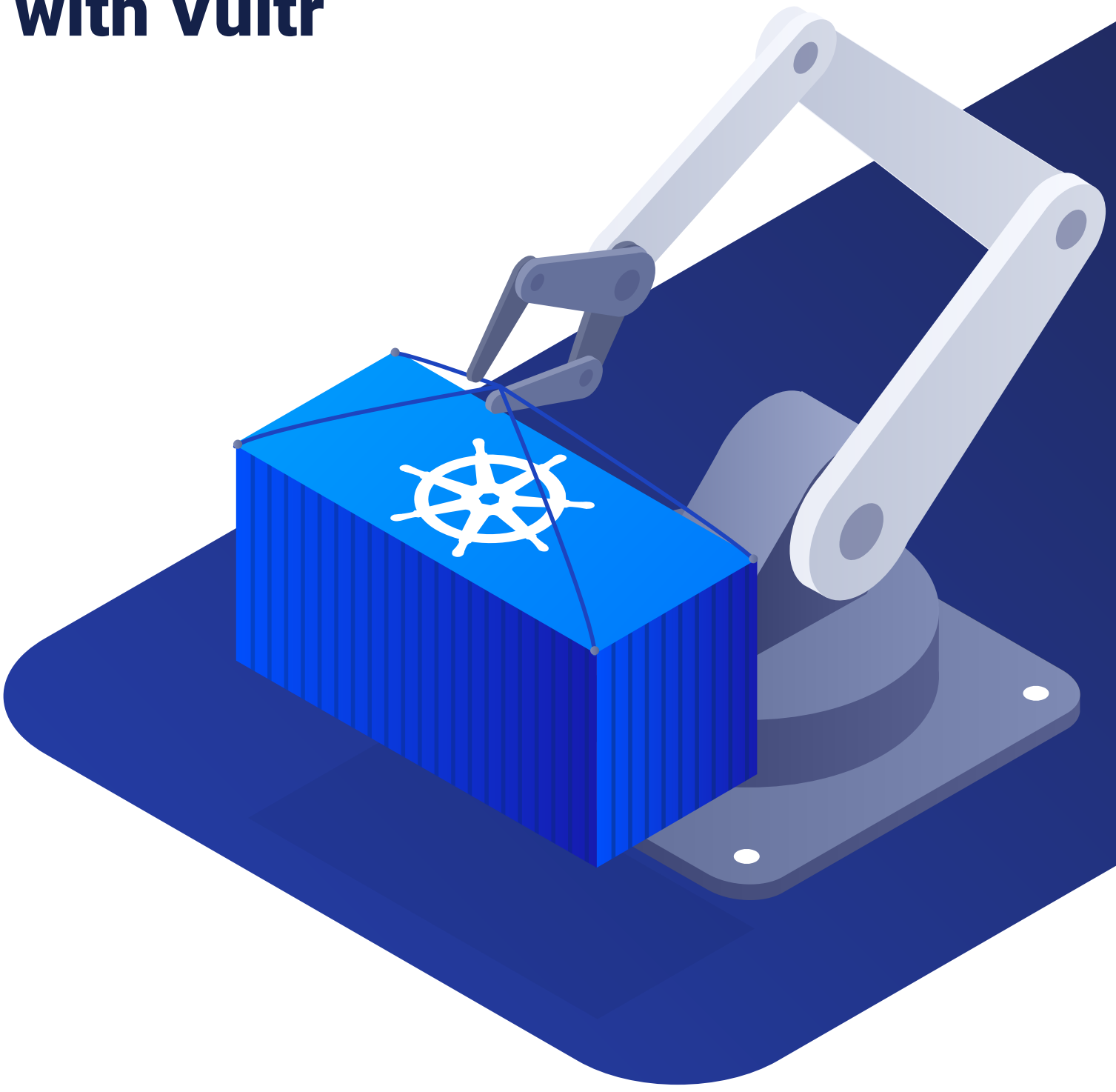
VULTR

The Developer's Guide to Kubernetes with Vultr



Kubernetes, an open-source system for managing containers at scale, was initially developed by Google and released in 2014. Today, Kubernetes is maintained by a vibrant community that includes many of the original engineers and those from other organizations with a vested interest in its long-term success.

Long before Kubernetes's inception, Google had adopted containers for their portability, efficiency, and isolation. Kubernetes evolved from over 15 years of experience managing containers at Google and the development of two previous container orchestration systems. Production workloads at their scale run billions of containers per week. The Cloud Native Computing Foundation (CNCF) was founded alongside the release of Kubernetes 1.0 with the mission of aligning the industry on leveraging containerization technology to build and run resilient, manageable, and observable applications.

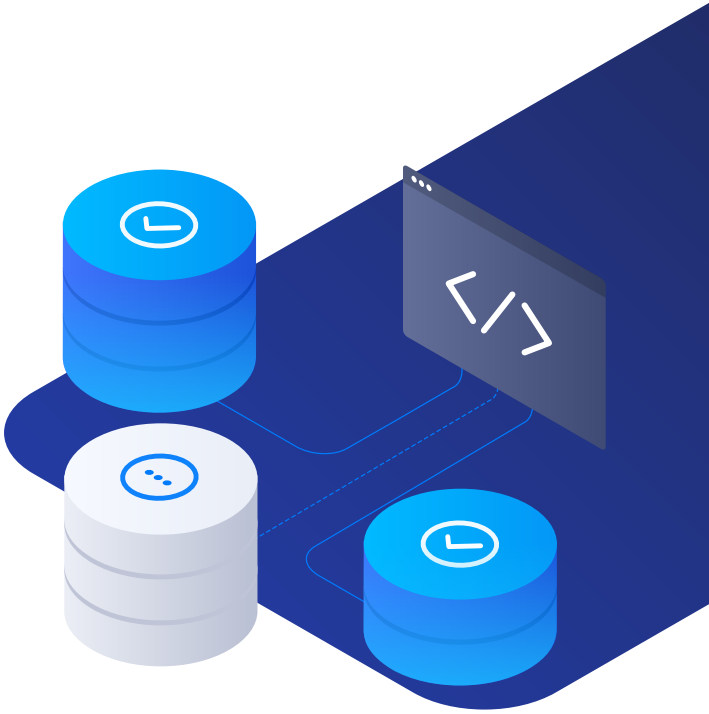
Unlike its predecessors, Kubernetes was built with a stronger developer bent. Its engineers aimed to make it easier for developers to deploy and manage complex, containerized, distributed systems. Its key features include automated rollouts and rollbacks, service discovery, load balancing, storage orchestration, secret and configuration management, self-healing, and so much more. Kubernetes was designed to facilitate extensibility, and a growing ecosystem of tools, applications, and managed services is evolving alongside it.

The complex architecture and extensibility of Kubernetes are what make it so powerful. However, there are so many moving parts that managing a system of such complexity can be intimidating. Many organizations and individuals who have adopted Kubernetes for their containerized environments have also attempted to solve these complexities.

Kubernetes is rapidly becoming ubiquitous in cloud computing. As a leading open-source solution, a vast and diverse community of support is available. Additionally, an increasing number of commercial support options have entered the market.

It's no surprise that the versatility Kubernetes offers is virtually unrivaled. But getting the most out of it is a challenging undertaking that often requires an expert touch.

Container Management Explained



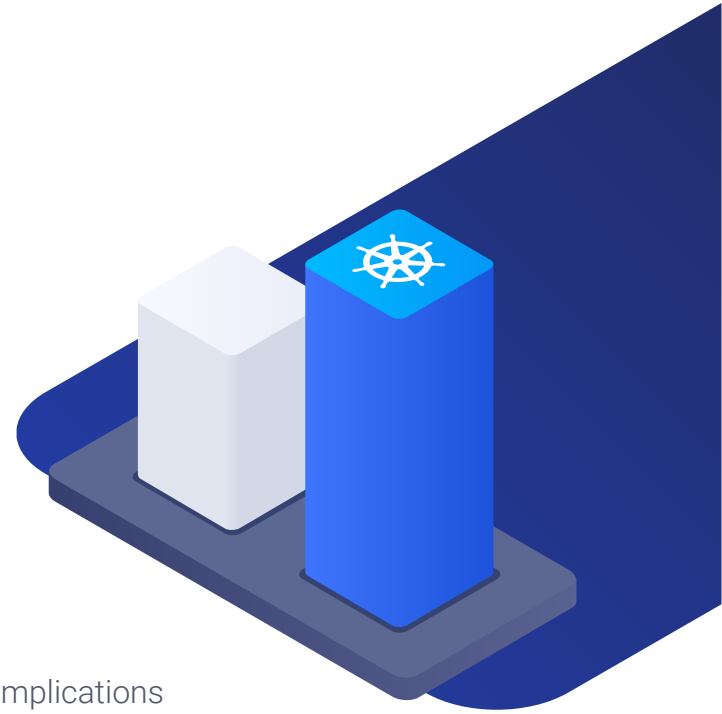
Containers have become a fundamental part of cloud deployment. They allow you to include everything you need to run an application – including code, settings, and dependencies – in an immutable file that can run in any environment with the same kernel. Each container is built from a container image, an executable software package containing everything required to run as a standalone application. The ability to build, maintain, and debug applications that are packed as containers has dramatically improved the development pipeline. The utility offered by this approach is key to successful cloud approaches, but monitoring, deploying, and maintaining containers is a complicated task. Doing so properly requires using a container orchestration (CO) system. These tools automate the deployment and management of containerized applications. Kubernetes is far and away the most popular such tool.

A CO is used to supervise the entire lifecycle of a containerized application, from deployment to deletion. The CO is the control center of a containerized system, governing the allocation of resources and health of the applications. If an application needs to be restarted, the CO is there to take care of it. If a container needs some love in terms of resources, the CO will find them. It takes all the manual work out of creating and maintaining a container.

Any good CO system adds value to a containerized approach in many ways. They add versatility and flexibility by managing container resources, allowing you to do more with less. Security is much more streamlined, as the CO governs several containers, insulating them from other networks and applying consistent policies across them all. The CO also assists with things such as scalability, as the CO can provision new containers automatically.

However, as the size and complexity of cloud systems grow and grow, proper operation of a CO becomes increasingly difficult. A sprawling containerized system can easily overwhelm traditional orchestration processes and lead to errors and wasted resources. Often, a CO Management System is needed to ensure consistent container architecture. However, there's no question that it's worth the extra work.

Why You Need Kubernetes



A 2021 report from the Cloud Native Computing Foundation found that over half of all companies operating in the cloud use Kubernetes, and that number is steadily rising. There's a reason for this, and it's that Kubernetes works. It helps development in several different ways, including the following.

Ease of Development

Packaging entire applications in a container has many implications for the application development lifecycle. Since everything needed to run the application has been encapsulated into a container image, the difference between development, production, and testing environments is a matter of configuration at runtime. This dramatically aids in building out continuous integration and continuous deployment (CI/CD) setups and provides confidence that what you're testing is identical to what you'll be running in production.

Cloud deployment, monitoring, and maintenance is a technically challenging assignment, but Kubernetes makes it much easier by taking a ton of the grunt work off your plate. By tackling the deployment and management of these applications, developers are free to focus on actual development instead of the technical hurdles behind safe cloud deployment.

Security

Historically, developers would place untrusted code in a virtual machine (VM), and the hypervisor would provide a stronger security boundary that containers can't achieve. This is because VMs emulate their hardware, so they are fully standalone environments.

However, while containers provide some level of resource isolation, they weren't designed with security in mind. Running third-party applications in containers can still affect other processes through resources that the host operating system kernel doesn't manage, such as L3 processor cache and memory bandwidth. Because underlying hardware is shared between containers, an exploit in one container can impact other containers on the same host if that attack extends to this shared hardware.

In a containerized environment, a container orchestration system such as Kubernetes can provide additional layers of security. For starters, the nature of a Kubernetes deployment means that application containers are not updated directly. Instead, the image file is replaced with a completely new version, which avoids security issues related to updating. Beyond that, Kubernetes offers the ability to apply security policies across containers and manage privileges in one central location.

By allowing security to be handled by a single entity, there is far less chance of security violations happening on individual container applications. And while Kubernetes offers some inherent security advantages over manual container deployment, vulnerabilities still arise. This is why an additional layer of Kubernetes management can help tighten things even further.

Versatility

That said, container orchestration is about more than security. Kubernetes provides many features that have been popularized by Platform-as-a-Service (PaaS) offerings, such as deployment rollout patterns, scaling, and load balancing. The defaults are optional and pluggable. Kubernetes understands that every organization has a different set of requirements and offers integration points for logging, monitoring, and alerting solutions and mechanisms to collect and export metrics – but it doesn't force you to use any particular solution.

Kubernetes provides decoupled building blocks that scale from testing locally to running a global enterprise without expanding your operations team. Kubernetes, and to an even greater degree, the package ecosystem, are imbued with years of container management wisdom. For everything else, Kubernetes provides fine-grained configuration and integration points that enable extensibility at all layers of the environment for unparalleled flexibility.

The cumulative experience of the engineers behind Kubernetes and the greater community has helped lay the foundation for standardizing across a set of interfaces upon which to build. As more organizations move toward leveraging Kubernetes for their digital transformation needs, they benefit from the reduced time and cost of implementing and maintaining complex container management solutions.

Because of its popularity, knowledge of and experience with Kubernetes are highly marketable, incentivizing engineers to learn and master the system. Organizations of any size also benefit because new hires can make an impact on their first day.

If you're not already using Kubernetes, you will likely be in the future.

The Double-Edged Sword of Open-Source

The open and extensible nature of Kubernetes allows anyone to build solutions to the container management problems they face. This is part of what makes it a versatile tool, but it also leads to confusion and complexity. While Kubernetes is designed to be user-friendly, and it might be straightforward to deploy a small application using it, managing an enterprise-level Kubernetes system is challenging. As your system grows, adding more pods, nodes – and eventually, potentially clusters – maintaining and updating the system becomes exponentially more complicated. This complexity is necessary to ensure your system's security, accessibility, and reliability.

Security Challenges

Security is best approached in layers. The cluster, nodes, and pods within Kubernetes provide additional layers of security beyond the container itself. However, Kubernetes is still a very complex system. You must be prepared to overcome the challenges of securing your clusters across the many moving parts that make up a Kubernetes cluster. Your choice of cloud provider forms the outer layer of securing a cluster. A provider with strong security is particularly important to handle the network communication between your Kubernetes control plane – the layer that manages the nodes, pods, and container lifecycle – and the rest of your cluster. This is difficult to get right. That's why using a Kubernetes management system tailored to your cloud provider is recommended to deal with the intricacies of creating and protecting the resources it administers.

Networking Challenges

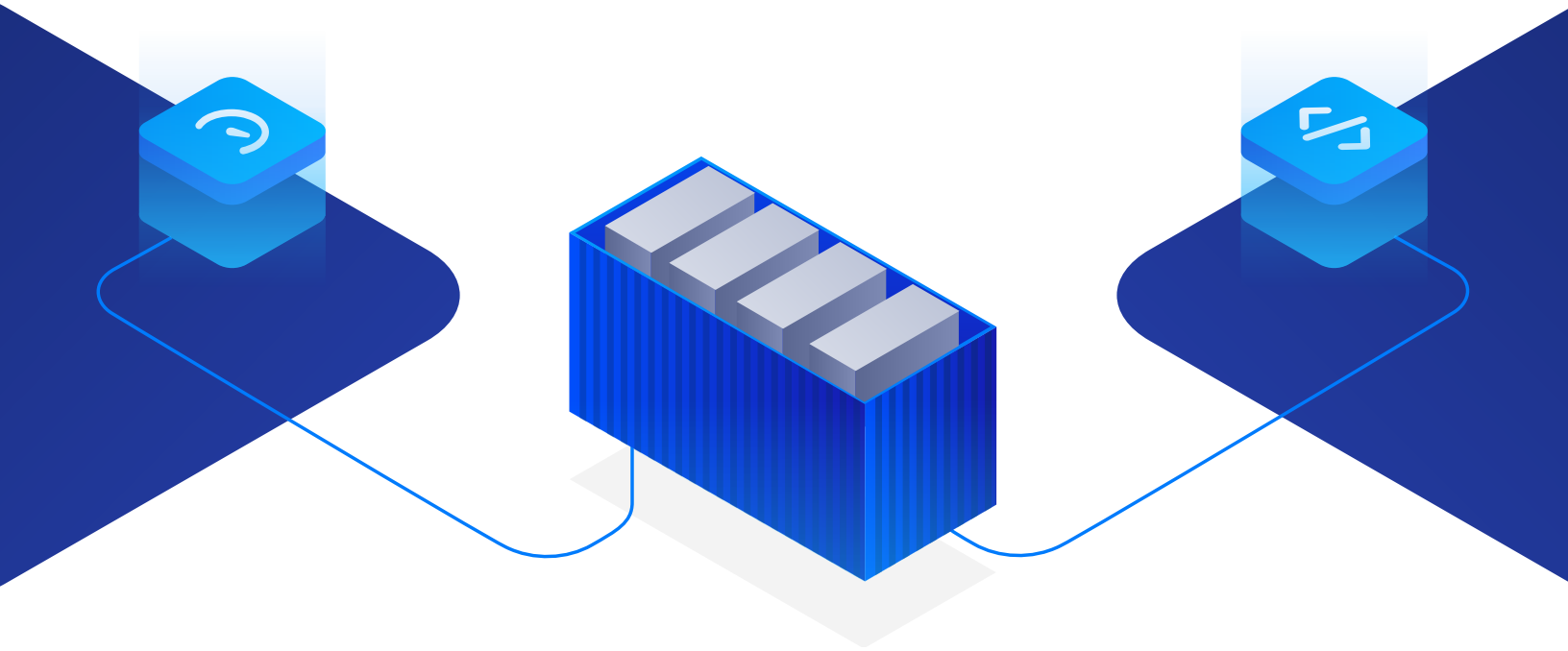
Networking presents its own set of challenges when coming from a more traditional approach to cloud environments. Kubernetes requires a bit of a shift in thinking. Every pod in a cluster is assigned a network namespace and an IP address from the range assigned to a node.

All pods within a cluster can communicate with each other, even across nodes, without network address translation (NAT). Containers within a pod can communicate with each other using localhost or inter-process communication but need to coordinate their shared use of the pod's IP address and port space when communicating with entities outside the pod. Pods can be viewed almost like VMs or physical servers regarding standard networking tasks such as port allocation, load balancing, or application configuration.

Scaling Challenges

Scaling within Kubernetes isn't magic. Kubernetes will run almost anything as long as you can containerize it, but if your applications aren't built to be scaled, you may be limited to vertical pod scaling. This means provisioning more resources such as CPU, memory, and storage for each pod. Your application should make effective use of Kubernetes horizontal pod autoscaling – adding more pods that run the same application – and be built with capability in mind. The best first step toward moving to Kubernetes is to start containerizing your applications.

Finally, updates cause constant changes in a cluster, which makes running a Kubernetes cluster a challenge. You must continuously monitor the cluster configurations, components' health, and resource usage to avoid security lapses, downtime, and unexpected cost increases.



Vultr Kubernetes Orchestration

Kubernetes is undoubtedly a powerful technology, but using it properly can be challenging. On top of the complexities described above, achieving a consistent and secure Kubernetes approach across a large and dynamic structure is time-consuming and fraught with issues. As such, it's often recommended that a Kubernetes Management System is employed. However, finding a suitable Kubernetes management system is difficult. The explosion in its popularity has several providers competing for your attention, each with a somewhat unique value proposition, differentiators, support, and pricing structures. Most managed Kubernetes providers offer easy initial setup. Still, it's important to pay attention to the native plugin support for managing provider-specific cloud resources, such as node instances, load balancers, and block storage.

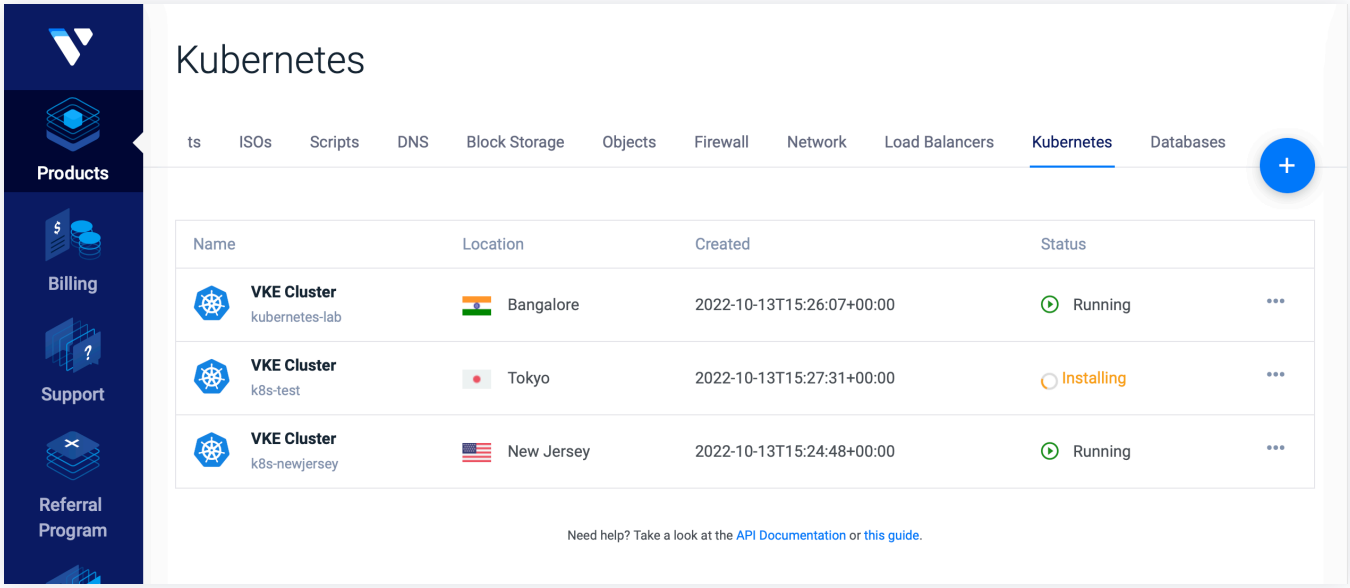
Vultr Kubernetes Engine (VKE) is a fully-managed Kubernetes service that removes a lot of the complexities of bootstrapping and maintaining a custom cluster. Creating a Kubernetes cluster from Vultr's control panel is as easy as selecting a few details, such as a cluster name, region, and initial nodes. Vultr manages the entire Kubernetes control plane and removes the complexities of cluster management, enabling you to focus on your specific application needs.

Integrations

As Kubernetes allows for deep integration between cloud providers, VKE includes the Vultr Cloud Controller Manager (CCM) when creating a new cluster. The Vultr CCM is responsible for maintaining secure communication between the Vultr API and your Kubernetes cluster, enabling deep integration between Kubernetes and Vultr features. The etcd cluster, used to store secrets and control plane configuration data, is encrypted and regularly backed up on a VKE cluster.

The Vultr CCM node controller comes preinstalled on VKE clusters. It updates Kubernetes node objects with information about the Vultr hosts as instances are created and deleted within the cluster. Furthermore, the Vultr CCM ensures that Kubernetes nodes are aware of Vultr-specific information, such as the instance hostnames, region, service plan identifier, and IP addresses. If nodes become unresponsive, the node controller communicates with Vultr to determine if the

server has been shut down or deleted. If so, the node object is removed from the cluster, and pods are rescheduled. Essentially, the Vultr CCM keeps your Vultr cloud resources in sync with your Kubernetes cluster.



Deploying a Kubernetes service configured as a load balancer will provision and configure a Vultr-managed load balancer instance. This results in fully automated, durable horizontal infrastructure scaling. VKE includes a Kubernetes autoscaler implementation that will dynamically add or remove nodes to optimize resource usage so that you're ready for bursts of traffic and save money on resources when traffic is low.

Storage

The Vultr CSI driver also comes preinstalled on VKE clusters. This driver integrates Kubernetes with the Vultr Block Storage managed service for both traditional rotating HDD and high-perfomance NVMe drives. Moreover, it handles the dynamic provisioning and mounting of persistent block storage devices within a Kubernetes cluster.

Use VKE Cert Manager to simplify the process of obtaining, renewing, and using TLS/SSL certificates issued from a variety of sources, including Let's Encrypt, a free certificate authority. Additionally, VKE ExternalDNS will dynamically maintain DNS records of publicly accessible Kubernetes resources with Vultr DNS and automatically configure your Vultr managed load balancer.

Certification

Moreover, VKE is certified by the Cloud Native Computing Foundation (CNCF) Certified Kubernetes Conformance Program. Products that have been certified through CNCF have been tested and verified to conform to superior standards of interoperability and consistency among Kubernetes installations. Software vendors that wish to remain certified by the CNCF must provide regular updates.

With this multitude of offerings, Vultr's VKE and growing integrations cover every Kubernetes situation you'll encounter.

Cost-Efficient Solutions

Vultr does not charge management fees for managed Kubernetes offerings. You're only billed for the resources that are provisioned – not the control plane itself. If you have a three-node cluster with a managed load balancer and block storage, then you only pay for the infrastructure and not the management that Vultr provides.

In comparison, other well-known managed Kubernetes offerings typically charge at least $70 per month for the control plane alone – before any resources are provisioned. This isn't always so clear because the pricing is often buried in confusing pricing calculators.

While not novel, VKE also includes a configured Kubernetes autoscaler implementation designed to ensure you get the most out of your system. The node-pooling approach ensures that when the cluster has determined that the current resource use doesn't require the application-specified resources from additional nodes, some instances can be deprovisioned to save on costs. Then, when resource use increases to the point that resources from an additional instance are needed, VKE will automatically provision them from the available pool.

Vultr's simplified and transparent pricing model is attractive for individuals and enterprises looking for a more predictable monthly bill and is already familiar to existing customers. If you encounter any issues with the managed services that Vultr provides, you can open a support ticket from the control panel, and a Vultr support representative will reach out to you promptly.

Conclusion

Kubernetes and the rise of containerization have changed the landscape of building and managing distributed systems of all sizes and levels of complexity. Proper separation of concerns allows organizations to ensure that their application developers can focus on building applications and that infrastructure teams can manage the infrastructure.

However, containers alone aren't enough. Rather, they serve as the base for application-oriented infrastructure. Organizations require a container management system to deal with

the intricacies of deploying, scaling, and error handling within complex distributed systems. Proper container management includes service discovery, load balancing, storage orchestration, automated rollouts and rollbacks, automatic resource bin packing, self-healing, secrets, and configuration management. This is no small undertaking.

While Kubernetes is great at handling low-level container orchestration, maintaining a sophisticated container architecture requires an added layer. A solid Kubernetes Management System is required to ensure a consistent and secure environment that lets you get the most out of Kubernetes. Manually dealing with an enterprise-level Kubernetes architecture is simply asking for trouble.

Organizations that choose to build their own Kubernetes Management System spend considerable time and money to build and maintain systems for automating the deployment, scaling, and management of containerized applications. Moreover, the patterns and problems addressed are those for which industry experts have already built proven solutions for Kubernetes and the extended ecosystem.

Kubernetes provides all these features, but bootstrapping and maintaining a cluster yourself is cumbersome. Leveraging a fully managed Kubernetes service like Vultr Kubernetes Engine can drastically reduce the effort of deploying and maintaining a Kubernetes cluster, lowering the barrier to benefitting from all the features Kubernetes provides. Additionally, using a fully managed Kubernetes service removes an entire layer of security responsibility.

Creating and upgrading clusters with VKE is straightforward and automated, requiring minimal user input. A few minutes later, the Kubernetes cluster and associated resources will be ready for deploying resilient applications.

Moreover, you don't need to pay anything to have Vultr manage your cluster, which is a rarity among its competitors. You only need to pay for the resources your cluster uses – you get the management for free. Overall, this makes Vultr one of the most affordable options for deploying and maintaining a Kubernetes cluster.

When you're ready to get the most out of your container management system, contact Vultr for more information on Vultr Kubernetes Engine.