Spot by NetApp

Guide

Autoscaling and 6 other Kubernetes automation challenges

Containers and Kubernetes make for a powerful combination to scale applications in the cloud, but they also introduce significant challenges and complexities that organizations must solve for if they want reliable, efficient container environments that support highly available applications.

Automation for container infrastructure has proven to be one of those challenges, even as organizations mature in their Kubernetes operations. Existing tooling and processes for scaling cloud infrastructure leaves much to be desired by the DevOps engineer, who still bears the burden of many manual operating tasks.

The major cloud providers do offer customers core automation capabilities for deployment, management and scaling of infrastructure through AWS Auto Scaling Groups (ASGs), GCP Instance groups and Azure Scale sets. However, these have some limitations, while using them effectively requires expertise that many organizations don't have.

Here, we'll identify and outline hurdles to automation in Kubernetes operations that are preventing you from building dynamic, self-healing and self-reliant infrastructure your containers need:

1. **Container-driven autoscaling**
2. **Managing Autoscaling Groups and Cluster Autoscaler**
3. **Using mixed instances and multiple availability zones**
4. **Blue/Green and rolling upgrades**
5. **Right-sizing application resources**
6. **Avoid scale out blackouts during Spot market capacity churn**
7. **Maintain spare infrastructure capacity to service dyanamic**

1 Container-driven autoscaling

Kubernetes offer native scaling services for its pods and containers, but it doesn't automatically scale infrastructure. As long as there is capacity for healthy pods to run, Kubernetes will scale and it's up to the operators to ensure enough compute has been planned for and provisioned. This approach doesn't account for actual resource utilization of containers, and can often result in significant inefficiencies within your cluster. Instead, infrastructure should benefit from the same flexibility as the containers it supports, and real-time container requirements should determine how infrastructure is provisioned. This container-driven approach to autoscaling ensures that applications have the right resources to scale as much and as quickly as they need.

2 Managing Autoscaling Groups and Cluster Autoscaler

AWS Auto scaling groups (and similarly Scale Sets/Instance Groups) are inherently infrastructure focused and rely on advanced capacity planning. Users try to estimate what their applications will consume, and then configure their ASGs, which then work to maintain health based on user-set scaling policies. Scaling decisions made by ASGs are based only on infrastructure considerations (i.e. the number of nodes and resource utilization) so users often need to maintain another management layer for their containers—one that makes decisions based on the state of the workload.

The open source Cluster Autoscaler can be implemented with ASGs as a do-it-yourself solution that adds more instances to your cluster when resources are insufficient to meet dynamic workload requirements, however Cluster Autoscaler has its own set of limitations.

Cluster Autoscaler will automatically add more nodes to your cluster but overprovisioning is common and flexbility in instance types are limited

3 Using mixed instances and multiple availability zones

Cloud providers offer a variety of instance families that range in attributes like RAM, CPU, and disk resources. Having a mixed instance strategy can help ensure maximum availability, flexibility, and efficiency however implementing mixed instances comes with some scaling hurdles. While you can scale up the instances in your ASGs based on metrics like CPU and network utilization, using multiple instance types results in inconsistent metrics since each instance is using a different kind of resource. An alternative is to use Cluster Autoscaler, which does allow for mixed instance types within a node group, but with the limitation that instances need to have the same capacity (CPU and memory).

Similarly, having instances that span across multiple availability zones is important for your application's availability and promotes redundancy, but in practice, a single auto scaling group cannot span AZs without consideration for rebalancing. Alternatively, you can manage one ASG per AZ.

It's common for users to customize their applications for specific environments, but Kubernetes doesn't provide any templating mechanism on its own to control common parameters and many tasks require manual configuration. For example, users can deploy their application using the kubectl, but to automate the process, it's best practice for users to configure a load balancer. When it comes to using computing power, users need to also configure resource requests on each pod. For instance, different ASGs are needed for different workload types (small, medium and large, at minimum) for each availability zone. Users will also need to configure seperate ASGs if they want to run their workloads using dfferent pricing models (i.e. spot and on-demand instances) or if you want to run GPU nodes.

4 Blue-green and rolling updates

The continuous delivery strategy blue/green deployments can help keep applications running and reduce risk when software updates are needed, but automating these processes for deployments in Kubernetes can be challenging. First, having two environments means DevOps engineers and SREs have to monitor and maintain both while the application is being upgraded. Users then need to determine and implement how they'll ensure newly launched nodes are healthy, what happens during a failure (rollbacks?), and how pipelines will be monitored, among other considerations.

Like blue/green upgrades, rolling updates enables your application to be updated quickly and without any downtime. In Kubernetes environments however, support for rolling upgrades is limited, regardless of Kubernetes provider. It's a resource intensive and complicated process for users to implement rolling upgrade especially in multi-cluster and multi-cloud container environments. Users have to take into account similar considerations as they do for blue-green updates, including how they will ensure the success of workload migration, how they will check for compliance and how to determine new versions of Kubernetes components.

What are blue/green deployments?

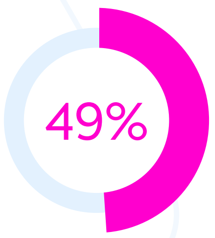
The blue/green deployment technique is used to reduce downtime and risk when updating software. Two identical production environments run, with only one live (blue) to serve traffic. As software is updated, testing takes place on the environment that is not live (green), and once ready, incoming requests are routed to green instead of blue.

5 Right-sizing application resources

Kubernetes provides users with the ability to define resource guidelines for containers based on CPU and memory needs, but it can be difficult to define and maintain these resource requirements for dynamic applications especially in fast scaling scenarios. Developers will often attempt to configure resource requests based on a guess (can be inaccurate), trial and error (can be extensive) or simulate with a test deployment (can be ineffective as test metrics often differ from production usage). Incorrect provisioning can lead to idle resource and higher operational costs or result in performance issues if the cluster doesn't have enough capacity to run on. Most organizations are unwilling to risk performance, and in order to be prepared for a scaling burst, will typically overprovision clusters or set pod priorities, leaving some pods potentially unscheduled.

6 Avoid scale out blackouts during Spot market capacity churn

While spot instances aren't for every workload, they have enormous cost saving benefits and can help make infrastructure run more efficiently if users can effectively mitigate the risk of using them. Running, monitoring and managing workloads on spot instances, however, requires specific expertise and significant time to limit disruptions. Spot market capacity churn during scale out periods is especially difficult to manage, as handling spot interruptions requires an understanding of and running node-termination-handler. In the case that request constraints can't be met (e.g. if you specifcy a maximum price that is below the current Spot price) or there is not enough capacity, scale up can be paused or delayed.



The majority of Kubernetes workloads are underutilizing CPU and memory, with 49% of containers using less CPU then their defined requests[1]

7 Maintain spare infrastructure capacity to service dyanamic user traffic

When starting a new node, it can take anywhere from 2 minutes to sometimes 30 minutes to boot up the node and declare it healthy enough to run workloads. This cold start keeps pods waiting to be scheduled, and can result in slow or interrupted services. Instead, maintaining spare headroom capacity at the infrastructure level can help users scale their infrastructure quickly, so when pods need to be executed they can be scheduled immediately. Users can configure this headroom using the open source cluster over-provisioner and cluster-proportional-autoscaler, but like other DIY solutions for Kubernetes infrastructure, it is a manual, time-consuming and complex process.

1. https://www.datadoghq.com/container-report/

Ocean

Automation made easy with Ocean by Spot

For all the ways that Kubernetes manages container orchestration, there is still much for the DevOps and Site Reliability Engineers to worry about when it comes to operations and infrastructure. Without expertise in the areas outlined above, Kubernetes environments can quickly become cloud cost centers, running inefficiently with higher risk for down time. However, Spot by NetApp offers Ocean, a Kubernetes data plane manager that automates and optimizes Day 2 operations. With Ocean, Kubernetes operators focus on running their workloads, while Ocean automates everything from container right-sizing and headroom, to blue/green and rolling updates.

Visit our Ocean product page to learn more about hands-free container infrastructure.