

# Cloud



# Optimization Guide

## Compute, Storage & Database

---

Check out this checklist of technical recommendations that can be made to your **AWS cloud infrastructure** to launch or improve your organization's cost optimization processes.

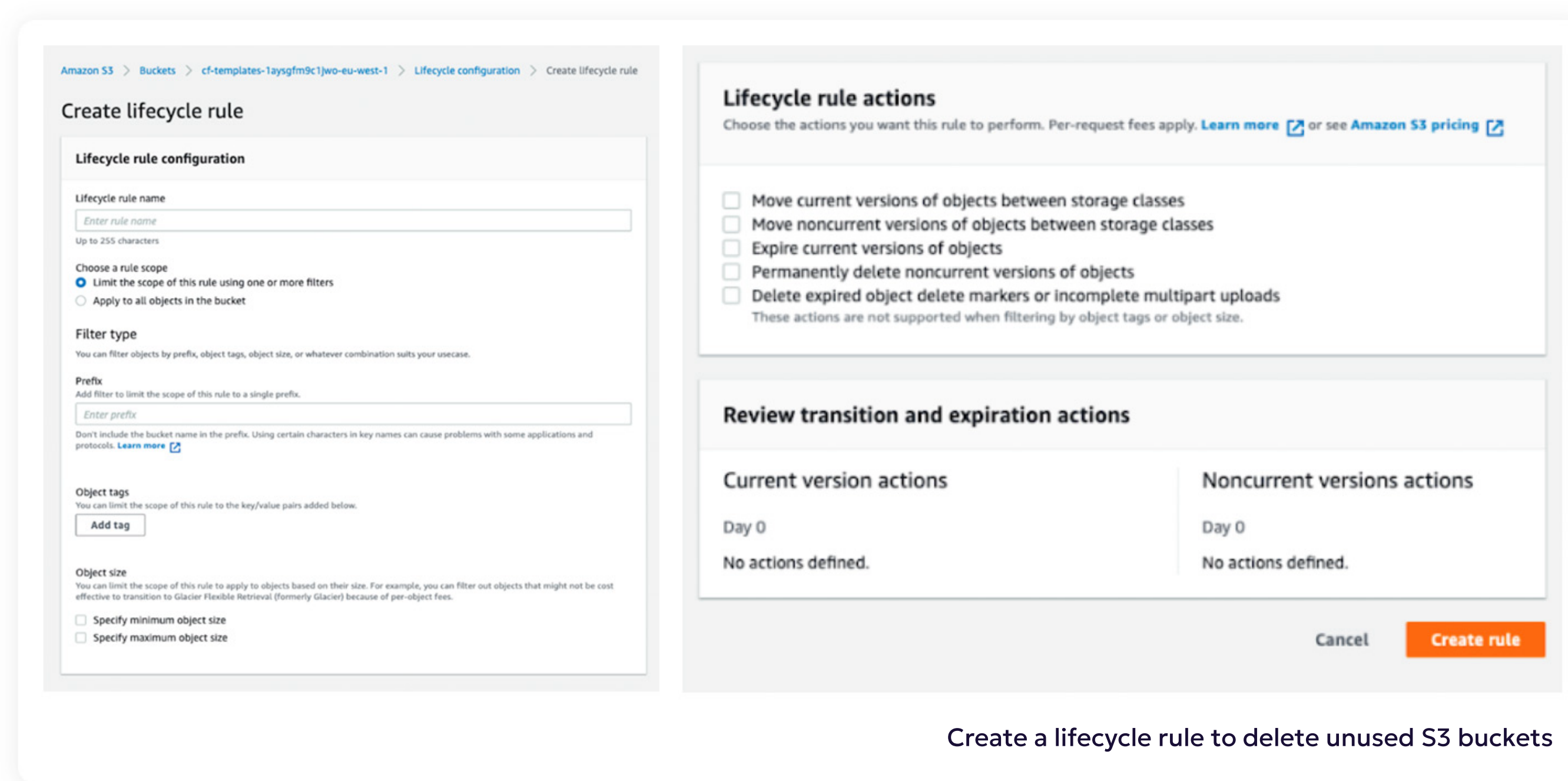


## Storage

ROT is not doing you any favors. Redundant, Obsolete, and Trivial data clogs your cloud infrastructure, slows down compute times, and unnecessarily compounds costs. Here are a few strategies to eliminate ROT.

### Delete unused object storage files

You can use an S3 lifecycle rule to set a policy that will automate the process of moving objects that aren't accessed frequently to an S3 bucket which takes slightly more time for data to be retrieved but provides a cost-effective storage location. You can set the policy of the time duration in which data can be moved across, and eventually deleted. Alternatively, this process can be automated by using [S3 intelligent tiering](#).

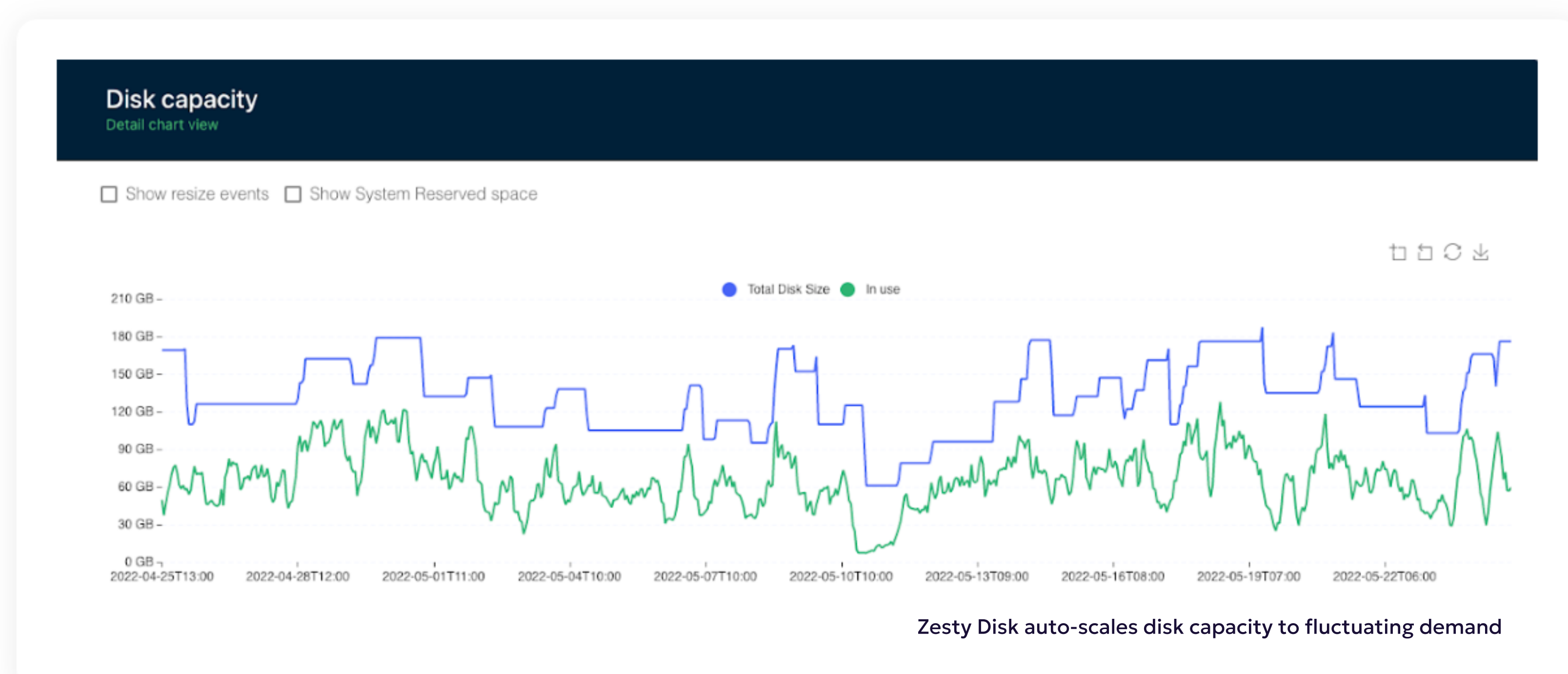


### Identify volumes that have very low activity

This data is clogging up your infrastructure and may be good candidates for deletion. To do this you can use [EBS Volumes Check](#) in Trusted Advisor to identify these underutilized and possibly orphaned volumes. You can also do this using [storage management and monitoring tools](#). Take a snapshot to ensure that the data isn't lost, and then delete the volume.

### Auto Scale your EBS Volumes

With the need to avoid application failure or slowdown, there is an industry-wide tendency to over-provision EBS storage volumes which results in paying between 2-5 times extra in cloud storage that doesn't end up getting used. However, it is possible to automatically scale volumes to application demand by using solutions like [Zesty Disk](#), which automatically adds filesystem storage when demand rises and removes them when demand drops off. This ensures that your application is always running optimally and cost-efficiently.



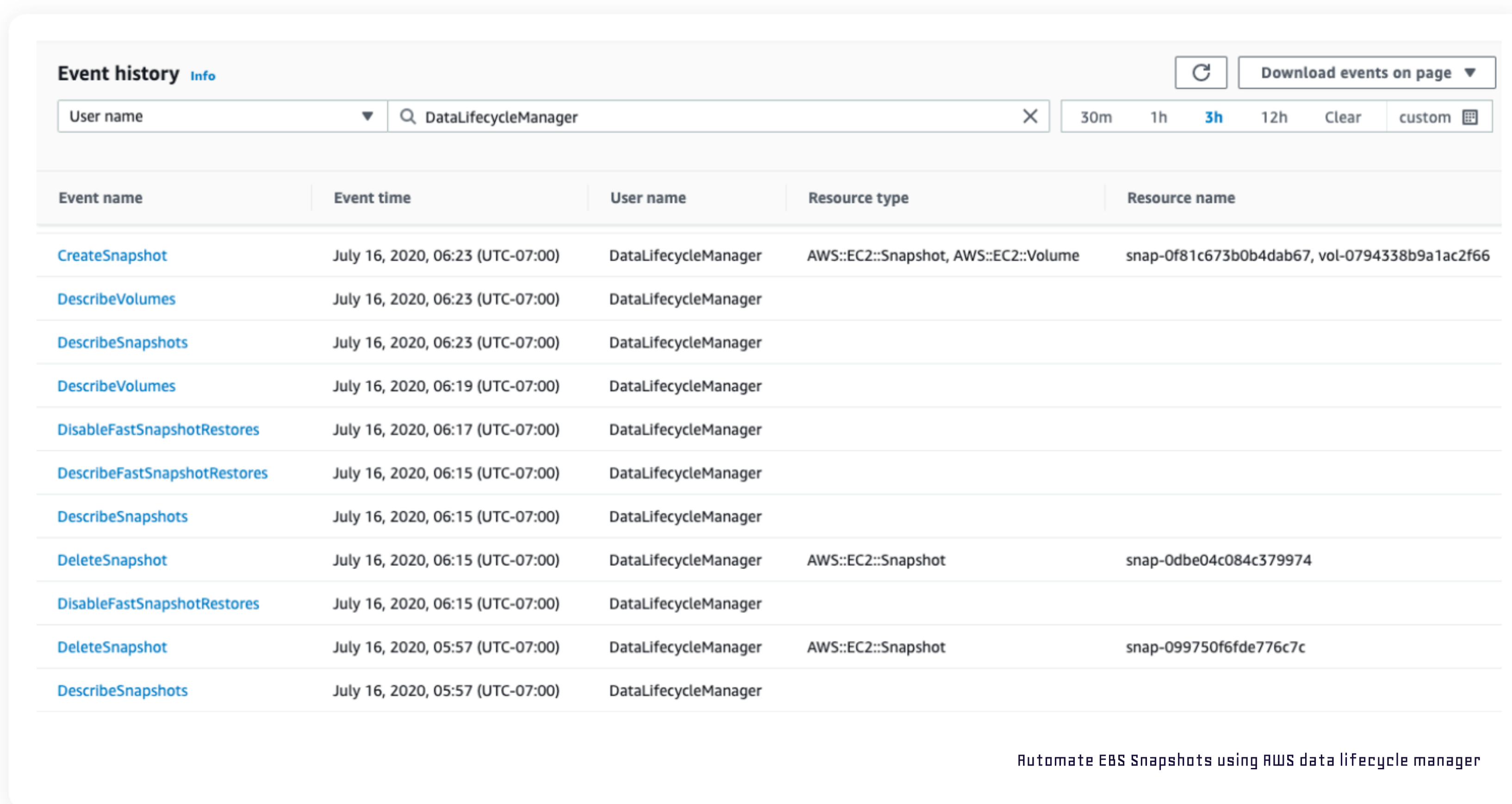


## Automate the management of Snapshots

To make this process more efficient, you can automate the management of snapshots for volumes that are getting old and are rarely used by using [Amazon Data Lifecycle Manager](#).

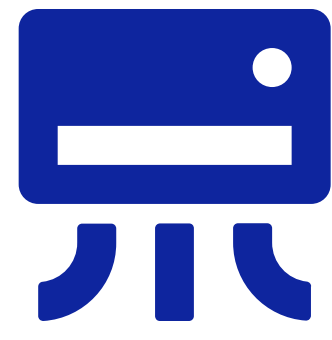
## Remove redundant snapshots

These may be orphaned snapshots or old snapshots that haven't been used for a while (the common parameter is 30 days). Snapshot retention policies can be set and modified so snapshots that aren't needed are no longer kept.



The screenshot shows the AWS Event History console. At the top, there's a search bar with 'DataLifecycleManager' entered and a filter set to '3h'. Below the search bar is a table with the following columns: Event name, Event time, User name, Resource type, and Resource name. The table contains 12 rows of event data. At the bottom right of the table area, there is a link: 'Automate EBS Snapshots using RWS data lifecycle manager'.

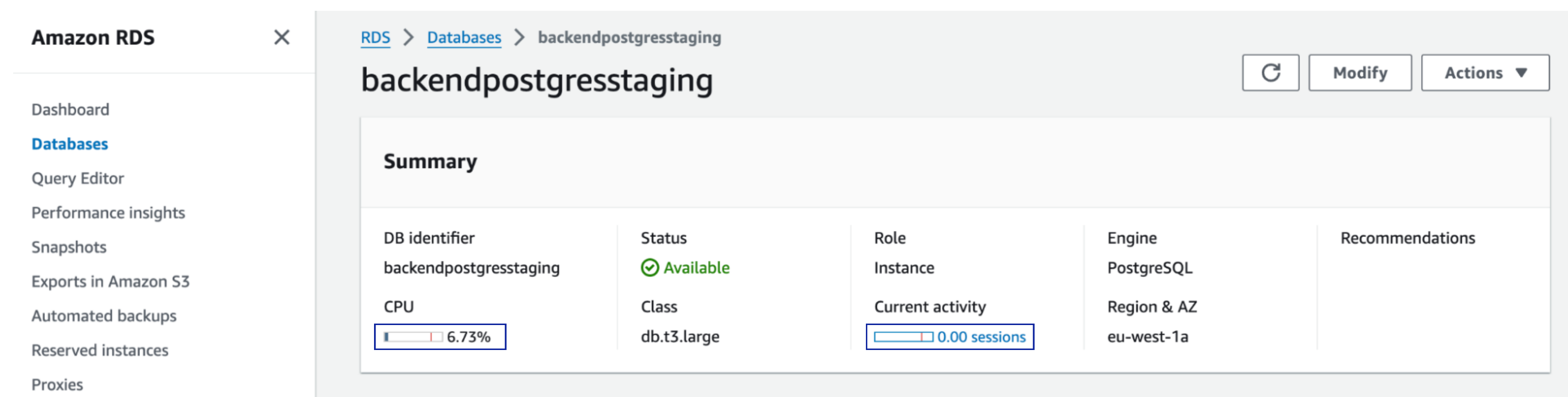
Event name	Event time	User name	Resource type	Resource name
CreateSnapshot	July 16, 2020, 06:23 (UTC-07:00)	DataLifecycleManager	AWS::EC2::Snapshot, AWS::EC2::Volume	snap-0f81c673b0b4dab67, vol-0794338b9a1ac2f66
DescribeVolumes	July 16, 2020, 06:23 (UTC-07:00)	DataLifecycleManager		
DescribeSnapshots	July 16, 2020, 06:23 (UTC-07:00)	DataLifecycleManager		
DescribeVolumes	July 16, 2020, 06:19 (UTC-07:00)	DataLifecycleManager		
DisableFastSnapshotRestores	July 16, 2020, 06:17 (UTC-07:00)	DataLifecycleManager		
DescribeFastSnapshotRestores	July 16, 2020, 06:15 (UTC-07:00)	DataLifecycleManager		
DescribeSnapshots	July 16, 2020, 06:15 (UTC-07:00)	DataLifecycleManager		
DeleteSnapshot	July 16, 2020, 06:15 (UTC-07:00)	DataLifecycleManager	AWS::EC2::Snapshot	snap-0dbe04c084c379974
DisableFastSnapshotRestores	July 16, 2020, 06:15 (UTC-07:00)	DataLifecycleManager		
DeleteSnapshot	July 16, 2020, 05:57 (UTC-07:00)	DataLifecycleManager	AWS::EC2::Snapshot	snap-099750f6fde776c7c
DescribeSnapshots	July 16, 2020, 05:57 (UTC-07:00)	DataLifecycleManager		



## Database

### Remove idle DB instances

If you have any DB instances that have not had any connection over the last seven days, it might be time to let go of them. Otherwise, you're paying for them to just sit around. You can identify them by using the [RDS Idle DB instances check](#) in Trusted Advisor. Additionally, you can have them automatically deleted by setting up [the stop and start capability](#) of Amazon RDS databases.



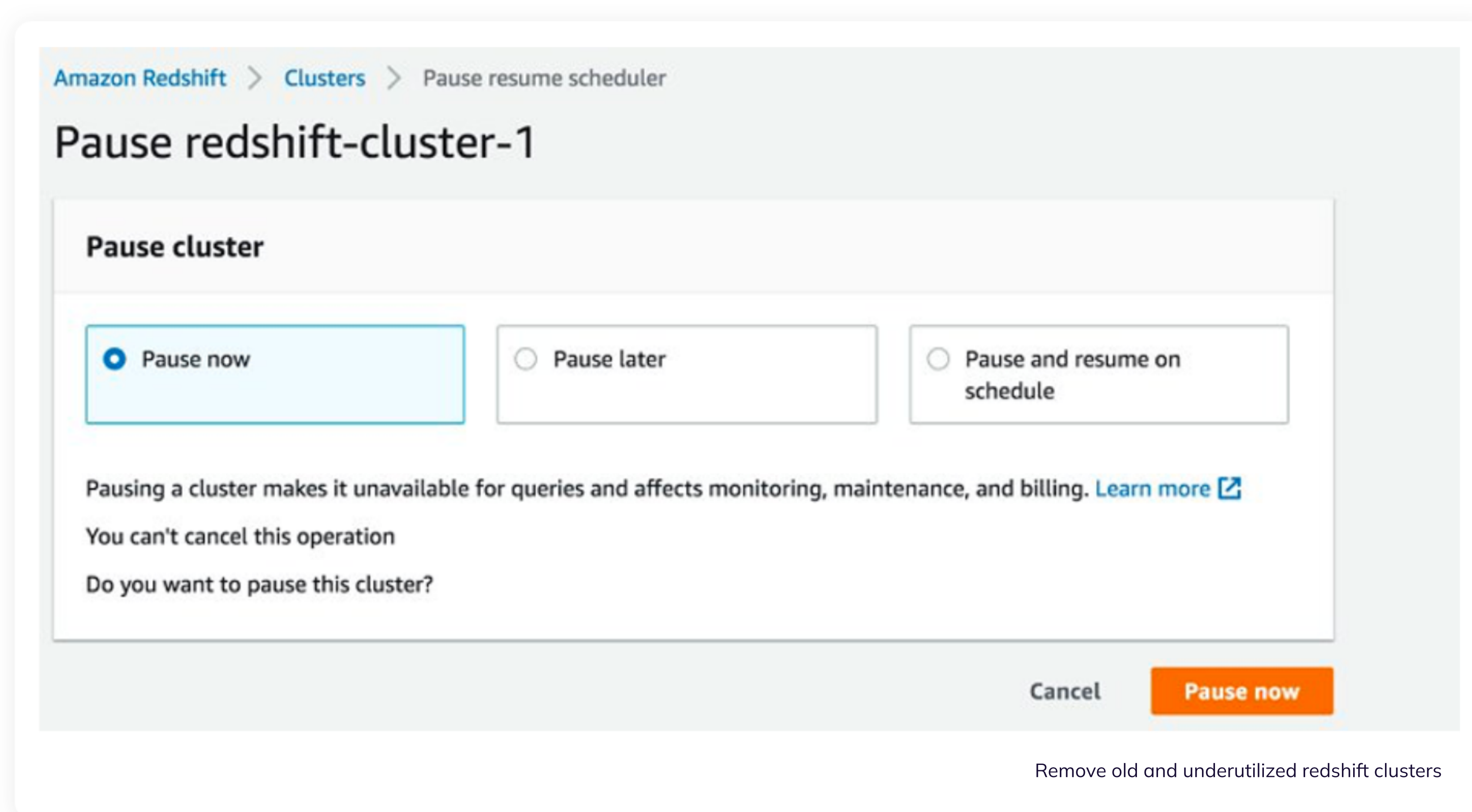
RDS idle DB instance check

### Stop paying for idle Redshift clusters

It's easy to overprovision Redshift clusters or leave them running during the evenings, weekends, and holidays causing you to pay for Redshift nodes when they are idle. Instead, it's possible to resize, pause and later resume nodes in line with your fluctuating needs. This equally applies to ElastiCache and Elasticsearch.

### Remove old Redshift clusters

It is recommended to delete Redshift clusters that have had no connection for over seven days and less than 5% cluster-wide average CPU utilization for 99% of the last seven days. To put a stop to the cost generated by these underutilized clusters you can identify them by doing a [Redshift clusters check](#) and then pause them to suspend their compute and still retain the underlying data structures. This can be configured on the Amazon Redshift console or CLIs.



Remove old and underutilized redshift clusters



## Monitor your DynamoDB usage

It's easy for DynamoDB usage to fluctuate and have costs spiral out of control. There are two metrics for analyzing usage; read capacity using ConsumedReadCapacityUnits and write capacity using ConsumedWriteCapacityUnits. Once you've identified general usage patterns, it is recommended to put your steady-state usage on a discount savings program. Only use On-demand to pay per request for data reads and writes your applications perform as the workload ramps up or down. This ensures you only pay for what you use without the risk of overprovisioning capacity.

### Edit read/write capacity

**Capacity mode** [Info](#)

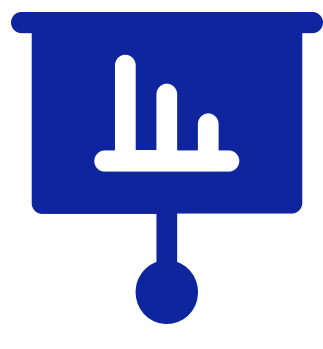
**Provisioned**  
Manage and optimize your costs by allocating read/write capacity in advance.

**On-demand**  
Simplify billing by paying for the actual reads and writes your application performs.

Cancel **Save changes**

**Zesty** is breaking new ground with intelligent cloud management technology that auto-scales cloud resources to fit real-time application needs.

As today's cloud environments become increasingly dynamic, Zesty enhances cloud efficiency, improves DevOps productivity, and reduces cloud costs with zero human input. As a result, DevOps engineers can spend less time on manual cloud infrastructure tasks and can enjoy the cloud's flexibility and scalability without worrying about cost or maintenance concerns. Zesty was founded in 2019 in Tel Aviv and is used by leading organizations such as Armis, Gong, GrubHub, Heap, and others. For more information, visit [Zesty.co](https://zesty.co).



# Compute

## Always, but always, tag

Aside from being a cornerstone tool to support cost optimization, tagging is needed to identify numerous resources you might have, allowing you to collect metrics based on the varied purposes of those applications. Tagging supports the cost allocation of resources, providing insight into the costs generated by each instance, supporting chargeback and payback, and alerting regarding budgets that are about to be exceeded. A good way to start implementing a tagging strategy is to define a tagging dictionary, where you define the “rules of the game” of how every resource should be labeled.

**Tagging Strategy**

- app
- app-component
- env

*Technical*

- owner
- cost-center
- project
- environment

*Finance*

- Start-time
- stop-time

*Ops/Automation*

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Confidential and Trademark.

Apply tags per instance in Cost Explorer

## Indicate tags for cost allocation

Tags should be activated for cost allocation to indicate to AWS that the associated cost data should be made available throughout the billing pipeline. Once activated, cost-allocated tags can be used to group or filter resources in Cost Explorer and to refine AWS budget criteria.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

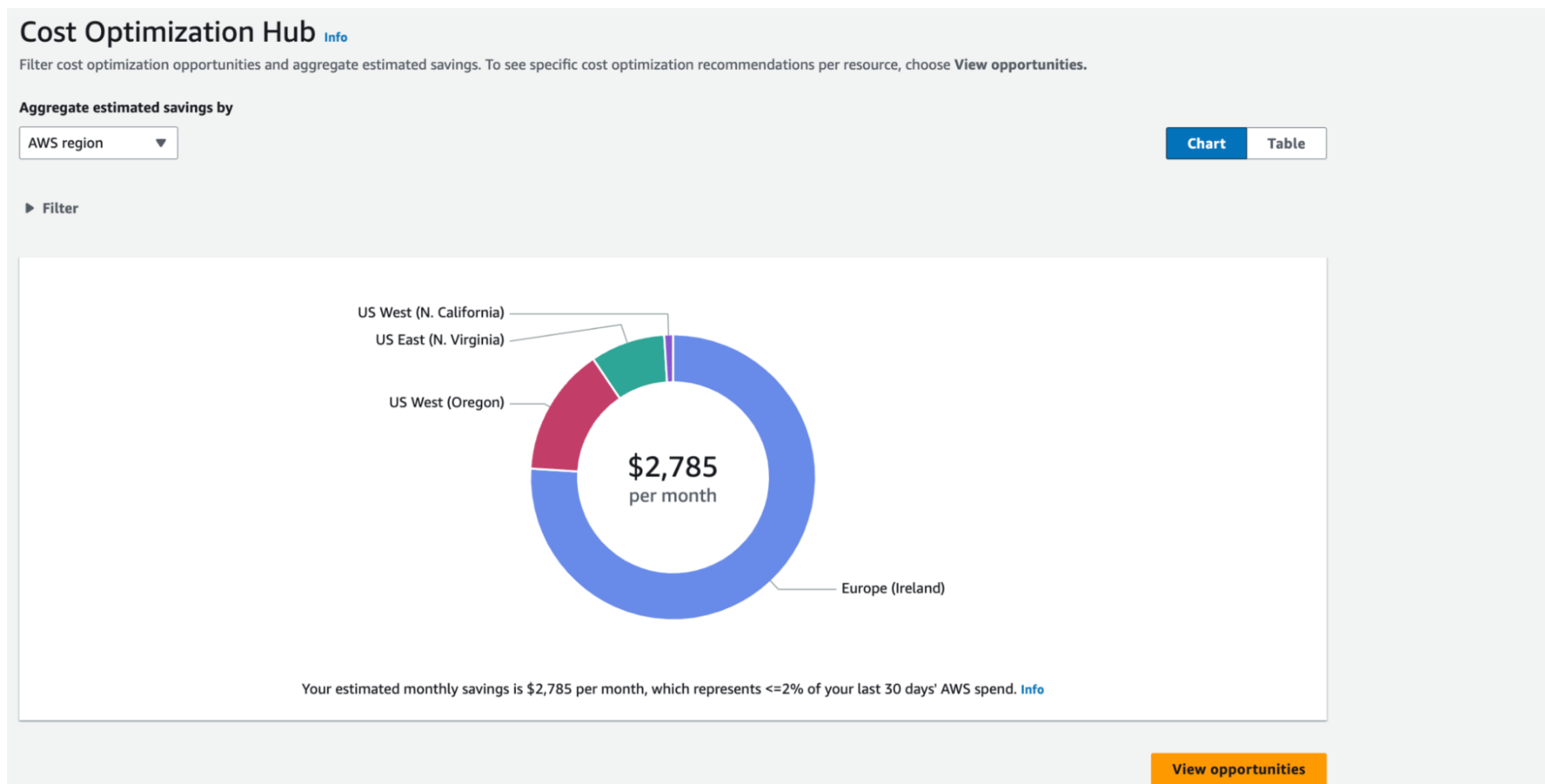
Key (127 characters maximum)	Value (255 characters maximum)	Instances	Volumes	
Name	leBlogTestInstance	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

**Add another tag** (Up to 50 tags maximum)



## Check optimization opportunities using the AWS Cost Optimization Hub

Using the Cost Optimization Hub, you can identify, filter, and consolidate over 15 types of AWS cost optimization recommendations, such as rightsizing your EC2 instances and your Lambda function or upgrading your EBS volume type. You can also find recommendations related to Graviton migration, idle resource detection, or Savings Plans. To help you compare and prioritize the recommendations, the Cost Optimization Hub provides an estimation of the potential savings, taking into account your specific AWS discount plans.



**Savings opportunities** [Info](#)

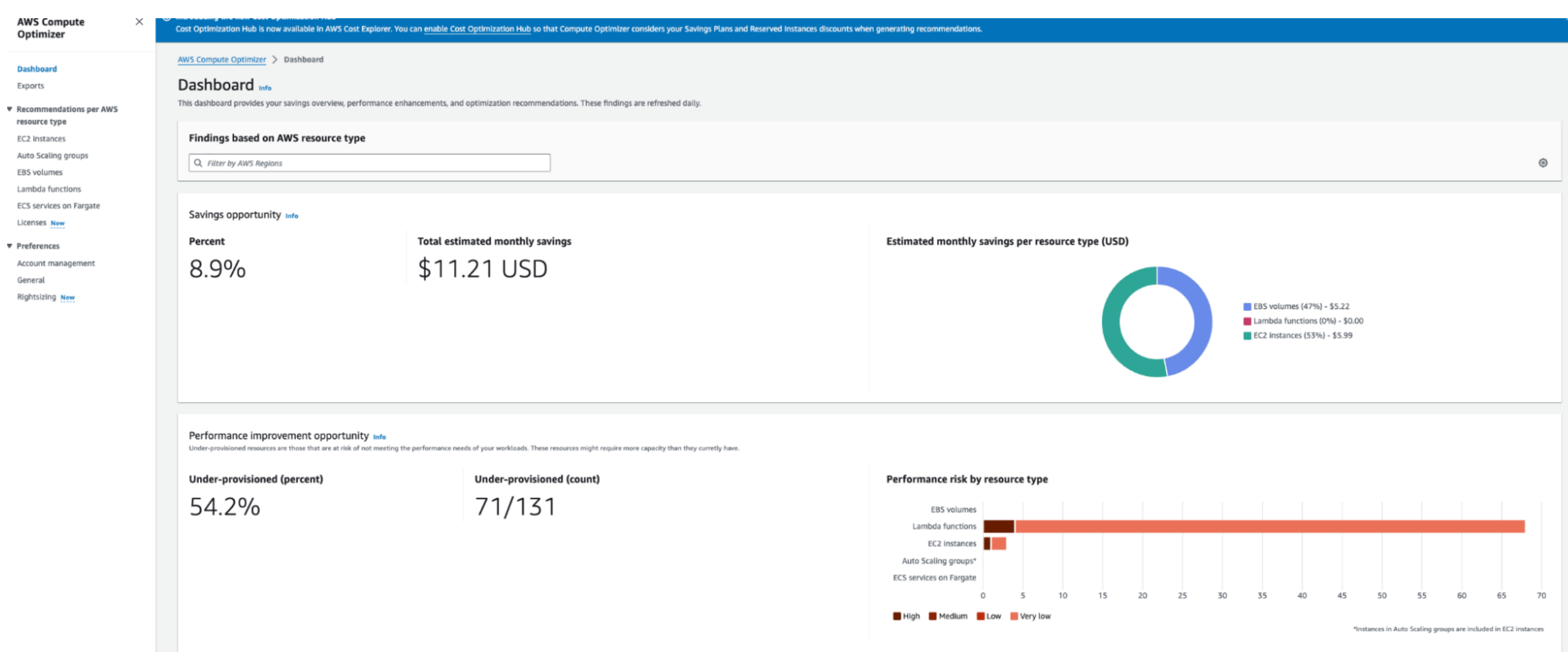
Explore your savings opportunities using the filters below.

**Resources with estimated savings** Group related recommendations

Filter distributions by text, property or value

	Estimated monthly savings ▲	Resource type ▼	Resource ID ▼	Top recommended action ▼	Current resource summary	Recommended r
<input type="radio"/>	\$865.52	RDS Reserved Instances	-	Purchase Reserved Instances (Reserved Node)	-	104 db.t3.micro f
<input type="radio"/>	\$473.05	RDS Reserved Instances	-	Purchase Reserved Instances (Reserved Node)	-	2 db.r5.large Pos
<input type="radio"/>	\$315.91	RDS Reserved Instances	-	Purchase Reserved Instances (Reserved Node)	-	2 db.m5.large Po
<input type="radio"/>	\$283.92	RDS Reserved Instances	-	Purchase Reserved Instances (Reserved Node)	-	8 db.t3.medium /
<input type="radio"/>	\$271.33	RDS Reserved Instances	-	Purchase Reserved Instances (Reserved Node)	-	8 db.t3.medium /
<input type="radio"/>	\$237.74	ElastiCache Reserved Nodes	-	Purchase Reserved Instances (Reserved Node)	-	2 cache.m6g.xlan
<input type="radio"/>	\$138.36	RDS Reserved Instances	-	Purchase Reserved Instances (Reserved Node)	-	20 db.t3.micro Pc
<input type="radio"/>	\$96.68	ElastiCache Reserved Nodes	-	Purchase Reserved Instances (Reserved Node)	-	1 cache.r5.large i
<input type="radio"/>	\$86.30	ElastiCache Reserved Nodes	-	Purchase Reserved Instances (Reserved Node)	-	6 cache.t3.small i
<input type="radio"/>	\$28.88	ElastiCache Reserved Nodes	-	Purchase Reserved Instances (Reserved Node)	-	4 cache.t3.micro
<input type="radio"/>	\$27.26	ElastiCache Reserved Nodes	-	Purchase Reserved Instances (Reserved Node)	-	2 cache.t3.small i
<input type="radio"/>	\$6.79	ElastiCache Reserved Nodes	-	Purchase Reserved Instances (Reserved Node)	-	1 cache.t3.micro
<input type="radio"/>	\$5.23	EBS volume	-	Rightsize	8.0 GB Storage/4000.0 IOPS/125.0 MB/s Throughput	8.0 GB Storage/3
<input type="radio"/>	\$0.67	EBS volume	-	Upgrade	gp2	gp3

Later, we recommend you to enter the AWS Compute Optimizer which provides you with a better granularity of EC2, Lambda, ECS, Fargate improvement opportunities:





## Shutdown idle EC2 instances

EC2 instances that are idle or have low utilization are still costing you money by the hour. To identify and remove these instances, use the Resource Optimization in Cost Explorer.

The screenshot displays the AWS Cost Explorer interface for Resource Optimization. At the top, it shows 3 optimization opportunities, an estimated monthly savings of \$110, and an estimated savings percentage of 50.00%. Below this, a summary states that 3 instances have been identified as idle and underutilized based on the last 14 days, with an estimated \$110 monthly savings (50.00% of EC2 On-Demand instance costs). A 'Download CSV' button is available. The main table lists three recommendations:

Recommendation	Instance ID	Account ID	Tag(s)	CPU (%)	Monthly estimated savings	Action
Modify instance	i-0b18d304a1...	AWS Insights Demo...	3	6.6%	\$72	<a href="#">View</a>
Modify instance	i-0196e32825...	AWS Insights Demo...	2	4.0%	\$33	<a href="#">View</a>
Modify instance	i-0a9909f442...	AWS Insights Demo...	2	7.5%	\$4	<a href="#">View</a>

Additional filters on the right include 'Show recommendations for' (Idle instances, Underutilized instances), 'Additional Filters' (Linked Account, Region, Tag), and a 'Viewing 1 to 3 of 3 recommendations' indicator.

Recommendations for idle and underutilized instances

## Pause instances when they're not needed

For instances that you want to keep, but are only used intermittently, you can stop or pause these instances when they're not needed using the AWS instance scheduler.

## Tune your EC2 autoscaling groups configuration

The auto-scaling group enables you to expand or shrink your EC2 fleet based on demand.

To improve cost efficiency, the scaling policy can be tuned to add instances less aggressively.

It can also be tuned to set a lower minimum for the number of instances that are needed to serve end-user requests.

The screenshot shows the AWS Config console 'Resource inventory' page. The left sidebar contains navigation options like 'Dashboard', 'Rules', 'Resources', and 'Settings'. The main content area shows search filters for 'Resources' (AutoScaling: AutoScalingGroup) and 'Tag'. A 'Look up' button is present. Below the search filters, there is a table with columns for 'Resource type', 'Config timeline', 'Compliance', and 'Manage resource'. The table lists one resource: 'AutoScaling AutoScalingGroup' with 'AS- APP1 - AS group' as the config timeline. The 'Manage resource' button for this resource is circled in orange.

Change configurations to auto-scaling groups



## Rightsize instances

There is a tendency to overprovision instances with more memory and CPU than what's actually needed. To check whether your instances exceed your needs, you can use Cost Explorer which gives recommendations for downsizing within or across instance families, upsizing recommendations to remove performance bottlenecks, and recommendations for EC2 instances that are part of an Auto Scaling group.

**49.81%**  
Estimated savings (%)

n these instances could help you save an estimated \$58.69 monthly

[Download CSV](#)

Tag(s)	CPU (%)	Monthly estimated savings	
1	1.3%	\$17.08	<a href="#">View</a>
2	2.6%	\$8.32	<a href="#">View</a>
6	1.8%	\$8.32	<a href="#">View</a>
3	1.7%	\$8.32	<a href="#">View</a>

**Recommendation parameters**

**Generate recommendations**

- Within the same instance families
- Across instance families

**Resource optimization types**

- Idle instances
- Underutilized instances

**Additional Filters**

Linked Account [Include all](#)

Region [Include all](#)

Tag [Include all](#)

**Advanced options**

- Include Savings Plans and Reserved Instances

Recommendations for idle and underutilized instances

## Consider using Spot instances

For stateless, fault-tolerant, and loosely coupled workloads, consider using Spot instances which can give discounts of up to 90% off On-Demand costs but can be reclaimed with just a two-minute warning if AWS needs these instances back. Spot instances are ideal for test and development workloads such as CI/CD, and high-performance computing.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

**Step 3: Configure Instance Details**  
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances  [Launch into Auto Scaling Group](#)

**Purchasing option**  Request Spot instances

**Current price**

Availability Zone	Current price
us-west-2a	\$0.2722
us-west-2b	\$0.306
us-west-2c	\$0.2763
us-west-2d	\$0.3117

**Maximum price** \$

**Persistent request**  Persistent request

**Launch group**

**Request valid from**  [Edit](#)

Select Spot Instances when configuring insurance details



## Apply Savings Plans to Fargate and Lambda costs

For these compute operations, Savings Plans can be applied to provide a discount of up to 17% from On-Demand costs.

The screenshot shows the AWS Cost Management console's 'Purchase Recommendations' page for Savings Plans. The interface includes a navigation sidebar on the left with options like Home, Cost Explorer, Budgets, and Savings Plans. The main content area displays 'Recommendation options' with filters for Savings Plans type (Compute selected), term (1-year selected), payment option (No upfront selected), and based-on-the-past (30 days selected). Below this, a recommendation is shown: 'Purchase a Compute Savings Plan at a commitment of \$0.10/hour'. A summary states: 'You could save an estimated \$29 monthly by purchasing the recommended Compute Savings Plan.' A comparison table shows the following data:

Before recommended purchase	After recommended purchase (based on your past 30 days of usage)	
Monthly On-Demand spend	Estimated monthly spend	Estimated monthly savings
<b>\$177</b> (\$0.24/hour)	<b>\$148</b> (\$0.20/hour)	<b>\$29</b> (\$0.04/hour)
Your estimated On-Demand spend based on your usage over the past 30 days (including all active Savings Plans)	Your recommended \$0.10/hour Savings Plans commitment + an average \$0.10/hour of On-Demand spend	16% monthly savings over On-Demand \$177 - \$148 = \$29

## Migrate to Graviton Instances (where you can)

The use of AWS on ARM processing technology is much more efficient and powerful at running servers, requiring just 60% less electricity to run just one server. This entails that for the same cost in power that it takes to run one intel CPU, AWS can run two ARM servers. These ARM servers can be used by selecting Graviton instance types. Graviton processors will reduce your EC2 instance bill by 40% yet still achieve the same level of performance. The challenge is that ARM servers are not similar to Intel processes and the two technologies are not compatible, making it difficult to migrate over. To do so, you need to rewrite and compile your code. While not always a feasible solution, when Graviton can be utilized it can be enormously cost-effective. Graviton can be a good fit for managed services, with users often starting with OpenSearch and RDS. Furthermore, by using Graviton 3 which was introduced at re:invent 2022, you can add another 25% on top of the 40% discount of Graviton 2, compared to the cost of an intel-based M5.

The screenshot shows the AWS Compute Optimizer 'Recommendations for EC2 instances' page. It features a 'CPU architecture preference' dropdown set to 'Current, Graviton (aws-ar...)'. Below this, a table lists recommendations for three instances:

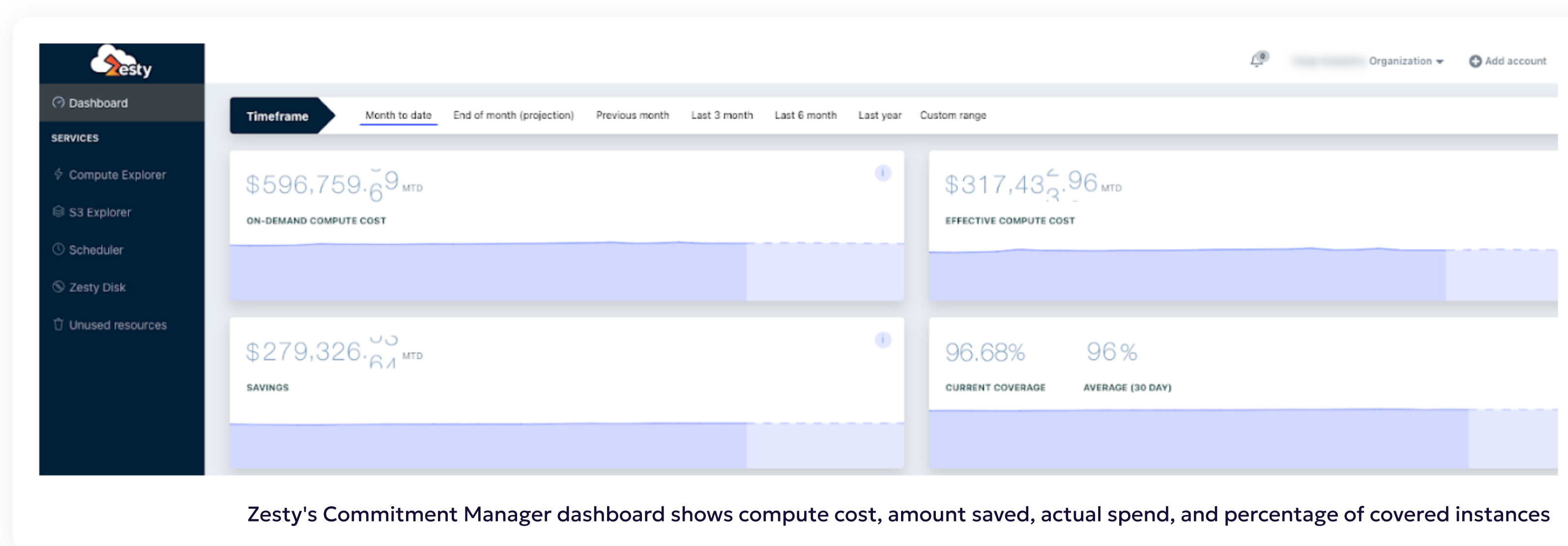
Instance ID	Finding Info	Current instance type	Current On-Demand price	Recommended instance type	Recommended On-Demand price	Migration effort	Inferred workload types
i-0b5ec1bb9daabf0f3	Under-provisioned	r5.large	\$0.1260 per hour	r6g.large	\$0.1008 per hour	Low	Apache Hadoop
i-033868420bdc7d29a	Over-provisioned	c5.2xlarge	\$0.3400 per hour	r6g.large	\$0.1008 per hour	Medium	-
i-0bc9a76f2ed1a5e75	Over-provisioned	i3.2xlarge	\$0.6240 per hour	i3.xlarge	\$0.3120 per hour	Very Low	-

Select Graviton instances in AWS Compute Optimizer



## Purchase 1 or 3 Year Discount Plans

If you're in a position where you have a fairly stable compute workload and can forecast the bulk of your usage a year in advance, or even three years in advance, then it is highly worthwhile to purchase Reserved Instances and Savings Plans commitments that will deliver substantial discounts from the cost of your EC2 On-Demand instances. The challenge is all too few of us have a workload that is so stable that you're able to predict what your EC2 usage will be next month, let alone, for the following year or three! In this situation, you may want to consider [Zesty's Commitment Manager](#). The solution automatically manages your discounted commitments for you, making it possible to leverage AWS's deepest discounts, without taking on the risk of over-committing. Commitment Manager saves users up to 50% of their EC2 workload. To find out more, [read the solution overview](#).



While some of these recommendations will be easy to implement, there will be many others that will require time, effort, and even a holistic cultural change to organizational processes. Towards that end, you may want to set up a CCoE (Cloud Center of Excellence) with representatives from different departments, be they DevOps, Finance, Procurement, and Executives, that will work as a steering committee to implement many of these FinOps best practices.

As a rule of thumb, seek to automate wherever you can, as that will deliver cost savings without adding any manual effort, and will continue to scale as your business grows. For the rest, we wish you the best of luck on your cost optimization journey!

**Zesty** is breaking new ground with intelligent cloud management technology that auto-scales cloud resources to fit real-time application needs.

As today's cloud environments become increasingly dynamic, Zesty enhances cloud efficiency, improves DevOps productivity, and reduces cloud costs with zero human input. As a result, DevOps engineers can spend less time on manual cloud infrastructure tasks and can enjoy the cloud's flexibility and scalability without worrying about cost or maintenance concerns. Zesty was founded in 2019 in Tel Aviv and is used by leading organizations such as Armis, Gong, GrubHub, Heap, and others. For more information, visit [Zesty.co](https://zesty.co).