



White Paper

---

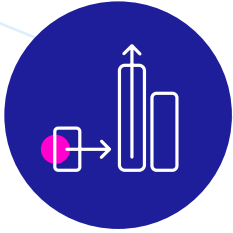
# Compute infrastructure right-sizing in a cloud-native world





# Table of contents

- 03 Introduction
  - 04 Visibility and monitoring
  - 05 Where to start right-sizing
  - 06 Criteria for downsizing and testing
  - 07 Right-sizing automation
  - 10 The future - matching performance requirements to infrastructure resources
- 



**Get accurate data**  
for accurate right-sizing

## Introduction

Unless you have prophetic powers, knowing in advance exactly how much CPU, memory, disk and network your applications are going to require is one of the **trickiest areas** during deployment. This gap, in our experience, leads many companies to accept a t-shirt sizing methodology for selecting instances - small, medium, large and extra large - with engineers often rounding up and greatly overprovisioning. As this is still an area that has not been automated for precise tracking, data on potential savings is anecdotal, but seems to be in the range of 25% and **possibly much higher**.

Fortunately, in the cloud you are not stuck with physical servers as you would have been had you installed them in your datacenter, and can always right-size your compute infrastructure even mid-flight. The challenge however, is in knowing when it's appropriate to right-size, and having the right metrics to make that decision. Once you have the relevant data, we highly recommend investing in resources for right-sizing efforts, particularly for any workloads that will run long-term. Let's take a look at what this would entail.



**Supersizing really does save you money!**

As seen in the chart below, by going a few sizes up, we can achieve better resource allocation and **reduce the cost by 20%**.

	Pod count	Pod M	Instance M	Memory %	Pod vCPU	Instance vCPU	CPU %	Cost per pod (OD hourly)
m5.large	1	6,500	8,000	81.25%	1.5	2	75%	\$0.096
m5.xlarge	2	13,000	16,000	81.25%	3	4	75%	\$0.096
m5.2xlarge	5	31,500	32,000	98.44%	7.5	8	93.75%	\$0.077

**Example:** Pod requires 6,500mb of memory and 1.5vCPUs



While autoscaling solutions provide basic optimization by adding or removing resources based on demand, there still is a **significant level of over and under provisioning** that needs to be addressed.

## Visibility and monitoring

To ascertain whether your workloads can be moved to machines with less resources requires, at a minimum, visibility into their CPU and memory utilization. Metrics such as disk and network consumption can also play an important role in determining whether to increase or decrease resource requirements. Let's take a look at the various options for both non-containerized and containerized workloads.

### Monitoring legacy and non-containerized applications

[AWS Cloudwatch](#), [Azure Monitor](#) and [Google's Cloud Monitoring](#) all allow you to monitor and collect CPU, Memory and other custom metrics (for a fee), with built-in dashboards and more. Amazon's [Compute Optimizer](#) was introduced in 2019 (leveraging Cloudwatch metrics) to provide both recommendations for right-sizing along with a simulation of how your workload would perform on the recommended, right-sized instances.

Third-party infrastructure monitoring vendors such as [Datadog](#), [LogicMonitor](#), [Dynatrace](#), [Solarwinds](#) and others also allow you to collect all the relevant utilization metrics from your AWS, Azure and Google Cloud compute infrastructure.

### Monitoring containerized workloads

When it comes to containerized workloads the trend to overprovision the underlying compute infrastructure (commonly referred to as nodes), is much the same as with legacy and non-containerized workloads. Likewise, the right-sizing approach is essentially the same, the main difference being where metrics collection and the actual right-sizing occur. With container workloads, this happens at the container-level and your autoscaler launches, retires and bin-packs nodes to meet the desired capacity of your freshly right-sized resource requirements.



### Insights

into compute utilization allows proper right-sizing of container requirements.

**To learn more about autoscaling for containerized workloads see [Ocean by Spot.io](#) or [open source Cluster Autoscaler](#).**

For AWS users, metrics collection can be done via [Amazon CloudWatch Container Insights](#), enabling you to collect all relevant metrics for your ECS workloads as well as for [EKS and Kubernetes](#). Other options such as [Metrics Server](#), [cAdvisor](#) and [Prometheus](#) can be used to understand how your pods and containers utilize CPU and memory. There are a number of considerations to take into account when deciding which monitoring tool is right for your organization, including cost, ease-of-use, and multi-vendor capabilities.


With your cluster utilization insights in hand, you can now right-size your tasks and pods while your autoscaling solution handles the underlying infrastructure.

## Where to start right-sizing

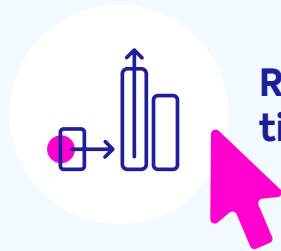
It's tempting to start your right-sizing project off with your most expensive applications and environments where your upfront investment may yield higher savings, especially as monitoring costs are typically linked to the [number of assets and metrics being monitored](#).

However, often the best way to show tangible and quick results is to start off with lower systems that you can easily identify as overprovisioned. Most likely, this low hanging fruit is already on your engineers' radar and can prove to be a good testing ground for your first right-sizing project. From that initial success your organization will be more willing to allocate time and resources to additional right-sizing exercises.

To obtain the most accurate assessment of your environment, regardless of which one you choose to focus on, you should monitor all instances running in it. Just looking at a subset of resources can lead to an incomplete picture and jeopardize the efficacy of your right-sizing.



> See how  
Ocean identifies  
right-sizing  
candidates.



### Right-sizing tip

Sometimes right-sizing can yield **dramatic results** as seen here with the c5n.large.

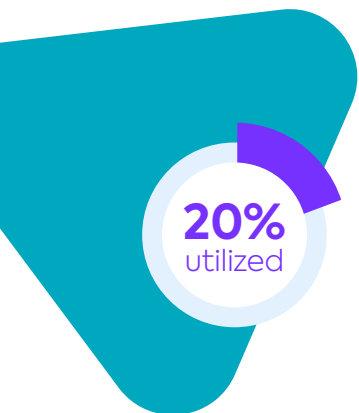
It provides all the networking capacity needed for communication-intensive workloads without the excess CPU and Memory baggage that the c4.8xlarge has and for **94% cost savings!**

Instance	Price	vCPU	Memory (GiB)	Instance storage	Network performance
c4.8xlarge	\$1.591 per hour	36	60	<ul style="list-style-type: none"><li>• EBS only</li><li>• Dedicated EBS</li><li>• Bandwidth (Mbps): 4,000</li></ul>	10 Gigabit
c5n.large	\$0.108 per hour	2	5.25	<ul style="list-style-type: none"><li>• EBS only</li><li>• EBS Bandwidth (Mbps): Up to 4,750</li></ul>	Up to 25 Gigabit

## Criteria for downsizing and testing

After observing workload resource utilization, anything less than 20% utilized is typically an excellent candidate for downsizing. Even if utilization is higher, but still nowhere near max capacity, you might consider downsizing.

Testing workload performance on right-sized instances to make sure there is no performance degradation is key. If you have both higher and lower environments for the same application workload, first right-size on the lower environments and then use load testing tools such as [Jmeter](#), [Gatling](#) or others to evaluate performance. Once that's done, ongoing monitoring of your application's behavior in your production environment should be done with APM tools such as [New Relic](#) and [AppDynamics](#) to ensure that the right-sizing does not impact performance at any point.



## Downsizing candidate!

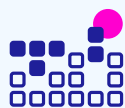
Seriously consider moving down a size when utilization is under 20%.

# Right-sizing automation

Even with properly defined resource requirements there is still a potential risk of misconfiguration, unless you have implemented automated right-sizing. Someone or something (a CI/CD system for example) can revert any resource update, resulting in misconfigured resource requirements for your services/deployments. Additionally, production resource consumption typically changes over time with a resulting mismatch between requirements and actual resources.

To ensure this doesn't happen there are several options for automating right-sizing. For a do-it-yourself approach, you can either implement automated right-sizing at the beginning of the CI process or at the "end" of the CD process.

For the beginning of the CI process, adding a resource update step to your existing automation of your Deployment Yaml generator will do the trick. This will result with a ready-to-apply Kubernetes Deployment object with the right-sized resource requirements configuration.



When it comes to right-sizing automation, we are focusing on **containerized workloads** as they are a natural fit for this. Here are some of the main reasons why:

- **Infrastructure and application** configuration is done in a single object, (e.g. a Kubernetes deployment)
- The ephemeral nature of containers allows you to use an **"immutable upgrade"** approach so when you need to change something (whether application or infrastructure) you create a new object with the new configurations and simply delete the old object
- Kubernetes (and containers) introduced **consistent API specifications** for updating objects (e.g. a Kubernetes Deployment) for on-premise as well as the major cloud platforms (AWS, Azure or Google)

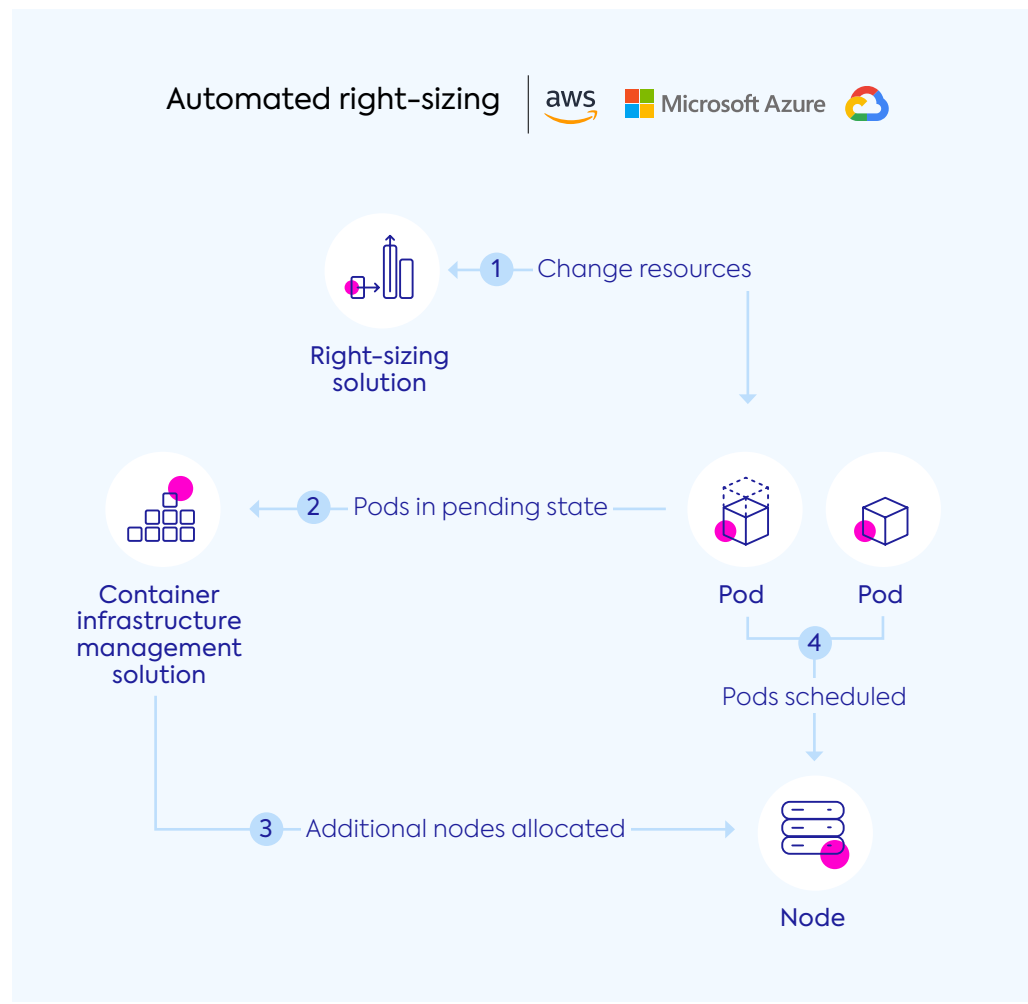
For the end of the CD process, an in-cluster mechanism to intercept any updates to the Kubernetes Deployment object will override any misconfigured resource requests and right-size them on the fly.

If you are looking for a more out-of-box solution that only requires minor configuration and will also dynamically assess whether your deployments are right-sized, you might consider using Kubernetes' native [VPA or vertical pod autoscaler](#). For a fully automated process use VPA's auto mode to restart pods automatically after their resource requirements are updated.

All of these options should be integrated with the open-source Cluster Autoscaler to translate the right-sized resource requirements into matching compute infrastructure, i.e. nodes.



### Automated right-sizing

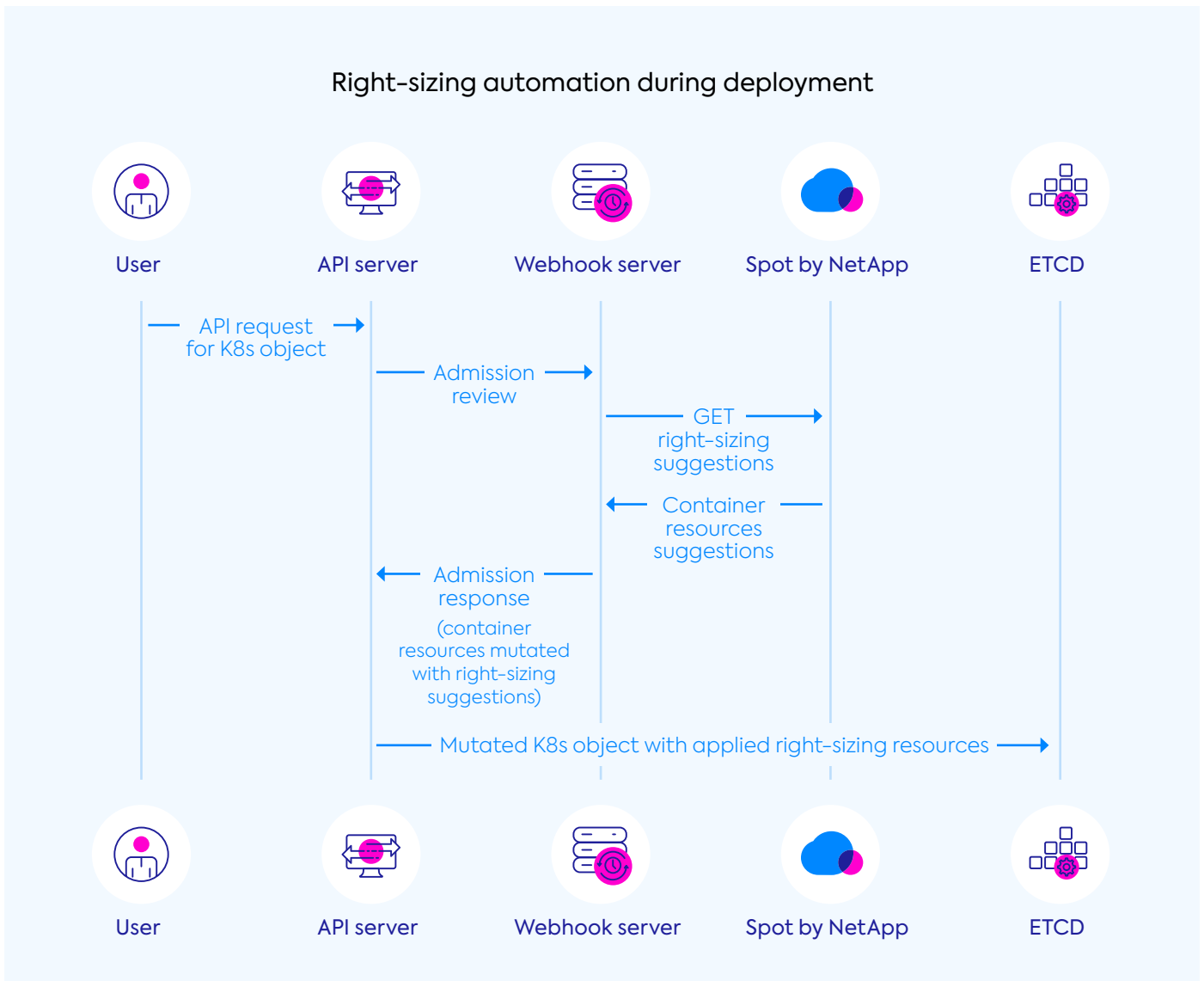






For a completely turn-key solution Ocean by Spot provides resource utilization analysis and sizing recommendations for all containerized workloads which can be implemented as part of the CI process or when a Deployment is being created on the cluster.

Based on the right-sized requirements, Ocean continuously manages the underlying nodes ensuring that you always have the optimal compute power needed by your cluster.



# The future: Matching performance requirements to infrastructure resources

In the near future we expect that cloud consumers will be able to easily right-size compute infrastructure simply by defining the workload performance requirements. For example, if an application needs to support 1,500 requests per second with a latency of 50ms, an advanced automation solution will run the workload on infrastructure with the right amount of vCPUs, memory, disk and network - all at the best price.

Until that day arrives however, a perfectly right-sized environment is completely achievable with a wide range of options and tools available to accomplish this.



Ocean by Spot simplifies infrastructure management for container orchestration tools. With robust, container-driven infrastructure auto-scaling and intelligent right-sizing for container resource requirements, engineers can code more, while operations can literally "set and forget" the underlying spot instance cluster.

[Try Ocean for free! >>](#)

