

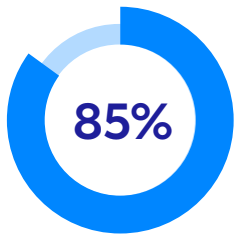
CloudOps Guide

---

# Automating Kubernetes infrastructure

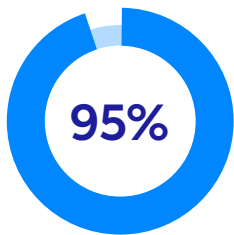
Essential automation tools and practical guidance for building & maintaining scalable, resilient & cost-efficient cloud-native infrastructure





## Managing infrastructure doesn't deliver value

DevOps teams feel that managing infrastructure gets in the way of delivering value



## Lack critical automation

DevOps don't have effective container orchestration in place

# Introduction

Kubernetes makes it easier to run and deploy applications in the cloud, delivering significant benefits like speed, agility and cost efficiency. But to unlock those strategic advantages, you'll need to navigate the operational hurdles of managing and scaling these applications and the infrastructure underpinning them.

Kubernetes 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 take on manual tasks of operationalizing and optimizing container infrastructure.

If DevOps and platform teams are to avoid becoming completely overwhelmed, it is critical to think about the long-term operational impact of maintaining and scaling clusters.

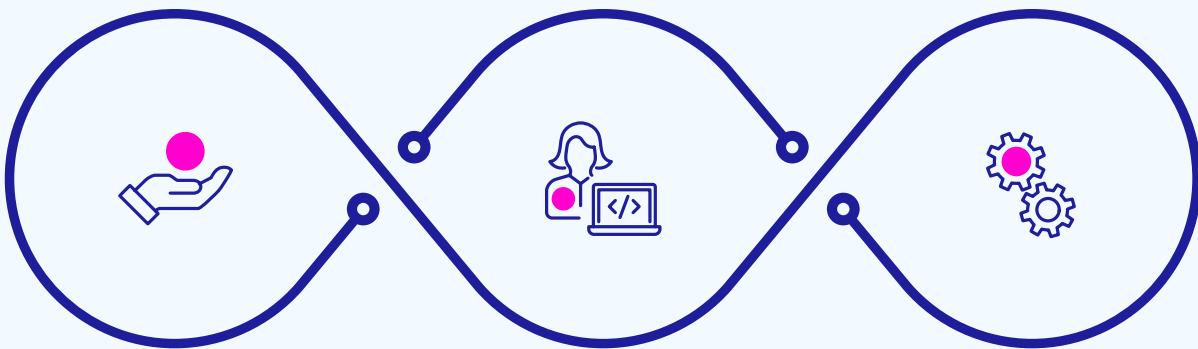
**For sustained success, and to reap the benefits of the cloud in the long term, users need to find ways to effectively and efficiently automate cloud native operations to deliver infrastructure that is performant, highly available and always meets application demands.**

Let's start by looking at some of the biggest challenges that automation must address when it comes to managing containerized infrastructure.

# The must-solve challenges of Kubernetes operations

While Kubernetes handles the application scaling and deployment, it doesn't handle the underlying cloud infrastructure, nor does Kubernetes and cloud infrastructure speak the same language. This gap is a fundamental challenge to building properly functioning Kubernetes environments that can scale with the needs of your business.

## Key concerns that ops teams are looking to address with Kubernetes infrastructure



### High availability

The ability to service SLOs and ensure minimal to no downtime during upgrades. This also involves fast recovery during inevitable failures.

### Developer enablement

A key task of DevOps and platform teams is to provision environments for Dev, QA, and Prod in a quick, self-service manner.

### Resource utilization

DevOps teams need to ensure there is enough capacity for peak times, and only the required number of instances are being utilized at any given time.

# Bridging the gap between infrastructure & workloads

Workloads describe themselves in ways that are completely different from how infrastructure describes itself. This mismatch creates a challenge that needs to be solved when building an optimized Kubernetes environment that can scale with your business. Infrastructure is usually described in terms of size, families and types of pricing—there are small, medium, large, and extra-large sized instances. They are available on demand, as reserved instances, or as excess capacity (such as spot instances). Meanwhile, Kubernetes workloads speak in terms of CPU/Memory/GPU resource requests and limits, labels, taints, tolerations, network & storage requirements, and affinities.



## Understanding compute purchasing models

### On-demand instances

Scales up and down for variable demand, and has expensive per-unit costs.




### Reserved capacity

Committed future use that is more affordable per unit but less flexible.




### Spot instances

Discounted spare capacity that can be taken away at any moment.

## Compute capacity

-  Instance size
-  Instance family
-  Instance type

## Container requests and consumption

-  vCPU
-  Memory
-  GPU

To bridge the gap between infrastructure and workloads, organizations need to find ways to abstract infrastructure. They need to spend less time on undifferentiated maintenance tasks, and more time running workloads.

# The DIY Kubernetes data plane automation stack

Many companies have taken a DIY approach, using open-source and cloud provider tools to manage container infrastructure. While these require DevOps teams to learn and operate various tools and processes necessary for building healthy Kubernetes environments, they do offer a route to **automating some of those critical infrastructure tasks, such as:**



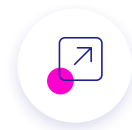
Scaling infrastructure up and down automatically



Using excess capacity reliably for cost savings



Handling interruptions



Rightsizing resources across pods



Maintaining capacity for scaling bursts



Matching workloads with the right instances

## Cluster-autoscaler:

### Scaling infrastructure to meet application demands



#### Challenge

**Kubernetes doesn't handle infrastructure. It just knows the number of registered and healthy nodes in the cluster and allocates workloads to them.**

This means that once all available nodes are fully allocated, Kubernetes will keep additional workloads pending until new infrastructure is present in the cluster.

That makes it the user's responsibility to scale clusters in and out to accommodate pending workloads, leading to inefficiency, hands-on work and the potential for failures.



#### Solution

**Cluster-autoscaler is an open-source tool that is commonly used to bridge the gap between Autoscaling Groups (AWS) and Kubernetes.**

It automatically adjusts the size of your cluster when resources are insufficient to meet dynamic workload requirements, or when a node is underutilized and its pods can be placed on an existing node.

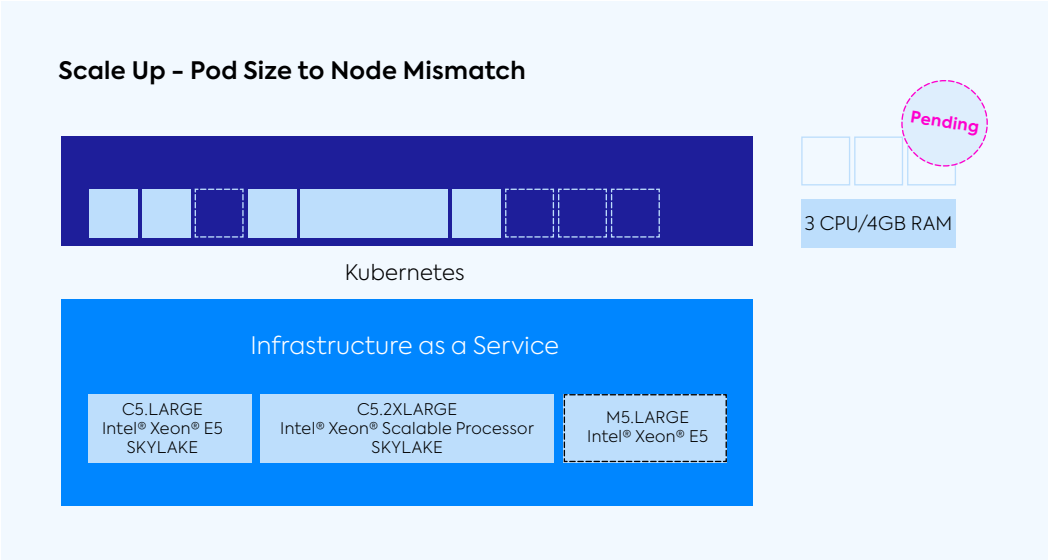


#### Need to know

**While cluster-autoscaler is a powerful automation tool, the user requires a keen understanding of its operational capabilities to prevent issues, and it still falls on the operations team to establish processes to handle failures and manage overprovisioning.**

If your application utilizes different compute types, you'll also have to manage multiple node pools.

# Kubernetes scaling challenge: Provisioning infrastructure is the user's responsibility



In this diagram a pod is waiting to be scheduled. The underlying infrastructure has enough space for the pod, which needs 3vCPU and 4GBs of memory, but there isn't a single node with 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.

## Node pools:

### Scaling in a complex, multi-cluster environment



#### Challenge

As cloud applications and services grow, with clusters of many microservices running across regions and availability zones, **it is essential to find a way to scale operations in a way that minimizes complexity and inefficiency. By managing nodes as groups rather than individually, software delivery teams can take more control.**



#### Solution

**Node pools are a group of nodes in a cluster that all have the same configuration.** Different node pools serve different purposes. Each node pool would have different instance types, sizes, and families to cater to changing needs after a cluster is created.

Cluster-autoscaler can manage numerous node pools, giving you the ability to leverage its capabilities across the different compute types required by your application.



#### Need to know

**There are limitations with node pools that need to be considered.** Within a node pool all nodes need to be of the same size. Additionally, if a cluster spans multiple availability zones or regions, changes in one node pool are replicated across all zones and regions. This can consume a lot of resources. Similarly, when a node pool is deleted, it is deleted across all regions and zones.



## Node termination handler:

### Anticipating & planning for interruptions to workloads



#### Challenge

Excess capacity – such as AWS Spot Instances – offer a powerful way for organizations to scale large workloads cost effectively.

However, **the challenge with this is to balance the cost benefit with the potential sudden termination that is part and parcel of these excess capacity instances.** When this happens, the workloads on the instance needs to be drained or moved to another instance for uninterrupted processing.



#### Solution

**This is a job for the node termination handler.** It listens for interruptions and automatically transitions workloads from existing nodes to new nodes so that the workloads can continue execution and serve their purpose.



#### Need to know

It is the responsibility of the software delivery team to install node termination handler, manage updates to it, deal with bugs as they arise, scale, upgrade, or downgrade as needed. This added overhead can be a drain on productivity and makes routine maintenance inefficient.

## Cluster overprovisioner: Maintaining capacity for scaling bursts



### Challenge

When a cluster has maxed out all the underlying instances provisioned for it, a new instance needs to be provisioned, which can take two minutes or more. During this time, the excess pods go into a pending state until the new instances are ready to host them. **The few minutes it takes to spin up a new instance can significantly impact time-sensitive workloads and even cause end users to experience service disruption.**



### Solution

**Cluster-autoscaler has no notion of overprovisioning, so to avoid this pending state, it can be used in conjunction with cluster overprovisioner**, which allocates a buffer of free instances that can be utilized when a cluster reaches its limit.

It does this by using Kubernetes' native capability to assign priority to pods. In this method, low priority buffer pods are placed in the cluster, ready to be evicted and replaced with incoming pods of higher priority when they are scheduled.

Cluster proportional autoscaler can be added to this stack so that as more worker nodes are added to your deployment the available buffer size increases proportionally.



### Need to know

**Cluster overprovisioner allows scheduling of low priority pods, but it has no intelligence about how to deploy workloads.** It can simply provision infrastructure resources without actually knowing what workloads require. For example, it doesn't understand which workloads are constantly scaling up or down, and it has no context on how to allocate resources. In this case, the software delivery team needs to build in logic for cluster overprovisioner to make decisions on how to better allocate workloads on the available buffer instances. In other words, deployments need to be clearly defined and the necessary overprovisioned resources need to be available.

## Vertical Pod Autoscaler (VPA): Rightsizing resources across pods



### Challenge

Since Kubernetes doesn't "see" infrastructure or adjust requests based on actual usage, **it's common to get into a situation where pods are either consistently requesting unneeded resources or requesting insufficient resources.**

The first issue results in inefficiencies that will grow with your environment, the second can result in pods going unscheduled or being killed, resulting in end-user disruptions.



### Solution

**Vertical Pod Autoscaler (VPA) is an open-source tool to manage resource requests and limits for containers within a cluster to eliminate waste, improve efficiency and prevent performance issues.**

It does this by analyzing the historical resource usage of pods and automatically rightsizing resource requests and limits accordingly.



### Need to know

The challenges with VPA are when updating the available resources for a pod. VPA restarts all containers in the pod during an update and may even reprovision these containers on a different node. At times there can be conflicting policies for a single pod, and these conflicts will need to be manually resolved.

## Bin-packing: Matching workloads with the right instances



### Challenge

We've already discussed how important rightsizing is to prevent overprovisioning resources and improve efficiency.

While rightsizing focuses on aligning resource requests with actual usage, bin packing focuses on rescheduling pods on underallocated nodes to drive greater efficiency without affecting performance.



### Solution

Collecting metrics comparing requested resources to allocable resources – using either native cloud provider monitoring tools or open-source options like Prometheus – can help gain valuable insights.

Cluster-autoscaler comes with some bin packing capabilities, which kick in when an instance is 50% underallocated.



### Need to know

While metrics offer one route to improving utilization, it still leaves us a long way from automated, proactive bin packing that actively reschedules pods on nodes with any free capacity to host them. While cluster autoscaler's automated bin packing capabilities are welcome, the 50% threshold means the prospect of considerable unused capacity remains.

# The risk of a DIY Kubernetes data plane management stack

In a typical organization, the ratio of DevOps engineers to application developers is imbalanced. A software delivery team of ten people typically supports hundreds of application developers. DevOps engineers are stretched thin and have an ever-growing list of developer requirements and Jira tickets to be responded to. Even if a highly productive software delivery team has initial success and a few quick wins, the ever-changing demands of the business will become overwhelming. Add to this the shortage of high-quality DevOps engineers, keeping the team continually well-staffed is a challenge. For all these reasons, a DIY Kubernetes management stack may just about do the job but is far from ideal as operations scale.

The alternative is to opt for a commercial container infrastructure platform like Ocean from Spot by NetApp, which handles the infrastructure plumbing, and allows software delivery teams to enable better developer experiences.

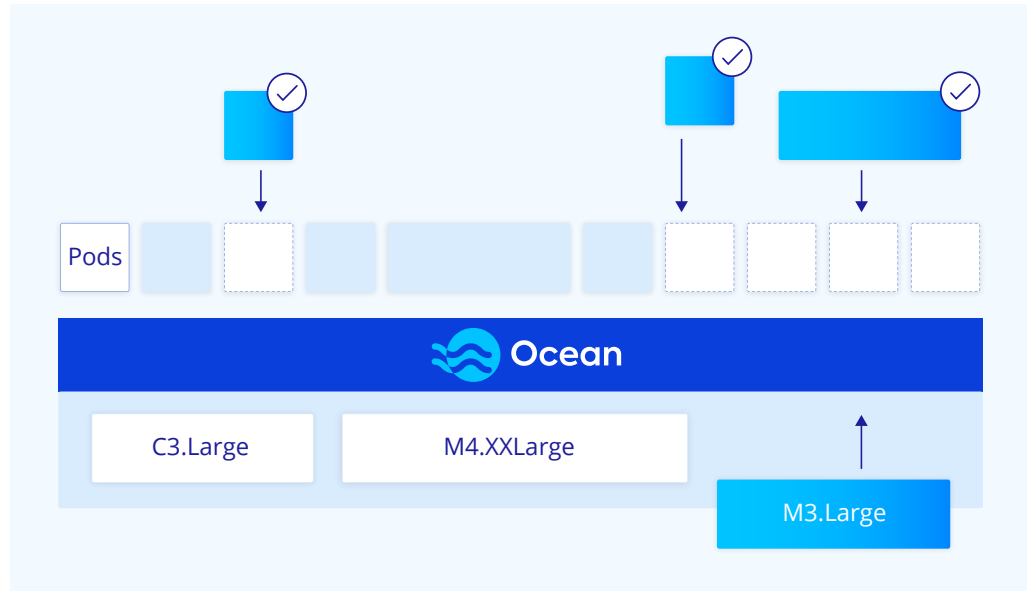


## Automated infrastructure engine for containers

Spot Ocean frees up software delivery teams from manual maintenance of Kubernetes clusters, and enables them to focus on improving workflows, and providing a better developer experience for teams that rely on them. It continually analyzes container infrastructure to drive better resource allocation and save cloud costs in the process. Ocean can integrate with all major managed Kubernetes services, and lets you get the most out of each of them.

### The key components of Spot Ocean:

- **Ocean autoscaler:** Automatically scales the number of nodes up or down making sure there are enough resources to run workloads.



Autoscaler takes a container-driven approach where the infrastructure is scaled according to the Pods' requirements and constraints.

- **Virtual Node Group:** VNGs allow users to configure multiple types of node pools on the same cluster. They provide a layer of abstraction for different types of workloads. Ocean enables the use of multiple machine types, sizes, availability zones, and life cycles within the same cluster. Additionally, it can leverage excess capacity instances seamlessly, managing their interruptions by draining resources to new instances immediately.
- **Ocean-controller:** This handles interruptions to workloads and ensures that underlying infrastructure unavailability or failure doesn't impact running workloads. It acts as the agent that is located in the cluster and sends relevant data to Ocean. This data is used to perform actions such as draining excess capacity instances that are about to be terminated and moving its workloads to a new instance.

- **Headroom:** Spare capacity, or headroom, is maintained to ensure that compute capacity is instantly available to handle scaling bursts. This feature allows the user to set a buffer of resources to allow Ocean to dynamically adjust infrastructure to the demands of applications. By understanding the history, metrics and demands of an application overtime, Ocean maintains the performance and stability of your applications during traffic spikes while limiting overprovisioning.
- **Cluster roll:** Allows updates to the entire cluster or just part of it in rolling batches. The benefit is uninterrupted workloads.
- **Right-sizing:** Ocean analyzes the usage of CPU and memory of container instances continually and provides recommendations on what should be the requested resources on a container level. It does this by comparing the current requests against the actual consumption of resources in each container. This results in cost savings and better resource utilization.
- **Cost analysis:** Ocean tracks cloud spend and enables cost showback to each team and application so you can have better visibility into cloud expenses and optimize them as needed. As you scale your usage of Kubernetes beyond Day 2, you need not have your software delivery team spend its best time managing Kubernetes plumbing tasks. Instead, leverage a serverless container infrastructure platform like Ocean and free your software delivery teams to better empower your application developers.



**Get hands-on with the leading cloud infrastructure automation & optimization solution today!**

Connect to [Spot Ocean](#) and start your free trial today.