White Paper

The state of Amazon EC2 spot instances 2020

Up to
90%
cost reduction

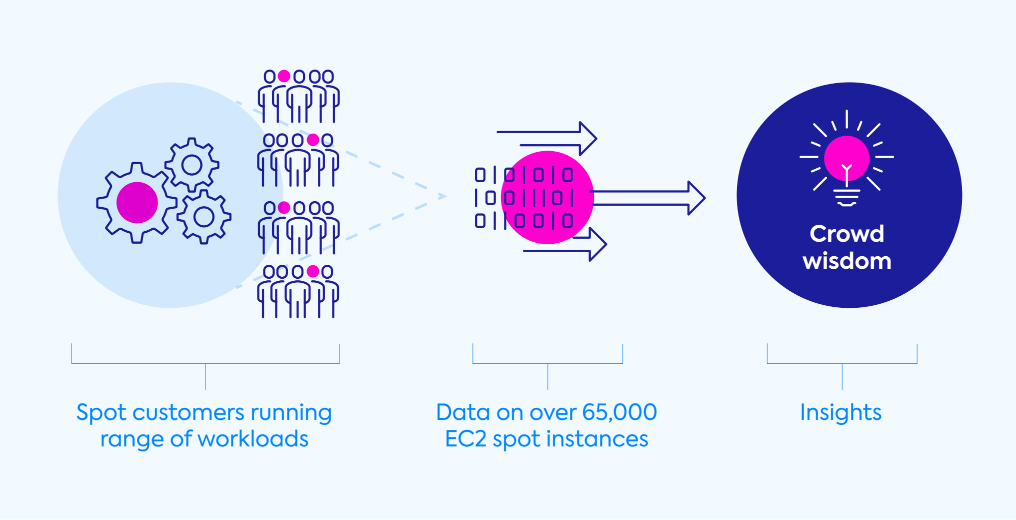
Introduction

As one of the most effective ways to dramatically reduce compute infrastructure costs, EC2 spot instances have always played a role in managing cloud costs, with the eye-popping potential of up to 90% cost reduction.

Managing cloud costs is particularly important in periods of volatility, whether volatility caused by a product or service suddenly "going viral" or by global disruptions such as COVID-19. Massive and unexpected changes in the consumption of all things digital force companies to find a way to service the surge in customer traffic (which in many cases may not come with the same order of magnitude increase in profit) without breaking the bank.

For a very savvy, albeit small portion of AWS's 1 million+ customers, EC2 spot instances were already a part of their cost optimization arsenal of tools, and these AWS customers have been able to match the surge in customer demand while keeping costs low.

In this paper we share unique insights gleaned from the data of over 1,000 Spot.io customers running a broad range of workloads, on over 65,000 EC2 spot instances (~14.2 million CPU resources) on a daily basis. These insights represent a form of "crowd-wisdom" and can prove instructive in areas such as container orchestration, stateful applications, and best performing EC2 instances.


Crowd wisdom
Spot customers running range of workloads
Data on over 65,000 EC2 spot instances
Insights

First some spot instance basics

AWS offers four primary pricing models to pay for and use EC2 (Elastic Compute Cloud) instances. Familiarity with these pricing models is important for creating a well-balanced cost-optimization strategy that properly leverages all models.

It's important to note that irrespective of the pricing model you select, there is absolutely no difference in the actual compute resource you receive (i.e. the underlying virtual machines will be the same). The only differences are the amount you pay, the level of financial lock-in and the SLA for availability that you receive.

On-demand

This option is essentially pay-as-you-go, allowing you to spin up and down EC2 instances at will, provided that the instance type you want is available when you want it. EC2 on-demand is the most expensive pricing option.

Reserved instances

This option consists of an upfront financial commitment for 1 or 3 years of EC2 usage which in turn provides users with guaranteed capacity for the instance type selected. Savings compared to on-demand are roughly in the 75% range. However, reserved instances create financial lock-in, so if you don't use what you committed to, you could potentially end up with a negative ROI.

Savings Plans

This option is similar to reserved instances in the commitment terms of 1 or 3 years, but does not require you to select a specific instance and rather can be applied to any EC2 instance (as well as other AWS services). For example, you can commit to spend a desired amount per hour, e.g. $35/hour, for either 1 or 3 years. Anything spent up to $35 will be charged in accordance with Savings Plans rates (between 66-72% savings). Any spend above the committed amount will be charged at on-demand rates.

Spot instances

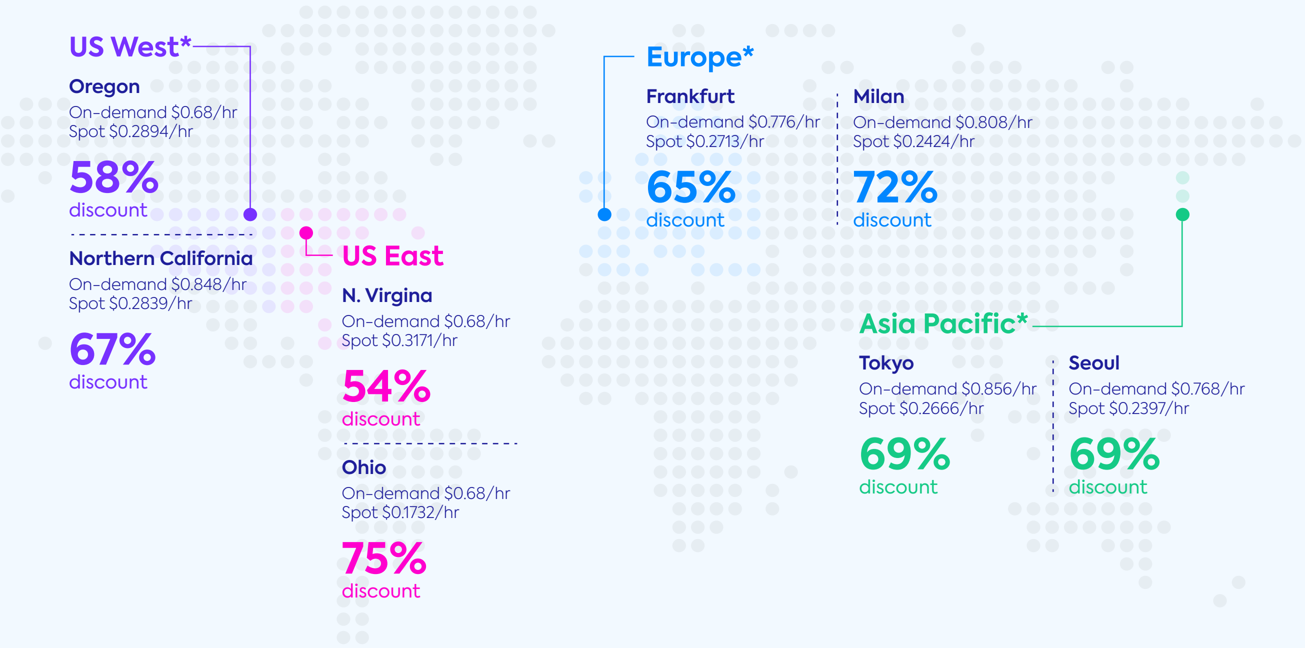
This option offers up to 90% cost reduction when compared to on-demand pricing. These spot instances represent AWS's excess capacity which as a cloud provider, they absolutely need to have available for any surges in customer demand. To offset the loss of idle infrastructure, AWS offers this excess capacity at a massive discount to drive usage. However, this discounted pricing comes with the caveat that AWS can "pull the plug" and terminate spot instances with just a 2 minute warning. These interruptions occur when AWS needs to draw from the excess capacity to service customers who purchased reserved instances, Savings Plans or on-demand instances. Of course, sudden interruption of EC2 instances can result in data loss, service degradation, unavailable services and the like, theoretically making spot instances a challenge for mission-critical, production workloads.

Use cases for spot instances

The general perception of spot instances is that they are only appropriate for web services, containerized applications or other stateless, fault-tolerant workloads. However, in reality with the right set of tools and solutions, they can be used for a much broader set of use cases, without any significant impact on availability or performance. Here are the use cases that we see running on spot instances:

- **Auto Scaling apps,** such as those that are part of auto scaling groups and web apps running behind ELBs, are a classic use case for spot instances, as covered in this case study.

On-demand and spot instance pricing for Linux C5.4xlarge (in select regions)



US West*

Oregon
On-demand $0.68/hr
Spot $0.2894/hr
58%
discount

Northern California
On-demand $0.848/hr
Spot $0.2839/hr
67%
discount

US East

N. Virgina
On-demand $0.68/hr
Spot $0.3171/hr
54%
discount

Ohio
On-demand $0.68/hr
Spot $0.1732/hr
75%
discount

Europe*

Frankfurt
On-demand $0.776/hr
Spot $0.2713/hr
65%
discount

Milan
On-demand $0.808/hr
Spot $0.2424/hr
72%
discount

Asia Pacific*

Tokyo
On-demand $0.856/hr
Spot $0.2666/hr
69%
discount

Seoul
On-demand $0.768/hr
Spot $0.2397/hr
69%
discount

***** On-demand pricing can also vary by region and impacts the actual savings that can be achieved for location-agnostic workloads.
Note: Even within regions, spot instance pricing can significantly vary between AZs.

- **Containers and microservices** are typically self-contained, fault-tolerant and highly available making them ideal for handling spot instance interruptions with dramatic savings for containerized applications, such as the EKS workloads described in this case study.

- **High Performance Computing (HPC)** applications often require high network performance, fast storage, large amounts of memory, and very high compute capabilities, all of which can be supported by spot instances, particularly when used for bursting, but even as the primary compute infrastructure being used.

- **Big Data ETL** running on Amazon EMR, Hadoop, Spark and batch processing in general are all great candidates for spot instances as shown in this case study.

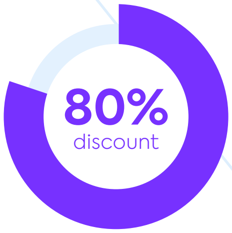
- **Machine Learning and AI** is another area in which deep learning and training progress can be negatively impacted by unplanned spot instance interruptions. But with the right tools, you can successfully run all your ML projects on spot instances as documented here.

- **Dev/Test apps** that run as stateful workloads typically require data and IP persistence. With automated solutions, even in the event of spot instance replacement, your workload will immediately restart in the desired Availability Zone, from the same exact data point, maintaining root and data volumes as well as private and public IPs.

- **CI/CD** operations whether handled by Jenkins, Chef, Gitlab or others, can be run, at scale, on spot instances quite easily as demonstrated in this case study.

- **Distributed DBs** such as Elasticsearch, Cassandra and Mongo which can handle a “reboot” of a single instance without losing data or affecting service, can also run on spot instances.

Hot deals
US East



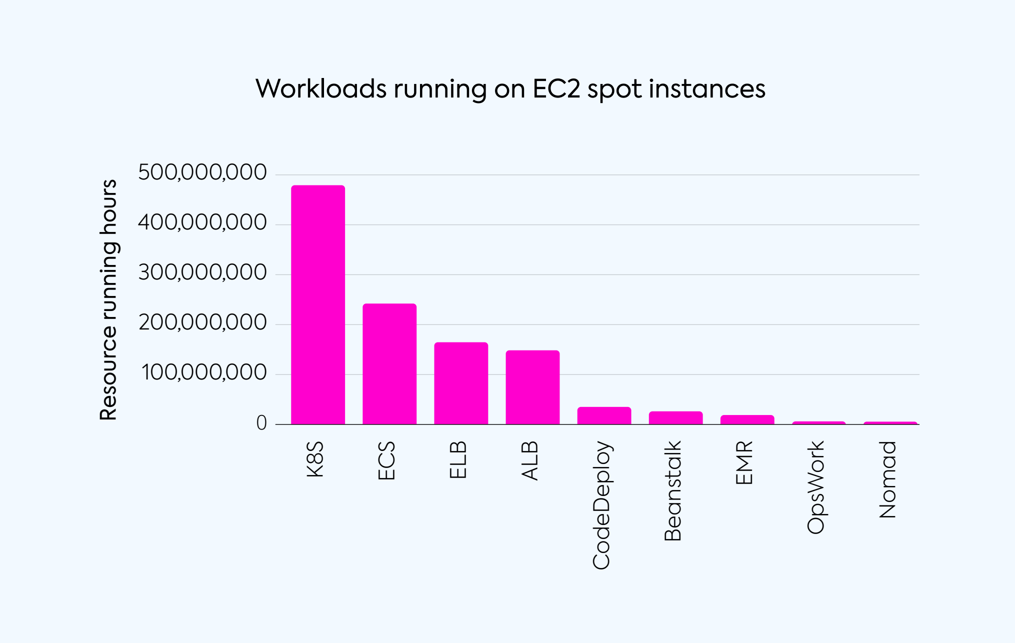
Linux R5.4xlarge
(N. Virginia)
On-demand $1.008/hr
Spot $0.1922/hr*

***Note:** Even if the price is attractive, spot instance longevity needs to be determined before deciding to deploy workload.

Crowd-wisdom & insights into spot instance usage

The insights we have collected here provide you with crowd-wisdom of what workloads can be successfully run on spot instances, which instance types are most popular and what the ideal balance of spot instances, on-demand and reserved capacity should be within production environments.

Analysis of spot instance usage



With microservices and containerization becoming the gold standard in software engineering, we have seen Kubernetes and ECS workloads go from roughly 30% of all spot instance usage in 2017 to today where they make up more than 70% of all workloads running on spot instances.

This level of usage is not surprising considering that modern containerized applications built with microservices architecture are typically self-contained, fault-tolerant and highly available. These characteristics make them, in theory, perfectly suited for ephemeral spot instances, as even if a spot instance is terminated, the containers running on it can be moved to a new, replacement instance.

Hot deals
US East

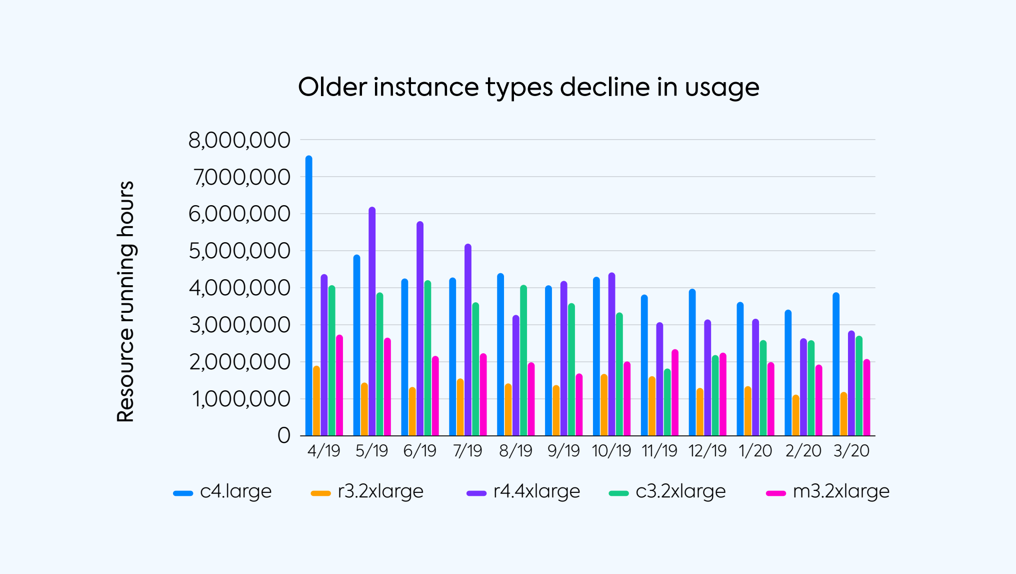
Linux C5d.4xlarge
(Ohio)
On-demand $0.768/hr
Spot $0.1581/hr*

***Note:** Even if the price is attractive, spot instance longevity needs to be determined before deciding to deploy workload.

As containerized workloads comprise such a significant portion of spot instance consumption, we will dedicate an entire section to containers and Kubernetes later on in this white paper.

For non-containerized workloads, let's take a look at how AWS customers are voting in terms of popular instance types, new types of workloads and pricing models.

Instance type popularity across all workloads



With AWS introducing new instance types on a regular basis, monitoring what instances customers are successfully using can be deemed a cloud computing best practice.

In the graph above, we can see that c3, r3, m3, c4, and r4 instance families have decreased in usage since April 2019. This likely reflects these older instance types being deployed less as AWS recommends moving workloads to newer instance types.

What exactly are spot instances

As the leading cloud provider, AWS must maintain excess capacity to satisfy any surges in customer demand. To offset the loss of idle infrastructure and drive consumption, AWS offers this excess capacity as spot instances at a massive discount.

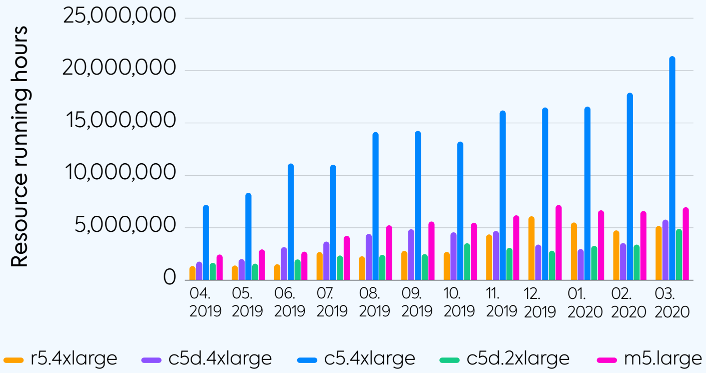
So why doesn't everyone use them?

The only catch is that AWS can pull the plug with just a 2 minute warning, making spot instances tricky to manage for production workloads.

Hot deals
US East

Linux M3.medium
(N. Virginia & most regions)
On-demand $0.067/hr
Spot $0.0067/hr*

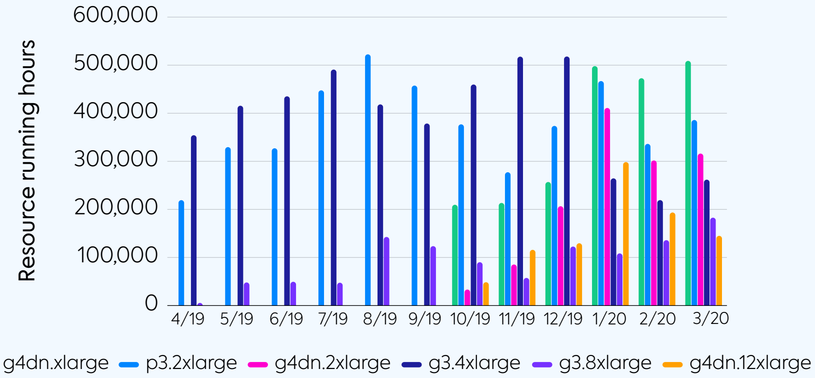
Instances with significant growth in usage since 2019



From the above graph we see that the c5 family (with the c5.4xlarge in particular) as well as m5.large and r5.4xlarge have become quite popular with, in some cases, an over 200% increase in usage compared to 2019. This is no doubt helped by AWS making these instance types available in more regions.

Machine learning and graphics-intensive workloads

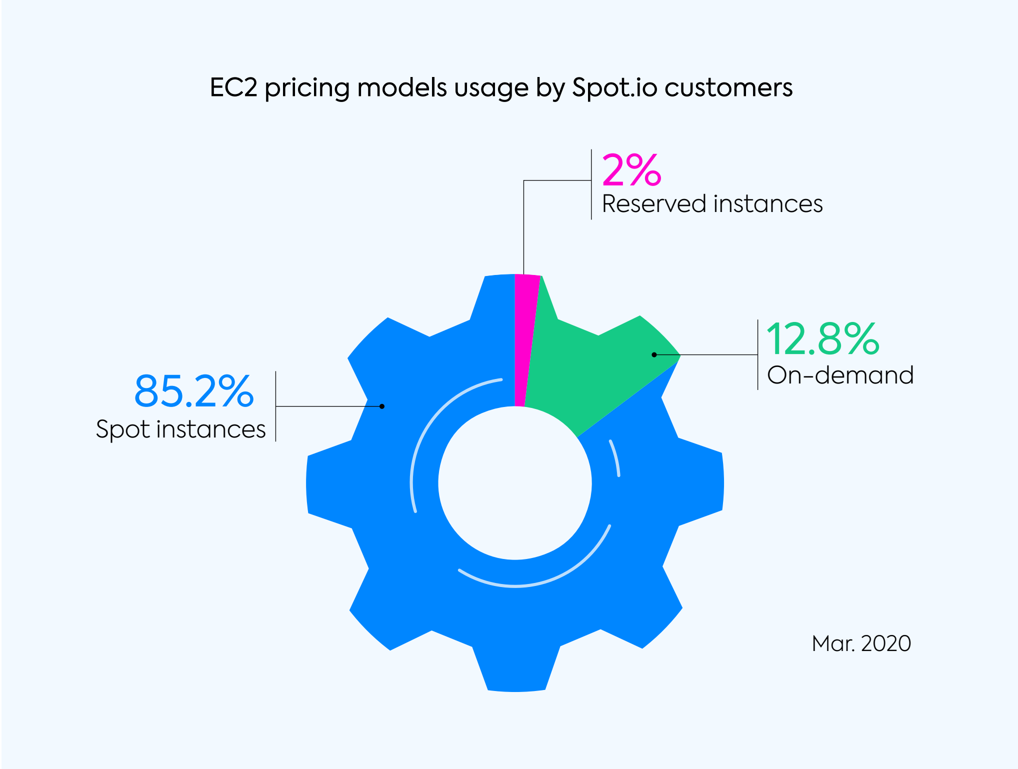
GPU spot instances shifting to newest versions



***Note:** Even if the price is attractive, spot instance longevity needs to be determined before deciding to deploy workload.

As machine learning and graphics-intensive workloads benefit from better specifications (e.g. NVIDIA T4 Tensor Core GPUs) even more so than other workload types, the clear shift towards the newer, more powerful instance types such as the older g3.4xlarge to newer ones such as the g4dn.xlarge, is to be expected. Of note, although many machine learning workloads are stateful, as can be seen here, running them on spot instances has become standard.

How 3rd party tools impact the selection of EC2 pricing models used for production workloads


EC2 pricing models usage by Spot.io customers

Mar. 2020

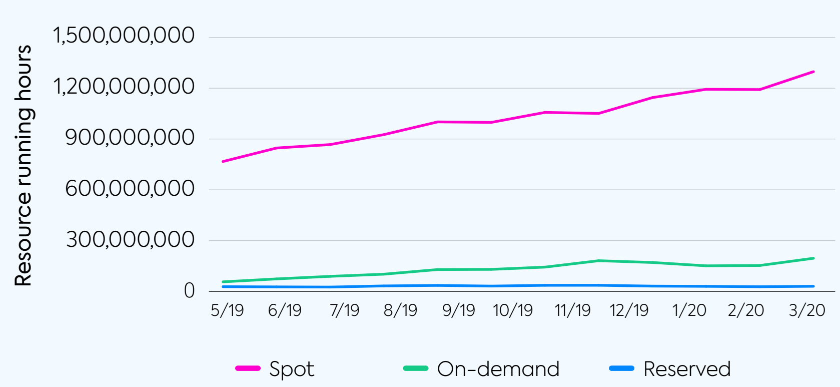
The above graph (which represents over ~1,000 companies' production environments running via Spot.io) shows that when AWS customers can easily run their workloads on spot instances, they will opt to do so.

Hot deals
US East

86%
discount

Linux R5n.12xlarge
(Ohio)
On-demand $3.576/hr
Spot $0.5017/hr*

***Note:** Even if the price is attractive, spot instance longevity needs to be determined before deciding to deploy workload.


EC2 pricing model consumption trends

Hot deals
US East

84%
discount

Linux M5d.8xlarge
(Ohio)
On-demand $1.808/hr
Spot $0.3193/hr*

Furthermore, the use of affordable spot instances, especially when doing so involves little effort and availability is guaranteed, has grown by over 62% since May 2019.

Containerized workloads and spot instances

Analyzing Spot.io data from thousands of clusters, tens of thousands of nodes and over 1.5 million containers running on a daily basis, we see many significant trends occurring in the AWS cloud ecosystem. In the following sections we examine data on the types of instances being used, popular container orchestrators and other technology consumption patterns.

Analyzing Spot.io data

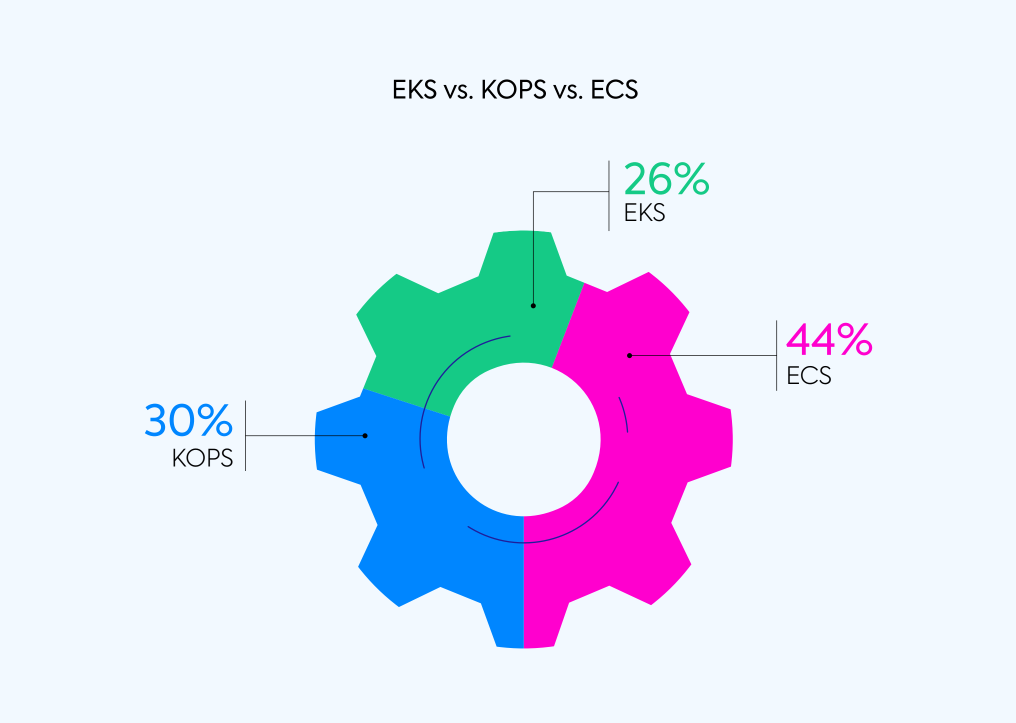
> 1,000 clusters

> 10,000 nodes

> 1.5 million containers

***Note:** Even if the price is attractive, spot instance longevity needs to be determined before deciding to deploy workload.

Types of container orchestrators being used

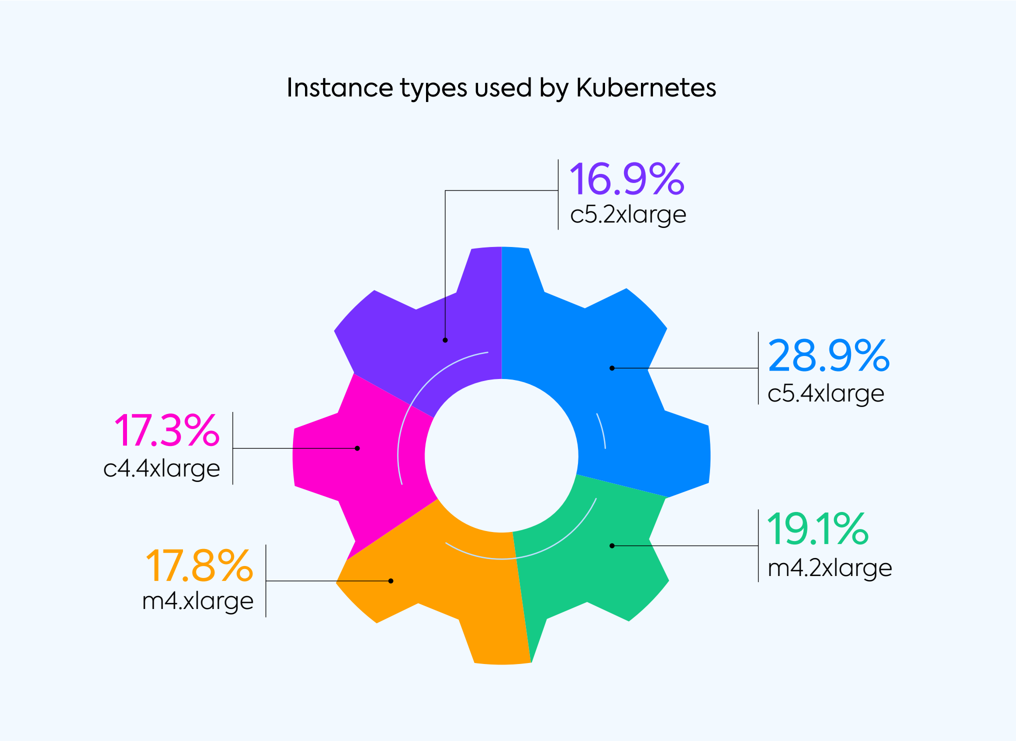


A quick look at the above chart shows that Kubernetes-based orchestrators represent the majority of container orchestrators with KOPS and EKS together showing 56% of total usage and ECS representing just 44%.

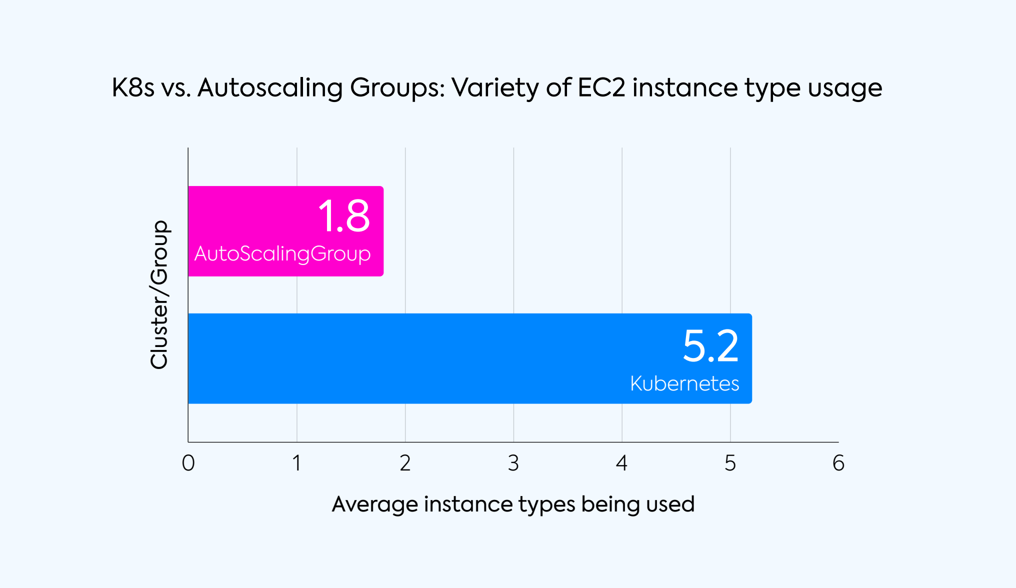
This is quite possibly due to concerns about vendor lock-in. ECS containerized workloads only can be run in AWS, whilst Kubernetes based orchestration, such as KOPS and even EKS, can be easily ported to other cloud vendors such as Azure and GCP.

Interestingly, while many comparative analyses found online indicate that EKS overall provides greater ease of usage, a significant share of operations teams still seem to prefer KOPS, perhaps because managing their master servers directly allows them to make custom configurations as needed. Alternatively, it might simply be because KOPS has been around longer.

Containerized workloads leverage larger instances and a broader variety of instance types



When we examine the most used instance types in Kubernetes deployments, we see that more than 80% is made up of 2XL and 4XL instance sizes.

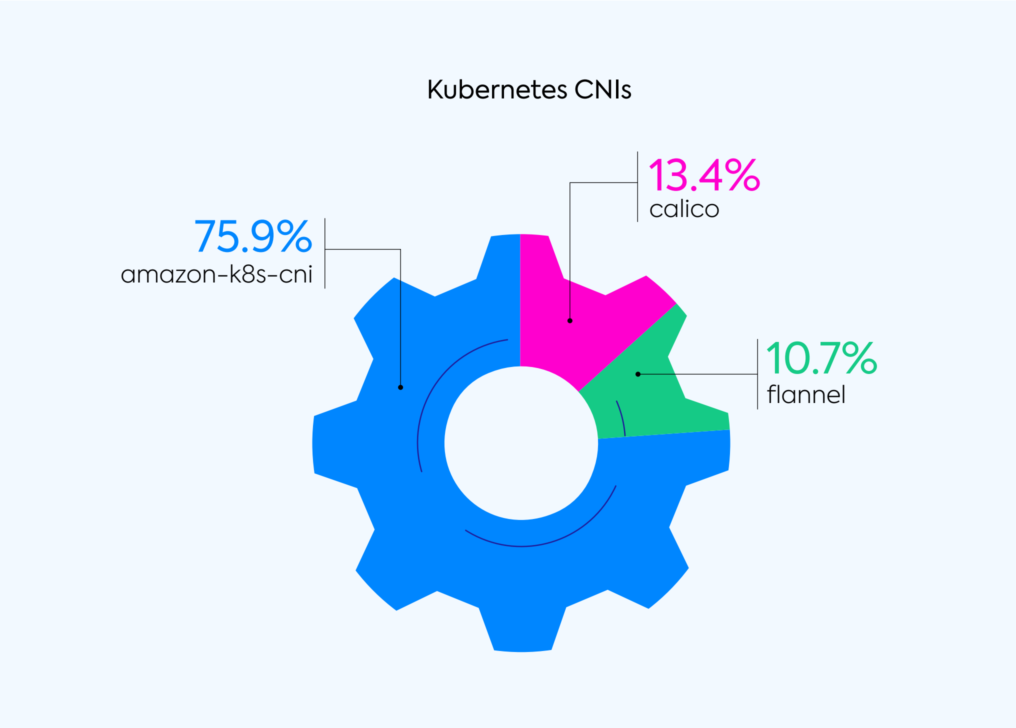


Additionally, as the graph above shows, containerized clusters leverage more than twice the average number of spot instance types than those being used in non-containerized clusters/groups.

These trends reflect how Kubernetes (whether EKS, KOPS or similar) and ECS are designed to utilize all available node resources, irrespective of size and type, in accordance with pod/task requirements.

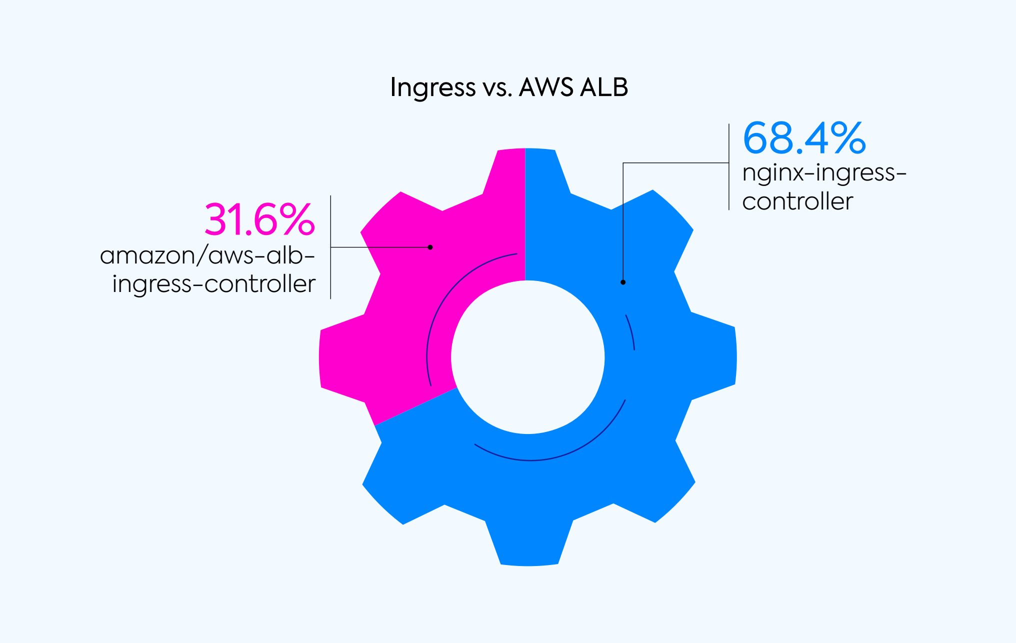
This is in sharp contrast to how non-containerized clusters with classic autoscaling groups operate, typically sticking to smaller, homogenous instance types. In these clusters, if there are multiple instances of varying sizes, the collective average capacity might be perceived by the load balancer to be within the defined range (e.g. 80% overall CPU utilization). As such, it will distribute traffic equally between the different instances, potentially overwhelming already struggling, smaller instances while leaving the larger instances underutilized.

CNI (container network interface) distribution



We find that the majority of the clusters are running on AWS with the amazon-K8s-cni. Flannel and Calico take second and third place respectively. What is interesting to note is that even though KOPS has a slightly larger market share than EKS (as seen above in the section on container orchestrators being used), KOPS users still seem to prefer using the AWS-native CNI.

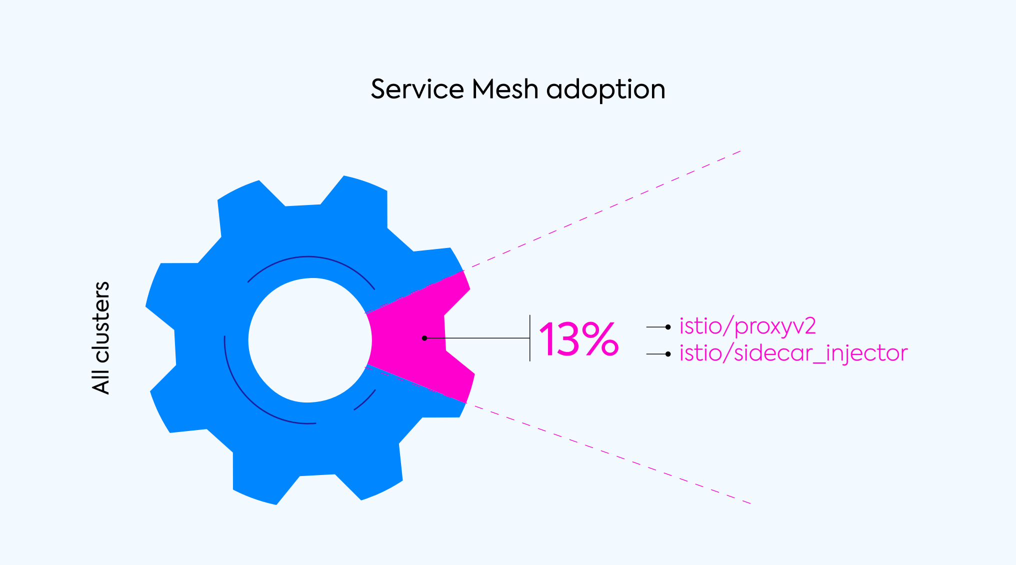
Load balancing inside Kubernetes clusters



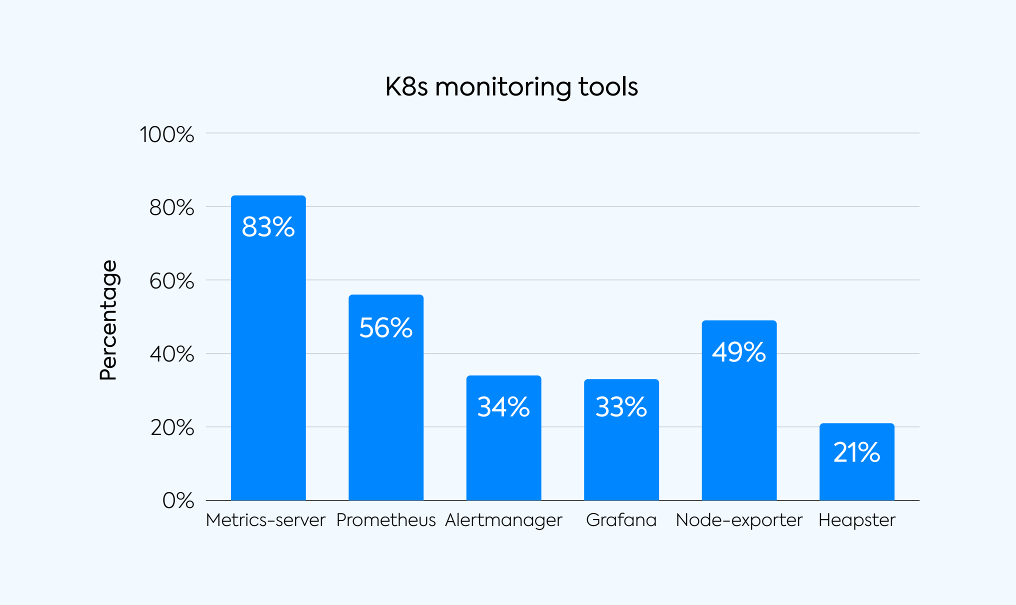
Ingress allows for rule-based traffic routing within Kubernetes without the need to create a load balancer per service. Our data indicates that roughly 70% of Kubernetes clusters are using Ingress. The most commonly used Ingress controllers are nginx-ingress-controller and aws alb-ingress-controller.

Service Mesh stats

Although service mesh is becoming popular, we see Istio and other types of service mesh installed on only 13% of all container workload clusters. This would indicate that service mesh is still in its infancy and quite possibly not yet as relevant as commonly thought.



Monitoring of Kubernetes clusters

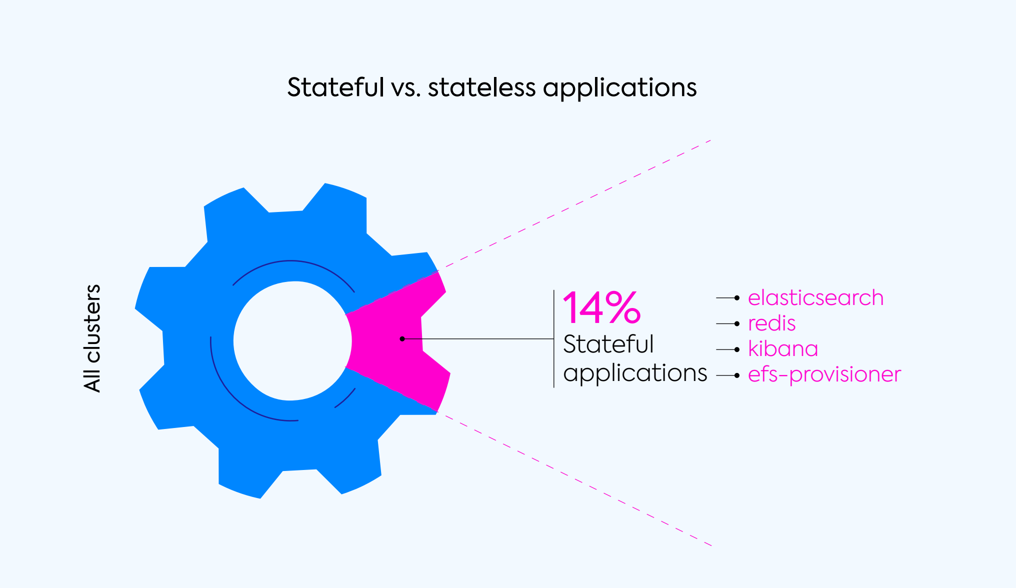


When analyzing monitoring, we've found that more than 50% of our customers are managing Prometheus **inside Kubernetes** (and not as an installation outside of the cluster). It's interesting to note that 30% of the Prometheus installations (which reflect 60% of all the running installations in Kubernetes) are managed by CoreOS Prometheus Operator.

Metrics-server has the broadest usage and is installed on more then 80% of the clusters. Incredibly, it seems that more than 20% of deployments still have heapster installed despite it having been deprecated since Kubernetes version 1.13.

As Kubernetes continues to grow, observability tools will play a key role in staying on top of underlying infrastructure, system application behaviour and any unusual incidents.

Stateful applications running in Kubernetes



Helm

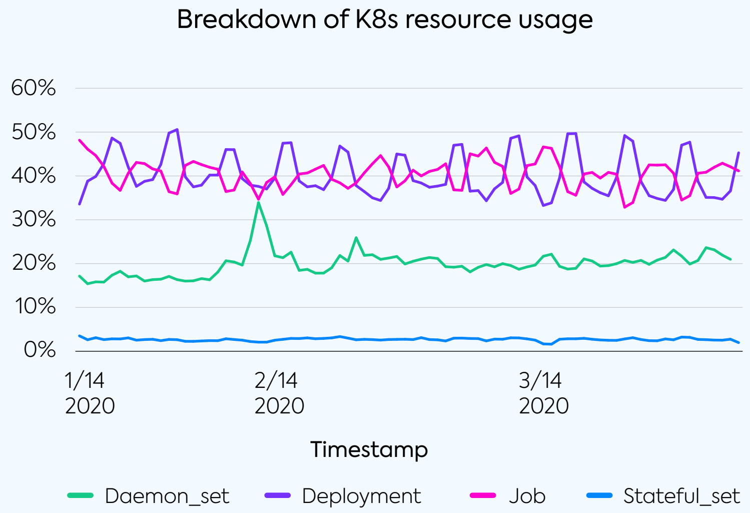
58%
customers

More than 58% of the customers examined are using helm to deploy applications to their clusters...with the remainder comprised of other automation tools.

Despite stateful applications requiring more sensitive handling for data persistence, we still see growing numbers of stateful workloads in Kubernetes clusters with Elasticsearch, Kibana, Redis and efs-provisioner most prominent.

While overall these numbers are low, likely due to the misconception that containerized environments are mainly useful for running stateless workloads, the data we see above would indicate that as awareness grows about the possibility of running stateful applications on Kubernetes, the actual number of workloads will grow as well.

Cluster distribution of Kubernetes objects



When it comes to Kubernetes objects running on spot instance clusters, we've found that jobs and deployments are the most common, alternating activity during day and night periods. While improvements are being made around stateful applications in Kubernetes StatefulSets (STS), it still is a relatively smaller portion of activity but as mentioned earlier, will likely grow in volume.

Amazon EKS Workshop

Get started with Kubernetes and spot instances

Amazon's very own EKS Workshop highlights the natural affinity between spot instances and containerized workloads and includes modules on using them for EKS workloads, including an entire section on Ocean by Spot.io which automates and optimizes everything from ensuring sufficient resources for pod requirements to leveraging the best range of instance types and pricing models, thereby helping AWS customers with increasing their usage of spot instances for EKS and other containerized workloads.

Summary

In summary we see that spot instances are being used for an increasingly broad variety of workloads and that in particular, they are a great fit for cloud-native applications such as those running on EKS, KOPS, ECS and others.

We also see that in addition to all the advantages commonly associated with containerized workloads, such as portability, component reuse, efficient hardware utilization, and greater security, when it comes to instance type diversification, users of container orchestrators are able to leverage a much broader range of instance types and sizes than traditional autoscaling workloads.

Finally, growth in spot instance usage is happening rapidly, at scale and across a variety of workloads and demands. This makes it clear that this pricing model, once considered relevant only for non-essential or test environments, actually is playing an essential role in optimizing cloud spend in mission-critical and production systems, not only in times of business volatility, but even during stable periods.

More information on spot instances

AWS introduces EC2 spot instances in 2009
What are EC2 spot instances
What is spare cloud capacity

7-17jun20