Enterprise Kubernetes Made Simple

Version 1.2 – June 2024

Abstract

Application container technologies are becoming the de facto leading standard for packaging, deploying, and managing applications with increased levels of agility and efficiency. Kubernetes is a widely used tool for the orchestration of containers on clusters. OpenNebula supports deploying Kubernetes clusters through its OneKE virtual appliance. OneKE (OpenNebula Kubernetes Engine) is available in the OpenNebula Public Marketplace. It is an enterprise-grade, CNCF-certified Kubernetes distribution that simplifies the provisioning, operations, and lifecycle management of Kubernetes. OneKE makes it possible to run any type of containerized application on an OpenNebula cloud using a single toolset, whether on premises, on a public cloud, or at the edge.

Contents

Glossary

BGP	Border Gateway Protocol
CMP	Cloud Management Platform
CNCF	Cloud Native Computing Foundation
CNI	Container Network Interface
CRD	Custom Resource Definition
HA	High Availability
K8s	Kubernetes
OS	Operating System
VIP	Virtual IP
VM	Virtual Machine
VNF	Virtual Network Function

1. An Overview of Containers

The Container Revolution

The past few years have seen a rapid evolution of information and communications technologies, which have dramatically changed the way information systems and applications are built. Software development has brought a slew of changes and revolutions, thanks to which organizations can now focus mainly on business applications. The key drivers of this shift have been (1) the ability to **package and run applications anywhere** regardless of the underlying computing architecture, and (2) the means of **keeping applications isolated from each other** in order to avoid security risks and interference during operations or maintenance processes.

Keeping applications isolated on the same host or cluster can be difficult, owing to the packages, libraries, and other software components that are normally required to run them. Hardware virtualization came as a solution to this problem, since it made it possible to keep applications isolated from each other on the same hardware, by using Virtual Machines. Packaging an application within a VM also allowed it to run on any infrastructure that supported virtualization, which provided a lot of flexibility to the whole concept. However, Virtual Machines come with some serious limitations: moving them around is not that easy since they are typically quite heavy and there are always difficulties associated with maintaining and upgrading applications running within a VM.

In recent years, we have all witnessed how container technologies have revolutionized the way enterprise and distributed applications are being developed and deployed. Containers clearly offer a more portable and flexible way of packaging, maintaining, and running applications. They allow admins to deploy, move, and replicate workloads more quickly and easily than using Virtual Machines. While containers as a concept have been around for a while, **Docker** was the technology that introduced several crucial changes to the existing container technology, making containers more portable and flexible to use. This resulted in a turning point towards the adoption of containerization and microservices in software development (e.g. cloud-native development).

Docker gave us an easy way to create container-based applications and to package them in portable images containing the specifications for the software components the container would run. Docker's technology brought cloud-like flexibility to any infrastructure capable of running containers: its container image tools allowed developers to build libraries of images, compose applications from multiple images and launch those containers and applications on local and remote infrastructures alike.

Orchestrating Containers

Nowadays, many companies have embraced a **cloud-native paradigm** in developing applications, and have shifted from a "monolithic" approach to a microservice approach. While deploying a single container can be an easy task, things get a bit more complicated when deploying multi-container applications on distributed hosts, since in these cases a Docker Engine alone is not enough. This is where container orchestrators (like **Kubernetes** or **Docker Swarm**) play an important role in scheduling containers to run on different servers, moving containers to a new host when the host becomes unhealthy, restarting containers when they fail, managing overlay networks to allow containers on different hosts to communicate, orchestrating storage to provide persistent volumes to stateful applications, and so on.

However, container technologies (e.g. Docker, Kubernetes) also come with some serious limitations, such as **security** (application containers share the kernel OS) and **multi-tenant environments**. In order to provide a multi-tenant and secure environment to deploy containerized applications, one has to provision different "virtual environments" to each user or group of users, typically by deploying several isolated Kubernetes clusters on top of a Cloud Management Platform. The CMP is then responsible for managing and orchestrating the underlying virtual resources (i.e. Virtual Machines, virtual networks and storage) that are used by the different Kubernetes deployments in charge of scheduling application containers within those isolated environments.

Why Kubernetes on OpenNebula?

OpenNebula incorporates **OneKE**, OpenNebula's Kubernetes management solution. In less than 5 minutes, you can configure and deploy a High Availability Kubernetes cluster, integrated with persistent storage solutions for stateful applications. OneKE simplifies Kubernetes management across the entire lifecycle, provides a consistent and secure experience from the datacenter to the edge, and fast-tracks your way to production-ready Kubernetes with push-button simplicity while preserving a native user experience.

Automated Multicluster Operations

Simplify operations and automate lifecycle management of large-scale, multicluster K8s environments, and keep your workloads properly isolated

Centralized Management for All Workloads

Encompass K8s clusters within other virtualized workloads using a single control layer to reduce complexity, consumption and operating costs

Kubernetes as a Service

Build a multi-tenant self-service environment for the execution of K8s clusters on a shared physical infrastructure

Enhanced Security

Enhance security thanks to the additional layer provided by hardware virtualization to isolate resources pools on the same host

Fast Deployment on Any Infrastructure

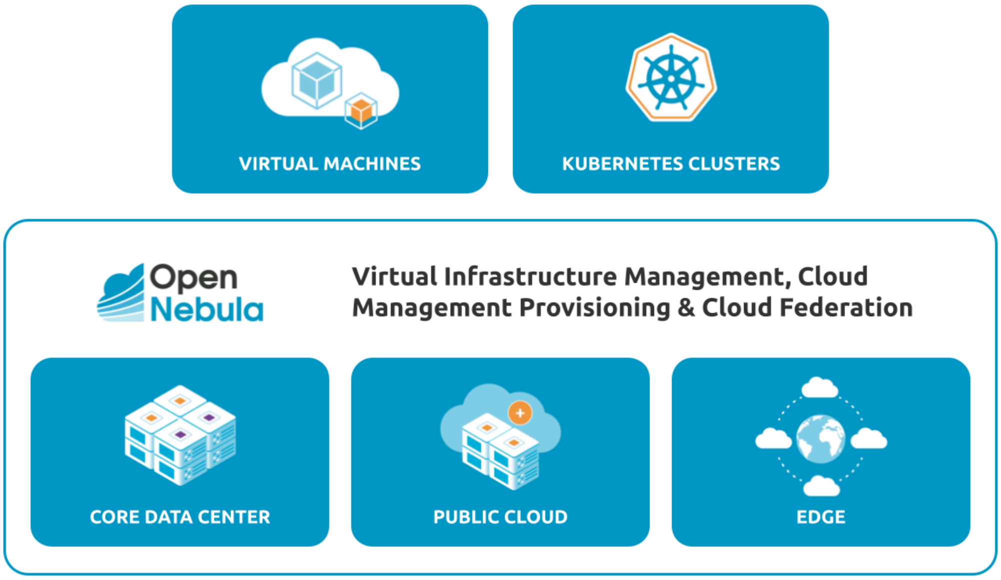
Automatically deploy in minutes and manage multiple K8s clusters across on-premises, edge, and cloud locations to enable large-scale container orchestration

No Provider Lock-in

Deliver a native K8s user experience with open APIs anywhere, with the configuration that you want and following the same process

2. What is OpenNebula?

OpenNebula[1] is a **simple, but powerful, open source solution to build and manage Enterprise Clouds and Edge environments**. It combines virtualization and container technologies with multi-tenancy, automatic provision, and elasticity to offer on-demand applications and services. It provides a single, feature-rich and flexible platform with **unified management of IT infrastructure and applications** that **avoids vendor lock-in** and **reduces complexity, resource consumption, and operational costs**.



OpenNebula manages:

- **Any Application**: Combine containerized applications from Kubernetes with Virtual Machine workloads in a common shared environment to offer the best of both worlds: mature virtualization technology and orchestration of application containers.
- **Any Infrastructure**: Open cloud architecture to orchestrate compute, storage, and networking driven by software.
- **Any Cloud**: Unlock the power of a true hybrid, edge and multi-cloud platform by combining your private cloud with infrastructure resources from third-party virtual and bare-metal cloud providers such as AWS and Equinix Metal, and manage all cloud operations under a single control panel and interoperable layer.
- **Any Time**: Add and remove new clusters automatically in order to meet peaks in demand, or to implement fault tolerance strategies or latency requirements.

OpenNebula provides the necessary tools for running containerized applications from Kubernetes while ensuring enterprise requirements for your DevOps practices. It helps organizations to easily embrace Hybrid and Edge Computing, allowing them to grow their Enterprise Cloud on demand with infrastructure resources from third-party Public Cloud and bare-metal providers such as AWS and Equinix Metal.
This white paper describes how OpenNebula integrates with Kubernetes. If you are interested in an OpenNebula cloud fully based on open source platforms and technologies, please refer to our Open Cloud Reference Architecture.

OpenNebula brings the provisioning tools and methods needed to dynamically grow a private cloud infrastructure that includes resources running on remote cloud and edge providers, enabling powerful, true hybrid and multi-cloud computing with support for all major clouds. This disaggregated cloud approach

[1] https://support.opennebula.pro/hc/en-us/articles/360036935791-OpenNebula-Overview-Datasheet

allows for a seamless transition from centralized private clouds to distributed edge-like cloud environments. Companies can grow their private cloud with resources at cloud and edge datacenter locations, to meet peaks in demand or the latency and bandwidth needs of their workload. This approach involves a single management layer where organizations can continue using existing OpenNebula images and templates, keep complete control over their infrastructure, and avoid vendor lock-in.

OpenNebula allows you to deploy a fully operational **Edge Cluster**[2] in a remote provider, and to manage its full life cycle from provisioning and maintenance to unprovisioning. Each cloud or edge location (the "**provision**") is defined as a group of physical hosts allocated from the remote bare-metal or virtual provider. They are fully configured with the user-selected hypervisor and enabled in the cloud stack, available for end-users.

Recommended Configurations for Kubernetes Clusters:

Use Case	Hypervisor	Edge Cluster
Execute our **CNCF-certified OneKE** virtual appliance on **VMs** within cloud **bare-metal servers.**	KVM	Bare Metal

3. The OpenNebula Kubernetes Engine (OneKE)

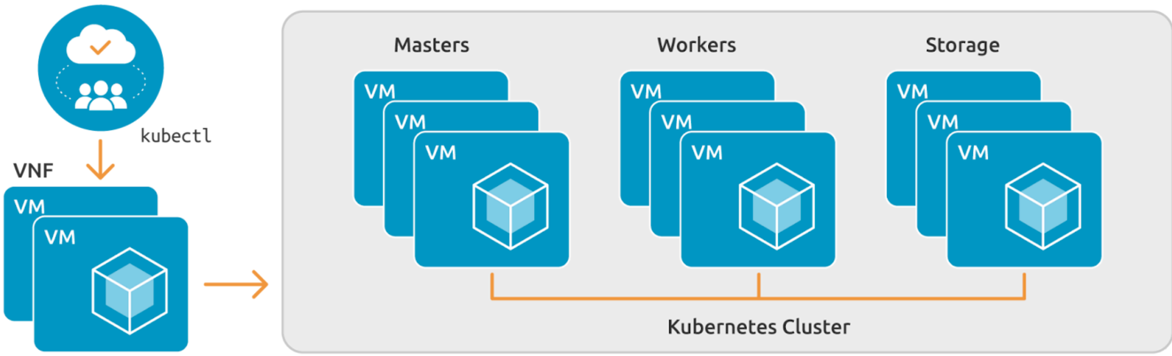
OpenNebula supports deploying, managing and scaling Kubernetes clusters through its CNCF-certified OneKE virtual appliance, available for download from our OpenNebula Public Marketplace. OneKE is based on the Linux distribution **Ubuntu** and on SUSE Rancher's Kubernetes distribution **RKE2**. It allows you to build a multi-master Kubernetes cluster ready for production environments, and may be integrated with the Canal, Cilium, Calico or Multus Container Network Interface (CNI) plugins. It supports persistent volumes through Longhorn distributed storage and is integrated with the MetalLB load balancer for exposing Kubernetes services deployed in on-prem clusters. Additionally, it features an integrated HAProxy/Traefik solution to export HTTP/HTTPS apps via IngressRoute resources.

★ Based on open source components
★ Multi-master ready
★ Canal, Cilium, Calico and Multus CNI networking
★ Longhorn distributed storage
★ MetalLB or Cilium load balancers
★ Traefik Ingress Controller

OneKE is implemented as a **OneFlow Service.** OneFlow allows the definition, execution and management of multi-tiered applications—so-called Services—composed of interconnected Virtual Machines with deployment dependencies between them. Each group of Virtual Machines is deployed and managed as a single entity (known as a "role"). OneKE has four different roles:

- **VNF**: Load balancer for Control-Plane and Ingress traffic
- **Master**: Control-Plane nodes (managing the `etcd` database, API server, controller manager and scheduler, along with the worker nodes)
- **Worker**: Nodes to run application workloads
- **Storage**: Dedicated storage nodes for persistent volume replicas

[2] https://support.opennebula.pro/hc/en-us/articles/360050302811-Edge-Cloud-Architecture-White-Paper

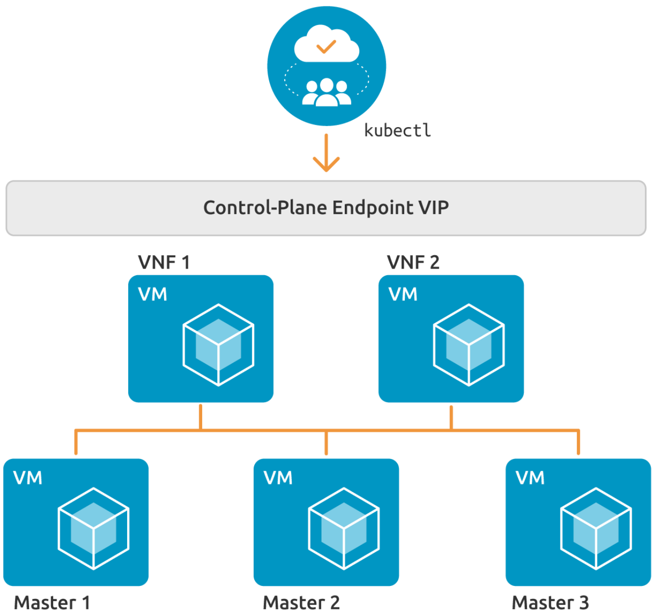


By default, each OneKE role is preconfigured to deploy a single server to work as a non-Highly-Available (non-HA) cluster. However, enabling Virtual IP addresses (VIPs) during initial setup allows you to deploy with a configuration enabled for High-Availability (HA). Since OneFlow is able to implement elasticity policies, service can scale up or down depending on needs, in order to add or remove Virtual Machines (nodes). Each role can be scaled up to achieve a High Availability setup.

3.1. High Availability

A High Availability (HA) configuration aims to avoid a single point of failure for the Kubernetes cluster. Non-HA Kubernetes clusters are single-master clusters, in which a single master node controls all worker nodes and all of the essential components (the `etcd` database, API server, controller manager, scheduler, etc.). In this configuration, failure of the master node means the failure of the whole system: users lose the ability to create services and pods, all worker nodes fail and the whole cluster could be lost.

An HA Kubernetes cluster is a multi-master cluster, which as its name implies uses multiple master nodes (usually three). Essential Kubernetes components are replicated across the multiple masters; if a master fails, the other masters keep the cluster up and running. Each master node runs its own copy of the API server, which can be used for load balancing among the master nodes. Each master also runs its own copy of the `etcd` database (which stores all data for the cluster), as well as its own controller manager (which handles replication), and its own scheduler (which schedules pods to nodes). Thus a multi-master configuration protects against an ample variety of failure scenarios, from the loss of a single worker node to the failure of essential components on a master node.

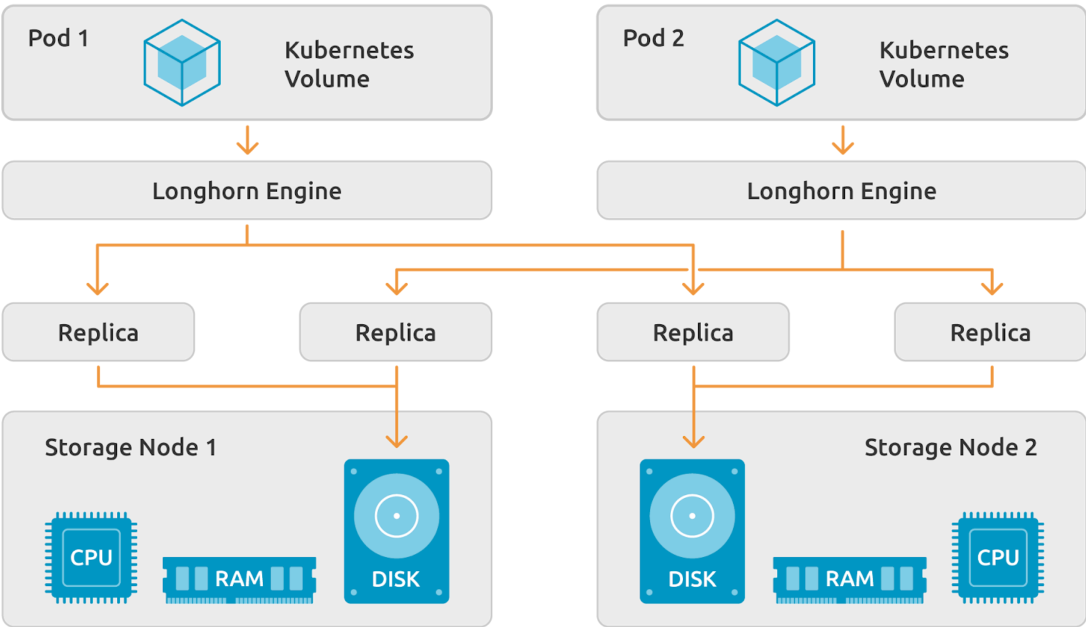


To achieve high availability, OneKE uses the VNF appliance (Linux Virtual Server) as a Load Balancer for the HA Multi-Master Control Plane. VNF is based on Keepalived and can be scaled up to run on multiple VMs, in order to provide HA for the VNF itself. Users must provide the requisite VIP for the Control-Plane Endpoint when instantiating the OneKE appliance, by using the GUI or by editing the ONEAPP_VNF_LB0_IP context parameter.

3.2. Storage

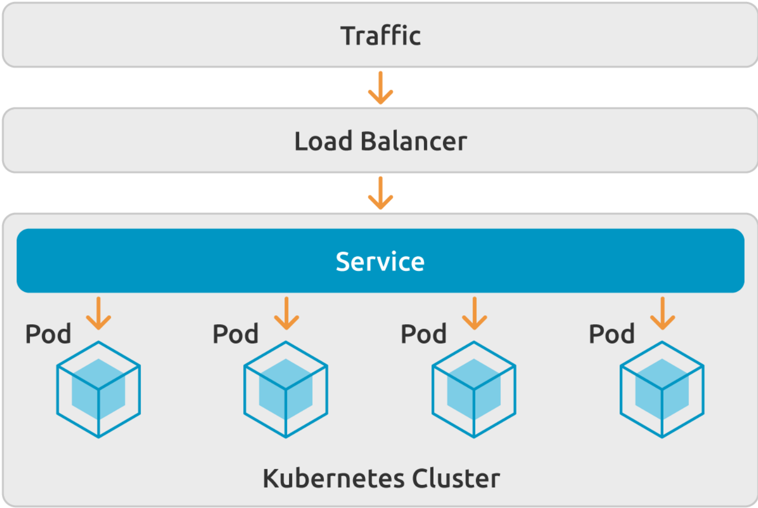
OneKE simplifies deployment of highly-available persistent block storage in your Kubernetes environment—and ensures that it is fast and reliable—by integrating Longhorn. Longhorn is a lightweight, reliable, and powerful distributed block storage system for Kubernetes. It implements distributed block storage using containers and microservices, creates a dedicated storage controller for each block device volume, and synchronously replicates the volume across multiple replicas stored on multiple nodes. The storage controller and replicas are themselves orchestrated using Kubernetes.

The OneKE storage node role can be scaled up to provide highly-available persistent block storage for Kubernetes pods.



3.3. Load Balancer Service

Pods or deployments can be exposed as a service of the type LoadBalancer, a more flexible alternative to the simple type NodePort. Our OneKE appliance offers the option of integrating the bare-metal load balancer MetalLB (or the BGP load balancers from Cilium, if Cilium is used as a CNI). By default, MetalLB is configured as an ARP/Layer 2 LoadBalancer, which means that the exposed LoadBalancer IP must be routed to one of the Kubernetes cluster nodes—which is not within the scope of the appliance itself, and so must be achieved by other means. MetalLB also supports BGP/Layer3 load balancing. After setting up the network for this dynamic routing protocol, the user can provide the appliance with the proper configuration by using contextualization parameters.



3.4. Ingress Controller Service

To expose HTTP and HTTPS routes from outside the Kubernetes cluster to services deployed within the cluster, OneKE provides an Ingress Controller based on Traefik. Traffic routing is controlled by rules defined on the Ingress resource. Traefik is exposed on a Kubernetes service of type NodePort. By default, the HAProxy instance (running on the leader VNF node) connects all worker nodes to ports 32080 and 32443, then forwards all traffic coming to HAProxy to ports 80 and 443 of the Traefik instance (running inside Kubernetes). An anti-affinity rule is applied to Traefik pods to minimize potential downtime during failures and upgrades. An Ingress Controller does not expose arbitrary ports or protocols; for exposing services other than HTTP or HTTPS to the internet, a LoadBalancer service is typically used instead..

3.5. Upgrades

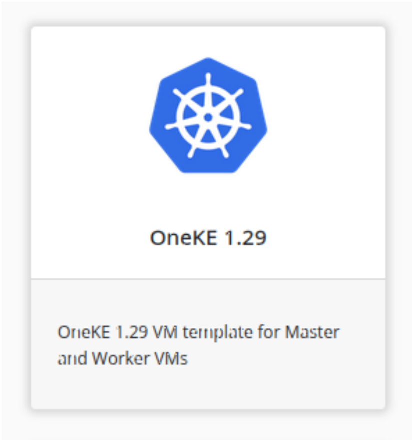
To manage OneKE Kubernetes cluster upgrades, you can use Rancher's RKE2 `system-upgrade-controller`, a Kubernetes-native, general-purpose upgrade controller for nodes. It uses a Custom Resource Definition (CRD) known as the Plan to define upgrade policies and requirements. Upgrades are scheduled by a controller based on user-defined plans.

On initial deployment, OneKE offers the option of enabling Longhorn, Traefik and MetalLB during cluster bootstrap.. These apps are deployed as add-ons using RKE2's Helm Integration and official Helm charts. Add-ons can be easily upgraded using the Helm chart CRD that allows the user to override and patch helm charts.

4. Using OneKE with OpenNebula

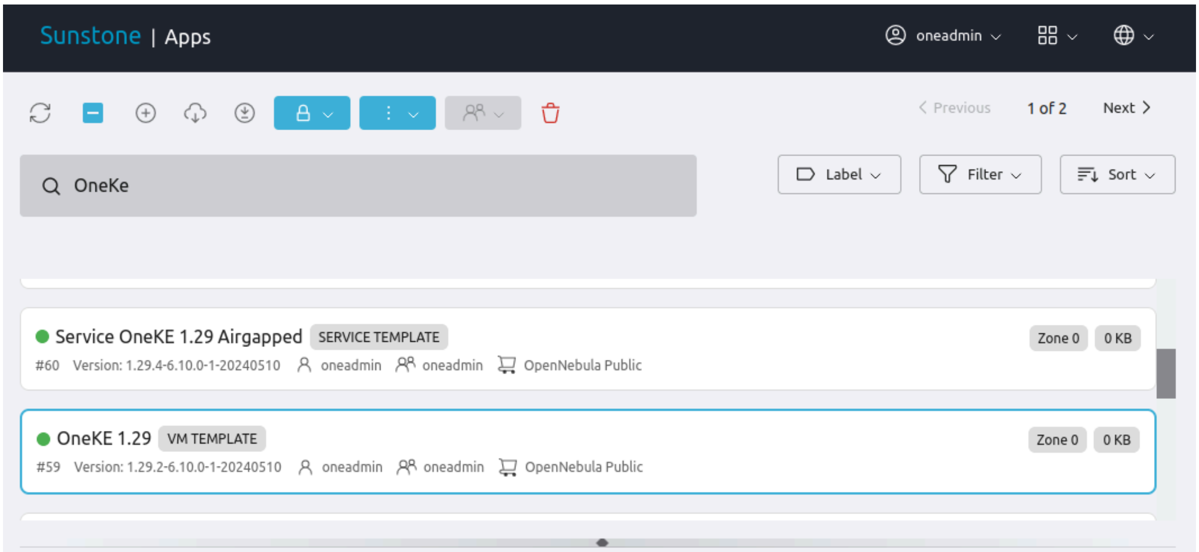
Running Your Kubernetes Clusters on OpenNebula

When companies need complete container orchestration services based on Kubernetes for the deployment and management of containerized workflows, OpenNebula provides them with a simple "press-of-a-button" option to create and deploy a fully functional Kubernetes cluster, thanks to the **OneKE virtual appliance** available from the OpenNebula Public Marketplace.



Our Virtual Appliance allows you to build a multi-master Kubernetes cluster ready for production environments, with the option of integrating with the Canal, Cilium, Calico or Multus CNI plugins. It supports persistent volumes through the CNCF Longhorn distributed storage, and offers the option of integrating with the MetalLB load balancer for exposing services deployed in on-prem clusters, or with Cilium BGP load balancers if Cilium is used as CNI. Lastly, it features an integrated HAProxy/Traefik solution to export HTTP/HTTPS apps via IngressRoute resources. It supports multiple contextualization parameters to adapt to your needs and required configuration. Using the OneFlow service enhances simplicity and versatility, enabling the appliance to function as an automatically managed multi-node cluster.

This virtual appliance provides you with a Kubernetes cluster in which every node is managed by OpenNebula as a regular VM (and you can always add more nodes to the cluster at any time using OneFlow elasticity features), but OpenNebula does not manage containers or pods inside the Kubernetes cluster. The Kubernetes cluster exposes the **Kubernetes API** so that you can then access it via `kubectl` or Kubernetes's UI dashboard to create pods, deployments, services, etc. Before deploying a Kubernetes Service, we recommend you to check out the associated documentation.



The OneKE virtual appliance available from our Marketplace:

- Enables the provisioning of managed K8s clusters on demand with just one click. Easily deploys on different architectures (on-prem, edge, and cloud) for different users and applications
- Provides an auto-configured cluster with information exchanged over additional OpenNebula services (OneGate), managed as one entity (OneFlow)
- Provides elasticity features to scale up/down K8s cluster nodes for High Availability and/or workload needs
- Runs anywhere, with a built-in configuration of components (e.g. networking, storage) selected to handle restrictions on the end-user side

Interested in deploying Kubernetes? 🚀

6. Ready for a Test Drive?

You can evaluate OpenNebula and build a cloud in just a few minutes by using miniONE, our deployment tool for quickly installing an OpenNebula Front-end inside a Virtual Machine or physical host, which you can then use to easily add remote resources.

7. Conclusions

OpenNebula introduces support for deploying, managing and scaling Kubernetes clusters through its CNCF-certified OneKE virtual appliance available from the OpenNebula Public Marketplace. If you require any assistance in adapting these technologies to your specific **DevOps** requirements, or in integrating any other Kubernetes distributions or container orchestration solutions in your organization, don't hesitate to contact us—we look forward to helping you at any stage of your cloud computing journey.

LET US HELP YOU DESIGN, BUILD, AND OPERATE YOUR CLOUD

CONSULTING & ENGINEERING

Our experts will help you design, integrate, build, and operate an OpenNebula cloud infrastructure

OPENNEBULA SUBSCRIPTION

Get access to our Enterprise Edition and to our support and exclusive services for Corporate Users

CLOUD DEPLOYMENT

Focus on your business and let us take care of setting up your OpenNebula cloud infrastructure.

Sign up for updates at OpenNebula.io/getupdated

Rev1.2_20240601