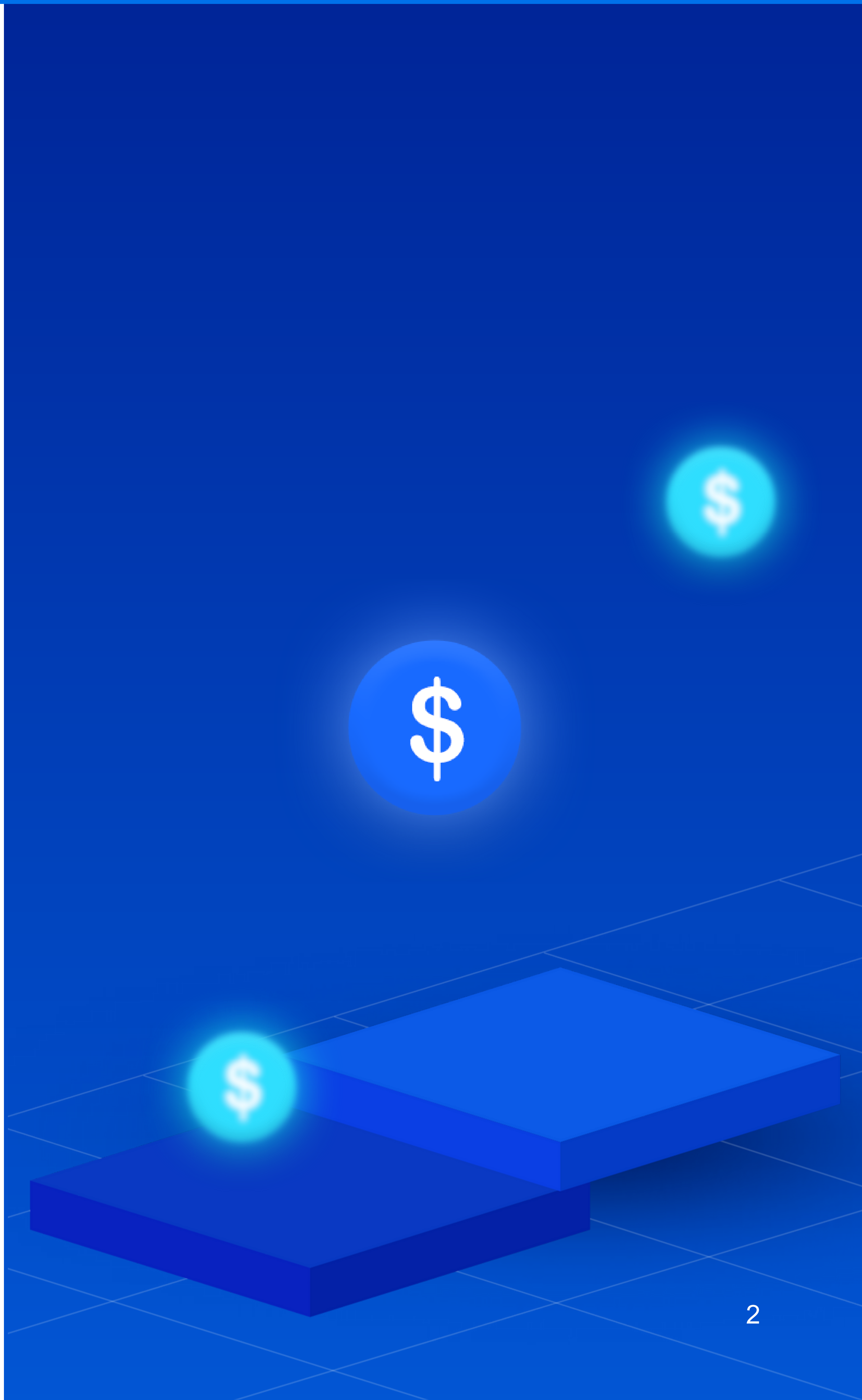
EBOOK

ECS Cost Optimization: The Complete Guide

www.nops.io

Table of Contents



Introduction

Many organizations are currently overspending on the cloud,[1] with EC2 likely the most expensive item on your AWS bill. AWS ECS (Elastic Container Service) is the most widely-used EC2 container orchestration service. However, it presents some unique[1] challenges for cost-optimization — particularly if you're using advanced strategies like running on Spot.

That's why we wrote this highly comprehensive guide, based on the insights we learned from managing $1.5 billion in AWS spend. We'll cover everything you need to know about pricing strategies, best practices for scaling, and implementing ECS cost-optimization strategies.

This practical guide will focus on actionable tips, how-to guides, screenshots, and the other critical info you need to start saving on your ECS costs.

1 " *Greene, T. (2023). The hidden costs of cloud and where to find overspending. Forbes. Retrieved Feb 1 2024, from* https://www.forbes.com/sites/forbestechcouncil/2023/01/19/the-hidden-costs-of-cloud-and-where-to-find-overspending

ECS: The Basics

01. What is AWS ECS?

AWS Elastic Container Service (ECS) is a highly scalable, high-performance container management service that supports Docker containers and allows you to run applications on a managed cluster of Amazon EC2 instances easily. ECS eliminates the need to install, operate, and scale your cluster management infrastructure.

02. What are the key components of AWS ECS?

Clusters

A cluster is a collection of EC2 instances that together run containerized applications. Each cluster defines the computing environment for its containers.

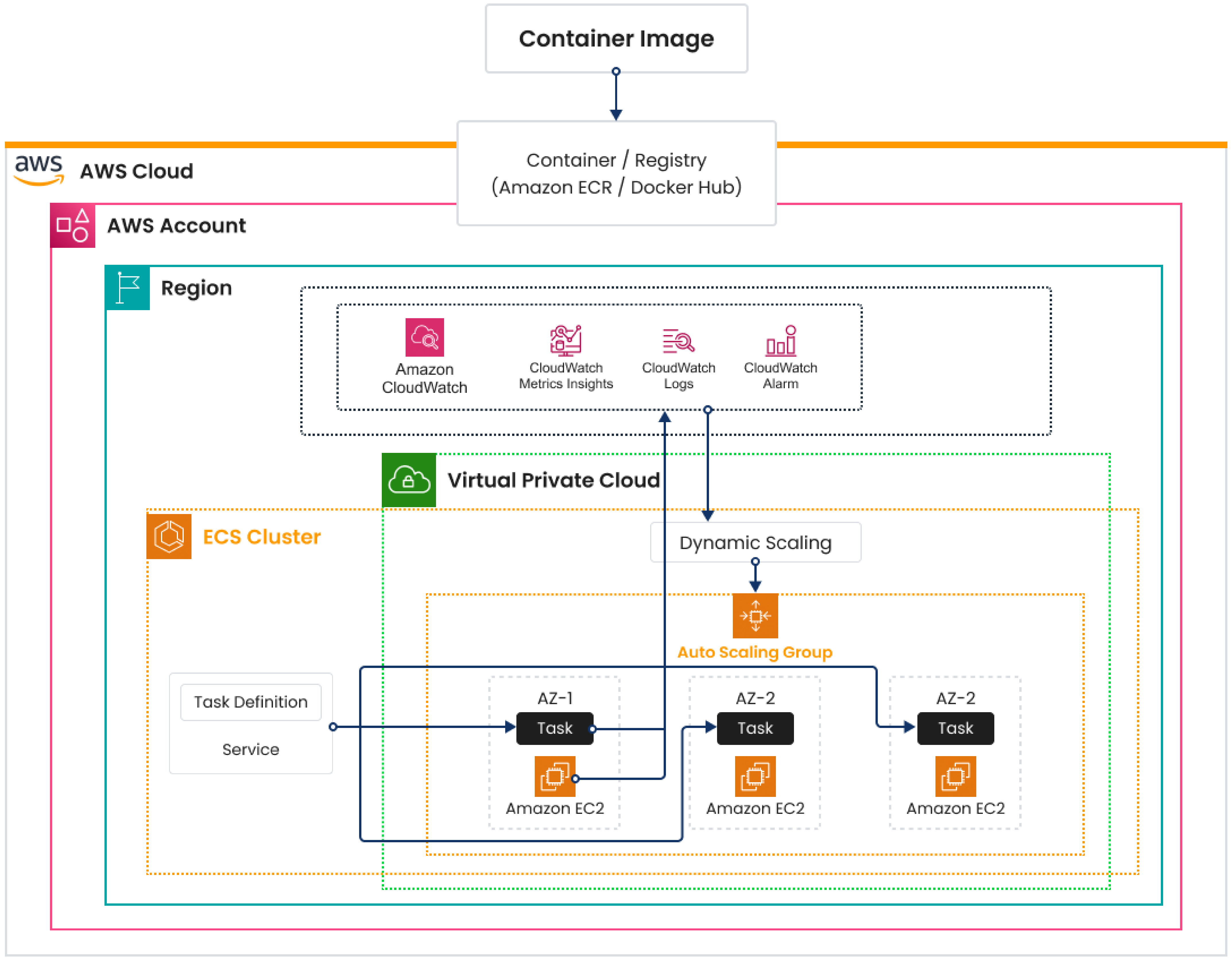
Tasks and Services

- Task Definitions: These are blueprints for your application that define how containers should be deployed. They include container definitions, volumes, and networking information.
- Services: They maintain a specified number of instances of a task definition simultaneously in a cluster. If a container instance fails, the service scheduler launches another instance to replace it, maintaining the desired instance count.

Container Agent

A piece of software that runs on each container instance within a cluster. It sends information about the instance's current running tasks and resource utilization to ECS, and starts and stops tasks whenever it receives a request from ECS.

ECS architecture diagram illustrating the relationship between Clusters, Task Definitions, and ECS Scheduler.



03. ECS vs. Traditional Container Management

Let’s briefly outline the difference in approach between ECS and traditional container management.

Feature	AWS ECS	Traditional Container Management
Setup and Scaling	Automates and simplifies container orchestration, easy to scale based on demand	Manual — meaning scaling can be complex and resource-intensive
Resource Optimization	Optimized for AWS resources — high-density container packing for efficiency	Depends on underlying infrastructure
AWS Integration	Deep integration with AWS services like IAM for security, VPC for networking, CloudWatch for logging and monitoring	Requires third-party tools for integration

04. ECS vs. EKS: A detailed comparison of benefits & costs

While both AWS Elastic Container Service (ECS) and Elastic Kubernetes Service (EKS) offer robust solutions for container orchestration, they differ in terms of architecture, ease of use, scalability, and integration capabilities. Let’s compare.

i. Architectural comparison

ECS Architecture	EKS Architecture
Designed as a proprietary AWS solution.	Based on Kubernetes, an open-source system
Integrates deeply with other AWS services.	Provides more flexibility with community-driven plugins and tools
Offers a simpler control plane and API.	Complex control plane that requires more Kubernetes expertise
Architecture focuses on tasks and services.	Centers around pods, deployments, and services

ii. Feature comparison

Feature/Aspect	AWS ECS	AWS EKS
Architecture	Proprietary AWS, simpler setup	Kubernetes-based, more components to manage
Ease of Use	More straightforward to set up and manage, with a simpler user interface and fewer components to manage. User-friendly, particularly for AWS users.	Steeper learning curve, requiring a deeper understanding of Kubernetes. However, it offers more flexibility and a broader range of features for complex deployments.
Scalability	Integrated with Fargate for serverless scaling Kubernetes	Kubernetes auto-scaling, more granular control
Security	Integrated with AWS IAM to benefit from AWS built-in security features	Leverages Kubernetes role-based access control (RBAC), community-driven but equally robust security features
Cost	No additional charge for ECS, pay for resources (e.g. EC2 instances or Fargate) used	Charges for EKS control plane plus resources used

The takeaway is that ECS offers a more AWS-centric experience, making it a good choice for teams looking for ease of use and deep integration with AWS services. On the other hand, EKS is ideal for teams that require the flexibility and extensive feature set of Kubernetes, along with its wide community support.

ECS pricing

ECS pricing

The first step is understanding how your ECS costs are generated and billed. There are four main billing models for ECS.

01. ECS launch type

There is no additional cost for using ECS on AWS, beyond the regular cost of the instances used (based on EBS storage volumes, CPU, memory, etc.), as well as any additional AWS services integrated with ECS, like Elastic Load Balancing (ELB), EBS volumes, and data transfer costs. There are 4 common types of EC2 instance pricing:

On-Demand Instances

Let you pay for compute capacity by the hour or second (minimum of 60 seconds) with no long-term commitments. This frees you from the costs and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into much smaller variable costs.

Spot Instances

These are spare AWS capacity that users can purchase at a heavy discount (up to 90%), but AWS may terminate these instances at any time with a 2-minute warning.

Reserved Instances

(RIs) pricing provides significant savings in exchange for a commitment of up to three years. However, it can be difficult to forecast how much compute you will need, and you will have to pay for commitments regardless of actual usage. They are ideal for steady, predictable usage.

Saving Plans

(SPs) are more flexible than RIs, as they automatically apply to EC2 instance usage regardless of instance family, size, AZ, Region, operating system, or tenancy, and also apply to Fargate and Lambda usage. Discounts can be slightly lower than RIs.

02. Fargate

There are no upfront costs and you pay only for the resources you use. AWS Fargate pricing is calculated based on the vCPU, memory, Operating Systems, CPU Architecture, and storage resources used from the time you start to download your container image until the Amazon ECS Task terminates, rounded up to the nearest second. As with normal EC2 instances, you can use Fargate with AWS RI & SP commitments or Spot.

03. ECS Anywhere

Lets you run and manage container workloads on your infrastructure. The pricing model is based on the resources consumed in your on-premises environment, such as compute and memory capacity. Use cases include meeting compliance requirements or scaling your business without sacrificing your on-premises investments. You pay $0.01025 per hour for each managed ECS Anywhere on-premises instance.

04. AWS Outposts

Fully managed service that extends AWS infrastructure, services, APIs, and tools to customer premises. By providing local access to AWS managed infrastructure, AWS Outposts enables customers to build and run applications on-premises using the same programming interfaces as in AWS Regions, while using local compute and storage resources for lower latency and local data processing needs.

Key Strategies for ECS Cost Optimization

01. Monitoring

The first step in controlling costs is understanding your cloud environment. Let's discuss the essential tools and strategies for monitoring ECS environments and best practices for managing them efficiently.

The most common ECS monitoring tools include:

AWS CloudWatch:

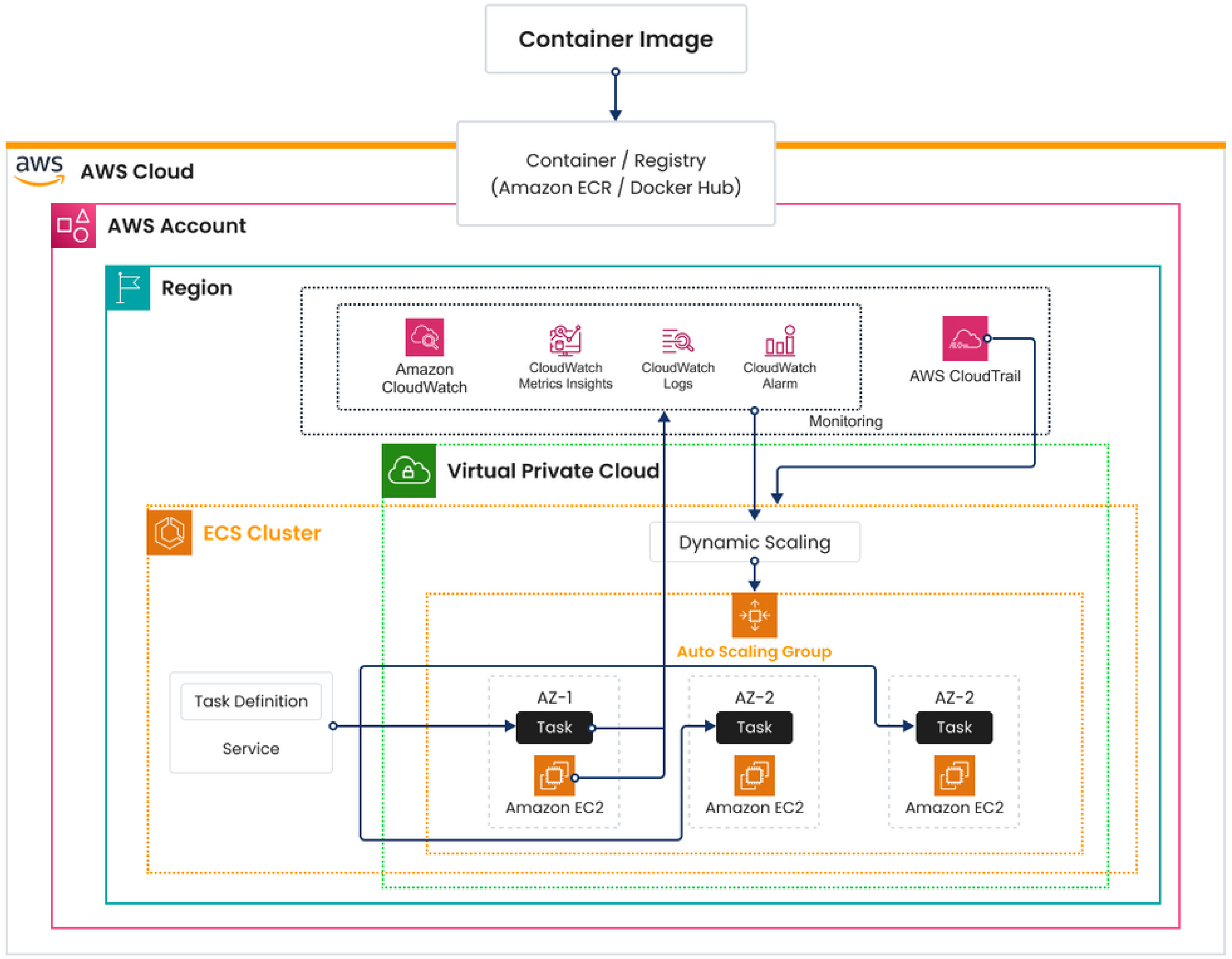
CloudWatch Provides monitoring and management for AWS cloud resources and the applications running on AWS. Includes logs, metrics, alarms, and events. Direct integration with ECS for monitoring CPU and memory utilization, network, and disk metrics.

AWS CloudTrail:

AWS CloudTrail records AWS API calls for your account. Provides a history of AWS API calls for your account, including calls which are useful for auditing changes in ECS services and tasks.

Third-Party Monitoring Tools:

Tools like Datadog, New Relic, and Prometheus can integrate with ECS through APIs for additional monitoring capabilities.



Monitoring tools shown in the ECS architecture

Aspect	Best Practice	Tools/Strategies Used
Logging and Analysis	Aggregate and analyze container logs	CloudWatch Logs, ELK stack
Alarms and Notifications	Set up alerts for critical metrics to monitor and control AWS costs; receive alerts when spending exceeds predefined thresholds.	CloudWatch Alarms, SNS
Cluster Health	Monitor cluster resources and task health	CloudWatch Metrics
Service and Task Management	Optimize service definitions and tasks, implement version control and change management processes	ECS Console, AWS CLI
Security Management	Regularly audit and enforce AWS IAM policies and compliance	IAM, AWS Config
Resource Optimization	Rightsize resources based on performance data, use cost-saving strategies such as leveraging Spot instances	CloudWatch, ECS Capacity Providers

It's worth noting that proper tagging of your EC2 resources is a prerequisite for proper monitoring. The benefits of a robust tagging strategy are to:

- Improve visibility into your workloads
- Attribute shared costs to individual teams, projects, departments, environments, and business units
- Programmatically manage infrastructure
- Identify underutilized resources
- Investigate spikes in usage and trace them back to specific causes

Tip: Use Container-Level Metrics

Implement detailed container-level monitoring to gain insights into specific containers or services driving costs. Third-party tools can help you filter spending by resource type, compute type, tags, and other dimensions to investigate spikes and the biggest drivers of your cloud costs. For more information on allocating your AWS costs, you can download the complete guide.

02. Right-Sizing ECS Resources

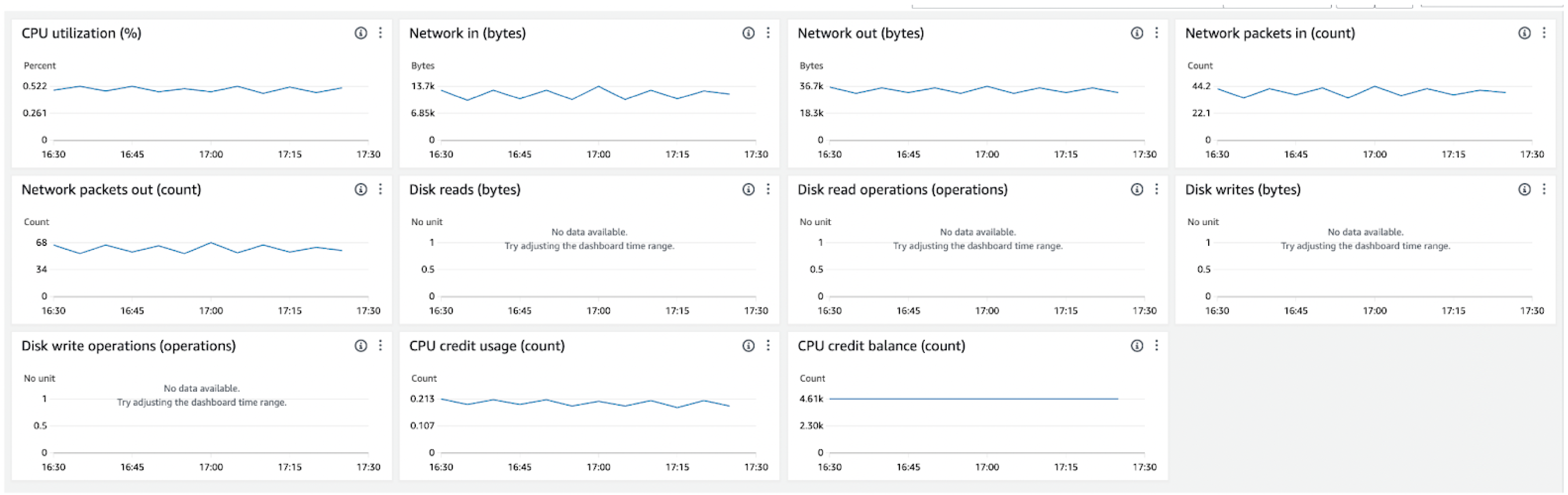
By analyzing historical usage and performance, you can identify and rightsize instances that are not consuming all the resources currently available to them. Keep in mind:

Rightsize regularly.

As your dynamic usage changes, regularly assess and adjust the ECS task definitions to ensure that the allocated CPU and memory are in line with the actual usage. Over-provisioning leads to unnecessary costs, whereas under-provisioning can affect performance.

Utilization Metrics.

The key to safe rightsizing is data. Use AWS CloudWatch, Datadog, another third-party tool or a custom engineering solution to monitor resource utilization and identify optimization opportunities. AWS's general rule for EC2 instances is that if your maximum CPU and memory usage is less than 40% over a four-week period, you can safely reduce capacity.



CloudWatch metrics used for rightsizing

For extensive and specific information and best practices on how to rightsize with CloudWatch, Datadog or other metrics, check out this free ebook.

03. Effective Auto-Scaling

ECS auto scaling involves automatically adjusting the number of running ECS tasks in response to configured CloudWatch alarms. You can also scale the underlying EC2 instances based on demand, ensuring that there are always enough resources to run tasks without over-provisioning.

AWS ECS supports various scaling strategies to ensure that applications can handle varying loads efficiently: Horizontal, Vertical, Scheduled, Dynamic, and Predictive Scaling. Let’s talk about the differences and how to use them.

General Autoscaling Strategies

Type	Definition	Best suited for	Best practices
Horizontal Scaling	Increasing or decreasing the number of container instances or tasks to handle the load.	Applications with variable workloads, where the load can increase or decrease unpredictably	Auto-scaling policies based on metrics
Vertical Scaling	Changing the compute (CPU, memory) capacity of existing instances.	Applications with predictable, stable workloads that occasionally require more resources	Resize instances based on steady trends
Scheduled Scaling	Scaling actions are automatically triggered based on a specified schedule.	Workloads that have predictable load changes, like batch processing jobs that run at specific times	Predefine scaling actions for known load patterns

Now let’s talk about some of the best practices for scaling effectively.

Implement ECS Service Auto Scaling

Use ECS service auto-scaling to adjust the number of tasks automatically based on demand. Leverage EC2 to manage the scaling of the underlying instances. Ensure instances have the necessary resources to support the maximum number of tasks you expect to run. Define scaling policies based on CloudWatch metrics such as CPU and memory utilization.

Dynamic Scaling

Dynamic Scaling adjusts the number of active tasks in an ECS service automatically in response to real-time changes in demand.

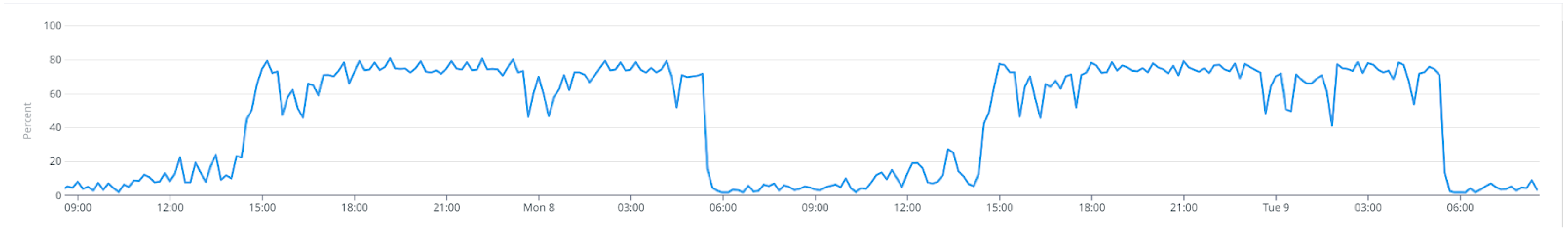
Here are the basic steps:

01. Identify the metrics most critical for your application's performance and stability. Common metrics include CPU utilization, network throughput, memory usage or custom application-specific metrics.

02. Establish appropriate minimum, maximum, and desired capacity settings for your ASGs. These should align with your anticipated workload demands, ensuring that the ASG can scale efficiently without over-provisioning.

For example, scale out (add instances) when CPU utilization exceeds a certain threshold, and scale in (remove instances) when the utilization decreases.

One strategy to consider is adjusting settings proactively in advance of predicted changes in demand, optimizing resource utilization and cost. We can also schedule the scaling based on the monitoring metrics where we can observe spikes and create the ASG to scale out during certain events or times of the week.

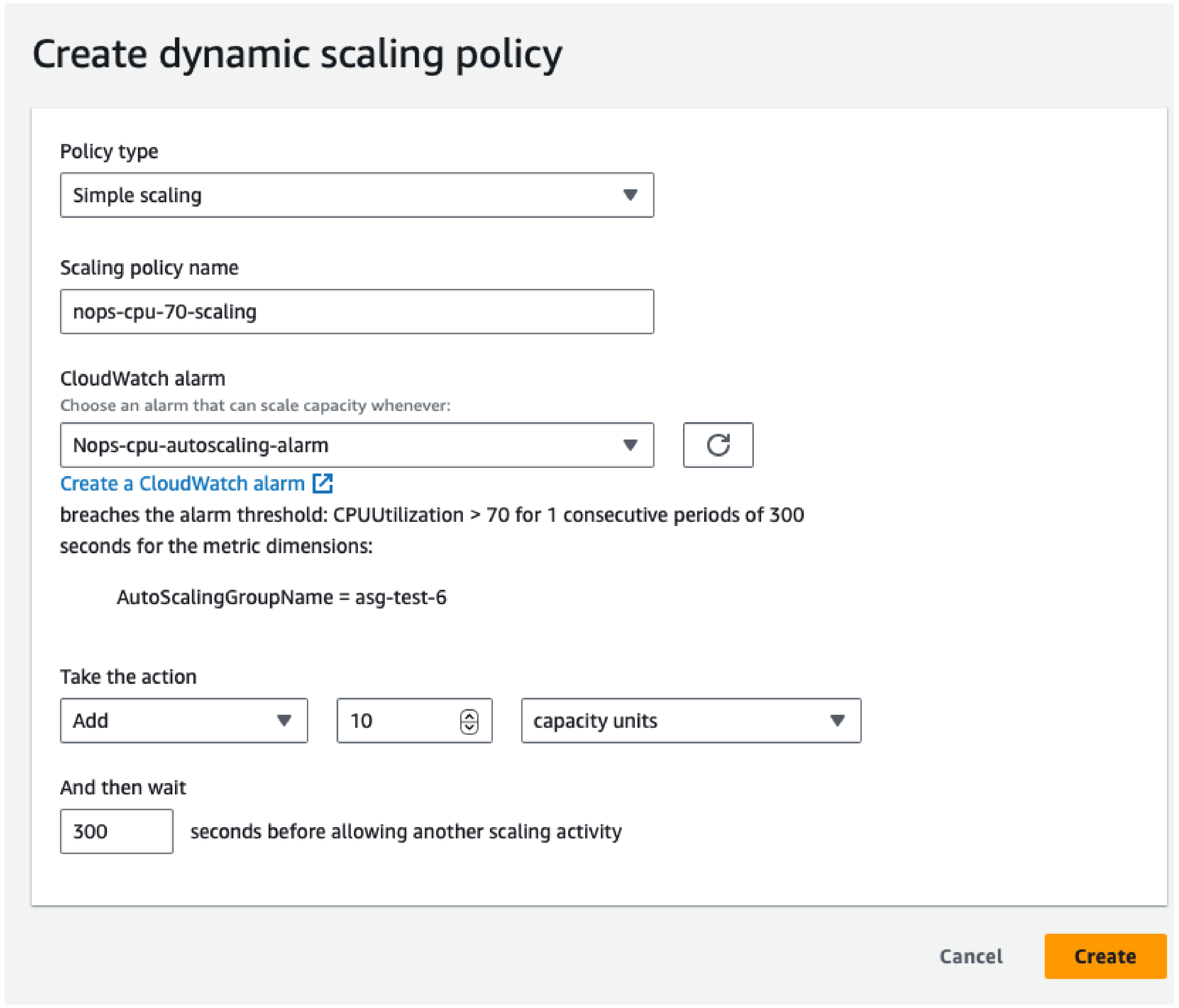


Cost-optimize this workload by scaling in during off-usage.

On the other hand, if your application has sporadic or unpredictable loads, consider using a mix of instances, possibly including burstable types (e.g. AWS T-series) that offer baseline performance with the ability to burst.

03. Create Dynamic Scaling Policies to properly utilize your resources and maintain cost-effective performance as your usage changes. In the AWS Management Console, go to the Auto Scaling service. Define policies for scaling out and scaling in based on the CloudWatch alarms. Configure your ASG to actively adjust its capacity in anticipation of idle or loading windows, i.e. periods of low and high demand. You'll need to regularly review historical data in CloudWatch to fine-tune your scaling policies continually.

04. Configure Scaling Actions: This may include adjusting the desired capacity, minimum and maximum instance counts, and cooldown periods that prevent rapid scaling events in response to fluctuations, ensuring stability.


Create dynamic scaling policy

Policy type

Simple scaling

Scaling policy name

nops-cpu-70-scaling

CloudWatch alarm

Choose an alarm that can scale capacity whenever:

Nops-cpu-autoscaling-alarm

Create a CloudWatch alarm

breaches the alarm threshold: CPUUtilization > 70 for 1 consecutive periods of 300 seconds for the metric dimensions:

AutoScalingGroupName = asg-test-6

Take the action

Add 10 capacity units

And then wait

300 seconds before allowing another scaling activity

Steps 3 & 4: Create dynamic scaling policies and configure scaling actions based on CloudWatch metrics

Predictive Scaling

Predictive scaling works by analyzing historical load data (e.g. via Amazon CloudWatch Integration) to detect daily or weekly patterns in traffic flows.

It uses this information to forecast future capacity needs, automatically scheduling scaling actions to proactively increase the capacity of your Auto Scaling group to match the anticipated load ahead of expected demand spikes.

Some best practices include:

1. Ensure sufficient historical data (typically at least 30 days) is available for accurate predictions.
2. Combine predictive scaling with dynamic scaling to handle unexpected surges in demand
3. Regularly review predictive scaling decisions and adjust as necessary based on actual demand patterns.

Let’s summarize the differences between the above strategies:

Scaling Type	Trigger	Best For	Considerations
Dynamic	Real-time metrics (e.g., CPU load)	Unpredictable workloads	Requires fine-tuning of metrics and thresholds
Predictive	Historical data analysis	Predictable, significant fluctuations in load	Depends on quality and quantity of historical data
Scheduled	Predefined times	Known events or maintenance windows	Must be manually set and adjusted as patterns change

Additional quick tips for effective scaling

Container packing.

Implement task placement strategies that optimize the packing of containers in each EC2 instance to maximize resource utilization. Strategies like 'binpack' and 'spread' can optimize resource utilization and availability. You can increase container density on each instance as long as it doesn't compromise performance.

i. Binpack

Tasks are placed on container instances to leave the least amount of unused CPU or memory. When this strategy is used and a scale-in action is taken, Amazon ECS terminates tasks. It does this based on the amount of resources that are left on the container instance after the task is terminated. The container instance that has the most available resources left after task termination has that task terminated.

ii. Spread

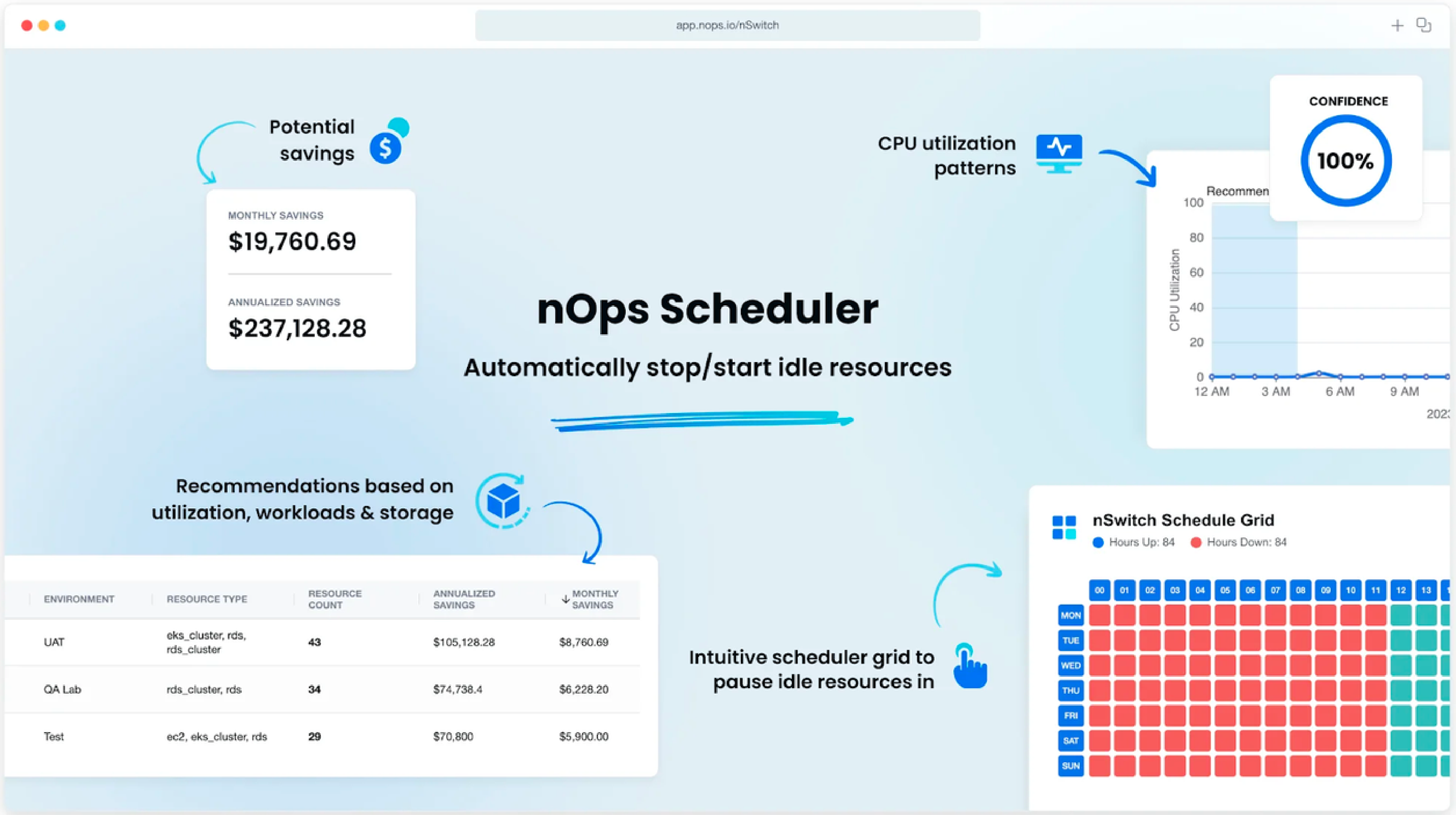
Tasks are placed evenly based on the specified value. Accepted values are instanceId (or host, which has the same effect), or any platform or custom attribute that's applied to a container instance, such as attribute:ecs.availability-zone. Service tasks are spread based on the tasks from that service.

04. Scheduling resources

Stop instances used in development and production during hours when these instances are not in use. Assuming a 50-hour work week, you can save 70% by automatically stopping dev/test/production instances during non-business hours.

Scheduling Tools

Include Amazon EC2 Scheduler, AWS Lambda, and AWS Data Pipeline. Or, third-party tools such as nOps Scheduler use AI to learn your usage patterns, identify the hours when an instance is not needed, and implement recommendations with a single-click.



ENVIRONMENT	RESOURCE TYPE	RESOURCE COUNT	ANNUALIZED SAVINGS	↓ MONTHLY SAVINGS
UAT	eks_cluster, rds, rds_cluster	43	$105,128.28	$8,760.69
QA Lab	rds_cluster, rds	34	$74,738.4	$6,228.20
Test	ec2, eks_cluster, rds	29	$70,800	$5,900.00

You can read further about how to schedule resources with this dedicated guide.

05. Optimizing your commitments

Reserved Instances (RI) and **Savings Plans (SP)** offer lower prices than On-Demand pricing in exchange for a specific usage commitment. However, forecasting commitments is an art as dynamic usage continually fluctates. If you commit too heavily, every hour you're paying for unused capacity. But if you under-commit, you'll have to pay the On-Demand premium for additional resources.

Here is a quick comparison of different commitment options to consider.

	Standard Reserved Instances	Convertible Reserved Instances	EC2 Instances Savings Plans	Compute Savings Plans
Applicable to	EC2 only	EC2 only	EC2 only	EC2, Lambda, & Fargate
Commitment term	1 year, 3 years	1 year, 3 years	1 year, 3 years	1 year, 3 years
Best case scenario savings	75%	54%	72%	66%
Payment Options	All Upfront, Partial Upfront, and No Upfront	All Upfront, Partial Upfront, and No Upfront	All Upfront, Partial Upfront, and No Upfront	All Upfront, Partial Upfront, and No Upfront
Instance Family	Fixed	Any	Fixed	Any
Instance Size	Fixed, except for regional scopes that use Linux/Unix	Any	Any	Any
Instance Operating System	Fixed	Any	Any	Any

	Standard Reserved Instances	Convertible Reserved Instances	EC2 Instances Savings Plans	Compute Savings Plans
Instance AZ	Any (Regional), Fixed (Zonal)	Any	Any	Any
Instance Tenancy	Fixed	Any	Any	Any
Need to reserve capacity?	Any (Regional), Fixed (Zonal)	No	No	No

Here are some quick tips:

1. **Understand what is, and what isn't, covered by a Savings Plan.** Savings Plans don't cover everything — they cover EC2 Compute, Fargate, and Lambda. They do not cover non-compute elements such as EBS volumes or data transfer.
2. **Analyze your On-Demand spend on an hourly basis.** Savings Plans and Reserved Instances apply on an hourly basis, meaning that you'll need to get fairly granular in your breakdown. They operate on a "use it or lose it" basis, meaning that if you commit to $10 per hour but only consume $6 of services in any hour, you will lose the remaining $4. That's why it is critical to monitor usage of commitments continually.
3. **Recognize that Savings Plans don't normally cover spikes**. Savings Plans are ideal for a steady and predictable compute spend level. You'll commit to using the agreed capacity per hour. If you exceed this usage, AWS will use On-Demand billing for extra charges.
4. **Understand how commitments are applied**. Savings Plans and RIs are applied in the following order:
    a. Standard Reserved Instances
    b. Convertible Reserved Instances
    c. EC2 Savings Plans
    d. Compute Savings Plans: For advantages and disadvantages of the various SP and RI options, as well as practical use cases and best practices for squeezing more out of commitments, check out our recent ebook The Ultimate Guide to AWS Commitments.

06. Spot Instances and Fargate Spot

If you're looking to save on ECS, Spot is key to reducing cost. Spot Instances allow users to take advantage of unused EC2 capacity at a fraction of the standard On-Demand price.

Instance Type	Spot vs. On-Demand Savings	Expected Savings
General Purpose	Up to 90% cheaper than On-Demand	Very High
Compute Optimized	Up to 80% cheaper than On-Demand	High
Memory Optimized	Up to 75% cheaper than On-Demand	Moderate to High

However, they can be reclaimed by AWS with just a 2-minute warning if there's higher demand for the capacity. This characteristic introduces a few risks and considerations that need to be managed carefully:

- **Service Interruption**: The most direct risk is the potential for service interruption. If AWS reclaims Spot Instances, any containers running on those instances will be stopped, which disrupts running services or batch jobs if not properly managed.
- **Task Interruptions and Failures**: For ECS tasks that are stateful or require longer processing times, interruptions can lead to incomplete transactions or data corruption if not properly handled.
- **Increased Management Overhead**: To effectively use Spot Instances, you need to implement more sophisticated management strategies such as including handling instance interruptions, increasing the complexity of your infrastructure.
- **Variable Costs**: While Spot Instances can offer significant cost savings, market prices continually fluctuate based on demand. If not monitored and controlled, costs can potentially spike.

- **Capacity Availability:** Spot Instance market availability varies and might not always be able to match the scale or specific instance types/sizes your applications require, leading to potential scaling limitations.

However, with the right techniques, you can successfully utilize AWS Spot Instances in conjunction with ECS to significantly reduce costs while maintaining performance and reliability.

Best Practices for Using Spot Instances with ECS

01. Analyze the suitability of your application.

Stateless, fault-tolerant, or flexible applications are ideal candidates for Spot. Implement a robust failover strategy using ECS service auto-scaling to maintain application availability and regularly test the resilience of your ECS environment on Spot Instances to ensure that it can handle instance interruptions gracefully.

02. Use ECS Capacity Providers with Spot Fleet Integration.

Capacity providers are used to manage the infrastructure on which ECS tasks are run. They can be set up to use a mix of On-Demand and Spot Instances. You'll need to use a capacity provider strategy that prioritizes Spot Instances but falls back on On-Demand Instances when needed.

Spot Fleets manage a collection of Spot Instances and optionally On-Demand Instances to meet a certain capacity target. You can combine Spot Fleets with ECS to manage instances and scale capacity efficiently.

Spot Fleet has the ability to mix instance types and sizes to optimize cost and maintain availability, and you can configure Spot Fleet to replace terminated instances automatically.

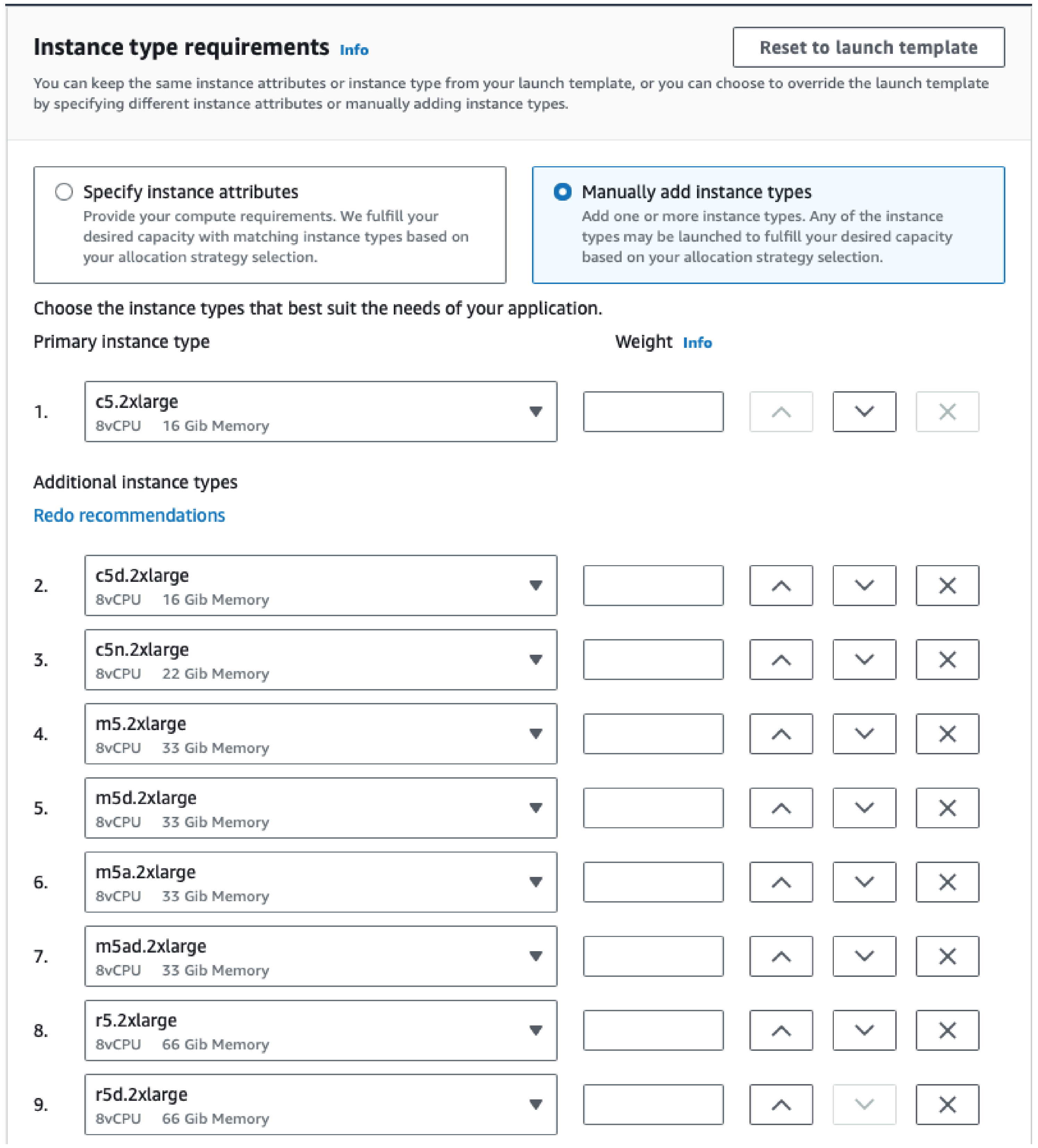
03. Implement Spot Instance Diversification

Rather than relying on a single instance type, diversify your Spot Instances across various instance types and availability zones. This strategy reduces the impact of sudden market fluctuations on your workload.

When it comes to critical applications, you want to *always maintain a core base capacity of On-Demand instances* to ensure they remain unaffected by interruptions in the event that no suitable Spot is available.

04. Implementing Spot Instance Draining

Use ECS's [managed instance draining]() feature to gracefully handle Spot Instance terminations by stopping tasks on the instance and rescheduling them on other instances. Monitor Spot Instance advisories and trigger the draining process automatically.



Instance type requirements Info

Reset to launch template

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

( ) Specify instance attributes
Provide your compute requirements. We fulfill your desired capacity with matching instance types based on your allocation strategy selection.

[✓ Manually add instance types]
Add one or more instance types. Any of the instance types may be launched to fulfill your desired capacity based on your allocation strategy selection.

Choose the instance types that best suit the needs of your application.

Primary instance type — Weight Info

	Instance type	vCPU	Memory
1.	c5.2xlarge	8vCPU	16 Gib Memory

Additional instance types

Redo recommendations

	Instance type	vCPU	Memory
2.	c5d.2xlarge	8vCPU	16 Gib Memory
3.	c5n.2xlarge	8vCPU	22 Gib Memory
4.	m5.2xlarge	8vCPU	33 Gib Memory
5.	m5d.2xlarge	8vCPU	33 Gib Memory
6.	m5a.2xlarge	8vCPU	33 Gib Memory
7.	m5ad.2xlarge	8vCPU	33 Gib Memory
8.	r5.2xlarge	8vCPU	66 Gib Memory
9.	r5d.2xlarge	8vCPU	66 Gib Memory

05. Use Spot Placement Scores

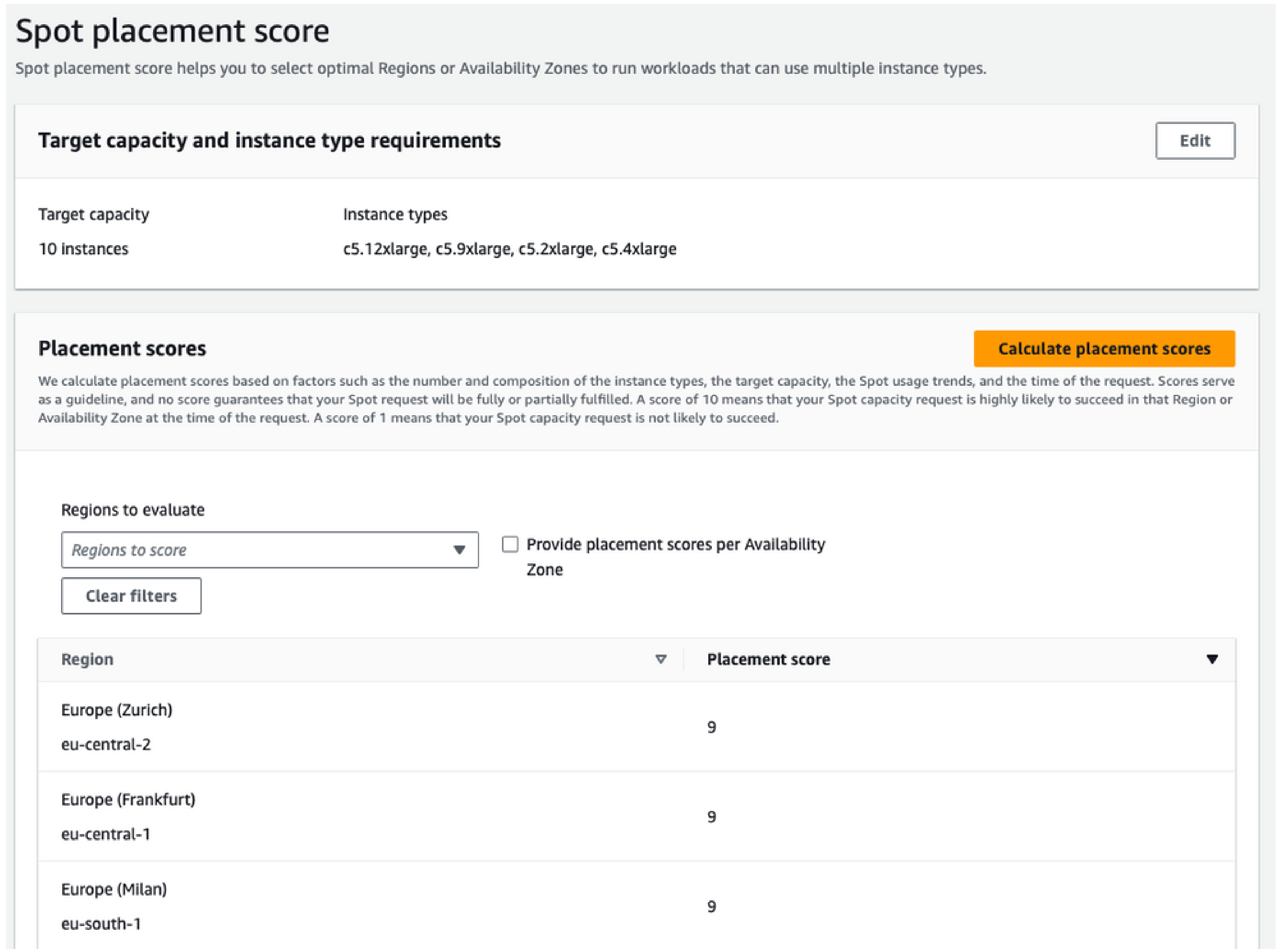
To better understand the market conditions for your selected instance types you can use Spot Placement Scores provided by AWS.

However, it's worth noting understanding and interpreting placement scores accurately demands a deep understanding of how AWS calculates these scores and their implications on Spot instance availability.

Incorporating Spot placement scores into auto-scaling strategies adds complexity, because you need to balance cost, performance, and availability. Changing Spot market pricing and availability means users must continuously monitor and react quickly to score changes to optimize their Spot instance use.

Failing to do so properly can result in a volatile Spot instance environment, leading to application downtime, data loss or corruption, and increased overhead due to operational challenges. And if you don't pick the right Spot instances, you might actually spend more for your trouble.

If these challenges sound familiar, **nOps Compute Copilot** can help. Just simply integrate it with EC2, ASG, EKS, Batch or other compute-based workload and let nOps handle the rest.



Spot placement score

Spot placement score helps you to select optimal Regions or Availability Zones to run workloads that can use multiple instance types.

Target capacity and instance type requirements

Target capacity	Instance types
10 instances	c5.12xlarge, c5.9xlarge, c5.2xlarge, c5.4xlarge

Placement scores

We calculate placement scores based on factors such as the number and composition of the instance types, the target capacity, the Spot usage trends, and the time of the request. Scores serve as a guideline, and no score guarantees that your Spot request will be fully or partially fulfilled. A score of 10 means that your Spot capacity request is highly likely to succeed in that Region or Availability Zone at the time of the request. A score of 1 means that your Spot capacity request is not likely to succeed.

Regions to evaluate: Regions to score

☐ Provide placement scores per Availability Zone

Region	Placement score
Europe (Zurich) eu-central-2	9
Europe (Frankfurt) eu-central-1	9
Europe (Milan) eu-south-1	9

About nOps

nOps offers proprietary AI-driven management of instances for the best price in real time. It continually analyzes market pricing and your existing commitments to ensure you are always on the best blend of Spot, Reserved, and On-Demand, gracefully replacing nodes before termination.

Here are the key benefits of delegating the hassle of cost optimization to nOps.

- **Hands free.** Copilot automatically selects the optimal instance types for your EC2 or other workloads, freeing up your time to focus on building and innovating.
- **Cost savings.** Copilot ensures you are always on the most cost-effective and stable Spot options.
- **Enterprise-grade SLAs** for the highest standards of reliability. Run production and mission-critical workloads on Spot with complete confidence.
- **No vendor-lock in.** Just plug in your AWS-native ECS to start saving effortlessly, and change your mind at any time.
- **No upfront cost**. You pay only a percentage of your realized savings, making adoption risk-free.

nOps manages over $1.5 billion in cloud spend and was recently ranked #1 in G2's cloud cost management category. Join our customers using nOps to slash your cloud costs and leverage automation with complete confidence by **booking a demo** today!