and web services. You can't use SQL Server Web edition to support line-of-business applications (such as customer relationship management or enterprise resource management applications).

- **Developer** – SQL Server Developer edition includes all the functionality of Enterprise edition, but it's intended for development purposes only.
- **Express** – SQL Server Express edition is a free database and can be used for learning or for building desktop applications. You can update Express edition to other editions.

> **Note**
>
> SQL Server Evaluation edition is available for a 180-day trial period.

Cost impact

You can purchase SQL Server licenses from a Microsoft reseller and bring them to AWS with Software Assurance. Alternatively, you can use SQL Server licenses with a pay-as-you-go model that has license-included Amazon EC2 AMIs.

If you purchase SQL Server licenses from Microsoft resellers, the core licenses are sold in packs of two and you must license a minimum of four cores per server. The following table shows a cost comparison between Enterprise and Standard editions.

Version	SQL Server Enterprise edition (2 cores pack)	SQL Server Standard edition (2 cores pack)	Savings
2022	$15,123	$3,945	74%
2019	$13,748	$3,586	74%

Note

Pricing in the preceding table is based on Microsoft's public pricing for SQL Server 2022 and SQL Server 2019.

The following cost comparison shows hosting different editions of SQL Server with license-included Amazon EC2 AMIs. In this comparison, SQL Server is hosted on r6i.xlarge (4 vCPU) in the `us-east-1` Region.

Instance	Compute cost	Windows license cost	SQL Server license cost	Total
R6i.xlarge (Linux)	$183.96	–	–	$183.96
R6i.xlarge + Windows	$183.96	$134.32	–	$318.28

Instance	Compute cost	Windows license cost	SQL Server license cost	Total
R6i.xlarge + SQL Server Web edition	$183.96	$134.32	$49.35	$367.63
R6i.xlarge + SQL Server Standard edition	$183.96	$134.32	$350.4	$668.68
R6i.xlarge + SQL Enterprise edition	$183.96	$134.32	$1,095	$1,413.28

You can save up to 95 percent on SQL Server licensing costs by selecting the right SQL Server edition for your workload. The following table compares the cost of SQL Server licenses on r6i.xlarge instances.

Edition	% savings
Standard compared to Enterprise	68%
Web compared to Standard	86%
Web compared to Enterprise	95%

In most scenarios, organizations switch from Enterprise to Standard edition, but there are some cases where switching from Standard or Enterprise edition down to Web edition is possible.

Cost optimization recommendations

You can choose the best edition for your workload based on scaling limits, high availability, performance, and security. The following table shows features that are supported across SQL Server editions. This can help you decide which edition to use. This comparison applies to SQL Server 2016 SP1 and later versions.

Scaling limits

The following table compares the scaling limits of the different SQL Server editions.

Feature	Enterprise edition	Standard edition	Web edition	Express edition
Maximum compute capacity used by a single instance of SQL Server Database Engine, SQL Server Analysis Services (SSAS), or SQL Server Reporting Services (SSRS)	Operating system maximum	Limited to lesser of 4 sockets or 24 cores	Limited to lesser of 4 sockets or 16 cores	Limited to lesser of 4 sockets or 4 cores
Maximum memory for buffer pool per instance of SQL Server Database Engine	Operating system maximum	128 GB	64 GB	1410 MB
Maximum capacity for buffer pool extension per instance of SQL Server Database Engine	32 times max memory configured	4 times max memory configured	N/A	N/A

Feature	Enterprise edition	Standard edition	Web edition	Express edition
Maximum relational database size	524 PB	524 PB	524 PB	10 GB
Maximum memory for Columnstore caches or memory optimized data	Operating system maximum	32 GB	16 GB	352 MB

If your application requires fewer than 16 cores (32 vCPUs) and 64 GB of RAM, then you can start evaluating from SQL Server Web edition. If your workload requires more than 64 GB of memory or other high availability options, then you must upgrade to SQL Server Standard edition.

You can use SQL Server Web edition to support public and internet accessible webpages, websites, web applications, and web services, but you can't use SQL Server Web edition to support line of business applications. For more information about use cases for SQL Server Web edition, contact Microsoft Licensing Support or your Microsoft reseller.

You can use SQL Server Standard edition for workloads up to 24 cores (48 vCPUs) and 128 GB memory. However, you can use buffer pool extensions to enable SQL Server Standard edition to utilize local instance storage, like those present in r6id EC2 instances. This extends memory up to the size of four times the maximum memory configuration. This combination of features can delay a server from having to upgrade to Enterprise edition when memory requirements start to rise.

You can identify memory utilization by finding the databases pages in the buffer pool and page life expectancy counters. Page life expectancy tells you how long the page is in memory before being flushed back to disk. This counter default value is 300. If pages are residing in memory for hours or days, then there is a chance of reducing allocated memory.

High availability

The following table compares the high availability capabilities of the different SQL Server editions.

Feature	Enterprise edition	Standard edition	Web edition	Express edition
Server core support 1	Yes	Yes	Yes	Yes
Log shipping	Yes	Yes	Yes	No
Database mirroring	Yes	FULL safety mode	Only as witness	Only as witness
Backup compression	Yes	Yes	No	No
Always On failover cluster instances	16 nodes	2 nodes	No	No
Always On availability groups	Up to 8 secondary replicas, including 2 synchrono us secondary replicas	No	No	No
Basic availability groups	No	2 nodes	No	No
Online page and file restore	Yes	No	No	No
Online indexing	Yes	No	No	No
Online schema change	Yes	No	No	No
Fast recovery	Yes	No	No	No

Feature	Enterprise edition	Standard edition	Web edition	Express edition
Mirrored backups	Yes	No	No	No
Hot add memory and CPU	Yes	No	No	No
Encrypted backup	Yes	Yes	No	No
Hybrid backup to Microsoft Azure (backup to URL)	Yes	Yes	No	No
Failover server for disaster recovery	Yes	Yes	No	No
Failover servers for high availability	Yes	Yes	No	No

Other common features

The following table compares the most common features of the different SQL Server editions. For an extensive list of features, see Editions and supported features of SQL Server 2019 in the Microsoft documentation.

Feature	Enterprise edition	Standard edition	Web edition	Express edition
(Performance) Resource governor	Yes	No	No	No
(Security) Transparent Database Encryption (TDE)	Yes	Yes	No	No
(Security) Extensible key management (EKM)	Yes	No	No	No
(Replication) Oracle publication	Yes	No	No	No
(Replication) Peer to peer transactional replication	Yes	No	No	No
Change data capture	Yes	Yes	No	No

SQL Server Developer edition

All non-production workloads, such as development, QA, testing, staging, and UAT environments, can use SQL Server Developer edition to save 100 percent on SQL Server licensing costs. After you download SQL Server, you can install SQL Server Developer edition on EC2 instances by using shared tenancy. Dedicated infrastructure isn't required for SQL Server Developer edition. For more information, see this guide's recommendation for SQL Server Developer edition.

Switching editions

For existing workloads, switching from one edition to another edition requires extensive testing. It's a best practice to check workloads running on Enterprise or Standard editions to see if edition-specific features are used and if there are any alternative solutions for those features. For example, if you want to see if your databases are using any Enterprise-level features, you can run dynamic management views (DMV) on all databases as the following example command shows.

```
SELECT feature_name FROM sys.dm_db_persisted_sku_features; GO
```

There are some Enterprise edition features that can't be captured in T-SQL, such as online re-indexing as part of SQL maintenance jobs. These must be manually verified.

Migration considerations

How you license SQL Server will determine your options for switching editions. AMIs, including SQL Server AMIs, have the licensing cost included in the price of the EC2 instance—the licensing cost is bound to the AMI. You can use AWS billing codes to verify the SQL Server version included in your AMI. For AWS license-included instances, changing the SQL Server edition inside the operating system won't change the billing associated with the AMI. You must migrate your databases to a new EC2 instance with an AMI running the new edition of SQL Server.

If you're bringing your own license, then you have more flexibility. It's usually still recommended to migrate to another EC2 instance that's running the new version. This allows for easy failback if something doesn't go as planned. However, if you must use the existing server, you can still do a side-by-side installation of SQL Server and migrate the databases between instances. For more detailed steps about side-by-side edition downgrades, see Edition Upgrade and Downgrade in SQL Server on the MSSQLTips website.

Additional resources

- Editions and supported features of SQL Server 2022 (Microsoft Learn)
- sys.dm_db_persisted_sku_features (Transact-SQL) (Microsoft Learn)
- Which Version of SQL Server Should You Use? (Brent Ozar Unlimited)
- AWS Pricing Calculator (AWS)

Evaluate SQL Server Developer edition

Overview

SQL Server Developer edition is a free edition of SQL Server that contains all the features of the Enterprise edition and can be used in any non-production environment. In the cloud, where Microsoft Developer Network (MSDN) licensing can't be used, SQL Server Developer edition is a good way to save on costs without having to provide licenses for development and testing workloads. This is especially true for teams that run large development and testing environments and seek to reduce unnecessary costs.

A production environment is defined as an environment that's accessed by the end users of an application (such as an internet website) and is used for more than gathering feedback or acceptance testing of that application. Other scenarios that constitute production environments include:

- Environments that connect to a production database
- Environments that support disaster recovery or backup for a production environment
- Environments that are used for production at least some of the time, such as a server that is rotated into production during peak periods of activity

For more licensing information, see Amazon Web Services and Microsoft: Frequently Asked Questions in the AWS documentation.

Cost impact

If you use SQL Server Developer edition for non-production workloads, you can save 100 percent of your current SQL Server licensing costs for development and testing environments.

SQL Server version	SQL Server Enterprise edition (2 cores pack)	SQL Server Standard edition (2 cores pack)	SQL Server Developer edition
2022	$15,123	$3,945	Free
2019	$13,748	$3,586	Free

Note

Pricing in the preceding table is based on Microsoft's public pricing for SQL Server 2022 and SQL Server 2019.

The following table compares the cost of different SQL Server editions running with 4 vCPUs and using on-demand pricing in the `us-east-2` Region. This applies to scenarios that rely on license-included instances from AWS.

EC2 instance	AMI	Compute price	Windows license price	SQL Server license price	Total price
r5.xlarge	Linux (compute pricing)	$183.96	–	–	$183.96
r5.xlarge	Linux + SQL Server Developer edition	$183.96	$0	$0	$183.96
r5.xlarge	Windows Server (LI)	$183.96	$134.32	–	$318.28
r5.xlarge	Windows + SQL Server Developer edition	$183.96	$134.32	$0	$318.28
r5.xlarge	Windows + SQL Server Web edition (LI)	$183.96	$134.32	$49.64	$367.92
r5.xlarge	Windows + SQL Server	$183.96	$134.32	$350.4	$668.68

EC2 instance	AMI	Compute price	Windows license price	SQL Server license price	Total price
	Standard edition (LI)				
r5.xlarge	Windows + SQL Server Enterprise edition (LI)	$183.96	$134.32	$1095	$1413.28

Cost optimization scenario

After a data integrity company made a new acquisition, it wanted to migrate the newly acquired workload from its current location on a managed hosting provider to consolidate with its other workloads in the AWS Cloud. Initial pricing showed that the company's SQL Server workload would cost 60 percent more running on AWS than on the current managed service provider. A MACO SME evaluated the estimation and found that the customer was actually paying for SQL Server licenses at the managed hosting provider for their development and testing environments. By switching the non-production workloads to SQL Server Developer edition during the migration, the company reduced their SQL Server licensing by 40 percent.

SQL Server license included on Amazon EC2

If you have SQL Server on EC2 instances that use license-included AMIs, it's not possible to do a direct conversion from Enterprise edition to Developer edition. The licensing costs for license-included instances are bound to the AMI. Even if SQL Server is uninstalled from within the operating system, the EC2 instance is still charged for licensing costs.

To convert to Developer edition, you must download SQL Server Developer edition, install it on a new EC2 instance, and then migrate your databases. You can migrate SQL Server databases between EC2 instances by using a variety of methods. For more information, see SQL Server database migration methods in the *Migrating Microsoft SQL Server databases to the AWS Cloud* guide. You can also use the Automated SQL Server Developer solution to prep the new instance that you plan to migrate to.

SQL Server BYOL on Amazon EC2

If you have SQL Server instances that use a BYOL, you can choose from the following in-place conversion or side-by-side downgrade options:

- Download SQL Server Developer edition from the Microsoft website. For manual or automated installation instructions, see the Automating SQL Server Developer deployments post on the AWS Blog.
- Use SQL Server native backup and restore to migrate databases or detach/attach databases from one SQL instance to another.
- Use an automation tool for bulk deployments.

> **Note**
>
> SQL Server Developer edition is for non-production environments only.

Additional resources

- Automating SQL Server Developer deployments for deploying SQL Server Developer Edition on EC2 (AWS Blog)
- SQL 2022 pricing (Microsoft)
- SQL 2019 pricing (Microsoft)
- Licensing options (SQL Server on Amazon EC2)
- AWS Pricing Calculator (SQL Server on Amazon EC2 documentation)
- Microsoft SQL Server 2019 Licensing guide (download from Microsoft)
- SQL Server 2022 Developer edition (download from Microsoft)

Evaluate SQL Server on Linux

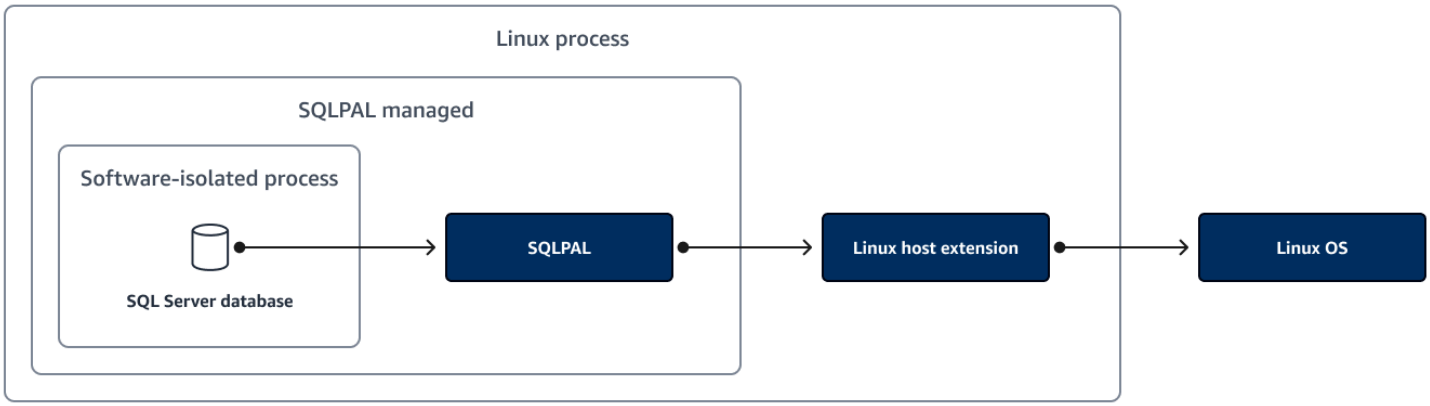
Overview

Since SQL Server 2017, it's been possible to install SQL Server on Linux operating systems. SQL Server on Linux is enterprise ready and offers flexibility, high performance, security features,

reduced TCO, HA/DR features, and a great user experience. You can switch from SQL Server on Windows Server to SQL Server on Linux to save on Windows Server licensing costs.

For Linux, SQL Server is available to deploy on Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), Ubuntu, and Amazon Linux 2. The SQL Server database engine runs the same way on both Windows Server and Linux, but there are some fundamental changes to certain tasks when using Linux. One key difference between running SQL Server Always On applications on Linux and Windows is related to failover clustering. If you deploy Always On availability groups on a Windows Server host, you can take advantage of Windows Server Failover Clustering (WSFC) and Active Directory as built-in features that support failover clustering. However, neither WSFC nor Active Directory are available to support failover clustering on Linux. If you want to launch failover clustering for SQL Server on Linux, you can use AWS Launch Wizard to simplify cluster setup and SQL installation on Linux instances by using ClusterLabs Pacemaker.

SQL Server on Windows and Linux share a common code base. That is, the SQL Server core engine hasn't been changed, at all, to run on Linux. SQL Server introduced a Platform Abstraction Layer (SQLPAL), as shown in the following diagram.



SQLPAL is responsible for abstraction of calls and communication between SQL Server and the underlying operating system. The host extension is simply a native Linux application. Low-level operating system functions are native calls to optimize the I/O, memory, and CPU usage. When the host extension starts, it loads and initializes SQLPAL, which then brings up SQL Server. SQLPAL launches isolated software processes that provide the required translation for the rest of the code. Adding this new layer to the SQL Server architecture means that the same enterprise-level core features and benefits that have made SQL Server so powerful on Windows are available regardless of operating system.

Cost impact

For r5.2xlarge instances, the Windows Server licensing cost reduction is about $268 in each scenario. The reduction is a higher percentage of the total server cost compared to using less expensive SQL Server editions. The following table shows the costs savings.

Instance	Edition	Monthly cost of SQL Server on Windows	Monthly cost of SQL Server on Linux	Savings
r5.2xlarge	Web	$735	$466	37%
r5.2xlarge	Standard	$1,337	$1,068	20%
r5.2xlarge	Enterprise	$2,826	$2,558	10%

Note

The price estimation in the preceding table is based on on-demand pricing in the `us-east-1` Region and can be viewed directly in the AWS Pricing Calculator.

Consider an example scenario where an ISV customer in the SMB segment is looking to save costs on their development environment. They're already using SQL Server Developer edition on a set of Windows servers. By switching from Windows with SQL Server Developer edition to Linux with SQL Server Developer edition, the ISV customer can save 33 percent on their development workload. The following table shows the following estimated costs for this scenario.

Estimate	Monthly cost
Windows + SQL Server	$9,307.72
Linux + SQL Server	$6,218.36
Estimated cost savings	$3,089.36 (33%)

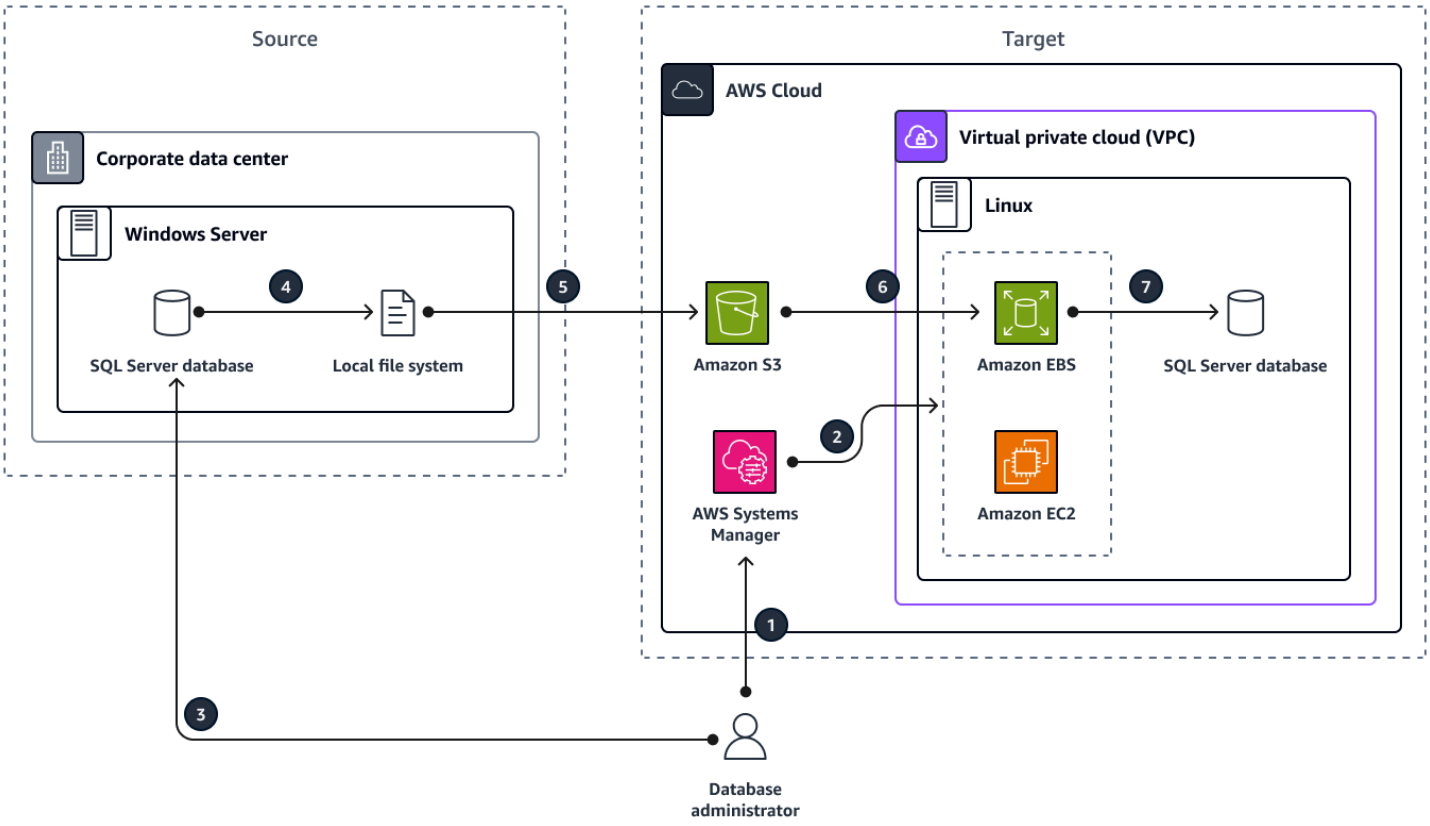
In another example scenario, a company migrates license-included SQL Server EC2 instances from Windows to Linux. The company saves a total of $300,000 per year on Windows Server licensing costs—roughly 20 percent of their total AWS bill.

Cost optimization recommendations

We recommend that you consider the following:

- SQL Server on Linux is supported starting with SQL Server 2017.
- To help make the switch, you can use the Windows to Linux replatforming assistant for Microsoft SQL Server Databases. The replatforming assistant is a scripting tool that can help you move existing SQL Server workloads from Windows to Linux operating systems by checking for common incompatibilities, exporting the databases from the Windows host, and then importing the databases into an EC2 instance running Microsoft SQL Server 2017 on Ubuntu 16.04.
- You can also use the backup and restore capabilities in SQL Server to switch from SQL Server on Windows to Linux.
- You can easily and quickly deploy to SQL Server on Linux or Ubuntu by using the AWS Launch Wizard. The Launch Wizard can deploy SQL Server on Linux or Ubuntu in both standalone and high availability scenarios based on your application needs. For more information, see the Deploying to SQL Server Always on Linux with AWS Launch Wizard post in the Microsoft Workloads on AWS blog.

The following diagram shows an architecture for a solution that uses the Windows to Linux replatforming assistant for Microsoft SQL Server Databases.



Additional resources

- Overview of SQL Server on Linux (Microsoft Learn)
- Installation guide for SQL Server on Linux (Microsoft Learn)
- Deploying to SQL Server Always on Linux with AWS Launch Wizard (Microsoft Workloads on AWS Blog)
- Highly Available SQL Server on Linux (AWS Open Source Blog)

Optimize SQL Server backup strategies

Overview

Most organizations are looking for the right solution to safeguard their data on SQL Server on Amazon EC2 to meet their current requirements for recovery point objective (RPO), the maximum acceptable amount of time since the last backup, and recovery time objective (RTO), the maximum acceptable delay between the interruption of service and restoration of service. If you're running SQL Server on EC2 instances, you have multiple options for creating backups of your data and

restoring your data. Backup strategies for safeguarding data for SQL Server on Amazon EC2 include the following:

- Server-level backup using Windows Volume Shadow Copy Service (VSS)-enabled Amazon Elastic Block Store (Amazon EBS) snapshots or AWS Backup
- Database-level backup using native backup and restore in SQL Server

You have the following storage options for database-level native backup:

- A local backup with an Amazon EBS volume
- A network file system backup with Amazon FSx for Windows File Server or Amazon FSx for NetApp ONTAP
- A network backup to Amazon Simple Storage Service (Amazon S3) using AWS Storage Gateway
- Direct backup to Amazon S3 for SQL Server 2022

This section does the following:

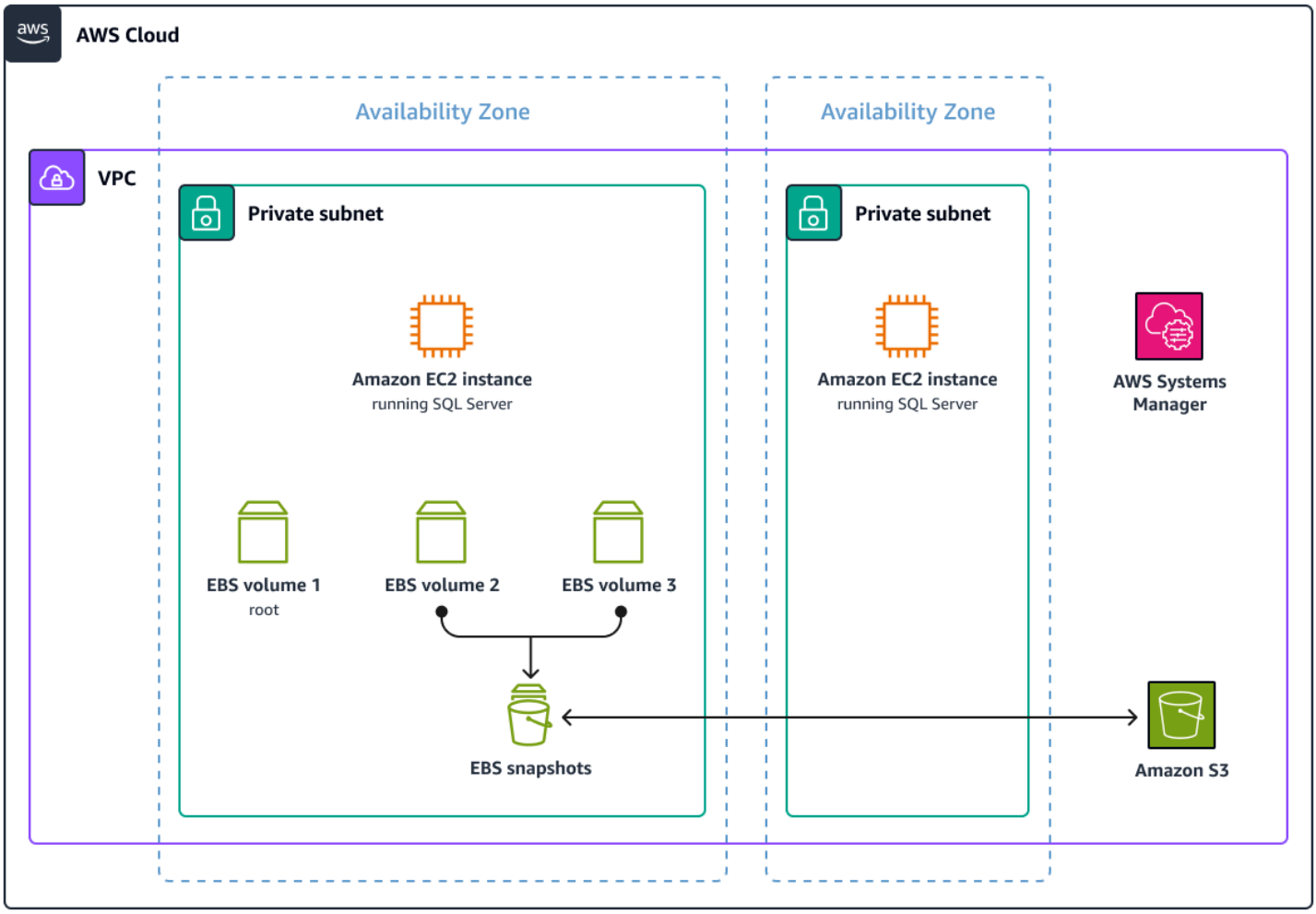
- Highlights features to help you save on storage space
- Compares the costs between different backend storage options
- Provides links to in-depth documentation to help implement these recommendations

Server-level backup using VSS-enabled snapshots

A VSS-enabled snapshots architecture uses the AWS Systems Manager Run Command to install the VSS agent on your SQL Server instances. You can also use the Run Command to invoke the entire workflow of flushing operating system and application buffers to the disk, pausing I/O operations, taking a point-in-time snapshot of the EBS volumes, and then resuming I/O.

This Run Command creates automated snapshots of all EBS volumes attached to a target instance. You also have the option to exclude the root volume, because user database files are usually stored on other volumes. In case you stripe multiple EBS volumes to create a single file system for SQL Server files, Amazon EBS also supports crash-consistent multivolume snapshots using a single API command. For more information about application-consistent VSS-enabled EBS snapshots, see Create a VSS application-consistent snapshot in the Amazon EC2 documentation.

The following diagram shows an architecture for server-level backup using VSS-enabled snapshots.



Consider the following benefits of using VSS-enabled snapshots:

- The first snapshot of a DB instance contains the data for the full DB instance. Subsequent snapshots of the same DB instance are incremental, which means that only the data that has changed after your most recent snapshot is saved.
- EBS snapshots provide point-in-time recovery.
- You can restore to a new SQL Server EC2 instance from a snapshot.
- If an instance is encrypted using Amazon EBS or if a database is encrypted in the instance using TDE, that instance or database is automatically restored with the same encryption.
- You can copy your automated cross-Region backups.
- When you restore an EBS volume from a snapshot it becomes immediately available for applications to access it. This means that you can immediately bring SQL Server online after restoring one or more of its underlying EBS volumes from snapshots.

- By default, restored volumes fetch underlying blocks from Amazon S3 the first time an application tries to read them. This means that there can be a lag in performance after an EBS volume is restored from a snapshot. The volume eventually catches up with the nominal performance. However, you can avoid that lag by using fast snapshot-restore (FSR) snapshots.
- You can use lifecycle management for EBS snapshots.

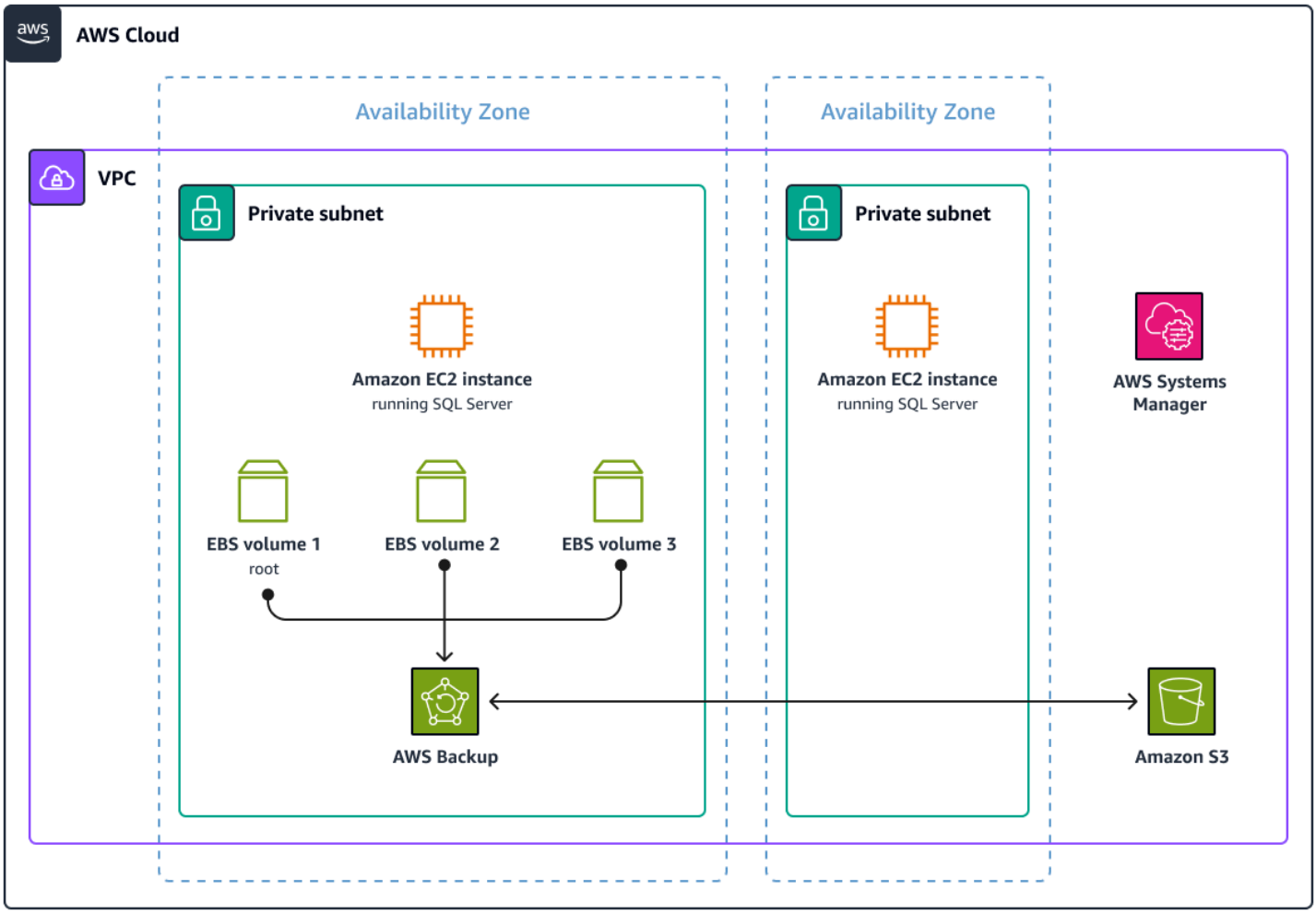
Consider the following limitations of using VSS-enabled snapshots:

- You can't perform cross-Region point-in-time recovery with an encrypted snapshot for a SQL Server instance.
- You can't create an encrypted snapshot of an unencrypted instance.
- You can't restore an individual database because the snapshot is taken at the EBS volume level.
- You can't restore the instance to itself.
- A snapshot of the DB instance must be encrypted by using the same AWS Key Management Service (AWS KMS) key as the DB instance.
- Storage I/O is suspended for a fraction of a second (approximately 10 milliseconds) during the snapshot backup process.

SQL Server backup using AWS Backup

You can use AWS Backup to centralize and automate data protection across AWS services. AWS Backup offers a cost-effective, fully managed, policy-based solution that simplifies data protection at scale. AWS Backup also helps you support your regulatory compliance obligations and meet your business continuity goals. Together with AWS Organizations, AWS Backup enables you to centrally deploy data protection (backup) policies to configure, manage, and govern your backup activity across your organization's AWS accounts and resources.

The following diagram shows the architecture of a backup and restore solution for SQL Server on EC2 by using AWS Backup.



Consider the following benefits of backing up SQL Server by using AWS Backup:

- You can automate backup scheduling, retention management, and lifecycle management.
- You can centralize your backup strategy across your organization, spanning multiple accounts and AWS Regions.
- You can centralize monitoring your backup activity and alerting across AWS services.
- You can implement cross-Region backups for disaster recovery planning.
- The solution supports cross-account backups.
- You can perform secure backups with secondary backup encryption.
- All backups support encryption by using AWS KMS encryption keys.
- The solution works with TDE.
- You can restore to a specific recovery point from the AWS Backup console.
- You can back up an entire SQL Server instance, which includes all SQL Server databases.

Database-level backup

These approaches use native Microsoft SQL Server backup functionality. You can take backups of individual databases on the SQL Server instance and restore an individual database.

Each of these options for native SQL Server backup and restore also support the following:

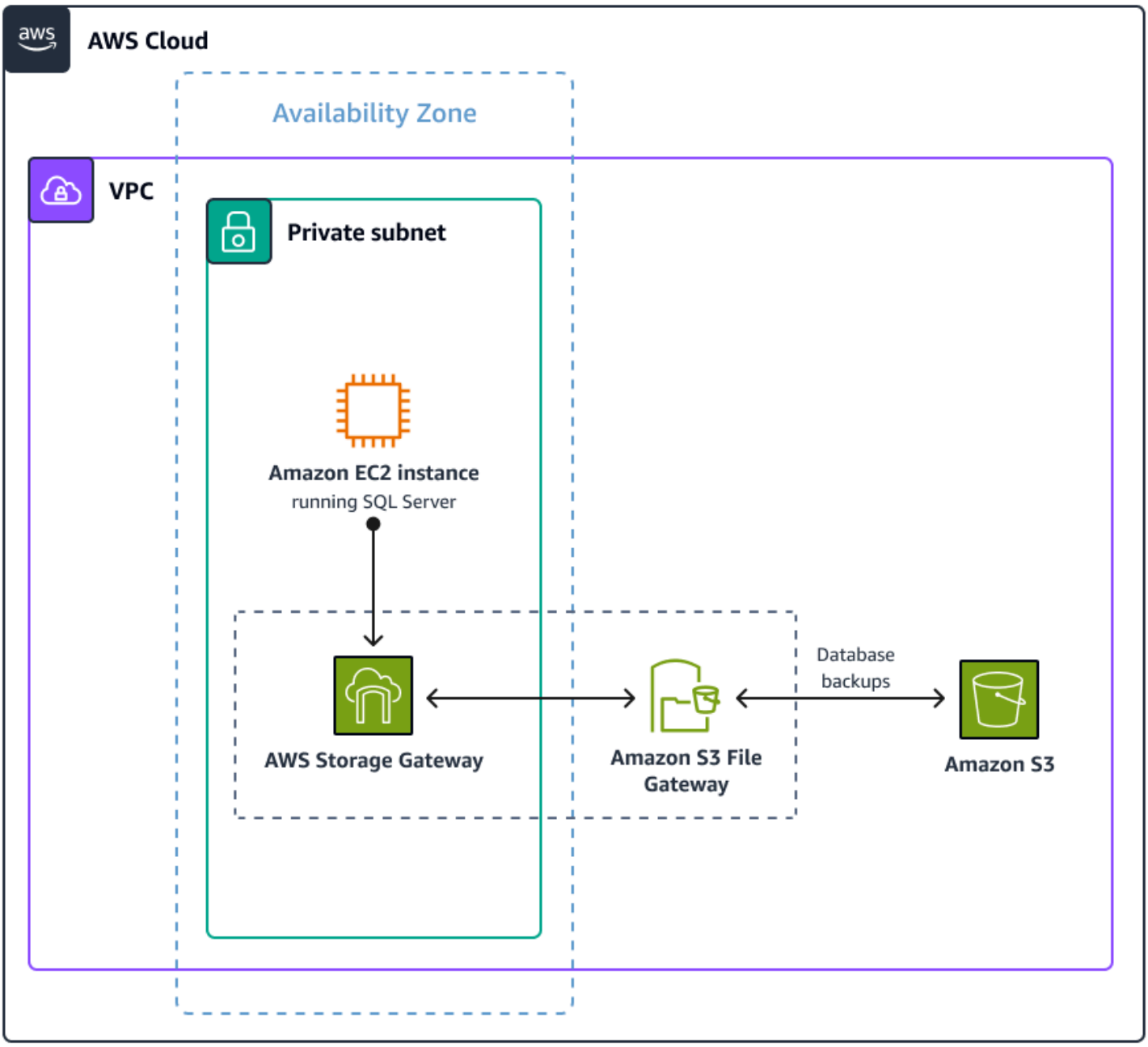
- Compression and multiple-file backup
- Full, differential, and T-log backups
- TDE-encrypted databases

SQL Server native backup and restore to Amazon S3

SQL Server on Amazon EC2 supports native backup and restore for SQL Server databases. You can take a backup of your SQL Server database and then restore the backup file to an existing database or to a new SQL Server EC2 instance, Amazon RDS for SQL Server, or an on-premises server.

Storage Gateway is a hybrid cloud storage service that provides on-premises applications with access to virtually unlimited cloud storage. You can use Storage Gateway to back up your Microsoft SQL Server databases directly to Amazon S3, reducing your on-premises storage footprint and using Amazon S3 for durable, scalable, and cost-effective storage.

The following diagram shows the architecture of a native backup and restore solution that uses Storage Gateway and Amazon S3.



Consider the following benefits of using native SQL Server backup with Storage Gateway:

- You can map a storage gateway as a Server Message Block (SMB) file share on the EC2 instance and send the backup to Amazon S3.
- The backup goes directly to the S3 bucket or through the Storage Gateway file cache.
- Multiple-file backups are supported.

Consider the following limitations of native backup using Storage Gateway:

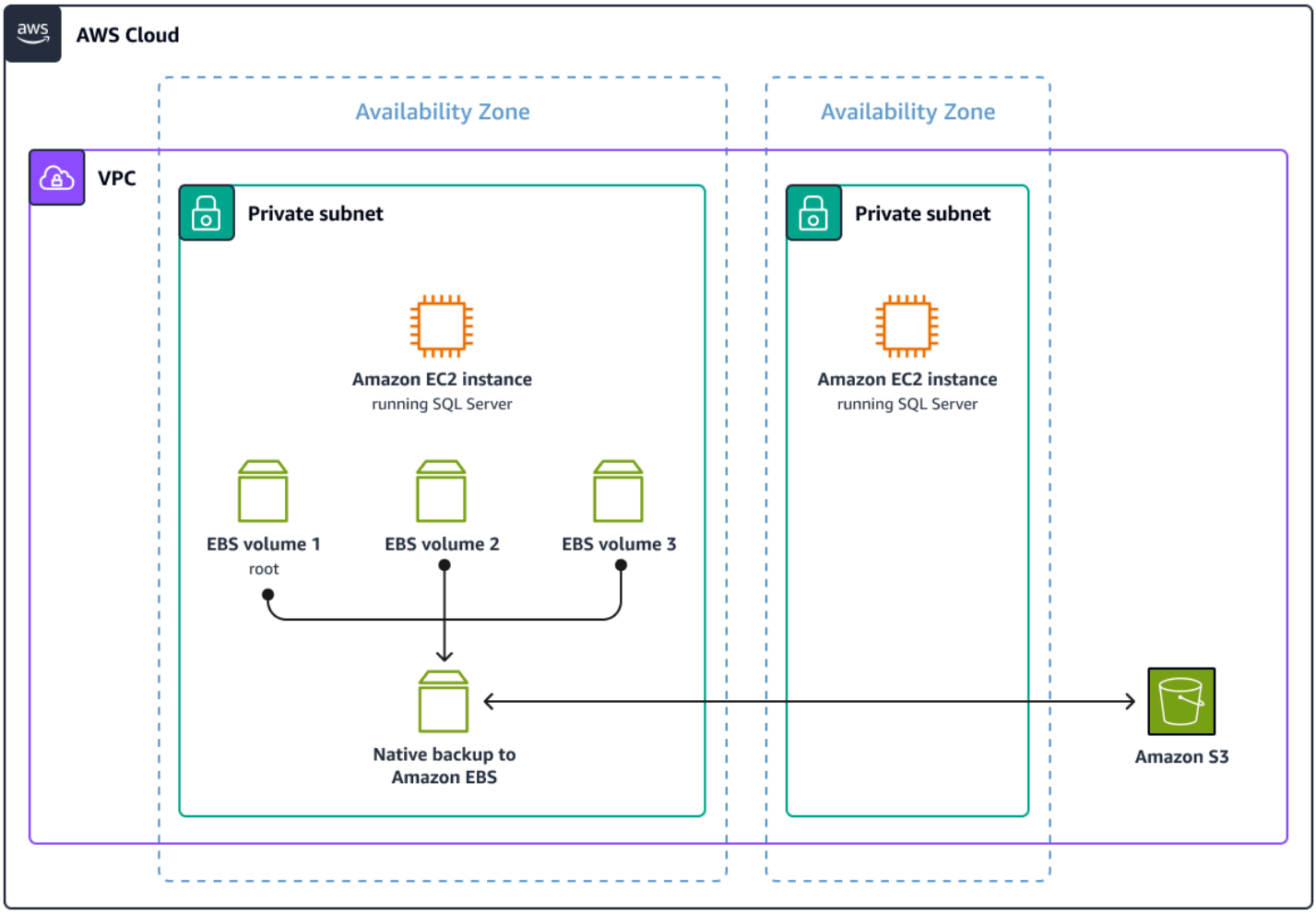
- You must set up backup and restore for each individual database.
- You must manage the Amazon S3 lifecycle policy for the backup files.

For more information about how to set up Storage Gateway, see the Store SQL Server backups in Amazon S3 using AWS Storage Gateway post on the AWS Blog.

SQL Server native backup to EBS volumes

You can take a native backup of your SQL Server database and store the file in an Amazon EBS volume. Amazon EBS is a highly performant block storage service. EBS volumes are elastic, which supports encryption. They can be detached and attached to an EC2 instance. You can back up SQL Server on an EC2 instance on the same EBS volume type or on a different EBS volume type. One advantage of backing up to a different EBS volume is cost savings.

The following diagram shows the architecture of a native backup to an EBS volume.



Consider the following benefits of using SQL Server native backup to EBS volumes:

- You can take backups of individual databases on a SQL Server EC2 instance and restore an individual database instead of having to restore the complete instance.
- Multiple-file backups are supported.
- You can schedule backup jobs by using SQL Server Agent and the SQL Server job engine.
- You can get performance benefits through your hardware choices. For example, you can use st1 storage volumes to achieve higher throughput.

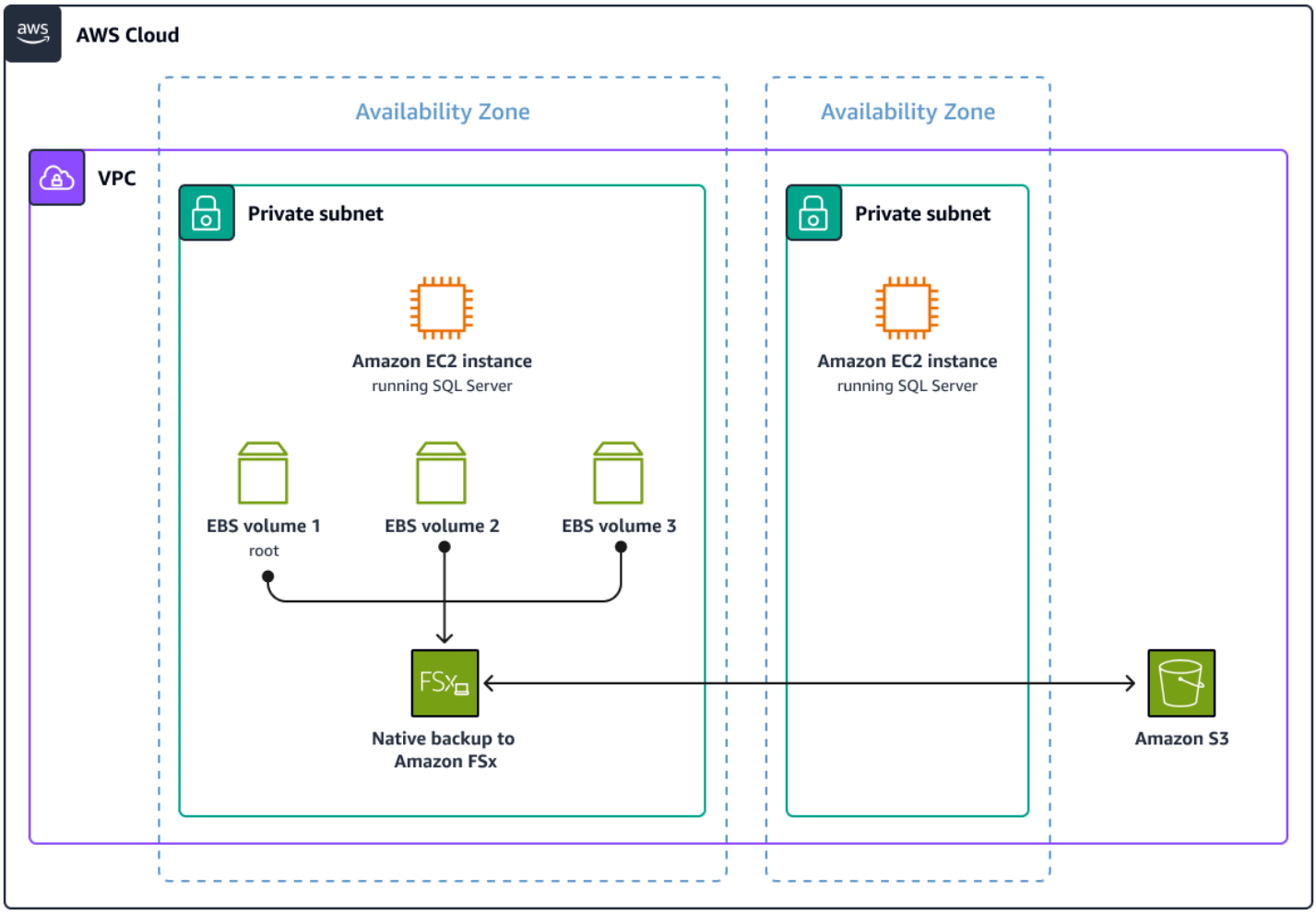
Consider the following limitations of using native backup to EBS volumes:

- You must manually move backups to Amazon S3 from the EBS volume.
- For large backups, you must manage disk space on Amazon EC2.
- On the EC2 instance, Amazon EBS throughput can be a bottleneck.
- Additional storage is required to store backups on Amazon EBS.

SQL Server native backup to Amazon FSx for Windows File Server

Amazon FSx for Windows File Server is a fully managed native Windows file system that offers up to 64 TB of storage designed to deliver fast, predictable, and consistent performance. AWS introduced native support for Multi-AZ file system deployments on FSx for Windows File Server. Native support makes it easier to deploy Windows file storage on AWS with high availability and redundancy across multiple Availability Zones. AWS also introduced support for SMB Continuously Available (CA) file shares. You can use FSx for Windows File Server as backup storage for a SQL Server database.

The following diagram shows the architecture of a native SQL Server backup to FSx for Windows File Server.



Consider the following benefits of using native SQL Server backup to FSx for Windows File Server:

- You can back up your SQL Server database to an Amazon FSx file share.
- You can take backups of individual databases on a SQL Server instance and restore an individual database instead of having to restore the complete instance.
- Multi-part backups are supported.
- You can schedule backup jobs by using SQL Server Agent and the job engine.
- The instances have higher network bandwidth compared to Amazon EBS.

Consider the following limitations of using native SQL Server backup to FSx for Windows File Server:

- You must manually move backups to Amazon S3 from Amazon FSx by using AWS Backup or AWS DataSync.

- Large backups might require additional overhead for disk space management on Amazon FSx.
- EC2 instance network throughput can be a bottleneck.
- Additional storage is required to store backups on FSx for Windows File Server.

SQL Server backup to Amazon FSx for NetApp ONTAP

Snapshots with FSx for ONTAP are always crash consistent, but they require you to quiesce (or pause the I/O of) your database in order to create an application-consistent snapshot. You can use NetApp SnapCenter (an orchestration tool with plug-ins for specific applications, including SQL Server) with FSx for ONTAP to create application-consistent snapshots and protect, replicate, and clone your databases at no additional cost.

NetApp SnapCenter

NetApp SnapCenter is a unified platform for application-consistent data protection. SnapCenter refers to snapshots as backups. This guide adopts that same naming convention. SnapCenter provides a single pane of glass for managing application-consistent backups, restores, and clones. You add a SnapCenter plug-in for your specific database application to create application-consistent backups. The SnapCenter plug-in for SQL Server provides the following functionality that simplifies your data protection workflow.

- Backup and restore options with granularity for full and log backups
- In-place restore and restore to an alternate location

For more information about SnapCenter, see the Protect your SQL Server workloads using NetApp SnapCenter with Amazon FSx for NetApp ONTAP post on the AWS Storage Blog.

Cost optimization for backups

The following options can help you reduce the cost of storing SQL Server backups on AWS.

- Enable SQL Server compression during the creation of the backup file and send the smallest possible file to the storage. For example, a 3:1 compression ratio indicates that you are saving about 66 percent on disk space. To query on these columns, you can use the following Transact-SQL statement: `SELECT backup_size/compressed_backup_size FROM msdb..backupset;`.
- For backups going to S3 buckets, enable the Amazon S3 Intelligent-Tiering storage class to reduce storage costs by 30 percent.

- For backups going to FSx for Windows File Server or FSx for ONTAP, use a single Availability Zone for 50 percent cost savings (as compared to using multiple Availability Zones). For pricing information, see Amazon FSx for Windows File Server Pricing and Amazon FSx for NetApp ONTAP Pricing.
- The most efficient option for SQL Server 2022 is direct backup to Amazon S3. You can save additional costs by avoiding Storage Gateway.

Benchmark test results for backups

This section compares the following options from a cost and performance point of view for a sample 1 TB database, based on the results of performance benchmark testing on the backup solutions covered in this guide.

- **EC2 instance specification** – r5d.8xlarge with Windows Server 2019 and SQL Server 2019 Developer edition
- **Database specification** – 1 TB in size with TDE disabled

The tests were performed with an r5d.8xlarge instance and 1 TB SQL Server database as the source. The source system was configured according to best practices, and the source database contained four data files (250 GB each) and one log file (50 GB) spread across separate gp3 volumes. The SQL Server native `BACKUP` command includes writing to 10 backup files, using compression to optimize backup performance and reduce the amount of data sent across the network and written to the target. In all test cases, storage performance was the bottleneck.

There is an almost endless variety of possible configurations for these types of test. This test focused on optimizing for performance, cost, scalability, and real-world use cases. The following table shows the performance metrics that were captured for the backup target options.

Backup options	Level	Run duration (Appx)	Backup rate	Cost USD per month*
Native backup to local EBS st1 HDD, 2 TB	Database	00:30:46 min	554.7 Mbps	$92.16

Backup options	Level	Run duration (Appx)	Backup rate	Cost USD per month*
Native backup to local EBS SSD gp3, 2 TB	Database	00:22:00 min	512 Mbps	$193.84
Native backup to FSx for Windows File Server HDD, 2 TB @512 Mbps throughput	Database	00:20:58 min	814.0 Mbps	$1,146
Native backup to FSx for Windows File Server SSD, 2 TB @512 Mbps throughput	Database	00:20:00 min	814.0 Mbps	$1,326
Native backup to S3 File Gateway m6i.4xlarge (16 vCPU, 64 GB) with 2 TB gp3	Database	00:23:20 min	731.5 Mbps	$470.42
EBS VSS snapshot	EBS volume	00:00:02 sec 00:00:53 sec	N/A snapshot	$51
AWS Backup (AMI backup)	AMI	00:00:04 sec 00:08:00 min	N/A snapshot	$75

Backup options	Level	Run duration (Appx)	Backup rate	Cost USD per month*
Native SQL Server backup directly to Amazon S3 (SQL Server 2022)	Database	00:12:00 min	731.5 Mbps	First 50 TB / Month, $0.023 per GB $23.55 per month
Native backup to FSx for ONTAP (using SnapCenter)	Database	–	–	$440.20

The preceding table assumes the following:

- Data transfer and Amazon S3 costs are not included.
- Storage price is included in instance pricing.
- The costs are based in the `us-east-1` Region.
- Throughput and IOPS grow by 10 percent with multiple backups that have an overall rate of change of 10 percent over the month.

The test results show that the fastest option is a native SQL Server database backup to FSx for Windows File Server. A backup to Storage Gateway and locally attached EBS volumes is the more cost-efficient option but has slower performance. For server-level backups (AMI), we recommend using AWS Backup for optimal performance, cost, and manageability.

Cost optimization recommendations

Understanding the possible solutions for backing up SQL Server on Amazon EC2 is key to safeguarding your data, ensuring that you meet your backup needs, and putting a plan in place to recover from critical events. The different ways to back up and restore your SQL Server instances and databases explored in this section can help you devise a backup and restore strategy that protects your data and meets your organization's requirements.

This section covers the following backup options:

- Compression
- Amazon S3 Intelligent-Tiering
- Single Availability Zone
- Backup to URL

The guidance provided for each of these options is high level. If you wish to implement any of these recommendations in your organization, we recommend that you reach out to your account team. The team can then engage with a Microsoft Specialist SA to lead the conversation. You can also reach out by emailing optimize-microsoft@amazon.com.

In summary, we recommend the following:

- If you're using SQL Server 2022, then backing up to Amazon S3 is the most cost-efficient option.
- If you're using SQL Server 2019 and earlier SQL Server editions, consider backing up to Storage Gateway backed by Amazon S3 as the most cost-efficient option.

Compression

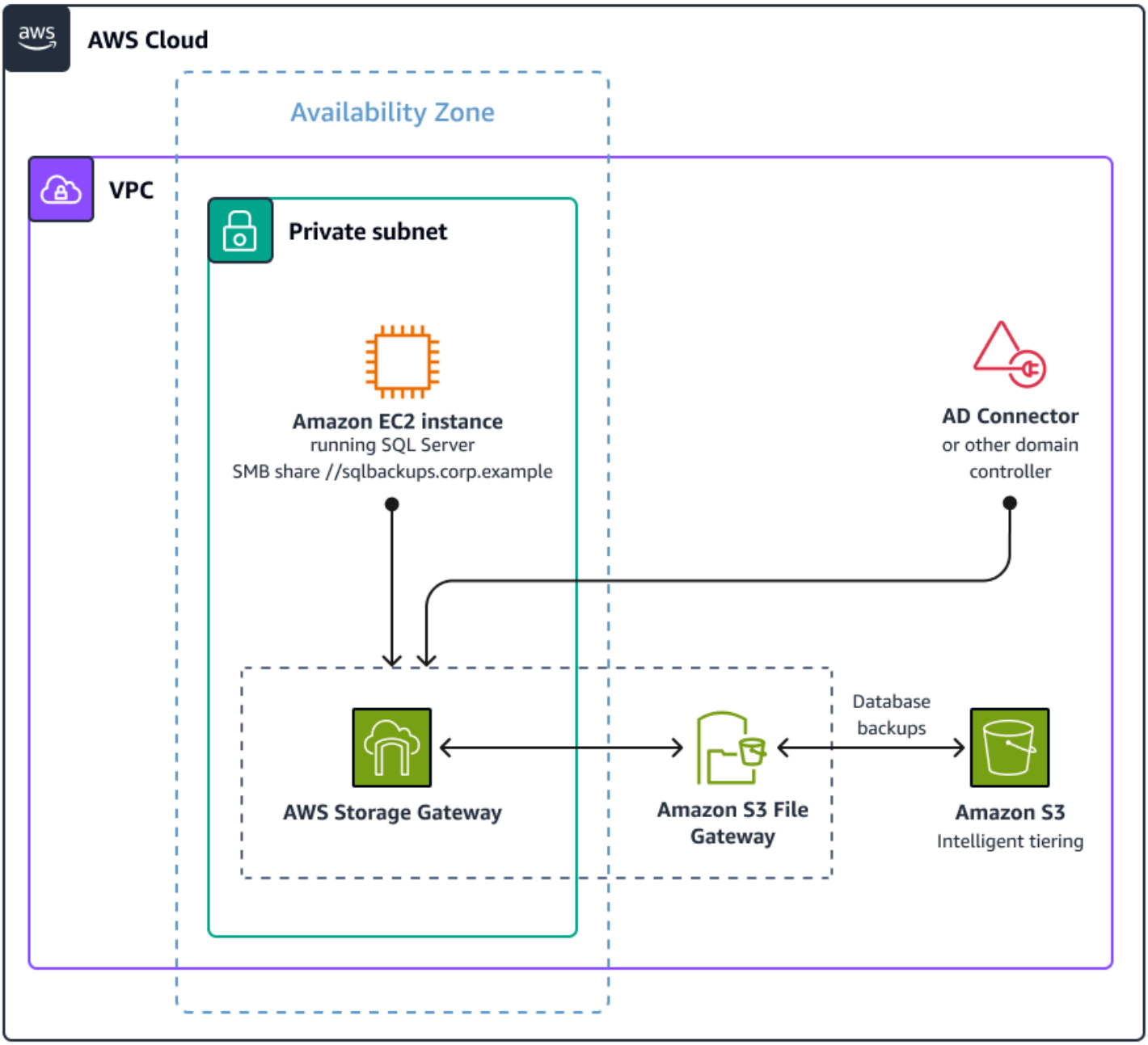
The goal of compression is to have less storage consumed by each backup, which is beneficial for various storage options. You must enable compression for a SQL Server backup at the level of the SQL Server instance. The following example shows how to add the compression keyword with a backup database:

```
BACKUP DATABASE <database_name> TO DISK WITH COMPRESSION (ALGORITHM =
QAT_DEFLATE)
```

Amazon S3 Intelligent-Tiering

For backups going to Amazon S3 buckets, you can enable Amazon S3 Intelligent-Tiering as your Amazon S3 File Gateway storage class. This can reduce storage costs by up to 30 percent. You then mount S3 File Gateway to your SQL servers by using an SMB file share that can be integrated with your Active Directory domain. This provides you with access control for your share, the ability to leverage existing service accounts, and access to Amazon S3 using a common Microsoft focused file protocol. For accounts that might not have direct connectivity to a domain controller, you can use the Active Directory Connector to facilitate communication with Active Directory on-premises or in the cloud. To configure the Active Directory settings on the gateway, you must specify the Active Directory Connector IPs for the domain controller to proxy requests to Active Directory.

The following diagram shows an architecture for a solution based on S3 Intelligent-Tiering.



By default, the backup files written to the S3 bucket use the Standard tier. To convert the backup files from the Standard tier to S3 Intelligent-Tiering, you must create a lifecycle rule. You can also use the AWS Management Console to enable S3 Intelligent-Tiering. For more information, see Getting Started Using Amazon S3 Intelligent-Tiering in the AWS documentation.

Single Availability Zone

To create a Single Availability Zone file system, choose the Single-AZ option when you create an FSx for Windows File Server file system. Amazon FSx also takes highly durable backups (stored in Amazon S3) of your file system daily using the Windows Volume Shadow Copy Service, and allows you to take additional backups at any point. Keep in mind some of the issues with using a Single Availability Zone. For example, the SMB file share becomes inaccessible if an affected Availability Zone where the file system is provisioned goes down for hours at a time. If you require access to the data, you must restore it from backups in an available Availability Zone within the source Region. For more information, see the Use a single Availability Zone section of this guide.

Backup to URL

For SQL Server 2022, the backup to URL feature allows direct backup to Amazon S3. This is the ideal backup approach for SQL Server 2022 running in AWS as you get the full feature set of Amazon S3 at the storage layer and remove the cost of the AWS Storage Gateway appliance needed in prior versions to facilitate this functionality. There are two primary costs to consider when implementing this feature: data transfer costs and the S3 storage class chosen. If you want the native disaster recovery capabilities of Amazon S3, then you must factor in that cross-Region Replication incurs cross-Region data egress costs. To learn more about how to configure this option, see the Backup SQL Server databases to Amazon S3 post on the Microsoft Workloads on AWS blog.

Additional resources

- Backup and restore options for SQL Server on Amazon EC2 (AWS Prescriptive Guidance)
- Point-in-time recovery and continuous backup for Amazon RDS with AWS Backup (AWS Storage Blog)
- Protect your SQL Server workloads using NetApp SnapCenter with Amazon FSx for NetApp ONTAP (AWS Storage Blog)
- Getting Started Using Amazon S3 Intelligent-Tiering (AWS Getting Started Resource Center)
- Backup and Restore Strategies for Amazon RDS for SQL Server (AWS Database Blog)
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2 (AWS Prescriptive Guidance)
- Best Practices for Deploying Microsoft SQL Server on Amazon EC2 (AWS Whitepaper)

Modernize SQL Server databases

Overview

If you're starting on a journey toward modernizing legacy databases for scalability, performance, and cost optimization, you may be facing challenges with commercial databases like SQL Server. Commercial databases are expensive, lock customers in, and offer punitive licensing terms. This section provides a high-level overview of the options for migrating and modernizing from SQL Server to open-source databases and information about choosing the best option for your workload.

You can refactor your SQL Server databases to open-source databases like Amazon Aurora PostgreSQL to save on Windows and SQL Server licensing costs. Cloud-native modern databases like Aurora merge the flexibility and low cost of open-source databases with the robust, enterprise-grade features of commercial databases. If you have variable workloads or multi-tenant workloads, you can also migrate to Aurora serverless V2. This can reduce costs by to 90 percent, depending on workload characteristics. Additionally, AWS offers capabilities like Babelfish for Aurora PostgreSQL, tools like AWS Schema Conversion Tool (AWS SCT), and services like AWS Database Migration Service (AWS DMS) to simplify the migration and modernization of SQL Server databases on AWS.

Database offerings

Migrating from SQL Server on Windows to open-source database like Amazon Aurora, Amazon RDS for MySQL, or Amazon RDS for PostgreSQL can offer significant cost savings without compromise on performance or features. Consider the following:

- Switching from SQL Server Enterprise edition on Amazon EC2 to Amazon RDS for PostgreSQL or Amazon RDS for MySQL can result in cost savings up to 80 percent.
- Switching from SQL Server Enterprise edition on Amazon EC2 to Amazon Aurora PostgreSQL-Compatible Edition or Amazon Aurora MySQL-Compatible Edition can result in cost savings up to 70 percent.

For traditional database workloads, Amazon RDS for PostgreSQL and Amazon RDS for MySQL address requirements and provide a cost-effective solution for relational databases. Aurora adds numerous availability and performance features previously limited to expensive commercial vendors. The resiliency features in Aurora are an added cost. However, in comparison to similar features by other commercial vendors, the resiliency costs of Aurora are still cheaper than what

commercial software charges for the same type of features. Aurora architecture is optimized to deliver significant improvements in performance as compared to standard MySQL and PostgreSQL deployments.

Because Aurora is compatible with open-source PostgreSQL and MySQL databases, there is the additional benefit of portability. Whether the best option is Amazon RDS for PostgreSQL, Amazon RDS for MySQL, or Aurora comes down to understanding business requirements and mapping necessary features to the best option.

Amazon RDS and Aurora comparison

The following table summarizes the key differences between Amazon RDS and Amazon Aurora.

Category	Amazon RDS for PostgreSQL or Amazon RDS for MySQL	Aurora PostgreSQL or Aurora MySQL
Performance	Good performance	3x or better performance
Failover	Typically 60–120 seconds*	Typically 30 seconds
Scalability	Up to 5 read replica	Up to 15 read replicas
	Lag in seconds	Lag in milliseconds
Storage	Up to 64 TB	Up to 128 TB
Storage HA	Multi-AZ with one or two standby, each with database copy	6 copies of data across 3 Availability Zones by default
Backup	Daily snapshot and log backups	Continuous, asynchronous backup to Amazon S3
Innovations with Aurora	NA	100 GB
		Fast database cloning
	Auto-scaling read replicas	
	Query plan management	

Category	Amazon RDS for PostgreSQL or Amazon RDS for MySQL	Aurora PostgreSQL or Aurora MySQL
	Aurora Serverless	
	Cross-Region replicas with Global Database	
	Cluster cache management**	
	Parallel Query	
	Database activity streams	

*Large transactions can increase failover times

**Available in Aurora PostgreSQL

The following table shows the estimated monthly cost of the different database services covered in this section.

Database service	Cost USD per month*	AWS Pricing Calculator (requires AWS account)
Amazon RDS for SQL Server Enterprise edition	$3,750	Estimate
Amazon RDS for SQL Server Standard edition	$2,318	Estimate
SQL Server Enterprise edition on Amazon EC2	$2,835	Estimate
SQL Server Standard edition on Amazon EC2	$1,345	Estimate
Amazon RDS for PostgreSQL	$742	Estimate
Amazon RDS for MySQL	$712	Estimate

Database service	Cost USD per month*	AWS Pricing Calculator (requires AWS account)
Aurora PostgreSQL	$1,032	Estimate
Aurora MySQL	$1,031	Estimate

* Storage price is included in instance pricing. Costs are based on the `us-east-1` Region. The throughput and IOPS are assumptions. The calculations are for r6i.2xlarge and r6g.2xlarge instances.

Cost optimization recommendations

Heterogenous database migrations typically require converting database schema from the source to the target database engine and migrating data from source to target database. The first step toward migration is to evaluate and convert SQL server schema and code objects to the target database engine.

You can use the AWS Schema Conversion Tool (AWS SCT) to evaluate and assess the database for compatibility with various target open-source database options like Amazon RDS for MySQL or Amazon RDS for PostgreSQL, Aurora MySQL, and PostgreSQL. You can also use the Babelfish Compass tool for assessing compatibility with Babelfish for Aurora PostgreSQL. This makes the AWS SCT and Compass powerful tools to understand the upfront work involved before deciding on a migration strategy. Should you decide to proceed, AWS SCT automates the changes required to the schema. The core philosophy behind Babelfish Compass is to allow the SQL database to move to Aurora with no, or very few, modifications. Compass will evaluate the existing SQL database to determine if this can be accomplished. This way, the outcome is known before any effort is spent on migrating data from SQL Server to Aurora.

AWS SCT automates conversion and migration of the database schema and code to the target database engine. You can use Babelfish for Aurora PostgreSQL to migrate your database and application from SQL Server to Aurora PostgreSQL with no or minimal schema changes. This can accelerate your migrations.

After the schema is migrated, you can use AWS DMS to migrate the data. AWS DMS can perform full data load and replicate changes to perform migration with minimal downtime.

This section explores the following tools in more detail:

- AWS Schema Conversion Tool
- Babelfish for Aurora PostgreSQL
- Babelfish Compass
- AWS Database Migration Service

AWS Schema Conversion Tool

You can use AWS SCT to evaluate your existing SQL Server databases and assess compatibility with Amazon RDS or Aurora. To simplify the migration process, you can also use AWS SCT to convert the schema from one database engine to another in a heterogeneous database migration. You can use AWS SCT to evaluate your application and convert embedded application code for applications written C#, C++, Java, and other languages. For more information, see Converting application SQL using AWS SCT in the AWS SCT documentation.

AWS SCT is a free AWS tool that supports many database sources. To use AWS SCT, you point it to the source database and then run an assessment. Then, AWS SCT evaluates the schema and generates the assessment report. Assessment reports include an executive summary, complexity and migration effort, suitable target database engines, and recommendations for conversion. To download AWS SCT, see Installing, verifying, and updating AWS SCT in the AWS SCT documentation.

The following table shows an example Executive Summary generated by AWS SCT to show the complexity involved with changing the database to different target platforms.

Target platform	**Auto or minimal changes**			**Complex actions**			
	Storage objects	**Code objects**	**Conversion actions**	**Storage objects**		**Code objects**	
Amazon RDS for MySQL	60 (98%)	8 (35%)	42	1 (2%)	1	15 (65%)	56
Amazon Aurora MySQL-Com	60 (98%)	8 (35%)	42	1 (2%)	1	15 (65%)	56

patible Edition							
Amazon RDS for PostgreSQL	60 (98%)	12 (52%)	54	1 (2%)	1	11 (48%)	26
Amazon Aurora PostgreSQL-Compatible Edition	60 (98%)	12 (52%)	54	1 (2%)	1	11 (48%)	26
Amazon RDS for MariaDB	60 (98%)	7 (30%)	42	1 (2%)	1	16 (70%)	58
Amazon Redshift	61 (100%)	9 (39%)	124	0 (0%)	0	14 (61%)	25
AWS Glue	0 (0%)	17 (100%)	0	0 (0%)	0	0 (0%)	0
Babelfish	59 (97%)	10 (45%)	20	2 (3%)	2	12 (55%)	30

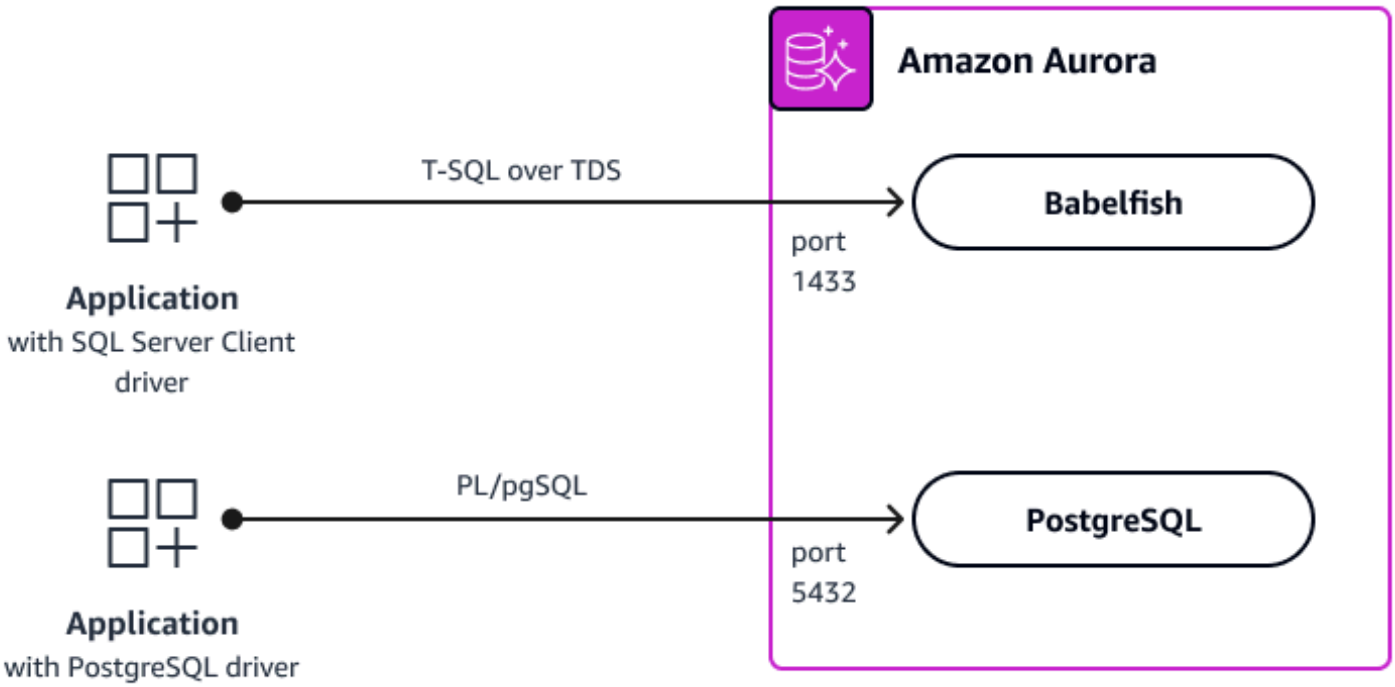
An AWS SCT report also provides details on the schema elements that cannot be automatically converted. You can close the AWS SCT conversion gaps and optimize target schemas by referring to AWS migration playbooks. There are many database migration playbooks to assist with heterogeneous migrations.

Babelfish for Aurora PostgreSQL

Babelfish for Aurora PostgreSQL extends Aurora PostgreSQL with the ability to accept database connections from SQL Server clients. Babelfish enables applications that were originally built for SQL Server to work directly with Aurora PostgreSQL, with few code changes and without changing database drivers. Babelfish turns Aurora PostgreSQL bilingual so that Aurora PostgreSQL can

work with both the T-SQL and PL/pgSQL languages. Babelfish minimizes the efforts to migrate from SQL Server to Aurora PostgreSQL. This accelerates migrations, minimizes risk, and reduces migration costs significantly. You can continue to use T-SQL post migrations, but there is also an option of using PostgreSQL native tools for development.

The following diagram illustrates how an application using T-SQL connects to the default port 1433 in SQL Server and uses the Babelfish translator to communicate with the Aurora PostgreSQL database, while an application using PL/pgSQL can directly and simultaneously connect to the Aurora PostgreSQL database using the default port 5432 in Aurora PostgreSQL.



Babelfish doesn't support certain SQL Server T-SQL features. For this reason, Amazon provides assessment tools to do a line-by-line analysis of your SQL statements and determine if any of them are unsupported by Babelfish.

There are two options for Babelfish assessments. AWS SCT can assess the compatibility of your SQL Server database with Babelfish. Another option is the Babelfish Compass tool, which is a recommended solution because the Compass tool is updated in line with new releases of Babelfish for Aurora PostgreSQL.

Babelfish Compass

Babelfish Compass is a free downloadable tool that aligns with the latest release of Babelfish for Aurora PostgreSQL. In contrast, AWS SCT will support newer Babelfish versions after some

time. Babelfish Compass is run against the SQL Server database schema. You can also extract the source SQL Server database schema by using tools like SQL Server Management Studio (SSMS). Then, you can run the schema through Babelfish Compass. This generates the report detailing the compatibility of SQL Server schema with Babelfish and if any changes are needed before migrating. The Babelfish Compass tool can also automate many of these changes and ultimately accelerate your migrations.

After the assessment and changes are completed, you can migrate the schema to Aurora PostgreSQL by using SQL Server native tools like SSMS or sqlcmd. For instructions, see the Migrate from SQL Server to Amazon Aurora using Babelfish post on the AWS Database Blog.

AWS Database Migration Service

After the schema is migrated, you can use AWS Database Migration Service (AWS DMS) to migrate the data to AWS with minimal downtime. AWS DMS not only does a full data load, but also replicates changes from source to destination while the source system is up and running. After both source and target databases are in sync, the cutover activity can take place where the application is pointed to the target database completing the migration. AWS DMS currently only performs full data load with Babelfish for an Aurora PostgreSQL target and doesn't replicate changes. For more information, see Using Babelfish as a target for AWS Database Migration Service in the AWS DMS documentation.

AWS DMS can do both homogeneous (across the same database engine) and heterogeneous (across different database engines) migrations. AWS DMS supports many source and destination database engines. For more information, see the Migrating your SQL Server database to Amazon RDS for SQL Server using AWS DMS post in the AWS Database Blog.

Additional resources

- Goodbye Microsoft SQL Server, Hello Babelfish (AWS News Blog)
- Convert database schemas and application SQL using the AWS Schema Conversion Tool CLI (AWS Database Blog)
- Migrate SQL Server to Amazon Aurora PostgreSQL using best practices and lessons learned from the field (AWS Database Blog)
- Validate database objects post-migration from Microsoft SQL Server to Amazon RDS for PostgreSQL and Amazon Aurora PostgreSQL (AWS Database Blog)

Optimize storage for SQL Server

Overview

This section focuses on cost optimizations for Amazon Elastic Block Store (Amazon EBS) SSD storage for SQL Server on EC2 workloads.

You have a wide variety of storage options for deploying and running SQL Server workloads on AWS. Selecting the right storage should be based on purpose, architecture, durability, performance, capacity, and cost. AWS customers running SQL Server workloads usually utilize a combination of Amazon EBS, NVMe, Amazon FSx, and Amazon Simple Storage Service (Amazon S3) storage.

Amazon EBS is network attached storage connected to EC2 compute instances and utilized to store and process general operating system, application, database, and backup files. Amazon EBS solid state drive (SSD) storage includes General Purpose SSD (gp2 and gp3) and Provisioned IOPS SSD (io1, io2, and io2BX). Consider the following:

- Some EC2 instances, like r5d, have local NVMe SSDs physically attached to the host instance. These volumes provide block-level storage that's commonly used for SQL Server tempdb or buffer pool extension.
- Amazon FSx for Windows File Server is a fully managed file storage service, while Amazon FSx for NetApp ONTAP is fully managed shared storage built on NetApp's popular ONTAP file system. Amazon FSx is frequently used to run SQL Server workloads in a high availability, SQL Server Failover Clustered Instance (FCI) configuration. This solution hosts SQL Server data and log files, which reduces the EBS performance requirements on EC2 instances.
- Amazon S3 is an object storage service offering industry-leading scalability, data availability, security, and performance. You can store SQL Server native backup files, AMIs, EBS snapshots, application logs, and more on Amazon S3.

SSD storage types, performance, and cost for Amazon EBS

SSD storage costs for Amazon EBS generally increase as durability and performance increase. The storage currently comes in five volume types, each with their own unique performance metrics. For a summary of the use cases and characteristics of SSD-backed volumes, see the table in the Solid state drive (SSD) volumes section of the Amazon EBS documentation.

You can use Amazon CloudWatch to monitor SSD performance, capture trending data, and set alarms when certain thresholds are met. If you're running SQL Server workloads on AWS, consider enabling detailed monitoring and deploying CloudWatch custom metrics to capture detailed volume performance metrics such as disk latency, IOPS, throughput, disk queue length, used vs. free capacity, and more. You can use these CloudWatch performance metrics to identify under-provisioned and over-provisioned storage and provide historic data points to accurately define storage requirements.

SSD storage costs for Amazon EBS also vary based on allocated capacity. The table below shows a comparison of the different volume types. All of the volume types have 1 TB of capacity and similar performance configurations.

Volume type	Max IOPS (16 KiB I/O)	Max throughput (128 KiB I/O)	Price per 1TB	Percent cost savings
gp2	3,000	250	$102.40	
gp3	3,000	250	$86.92	15%
io1	16,000	500	$1,168	
io2	16,000	500	$1,168	
gp3	16,000	500	$146.92	87%
io2bx	16,000	4,000	$1,168	
gp3	16,000	1,000	$181.92	84%

Note

The performance and cost metrics in the preceding table are per volume, based on an estimate from the AWS Pricing Calculator. An AWS account is required to access the estimate in the AWS Pricing Calculator.

Amazon EBS SSD gp3 volumes provide excellent performance at a low cost. You can save up to 87 percent if you choose a gp3 volume over io1 or io2 volumes for workloads requiring less than 16,000 IOPS and 500 MiBps throughput.

io2 Block Express (io2BX) volumes offer increased performance over regular io2 volumes. At 16,000 IOPS, io1 or io2 volumes are only capable of 500 MiBps throughput, while io2BX volumes can be configured up to 4,000 MiBps throughput. Compared to io1 and io2 volumes, io2BX volumes provide more than four times the throughput between 16,000 to 64,000 IOPS, at the exact same price. Regular io2 volumes can be converted to io2BX volumes by attaching them to io2BX-supported EC2 instances. For a list of io2BX-supported EC2 instances, see Provisioned IOPS SSD volumes in the Amazon EBS documentation. Before deploying new storage, you can use the AWS Pricing Calculator to estimate your monthly cost and understand the impact on cost based on the trade-offs between durability, performance, and capacity.

General SSD cost optimization for Amazon EBS

We recommend that you evaluate what you're storing and ensure that you're using the right storage type and class. For example, Amazon S3 provides a great price point, built-in lifecycle policies, and replication options ideal for SQL Server backups. SQL Server 2022 has the ability to backup directly to Amazon S3, while previous versions of SQL Server rely on native local backups. If you're running older versions of SQL Server, consider backing up to Amazon EBS HDD volumes and then copying the backup to Amazon S3. This solution can save 53 percent as opposed to using gp3 volumes for backups.

The following table shows the price difference for 1 TB of storage on Amazon EBS gp3, Amazon EBS HDD st1, and Amazon S3.

Storage type	Capacity	Price pm
EBS gp3 500 MiBps	1 TB	$96.92
EBS st1 burst 500 MiBps		$46.08
S3 Standard		$23.55
S3 Standard (infrequent access)		$12.80
S3 Glacier Deep Archive		$1.03

> **Note**
>
> The cost metrics in the preceding table are based on an estimate in the AWS Pricing Calculator. An AWS account is required to access the estimate in the AWS Pricing Calculator.

We recommend that you consider the following:

- Enable detailed monitoring and deploy CloudWatch custom metrics to accurately capture their storage performance requirements.
- Upgrade Amazon EBS storage from gp2 to gp3 to reduce costs, increase flexibility, and improve performance.
- Upgrade Amazon EBS storage from io1 to io2 for increased durability and performance flexibility.
- Use io2BX instead of io1 or io2 when possible for increased durability and performance.
- Consider a mix-and-match approach when choosing storage to help reduce capacity requirements and the cost of high-performance volumes. For example, you could use low-cost gp3 volumes for your root volume (operating system), SQL Server installation, system databases (excluding tempdb), and lower performing user databases. This could help reduce the capacity and cost of an io2 volume, which can be dedicated to high-performance user databases.
- If you're hosting SQL Server databases on AWS, we recommend that you use multiple SQL Server data files per database. This allows the opportunity to distribute read/write workloads across multiple volumes, reducing performance and capacity requirements per volume and consequently reducing cost.
- Even if production workloads require higher performing storage, such as io1 or io2/io2BX, consider gp3 volumes for non-production workloads to help reduce costs.
- Track and trend storage utilization over time to easily identify usage spikes and unexpected costs.
- Use AWS Compute Optimizer for recommendations on scaling EBS volumes up or down based on actual utilization.
- Use the elasticity of AWS to adjust the performance and capacity needs of your SSD volumes for Amazon EBS. Unlike for on-premises environments, you don't need to overprovision storage performance and capacity for future workloads. You can migrate your existing SQL Server workloads onto AWS and adjust performance or capacity as needed, while keeping your databases online.

Additional resources

- Amazon EBS volume types (Amazon EBS documentation)
- Amazon Elastic Block Store (Amazon EBS) (Amazon EBS documentation)
- Provisioned IOPS SSD volumes (Amazon EBS documentation)
- SSD instance store volumes (Amazon EC2 documentation)
- Amazon CloudWatch metrics for Amazon EBS (Amazon EBS documentation)
- Specifications for Amazon EC2 storage optimized instances (Amazon EC2 documentation)
- Protect your SQL Server workloads using NetApp SnapCenter with Amazon FSx for NetApp ONTAP (AWS Storage Blog)
- Amazon EC2 FAQ (AWS product page)

Optimize SQL Server licensing by using Compute Optimizer

Guidance on how to optimize licenses for SQL Server by using AWS Compute Optimizer.

Overview

AWS Compute Optimizer can recommend licensing optimization opportunities for Microsoft SQL Server workloads on Amazon Elastic Compute Cloud (Amazon EC2). Compute Optimizer can provide automated recommendations to reduce licensing costs. The recommendations from Compute Optimizer are listed next to each of your EC2 instances with Microsoft SQL Server licenses. The information that's provided includes recommended saving opportunities, EC2 instance On-Demand prices, and hourly bring your own license (BYOL) prices. This information can help you decide if you should downgrade your license edition.

Compute Optimizer automatically discovers your SQL Server instances on Amazon EC2 by inferred workload type. To view the licensing recommendations, you can select the SQL Server instances in Compute Optimizer and then authenticate with Amazon CloudWatch Application Insights by using your read-only database credentials. Compute Optimizer analyzes if you are using any SQL Server Enterprise edition features. If no Enterprise edition features are being used, Compute Optimizer recommends that you downgrade to Standard edition to reduce licensing costs.

You can also use Compute Optimizer to make sizing recommendations for your Amazon EC2 instances that run SQL Server workloads. For more information, see Optimize SQL Server sizing by using Compute Optimizer in this guide.

Cost optimization recommendations

The license recommendations in Compute Optimizer can help you evaluate the features you are using in Microsoft SQL Server and choose the most cost-effective edition for your workloads. SQL Server Enterprise edition is significantly more expensive than Standard edition. For more information, see Compare SQL Server editions in this guide and see SQL Server 2022 pricing on the Microsoft website. Investing the time to configure Compute Optimizer to evaluate your SQL Server fleet and provide recommendations can dramatically reduce your licensing costs.

The **License details** page provides the following information:

- Use the table to compare your current license settings (such as edition, model, and number of instance cores) with Compute Optimizer recommendations.
- Use the utilization graphs to review the number of Enterprise edition features that were used during the analysis period.

For more information, see Viewing details of a commercial software license recommendation in the Compute Optimizer documentation.

Configure Compute Optimizer

Compute Optimizer analyzes commercial software licenses by using the `mssql_enterprise_features_used` metric. For more information about this metric, see Metrics for commercial software licenses.

1. Make sure that you have the appropriate permissions to opt in to Compute Optimizer. For more information, see the following:
   - Policy to opt in to Compute Optimizer
   - Policies to grant access to Compute Optimizer for standalone AWS accounts
   - Policies to grant access to Compute Optimizer for a management account of an organization
2. Attach the required instance roles and policy for CloudWatch Application Insights. For instructions, see Policies to enable commercial software license recommendations.
3. Enable CloudWatch Application Insights by using your Microsoft SQL Server database credentials. For instructions, see Set up application for monitoring in the CloudWatch documentation.

> **Note**
>
> To generate recommendations for commercial software licenses, at least 30 consecutive hours of CloudWatch metric data is required. For more information, see CloudWatch metric requirements.

4. Use the following SQL query to configure least-privilege access for CloudWatch Application Insights.

```
GRANT VIEW SERVER STATE TO [LOGIN];
GRANT VIEW ANY DEFINITION TO [LOGIN];
```

   This enables a new service, PrometheusSqlExporterSQL.

5. From the target AWS account or organization management account, opt in to Compute Optimizer. For instructions, see Opting in your account.

> **Note**
>
> After you opt in, findings and optimization recommendations can take up to 24 hours to be generated.

6. In the Compute Optimizer console, choose **Licenses** in the navigation pane.
7. In the **Findings** column, search for any instances that have the **Insufficient metrics** finding. Compute Optimizer returns this finding if it detects that CloudWatch Application Insights isn't enabled or has insufficient permissions. For more information, see Finding reasons. Do the following to resolve these findings:
   a. Choose the instance.
   b. Add a secret.
   c. Confirm the instance role and policy are attached.
   d. Choose **Enable license recommendations**.
8. In the **Findings** column, search for any instances that have the **Not optimized** finding. Compute Optimizer returns this finding if it detects that your Amazon EC2 infrastructure isn't using any of the Microsoft SQL Server license features that you're paying for. For more information, see Finding reasons. Do the following to resolve these findings:
   a. Choose the instance.

b. Compare the current license edition with the recommended edition.
c. Review the current license utilization graph.
d. If you want to downgrade the license, choose **Implement recommendation**.
e. Review the requirements and follow the instructions to downgrade the license. If you want to automate the process, see Downgrade SQL Server Enterprise edition using AWS Systems Manager Document to reduce cost (AWS Blog).

Additional resources

- Reduce Microsoft SQL Server licensing costs with AWS Compute Optimizer (AWS Blog)
- What is AWS Compute Optimizer? (AWS documentation)
- Viewing commercial software license recommendations (AWS documentation)
- Downgrade your Microsoft SQL Server edition (AWS documentation)
- Microsoft SQL Server on AWS (AWS)
- Microsoft Licensing on AWS (AWS)
- Microsoft SQL Server 2019 Pricing (Microsoft)
- Microsoft SQL Server 2022 Pricing (Microsoft)

Optimize SQL Server sizing by using Compute Optimizer

Overview

AWS Compute Optimizer helps database administrators (DBAs) discover Microsoft SQL Server workloads on Amazon Elastic Compute Cloud (Amazon EC2) and rightsize EC2 instances to reduce license costs by up to 25%. The inferred workload type feature in Compute Optimizer uses machine learning (ML) and automatically detects the applications that might be running on your AWS resources. Compute Optimizer includes support for SQL Server as an inferred workload type. By using the inferred workload type feature, you can pinpoint cost-saving opportunities based on the specific workload running on your Amazon EC2 instances.

With this feature, you can categorize cost-saving opportunities by supported inferred workload types, such as SQL Server. Compute Optimizer can automatically discover SQL Server EC2 instances that are over-provisioned. You can switch to the EC2 console to downsize the instance, which helps reduce the licensing and infrastructure costs.

You can also use Compute Optimizer to make SQL Server licensing recommendations. For more information, see Optimize SQL Server licensing by using Compute Optimizer in this guide.

Configure Compute Optimizer

For instructions for using Compute Optimizer with SQL Server inferred workloads, see Optimizing performance and reducing licensing costs: Leveraging AWS Compute Optimizer for Amazon EC2 SQL Server instances (AWS Blog). You can opt in for standalone accounts, accounts that are a member of an organization, and management accounts of an organization. For standalone and member accounts, opting in enables Compute Optimizer for that account only. For an organization management account, you can choose whether to enable Compute Optimizer in that account only or for all member accounts of the organization.

The Compute Optimizer opt-in process automatically creates an AWS Identity and Access Management (IAM) service-linked role. For more information, see Using service-linked roles for AWS Compute Optimizer.

Compute Optimizer analyzes your resources based on Amazon CloudWatch metrics, such as CPU, I/O, network, and Amazon Elastic Block Store (Amazon EBS) usage. To generate recommendations, at least 30 consecutive hours of CloudWatch metric data is required in the past 14 days. If you enable the enhanced infrastructure metrics feature, it extends the utilization metrics to 93 days. For more information, see CloudWatch metric requirements and Enhanced infrastructure metrics in the Compute Optimizer documentation.

Compute Optimizer provides options and the savings associated with each option, based on vCPU, memory, storage, network, risk, and migration effort. You can use the CloudWatch metrics dashboard to analyze the data being used to make the recommendation. With this data, you can rightsize your EC2 instances that are running SQL Server workloads. For more information about how to change your instance type, see Change the instance type in the Amazon EC2 documentation.

Additional resources

- AWS Compute Optimizer identifies and filters Microsoft SQL Server workloads (AWS)
- Optimizing performance and reducing licensing costs: Leveraging AWS Compute Optimizer for Amazon EC2 SQL Server instances (AWS Blog)
- What is AWS Compute Optimizer? (AWS documentation)
- Viewing EC2 instance recommendations (AWS documentation)

Review Trusted Advisor recommendations for SQL Server workloads

Overview

AWS Trusted Advisor provides recommendations that help you follow AWS best practices. By analyzing your usage, configuration, and spend, Trusted Advisor provides actionable recommendations to reduce your costs, improve system availability and performance, or help close security gaps. This section focuses on Trusted Advisor checks that can help you reduce the costs of operating SQL Server workloads in the AWS Cloud.

Cost optimization recommendations

Trusted Advisor provides recommendations that help you optimize your SQL Server workloads on Amazon Elastic Compute Cloud (Amazon EC2). The checks inspect your SQL Server workloads and automatically list the instances that need optimization. Operationalizing Trusted Advisor recommendations can reduce costs and improve the security posture of your organization.

The following are Trusted Advisor checks that focus on Microsoft SQL Server:

- Amazon EC2 instances over-provisioned for Microsoft SQL Server – This check analyzes your Amazon EC2 instances that are running SQL Server and alerts you if an instance exceeds the SQL Server software vCPU limit. For example, an instance with SQL Server Standard edition can use up to 48 vCPUs. An instance with SQL Server Web can use up to 32 vCPUs.

Edition	vCPU Min.	vCPU Max.
Web	4	32
Standard	4	48
Enterprise	4	OS Limits

- Amazon EC2 instances consolidation for Microsoft SQL Server – This check analyzes your Amazon EC2 instances and alerts you if your instance has less than the minimum number of SQL Server licenses. You can consolidate smaller SQL Server instances to help lower costs. If you have many small, license-included SQL Server instances, then consider consolidating. According to the Microsoft SQL Server 2019 licensing guide, SQL Server requires a minimum of 4 vCPU

licenses per instance. If you consolidate these databases, you can save on licensing costs. You can make your decision based on the number of databases on the instance, the maximum database size, and the total size of the databases. Consolidation is supported for Web, Standard, and Enterprise editions of SQL Server. For more information, see Consolidating SQL Server Databases (Microsoft blog post).

AWS does not recommend putting large production databases on only one server. However, you can consolidate smaller ones used for non-production environments, such as for development, testing, and staging. This depends on your current SQL Server usage; if you have low-usage databases, you can consolidate on one server.

Configure Trusted Advisor

Do the following to evaluate SQL Server focused checks in Trusted Advisor.

1. Sign in to the AWS Management Console.
2. Open the AWS Trusted Advisor console.
3. In the navigation pane, under **Recommendations**, choose **Cost optimization**.
4. In the **Cost optimization checks** list, review the status of the **Amazon EC2 instances consolidation for Microsoft SQL Server** and **Amazon EC2 instances over-provisioned for Microsoft SQL Server** checks.
   - Green check symbols indicate that your Amazon EC2 instances are optimally configured.
   - Orange alert symbols indicate that there are opportunities for improvement.
5. Choose a check to see its details and recommendations.
6. Follow the instructions provided by the check to optimize your Amazon EC2 instances that are running SQL Server workloads.
7. Monitor your instances regularly, and refresh the checks periodically.

Additional resources

- Trusted Advisor check reference (AWS documentation)
- Microsoft SQL Server on AWS (AWS)
- Microsoft Licensing on AWS (AWS)
- SQL Server 2019 pricing (Microsoft)

- AWS Launch Wizard for SQL Server (AWS documentation)

Containers

Modernization is a transformational journey that offers many options, including decomposing monoliths to microservices, re-architecting applications to be event-driven by using serverless functions (AWS Lambda), and repurposing databases from SQL Server to Amazon Aurora or purpose-built managed databases. The modernization pathways to replatform .NET Framework applications to Linux and Windows containers require less effort than other modernization options. Containers offer the following benefits:

- **Accelerate innovation** – Moving to containers makes it easier to automate stages of the development lifecycle that include building, testing, and deploying applications. By automating these processes, development and operations teams have more time to focus on innovating.
- **Reduce total cost of ownership (TCO)** – Moving to containers can also reduce your reliance on license management and endpoint protection tools. Because containers are ephemeral units of compute, you can automate and simplify management tasks such as patching, scaling, and backup and restore. This reduces the TCO of administering and operating container-based workloads. Finally, containers are more efficient in comparison to virtual machines because you can use containers to maximize the placement of your applications by providing better isolation. This increases the utilization of your application's infrastructure resources.
- **Improve resource utilization** – Containers are more efficient in comparison to virtual machines because you can use containers to maximize the placement of your applications. This increases the utilization of your application's infrastructure resources by providing better isolation.
- **Close the skills gap** – AWS offers immersion days to upskill your development teams on container technology and DevOps practices.

This section covers the following topics:

- Move Windows applications to containers
- Optimize costs for AWS Fargate tasks on Amazon ECS
- Gain visibility into your Amazon EKS costs
- Replatform Windows applications with App2Container

For licensing information, see the *Licensing* section of Amazon Web Services and Microsoft: Frequently Asked Questions or email your questions to microsoft@amazon.com.

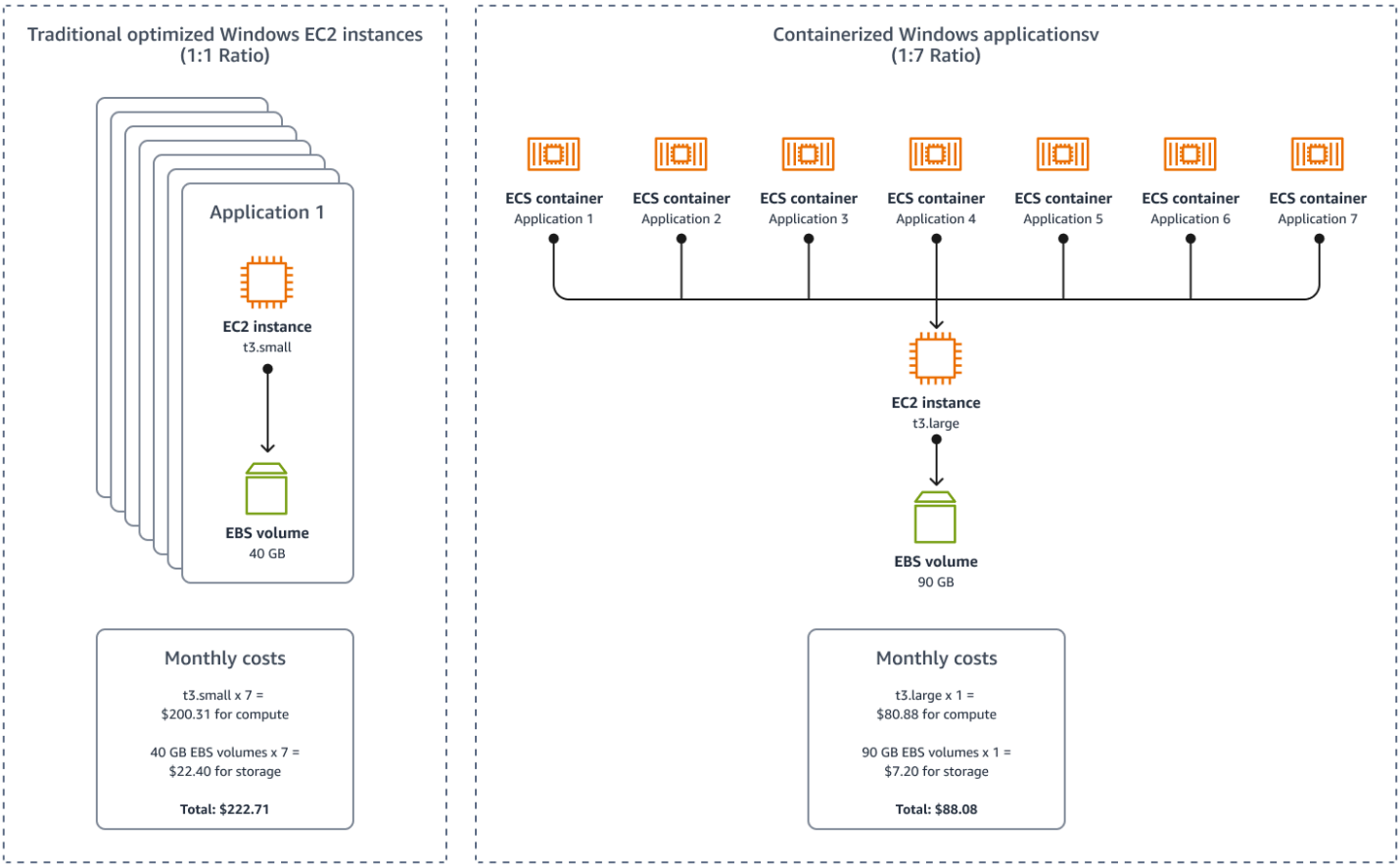
Move Windows applications to containers

Overview

According to the CNCF Annual Survey 2021, 96 percent of organizations are either using or evaluating containers to modernize their infrastructure. This is because containers can help your organization reduce risk, increase operational efficiency and speed, and enable agility. You can also use containers to reduce the cost of running your applications. This section offers recommendations for cost-effectively running containers across AWS container services, including Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and AWS Fargate.

Cost benefits

The following infographic shows the cost savings that a business can achieve by consolidating their ASP.NET Framework applications onto Amazon Elastic Compute Cloud (Amazon EC2) instances based on an AWS Optimization and Licensing Assessment (AWS OLA) recommendation. The following infographic shows what additional savings can be achieved by moving applications to a Windows container.



The AWS OLA recommended that the business do a lift and shift to individual t3.small instances. The business could accomplish these savings by running seven ASP.NET applications on on-premises servers, as the following performance utilization analysis shows.

	Server name	Storage	Operating system	On-premises CPU AVG utilization	On-premises CPU peak utilization	On-premises RAM (GB)	On-premises RAM AVG utilization (GB)	On-premises RAM peak utilization (GB)	Instance size	vCPU	RAM (GB)
		Not sharable	Nearing end of life	Approximately 15% overhead		The OS accounts for 1.2 GB per instance			Optimized for EC2		
1	AppServer01	60	Windows Server 2012	7.00%	17.00%	8	13.50%	17.10%	t3.small	2	2
2	AppServer02	39	Windows Server 2012	20.07%	22.00%	16	7.50%	12.40%	t3.small	2	2
3	AppServer03	39	Windows Server 2012	24.00%	25.50%	16	8.80%	11.90%	t3.small	2	2
4	AppServer04	4	Windows Server 2012	21.40%	24.00%	16	7.80%	10.70%	t3.small	2	2
5	AppServer05	40	Windows Server 2012	21.30%	23.00%	16	8.20%	12.00%	t3.small	2	2
6	AppServer06	39	Windows Server 2012	21.50%	23.50%	16	7.90%	10.90%	t3.small	2	2
7	AppServer07	39	Windows Server 2012	21.60%	22.90%	16	8.40%	11.50%	t3.small	2	2

Further analysis revealed that the business could save even more on costs by running its workloads on containers. Containers reduce the operating system overhead on system resources like CPU, RAM, and disk usage (explained in the next section). In this scenario, the business could consolidate all seven applications onto one t3.large instance and still have 3 GB RAM to spare. Migrating to

containers can help the business achieve an average of 64 percent cost savings across compute and storage by using containers instead of Amazon EC2.

Cost optimization recommendations

The following section offers recommendations for optimizing costs by consolidating applications and using containers.

Reduce your Windows on Amazon EC2 footprint

Windows containers can reduce your Windows on Amazon EC2 footprint by enabling you to consolidate more applications onto fewer EC2 instances. For example, assume that you have 500 ASP.NET applications. If you're running one core per one application for Windows on Amazon EC2, that equals 500 Windows instances (t3.small). If you assume a 1:7 ratio (which can significantly increase depending on EC2 instance type/size) for using Windows containers (with t3.large), then you only need approximately 71 Windows instances. That represents an 85.8 percent decrease in your Windows on Amazon EC2 footprint.

Reduce Windows licensing costs

If you license a Windows instance, you don't need to license containers running on that instance. As a result, consolidating your ASP.NET applications using Windows containers can significantly reduce your Windows licensing costs.

Reduce your storage footprint

Every time you launch a new EC2 instance, you create and pay for a new Amazon Elastic Block Store (Amazon EBS) volume to house the operating system. As this scales, the cost scales with it. If you use containers, you can reduce storage costs because all the containers share the same base operating system. Additionally, containers use the concept of layers to reuse immutable portions of a container image for all running containers based on that image. In the preceding example scenario, all the containers are running .NET Framework and therefore all share the intermediate and immutable ASP.NET framework layer.

Migrate end-of-support servers to containers

Support for Windows Server 2012 and Windows Server 2012 R2 ended on October 10, 2023. You can migrate your applications running on Windows Server 2012 or previous versions by containerizing them to run on new operating systems. This way, you avoid running your

applications on non-compliant operating systems while taking advantage of the cost efficiencies, reduced risk, operational efficiency, speed, and agility that containers provide.

A caveat to consider with this approach is if your application requires specific APIs related to the operating system version currently in use (COM Interop, for example). In this case, you must test moving your application to a newer Windows version. Windows containers align their base container image (for example, Windows Server 2019) with the operating system of the container host (for example, Windows Server 2019). Testing and moving to containers can enable easier operating system upgrades in the future by changing the base image within your Dockerfile and deploying to a fresh set of hosts running the latest version of Windows.

Remove third-party management tools and licenses

Managing your server fleet requires using several third-party system operation tools for patching and configuration management. These can make infrastructure management complex and you often incur third-party licensing costs. If you use containers on AWS, you don't need to manage anything on the operating system side. The container runtime manages the containers. This means the underlying host is ephemeral and can easily be replaced. You can run your containers without the need for directly managing the container host. Additionally, you can use free tools like AWS Systems Manager Session Manager to easily access the host and troubleshoot issues.

Improve control and portability

Containers give you more granular control over server resources like CPU and RAM than you have over EC2 instances. For EC2 instances, you can control CPU and RAM by selecting an instance family, instance type, and CPU options. However, with containers, you can define exactly how much CPU or RAM you want to allocate to a container in your ECS task definitions or to pods in Amazon EKS. In fact, we recommend specifying container-level CPU and memory for Windows containers. This level of granularity brings cost benefits. Consider the following example code:

```
json
{
    "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/demo-
service:1",
    "containerDefinitions": [
        {
            "name": "demo-service",
            "image": "mcr.microsoft.com/dotnet/framework/samples:aspnetapp-
windowsservercore-ltsc2019",
            "cpu": 512,
```

```
            "memory": 512,
            "links": [],
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 0,
                    "protocol": "tcp"
                }
            ],
```

Accelerate innovation

Moving to containers makes it easier to automate stages of the development lifecycle that include building, testing, and deploying applications. If you automate these processes, then you give your development and operations teams more time to focus on innovating.

Reduce TCO

Moving to containers often reduces the reliance on license management and endpoint protection tools. Because containers are ephemeral units of compute, you can automate and simplify management tasks such as patching, scaling, and backup and restore. This can reduce the TCO of administering and operating container-based workloads. Containers are more efficient in comparison to virtual machines because they allow you to maximize the placement of your applications so that you can increase the utilization of your application's infrastructure resources.

Close the skills gap

AWS offers programs and immersion days to upskill customer development teams on containers and DevOps technology. This includes hands-on consulting and enablement.

Refactor to .NET 5+ and use Linux containers

While you can reduce costs by moving your .NET Framework applications to containers, you can realize even further cost savings when you refactor legacy .NET applications to cloud-native alternatives on AWS.

Remove licensing costs

Refactoring your application from .NET Framework on Windows to .NET Core on Linux results in a cost savings of approximately 45 percent.

Access the latest enhancements

Refactoring your applications from .NET Framework on Windows to .NET Core on Linux gives you access to the latest enhancements such as Graviton2. Graviton2 offers 40 percent better price for performance over comparable instances.

Improve security and performance

Refactoring your application from .NET Framework on Windows to .NET Core on Linux containers brings improvements to security and performance. This is because you get the latest security patches, benefit from container isolation, and have access to new features.

Use Windows containers instead of running many applications on one instance of IIS

Consider the following advantages of using Windows containers instead of running multiple applications on one EC2 Windows instance with Internet Information Services (IIS):

- **Security** – Containers provide a level of security out of the box that isn't achieved through isolation at the IIS level. If one IIS website or application is compromised, all the other hosted sites are exposed and vulnerable. Container escape is rare and a harder vulnerability to exploit than gaining control of a server through a web vulnerability.
- **Flexibility** – The ability to run containers in process isolation and have their own instance allows for more granular networking options. Containers also offer complex distribution methods across many EC2 instances. You don't get these benefits when you consolidate applications on a single IIS instance.
- **Management overhead** – Server Name Indication (SNI) creates overhead that requires management and automation. Also, you have to grapple with typical operating system management operations like patching, troubleshooting BSOD (if auto scaling isn't in place), endpoint protection, and so on. Configuring IIS sites according to security best practices is a time consuming and ongoing activity. You might even need to set up trust levels, which also adds to management overhead. Containers are designed to be stateless and immutable. Ultimately, your deployments are faster, more secure, and repeatable if you use Windows containers instead.

Next steps

Investing in modern infrastructure to run your legacy workloads brings immense benefits to your organization. AWS container services make it easier to manage your underlying infrastructure,

whether on premises or in the cloud, so you can focus on innovation and your business needs. Nearly 80 percent of all containers in the cloud run on AWS today. AWS provides a rich set of container services for just about all use cases. To get started, see Containers at AWS.

Additional resources

- Optimize cost for container workloads with ECS capacity providers and EC2 Spot Instances (AWS Blog)
- Cost Optimization Checklist for Amazon ECS and AWS Fargate (AWS Blog)
- Amazon EKS on AWS Graviton2 generally available: considerations on multi-architecture apps (AWS Blog)
- Cost optimization for Kubernetes on AWS (AWS Blog)
- Optimizing your Kubernetes compute costs with Karpenter consolidation (AWS Blog)

Optimize costs for AWS Fargate tasks on Amazon ECS

Overview

Right sizing AWS Fargate tasks is an important step for cost optimization. Too often, applications are built with arbitrary sizing for Fargate tasks and never revisited. This can cause overprovisioning of Fargate tasks and unnecessary spending. This section shows you how to use AWS Compute Optimizer to deliver actionable recommendations so you can optimize task CPU and memory for your Amazon Elastic Container Service (Amazon ECS) services running on Fargate. Compute Optimizer also quantifies the cost impact of adopting these recommendations. This enables you to prioritize your optimization efforts based on the size of the savings opportunity. Compute Optimizer recommendations provide container-level CPU and memory configurations for downsizing tasks.

Cost benefits

Right sizing Amazon ECS tasks on Fargate can reduce costs by 30–70 percent for long running tasks. Without reviewing application performance metrics to right size your task size, you can apply the same mindset used on EC2 compute instances to container sizing. This leads to oversized Fargate tasks that increase costs for idle resources. You can use Compute Optimizer to surface the right sizing opportunities reactively. Ideally, the application owner reviews the specific application performance metrics and removes the operating system overhead to ensure the proper task size is

specified. For more information, see the Move Windows applications to containers section of this guide.

Cost optimization recommendations

This section offers recommendations for using Compute Optimizer to right size your Amazon ECS on Fargate tasks.

As part of the cost-optimization process, we recommend that you do the following:

- Enable Compute Optimizer
- Consume Compute Optimizer results
- Tag tasks to be right sized
- Enable the cost allocation tag to work with AWS billing tools
- Implement right sizing recommendations
- Review before and after costs in Cost Explorer

Enable Compute Optimizer

You can enable AWS Compute Optimizer at the organization or single account level in AWS Organizations. The organization-wide configuration provides ongoing reports for new and existing instances across your entire fleet for all member accounts. This enables right sizing to be a recurring activity instead of a point-in-time activity.

Organization level

For most organizations, the most efficient way to use Compute Optimizer is at the organization level. This provides multi-account and multi-Region visibility into your organization and centralizes the data into one source for review. To enable this at the organization level, do the following:

1. Sign in to your AWS Organizations management account with a role that has the required permissions and choose to opt in for all accounts within this organization. Your organization must have all features enabled.
2. After you enable the management account, you can sign in to the account, see all other member accounts, and browse their recommendations.

> **Note**
>
> It's a best practice to configure a delegated administrator account for Compute Optimizer. This enables you to exercise the principle of least privilege, minimizing access to the AWS Organizations management account while still providing access to the organization-wide service.

Single account level

If you're targeting an account with high costs but don't have access to AWS Organizations, you can still enable Compute Optimizer for that account and Region. To learn about the opt-in process, see Getting started with AWS Compute Optimizer.

> **Note**
>
> The recommendations are refreshed daily and they can take up to 12 hours to generate. Keep in mind that Compute Optimizer requires 24 hours of metrics in the past 14 days to generate recommendations for Amazon ECS on Fargate. For more information, see Requirements for Amazon ECS services on Fargate in the Compute Optimizer documentation.

Compute Optimizer automatically analyzes the following Amazon CloudWatch and Amazon ECS utilization metrics for your Amazon ECS services on Fargate:

- `CPUUtilization` – The percentage of CPU capacity that's used in the service.
- `MemoryUtilization` – The percentage of memory that's used in the service.

Consume Compute Optimizer results

Consider an example that focuses on making right sizing changes within a single account and single Region. In this example, Compute Optimizer is enabled at the organization level across all accounts. Keep in mind that right sizing is a disruptive process that in most cases is carried out with precision by the application owners during a scheduled maintenance window over several weeks.

If you navigate to Compute Optimizer from within an organization's management account (as shown in the following steps), you can choose the account that you want to investigate. In this

example, one task is running in a single account that's overprovisioned in `us-east-1`. The focus is on resizing to the recommended size for the Amazon ECS service.

1. Open the Compute Optimizer console.
2. On the **Dashboard** page, filter by **Findings=Over-provisioned** to see all the Amazon ECS services on Fargate.
3. To review detailed recommendations for **Over-provisioned ECS services on Fargate**, scroll down and then choose **View recommendations**.
4. Choose **Export** and save the file for future use.

> **Note**
>
> To save recommendations for future review, you must have an S3 bucket available for Compute Optimizer to write to in each Region. For more information, see Amazon S3 bucket policy for AWS Compute Optimizer in the Compute Optimizer documentation.

To see recommendations from Compute Optimizer, do the following:

1. In the Compute Optimizer console, go the **Export recommendations** page.
2. For **S3 bucket destination**, choose your S3 bucket.
3. In the **Export filters** section, for **Resource type**, choose **ECS services on Fargate**.
4. On the **Recommendations for ECS services on Fargate** page, drill into one of the ECS services on Fargate and see the CPU and memory recommendations from Compute Optimizer. For example, review the recommendations in the **Compare current settings with recommend task size** and **Compare current settings with recommended container size** sections.

To get the list of ECS services for Fargate that you need to right size, do the following:

1. Open the Amazon S3 console.
2. In the navigation pane, choose **Buckets**, and then choose the bucket where you exported your results.
3. On the **Objects** tab, select your object and choose **Download**.
4. In your downloaded results, filter the finding column to show only **OVER_PROVISIONED** Amazon ECS services on Fargate. This shows the Amazon ECS services that you plan to target for right sizing.

5. Store the task definitions in a text editor for use later.

Right sizing tag tasks

Tagging your workloads is a powerful tool for organizing your resources in AWS. You can use tags to gain fine-grain visibility into costs and enable chargeback. There are many methods and strategies for adding tags to AWS resources to handle chargeback and automation. For more information, see the AWS Whitepaper Best Practices for Tagging AWS Resources. The following example uses AWS CloudShell to tag all the tasks that are part of any Amazon ECS service within the target account and AWS Region.

```
#!/bin/bash
# Set variables
TAG_KEY="rightsizing"
TAG_VALUE="enabled"
# Get a list of ECS Clusters
ClustersArns=$( aws ecs list-clusters –query 'clusterArns' –output text)
for ClustersArn in $ClustersArns; do
 ServiceArns=$( aws ecs list-services –cluster $ClustersArn –query 'serviceArns' –
output text)
 for ServiceArn in $ServiceArns; do
  TasksArns=$( aws ecs list-tasks –cluster $ClustersArn –service-name $ServiceArn –
query 'taskArns' –output text)
  for TasksArn in $TasksArns; do
    aws ecs tag-resource –resource-arn $TasksArn –tags key=$TAG_KEY,value=$TAG_VALUE
  done
 done
done
```

The following code example shows how to enable tag propagation to all Amazon ECS services.

```
#!/bin/bash
# Set variables
TAG_KEY="rightsizing"
TAG_VALUE="enabled"
# Get a list of ECS Clusters
ClustersArns=$(aws ecs list-clusters --query 'clusterArns' --output text)
for ClustersArn in $ClustersArns; do
 ServiceArns=$(aws ecs list-services --cluster $ClustersArn --query 'serviceArns' --
output text)
 for ServiceArn in $ServiceArns; do
```

```
  aws ecs update-service --cluster $ClustersArn --service $ServiceArn --propagate-tags
 SERVICE &>/dev/null
  aws ecs tag-resource --resource-arn $ServiceArn --tags key=$TAG_KEY,value=$TAG_VALUE
 done
done
```

Enable the cost allocation tag to work with AWS billing tools

We recommend activating the user-defined cost allocation tag. This enables the **Rightsizing** tag to be recognized and filterable in the AWS billing tools (for example, AWS Cost Explorer and AWS Cost and Usage Report). If you don't enable this, the tag filtering option and data won't be available. For information about using cost allocation tags, see Activating user-defined cost allocation tags in the AWS Billing and Cost Management documentation.

After waiting for 24 hours, you can see the tag in Cost Explorer before implementing right sizing recommendations in the next section. To do this, search for the **Rightsizing** tag in Cost Explorer.

Implement right sizing recommendations

Compute Optimizer will provide either task or container size recommendations. To implement right sizing recommendations, do the following.

1. Open the Amazon ECS console.
2. From the navigation bar, choose the Region that contains your task definition.
3. In the navigation pane, choose **Task definitions**.
4. On the **Task definitions** page, choose the task, and then choose **Create new revision**.
5. On the **Create new task definition revision** page, make changes. To update the container size recommendation, update `cpu` and `memory` under the **containerDefinitions** block in your ECS task definition. For example:

```
"containerDefinitions": [
 {
  "name": "your-container-name",
  "image": "your-image",
  "cpu": 1024,
  "memory": 2048,
 }
],
```

6. Verify the information, and then choose **Create**.

To update the Amazon ECS service, do the following:

1. Open the Amazon ECS console.
2. On the **Clusters** page, select the cluster.
3. On the **Cluster overview** page, select the service, and then choose **Update**.
4. For **Task definition**, choose the task definition family and revision to use.

For advanced operators, you could use CloudShell to update the Amazon ECS service. For example:

```
bash
#!/bin/bash
# Set variables
ClustersName="workshop-cluster"
ServiceName="lab7-fargate-service"
TaskDefinition="lab7-fargate-demo:3"
# update the service
aws ecs update-service --cluster $ClustersName --service $ServiceName --task-definition
 $TaskDefinition
```

Review before and after costs

After you have right sized your resources, you can use Cost Explorer to show before and after costs by using the **Rightsizing** tag. Recall that you can use resource tags to track costs. By using several layers of tags, you can achieve granular visibility into your costs. In the example covered in this guide, the **Rightsizing** tag is used to apply a generic tag to all targeted instances. Then, a **team** tag is used to further organize resources. The next step is to introduce application tags to further show the cost impact for operating a specific application.

Consider an example of the cost reduction that can be achieved by using the **Rightsizing** tag for a single account level. In this example, operating costs go from $30.26 per day to $7.56 per day. Assuming 744 hours per month, the annual cost before right sizing is $11,044.9. After right sizing, the annual cost drops to $2,759.4. This translates to a 75 percent decrease in compute costs for this account. Imagine the impact of this across a large organization.

Before embarking on your right sizing journey, consider the following:

- AWS offers many options for cost reduction. This includes AWS OLA, where AWS reviews your on-premises instances prior to moving to AWS. The AWS OLA also provides you with right-sizing recommendations and licensing guidance.

- Complete all of your right sizing before purchasing Savings Plans. This can help you avoid over purchases on your Savings Plans commitment.

Next steps

We recommend the following next steps:

1. Review your existing landscape and consider converting Amazon EBS gp2 volumes to gp3 volumes.
2. Review Savings Plans.

Additional resources

- Getting started with Compute Optimizer (AWS documentation)
- Best Practices for Tagging AWS Resources (AWS Whitepapers)
- Windows Containers on AWS (AWS Workshop Studio)

Gain visibility into your Amazon EKS costs

Overview

A holistic view is necessary for effectively monitoring the cost of a Kubernetes deployment. The only fixed and known cost is for the Amazon Elastic Kubernetes Service (Amazon EKS) control plane. This includes every other component that makes up the deployment, from compute and storage to networking, being a variable amount based on your application needs.

You can use Kubecost to analyze the cost of your Kubernetes infrastructure all the way from the Namespaces and Services down to the individual Pods, and then display the data in a dashboard. Kubecost surfaces in-cluster costs like compute and storage and out-of-cluster costs like Amazon Simple Storage Service (Amazon S3) buckets and Amazon Relational Database Service (Amazon RDS) instances. Kubecost will make right-sizing recommendations based on this data and display critical alerts that may impact the system. Kubecost can integrate with AWS Cost and Usage Report to show savings from Compute Savings Plans, Reserved Instances, and other discount programs.

Cost benefits

Kubecost provides reports and dashboards that visualize the cost of your Amazon EKS deployments. It enables you to drill down from the cluster into each of the various components such as the controllers, services, nodes, pods, and volumes. This gives you a holistic view of your applications running in an Amazon EKS environment. By enabling this visibility, you can act on the Kubecost recommendations or view the costs of each application at a granular level. Right sizing an Amazon EKS node group offers the same potential savings as standard EC2 instances. If you can right size your containers and nodes, then you can remove compute bloat from the size of the instance needed to run the container and the number of EC2 instances required in the auto scaling group.

Cost optimization recommendations

To take advantage of Kubecost, we recommend that you do the following:

1. Deploy Kubecost into your environment
2. Get a granular cost breakdown of Windows applications
3. Right size cluster nodes
4. Right size container requests
5. Manage underutilized nodes
6. Remedy abandoned workloads
7. Act on recommendations
8. Update self-managed nodes

Deploy Kubecost into your environment

The Amazon EKS Finhack Workshop teaches you how to deploy an Amazon EKS environment that's configured to use Kubecost in an AWS owned account. This allows you to get hands-on experience with the technology. If you're interested in running this workshop in your organization, contact your account team.

To deploy Kubecost to your Amazon EKS cluster using Helm, see the AWS and Kubecost collaborate to deliver cost monitoring for EKS customers post on the AWS Blog. Alternatively, you can refer to the official Kubecost documentation for instructions on installing and configuring Kubecost. For information about Kubecost support for Windows nodes, see Windows Node Support in the Kubecost documentation.

Get a granular cost breakdown of Windows applications

Although you can achieve significant cost savings by using Amazon EC2 Spot Instances, you can also benefit from the fact that Windows workloads tend to be stateful. The use of Spot Instances is application-dependent, and we encourage you to verify if they will be applicable for your use case.

To get a granular cost breakdown of your Windows applications, log in to Kubecost. In the navigation page, choose **Savings**.

Right size cluster nodes

In Kubecost, choose **Savings** from the navigation bar, and then choose **Right-size your cluster node**.

Consider an example where Kubecost reports that the cluster is over-provisioned both in terms of vCPU and RAM. The following table shows the details and recommendations from Kubecost.

	Current	Recommendation: Simple	Recommendation: Complex
Total count	US $3462.57 per month	US $137.24 per month	US $303.68 per month
Node count	4	5	4
CPU	74 VCPUs	10 VCPUs	8 VCPUs
RAM	152 GB	20 GB	18 GB
Instance breakdown	2 c5.xlarge + 2 more	5 t3a.medium	2 c5n.large + 1 more

As described in the Kubecost blog post Find an optimal set of nodes for a Kubernetes cluster, the simple option utilizes a single node group, whereas the complex one utilizes a multi-node group approach. The **Learn how to adopt** button can perform one-click cluster resizing. It requires the installation of the Kubecost Cluster Controller.

If you're using self-managed Windows nodes that aren't created by eksctl, see Updating an existing self-managed node group. These instructions show you how to change the instance type in the Amazon EC2 launch template used by the Auto Scaling group.

Right size container requests

In Kubecost, choose **Savings** from the navigation bar, and the go to the **Request right-sizing recommendations** page. This page shows the efficiency of the pods, right-sizing recommendations, and estimated cost savings. You can use the **Customize** button to filter by **Cluster**, **Node**, **Namespace\Controller**, and more.

As an example, consider that Kubecost has calculated that some of your pods are overprovisioned in terms of CPU and RAM (memory). Then, Kubecost recommends that you adjust to new CPU and RAM values to achieve its estimated monthly savings. To change the CPU and RAM values, you must update your deployment manifest file.

Manage underutilized nodes

In Kubecost, choose **Savings** from the navigation bar, and then choose **Manage underutilized nodes**.

Consider an example where the page shows that one node in the cluster is underutilized in terms of CPU and RAM (memory) and can therefore be drained and either terminated or resized. Choosing the nodes that don't pass the node and pod checks will give you more information about why they cannot be drained.

Remedy abandoned workloads

In Kubecost, choose **Savings** from the navigation bar, and then choose the **Abandoned Workloads** page. In this example, you filter by Namespace called **windows**. This page shows the pods that have not met the traffic threshold and are considered abandoned. Pods need to send or receive a certain amount of network traffic over the defined period.

After careful consideration that one or more pods are abandoned, you can save on costs by scaling down the number of replicas, deleting the deployment, resizing it to consume fewer resources, or notifying the application owner that you believe the deployment is abandoned.

Act on recommendations

In the **Right-size your cluster nodes** section, Kubecost analyzes the usage of the worker nodes in the cluster, and makes recommendations about right sizing the nodes to reduce cost. There are two types of node groups that can be used with Amazon EKS: self-managed and managed.

Update self-managed nodes

For information about updating self-managed nodes, see Self-managed node updates in the Amazon EKS documentation. It states that node groups created with `eksctl` can't be updated and must be migrated to a new node group with the new configuration.

As an example, assume that you have a Windows node group called `ng-windows-m5-2xlarge` (which uses an m5.2xlarge EC2 instance) and you want to migrate the pods to a new node group called `ng-windows-t3-large` (which is backed by a t3.large EC2 instance to save cost).

To migrate to a new node group when you use node groups deployed by `eksctl`, do the following:

1. To find the node that the pod is currently, run the `kubectl describe pod <pod_name> -n <namespace>` command.
2. Run the `kubectl describe node <node_name>` command. The output shows that the node is running on a m5.2xlarge instance. It also matches the node group name (`ng-windows-m5-2xlarge`).
3. To change the deployment to use node group `ng-windows-t3-large`, delete node group `ng-windows-m5-2xlarge` and run `kubectl describe svc,deploy,pod -n windows`. The deployment immediately starts to redeploy now that its node group has been deleted.

> **Note**
>
> There will be downtime of the service when you delete the node group.

4. Run the `kubectl describe svc,deploy,pod -n windows` command again after a few minutes. The output shows that the pods are all in a **Running** state again.
5. To show that the pods are now running on node group `ng-windows-t3-large`, run the `kubectl describe pod <pod_name> -n <namespace>` and `kubectl describe node <node_name>` commands again.

Alternative resizing methods

This method applies to any combination of self-managed or managed node groups. The Seamlessly migrate workloads from EKS self-managed node group to EKS-managed node groups blog post provides guidance on how to migrate your workloads from one node group with the oversized instance type to the node group that has been right sized without any downtime.

Next steps

Kubecost makes it easy to visualize the cost of your Amazon EKS environments. The deep integration of Kubecost with Kubernetes and the AWS APIs can help you find potential cost savings. You can see these as recommendations in the **Savings** dashboard of Kubecost. Kubecost can also implement some of these recommendations for you through its cluster controller feature.

We recommend that you review the step-by-step deployment in the AWS and Kubecost collaborate to deliver cost monitoring for EKS customers blog post from the AWS Containers blog.

Additional resources

- Amazon EKS Workshop (Amazon EKS Workshop)
- AWS and Kubecost collaborate to deliver cost monitoring for EKS customers (AWS Blog)
- Amazon EKS Finhack Workshop (AWS Workshop Studio)
- Windows Containers on AWS (AWS Workshop Studio)

Replatform Windows applications with App2Container

Overview

AWS App2Container is a command-line tool for migrating and modernizing Java and .NET web applications into containers. App2Container analyzes and builds an inventory of all applications running in bare metal, virtual machines, Amazon Elastic Compute Cloud (Amazon EC2) instances, or in other cloud providers. You select the application you want to containerize. App2Container packages the application artifacts and dependencies into container images, configures the network ports, and generates the necessary Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS) deployment artifacts, which are infrastructure as code (IaC) templates. App2Container provisions the cloud infrastructure and CI\CD pipelines required to deploy the containerized application into a production environment. For more information, see How App2Container works in the App2Container documentation.

With App2Container, you can migrate to AWS and modernize your applications as containers while also standardizing the deployment and operations for your applications. You can use App2Container to help quickly build a proof of concept (PoC) or accelerate deployment of production workloads in containers.

There are several things to keep in mind when working with Windows applications. App2Container supports containerization of ASP.NET applications deployed on Microsoft Internet Information Services (IIS), including IIS-hosted Windows Communication Foundation (WCF) applications that run on Windows Server 2016, Windows Server 2019, or Windows Server Core 2004. For more information, see Supported applications for Windows in the App2Container documentation. App2Container uses Windows Server Core as a base image for its container artifacts, matching the Windows Server Core container version to the operating system (OS) version of the server where you run containerization commands. This approach decouples the application from the underlying OS so that you can upgrade the OS without performing a traditional migration.

If you use a worker machine to containerize your application, the container base image, such as Windows Server 2019 long-term servicing channel (LTSC), matches your worker machine OS, such as Windows Server 2019. If you are running containerization directly on application servers, the version matches your application server OS. If your applications are running on Windows Server 2008 or 2012 R2, you can still use App2Container by setting up a worker machine for containerization and deployment steps. App2Container does not support applications running on Windows client operating systems, such as Windows 7 or Windows 10. App2Container supports Tomcat, TomEE, and JBoss (standalone mode) frameworks for Java processes. For more information, see App2Container compatibility.

Cost benefits

Containerizing and consolidating your applications can yield up to 60% compute savings when compared to a one-application-to-one-server deployment design pattern. App2Container helps expedite the application containerization process. The following are some of the benefits of using App2Container for your modernization needs:

- App2Container is offered at no additional charge.
- App2Container supports multiple applications in a container image.
- Address operating systems that are approaching end of support by using App2Container to move your legacy .NET applications to containers. You can move to a newer operating system, avoid paying for extended support, and reduce security risks.
- Containers are an efficient and cost-effective method of packaging your .NET applications. Review the benefits of containers in the MACO Recommendation - Moving to containers.
- Application consolidation and containerization help reduce your compute, storage, and licensing footprint by using your compute resources more efficiently.

- Moving to containers can reduce operational overhead and infrastructure costs and increase development portability and deployment agility.

Cost optimization recommendations

For instructions about how to use App2Container, see Getting started with AWS App2Container. For information about the App2Container commands, see the App2Container command reference.

Next steps

App2Container can accelerate the process of containerizing applications and deploying to Amazon EKS or Amazon ECS. The deployment of applications to containers reduces compute, networking, and storage costs and reduces the operational overhead for application operators.

For a hands-on experience with App2Container, see the Modernize with AWS App2Container Workshop. If you would like to have a deep-dive learning experience, ask your AWS account team to set up an App2Container immersion day.

Additional resources

- Containerizing Complex Multi-tier Windows Applications using AWS App2Container (AWS blog post)
- Containerizing legacy ASP.NET applications using AWS App2Container (AWS blog post)
- App2Container supported applications (AWS documentation)
- Modernize with AWS App2Container Workshop (AWS Workshop Studio)
- AWS App2Container FAQs (AWS website)

Storage

Choosing the right storage for your Microsoft workloads is a critical architectural decision. As part of the decision-making process, we recommend that you develop a storage plan and determine functional requirements for your applications and services. This chapter provides an overview of the following storage options that could factor into your planning.

Sections:

- Amazon EBS
- Amazon FSx
- AWS Storage Gateway

Amazon EBS

Amazon Elastic Block Store (Amazon EBS) is a fully managed block storage service that enables you to store persistent block-level storage volumes that you can use with Amazon Elastic Compute Cloud (Amazon EC2) instances. You can take advantage of several features in Amazon EBS to effectively manage and optimize your storage resources for Windows workloads in the cloud. For example, you can use Amazon EBS to provision the exact amount of IOPS and throughput that you require for your workload, select from a range of volume types to match your workload requirements, and use tools to identify and eliminate wasted storage resources. This granular control over storage performance and usage helps you optimize your storage resources while avoiding unnecessary costs.

This section covers the following topics:

- Migrate Amazon EBS volumes from gp2 to gp3
- Modify Amazon EBS snapshots
- Delete unattached Amazon EBS volumes

Migrate Amazon EBS volumes from gp2 to gp3

Overview

A solid-state drive (SSD) is the standard storage option for production and high-performance workloads. Amazon EBS offers a general purpose SSD volume for mid- to high-performance

workloads. The standard across many AWS services (including Amazon EC2) is gp2, the second generation of these general purpose SSD volumes. The third generation of general purpose SSDs, called gp3, was released in December 2020.

The gp3 offering made significant improvements to the performance customization aspects over the previous generation. For Amazon EBS gp2 volumes, the performance is closely coupled with the size of the volume. For every 1 GB of capacity, gp2 volumes get 3 IOPS of performance. That is, a 2,000 GB gp2 volume is capable of 6,000 IOPS. For gp3 volumes, performance can be customized independently from the storage capacity. This enables even small capacity volumes to achieve performance capabilities up to 80,000 IOPS and 2,000 Mb/s throughput.

Another major change with gp3 volumes is the baseline IOPS performance. The gp3 volumes start at 3,000 IOPS. By comparison, gp2 volumes must reach 1 TiB in size before reaching the same performance capability. For Windows Server, which usually has a C: drive much smaller than 1 TiB, upgrading from gp2 to gp3 is a significant performance improvement.

Finally, the price of gp3 volumes is one of the largest improvements compared to gp2 volumes. The gp3 volumes offer enhanced performance capabilities at 20 percent lower cost than gp2 volumes.

Cost impact

With the ability to scale performance independently from capacity, it's important to understand the pricing aspects of adding additional IOPS and throughput. For gp2 volumes, pricing is based on provisioned capacity at $0.10 per GiB-month. For gp3 volumes, pricing is similar to the high performance provisioned IOPS SSD volumes, which have one cost for capacity and a separate cost for additional IOPS and throughput.

As noted in the following table, gp3 volumes have a capacity price at $0.08 per GiB-month (20 percent less expensive than gp2) and separate costs for IOPS at $0.005 per provisioned IOPS-month over 3,000 and $0.04 per provisioned MiBs-month over 125 MiBs for throughput.

	gp3	gp2
Volume size	1 GiB – 64 TiB	1 GiB – 16 TiB
Baseline IOPS	3,000	3 IOPS/GiB (minimum 100 IOPS) to a maximum of 16,000 IOPS

	gp3	gp2
		Volumes smaller than 1 TiB can burst up to 3,000 IOPS
Max IOPS/volume	80,000	16,000
Baseline throughput	125 MiBs	Throughput limit is between 128 MiBs – 250 MiBs, depending on volume size
Max throughput/volume	2,000 MiBs	250 MiBs
Price	$0.08/GiB-month 3,000 IOPS free and $0.005/provisioned IOPS-month over 3,000 125 MiBs free and $0.04/provisioned MiBs-month over 125 MiBs	$0.10/GiB-month

Important

Even though gp3 volumes have separate costs for capacity and performance, gp3 volumes are always cheaper than gp2 volumes if they're configured at the same performance levels.

The following tables show examples of cost savings that can be achieved by converting gp2 to gp3 volumes at various capacity and performance configurations.

Example for gp2 configuration

Volume size (GiB)	Max IOPS	Throughput (MiBs)	Cost (USD/month)
30	3000	128	$3.00
100	3000	128	$10.00

Volume size (GiB)	Max IOPS	Throughput (MiBs)	Cost (USD/month)
500	3000	250	$50.00
1000	3000	250	$100.00
2000	6000	250	$200.00
6000	16000	250	$600.00

Example for gp3 (baseline) configuration

Max IOPS	Throughput (MiBs)	Cost (USD/month)	Cost reduction (compared to gp2)
3000	125	$2.40	20%
3000	125	$8.00	20%
3000	125	$40.00	20%
3000	125	$80.00	20%
3000	125	$160.00	20%
3000	125	$480.00	20%

Example for gp3 (gp2 matching) configuration

Max IOPS	Throughput (MiBs)	Cost (USD/month)	Cost reduction (compared to gp2)
3000	128	$2.52	16%
3000	128	$8.12	19%
3000	250	$45.00	10%

Max IOPS	Throughput (MiBs)	Cost (USD/month)	Cost reduction (compared to gp2)
3000	250	$85.00	15%
6000	250	$180.00	10%
16000	250	$550.00	8%

For a cost analysis, see the *EBS gp2 to gp3 migration cost savings calculator* section in Amazon EBS resource. You can download the calculator and use it to find out how much you can save by migrating your gp2 volumes to gp3.

Cost optimization recommendations

For instructions on how to complete the migration process, see the Migrate your Amazon EBS volumes from gp2 to gp3 and save up to 20% on costs post on the AWS Storage Blog.

Additional resources

- Migrate your Amazon EBS volumes from gp2 to gp3 and save up to 20% on costs (AWS Storage Blog)
- Build an AWS Config Custom Rule to Optimize Amazon EBS Volume Types (AWS Cloud Operations & Migrations Blog)
- Controlling your AWS costs by deleting unused Amazon EBS volumes (AWS Cloud Operations & Migrations Blog)
- Amazon EBS Migration Utility (GitHub)
- Finding savings from 2020 re:Invent announcements (AWS Cloud Financial Management)
- Cost Optimization Workshop (AWS Well-Architected Labs)
- gp2 to gp3 migration cost savings calculator (download)

Modify Amazon EBS snapshots

Overview

Deleting EBS volumes and managing the retention and archiving of snapshots is an important aspect to control costs from the start. You can back up the data on your EBS volumes to Amazon

Simple Storage Service (Amazon S3) by taking point-in-time snapshots. Snapshots are incremental backups, so they save only the blocks on the devices that changed after your most recent snapshot. This minimizes the time required to create the snapshot and saves on storage costs by not duplicating data. Each snapshot contains all the information that's required to restore your data (from when the snapshot was created) to a new EBS volume.

Charges for EBS snapshots are calculated by the gigabyte-month. You're billed for how large the snapshot is and how long you keep the snapshot. Pricing varies depending on the storage tier. For the Standard tier, you're billed only for changed blocks that are stored. For the Archive tier, you're billed for all snapshot blocks that are stored. You're also billed for retrieving snapshots from the Archive tier. The following are example scenarios for each storage tier:

- **Standard tier** – You have a volume that's storing 100 GB of data. You're billed for the full 100 GB of data for the first snapshot (snap A). At the time of the next snapshot (snap B), you have 105 GB of data. You're then billed for only the additional 5 GB of storage for incremental snap B.
- **Archive tier** – You archive snap B. The snapshot is then moved to the Archive tier, and you're billed for the full 105 GB snapshot block.

You can use Amazon Data Lifecycle Manager to help you get set up a lifecycle to retain and manage your snapshots on schedule.

Cost impact

Charges for EBS volumes and snapshots are managed separately. EBS snapshots are billed at a lower rate than active EBS volumes. When an instance terminates, the value of the DeleteOnTermination attribute for each attached EBS volume determines whether to preserve or delete the volume. By default, the `DeleteOnTermination` attribute is set to `True` for the root volume. It's set to `False` for all other volume types. This creates situations where the operator intends to delete an EC2 instance, but leaves behind volumes that were added to the instance in addition to the root volume. For instructions about checking volumes (and their associated snapshots) that you no longer require, see View information about an Amazon EBS volume in the Amazon EBS documentation.

By default, when you create a snapshot, it's stored in the Amazon EBS Snapshot Standard tier (standard tier). Snapshots stored in the standard tier are incremental. This means that only the blocks on the volume that have changed after your most recent snapshot are saved. The Amazon EBS Snapshots Archive is a new storage tier that you can use for low-cost, long-term storage of your rarely-accessed snapshots that don't require frequent or fast retrieval. The pricing difference

from standard to archival is significant and should be a key consideration when setting up your snapshot strategy. Amazon EBS Snapshots Archive offers up to 75 percent lower snapshot storage costs for snapshots that you plan to store for 90 days or longer and that you rarely need to access.

Amazon EBS Snapshot Storage	Cost
Standard	$0.05/GB-month
Archive	$0.0125/GB-month

In smaller environments, the cost savings may not be significant. The savings are more significant at a large scale where there are multiple accounts and thousands of EC2 instances with TBs of EBS snapshots sitting even when the EBS volumes have been deleted.

The following table compares the standard and archive tiers per month at just 50 TB of usage. Even at this lower scale it's still thousands of dollars of savings annually.

Amazon EBS Snapshot Storage	Cost per month	Cost per year
Standard 50 TB	$312.50	$3,750
Archive 50 TB	$78.13	$937.60
	Annual savings	**$2,812.40**

Cost optimization recommendations

Deleting a snapshot might not reduce your organization's data storage costs. Other snapshots might reference that snapshot's data, and referenced data is always preserved. For example, when you take the first snapshot of a volume with 10 GiB of data, the size of the snapshot is also 10 GiB. Because snapshots are incremental, the second snapshot that you take of the same volume contains only blocks of data that changed since the first snapshot was taken. The second snapshot also references the data in the first snapshot. If you change 4 GiB of data and take a second snapshot, the size of the second snapshot is 4 GiB. In addition, the second snapshot references the unchanged 6 GiB in the first snapshot. For more information, see Why did my storage costs not

reduce after I deleted a snapshot of my EBS volume and then deleted the volume itself? in the AWS Knowledge Center.

Consider the following:

- You aren't billed for snapshots that another AWS account owns and shares with your account. You're billed only when you copy the shared snapshot to your account. You're also billed for EBS volumes that you create from the shared snapshot.
- If a snapshot (snap A) is referenced by another snapshot (snap B), then deleting snap B might not reduce the storage costs. When you delete a snapshot, only the data that's unique to that snapshot is removed. Data that's referenced by other snapshots remain, and you're billed for this referenced data. To delete an incremental snapshot, see Incremental snapshot deletion in the Amazon EBS documentation.

Snapshot cleanliness is standard operating practice when running your workloads in AWS. Over time, snapshots can add up to costly charges for data that you don't need.

Additional resources

- Controlling your AWS costs by deleting unused Amazon EBS volumes (AWS Cloud Operations & Migrations Blog)
- Delete an Amazon EBS snapshot (Amazon EBS documentation)
- Cost Optimization Workshop (AWS Well-Architected Labs)
- Automatically archive Amazon EBS Snapshots with Amazon Data Lifecycle Manager (AWS Storage Blog)

Delete unattached Amazon EBS volumes

Overview

Unattached (orphaned) EBS volumes can lead to unnecessary storage costs in your AWS environment. It's essential to incorporate the regular review and deletion of unused and unutilized EBS volumes as part of your AWS environment hygiene. It's a best practice to have a process in place to continually review the usage of EBS volumes. You can use the AWS Compute Optimizer to review underutilized instances. This section helps you identify, manage, and delete EBS volumes that are unattached or underutilized.

Amazon EBS

Amazon Elastic Block Store (Amazon EBS) is a block-level device that offers storage volumes for Amazon Elastic Compute Cloud (Amazon EC2) instances. EBS provides persistent storage, with the flexibility to attach and detach from EC2 instances. This means the lifecycle of EBS volumes persists even if an EC2 instance is terminated. The DeleteOnTermination attribute is a feature that controls whether to preserve or delete attached EBS volumes upon instance termination. By default, the attribute is set to `True` for the root volume, resulting in deletion. It's set to `False` for other volumes, resulting in preservation.

Cost impact

Unattached EBS volumes, also referred to as unused or orphaned volumes, incur the same charges as attached volumes based on the provisioned storage size and storage type. Although the average cost of Amazon EBS charges may seem minimal at $0.10 per GB-month, it's crucial to recognize that the accumulation of unused EBS volumes can result in significant costs over time.

For example, consider the ramifications of retaining 50 unused EBS volumes, each provisioned with a storage size of 100 GB, as the following table shows.

Number of storage volumes	Volume type	Size	Total monthly cost
50 volumes	gp2 ($0.10 USD)	100 GB	100 GB 50.00 EBS volumes months $0.10 USD = $500.00 USD

The scenario from the preceding table yields a cost reduction of approximately $500 per month or $6,000 annually. This is an effective step toward cost reduction. Be sure to incorporate the deletion of unattached EBS volumes as a regular practice in your AWS environment hygiene.

Cost optimization recommendations

You can use AWS to easily automate the deletion of unattached EBS volumes. For example, you can use AWS Lambda, AWS Config, Amazon CloudWatch, and AWS Systems Manager to define criteria for deleting unattached volumes based on age, tags, and other specifications. You can also use these AWS services to automate the cleanup process at scale.

To avoid unintended consequences, we recommend that you perform your due diligence before deleting unattached EBS volumes.

Manage unattached EBS volumes

We recommend that you consider the follow best practices:

- **Meet compliance requirements** – Verify that the deletion of unattached EBS volumes complies with your organization's governance and compliance requirements.
- **Set data backup and retention policies** – Before deleting an unattached EBS volume, back up any important data to another storage repository (for example, Amazon S3). For data retention, Amazon EBS snapshots are a more cost-effective way to retain data than EBS volumes, and they can restore the volume if needed in the future. For more information about effectively managing snapshots, see the Modify Amazon EBS snapshots section of this guide.
- **Check for dependencies** – Check for any dependencies between unattached EBS volumes and other AWS resources. You can use the AWS Management Console or an API to gather descriptive information about your EBS volumes, such as size, status, and associated resources. This is an important step to safeguard against deleting any temporarily unattached resources.
- **Create a retention policy** – Establish a retention period for unattached EBS volumes. This can help you identify the appropriate time to delete unattached volumes, ensuring that your AWS environment remains optimized. For example, you can create an Amazon EventBridge rule to initiate a Lambda function on a scheduled basis. The Lambda function can use the AWS SDK to actively identify any unattached EBS volumes, apply a tagging mechanism for easy tracking, and send out notifications when an unattached EBS volume reaches or exceeds a defined threshold.
- **Tag unattached EBS volumes** – Tagging EBS volumes is a useful practice that can aid in organizing and identifying volumes based on attributes such as environment, application, or owner. This can be particularly helpful when deciding which unattached volumes to delete, because it enables you to quickly identify volumes that are no longer needed based on their tags.
- **Ensure safe deletion** – Reviewing when an EBS volume was last attached can help you determine whether it's safe to delete the volume. For more information, see How do I use AWS CLI commands to list the attachments or detachments history of a specific Amazon EBS volume? in the AWS Knowledge Center.
- **Identify underutilized EBS volumes** – Identifying and removing underutilized EBS volumes is a highly recommended practice for reducing storage costs and maintaining an optimized AWS environment. AWS Trusted Advisor and AWS Compute Optimizer can help you identify underutilized EBS volumes and provide recommendations to reduce costs and improve efficiency.

For example, see Setting up automation for optimizing EBS volumes with AWS Trusted Advisor (GitHub), Establishing a Trusted Advisor Organization (TAO) dashboard (AWS Workshop Studio), and Cost-optimizing Amazon EBS volumes using AWS Compute Optimizer (AWS Storage Blog).

Automate the cleaning of unattached EBS volumes

We recommend that you consider the following tools to help you automate the cleaning of unattached EBS volumes:

- AWS APIs (DescribeVolumes) – You can filter and find unattached EBS volumes by using AWS SDKs or the AWS Command Line Interface (AWS CLI). You can save time and effort by automating this process with a script or a Lambda function that runs on a schedule. A sample script from GitHub demonstrates how this works. The script uses Lambda to analyze AWS CloudTrail logs and identify unattached EBS volumes.
- AWS Systems Manager Automation – This enables you to automate routine maintenance and remediation tasks in your infrastructure. To get started, create an automation runbook, which defines a series of steps to be executed in a specific order. For example, you can create a runbook that first creates a snapshot of the unattached EBS volume and then deletes the volume itself. This can help you automate tasks that would otherwise be time-consuming and error-prone if done manually.
- AWS Config – This enables you to assess, audit, and track changes to your AWS resources over time. By capturing configuration changes, you can use AWS Config to evaluate compliance, governance, and resource utilization in your environment. For example, AWS Config can identify unused EBS volumes. Furthermore, you can associate AWS Systems Manager Automation with AWS Config to automatically remediate the deletion of unused EBS volumes.

Additional resources

- Delete unused Amazon Elastic Block Store (Amazon EBS) volumes by using AWS Config and AWS Systems Manager (AWS Prescriptive Guidance)
- Controlling your AWS costs by deleting unused Amazon EBS volumes (AWS Cloud Operations & Migrations Blog)
- AWSConfigRemediation-DeleteUnusedEBSVolume (AWS Systems Manager Automation runbook reference)

Amazon FSx

Amazon FSx for Windows File Server is a fully managed file storage service that's optimized for Windows workloads. It provides you with a simple and scalable solution to run your Windows-based applications and workloads, without the need for complex storage infrastructure management. You can use FSx for Windows File Server to easily provision and access shared file storage that supports your Windows applications natively, including Microsoft SQL Server, Microsoft SharePoint, and custom .NET applications. Additionally, FSx for Windows File Server helps you manage costs by providing flexible pricing options, such as pay-as-you-go and storage quotas, and automatic data deduplication to reduce storage footprint and optimize performance and cost.

This section covers the following topics:

- Choose the right SMB file storage
- Enable data deduplication in Amazon FSx
- Understand data sharding in FSx for Windows File Server
- Understand HDD volume usage in Amazon FSx
- Use a single Availability Zone

Choose the right SMB file storage

Overview

AWS offers a variety of fully managed storage services that give you the rich capabilities of industry-leading file services, while combining the latest AWS infrastructure innovations and security. You can incorporate AWS services into infrastructure as code (IaC) workflows and integrate them with AWS compute, monitoring, and data protection services. For Windows workloads, you can choose from two fully managed file services that can be used to match your application needs: FSx for Windows File Server and Amazon FSx for NetApp ONTAP.

FSx for Windows File Server

Amazon FSx for Windows File Server provides fully managed shared storage built on Windows Server, and delivers a wide range of data access, data management, and administrative capabilities. FSx for Windows File Server integrates easily with Windows environments because it's a Windows-native service. We recommend using FSx for Windows File Server for user and group shares,

Always On Failover Cluster Instances for SQL Server, Windows applications, and virtual desktop infrastructure (VDI). FSx for Windows File Server also integrates well with Amazon FSx File Gateway, Amazon Kendra, audit logs for Amazon S3, and Amazon Data Firehose.

FSx for ONTAP

FSx for ONTAP is based on NetApp's proprietary ONTAP file system. It takes some level of upskilling and is recommended mostly to existing on-premises NetApp users. Typical use cases include user and group shares, Always On Failover Cluster Instances for SQL Server, and Windows applications. FSx for ONTAP supports multiple protocols, larger than 64 TB file systems (PB scale without a DFS namespace server), cloning, replication, snapshots, compression (storage efficiency), and intelligent tiering of data.

Cost impact

FSx for Windows File Server

FSx for Windows File Server was the first shared storage solution on AWS for deploying Failover Cluster Instances for SQL Server. With FSx for Windows File Server, you could launch Failover Cluster Instances using SQL Standard edition licensing. This, however, prevents you from relying on Always On availability groups, which require SQL Server Enterprise edition licenses. By switching from SQL Server Enterprise Standard edition to SQL Server Standard edition, you could save 65–75 percent on your SQL Server licensing.

You can use FSx for Windows File Server for Failover Cluster Instances to offload storage I/O from typical EBS storage. By offloading I/O to FSx for Windows File Server, you could scale down EC2 instances, which rely on high Amazon EBS throughput and IOPS, without affecting storage throughput.

FSx for ONTAP

You can use FSx for ONTAP to run your Microsoft failover cluster on block protocol iSCSI and benefit from SQL Server instant file initialization, cross-Region replication using SnapMirror, antivirus support, and cloning. If you create multiple copies of databases for testing, cloning can make a significant difference in both space consumption and how quick those database copies can be created. Additionally, you can use NetApp SnapCenter to manage backup, restore, and clone functionality with your EC2 instances for SQL Server by using FSx for ONTAP. FSx for ONTAP also provides automatic tiering from SSD to a low cost capacity pool storage for a mixture of performance and cost efficiency.

FSx for ONTAP supports NetApp's file system (ONTAP), unlike FSx for Windows File Server, which supports a Windows native NTFS file system. The minimum size for FSx for ONTAP is 1024 GB, while FSx for Windows File Server can start as low as 32 GB.

Integration with Microsoft Distributed File System

FSx for Windows File Server and FSx for ONTAP integrate with Microsoft's Distributed File System (DFS) for seamless integration into existing deployments. Keep the following in mind when planning your architecture:

- FSx for Windows File Server and FSx for ONTAP support DFS Namespaces (DFSN) on both deployment types (multiple Availability Zones and single Availability Zones).
- Only FSx for Windows File Server supports DFS Replication (DFSR), and only when using single Availability Zones.

Cost optimization recommendations

Performance for both FSx for Windows File Server and FSx for ONTAP is very configuration dependent, as is their pricing. FSx for Windows File Server pricing primarily depends on storage capacity and storage type, throughput capacity, backup, and data transferred. With FSx for ONTAP, you pay for SSD storage, SSD IOPS, capacity pool usage, throughput capacity, and backup.

File service	Cost for 5 TB storage	Configuration	Region
FSx for Windows File Server	$982.78	Single Availability Zone SSD (15,000 IOPS) 32 MBps 5 TB backup (no deduplication savings)	US East (N. Virginia)
FSx for ONTAP	$979.28	Single Availability Zone 100% SSD	US East (N. Virginia)

File service	Cost for 5 TB storage	Configuration	Region
		15,000 read-write capacity tier 15,000 SSD IOPS 128 MBps 5 TB backup (no deduplication savings)	

Keep in mind the following:

- Deduplication and compression enable you to store more data on physical devices by shrinking data size, but you pay for the provisioned solid state drive (SSD) or hard disk drive (HDD) storage.
- You can use FSx for ONTAP to tier your data. It's extremely rare for 100 percent of your data to be accessed regularly and require SSD storage. You can move cold and infrequently accessed data to a capacity tier for cost savings.
- The prices mentioned here are calculated with 100 percent data on the SSD tier and 15,000 IOPS on the SSD tier.

Backup

By default, both FSx for ONTAP and FSx for Windows File Server store their fully managed backup on Amazon S3. However, with FSx for ONTAP there is an additional option for backup using SnapVault, which can configure backups to reside in the capacity tier. Backing up with SnapVault is a self-managed mechanism that's more cost-efficient than the default fully-managed backup option. The fully-managed backup option is $0.05 per GB-month. The SnapVault backup on FSx for ONTAP (10:1 SSD to capacity pool storage) is $0.03221 (0.9x0.0219+0.1x0.125).

Keep in mind the following:

- AWS managed backups offer granularity of one hour. SnapVault enables you to go as low as five minutes.
- You can use NetApp's tools (such as the CLI and API) to configure the SnapVault relationships and snapshot replication.

- Enable the `all` tiering policy on a SnapVault volume to use the capacity tier as storage for the backup data.
- SnapVault destinations can be in the same AWS Region, cross-Region, or on-premises. This is usually to a single Availability Zone or multiple Availability Zone file system backup destination. In comparison, AWS Backup is backed by the regional resiliency of Amazon S3.

Right sizing

You can also save on costs and get the most out of your file system by right sizing and preventing over provisioning.

To right size, do the following:

1. Identify your current needs based on data. For typical Windows workloads, you can use built-in operating system tools like Performance Monitor.
2. In Performance Monitor, use the following counters to gauge your current performance needs. The capture interval is set to one second, with a maximum log size of 1,000 MB and overwrite enabled.

```
Logman.exe create counter PerfLog-Short -o "c:\perflogs\PerfLog-Long.blg" -f bincirc
 -v mmddhhmm -max 1024 -c "\LogicalDisk(*)\*" "\Memory\*" "\.NET CLR Memory(*)\*"
 "\Cache\*" "\Network Interface(*)\*" "\Paging File(*)\*" "\PhysicalDisk(*)\*"
 "\Processor(*)\*" "\Processor Information(*)\*" "\Process(*)\*" "\Thread(*)\*"
 "\Redirector\*" "\Server\*" "\System\*" "\Server Work Queues(*)\*" "\Terminal
 Services\*" -si 00:00:01
```

3. To start the log capture, run the `logman start PerfLog-Short` command. To stop the log capture, run the `logman stop PerfLog-Short` command.

> **Note**
>
> You can find performance log files in **c:\perflogs** on the server running the capture. For more information, see Windows Performance Monitor Overview in the Microsoft documentation.

4. After you identify the correct the configuration, test if your estimate is correct on the Amazon FSx file system by using disk stress tools like Microsoft DISKSPD.
5. If you're satisfied with the performance, cut over to the file share.

We recommend a conservative approach to storage capacity as it can only be scaled up. Throughput capacity can be scaled up and down as needed.

Additional resources

- Amazon FSx for NetApp ONTAP FAQs (AWS website)
- Optimizing Amazon FSx for Windows File Server performance with new metrics (AWS Storage Blog)

Enable data deduplication in Amazon FSx

Overview

Data deduplication is a feature that enables you store your data more efficiently and with less capacity requirements. It involves finding and removing duplication within data without compromising its fidelity or integrity. Data deduplication uses subfile variable-size chunking and compression, which deliver optimization ratios of 2:1 for general file servers and up to 20:1 for virtualization data. Data deduplication is much more effective than NTFS compression. Inherent in the deduplication architecture is resiliency during hardware failures—with full checksum validation on data and metadata, including redundancy for metadata and the most accessed data chunks.

FSx for Windows File Server fully supports data deduplication. Using it can lead to an average savings of 50–60% for general-purpose file shares. Within shares, savings range from 30–50% for user documents and up to 70–80% for software development datasets. It's important to understand that the storage savings that you can achieve with data deduplication depend on the nature of your dataset, including how much duplication exists across files. Deduplication is not a good option if the data stored is dynamic in nature.

Cost impact

To cope with data storage growth in the enterprise, administrators consolidate servers and make capacity scaling and data optimization key goals. Data deduplication's default settings can provide savings immediately, or administrators can fine-tune the settings to see additional gains. For example, you can configure deduplication to run only on certain file types, or you can create a custom job schedule.

At a high level, deduplication has three types of jobs: optimization, garbage collection, and scrubbing. Be aware that space won't be freed until you run a garbage collection job after optimization. You can schedule the job or you can manually run it. All settings available when you

schedule a data deduplication job are also available when you start a job manually (except for those which are scheduling-specific).

Even with only a 25-percent effective savings from deduplication, there's a significant cost savings for FSx for Windows File Server. These projected savings are based on an estimate in the AWS Pricing Calculator.

Cost optimization recommendations

Deduplication on FSx for Windows File Server file systems is not enabled by default. To enable deduplication by using remote management on PowerShell, you must run the `Enable-FSxDedup` command and then use the `Set-FSxDedupConfiguration` command to set the configuration. For more information, see Administering file systems in the FSx for Windows File Server documentation.

To enable deduplication, run the following command:

```
PS C:\Users\Admin> Invoke-Command -ComputerName amznfsxzzzzzzzz.corp.example.com -
ConfigurationName FSxRemoteAdmin -ScriptBlock {Enable-FsxDedup }
```

To verify your deduplication configuration, run the following command:

```
Invoke-Command -ComputerName amznfsxzzzzzzzz.corp.example.com -ConfigurationName
 FSxRemoteAdmin -ScriptBlock {
Set-FSxDedupSchedule -Name "CustomOptimization" -Type Optimization -Days
 Mon,Tues,Wed,Sat -Start 09:00 -DurationHours 7
}
```

By running the PowerShell `Measure-DedupFileMetadata` cmdlet, you can determine how much potential disk space can be reclaimed on a volume if you delete a group of folders, a single folder, or a single file, and then run a garbage collection job. Specifically, the `DedupDistinctSize` value tells you how much space you get back if you delete those files. Files often have chunks that are shared across other folders, so the deduplication engine calculates which chunks are unique and would be deleted after the garbage collection job.

The default data deduplication job schedules are designed to work well for recommended workloads and be as non-intrusive as possible (excluding the priority optimization job that's enabled for the backup usage type). If workloads have large resource requirements, we recommend that you schedule jobs run only during idle hours, or to reduce or increase the amount of system resources that a data deduplication job is allowed to consume.

By default, data deduplication uses 25 percent of the memory available. However, this can be increased by using `-memory switch`. For optimization jobs, we recommend that you set a range from 15 to 50. For scheduled jobs, you can use higher memory consumption. For example, with garbage collection and scrubbing jobs (which you typically schedule to run in off hours), you can set higher memory consumption (such as 50).

For additional information regarding data deduplication settings, see Reducing storage costs with Data Deduplication in the FSx for Windows File Server documentation.

Additional resources

- Understanding Data Deduplication (Microsoft documentation)
- Reducing storage costs with Data Deduplication (FSx for Windows File Server documentation)

Understand data sharding in FSx for Windows File Server

Overview

FSx for Windows File Server performance is configuration dependent. It's primarily based on storage type, storage capacity, and throughput configuration. The throughput capacity that you select determines the performance resources available for the file server—including the network I/O limits, the CPU and memory, and the disk I/O limits imposed by the file server. The storage capacity and storage type that you select determine the performance resources available for the storage volumes—the disk I/O limits imposed by the storage disks. In addition to performance, the configuration choices also influence the cost. FSx for Windows File Server pricing primarily depends on storage capacity and storage type, throughput capacity, backup, and data transferred.

If you have relatively large file storage and performance requirements, you can benefit from data sharding. Data sharding involves dividing your file data into smaller datasets (shards) and storing them across different file systems. Applications accessing your data from multiple instances can achieve high levels of performance by reading and writing to these shards in parallel. At the same time, you can still present a unified view under a common namespace to your applications. In addition, it can also help to scale file data storage beyond what each file system supports (64 TB) for large file datasets—up to hundreds of petabytes.

Cost impact

For large datasets, it's typically more effective to deploy multiple small FSx for Windows File Server file systems, rather than one large SSD share to achieve the same level of performance. Using

a combination of FSx for Windows File Server HDD and SSD storage types enables better cost savings, and enables you to match the workload with the best underlying disk subsystem. In the following tables, you can see the difference between a single 17 TB file system and compare it to multiple smaller file systems which add to the same capacity.

Large SSD file system with multiple workloads

Server name	Cost	Configuration	Region
Amazon FSx for Windows File Server	$5,716 USD	17 TB SSD 30 percent deduplication 256 Mbps 17 TB backup	US East (N. Virginia)

Partitioned workload using DFSN

Server name	Cost	Configuration	Region	Share
Amazon FSx for Windows File Server	$1,024 USD	2 TB SSD 20% deduplication 128 Mbps 2 TB backup Multi-AZ	US East (N. Virginia)	Share 1
Amazon FSx for Windows File Server	$2,132 USD	5 TB SSD 30% deduplication 256 Mbps	US East (N. Virginia)	Share 2

Server name	Cost	Configuration	Region	Share
		5 TB backup Multi-AZ		
Amazon FSx for Windows File Server	$1,036 USD	10 TB HDD 40% deduplication 128 Mbps 10 TB backup Multi-AZ	US East (N. Virginia)	Share 3
DFSN Windows EC2 instances	$27 USD	t3a.medium 2 vCPUs 4 GiB memory	US East (N. Virginia)	DFSN Instances

The annual cost for a large SSD file system is $68,592. The annual cost of a partitioned workload is $50,640. In this example, a 26 percent savings can be achieved while matching the workload to the appropriate backend storage. For more information about pricing estimation, see the AWS Pricing Calculator estimate.

Cost optimization recommendations

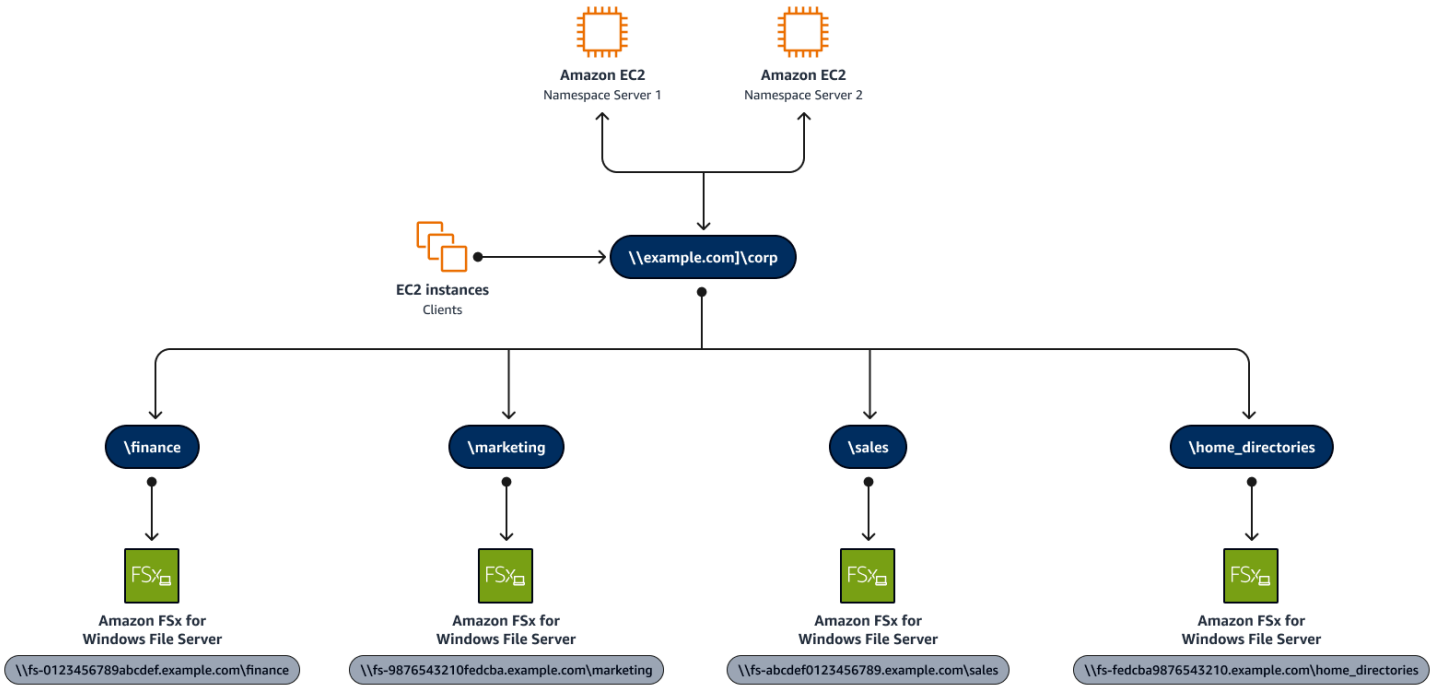
To deploy a data deduplication solution, you must set up a Microsoft DFS Namespace based on the type of data, I/O size, and I/O access pattern. Each namespace supports up to 50,000 file shares and hundreds of petabytes of storage capacity in aggregate.

It works most efficiently to choose a sharding convention that distributes I/O evenly across all the file systems you plan on using. Monitoring your workload will help with additional optimization or cost reduction. If you need help gauging performance information for the Amazon FSx file system, see FSx for Windows File Server performance in the FSx for Windows File Server documentation.

After you choose a sharding strategy, you can group the file systems for easy access to your shares by using DFS Namespaces. This enables users to see one homogenous file system, when in reality

they're accessing a variety of different file systems with purpose-built use cases. It's important to create the shares with a proper naming convention so your end users can easily decipher what workload the shares are designed for. It's also important to label production and non-production shares, so end users don't place files in the wrong file system by mistake.

The following diagram shows how a single DFS Namespace can be used as the access point for multiple Amazon FSx file systems.



Keep in mind the following:

- You can add existing FSx for Windows File Server shares to a DFS tree.
- Amazon FSx can't be added to the root of the DFS share path. You have only one subfolder.
- You must deploy an EC2 instance to serve the DFS namespace configuration.

For more information about DFS-N configuration, see DFS Namespaces overview in the Microsoft documentation. For more information about using DFS namespaces, see the Using DFS Namespaces with Amazon FSx for Windows File Server video on YouTube.

Additional resources

- Grouping multiple file systems with DFS Namespaces (Amazon FSx documentation)
- Walkthrough 6: Scaling out performance with shards (Amazon FSx documentation)

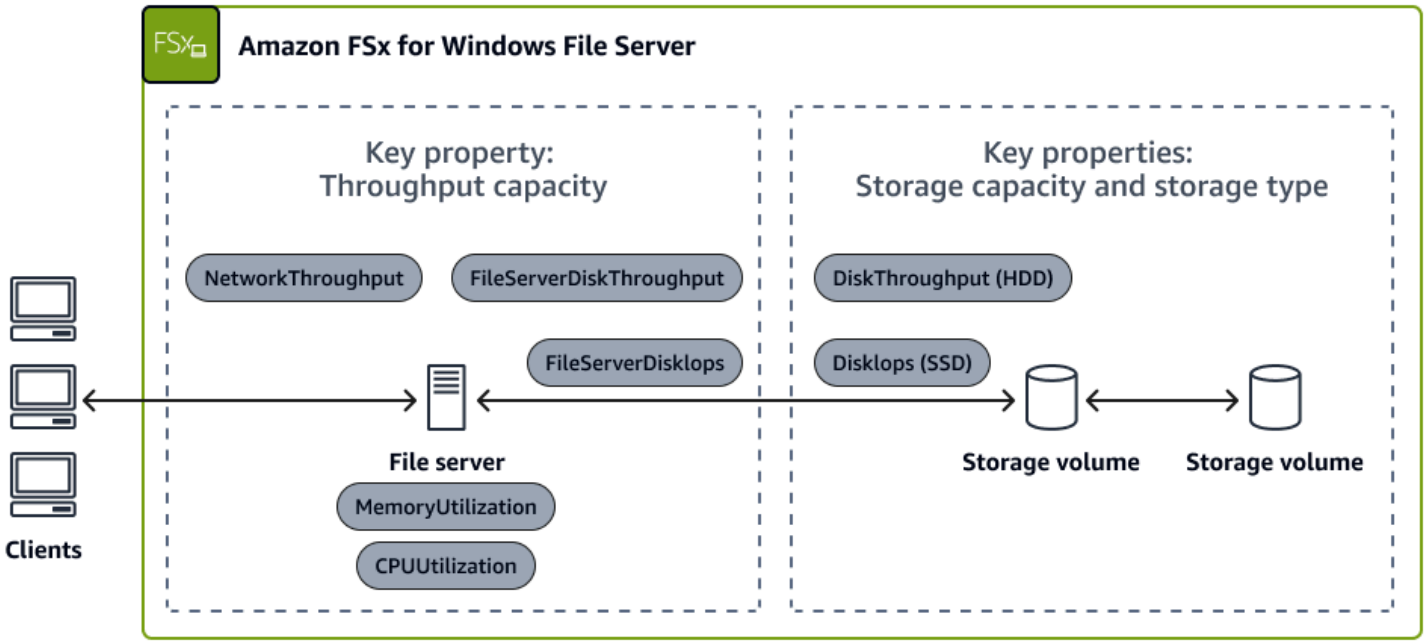
- Using DFS Namespaces with Amazon FSx for Windows File Server (AWS Labs)

Understand HDD volume usage in Amazon FSx

Overview

Amazon FSx for Windows File Server offers the flexibility to choose throughput independently of file system capacity. Two capacity settings are available: hard disk drive (HDD) and solid state drive (SSD).

The following diagram shows the relationship between throughput and storage settings.



With HDD-based storage, you receive a 12 IOPS baseline with 80 burst disk IOPS (IOPs per TiB of storage) and throughput of 12 Megabytes/second baseline with 80 burst Megabytes/second (per TiB of storage). For example, if your share is 50 TB in size, you get 50 * 12 = 600 as baseline for both throughput and IOPS.

Amazon FSx for Windows File Server provides 80 burst IOPS. Burst credits are refilled automatically when your utilization is below your baseline rate and are automatically consumed when your utilization is above your baseline rate. For example, if your workload is only utilizing 10 IOPS/TB for an hour (2 IOPS/TB below your baseline rate), you can then utilize 14 IOPS/TB (2 IOPS/TB above your baseline) for the following hour before running out of burst credits again.

For file operations, Amazon FSx for Windows File Server provides consistent sub-millisecond latencies with SSD storage and single-digit millisecond latencies with HDD storage. For all file systems, including those with HDD storage, Amazon FSx for Windows File Server provides a fast (in-memory) cache on the file server, so you can get high performance and sub-millisecond latencies for actively accessed data, irrespective of storage type.

When appropriate, the usage of HDD storage can help to reduce the cost of your overall storage capacity and provide a reliable storage platform for your needs.

Cost impact

Amazon FSx for Windows File Server performance depends on three factors: storage capacity, storage type, and throughput. Network I/O performance and in-memory cache size are solely determined by throughput capacity, while the disk I/O performance is determined by a combination of throughput capacity, storage type, and storage capacity.

While SSD is recommended for I/O intensive workloads, there are a variety of workloads whose needs can be met with HDD performance specs. HDD storage is designed for a broad spectrum of workloads, including home directories, user and departmental shares, and content management systems. For example, if your users only need low-latency access to data supporting current projects, then most of the data you're storing is infrequently accessed.

You can use the AWS Pricing Calculator to provide a comparison of a 20 TB SSD to an HDD file system in `us-east-1`. As the following table shows, even with no deduplication savings, the cost difference is significant when comparing HDD file systems to SSD file systems.

Amazon FSx file system configuration	Monthly costs
20 TB multi-AZ SSD (`us-east-1`)	$4,699.30
20 TB multi-AZ HDD (`us-east-1`)	$542.88
Estimated monthly savings	**$4,156.42**

> **Note**
>
> For additional FSx for Windows File Server savings, see the Enable data deduplication in Amazon FSx section of this guide.

By correctly identifying your performance needs, you can select right storage for your workload and reduce your costs.

Cost optimization recommendations

If you decide to use HDD storage, test your file system to ensure it can meet your performance requirements. HDD storage comes at a lower cost relative to SSD storage, but with lower levels of disk throughput and disk IOPS per unit of storage. It might be suitable for general-purpose user shares and home directories with low I/O requirements, large content management systems where data is retrieved infrequently, or datasets with small numbers of large files.

The storage type for an existing file system can't be changed. To convert the storage type for an Amazon FSx for Windows File Server file system, you must back up your existing file system and restore it to a new file system with the desired storage type. If you're looking to convert an existing SSD file system to an HDD file system, be aware that HDD has a much higher minimum capacity of 2 TB.

To restore a backup with a different storage type, do the following:

1. Back up your existing file system.
2. Create a new Amazon FSx file system with the HDD storage type.
3. Restore the backup to the new file system with the desired storage type.
4. Verify the new file system has the correct storage type and your data is intact.

Before moving your changes to production, we recommend that you analyze the performance of your Amazon FSx file system and verify the change is acceptable. For more guidance, see the Optimizing Amazon FSx for Windows File Server performance with new metrics post on the AWS Storage Blog.

Additional resources

- Optimizing costs with Amazon FSx (Amazon FSx documentation)

Use a single Availability Zone

Overview

This section explains when it's more beneficial to use a single Availability Zone implementation of Amazon FSx for Windows File Server. It covers scenarios where moving to a single Availability Zone reduces costs, while still enabling you to use Amazon FSx for Windows File Server as your managed file storage service. We recommend that you implement a single Availability Zone for Amazon FSx for production workloads. This can help ensure that you have the redundancy of multiple Availability Zones.

Cost impact

A single Availability Zone file system offers an approximately 40 percent cost reduction compared to a multiple Availability Zone implementation. With a multiple Availability Zone file system, you pay $0.230 per GB-month in SSD and $0.025 per GB-month in HDD compared to $0.130 per GB-month for SSD and $0.013 per GB-month for HDD on a single Availability Zone file system. You can see a comparison of costs and create your own estimates by using the AWS Pricing Calculator.

For a 10 TB file system this can be a difference of paying approximately $1,200 per month for multiple Availability Zones or $680 per month for a single Availability Zone. This example uses a 10 TB FSx for Windows File Server file system with SSD. The estimated savings for deduplication is 50 percent. Overall, a single Availability Zone has a lower cost of entry but comes with a few caveats that are covered in the next section.

Cost optimization recommendations

Single Availability Zone deployments

To make sure that a single Availability Zone is the right fit, take into consideration your own internal SLAs for the data being stored on FSx for Windows File Server. This entails understanding if you have SLAs to provide to your customers (internal and external) and if the three nines of availability for an Amazon FSx single Availability Zone will still allow you to meet those SLAs. FSx for Windows File Server with a single Availability Zone still has an uptime of 99.9 percent. The SLA for Amazon FSx for multiple Availability Zones is greater than 99.99 percent. For mission-critical workloads, we recommend that you use multiple Availability Zones over a single Availability Zone, even at additional cost.

Single Availability Zone deployments are ideal for workloads such as backups for SQL Server databases. They can provide low cost storage with an HDD tier, while still providing you with

consistent uptime. If you require a higher level of availability for a production workload, such as highly-available SQL servers or production application access, then a single Availability Zone isn't the right fit for your workloads. For backups, non-production testing, and development environments, an Amazon FSx single Availability Zone implementation can reduce your operational costs.

One use case where an Amazon FSx single Availability Zone file system works well is in a production situation where multiple Amazon FSx single Availability Zone file systems are in use, as the per-server storage in a highly available SQL Server cluster using Always On availability groups. For more information, see the Optimizing cost for your high availability SQL Server deployments on AWS post on the AWS Storage Blog.

Multi-Region replication

A potential option for reducing costs with a single Availability Zone file system (one where only a single Availability Zone file system works) is if you want to take advantage of a multi-Region replication with Amazon FSx. You can deploy Single-AZ file systems that support usage with native Microsoft DFS-R. DFS-R has the capability to automatically replicate data across Regions and multiple sites. For more information about configuring DFS-R using Amazon FSx, see Using Microsoft Distributed File System Replication in the Amazon FSx documentation.

Another alternative for multi-Region cost savings is using AWS Storage Gateway. This enables you to implement an Amazon FSx File Gateway in another Region for multi-Region access of Amazon FSx. For more information, see the AWS Storage Gateway section of this guide.

If you work across Regions, you must consider the data transfer cost for cross-Region data traffic. Traffic moving across Regions incurs a $0.02/Gb charge. So, if you have consistent data change at high volumes, this will add to your overall cost. For example, 1 TB of data transfer equals approximately $20.48.

Maintenance window

The maintenance window is a key consideration if you're using a Single Availability Zone with Amazon FSx. During the maintenance window, the Amazon FSx file system is unavailable for roughly 20 minutes, due to routine software patching for the underlying Windows Server. If you're using the file system for overnight backups, adjust the Amazon FSx maintenance window accordingly to avoid interruptions during your backup. You can adjust the maintenance window after creating your Amazon FSx file system.

Additional resources

- Availability and durability: Single-AZ and Multi-AZ file systems (Amazon FSx documentation)
- Amazon FSx for Windows File Server Pricing (AWS website)

AWS Storage Gateway

AWS Storage Gateway is a hybrid cloud storage service that connects on-premises environments with AWS cloud storage. It allows you to seamlessly integrate your existing on-premises infrastructure with AWS, enabling you to store and retrieve data from the cloud and run applications in a hybrid environment. For Windows workloads, you can use Storage Gateway to store and access data using native Windows protocols such as SMB and NFS. You can use Storage Gateway to reduce costs associated with running Windows workloads on AWS by using on-premises hardware and software as a bridge to the cloud. This enables you to take advantage of the scalability and cost-efficiency of AWS without having to make significant changes to your existing infrastructure.

Under the umbrella of Storage Gateway, you get Amazon S3 File Gateway, Amazon FSx File Gateway, Tape Gateway, and Volume Gateway. S3 File Gateway and FSx File Gateway are most commonly used with Microsoft workloads.

Amazon S3 File Gateway

Amazon S3 File Gateway enables you to store your files in Amazon S3 while providing access to your users by using traditional SMB shares. This provides a familiar user interface and helps reduce costs by storing your data in Amazon S3 and taking advantage of the various Amazon S3 storage tiers. You can implement Storage Gateway with S3 Intelligent Tiering to help you automatically move lifecycle files to the lowest cost storage tiers to lower your costs even further. We recommend S3 File Gateway for scale-out, read-only access, fast repeated reads (from cache), and database dumps. It's not generally recommended for high performance or high availability writes, editing files, or departmental shares.

Amazon FSx File Gateway

Amazon FSx File Gateway can also offer cost savings when working with Amazon FSx Windows file systems. You can stand up an FSx File Gateway to provide localized access to an Amazon FSx file system in another Region to avoid the costs of having two independent files systems. This can also

be helpful if you have multiple on-premises file servers and want to consolidate those to avoid paying for multiple hardware devices.

Cost impact

Amazon S3 File Gateway

Setting up S3 File Gateway is easy because you can use the launch wizard for Storage Gateway. You can deploy the gateway in a matter of minutes by using an EC2 instance in your AWS environment. After the gateway is set up, you can configure Storage Gateway shares to be accessible through the SMB and NFS protocols. For typical Windows workloads, you can also use this setup to take advantage of an Active Directory environment and set permissions on your file shares. You can effectively integrate a Storage Gateway into your normal usage, as it will work as a typical Windows file share. Files and folders are stored as objects and NTFS access control lists (ACLs) as metadata.

The following table compares the costs of 10 TB of storage with three available storage options:

- FSx for Windows File Server
- Amazon S3 File Gateway
- Amazon Elastic Block Store (Amazon EBS)

The price to store 10 TB of storage in considerably less expensive if you use Amazon S3, because you can partition your data into various usage tiers. In the pricing estimate, S3 Intelligent Tiering is used for its pricing flexibility. This includes 80 percent in S3 Standard, 10 percent in Infrequent Access, and 10 percent in Amazon Glacier. Although you can use Amazon Glacier, it's important to set the proper lifecycle rules to make sure that any files moved to Amazon Glacier won't need to be accessed immediately. Amazon Glacier is purely for archival usage, not regular access usage.

Storage systems	Cost for 10 TB of storage	Region
FSx for Windows File Server (assuming 50% savings for deduplication)	$683.20 USD SSD	US East (N. Virginia)
Amazon S3 File Gateway	$449.51 USD Intelligent Tiering	US East (N. Virginia)

Storage systems	Cost for 10 TB of storage	Region
Amazon EBS	$1,335.69 USD GP3	US East (N. Virginia)

Consider the following:

- In Amazon Glacier, you receive generic I/O errors unless you use the RestoreObject API to restore the object back to Amazon S3. We recommend that you use a notification for this I/O error by using Amazon CloudWatch Events. That way, your operations team can react to a user getting this error on a file they might need to access. For more information about these errors, see Error: InaccessibleStorageClass in the Amazon S3 File Gateway documentation.
- In addition to the Amazon Glacier limitation on access, there are only 10 ACLs allowed per object/folder on Storage Gateway. Before you decide to use Storage Gateway, make sure that you don't need more than 10 ACL entries.

Amazon FSx File Gateway

Similar to an Amazon S3 File Gateway, an FSx File Gateway provides access to a file system that retains the data long-term. In the Amazon S3 File Gateway, the data resides in Amazon S3. For FSx File Gateway, your data resides on FSx for Windows File Server. Although Multi-AZ options are available for FSx for Windows File Server, there isn't a multi-Region option. If you have a global company or remote office, you may need to provide a shared storage platform that's geographically closer to the end user to avoid latency. If you were to deploy another Amazon FSx file system, this would add the cost of an entirely new Amazon FSx for Windows File Server file system and the necessary storage. To avoid creating an entirely new file system and duplicating costs, you can deploy FSx File Gateway in the secondary Region. This provides localized access to files for users, while helping to reduce your overall costs.

Storage systems	Cost for 10 TB of storage	Region
Amazon FSx for Windows File Server	$683.20 USD SSD	US East (N. Virginia)
Amazon FSx File Gateway	$503.70 / Single Gateway	US East (N. Virginia)

> **Note**
>
> Prices in the preceding table are based on Storage Gateway pricing.

Keep in mind the following:

- FSx File Gateway can help you save approximately $180 per month (or $2100 annually) for multi-Region workloads.
- Data transfer charges are much lower with FSx File Gateway, because it only needs to cache the files being accessed regularly and not a full secondary copy.
- Although you can have two deployments of FSx for Windows File Server in different Regions and keep them updated with AWS Backup or AWS DataSync, neither option is near real time.

Cost optimization recommendations

Amazon S3 File Gateway

S3 File Gateway provides a low-cost option for storing files, but there are some issues to consider regarding how it implements and uses the file system. For instance, S3 File Gateway requires the usage of a virtual machine to run Storage Gateway software. In AWS, Storage Gateway is deployed in Amazon EC2 by using an m5.xlarge instance, by default. If you want to reduce your on-premises storage costs, you can deploy Storage Gateway as a virtual appliance on virtualization platforms such as VMware and Hyper-V.

High availability considerations

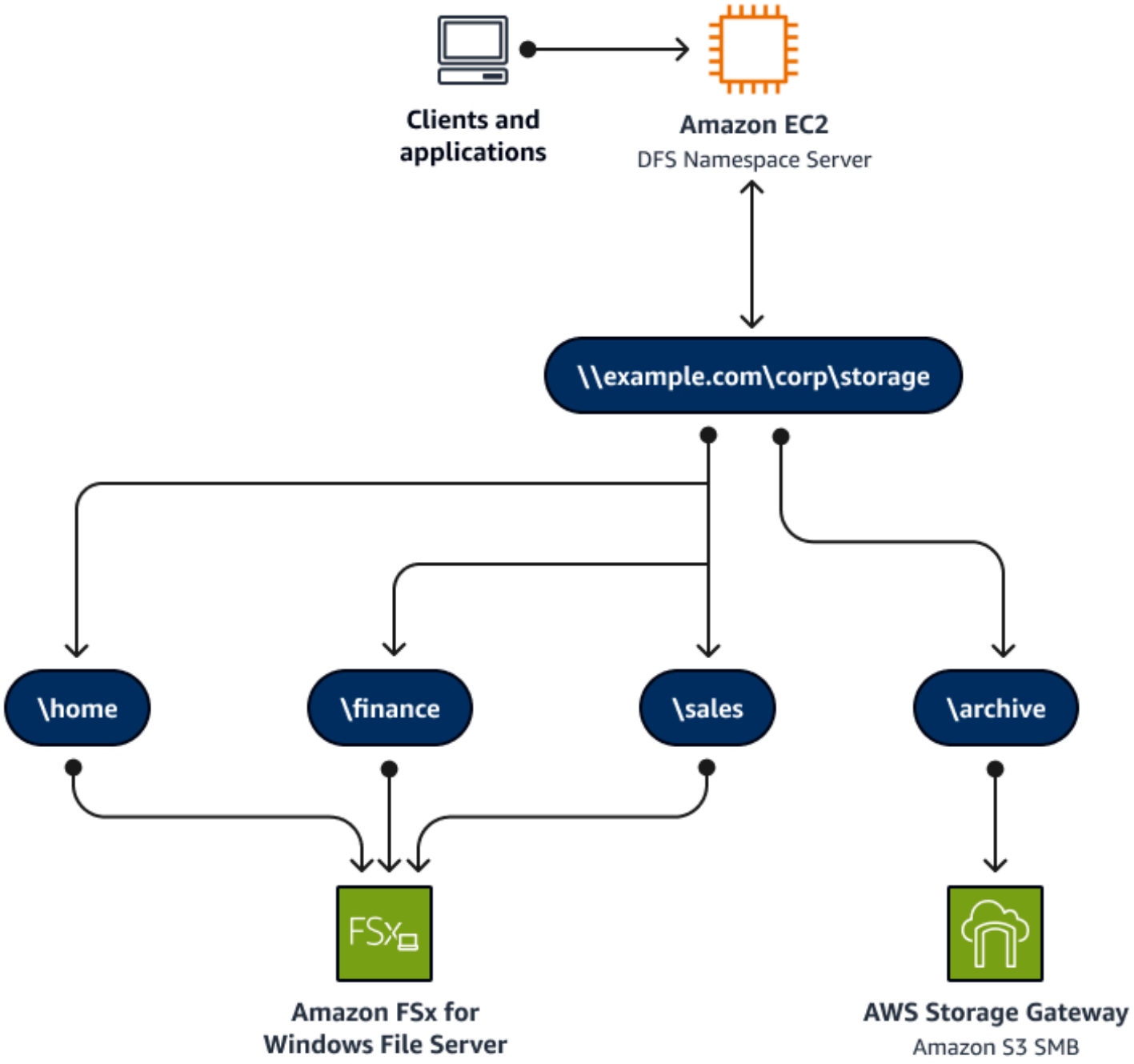
Running Storage Gateway is a single point of failure for access to the files. To prevent unnecessary downtime, we recommend that you implement strict access control on which users can make changes to or stop and start the Storage Gateway instance. Additionally, for deployments on AWS, it's beneficial to use Amazon Data Lifecycle Manager to create routing snapshots to quickly recover your Storage Gateway implementation. If you're running Storage Gateway on-premises using VMware, you can configure it for high availability.

Running multiple file systems

Separating your daily-use file workloads from your archive workloads can help you avoid unnecessary storage costs. Storage Gateway has the ability to be deployed alongside an FSx for Windows File Server file system. By using DFS Namespaces, you can present your primary daily-

use storage running on FSx for Windows File Server and your storage running in Amazon S3 (that's accessed through Storage Gateway).

The following diagram shows how a single DFS Namespace can be used as the frontend access point for different backend storage options.



Clients are directed to a folder structure, such as **\\example.com\storage**. This main directory contains the sub-directories. An FSx for Windows File Server file system contains the file shares

accessed on a normal basis. You can use a file share created on Storage Gateway for archive data. Users can manually archive items to the archive folder or you can build a process to automate moving some files from your normal file shares to the archive folder.

Consider the following:

- Review your storage requirements and provide adequate storage for the cache.
- Add your gateway to your Active Directory configuration and use standard Windows ACLs for access to files.

FSx File Gateway

The deployment of FSx File Gateway is similar to the deployment of S3 File Gateway, but it's even easier if you use the launch wizard. For detailed instructions, see Step 3: Create and activate an Amazon FSx File Gateway in the Amazon FSx File Gateway documentation. After you deploy FSx File Gateway in your environment, you can associate it to your existing Amazon FSx file systems and gain access to your files.

Storage is the primary consideration when deploying FSx File Gateway. The default storage provides 150 GB, which is a decent amount of space for caching files. Creating monitoring alerts for low free space can help with storage right sizing without overallocation.

Additional resources

- AWS Storage Gateway resources (AWS documentation)

Active Directory

Amazon Elastic Compute Cloud (Amazon EC2) running Windows Server is a secure, reliable, and high-performance environment for deploying Windows-based applications and workloads. You can provision instances quickly and scale up or scale down as needed, while only paying for what you use. Active Directory services are used as the primary source of identity management in Windows Server environments.

This section covers the following topics:

- Self-managed Active Directory on Amazon EC2
- AWS Managed Microsoft AD
- AD Connector

Self-managed Active Directory on Amazon EC2

Overview

This section provides recommendations for reducing the cost of running Active Directory on Amazon Elastic Compute Cloud (Amazon EC2). The primary focus is on making sure that you can size the Active Directory domain controllers appropriately and use the flexibility of the AWS Cloud to adjust as needed for your environment. AWS can help you easily stop an instance and resize it to meet your changing needs, or downsize the instance if you scale up too fast. Choosing the right instance size and type can result in significant savings.

Cost impact

The following table shows the difference between choosing a burstable instance family instance over a general purpose instance. This choice can save you a considerable amount of money each month. Appropriate planning and sizing of your instance can help you to manage costs.

Instance type	Number of instances	vCPU	Memory	Cost
t3a.medium	2	2	8	$81.76/month
m5a.large	2	2	8	$259.88/month

For more information about costs, see the AWS Pricing Calculator estimate.

A savings of $178.12 per month ends up being over $2,000 in savings per year for your domain controllers. Keep in mind that is for a small footprint of just two domain controllers in one account. At scale with multiple accounts and additional domain controllers, such savings can add up to a significant cost reduction.

Cost optimization recommendations

Microsoft provides capacity planning recommendations for when you're deploying your Active Directory environment. We recommend that you take the following main components into consideration when you plan or scale your Active Directory environment:

- Memory
- Network
- Storage
- Processor

While keeping these main components in mind, you can work through selecting an instance type that makes sense for your Active Directory environment on AWS. This section covers a few example Active Directory to AWS deployment scenarios. These scenarios make it clear that it's not necessary to replicate your on-premises environment in AWS, if you don't plan to handle the same number of users and computers as you do in your on-premises environment.

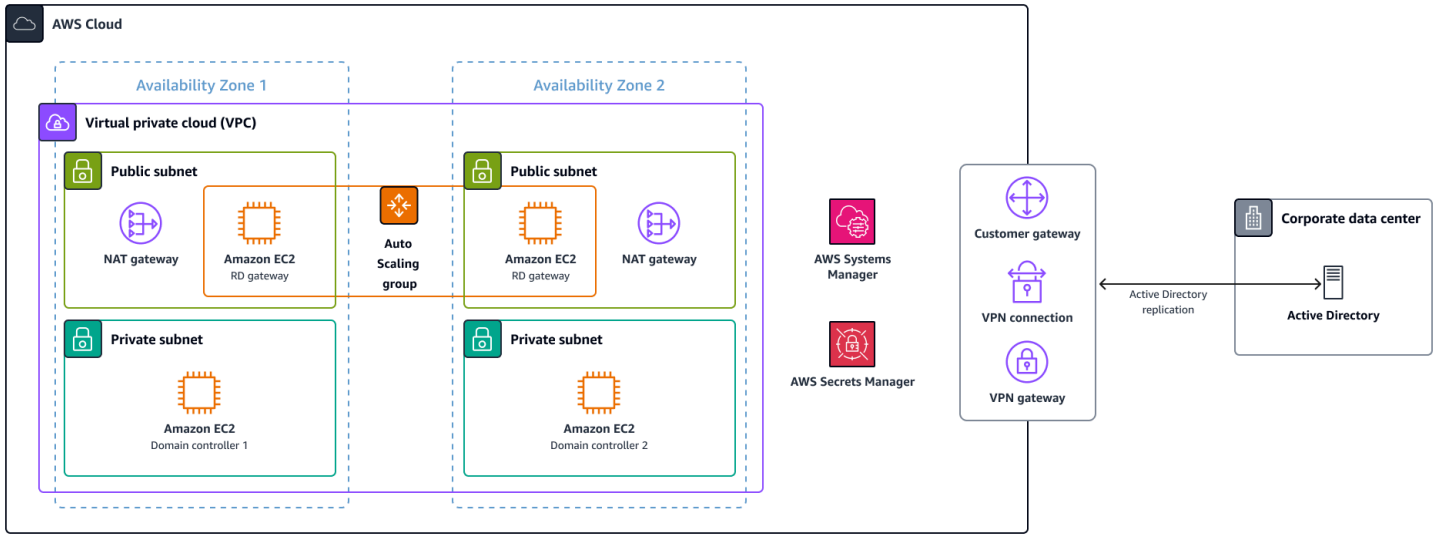
The following table highlights important components regarding vCPU, memory, and disk for your AWS footprint.

Component	Estimates
Storage/database size	40–60 KB for each user
RAM	Database size Base operating system recommendations Third-party applications
Network	1 GB

Component	Estimates
CPU	1,000 concurrent users for each core

Hybrid deployment scenario

The following diagram shows an example architecture for a hybrid deployment of Active Directory.



As the diagram shows, you typically have an on-premises footprint and then expand this into the AWS Cloud. In the initial phases of a migration, you typically won't have all your users and servers deployed in AWS. That's why it's important to initially deploy a smaller sized footprint to save money on the migration efforts.

If you're going to maintain an on-premises footprint with servers and users authenticating on premises, then you won't need the same footprint for domain controllers in AWS. By following Active Directory best practices, you can implement proper Active Directory sites and services to authenticate users and computers to your on-premises footprint, while only authenticating your AWS footprint to the domain controllers in AWS. This enables you to avoid oversizing your Active Directory footprint on AWS by limiting the use to just AWS resources and not all of your on-premises infrastructure. For guidance designing a hybrid setup, see Proper placement of domain controllers and site considerations in the Microsoft documentation.

Optimize for an AWS migration by right sizing

If you're deploying a new instance of Active Directory for your users or plan to fully migrate to AWS for your Active Directory infrastructure, we recommend that you plan the sizing against Microsoft's recommendations for vCPU, memory, and disk space for the instances choice in the preceding table.

If this is a new footprint, you can start small and take advantage of the ability to easily change instance types to resize your environment as it grows on AWS. The Windows on Amazon EC2 section of this guide shows you how to monitor and review your CPU and memory utilization on AWS. That way, you know when to increase size of your EC2 instance.

If you're fully migrating your on-premises Active Directory environment to AWS, you can implement the same sizing plans to ensure proper performance. Before duplicating what you have on premises in AWS, we recommend that you complete a thorough review of your Active Directory environment. This can help you prevent overprovisioning. Be sure to use Performance Monitor to collect information about the amount of traffic and utilization for your existing domain controllers. This can give you an understanding of the overall usage so that you can right size and ultimately reduce your costs.

Optimize Active Directory on AWS

If you're running Active Directory on AWS, it's important to also continuously monitor utilization and change instance sizes as needed to reduce your spending. You can use AWS Compute Optimizer to get information about the resources that you're running in AWS. For information about using Compute Optimizer to right size your Windows workloads, see the Windows on Amazon EC2 section of this guide. For a more comprehensive deep dive, you can use Performance Monitor to monitor the utilization of your Active Directory domain controllers, assess performance, and then resize accordingly.

You can also use CloudWatch to monitor the performance of domain controllers. To optimize your domain controllers (scaling up or down), you can use the metrics available in CloudWatch to help you make the right decisions. You can use the CloudWatch agent to configure custom Performance Monitor metrics to be sent for data collection. For instructions, see How can I use the CloudWatch agent to view metrics for Performance Monitor on a Windows server? in the AWS Knowledge Center.

After you deploy the CloudWatch agent, you can configure the following metrics within the agent configuration file under `metrics_collected`:

Metric category	Metric name
Database to instances (NTDSA)	Database cache % hit
I/O database reads average latency	
I/O database reads/sec	
I/O log writes average latency	
DirectoryServices (NTDS)	LDAP bind time
DRA pending replication operations	
DRA pending replication synchronizations	
DNS	Recursive queries/sec
Recursive query failure/sec	
TCP query received/sec	
Total query received/sec	
Total response sent/sec	
UDP query received/sec	
LogicalDisk	Avg. disk queue length
% free space	
Memory	% committed bytes in use
Long-term average standby cache lifetime(s)	
Network interface	Bytes sent/sec
Bytes Received/sec	
Current bandwidth	

Metric category	Metric name
NTDS	ATQ estimated queue delay
ATQ request latency	
DS directory reads/sec	
DS directory searches/sec	
DS directory writes/sec	
LDAP client sessions	
LDAP searches/sec	
LDAP successful binds/sec	
Processor	% processor time
Security system-wide statistics	Kerberos authentications
NTLM authentications	

Additional resources

- Active Directory Domain Services on AWS: Partner Solution Deployment Guide (AWS documentation)
- Capacity planning for Active Directory Domain Services (Microsoft documentation)
- Design considerations for running Active Directory on EC2 instances (AWS Whitepapers)

AWS Managed Microsoft AD

Overview

AWS Directory Service for Microsoft Active Directory, also known as AWS Managed Microsoft AD, is powered by a Windows Server Active Directory and managed by AWS. You can use AWS Managed Microsoft AD to migrate a broad range of Active Directory–aware applications to the AWS

Cloud. AWS Managed Microsoft AD works with a variety of native Active Directory applications and services. It also supports AWS managed applications and services. While there are not many cost optimization levers for AWS Managed Microsoft AD due to the service and its billing mechanisms, there are some design tenets that can help you keep costs at a minimum.

Cost impact

Since AWS Managed Microsoft AD is a managed service based on present SKUs, sizing is a relatively straightforward process. Currently there are two sizing SKUs available: Standard and Enterprise editions. Other SKUs include directory sharing, adding additional domain controllers (including additional Regions), and cross-Region data transfer.

Cost optimization recommendations

There are differences between AWS Managed Microsoft AD Standard Edition and AWS Managed Microsoft AD Enterprise Edition. Enterprise Edition supports up to 500,000 Active Directory objects, 500 account shares (soft limit), and has multi-Region support. Standard Edition supports up to 30,000 Active Directory objects, five account shares (soft limit to approximately 25 maximum), and doesn't have multi-Region support.

> **Note**
>
> The upper limits of Active Directory objects are approximations. Your directory might support more or fewer objects depending on their size and the behavior and performance needs of your applications.

The questions to consider prior to selecting your directory type are:

- Is multi-Region support required?
- Is the directory going to be shared with over 25 accounts?
- Is the Active Directory object count going to be over 30,000?

If the answer is yes to any of the above questions, then Enterprise Edition is required. If the answer to all the questions is no, we recommend that you start with Standard Edition.

> **Note**
>
> You can upgrade a directory from Standard Edition to Enterprise Edition but a directory cannot be downgraded. Deploying Standard Edition isn't going through a one-way door. If you desire to upgrade your directory to Enterprise Edition, contact AWS.

There is a cost for each share when you share directories in AWS Managed Microsoft AD Enterprise Edition. This is less than the cost of deploying a directory in each account, but keep in mind that sharing costs can creep up if left unchecked. We recommend that you only share directories with accounts containing Amazon Relational Database Service (Amazon RDS) and Amazon FSx for Windows File Server, since only those services support this feature. Keep in mind that you have the option to integrate FSx for Windows File Server with your self-managed Active Directory, including an AWS Managed Microsoft AD. If only Amazon FSx is required in another account, then you can do a self-managed Amazon FSx deployment against the AWS Managed Microsoft AD without the need to share the directory.

When deciding when to deploy additional domain controllers, keep in mind that AWS Managed Microsoft AD supports only two subnets in separate Availability Zones in the same VPC. Adding additional domain controllers doesn't allow you to add additional subnets. To determine if you must add additional domain controllers due to performance issues, review the domain controller performance metrics in CloudWatch. This tells you if one or all domain controllers are being overwhelmed. If you determine that only one domain controller is being overwhelmed, adding additional domain controllers won't alleviate the load and you'll need to dig deeper into applications not load balancing across the currently available domain controllers. If all domain controllers are being heavily used, adding an additional domain controller could reduce the load on the existing domain controllers. For instructions on how to automate scaling, see How to automate AWS Managed Microsoft AD scaling based on utilization metrics in the AWS Security Blog.

If you extended your directory to multiple Regions, we recommend that you don't use the directory NETLOGON or SYSVOL shares for file storage. All domain controllers replicate the contents of those shares. Not using the shares for file storage keeps data transfer costs to a minimum.

You also have the option to enroll in an Enterprise Agreement with AWS. Enterprise Agreements give you the option to tailor agreements that best suit your needs. For more information, see Enterprise Customers.

Additional resources

- AWS Managed Microsoft AD quotas (AWS Directory Service documentation)
- AWS Directory Service Pricing (AWS website)
- Active Directory Domain Services on AWS (AWS Whitepapers)

AD Connector

Overview

AD Connector is a proxy service that provides an easy way to connect your existing on-premises Microsoft Active Directory to compatible AWS applications, such as Amazon WorkSpaces, Amazon Quick Suite, and seamless domain join for Amazon Elastic Compute Cloud (Amazon EC2) instances, without caching any information in the cloud. You can use AD Connector to add one service account to your Active Directory. AD Connector eliminates the need for directory synchronization or the cost and complexity of hosting a federation infrastructure. While there are not many cost optimization levers for AD Connector due to the nature of the service and its billing mechanisms, you can follow the design recommendations in this section to keep costs to a minimum.

Cost impact

AD Connector is a managed service based on preset SKUs. This makes sizing a straightforward process. There are two sizing SKUs available: small and large sizes. You can use the AWS Pricing Calculator for cost estimations involving AD Connector.

Cost optimization recommendations

Other than backend compute resources there is no difference between the small and large connector sizes.

The questions to consider prior to selecting your directory type are:

- Is there a large number (10,000+) of active users using AWS applications integrated with the AD Connector?
- Is the user a member of many, deep, or circular nested groups?

If the answer to both questions is no, we recommend you start with the small size. If you answer yes to any of the above questions, then a large size might be worth considering. You can start with a small size AD Connector and, if the directory becomes impaired due to performance, you can request the directory be upgraded to the large size.

> **Note**
>
> You can upgrade an AD Connector from small to large, but an AD Connector cannot be downgraded.

Most of the performance issues are not related to the AD Connector, but the on-premises Active Directory domain controllers being overwhelmed due to many users being a member of many, deep, or circular nested groups.

You also have the option to enroll in an Enterprise Agreement with AWS. Enterprise Agreements give you the option to tailor agreements that best suit your needs. For more information, see Enterprise Customers.

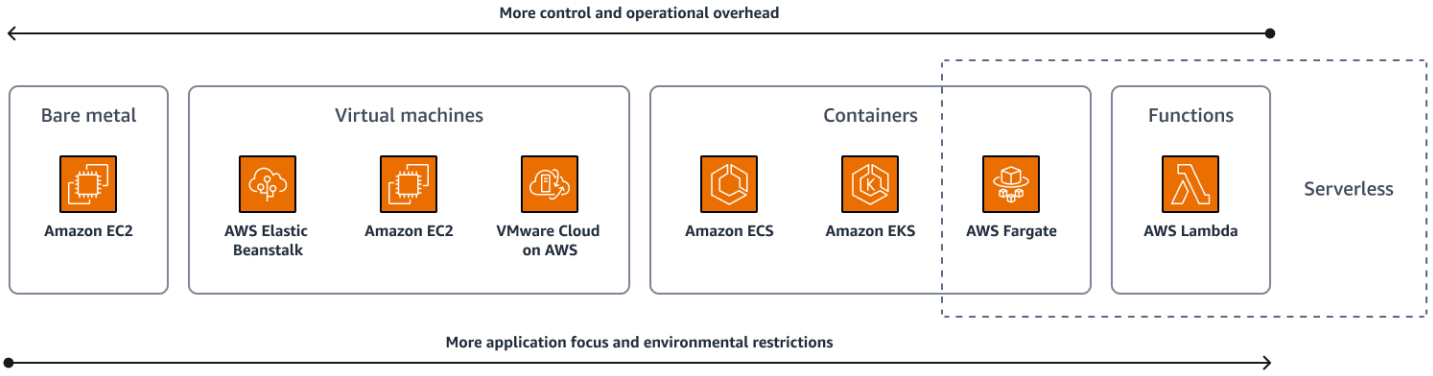
Additional resources

- AD Connector quotas (AWS Directory Service documentation)
- Other directory types pricing (AWS website)
- Active Directory Domain Services on AWS (AWS Whitepapers)

.NET

Developing and deploying .NET applications is an important key in helping you achieve the scale and agility offered by cloud computing. For many legacy .NET applications, the most suitable compute choice for running applications in AWS is using virtual machines, either through AWS Elastic Beanstalk or Amazon Elastic Compute Cloud (Amazon EC2). It's also possible to run .NET applications in Windows and Linux containers.

The introduction of .NET core enables you to design modern .NET applications that take advantage of all the cloud benefits. Modern applications can use the traditional set of compute choices and also target various types of serverless environments, including AWS Fargate or AWS Lambda. .NET 6+ now offers performant hosting of workloads on ARM64 EC2 instances such as the Graviton2 EC2 families. This enables access to the latest generation of processors available on Amazon EC2. This means that your applications can be hosted on compute specialized to your workload type, such as video encoding, web servers, and high-performance computing (HPC).



This section provides recommendations for helping you adapt your .NET applications to take advantage of the benefits of the cloud with a focus on cost efficiency.

This section covers the following topics:

- Refactor to modern .NET and move to Linux
- Containerize .NET apps
- Use Graviton instances and containers
- Support dynamic scaling for static .NET Framework apps
- Use caching to reduce database demand
- Consider serverless .NET

- Consider purpose-built databases

Refactor to modern .NET and move to Linux

Overview

Modernizing legacy .NET Framework apps can help you improve security, performance, and scalability. An effective way to modernize .NET Framework apps is to migrate them to a modern .NET version (6+). Here are some of the key benefits for moving these applications to open-source .NET:

- To reduce Windows licensing costs by running them on a Linux operating system
- Take advantage of the availability of modern languages
- Get performance that's optimized to run on Linux

Many organizations are still running older versions of the .NET Framework. This can pose security risks, since the vulnerabilities in the older versions are no longer addressed by Microsoft. Microsoft has ended support for recent versions of .NET Framework 4.5.2, 4.6, and 4.6.1. It's very important to evaluate the risks and benefits for continuing to run the older versions of the framework. To reduce risk and reduce costs, it can be worth investing the time and effort into refactoring to a modern version of .NET.

Cost impact

Consider a general purpose EC2 instance type (m5), which offers a balance of compute, memory, and networking resources. These instances are suitable for a variety of applications such as web servers, mid-sized databases, and source code repositories.

For example, an on-demand m5.xlarge instance with 4 vCPUs and 16 GB memory on Windows Server (license included) in US East (N. Virginia) costs $274.48 monthly. The same resources on a Linux server cost $140.16 monthly. In this example, there is a 49 percent reduction in cost when you migrate your application from .NET Framework to a modern version of .NET and run your application on a Linux server. Your cost can vary depending on the options (for example, instance type, operating system, storage) you choose when selecting an EC2 instance. You can further optimize the costs by using Savings Plans or Reserved Instances. For more details, use the AWS Pricing Calculator to run cost estimates. For Windows-included instances, the license cost is $0.046 per vCPU per hour, regardless of pricing model.

Porting these .NET Framework applications to modern .NET requires developer effort. You must assess your applications and their dependencies to see if they're compatible with the target platform version. AWS Porting Assistant for .NET is an assistive tool that scans .NET Framework applications and generates a .NET compatibility assessment, helping you port your applications to be compatible with Linux faster. The Porting Assistant for .NET identifies incompatibilities with .NET, finds known replacements, and generates a detailed compatibility assessment. After porting your solution, you must make manual code changes for your project to be compiled successfully with dependencies. This reduces the manual effort involved in modernizing your applications to Linux. If your application supports ARM processors, moving to Linux unlocks the ability to use Graviton instances. This can help you achieve an additional 20 percent in further cost reductions. For more information, see Powering .NET 5 with AWS Graviton2: Benchmarks in the AWS Compute Blog.

There are other tools, such as AWS Toolkit for .NET Refactoring and the .NET Upgrade Assistant, which can help you with porting legacy .NET framework applications to modern .NET.

Cost optimization recommendations

To migrate .NET Framework apps, do the following:

1. **Prerequisites** – To use Porting Assistant for .NET, you must install .NET 5+ on the machine where you plan to analyze the application source code. The resources on the machine must have a minimum of 1.8 GHz processing speed, 4 GB of memory, and 5 Gb of storage space. For more information, see Prerequisites in the Porting Assistant for .NET documentation.
2. **Assessment** – Download Porting Assistant for .NET as an executable (download) file. You can download and install the tool on your machine to start the assessment of your applications. The assessment page contains ported projects, packages, and APIs that are incompatible with modern .NET. For this reason, you get build errors in the solution after the assessment. You can view or download the assessment findings to a CSV file. For more information, see Port a solution in the Porting Assistant for .NET documentation.
3. **Refactoring** – After assessing the application, you can port your projects to the target framework version. When you port a solution, your project files and some of the code will be modified by the Porting Assistant. You can check the logs to review the changes to your source code. In most cases, the code will require additional effort to complete the migration and testing to make it production ready. Depending up on the application, some of the changes may include entity framework, identity, and authentication. For more information, see Port a solution in the Porting Assistant for .NET documentation.

This is a first step to modernizing your applications to containers. There could be a number of business and technical drivers to modernize your .NET Framework apps to Linux containers. One of the significant drivers is reducing the total cost of ownership by moving away from a Windows operating system to Linux. This reduces licensing costs when migrating your application to a cross-platform version of .NET and to containers to optimize resource utilization.

After your application is ported to Linux, you can use AWS App2Container to containerize your application. App2Container uses Amazon ECS or Amazon EKS as endpoint services that you can deploy directly to. App2Container provides all the necessary infrastructure as code (IaC) deployment artifacts to containerize your applications repeatedly.

Additional considerations and resources

- If you have applications built on VB.NET (a legacy framework from 2002) and want to port them to .NET 6, see the Port legacy VB.NET applications to .NET 6.0 with Porting Assistant for .NET post on the Microsoft Workloads on AWS blog.
- If you have legacy applications on Windows Communication Foundation (WCF) and want to run them on modern .NET, you can adopt CoreWCF. For more information, see the Modernizing legacy WCF applications to CoreWCF using Porting Assistant for .NET post on the Microsoft Workloads on AWS blog.
- You can add porting assistant as an extension to your Visual Studio IDE. This enables you to perform all of the tasks necessary to convert your code without needing to switch between your IDE and the Porting Assistant for .NET tool. For more information, see the Accelerate .NET application modernization with Porting Assistant for .NET Visual Studio IDE extension post on the Microsoft Workloads on AWS blog.
- AWS Porting Assistant for .NET is now open source tool with the source code and compatibility analysis components of the assessment. This can encourage your developers to use and share .NET porting knowledge and best practices.
- You can port .NET framework applications to modern .NET on Linux by using the AWS Toolkit for .NET Refactoring. For more information, see the Accelerate .NET modernization with AWS Toolkit for .NET Refactoring post on the Microsoft Workloads on AWS blog.
- You can accelerate containerization and migration of ASP.NET Core applications to AWS using AWS App2Container.

Containerize .NET apps

Overview

Containers are a lightweight and efficient way to package and deploy applications in a consistent and reproducible manner. This section explains how you can use AWS Fargate, a serverless container service, to reduce the costs of your .NET applications while also providing scalable and reliable infrastructure.

Cost impact

Some factors that influence the effectiveness of using containers for cost savings include the size and complexity of the application, the number of applications that need to be deployed, and the level of traffic and demand on the applications. For small or simple applications, containers may not provide significant cost savings compared to traditional infrastructure approaches because the overhead of managing the containers and the associated services may actually increase costs. However, for larger or more complex applications, using containers can provide cost savings by improving resource utilization and reducing the number of required instances.

We recommend that you keep the following in mind when using containers for cost savings:

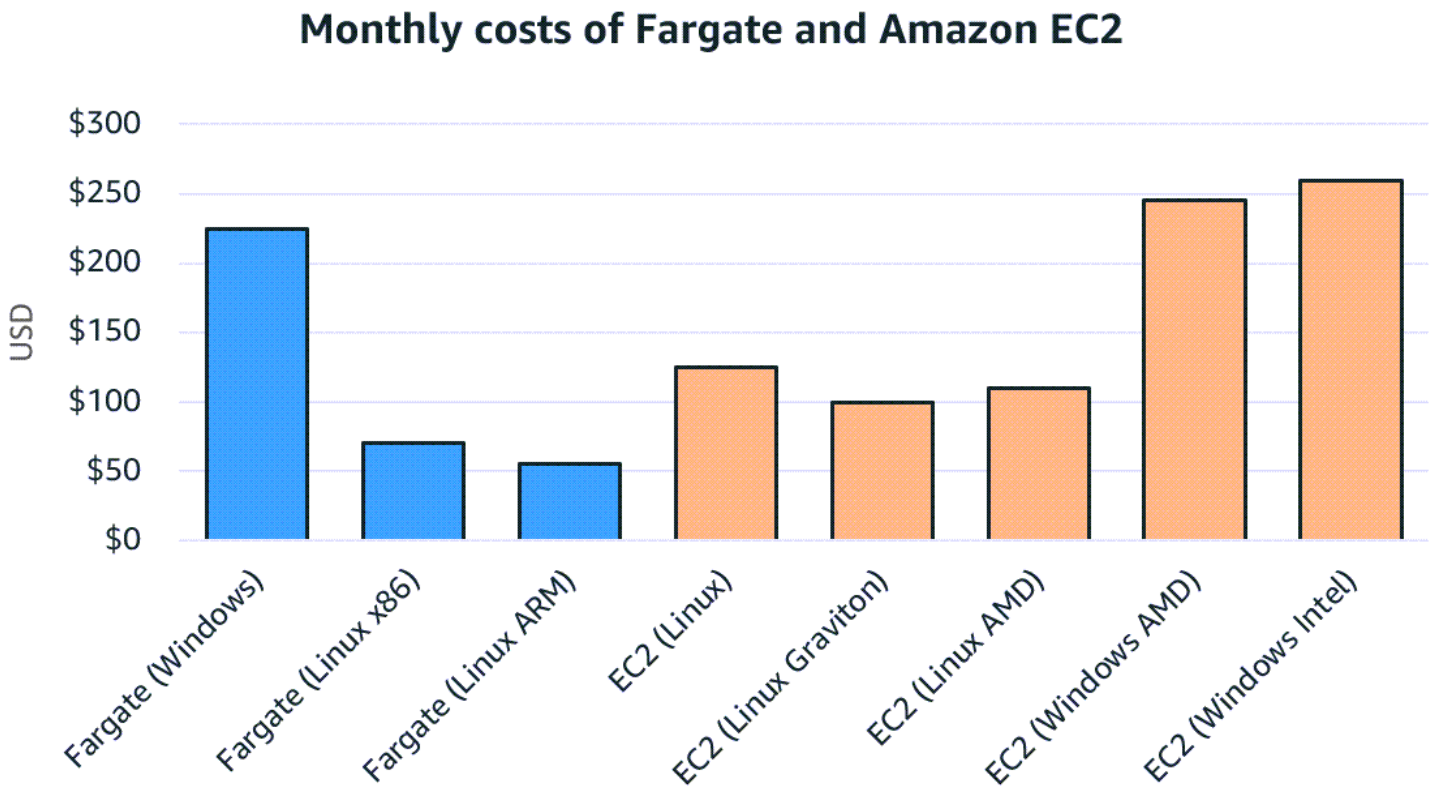
- **Application size and complexity** – Larger and more complex applications are better suited for containerization because they tend to require more resources and can benefit more from improved resource utilization.
- **Number of applications** – The more applications that your organization must deploy, the more cost savings can be achieved through containerization.
- **Traffic and demand** – Applications that experience high traffic and demand can benefit from the scalability and elasticity that containers provide. This can lead to cost savings.

Different architectures and operating systems affect container costs. If you're using Windows containers, costs may not decrease because of licensing considerations. Licensing costs are lower or absent with Linux containers. The following chart uses a basic configuration on AWS Fargate in the US East (Ohio) Region with the following settings: 30 tasks per month each running for 12 hours with 4 vCPUs and 8 GB of memory allocated.

You can choose from two primary compute platforms to run your containers on AWS: EC2-based container hosts and serverless or AWS Fargate. If you use Amazon Elastic Container Service

(Amazon ECS) instead of Fargate, then you must maintain running compute (instances) to allow the placement engine to instantiate containers when needed. If you use Fargate instead, only the compute capacity that is needed is provisioned.

The following chart shows the difference for equivalent containers using Fargate versus Amazon EC2. Because of the flexibility of Fargate, tasks for an application can run 12 hours per day, with zero utilization during off hours. However, for Amazon ECS, you must control compute capacity by using an Auto Scaling group of EC2 instances. This can lead to capacity running 24 hours a day, which can ultimately increase costs.



Cost optimization recommendations

Use Linux containers rather than Windows

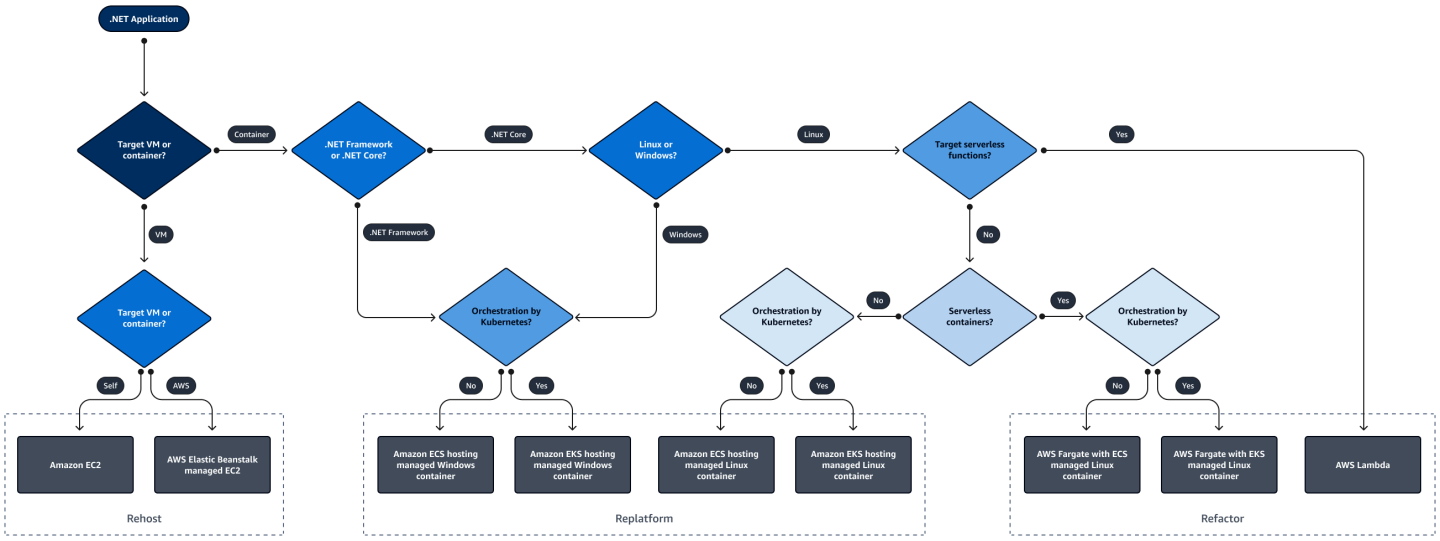
You can achieve significant savings if you use Linux containers instead of Windows containers. For example, you can achieve an approximately 45 percent savings on compute costs if you run the .NET Core on EC2 Linux instead of running the .NET Framework on EC2 Windows. You can get an additional 40 percent savings if you use the ARM architecture (AWS Graviton) instead of x86.

If you plan to run Linux-based containers for existing .NET Framework applications, you must port these applications to modern, cross-platform versions of .NET (such as .NET 6.0) in order to use

Linux containers. A major consideration is weighing the cost of refactoring compared with the cost savings gained through the reduced cost of Linux containers. For more information about porting your applications to modern .NET, see Porting Assistant for .NET in the AWS documentation.

Another benefit of moving to modern .NET (that is, away from the .NET Framework) is that additional modernization opportunities become available. For instance, you can consider rearchitecting your application to a microservices-based architecture that's more scalable, agile, and cost effective.

The following diagram illustrates the decision-making process for exploring modernization opportunities.



Take advantage of Savings Plans

Containers can help you take advantage of Compute Savings Plans to reduce your Fargate costs. The flexible discount model offers the same discounts as Convertible Reserved Instances. Fargate pricing is based on the vCPU and memory resources used from the time you start to download your container image until the Amazon ECS task terminates (rounded up to the nearest second). Savings Plans for Fargate offer savings of up to 50 percent on Fargate usage in exchange for a commitment to use a specific amount of compute usage (measured in dollars per hour) for a one-year or three-year term. You can use AWS Cost Explorer to help you choose a Savings Plans.

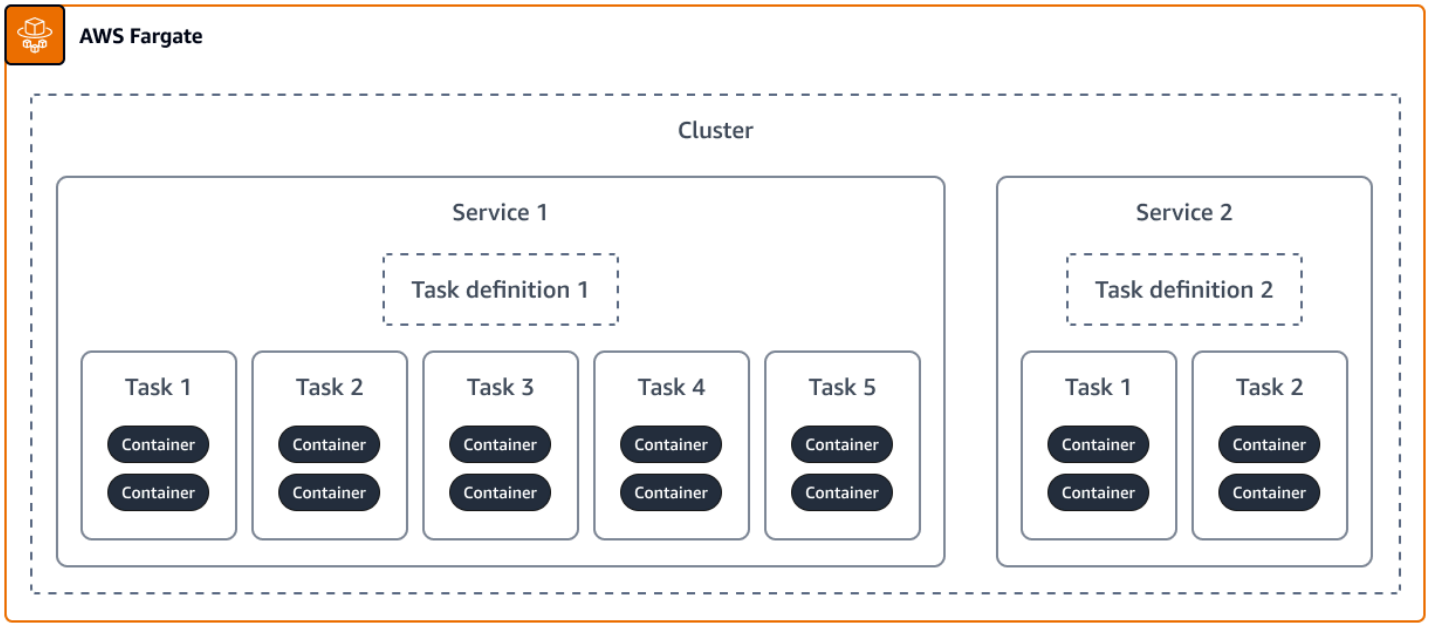
It's important to understand that Compute Savings Plans are applied to the usage that gets you the largest savings first. For example, if you're running a t3.medium Linux instance in `us-east-2` and an identical Windows t3.medium instance, the Linux instance receives the Savings Plans benefit first. This is because the Linux instance has a 50 percent savings potential whereas the

same Windows instance has a 35 percent savings potential. If you have other Savings Plans eligible resources running in your AWS account, such as Amazon EC2 or Lambda, then it's not necessary for your Savings Plans to be applied to Fargate first. For more information, see Understanding how Savings Plans apply to your AWS usage in the Savings Plans documentation and the Optimize spending for Windows on Amazon EC2 section of this guide.

Right size Fargate tasks

It's important to ensure that Fargate tasks are correctly sized to achieve the maximum degree of cost optimization. Frequently, developers don't have all the necessary usage information when initially determining the configurations for the Fargate tasks used in their applications. This can lead to overprovisioning of tasks and then result in unnecessary spending. To avoid this, we recommend that you load test applications running on Fargate to understand how a specific task configuration performs under different usage scenarios. You can use the load testing results, vCPU, memory allocation of the tasks, and auto scaling policies to find the right balance between performance and cost.

The following diagram shows how Compute Optimizer generates recommendations for the optimal task and container size.



One approach is to use a load testing tool, such as the one described in Distributed Load Testing on AWS, to establish a baseline for vCPU and memory utilization. After you run the load test to simulate a typical application load, then you can fine-tune the vCPU and memory configuration for the task until the baseline utilization is achieved.

Additional resources

- Cost Optimization Checklist for Amazon ECS and AWS Fargate (AWS Containers blog post)
- Theoretical cost optimization by Amazon ECS launch type: Fargate vs EC2 (AWS Containers blog post)
- Porting Assistant for .NET (AWS documentation)
- Distributed Load Testing on AWS (AWS Solutions Library)
- AWS Compute Optimizer launches support for Amazon ECS services on AWS Fargate (AWS Cloud Financial Management blog post)

Use Graviton instances and containers

Overview

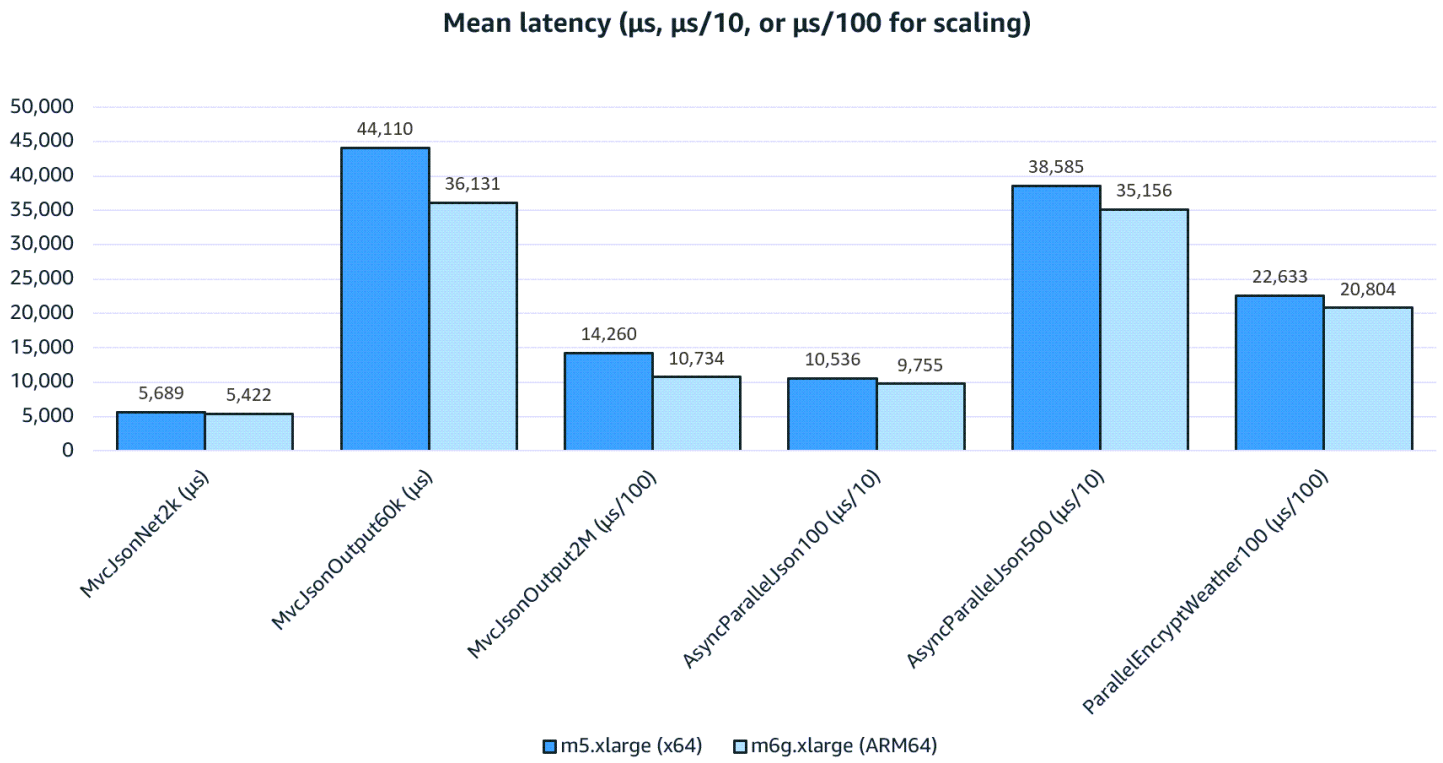
AWS Graviton instances are powered by ARM processors designed by AWS to deliver the best price performance for your cloud workloads running in Amazon Elastic Compute Cloud (Amazon EC2), including containers running in AWS. There are currently three generations of Graviton available for use on Amazon EC2. This guide focuses on the use of Graviton 2 and 3 with .NET applications because there is a significant cost savings when you use the latest versions of Graviton. Keep in mind that Graviton instances only run the Linux operating system. As a result, Graviton instances are a strong offering for .NET running on Linux, but aren't an option for the Windows operating system or legacy .NET Framework applications.

Graviton 3 is 60 percent more efficient over comparable EC2 instances with up to 40 percent better performance. This guide focuses on the cost benefits of using Graviton, but it's important to note that Graviton offers the additional benefits of performance improvements and improved environmental sustainability.

Cost impact

You can achieve up to 45 percent savings when you switch to Graviton. After you refactor any legacy .NET Framework applications to a modern .NET version, you unlock the capability of using Graviton instances. Moving to Graviton is an effective cost optimization technique for .NET developers.

The example in the following table shows the potential for performance improvements that you can achieve by migrating to Graviton instances.



For a full breakdown and explanation of the benchmarking approach used to create the results in the preceding diagram, see Powering .NET 5 with AWS Graviton2: Benchmarks in the AWS Compute Blog.

One of the reasons for the improved efficiency is the difference in the meaning of vCPU between x86 and Graviton. In the x86 architecture, a vCPU is a logical core achieved by hyperthreading. In Graviton, vCPU equates to a physical core which allows the vCPU to be fully committed to the workload.

The result with Graviton2 is a 40 percent better price performance over comparable x86/x64 instances. Graviton3 offers the following over Graviton2:

- An increased performance profile with up 25 percent better performance
- Up to two times higher floating-point performance
- Up to two times faster cryptographic workload performance
- Up to three times better machine learning performance

In addition, Graviton3 is the first instance in the cloud to feature DDR5 memory.

The following tables show the difference in cost savings between Graviton-based instances and equivalent x86-based instances.

This table shows Graviton savings of 19.20 percent.

Instance type	Architecture	vCPU	Memory (GB)	Hourly cost (on demand)
t4g.xlarge	ARM	4	16	$0.1344
t3.xlarge	x86	4	16	$0.1664

This table shows Graviton savings of 14.99 percent.

Instance type	Architecture	vCPU	Memory (GB)	Hourly cost (on demand)
c7g.4xlarge	ARM	16	32	$0.5781
c6i.4xlarge	x86	16	32	$0.6800

It's important to test your application's performance profile when considering Graviton. Graviton isn't a replacement for solid software development practices. You can use testing to verify if you're getting the most out of your underlying compute resources.

Cost optimization recommendations

There are several ways to take advantage of the Graviton processors/instances. This section walk you through the changes required to move from using an x86-architecture machine to Graviton (ARM) instances.

Change the runtime setting in Lambda

We recommend that you switch the runtime settings in AWS Lambda. For more information, see Modifying the runtime environment in the Lambda documentation. Since .NET is a compiled language, you must follow a build process to get this working. For an example of how to do this, see .NET on Graviton in GitHub.

Containers

For a containerized workload, create a multi-architecture container image. You can do this by specifying multiple architectures in the Docker build command. For example:

```
docker buildx build -t "myImageName:latest" --platform linux/amd64,linux/arm64 --push .
```

You can also use a tool such as AWS Cloud Development Kit (AWS CDK) to help orchestrate the build. For examples from Docker, see Building Multi-Arch Images for Arm and x86 with Docker Desktops in the Docker documentation.

Amazon EC2

To migrate to ARM from x86/x64, target the ARM architecture in the compile step. In Visual studio, you can create an ARM64 CPU. For instructions, see To configure a project to target Arm64 and other platforms in the Microsoft documentation.

If you're using the .NET CLI, then running the build on an ARM machine produces a Graviton compatible build. To see a demo, watch Accelerate .NET 6 performance with Arm64 on AWS Graviton2 on YouTube. Dependency issues will result in compile time errors that can then be addressed individually. As long as there are ARM libraries for any dependency, the transition should be relatively straightforward.

Additional resources

- How to build your containers for ARM and save with Graviton and Spot instances on Amazon ECS (AWS blog)
- AWS Lambda Functions Powered by AWS Graviton2 Processor – Run Your Functions on Arm and Get Up to 34% Better Price Performance (AWS blog)
- Migrating AWS Lambda functions to Arm-based AWS Graviton2 processors (AWS blog)
- Build and deploy .NET web applications to ARM-powered AWS Graviton 2 Amazon ECS Clusters using AWS CDK (AWS blog)
- Graviton Fast Start – A New Program to Help Move Your Workloads to AWS Graviton (AWS blog)
- Powering .NET 5 with AWS Graviton2: Benchmarks (AWS blog)

Support dynamic scaling for static .NET Framework apps

Overview

One of the key benefits of using the cloud for applications is elasticity, or the ability to scale compute in or out based on demand. This enables you to only pay for the compute capacity that you need, rather than provisioning for peak usage. Cyber Monday, in which online retailers can quickly get many times more traffic than normal (for example, thousands of percent within minutes), is a good example of elasticity.

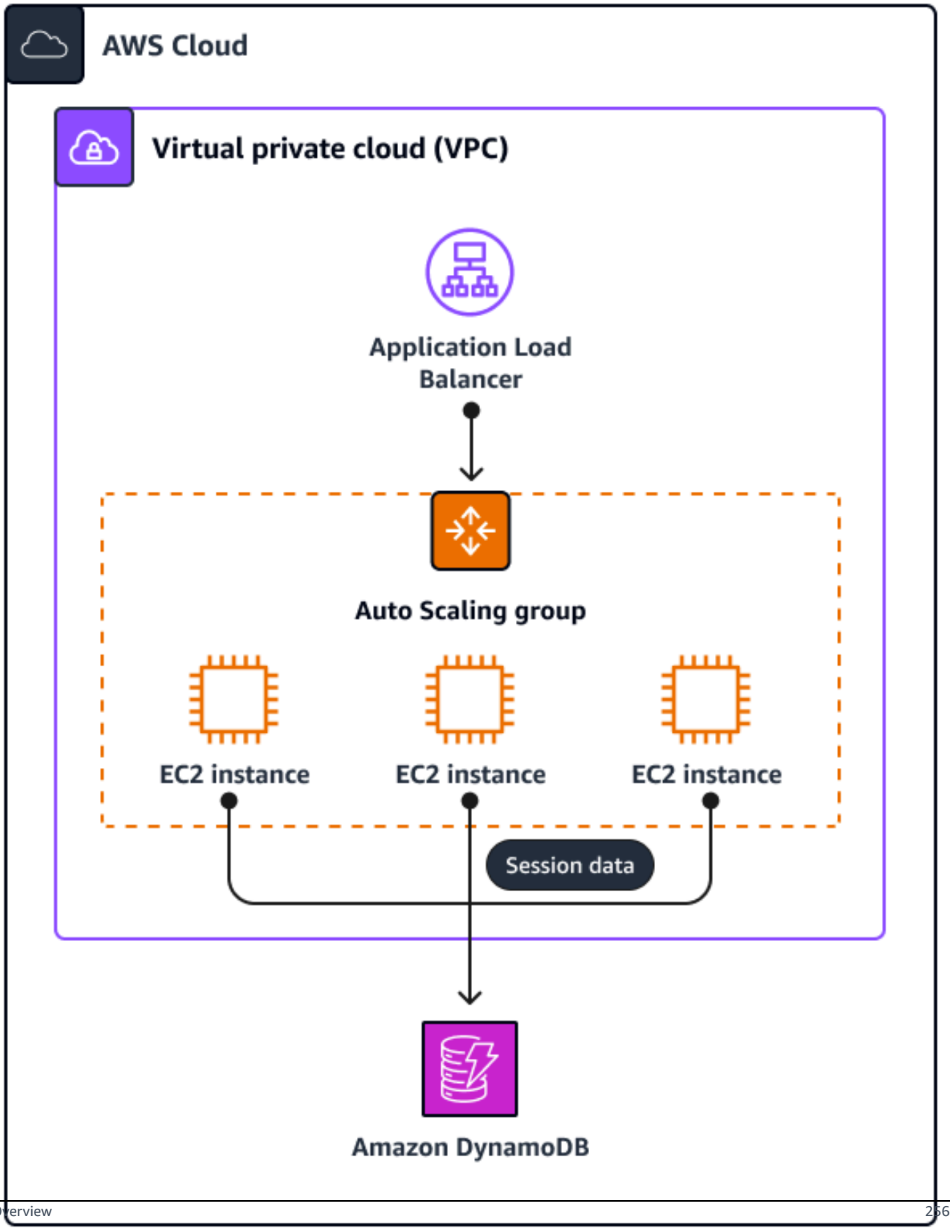
If you're bringing legacy .NET web applications to the cloud (for example, ASP.NET Framework applications running on IIS), the ability to quickly scale load balanced server farms may be difficult or impossible due to the stateful nature of the application. User session data is stored within the memory of the application, usually with ASP.NET session state or static variables that hold cross-request data that must be persisted. User session affinity is usually maintained through load balancer sticky sessions.

This proves to be challenging operationally. When increased capacity is required, you must intentionally provision and add servers. This can be a slow process. Taking nodes out of service in the event of patching or for unexpected failures can be problematic for the end user experience, losing state for all users associated with affected nodes. At best, this would require users to log in again.

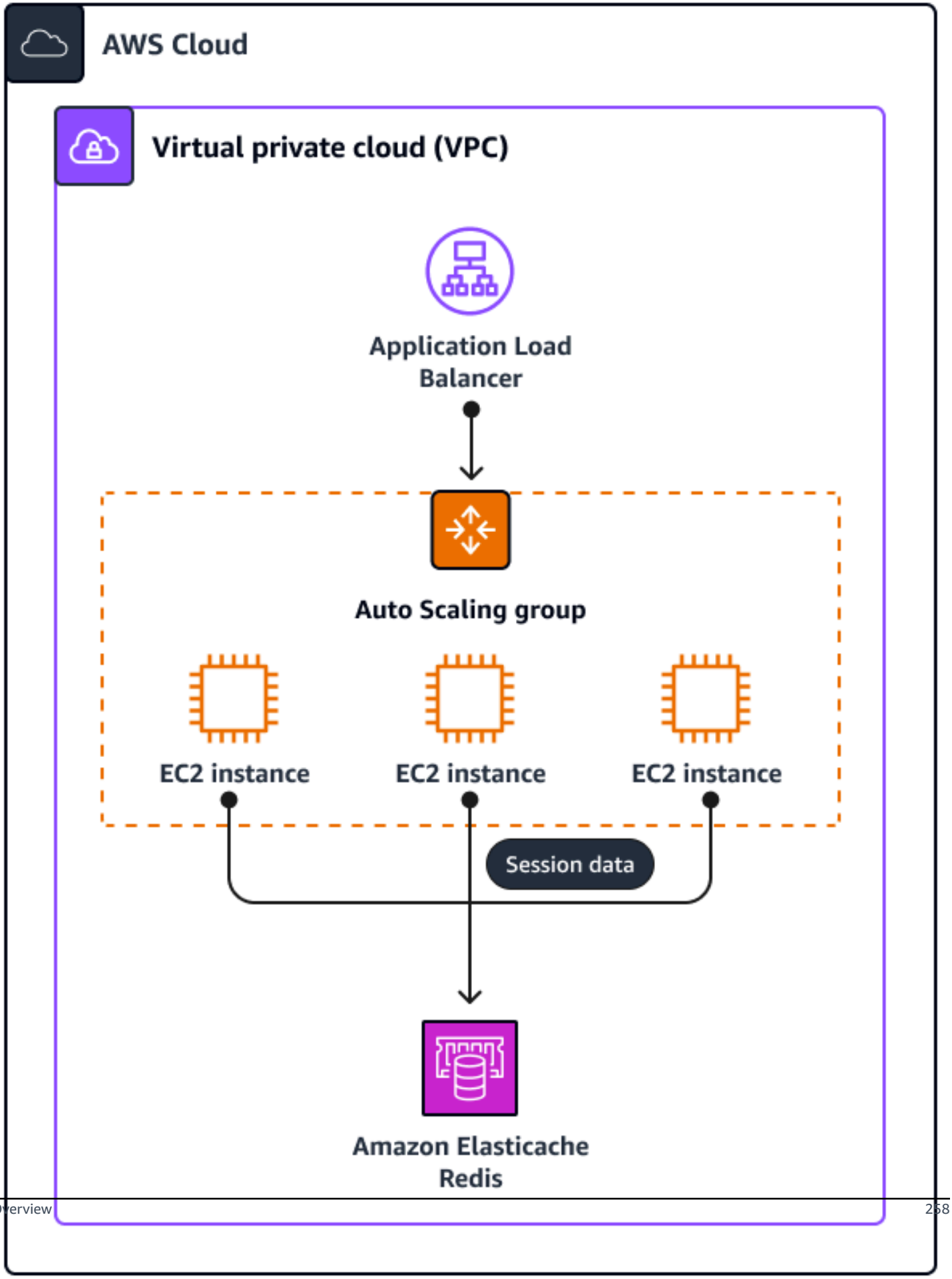
By centralizing session state for ASP.NET applications and applying autoscaling rules to legacy ASP.NET applications, you can take advantage of the elasticity of the cloud and potentially take advantage of cost savings when running applications. For example, you get cost reductions through compute scalability, but you also get to choose from different pricing models available, such as reducing reserved instance usage and using Amazon Spot Instance pricing.

Two common techniques include using Amazon DynamoDB as a session state provider and using Amazon ElastiCache (Redis OSS) as an ASP.NET session store.

The following diagram shows an architecture that uses DynamoDB as a session state provider.


AWS Cloud
Virtual private cloud (VPC)
Application Load Balancer
Auto Scaling group
EC2 instance
EC2 instance
EC2 instance
Session data
Amazon DynamoDB

The following diagram shows an architecture that uses ElastiCache (Redis OSS) as a session state provider.

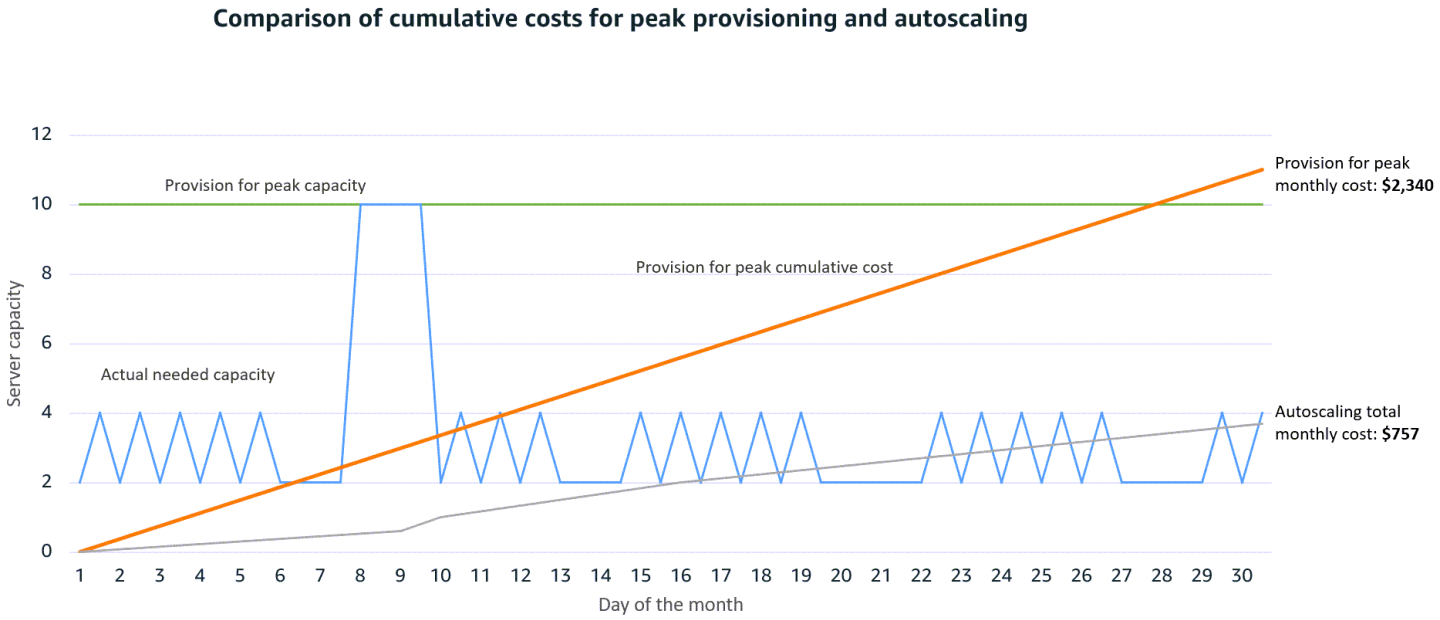

AWS Cloud
Virtual private cloud (VPC)
Application Load Balancer
Auto Scaling group
EC2 instance
EC2 instance
EC2 instance
Session data
Amazon Elasticache Redis

Cost impact

To determine the benefits of scaling for a production application, we recommend that you model your actual demand. This section makes the following assumptions to model a sample application:

- Instances that are added and removed from rotation are identical and no instance size variation is introduced.
- Server utilization never drops below two active servers in order to maintain high availability of the application.
- The quantity of servers scales linearly with the traffic (that is, twice as much traffic will require twice as much compute).
- Traffic is modeled over the course of a month in six hour increments, with intra-day variation and one abnormal peak of traffic (for example, a promotional sale) for one day of 10x traffic. Weekend traffic is modeled at base utilization.
- Nighttime traffic is modeled at base utilization, while weekday traffic is modeled at 4x utilization.
- Reserved Instance pricing uses one-year, no-upfront pricing. Normal daytime pricing uses on-demand pricing while burst demand uses Spot Instance pricing.

The following diagram illustrates how this model takes advantage of elasticity in a .NET application rather than provisioning for peak usage. This results in a savings of approximately 68 percent.



If you use DynamoDB as a session state storage mechanism, then use the following parameters:

```
Storage: 20GB
Session Reads: 40 million
Session Writes: 20 million
Pricing Model: On demand
```

The estimated monthly cost for this service is approximately $35.00 per month.

If you use ElastiCache (Redis OSS) as a session state storage mechanism, then use the following parameters:

```
Number of Nodes: 3
Node size: cache.t4g.medium
Pricing Model: 1y reserved
```

The estimated monthly cost for this service is approximately $91.00 per month.

Cost optimization recommendations

The first step is to implement session state in a legacy .NET application. If you're using ElastiCache as your state storage mechanism, follow the guidance from ElastiCache as an ASP.NET Session Store in the AWS Developer Tools Blog. If you're using DynamoDB, then follow the guidance from What is the AWS SDK for .NET in the SDK for .NET documentation.

If the application uses the **InProc** session to begin with, make sure that all objects that you plan to store in the session can be serialized. To do this, use the `SerializableAttribute` attribute to decorate classes whose instances will be stored in the session. For example:

```
[Serializable()]
public class TestSimpleObject {
public string SessionProperty {get;set;}
}
```

Additionally, the .NET `MachineKey` must be the same between all the servers in use. This is normally the case when instances are created from a common Amazon Machine Image (AMI). For example:

```
<machineKey
validationKey="some long hashed value"
decryptionKey="another long hashed value"
```

```
validation="SHA1"/>
```

However, it's important to ensure that if a base image is changed, it's configured with the same .NET machine image (either configurable at the IIS or server level). For more information, see SystemWebSectionGroup.MachineKey Property in the Microsoft documentation.

Finally, you must determine the mechanism for adding servers to an Auto Scaling group in response to a scaling event. There are several ways to accomplish this. We recommend the following methods to seamlessly deploy .NET Framework applications to an EC2 instance in an Auto Scaling group:

- Use EC2 Image Builder to configure an AMI that contains the fully configured server and application. You can then use this AMI to configure the launch template of your Auto Scaling group.
- Use AWS CodeDeploy to deploy your application. CodeDeploy enables integration directly with Amazon EC2 Auto Scaling. This provides an alternative to creating a new AMI for each release of the application.

Additional resources

- Create images with EC2 Image Builder (EC2 Image Builder documentation)
- Deploying .NET Web Applications Using AWS CodeDeploy with Visual Studio Team Services (AWS Developer Tools Blog)

Use caching to reduce database demand

Overview

You can use caching as an effective strategy to help reduce costs for your .NET applications. Many applications use backend databases, such as SQL Server, when applications require frequent access to data. The cost of maintaining these backend services to cope with demand can be high, but you can use an effective caching strategy to reduce load on backend databases by reducing sizing and scaling requirements. This can help you reduce costs and improve the performance of your applications.

Caching is a useful technique to save on costs related to read heavy workloads that use more expensive resources such as SQL Server. It's important to use the right technique for your workload.

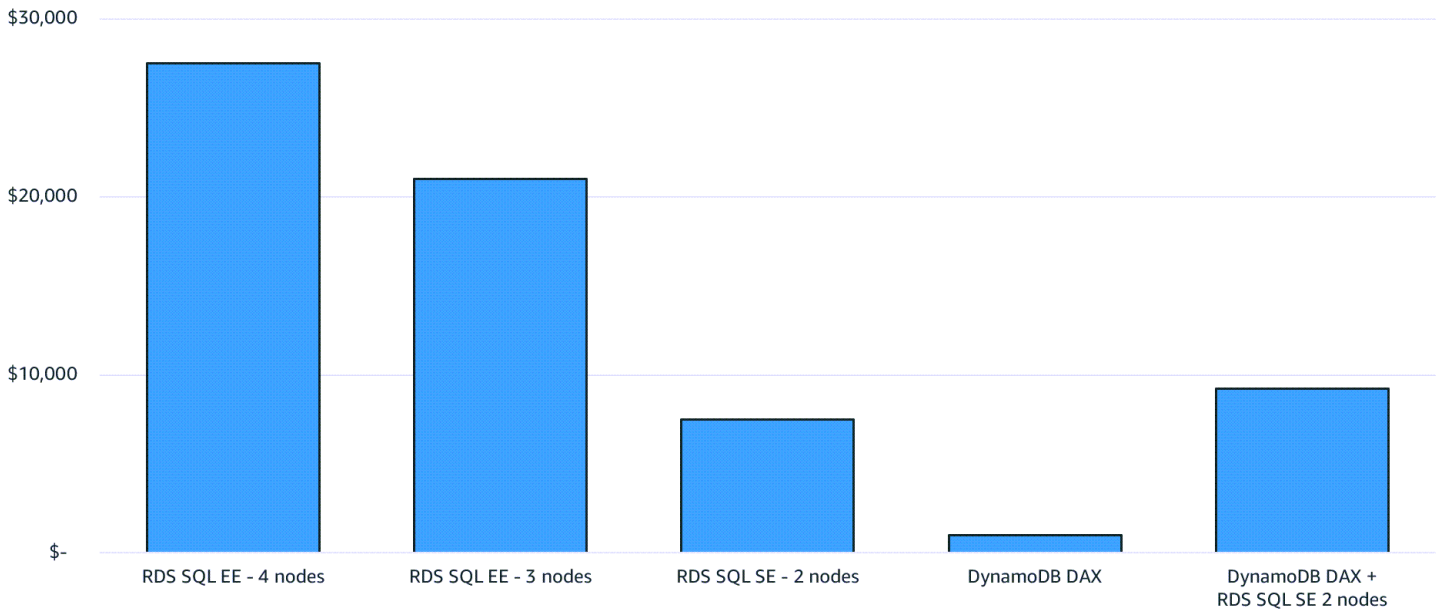
For instance, local caching isn't scalable and requires you to maintain a local cache for each instance of an application. You should weigh the performance impact compared to the potential costs, so that the lower cost of the underlying data source offsets any added costs related to the caching mechanism.

Cost impact

SQL Server requires you to factor in read requests when sizing your database. This could affect costs, since you may need to introduce read replicas to accommodate the load. If you're using read replicas, it's important to understand that they are only available on SQL Server Enterprise edition. This edition requires a more expensive license than SQL Server Standard edition.

The following diagram is designed to help you understand caching effectiveness. It shows Amazon RDS for SQL Server with four db.m4.2xlarge nodes running SQL Server Enterprise edition. It's deployed in a Multi-AZ configuration with one read replica. Exclusive read traffic (for example, SELECT queries) is directed to the read replicas. In comparison, Amazon DynamoDB uses an r4.2xlarge two node DynamoDB Accelerator (DAX) cluster.

The following chart shows the results of removing the need for dedicated read replicas that handle high read traffic.



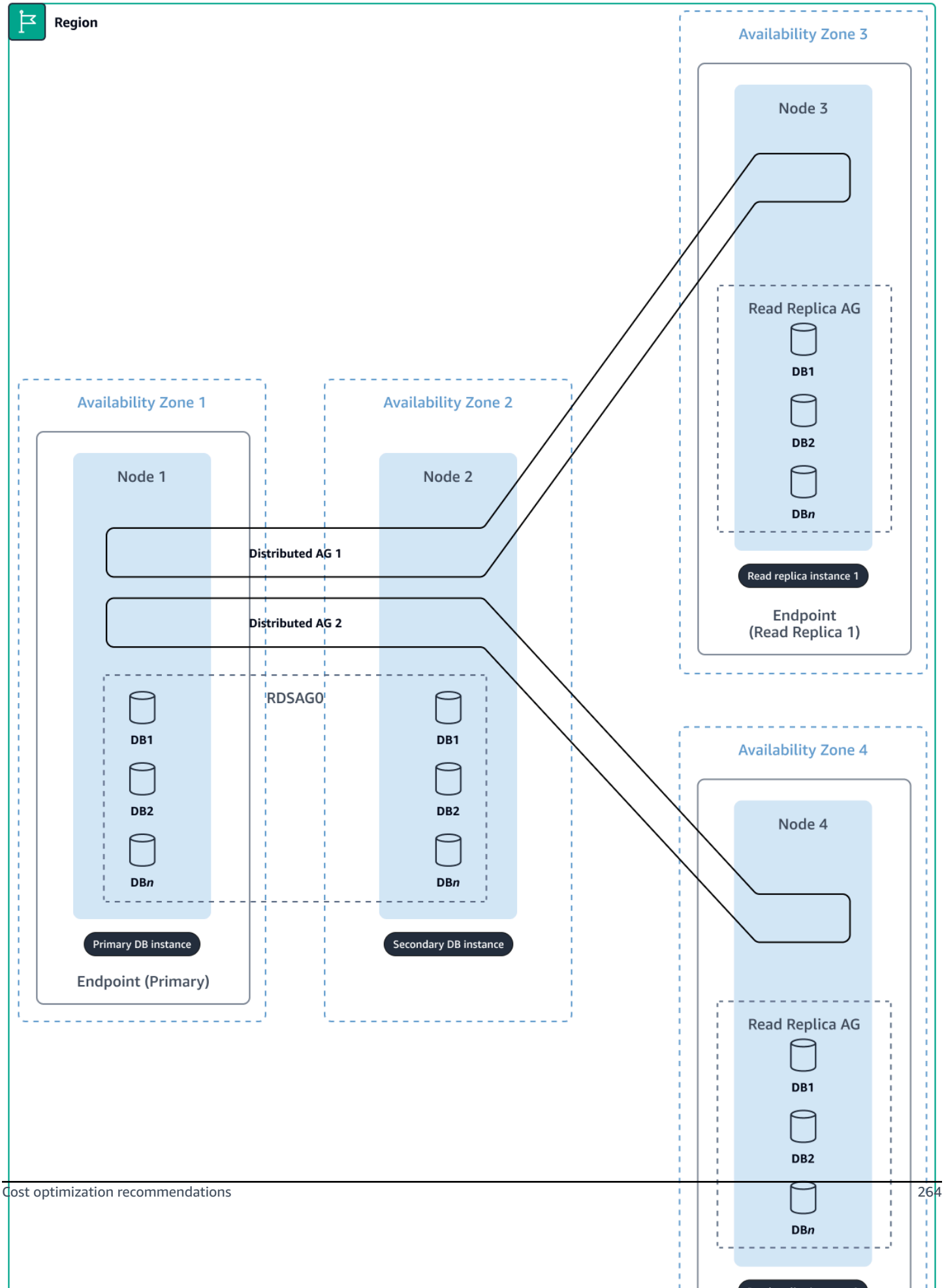
You can achieve significant cost savings by using local caching with no read replicas or by introducing DAX side by side with SQL Server on Amazon RDS as a caching layer. This layer offloads from SQL Server and reduces the size of the SQL Server required to run the database.

Cost optimization recommendations

Local caching

Local caching is one of the most commonly used ways to cache content for applications hosted both in on-premises environments or in the cloud. This is because it's relatively easy and intuitive to implement. Local caching involves taking content from a database or other source and either caching locally in memory or on disk for quicker access. This approach, although easy to implement, isn't ideal for some use cases. For example, this includes use cases when the caching content needs to persist over time, such as preserving application state or user state. Another use case is when cached content is required to be accessed from other application instances.

The diagram below illustrates a highly available SQL Server cluster with four nodes and two read replicas.


Region
Availability Zone 3
Node 3
Read Replica AG
DB1
DB2
DBn
Read replica instance 1
Endpoint
(Read Replica 1)
Availability Zone 1
Availability Zone 2
Node 1
Node 2
Distributed AG 1
Distributed AG 2
RDSAG0
DB1
DB2
DBn
DB1
DB2
DBn
Primary DB instance
Secondary DB instance
Endpoint (Primary)
Availability Zone 4
Node 4
Read Replica AG
DB1
DB2
DBn

With local caching, you might need to load balance traffic across multiple EC2 instances. Each instance must maintain its own local cache. If the cache stores stateful information, there needs to be regular commits to the database, and users may need to be forwarded to the same instance for each subsequent request (sticky session). This presents a challenge when trying to scale applications because some instances could be overutilized, while some are underutilized because of the uneven distribution of traffic.

You can use local caching, either in-memory or using local storage, for .NET applications. To do so, you can add functionality to either store objects on disk and retrieve them when required, or query data from the database and persist it in memory. To perform local caching in-memory and on local storage of data from a SQL Server in C#, for example, you can use a combination of `MemoryCache` and `LiteDB` libraries. `MemoryCache` provides in-memory caching, while `LiteDB` is an embedded NoSQL disk-based database that's fast and lightweight.

To perform in-memory caching, use the .NET Library `System.Runtime.MemoryCache`. The following code example shows how to use the `System.Runtime.Caching.MemoryCache` class to cache data in-memory. This class provides a way to temporarily store data in the application's memory. This can help improve the performance of an application by reducing the need to fetch data from a more expensive resource, like a database or an API.

Here's how the code works:

1. A private static instance of `MemoryCache` named `_memoryCache` is created. The cache is given a name (`dataCache`) to identify it. Then, the cache stores and retrieves the data.
2. The `GetData` method is a generic method that takes two arguments: a `string` key and a `Func<T>` delegate called `getData`. The key is used to identify the cached data, while the `getData` delegate represents the data retrieval logic that gets executed when the data isn't present in the cache.
3. The method first checks if the data is present in the cache using the `_memoryCache.Contains(key)` method. If the data is in the cache, the method retrieves the data by using `_memoryCache.Get(key)` and casts it to the expected type T.
4. If the data isn't in the cache, the method calls the `getData` delegate to fetch the data. Then, it adds the data to the cache by using `_memoryCache.Add(key, data, DateTimeOffset.Now.AddMinutes(10))`. This call specifies that the cache entry should expire after 10 minutes, at which point the data is removed from the cache automatically.
5. The `ClearCache` method takes a `string` key as an argument and removes the data associated with that key from the cache by using `_memoryCache.Remove(key)`.

```
using System;
using System.Runtime.Caching;

public class InMemoryCache
{
    private static MemoryCache _memoryCache = new MemoryCache("dataCache");

    public static T GetData<T>(string key, Func<T> getData)
    {
        if (_memoryCache.Contains(key))
        {
            return (T)_memoryCache.Get(key);
        }

        T data = getData();
        _memoryCache.Add(key, data, DateTimeOffset.Now.AddMinutes(10));

        return data;
    }

    public static void ClearCache(string key)
    {
        _memoryCache.Remove(key);
    }
}
```

You can use the following code:

```
public class Program
{
    public static void Main()
    {
        string cacheKey = "sample_data";

        Func<string> getSampleData = () =>
        {
            // Replace this with your data retrieval logic
            return "Sample data";
        };

        string data = InMemoryCache.GetData(cacheKey, getSampleData);
        Console.WriteLine("Data: " + data);
    }
```

```
}
```

The following example shows you how to use LiteDB to cache data in local storage. You can use LiteDB as an alternative or complement to in-memory caching. The following code demonstrates how to use the LiteDB library to cache data in local storage. The `LocalStorageCache` class contains the main functions for managing the cache.

```
using System;
using LiteDB;

public class LocalStorageCache
{
    private static string _liteDbPath = @"Filename=LocalCache.db";

    public static T GetData<T>(string key, Func<T> getData)
    {
        using (var db = new LiteDatabase(_liteDbPath))
        {
            var collection = db.GetCollection<T>("cache");
            var item = collection.FindOne(Query.EQ("_id", key));

            if (item != null)
            {
                return item;
            }
        }

        T data = getData();

        using (var db = new LiteDatabase(_liteDbPath))
        {
            var collection = db.GetCollection<T>("cache");
            collection.Upsert(new BsonValue(key), data);
        }

        return data;
    }

    public static void ClearCache(string key)
    {
        using (var db = new LiteDatabase(_liteDbPath))
        {
            var collection = db.GetCollection("cache");
```

```
            collection.Delete(key);
        }
    }
}

public class Program
{
    public static void Main()
    {
        string cacheKey = "sample_data";

        Func<string> getSampleData = () =>
        {
            // Replace this with your data retrieval logic
            return "Sample data";
        };

        string data = LocalStorageCache.GetData(cacheKey, getSampleData);
        Console.WriteLine("Data: " + data);
    }
}
```

If you have a static cache or static files that don't change frequently, you can also store these files in Amazon Simple Storage Service (Amazon S3) object storage. The application can retrieve the static cache file on startup to use locally. For more details on how to retrieve files from Amazon S3 by using .NET, see Downloading objects in the Amazon S3 documentation.

Caching with DAX

You can use a caching layer that can be shared across all application instances. DynamoDB Accelerator (DAX) is a fully managed, highly available in-memory cache for DynamoDB that can deliver a tenfold performance improvement. You can use DAX to reduce costs by reducing the need to overprovision read capacity units in DynamoDB tables. This is especially useful for workloads that are read heavy and require repeated reads for individual keys.

DynamoDB is priced on-demand or with provisioned capacity, so the number of reads and writes per month contributes to the cost. If you have read heavy workloads, DAX clusters can help lower costs by reducing the number of reads on your DynamoDB tables. For instructions on how to set up DAX, see In-memory acceleration with DynamoDB Accelerator (DAX) in the DynamoDB documentation. For information about .NET application integration, watch Integrating Amazon DynamoDB DAX into Your ASP.NET Application on YouTube.

Additional resources

- In-memory acceleration with DynamoDB Accelerator (DAX) - Amazon DynamoDB (DynamoDB documentation)
- Integrating Amazon DynamoDB DAX into Your ASP.NET Application (YouTube)
- Downloading objects (Amazon S3 documentation)

Consider serverless .NET

Overview

Serverless computing has become a popular approach for building and deploying applications. This is mainly because of the scalability and agility that the serverless approach offers when building a modern architecture. However, it's important to consider the cost impact of serverless computing in some scenarios.

Lambda is a serverless computing platform that enables developers to run code without the need for dedicated servers. Lambda is a particularly attractive option for .NET developers looking to reduce infrastructure costs. With Lambda, .NET developers can develop and deploy applications that are highly scalable and potentially cost-effective. By using a serverless approach, developers no longer provision servers to handle application requests. Instead, developers can create functions that are executed on-demand. This makes a serverless approach more scalable, manageable, and potentially more cost-effective than running, managing, and scaling virtual machines. As a result, you only pay for the resources used by the application, without having to worry about underutilized resources or server maintenance costs.

Developers can use modern, cross-platform .NET versions to build serverless applications that are fast, efficient, and cost-effective. The .NET Core and newer versions are a free and open-source framework that's better suited to running on serverless platforms over previous .NET Framework versions. This enables developers to reduce development time and increase application performance. Modern .NET also supports a range of programming languages, including C# and F#. For this reason, it's an attractive option for developers looking to build modern architectures in the cloud.

This section explains how you can achieve costs savings by using Lambda as a serverless option. You can further optimize costs by fine-tuning the execution profiles of your Lambda functions,

right sizing the memory allocation of your Lambda functions, using Native AOT, and moving to Graviton-based functions.

Cost impact

How much you can reduce costs depends on several factors, including how many executions your serverless functions will execute in addition to the amount of memory allocated and duration of each function. AWS Lambda offers a free tier, which includes one million free requests per month and 400,000 GB seconds of compute time per month. You can significantly reduce your monthly costs for workloads that are on or around these free tier limits.

There may also be additional costs when using a load balancer with Lambda functions as the target. This is calculated as the amount of data processed by the load balancer for the Lambda targets.

Cost optimization recommendations

Right size your Lambda functions

Right sizing is an essential practice for cost optimization in .NET-based Lambda functions. This process involves identifying the optimal memory configuration that balances performance with cost-effectiveness, without requiring changes to the code.

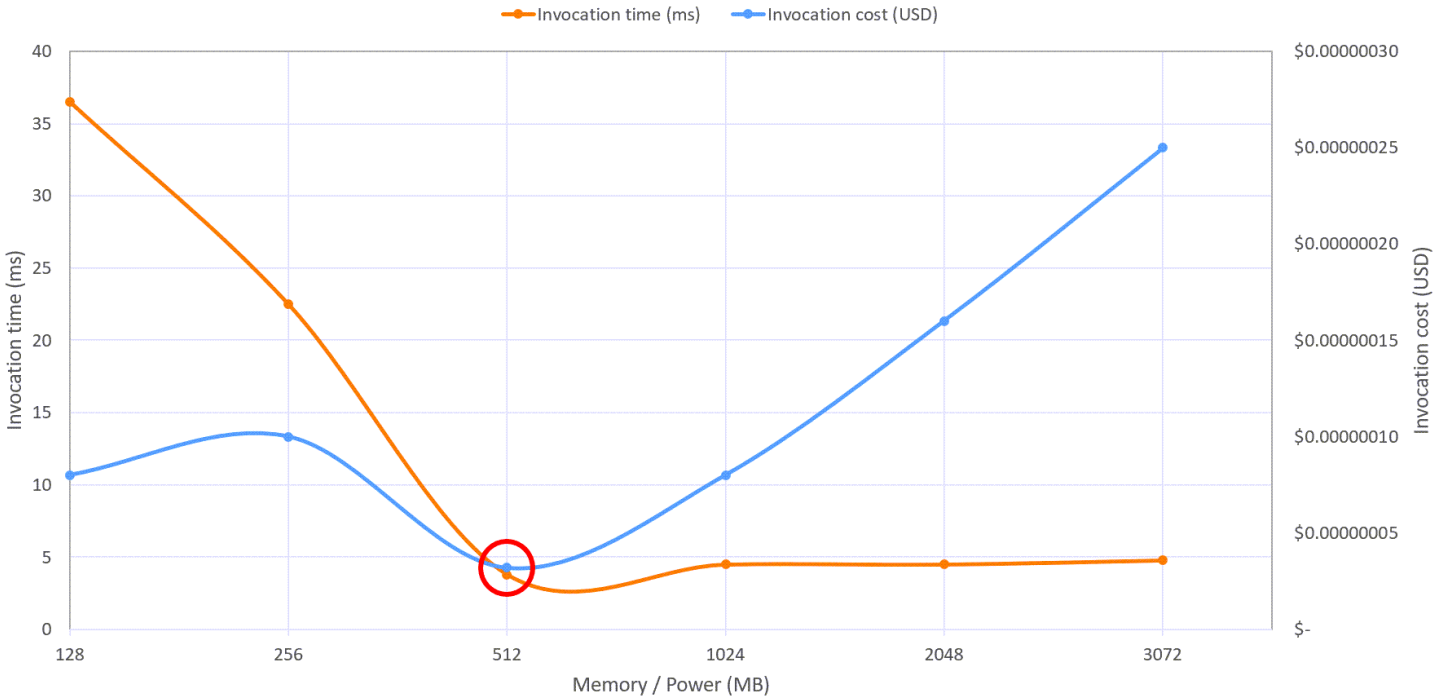
By configuring the memory for a Lambda function, ranging from 128 MB up to 10,240 MB, you also adjust the amount of vCPU available during invocation. This enables memory or CPU-bound applications to access additional resources during execution, leading to a potential reduction in invocation duration and overall cost.

However, identifying the optimal configuration for your .NET-based Lambda functions can be a manual and time-intensive process, especially if changes are frequent. The AWS Lambda Power Tuning tool can help you identify the appropriate configuration by analyzing a set of memory configurations against an example payload.

For example, increasing the memory for a .NET-based Lambda function can lead to improved total invocation time and reduced cost without affecting performance. The optimal memory configuration for a function can vary. The AWS Lambda Power Tuning tool can help identify the most cost-effective configuration for each function.

In the following example chart, the total invocation time improves as the memory increases for this Lambda function. This leads to a reduction in the cost for the total execution without affecting the

original performance of the function. For this function, the optimal memory configuration for the function is 512 MB, as this is where the resource utilization is most efficient for the total cost of each invocation. This varies per function, and using the tool on your Lambda functions can identify if they benefit from right sizing.



We recommend that you complete this exercise regularly, as part of any integration testing when new updates are released. If infrequently updated, then do this exercise periodically to ensure functions are tuned and right sized. After you have identified the appropriate memory setting for your Lambda functions, you can add right sizing to your processes. The AWS Lambda Power Tuning tool generates programmatic output that can be used by your CI/CD workflows during the release of new code. This enables you to automate memory configuration.

You can download the AWS Lambda Power Tuning tool for free. For instructions on how to use the tool, see How to execute the state machine in GitHub.

Lambda also supports native AOT, which enables .NET applications to be pre-compiled. This can help reduce costs by reducing execution times for .NET functions. For more information about creating Native AOT functions, see .NET functions with native AOT compilation in the Lambda documentation.

Avoid idle wait time

Lambda function duration is one dimension used for calculating billing. When function code makes a blocking call, you are billed for the time that it waits to receive a response. This wait time can grow when Lambda functions are chained together, or a function is acting as an orchestrator for other functions. If you have workflows such as batch operations or order delivery systems, this adds management overhead. Additionally, it may not be possible to complete all workflow logic and error handling within the maximum Lambda timeout of 15 minutes.

Instead of handling this logic in function code, we recommend that you re-architect your solution to use AWS Step Functions as an orchestrator of the workflow. When using a standard workflow, you are billed for each state transition within the workflow rather than the total duration of the workflow. In addition, you can move support for retries, wait conditions, error workflows, and callbacks into the state condition to allow your Lambda functions to focus on business logic. For more information, see Optimizing your AWS Lambda costs – Part 2 in the AWS Compute Blog.

Move to Graviton-based functions

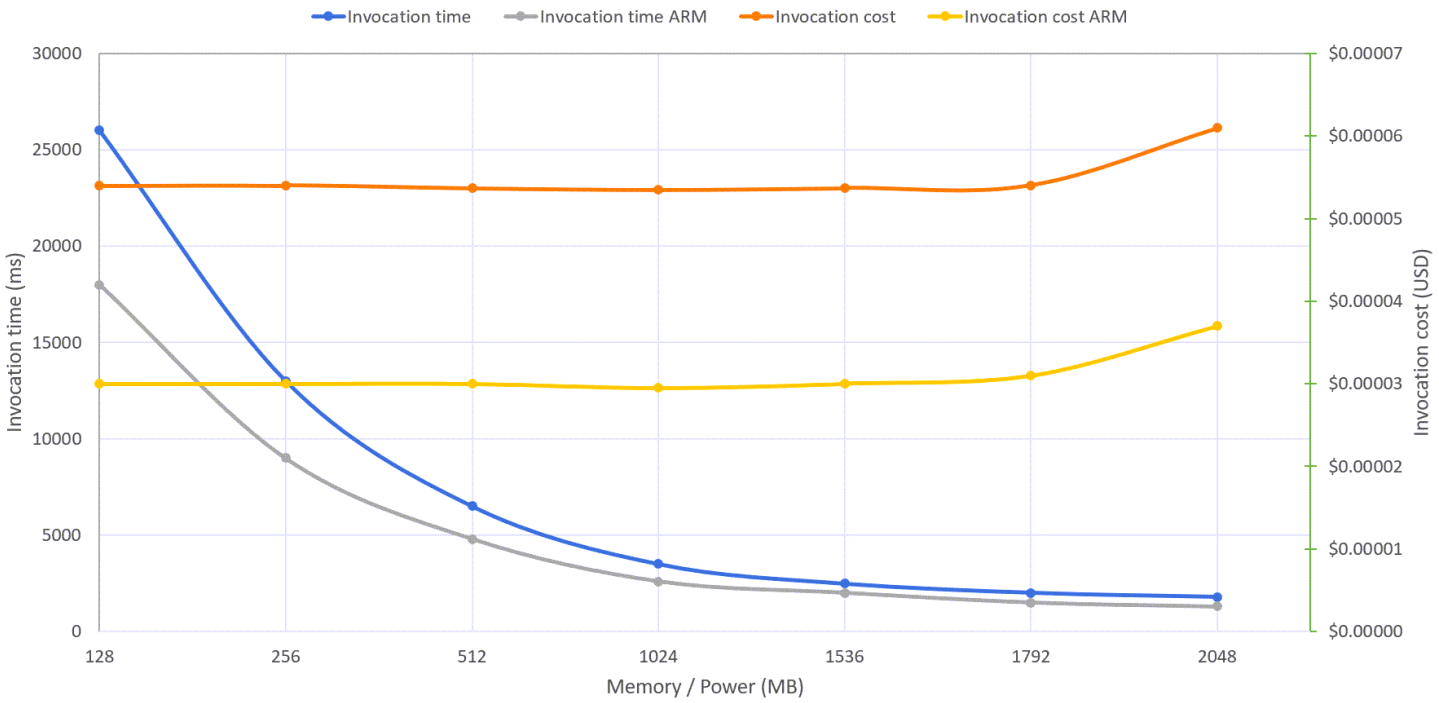
Lambda functions powered by next-generation Graviton2 processors are now generally available. Graviton2 functions, using an ARM-based processor architecture, are designed to deliver up to 19 percent better performance at a 20 percent lower cost for a variety of serverless workloads. With lower latency and better performance, functions powered by Graviton2 processors are ideal for powering mission-critical serverless applications.

Migrating to Graviton-based Lambda functions can be a cost-effective option for .NET developers looking to optimize their Lambda costs. Graviton-based functions use ARM-based processors instead of traditional x86 processors. This can lead to significant cost savings without sacrificing performance.

While there are several benefits to moving to Graviton-based functions, there are also several challenges and considerations that we recommend you consider. For example, Graviton-based functions require the use of Amazon Linux 2, which may not be compatible with all .NET applications. Additionally, there may be compatibility issues with third-party libraries or dependencies that aren't compatible with ARM-based processors.

If you're running .NET Framework applications and want to take advantage of serverless with Lambda, you can consider porting the applications to modern .NET by using the Porting Assistant for .NET. This can help you to accelerate the porting of legacy .NET applications to modern .NET, enabling the application to run on Linux.

The following chart compares x86 and ARM/Graviton2 architecture results for a function that computes prime numbers.



The function is using a single thread. The lowest duration for both architectures is reported when memory is configured with 1.8 GB. Above that, Lambda functions have access to more than 1 vCPU, but in this case, the function can't use the additional power. For the same reason, costs are stable with memory up to 1.8 GB. With more memory, costs increase because there are no additional performance benefits for this workload. The Graviton2 processor is clearly providing better performance and lower costs for this compute-intensive function.

To configure your function to use and ARM-based processor with Graviton, do the following:

1. Sign in to the AWS Management Console and then open the Lambda console.
2. Choose **Create function**.
3. For **Function name**, enter a name.
4. For **Runtime**, choose **.NET 6 (C#/PowerShell)**.
5. For **Architecture**, select **arm64**.
6. Make any additional configurations that you require, and then choose **Create function**.

Additional resources

- Lambda functions as targets (AWS documentation)
- Optimizing AWS Lambda cost and performance using AWS Compute Optimizer (AWS Compute Blog)
- Optimizing your AWS Lambda costs – Part 1 (AWS Compute Blog)
- Optimizing your AWS Lambda costs – Part 2 (AWS Compute Blog)
- Building serverless .NET applications on AWS Lambda using .NET 7 (AWS Compute Blog)

Consider purpose-built databases

Overview

One of the costliest aspects of running Microsoft-based workloads comes from the licensing of commercial databases, such as SQL Server. Businesses often standardize on SQL Server as the database platform of choice and it becomes ingrained in the development culture of the organization. Developers generally choose a relational SQL Server-based model regardless of the use case. Reasons include:

- The business already has SQL Server instances and/or licenses available.
- Teams have habituated to the SQL programming model through the use of shared libraries, ORMs, and business logic.
- Management is not aware of alternatives.
- Developers are not aware of alternatives.

Purpose-built databases can accommodate the data access patterns of your use case. These databases are increasingly adopted by businesses as they adopt more modern architectures (such as microservices) and as the scope of individual applications narrow.

A database being purpose-built does not preclude a relational model, or require a NoSQL (non-relational) model. In fact, a relational database is considered purpose-built when selected in response to a workload's specific needs. The use of purpose-built databases can help teams to reduce the database costs associated with their .NET applications, while also gaining standard cloud benefits, such as scalability, resilience, and the reduction of undifferentiated heavy lifting.

The following table shows the purpose-built databases offered by AWS.

Database	Type	Characteristics
Amazon Aurora PostgreSQL or Amazon Aurora MySQL	Relational	Use cases where data has a fixed structure Relational databases naturally maintain data consistency through ACID transactions
Amazon DynamoDB	Key-value pair	NoSQL database that stores data using a hash table data structure High performance storage and retrieval of unstructured data Use cases include user profiles, session state, and shopping cart data
Amazon ElastiCache	In-memory	High performance NoSQL database that stores unstructured data in memory with sub-millisecond access time Used for frequently accessed, ephemeral data such as user sessions, and as a caching layer in front of other, slower, data stores Includes support for both ElastiCache (Redis OSS) and ElastiCache (Memcached)

Database	Type	Characteristics
Amazon MemoryDB	Durable in-memory	Redis compatible purpose-built database with durable storage
Amazon Timestream	Time series	Database designed for high throughput data ingestion in temporal order Use cases include Internet of Things (IoT) applications and storing metrics or telemetry data
Amazon DocumentDB	Document	NoSQL database that stores data without a prescribed structure or enforced relationships to other data Often used for read-intensive workloads such as product catalogs
Amazon Neptune	Graph	NoSQL database that holds both data and a representation of the connections among data items Use cases include fraud detection, recommendation engines, and social applications

Database	Type	Characteristics
Amazon Keyspaces	Wide-column	High performance distributed database based on Apache Cassandra Use cases include IoT applications, event processing, and game applications

A significant driver of purpose-built database adoption can be attributed to the elimination of commercial licensing. However, the auto-scaling capability of databases such as DynamoDB (including on-demand mode), Aurora, Amazon Neptune, and Amazon Keyspaces enable you to provision capacity for the average case, rather than for peak usage. Purpose-built databases, such as Timestream, are serverless and automatically scale to meet demand without any pre-provisioning.

AWS offers Babelfish for Aurora PostgreSQL if you want to use a purpose-built, open-source compatible relational database, but are unable or unwilling to make significant code changes to your application. In some cases, Babelfish allows you to use an existing SQL Server access code, with almost no changes.

When choosing a purpose-built relational database for applications, it's important to retain the same (or functionally equivalent) features that you require for your applications. This recommendation addresses purpose-built databases as a primary data store for applications. Specific applications (such as caching) are addressed in other recommendations.

Cost impact

Adopting purpose-built databases for .NET workloads, while unlikely to affect compute consumption/cost directly, can directly influence the cost of the database services consumed by the .NET applications. In fact, cost savings may be a secondary goal, when compared with the added benefits of agility, scalability, resilience, and data durability.

It's outside the scope of this guide to explain the full process of choosing a purpose-built database for applications and rearchitecting a data strategy to use them effectively. For more information, see Purpose-built databases in the AWS Tutorials Directory.

The following tables show several examples of how the replacement of SQL Server with a purpose-built database can alter application costs. Note that these are simply rough estimates. Benchmarks and optimization of actual workloads is required to calculate the exact production cost.

These are some commonly used purpose-built database estimates that include on-demand compute and 100 GB SSD, single instance databases in `us-east-1`. License costs include SQL Server license plus software assurance.

The following table shows estimated costs for commercial database examples.

Database engine	Licensing model	Instance type/specs	AWS compute + storage cost	License cost	Total monthly cost
SQL Server Standard edition on Amazon EC2	License included	r6i.2xlarge (8 CPU/64 GB RAM)	$1,345.36	$0.00	$1,345.36
SQL Server Enterprise edition on Amazon EC2	License included	r6i.2xlarge (8 CPU/64 GB RAM)	$2,834.56	$0.00	$2,834.56
SQL Server Standard edition on Amazon EC2	BYOL	r6i.2xlarge (8 CPU/64 GB RAM)	$644.56	$456.00	$1,100.56
SQL Server Enterprise edition on Amazon EC2	BYOL	r6i.2xlarge (8 CPU/64 GB RAM)	$644.56	$1,750.00	$2,394.56
SQL Server Standard edition on Amazon RDS		db.r6i.2x large (8 CPU/64 GB RAM)	$2,318.30	$0.00	$2,318.30

Database engine	Licensing model	Instance type/specs	AWS compute + storage cost	License cost	Total monthly cost
SQL Server Enterprise edition on Amazon RDS		db.r6i.2x large (8 CPU/64 GB RAM)	$3,750.56	$0.00	$3,750.56

The following table shows estimated costs for purpose-built examples.

Database engine	Instance type/ specs	AWS compute + storage cost	License cost	Total monthly cost
Amazon Aurora PostgreSQL	r6g.2xlarge (8 CPU/64 GB RAM)	$855.87	$0.00	$855.87
DynamoDB	Provisioned base 100 WCU/400 RCU	$72.00		$72.00
Amazon DocumentDB	db.r6i.2xlarge (8 CPU/64 GB RAM)	$778.60		$778.60

Important

The table is based on estimated licensing costs for SQL Server with Software Assurance, during the first three years of purchase. For SQL Server Standard edition: $4,100, 2 core pack, 3 years. For SQL Server Enterprise edition: $15,700, 2 core pack, 3 years.

We recommend that you consider the cost implications before you adopt purpose-built databases. For example, the cost to update applications to use a purpose-built database is related to the complexity of the application and the source database. Be sure to factor in the total cost of

ownership when planning this architecture switch. This includes refactoring your applications, upskilling staff on new technologies, and carefully planning the performance and consumption anticipated for each workload. From there, you can make the determination if the investment is worth the cost savings. In most cases, maintaining an end-of-support product is a security and compliance risk, and the cost of remediating it is worth the effort and initial investment.

Cost optimization recommendations

For .NET applications that are accessing SQL Server, there are replacement libraries for purpose-built relational databases. You can implement these libraries in your application to replace similar SQL Server application functionality.

The following table highlights some libraries that can be used in many common scenarios.

Library	Database	Replacement for	Framework compatibility
Npgsql Entity Framework Core Provider	Amazon Aurora PostgreSQL	Entity Framework Core SQL Server Provider	Modern .NET
Npgsql Entity Framework 6 Provider	Amazon Aurora PostgreSQL	Entity Framework 6.0 SQL Server Provider	.NET Framework
Npgsql (ADO.NET compatible PostgreSQL library)	Amazon Aurora PostgreSQL	ADO.NET	.NET Framework/ Modern .NET
MySQL Entity Framework Core Provider	Amazon Aurora MySQL	Entity Framework Core SQL Server Provider	Modern .NET
Pomelo.EntityFrameworkCore.MySql	Amazon Aurora MySQL	Entity Framework Core SQL Server Provider	Modern .NET

Connecting to Amazon Aurora PostgreSQL by using Babelfish doesn't require any special coding to connect. However, all code should be thoroughly tested prior to use.

Other purpose-built databases have libraries for accessing .NET compatible libraries that enable you to access purpose built databases. Examples include:

- Using Amazon DynamoDB NoSQL databases (AWS SDK for .NET documentation)
- MongoDB C# Driver (MongoDB documentation)
- .NET (Timestream documentation)
- Using a Cassandra .NET Core client driver to access Amazon Keyspaces programmatically (Amazon Keyspaces documentation)
- Using .NET to connect to a Neptune DB instance (Neptune documentation)

If you migrate to purpose built-databases, you can use these tools from AWS to help with the migration process:

- AWS Schema Conversion Tool (AWS SCT) can help you transform SQL Server schemas to Amazon Aurora and Amazon DynamoDB.
- AWS Database Migration Service (AWS DMS) can help you migrate data, either one time or on an ongoing basis, from SQL Server to Aurora or DynamoDB.
- Babelfish Compass can help you check the compatibility of your SQL Server database for use with Babelfish for Aurora PostgreSQL.

Additional resources

- Guidance for migrating SQL Server to Amazon Aurora PostgreSQL (AWS Database Blog)
- Babelfish APP Modernization Immersion Day (AWS Workshop Studio)
- .NET Immersion Day (AWS Workshop Studio)
- Getting started with Amazon Timestream with .NET (GitHub)
- Purpose-built databases for modern .NET applications on AWS (AWS presentation)

Next steps

After you finish reviewing this guide, we recommend that you take the following next steps to implement MACO:

1. **Reach out to a MACO expert.** A MACO expert can help answer your questions and address your concerns. If you're already working with an AWS account team, reach out to the team and request help from a MACO expert. If you don't have an account team, contact optimize-microsoft@amazon.com.
2. **Apply the recommendations.** Apply the recommendations, best practices, and strategies that you learned in this guide and from speaking with a MACO expert.
3. **Track costs changes.** Tag your workloads and use services like AWS Cost Explorer and AWS Budgets for detailed cost tracking, monitoring, and control.

Document history

The following table describes significant changes to this guide. If you want to be notified about future updates, you can subscribe to an RSS feed.

Change	Description	Date
SQL Server updates	We updated the Optimize CPU for SQL Server workloads section to add more informati on about the Optimize CPU feature for Amazon EC2 instances.	October 22, 2025
SQL Server updates	We updated the Optimize CPU for SQL Server workloads section.	October 25, 2024
SQL Server and Container updates	We added the Optimize SQL Server sizing by using Compute Optimizer, Review Trusted Advisor recommend ations for SQL Server workloads, and Replatform Windows applications with App2Container sections.	June 29, 2024
SQL Server licensing optimization	We added the Optimize SQL Server licensing by using Compute Optimizer section.	May 22, 2024
Initial publication	—	December 21, 2023

AWS Prescriptive Guidance glossary

The following are commonly used terms in strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the **Provide feedback** link at the end of the glossary.

Numbers

7 Rs

Seven common migration strategies for moving applications to the cloud. These strategies build upon the 5 Rs that Gartner identified in 2011 and consist of the following:

- Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.
- Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.
- Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.
- Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.
- Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. You migrate servers from an on-premises platform to a cloud service for the same platform. Example: Migrate a Microsoft Hyper-V application to AWS.
- Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later time, and legacy applications that you want to retain, because there's no business justification for migrating them.

- Retire – Decommission or remove applications that are no longer needed in your source environment.

A

ABAC

See attribute-based access control.

abstracted services

See managed services.

ACID

See atomicity, consistency, isolation, durability.

active-active migration

A database migration method in which the source and target databases are kept in sync (by using a bidirectional replication tool or dual write operations), and both databases handle transactions from connecting applications during migration. This method supports migration in small, controlled batches instead of requiring a one-time cutover. It's more flexible but requires more work than active-passive migration.

active-passive migration

A database migration method in which the source and target databases are kept in sync, but only the source database handles transactions from connecting applications while data is replicated to the target database. The target database doesn't accept any transactions during migration.

aggregate function

A SQL function that operates on a group of rows and calculates a single return value for the group. Examples of aggregate functions include `SUM` and `MAX`.

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

The process of permanently deleting personal information in a dataset. Anonymization can help protect personal privacy. Anonymized data is no longer considered to be personal data.

anti-pattern

A frequently used solution for a recurring issue where the solution is counter-productive, ineffective, or less effective than an alternative.

application control

A security approach that allows the use of only approved applications in order to help protect a system from malware.

application portfolio

A collection of detailed information about each application used by an organization, including the cost to build and maintain the application, and its business value. This information is key to the portfolio discovery and analysis process and helps identify and prioritize the applications to be migrated, modernized, and optimized.

artificial intelligence (AI)

The field of computer science that is dedicated to using computing technologies to perform cognitive functions that are typically associated with humans, such as learning, solving problems, and recognizing patterns. For more information, see What is Artificial Intelligence?

artificial intelligence operations (AIOps)

The process of using machine learning techniques to solve operational problems, reduce operational incidents and human intervention, and increase service quality. For more information about how AIOps is used in the AWS migration strategy, see the operations integration guide.

asymmetric encryption

An encryption algorithm that uses a pair of keys, a public key for encryption and a private key for decryption. You can share the public key because it isn't used for decryption, but access to the private key should be highly restricted.

atomicity, consistency, isolation, durability (ACID)

A set of software properties that guarantee the data validity and operational reliability of a database, even in the case of errors, power failures, or other problems.

attribute-based access control (ABAC)

The practice of creating fine-grained permissions based on user attributes, such as department, job role, and team name. For more information, see ABAC for AWS in the AWS Identity and Access Management (IAM) documentation.

authoritative data source

A location where you store the primary version of data, which is considered to be the most reliable source of information. You can copy data from the authoritative data source to other locations for the purposes of processing or modifying the data, such as anonymizing, redacting, or pseudonymizing it.

Availability Zone

A distinct location within an AWS Region that is insulated from failures in other Availability Zones and provides inexpensive, low-latency network connectivity to other Availability Zones in the same Region.

AWS Cloud Adoption Framework (AWS CAF)

A framework of guidelines and best practices from AWS to help organizations develop an efficient and effective plan to move successfully to the cloud. AWS CAF organizes guidance into six focus areas called perspectives: business, people, governance, platform, security, and operations. The business, people, and governance perspectives focus on business skills and processes; the platform, security, and operations perspectives focus on technical skills and processes. For example, the people perspective targets stakeholders who handle human resources (HR), staffing functions, and people management. For this perspective, AWS CAF provides guidance for people development, training, and communications to help ready the organization for successful cloud adoption. For more information, see the AWS CAF website and the AWS CAF whitepaper.

AWS Workload Qualification Framework (AWS WQF)

A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.

B

bad bot

A bot that is intended to disrupt or cause harm to individuals or organizations.

BCP

See business continuity planning.

behavior graph

A unified, interactive view of resource behavior and interactions over time. You can use a behavior graph with Amazon Detective to examine failed logon attempts, suspicious API calls, and similar actions. For more information, see Data in a behavior graph in the Detective documentation.

big-endian system

A system that stores the most significant byte first. See also endianness.

binary classification

A process that predicts a binary outcome (one of two possible classes). For example, your ML model might need to predict problems such as “Is this email spam or not spam?" or "Is this product a book or a car?"

bloom filter

A probabilistic, memory-efficient data structure that is used to test whether an element is a member of a set.

blue/green deployment

A deployment strategy where you create two separate but identical environments. You run the current application version in one environment (blue) and the new application version in the other environment (green). This strategy helps you quickly roll back with minimal impact.

bot

A software application that runs automated tasks over the internet and simulates human activity or interaction. Some bots are useful or beneficial, such as web crawlers that index information on the internet. Some other bots, known as *bad bots*, are intended to disrupt or cause harm to individuals or organizations.

botnet

Networks of bots that are infected by malware and are under the control of a single party, known as a *bot herder* or *bot operator*. Botnets are the best-known mechanism to scale bots and their impact.

branch

A contained area of a code repository. The first branch created in a repository is the *main branch*. You can create a new branch from an existing branch, and you can then develop features or fix bugs in the new branch. A branch you create to build a feature is commonly referred to as a *feature branch*. When the feature is ready for release, you merge the feature branch back into the main branch. For more information, see About branches (GitHub documentation).

break-glass access

In exceptional circumstances and through an approved process, a quick means for a user to gain access to an AWS account that they don't typically have permissions to access. For more information, see the Implement break-glass procedures indicator in the AWS Well-Architected guidance.

brownfield strategy

The existing infrastructure in your environment. When adopting a brownfield strategy for a system architecture, you design the architecture around the constraints of the current systems and infrastructure. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

buffer cache

The memory area where the most frequently accessed data is stored.

business capability

What a business does to generate value (for example, sales, customer service, or marketing). Microservices architectures and development decisions can be driven by business capabilities. For more information, see the Organized around business capabilities section of the Running containerized microservices on AWS whitepaper.

business continuity planning (BCP)

A plan that addresses the potential impact of a disruptive event, such as a large-scale migration, on operations and enables a business to resume operations quickly.

C

CAF

See AWS Cloud Adoption Framework.

canary deployment

The slow and incremental release of a version to end users. When you are confident, you deploy the new version and replace the current version in its entirety.

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

The process of tracking changes to a data source, such as a database table, and recording metadata about the change. You can use CDC for various purposes, such as auditing or replicating changes in a target system to maintain synchronization.

chaos engineering

Intentionally introducing failures or disruptive events to test a system's resilience. You can use AWS Fault Injection Service (AWS FIS) to perform experiments that stress your AWS workloads and evaluate their response.

CI/CD

See continuous integration and continuous delivery.

classification

A categorization process that helps generate predictions. ML models for classification problems predict a discrete value. Discrete values are always distinct from one another. For example, a model might need to evaluate whether or not there is a car in an image.

client-side encryption

Encryption of data locally, before the target AWS service receives it.

Cloud Center of Excellence (CCoE)

A multi-disciplinary team that drives cloud adoption efforts across an organization, including developing cloud best practices, mobilizing resources, establishing migration timelines, and leading the organization through large-scale transformations. For more information, see the CCoE posts on the AWS Cloud Enterprise Strategy Blog.

cloud computing

The cloud technology that is typically used for remote data storage and IoT device management. Cloud computing is commonly connected to edge computing technology.

cloud operating model

In an IT organization, the operating model that is used to build, mature, and optimize one or more cloud environments. For more information, see Building your Cloud Operating Model.

cloud stages of adoption

The four phases that organizations typically go through when they migrate to the AWS Cloud:

- Project – Running a few cloud-related projects for proof of concept and learning purposes
- Foundation – Making foundational investments to scale your cloud adoption (e.g., creating a landing zone, defining a CCoE, establishing an operations model)
- Migration – Migrating individual applications
- Re-invention – Optimizing products and services, and innovating in the cloud

These stages were defined by Stephen Orban in the blog post The Journey Toward Cloud-First & the Stages of Adoption on the AWS Cloud Enterprise Strategy blog. For information about how they relate to the AWS migration strategy, see the migration readiness guide.

CMDB

See configuration management database.

code repository

A location where source code and other assets, such as documentation, samples, and scripts, are stored and updated through version control processes. Common cloud repositories include GitHub or Bitbucket Cloud. Each version of the code is called a *branch*. In a microservice structure, each repository is devoted to a single piece of functionality. A single CI/CD pipeline can use multiple repositories.

cold cache

A buffer cache that is empty, not well populated, or contains stale or irrelevant data. This affects performance because the database instance must read from the main memory or disk, which is slower than reading from the buffer cache.

cold data

Data that is rarely accessed and is typically historical. When querying this kind of data, slow queries are typically acceptable. Moving this data to lower-performing and less expensive storage tiers or classes can reduce costs.

computer vision (CV)

A field of AI that uses machine learning to analyze and extract information from visual formats such as digital images and videos. For example, Amazon SageMaker AI provides image processing algorithms for CV.

configuration drift

For a workload, a configuration change from the expected state. It might cause the workload to become noncompliant, and it's typically gradual and unintentional.

configuration management database (CMDB)

A repository that stores and manages information about a database and its IT environment, including both hardware and software components and their configurations. You typically use data from a CMDB in the portfolio discovery and analysis stage of migration.

conformance pack

A collection of AWS Config rules and remediation actions that you can assemble to customize your compliance and security checks. You can deploy a conformance pack as a single entity in an AWS account and Region, or across an organization, by using a YAML template. For more information, see Conformance packs in the AWS Config documentation.

continuous integration and continuous delivery (CI/CD)

The process of automating the source, build, test, staging, and production stages of the software release process. CI/CD is commonly described as a pipeline. CI/CD can help you automate processes, improve productivity, improve code quality, and deliver faster. For more information, see Benefits of continuous delivery. CD can also stand for *continuous deployment*. For more information, see Continuous Delivery vs. Continuous Deployment.

CV

See computer vision.

D

data at rest

Data that is stationary in your network, such as data that is in storage.

data classification

A process for identifying and categorizing the data in your network based on its criticality and sensitivity. It is a critical component of any cybersecurity risk management strategy because it helps you determine the appropriate protection and retention controls for the data. Data classification is a component of the security pillar in the AWS Well-Architected Framework. For more information, see Data classification.

data drift

A meaningful variation between the production data and the data that was used to train an ML model, or a meaningful change in the input data over time. Data drift can reduce the overall quality, accuracy, and fairness in ML model predictions.

data in transit

Data that is actively moving through your network, such as between network resources.

data mesh

An architectural framework that provides distributed, decentralized data ownership with centralized management and governance.

data minimization

The principle of collecting and processing only the data that is strictly necessary. Practicing data minimization in the AWS Cloud can reduce privacy risks, costs, and your analytics carbon footprint.

data perimeter

A set of preventive guardrails in your AWS environment that help make sure that only trusted identities are accessing trusted resources from expected networks. For more information, see Building a data perimeter on AWS.

data preprocessing

To transform raw data into a format that is easily parsed by your ML model. Preprocessing data can mean removing certain columns or rows and addressing missing, inconsistent, or duplicate values.

data provenance

The process of tracking the origin and history of data throughout its lifecycle, such as how the data was generated, transmitted, and stored.

data subject

An individual whose data is being collected and processed.

data warehouse

A data management system that supports business intelligence, such as analytics. Data warehouses commonly contain large amounts of historical data, and they are typically used for queries and analysis.

database definition language (DDL)

Statements or commands for creating or modifying the structure of tables and objects in a database.

database manipulation language (DML)

Statements or commands for modifying (inserting, updating, and deleting) information in a database.

DDL

See database definition language.

deep ensemble

To combine multiple deep learning models for prediction. You can use deep ensembles to obtain a more accurate prediction or for estimating uncertainty in predictions.

deep learning

An ML subfield that uses multiple layers of artificial neural networks to identify mapping between input data and target variables of interest.

defense-in-depth

An information security approach in which a series of security mechanisms and controls are thoughtfully layered throughout a computer network to protect the confidentiality, integrity, and availability of the network and the data within. When you adopt this strategy on AWS, you add multiple controls at different layers of the AWS Organizations structure to help secure resources. For example, a defense-in-depth approach might combine multi-factor authentication, network segmentation, and encryption.

delegated administrator

In AWS Organizations, a compatible service can register an AWS member account to administer the organization's accounts and manage permissions for that service. This account is called the *delegated administrator* for that service. For more information and a list of compatible services, see Services that work with AWS Organizations in the AWS Organizations documentation.

deployment

The process of making an application, new features, or code fixes available in the target environment. Deployment involves implementing changes in a code base and then building and running that code base in the application's environments.

development environment

See environment.

detective control

A security control that is designed to detect, log, and alert after an event has occurred. These controls are a second line of defense, alerting you to security events that bypassed the preventative controls in place. For more information, see Detective controls in *Implementing security controls on AWS*.

development value stream mapping (DVSM)

A process used to identify and prioritize constraints that adversely affect speed and quality in a software development lifecycle. DVSM extends the value stream mapping process originally designed for lean manufacturing practices. It focuses on the steps and teams required to create and move value through the software development process.

digital twin

A virtual representation of a real-world system, such as a building, factory, industrial equipment, or production line. Digital twins support predictive maintenance, remote monitoring, and production optimization.

dimension table

In a star schema, a smaller table that contains data attributes about quantitative data in a fact table. Dimension table attributes are typically text fields or discrete numbers that behave like text. These attributes are commonly used for query constraining, filtering, and result set labeling.

disaster

An event that prevents a workload or system from fulfilling its business objectives in its primary deployed location. These events can be natural disasters, technical failures, or the result of human actions, such as unintentional misconfiguration or a malware attack.

disaster recovery (DR)

The strategy and process you use to minimize downtime and data loss caused by a disaster. For more information, see Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

See database manipulation language.

domain-driven design

An approach to developing a complex software system by connecting its components to evolving domains, or core business goals, that each component serves. This concept was introduced by Eric Evans in his book, *Domain-Driven Design: Tackling Complexity in the Heart of Software* (Boston: Addison-Wesley Professional, 2003). For information about how you can use domain-driven design with the strangler fig pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

DR

See disaster recovery.

drift detection

Tracking deviations from a baselined configuration. For example, you can use AWS CloudFormation to detect drift in system resources, or you can use AWS Control Tower to detect changes in your landing zone that might affect compliance with governance requirements.

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

EDI

See electronic data interchange.

edge computing

The technology that increases the computing power for smart devices at the edges of an IoT network. When compared with cloud computing, edge computing can reduce communication latency and improve response time.

electronic data interchange (EDI)

The automated exchange of business documents between organizations. For more information, see What is Electronic Data Interchange.

encryption

A computing process that transforms plaintext data, which is human-readable, into ciphertext.

encryption key

A cryptographic string of randomized bits that is generated by an encryption algorithm. Keys can vary in length, and each key is designed to be unpredictable and unique.

endianness

The order in which bytes are stored in computer memory. Big-endian systems store the most significant byte first. Little-endian systems store the least significant byte first.

endpoint

See service endpoint.

endpoint service

A service that you can host in a virtual private cloud (VPC) to share with other users. You can create an endpoint service with AWS PrivateLink and grant permissions to other AWS accounts or to AWS Identity and Access Management (IAM) principals. These accounts or principals can connect to your endpoint service privately by creating interface VPC endpoints. For more

information, see Create an endpoint service in the Amazon Virtual Private Cloud (Amazon VPC) documentation.

enterprise resource planning (ERP)

A system that automates and manages key business processes (such as accounting, MES, and project management) for an enterprise.

envelope encryption

The process of encrypting an encryption key with another encryption key. For more information, see Envelope encryption in the AWS Key Management Service (AWS KMS) documentation.

environment

An instance of a running application. The following are common types of environments in cloud computing:

- development environment – An instance of a running application that is available only to the core team responsible for maintaining the application. Development environments are used to test changes before promoting them to upper environments. This type of environment is sometimes referred to as a *test environment*.
- lower environments – All development environments for an application, such as those used for initial builds and tests.
- production environment – An instance of a running application that end users can access. In a CI/CD pipeline, the production environment is the last deployment environment.
- upper environments – All environments that can be accessed by users other than the core development team. This can include a production environment, preproduction environments, and environments for user acceptance testing.

epic

In agile methodologies, functional categories that help organize and prioritize your work. Epics provide a high-level description of requirements and implementation tasks. For example, AWS CAF security epics include identity and access management, detective controls, infrastructure security, data protection, and incident response. For more information about epics in the AWS migration strategy, see the program implementation guide.

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

The process of analyzing a dataset to understand its main characteristics. You collect or aggregate data and then perform initial investigations to find patterns, detect anomalies, and check assumptions. EDA is performed by calculating summary statistics and creating data visualizations.

F

fact table

The central table in a star schema. It stores quantitative data about business operations. Typically, a fact table contains two types of columns: those that contain measures and those that contain a foreign key to a dimension table.

fail fast

A philosophy that uses frequent and incremental testing to reduce the development lifecycle. It is a critical part of an agile approach.

fault isolation boundary

In the AWS Cloud, a boundary such as an Availability Zone, AWS Region, control plane, or data plane that limits the effect of a failure and helps improve the resilience of workloads. For more information, see AWS Fault Isolation Boundaries.

feature branch

See branch.

features

The input data that you use to make a prediction. For example, in a manufacturing context, features could be images that are periodically captured from the manufacturing line.

feature importance

How significant a feature is for a model's predictions. This is usually expressed as a numerical score that can be calculated through various techniques, such as Shapley Additive Explanations (SHAP) and integrated gradients. For more information, see Machine learning model interpretability with AWS.

feature transformation

To optimize data for the ML process, including enriching data with additional sources, scaling values, or extracting multiple sets of information from a single data field. This enables the ML model to benefit from the data. For example, if you break down the “2021-05-27 00:15:37” date into “2021”, “May”, “Thu”, and “15”, you can help the learning algorithm learn nuanced patterns associated with different data components.

few-shot prompting

Providing an LLM with a small number of examples that demonstrate the task and desired output before asking it to perform a similar task. This technique is an application of in-context learning, where models learn from examples (*shots*) that are embedded in prompts. Few-shot prompting can be effective for tasks that require specific formatting, reasoning, or domain knowledge. See also zero-shot prompting.

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

The use of multiple conditions to allow or deny an access request.

flash-cut migration

A database migration method that uses continuous data replication through change data capture to migrate data in the shortest time possible, instead of using a phased approach. The objective is to keep downtime to a minimum.

FM

See foundation model.

foundation model (FM)

A large deep-learning neural network that has been training on massive datasets of generalized and unlabeled data. FMs are capable of performing a wide variety of general tasks, such as understanding language, generating text and images, and conversing in natural language. For more information, see What are Foundation Models.

G

generative AI

A subset of AI models that have been trained on large amounts of data and that can use a simple text prompt to create new content and artifacts, such as images, videos, text, and audio. For more information, see What is Generative AI.

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

In Amazon CloudFront, an option to prevent users in specific countries from accessing content distributions. You can use an allow list or block list to specify approved and banned countries. For more information, see Restricting the geographic distribution of your content in the CloudFront documentation.

Gitflow workflow

An approach in which lower and upper environments use different branches in a source code repository. The Gitflow workflow is considered legacy, and the trunk-based workflow is the modern, preferred approach.

golden image

A snapshot of a system or software that is used as a template to deploy new instances of that system or software. For example, in manufacturing, a golden image can be used to provision software on multiple devices and helps improve speed, scalability, and productivity in device manufacturing operations.

greenfield strategy

The absence of existing infrastructure in a new environment. When adopting a greenfield strategy for a system architecture, you can select all new technologies without the restriction of compatibility with existing infrastructure, also known as brownfield. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

guardrail

A high-level rule that helps govern resources, policies, and compliance across organizational units (OUs). *Preventive guardrails* enforce policies to ensure alignment to compliance standards. They are implemented by using service control policies and IAM permissions boundaries.

Detective guardrails detect policy violations and compliance issues, and generate alerts for remediation. They are implemented by using AWS Config, AWS Security Hub CSPM, Amazon GuardDuty, AWS Trusted Advisor, Amazon Inspector, and custom AWS Lambda checks.

H

HA

See high availability.

heterogeneous database migration

Migrating your source database to a target database that uses a different database engine (for example, Oracle to Amazon Aurora). Heterogeneous migration is typically part of a re-architecting effort, and converting the schema can be a complex task. AWS provides AWS SCT that helps with schema conversions.

high availability (HA)

The ability of a workload to operate continuously, without intervention, in the event of challenges or disasters. HA systems are designed to automatically fail over, consistently deliver high-quality performance, and handle different loads and failures with minimal performance impact.

historian modernization

An approach used to modernize and upgrade operational technology (OT) systems to better serve the needs of the manufacturing industry. A *historian* is a type of database that is used to collect and store data from various sources in a factory.

holdout data

A portion of historical, labeled data that is withheld from a dataset that is used to train a machine learning model. You can use holdout data to evaluate the model performance by comparing the model predictions against the holdout data.

homogeneous database migration

Migrating your source database to a target database that shares the same database engine (for example, Microsoft SQL Server to Amazon RDS for SQL Server). Homogeneous migration is typically part of a rehosting or replatforming effort. You can use native database utilities to migrate the schema.

hot data

Data that is frequently accessed, such as real-time data or recent translational data. This data typically requires a high-performance storage tier or class to provide fast query responses.

hotfix

An urgent fix for a critical issue in a production environment. Due to its urgency, a hotfix is usually made outside of the typical DevOps release workflow.

hypercare period

Immediately following cutover, the period of time when a migration team manages and monitors the migrated applications in the cloud in order to address any issues. Typically, this period is 1–4 days in length. At the end of the hypercare period, the migration team typically transfers responsibility for the applications to the cloud operations team.

I

IaC

See infrastructure as code.

identity-based policy

A policy attached to one or more IAM principals that defines their permissions within the AWS Cloud environment.

idle application

An application that has an average CPU and memory usage between 5 and 20 percent over a period of 90 days. In a migration project, it is common to retire these applications or retain them on premises.

IIoT

See industrial Internet of Things.

immutable infrastructure

A model that deploys new infrastructure for production workloads instead of updating, patching, or modifying the existing infrastructure. Immutable infrastructures are inherently more consistent, reliable, and predictable than mutable infrastructure. For more information, see the Deploy using immutable infrastructure best practice in the AWS Well-Architected Framework.

inbound (ingress) VPC

In an AWS multi-account architecture, a VPC that accepts, inspects, and routes network connections from outside an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

incremental migration

A cutover strategy in which you migrate your application in small parts instead of performing a single, full cutover. For example, you might move only a few microservices or users to the new system initially. After you verify that everything is working properly, you can incrementally move additional microservices or users until you can decommission your legacy system. This strategy reduces the risks associated with large migrations.

Industry 4.0

A term that was introduced by Klaus Schwab in 2016 to refer to the modernization of manufacturing processes through advances in connectivity, real-time data, automation, analytics, and AI/ML.

infrastructure

All of the resources and assets contained within an application's environment.

infrastructure as code (IaC)

The process of provisioning and managing an application's infrastructure through a set of configuration files. IaC is designed to help you centralize infrastructure management, standardize resources, and scale quickly so that new environments are repeatable, reliable, and consistent.

industrial Internet of Things (IIoT)

The use of internet-connected sensors and devices in the industrial sectors, such as manufacturing, energy, automotive, healthcare, life sciences, and agriculture. For more information, see Building an industrial Internet of Things (IIoT) digital transformation strategy.

inspection VPC

In an AWS multi-account architecture, a centralized VPC that manages inspections of network traffic between VPCs (in the same or different AWS Regions), the internet, and on-premises networks. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

Internet of Things (IoT)

The network of connected physical objects with embedded sensors or processors that communicate with other devices and systems through the internet or over a local communication network. For more information, see What is IoT?

interpretability

A characteristic of a machine learning model that describes the degree to which a human can understand how the model's predictions depend on its inputs. For more information, see Machine learning model interpretability with AWS.

IoT

See Internet of Things.

IT information library (ITIL)

A set of best practices for delivering IT services and aligning these services with business requirements. ITIL provides the foundation for ITSM.

IT service management (ITSM)

Activities associated with designing, implementing, managing, and supporting IT services for an organization. For information about integrating cloud operations with ITSM tools, see the operations integration guide.

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

An implementation of mandatory access control (MAC) where the users and the data itself are each explicitly assigned a security label value. The intersection between the user security label and data security label determines which rows and columns can be seen by the user.

landing zone

A landing zone is a well-architected, multi-account AWS environment that is scalable and secure. This is a starting point from which your organizations can quickly launch and deploy workloads and applications with confidence in their security and infrastructure environment. For more information about landing zones, see Setting up a secure and scalable multi-account AWS environment.

large language model (LLM)

A deep learning AI model that is pretrained on a vast amount of data. An LLM can perform multiple tasks, such as answering questions, summarizing documents, translating text into other languages, and completing sentences. For more information, see What are LLMs.

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

The security best practice of granting the minimum permissions required to perform a task. For more information, see Apply least-privilege permissions in the IAM documentation.

lift and shift

See 7 Rs.

little-endian system

A system that stores the least significant byte first. See also endianness.

LLM

See large language model.

lower environments

See environment.

M

machine learning (ML)

A type of artificial intelligence that uses algorithms and techniques for pattern recognition and learning. ML analyzes and learns from recorded data, such as Internet of Things (IoT) data, to generate a statistical model based on patterns. For more information, see Machine Learning.

main branch

See branch.

malware

Software that is designed to compromise computer security or privacy. Malware might disrupt computer systems, leak sensitive information, or gain unauthorized access. Examples of malware include viruses, worms, ransomware, Trojan horses, spyware, and keyloggers.

managed services

AWS services for which AWS operates the infrastructure layer, the operating system, and platforms, and you access the endpoints to store and retrieve data. Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB are examples of managed services. These are also known as *abstracted services*.

manufacturing execution system (MES)

A software system for tracking, monitoring, documenting, and controlling production processes that convert raw materials to finished products on the shop floor.

MAP

See Migration Acceleration Program.

mechanism

A complete process in which you create a tool, drive adoption of the tool, and then inspect the results in order to make adjustments. A mechanism is a cycle that reinforces and improves itself as it operates. For more information, see Building mechanisms in the AWS Well-Architected Framework.

member account

All AWS accounts other than the management account that are part of an organization in AWS Organizations. An account can be a member of only one organization at a time.

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

A lightweight, machine-to-machine (M2M) communication protocol, based on the publish/subscribe pattern, for resource-constrained IoT devices.

microservice

A small, independent service that communicates over well-defined APIs and is typically owned by small, self-contained teams. For example, an insurance system might include microservices that map to business capabilities, such as sales or marketing, or subdomains, such as purchasing, claims, or analytics. The benefits of microservices include agility, flexible scaling, easy deployment, reusable code, and resilience. For more information, see Integrating microservices by using AWS serverless services.

microservices architecture

An approach to building an application with independent components that run each application process as a microservice. These microservices communicate through a well-defined interface by using lightweight APIs. Each microservice in this architecture can be updated, deployed, and scaled to meet demand for specific functions of an application. For more information, see Implementing microservices on AWS.

Migration Acceleration Program (MAP)

An AWS program that provides consulting support, training, and services to help organizations build a strong operational foundation for moving to the cloud, and to help offset the initial cost of migrations. MAP includes a migration methodology for executing legacy migrations in a methodical way and a set of tools to automate and accelerate common migration scenarios.

migration at scale

The process of moving the majority of the application portfolio to the cloud in waves, with more applications moved at a faster rate in each wave. This phase uses the best practices and lessons learned from the earlier phases to implement a *migration factory* of teams, tools, and processes to streamline the migration of workloads through automation and agile delivery. This is the third phase of the AWS migration strategy.

migration factory

Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners,

migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the Cloud Migration Factory guide in this content set.

migration metadata

The information about the application and server that is needed to complete the migration. Each migration pattern requires a different set of migration metadata. Examples of migration metadata include the target subnet, security group, and AWS account.

migration pattern

A repeatable migration task that details the migration strategy, the migration destination, and the migration application or service used. Example: Rehost migration to Amazon EC2 with AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

An online tool that provides information for validating the business case for migrating to the AWS Cloud. MPA provides detailed portfolio assessment (server right-sizing, pricing, TCO comparisons, migration cost analysis) as well as migration planning (application data analysis and data collection, application grouping, migration prioritization, and wave planning). The MPA tool (requires login) is available free of charge to all AWS consultants and APN Partner consultants.

Migration Readiness Assessment (MRA)

The process of gaining insights about an organization's cloud readiness status, identifying strengths and weaknesses, and building an action plan to close identified gaps, using the AWS CAF. For more information, see the migration readiness guide. MRA is the first phase of the AWS migration strategy.

migration strategy

The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs entry in this glossary and see Mobilize your organization to accelerate large-scale migrations.

ML

See machine learning.

modernization

Transforming an outdated (legacy or monolithic) application and its infrastructure into an agile, elastic, and highly available system in the cloud to reduce costs, gain efficiencies, and take advantage of innovations. For more information, see Strategy for modernizing applications in the AWS Cloud.

modernization readiness assessment

An evaluation that helps determine the modernization readiness of an organization's applications; identifies benefits, risks, and dependencies; and determines how well the organization can support the future state of those applications. The outcome of the assessment is a blueprint of the target architecture, a roadmap that details development phases and milestones for the modernization process, and an action plan for addressing identified gaps. For more information, see Evaluating modernization readiness for applications in the AWS Cloud.

monolithic applications (monoliths)

Applications that run as a single service with tightly coupled processes. Monolithic applications have several drawbacks. If one application feature experiences a spike in demand, the entire architecture must be scaled. Adding or improving a monolithic application's features also becomes more complex when the code base grows. To address these issues, you can use a microservices architecture. For more information, see Decomposing monoliths into microservices.

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

A process that helps generate predictions for multiple classes (predicting one of more than two outcomes). For example, an ML model might ask "Is this product a book, car, or phone?" or "Which product category is most interesting to this customer?"

mutable infrastructure

A model that updates and modifies the existing infrastructure for production workloads. For improved consistency, reliability, and predictability, the AWS Well-Architected Framework recommends the use of immutable infrastructure as a best practice.

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

A migration method in which the source workload is taken down during the migration process. This method involves extended downtime and is typically used for small, non-critical workloads.

OI

See operations integration.

OLA

See operational-level agreement.

online migration

A migration method in which the source workload is copied to the target system without being taken offline. Applications that are connected to the workload can continue to function during the migration. This method involves zero to minimal downtime and is typically used for critical production workloads.

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

A machine-to-machine (M2M) communication protocol for industrial automation. OPC-UA provides an interoperability standard with data encryption, authentication, and authorization schemes.

operational-level agreement (OLA)

An agreement that clarifies what functional IT groups promise to deliver to each other, to support a service-level agreement (SLA).

operational readiness review (ORR)

A checklist of questions and associated best practices that help you understand, evaluate, prevent, or reduce the scope of incidents and possible failures. For more information, see Operational Readiness Reviews (ORR) in the AWS Well-Architected Framework.

operational technology (OT)

Hardware and software systems that work with the physical environment to control industrial operations, equipment, and infrastructure. In manufacturing, the integration of OT and information technology (IT) systems is a key focus for Industry 4.0 transformations.

operations integration (OI)

The process of modernizing operations in the cloud, which involves readiness planning, automation, and integration. For more information, see the operations integration guide.

organization trail

A trail that's created by AWS CloudTrail that logs all events for all AWS accounts in an organization in AWS Organizations. This trail is created in each AWS account that's part of the organization and tracks the activity in each account. For more information, see Creating a trail for an organization in the CloudTrail documentation.

organizational change management (OCM)

A framework for managing major, disruptive business transformations from a people, culture, and leadership perspective. OCM helps organizations prepare for, and transition to, new systems and strategies by accelerating change adoption, addressing transitional issues, and driving cultural and organizational changes. In the AWS migration strategy, this framework is called *people acceleration*, because of the speed of change required in cloud adoption projects. For more information, see the OCM guide.

origin access control (OAC)

In CloudFront, an enhanced option for restricting access to secure your Amazon Simple Storage Service (Amazon S3) content. OAC supports all S3 buckets in all AWS Regions, server-side encryption with AWS KMS (SSE-KMS), and dynamic PUT and DELETE requests to the S3 bucket.

origin access identity (OAI)

In CloudFront, an option for restricting access to secure your Amazon S3 content. When you use OAI, CloudFront creates a principal that Amazon S3 can authenticate with. Authenticated principals can access content in an S3 bucket only through a specific CloudFront distribution. See also OAC, which provides more granular and enhanced access control.

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

In an AWS multi-account architecture, a VPC that handles network connections that are initiated from within an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

P

permissions boundary

An IAM management policy that is attached to IAM principals to set the maximum permissions that the user or role can have. For more information, see Permissions boundaries in the IAM documentation.

personally identifiable information (PII)

Information that, when viewed directly or paired with other related data, can be used to reasonably infer the identity of an individual. Examples of PII include names, addresses, and contact information.

PII

See personally identifiable information.

playbook

A set of predefined steps that capture the work associated with migrations, such as delivering core operations functions in the cloud. A playbook can take the form of scripts, automated runbooks, or a summary of processes or steps required to operate your modernized environment.

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

An object that can define permissions (see identity-based policy), specify access conditions (see resource-based policy), or define the maximum permissions for all accounts in an organization in AWS Organizations (see service control policy).

polyglot persistence

Independently choosing a microservice's data storage technology based on data access patterns and other requirements. If your microservices have the same data storage technology, they can encounter implementation challenges or experience poor performance. Microservices are more easily implemented and achieve better performance and scalability if they use the data store best adapted to their requirements. For more information, see Enabling data persistence in microservices.

portfolio assessment

A process of discovering, analyzing, and prioritizing the application portfolio in order to plan the migration. For more information, see Evaluating migration readiness.

predicate

A query condition that returns `true` or `false`, commonly located in a `WHERE` clause.

predicate pushdown

A database query optimization technique that filters the data in the query before transfer. This reduces the amount of data that must be retrieved and processed from the relational database, and it improves query performance.

preventative control

A security control that is designed to prevent an event from occurring. These controls are a first line of defense to help prevent unauthorized access or unwanted changes to your network. For more information, see Preventative controls in *Implementing security controls on AWS*.

principal

An entity in AWS that can perform actions and access resources. This entity is typically a root user for an AWS account, an IAM role, or a user. For more information, see *Principal* in Roles terms and concepts in the IAM documentation.

privacy by design

A system engineering approach that takes privacy into account through the whole development process.

private hosted zones

A container that holds information about how you want Amazon Route 53 to respond to DNS queries for a domain and its subdomains within one or more VPCs. For more information, see Working with private hosted zones in the Route 53 documentation.

proactive control

A security control designed to prevent the deployment of noncompliant resources. These controls scan resources before they are provisioned. If the resource is not compliant with the control, then it isn't provisioned. For more information, see the Controls reference guide in the AWS Control Tower documentation and see Proactive controls in *Implementing security controls on AWS*.

product lifecycle management (PLM)

The management of data and processes for a product throughout its entire lifecycle, from design, development, and launch, through growth and maturity, to decline and removal.

production environment

See environment.

programmable logic controller (PLC)

In manufacturing, a highly reliable, adaptable computer that monitors machines and automates manufacturing processes.

prompt chaining

Using the output of one LLM prompt as the input for the next prompt to generate better responses. This technique is used to break down a complex task into subtasks, or to iteratively refine or expand a preliminary response. It helps improve the accuracy and relevance of a model's responses and allows for more granular, personalized results.

pseudonymization

The process of replacing personal identifiers in a dataset with placeholder values. Pseudonymization can help protect personal privacy. Pseudonymized data is still considered to be personal data.

publish/subscribe (pub/sub)

A pattern that enables asynchronous communications among microservices to improve scalability and responsiveness. For example, in a microservices-based MES, a microservice can publish event messages to a channel that other microservices can subscribe to. The system can add new microservices without changing the publishing service.

Q

query plan

A series of steps, like instructions, that are used to access the data in a SQL relational database system.

query plan regression

When a database service optimizer chooses a less optimal plan than it did before a given change to the database environment. This can be caused by changes to statistics, constraints, environment settings, query parameter bindings, and updates to the database engine.

R

RACI matrix

See responsible, accountable, consulted, informed (RACI).

RAG

See Retrieval Augmented Generation.

ransomware

A malicious software that is designed to block access to a computer system or data until a payment is made.

RASCI matrix

See responsible, accountable, consulted, informed (RACI).

RCAC

See row and column access control.

read replica

A copy of a database that's used for read-only purposes. You can route queries to the read replica to reduce the load on your primary database.

re-architect

See 7 Rs.

recovery point objective (RPO)

The maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable loss of data between the last recovery point and the interruption of service.

recovery time objective (RTO)

The maximum acceptable delay between the interruption of service and restoration of service.

refactor

See 7 Rs.

Region

A collection of AWS resources in a geographic area. Each AWS Region is isolated and independent of the others to provide fault tolerance, stability, and resilience. For more information, see Specify which AWS Regions your account can use.

regression

An ML technique that predicts a numeric value. For example, to solve the problem of "What price will this house sell for?" an ML model could use a linear regression model to predict a house's sale price based on known facts about the house (for example, the square footage).

rehost

See 7 Rs.

release

In a deployment process, the act of promoting changes to a production environment.

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

An application's ability to resist or recover from disruptions. High availability and disaster recovery are common considerations when planning for resiliency in the AWS Cloud. For more information, see AWS Cloud Resilience.

resource-based policy

A policy attached to a resource, such as an Amazon S3 bucket, an endpoint, or an encryption key. This type of policy specifies which principals are allowed access, supported actions, and any other conditions that must be met.

responsible, accountable, consulted, informed (RACI) matrix

A matrix that defines the roles and responsibilities for all parties involved in migration activities and cloud operations. The matrix name is derived from the responsibility types defined in the matrix: responsible (R), accountable (A), consulted (C), and informed (I). The support (S) type is optional. If you include support, the matrix is called a *RASCI matrix*, and if you exclude it, it's called a *RACI matrix*.

responsive control

A security control that is designed to drive remediation of adverse events or deviations from your security baseline. For more information, see Responsive controls in *Implementing security controls on AWS*.

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

A generative AI technology in which an LLM references an authoritative data source that is outside of its training data sources before generating a response. For example, a RAG model might perform a semantic search of an organization's knowledge base or custom data. For more information, see What is RAG.

rotation

The process of periodically updating a secret to make it more difficult for an attacker to access the credentials.

row and column access control (RCAC)

The use of basic, flexible SQL expressions that have defined access rules. RCAC consists of row permissions and column masks.

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

A set of manual or automated procedures required to perform a specific task. These are typically built to streamline repetitive operations or procedures with high error rates.

S

SAML 2.0

An open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS API operations without you having to create user in IAM for everyone in your organization. For more information about SAML 2.0-based federation, see About SAML 2.0-based federation in the IAM documentation.

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

In AWS Secrets Manager, confidential or restricted information, such as a password or user credentials, that you store in encrypted form. It consists of the secret value and its metadata.

The secret value can be binary, a single string, or multiple strings. For more information, see What's in a Secrets Manager secret? in the Secrets Manager documentation.

security by design

A system engineering approach that takes security into account through the whole development process.

security control

A technical or administrative guardrail that prevents, detects, or reduces the ability of a threat actor to exploit a security vulnerability. There are four primary types of security controls: preventative, detective, responsive, and proactive.

security hardening

The process of reducing the attack surface to make it more resistant to attacks. This can include actions such as removing resources that are no longer needed, implementing the security best practice of granting least privilege, or deactivating unnecessary features in configuration files.

security information and event management (SIEM) system

Tools and services that combine security information management (SIM) and security event management (SEM) systems. A SIEM system collects, monitors, and analyzes data from servers, networks, devices, and other sources to detect threats and security breaches, and to generate alerts.

security response automation

A predefined and programmed action that is designed to automatically respond to or remediate a security event. These automations serve as detective or responsive security controls that help you implement AWS security best practices. Examples of automated response actions include modifying a VPC security group, patching an Amazon EC2 instance, or rotating credentials.

server-side encryption

Encryption of data at its destination, by the AWS service that receives it.

service control policy (SCP)

A policy that provides centralized control over permissions for all accounts in an organization in AWS Organizations. SCPs define guardrails or set limits on actions that an administrator can delegate to users or roles. You can use SCPs as allow lists or deny lists, to specify which services or actions are permitted or prohibited. For more information, see Service control policies in the AWS Organizations documentation.

service endpoint

The URL of the entry point for an AWS service. You can use the endpoint to connect programmatically to the target service. For more information, see AWS service endpoints in *AWS General Reference*.

service-level agreement (SLA)

An agreement that clarifies what an IT team promises to deliver to their customers, such as service uptime and performance.

service-level indicator (SLI)

A measurement of a performance aspect of a service, such as its error rate, availability, or throughput.

service-level objective (SLO)

A target metric that represents the health of a service, as measured by a service-level indicator.

shared responsibility model

A model describing the responsibility you share with AWS for cloud security and compliance. AWS is responsible for security *of* the cloud, whereas you are responsible for security *in* the cloud. For more information, see Shared responsibility model.

SIEM

See security information and event management system.

single point of failure (SPOF)

A failure in a single, critical component of an application that can disrupt the system.

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your

organization's capabilities and services, improves developer productivity, and supports rapid innovation. For more information, see Phased approach to modernizing applications in the AWS Cloud.

SPOF

See single point of failure.

star schema

A database organizational structure that uses one large fact table to store transactional or measured data and uses one or more smaller dimensional tables to store data attributes. This structure is designed for use in a data warehouse or for business intelligence purposes.

strangler fig pattern

An approach to modernizing monolithic systems by incrementally rewriting and replacing system functionality until the legacy system can be decommissioned. This pattern uses the analogy of a fig vine that grows into an established tree and eventually overcomes and replaces its host. The pattern was introduced by Martin Fowler as a way to manage risk when rewriting monolithic systems. For an example of how to apply this pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

subnet

A range of IP addresses in your VPC. A subnet must reside in a single Availability Zone.

supervisory control and data acquisition (SCADA)

In manufacturing, a system that uses hardware and software to monitor physical assets and production operations.

symmetric encryption

An encryption algorithm that uses the same key to encrypt and decrypt the data.

synthetic testing

Testing a system in a way that simulates user interactions to detect potential issues or to monitor performance. You can use Amazon CloudWatch Synthetics to create these tests.

system prompt

A technique for providing context, instructions, or guidelines to an LLM to direct its behavior. System prompts help set context and establish rules for interactions with users.

T

tags

Key-value pairs that act as metadata for organizing your AWS resources. Tags can help you manage, identify, organize, search for, and filter resources. For more information, see Tagging your AWS resources.

target variable

The value that you are trying to predict in supervised ML. This is also referred to as an *outcome variable*. For example, in a manufacturing setting the target variable could be a product defect.

task list

A tool that is used to track progress through a runbook. A task list contains an overview of the runbook and a list of general tasks to be completed. For each general task, it includes the estimated amount of time required, the owner, and the progress.

test environment

See environment.

training

To provide data for your ML model to learn from. The training data must contain the correct answer. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict). It outputs an ML model that captures these patterns. You can then use the ML model to make predictions on new data for which you don't know the target.

transit gateway

A network transit hub that you can use to interconnect your VPCs and on-premises networks. For more information, see What is a transit gateway in the AWS Transit Gateway documentation.

trunk-based workflow

An approach in which developers build and test features locally in a feature branch and then merge those changes into the main branch. The main branch is then built to the development, preproduction, and production environments, sequentially.

trusted access

Granting permissions to a service that you specify to perform tasks in your organization in AWS Organizations and in its accounts on your behalf. The trusted service creates a service-linked role in each account, when that role is needed, to perform management tasks for you. For more information, see Using AWS Organizations with other AWS services in the AWS Organizations documentation.

tuning

To change aspects of your training process to improve the ML model's accuracy. For example, you can train the ML model by generating a labeling set, adding labels, and then repeating these steps several times under different settings to optimize the model.

two-pizza team

A small DevOps team that you can feed with two pizzas. A two-pizza team size ensures the best possible opportunity for collaboration in software development.

U

uncertainty

A concept that refers to imprecise, incomplete, or unknown information that can undermine the reliability of predictive ML models. There are two types of uncertainty: *Epistemic uncertainty* is caused by limited, incomplete data, whereas *aleatoric uncertainty* is caused by the noise and randomness inherent in the data. For more information, see the Quantifying uncertainty in deep learning systems guide.

undifferentiated tasks

Also known as *heavy lifting*, work that is necessary to create and operate an application but that doesn't provide direct value to the end user or provide competitive advantage. Examples of undifferentiated tasks include procurement, maintenance, and capacity planning.

upper environments

See environment.

V

vacuuming

A database maintenance operation that involves cleaning up after incremental updates to reclaim storage and improve performance.

version control

Processes and tools that track changes, such as changes to source code in a repository.

VPC peering

A connection between two VPCs that allows you to route traffic by using private IP addresses. For more information, see What is VPC peering in the Amazon VPC documentation.

vulnerability

A software or hardware flaw that compromises the security of the system.

W

warm cache

A buffer cache that contains current, relevant data that is frequently accessed. The database instance can read from the buffer cache, which is faster than reading from the main memory or disk.

warm data

Data that is infrequently accessed. When querying this kind of data, moderately slow queries are typically acceptable.

window function

A SQL function that performs a calculation on a group of rows that relate in some way to the current record. Window functions are useful for processing tasks, such as calculating a moving average or accessing the value of rows based on the relative position of the current row.

workload

A collection of resources and code that delivers business value, such as a customer-facing application or backend process.

workstream

Functional groups in a migration project that are responsible for a specific set of tasks. Each workstream is independent but supports the other workstreams in the project. For example, the portfolio workstream is responsible for prioritizing applications, wave planning, and collecting migration metadata. The portfolio workstream delivers these assets to the migration workstream, which then migrates the servers and applications.

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

A storage model that writes data a single time and prevents the data from being deleted or modified. Authorized users can read the data as many times as needed, but they cannot change it. This data storage infrastructure is considered immutable.

Z

zero-day exploit

An attack, typically malware, that takes advantage of a zero-day vulnerability.

zero-day vulnerability

An unmitigated flaw or vulnerability in a production system. Threat actors can use this type of vulnerability to attack the system. Developers frequently become aware of the vulnerability as a result of the attack.

zero-shot prompting

Providing an LLM with instructions for performing a task but no examples (*shots*) that can help guide it. The LLM must use its pre-trained knowledge to handle the task. The effectiveness of zero-shot prompting depends on the complexity of the task and the quality of the prompt. See also few-shot prompting.

zombie application

An application that has an average CPU and memory usage below 5 percent. In a migration project, it is common to retire these applications.