White Paper

DevOps guide to Kubernetes infrastructure automation

Spot by NetApp

Introduction

Even as organizations become more mature in their Kubernetes operations, achieving scale and speed by running containers on the cloud, many DevOps teams are still burdened with the task of managing the underlying infrastructure to support their applications. Kubernetes and containers both add a layer of abstraction from the underlying infrastructure, with neither layer have much visibility into the other. This decoupling has made it increasingly difficult to deploy and use containerized applications successfully.

Kubernetes is a powerful tool for managing the deployment and lifecycle of containers, but it doesn’t actually manage the cloud infrastructure that containers are running on. It scales pods and containers as long as there are healthy nodes for them to run on, but leaves provisioning and management of infrastructure up to the user to solve. Often, this means DevOps engineers, and even application developers, take on manual tasks of operationalizing and optimizing container infrastructure.

This white paper will outline the challenges of common scaling infrastructure practices on Kubernetes, identify key areas where automation can be applied, and present Ocean by Spot as a solution to do so through container-driven autoscaling.

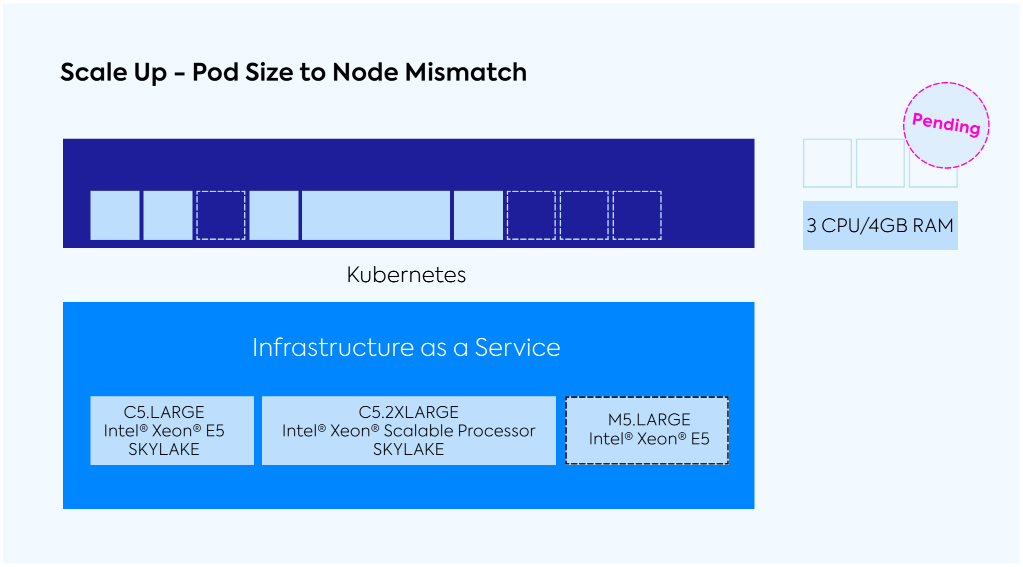
The trouble with scaling in Kubernetes

With Kubernetes, everything scales by pods. Kubernetes natively offers pod scaling services (horizontal and vertical pod autoscaling), and while it will schedule a pod to run on any node that meets its requirements, it doesn't automatically scale infrastructure.

Horizontal Pod Autoscaler (HPA)	Vertical Pod Autoscaler (VPA)
Both the HPA and VPA are fed by the metrics server, which reports CPU and memory utilizations	
HPA measures metrics on deployments and automatically scales the number of pods available in a cluster by replicating them across the environment. As cluster complexity grows, **Ocean** observes scaling decisions made by the HPA and ensures that the cluster has the most optimized infrastructure in real time.	Allocates more or less CPU and memory to a single pod. **Ocean** offers its own vertical container auto scaling solution, which measures in real-time the CPU/memory of pods and provides revsource suggestions based on cluster consumption.

Kubernetes users can configure the open source Cluster Autoscaler to automatically adjust the size of a cluster and add more resources if there are pods waiting to be run. However, there are limitations to using this DIY tool, especially for users who are looking to take a more hands-off approach to infrastructure.

When configured right, a tool like Cluster Autoscaler can ensure that there are enough nodes for pods to run on, but it can also result in significant efficiencies since Kubernetes doesn't care about the type or size of instance and will schedule a pod on any healthy, available node. For example, in the diagram below, a pod is waiting to be scheduled. The underlying infrastructure has enough space for the pod, which needs 3vCPU and 4GBs of memory, but no single node has enough capacity. Since a pod can only run on a single node, it will wait to be scheduled until one with enough capacity becomes available. This delay could potentially translate into an interruption of service to the customer and wasted resources.


Scale Up - Pod Size to Node Mismatch
Pending
3 CPU/4GB RAM
Kubernetes
Infrastructure as a Service
C5.LARGE Intel® Xeon® E5 SKYLAKE
C5.2XLARGE Intel® Xeon® Scalable Processor SKYLAKE
M5.LARGE Intel® Xeon® E5

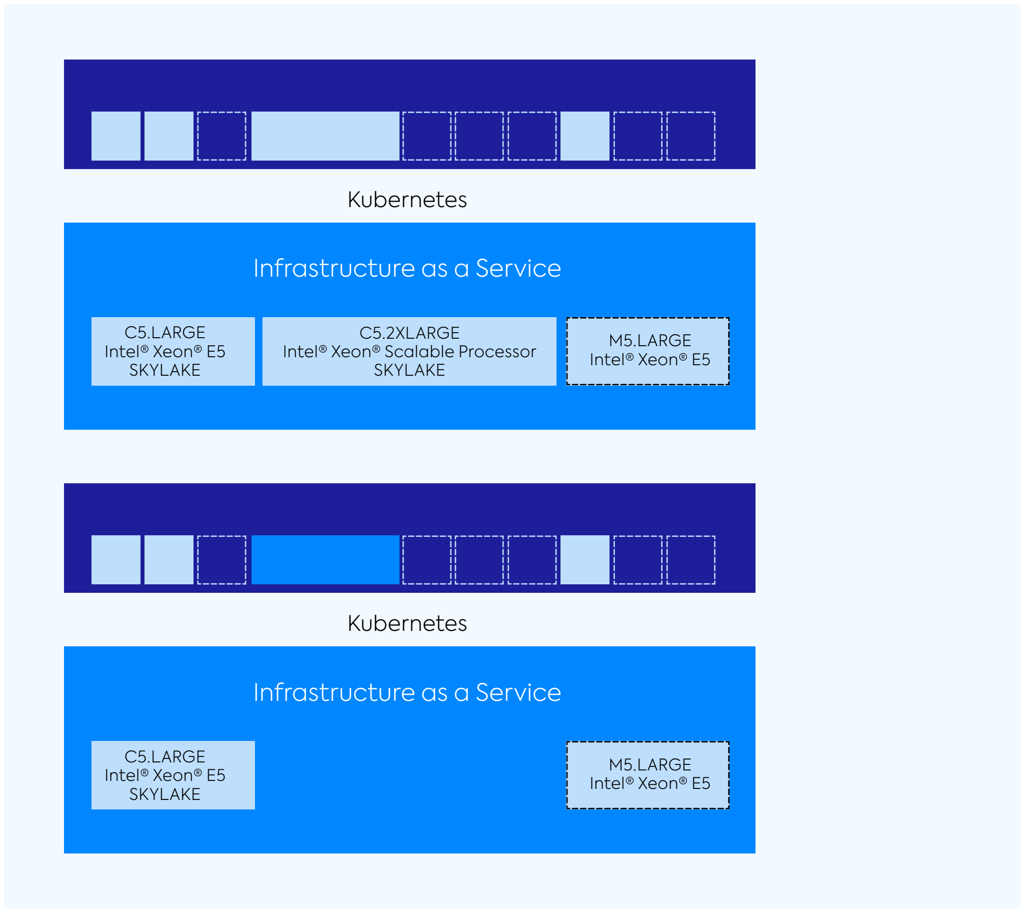
You can try to get ahead of this lag by **adding different types of instances to match different types of pods.** But if you bring that same pod from the graphic above to a cluster that is not yet fully provisioned and spin up the wrong instance, your pod will stay unscheduled.

Leveraging different instance types and sizes is limited when using Cluster Autoscaler and Autoscaling Groups. Instances need to have the same capacity (CPU and memory) if they are in the same node group. Managing multiple node pools is complex and ASGs need to be managed independently by the user.

80%

80% of instance types used in Kubernetes deployments are made up from 2XL and 4XL instances sizes

Scaling down applications also presents infrastructure inefficiencies, and can result in over provisioned nodes. When traffic is low during off hours or night time, it makes sense to reduce capacity. In the case below, there are only a few pods running across a lot of infrastructure.



Kubernetes
Infrastructure as a Service
C5.LARGE Intel® Xeon® E5 SKYLAKE
C5.2XLARGE Intel® Xeon® Scalable Processor SKYLAKE
M5.LARGE Intel® Xeon® E5
Kubernetes
Infrastructure as a Service
C5.LARGE Intel® Xeon® E5 SKYLAKE
M5.LARGE Intel® Xeon® E5

In this scenario, it would be better to take away the C5.Large or M5.Large, rather than the C5.XLarge. The pods on those smaller instances could have been rescheduled on another node, and the cluster would have remained efficient, running on only what it needs, with all pods scheduled. These kinds of mismatches between pods and nodes hampers Kubernetes' inherent agility. To be at maximum efficiency, infrastructure should benefit from the same scaling flexibilities as containers.

With little to no visibility into infrastructure however, Kubernetes can't make these decisions. A different approach is needed, where pods and containers are treated as the unique entities that they are, and infrastructure automatically responds to their needs.

This is the concept of container-driven autoscaling, which uses real-time container requirements when provisioning infrastructure instead of trying to fit containers into pre-determined or pre-existing instances. Container and pod characteristics, including labels, taints and tolerations, define the kind of instances that it gets matched to.

Easier said than done, however. Autoscaling groups are inherently infrastructure-focused, making decisions based on scaling policies and resource utilization, not what your containers actually demand. Without well-defined monitoring however, it's difficult to gauge what those needs actually are, and many DevOps teams simply don't have the the resources to continuously observe containers and provision infrastructure appropriately.

Ocean

As a fully managed data plane for container orchestration, Ocean by Spot is inherently container-driven. With Kubernetes, Ocean connects to your cluster's control plane via a controller pod, and continuously monitors the status of the cluster, its resources and implements changes accordingly. Events are observed at the Kubernetes API Server, giving Ocean visibility that enables accurate provisioning for performance, and up to 90% lower cloud infrastructure costs.

Right-sizing containers for optimized container utilization

There are more than
1.7 million
possible compute instance combinations to choose from

While the concept of container-driven scaling is a paradigm shift that can ensure infrastructure meets application requests, for Ops teams, scaling infrastructure is just one piece of the puzzle. If pods are asking for more infrastructure than they actually use, it doesn't matter how efficiently your infrastructure scales, you'll still end up paying for more compute than you need.

Kubernetes provides users with the option to define resource guidelines for containers, based on specific CPU and memory needs. Developers will often attempt to configure resource requests based on a guess (can be inaccurate), trial and error (can be extensive) or simulations with a test deployment (can be ineffective as test metrics often differ from production usage).

Incorrect provisioning can lead to idle resources and increase operational costs, or can create performance issues within your application because a cluster doesn't have enough capacity to run. Most organizations are unwilling to risk performance, and in order to be prepared for a scaling burst, will typically do one of two things:

Overprovision clusters with more resources than needed, resulting in spare capacity that you're paying for, but that often sits idle.

Pod Priority indicates the importance of a pod relative to other pods, and schedules them as such. Lower priority pods may remain unscheduled.

70%
cloud cost wasted

As much as 70% of cloud costs are wasted[1]
– Gartner

Ocean

With Ocean, users are presented with better right sizing and container utilization approaches that help to reduce container costs. In order to more accurately estimate and apply the right resource requirements to containers, Ocean's vertical container auto scaling feature measures, in real time, the CPU and memory of pods and provides actionable recommendations to improve resource configurations.

Users also have the option to configure adjustable headroom, a unique concept introduced by Spot, to better provision spare capacity for fast scaling. Adjustable headroom is a spare capacity unit that acts as a placeholder for workloads, configured based on the characteristics of applications running in the cluster. This kind of intelligent overprovisioning is a cost effective way to limit idle resources and ensure pods always have a place to land.

1) https://www.infosys.com/about/knowledge-institute/insights/rationalizing-cloud-costs.html#:~:text=Gartner%20estimates%20that%20up%20to%2070%25%20of%20cloud%20costs%20are%20wasted.&text=Financial%20CapEx%20models%20or%20manual,CapEx%20rather%20than%20OpEx%20model

Leverage spot instances for lower cost computing

As cloud applications scale and grow, so does the cost of cloud operations, exacerbated by the challenges we laid out above. DevOps teams and cloud architects are now being tasked with considering cost savings as part of their strategies. While the cloud providers offer deep discounts with spot instances, often they are not leveraged because of the risk of termination and complexities of managing cluster running on spot instance.

On-demand is the **easiest to set up** and implement, but its flexibility comes at a higher cost than other options.

Reserved instances are more **cost effective**, but the dynamic nature of containers make it difficult to make long term commitments.

Spot instances are the **cheapest way to buy** compute capacity, but require a significant amount of configuration, maintenance and expertise to ensure availability.

Spot instances offer users up to 90% cost reduction compared to on-demand pricing, but their low cost comes with the caveat that cloud providers can take capacity back whenever they need it. Interruption of your EC2 instances can impact and degrade services, or result in data loss, making developers weary to work with them on mission-critical, production workloads. With intelligent orchestration, however, spot instances can be a good fit for container workloads since these are typically fault tolerant and highly available.

For companies that want to take advantage of the discounted pricing of spot instances, Ocean uses predictive analytics to identify and anticipate spare capacity interruptions, and proactively replaces at-risk instances with new ones. Machine learning and analytics also enable Ocean to implement the optimal mix of pricing options while sustaining highly available applications.

Conclusion

It’s clear that there is a path to automated scalability when users leverage Kubernetes and containers in the cloud--if they can overcome the operational complexities of managing cloud infrastructure.

At Spot, our customers are facing the same challenges and we’ve worked to create a solution that addresses infrastructure management with containers, following the approaches laid out above. Ocean is a container-driven data plane management platform that automates capacity provisioning and scaling so developers can focus on applications, not on infrastructure. Using these intelligent features for auto scaling, right sizing and capacity purchasing, Ocean is able to help customers realize up to 90% savings on cloud infrastructure, and give their teams a simpler, efficient way to scale out their containers and applications.

More information on containers and Kubernetes

Ocean on AWS EKS Workshop

Limitations of Cluster Autoscaler