

.

ThuyVT2

VKS

•

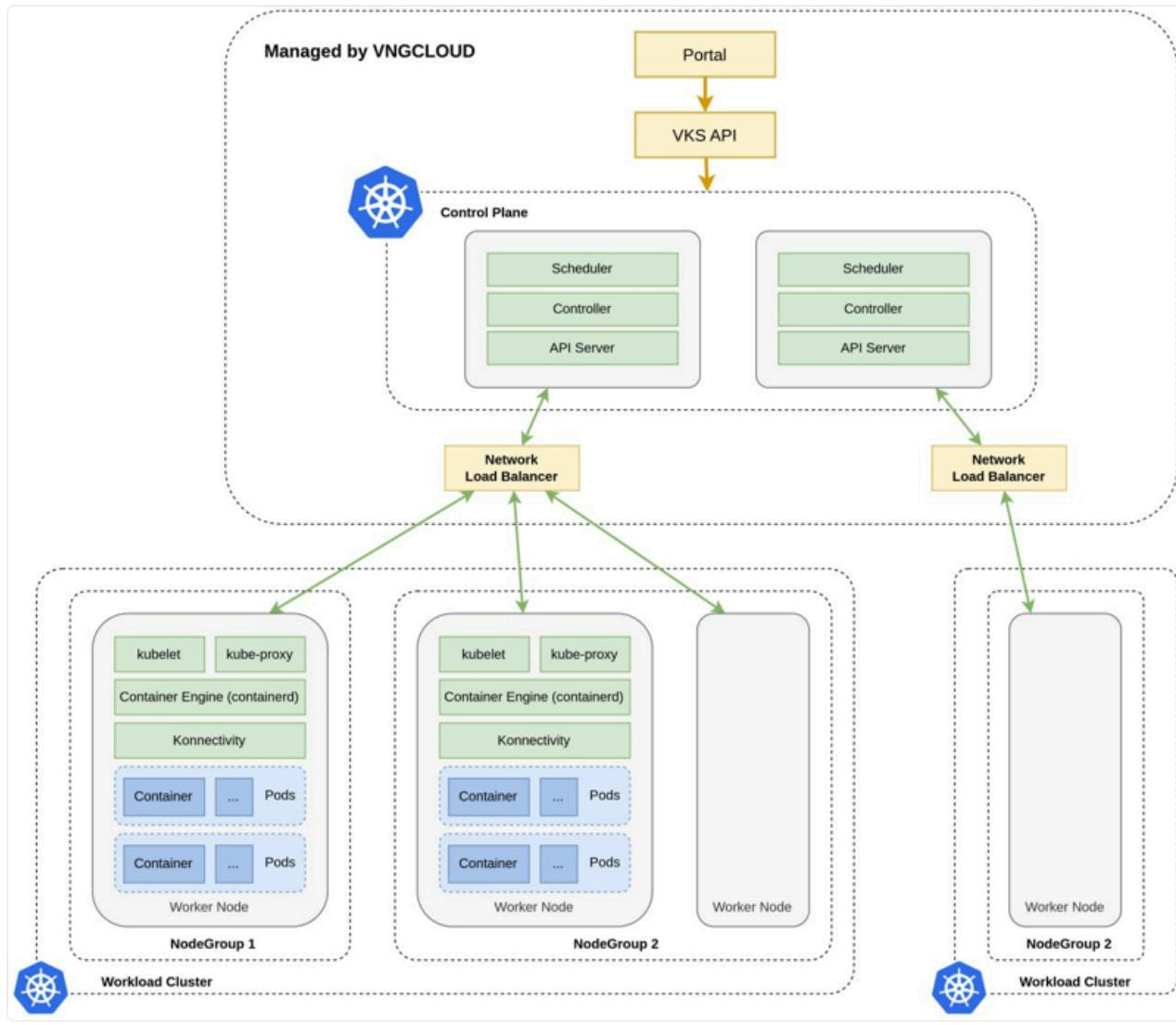
VKS (VNGCloud Kubernetes Service) is a managed service on VNGCloud that helps you simplify the deployment and management of container-based applications. Kubernetes is an open-source platform developed by Google, widely used for managing and deploying containerized applications in distributed environments.

VKS Demo Video - Giải Pháp Quản Lý Kubernetes Toàn Diện Của VN...



What is VKS?

VKS (VNGCloud Kubernetes Service) is a managed service on VNGCloud that simplifies the deployment and management of container-based applications. Kubernetes, an open-source platform developed by Google, is widely used to manage and deploy containerized applications in distributed environments.



Highlights of VKS

- **Fully Managed control plane:** VKS will free you from the burden of managing the Kubernetes Control Plane, allowing you to focus on developing applications.

- **Support for the latest Kubernetes versions:** VKS is always updating to the latest Kubernetes versions (minor versions from 1.27, 1.28, 1.29) to ensure you can take advantage of the most advanced features.
- **Kubernetes Networking:** VKS integrates Calico CNI, providing high efficiency and security.
- **Upgrade seamlessly:** VKS supports easy and fast upgrades between Kubernetes versions, helping you stay updated with the latest improvements.
- **Scaling & Healing Automatically:** VKS automatically scales the Node group when needed and repairs issues when nodes encounter problems, saving you time and effort in management.
- **Reduce costs and enhance reliability:** VKS deploys the Kubernetes Control Plane in a highly available mode and completely for free, helping you save costs and improve system reliability.
- **Integration of Native Blockstore (Container Storage Interface - CSI):** VKS allows you to manage Blockstore through Kubernetes YAML, providing persistent storage for containers and supporting important features such as resizing, changing IOPS, and snapshotting volumes.
- **Integration of Load Balancer (Network Load Balancer, Application Load Balancer)** through built-in drivers such as VNGCloud Controller Manager, VNGCloud Ingress Controller: VKS provides the capability to manage NLB/ALB through Kubernetes YAML, making it easy to expose Services in Kubernetes to the outside.
- **Enhanced security:** VKS allows you to create Private Node Groups with only Private IPs and control access to the cluster via the IP Whitelist feature, ensuring the safety of your system.

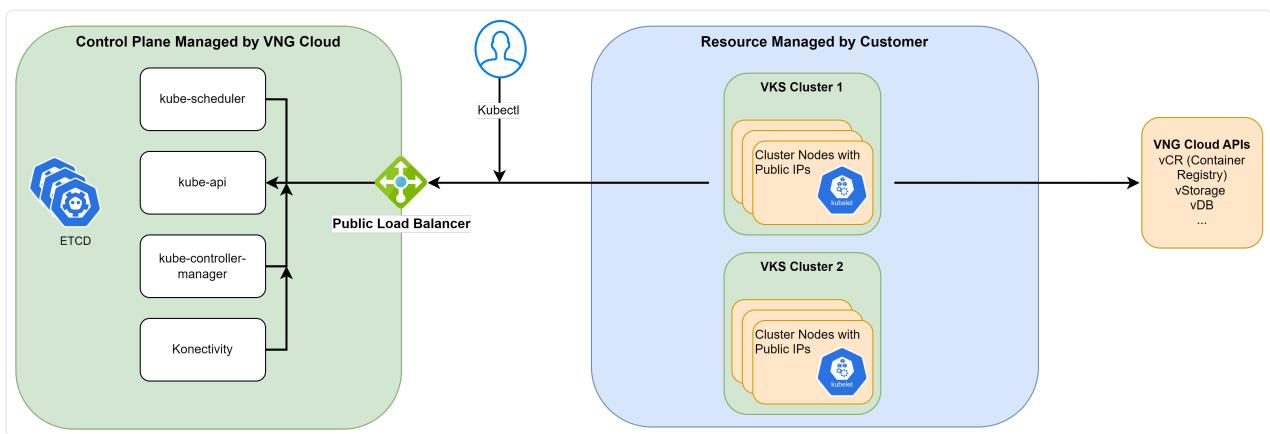
In addition, VKS has the following advantages:

- **Easy to use:** VKS provides a simple and user-friendly interface.
- **Affordable pricing:** VKS offers competitive pricing for its services.

How VKS works?

Below are the current concepts being provided to you by VKS:

1. Public Cluster



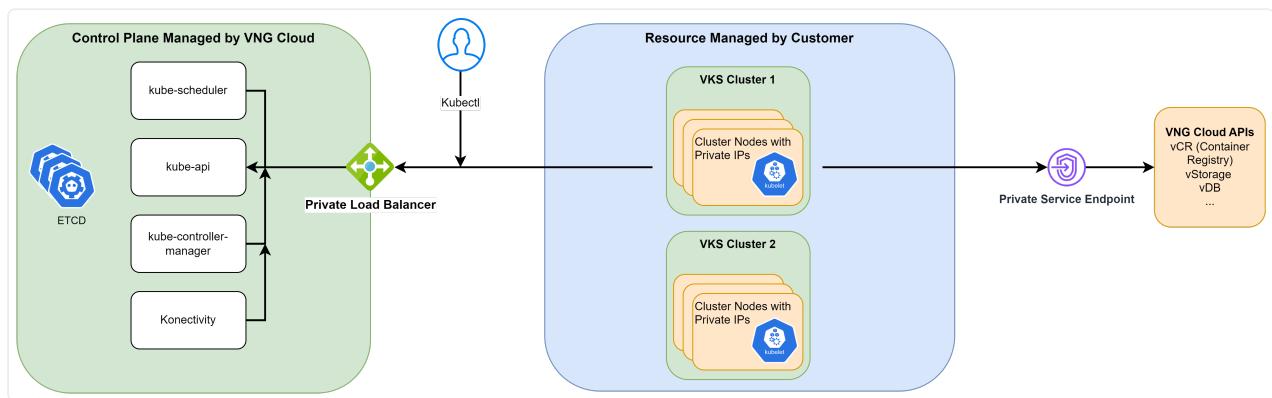
When you create a **Public Cluster with Public Node Group**, the VKS system will:

- Create a VM with Floating IP (ie Public IP). Now these VMs (Nodes) can directly join the K8S cluster through this Public IP. By using Public Cluster and Public Node Group, you can easily create Kubernetes clusters and expose services without using Load Balancer. This will contribute to cost savings for your cluster.

When you create a **Public Cluster with a Private Node Group**, the VKS system will:

- Create VM without Floating IP (ie without Public IP). At this time, these VMs (Nodes) cannot join the K8S cluster directly. In order for these VMs to join the K8S cluster, you need to use a NAT Gateway (**NATGW**). **NATGW** acts as a relay station, allowing VMs to connect to the K8S cluster without needing a Public IP. With VNG Cloud, we recommend you use Pfsense or Palo Alto as a NATGW for your Cluster. Pfsense will help you manage incoming and outgoing network traffic (inbound and outbound traffic) effectively, ensuring network security and access management. Besides, using Private Node Group will help you control applications in the cluster more securely, specifically you can limit control plane access rights through the Whitelist IP feature.

2. Private Cluster



When you create a **Public Cluster with Public/Private Node Group**, the VKS system will:

- To enhance the security of your cluster, we have introduced the private cluster model. The Private Cluster feature helps make your K8S cluster as secure as possible, all connections are completely private from the connection between nodes to the control plane, the connection from the client to the control plane, or the connection from nodes to products. Other services in VNG Cloud such as: vStorage, vCR, vMonitor, VNGCloud APIs,...Private Cluster is the ideal choice for **services that require strict access control, ensuring compliance with security regulations and data privacy**.

3. Comparison between using Public Cluster and Private Cluster

Below is a comparison table between creating and using Public Cluster and Private Cluster on the VKS system:

Criteria	Public Cluster	Private Cluster
Connect	Use Public IP addresses to communicate between nodes and control plane, between clients and control plane, between nodes and	Use Private IP addresses to communicate between nodes and control plane, between clients and control plane, between nodes and

	other services in VNG Cloud.	other services in VNG Cloud.
Security	Medium security since connections use Public IP.	Higher security with all connections private and limited access.
Access management	More difficult to control, access can be managed through the Whitelist feature	Strict access control, all connections are within VNG Cloud's private network, thereby minimizing the risk of external network attacks.
Scalability (AutoScaling)	Easily scalable through Auto Scaling feature .	Easily scalable through Auto Scaling feature .
AutoHealing	Automatically detect errors and restart the node (Auto Healing)	Automatically detect errors and restart the node (Auto Healing)
Accessibility from outside	Easy access from anywhere with internet.	Access from outside must be through other security solutions.
Configuration and deployment	Simpler because it does not require setting up an internal network.	More complex, requires private and secure network configuration.
Cost	Usually lower because there is no need to set up a complex security infrastructure.	Higher cost due to additional security and management components required. Specifically, when using a private cluster, you need to pay for 4 automatically created private service endpoints to connect to services on VNG Cloud.
Flexibility	High, easy to change and access services.	More flexible in applications that require security, but less flexible for applications that require external access.

Therefore:



- **Public Cluster** : Suitable for applications that do not require high security and need flexibility and access from multiple locations. Easy to deploy and manage but has higher security risks.
- **Private Cluster** : Suitable for applications that require high security, strictly complying with security and privacy regulations. Provides stable and secure connectivity, but requires more complex configuration and management, as well as higher costs.

Announcements and Updates

Release notes

•

Dec 5, 2024

VKS (VNGCloud Kubernetes Service) has just released the latest update, bringing many new features to users. Here are the highlights of the update:

New features:

- **Force-Upgrade, Auto-Upgrade** : Automatically upgrade the Kubernetes version for the cluster/node group on schedule or when the current version is about to expire. For more details, please refer [here](#) .
-

Oct 23, 2024

VKS (VNGCloud Kubernetes Service) has just released the latest update, bringing many improvements to users. Here are the highlights of the update:

Improve:

- **Support POC/ Stop POC for Cluster** : Users can now perform POC/ Stop POC for resources on VKS such as Server, Volume, Load Balancer, Endpoint. This feature brings high flexibility to users who want to experience VKS. For more details, please refer [here](#) .
- **Upgrade VNGCloud BlockStorage CSI Driver Plugin**: Bugs discovered in previous versions have been fixed, making the system run smoother and more reliably.
- **Freely choose/edit configuration with/without using VNGCloud Controller Manager plugin, VNGCloud Ingress Controller plugin on existing VKS cluster**: The ability to customize plugin configuration allows users to optimize the VKS cluster according to their specific needs. This helps increase flexibility and meet the special requirements of each application.

- **Additionally**, in this update, we have also fixed some minor bugs to provide a better user experience.
-

Oct 03, 2024

VKS (VNGCloud Kubernetes Service) has just released its latest update, bringing many features and improvements to users. Here are the highlights of the update:

New Region:

- In addition to Region HCM03, VKS now supports Region HAN01. This addition gives customers more options in deploying applications, especially useful for businesses with data location requirements.

New features:

- **Network Type: Cilium Overlay, Cilium VPC Native Routing:** In addition to Calico Overlay, this release we have added two new network types: Cilium Overlay and Cilium VPC Native Routing. Cilium Overlay allows you to build flexible overlay networks, while Cilium VPC Native Routing integrates tightly with VNG Cloud's VPC, optimizing performance and security for your applications. For more details, please refer [here](#).

Improve:

- **Multiple Subnets for Clusters on VKS:** VKS now supports using multiple subnets for a cluster. This allows you to configure each node group in the cluster to be located on different subnets within the same VPC, optimizing resource allocation and network management.
- **Edit Labels/Taints on an existing VKS cluster:** With the ability to directly edit Labels/Taints on a deployed VKS cluster, you can control Pod scheduling, apply different policies to Node Groups, and customize node selection rules for applications. This helps manage and classify resources more efficiently.
- **Enable/Disable Private Service Endpoint usage option:** Previously, when creating a private cluster on VKS, creating a private service endpoint was required. Now, you can easily enable/disable this feature, allowing services in

the VKS cluster to communicate via internal IP addresses, enhancing security and minimizing the risk of external attacks.

- **Enable/Disable Volume Encryption Option:** Volume encryption feature allows you to protect sensitive data stored in the Persistent Volumes of the VKS cluster. This ensures data security and compliance with information protection regulations. Now, you can enable/disable encryption for each Volume as needed.
-

Aug 28, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the existing VKS, bringing many new features to users. Here are the details about the update:

New features:

- **Private Cluster:** Previously, public clusters on VKS were using Public IP addresses to communicate between nodes and the control plane. To improve the security of your cluster, we have launched the private cluster model. The Private Cluster feature helps your K8S cluster to be as secure as possible, all connections are completely private from the connection between the nodes to the control plane, the connection from the client to the control plane, or the connection from the nodes to other products and services in VNG Cloud such as: vStorage, vCR, vMonitor, VNGCloud APIs,... **Private Cluster is the ideal choice for services that require strict access control, ensuring compliance with regulations on security and data privacy.** For details on the two operating models of Cluster, you can refer to [here](#) and refer to the steps to create a private Cluster [here](#).
-

Aug 26, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update for VKS, bringing numerous new improvements for users. Here are the details of the update:

Improve:

- **Kubernetes Version:** VKS has added new images to optimize the size, features, and network compared to the old images. The creation of these images also aims to serve both Public and Private clusters that VKS is about to launch. Specifically, in this release, we have added the following images:
 - Ubuntu-22.kube_v1.27.12-vks.1724605200
 - Ubuntu-22.kube_v1.28.8-vks.1724605200
 - Ubuntu-22.kube_v1.29.1-vks.1724605200



Chú ý:

- Để khởi tạo một **Private Cluster**, bạn cần chọn sử dụng một trong 3 image mới này. Đối với Public Cluster, bạn có thể chọn sử dụng bất kỳ image cũ hoặc mới tùy theo nhu cầu của bạn.

Aug 13, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the existing VKS, bringing many new improvements to users. Here are the details about the update:

Improve:

- **Event History:** VKS has added events for **Auto Scaling** and **Auto Healing**. Now, with Event History, you can track every change occurring within your Cluster, from automatic scaling to automatic healing. These events enhance your ability to monitor and manage your Kubernetes cluster.

Aug 01, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the already available VKS, bringing many new features and improvements to users. Here are the details about the update:

New feature:

- **VKS resource monitoring:** Users can directly monitor the operating status of Cluster, Node, CPU usage, RAM, Memory,... status of Node through intuitive dashboards. To display data on the dashboard, users need to install it `vmonitor-metric-agent` on the cluster where they want to perform monitoring. For more details, refer [here](#).
-

July 25, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the existing VKS, bringing many new improvements to users. Here are the details about the update:

Improve:

- **Upgrade VKS management through Terraform:** Users can simultaneously adjust the number of nodes and change the number of nodes for autoscale (Minimum/ Maximum node Autoscale) right during the configuration editing process. With the ability to adjust multiple parameters at the same time, managing a Kubernetes cluster becomes more flexible and convenient. For more details, see examples [here](#).
-

July 23, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the existing VKS, bringing many new improvements to users. Here are the details about the update:

Improve:

- **Upgrade VNGCloud Controller Manager Plugin, VNGCloud Ingress Controller Plugin:** Errors discovered in previous versions have been fixed, helping the system operate smoother and more reliably.
-

July 18, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the existing VKS, bringing many new improvements to users. Here are the details about the update:

Improve:

- **Upgrade VNGCloud BlockStorage CSI Driver Plugin:** Errors discovered in previous versions have been fixed, helping the system operate smoother and more reliably.
-

July 17, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the already available VKS, bringing many improvements to users. Here are the details about the update:

Improve:

- **Private Node Group :** The MTU for Nodes belonging to the Private Node Group has been updated to 1450. This improves network performance for applications running in the Private Node Group.
 - **Number of nodes and AutoScale :** You can now edit both of these properties in the same API. This simplifies your Cluster management.
-

July 02, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the already available VKS, bringing many new features and improvements to users. Here are the details about the update:

New feature:

- **Stop POC support for Cluster :** Users can now easily perform Stop POC for all resources being POC on a Cluster, instead of having to perform Stop POC individually for each resource. This helps save time and effort when moving Cluster from test resources to real resources. For more details, refer [here](#) .

Improve:

- **Node Group status : Added " Degraded "** status so users can monitor the operating status of the Node Group more accurately. This status will display when the number of active nodes is less than the actual number of replicas.
- **Timeout for Cluster and Node Group :** Added timeout for Cluster and Node Group creation, this improvement ensures VKS operates smoothly and efficiently, while providing clear and timely information to users. use. Timeout for Cluster creation is **1 hour** and for Node Group is **3 hours** . If after this time your Cluster or Node Group has not been successfully created, we will update their status to ERROR. At this point, you can delete and create another Cluster or Node Group instead.
- **KubeConfig Access:** Access to the KubeConfig file is now only allowed when the Cluster is Active. This improvement helps users avoid configuration errors when using Terraform to automate Kubernetes deployments.

June 27, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the existing VKS, bringing many new improvements to users. Here are the details about the update:

Improve:

- **Upgrade VNGCloud Controller Manager Plugin, VNGCloud Ingress Controller Plugin:** Errors discovered in previous versions have been fixed, helping the system operate smoother and more reliably.
-

June 19, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the existing VKS, bringing many new improvements to users. Here are the details about the update:

Improve:

- **Upgrade PVC size setting feature (Persistent Volume Claim Size):** Users can now specify the minimum size for CSI drives to be **1GB** instead of the minimum size of 20GB as before. For more details, you can refer to Volume and Integrate with Container Storage Interface .
- **Change the default Storage Class used for Cluster:** change the default from SSD type drives - IOPS 200 to the default SSD type drives - IOPS 3000.
- **Upgrade VNGCloud Controller Manager Plugin, VNGCloud Ingress Controller Plugin:** plugin improvements help avoid duplicate Load Balancer naming.

Attention:

Because the old default Storage Class has been removed from the system, if you want to continue using and resize this storage class, you can:

- Create a Storage Class named sc-iops-200-retain with the Volume Type you desire.
- Resize Storage Class via command:

Copy

```
kubectl patch pvc sc-iops-200-retain -p '{"spec": {"resources": {"requests":
```

For more details, refer to Integrate with Container Storage Interface .

June 12, 2024

VKS (VNGCloud Kubernetes Service) introduces the latest update to the already available VKS, bringing many new features and improvements to users. Here are the details about the update:

New feature:

- **Support users working with VKS through Terraform:** Users can easily create Clusters and Node Groups in VKS using Terraform. For more details, refer [here](#) .

Improve:

- **Upgrade VNGCloud Controller Manager Plugin:** Add Annotation to configure Load Balancer to support Proxy Protocol. For more details, refer [here](#) .
-

May 30, 2024

We are extremely pleased to announce that the official release (**General Availability**) of VNGCloud Kubernetes Service is available. With this official release, in addition to the features we have provided on previous releases, this version will bring many new features and improvements to users. Here are the details about the update:

New feature:

- **Re-activate:** VKS allows you to request the system to automatically re-initialize the default IAM Service Account when you mistakenly delete or change previously created default IAM Service Account information. The default IAM Service Account is the IAM Service Account that is automatically created by the

VKS system when you start working with VKS. We will use this IAM Service Account to initialize resources for your Cluster.

- **Event History** : VKS will display the history of events that occur when users work with the Cluster or each Node Group. This will be a way to help you monitor activities occurring with your Cluster, thereby limiting unusual activities from occurring.
- **Volume** : VKS has integrated the display of the Volume list at the Resource Tab, helping you easily manage the Volumes that are attached to your Cluster.
- **Load Balancer** : VKS has integrated the display of the Load Balancer list in the Resource Tab, helping you easily manage the Load Balancers being used for your Cluster.

Improve:

- **Performance** : VKS has optimized performance when initializing the Cluster. Specifically, in the Alpha version, the time from the start of Cluster initialization (with Default Node Group) to the time the Cluster switches to **ACTIVE** state is about 04:00s to 04:30s. Currently, this time has been optimized by us to **02:30s to 03:00s** depending on each Cluster and each time you initialize.
- **Garbage collection of unused containers and images** : VKS will automatically delete unused images when the disk reaches the usage limit (usage/quota ratio $\geq 85\%$).
- **In addition** , this GA version has improved a few other issues such as:
 - Changing **the Node Name** helps you easily use and manage your Cluster. Specifically, the Node Name will have additional information: **Cluster Name** , **Node Group Name** .
 - Delete **User Builder** on User's Worker Node.
 - **Change SSH** mechanism from Port 22 to Port 234.

If you encounter any problems with this official release, please contact VKS support for assistance.

May 03, 2024

The latest update for VKS is available, bringing many new features and improvements to users. Here are the details about the update:

New feature:

- **Supports Whitelist feature:** VKS allows creating a Private Node Group with only Private IP and also allows any IP to connect to the Cluster through the Whitelist IP feature. For more details, please refer to Whitelist .

Improve:

- **System optimization:** Helps the system operate more smoothly and efficiently.
- **Bug Fixes:** Fixed some minor bugs to provide a better user experience.

If you encounter any problems after updating, please contact VKS support for assistance.

April 17, 2024

We're excited to introduce a new update to the VKS service, giving you a more powerful and efficient Kubernetes management experience than ever before!

Highlights:

- **Fully Managed control plane:** VKS will free you from the burden of managing Kubernetes' Control Plane, helping you focus on application development.
- **Supports the latest versions of Kubernetes:** VKS always updates the latest versions of Kubernetes (minor versions from 1.27, 1.28, 1.29) to ensure you always take advantage of the most advanced features.
- **Kubernetes Networking:** VKS integrates Calico CNI, providing high efficiency and security.
- **Upgrade seamlessly:** VKS supports upgrading between Kubernetes versions easily and quickly, helping you stay up to date with the latest improvements.

- **Scaling & Healing Automatically:** VKS automatically expands the Node group when necessary and automatically fixes errors when the node has problems, helping you save time and effort on management.
- **Reduce costs and improve reliability:** VKS deploys Control Plane of Kubernetes in high availability mode and is completely free, helping you save costs and improve system reliability.
- **Blockstore Native (Container Storage Interface - CSI) integration:** VKS allows you to manage Blockstore through Kubernetes' YAML, providing persistent storage for containers and supporting important features such as resizing, IOPS scaling and snapshot volumes.
- **Integrate Load Balancer (Network Load Balancer, Application Load Balancer) through built-in drivers such as VNGCloud Controller Manager, VNGCloud Ingress Controller:** VKS provides the ability to manage NLB/ALB through YAML of Kubernetes, making it easy for you expose Service in Kubernetes to the outside.
- **Enhance security:** VKS allows you to create a Private Node Group with only Private IP and control access to the cluster through the IP Whitelist feature, ensuring the safety of your system.

With these breakthrough features, VKS promises to bring you a completely new Kubernetes management experience, helping you optimize efficiency and save costs!

Please contact us for further advice and support!

Getting Started with VKS

.

Instructions for installing and configuring the kubectl in Kubernetes

Necessary conditions

You need to use a kubectl version that is no more than one version different from the cluster version. For example, a client v1.2 should work with master v1.1, v1.2 and v1.3. Using the latest version of kubectl helps avoid unforeseen problems.

Install kubectl on Linux

Install kubectl binary with curl on Linux

Step 1: Download the latest version with the command:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl
```

To download a specific version, replace the section in the command with a specific version. `$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)`

For example, to download version v1.17.0 on Linux, run the command:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.17.
```

Step 2: Create kubectl binary executable via command:

```
chmod +x ./kubectl
```

Step 3: Put the binary into your PATH environment variable via the command:

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

•

Step 4: Check to make sure that the version you installed is the latest via command:

```
kubectl version
```

Install kubectl using the package manager

- With CentOS, RHEL or Fedora operating systems, you can run the command:

```
sudo apt-get update && sudo apt-get install -y apt-transport-https curl -s https://apt.kubernetes.io/  
kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes
```

Install kubectl with snap

If you are using [Ubuntu](#) or another Linux distro that supports the [snap](#) package manager , kubectl is already available in [snap](#) .

Step 1: Switch to user snap and execute the installation command:

```
sudo snap install kubectl --classic
```

Step 2: Check the version you just installed is the latest:

```
kubectl version
```

Install kubectl on macOS

Install kubectl binary with curl on macOS

Step 1: Download the latest version:

```
kubectl version
curl -LO "
https://storage.googleapis.com/kubernetes-release/release/$(curl
-s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin
"
```

To download a specific version, replace the section in the command with the specific version. For example, to download v1.17.0 on macOS, run the command:

```
$(curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt )
```

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.17.
```

Step 2: Create kubectl binary executable via command:

```
chmod +x ./kubectl
```

Step 3: Put the binary into your PATH environment variable via the command:

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

Step 4: Check to make sure that the version you installed is the latest via command:

```
kubectl version
```

Install with Homebrew on macOS

If you are on macOS and using the [Homebrew](#) package manager , you can install kubectl with Homebrew.

Step 1: Run the installation command:

```
brew install kubernetes-cli
```

or command:

```
brew install kubectl
```

Step 2: Check to make sure the version you installed is the latest:

```
kubectl version
```

Install with Macports on macOS

If you are on macOS and using the [Macports](#) package manager , you can install kubectl with Macports.

Step 1: Run the installation command:

```
sudo port selfupdatessudo port install kubectl
```

Step 2: Check to make sure the version you installed is the latest:

```
sudo port selfupdatessudo port install kubectl
```

Install kubectl on Windows

Install kubectl binary with curl on Windows

Step 1: Download the latest version v1.17.0 from [this link](#) . Or if you have already installed it `curl` , use the following command:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.17.
```

To find out the latest stable version, see
<https://storage.googleapis.com/kubernetes-release/release/stable.txt>.

Step 2: Include the binary in your PATH environment variable.

Step 3: Check to make sure the version `kubectl` is the same as the downloaded version:

```
kubectl version
```

Note: [Docker Desktop for Windows](#) `kubectl` adds its own version to the PATH. If you have previously installed Docker Desktop, you may need to set your PATH before the Docker Desktop installation adds a PATH to or removes `kubectl` Docker Desktop's.

Install with Powershell from PSGallery

[If you are on Windows and using the Powershell Gallery package manager](#), you can install and update `kubectl` with Powershell.

Step 1: Execute the following installation commands (make sure you define your own `DownloadLocation`):

```
Install-Script -Name install-kubectl -Scope CurrentUser -Forceinstall-kub
```

Note: If you do not define it `DownloadLocation`, `kubectl` will be installed in the user's temp directory.

The installation will generate `$HOME/.kube` and instruct you to create a configuration file

Step 1: Check to make sure the version you installed is the latest:

```
kubectl version
```

Note: Installation updates will be performed when re-running the commands from step 1.

Install on Windows using Chocolatey or Scoop

Step 1: To install kubectl on Windows you can use the [Chocolatey](#) package manager or the [Scoop](#) command installer .

- If you use Choco

```
choco install kubernetes-cli
```

- If you use Scoop

```
scoop install kubectl
```

Step 2: Check to make sure the version you installed is the latest:

```
kubectl version
```

Step 3: Move to your home directory:

```
cd %USERPROFILE%
```

Step 4: Create folder `.kube` :

```
mkdir .kube
```

Step 5: Move to the folder `.kube` you just created:

```
cd .kube
```

Step 6: Configure kubectl to use a remote Kubernetes cluster:

```
New-Item config -type file
```



Note: Edit the configuration file with a text editor, such as Notepad.

Download from part of the Google Cloud SDK

You can install kubectl from part of the Google Cloud SDK.

Step 1: Install [Google Cloud SDK](#).

Step 2: Execute the installation command `kubectl`:

```
gcloud components install kubectl
```

Step 3: Check to make sure the version you installed is the latest:

```
kubectl version
```

Verify kubectl configuration

1. For kubectl to search and access your Kubernetes cluster, it needs a kubeconfig file, which is automatically created when you create a new cluster using [kube-up.sh](#) or successfully deploy a Minikube cluster. By default, kubectl's configuration is defined at `~/.kube/config`.
2. Check kubectl is configured correctly by viewing the cluster status: `kubectl cluster-info`

```
kubectl cluster-info
```

1. If you see a response URL, kubectl is properly configured to access your cluster.
2. If you see a message similar to the one below, kubectl is not configured correctly or cannot connect to the Kubernetes cluster.

The connection to the server <server-name:port> was refused - did you specify the right host or port?

For example, if you are planning to run a Kubernetes cluster on your laptop (locally), you will need a tool like Minikube installed previously and run the commands above again. If kubectl cluster-info returns the url but you cannot access your cluster, then check if it is configured correctly, by:

```
kubectl cluster-info dump
```

kubectl configuration options

Enable shell completion

- kubectl provides autocomplete support for Bash and Zsh, helping you reduce the need to type many commands.
- Below are the steps to set up autocomplete for Bash (including the differences between Linux and macOS) and Zsh.

Bash on Linux

Introduce

- Kubelet completion script for Bash is generated with the command . After the script is created, you need to source (execute) the script to enable the autocomplete feature. `kubectl completion bash`
- However, the completion script depends on [bash-completion](#), so you must install bash-completion beforehand (check bash-completion exists with the command `type _init_completion`).

Install bash-completion

1. bash-completion is provided by many package managers (see [here](#)). You can install with command `apt-get install bash-completion` or `yum install bash-completion`.
2. The above commands generate `/usr/share/bash-completion/bash_completion`, which is the main script of bash-completion. Depending on your package manager, you may have to source this file in a `~/.bashrc`.
3. To find this file, reload your current shell and run the `type _init_completion`. If successful, you are done setting up, otherwise add the following to `~/.bashrc` your file:

```
source /usr/share/bash-completion/bash_completion
```

1. Reload your shell and confirm that bash-completion is installed correctly with the command `type _init_completion`.

Enable kubectl autocompletion

Now you need to ensure that the kubectl completion script is sourced across all shell sessions. There are 2 ways to do this:

- Source script in file `~/.bashrc`:

```
echo 'source <(kubectl completion bash)' >>~/.bashrc
```

- Add script to folder `/etc/bash_completion.d`:

```
kubectl completion bash >/etc/bash_completion.d/kubectl
```

- If you have an alias for kubectl, you can add another shell completion to that alias:

Copy

```
echo 'alias k=kubectl' >>~/.bashrc
echo 'complete -F __start_kubectl k'
```

Note: bash-completion sources all completion scripts in `/etc/bash_completion.d`.

The above methods are equally effective. After reloading the shell, kubectl autocomplete will work.

Bash on macOS

Introduce

Kubectl completion script on Bash is generated by `kubectl completion bash`. This source script will enable the kubectl completion feature.

However, the kubectl completion script depends on [the bash-completion](#) you installed earlier.

Warning: There are two versions of bash-completion, v1 and v2. V1 is for Bash 3.2 (default Bash on macOS), and v2 is for Bash 4.1+. Kubectl completion script **does not work** properly with bash-completion v1 and Bash 3.2. It is compatible with **bash-completion v2 and Bash 4.1+**. Therefore, to use kubectl completion correctly on macOS, you must install Bash 4.1+ ([instructions](#)). The instructions that follow assume that you are using Bash 4.1+ (that is, any Bash version 4.1 or later).

Install bash-completion

Note: As mentioned, these instructions assume you are using Bash 4.1+, which means that you will be installing bash-completion v2 (as opposed to Bash 3.2 and bash-completion v1, in which case, kubectl completion will not work).

1. You can check if bash-completion v2 was previously installed with the command `type _init_completion`. If not, you can install it with Homebrew:

```
brew install bash-completion@2
```

- From the output of this command, add the following to `~/.bashrc` your file:
`export BASH_COMPLETION_COMPAT_DIR="/usr/local/etc/bash_completion.d"`

```
export BASH_COMPLETION_COMPAT_DIR="/usr/local/etc/bash_completion.d" [[ -r
```

Reload your shell and verify that bash-completion v2 is installed correctly using the
`type _init_completion`.

Enable kubectl autocompletion

Now you must ensure that the kubectl completion script has been sourced in all your shell sessions. There are many ways to achieve this:

- Source completion script in file `~/.bashrc` :

```
echo 'source <(kubectl completion bash)' >>~/.bashrc
```

- Add completion script to the folder `/usr/local/etc/bash_completion.d` :

```
kubectl completion bash >/usr/local/etc/bash_completion.d/kubectl
```

- If you have an alias for kubectl, you can extend the completion shell to work with that alias:

Copy

```
echo 'alias k=kubectl' >>~/.bashrcecho 'complete -F __start_kubectl k'
```

- If you have installed kubectl with Homebrew (as introduced [above](#)) then the `/usr/local/etc/bash_completion.d/kubectl`. In this case, you don't need to do anything.

Note: Installing the Homebrew way already sources all the files in the `BASH_COMPLETION_COMPAT_DIR`, which is why the following two methods work.

In any case, after reloading your shell, kubectl completion should work.

Zsh

1. Kubectl completion script for Zsh is generated with the command

```
kubectl completion zsh
```

. Source completion script in your shell will enable kubectl autocompletion. To make it work for all shells, add the following line to the file `~/.zshrc`:

```
source <(kubectl completion zsh)
```

1. If you have an alias for kubectl, you can extend the completion shell to work with that alias:

```
echo 'alias k=kubectl' >>~/.zshrc
echo 'complete -F __start_kubectl k' >>~/.zshrc
```

After reloading the shell, kubectl autocompletion should work.

1. If you get the error complete:13: command not found: compdef, add the following line at the beginning of the file `~/.zshrc`:

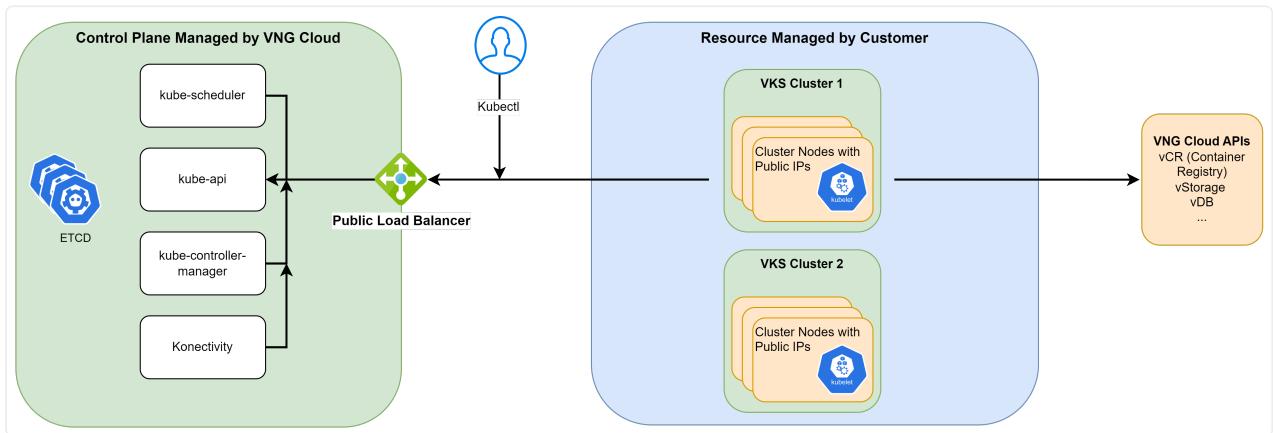
```
autoload -Uz compinit
compinit
```

For more details, see kubernetes.io

Create a Public Cluster

.

Model



When you create a **Public Cluster with Public Node Group**, the VKS system will:

- Create a VM with Floating IP (ie Public IP). Now these VMs (Nodes) can directly join the K8S cluster through this Public IP. By using Public Cluster and Public Node Group, you can easily create Kubernetes clusters and expose services without using Load Balancer. This will contribute to cost savings for your cluster.

When you create a **Public Cluster with a Private Node Group**, the VKS system will:

- Create VM without Floating IP (ie without Public IP). At this time, these VMs (Nodes) cannot join the K8S cluster directly. In order for these VMs to join the K8S cluster, you need to use a NAT Gateway (**NATGW**). **NATGW** acts as a relay station, allowing VMs to connect to the K8S cluster without needing a Public IP. With VNG Cloud, we recommend you use Pfsense or Palo Alto as a NATGW for your Cluster. Pfsense will help you manage incoming and outgoing network traffic (inbound and outbound traffic) effectively, ensuring network security and access management. Besides, using Private Node Group will help you control applications in the cluster more securely, specifically you can limit control plane access rights through the Whitelist IP feature.

Create a Public Cluster with Public Node Group

•

Prerequisites

To be able to initialize a **Cluster** and **Deploy a Workload**, you need:

- There is at least 1 **VPC** and 1 **Subnet in ACTIVE state**. If you do not have a VPC or Subnet yet, please create a VPC or Subnet according to the instructions here [.](#)
 - There is at least 1 **SSH key in ACTIVE state**. If you do not have any SSH key, please create an SSH key according to the instructions here [.](#)
 - Installed and configured **kubectl** on your device. Please refer here [if](#) you are not sure how to install and use kubectl. In addition, you should not use a kubectl version that is too old, we recommend that you use a kubectl version that is no more than one version different from the cluster version.
-

Initialize Cluster

A **cluster in Kubernetes** is a collection of one or more virtual machines (VMs) connected together to run containerized applications. Cluster provides a unified environment to deploy, manage, and operate containers at scale.

To initialize a Cluster, follow the steps below:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select **Activate**.

Step 3: Wait until we successfully create your VKS account. After Activate successfully, select **Create a Cluster**

Step 4: At the Cluster initialization screen, we have set up information for the Cluster and a **Default Node Group** for you. You can keep these default values or adjust the desired parameters for the Cluster and Node Group at Cluster Configuration, Default Node Group Configuration, Plugin. **By default we will create a Public Cluster for you with Public Node Group.**

Step 5: Select **Create Kubernetes cluster**. Please wait a few minutes for us to initialize your Cluster, the Cluster's status is now **Creating**.

Step 6: When the Cluster status is **Active**, you can view Cluster information and Node Group information by selecting Cluster Name in the **Name** column.

Connect and check the newly created Cluster information

After the Cluster is successfully initialized, you can connect and check the newly created Cluster information by following these steps:

Step 1: Visit <https://vks.console.vngcloud.vn/k8s-cluster>

Step 2: The Cluster list is displayed, select the icon and select **Download Config File** to download the kubeconfig file. This file will give you full access to your Cluster.

Step 3 : Rename this file to config and save it to the **~/.kube/config** directory

Step 4: Perform Cluster check via command:

- Run the following command to test **node**

```
kubectl get nodes
```

- If the results are returned as below, it means your Cluster was successfully initialized with 3 nodes as below.

NAME	STATUS	•	ROLES	AGE
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready		<none>	50m
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready		<none>	23m
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready		<none>	28m

Deploy a Workload

The following is a guide for you to deploy the nginx service on Kubernetes.

Step 1 : Create Deployment and Service for Nginx app

- Create **nginx-service.yaml** file with the following content:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
  spec:
    containers:
      - name: nginx
        image: nginx:1.19.1
        ports:
          - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80

```

- Deploy This deployment equals:

```
kubectl apply -f nginx-service.yaml
```

Step 2: Check Deployment and Service information before exposing it to the Internet.

- Run the following command to test **Deployment**

```
kubectl get svc,deploy,pod -owide
```

- If the results are returned as below, it means you have successfully deployed the nginx service.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP
service/nginx-service	ClusterIP	10.96.178.229	<none>	80/TCP
NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx-app	1/1	1	1	74s
NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-app-7f45b65946-5pcvz	1/1	Running	0	74s
				172.1

Step 3: Expose Nginx Service to the Internet

- Run the following command to expose nginx-service to the internet:

```
kubectl expose deployment nginx-app --type=NodePort --port=30080 --target
```

- If the results are returned as below, it means you have successfully exposed the Service to the Internet.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP
service/nginx-app	NodePort	10.96.215.192	<none>	30080:3
service/nginx-service	ClusterIP	10.96.178.229	<none>	80/TCP
NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx-app	1/1	1	1	2m43s
NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-app-7f45b65946-5pcvz	1/1	Running	0	2m43s
				172

Step 4: To access the just exported Nginx app, you can use the URL with the format:

`http://<node_ip>:31289/`



Where node_ip can be the node_port address of any node in the cluster. You can get Node's External IP information at the vServer interface. Specifically, access at <https://hcm-3.console.vngcloud.vn/vserver/v-server/cloud-server> .

For example, below I have successfully accessed the nginx app with the address:
<http://61.28.231.65:31007/>

If you want to expose this service through vLB Layer4, vLB Layer7, please refer to:

- [Expose a service through vLB Layer4](#)
- [Expose a service through vLB Layer7](#)

Create a Public Cluster with Private Node Group

•

Prerequisites

To be able to initialize a **Cluster** and **Deploy a Workload**, you need:

- There is at least 1 **VPC** and 1 **Subnet in ACTIVE state**. If you do not have a VPC or Subnet yet, please create a VPC or Subnet according to the instructions here [.](#)
- There is at least 1 **SSH key in ACTIVE state**. If you do not have any SSH key, please create an SSH key according to the instructions here [.](#)
- Installed and configured **kubectl** on your device. Please refer here [if](#) you are not sure how to install and use kubectl. In addition, you should not use a kubectl version that is too old, we recommend that you use a kubectl version that is no more than one version different from the cluster version.



Attention:

- To ensure that VMs in NodeGroups on the subnet can go outbound to the internet and connect to the Control Plane, you must set up a NAT Gateway. For more details, please refer to the section below.

Create Palo Alto or Pfsense as an alternative to NAT Gateway



Attention:

- For the best support when using Palo Alto or Pfsense, please contact our team of experts via Hotline **1900 1549** or email **support@vngcloud.vn**

Or you can choose to use Palo Alto or Pfsense to work with Private Node Group according to instructions at:

- [Palo Alto as a NAT Gateway](#)
 - [Pfsense as a NAT Gateway](#)
-

Initialize Route Table

After Palo Alto, Pfsense is successfully initialized, you need to create a Route table to connect to different networks. Specifically, follow these steps to create a Route table:

Step 1: Visit <https://hcm-3.console.vngcloud.vn/vserver/network/route-table>

Step 2: In the navigation menu bar, select **Network Tab/ Route table**.

Step 3: Select **Create Route table**.

Step 4: Enter a descriptive name for the Route table. Route table names can include letters (az, AZ, 0-9, '_', '-'). The input data length is between 5 and 50. It must not include leading or trailing spaces.

Step 5: Select **VPC** for your Route table. If you do not have a VPC, you need to create a new VPC according to the instructions on [the VPC Page](#). **The VPC used to set up the Route table must be the VPC selected for Palo Alto or Pfsense and your Cluster.**

Step 6 : Select **Create** to create a new Route table.

Step 7: Select the newly created Route table then select **Edit Routes**.

Step 8: In the add new **Route** section , enter the following information:

- For Destination, enter **Destination CIDR as 0.0.0.0/0**
 - For Target, enter **Target CIDR as the corresponding Palo Alto or Pfsense Network Interface IP address.**
-

Initialize Cluster

A cluster in Kubernetes is a collection of one or more virtual machines (VMs) connected together to run containerized applications. Cluster provides a unified environment to deploy, manage, and operate containers at scale.

To initialize a Cluster, follow the steps below:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select Activate.

Step 3: Wait until we successfully create your VKS account. After Activate successfully, select **Create a Cluster**

Step 4: At the Cluster initialization screen, we have set up information for the Cluster and a **Default Node Group** for you. You can keep these default values or adjust the desired parameters for the Cluster and Node Group at Cluster Configuration, Default Node Group Configuration, Plugin. **By default we will create a Public Cluster for you with Public Node Group. You need to change your selection to Private Node Group .**

Step 5: Select **Create Kubernetes cluster**. Please wait a few minutes for us to initialize your Cluster, the Cluster's status is now **Creating** .

Step 6: When the Cluster status is Active , you can view Cluster information and Node Group information by selecting Cluster Name in the Name column .

Connect and check the newly created Cluster information

After the Cluster is successfully initialized, you can connect and check the newly created Cluster information by following these steps:

Step 1: Visit <https://vks.console.vngcloud.vn/k8s-cluster>

Step 2: The Cluster list is displayed, select the icon and select **Download config file** to download the kubeconfig file. This file will give you full access to your Cluster.

Step 3 : Rename this file to config and save it to the **~/.kube/config** directory

Step 4: Perform Cluster check via command:

- Run the following command to test **node**

```
kubectl get nodes
```

- If the results are returned as below, it means your Cluster was successfully initialized with 3 nodes as below.

NAME	STATUS	ROLES	AGE
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready	<none>	50m
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready	<none>	23m
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready	<none>	28m

Deploy a Workload

The following is a guide for you to deploy the nginx service on Kubernetes.

Step 1: Create Deployment for Nginx app.

- Create **nginx-service-lb4.yaml** file with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.19.1
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- Deploy This deployment equals:

```
kubectl apply -f nginx-service-lb4.yaml
```

Step 2: Check Deployment and Service information before exposing it to the Internet.

- Run the following command to test **Deployment**

```
kubectl get svc,deploy,pod -owide
```

- If the results are returned as below, it means you have successfully deployed the nginx service.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/
service/nginx-app	NodePort	10.96.215.192	<none>	3008
service/nginx-service	LoadBalancer	10.96.179.221	<pending>	80:3
NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx-app	1/1	1	1	2m16s
NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-app-7f45b65946-t7d7k	1/1	Running	0	2m16s
				172

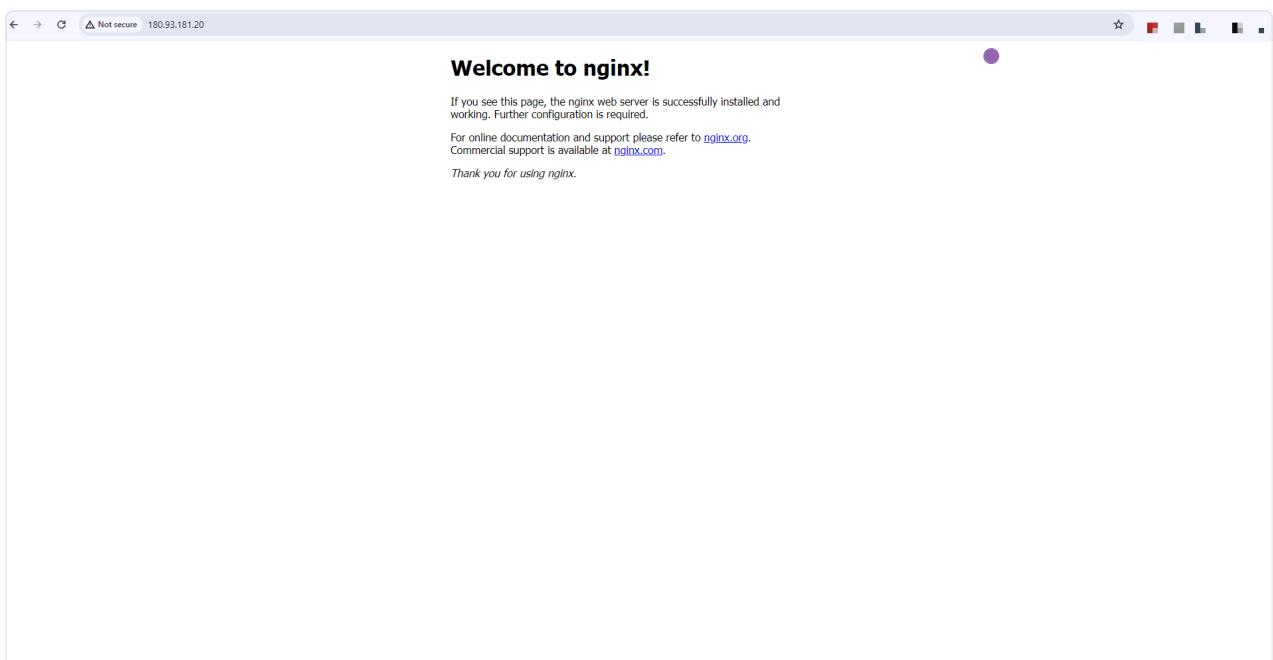
Step 3: To access the just exported nginx app, you can use the URL with the format:

```
http://Endpoint/
```

You can get Load Balancer Public Endpoint information at the vLB interface.

Specifically, access at <https://hcm-3.console.vngcloud.vn/vserver/load-balancer/vlb/>

For example, below I have successfully accessed the nginx app with the address:
<http://180.93.181.20/>



Palo Alto as a NAT Gateway •

Use the instructions below to work with a Private Node group through Palo Alto.

Prerequisites

To be able to use Palo Alto as NAT Gateway for Cluster on VKS system, you need:

- A **Windows server (VM)** has been initialized on the **vServer** system with the following configuration:

Item	Configuration
Flavor	2×4
Volume	20GB
VPC	10.76.0.0/16
Subnet	10.76.0.4/24
Network Interface 1	10.76.0.3

- A **Palo Alto server (VM)** is initialized on the **vMarketPlace** system according to the instructions below with the following configuration:

Item	Configuration
Flavor	2×8
Volume	60GB
VPC	10.76.0.0/16
Network Interface 1	10.76.255.4
Network Interface 2	10.76.0.4

Initialize Palo Alto

Step 1: Visit <https://marketplace.console.vngcloud.vn/>

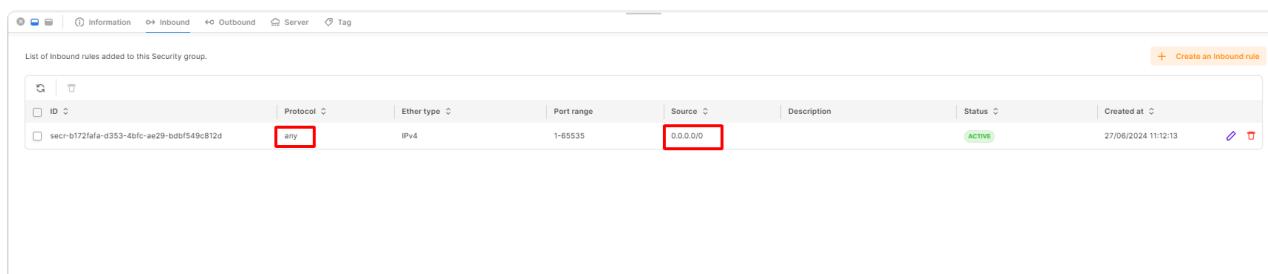
Step 2: At the main screen, search for **Palo Alto** , at **Palo Alto services** , select **Launch** .

Step 3: Now, you need to configure **Palo Alto**. Specifically, you can select the desired **Volume, IOPS, Network, Security Group** . You need to choose the same **VPC and Subnet** as the **VPC and Subnet** you choose to use for your Cluster. In addition, you also need to select an existing Server Group or select **Dedicated SOFT ANTI AFFINITY group** so we can automatically create a new server group.

Step 4: Proceed to pay like normal resources on VNG Cloud.

Configure parameters for Palo Alto

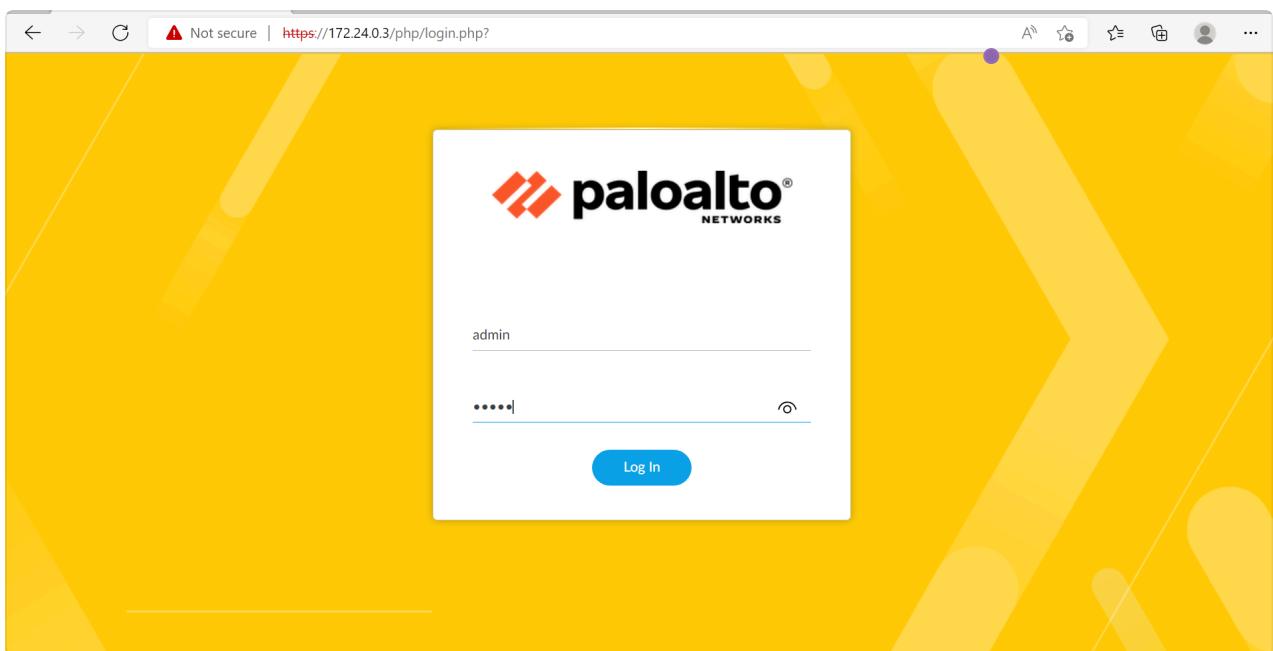
Step 1: After initializing Palo Alto from vMarketPlace according to the instructions above, you can access the vServer interface here [to](#) check if the server running Palo Alto has been initialized. **Next, open the Any rule on the Security Group for the Palo Alto server you just created. Opening the Any rule on the Security Group will allow all traffic to the Palo Alto server.**



ID	Protocol	Ether type	Port range	Source	Description	Status	Created at	Actions
secr-b172fafa-d353-4bfc-ae29-bdbf549c812d	any	IPv4	1-65535	0.0.0.0/0		ACTIVE	27/06/2024 11:12:13	Edit Delete

Step 2: After the server running Palo Alto is successfully initialized . To access the Palo Alto GUI you need a vServer running Windows. Then you access it using IP Internal Interface with the default login name and password: **admin/admin**

Note: Go to the Network section of vServer Windows to access the Palo Alto GUI. You need to create the same VPC and use a different subnet than the subnet with priority 1 when initializing Palo Alto



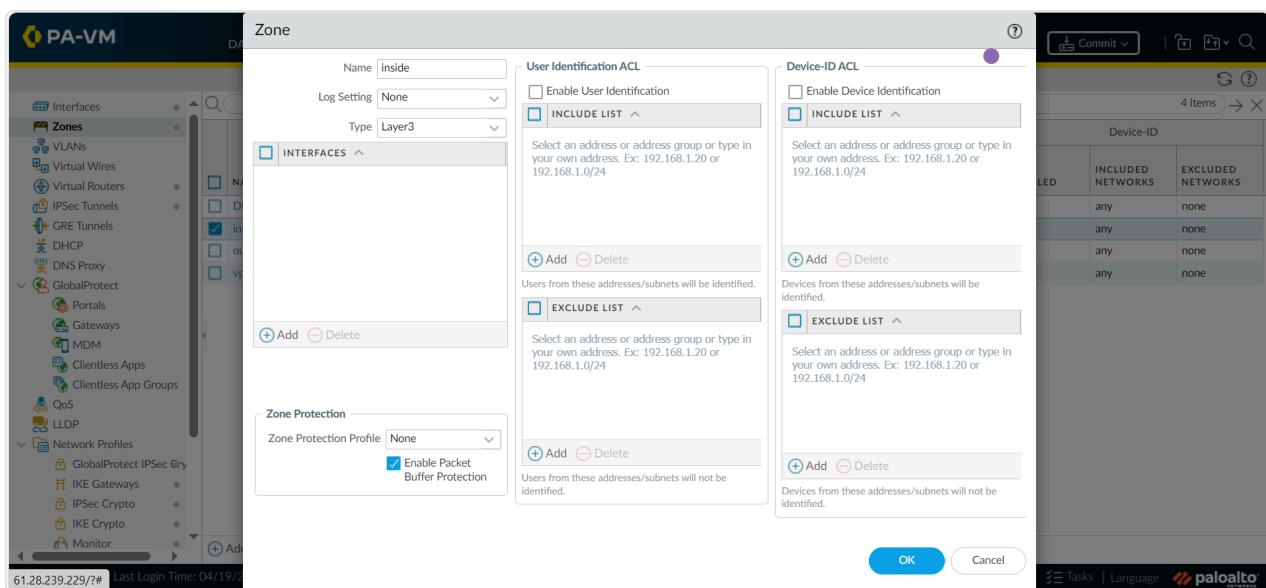
Step 3 : After logging in, you need to change your password for the first time. Please enter a new password according to your wishes.

Step 4: You need to create 1 Zone Inside and 1 Zone Outside according to the instructions below:

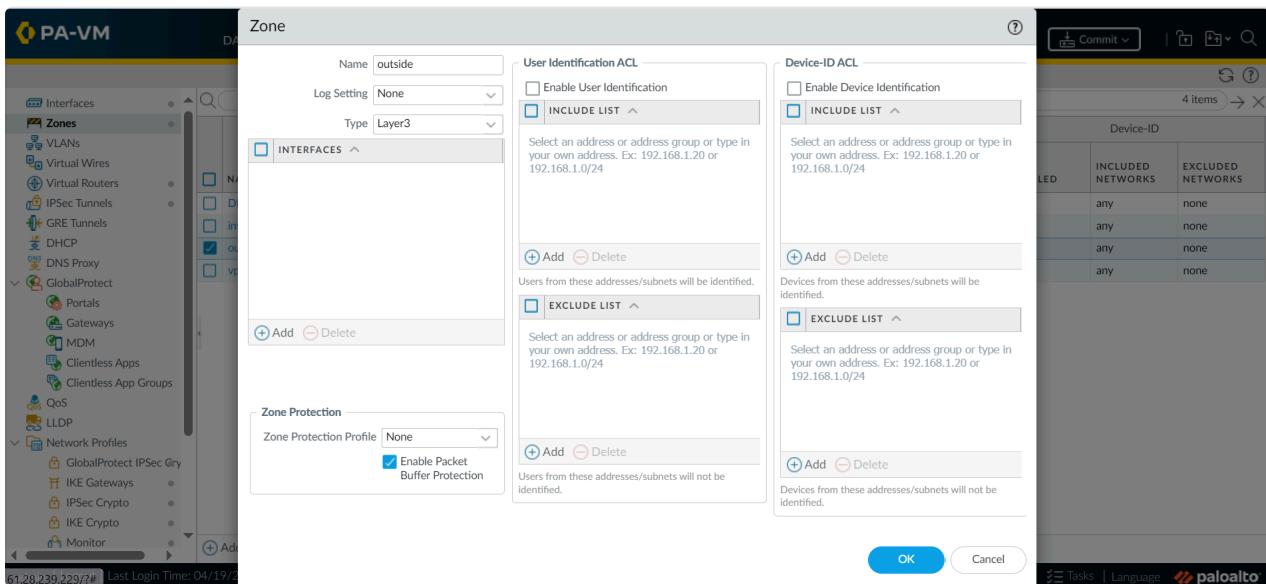
- Select **Add pen**

A screenshot of the Palo Alto Networks PA-VM interface. The left sidebar shows navigation categories: Interfaces, Zones, IPsec Tunnels, GlobalProtect, Portals, Gateways, MDM, Clientless Apps, Clientless App Groups, SD-WAN, and Network Profiles. The 'Network' tab is selected. On the right, there is a table titled 'Zones' with columns: NAME, TYPE, INTERFACES / VIRTUAL SYSTEMS, ZONE PROTECTION PROFILE, PACKET BUFFER PROTECTION, LOG SETTING, User-ID, Device-ID, INCLUDED NETWORKS, EXCLUDED NETWORKS, ENABLED, INCLUDED NETWORKS, and EXCLUDED NETWORKS. At the bottom left of the table area, there is a red box around the 'Add' button.

- Name the Zone : **Inside** then select **OK**

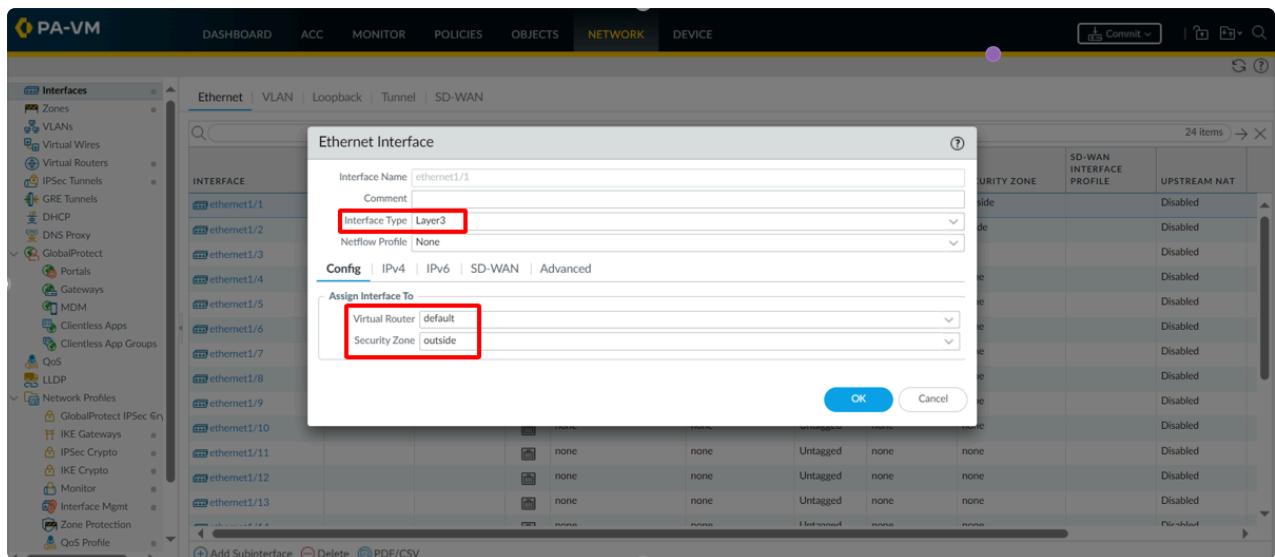


- Do the same for **Zone Outside**

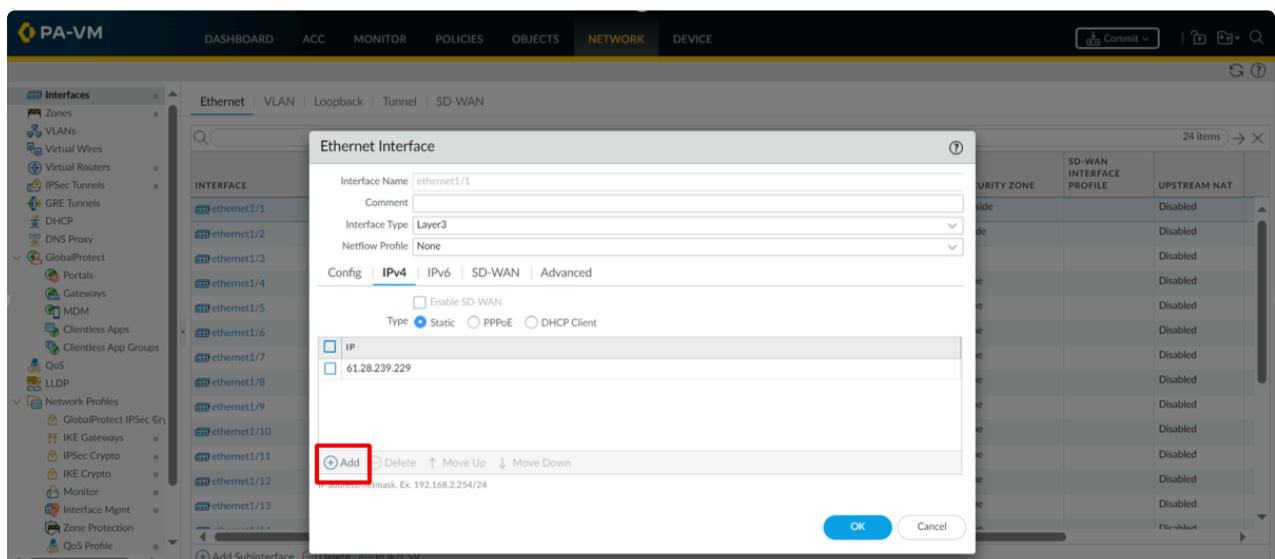


Step 5 : Configure External Interface

- Interface Type: **Layer 3**
- Virtual Router: **default**
- Security Zone: **Outside**



- Switch to **IPv4 Tab** and select **Add** to enter **Static IP** for **External Interface**



- To get this IP information, go to **Palo Alto's Network Interface** section to view the information

mp-Palo_Alto_PA-10_1_0 STOPPED
Created on 26/04/2024 09:44:08

ID	mp-ins-56d07f69-dcac-4669-8303-bbaa794e8c27	Memory	8 GB	SSH key	No
Instance type	s1-standard-2x8	CPU	2 Core(s)	Encryption volume	No
OS	Marketplace image Palo_Alto_PA-10.1.0	GPU	0	Server group	soft-anti-affinity
OS Licence	No				

Detail information
Details of your server while it is in use

Volume Log History Monitor Network Interface Security Associated snapshot Tag

INTERNAL INTERFACE

ID	VPC ID	Subnet ID	Fixed IP	Floating IP Association
net-in-fcc60993-3643-4447-a8f6-e19ff622dbc6	net-a4aeef856-7c16-4557-96c9-2851262dbefb	sub-e08ae230-4040-41d3-9d57-7b8e88fc302c	10.76.255.4	
net-in-fbf0a9bf-ec2f-4cc0-9169-6045a3befa7d	net-a4aeef856-7c16-4557-96c9-2851262dbefb	sub-3add2e0e-a8cb-4ccb-89de-5f77a8eb4581	10.76.0.4	

EXTERNAL INTERFACE
You can only attach one external network interface at the same time.

ID	VPC ID	Subnet ID	Fixed IP
net-in-beac1c8c-10aa-48eb-8ab4-905a3af3db4c	d1fa3c18-c904-41b9-bf50-d7b47cf7b2cd	marketplacePublicInterface	61.28.239.226

- Switch to the **Advanced** tab , in the **MTU** section you need to set it to **1400**

Interface Name: ethernet1/1

Comment:

Interface Type: Layer3

Netflow Profile: None

Config | IPv4 | IPv6 | SD-WAN | **Advanced**

Link Settings

Link Speed: auto	Link Duplex: auto	Link State: auto
------------------	-------------------	------------------

Other Info | ARP Entries | ND Entries | NDP Proxy | LLDP | DDNS

Management Profile: None

MTU: **1400**

Adjust TCP MSS

IPv4 MSS Adjustment: 40

IPv6 MSS Adjustment: 60

Untagged Subinterface

OK Cancel

Step 6: Perform similar configuration for Internal Interfaces

Ethernet Interface

Interface Name: ethernet1/2

Comment:

Interface Type: Layer3

Netflow Profile: None

Config | IPv4 | IPv6 | SD-WAN | Advanced

Assign Interface To

Virtual Router: default

Security Zone: inside

OK **Cancel**

- At the **IPv4 tab**: you proceed to set up **Static IP**

Ethernet Interface

Interface Name: ethernet1/2

Comment:

Interface Type: Layer3

Netflow Profile: None

Config | **IPv4** | IPv6 | SD-WAN | Advanced

Enable SD-WAN

Type: Static PPPoE DHCP Client

<input type="checkbox"/>	IP
<input type="checkbox"/>	10.76.0.4/24

+ Add **- Delete** **↑ Move Up** **↓ Move Down**

IP address/netmask. Ex. 192.168.2.254/24

OK **Cancel**

- Switch to the **Advanced** tab , in the **MTU** section , set it to 1400

Ethernet Interface

Interface Name: ethernet1/2

Comment:

Interface Type: Layer3

Netflow Profile: None

Config | IPv4 | IPv6 | SD-WAN | **Advanced**

Link Settings

Link Speed: auto | Link Duplex: auto | Link State: auto

Other Info | ARP Entries | ND Entries | NDP Proxy | LLDP | DDNS

Management Profile: None

MTU: **1400** (highlighted with a red box)

Adjust TCP MSS

IPv4 MSS Adjustment: 40 | IPv6 MSS Adjustment: 60

Untagged Subinterface

OK | **Cancel**

Step 7: Create static route

- Go to Network → Virtual Routers → Select default → Switch to Static Routes

PA-VM

DASHBOARD ACC MONITOR POLICIES OBJECTS NETWORK DEVICE

Virtual Router - default

Router Settings

Static Routes (highlighted with a red box)

Redistribution Profile

RIP | OSPF | OSPFv3 | BGP | Multicast

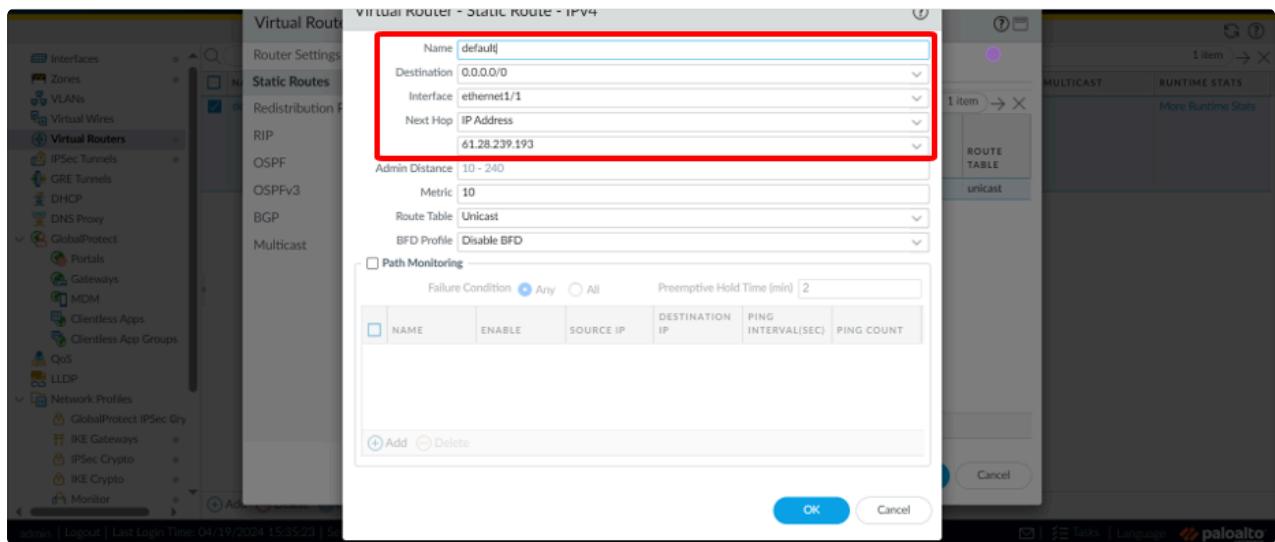
IPv4 | IPv6

Next Hop						
NAME	DESTINA...	INTERFACE	TYPE	VALUE	ADMIN DISTANCE	METRIC

Add | Delete | Clone

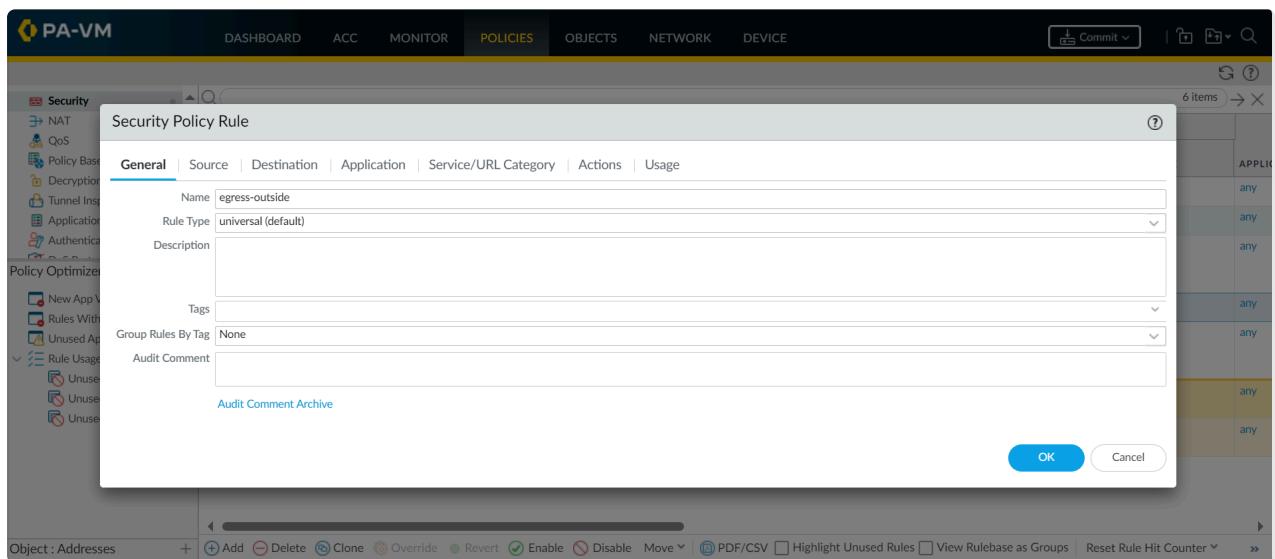
OK | Cancel

- Create a route as shown below

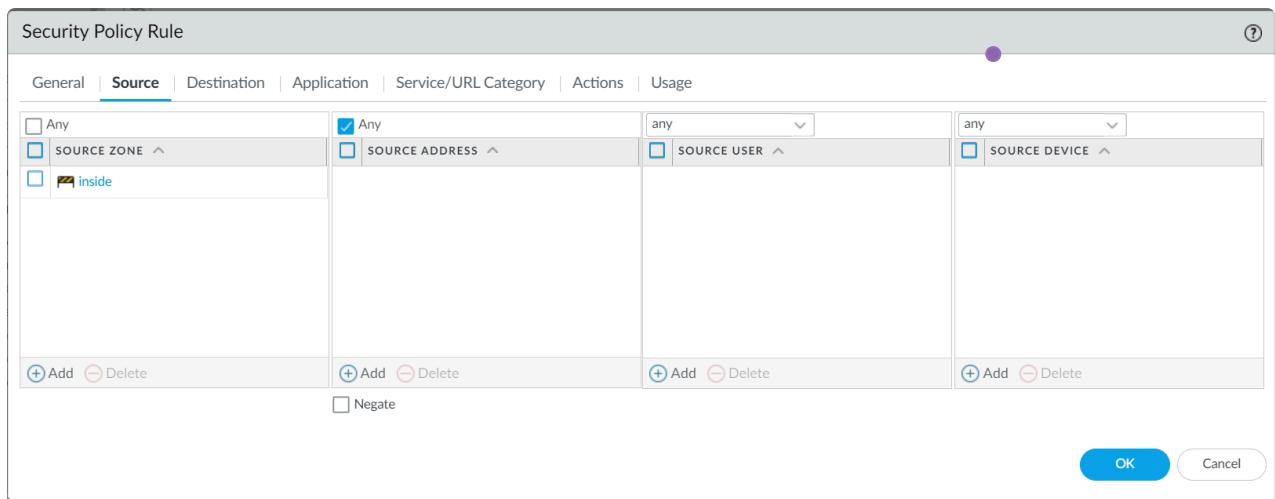


Step 8: Create Security Policy Rule

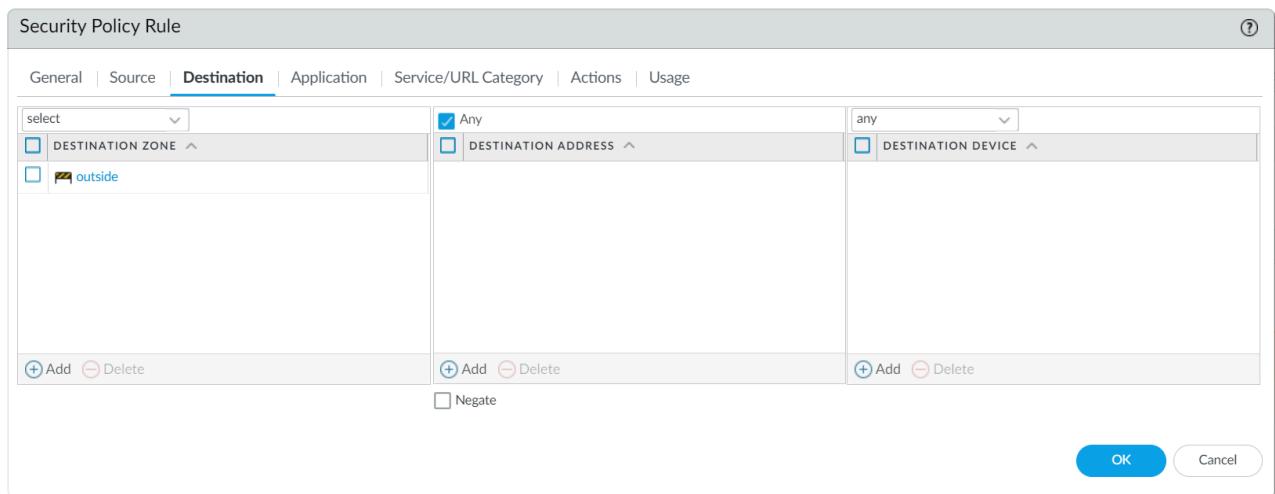
- Go to **Policies** → **Security** → **Add**
- On the **General** tab , you need to name the rule



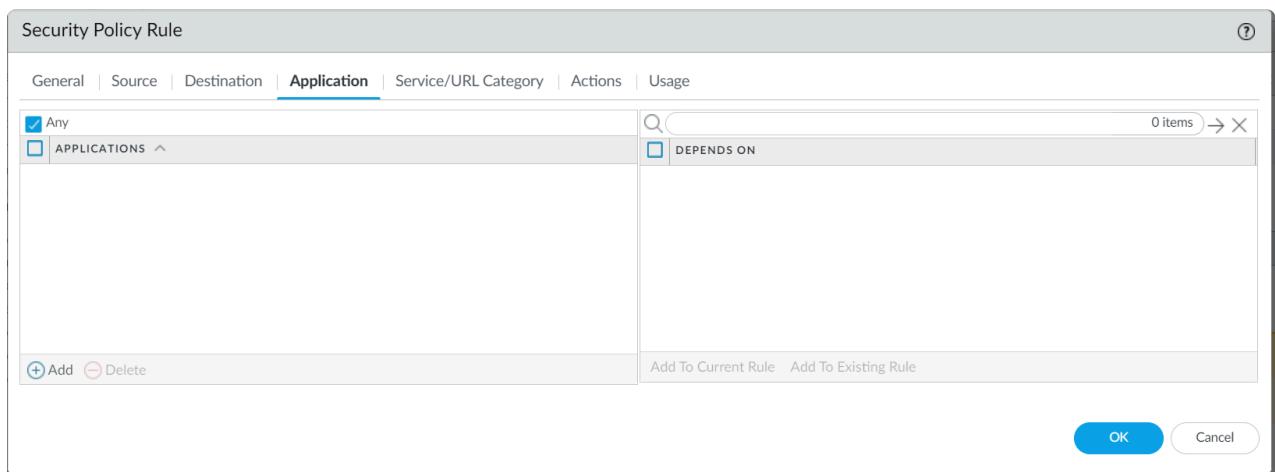
- At the **Source** tab , set information such as **Source Zone , Source Address , Source User, Source Device**



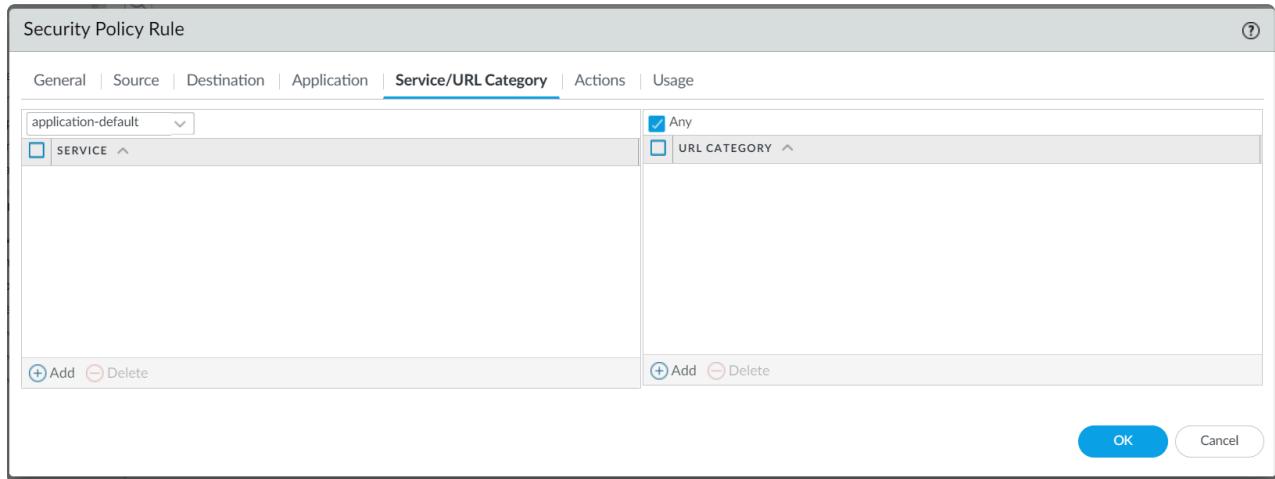
- At the **Destination** tab , set information such as **Destination Zone, Destination Address, Destination Device**



- At the **Application** tab , set information such as **Application, Depend On**



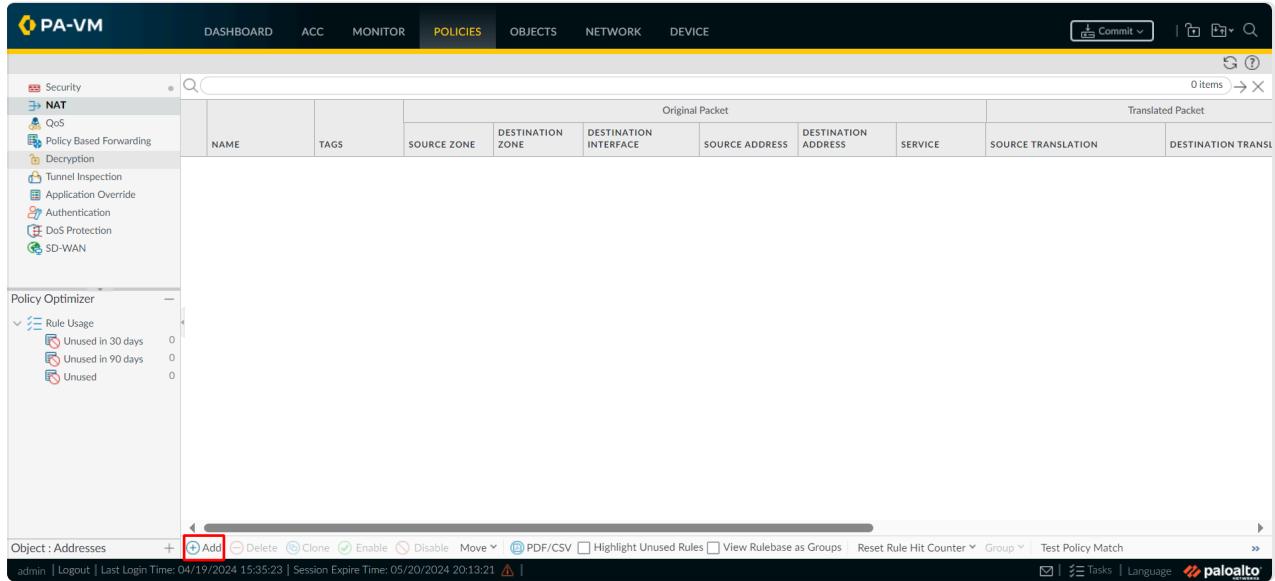
- At the **Service/URL Category** tab , set information such as **Service, URL Category**



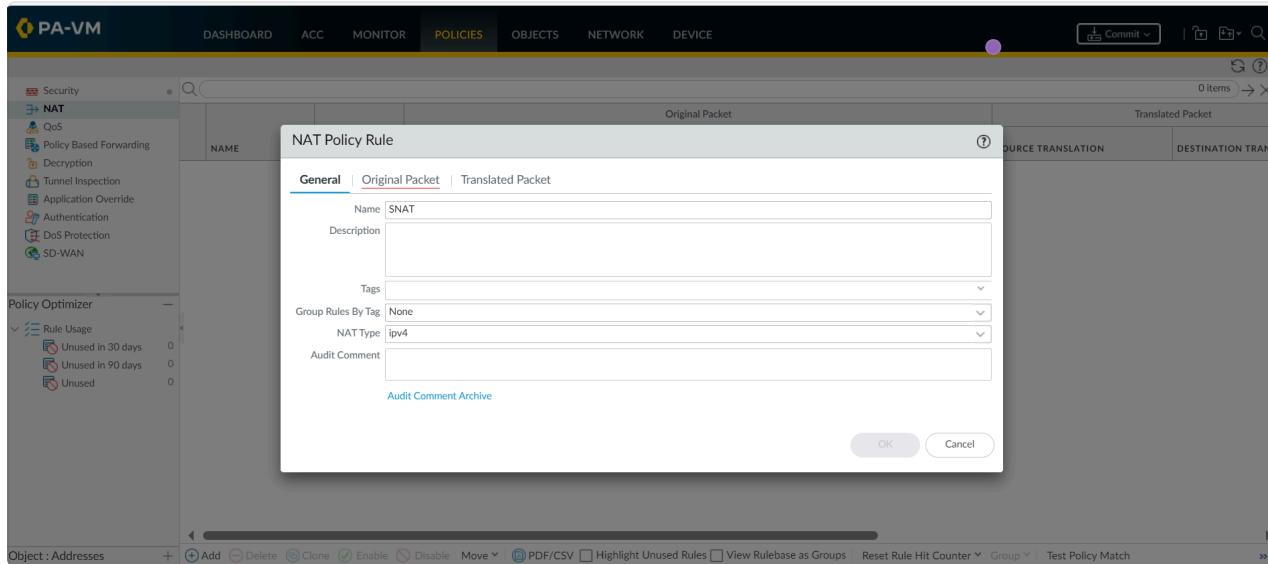
- At the **Actions** tab , set information such as **Action, Log, Profile, Other Settings**

Step 9 : Create a NAT rule so that VMs can go out to the Internet

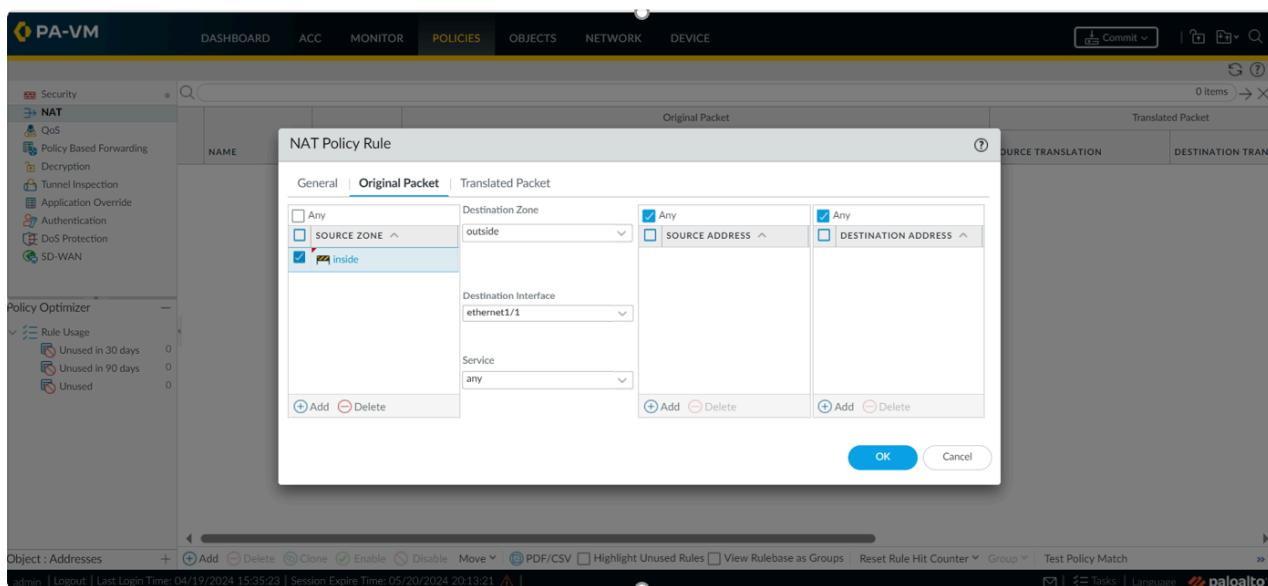
- Go to **Policies → NAT → Add**



- On the **General** tab , name **the NAT rule**

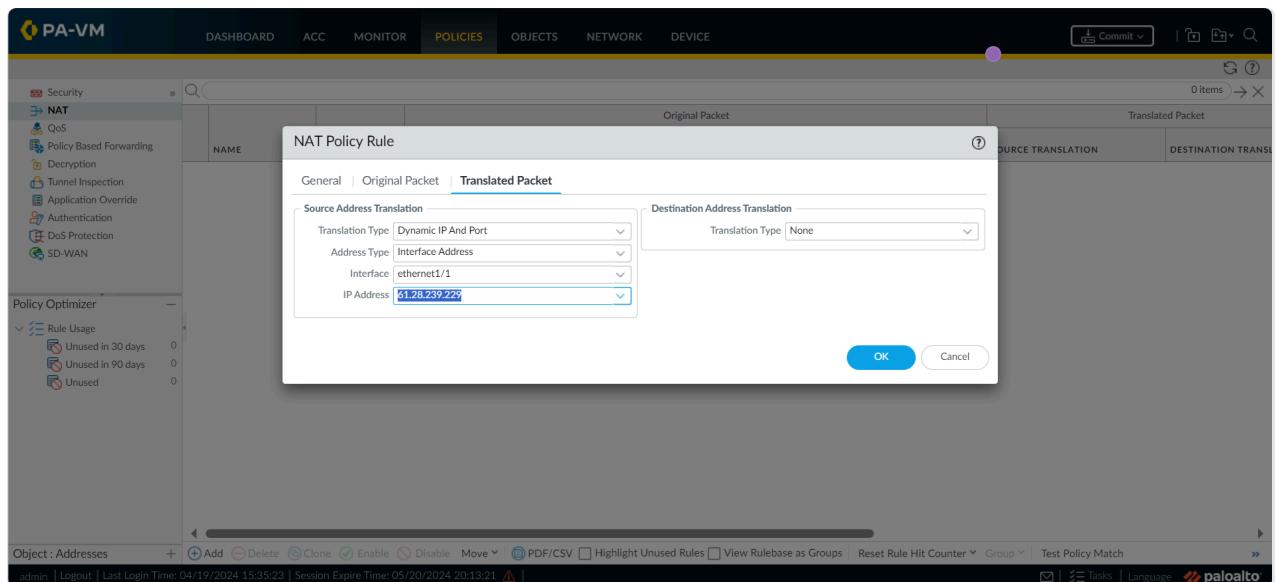


- At the **Original Packet** tab, select **Source Zone, Destination Zone, Destination Interface, Service, Source Address, Destination Address**

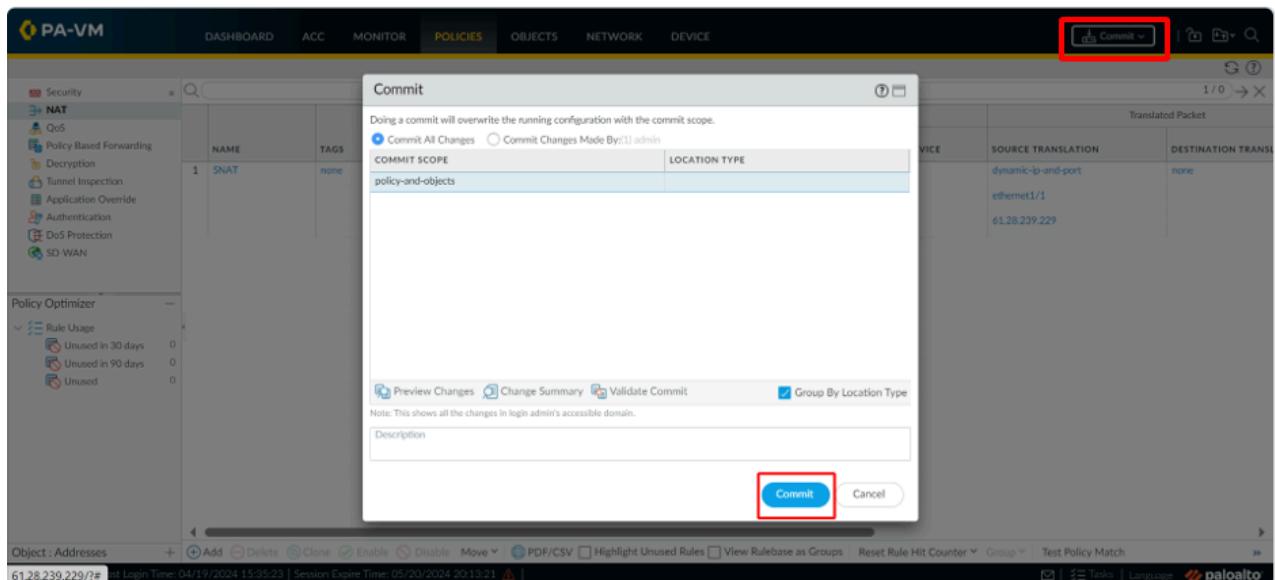


- Create the **Translated Packet** tab and perform configuration as shown below

Note: Need to change **the IP Address to the Static IP** address that you configured in step 6



Step 10 : Proceed to Commit



Initialize Route Table

After Palo Alto is successfully initialized and configured, you need to create a Route table to connect to different networks. Specifically, follow these steps to create a Route table:

Step 1: Visit <https://hcm-3.console.vngcloud.vn/vserver/network/route-table>

Step 2: In the navigation menu bar, select **Network Tab/ Route table**.

Step 3: Select **Create Route table**.

Step 4: Enter a descriptive name for the Route table. Route table names can include letters (az, AZ, 0-9, '_', '-'). The input data length is between 5 and 50. It must not include leading or trailing spaces.

Step 5: Select **VPC** for your Route table. If you do not have a VPC, you need to create a new VPC according to the instructions on [the VPC Page](#). **The VPC used to set up the Route table must be the VPC selected for your Palo Alto and Cluster.**

Step 6 : Select **Create** to create a new Route table.

Step 7: Select the newly created Route table then select **Edit Routes**.

Step 8: In the add new **Route** section , enter the following information:

- For Destination, enter **Destination CIDR as 0.0.0.0/0**
- For Target, enter **Target CIDR as the Palo Alto Network Interface 2 IP address.**

For example:

Name	VPC	Status	Created at	Action
rt-7f0108bb-8ab3-4e24-9e49-4ed513412229 phuctm3	net-a4aef856-7c16-4557-96c9-2851262dbefb	ACTIVE	24/04/2024 14:03:11	trash

List of Route attached to this Route Table.

Destination	Target
0.0.0.0/0	10.76.0.4

IP Network Interface 2

Checking connection

- Proceed to ping 8.8.8.8 or google.com

```
stackops@instance-4b946a04-0f:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=26.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=26.0 ms

64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=25.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=26.0 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=26.1 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=116 time=26.0 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=116 time=26.0 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=116 time=25.9 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=116 time=25.9 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=116 time=25.9 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=116 time=25.9 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=116 time=26.0 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=116 time=26.0 ms
64 bytes from 8.8.8.8: icmp_seq=14 ttl=116 time=25.9 ms
64 bytes from 8.8.8.8: icmp_seq=15 ttl=116 time=26.0 ms
64 bytes from 8.8.8.8: icmp_seq=16 ttl=116 time=25.9 ms
64 bytes from 8.8.8.8: icmp_seq=17 ttl=116 time=26.0 ms
64 bytes from 8.8.8.8: icmp_seq=18 ttl=116 time=25.9 ms
64 bytes from 8.8.8.8: icmp_seq=19 ttl=116 time=26.1 ms
64 bytes from 8.8.8.8: icmp_seq=20 ttl=116 time=26.0 ms
64 bytes from 8.8.8.8: icmp_seq=21 ttl=116 time=26.0 ms
```

Pfsense as a NAT Gateway •

Use the instructions below to work with Private Node groups through Pfsense

Prerequisites

To be able to use Pfsense as NAT Gateway for Cluster on VKS system, you need:

- A **Pfsense server (VM)** is initialized on the **vMarketPlace** system according to the instructions below with the following configuration:

Item	Configuration
Flavor	2x4
Volume	80 GB
VPC	10.3.0.0/16
Network Interface 1	10.3.0.3

Initialize Pfsense

Step 1: Visit <https://marketplace.console.vngcloud.vn/>

Step 2: At the main screen, search for **Pfsense** , at **Pfsense** service , select **Launch**

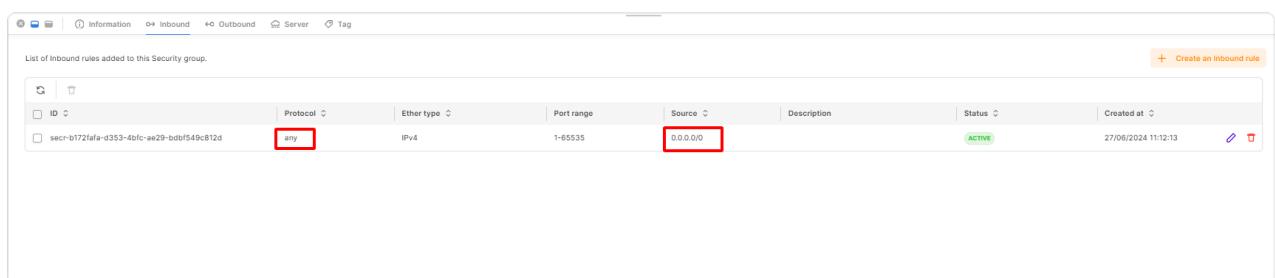
.

Step 3: Now, you need to configure **Pfsense**. Specifically, you can select the desired **Volume, IOPS, Network, Security Group** . You need to choose the same **VPC and Subnet as the VPC and Subnet you choose to use for your Cluster**. In addition, you also need to select an existing Server Group or select **Dedicated SOFT ANTI AFFINITY group** so we can automatically create a new server group.

Step 4: Proceed to pay like normal resources on VNG Cloud.

Configure parameters for PfSense

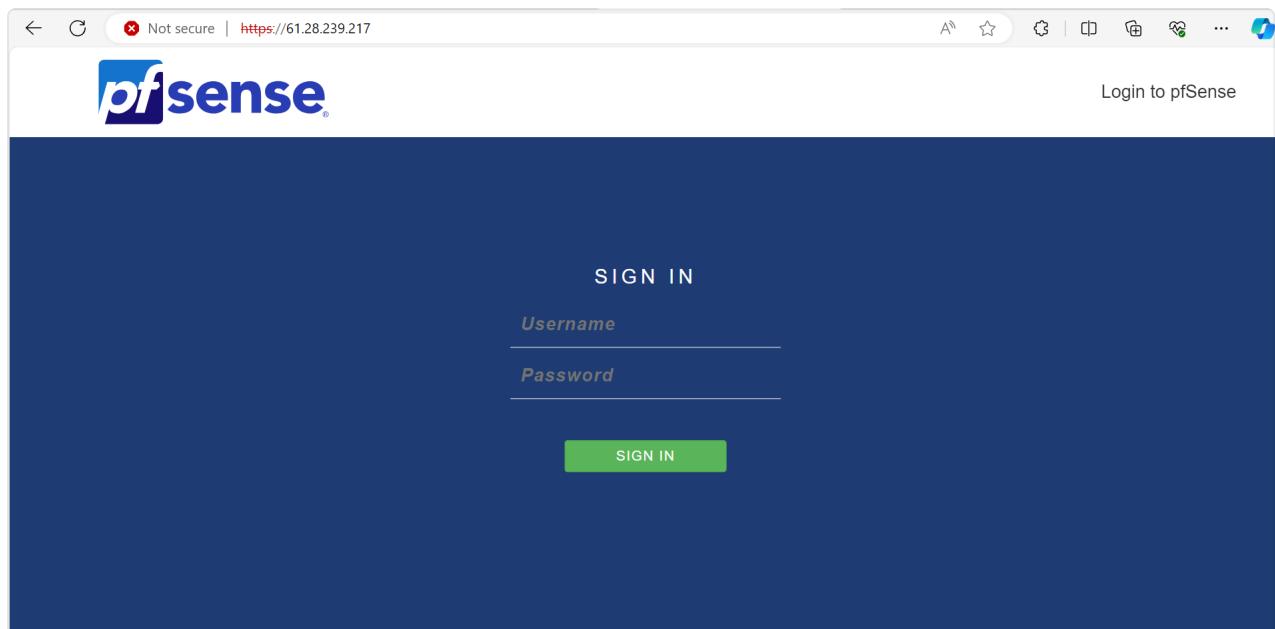
Step 1: After initializing PfSense from vMarketPlace according to the instructions above, you can access the vServer interface here [to](#) check whether the server running PfSense has been initialized. **Next, open the Any rule on the Security Group for the PfSense server you just created. Opening the Any rule on the Security Group will allow all traffic to the PfSense server.**



The screenshot shows the 'Inbound' rules configuration screen in the PfSense web interface. A single rule is listed:

ID	Protocol	Ether type	Port range	Source	Description	Status	Created at	Action
secr-b172faf-a-d353-4bf0-ae29-bdbf549cB12d	Any	IPv4	1-65535	0.0.0.0/0		ACTIVE	27/06/2024 11:12:13	

Step 2: After the server running PfSense is successfully initialized . To access the PfSense GUI, you need to use the IP address of the External Interface to log in with the default Username and password **admin/pfsense**.



The screenshot shows the PfSense login page. It features a blue header with the PfSense logo and a 'Login to pfSense' link. The main area is dark blue with a 'SIGN IN' button at the top. Below it are two input fields labeled 'Username' and 'Password', followed by a green 'SIGN IN' button.

- To get this IP information, go to **the Network Interface** section of **PfSense** to view the information

INTERNAL INTERFACE

ID	VPC ID	Subnet ID	Fixed IP	Floating IP Association
net-in-fcc60993-3643-4447-a8f6-e19ff622dbc6	net-a4aef856-7c16-4557-96c9-2851262dbebf	sub-e08ae230-4040-41d3-9d57-7b8e88fc302c	10.76.255.4	
net-in-fbf0a9bf-ec2f-4cc0-9169-6045a3befa7d	net-a4aef856-7c16-4557-96c9-2851262dbebf	sub-3add2e0e-a8cb-4cbb-89de-5f77a8eb4581	10.76.0.4	

EXTERNAL INTERFACE
You can only attach one external network interface at the same time.

ID	VPC ID	Subnet ID	Fixed IP
net-in-beac1c8c-10aa-48eb-8ab4-905a3af3db4c	d1fa3c18-c904-41b9-bf50-d7b47cf7b2cd	marketplacePublicInterface	61.28.239.226

Step 3 : Open the rule on the firewall

- Proceed to **Add rule**

Rules (Drag to Change Order)

ID	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓	0/1.14 MiB	*	*	*	WAN Address	443 80	*	*		Anti-Lockout Rule	
✗	0/0 B	*	Reserved Not assigned by IANA	*	*	*	*	*		Block bogon networks	

No rules are currently defined for this interface
All incoming connections on this interface will be blocked until pass rules are added. Click the button to add a new rule.

Actions: Add Delete Toggle Copy Save

- You can open the rule as below to access the GUI using **External Interface** .

Attention:

- You should limit the IP Range allowed to connect to the Pfsense GUI to limit users allowed to access the Pfsense GUI.

Firewall / Rules / Edit

Edit Firewall Rule

Action	Pass
Choose what to do with packets that match the criteria specified below. Hint: the difference between block and reject is that with reject, a packet (TCP RST or ICMP port unreachable for UDP) is returned to the sender, whereas with block the packet is dropped silently. In either case, the original packet is discarded.	
Disabled	<input type="checkbox"/> Disable this rule Set this option to disable this rule without removing it from the list.
Interface	WAN
Choose the interface from which packets must come to match this rule.	
Address Family	IPv4
Select the Internet Protocol version this rule applies to.	
Protocol	Any
Choose which IP protocol this rule should match.	
Source	
Source	<input type="checkbox"/> Invert match any Source Address /
Destination	
Destination	<input type="checkbox"/> Invert match any Destination Address /
Extra Options	
Log	<input type="checkbox"/> Log packets that are handled by this rule Hint: the firewall has limited local log space. Don't turn on logging for everything. If doing a lot of logging, consider using a remote syslog server (see the Status: System Logs: Settings page).
Description	

- Select **Save**
- Then select **Apply changes**

Step 4 : Proceed with **General Setup** , please do as below

pfSense COMMUNITY EDITION

System ▾ Interfaces ▾ Firewall ▾ Services ▾ VPN ▾ Status ▾ Diagnostics ▾ Help ▾

WARNING: The 'admin' account password is set to the default value. Change the password in the User Manager.

Wizard / pfSense Setup /

pfSense Setup

Welcome to pfSense® software!
This wizard will provide guidance through the initial configuration of pfSense.
The wizard may be stopped at any time by clicking the logo image at the top of the screen.
pfSense® software is developed and maintained by Netgate®

[Learn more](#)

>> Next

pfSense is developed and maintained by Netgate. © ESF 2004 - 2024 [View license](#).

The screenshot shows the pfSense Setup Wizard at Step 1 of 9. The title bar says "Wizard / pfSense Setup / Netgate® Global Support is available 24/7". A red warning message at the top states: "WARNING: The 'admin' account password is set to the default value. Change the password in the User Manager." Below this, a section titled "Netgate® Global Support is available 24/7" contains text about the support team's qualifications and a list of reasons why companies choose Netgate support. A "Learn more" button is present, and a "» Next" button is at the bottom.

The screenshot shows the pfSense Setup Wizard at Step 3 of 9. The title bar says "Wizard / pfSense Setup / Time Server Information". A red warning message at the top states: "WARNING: The 'admin' account password is set to the default value. Change the password in the User Manager." Below this, a section titled "Time Server Information" asks for time, date and time zone information. It includes fields for "Time server hostname" (set to "2.pfsense.pool.ntp.org") and "Timezone" (set to "Asia/Ho_Chi_Minh"). A "» Next" button is at the bottom. A footer note at the bottom of the page reads: "pfSense is developed and maintained by Netgate. © ESF 2004 - 2024 [View license](#)".

On this screen the general pfSense parameters will be set.

Hostname	pfSense
Name of the firewall host, without domain part.	
Examples: pfsense, firewall, edgefw	
Domain	home.arpa
Domain name for the firewall.	
Examples: home.arpa, example.com	
Do not end the domain name with '.local' as the final part (Top Level Domain, TLD). The 'local' TLD is widely used by mDNS (e.g. Avahi, Bonjour, Rendezvous, Airprint, Airplay) and some Windows systems and networked devices. These will not network correctly if the router uses 'local' as its TLD. Alternatives such as 'home.arpa', 'local.lan', or 'mylocal' are safe.	
The default behavior of the DNS Resolver will ignore manually configured DNS servers for client queries and query root DNS servers directly. To use the manually configured DNS servers below for client queries, visit Services > DNS Resolver and enable DNS Query Forwarding after completing the wizard.	
Primary DNS Server	8.8.8.8
Secondary DNS Server	8.8.4.4
Override DNS	<input checked="" type="checkbox"/>
Allow DNS servers to be overridden by DHCP/PPP on WAN	
>> Next	

- Configure WAN Interface

Wizard / pfSense Setup / Configure WAN Interface

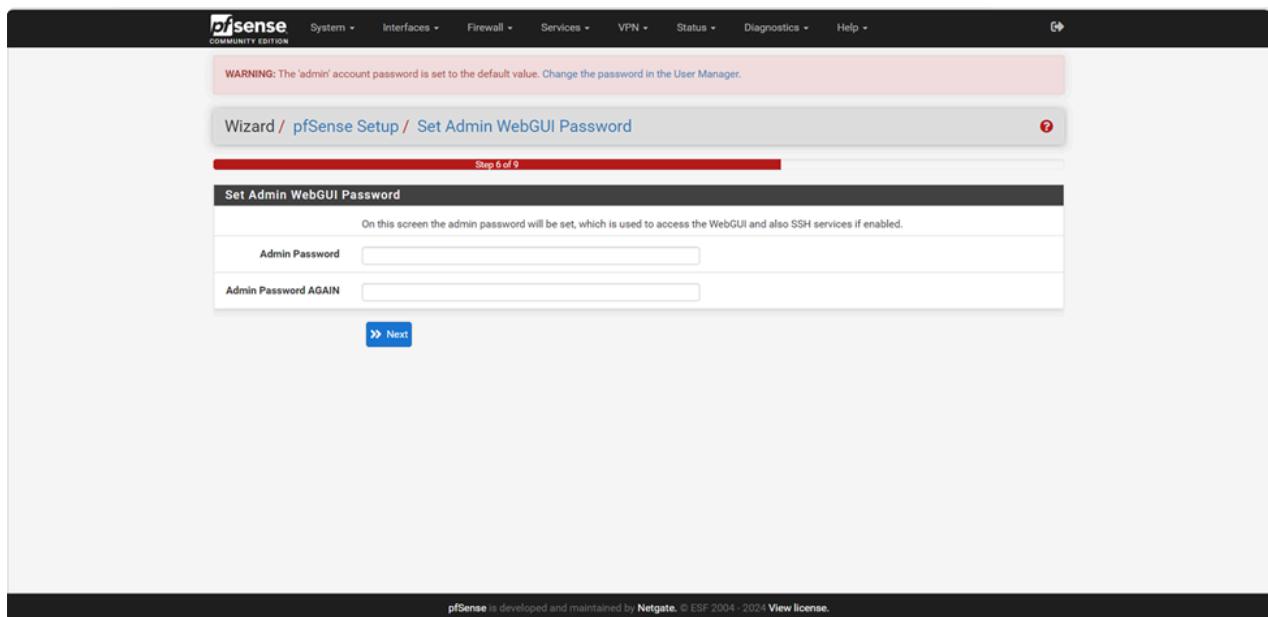
Step 4 of 9

Configure WAN Interface

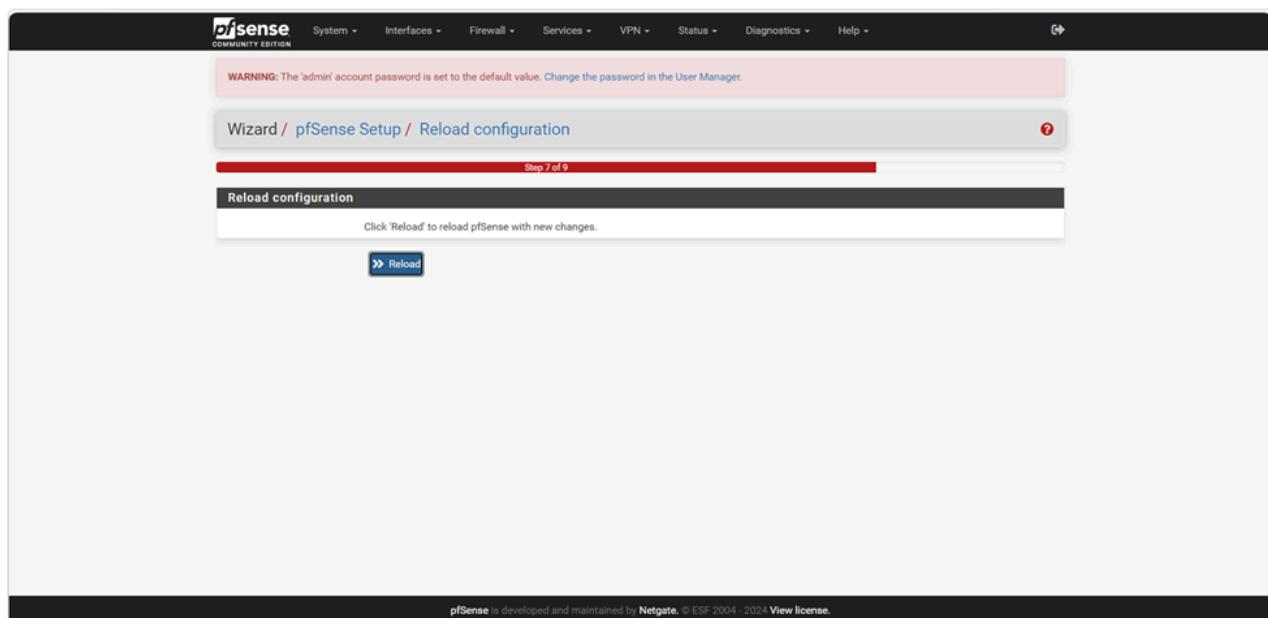
On this screen the Wide Area Network information will be configured.

SelectedType	Static
General configuration	
MAC Address	
This field can be used to modify ("spoof") the MAC address of the WAN interface (may be required with some cable connections). in the following format: xx:xxxx:xxxx:xxxx or leave blank.	
MTU	1400
If this field is blank, the adapter's default MTU will be used. This is typically 1500 bytes but can vary in some circumstances. assumed.	
MSS	
If a value is entered in this field, then MSS clamping for TCP connections to the value entered above minus 40 (TCP/IP header size) this field is left blank, an MSS of 1492 bytes for PPPoE and 1500 bytes for all other connection types will be assumed. This should MTU value in most all cases.	
Static IP Configuration	
IP Address	61.28.239.217
Subnet Mask	26
Upstream Gateway	61.28.239.193
DHCP client configuration	
DHCP Hostname	

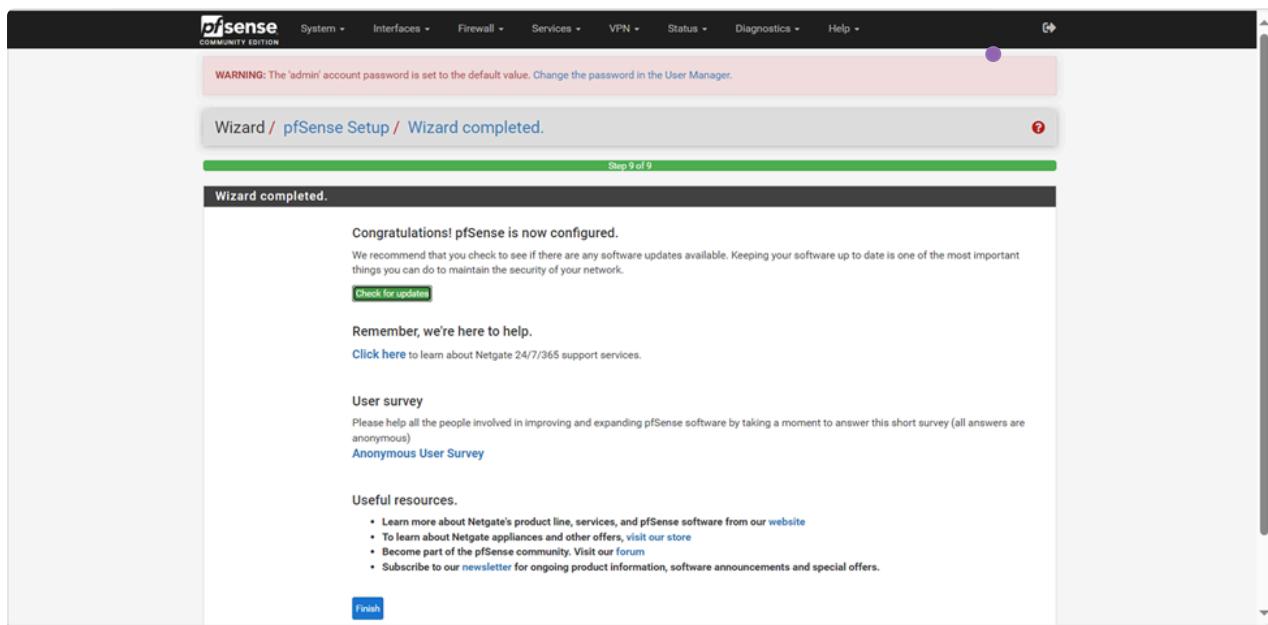
- Change password in GUI



- Proceed to reload



- General Setup completed



Step 5: Configure LAN Interface

- Go to **Interfaces → Assignments** to add a LAN Interface

The screenshot shows the pfSense Status/Dashboard page. The navigation bar has "Interfaces" selected, with "Assignments" highlighted by a red box. A pink banner at the top reads: "WARNING: The 'admin' account password is set to the default value. Change the password in the User Manager." The main content area includes sections for "System Information" (with tabs for WAN and LAN), "Netgate Services And Support" (with tabs for Community Support and Contract type), and "NETGATE AND pfSense COMMUNITY SUPPORT RESOURCES". A note at the bottom right says: "If you decide to purchase a Netgate Global TAC Support subscription, you MUST have your Netgate Device ID (NDI) from your firewall in order to validate support for this unit. Write down your NDI and store it in a safe place. You can purchase TAC supports [here](#)".

- Click Add

WARNING: The 'admin' account password is set to the default value. Change the password in the User Manager.

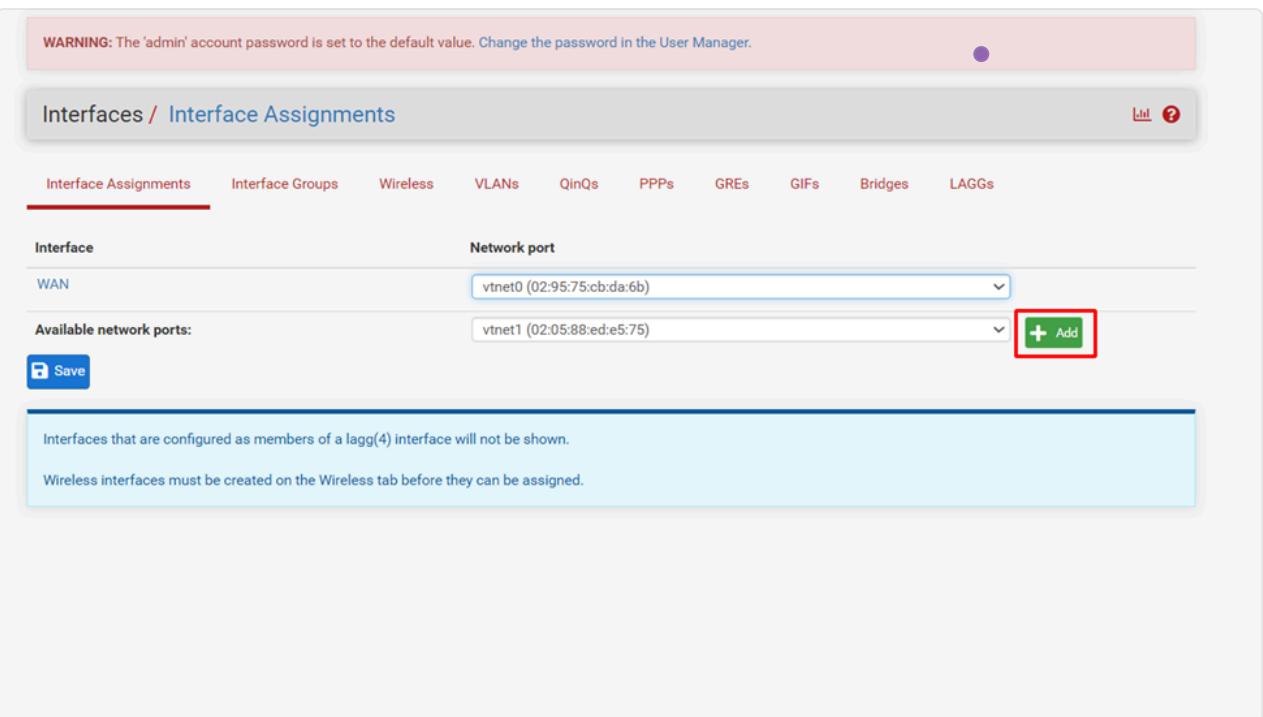
Interfaces / Interface Assignments

Interface Assignments Interface Groups Wireless VLANs QinQs PPPs GREs GIGe Bridges LAGGs

Interface	Network port
WAN	vtnet0 (02:95:75:cb:da:6b)
Available network ports:	vtnet1 (02:05:88:ed:e5:75)

Save + Add

Interfaces that are configured as members of a lagg(4) interface will not be shown.
Wireless interfaces must be created on the Wireless tab before they can be assigned.



- Then click **Save**

WARNING: The 'admin' account password is set to the default value. Change the password in the User Manager.

Interfaces / Interface Assignments

Interface has been added.

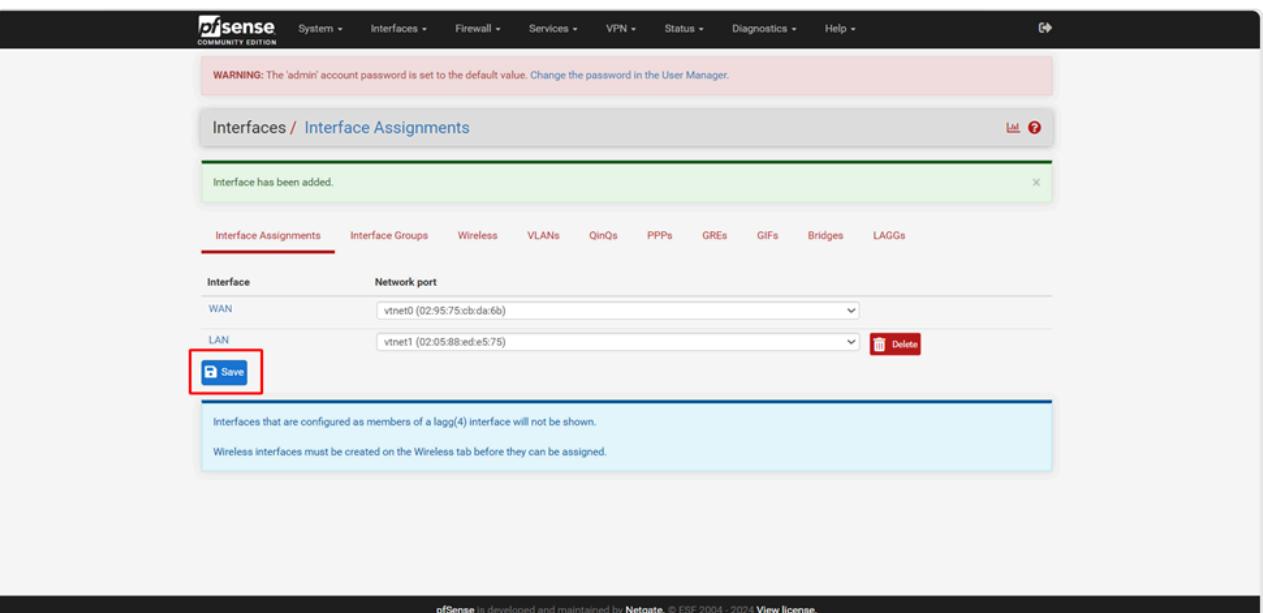
Interface Assignments Interface Groups Wireless VLANs QinQs PPPs GREs GIGe Bridges LAGGs

Interface	Network port
WAN	vtnet0 (02:95:75:cb:da:6b)
LAN	vtnet1 (02:05:88:ed:e5:75)

Save Delete

Interfaces that are configured as members of a lagg(4) interface will not be shown.
Wireless interfaces must be created on the Wireless tab before they can be assigned.

pfSense is developed and maintained by Netgate. © ESF 2004 - 2024 [View license](#).



- Go to **Interfaces → Assignments** to enable LAN Interface

- You make the configuration as below

- Configure IP for LAN

- Then proceed to **Add a new gateway**: enter **Gateway for LAN Interface**

New IPv4 Gateway

<input checked="" type="checkbox"/> Default	<input type="checkbox"/> Default gateway
Gateway name	LANGW
Gateway IPv4	<input type="text" value="10.3.0.1"/>
Description	
<input type="button" value="Add"/> <input type="button" value="Cancel"/>	

- To get this IP information, go to the **Network Interface** section of the PfSense server to view the information:

Detail information

INTERNAL INTERFACE

ID	VPC ID	Subnet ID	Fixed IP	Floating IP Association
net-in-fcc80093-3643-4447-a8f6-e10ff822dbc6	net-a4aeef856-7c16-4557-96c9-2851262dbefb	sub-e08ae230-4040-41d3-9d57-7b8e88fc302c	10.76.255.4	
net-in-fbf0a9bf-ec2f-4cc0-9169-6045a3befa7d	net-a4aeef856-7c16-4557-96c9-2851262dbefb	sub-3add2e0e-abcb-4ccb-89de-5f77a8eb4581	10.76.0.4	

- Proceed to **Save** again

Static IPv4 Configuration

IPv4 Address	<input type="text" value="10.3.0.3"/>	/ 24
IPv4 Upstream gateway	<input type="text" value="LANGW - 10.3.0.1"/>	<input type="button" value="Add a new gateway"/>
<small>If this interface is an Internet connection, select an existing Gateway from the list or add a new one using the "Add" button. On local area network interfaces the upstream gateway should be "none". Selecting an upstream gateway causes the firewall to treat this interface as a WAN type interface. Gateways can be managed by clicking here.</small>		

Step 6 : Review configuration information

The screenshot shows the pfSense Status / Dashboard interface. On the left, there's a 'System Information' panel with details like Name (pfSense.home.arpa), User (admin@116.110.40.116), System (KVM Guest, Netgate Device ID: 43b3ee4234d346b91a6e), BIOS (Vendor: SeaBIOS, Version: 1.11.0-2.el7, Release Date: Tue Apr 1 2014), Version (2.7.0-RELEASE (amd64) built on Wed Jun 28 03:53:34 UTC 2023, FreeBSD 14.0-CURRENT), CPU Type (Intel(R) Xeon(R) Silver 4310 CPU @ 2.10GHz, AES-NI CPU Crypto: Yes (inactive), QAT Crypto: No), Hardware crypto (Inactive), Kernel PTI (Disabled), MDS Mitigation (Inactive), Uptime (00 Hour 05 Minutes 35 Seconds), Current date/time (Sun Apr 21 12:40:53 +07 2024), and DNS server(s) (127.0.0.1, 8.8.8, 8.8.4.4). A 'Last config change' entry is also present. On the right, there's a 'Netgate Services And Support' panel with contract type (Community Support, Community Support Only) and a section on NETGATE AND pfSense COMMUNITY SUPPORT RESOURCES. It includes links for upgrading support, Netgate Global Support FAQ, official pfSense training, professional services, and visiting Netgate.com. A note about purchasing a TAC support subscription is also shown.

Step 7 : Open the Internet outbound rule for the LAN interface

The screenshot shows the pfSense Firewall / Rules / LAN configuration page. The LAN tab is selected. The rules table lists three entries: an Anti-Lockout Rule (0/0 B, LAN Address, 443, 80, *), a Default allow LAN to any rule (0/76 B, LAN net, *, *, none, *), and a Default allow LAN IPv6 to any rule (0/0 B, LAN net, *, *, none, *). At the bottom, there are buttons for Add, Delete, Toggle, Copy, Save, and Separator. The 'Add' button is highlighted with a red box.

- At source, select the IP range that is allowed to go out to the Internet

Firewall / Rules / Edit

Edit Firewall Rule

Action	Pass
Choose what to do with packets that match the criteria specified below. Hint: the difference between block and reject is that with reject, a packet (TCP RST or ICMP port unreachable for UDP) is returned to the sender, whereas with block the packet is dropped silently. In either case, the original packet is discarded.	
Disabled	<input type="checkbox"/> Disable this rule Set this option to disable this rule without removing it from the list.
Interface	LAN
Choose the interface from which packets must come to match this rule.	
Address Family	IPv4
Select the Internet Protocol version this rule applies to.	
Protocol	Any
Choose which IP protocol this rule should match.	
Source	
Source	<input type="checkbox"/> Invert match Network 10.3.1.0 / 24
Destination	
Destination	<input type="checkbox"/> Invert match any Destination Address
Extra Options	
Log	<input type="checkbox"/> Log packets that are handled by this rule Hint: the firewall has limited local log space. Don't turn on logging for everything. If doing a lot of logging, consider using a remote syslog server (see the Status: System Logs: Settings page).
Description	

Step 8: Configure NAT so that vServers can go out to the Internet

- Go to Firewall → NAT

The screenshot shows the pfSense web interface. The top navigation bar includes links for System, Interfaces, Firewall (selected), Services, VPN, Status, Diagnostics, and Help. A warning message states: "WARNING: The 'admin' account password is set to 'admin'. Please change the password in the User Manager." Below the navigation is a sidebar with "Status / Dashboard" and a "System Information" table. The "System Information" table contains the following details:

Name	pfSense.home.arpa
User	admin@116.110.40.116 (Local Database)
System	KVM Guest Netgate Device ID: 43b3ee4234d346b91a6e
BIOS	Vendor: SeaBIOS Version: 1.11.0-2.e17 Release Date: Tue Apr 1 2014
Version	2.7.0-RELEASE (amd64) built on Wed Jun 28 03:53:34 UTC 2023 FreeBSD 14.0-CURRENT
CPU Type	Intel(R) Xeon(R) Silver 4310 CPU @ 2.10GHz AES-NI CPU Crypto: Yes (inactive) QAT Crypto: No
Hardware crypto	Inactive
Kernel PTI	Disabled
MDS Mitigation	Inactive
Uptime	00 Hour 08 Minutes 25 Seconds
Current date/time	Sun Apr 21 12:43:43 +07 2024

The main content area shows the "NAT" tab selected in a dropdown menu. To the right, there's a "Netgate Services And Support" section with a "Community Support" contract type. It also features a "NETGATE AND pfSense COMMUNITY SUPPORT RESOURCES" section with links to upgrade support, community resources, and professional services. A note at the bottom encourages purchasing TAC support.

- Select NAT mode then proceed to configure NAT

Outbound NAT Mode

Mode

- Automatic outbound NAT rule generation. (IPsec passthrough included)
- Hybrid Outbound NAT rule generation. (Automatic Outbound NAT + rules below)
- Manual Outbound NAT rule generation. (AON - Advanced Outbound NAT)**
- Disable Outbound NAT rule generation. (No Outbound NAT rules)

Save

- Click **Add** to add the rule

Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description	Actions
WAN	127.0.0.0/8 ::1/128	*	*	500	WAN address	*	✓	Auto created rule for ISAKMP	
WAN	127.0.0.0/8 ::1/128	*	*	*	WAN address	*	✗	Auto created rule	
LAN	127.0.0.0/8 ::1/128	*	*	500	LAN address	*	✓	Auto created rule for ISAKMP	
LAN	127.0.0.0/8 ::1/128	*	*	*	LAN address	*	✗	Auto created rule	

Automatic Rules

- Select source , destination NAT

Edit Advanced Outbound NAT Entry

Disabled Disable this rule

Do not NAT Enabling this option will disable NAT for traffic matching this rule and stop processing Outbound NAT rules
In most cases this option is not required.

Interface WAN
The interface on which traffic is matched as it exits the firewall. In most cases this is "WAN" or another externally-connected interface.

Address Family IPv4+IPv6
Select the Internet Protocol version this rule applies to.

Protocol any
Choose which protocol this rule should match. In most cases "any" is specified.

Source Network 10.3.1.0 / 24
Source network for the outbound NAT mapping.

Destination Any Port or Range
Destination network for the outbound NAT mapping.

Not
Invert the sense of the destination match.

Initialize Route Table

After Pfsense is successfully initialized and configured, you need to create a Route table to connect to different networks. Specifically, follow these steps to create a Route table:

Step 1: Visit <https://hcm-3.console.vngcloud.vn/vserver/network/route-table>

Step 2: In the navigation menu bar, select **Network Tab/ Route table**.

Step 3: Select **Create Route table**.

Step 4: Enter a descriptive name for the Route table. Route table names can include letters (az, AZ, 0-9, '_', '-'). The input data length is between 5 and 50. It must not include leading or trailing spaces.

Step 5: Select **VPC** for your Route table. If you do not have a VPC, you need to create a new VPC according to the instructions on [the VPC Page](#). **The VPC used to set up the Route table must be the VPC selected for your Pfsense and Cluster.**

Step 6 : Select **Create** to create a new Route table.

Step 7: Select the newly created Route table then select **Edit Routes**.

Step 8: In the add new **Route** section , enter the following information:

- For Destination, enter **Destination CIDR as 0.0.0.0/0**
- For Target, enter **Target CIDR as the Pfsense Network Interface 2 IP address**.

For example:

Name	VPC	Status	Created at	Action
rt-552332bc-9af3-4cb8-9037-1d2e3e780e6a phuctm3	net-9235c741-0f87-41cf-b70d-8fb71ab550dd	ACTIVE	19/04/2024 14:31:10	Delete

List of Route attached to this Route Table.

Destination * Target *

0.0.0.0/0 172.24.0.3

Add a route

Cancel Save

Checking connection

Proceed to ping google.com or 8.8.8.8 to check

- Before **Enable NAT** the server could not access the internet

```
stackops@instance-cc789c5d-f1:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
stackops@instance-cc789c5d-f1:~$ ping 8.8.8.8
```

- After **configuring NAT**, ping 8.8.8.8 to check

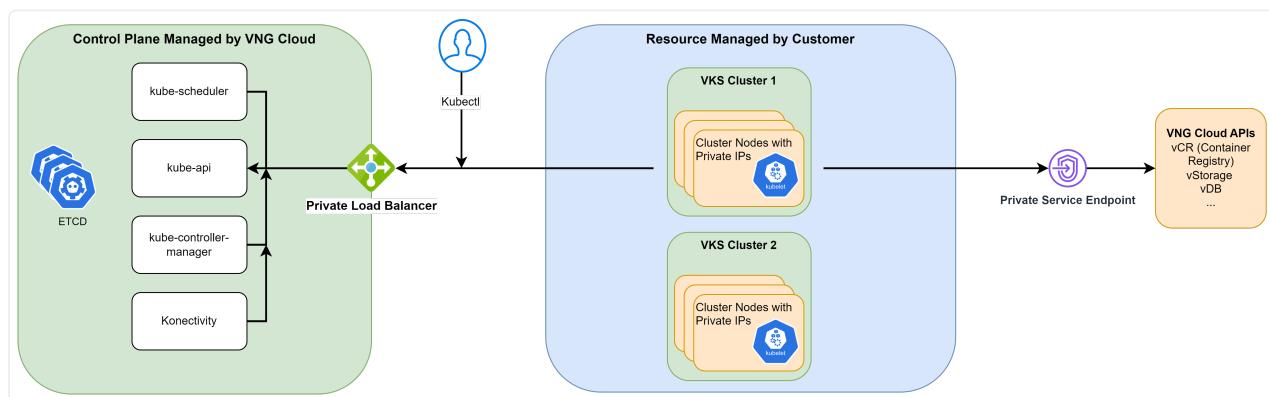
```
stackops@instance-cc789c5d-f1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:a5:8a:57:a6:89 brd ff:ff:ff:ff:ff:ff
        altname enp0s3
        inet 10.3.1.3/24 metric 100 brd 10.3.1.255 scope global ens3
            valid_lft forever preferred_lft forever
        inet6 fe80::a5:8aff:fe57:a689/64 scope link
            valid_lft forever preferred_lft forever
stackops@instance-cc789c5d-f1:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=27.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=26.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=26.7 ms
```

Create a Private Cluster

•

Previously, public clusters on VKS were using Public IP addresses to communicate between nodes and the control plane. To improve the security of your cluster, we have launched the private cluster model. The Private Cluster feature helps your K8S cluster to be as secure as possible, all connections are completely private from the connection between nodes to the control plane, the connection from the client to the control plane, or the connection from nodes to other products and services in VNG Cloud such as: vStorage, vCR, vMonitor, VNGCloud APIs,... Private Cluster is the ideal choice for services that require strict access control, ensuring compliance with regulations on security and data privacy.

Model



In which:

- **Control plane :** Managed by VNG Cloud, responsible for coordinating and managing the entire cluster.
- **Nodes :** When created, Nodes in the Cluster will only have internal IPs and cannot go to the public internet. If you want the node to access the internet, you need to use a NAT Gateway. For more details, refer [here](#) .
- **Private Load Balancer :** Managed by VNG Cloud, responsible for helping Private Nodes communicate with Control Plane.
- **Private Service Endpoint :** When you create a private cluster, the system automatically creates 4 endpoints to help connect to other services on VNG Cloud including:

- **Endpoint** to connect to the **IAM** service (Endpoint Name: vks-iam-endpoint-...)
- **Endpoint** to connect to **vCR** service (Endpoint Name: vks-vcr-endpoint-...)
- **Endpoint to connect to vServer** service (Endpoint Name: vks-vserver-endpoint-...)
- **Endpoint** to connect to **vStorage** service (Endpoint Name: vks-vstorage-endpoint-...)

You can view information about the 4 private service endpoints through the vServer portal by following the link [here](#).

Name	Status	Description	VPC	Endpoint IP	Service Name	Action
eng-fec284ec-5bf2-466e-ab04-8f6e29fc277 vks-vcr-endpoint-condescending-hamilton-2edb8 21/08/2024 15:58:55	ACTIVE	This is the service endpoint for vCR APIs established by the VKS product. Please DO NOT DELETE IT.	net-96ecfdf1-b390-4454-8126-3b0381c259f9 demo_vpc	10.7.8.6	vcr	⋮
eng-0d5d9e46-e735-4e07-94c7-bbf79ebf752d vks-vserver-endpoint-condescending-hamilton-5413b 21/08/2024 15:58:49	ACTIVE	This is the service endpoint for vServer APIs established by the VKS product. Please DO NOT DELETE IT.	net-96ecfdf1-b390-4454-8126-3b0381c259f9 demo_vpc	10.7.8.5	vserver.hcm03	⋮
eng-7c107f3-68a2-4332-bbec-ef806894deb9 vks-vstorage-endpoint-condescending-hamilton-71091 21/08/2024 15:58:45	ACTIVE	This is the service endpoint for vStorage APIs established by the VKS product. Please DO NOT DELETE IT.	net-96ecfdf1-b390-4454-8126-3b0381c259f9 demo_vpc	10.7.8.4	vstorage.hcm03	⋮
eng-1bc3e83a-6aa6-49ab-b48d-6012b64f99a4 vks-iam-endpoint-condescending-hamilton-cdc2d 21/08/2024 15:58:40	ACTIVE	This is the service endpoint for IAM APIs established by the VKS product. Please DO NOT DELETE IT.	net-96ecfdf1-b390-4454-8126-3b0381c259f9 demo_vpc	10.7.8.3	iam	⋮

Warning:

- **Do not delete Private Service Endpoints :** To ensure stable operation of the cluster, you should not delete the 4 pre-created service endpoints. If you accidentally delete or edit these 4 endpoints, within a maximum of 5 minutes, the system will automatically recreate them but may cause disruption to running services. At this time, because the recreated service endpoint may have changed the Endpoint IP compared to the original, in order for the cluster to work, you need to manually add Endpoint IP to the previously running servers via command:

```
vks-bootstraper add-host -i <IP> -d <DOMAIN>
```

Example, if you delete private service endpoint of vCR, you must add host via command:

```
vks-booster add-host -i 10.10.10.10 -d vcr.vngcloud.vn
```

- **Reuse Private Service Endpoints:** Service endpoints can be used by multiple private clusters. When private clusters share a VPC, we will reuse them for these clusters.
- **Delete Private Service Endpoints automatically:** When you delete a cluster, if there are no more clusters that reuse these service endpoints, the system will automatically delete them.
- **Cost of using Private Service Endpoint:** Using a private cluster will incur additional costs for 4 private service endpoints, but it brings many security benefits to your project. Please carefully consider the factors to decide whether to use public or private for your cluster.

Prerequisites

To be able to initialize a **Cluster** and **Deploy a Workload**, you need:

- There is at least 1 **VPC** and 1 **Subnet in ACTIVE state**. If you do not have a VPC or Subnet yet, please create a VPC and Subnet according to the instructions [here](#).
- There is at least 1 **SSH key in ACTIVE state**. If you do not have any SSH key, please create an SSH key according to the instructions [here](#).
- Installed and configured **kubectl** on your device. Please refer [here](#) if you are not sure how to install and use kubectl. In addition, you should not use a kubectl version that is too old, we recommend that you use a kubectl version that is no more than one version different from the cluster version.

Initialize Cluster

A cluster in Kubernetes is a collection of one or more virtual machines (VMs) connected together to run containerized applications. Cluster provides a unified environment to deploy, manage, and operate containers at scale.

To initialize a Cluster, follow the steps below:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select **Activate**.

Step 3: Wait until we successfully create your VKS account. After Activate successfully, select **Create a Cluster**

Step 4: At the Cluster initialization screen, we have set up information for the Cluster and a **Default Node Group** for you. You can keep these default values or adjust the desired parameters for the Cluster and Node Group at Cluster Configuration, Default Node Group Configuration, Plugin.

The screenshot shows the 'Create a Cluster' wizard on the VNG Cloud platform. The first step, 'Cluster configuration', is completed. In the 'Information' section, the 'Cluster Name' is 'demo-cluster' and the 'Kubernetes Version' is 'Version 1.29.1-vks.1'. The 'Description' field is empty. In the 'Network setting' section, the 'Network type' is 'Calico' and the 'IPIP encapsulation mode' is 'Always'. The 'CIDR' is '172.16.0.0/16'. The 'Private Cluster' option is selected, highlighted with a red box. The right sidebar displays 'Cluster configuration' details: Node group info (Cluster Name: demo-cluster, Kubernetes Version: Version 1.29.1-vks.1), Network settings (Access: Private, Network type: Calico, IPIP encapsulation mode: Always, CIDR: 172.16.0.0/16, VPC: demo_vpc (10.7.0.0/16), Subnet: demo_subnet (10.7.8.0/24)), and Default Node group configuration (Node group name: nodegroup-360f5, Number of nodes: 3). The total cost is listed as 4,356,000 VND (VAT included). A large orange button at the bottom right says 'CREATE KUBERNETES CLUSTER'.

Step 5: Select **Create Kubernetes cluster**. Please wait a few minutes for us to initialize your Cluster, the Cluster's status is now **Creating** .

Step 6: When the Cluster status is **Active** , you can view Cluster information and Node Group information by selecting Cluster Name in the **Name** column .



Warning:

- A Cluster can have **many Node Groups**, each Node Group can operate in Public/Private mode depending on your needs.
- Because your **Cluster** is initialized in **Private** mode, to be able to access Control Plane's **kube-api**, you need to be in the VPC you choose to use for your Cluster.

Connect and check the newly created Cluster information

After the Cluster is successfully initialized, you can connect and check the newly created Cluster information by following these steps:

Step 1: Visit <https://vks.console.vngcloud.vn/k8s-cluster>

Step 2: The Cluster list is displayed, select the **Download** icon and select **Download config file** to download the kubeconfig file. This file will give you full access to your Cluster.

Step 3 : Rename this file to config and save it to the **~/.kube/config** folder

Step 4: Because your Cluster is initialized in Private mode, to be able to access kube-api, you need to be in the VPC you have chosen to use for your Cluster. For example, when you are not in the VPC and execute get nodes, the results will display as follows:

```
kubectl get nodes
E0821 14:27:03.793829    23348 memcache.go:265] couldn't get current serve
E0821 14:27:05.866230    23348 memcache.go:265] couldn't get current serve
E0821 14:27:07.922272    23348 memcache.go:265] couldn't get current serve
E0821 14:27:09.989832    23348 memcache.go:265] couldn't get current serve
E0821 14:27:12.055864    23348 memcache.go:265] couldn't get current serve
Unable to connect to the server: dial tcp 10.7.8.9:6443: connectex: No co
```

In the example below I will stand at a server with a VPC along with the VPC used for the Cluster. You can perform SSH to the server according to instructions [here](#).

After SSH into the server, install kubectl according to the instructions [here](#).

- For example, I am using a linux server to perform get nodes, I can install kubectl via command:

```
sudo snap install kubectl --classic
```

- Then I tested kubectl via command:

```
kubectl version
```

- Create folder .`kube` via command:

```
mkdir -p .kube
```

- Then, enter the kubeconfig file via the command:

```
vim .kube/config
```

- Then, enter :wq to save the kubeconfig file and exit vim.
- Run the following command to test **the cluster**

```
kubectl get svc
```

- You should see a return similar to the following:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	10m

- Run the following command to test **node**

```
kubectl get nodes
```

- If the results are returned as below, it means your Cluster was successfully initialized with 1 node as below.

NAME	STATUS	ROLES	AGE	VERSION
vks-demo-cluster-nodegroup-demo-7c9aa	Ready	<none>	8m11s	v1.28.8

Use Docker to Pull/Push images

Because Private Cluster can only connect privately to the vContainer Registry (vCR) system and cannot connect to other Container Registry outside the internet, you need to pull/push the image to vCR to use according to the following instructions:

Step 1: Install Docker

- Perform docker installation according to instructions [here](#) .

Step 2: Initialize Public Repository and Repository User on vContainer Registry Portal:

- Log in to the vCR portal at the link: <https://vcr.console.vngcloud.vn/list>
- Perform Repository and Repository initialization according to instructions [here](#) .
For example in the image below, I have initialized demo_repo with demo_user who can pull/push images:

Repository
The Repository is a hub for docker images, enabling image management, including pull, push, and versioning.

Create a repository

Repositories 4 **Images** 4 **Quota used** 10.57 GB

Name	Access level	Image count	Quota Used/Limit	Created At	Action
repo-38997b70-663f-4965-b70c-dd4a5bb12eb5	PUBLIC	0	0 %	0 B / 20 GB	21/08/2024 16:55:21
53461-repo_demo	PUBLIC	1	0.33 %	67.73 MB / 20 GB	21/08/2024 16:28:35
19	PRIVATE	0	0 %	0 B / 10 GB	21/08/2024 13:15:24
19	PUBLIC	3	52.52 %	10.50 GB / 20 GB	30/07/2024 11:21:42

© 2022, VINA DATA IT JSC. Helios Building, Lot 3, Road #3, Quang Trung Software City. Tax code: 0304851362. Dept. of Planning and Investment of HCMC on 26/02/2007. support@vngcloud.vn - 1900 1549. Terms Privacy Policy SLA PCI DSS

Repository / Repository
repo-38997b70-663f-4965-b70c-dd4a5bb12eb5
53461-repo_demo

Quota used: 0 B / 20 GB

Images **Repository User** **History**

List of users attached to this repository.

User Name	Status	Permission	Description	Expired At	Action
53461-user_demo	ACTIVE	PULL AND PUSH		25/08/2024 16:56:02	

© 2022, VINA DATA IT JSC. Helios Building, Lot 3, Road #3, Quang Trung Software City. Tax code: 0304851362. Dept. of Planning and Investment of HCMC on 26/02/2007. support@vngcloud.vn - 1900 1549. Terms Privacy Policy SLA PCI DSS



Warning:

- If you want to create a Private Repository, to pull an image from the Private Repository, you need to create a secret key according to the instructions [here](#).

Step 3: Pull the nginx image according to the command:

```
docker pull nginx:latest
```

Step 4: Log in to vCR via command:

```
docker login vcr.vngcloud.vn -u <repository_user>
```

- For example, the command below I use to login to the demo repo:

```
docker login vcr.vngcloud.vn -u 53461-user_demo
```

Step 5: Assign tags to the nginx image

```
docker tag SOURCE_IMAGE[:TAG] vcr.vngcloud.vn/REPO_NAME/IMAGE[:TAG]
```

- For example, the command below I use to assign tags to the nginx image:

```
docker tag nginx:latest vcr.vngcloud.vn/53461-repo_demo/nginx-demo:latest
```

Step 6: Push the image to the repo via command:

```
docker push vcr.vngcloud.vn/REPO_NAME/IMAGE[:TAG]
```

- For example, the command below I use to push images to demo_repo:

```
docker push vcr.vngcloud.vn/53461-repo_demo/nginx-demo:latest
```

Deploy a Workload

The following are instructions for you to deploy the nginx service and expose this service via Network Load Balancer

Step 1: Create the nginx-service-lb4.yaml file via the command:

```
vi nginx.yaml
```

Then, enter the content for this file as follows: you need to replace the image with the image path saved on the vCR that you pushed in the step above:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: vcr.vngcloud.vn/53461-repo_demo/nginx-demo:latest
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- Enter :wq to save this file.
- Deploy This deployment equals:

```
kubectl apply -f nginx-service-lb4.yaml
```

Step 2: Check Deployment and Service information before exposing it to the Internet.

- Run the following command to test **Deployment**

```
kubectl get svc,deploy,pod -owide
```

- If the results are returned as below, it means you have successfully deployed the nginx service.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PO
service/kubernetes	ClusterIP	10.96.0.1	<none>	44
service/nginx-service	LoadBalancer	10.96.81.236	116.118.88.236	80
NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx-app	1/1	1	1	3m32s
NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-app-56bbc8fdd8-4pz68	1/1	Running	0	3m32s
				172

At this time, the vLB system will initialize a Network Load Balancer, you can view this LB information through the vLB portal [here](#).

The screenshot shows the VNG Cloud Load Balancer interface. On the left, there's a sidebar with navigation links like Overview, Limit, Network, VPCs, Floating IPs, etc. The main area has a title 'Load Balancer' with a sub-instruction: 'Load balancer is a service that distributes incoming traffic to multiple servers. It is a highly available service that can be used to distribute traffic to multiple servers in a single region or multiple regions.' Below this is a 'Create a Load Balancer' button. The main table lists six load balancers:

Name	Status	End point	Schema	Type	Package	AutoScale	Created at	Action
lb-9680258-9a83-478a-8dc4-864fd652127	ACTIVE	116.118.88.236	Internet	Network	NLB_Small	false	21/08/2024 17:05:36	⋮
vtk-k8s-5c0448-default-nginx-serv-3ab10								

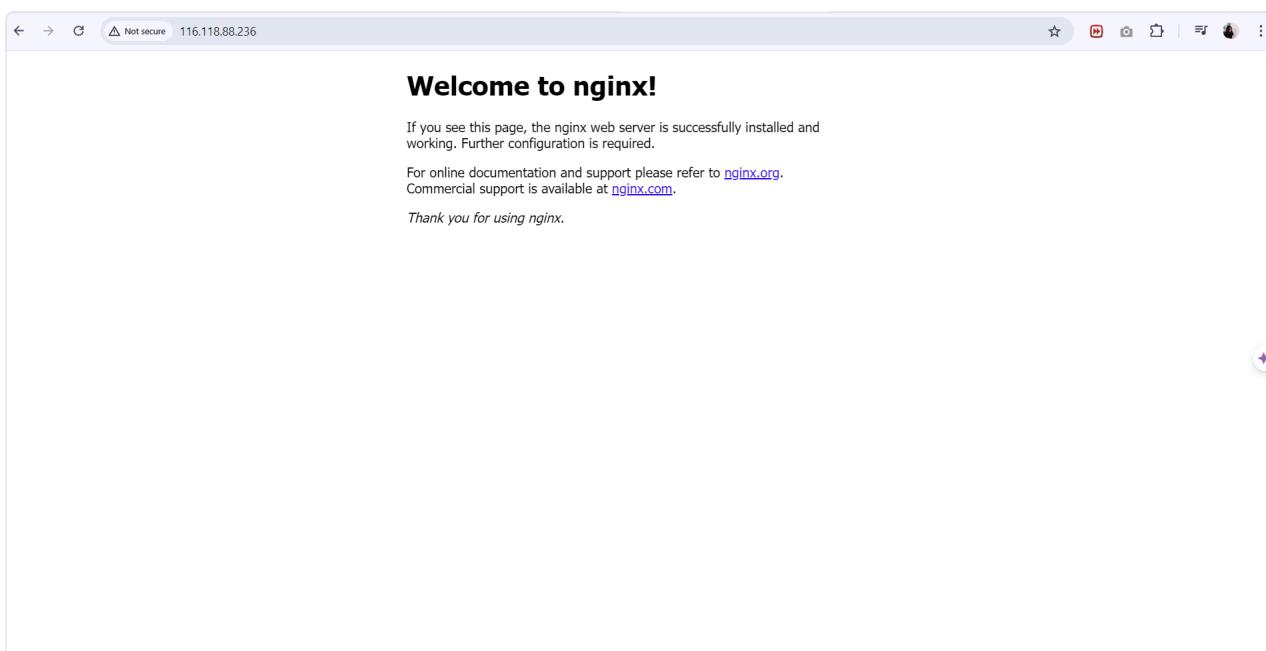
At the bottom, there are buttons for 'Total: 6', 'Show: 10', and a 'refresh' icon.

Step 3: To access the just exported nginx app, you can use the Endpoint of Load Balancer URL with the format:

```
http://Endpoint/
```

You can get Load Balancer Public Endpoint information at the vLB interface. Specifically, access at <https://hcm-3.console.vngcloud.vn/vserver/load-balancer/vlb/>

For example, below I have successfully accessed the nginx app with the address: <http://116.118.88.236/>



A few other notes:

- Above is an example showing you how to expose a service through vLB Layer 4. You can expose a service through vLB Layer 7 according to the instructions [here](#).
- To ensure the private cluster works effectively, we have automatically added the Subnet you choose to use for the Cluster to the cluster's Whitelist. You can use the Whitelist feature to limit the Subnets in the VPC that have access to kube-api. Details on how to use the Whitelist feature please refer [here](#).

Expose a service through vLB Layer4

Prerequisites

To be able to initialize a **Cluster** and **Deploy a Workload**, you need:

- There is at least 1 **VPC** and 1 **Subnet in ACTIVE state**. If you do not have a VPC or Subnet yet, please create a VPC or Subnet according to the instructions here [.](#)
 - There is at least 1 **SSH key in ACTIVE state**. If you do not have any SSH key, please create an SSH key according to the instructions here [.](#)
 - Installed and configured **kubectl** on your device. Please refer here [if](#) you are not sure how to install and use kubectl. In addition, you should not use a kubectl version that is too old, we recommend that you use a kubectl version that is no more than one version different from the cluster version.
-

Initialize Cluster

A **cluster in Kubernetes** is a collection of one or more virtual machines (VMs) connected together to run containerized applications. Cluster provides a unified environment to deploy, manage, and operate containers at scale.

To initialize a Cluster, follow the steps below:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select Activate.

Step 3: Wait until we successfully create your VKS account. After Activate successfully, select **Create a Cluster**

Step 4: At the Cluster initialization screen, we have set up information for the Cluster and a **Default Node Group** for you. You can keep these default values or

adjust the desired parameters for the Cluster and Node Group at Cluster Configuration, Default Node Group Configuration, Plugin. **When you choose to enable option, by default we will pre-install this plugin into your Cluster.**

Step 5: Select **Create Kubernetes cluster**. Please wait a few minutes for us to initialize your Cluster, the Cluster's status is now **Creating**.

Step 6: When the Cluster status is **Active**, you can view Cluster information and Node Group information by selecting Cluster Name in the **Name** column.

Connect and check the newly created Cluster information

After the Cluster is successfully initialized, you can connect and check the newly created Cluster information by following these steps:

Step 1: Visit <https://vks.console.vngcloud.vn/k8s-cluster>

Step 2: The Cluster list is displayed, select the icon and select **Download Config File** to download the kubeconfig file. This file will give you full access to your Cluster.

Step 3 : Rename this file to config and save it to the **~/.kube/config** directory

Step 4: Perform Cluster check via command:

- Run the following command to test **node**

```
kubectl get nodes
```

- If the results are returned as below, it means your Cluster was successfully initialized with 3 nodes as below.

NAME	STATUS	ROLES	AGE
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready	<none>	50m
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready	<none>	23m
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready	<none>	28m

Create Service Account and install VNGCloud Controller Manager

 **Note:**

When you initialize the Cluster according to the instructions above, if you have not enabled the **Enable vLB Native Integration Driver** option , by default we will not pre-install this plugin into your Cluster. You need to manually create Service Account and install VNGCloud Controller Manager according to the instructions below. If you have enabled the **Enable vLB Native Integration Driver** option , then we have pre-installed this plugin into your Cluster, skip the Service Account Initialization step, install VNGCloud Controller Manager and continue following the instructions from Deploy once. Workload.

- ✓ Instructions for creating Service Account and installing VNGCloud Controller Manager

Initialize Service Account

- Create or use a **service account** created on IAM and attach policy: **vLBFULLAccess** , **vServerFullAccess** . To create a service account, go here [and](#) follow these steps:
 - Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account
 - Find and select **Policy: vLBFULLAccess and Policy: vServerFullAccess** , then click " **Create a Service Account** " to create Service Account, Policy: vLBFULLAccess and Policy:

vServerFullAccess created by VNG Cloud, you cannot delete these policies.

- After successful creation, you need to save the **Client_ID** and **Secret_Key** of the Service Account to perform the next step.
- Uninstall cloud-controller-manager

```
kubectl get daemonset -n kube-system | grep -i "cloud-contro...  
# if your output is similar to the following, you MUST delete t...  
kubectl delete daemonset cloud-controller-manager -n kube-syste...
```

- Besides, you can delete the Service Account being used for the cloud-controller-manager you just removed

```
kubectl get sa -n kube-system | grep -i "cloud-controller-manag...  
# if your output is similar to the above, you MUST delete this :...  
kubectl delete sa cloud-controller-manager -n kube-system --force...
```

Install VNGCloud Controller Manager

- Install Helm version 3.0 or higher. Refer to <https://helm.sh/docs/intro/install/> for instructions on how to install.
- Add this repo to your cluster via the command:

```
helm repo add vks-helm-charts https://vngcloud.github.io/vks-he...  
helm repo update
```

- Replace your K8S cluster's ClientID, Client Secret, and ClusterID information and continue running:

```
helm install vngcloud-controller-manager vks-helm-charts/vngclou...  
--namespace kube-system \  
--set cloudConfig.global.clientID= <Lấy ClientID của Service />  
--set cloudConfig.global.clientSecret= <Lấy ClientSecret của Service />  
--set cluster.clusterID= <Lấy Cluster ID của cluster mà bạn đã...
```

- After the installation is complete, check the status of vngcloud-Integrate-controller pods:

```
kubectl get pods -n kube-system | grep vngcloud-controller-mana...
```

For example, in the image below you have successfully installed vngcloud-controller-manager:

NAME	READY	STATUS
vngcloud-controller-manager-8864c754c-bqhvz	1/1	Running

Deploy a Workload

The following is a guide for you to deploy the nginx service on Kubernetes.

Step 1 : Create Deployment, Service for Nginx app.

- Create **nginx-service-lb4.yaml** file with the following content:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
  spec:
    containers:
      - name: nginx
        image: nginx:1.19.1
        ports:
          - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80

```

- Deploy this Service using:

```
kubectl apply -f nginx-service-lb4.yaml
```

Step 2: Check the Deployment and Service information just deployed

- Run the following command to test **Deployment**

```
kubectl get svc,deploy,pod -owide
```

- If the results are returned as below, it means you have deployed Deployment successfully.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/T
service/nginx-service	LoadBalancer	10.96.74.154	<pending>	80:31

NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINER
deployment.apps/nginx-app	0/1	1	0	2s	nginx

NAME	READY	STATUS	RESTARTS	A
pod/nginx-app-7f45b65946-bmrcf	0/1	ContainerCreating	0	2

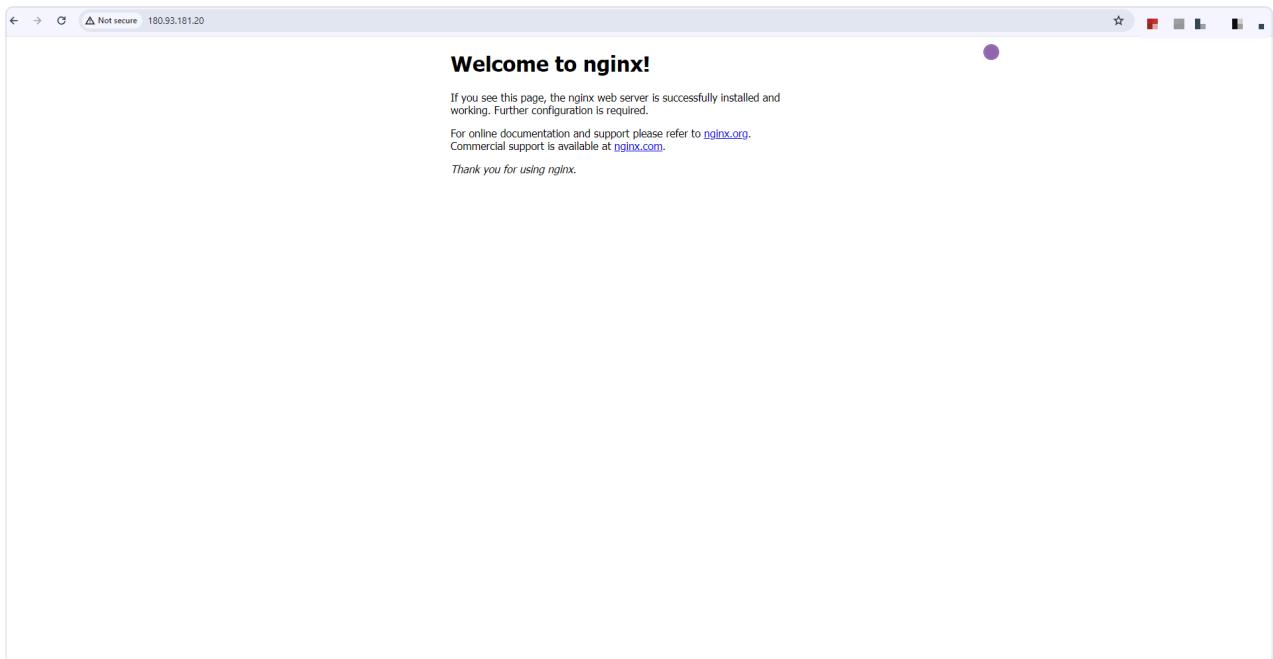
At this point, the vLB system will automatically create a corresponding LB for the deployed nginx app, for example:

Step 3: To access the just exported nginx app, you can use the URL with the format:

```
http://Endpoint/
```

You can get Load Balancer Public Endpoint information at the vLB interface. Specifically, access at <https://hcm-3.console.vngcloud.vn/vserver/load-balancer/vlb/>

For example, below I have successfully accessed the nginx app with the address: <http://180.93.181.20/>



You can see more about ALB at [Working with Network load balancing \(NLB\)](#) .



Attention:

- Changing the name or size (Rename, Resize) of the Load Balancer resource on vServer Portal can cause incompatibility with resources on the Kubernetes Cluster. This can lead to resources becoming inactive on the Cluster, or resources being resynchronized, or resource information between vServer Portal and the Cluster not matching. To prevent this problem, use `kubectl` Cluster resource management.

Expose a service through vLB Layer7

Prerequisites

To be able to initialize a **Cluster** and **Deploy a Workload**, you need:

- There is at least 1 **VPC** and 1 **Subnet in ACTIVE state**. If you do not have a VPC or Subnet yet, please create a VPC or Subnet according to the instructions here [.](#)
 - There is at least 1 **SSH key in ACTIVE state**. If you do not have any SSH key, please create an SSH key according to the instructions here [.](#)
 - Installed and configured **kubectl** on your device. Please refer here [if](#) you are not sure how to install and use kubectl. In addition, you should not use a kubectl version that is too old, we recommend that you use a kubectl version that is no more than one version different from the cluster version.
-

Initialize Cluster

A **cluster in Kubernetes** is a collection of one or more virtual machines (VMs) connected together to run containerized applications. Cluster provides a unified environment to deploy, manage, and operate containers at scale.

To initialize a Cluster, follow the steps below:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select Activate.

Step 3: Wait until we successfully create your VKS account. After Activate successfully, select **Create a Cluster**

Step 4: At the Cluster initialization screen, we have set up information for the Cluster and a **Default Node Group** for you. You can keep these default values or

adjust the desired parameters for the Cluster and Node Group at Cluster Configuration, Default Node Group Configuration, Plugin. **When you choose to enable option , by default we will pre-install this plugin into your Cluster.**

Step 5: Select **Create Kubernetes cluster**. Please wait a few minutes for us to initialize your Cluster, the Cluster's status is now **Creating** .

Step 6: When the Cluster status is **Active** , you can view Cluster information and Node Group information by selecting Cluster Name in the **Name** column .

Connect and check the newly created Cluster information

After the Cluster is successfully initialized, you can connect and check the newly created Cluster information by following these steps:

Step 1: Visit <https://vks.console.vngcloud.vn/k8s-cluster>

Step 2: The Cluster list is displayed, select the icon and select **Download Config File** to download the kubeconfig file. This file will give you full access to your Cluster.

Step 3 : Rename this file to config and save it to the **~/.kube/config** directory

Step 4: Perform Cluster check via command:

- Run the following command to test **node**

```
kubectl get nodes
```

- If the results are returned as below, it means your Cluster was successfully initialized with 3 nodes as below.

NAME	STATUS	ROLES	AGE
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready	<none>	50m
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready	<none>	23m
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready	<none>	28m

Create Service Account and install VNGCloud Ingress Controller

Attention:

When you initialize the Cluster according to the instructions above, if you have not enabled the **Enable vLB Native Integration Driver** option , by default we will not pre-install this plugin into your Cluster. You need to manually create Service Account and install VNGCloud Ingress Controller according to the instructions below. If you have enabled the **Enable vLB Native Integration Driver** option , then we have pre-installed this plugin into your Cluster, skip the Service Account Initialization step, install VNGCloud Ingress Controller and continue following the instructions from Deploy once. Workload.

- ✓ Create Service Account and install VNGCloud Ingress Controller

Initialize Service Account

- Create or use a **service account** created on IAM and attach policy: **vLBFULLAccess** , **vServerFullAccess** . To create a service account, go here [and](#) follow these steps:
 - Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account
 - Find and select **Policy: vLBFULLAccess and Policy: vServerFullAccess** , then click " **Create a Service Account** " to create Service Account, Policy: vLBFULLAccess and Policy:

vServerFullAccess created by VNG Cloud, you cannot delete these policies.

- After successful creation, you need to save the **Client_ID** and **Secret_Key** of the Service Account to perform the next step.

Install VNGCloud Ingress Controller

- Install Helm version 3.0 or higher. Refer to <https://helm.sh/docs/intro/install/> for instructions on how to install.
- Add this repo to your cluster via the command:

```
helm repo add vks-helm-charts https://vngcloud.github.io/vks-he:  
helm repo update
```

- Replace your K8S cluster's ClientID, Client Secret, and ClusterID information and continue running:

```
helm install vngcloud-ingress-controller vks-helm-charts/vngclo:  
--namespace kube-system \  
--set cloudConfig.global.clientID= <Lấy ClientID của Service /  
--set cloudConfig.global.clientSecret= <Lấy ClientSecret của S:  
--set cluster.clusterID= <Lấy Cluster ID của cluster mà bạn đ
```

- After the installation is complete, check the status of vngcloud-ingress-controller pods:

```
kubectl get pods -n kube-system | grep vngcloud-ingress-control
```

For example, in the image below you have successfully installed vngcloud-controller-manager:

NAME	READY	STATUS	RES
vngcloud-ingress-controller-0	1/1	Running	0

Deploy a Workload

The following is a guide for you to deploy the nginx service on Kubernetes.

Step 1 : Create Deployment for Nginx app.

- Create **nginx-service-lb7.yaml** file with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.19.1
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- Deploy This deployment equals:

```
kubectl apply -f nginx-service-lb7.yaml
```

Step 2: Check the Deployment and Service information just deployed

- Run the following command to test **Deployment**

```
kubectl get svc,deploy,pod -owide
```

- If the results are returned as below, it means you have deployed Deployment successfully.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP
service/nginx-service	NodePort	10.96.25.133	<none>	80:32572

NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONT
deployment.apps/nginx-app	1/1	1	1	2m50s	ngin

NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx-app-7f45b65946-6wlgw	1/1	Running	0	2m49s	172

Create Ingress Resource

- Create **nginx-ingress.yaml** file with the following content:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
spec:
  ingressClassName: "vngcloud"
  defaultBackend:
    service:
      name: nginx-service
      port:
        number: 80
  rules:
    - http:
        paths:
          - path: /path1
            pathType: Exact
            backend:
              service:
                name: nginx-service
                port:
                  number: 80
```

- Run the following command to deploy Ingress

```
kubectl apply -f nginx-ingress.yaml
```

At this time, the vLB system will automatically create a LB corresponding to the Ingress resource above, for example:



Attention:

- Currently Ingress only supports TLS port 443 and is the termination point for TLS (TLS termination). TLS Secret must contain fields with key names `tls.crt` and `tls.key`, which are the certificate and private key to use for TLS. If you want to use a Certificate for a host, please upload the Certificate according to the instructions at [Upload a certificate] and use them as an annotation. For example:

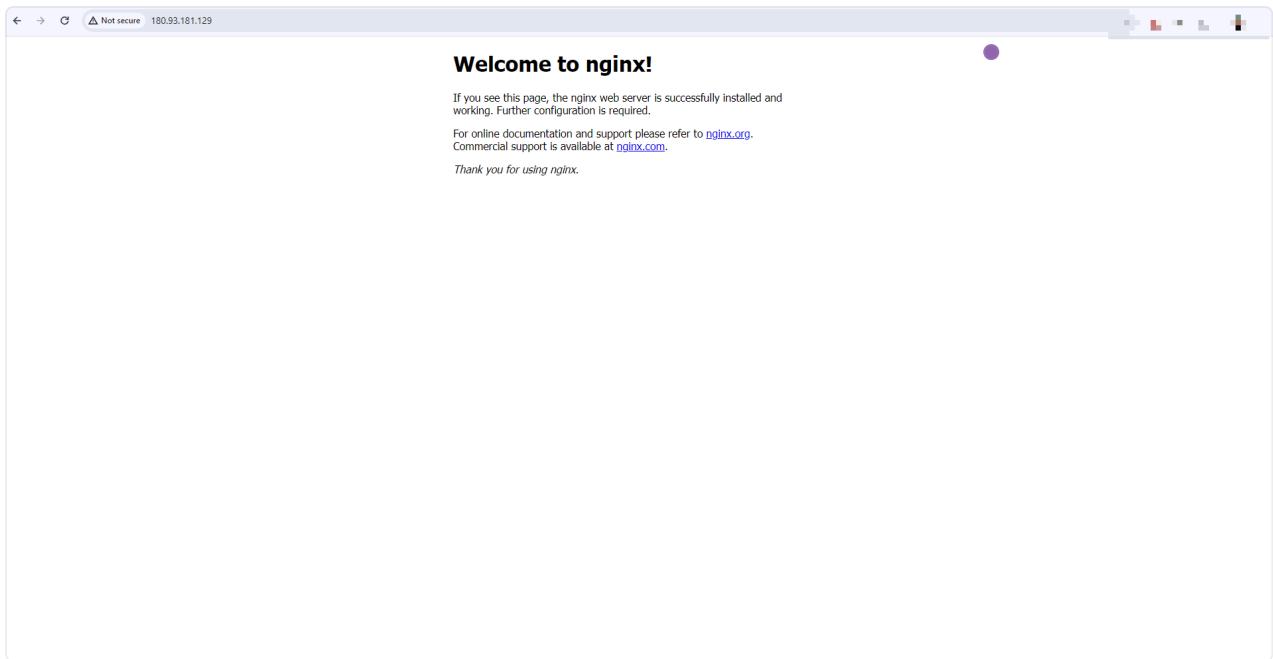
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    # kubernetes.io/ingress.class: "vngcloud" # this annotation is deprecated
    vks.vngcloud.vn/load-balancer-id: "lb-6cdea8fd-4589-410e-933f-c3bc46f"
    vks.vngcloud.vn/certificate-ids: "secret-a6d20ec6-f3e5-499a-981b-b148"
spec:
  ingressClassName: "vngcloud"
  defaultBackend:
    service:
      name: apache-service
      port:
        number: 80
  tls:
    - hosts:
      - host.example.com
  rules:
    - host: host.example.com
      http:
        paths:
          - path: /path1
            pathType: Exact
            backend:
              service:
                name: nginx-service
                port:
                  number: 80
```

To access the nginx app, you can use the Load Balancer Endpoint that the system has created.

```
http://Endpoint/
```

You can get Load Balancer Public Endpoint information at the vLB interface.
Specifically, access at

For example, below I have successfully accessed the nginx app with the address:
<http://180.93.181.129/>



You can see more about ALB at [Working with Application Load Balancer \(ALB\)](#).



Attention:

- Changing the name or size (Rename, Resize) of the Load Balancer resource on vServer Portal can cause incompatibility with resources on the Kubernetes Cluster. This can lead to resources becoming inactive on the Cluster, or resources being resynchronized, or resource information between vServer Portal and the Cluster not matching. To prevent this problem, use `kubectl` Cluster resource management.

Preserve Source IP when using NLB and Nginx Ingress Controller

Preserve Source IP when using vLB Layer 4 and Nginx Ingress Controller in Kubernetes is the process of maintaining the client's original IP address when traffic is forwarded through the load balancer and into the Kubernetes cluster. This is important in some cases when you need detailed information about the client's connection, such as the client's original IP address and root port, to be able to make traffic handling or logging decisions. Exactly. Below are our specific instructions to help you implement this usecase.

Prerequisites

- You have initialized the Cluster on the VKS system according to the instructions here [and](#) **VNGCloud Controller Manager** has been installed on your cluster with appversion from **v0.2.1** or higher. If your appversion is lower than this standard version, you can perform the upgrade according to the following instructions:
 - First, you need to get the release name of **vngcloud-controller-manager** installed on your cluster:

```
$ helm list -A | grep vngcloud-controller-manager  
vngcloud-controller-manager-1716448250          kube-system    10
```

- Then, please upgrade to the latest version via the command:

```
helm upgrade vngcloud-controller-manager-1716448250 oci://vcr.vngcloud  
--namespace kube-system
```

- Next, you need to install nginx-ingress-controller with the command:

```
helm install nginx-ingress-controller oci://ghcr.io/nginxinc/charts/nginx
```

ConfigMap for Nginx Ingress Controller

- Add to Nginx Ingress Controller's ConfigMap the settings to enable proxy protocol via command:

```
kubectl edit cm -n kube-system nginx-ingress-controller
```

- The code you need to add is as follows:

```
data:  
  proxy-protocol: "True"  
  real-ip-header: proxy_protocol  
  real-ip-recursive: "True"  
  set-real-ip-from: 0.0.0.0/0
```

Configure vLB Layer 4

- Next, you need to configure vLB Layer4 to allow the use of proxy protocol for the Load Balancer Nginx service. The input value is a list of service names in Load Balancer using Proxy Protocol.

```
kubectl annotate service -n kube-system nginx-ingress-controller-controll  
vks.vngcloud.vn/enable-proxy-protocol="http,https"
```

- Finally, please perform NLB testing on vLB Portal until these Load Balancers are ACTIVE with full listener and pool.

Using

- Suppose, you have a service prometheus-node-exporter with port 9100 in the default namespace, you can apply the following yaml to make it accessible via

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-ingress
  namespace: default
spec:
  ingressClassName: nginx
  rules:
  - host: kkk.example.com
    http:
      paths:
      - backend:
          service:
            name: prometheus-node-exporter
            port:
              number: 9100
        path: /metrics
        pathType: Exact
```

- Then I use IP 103.245.252.75 to curl to host kkk.example.com as follows:

```
curl -H 'Host: kkk.example.com' http://103.245.250.142/metrics
Client sent an HTTP request to an HTTPS server.
```

- The recorded log result has this Client IP information as shown:

```
103.245.252.75 - - [11/Jun/2024:10:04:10 +0000] "GET /metrics HTTP/1.1" 400 59 "-" "curl/7.81.0" "-"
103.245.252.75 - - [11/Jun/2024:10:04:10 +0000] "GET /metrics HTTP/1.1" 400 59 "-" "curl/7.81.0" "-"
```

Integrate with Container Storage Interface (CSI)

•

Prerequisites

To be able to initialize a **Cluster** and **Deploy a Workload**, you need:

- There is at least 1 **VPC** and 1 **Subnet in ACTIVE state**. If you do not have a VPC or Subnet yet, please create a VPC or Subnet according to the instructions here [.](#)
 - There is at least 1 **SSH key in ACTIVE state**. If you do not have any SSH key, please create an SSH key according to the instructions here [.](#)
 - Installed and configured **kubectl** on your device. Please refer here [if](#) you are not sure how to install and use kubectl. In addition, you should not use a kubectl version that is too old, we recommend that you use a kubectl version that is no more than one version different from the cluster version.
-

Initialize Cluster

A **cluster in Kubernetes** is a collection of one or more virtual machines (VMs) connected together to run containerized applications. Cluster provides a unified environment to deploy, manage, and operate containers at scale.

To initialize a Cluster, follow the steps below:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select **Activate**.

Step 3: Wait until we successfully create your VKS account. After Activate successfully, select **Create a Cluster**

Step 4: At the Cluster initialization screen, we have set up information for the Cluster and a **Default Node Group** for you. You can keep these default values or adjust the desired parameters for the Cluster and Node Group at Cluster Configuration, Default Node Group Configuration, Plugin. When you choose to enable the **Enable vLB Native Integration Driver** option , by default we will pre-install this plugin into your Cluster.

Step 5: Select **Create Kubernetes cluster**. Please wait a few minutes for us to initialize your Cluster, the Cluster's status is now **Creating** .

Step 6: When the Cluster status is Active , you can view Cluster information and Node Group information by selecting Cluster Name in the **Name** column .

Connect and check the newly created Cluster information

After the Cluster is successfully initialized, you can connect and check the newly created Cluster information by following these steps:

Step 1: Visit <https://vks.console.vngcloud.vn/k8s-cluster>

Step 2: The Cluster list is displayed, select **Download Config File** to download the kubeconfig file. This file will give you full access to your Cluster.

Step 3 : Rename this file to config and save it to the **~/.kube/config** directory

Step 4: Perform Cluster check via command:

- Run the following command to test **node**

```
kubectl get nodes
```

- If the results are returned as below, it means your Cluster was successfully initialized with 3 nodes as below.

NAME	STATUS	ROLES	AGE
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready	<none>	50m
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready	<none>	23m
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready	<none>	28m

Create Service Account and install VNGCloud BlockStorage CSI Driver



Attention:

- When you initialize the Cluster according to the instructions above, if you have not enabled the **Enable BlockStore Persistent Disk CSI Driver** option , by default we will not pre-install this plugin into your Cluster. You need to manually create Service Account and install VNGCloud BlockStorage CSI Driver according to the instructions below. If you have enabled the **Enable BlockStore Persistent Disk CSI Driver** option , we have pre-installed this plugin into your Cluster, skip the Service Account Initialization step, install VNGCloud BlockStorage CSI Driver and continue following the instructions from now on. Deploy a Workload.
- VNGCloud BlockStorage CSI Driver** only supports attaching volumes to a single node (VM) throughout the life of that volume. If you have a need for ReadWriteMany, you may consider using the NFS CSI Driver, as it allows multiple nodes to Read and Write on the same volume at the same time. This is very useful for applications that need to share data between multiple pods or services in Kubernetes.

✓ Create Service Account and install VNGCloud BlockStorage CSI Driver

Initialize Service Account

- Create or use a **service account** created on IAM and attach policy: **vServerFullAccess** . To create a service account, go here [and](#) follow these steps:

- Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account
- Find and select **Policy: vServerFullAccess** , then click " **Create a Service Account** " to create a Service Account, Policy: vLBFullAccess and Policy: vServerFullAccess are created by VNG Cloud, you cannot delete these policies.
- After successful creation, you need to save **the Client_ID** and **Secret_Key** of the Service Account to perform the next step.

Install VNGCloud BlockStorage CSI Driver

- Install Helm version 3.0 or higher. Refer to <https://helm.sh/docs/intro/install/> for instructions on how to install.
- Add this repo to your cluster via the command:

```
helm repo add vks-helm-charts https://vngcloud.github.io/vks-he
helm repo update
```

- Replace your K8S cluster's ClientID, Client Secret, and ClusterID information and continue running:

```
helm install vngcloud-blockstorage-csi-driver vks-helm-charts/vn
--replace --namespace kube-system \
--set vngcloudAccessSecret.keyId=${VNGCLOUD_CLIENT_ID} \
--set vngcloudAccessSecret.accessKey=${VNGCLOUD_CLIENT_SECRET} \
--set vngcloudAccessSecret.vksClusterId=${VNGCLOUD_VKS_CLUSTER}
```

- After the installation is complete, check the status of vngcloud-blockstorage-csi-driver pods:

```
kubectl get pods -n kube-system | grep vngcloud-ingress-control
```

For example, in the image below you have successfully installed vngcloud-controller-manager:

NAME	READY	STATUS
vngcloud-csi-controller-56bd7b85f-ctpns	7/7	Running
vngcloud-csi-controller-56bd7b85f-npp9n	7/7	Running
vngcloud-csi-node-c8r2w	3/3	Running

Deploy a Workload

The following is a guide for you to deploy the nginx service on Kubernetes.

Step 1 : Create Deployment for Nginx app.

- Create **nginx-service.yaml** file with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.19.1
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- Deploy This deployment equals:

```
kubectl apply -f nginx-service.yaml
```

Step 2: Check the Deployment and Service information just deployed

- Run the following command to test **Deployment**

```
kubectl get svc,deploy,pod -owide
```

- If the results are returned as below, it means you have deployed Deployment successfully.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/
service/nginx-app	NodePort	10.96.215.192	<none>	3008
service/nginx-service	LoadBalancer	10.96.179.221	<pending>	80:3
NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx-app	1/1	1	1	16s
NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-app-7f45b65946-t7d7k	1/1	Running	0	16s
				172.1

Create Persistent Volume

- Create a **persistent-volume.yaml** file with the following content:

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-expansion-storage-class          # [1] The StorageClass
provisioner: bs.csi.vngcloud.vn             # The VNG-CLOUD CSI
parameters:
  type: vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018 # The volume type U
allowVolumeExpansion: true                  # MUST set this val
---

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-expansion-pvc                   # [2] The PVC name, CA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi                         # [3] The PVC size, CA
  storageClassName: my-expansion-storage-class # [4] The StorageClass
---

apiVersion: v1
kind: Pod
metadata:
  name: nginx                            # [5] The Pod name, CA
spec:
  containers:
    - image: nginx
      imagePullPolicy: IfNotPresent
      name: nginx
      ports:
        - containerPort: 80
          protocol: TCP
    volumeMounts:
      - mountPath: /var/lib/www/html
        name: my-volume-name                 # MUST be the same as
    volumes:
      - name: my-volume-name                # [6] The volume name,
        persistentVolumeClaim:
          claimName: my-expansion-pvc       # MUST be the same as
          readOnly: false

```

- Run the following command to deploy Ingress

```
kubectl apply -f persistent-volume.yaml
```

At this time, the vServer system will automatically create a Volume corresponding to the yaml file above, for example:

The screenshot shows the VNG Cloud vServer BlockStore Volumes page. The left sidebar includes sections for vServer, Route tables, Peerings, Bandwidths, Interconnects, Network ACL, Server (Servers, Server Groups, SSH Keys, System Images, Flavors), and BlockStore (Volumes, Volume Types, Images, Backups, Snapshots). The main content area is titled 'Volume' and defines it as a resource for storing business data. It features a 'Create volume' button and a table listing seven volumes. The columns in the table are Name, Status, Type, IOPS, Size, Attachment, Multi-Attach, and Action. The volumes listed are:

Name	Status	Type	IOPS	Size	Attachment	Multi-Attach	Action
vol-2f295ad8-98c4-4043-af28-98637704bb59 pvc-14456f4a-e9e9-435d-a94f-5a2e820954e9 21/04/2024 21:16:57	IN USE	SSD	200	20	ins-4ae9966b-29a1-4c01-b6eb-a543b1d2abf4	No	⋮
vol-4403a56a-11fb-470c-bf5e-56d6f134c5c1 ng-3f06013a-f6a5-47ba-a51f-be5e9c2b10a7-ecea1 boot_volume 19/04/2024 18:27:26	IN USE	SSD	3000	20	ins-4ae9966b-29a1-4c01-b6eb-a543b1d2abf4	No	⋮
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	⋮
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	⋮
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	⋮
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	⋮
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	⋮

Total: 7 Show: 10

Create Snapshots

Snapshot is a low-cost, convenient and effective data backup method and can be used to create images, restore data and distribute copies of data. If you are a new user who has never used the Snapshot service, you will need to Activate Snapshot Service before you can create a Snapshot for your Persistent Volume.

Activate Snapshot Service

To be able to create Snapshots, you need to perform Activate Snapshot Service. You will not be charged for activating the snapshot service. After you create snapshots, costs will be calculated based on the storage capacity and storage time of these snapshots. Follow these steps to enable the Snapshot service:

Step 1: Visit <https://hcm-3.console.vngcloud.vn/vserver/block-store/snapshot/overview>

Step 2: Select Activate Snapshot Service .

For example:

The screenshot shows the VNG Cloud console interface. On the left, there is a sidebar with various service categories like vServer, BlockStore, and Snapshots. Under Snapshots, the 'Snapshot' option is selected and has a 'DISABLE' status. The main content area displays a 'Billing Note' stating that snapshots are low-cost, convenient, and efficient for backup and recovery. It also mentions that snapshots are billed based on storage space and time stored. A red arrow points to a prominent orange button labeled 'Activate Snapshot Service'. Below the button is a link 'What can I do with Snapshots? →'.

Install VNGCloud Snapshot Controller

- Install Helm version 3.0 or higher. Refer to <https://helm.sh/docs/intro/install/> for instructions on how to install.
- Add this repo to your cluster via the command:

```
helm repo add vks-helm-charts https://vngcloud.github.io/vks-helm-charts
helm repo update
```

- Continue running:

```
helm install vngcloud-snapshot-controller vks-helm-charts/vngcloud-snapshot-controller
--replace --namespace kube-system
```

- After the installation is complete, check the status of vngcloud-blockstorage-csi-driver pods:

```
kubectl get pods -n kube-system
```

For example, in the image below you have successfully installed vngcloud-controller-manager:

NAME	READY	STATUS
snapshot-controller-7fdd984f89-745tg	0/1	ContainerCreating
snapshot-controller-7fdd984f89-k94wq	0/1	ContainerCreating

Create a snapshot.yaml file with the following content:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: my-snapshot-storage-class  # [2] The name of the volume snapshot
driver: bs.csi.vngcloud.vn
deletionPolicy: Delete
parameters:
  force-create: "false"
---

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: my-snapshot-pvc  # [4] The name of the snapshot, CAN be changed
spec:
  volumeSnapshotClassName: my-snapshot-storage-class  # MUST match with [2]
  source:
    persistentVolumeClaimName: my-expansion-pvc  # MUST match with [3]
```

- Run the following command to deploy Ingress

```
kubectl apply -f snapshot.yaml
```

Check the newly created PVC and Snapshot

- After applying the file successfully, you can check the service and pvc list via:

```
kubectl get sc,pvc,pod -owide
```

NAME	PROVISIONER				
storageclass.storage.k8s.io/my-expansion-storage-class	bs.csi.vngclou				
storageclass.storage.k8s.io/sc-iops-200-retain (default)	bs.csi.vngclou				
NAME	STATUS	VOLUME			
persistentvolumeclaim/my-expansion-pvc	Bound	pvc-14456f4a-ee9e-435d-			
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx	1/1	Running	0	10m	172.1
pod/nginx-app-7f45b65946-t7d7k	1/1	Running	0	94m	172.1

Change the IOPS parameters of the newly created Persistent Volume

To change the IOPS parameters of the newly created Persistent Volume, follow these steps:

Step 1: Run the command below to list the PVCs in your Cluster

```
kubectl get persistentvolumes
```

Step 2: Edit the PVC YAML file according to the command

```
kubectl edit pvc my-expansion-pvc
```

- If you have not edited the IOPS of the Persistent Volume before, when you run the above command, add an annotation `bs.csi.vngcloud.vn/volume-type: "volume-type-id"`. For example, below I am changing the Persistent Volume IOPS from 200 (Volume type id = `vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018`) to 1000 (Volume type id = `vtype-85b39362-a360-4bbb-9afa-a36a40cea748`)

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    bs.csi.vngcloud.vn/volume-type: "vtype-85b39362-a360-4bbb-9afa-a36a40
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"PersistentVolumeClaim","metadata":{}}}
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: bs.csi.vngcloud.vn
    volume.kubernetes.io/storage-provisioner: bs.csi.vngcloud.vn
  creationTimestamp: "2024-04-21T14:16:53Z"
  finalizers:
    - kubernetes.io/pvc-protection
  name: my-expansion-pvc
  namespace: default
  resourceVersion: "11041591"
  uid: 14456f4a-ee9e-435d-a94f-5a2e820954e9
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: my-expansion-storage-class
  volumeMode: Filesystem
  volumeName: pvc-14456f4a-ee9e-435d-a94f-5a2e820954e9
status:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 20Gi
  phase: Bound

```

- If you have edited the IOPS of the Persistent Volume before, when you run the above command, your yaml file will already have the annotation `bs.csi.vngcloud.vn/volume-type: "volume-type-id"`. Now, edit this annotation to the Volume type id with the IOPS you desire.

Change the Disk Volume of the newly created Persistent Volume

To change the Disk Volume of the newly created Persistent Volume, run the following command:

For example, initially the PVC created was 20 Gi in size, now I will increase it to 30 Gi

```
kubectl patch pvc my-expansion-pvc -p '{"spec":{"resources":{"requests":{}}
```

 **Attention:**

- You can only increase Disk Volume but cannot reduce Disk Volume size.

Restore Persistent Volume from Snapshot

To restore Persistent Volume from Snapshot, follow these steps:

- Create file **restore-volume.yaml** with the following content:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-restore-pvc # The name of the PVC, CAN be changed
spec:
  storageClassName: my-expansion-storage-class
  dataSource:
    name: my-snapshot-pvc # MUST match with [4] from the section 5.2
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

Upgrading Control Plane Version

Currently, our VKS system has supported you to upgrade Control Plane Version, you can:

- Upgrade to a newer **Minor Version** (e.g. 1.24 to 1.25)
- Upgrade to a newer **Patch Version (for example: 1.24.2-VKS.100 to 1.24.5-VKS.200)**

To upgrade the Control Plane version, you can follow these instructions:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select the Kubernetes Cluster menu.

Step 3: Select the icon and select **Upgrade control plane version** to upgrade the control plane version.

Step 4: You can select a new version for the control plane. The new version needs to be valid and compatible with the current version of the cluster. Specifically: you can choose:

- Upgrade to a newer Minor Version (e.g. 1.24 to 1.25)
- Upgrade to a newer Patch Version (for example: 1.24.2-VKS.100 to 1.24.5-VKS.200)

Step 4: The VKS system will upgrade the Control Plane components of the Cluster to the new version. After the upgrade is complete, the Cluster status returns to **ACTIVE** .



Attention:

- Upgrading Control Plane Version is optional and independent of upgrading Node Group Version. However, Control Plane Version and Node Group Version in the same Cluster cannot differ by more than 1 minor version. Besides, the VKS system automatically upgrades

Control Plane Version when the current K8S Version used for your Cluster exceeds the supplier's support period.

- During the Control Plane Version upgrade, you cannot perform other actions on your Cluster.
- Below are a few notes before, during and after the upgrade process, please refer to:

Before getting into work:

- Back up data: You should back up cluster data before upgrading to ensure safety in case the upgrade fails.
- Check the current version: Visit Releases for a list of supported versions. Select a new version that is valid and compatible with the current version of the cluster.
- Ensure cluster availability: Cluster must be in ACTIVE status and all nodes must be HEALTHY.
- Stop running tasks: Stop running tasks on the cluster to avoid affecting the upgrade process.

While doing:

- Monitor cluster status: Monitor cluster status during the upgrade process. The cluster status will change to UPDATING and after completion will return to ACTIVE.
- Check system logs: Check system logs to detect any errors or warnings during the upgrade process.

After implementation:

- Check cluster availability: Confirm that the cluster has been upgraded successfully and all nodes are operating normally.
- Test applications: Test applications running on the cluster to ensure they work properly after the upgrade.

Note:

- Upgrading Control Plane Version may take some time depending on the size and complexity of the cluster.

- In some rare cases, the Control Plane Version upgrade may fail. If this happens, the VKS system will automatically rollback the cluster to the current version.

Upgrading Node Group Version

Currently, our VKS system has supported you to upgrade Node Group Version, you can upgrade Node Group Version to:

- **Control Plane Version** (For example upgrade from 1.24 (current Node Group version) to 1.25 (current Control Plane Version), but cannot upgrade to other versions.

To perform a Node Group Version upgrade, you can follow these instructions:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select the **Kubernetes Cluster menu**. Select a **Cluster** where you want to upgrade **Node Group Version** .

Step 3: Select the icon and select **Upgrade Node Group Version** to upgrade the node group version.

Step 4: You can select the new version for all Node Groups. The new version needs to be valid and compatible with the current version of the cluster. Specifically: you can choose:

- Upgrade Node Group to the same version as Control Plane Version (for example: 1.24 to 1.25)

Step 5: The VKS system will upgrade all Node Groups to the Control Plane version. After the upgrade is complete, the Node Group status returns to **ACTIVE** .



Attention:

- Upgrading Node Group Version is optional and independent of upgrading Control Plane Version. However, all Node Groups in a Cluster will be upgraded at the same time, as well as Control Plane Version and Node Group Version in the same Cluster cannot differ by more than 1 minor version. Besides, the VKS system automatically

upgrades the Node Group Version when the current K8S Version being used for your Cluster exceeds the supplier's support period.

- During the Node Group Version upgrade, you cannot perform other actions on your Node Group.
- Below are a few notes before, during and after the upgrade process, please refer to:

Before getting into work:

- Check the current version: Visit Releases for a list of supported versions. Select a new version that is valid and compatible with the current version of the cluster.
- Ensure Node Group availability: Node Group must be in ACTIVE status and all nodes must be HEALTHY.
- Stop running tasks: Stop running tasks on the cluster to avoid affecting the upgrade process.

While doing:

- Monitor Node Group status: Monitor Node Group status during the upgrade process. Node Group status will change to UPDATING and after completion will return to ACTIVE.
- Check system logs: Check system logs to detect any errors or warnings during the upgrade process.

After implementation:

- Check Node Group Availability: Confirm that the Node Group has been upgraded successfully and all nodes are operating normally.
- Test applications: Test applications running on the cluster to ensure they work properly after the upgrade.

Note:

- Upgrading Node Group Version may take some time depending on the size and complexity of the Node Group.
- In some rare cases, upgrading Node Group Version may fail. If this happens, the VKS system will automatically rollback the cluster to the current version.

Use Terraform to create a Cluster and Node Group

•

Overview

Terraform is an open source tool used to automate the provisioning and management of infrastructure such as virtual machines, networking, storage, and Kubernetes.

With Terraform, you can describe your desired infrastructure in code, then Terraform will perform the necessary operations to create or update the infrastructure to match your description.

Initialize Cluster and Node Group

To initialize a Kubernetes Cluster using Terraform, you need to perform the following steps:

1. **Access the IAM Portal here**, create a Service Account with [VKS Full Access](#) authority . Specifically, at the IAM site, you can:
 - Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account.
 - Find and select **Policy: VKSFullAccess** then click " **Create a Service Account** " to create a Service Account, **Policy: VKSFullAccess** is created by VNG Cloud, you cannot delete these policies.
 - After successful creation, you need to save **the Client_ID** and **Secret_Key** of the Service Account to perform the next step.
2. **Access the VKS Portal here**, **Activate the** VKS service on the **Overview tab**. Please wait until we successfully create your VKS account.
3. **Install Terraform:**

- Download and install Terraform for your operating system from <https://developer.hashicorp.com/terraform/install> .

4. Initialize Terraform configuration:

- Create a file `variable.tf` and declare Service Account information in this file.
- Create a file `main.tf` and define the Kubernetes Cluster resources you want to create.

For example:

- The file `variable.tf`: you need to replace the Client ID and Client Secret created in step 1 in this file.

```
variable "client_id" {
  type = string
  default = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
variable "client_secret" {
  type = string
  default = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

- The file `main.tf`: in this example I initialize a Cluster and a Node Group has the following information:
 - Cluster name: my-cluster
 - K8S Version: v1.28.8
 - Mode: Public Cluster and Public Node Group
 - Node Group name: my-nodegroup
 - Turn on AutoScaling: scale from 0 to 5 nodes

Attention:

- In the `main.tf` file, to initialize a cluster with a node group, you must pass in the following parameters:

```
vpc_id      = "net-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"  
subnet_id   = "sub-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"  
ssh_key_id= "ssh-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"
```

- The sample file `main.tf` helps you create Cluster and Node Group according to the configuration above:

```

terraform {
  required_providers {
    vngcloud = {
      source  = "vngcloud/vngcloud"
      version = "1.2.2"
    }
  }
}

provider "vngcloud" {
  token_url      = "https://iamapis.vngcloud.vn/accounts-api/v2/auth/to
  client_id      = var.client_id
  client_secret   = var.client_secret
  vserver_base_url = "https://hcm-3.api.vngcloud.vn/vserver/vserver-gatew
  vlb_base_url    = "https://hcm-3.api.vngcloud.vn/vserver/vlb-gateway"
}

resource "vngcloud_vks_cluster" "primary" {
  name      = "my-cluster"
  description = "VNGCLOUD uses terraform"
  version = "v1.28.8"
  cidr      = "172.16.0.0/16"
  enable_private_cluster = false
  network_type = "CALICO"
  vpc_id     = "net-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"
  subnet_id  = "sub-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"
  enabled_load_balancer_plugin = true
  enabled_block_store_csi_plugin = true
}

resource "vngcloud_vks_cluster_node_group" "primary" {
  cluster_id= vngcloud_vks_cluster.primary.id
  name= "my-nodegroup"
  num_nodes
  auto_scale_config {
    min_size = 0
    max_size = 5
  }
  upgrade_config {
    strategy = "SURGE"
    max_surge = 1
  }
  max_unavailable = 0
  image_id = "img-983d55cf-9b5b-44cf-aa72-23f3b25d43ce"
  flavor_id = "flav-9e88cfb4-ec31-4ad4-8ba5-243459f6dc4b"
  disk_size = 20
  disk_type = "vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018"
  enable_private_nodes = false
}

```

```
  ssh_key_id= "ssh-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"
  labels = {
    "mylabel" = "vngcloud"
  }
  taint {
    key      = "mykey"
    value    = "myvalue"
    effect   = "PreferNoSchedule"
  }
}
```

1. Initialize Terraform:

- Run the command `terraform init`. This command will download the necessary plugins and initialize the Terraform state.

1. Apply Terraform configuration:

- Run command `terraform apply`. This command will create a Kubernetes Cluster as described in the `main.tf`.

See more about how to use Terraform to work with VKS [here](#).

Working with NVIDIA GPU Node Group

•

Overview

- The [NVIDIA GPU Operator](#) is an operator that simplifies the deployment and management of GPU nodes in Kubernetes clusters. It provides a set of Kubernetes custom resources and controllers that work together to automate the management of GPU resources in a Kubernetes cluster.
- In this guide, we will show you how to:
 - Create a nodegroup with NVIDIA GPUs in a VKS cluster.
 - Install the NVIDIA GPU Operator in a VKS cluster.
 - Deploy your GPU workload in a VKS cluster.
 - Configure GPU Sharing in a VKS cluster.
 - Monitor GPU resources in a VKS cluster.
 - Autoscale GPU resources in a VKS cluster.

Create a nodegroup with NVIDIA GPUs in a VKS cluster

- A VKS cluster with at least **one NVIDIA GPU nodegroup**.
- `kubectl` command-line tool installed on your machine. For more information, see [Install and Set Up kubectl](#).
- `helm` command-line tool installed on your machine. For more information, see [Installing Helm](#).
- (Optional) Other tools and libraries that you can use to monitor and manage your Kubernetes resources:
 - `kubectl-view-allocations` plugin for monitoring cluster resources. For more information, see [kubectl-view-allocations](#).
- The image below shows my machine setup, it will be used in this guide:

```
# Check kubectl CLI version  
kubectl version  
  
# Check Helm version  
helm version  
  
# Check kubectl-view-allocations version  
kubectl-view-allocations --version
```

- And this is my VKS cluster with 1 NVIDIA GPU nodegroup, it will be used in this guide, execute the following command to check the nodegroup in your cluster:

```
kubectl get nodes -owide
```

Installing the GPU Operator

- This guide only focus on installing the NVIDIA GPU Operator, for more information about the NVIDIA GPU Operator, see [NVIDIA GPU Operator Documentation](#). We manually install the NVIDIA GPU Operator in a VKS cluster by using Helm charts, execute the following command to install the NVIDIA GPU Operator in your VKS cluster:

```
helm install nvidia-gpu-operator --wait --version v24.3.0 \  
-n gpu-operator --create-namespace \  
oci://vcr.vngcloud.vn/81-vks-public/vks-helm-charts/gpu-operator \  
--set dcgmExporter.serviceMonitor.enabled=true
```

- You **MUST** wait for the installation to complete (*about 5-10 minutes*), execute the following command to check all the pods in the `gpu-operator` namespace are **running**:

```
kubectl -n gpu-operator get pods -owide
```

- The operator will label the node with the `nvidia.com/gpu` label, which can be used to filter the nodes that have GPUs. The `nvidia.com/gpu` label is used by the NVIDIA GPU Operator to identify nodes that have GPUs. The NVIDIA GPU

Operator will only deploy the NVIDIA GPU device plugin on nodes that have the `nvidia.com/gpu` label.

```
kubectl get node -o json | jq '.items[].metadata.labels' | grep "nvidi
```

- For the above result, the single node in the cluster has the `nvidia.com/gpu` label, which means that the node has GPUs.
- These labels also tell that this node is using 1 card of RTX 2080Ti GPU, number of available GPUs, the GPU Memory and other information.
- On the pod `nvidia-device-plugin-daemonset` in the `gpu-operator` namespace, you can execute `nvidia-smi` command to check the GPU information of the node:

```
POD_NAME=$(kubectl -n gpu-operator get pods -l app=nvidia-device-plugi  
kubectl -n gpu-operator exec -it $POD_NAME -- nvidia-smi
```

Deploy your GPU workload

Cuda VectorAdd Test

- In this section, we will show you how to deploy a GPU workload in a VKS cluster. We will use the `cuda-vectoradd-test` workload as an example. The `cuda-vectoradd-test` workload is a simple CUDA program that adds two vectors together. The program is provided as a container image that you can deploy in your VKS cluster. See file [cuda-vectoradd-test.yaml](#).

```

# Apply the manifest
kubectl apply -f \
https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/main

# Check the pods
kubectl get pods

# Check the logs of the pod
kubectl logs cuda-vectoradd

# [Optional] Clean the resources
kubectl delete deploy cuda-vectoradd

```

TensorFlow Test

- In this section, we apply a `Deployment` manifest for a TensorFlow GPU application. The purpose of this `Deployment` is to create and manage a single pod running a TensorFlow container that utilizes GPU resource for executing the sum of random values from a normal distribution of size `(100000)` by `(100000)`. For more detail about the manifest, see file [tensorflow-gpu.yaml](#)

```

# Apply the manifest
kubectl apply -f \
https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/main

# Check the pods
kubectl get pods

# Check processes are running using nvidia-smi
kubectl -n gpu-operator exec -it <put-your-nvidia-driver-daemonset-pod> nvidia-smi

# Check the logs of the TensorFlow pod
kubectl logs <put-your-tensorflow-gpu-pod-name> --tail 20

# [Optional] Clean the resources
kubectl delete deploy tensorflow-gpu

```

Configure GPU Sharing

- GPU sharing strategies allow multiple containers to efficiently use your attached GPUs and save running costs. The following tables summarizes the difference between the GPU sharing modes supported by NVIDIA GPUs:

Sharing mode	Supported by VKS	Workload isolation level	Pros	Cons	Suitable for these workloads
Multi-instance GPU (MIG)	✗	Best	<ul style="list-style-type: none"> • Processes are executed in parallel • Full isolation (dedicated memory and compute resources) 	<ul style="list-style-type: none"> • Supported by fewer GPU models • (only Amperes or more recent architectures) • Coarse-grained control over memory and compute resources 	<ul style="list-style-type: none"> • Recommended for workloads running in parallel and that need certain resilience and QoS. • For example, when running AI inference workloads, multi-instance GPU multi-instance GPU

allow
s
multip
le
infere
nce
queri
es to
run
simult
aneou
sly
for
quick
respo
nses,
witho
ut
slowi
ng
each
other
down.

GPU Time- slicing	<input checked="" type="checkbox"/>	None	<ul style="list-style-type: none">• Processes are executed sequentially• Supported by older GPU architectures (Pascal or newer)	<ul style="list-style-type: none">• No resource limits• No memory isolation• Lower performance due to context-switching overhead	<ul style="list-style-type: none">• Recommended for bursty and interactive workloads that have idle periods. These workloads are not cost-
----------------------------------	-------------------------------------	------	--	--	--

- effective with a fully dedicated GPU. By using time-sharing, workloads get quick access to the GPU when they are in active phases.

- GPU time-sharing is optimal for scenarios to avoid idling costly GPUs where full isolation and continuous

- uous
GPU
acces
s
might
not
be
neces
sary,
for
exam
ple,
when
multip
le
users
test
or
protot
ype
workl
oads.

- Workl
oads
that
use
time-
sharin
g
need
to
tolera
te
certai
n
perfor
manc
e and
latenc
y
comp
romis
es.

Multi-process server (MPS)	✓	Medium	<ul style="list-style-type: none"> • Processes are executed parallelly over memory and compute resources allocation. • Fine-grained control over memory and compute resources allocation. 	<ul style="list-style-type: none"> • No error isolation and memory protection for small jobs because MPS maximizes the throughput and concurrent use of a GPU. • MPS allows batch jobs to efficiently process in parallel for small to medium sized workloads. • NVIDIA A 	<ul style="list-style-type: none"> • Recommended for batch processing for small jobs because MPS maximizes the throughput and concurrent use of a GPU.
-----------------------------------	---	--------	---	--	---

MPS is optim al for coop erativ e proces ses acting as a single applic ation. For exam ple, MPI jobs with inter- MPI rank parall elism. With these jobs, each small CUDA proces ss (typic ally MPI ranks) can run concu rrentl y on the GPU to

fully
satur
ate
the
whole
GPU.

- Workloads that use CUDA MPS need to tolerate the [memory protection and error containment](#) limitations.

GPU time-slicing

- VKS uses the built-in timesharing ability provided by the NVIDIA GPU and the software stack. Starting with the [Pascal architecture](#), NVIDIA GPUs support instruction level preemption. When doing context switching between processes running on a GPU, instruction-level preemption ensures every process gets a fair timeslice. GPU time-sharing provides software-level isolation between the workloads in terms of address space isolation, performance isolation, and error isolation.

Configure GPU time-slicing

- To enable GPU time-slicing, you need to configure a `ConfigMap` with the following settings:

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: gpu-sharing-config
data:
  any: |-
    version: v1
    flags:
      migStrategy: none          # Disable MIG, MUST be none in the
    sharing:
      timeSlicing:
        resources:
          - name: nvidia.com/gpu   # Only apply for the node with the
            replicas: 4           # Allow 4 pods to share the GPU, SH

```

- The above manifest allows 4 pods to share the GPU. The `replicas` field specifies the number of pods that can share the GPU. The `replicas` field should be less than the number of GPUs on the node. The `nvidia.com/gpu` label is used to filter the nodes that have GPUs. The `migStrategy` field is set to `none` to disable MIG.
- This configuration will apply to all nodes in the cluster that have the `nvidia.com/gpu` label. To apply the configuration, execute the following command:

```

kubectl -n gpu-operator create -f \
https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/m

```

- And then you need to patch the `ClusterPolicy` to enable GPU time-slicing using the `any` setting:

```

# Patch the ClusterPolicy
kubectl patch clusterpolicies.nvidia.com/cluster-policy \
-n gpu-operator --type merge \
-p '{"spec": {"devicePlugin": {"config": {"name": "gpu-sharing-confi
# Disable DCGM exporter, time-slicing not support DCGM exporter
kubectl patch clusterpolicies.nvidia.com/cluster-policy \
-n gpu-operator --type merge \
-p '{"spec": {"dcgmExporter": {"enabled": false}}}''

```

- Your new configuration will be applied to all nodes in the cluster that have the `nvidia.com/gpu` label.
- The configuration is considered successful if the `ClusterPolicy STATUS` is `ready`.
- Because of the `sharing.timeSlicing.resources.replicas` is set to 4, you can deploy up to 4 pods that share the GPU.
- My cluster has only 1 GPU node, so I can deploy up to 4 pods that share the GPU.

Verify GPU time-slicing

- Until now, we have configured the GPU time-slicing, now we will deploy 5 pods that share the GPU using `Deployment`, because of only 4 pods can share the GPU, the 5th pod will be in `Pending` state. See file [time-slicing-verification.yaml](#).

```
# Apply the manifest
kubectl apply -f \
  https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/master/time-slicing/deployment.yaml

# Check the pods
kubectl get pods

# Check the logs of the TensorFlow pod
kubectl logs <put-your-time-slicing-verification-pod-name> --tail 10

# Get the event of pending pod
kubectl events | grep "FailedScheduling"

# [Optional] Clean the resources
kubectl delete deploy time-slicing-verification
```

Multi-process server (MPS)

- VKS uses NVIDIA's [Multi-Process Service \(MPS\)](#). NVIDIA MPS is an alternative, binary-compatible implementation of the CUDA API designed to transparently enable co-operative multi-process CUDA workloads to run concurrently on a

single GPU device. GPU with NVIDIA MPS provides software-level isolation in terms of resource limits ([active thread percentage](#) and [pinned device memory](#)).

Configure MPS

- To enable GPU MPS, you need to update the previous `ConfigMap` with the following settings:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: gpu-sharing-config
data:
  any-mps: |-  
    version: v1  
    flags:  
      migStrategy: none          # MIG strategy is not used, this fi  
    sharing:  
      mps:                      # Enable MPS for the GPU  
        resources:  
        - name: nvidia.com/gpu   # Only apply for the node with the  
          replicas: 4            # Allow 4 pods to share the GPU
```

- Now let's apply this new `ConfigMap` and then patching the `ClusterPolicy` like the way at the GPU time-slicing section.

```
# Delete the old configmap
kubectl -n gpu-operator delete cm gpu-sharing-config
kubectl -n gpu-operator create -f \
  https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/m

# Patch the ClusterPolicy
kubectl patch clusterpolicies.nvidia.com/cluster-policy \
  -n gpu-operator --type merge \
  -p '{"spec": {"devicePlugin": {"config": {"name": "gpu-sharing-confi

# Disable DCGM exporter, MPS not support DCGM exporter
kubectl patch clusterpolicies.nvidia.com/cluster-policy \
  -n gpu-operator --type merge \
  -p '{"spec": {"dcgmExporter": {"enabled": false}}}''

# Check MPS server is running or not
kubectl -n gpu-operator get pods
```

- Your new configuration will be applied to all nodes in the cluster that have the `nvidia.com/gpu` label.
- The configuration is considered successful if the `ClusterPolicy STATUS` is `ready`.
- Because of the `sharing.mps.resources.replicas` is set to 4, you can deploy up to 4 pods that share the GPU.

Verify MPS

- Until now, we have configured the GPU MPS, now we will deploy 5 pods that share the GPU using `Deployment`, because of only 4 pods can share the GPU, the 5th pod will be in `Pending` state. See file [mps-verification.yaml](#).

```
# Apply the manifest
kubectl apply -f \
  https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/m

# Check the pods
kubectl get pods

# Check the logs of the TensorFlow pod
kubectl logs -l job-name=nbody-sample

# [Optional] Clean the resources
kubectl delete job nbody-sample
```

Applying Multiple Node-Specific Configurations

- An alternative to applying one cluster-wide configuration is to specify **multiple time-slicing configurations** in the `ConfigMap` and to **apply labels** node-by-node to control which configuration is applied to which nodes.
- In this guideline, I add a new RTX-4090 into the cluster.
- This configuration should be create if your cluster have multiple nodes with different GPU models. For example:
 - NodeGroup 1 includes the instance of GPU RTX 2080Ti.
 - NodeGroup 2 includes the instance of GPU RTX 4090.

- And if you want to run multiple GPU sharing strategies in the same cluster, you can apply multiple configurations to the same node by using labels. For example:
 - NodeGroup 1 includes the instance of GPU RTX 2080Ti with 4 pods sharing the GPU using time-slicing.
 - NodeGroup 2 includes the instance of GPU RTX 4090 with 8 pods sharing the GPU using MPS.

Configure Multiple Node-Specific Configurations

- To use this feature, you need to update the previous `ConfigMap` with the following settings:

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: gpu-multi-sharing-config
data:
  rtx-2080ti: |- # Same the name with t
    version: v1
    flags:
      migStrategy: none # MIG strategy is not
    sharing:
      timeSlicing:
        resources:
          - name: nvidia.com/gpu
            replicas: 4 # Allow the node using
  rtx-4090: |- # Same the name with t
    version: v1
    flags:
      migStrategy: none # MIG strategy is not
    sharing:
      mps:
        resources:
          - name: nvidia.com/gpu
            replicas: 8 # Allow the node using

```

- Apply the above configure.

```

kubectl -n gpu-operator create -f \
https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/m

# Patch the ClusterPolicy
kubectl patch clusterpolicies.nvidia.com/cluster-policy \
-n gpu-operator --type merge \
-p '{"spec": {"devicePlugin": {"config": {"name": "gpu-multi-sharing"

# Disable DCGM exporter
kubectl patch clusterpolicies.nvidia.com/cluster-policy \
-n gpu-operator --type merge \
-p '{"spec": {"dcgmExporter": {"enabled": false}}}'}

# Check the ClusterPolicy
kubectl get clusterpolicy

```

- Now, we need to label the node with the name that you specified in the ConfigMap :

```

# Get the node names
kubectl get nodes

# Label the node with the name that you specified in the ConfigMap
kubectl label node <node-name> nvidia.com/device-plugin.config=rtx-208
kubectl label node <node-name> nvidia.com/device-plugin.config=rtx-409

```

Verify Multiple Node-Specific Configurations

- In this example, we will training MNIST model in TensorFlow using the GPU RTX 2080Ti and RTX 4090. The RTX 2080Ti will be shared by 4 pods using time-slicing and the RTX 4090 will be shared by 8 pods using MPS. See file [tensorflow-mnist-sample.yaml](#).

```

# Apply the manifest
kubectl apply -f \
  https://github.com/vngcloud/kubernetes-sample-apps/raw/main/nvidia-g

# Check the pods
kubectl get pods -owide

# Check the logs of the TensorFlow pod
kubectl logs <put-your-favourite-tensorflow-mnist-pod-name> --tail 20

# [Optional] Clean the resources
kubectl delete deploy tensorflow-mnist

```

- The pods are running on the node with the GPU RTX 2080Ti and RTX 4090 within different GPU sharing strategies.

Monitoring GPU Resources

- Monitoring NVIDIA GPU resources in a Kubernetes cluster is essential for ensuring optimal performance, efficient resource utilization, and proactive issue resolution. This overview provides a comprehensive guide to setting up and leveraging Prometheus and the NVIDIA Data Center GPU Manager (DCGM) to monitor GPU resources in a Kubernetes environment.
- Firstly, we need to install **Prometheus Stack** and **Prometheus Adapter** to integrate with the Kubernetes API server. Execute the following command to install the Prometheus Stack and Prometheus Adapter in your VKS cluster:

```

# Install Prometheus Stack using Helm
helm install --wait prometheus-stack \
  --namespace prometheus --create-namespace \
  oci://vcr.vngcloud.vn/81-vks-public/vks-helm-charts/kube-prometheus-
  --version 60.0.2 \
  --set prometheus.prometheusSpec.serviceMonitorSelectorNilUsesHelmVal

# Install and configure Prometheus Adapter using Helm
prometheus_service=$(kubectl get svc -n prometheus -lapp=kube-promethe
helm install --wait prometheus-adapter \
  --namespace prometheus --create-namespace \
  oci://vcr.vngcloud.vn/81-vks-public/vks-helm-charts/prometheus-adapt
  --version 4.10.0 \
  --set prometheus.url=http://${prometheus_service}.prometheus.svc.clu

```

- After the installation is complete, execute the following command to check the resources of Prometheus are running:

```

# Check the resources of Prometheus are running
kubectl -n prometheus get all

```

- Now, we need to enable the DCGM exporter to monitor the GPU resources in the VKS cluster. Execute the following command to enable the DCGM exporter in your VKS cluster:

```

# Enable the DCGM exporter
kubectl patch clusterpolicies.nvidia.com/cluster-policy \
  -n gpu-operator --type merge \
  -p '{"spec": {"dcgmExporter": {"enabled": true}}}' 

# Confirm Prometheus can scrape the DCGM exporter metrics, sometime yo
# (about 1-3 mins) for the DCGM exporter to be ready
kubectl get --raw /apis/custom.metrics.k8s.io/v1beta1 | jq -r . | grep

```

- Let's forward the Prometheus Adapter to your local machine to check the GPU metrics by visit <http://localhost:9090>:

```

# Forward the Prometheus Adapter to your local machine
kubectl -n prometheus \
  port-forward svc/prometheus-stack-kube-prom-prometheus 9090:9090

```

- The following table lists some observable GPU metrics. For details about more metrics, see [Field Identifiers](#).

- Table 1:** Usage

Metric Name	Metric Type	Unit	Description
DCGM_FI_DEV_G PU_UTIL	Gauge	Percentage	GPU usage.
DCGM_FI_DEV_M EM_COPY_UTIL	Gauge	Percentage	Memory usage.
DCGM_FI_DEV_E NC_UTIL	Gauge	Percentage	Encoder usage.
DCGM_FI_DEV_D EC_UTIL	Gauge	Percentage	Decoder usage.

- Table 2:** Memory

Metric Name	Metric Type	Unit	Description
DCGM_FI_DEV_F B_FREE	Gauge	MB	Number of remaining frame buffers. The frame buffer is called VRAM.
DCGM_FI_DEV_F B_USED	Gauge	MB	Number of used frame buffers. The value is the same as the value of memory-usage in the nvidia-smi command.

- Table 3:** Temperature and power

Metric Name	Metric Type	Unit	Description
DCGM_FI_DEV_G PU_TEMP	Gauge	°C	Current GPU temperature of the device.



Autoscaling GPU Resources

- To enable this feature, you **MUST**:
 - Enable **Autoscale** for GPU Nodegroups that you want to scale on the VKS portal.
 - Install [Keda](#) using Helm chart in your VKS cluster.
- In the case you **DO NOT** install Keda in your cluster, **VKS autoscaler** feature will detect the `Pending` pods and scale the GPU Nodegroup automatically. This happens when the number of replicas of the `Deployment` is greater than the number of available GPUs that you configured in the `ConfigMap`.
- If you already installed Keda in your cluster, you can use the `ScaledObject` to scale the GPU Nodegroup based on the metrics that you want. For example, you can scale the GPU Nodegroup based on the GPU usage, memory usage, or any other metrics that you want. For example:

```

apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: scaled-object
spec:
  scaleTargetRef:
    name: scaling-app # The name of the Deployment, MUST in same namespace
    minReplicaCount: 1 # Optional. Default: 0
    maxReplicaCount: 3 # Optional. Default: 100
    triggers: # Will be trigger if either of these triggers is true
    - type: prometheus
      metadata: # prometheus-stack-kube-prom-prometheus
      serverAddress: http://prometheus-stack-kube-prom-prometheus.prometheus.stack.kube.svc.cluster.local:9090
      metricName: engine_active
      query: sum(DCGM_FI_DEV_GPU_UTIL) / count(DCGM_FI_DEV_GPU_UTIL)
      threshold: '0.5' # Scale the GPU Nodegroup when the GPU usage is above 50%
    - type: prometheus
      metadata: # prometheus-stack-kube-prom-prometheus
      serverAddress: http://prometheus-stack-kube-prom-prometheus.prometheus.stack.kube.svc.cluster.local:9090
      metricName: engine_active
      query: sum(DCGM_FI_DEV_MEM_COPY_UTIL) / count(DCGM_FI_DEV_MEM_COPY_UTIL)
      threshold: '0.5' # Scale the GPU Nodegroup when the GPU memory copy utilization is above 50%
  
```

- The above manifest scales the GPU Nodegroup based on the GPU usage and memory usage. The `query` field specifies the query to fetch the metrics from Prometheus. The `threshold` field specifies the threshold value to scale the GPU Nodegroup. The `minReplicaCount` and `maxReplicaCount` fields specify the minimum and maximum number of replicas that the GPU Nodegroup can scale to.
- Now let's install **Keda** in your cluster by executing the below command:

```
helm install --wait kedacore \
--namespace keda --create-namespace \
oci://vcr.vngcloud.vn/81-vks-public/vks-helm-charts/keda \
--version 2.14.2

kubectl -n keda get all
```

- Apply [scaling-app.yaml](#) manifest to generate resources for testing the autoscaling feature. This manifest run 1 pod of CUDA VectorAdd Test and the GPU Nodegroup will be scaled to 3 when the GPU usage is greater than 50%.

```
kubectl apply -f \
https://github.com/vngcloud/kubernetes-sample-apps/raw/main/nvidia-g
```

- Apply [scale-gpu.yaml](#) manifest to create the `ScaleObject` for the above application. This manifest will scale the GPU Nodegroup based on the GPU usage.

```
kubectl apply -f \
https://github.com/vngcloud/kubernetes-sample-apps/raw/main/nvidia-g

kubectl get deploy

# Check the ScaledObject
kubectl get scaledobject
```

- When the `ScaledObject Ready` value is `True`, the GPU Nodegroup will be scaled based on the GPU usage.

Clusters

A cluster in Kubernetes is a collection of one or more virtual machines (VMs) connected together to run containerized applications. Cluster provides a unified environment to deploy, manage, and operate containers at scale.

Prerequisites:

To be able to initialize a **Cluster** and **Deploy a Workload**, you need:

- There is at least 1 **VPC** and 1 **Subnet in ACTIVE state**. If you do not have a VPC or Subnet yet, please create a VPC and Subnet according to the instructions [here](#).
- There is at least 1 **SSH key in ACTIVE state**. If you do not have any SSH key, please create an SSH key according to the instructions [here](#).
- Installed and configured **kubectl** on your device. Please refer here [if](#) you are not sure how to install and use kubectl. In addition, you should not use a kubectl version that is too old, we recommend that you use a kubectl version that is no more than one version different from the cluster version.

Initialize Cluster

To initialize a Cluster, follow the steps below:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select Activate.

Step 3: Wait until we successfully create your VKS account. After Activate successfully, select **Create a Cluster**.

Step 4: At the Cluster initialization screen, we have set up information for the Cluster and a **Default Node Group** for you. You can keep these default values or

adjust the desired parameters for the Cluster and Node Group at Cluster Configuration, Default Node Group Configuration, Plugin.

- Cluster Configuration:
 - Cluster Information:
 - **Cluster Name:** Name for your Cluster. The name can only contain alphanumeric characters (az, AZ, 0-9, '_', '-'). Your input data length must be from 5 to 50. The name must be unique within the Region and VNG Cloud account you are creating the Cluster for.
 - **Kubernetes Version:** The version of Kubernetes that will be used for your Cluster. We recommend that you choose the latest version, unless you need an older version.
 - **Description:** Enter the information you want to note for the Cluster to create a separate mark for easier management in the future.
 - Network Setting:
 - **Network type, Encapsulation Mode** are selected by default by the system and you cannot change them, however you can re-enter **Calico CIDR** parameters (note that IP must be private and can be selected according to the following options (10.0.0.0 - 10.255.0.0 / 172.16.0.0 - 172.24.0.0 / 192.168.0.0)
 - **VPC:** Select an existing VPC that meets K8S requirements to create your Cluster. Before choosing a VPC, we recommend that you familiarize yourself with all VPC requirements and considerations as well as Subnet requirements and considerations. You cannot change which VPC you want to use after creating the Cluster. If no VPC is listed, you need to create one first. For more information, see [Create a VPC](#) .
 - **Subnet:** By default, all available subnets in the VPC specified in the previous field will be randomly selected in the first order, you can choose another Subnet again, but only 1 can be selected.
- Default Node group Configuration:
 - Node Group Information:
 - **Node Group Name :** A memorable name for your Node Group.
 - **Number of nodes:** Enter the number of Worker nodes for your Cluster, note that the number of nodes needs to be greater than or equal to 1 and less than or equal to 100.

- Node Group Automation Setting:
 - **Auto Healing:** By default we will enable the HA feature in your Cluster. When a node or pod fails, Kubernetes will automatically restart or create a new pod to replace it, ensuring your application always operates without interruption.
 - **Auto Scaling:** Enable auto-scaling in your Cluster. Auto scaling helps automatically adjust the number of pods (application deployment units) based on actual usage needs, avoiding wasting resources when demand is low or overloading when demand is high.
 - **Minimum node** : minimum number of nodes that the Cluster needs to have.
 - **Maximum node** : maximum number of nodes that the Cluster can scale to.
 - Node Group upgrade strategy: Node Group upgrade strategy. When you set up a **Node Group Upgrade Strategy** via the **Surge upgrade** method for a Node Group in VKS, the VKS system will update sequentially to upgrade the nodes, in an unspecified order .
 - **Max surge:** limits the number of nodes that can be upgraded simultaneously (the number of new nodes (surge) that can be created at the same time). Default **Max surge = 1** - upgrade only one node at a time. with maxUnavailable
 - **Max unavailable :** limits the number of nodes that cannot be reached during the upgrade (the number of existing nodes that can be offline at the same time). Default **Max unavailable = 0** - ensures all nodes are accessible during the upgrade.
- Node Group Setting:
 - **Image :** By default we provide one type of Image, Ubuntu with containerd.
 - **Instance type :** select the appropriate configuration instance type for the Worker node according to your usage needs.
- Node Group Volume Setting: **Boot Volume Configuration** – Parameters are set by default by the system to help optimize your Cluster
- Node Group Network Setting: You can choose **Public Node Group** or **Private Node Group** depending on your Cluster usage needs.

- Node Group Security Setting: You can choose **Security Group** and **SSH Key** for your Node Group.
 - Node Group Metadata Setting: You can enter the corresponding **Metadata for the Node Group**.
- Plugins
 - **Enable BlockStore Persistent Disk CSI Driver** : enable us to automatically install CSI Controller on your Cluster.
 - **Enable vLB Native Integration Driver** : enable us to automatically install LB Controller on your Cluster.

Step 5: Select **Create Kubernetes cluster**. Please wait a few minutes for us to initialize your Cluster, the Cluster's status is now **Creating** .

Step 6: When the Cluster status is **Active** , you can view Cluster information and Node Group information by selecting Cluster Name in the **Name** column .

Download the Kube Config file

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select the **Kubernetes Cluster** menu.

Step 3: In the successfully created **Cluster** , select the icon and select **Download Config File**.

Step 4: The config file will be saved to your computer, now you can use Kubectl to manage your Cluster on your personal device.

Delete a Cluster



Attention:

When you no longer need to use a Kubernetes Cluster, you should delete the resources associated with that cluster so you don't incur any unnecessary costs. When deleting a Kubernetes Cluster the following resources will be deleted:

- Control Plane Resource of the Cluster.
- All nodes in the Cluster (VM)
- Which Pods are all running on the nodes.
- The default Security Group is created for that Cluster.
- The default Load Balancer is created for that Cluster.
- ETCD.

The system may not delete the following resources:

- The Load Balancer is integrated into the Cluster by you.
- Persistent Volume is integrated into the Cluster by you.

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select the **Kubernetes Cluster** menu.

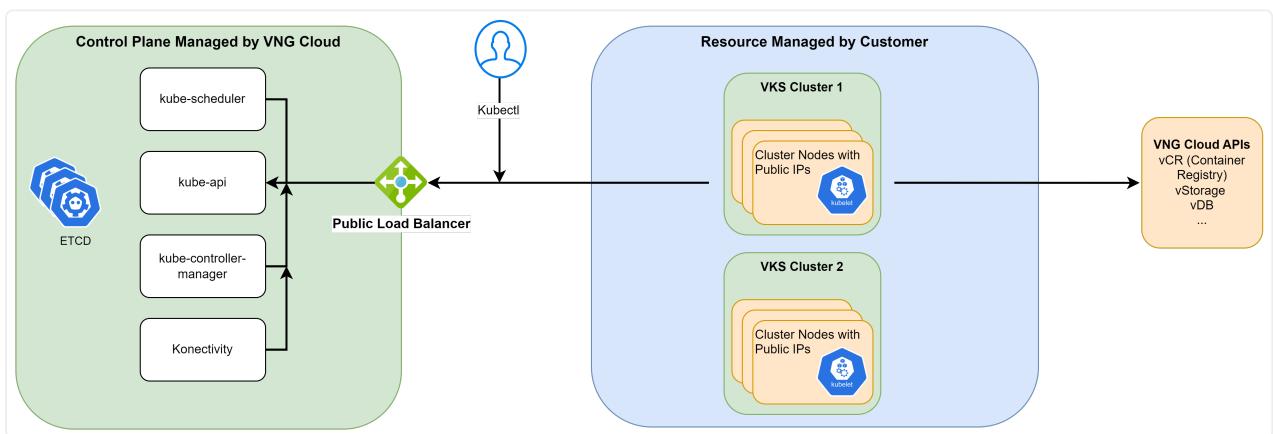
Step 3: In the successfully created **Cluster** , select the cluster you want to delete and select **Delete**

Step 4: Select **Delete** to completely delete your Cluster.

Public Cluster and Private Cluster

Below are the current concepts being provided to you by VKS:

1. Public Cluster



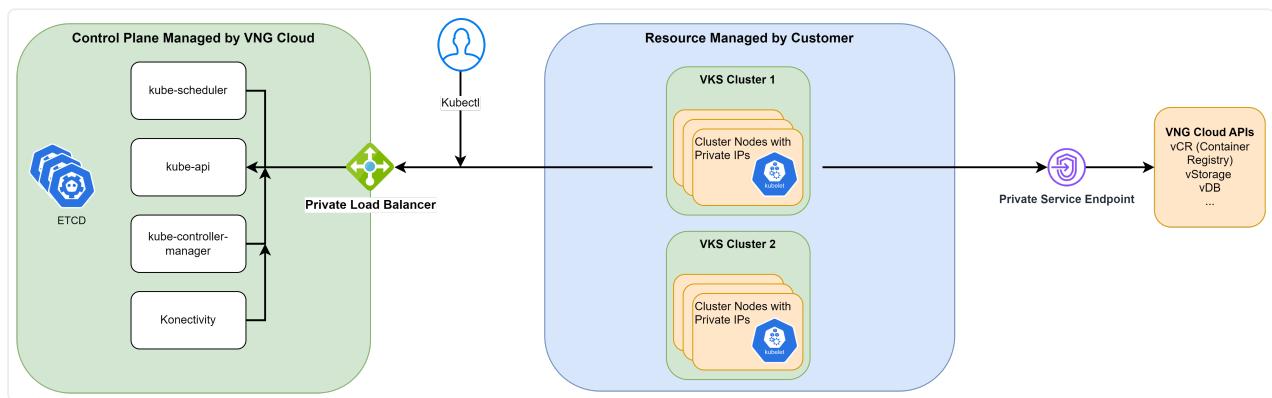
When you create a **Public Cluster with Public Node Group**, the VKS system will:

- Create a VM with Floating IP (ie Public IP). Now these VMs (Nodes) can directly join the K8S cluster through this Public IP. By using Public Cluster and Public Node Group, you can easily create Kubernetes clusters and expose services without using Load Balancer. This will contribute to cost savings for your cluster.

When you create a **Public Cluster with a Private Node Group**, the VKS system will:

- Create VM without Floating IP (ie without Public IP). At this time, these VMs (Nodes) cannot join the K8S cluster directly. In order for these VMs to join the K8S cluster, you need to use a NAT Gateway (**NATGW**). **NATGW** acts as a relay station, allowing VMs to connect to the K8S cluster without needing a Public IP. With VNG Cloud, we recommend you use Pfsense or Palo Alto as a NATGW for your Cluster. Pfsense will help you manage incoming and outgoing network traffic (inbound and outbound traffic) effectively, ensuring network security and access management. Besides, using Private Node Group will help you control applications in the cluster more securely, specifically you can limit control plane access rights through the Whitelist IP feature.

2. Private Cluster



When you create a **Public Cluster with Public/Private Node Group**, the VKS system will:

- To enhance the security of your cluster, we have introduced the private cluster model. The Private Cluster feature helps make your K8S cluster as secure as possible, all connections are completely private from the connection between nodes to the control plane, the connection from the client to the control plane, or the connection from nodes to products. Other services in VNG Cloud such as: vStorage, vCR, vMonitor, VNGCloud APIs,...Private Cluster is the ideal choice for **services that require strict access control, ensuring compliance with security regulations and data privacy**.

3. Comparison between using Public Cluster and Private Cluster

Below is a comparison table between creating and using Public Cluster and Private Cluster on the VKS system:

Criteria	Public Cluster	Private Cluster
Connect	Use Public IP addresses to communicate between nodes and control plane, between clients and control plane, between nodes and	Use Private IP addresses to communicate between nodes and control plane, between clients and control plane, between nodes and

	other services in VNG Cloud.	other services in VNG Cloud.
Security	Medium security since connections use Public IP.	Higher security with all connections private and limited access.
Access management	More difficult to control, access can be managed through the Whitelist feature	Strict access control, all connections are within VNG Cloud's private network, thereby minimizing the risk of external network attacks.
Scalability (AutoScaling)	Easily scalable through Auto Scaling feature .	Easily scalable through Auto Scaling feature .
AutoHealing	Automatically detect errors and restart the node (Auto Healing)	Automatically detect errors and restart the node (Auto Healing)
Accessibility from outside	Easy access from anywhere with internet.	Access from outside must be through other security solutions.
Configuration and deployment	Simpler because it does not require setting up an internal network.	More complex, requires private and secure network configuration.
Cost	Usually lower because there is no need to set up a complex security infrastructure.	Higher cost due to additional security and management components required. Specifically, when using a private cluster, you need to pay for 4 automatically created private service endpoints to connect to services on VNG Cloud.
Flexibility	High, easy to change and access services.	More flexible in applications that require security, but less flexible for applications that require external access.

Therefore:



- **Public Cluster** : Suitable for applications that do not require high security and need flexibility and access from multiple locations. Easy to deploy and manage but has higher security risks.
- **Private Cluster** : Suitable for applications that require high security, strictly complying with security and privacy regulations. Provides stable and secure connectivity, but requires more complex configuration and management, as well as higher costs.

Upgrading Control Plane Version

Currently, our VKS system has supported you to upgrade Control Plane Version, you can:

- Upgrade to a newer **Minor Version** (e.g. 1.24 to 1.25)
- Upgrade to a newer **Patch Version (for example: 1.24.2-VKS.100 to 1.24.5-VKS.200)**

To upgrade the Control Plane version, you can follow these instructions:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select the Kubernetes Cluster menu.

Step 3: Select the icon and select **Upgrade control plane version** to upgrade the control plane version.

Step 4: You can select a new version for the control plane. The new version needs to be valid and compatible with the current version of the cluster. Specifically: you can choose:

- Upgrade to a newer **Minor Version** (e.g. 1.24 to 1.25)
- Upgrade to a newer **Patch Version (for example: 1.24.2-VKS.100 to 1.24.5-VKS.200)**

Step 4: The VKS system will upgrade the **Control Plane** components of **the Cluster** to the new version. After the upgrade is complete, the Cluster state returns to **ACTIVE** .



Attention:

- Upgrading Control Plane Version is optional and independent of upgrading Node Group Version. However, Control Plane Version and Node Group Version in the same Cluster cannot differ by more than 1 minor version. Besides, the VKS system automatically upgrades

Control Plane Version when the current K8S Version used for your Cluster exceeds the supplier's support period.

- During the Control Plane Version upgrade, you cannot perform other actions on your Cluster.
- Below are a few notes before, during and after the upgrade process, please refer to:

Before getting into work:

- Back up data: You should back up cluster data before upgrading to ensure safety in case the upgrade fails.
- Check the current version: Visit Releases for a list of supported versions. Select a new version that is valid and compatible with the current version of the cluster.
- Ensure cluster availability: Cluster must be in active state (ACTIVE) and all nodes must be HEALTHY.
- Stop running tasks: Stop running tasks on the cluster to avoid affecting the upgrade process.

While doing:

- Monitor cluster status: Monitor cluster status during the upgrade process. The cluster status will change to UPDATING and after completion will return to ACTIVE.
- Check system log: Check the system log for any errors or warnings during the upgrade process.

After implementation:

- Check cluster availability: Confirm that the cluster has been upgraded successfully and all nodes are operating normally.
- Test applications: Test applications running on the cluster to ensure they work properly after the upgrade.

Note:

- Upgrading Control Plane Version may take some time depending on the size and complexity of the cluster.

- In some rare cases, the Control Plane Version upgrade may fail. If this happens, the VKS system will automatically rollback the cluster to the current version.

Whitelist

•

Overview

The IP Whitelist feature on VKS's Private Node Group mode allows you to only allow specific IP addresses to connect to your Cluster. This helps increase security for applications and sensitive data by restricting access from unknown sources.

Benefit

- **Enhanced security:** IP Whitelist helps protect your data and applications from potential threats on public networks, such as cyberattacks and data breaches.
- **Minimize risk:** By restricting access to sensitive nodes, Whitelist IP helps minimize the risk of spreading a data breach to other parts of your network.
- **Greater control:** Whitelist IP allows you to tightly control access to your nodes, ensuring only authorized users and applications can access.

Recommendations for Using Whitelist in Cluster Models:

1. Public Cluster Only Includes Public Node Group

- **Recommendation :** Not recommended to use whitelist.
- If you need to use Whitelist IP for security, please allow vServer's IP Range Public list according to the following list:

Copy

103.245.249.0/24
103.245.251.0/24
116.118.95.0/24
58.84.1.0/24
58.84.2.0/24
61.28.226.0/24
61.28.227.0/24
61.28.229.0/24
61.28.230.0/24
61.28.231.0/24
180.93.182.0/24
61.28.233.0/24
61.28.235.0/24
61.28.236.0/24
61.28.238.0/24
180.93.183.0/24

2. Public Cluster Includes Private Node Group Going Through NAT Gateway (Pfsense, PaloAlto)

- **Recommendation :** Can use whitelist feature.
- Need to whitelist additional IP of NAT Gateway.

3. Private Cluster Includes Public Node Group or Private Node Group

- **Recommendation:** Can use whitelist feature.

Edit Whitelist

To use the IP Whitelist feature on Private Node Group mode, you need to perform the following steps:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>



Step 2: At the Overview screen , select the **Kubernetes Cluster** menu.

Step 3: Select the icon and select **Edit Whitelist** or **select the Edit** icon when viewing details of a Cluster to add a Whitelist to your Cluster.

Step 4: Now, the **Edit Whitelist** screen displays, you can enter the IP address you want to allow access to the Cluster then select **Add** .

Step 5: Repeat step 4 if you want to add more **Whitelist IPs** to your Cluster. You can also select **Delete** to delete the Whitelist IP you added previously.

Step 6: Select **Save** to save the information or **Cancel** to cancel saving these parameters.

Stop POC

Before learning how to Stop POC for your resources on VKS, you should clearly understand the concepts and actions you can take on POC resources. For more details, refer [here](#).

Before your POC wallet for Cluster expires, you have two main options:

- Delete this POC Cluster and recreate another normal Cluster.
- Perform Stop POC to renew the POC Cluster into a normal Cluster.

To continue using the resource that just stopped POC as a normal resource (for the purpose of keeping the configuration intact), the user can do:

Step 1: Access [VKS Portal](#), select the Cluster where you want to Stop POC.

Step 2: Select the **Stop POC** button on the top right corner of the screen.

The screenshot shows the VNG Cloud VKS Portal interface. On the left, there's a sidebar with 'VKS' selected. The main area displays a 'Kubernetes cluster / Kubernetes cluster detail' page. At the top, it shows the cluster status as 'ACTIVE' and was created at '03/07/2024 11:23:18'. To the right of the status, there's a 'Stop POC' button with a red arrow pointing to it. Below the status, there are four summary boxes: '1 Total Node Groups', '2 Total Nodes', '1 Healthy/Active Nodes', and '1 Unhealthy Nodes'. Underneath these, there are two sections: 'General Information' and 'Detail Information'. The 'General Information' section contains details like ID, Version, Description, Created at, and Whitelist. The 'Detail Information' section includes tabs for 'Node group', 'Volume', 'Load balancer', and 'Event history', with an 'Add Node Group' button at the bottom.

Step 3: At this time, the screen displays a list of all Servers and Volumes in the Cluster (**including the Boot Volume and PVC that you attached to the node in your Cluster**) that have POC status. You can check the information then select **Stop POC**

Step 4 : Proceed to pay for resources with real money, you can select **the desired usage cycle, turn on and off Auto-renew, enter Coupon if available and select Continue to make Resource Payment**

Resource	Unit Price	Period	Price
vserver - instance ins-6e518279-ac75-41c4-9077-61bed1abf597	696,300 VND / month	<input checked="" type="checkbox"/> Auto-renew 1 month	696,300 VND
vserver - instance ins-d0ac32b2-bb91-4242-be01-0884fd2b1d2a	696,300 VND / month	<input checked="" type="checkbox"/> Auto-renew 1 month	696,300 VND
vserver - volume vol-7f24b175-a1d5-4c75-b8cb-8ae2d4a6b2b0	35,200 VND / month	<input checked="" type="checkbox"/> Auto-renew 1 month	35,200 VND

Coupon Enter the code Apply

Total (3 items) **1,427,800 VND** (Saved 1,427,800 VND)

CONTINUE

Step 5 : Make payment using credit balance or other forms of payment if available.

UNG CLOUD

AVAILABLE
5,799,868 credits

Cancel and go back
Check out > **Order Summary**

3 ITEMS

Resource	Period	Price
vserver - instance ins-6e518279-ac75-41c4-9077-61bed1abf597	1 month Until 02/08/2024 17:55	696,300 VND
vserver - instance ins-d0ac32b2-bb91-4242-be01-0884fd2b1d2a	1 month Until 02/08/2024 17:55	696,300 VND
vserver - volume vol-7f24b175-a1d5-4c75-b8cb-8ae2d4a6b2b0	1 month Until 02/08/2024 17:55	35,200 VND

ORDER SUMMARY

Original price	2,855,600 VND
Item discount	1,427,800 VND
Total	1,427,800 VND

I agree with the [Terms & Conditions](#)

Use 1,427,800 credits ⓘ

You'll pay for this order with 1,427,800 credits.

PAY

© 2022, VI Na Data Information Technology JSC
Sales@vngcloud.vn - 180001549

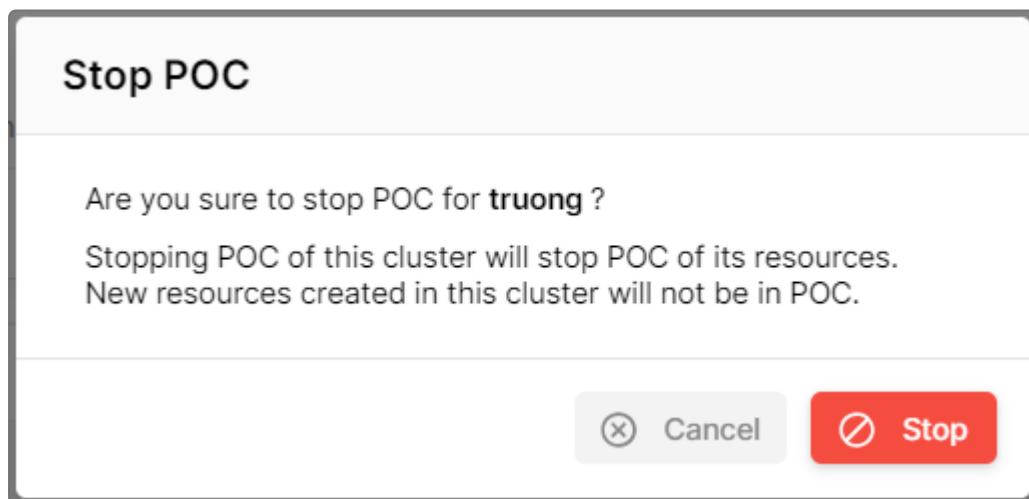
Helios Building, Quang Trung Software City
Tan Chanh Hiep Ward, District 12, HCMC

Tax code: 0304851362
Department of Planning and Investment of HCMC

[Terms & Conditions](#)

ISO 27001 ISO 22381 ISO 9001 SLA 99.99% PCI DSS

To ensure VKS works correctly, Stop POC implementation needs to be performed on VKS Portal instead of doing it individually on each resource server or volume on vServer Portal. If you have previously performed an individual POC stop for each resource on vServer Portal, you still need to perform a POC Stop for the Cluster at VKS Portal. At this time, the screen will display as follows. Please press **Stop** to turn off the POC option for your Cluster.



(i) Attention:

- After stopping POC on VKS, the "Stop POC" button will continue to display if there are still resources that have not been Stop POC after doing so on VKS Portal. You can continue to select and execute Stop POC until all resources are converted to real resources.

- Currently, VKS only applies payment by POC for Server and Volume and has not yet applied payment for Load Balancer and Snapshot by POC. Therefore, you do not need to perform Stop POC for these two resource types Load Balancer and Snapshot.c

Node Groups

Node Group is an important concept in Kubernetes, used to manage groups of **nodes** (VMs) with the same configuration in a cluster. For a Node Group, you can:

Create a Node Group

To initialize a Node Group, follow the steps below:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the previously created Cluster, select **Create a Node group**.

Step 3: At the Node Group initialization screen, we have set up information for your Node Group. You can keep these default values or adjust the desired parameters for your Node Group at:

- Node Group Information:
 - **Node Group Name** : A memorable name for your Node Group.
 - **Number of nodes**: Enter the number of Worker nodes for your Cluster, note that the number of nodes needs to be greater than or equal to 1 and less than or equal to 100.
- Node Group Automation Setting:
 - **Auto Healing**: By default we will enable the HA feature in your Cluster. When a node or pod fails, Kubernetes will automatically restart or create a new pod to replace it, ensuring your application always operates without interruption.
 - **Auto Scaling**: Enable auto-scaling in your Cluster. Auto scaling helps automatically adjust the number of pods (application deployment units) based on actual usage needs, avoiding wasting resources when demand is low or overloading when demand is high.
 - **Minimum node** : minimum number of nodes that the Cluster needs to have.

- **Maximum node** : maximum number of nodes that the Cluster can scale to.
- Node Group upgrade strategy: Node Group upgrade strategy. When you set up a **Node Group Upgrade Strategy** via the **Surge upgrade** method for a Node Group in VKS, the VKS system will update sequentially to upgrade the nodes, in an unspecified order .
 - **Max surge**: limits the number of nodes that can be upgraded simultaneously (the number of new nodes (surge) that can be created at the same time). Default **Max surge = 1** - upgrade only one node at a time. with maxUnavailable
 - **Max unavailable** : limits the number of nodes that cannot be reached during the upgrade (the number of existing nodes that can be offline at the same time). Default **Max unavailable = 0** - ensures all nodes are accessible during the upgrade.
- Node Group Setting:
 - **Image** : By default we provide one type of Image, Ubuntu with containerd.
 - **Instance type** : select the appropriate configuration instance type for the Worker node according to your usage needs.
- Node Group Volume Setting: **Boot Volume Configuration** – Parameters are set by default by the system to help optimize your Cluster
- Node Group Network Setting: You can choose **Public Node Group** or **Private Node Group** depending on your Cluster usage needs.
- Node Group Security Setting: You can choose **Security Group and SSH Key** for your Node Group.
- Node Group Metadata Setting: You can enter the corresponding **Metadata for the Node Group**.

Step 5: Select **Create Node Group**. Please wait a few minutes for us to initialize your Node Group. The status of the Node Group is currently **Creating** .

Step 6: When the Node Group status is **Active** , you can view Node Group information by selecting **Node Group Name** on the main screen.

Edit a Node Group

For Node Group, you can edit the parameters: **Number of Nodes, Auto Scaling, Upgrade Strategy, Security Group** in each separate edit . Specifically, you can follow these steps:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: In the previously created Cluster, select **the Cluster you want to edit the Node group.**

Step 3: On the screen containing the list of existing Node Groups, in the Node Group you want to edit, select one of the options:

- **Resize feature :** you can change
 - Number of nodes: Enter the number of Worker nodes for your Cluster, note that the number of nodes needs to be greater than or equal to 1 and less than or equal to 100.
- **Edit Auto Scaling feature :** you can change
 - Auto Scaling: Enable auto-scaling in your Cluster. Auto scaling helps automatically adjust the number of pods (application deployment units) based on actual usage needs, avoiding wasting resources when demand is low or overloading when demand is high.
 - Minimum node: minimum number of nodes that the Cluster needs to have.
 - Maximum node: maximum number of nodes that the Cluster can scale to.
- **Edit Upgrade Strategy feature :** you can change
 - Node Group upgrade strategy: Node Group upgrade strategy. When you set up a Node Group Upgrade Strategy via the Surge upgrade method for a Node Group in VKS, the VKS system will update sequentially to upgrade the nodes, in an unspecified order .
 - Max surge: limits the number of nodes that can be upgraded simultaneously (the number of new nodes (surge) that can be created at

the same time). Default Max surge = 1 - upgrade only one node at a time. with maxUnavailable

- Max unavailable: limits the number of nodes that cannot be reached during the upgrade (the number of existing nodes that can be offline at the same time). Default Max unavailable = 0 - ensures all nodes are accessible during the upgrade.

- **Edit Security Group** feature : you can change

- Node Group Security Setting: You can choose Security Group and SSH Key for your Node Group.

Delete a Node Group

Attention:

When you no longer need to use the Node Group, delete them to save costs. When deleting a Node Group, the following resources will be deleted:

- All nodes included in the Node Group (VM)

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select the **Kubernetes Cluster** menu.

Step 3: In the successfully created **Cluster** , select the **Node Group** you want to delete and select **Delete**.

Step 4: Select **Delete** to completely delete your **Node Group** .

Auto Healing

•

Overview

On the VKS system, the Auto Healing feature is applied to each Node Group and is always on. Enabling self-healing in Kubernetes provides many important benefits, helping to ensure high availability and reliability for your applications.

The Auto Healing feature has the following highlights:

- **Automatic error detection:** Kubernetes can automatically detect failed or problematic nodes through monitoring the status of the nodes. Some signs that a node is failing include: node reporting "NotReady" status, node unable to be pinged, node experiencing hardware failure, etc.
- **Automatically restart the node:** When a node detects an error, Kubernetes will automatically restart the node. Restarting the node can help fix temporary errors and return the node to normal operation.
- **Minimize manual intervention:** Auto Healing helps minimize manual intervention by system administrators, saving time and effort.
- **Improved performance:** Auto Healing helps improve system performance by ensuring that nodes always operate normally.

Mechanism of action

Auto Healing mechanism: VKS system activates auto healing when

- The Node reports the **NotReady** status on consecutive checks at **10 minute** intervals .

If the above conditions are met, the system will immediately perform auto healing. This process is performed in 2 steps:

- **Step 1:** The VKS system drains the node, which means moving all pods running on this NotReady node to other nodes in the node group before removing that node from the node group.
- **Step 2:** The system will recreate a new node with the configuration set up on the node group and join this node to the cluster. If after rebooting, the node still reports a "NotReady" status, the system will continue to reboot the node until the node returns to its normal operating state.



Attention:

- When the system performs Auto Healing, creating a new node may encounter an error if you do not have enough credits or you have run out of quota to create a VM on the vServer system. At this time, every 30 minutes the system will restart the node until the node returns to normal operating status. To avoid the error above, you need to:
 - **Make sure you have enough credits:** If you're a prepaid user, add more credits to your account.
 - **Request a quota increase:** You can request a quota increase for your account [here](#).

Turn on Auto Healing

Currently, the Auto Healing feature is applied to each Node Group and is always **on**. You do not need to manually enable it when initializing the Cluster or Node Group.

Auto Scaling

•

Overview

Auto Scaling for Cluster is a feature in Kubernetes that allows automatically adjusting the size of the cluster, specifically the number of nodes in the cluster to meet usage needs.

The Auto Scaling feature has the following highlights:

1. **Optimize performance:** Auto Scaling allows the cluster to automatically scale resources as needed. When workloads are higher, the cluster automatically creates additional nodes to ensure applications operate at their best performance.
2. **Cost savings:** Auto Scaling allows the cluster to automatically reduce resources when not needed. If workload decreases, the cluster automatically reclaims unused resources to save costs.
3. **Ensure availability:** Auto Scaling helps ensure that the cluster is available to meet demand and avoid resource overload or shortages.
4. **Auto Scaling:** Auto Scaling helps automatically recover from crashes or errors by creating new nodes to replace failed nodes.

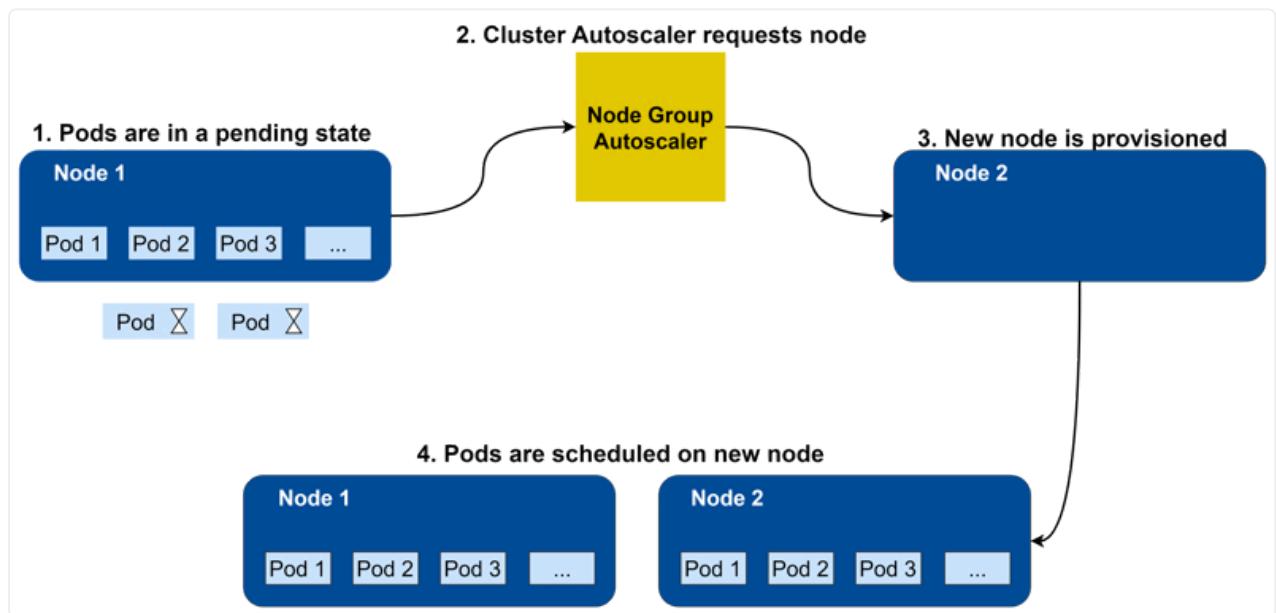
When deploying applications in a cloud environment, using Auto Scaling helps optimize resource usage, improve application availability and performance, and makes cluster management easy and convenient. more effective.

Mechanism of action

Scale up mechanism: the Procuracy system performs scale up when

- Pods cannot be scheduled on any existing node due to lack of resources.
- Adding 1 more node similar to the current node group configuration is useful and can handle this resource shortage problem.

Illustration:



If the above two conditions are met, the system will increase the number of nodes (one or more nodes) to accommodate all unscheduling pods. This process will be done immediately in 2 steps:

- **Step 1:** The VKS system creates a new node according to the current node group configuration.
- **Step 2:** The VKS system will deploy these unscheduling pods to new nodes.



Attention:

- When the system performs Auto Scaling, creating a new node may encounter an error if you do not have enough credits or you have run out of quota to create a VM on the vServer system. To avoid the error above, you need to:
 - Make sure you have enough credits:** If you're a prepaid user, add more credits to your account.
 - Request a quota increase:** You can request a quota increase for your account [here](#).

Scale down mechanism: the Procuracy system performs scale down when

- One or more nodes have continuously low load over a period of time. Specifically, the node has low utilization (availability) including CPU and memory requests of the pod at < 50%.
- All existing pods of that node, can be moved to another node without any problem.

If the above two conditions are met, by default within about 10 minutes, that node will be deleted from the Cluster. This deletion process will include 3 steps:

- Step 1:** The VKS system will mark that node as unschedulable.
- Step 2:** The system moves the entire pod to another node.
- Step 3:** After successfully moving all pods to another node, the VKS system will delete the marked node.

Turn on Auto Scaling

On the VKS system, you can turn on Auto Scaling when:

- **Initialize a Cluster**
- **Create a Node Group**
- **Edit a Node Group**

To enable Auto Scaling for your Kubernetes Cluster please enable the **Enable Auto Scaling option**. When enabling this option you need to enter::

- **Minimum node** : minimum number of nodes that the Cluster must have.
- **Maximum node** : maximum number of nodes that the Cluster can scale to.
- **Node Group upgrade strategy** : Node Group upgrade strategy. When you set up a **Node Group Upgrade Strategy** via the **Surge upgrade** method for a Node Group in VKS, the VKS system will update sequentially to upgrade the nodes, in an unspecified order .
 - **Max surge**: limits the number of nodes that can be upgraded simultaneously (the number of new nodes (surge) that can be created at the same time). Default **Max surge = 1** - upgrade only one node at a time.
 - **Max unavailable** : limits the number of nodes that cannot be reached during the upgrade (the number of existing nodes that can be offline at the same time). Default **Max unavailable = 0** - ensures all nodes are accessible during the upgrade.

For example, as shown below: I have initialized a Node Group with:

- **Number of nodes: 3 nodes**
- **Minimum nodes: 1 nodes**
- **Maximum nodes: 5 nodes**

At this time:

- If one or more pods cannot be scheduled on any node on the cluster due to lack of resources and adding a node similar to this node group configuration can solve the problem, the system will perform scale up. With this setup, the system can scale up to a maximum of **5 nodes** .
- If there is a node with low utilization (availability) at < 50% and all pods of that node can be scheduled on another node, the system will perform scale up. With this setup, the system can scale down to a minimum of **1 node**.

Upgrading Node Group Version

Currently, our VKS system has supported you to upgrade Node Group Version, you can upgrade Node Group Version to:

- **Control Plane Version** (For example upgrade from 1.24 (current Node Group version) to 1.25 (current Control Plane Version), but cannot upgrade to other versions.

To perform a Node Group Version upgrade, you can follow these instructions:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the Overview screen , select the **Kubernetes Cluster menu**. Select a **Cluster** where you want to upgrade **Node Group Version** .

Step 3: Select iconand select **Upgrade Node Group version** to upgrade the node group version.

Step 4: You can select the new version for all Node Groups. The new version needs to be valid and compatible with the current version of the cluster. Specifically: you can choose:

- Upgrade Node Group to the same version as Control Plane Version (for example: 1.24 to 1.25)

Step 5: The VKS system will upgrade all Node Groups to the Control Plane version. After the upgrade is complete, the Node Group status returns to **ACTIVE** .



Attention:

- Upgrading Node Group Version is optional and independent of upgrading Control Plane Version. However, all Node Groups in a Cluster will be upgraded at the same time, as well as Control Plane Version and Node Group Version in the same Cluster cannot differ by more than 1 minor version. Besides, the VKS system automatically

upgrades the Node Group Version when the current K8S Version being used for your Cluster exceeds the supplier's support period.

- During the Node Group Version upgrade, you cannot perform other actions on your Node Group.
- Below are a few notes before, during and after the upgrade process, please refer to:

Before getting into work:

- Check the current version: Visit Releases for a list of supported versions. Select a new version that is valid and compatible with the current version of the cluster.
- Ensure Node Group availability: Node Group must be in active state (ACTIVE) and all nodes must be HEALTHY.
- Stop running tasks: Stop running tasks on the cluster to avoid affecting the upgrade process.

While doing:

- Monitor Node Group status: Monitor Node Group status during the upgrade process. The Node Group status will change to UPDATING and after completion will return to ACTIVE.
- Check system log: Check the system log for any errors or warnings during the upgrade process.

After implementation:

- Check Node Group Availability: Confirm that the Node Group has been upgraded successfully and all nodes are operating normally.
- Test applications: Test applications running on the cluster to ensure they work properly after the upgrade.

Note:

- Upgrading Node Group Version may take some time depending on the size and complexity of the Node Group.
- In some rare cases, upgrading Node Group Version may fail. If this happens, the VKS system will automatically rollback the cluster to the current version.

Label and Taint

•

Label

Labels are an important feature in Kubernetes, used to organize and manage objects effectively. You can assign key-value pairs to Kubernetes objects such as Pod, Node, Service, Deployment, etc. Specifically:

- **Each Label is a key-value pair:** Key is a string of characters used to identify the name of the label. Value is an optional character string that provides detailed information about the label.
- **Keys and values must follow the naming rules:** Keys and values must not contain spaces or special characters other than (-, _, .).
- Label can be used for a variety of purposes, including:
 - Classify objects based on criteria such as environment, version, status, etc
 - Monitor and manage objects in a Kubernetes cluster.

For example:

- `app: nginx` - This label indicates the object is related to the Nginx application.
- `environment: production` - This label indicates that the object belongs to the production environment.
- `version: 1.7.2` - This label indicates the object is related to version 1.7.2.

Create Label

To create a Label for a Node Group, follow these instructions:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the previously created Cluster, select **Create a Node group**.

Step 3: At the Node Group initialization screen, we have set up information for your Node Group. You can keep these default values or adjust the desired parameters for your Node Group. In the **Node Group Metadata Setting section**, you need:

- Enter the key for your label. The key must begin and end with letters or numbers and include the characters az, AZ, 0-9, -, _, . Maximum 253 characters. Alternatively, you can enter the key as a DNS subdomain, for example: example.com/my-app
- Enter the value for this corresponding key.

Step 5: Select **Create Node Group**. Please wait a few minutes for us to initialize your Node Group. The status of the Node Group is currently **Creating**.

Step 6: When the Node Group status is Active, you can view Node Group information by selecting **Node Group Name** on the main screen.

Or you can create Lable through kubectl with the command:

```
kubectl label nodes my-node1 disktype=ssd
```

You can check the newly created label again with the command:

```
kubectl get nodes --show-labels
```

For example the result for this command would be as follows:

NAME	STATUS	ROLES	AGE	VERSION	LABELS
worker0	Ready	<none>	1d	v1.13.0	...,disktype=ssd,kube
worker1	Ready	<none>	1d	v1.13.0	...,kubernetes.io/hos
worker2	Ready	<none>	1d	v1.13.0	...,kubernetes.io/hos

Use Label with nodeSelector

nodeSelector is a parameter used in PodSpec to specify that Pods should only be scheduled on Nodes with a specific label. This is useful when you want to run Pods on Nodes with specific resources or properties.

- Create a `my-pod.yaml` file containing the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  nodeSelector:
    disktype: ssd
    region: hcm03
```

In this example, the Pod `my-pod` is scheduled only on Nodes with label `disktype: ssd` and `region: hcm03`.

- Deploy Pod on your Cluster:

```
kubectl -f apply my-pod.yaml
```

Taint

Taint is an important feature in Kubernetes, serving as a mechanism to tag Nodes and control Pod scheduling on those Nodes. Different from regular Label, Taint is used to specify special properties of Node and execute specific actions when Pod does not meet the conditions defined by Taint. Specifically:

Specifically:

- **Each Taint includes:**
 - Key is a string of characters used to identify the name of the taint.
 - Value is an optional character string that provides detailed information about the taint.
 - Effect:
 - **NoSchedule:** Prevent Pods from having a corresponding Toleration scheduled on the Node.

- **NoExecute:** Allows the Pod to be scheduled on the Node but the Pod will not be executed.
- **PreferNoSchedule:** Kubernetes will try to prioritize not scheduling the Pod to the Node with this Taint.
- **Keys and values must follow the naming rules:** Keys and values must not contain spaces or special characters other than (-, _, .).
- **Toleration:** In order for a Pod to be scheduled and run on a Node with Taint, the Pod needs to have a corresponding Toleration. Toleration is declared in PodSpec using `tolerations` field. For example:

```
tolerations: - key: node.role.kubernetes.io/master effect: NoSchedule
```

- **Relationship between Taint and Toleration:** When Kubernetes schedules a Pod, Kubernetes matches the Node's Taints with the Pod's Toleration. Pods are only scheduled on a Node if there is Toleration for all Taints of that Node.

For example:

- `node.role.kubernetes.io/master:NoSchedule` - prevents regular Pods from being run on this Node.

Create Taints

To create a Taint for a Node Group, follow these instructions:

Step 1: Visit <https://vks.console.vngcloud.vn/overview>

Step 2: At the previously created Cluster, select **Create a Node group**.

Step 3: At the Node Group initialization screen, we have set up information for your Node Group. You can keep these default values or adjust the desired parameters for your Node Group. In the **Node Group Metadata Setting section**, you need:

- Enter the key for your taint. The key must begin and end with letters or numbers and include the characters az, AZ, 0-9, -, _, . Maximum 253 characters.

Alternatively, you can enter the key as a DNS subdomain, for example:
example.com/my-app

- Enter the value for this corresponding key.
- Choose 1 of 3 effect types: **NoSchedule**, **NoExecute**, **PreferNoSchedule**.

Step 5: Select **Create Node Group**. Please wait a few minutes for us to initialize your Node Group. The status of the Node Group is currently **Creating**.

Step 6: When the Node Group status is **Active**, you can view Node Group information by selecting **Node Group Name** on the main screen.

Or you can create Taint through kubectl with the command:

```
kubectl taint node my-node node.role.kubernetes.io/master:NoSchedule.
```

Taint usage example:

Suppose you have a Node `master` used for management purposes and you want to prevent regular Pods from being run on this Node. You can use Taint as follows:

```
kubectl taint node my-master node.role.kubernetes.io/master:NoSchedule
```

In order for Pod to run on Node `master`, the Pod needs to have the corresponding Toleration:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  tolerations:
  - key: node.role.kubernetes.io/master
    effect: NoSchedule
```

Network

•

Working with Application Load Balancer (ALB)

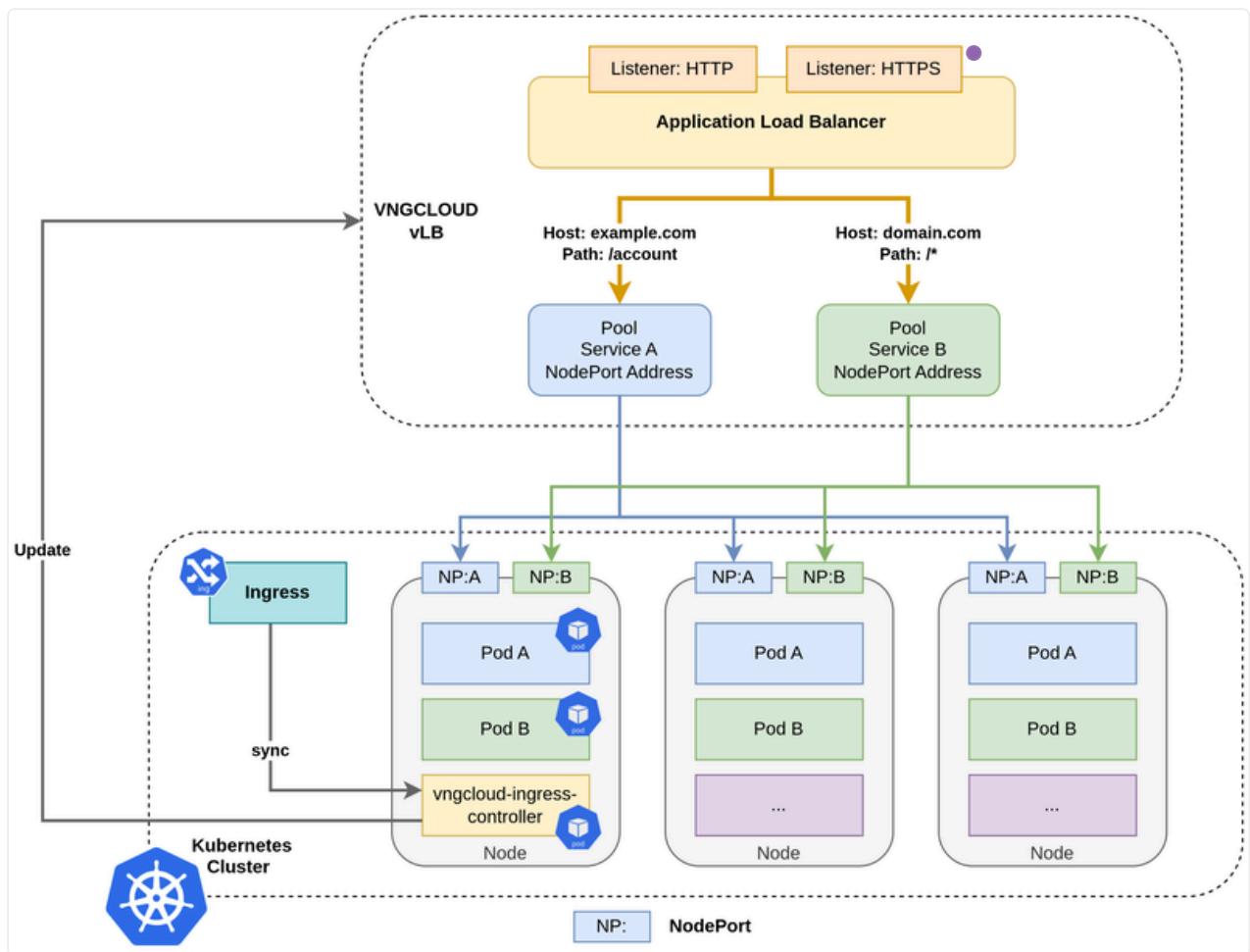
•

Overview

What is ALB?

- **Application Load Balancer (ALB)** is a tool in network and server infrastructure used to distribute network traffic to multiple servers or virtual machines to improve the performance and availability of applications. ALB operates at the application layer, allowing traffic distribution based on many factors such as request type, server state, and load distribution algorithm. ALB provides advanced routing capabilities, allowing traffic to be directed based on Host or Path Header. It also supports session persistence, which helps maintain user sessions to the same server. This is useful for applications that require consistency in user interactions. For more information about ALB, please refer to [How it works (ALB)]

Model:



In addition to the basic components of a K8S cluster and an ALB that you already know, in this model we use:

- **Ingress:** is a resource in Kubernetes that is configured to make Services accessible from outside the k8s cluster via URL, and can also load balance traffic, support SSL/TLS connections and provide virtual hosting based on names. An Ingress does not arbitrarily expose protocols other than HTTP and HTTPS. Ingress acts as a single entry point for HTTP and HTTPS requests from outside the cluster to internal services. Traffic routing is controlled by rules defined in the Ingress resource (Ingress Yaml File). An Ingress is managed by VNGCloud Ingress Controller: is an application that runs in the cluster and manages Ingress resources based on the Ingress Yaml File defined by the customer

Ingress for an Application Load Balancer

•

In order for the Ingress resource (Ingress Yaml file) to work, the cluster must have a running VNGCloud Ingress Controller. Unlike other Controller types that run as part of **kube-controller-manager** . VNGCloud Ingress Controller is not automatically started with the cluster. Please follow the instructions below to install VNGCloud Ingress Controller as well as work with Ingress Yaml files.

Prepare

- Create a Kubernetes cluster on VNGCloud, or use an existing cluster. Note: make sure you have downloaded the cluster configuration file once the cluster has been successfully initialized and accessed your cluster.
- Create or use a **service account** created on IAM and attach policy: **vLBFullAccess** , **vServerFullAccess** . To create a service account, go here [and](#) follow these steps:
 - Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account
 - Find and select **Policy: vLBFullAccess and Policy: vServerFullAccess** , then click " **Create a Service Account** " to create Service Account, Policy: vLBFullAccess and Policy: vServerFullAccess created by VNG Cloud, you cannot delete these policies.
 - After successful creation, you need to save **the Client_ID** and **Secret_Key** of the Service Account to perform the next step.
- Change **the Security Group** information to allow ALBs to connect to Nodes in your Node Group. You need to change them on vServer Portal when:
 - The Security Group attached to your **Cluster/Node Group is different from the default parameters** we created.
 - You need **to change the security level** for your Cluster or you need to **open more ports** for specific services to operate on the Cluster. Details information here [.](#)

Create Service Account and install VNGCloud Ingress Controller

Attention:

When you initialize the Cluster according to the instructions above, if you have not enabled the **Enable vLB Native Integration Driver** option , by default we will not pre-install this plugin into your Cluster. You need to manually create Service Account and install VNGCloud Ingress Controller according to the instructions below. If you have enabled the **Enable vLB Native Integration Driver** option , then we have pre-installed this plugin into your Cluster, skip the Service Account Initialization step, install VNGCloud Ingress Controller and continue following the instructions from Deploy once. Workload.

✓ Create Service Account and install VNGCloud Ingress Controller

Initialize Service Account

- Create or use a **service account** created on IAM and attach policy: **vLBFULLAccess** , **vServerFullAccess** . To create a service account, go here [and](#) follow these steps:
 - Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account
 - Find and select **Policy: vLBFULLAccess and Policy: vServerFullAccess** , then click " **Create a Service Account** " to create Service Account, Policy: vLBFULLAccess and Policy: vServerFullAccess created by VNG Cloud, you cannot delete these policies.
 - After successful creation, you need to save **the Client_ID** and **Secret_Key** of the Service Account to perform the next step.

Install VNGCloud Ingress Controller

- Install Helm version 3.0 or higher. Refer to <https://helm.sh/docs/intro/install/> for instructions on how to install.
- Add this repo to your cluster via the command:

```
helm repo add vks-helm-charts https://vngcloud.github.io/vks-he:  
helm repo update
```

- Replace your K8S cluster's ClientID, Client Secret, and ClusterID information and continue running:

```
helm install vngcloud-ingress-controller vks-helm-charts/vngclo:  
--namespace kube-system \  
--set cloudConfig.global.clientID= <Lấy ClientID của Service /  
--set cloudConfig.global.clientSecret= <Lấy ClientSecret của S  
--set cluster.clusterID= <Lấy Cluster ID của cluster mà bạn đ
```

- After the installation is complete, check the status of vngcloud-ingress-controller pods:

```
kubectl get pods -n kube-system | grep vngcloud-ingress-control
```

For example, in the image below you have successfully installed vngcloud-controller-manager:

NAME	READY	STATUS	RES
vngcloud-ingress-controller-0	1/1	Running	0

Deploy a Workload

The following is a guide for you to deploy the nginx service on Kubernetes.

Step 1 : Create Deployment for Nginx app.

- Create **nginx-service-lb7.yaml** file with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
  spec:
    containers:
      - name: nginx
        image: nginx:1.19.1
        ports:
          - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- Deploy This deployment equals:

```
kubectl apply -f nginx-service-lb7.yaml
```

Step 2: Check the Deployment and Service information just deployed

- Run the following command to test **Deployment**

```
kubectl get svc,deploy,pod -owide
```

- If the results are returned as below, it means you have deployed Deployment successfully.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	
service/nginx-service	NodePort	10.96.25.133	<none>	80:32572	
NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONT
deployment.apps/nginx-app	1/1	1	1	2m50s	nginx
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx-app-7f45b65946-6wlgw	1/1	Running	0	2m49s	172

Step 3: Create Ingress Resource

1. If you do not have an Application Load Balancer previously created on the vLB system.

Now, when creating an Ingress, leave the Load Balancer ID information blank at the vks.vngcloud.vn/load-balancer-id annotation .

- For example, suppose you have deployed a service named nginx-service. At this point, you can create the **nginx-ingress.yaml** file as follows:

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
spec:
  ingressClassName: "vngcloud"
  defaultBackend:
    service:
      name: nginx-service
      port:
        number: 80
  rules:
    - http:
        paths:
          - path: /4
            pathType: Exact
            backend:
              service:
                name: nginx-service
                port:
                  number: 80

```

- Run the following command to deploy Ingress

```
kubectl apply -f nginx-ingress.yaml
```

Once you have deployed Ingress, we will automatically create an ALB on your cluster. This ALB will be displayed on vLB Portal, details can be accessed here [.](#)
This ALB will have default information:

Ingredient	Quantity	Properties
ALB Package	first	VNG ALB_Small
Listener	2	<ul style="list-style-type: none"> 1 listener with HTTP protocol and port 80 1 listener with HTTPS protocol and port 443
Pool	first	<ul style="list-style-type: none"> 1 pool default HTTP protocol and ROUND ROBIN algorithm

Health Check

first

- Use TCP to check health of members.

For example:

The screenshot shows the VNG Cloud console interface for managing load balancers. The left sidebar navigation includes options like vServer, Region HCM03, BlockStore, Volumes, Images, Backups, Snapshots, Load Balancing, Load Balancers (selected), LB Packages, Certificates, Container, Kubernetes Clusters, Persistent Volumes, Container Registry, AutoScale, AutoScales, and Billing. The main content area displays a table of load balancers with the following data:

Name	Status	End point	Schema	Type	Package	Created at	Action
lb-1fcaeb35-15af-4b87-9e83-0f3209638fd9 vks-k8s-6d2acb-default-nginx-ingr-897e5	ACTIVE	180.93.181.129	Internet	Application	ALB_Small	21/04/2024 20:08:28	⋮
lb-8ea77772-0ffb-442c-81b7-f03e44dd9742 vks-k8s-6d2acb-default-nginx-serv-908c5	ACTIVE	180.93.181.20	Internet	Network	NLB_Small	21/04/2024 19:52:22	⋮

Total: 2

Bottom footer: © 2022, VI NA Data IT JSC. Helios Building, Lot 3, Road #3, Quang Trung Software City, Tan Chanh Hiep Ward, District 12, HCMC. Tax code: 0304851362. Dept. of Planning and Investment of HCMC on 28/02/2007. Terms Privacy Policy ISO 27001 ISO 22381 ISO 9001 SLA Payless.



Attention:

- Currently Ingress only supports TLS port 443 and is the termination point for TLS (TLS termination). TLS Secret must contain fields with key names `tls.crt` and `tls.key`, which are the certificate and private key to use for TLS. If you want to use a Certificate for a host, please upload the Certificate according to the instructions at [Upload a certificate] and use them as an annotation. For example:

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    # kubernetes.io/ingress.class: "vngcloud" # this annotation is deprecated
    vks.vngcloud.vn/certificate-ids: "secret-a6d20ec6-f3e5-499a-981b-b148"
spec:
  ingressClassName: "vngcloud"
  defaultBackend:
    service:
      name: apache-service
      port:
        number: 80
  tls:
    - hosts:
      - host.example.com
  rules:
    - host: host.example.com
      http:
        paths:
          - path: /4
            pathType: Exact
            backend:
              service:
                name: nginx-service
                port:
                  number: 80

```

2.If you already have a previously initialized Application Load Balancer on the vLB system and you want to reuse the ALB for your cluster.

Now, when creating an Ingress, enter the Load Balancer ID information into the **vks.vngcloud.vn/load-balancer-id annotation**. For example, in this case I reused the ALB with ID = lb-2b9d8974-3760-4d60-8203-9671f229fb96:

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    # kubernetes.io/ingress.class: "vngcloud" # this annotation is deprecated
    vks.vngcloud.vn/load-balancer-id: "lb-2b9d8974-3760-4d60-8203-9671f22"
    vks.vngcloud.vn/certificate-ids: "secret-a6d20ec6-f3e5-499a-981b-b148"
spec:
  ingressClassName: "vngcloud"
  defaultBackend:
    service:
      name: apache-service
      port:
        number: 80
  tls:
    - hosts:
      - host.example.com
  rules:
    - host: host.example.com
      http:
        paths:
          - path: /4
            pathType: Exact
            backend:
              service:
                name: nginx-service
                port:
                  number: 80

```

- After you have created ingress according to the instructions at [Ingress for an Application Load Balancer](#). If:
 - Your ALB currently has 2 listeners in it:
 - 1 listener has HTTP protocol configuration and port 80
 - If a listener has HTTPS protocol configuration and port 443, we will use these 2 listeners.
 - Your ALB does not have either or both listeners with the above configuration, we will automatically create them.

 **Attention:**

If your ALB has:

- 1 listener has HTTP protocol configuration and port 443
- Or a listener configured with HTTPS protocol and portal 80

then when creating Ingress an error will occur. At this point, you need to edit valid listener information on the vLB system and recreate ingress.

3. After successfully creating ingress with an ALB , you are good to go

- Edit your ingress configuration according to the specific instructions at [Configure for an Application Load Balancer](#) .
- Or you can add/edit/delete policies in your ALB by editing the following parameters in the ingress resource (Ingress Yaml file). For example, below, I have set up **2 rules** as follows:

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    vks.vngcloud.vn/certificate-ids: "secret-58542fb-f410-4095-9e1c-34cd
spec:
  ingressClassName: "vngcloud"
  defaultBackend:
    service:
      name: nginx-service
      port:
        number: 80
  tls:
    - hosts:
      - '*.example.com'
      - 'example.com'
  rules:
    - host: example.com
      http:
        paths:
          - path: /1
            pathType: Exact
            backend:
              service:
                name: nginx-service1
                port:
                  number: 80
    - http:
      paths:
        - path: /2
          pathType: Exact
          backend:
            service:
              name: nginx-service2
              port:
                number: 80

```

- Like other Kubernetes resources, Ingress has a structure including the following information fields:
 - **apiVersion:** API version for Ingress.
 - **kind:** Resource type, in this case "Ingress".
 - **ingressClassName :** you need to specify this field value as "vngcloud" to use vngcloud-ingress-controller.

- **metadata:** Information describing Ingress, including name, annotations.
- **spec:** Ingress configuration, including traffic route rules according to the conditions of incoming requests. Ingress resources only support rules to direct HTTP traffic.

For general information about working with Ingress resources (Ingress Yaml files), see [Configure for an Application Load Balancer]).

Step 4: Check and edit the created Ingress resource

- After successfully creating ingress, you can view the ingress list via command

```
kubectl get ingress
```

For example, below we have successfully created nginx-ingress:

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
nginx-ingress	vngcloud	*	180.93.181.129	80	103m

- Or view details of an ingress by

```
kubectl describe ingress nginx-ingress
```

For example, below are the details of nginx-ingress that I created:

```
Name: nginx-ingress
Labels: <none>
Namespace: default
Address: 180.93.181.129
Ingress Class: vngcloud
Default backend: nginx-service:80 (172.16.24.202:80)
Rules:
Host Path Backends
-----
*
      /path1  nginx-service:80 (172.16.24.202:80)
Annotations: vks.vngcloud.vn/load-balancer-id: lb-6cdea8fd-4589-410e-933
Events: <none>
```

- To update an existing nginx-ingress, we can do so by updating the Ingress Yaml file as follows:

Copy

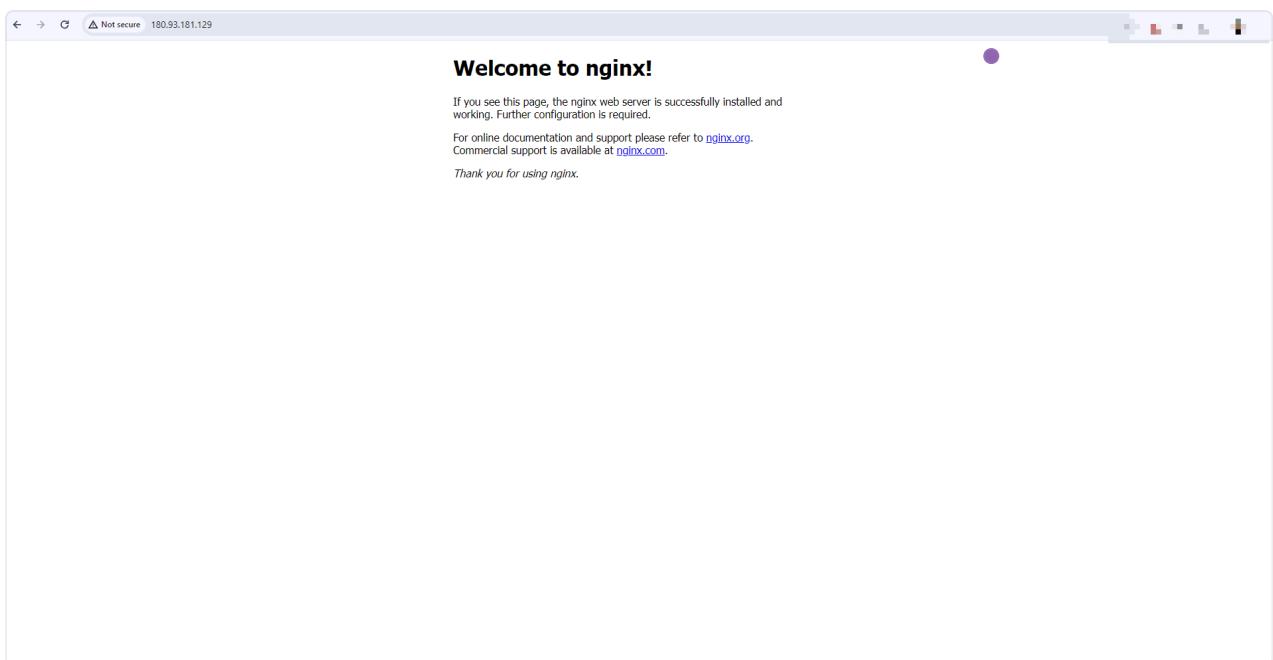
```
kubectl edit ingress nginx-ingress
```

Step 5: To access the nginx app, you can use the Load Balancer Endpoint that the system has created.

```
http://Endpoint/
```

You can get Load Balancer Public Endpoint information at the vLB interface. Specifically, access at

For example, below I have successfully accessed the nginx app with the address:
<http://180.93.181.129/>



Configure for an Application Load Balancer

On the [Ingress for an Application Load Balancer] page, we have shown you how to install the Ingress Controller and create ingress via the Ingress Yaml file. The following are detailed meanings of the information you can set for an Ingress

Annotations

Use the annotations below when creating ingress to customize the Load Balancer to suit your needs:

Annotations	Required/Not required	Meaning
vks.vngcloud.vn/load-balancer-id	Optional	<ul style="list-style-type: none">• If you do not already have a previously initialized Application Load Balancer on the vLB system. Now, when creating an Ingress, leave this information blank. After you have implemented Ingress deployment following the instructions at Ingress for an Application Load

[Balancer](#) . We

will automatically create an ALB on your cluster. This ALB will be displayed on vLB Portal, details can be accessed

[here](#)

- **If you already have a** previously initialized Application Load Balancer on the vLB system and you want to reuse the ALB for your cluster. Now, when creating an Ingress, enter the Load Balancer ID information into this annotation. After you have created Ingress according to the instructions at [Ingress for an Application Load Balancer](#) . If:

- ○ Your ALB currently has 2 listeners in it:

- 1 listener has HTTP protocol configuration and port 80

- If a listener has HTTPS protocol configuration and port 443, we will use these 2 listeners.

- Your ALB does not have either or

both listeners with the above configuration, we will automatically create them.

Attention:

If your ALB has:

- 1 listener has HTTP protocol configuration and port 443
- Or a listener configured with HTTPS protocol and portal 80

then when creating Ingress an error will occur. At this point, you need to edit valid listener information on the vLB system and recreate ingress.

vks.vngcloud.vn/load-balancer-name

Optional

- Annotation

vks.vngcloud.vn/load-balancer-name

will be used if you **do not** use annotation

load-
balancer-id

- Annotation
`vks.vngcloud.vn/load-balancer-name`

only makes sense when you create a new Ingress resource. After the Ingress resource is successfully created, this annotation **will be automatically deleted**. Using this annotation after the Ingress resource is created will **have no effect**.
- When you use this annotation, if you do not already have a previously initialized Application Load Balancer on the vLB system. We will automatically create an ALB

- on your cluster. This ALB will be displayed on vLB Portal, details can be accessed [here](#)

- If you already have a previously initialized Application Load Balancer on the vLB system and you want to reuse the ALB for your cluster. Now, please enter the Load Balancer Name information into this annotation.

vks.vngcloud.vn/package-id Optional

- If you do not enter this information, we will use the **ALB Small configuration by default**.
- If you already have an ACTIVE vLB host and you want to integrate this host into your K8S cluster,

		<p>please skip this information field.</p>
vks.vngcloud.vn/tags	Optional	<ul style="list-style-type: none"> The tag is added to your ALB.
vks.vngcloud.vn/scheme	Optional	<ul style="list-style-type: none"> Default is internet-facing, you can change it to internal depending on your needs.
vks.vngcloud.vn/security-groups	Optional	<ul style="list-style-type: none"> By default, a default security group will be created according to your Cluster.
vks.vngcloud.vn/inbound-cidrs	Optional	<ul style="list-style-type: none"> Default All CIDR: 0.0.0.0/0
vks.vngcloud.vn/healthy-threshold-count	Optional	<ul style="list-style-type: none"> Default 3
vks.vngcloud.vn/unhealthy-threshold-count	Optional	<ul style="list-style-type: none"> Default 3
vks.vngcloud.vn/healthcheck-interval-seconds	Optional	<ul style="list-style-type: none"> Default 30
vks.vngcloud.vn/healthcheck-timeout-seconds	Optional	<ul style="list-style-type: none"> Default 5
vks.vngcloud.vn/healthcheck-protocol	Optional	<ul style="list-style-type: none"> Default TCP. The user can select one of the TCP/HTTP values

vks.vngcloud.vn/healthcheck k-http-method	Optional	<ul style="list-style-type: none"> • Default GET. User can choose one of GET / POST / PUT values
vks.vngcloud.vn/healthcheck k-path	Optional	<ul style="list-style-type: none"> • Default /
vks.vngcloud.vn/healthcheck k-http-version	Optional	<ul style="list-style-type: none"> • Default 1.0. Users can choose one of the values 1.0, 1.1
vks.vngcloud.vn/healthcheck k-http-domain-name	Optional	<ul style="list-style-type: none"> • Default is empty
vks.vngcloud.vn/healthcheck k-port	Optional	<ul style="list-style-type: none"> • Default traffic port
vks.vngcloud.vn/success-codes	Optional	<ul style="list-style-type: none"> • Default 200
vks.vngcloud.vn/idle-timeout-client	Optional	<ul style="list-style-type: none"> • Default 50
vks.vngcloud.vn/idle-timeout-member	Optional	<ul style="list-style-type: none"> • Default 50
vks.vngcloud.vn/idle-timeout-connection	Optional	<ul style="list-style-type: none"> • Default 5
vks.vngcloud.vn/pool-algorithm	Optional	<ul style="list-style-type: none"> • Default ROUND_ROBIN. The user can select one of the values ROUND_ROBIN / LEAST_CONN ECTIONS / SOURCE_IP

vks.vngcloud.vn/enable-sticky-session	Optional	<ul style="list-style-type: none"> • Default false.
vks.vngcloud.vn/enable-tls-encryption	Optional	<ul style="list-style-type: none"> • Default false
vks.vngcloud.vn/target-node-labels	Optional	<ul style="list-style-type: none"> • Default is empty
vks.vngcloud.vn/certificate-ids	Optional	<ul style="list-style-type: none"> • Default is empty

IngressClassName

The Ingress installed by the VNGCloud Ingress Controller will have the information `IngressClassName = "vngcloud"`. You may not change this information.

DefaultBackend

- An Ingress without any rules will send all traffic to a single default service default backend, or if no `host` and `path` match the HTTP request in the Ingress Yaml file, traffic will be routed to the service default backend. For example below, we are configuring the default if the request does not satisfy any rule in the Ingress yaml file, it will go to service name: `example-svc-1` with port number `8080`

```
defaultBackend:
  service:
    name: example-svc-1
    port:
      number: 8080
```

TLS

You can secure Ingress by specifying a Secret that contains the TLS key and certificate. Currently Ingress only supports TLS port 443 and is the termination point for TLS (TLS termination). TLS Secret must contain fields with key names tls.crt and tls.key, which are the certificate and private key to use for TLS.

Specifically, you need to specify:

- Host: the specified hosts will use the cert.
 - SecretName: secret name containing cert.
-

Path types

Each path in Ingress has a corresponding pathType. There are three supported pathTypes:

- Exact: Matches the URL path with absolute precision and is case sensitive.
- Prefix: Matches based on the URL path prefix separated by /. Matching is case-sensitive and is performed on each element of the URL path. A component of the main URL path is a label separated by a / in the URL path (This means that the URL path can consist of multiple levels separated by /, each string is between two main / marks). is a label, each label is a component of the URL path). A URL request is considered to match a path field (configured in the Ingress specification) when the entire value of the path (which can include multiple components separated by /) matches the first labels (adjectives). left of the URL). For example /example1/path1 matches /example1/path1/path2, but not /example1/path1path2

Specific examples:

Path type	Path(s)	Request path(s)	Is there a match or not?
Exact	/example1	/example1	Have

Exact	/example1	/host1	Are not
Exact	/example1	/example1/	Are not
Exact	/example1/	/example1	Are not
Prefix	/	(all paths)	Have
Prefix	/example1	/example1 , /example1/	Have
Prefix	/example1/	/example1 , /example1/	Have
Prefix	/example1/host1 1	/example1/host1	Are not
Prefix	/example1/host1	/example1/host1	Have
Prefix	/example1/host1 /	/example1/host1	Have
Prefix	/example1/host1	/example1/host1 /	Have
Prefix	/example1/host1	/example1/host1 /ccc	Have
Prefix	/example1/host1	/example1/host1 xyz	Are not
Prefix	/ , /example1	/example1/ccc	Have
Prefix	/ , /example1 , /example1/host1	/example1/host1	Have
Prefix	/ , /example1 , /example1/host1	/ccc	Have
Prefix	/example1	/ccc	Are not

- In some cases, multiple paths within Ingress will match the path of the request URL. In those cases, priority will be given to the longest matching path first. If the two paths still have the same length, the priority will be in the order of the rule created on the Ingress Yaml file.

Ingress rule

Each HTTP rule contains the following information:

- **1 optional host** . If no host (we can understand it as a domain name) is specified, the rule will be applied to all HTTP traffic inbound to the specified IP address. If a host is specified (for example example1.com), the rule only applies to that host.
- **A list of paths** (for example /example1/host1), each path has a backend service associated with it defined by Service Name and Port Number. Both host and path must match the content of the incoming request before the load balancer directs traffic to the desired Services.
- **A backend** is a combination of the Service name and Port Number. HTTP and HTTPS requests going to Ingress and whose URL matches the host and path of the rule will be sent to the list of backends.

For example, does Host match the Host header according to the table:

Host	Host header	Is there a match or not?
*. example1.com	example2.example1.com	Have
*. example1.com	baz.example2.example1.com	Are not
*. example1.com	example1.com	Are not

ALB Limitation



Note

A few notes about the limitations of ingressing an ALB into a cluster:

- A cluster can have multiple Ingresses, each Ingress containing resource information for a single ALB.
 - One ALB can be used for many Ingress. From there, one ALB can be used for many clusters but must ensure these clusters have the same **Subnet**.
 - An ALB can include many listeners, many pools, and many policies. For limits on the number of listeners, number of pools, number of policies, please refer to [Resource limit]
-

Limit

- Ingress controller manager needs to configure annotations to specify ALB properties, such as protocol, port,... These annotations may vary depending on the cloud service provider, currently with **vngcloud ingress controller** is being used. provides a reference annotation list at [Configuration for an Application Load Balancer](#). We will further upgrade this part in the next release versions.
- Currently, vLB does not support the strip path feature in Load Balancer Layer 7, we will soon integrate this feature in the future.
- Changing the name or size (Rename, Resize) of the Load Balancer resource on vServer Portal can cause incompatibility with resources on the Kubernetes Cluster. This can lead to resources becoming inactive on the Cluster, or resources being resynchronized, or resource information between vServer Portal and the Cluster not matching. To prevent this problem, use `kubectl` Cluster resource management.

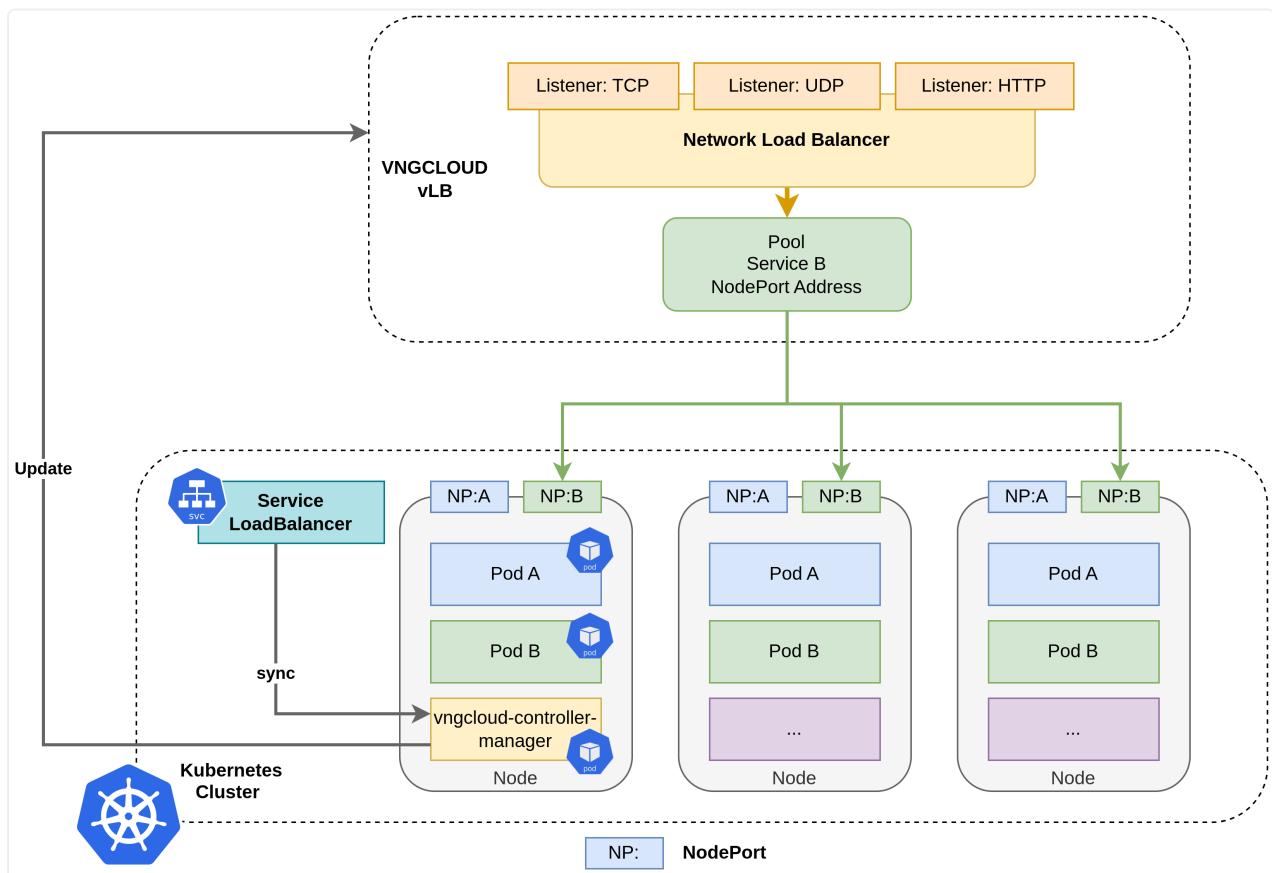
Working with Network load balancing (NLB)

.

What is NLB?

- **Network Load Balancer (NLB)** is a load balancer provided by VNGCloud that helps distribute network traffic to multiple back-end servers in a computer group (instance group). NLB operates at layer 4 of the OSI model, providing load balancing based on IP addresses and TCP/UDP ports. For more detailed information about NLB, please refer to [How it works (NLB)]

Model deployment



- **vngcloud-controller-manager :** VNG Cloud Controller Manager is a controller that runs on Kubernetes clusters deployed on VNG Cloud. It is responsible for managing VNG Cloud resources for Kubernetes clusters, including:
 - **Create and manage Network Load Balancer (NLB)** for Kubernetes Services with service type = Load Balancer.

Integrate with Network Load Balancer

To integrate a Network Load Balancer with a Kubernetes cluster, you can use a Service with type [LoadBalancer](#). When you create such a Service, VNGCloud Controller Manager will automatically create an NLB to forward traffic to pods on your [node](#). You can also use annotations to customize Network Load Balancer properties, such as port, protocol,...

Prepare

- Create a Kubernetes cluster on VNGCloud, or use an existing cluster. Note: make sure you have downloaded the cluster configuration file once the cluster has been successfully initialized and accessed your cluster.
- Create or use a **service account** created on IAM and attach policy: **vLBFullAccess**, **vServerFullAccess**. To create a service account, go here [and](#) follow these steps:
 - Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account
 - Find and select **Policy: vLBFullAccess and Policy: vServerFullAccess**, then click " **Create a Service Account** " to create Service Account, Policy: vLBFullAccess and Policy: vServerFullAccess created by VNG Cloud, you cannot delete these policies.
 - After successful creation, you need to save **the Client_ID** and **Secret_Key** of the Service Account to perform the next step.

Create Service Account and install VNGCloud Controller Manager

 **Attention:**

When you initialize the Cluster according to the instructions above, if you have not enabled the **Enable vLB Native Integration Driver** option , by

default we will not pre-install this plugin into your Cluster. You need to manually create Service Account and install VNGCloud Controller Manager according to the instructions below. If you have enabled the **Enable vLB Native Integration Driver** option , then we have pre-installed this plugin into your Cluster, skip the Service Account Initialization step, install VNGCloud Controller Manager and continue following the instructions from Deploy once. Workload.

- ✓ Instructions for creating Service Account and installing VNGCloud Controller Manager

Initialize Service Account

- Create or use a **service account** created on IAM and attach policy: **vLBFullAccess** , **vServerFullAccess** . To create a service account, go here [and](#) follow these steps:
 - Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account
 - Find and select **Policy: vLBFullAccess and Policy: vServerFullAccess** , then click " **Create a Service Account** " to create Service Account, Policy: vLBFullAccess and Policy: vServerFullAccess created by VNG Cloud, you cannot delete these policies.
 - After successful creation, you need to save **the Client_ID** and **Secret_Key** of the Service Account to perform the next step.
- Uninstall cloud-controller-manager

```
kubectl get daemonset -n kube-system | grep -i "cloud-controller"
```

```
# if your output is similar to the following, you MUST delete it  
kubectl delete daemonset cloud-controller-manager -n kube-system
```

- Besides, you can delete the Service Account being used for the cloud-controller-manager you just removed

```
kubectl get sa -n kube-system | grep -i "cloud-controller-manage"
# if your output is similar to the above, you MUST delete this service account
kubectl delete sa cloud-controller-manager -n kube-system --force
```

Install VNGCloud Controller Manager

- Install Helm version 3.0 or higher. Refer to <https://helm.sh/docs/intro/install/> for instructions on how to install.
- Add this repo to your cluster via the command:

```
helm repo add vks-helm-charts https://vngcloud.github.io/vks-helm-charts/
helm repo update
```

- Replace your K8S cluster's ClientID, Client Secret, and ClusterID information and continue running:

```
helm install vngcloud-controller-manager vks-helm-charts/vngcloud-controller-manager \
--namespace kube-system \
--set cloudConfig.global.clientID= <Lấy ClientID của Service / \
--set cloudConfig.global.clientSecret= <Lấy ClientSecret của Service / \
--set cluster.clusterID= <Lấy Cluster ID của cluster mà bạn đã cài đặt
```

- After the installation is complete, check the status of vngcloud-Integrate-controller pods:

```
kubectl get pods -n kube-system | grep vngcloud-controller-manager
```

For example, in the image below you have successfully installed vngcloud-controller-manager:

NAME	READY	STATUS
vngcloud-controller-manager-8864c754c-bqhvv	1/1	Running

Deploy a Workload

1.If you do not have a previously initialized Network Load Balancer available on the vLB system.

At this point, you need to do:

Step 1 : Create Deployment, Service for Nginx app.

- Create **nginx-service-lb4.yaml** file with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.19.1
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- Or use the following script file to deploy HTTP Apache Service with Internal LoadBalancer allowing internal access on port 8080:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: internal-http-apache2-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: apache2
  template:
    metadata:
      labels:
        app: apache2
    spec:
      containers:
        - name: apache2
          image: httpd
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: internal-http-apache2-service
  annotations:
    vks.vngcloud.vn/scheme: "internal"           # MUST set like this
spec:
  selector:
    app: apache2
  type: LoadBalancer                           # MUST set like this
  ports:
    - name: http
      protocol: TCP
      port: 8080                                # CAN be accessed via
      targetPort: 80

```

- Or sample YAML file to create Deployment and Service for a UDP server application in a Kubernetes cluster:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: udp-server-deployment
spec:
  selector:
    matchLabels:
      name: udp-server
  replicas: 5
  template:
    metadata:
      labels:
        name: udp-server
  spec:
    containers:
      - name: udp-server
        image: vcr.vngcloud.vn/udp-server
        imagePullPolicy: Always
        ports:
          - containerPort: 10001
            protocol: UDP
---

```

```

apiVersion: v1
kind: Service
metadata:
  name: udp-server-service
  annotations:
    vks.vngcloud.vn/pool-algorithm: "source-ip"
  labels:
    app: udp-server
spec:
  type: LoadBalancer
  sessionAffinity: ClientIP
  ports:
    - port: 10001
      protocol: UDP
  selector:
    name: udp-server

```

2.If you already have a previously initialized Network Load Balancer on the vLB system and you want to reuse the NLB for your cluster.

At this point, please enter the Load Balancer ID information into the **vks.vngcloud.vn/load-balancer-id annotation**. The example below is a sample

YAML file to deploy Nginx with External LoadBalancer using vngcloud-controller-manager to automatically expose the service to the internet using an L4 load balancer using an available NLB with ID = lb-2b9d8974-3760-4d60-8203-9671f229fb96

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: external-http-nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
---
kind: Service
apiVersion: v1
metadata:
  name: external-http-nginx-service
  annotations:
    vks.vngcloud.vn/package-id: "lbp-ddbf9313-3f4c-471b-af5-f6a3305159fc"
    vks.vngcloud.vn/load-balancer-id: "lb-2b9d8974-3760-4d60-8203-9671f229fb96"
spec:
  selector:
    app: nginx
  type: LoadBalancer
  ports:
    - name: http
      port: 80
      targetPort: 80
```

3.Once a new NLB has been automatically created by us , you can now proceed

- Edit your NLB configuration according to the specific instructions at [Configure for a Network Load Balancer](#). For example below, I have edited the protocol and port as follows:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: http-apache2-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: apache2
  template:
    metadata:
      labels:
        app: apache2
    spec:
      containers:
        - name: apache2
          image: httpd
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: http-apache2-service
  annotations:
    vks.vngcloud.vn/load-balancer-id: "lb-f8c0d85b-cb0c-4c77-b382-37982c4
spec:
  selector:
    app: apache2
  type: LoadBalancer
  ports:
    - name: http
      protocol: TCP
      port: 8000
      targetPort: 80

```

- Like other Kubernetes resources, **vngcloud-controller-manager** has a structure including the following information fields:
 - **apiVersion:** API version for Ingress.
 - **kind:** Resource type, in this case "Service".

- **metadata:** Information describing Ingress, including name, annotations.
- **spec:** Configure the conditions of incoming requests.

For general information about working with **vngcloud-controller-manager**, see [Configure for a Network Load Balancer]

- Deploy this Service using:

```
kubectl apply -f nginx-service-lb4.yaml
```

Step 2: Check Deployment and Service information just deployed

- Run the following command to test **Deployment**

```
kubectl get svc,deploy,pod -owide
```

- If the results are returned as below, it means you have deployed Deployment successfully.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/T
service/nginx-service	LoadBalancer	10.96.74.154	<pending>	80:31
NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx-app	0/1	1	0	2s
NAME	READY	STATUS	RESTARTS	A
pod/nginx-app-7f45b65946-bmrcf	0/1	ContainerCreating	0	2

At this point, the vLB system will automatically create a corresponding LB for the deployed nginx app, for example:

The screenshot shows the VNG Cloud Load Balancer interface. On the left, there's a sidebar with navigation links for vServer, Region HCM03, Overview, Limit, Network (VPCs, Floating IPs, Network Interfaces, Security Groups, Virtual IP Addresses, Route tables, Peering, Bandwidths, Interconnects, Network ACL), Server (Servers, Server Groups, SSH Keys), and System Images. The main content area has a header "Load Balancer" with a sub-header "Load balancer is a service that distributes incoming traffic to multiple servers. It is a highly available service that can be used to distribute traffic to multiple servers in a single region or multiple regions." Below this is a button "Create a Load Balancer". A search bar "Search with Suggestion" is at the top right. The main table lists one load balancer entry:

Name	Status	Endpoint	Schema	Type	Package	Created at	Action
lb-6ea7772-0ff8-442c-81b7-f03e44dde742 vks-k8s-6d2acb-default-nginx-serv-998c5	ACTIVE	180.93.181.20	Internet	Network	NLB_Small	21/04/2024 19:52:22	⋮

Total: 1 Show: 10 ⋮

At the bottom, there are footer links for Terms, Privacy Policy, and various compliance logos (ISAE30, ISO27001, ISO27017, ISO27018, SLA, PCI DSS). The footer also includes copyright information: © 2022, VI NA Data IT JSC, Helios Building, Lot 3, Road #3, Quang Trung Software City, Tan Chanh Hiep Ward, District 12, HCMC, Tax code: 0304851362, support@vngcloud.vn, 1800 1548, and Dept. of Planning and Investment of HCMC on 26/02/2007.

Step 3: To access the just exported nginx app, you can use the URL with the format:

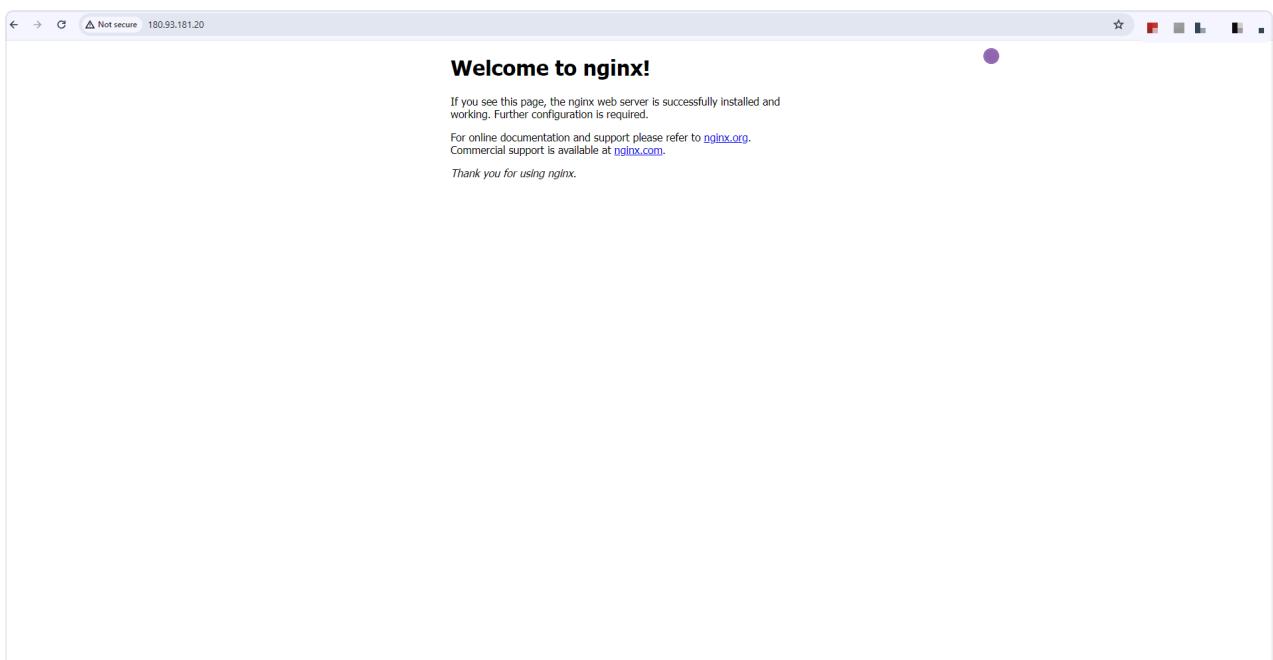
`http://Endpoint/`

You can get Load Balancer Public Endpoint information at the vLB interface.

Specifically, access at <https://hcm-3.console.vngcloud.vn/vserver/load-balancer/vlb/>

For example, below I have successfully accessed the nginx app with the address:

<http://180.93.181.20/>



Configure for a Network Load Balancer

On the [Integrate with Network Load Balancer] page, we have shown you how to install VNGCloud Controller Manager, create and apply yaml file. The following are detailed meanings of the information you can set in the yaml file:

Annotations

Use the annotations below to customize the Load Balancer to suit your needs:

Annotations	Required/Not required	Meaning
vks.vngcloud.vn/load-balancer-id	Optional	<ul style="list-style-type: none">If you do not already have a previously initialized Network Load Balancer on the vLB system. We will automatically create 1 NLB on your cluster. This NLB will be displayed on vLB Portal, details can be accessed hereIf you already have a previously initialized Network Load Balancer on the vLB system and you want to reuse the NLB for your cluster. Now, please enter the Load Balancer ID information into this annotation.

vks.vngcloud.vn/load-balancer-name

Optional

- **Annotation**

vks.vngcloud.vn/
load-balancer-
name

will be used if you
do not use
annotation
load-balancer-id .

- **Annotation**

vks.vngcloud.vn/
load-balancer-
name

only makes sense
when you create a
new load balancer.
After the load
balancer is
successfully
created, this
annotation **will be**
automatically
deleted . Using this
annotation after the
load balancer has
been created will
have no effect .

- **When you use this annotation, if you do not already have a previously initialized Network Load Balancer on the vLB system. We will automatically create 1 NLB on your cluster. This ALB will be displayed on vLB Portal, details can be accessed [here](#)**

- **If you already have a previously initialized Network Load Balancer on**

the vLB system and you want to reuse the NLB for your cluster. Now, please enter the Load Balancer Name information into this annotation.

vks.vngcloud.vn/package-id	Optional	<ul style="list-style-type: none">If you do not enter this information, we will use the NLB Small configuration by default.If you already have an ACTIVE vLB host and you want to integrate this host into your K8S cluster, please skip this information field.
vks.vngcloud.vn/tags	Optional	<ul style="list-style-type: none">The tag is added to your NLB.
vks.vngcloud.vn/scheme	Optional	<ul style="list-style-type: none">Default is internet-facing, you can change it to internal depending on your needs.
vks.vngcloud.vn/security-groups	Optional	<ul style="list-style-type: none">By default, a default security group will be created according to your Cluster.
vks.vngcloud.vn/inbound-cidrs	Optional	<ul style="list-style-type: none">Default All CIDR: 0.0.0.0/0
vks.vngcloud.vn/healthy-threshold-count	Optional	<ul style="list-style-type: none">Default 3
vks.vngcloud.vn/unhealthy-threshold-count	Optional	<ul style="list-style-type: none">Default 3

vks.vngcloud.vn/healthch eck-interval-seconds	Optional	<ul style="list-style-type: none"> • Default 30
vks.vngcloud.vn/healthch eck-timeout-seconds	Optional	<ul style="list-style-type: none"> • Default 5
vks.vngcloud.vn/healthch eck-protocol	Optional	<ul style="list-style-type: none"> • Default TCP . Users can select one of the values TCP/ HTTP/ HTTPS/ PING-UDP
vks.vngcloud.vn/healthch eck-http-method	Optional	<ul style="list-style-type: none"> • Default GET . User can choose one of GET / POST / PUT values
vks.vngcloud.vn/healthch eck-path	Optional	<ul style="list-style-type: none"> • Default /
vks.vngcloud.vn/healthch eck-http-version	Optional	<ul style="list-style-type: none"> • Default 1.0 . Users can choose one of the values 1.0, 1.1
vks.vngcloud.vn/healthch eck-http-domain-name	Optional	<ul style="list-style-type: none"> • Default is empty
vks.vngcloud.vn/healthch eck-port	Optional	<ul style="list-style-type: none"> • Default traffic port
vks.vngcloud.vn/success- codes	Optional	<ul style="list-style-type: none"> • Default 200
vks.vngcloud.vn/idle- timeout-client	Optional	<ul style="list-style-type: none"> • Default 50
vks.vngcloud.vn/idle- timeout-member	Optional	<ul style="list-style-type: none"> • Default 50
vks.vngcloud.vn/idle- timeout-connection	Optional	<ul style="list-style-type: none"> • Default 5
vks.vngcloud.vn/pool- algorithm	Optional	<ul style="list-style-type: none"> • Default ROUND_ROBIN . The user can select one of the values ROUND_ROBIN /

		LEAST_CONNECTIO NS / SOURCE_IP
vks.vngcloud.vn/target-node-labels	Optional	<ul style="list-style-type: none">• Default is empty
vks.vngcloud.vn/enable-proxy-protocol	Optional	<ul style="list-style-type: none">• Default is empty. The user specifies a list of service names in the Load Balancer to which the Proxy Protocol will be applied.

NLB Limitation

•

Note

A few notes about the limitations of integrating an NLB into a cluster:

- One NLB can be used for many clusters but must ensure these clusters have the same **Subnet** .
 - An NLB can include many listeners, many pools, and many policies. For limits on the number of listeners, number of pools, number of policies, please refer to [Resource Limits](#) .
-

Limit

- Changing the name or size (Rename, Resize) of the Load Balancer resource on vServer Portal can cause incompatibility with resources on the Kubernetes Cluster. This can lead to resources becoming inactive on the Cluster, or resources being resynchronized, or resource information between vServer Portal and the Cluster not matching. To prevent this problem, use `kubectl` Cluster resource management.

Overview

CNI (Container Network Interface) is a standard set of tools that provides networking capabilities to containers in a Kubernetes cluster. Simply put, CNI is an abstraction layer that helps Kubernetes manage and configure networking for pods (a collection of containers sharing the same network) in a flexible and efficient way.

How does CNI work?

When you create a new pod, Kubernetes calls CNI to create a network interface for that pod. The CNI plugin performs the following tasks:

- **Assign IP address:** Assign a unique IP address to the pod.
- **Routing configuration:** Set up routing rules that allow communication between pods,...

Additionally, the connections work as follows:

- **Connecting within the same VPC :** Nodes within the same VPC will connect directly to each other.
- **Connecting between different VPCs :** Use VPC Peering to connect nodes between different VPCs.
- **Connect to external infrastructure:** Use networking solutions such as site-to-site VPN or Direct Connect to connect from nodes in VPC to external infrastructures (On Cloud, On-premise).

This helps maintain a continuous, flexible, and secure network infrastructure in a multi-cloud or hybrid-cloud environment.

Comparison between CNI plugins

Currently, VKS is providing 3 popular CNI plugins: Calico Overlay, Cilium Overlay, Cilium VPC Native Routing. In which:

- **Calico Overlay** : Uses overlay model through tunneling (**IP-in-IP**). Compatible with many infrastructures but performance can be affected by tunnel **overhead**.
- **Cilium Overlay** : Also uses the overlay model but has strong integration with **eBPF**, which improves performance, security, and scalability.
- **Cilium VPC Native Routing** : Uses **eBPF** and **no overlay required**, leveraging the routing capabilities of the VPC infrastructure, providing the best performance and scalability.

When to use Calico Overlay : simple to use, does not require too high performance.

When to use Cilium Overlay : simple to use, does not require too high performance but requires intensive monitoring (Hubble).

When to use Cilium VPC Native Routing : high performance requirements, easy **connectivity to external systems**, and in-depth monitoring needs (Hubble).

Using CNI Calico Overlay

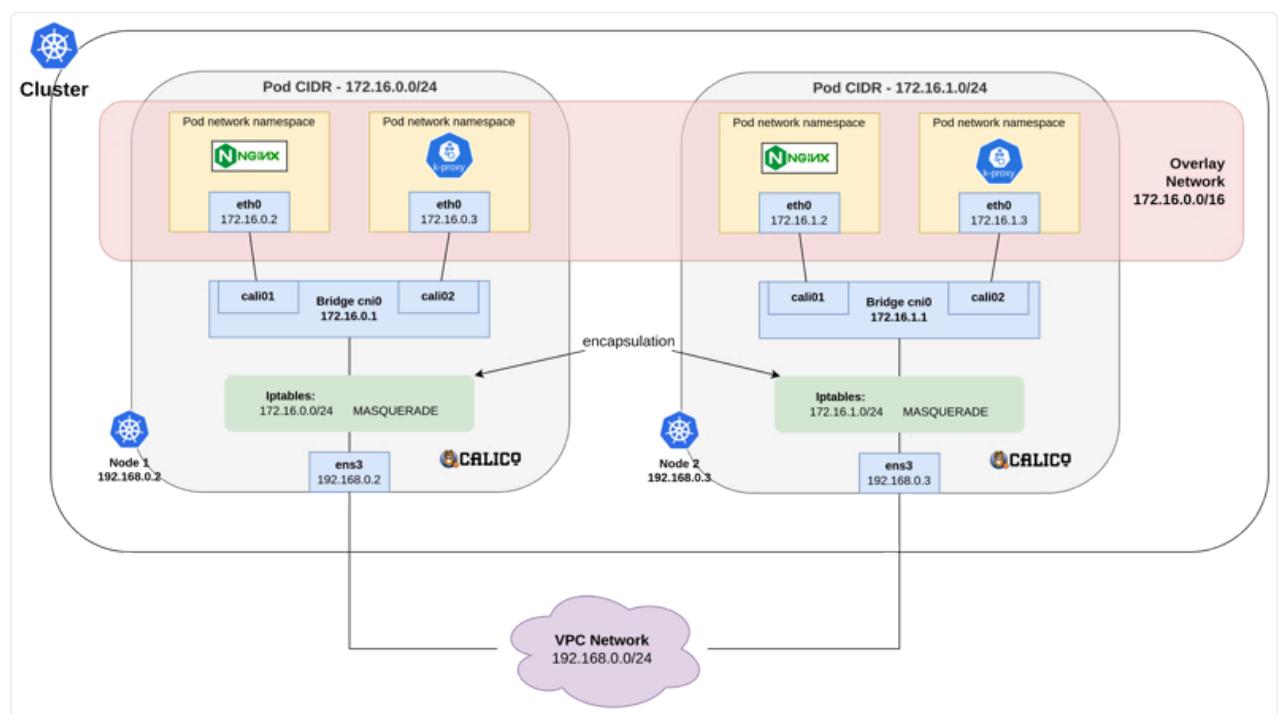
•

Overview

The **CNI Calico Overlay** in VKS is a type of **overlay network** that uses **IP-in-IP encapsulation** to create an overlay network. This allows pods to communicate with each other without changing the underlying physical network configuration. Pods will receive IP addresses from the IP address range configured for Calico, which is usually different from the IP address of your VPC or subnet.

Model

On VKS, **Calico Overlay** works according to the following model:



In there:

- **Pods** on each node communicate with each other via **the cali interface** and **bridge cni0**.

- When pods need to communicate with pods on other nodes, packets are encapsulated into overlay packets and sent over **the physical network (VPC Network)**.
 - **Calico** on each node is responsible for performing encapsulation and decapsulation so that pods can communicate across different nodes.
-

Necessary conditions

To be able to initialize a **Cluster** and **Deploy a Workload**, you need:

- There is at least **1 VPC** and **1 Subnet in ACTIVE state**. If you do not have any VPC, Subnet, please initialize VPC, Subnet according to the instructions here [.](#)
 - There is at least **1 SSH key in ACTIVE state**. If you do not have any SSH key, please initialize SSH key following the instructions here [.](#)
 - **kubectl** installed and configured on your device. please refer here [if](#) you are not sure how to install and use kubectl. In addition, you should not use an outdated version of kubectl, we recommend that you use a kubectl version that is no more than one version different from the cluster version.
-

Create a Cluster using Calico Overlay

To initialize a Cluster, follow the steps below:

Step 1: Access <https://vks.console.vngcloud.vn/overview>

Step 2: On the Overview screen, select **Activate**.

Step 3: Wait until we successfully initialize your VKS account. After successfully Activating, select **Create a Cluster**.

Step 4: At the Cluster initialization screen, we have set up the information for the Cluster and a **Default Node Group** for you. To use **Calico Overlay** for your **Cluster**,

please select:

- **Network type : Calico Overlay**

Field	Meaning	Illustrative example
VPC	The IP address range that the Cluster nodes will use to communicate.	In the picture, we choose VPC with IP range 10.111.0.0/16 , corresponding to 65536 IPs
Subnet	A smaller IP address range belonging to the VPC. Each node in the Cluster will be assigned an IP from this Subnet. The Subnet must be within the IP range of the selected VPC.	In the picture, we choose Subnet with Primary IP range of 10.111.0.0/24 , corresponding to 256 IPs
IP-IP encapsulation mode	IP-IP encapsulation mode in VKS is Always	In the figure, we select Always mode to always encapsulate packets.
CIDR	The virtual network range that the pods will use	In the picture, we choose the virtual network range as 172.16.0.0/16 . The pods will get IP from this IP range.

Attention:

- **Only one networktype:** In a cluster, you can use only one of three networktypes: Calico Overlay, Cilium Overlay, or Cilium VPC Native Routing
- **Multiple subnets for a cluster:** VKS supports the use of multiple subnets for a cluster. This allows you to configure each node group in the cluster to be located on different subnets within the same VPC, helping to optimize resource allocation and network management.

Step 5: Select **Create Kubernetes cluster**. Please wait a few minutes for us to initialize your Cluster, the status of the Cluster is now **Creating**.

Step 6: When the Cluster status is **Active**, you can view Cluster information and Node Group information by selecting Cluster Name in the **Name** column.

Deploy a Workload

Below are instructions for deploying an nginx deployment and testing IP assignment for the pods deployed in your cluster.

Step 1: Access <https://vks.console.vngcloud.vn/k8s-cluster>

Step 2: The Cluster list is displayed, select the **Download** icon and select **Download Config File** to download the kubeconfig file. This file will give you full access to your Cluster.

Step 3 : Rename this file to config and save it to the **~/.kube/config** folder

Step 4: Perform Cluster check via command:

- Run the following command to check **the node**

```
kubectl get nodes
```

- If the result is as below, it means your Cluster is successfully initialized with 5 nodes:

NAME	STATUS	ROLES	AGE	VERSION
vks-cluster01-nodegroup-536d9-452f1	Ready	<none>	15h	v1.28.8
vks-cluster01-nodegroup-998b1-14f64	Ready	<none>	16h	v1.28.8
vks-cluster01-nodegroup01-22e98	Ready	<none>	19h	v1.28.8
vks-cluster01-nodegroup01-36911	Ready	<none>	19h	v1.28.8
vks-cluster01-nodegroup01-9102e	Ready	<none>	19h	v1.28.8

- Continue running the following command to check the pods **deployed** on your kube-system namespace:

```
k get pods -A
```

- If the result is as below, it means that the pods supporting **Calico Overlay** have been successfully run:

NAMESPACE	NAME	READY	STAT
kube-system	calico-kube-controllers-868b574465-wbxlx	1/1	Runn
kube-system	calico-node-65ql2	1/1	Runn
kube-system	calico-node-d9hc7	1/1	Runn
kube-system	calico-node-gp2s7	1/1	Runn
kube-system	calico-node-hgk86	1/1	Runn
kube-system	calico-node-vj9ts	1/1	Runn
kube-system	calico-typha-74d79bf5f6-zzdn9	1/1	Runn
kube-system	coredns-1727334072-86776977c9-19tcp	1/1	Runn
kube-system	coredns-1727334072-86776977c9-xqcn9	1/1	Runn
kube-system	konnectivity-agent-bj7wc	1/1	Runn
kube-system	konnectivity-agent-fnm7j	1/1	Runn
kube-system	konnectivity-agent-gvnbl	1/1	Runn
kube-system	konnectivity-agent-jj764	1/1	Runn
kube-system	konnectivity-agent-vgmwf	1/1	Runn
kube-system	kube-proxy-8r85m	1/1	Runn
kube-system	kube-proxy-bddf5	1/1	Runn
kube-system	kube-proxy-kwsk1	1/1	Runn
kube-system	kube-proxy-zv6m4	1/1	Runn
kube-system	kube-proxy-zw65v	1/1	Runn
kube-system	vngcloud-controller-manager-67cf7f868c-jc66k	1/1	Runn
kube-system	vngcloud-csi-controller-746b67bcb8-dn5d7	7/7	Runn
kube-system	vngcloud-csi-controller-746b67bcb8-hqm24	7/7	Runn
kube-system	vngcloud-csi-node-24nlb	3/3	Runn
kube-system	vngcloud-csi-node-fgxpg	3/3	Runn
kube-system	vngcloud-csi-node-q9npf	3/3	Runn
kube-system	vngcloud-csi-node-tw5sv	3/3	Runn
kube-system	vngcloud-csi-node-z2sk9	3/3	Runn
kube-system	vngcloud-ingress-controller-0	1/1	Runn

Step 2: Deploy nginx on the newly created cluster:

- Initialize the **nginx-deployment.yaml** file with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 20
  template:
    metadata:
      labels:
        app: nginx
  spec:
    containers:
      - name: nginx
        image: nginx:latest
        ports:
          - containerPort: 80
```

- Perform this deployment via command:

```
kubectl apply -f nginx-deployment.yaml
```

Step 3: Check the deployed nginx pods and the IP address assigned to each pod

- Perform a check of the **pods** via the command:

```
kubectl get pods -o wide
```

- You can observe below, the **nginx pods** are assigned IPs 172.16.xx which satisfy the **Calico CIDR condition 172.16.0.0/16** that we specified above:

NAME	READY	STATUS	RESTARTS	AGE	IP
nginx-app-7c79c4bf97-2xbwd	1/1	Running	0	49s	172.16.19
nginx-app-7c79c4bf97-5hcds	1/1	Running	0	49s	172.16.19
nginx-app-7c79c4bf97-5hgwp	1/1	Running	0	49s	172.16.19
nginx-app-7c79c4bf97-5l79h	1/1	Running	0	49s	172.16.83
nginx-app-7c79c4bf97-5q2f4	1/1	Running	0	49s	172.16.22
nginx-app-7c79c4bf97-5szc6	1/1	Running	0	49s	172.16.83
nginx-app-7c79c4bf97-9272q	1/1	Running	0	49s	172.16.16
nginx-app-7c79c4bf97-cgwrj	1/1	Running	0	49s	172.16.67
nginx-app-7c79c4bf97-fhl4g	1/1	Running	0	49s	172.16.16
nginx-app-7c79c4bf97-fj865	1/1	Running	0	49s	172.16.83
nginx-app-7c79c4bf97-gh6hj	1/1	Running	0	49s	172.16.16
nginx-app-7c79c4bf97-hx2rn	1/1	Running	0	49s	172.16.83
nginx-app-7c79c4bf97-jv26j	1/1	Running	0	49s	172.16.16
nginx-app-7c79c4bf97-km7p4	1/1	Running	0	49s	172.16.22
nginx-app-7c79c4bf97-lrh2r	1/1	Running	0	49s	172.16.16
nginx-app-7c79c4bf97-lvj6g	1/1	Running	0	49s	172.16.67
nginx-app-7c79c4bf97-nhhdk	1/1	Running	0	49s	172.16.22
nginx-app-7c79c4bf97-qr2lm	1/1	Running	0	49s	172.16.67
nginx-app-7c79c4bf97-x4ztb	1/1	Running	0	49s	172.16.22
nginx-app-7c79c4bf97-xrqwx	1/1	Running	0	49s	172.16.67

- You can also perform a detailed description of each pod to check this pod information via the command:

```
kubectl describe pod nginx-app-7c79c4bf97-2xbwd
```

Using CNI Cilium Overlay

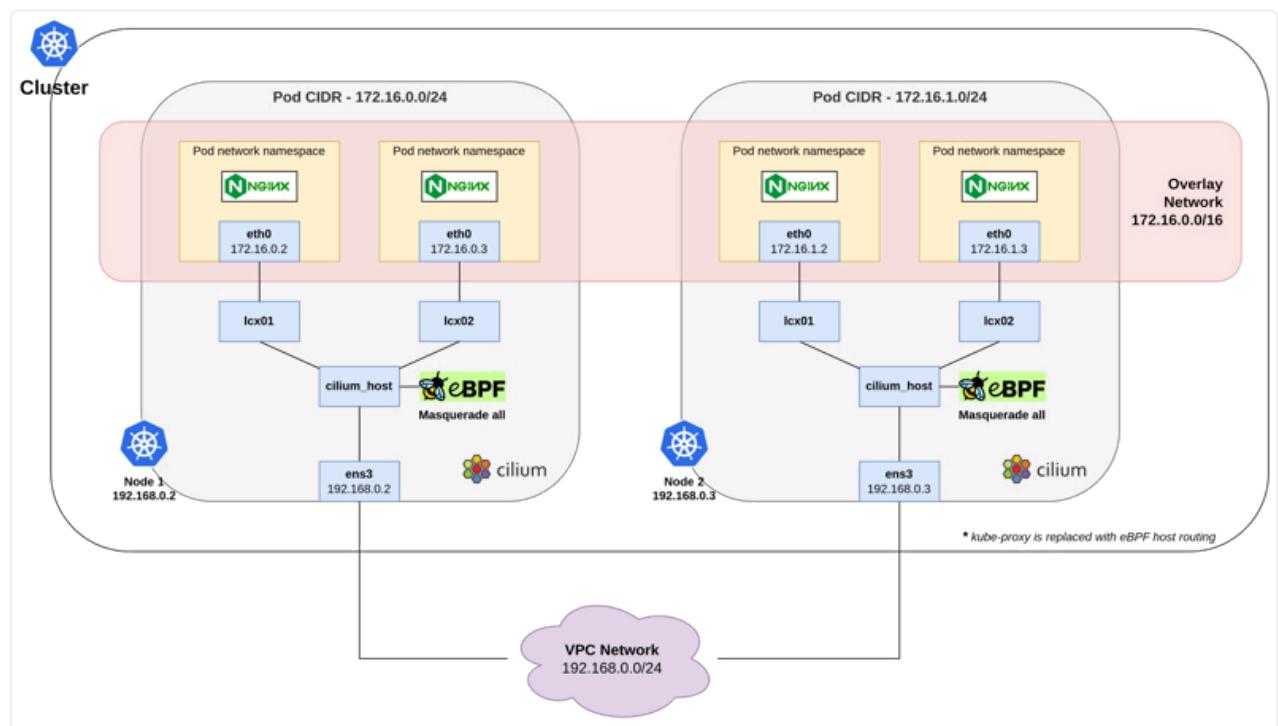
•

Overview

CNI Cilium Overlay in VKS is a type of **overlay network** that uses **eBPF (extended Berkeley Packet Filter)** to enhance network performance and security. Compared to solutions like **Calico Overlay**, Cilium offers higher performance thanks to its ability to process traffic directly in the kernel using eBPF. In addition, Calico often uses **iptables** to manage traffic, while Cilium with eBPF can handle network policies and application-specific behaviors (Layer 7).

Model

On VKS, **Cilium Overlay** works according to the following model:



In there:

- **Pod (eth0) → Ixc01/Ixc02** : Pods communicate over a virtual network created by Cilium.

- **Ixc01/Ixc02 → cilium_host** : Packets from Pods are forwarded to `cilium_host` , which is the intermediate layer between the Pod's network and the physical network.
 - **cilium_host → ens3** : After being processed by Cilium (and eBPF), packets are sent to the physical network via `ens3` .
 - **ens3 → VPC Network** : Finally, packets are transmitted over the physical network to other nodes or out of the cluster.
-

Necessary conditions

To be able to initialize a **Cluster** and **Deploy a Workload** , you need:

- There is at least **1 VPC** and **1 Subnet in ACTIVE state** . If you do not have any VPC, Subnet, please initialize VPC, Subnet according to the instructions here [.](#)
 - There is at least **1 SSH key in ACTIVE state** . If you do not have any SSH key, please initialize SSH key following the instructions here [.](#)
 - **kubectl** installed and configured on your device. please refer here [if](#) you are not sure how to install and use kubectl. In addition, you should not use an outdated version of kubectl, we recommend that you use a kubectl version that is no more than one version different from the cluster version.
-

Initialize a Cluster using Cilium Overlay

To initialize a Cluster, follow the steps below:

Step 1: Access <https://vks.console.vngcloud.vn/overview>

Step 2: On the Overview screen , select **Activate**.

Step 3: Wait until we successfully initialize your VKS account. After successfully Activating, select **Create a Cluster**.

Step 4: At the Cluster initialization screen, we have set up the information for the Cluster and a **Default Node Group** for you. To use **Cilium Overlay** for your **Cluster**, please select:

- **Network type** : Cilium Overlay

Field	Meaning	Illustrative example
VPC	The IP address range that the Cluster nodes will use to communicate.	In the picture, we choose VPC with IP range 10.111.0.0/16 , corresponding to 65536 IPs
Subnet	A smaller IP address range belonging to the VPC. Each node in the Cluster will be assigned an IP from this Subnet. The Subnet must be within the IP range of the selected VPC.	In the picture, we choose Subnet with Primary IP range of 10.111.0.0/24 , corresponding to 256 IPs
IP-IP encapsulation mode	IP-IP encapsulation mode in VKS is Always	In the figure, we select Always mode to always encapsulate packets.
CIDR	The virtual network range that the pods will use	In the picture, we choose the virtual network range as 172.16.0.0/16 . The pods will get IP from this IP range.

Attention:

- **Only one networktype:** In a cluster, you can use only one of three networktypes: Calico Overlay, Cilium Overlay, or Cilium VPC Native Routing
- **Multiple subnets for a cluster:** VKS supports the use of multiple subnets for a cluster. This allows you to configure each node group in the cluster to be located on different subnets within the same VPC, helping to optimize resource allocation and network management.

Step 5: Select **Create Kubernetes cluster**. Please wait a few minutes for us to initialize your Cluster, the status of the Cluster is now **Creating**.

Step 6: When the Cluster status is **Active**, you can view Cluster information and Node Group information by selecting Cluster Name in the **Name** column.

Deploy a Workload

Below are instructions for deploying an nginx deployment and testing IP assignment for the pods deployed in your cluster.

Step 1: Access <https://vks.console.vngcloud.vn/k8s-cluster>

Step 2: The Cluster list is displayed, select the **Download** icon and select **Download Config File** to download the kubeconfig file. This file will give you full access to your Cluster.

Step 3 : Rename this file to config and save it to the **~/.kube/config** folder

Step 4: Perform Cluster check via command:

- Run the following command to check **the node**

```
kubectl get nodes
```

- If the result is as below, it means your Cluster is successfully initialized with 3 nodes:

NAME	STATUS	ROLES	AGE	VERSION
vks-cluster-02-nodegroup-7fb09-3a594	Ready	<none>	5m48s	v1.29.1
vks-cluster-02-nodegroup-7fb09-3cb67	Ready	<none>	5m34s	v1.29.1
vks-cluster-02-nodegroup-7fb09-430aa	Ready	<none>	5m52s	v1.29.1

- Continue running the following command to check the pods **deployed** on your kube-system namespace:

```
k get pods -A
```

- If the result is as below, it means that the pods supporting **Cilium Overlay** have been successfully run:

NAMESPACE	NAME	READY	STAT
kube-system	cilium-8xtwz	1/1	Runn
kube-system	cilium-cpxvv	1/1	Runn
kube-system	cilium-envoy-b95pg	1/1	Runn
kube-system	cilium-envoy-dx8qg	1/1	Runn
kube-system	cilium-envoy-sqdn8	1/1	Runn
kube-system	cilium-operator-75b8c6f6d4-7x4f6	1/1	Runn
kube-system	cilium-operator-75b8c6f6d4-k7j45	1/1	Runn
kube-system	cilium-zs2cm	1/1	Runn
kube-system	coredns-1727408780-5fcf89468-7hmvp	1/1	Runn
kube-system	coredns-1727408780-5fcf89468-v9nbd	1/1	Runn
kube-system	hubble-relay-8899f8cdc-976zf	1/1	Runn
kube-system	hubble-ui-574c5bb99b-gg7jx	2/2	Runn
kube-system	konnectivity-agent-46nvd	1/1	Runn
kube-system	konnectivity-agent-qhq4m	1/1	Runn
kube-system	konnectivity-agent-xs7bq	1/1	Runn
kube-system	vngcloud-controller-manager-7c47d64584-z8827	1/1	Runn
kube-system	vngcloud-csi-controller-848f68f46-2hkxl	7/7	Runn
kube-system	vngcloud-csi-controller-848f68f46-bkvkg	7/7	Runn
kube-system	vngcloud-csi-node-8rxbx	3/3	Runn
kube-system	vngcloud-csi-node-mxknq	3/3	Runn
kube-system	vngcloud-csi-node-tfrsp	3/3	Runn
kube-system	vngcloud-ingress-controller-0	1/1	Runn

Step 2: Deploy nginx on the newly created cluster:

- Initialize the **nginx-deployment.yaml** file with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 20
  template:
    metadata:
      labels:
        app: nginx
  spec:
    containers:
      - name: nginx
        image: nginx:latest
        ports:
          - containerPort: 80
```

- Perform this deployment via command:

```
kubectl apply -f nginx-deployment.yaml
```

Step 3: Check the deployed nginx pods and the IP address assigned to each pod

- Perform a check of the pods via the command:

```
kubectl get pods -o wide
```

- You can observe below, the **nginx pods** are assigned IPs 172.16.xx which satisfy the **Cilium CIDR condition 172.16.0.0/16** that we specified above:

NAME	READY	STATUS	RESTARTS	AGE	IP
nginx-app-7c79c4bf97-4lcbn	1/1	Running	0	83s	172.16.0.
nginx-app-7c79c4bf97-669z9	1/1	Running	0	83s	172.16.1.
nginx-app-7c79c4bf97-7hqp5	1/1	Running	0	83s	172.16.1.
nginx-app-7c79c4bf97-8fjhm	1/1	Running	0	83s	172.16.2.
nginx-app-7c79c4bf97-8xmfm	1/1	Running	0	83s	172.16.0.
nginx-app-7c79c4bf97-9b4px	1/1	Running	0	83s	172.16.2.
nginx-app-7c79c4bf97-b7vlg	1/1	Running	0	83s	172.16.1.
nginx-app-7c79c4bf97-bc6r4	1/1	Running	0	83s	172.16.0.
nginx-app-7c79c4bf97-flkz5	1/1	Running	0	83s	172.16.2.
nginx-app-7c79c4bf97-k55j6	1/1	Running	0	83s	172.16.2.
nginx-app-7c79c4bf97-19p8p	1/1	Running	0	83s	172.16.2.
nginx-app-7c79c4bf97-llnfq	1/1	Running	0	83s	172.16.1.
nginx-app-7c79c4bf97-mg9t8	1/1	Running	0	83s	172.16.0.
nginx-app-7c79c4bf97-mlh7g	1/1	Running	0	83s	172.16.2.
nginx-app-7c79c4bf97-n946h	1/1	Running	0	83s	172.16.1.
nginx-app-7c79c4bf97-p9k42	1/1	Running	0	83s	172.16.0.
nginx-app-7c79c4bf97-sl4b8	1/1	Running	0	83s	172.16.1.
nginx-app-7c79c4bf97-tdtjc	1/1	Running	0	83s	172.16.2.
nginx-app-7c79c4bf97-zwxps	1/1	Running	0	83s	172.16.2.
nginx-app-7c79c4bf97-zxx87	1/1	Running	0	83s	172.16.0.

- You can also perform a detailed description of each pod to check this pod information via the command:

```
kubectl describe pod nginx-app-7c79c4bf97-4lcbn
```

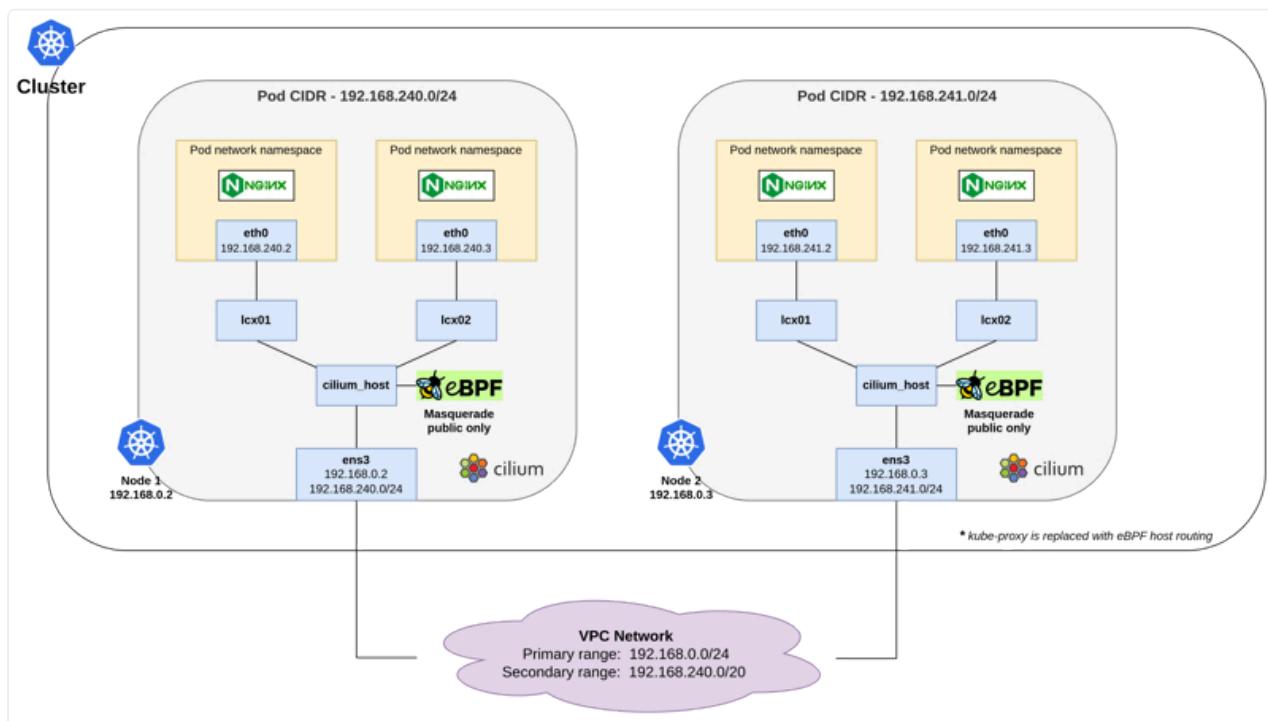
Using CNI Cilium VPC Native Routing

Overview

CNI (Container Network Interface) Cilium VPC Native Routing is a mechanism that helps Kubernetes manage networks without using overlay networks. Instead of using virtual network layers, **CNI Cilium VPC Native Routing** leverages the direct routing capabilities of cloud service providers' VPCs (Virtual Private Clouds) to optimize data transfer between nodes and pods in the Kubernetes cluster.

Model

On VKS, **CNI (Container Network Interface) Cilium VPC Native Routing** operates according to the following model:



In there:

- Each **Node** has a private IP address range for pods (Pod CIDR). Pods in each node use addresses from this CIDR and communicate over the virtual network.
 - **Cilium** and **eBPF** perform network management for all pods on each node, including handling traffic going from pod to pod, or from node to node. When necessary, eBPF performs **masquerading** to hide the internal IP address of the pod when communicating with the external network.
 - **Cilium** ensures that pods can communicate with each other both within the same node and between different nodes.
-

Necessary conditions

To be able to initialize a **Cluster** and **Deploy a Workload**, you need:

- There is at least **1 VPC** and **1 Subnet in ACTIVE state**. If you do not have any VPC, Subnet, please initialize VPC, Subnet according to the instructions below:
 - **Step 1:** Access the vServer homepage at the link <https://hcm-3.console.vngcloud.vn/vserver>
 - **Step 2:** Select the **VPCs** menu in the left menu of the screen.
 - **Step 3:** Here, if you don't have any VPC yet, please select **Create VPC** by entering the **VPC name** and defining the desired **CIDR/16 range**.
 - **Step 4:** After having at least 1 VPC, to create a subnet, you need to select **View Detail** to expand the control panel at the bottom, including the **Subnet** section.
 - **Step 5:** In the **Subnet** section, select **Add Subnet**. Now, you need to enter:
 - **Subnet name:** the subnet's mnemonic name
 - **Primary CIDR :** This is the primary IP address range of the subnet. All internal IP addresses of virtual machines (VMs) in this subnet will be taken from this address range. For example, if you set Primary CIDR to 10.1.0.0/24, the IP addresses of the VMs will be in the range of 10.1.0.1 to 10.1.0.254.
 - **Secondary CIDR :** This is a secondary IP address range, used to provide additional IP addresses or to separate different services within the same subnet. Each Node has a private IP address range for its pods (Pod CIDR).

CIDR). The pods in each node use addresses from this CIDR and communicate over the virtual network.

The screenshot shows the VNG Cloud VPC management interface. On the left, there's a sidebar with options like Overview, Limit, Network (VPCs selected), DHCPs, Floating IPs, Network Interfaces, Endpoint, Public NAT, Security Groups, Virtual IP Addresses, Route tables, Peering, and Bandwidths. The main area shows a table of existing VPCs with columns for Name, Status, CIDR, DHCP, Route Table, Created at, and Action. A red arrow points to the 'Name' column header. Below the table is a sub-section titled 'List of Subnets added to this VPC.' with a table showing subnets named 'subnet1' and 'subnet2'. A red arrow points to the '+ Add subnet' button at the top right of this section.

The screenshot shows the 'Create subnet' dialog box. It has fields for 'Subnet name' (set to 'my-subnet'), 'Primary CIDR (Optional)' (set to '10.111.0.0 /24'), and 'Secondary CIDR (Optional)'. Under 'Secondary CIDR', there are two entries: 'cidr-1' with CIDR Range '10.111.160.0/20' and 'cidr-2' with CIDR Range '10.111.128.0/20'. Both ranges are highlighted with a red box. At the bottom right of the dialog is a '+ Add a CIDR' button. The background shows the same VPC management interface as the previous screenshot.



Attention:

- The IP address ranges of **Primary CIDR** and **Secondary CIDR** cannot overlap. This means that the address range of **Secondary CIDR** must be outside the range of **Primary CIDR** and vice versa. For example, if Primary CIDR is 10.1.0.0/24, then Secondary CIDR cannot be 10.1.0.0/20

because it is within the range of Primary CIDR. Instead, you can use a different address range like 10.1.16.0/20.

- There is at least 1 **SSH key in ACTIVE state**. If you do not have any SSH key, please initialize SSH key following the instructions here [.](#)
- **kubectl** installed and configured on your device. please refer here [if](#) you are not sure how to install and use kubectl. In addition, you should not use an outdated version of kubectl, we recommend that you use a kubectl version that is no more than one version different from the cluster version.

 **Attention:**

- When using Cilium's native routing mode, it is crucial to configure **Security Groups** correctly to allow necessary connections. For example, when running an NGINX pod on a node, you must permit traffic on port 80 to ensure requests from other nodes can connect. This configuration is not required when using the network overlay mode.

Create a Cluster using CNI Cilium VPC Native Routing

To initialize a Cluster, follow the steps below:

Step 1: Access <https://vks.console.vngcloud.vn/overview>

Step 2: On the Overview screen , select **Activate**.

Step 3: Wait until we successfully initialize your VKS account. After successfully Activating, select **Create a Cluster**.

Step 4: At the Cluster initialization screen, we have set up the information for the Cluster and a **Default Node Group** for you. To use **CNI Cilium VPC Native Routing** for your **Cluster** , please select:

- **Network type** : Cilium VPC Native Routing and other parameters as follows:

Field	Meaning	Illustrative example
VPC	The IP address range that the Cluster nodes will use to communicate.	In the picture, we choose VPC with IP range 10.111.0.0/16 , corresponding to 65536 IPs
Subnet	A smaller IP address range belonging to the VPC. Each node in the Cluster will be assigned an IP from this Subnet. The Subnet must be within the IP range of the selected VPC.	In the picture, we choose Subnet with Primary IP range of 10.111.0.0/24 , corresponding to 256 IPs
Default Pod IP range	This is the secondary IP address range used for pods. It is called Secondary IP range because it does not match the primary IP range of the node. Pods in the Cluster will be assigned IPs from this range.	In the picture, we choose Secondary IP range as 10.111.160.0/20 - Corresponding to 4096 IPs for pods
Node CIDR mask size	CIDR size for nodes. This parameter indicates how many IP addresses each node will be assigned from the pod IP range. This size should be chosen to ensure that there are enough IP addresses for all pods on each node. You can refer to the table below to understand how to calculate the number of IP addresses that can be used to allocate to nodes and pods in your cluster.	In the picture, we choose Node CIDR mask size as /25 - Each node will have 128 IP addresses , suitable for the number of pods you want to run on a node.

Calculating the number of IPs for pods and nodes:



1. Số lượng IP khả dụng trên mỗi node:

$$\text{Số lượng IP khả dụng trên mỗi node} = 2^{(32 - \text{Node CIDR mask size})}$$

2. Số lượng node có thể tạo trong dài Secondary IP range:

$$\text{Số node có thể tạo} = \frac{\text{Số IP trong dài Secondary IP range}}{\text{Số IP mỗi node có thể sử dụng từ Node CIDR}}$$

3. Số lượng pod thực tế có thể tạo trên mỗi node:

$$\text{Số lượng pod thực tế có thể tạo} = \frac{\text{Số lượng IP khả dụng trên mỗi node}}{2}$$

Suppose, when initializing the cluster, I choose:

- **VPC** : 10.111.0.0/16
- **Subnet:**
 - **Primary IP Range:** 10.111.0.0/24
 - **Secondary IP Range:** 10.111.160.0/20
- **Node CIDR mask size:** Selectable values range from **/24** to **/26** .

Node CIDR mask size	Number of IPs per node	Number of nodes that can be created in the /20 range (4096 IPs)	Number of IPs allocated to pods on each node	Actual number of pods that can be created
/24	256	16	256	128
/25	128	32	128	64
/26	64	64	64	32

VNG CLOUD Search VNG Cloud services

Network Setting

Define how applications in this cluster communicate with each other and with the Kubernetes control plane, and how clients can reach them.

Public Cluster The VNG public clusters refer to a type of Kubernetes cluster configuration where the Kubernetes API server endpoint is publicly accessible over the internet. In a VNG public cluster, the API server endpoint is not restricted to private access within a VPC (Virtual Private Cloud) and can be accessed over the public internet.

Private Cluster The VNG private clusters are configured to have private access to the Kubernetes API server endpoint. This means that the API server endpoint is only accessible from within a specific Virtual Private Cloud (VPC) and is not exposed to the public internet. Private clusters provide enhanced security by restricting access to the Kubernetes API to resources within the VPC.

Network type * Cilium VPC Native Routing

VPC * (10.111.0.0/16)

Subnet * subnet1 (10.111.0.0/24)

Default Pod IP range Select a VPC for this server. [Click here to manage your VPCs.](#)

ADVANCED CILIUM NETWORK SETTING

Default Pod IP range Default Pod IP range will be applied to Node groups by default. Another range can be selected for Node groups if necessary.

Select range 10.111.160.0/20

Node CIDR mask size /25

Whitelist IP * 0.0.0.0 /0

2 Default Node group configuration Provide information of the default Node group.

Node group infomation

A node group is a template for groups of nodes created in this cluster. The new cluster will be created with at least one node group. More node pools can be added and removed after cluster creation.

Node group name *

Number of nodes *

Cluster configuration

Node group infomation

Cluster Name cluster-30c3ae61-be9

Kubernetes Version 1.29.1

Version vks.1724605200

Network setting

Access Public

Network type Cilium VPC Native Routing

VPC ann2 (10.111.0.0/16)

Subnet subnet1 (10.111.0.0/24)

Secondary range 10.111.160.0/20

Default Node group configuration

Node group infomation

Node group name nodegroup-0a6ba

Number of nodes 3

Node group automation setting

Auto Healing Enabled

Auto-renewal **Period** 1 month

Original price 4,356,000 VND

Total 4,356,000 VND (VAT included)

CREATE KUBERNETES CLUSTER

© 2024, VI NA DATA IT JSC Golden King Building, No. 15 Nguyen Luong Bang Street Tax code: 0304851962 Dept. of Planning and Investment of HCMC on 24/02/2003

Terms Privacy Policy SLA PCI DSS



Attention:

- **Only one networktype:** In a cluster, you can use only one of three networktypes: Calico Overlay, Cilium Overlay, or Cilium VPC Native Routing
- **Multiple subnets for a cluster:** VKS supports the use of multiple subnets for a cluster. This allows you to configure each node group in the cluster to be located on different subnets within the same VPC, helping to optimize resource allocation and network management.
- **Cilium VPC Native Routing and Secondary IP Range :** When using Cilium VPC Native Routing for a cluster, you can use multiple Secondary IP Ranges. However, each Secondary IP Range can only be used by a single cluster. This helps avoid IP address conflicts and ensures consistency in network management.
- When there are not enough IP addresses in **the Node CIDR range** or **Secondary IP range** to create more nodes, specifically:
 - If you **cannot use the new Node because of** running out of IP addresses in **the Secondary IP range** . At this time, new nodes will still be created and joined to the cluster but you cannot use them. The pods that are required to launch on this new node will be stuck in the " **ContainerCreating** " state because no suitable node can be found to deploy. At this time, you need to create a new node group with a secondary IP range that is not used on any cluster.

Step 5: Select **Create Kubernetes cluster**. Please wait a few minutes for us to initialize your Cluster, the status of the Cluster is now **Creating** .

Step 6: When the Cluster status is **Active** , you can view Cluster information and Node Group information by selecting Cluster Name in the **Name** column .

Deploy a Workload

Below are instructions for deploying an nginx deployment and testing IP assignment for the pods deployed in your cluster.

Step 1: Access <https://vks.console.vngcloud.vn/k8s-cluster>

Step 2: The Cluster list is displayed, select the icon **Download** and select **Download Config File** to download the kubeconfig file. This file will give you full access to your Cluster.

Step 3 : Rename this file to config and save it to the **~/.kube/config** folder

Step 4: Perform Cluster check via command:

- Run the following command to check **the node**

```
kubectl get nodes
```

- If the result is as below, it means your Cluster is successfully initialized with 3 nodes:

NAME	STATUS	ROLES	AGE
vks-cluster-democilium-nodegroup-558f4-39206	Ready	<none>	5m35s
vks-cluster-democilium-nodegroup-558f4-63344	Ready	<none>	5m45s
vks-cluster-democilium-nodegroup-558f4-e6e4d	Ready	<none>	6m24s

- Continue by running the following command to check the pods **deployed** on your kube-system namespace:

```
k get pods -A
```

- If the result is as below, it means that the **pods** supporting Cilium VPC Native have been running:

NAMESPACE	NAME	READY	STATUS
kube-system	cilium-envoy-2g221	1/1	Running
kube-system	cilium-envoy-h9mjb	1/1	Running
kube-system	cilium-envoy-ngz89	1/1	Running
kube-system	cilium-ft98g	1/1	Running
kube-system	cilium-operator-5fc5c56c4c-66l6d	1/1	Running
kube-system	cilium-operator-5fc5c56c4c-qfnw2	1/1	Running
kube-system	cilium-rfrr7	1/1	Running
kube-system	cilium-xmlq5	1/1	Running
kube-system	coredns-1727334052-85db76748b-fpmfr	1/1	Running
kube-system	coredns-1727334052-85db76748b-jqv79	1/1	Running
kube-system	hubble-relay-8578649fdb-bgzzz	1/1	Running
kube-system	hubble-ui-574c5bb99b-g716c	2/2	Running
kube-system	konnectivity-agent-hmf2x	1/1	Running
kube-system	konnectivity-agent-q69n2	1/1	Running
kube-system	konnectivity-agent-wgqbw	1/1	Running
kube-system	vngcloud-controller-manager-d4d4f7b84-m65nb	1/1	Running
kube-system	vngcloud-csi-controller-565c55dbcc-88pt4	7/7	Running
kube-system	vngcloud-csi-controller-565c55dbcc-v22q4	7/7	Running
kube-system	vngcloud-csi-node-665r2	3/3	Running
kube-system	vngcloud-csi-node-8x542	3/3	Running
kube-system	vngcloud-csi-node-gx7zd	3/3	Running
kube-system	vngcloud-ingress-controller-0	1/1	Running

Step 2: Deploy nginx on the newly created cluster:

- Initialize the **nginx-deployment.yaml** file with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 20
  template:
    metadata:
      labels:
        app: nginx
  spec:
    containers:
      - name: nginx
        image: nginx:latest
        ports:
          - containerPort: 80
```

- Perform this deployment via command:

```
kubectl apply -f nginx-deployment.yaml
```

Step 3: Check the deployed nginx pods and the IP address assigned to each pod

- Perform a check of the pods via the command:

```
kubectl get pods -o wide
```

- You can observe below, the **nginx pods** are assigned IPs 10.111.16x.x which satisfy **the Secondary IP range and Node CIDR mask size** conditions that we specified above:

NAME	READY	STATUS	RESTARTS	AGE	IP
nginx-app-7c79c4bf97-6v88s	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-754m7	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-9tjw7	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-c6vx7	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-c7nch	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-cggfq	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-cz4xc	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-d84rb	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-dbmt7	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-gtx8b	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-km7tx	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-lmk7c	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-mc24h	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-n4zvf	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-n84tc	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-qtjjx	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-rp4bt	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-sk7tf	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-x8jxm	1/1	Running	0	31s	10.111.16
nginx-app-7c79c4bf97-zlstg	1/1	Running	0	31s	10.111.16

- You can also perform a detailed description of each pod to check this pod information via the command:

```
kubectl describe pod nginx-app-7c79c4bf97-6v88s
```

Step 4: There are a few steps you can take to thoroughly test the performance of Cilium. Specifically:

- First, you need to install Cilium CLI following the instructions [here](#) .
- After installing Cilium CLS, check **the status** of Cilium in your cluster via the command:

```
cilium status wait
```

- If the result is displayed as below, it means that **Cilium is working properly and fully :**

```

/---\
/---\__/\---\ Cilium:          OK
\---/---\__/ Operator:        OK
/---\__/\---\ Envoy DaemonSet: OK
\---/---\__/ Hubble Relay:    OK
\---/ ClusterMesh:         disabled

DaemonSet           cilium-envoy      Desired: 3, Ready: 3/3, Available
Deployment          hubble-relay     Desired: 1, Ready: 1/1, Available
Deployment          hubble-ui       Desired: 1, Ready: 1/1, Available
Deployment          cilium-operator Desired: 2, Ready: 2/2, Available
DaemonSet           cilium          Desired: 3, Ready: 3/3, Available
Containers:
  hubble-ui        Running: 1
  cilium-operator   Running: 2
  cilium           Running: 3
  cilium-envoy     Running: 3
  hubble-relay     Running: 1

Cluster Pods:      32/32 managed by Cilium

Helm chart version:
Image versions
  cilium           vcr.vngcloud.vn/81-vks-public/c
  cilium-envoy     vcr.vngcloud.vn/81-vks-public/c
  hubble-relay     vcr.vngcloud.vn/81-vks-public/c
  hubble-ui       vcr.vngcloud.vn/81-vks-public/c
  hubble-ui       vcr.vngcloud.vn/81-vks-public/c
  cilium-operator vcr.vngcloud.vn/81-vks-public/c

```

Step 5: You can perform a healthy check to check Cilium in your cluster

- Run the following command to perform a healthy check

```
kubectl -n kube-system exec ds/cilium -- cilium-health status --probe
```

- Reference results

```

Probe time: 2024-09-26T07:11:57Z
Nodes:
  vks-cluster-democilium-nodegroup-558f4-e6e4d (localhost):
    Host connectivity to 10.111.0.8:
      ICMP to stack: OK, RTT=306.523µs
      HTTP to agent: OK, RTT=206.191µs
    Endpoint connectivity to 10.111.160.91:
      ICMP to stack: OK, RTT=307.205µs
      HTTP to agent: OK, RTT=365.113µs
  vks-cluster-democilium-nodegroup-558f4-39206:
    Host connectivity to 10.111.0.14:
      ICMP to stack: OK, RTT=1.90859ms
      HTTP to agent: OK, RTT=344.725µs
    Endpoint connectivity to 10.111.161.9:
      ICMP to stack: OK, RTT=1.889682ms
      HTTP to agent: OK, RTT=549.887µs
  vks-cluster-democilium-nodegroup-558f4-63344:
    Host connectivity to 10.111.0.9:
      ICMP to stack: OK, RTT=1.920985ms
      HTTP to agent: OK, RTT=706.376µs
    Endpoint connectivity to 10.111.160.223:
      ICMP to stack: OK, RTT=1.919709ms
      HTTP to agent: OK, RTT=1.090877ms

```

Additionally, you can also perform additional End-to-End connectivity tests or Network performance tests following the instructions at [End-To-End Connectivity Testing](#) or [Network Performance Test](#).

Step 6: Check the connection between Pods

- Perform a connectivity test between pods, ensuring that the **pods can communicate via the VPC IP address without going through overlay networks**. For example, below I perform a ping from the pod nginx-app-7c79c4bf97-6v88s with IP address: 10.111.161.53 to a server in the same VPC with IP address: 10.111.0.10:

```
kubectl exec -it nginx-app-7c79c4bf97-6v88s -- ping 10.111.0.10
```

- If the result is as follows, the connection is successful:

```
PING 10.111.0.10 (10.111.0.10): 56 data bytes
64 bytes from 10.111.0.10: seq=0 ttl=62 time=3.327 ms
64 bytes from 10.111.0.10: seq=1 ttl=62 time=0.541 ms
64 bytes from 10.111.0.10: seq=2 ttl=62 time=0.472 ms
64 bytes from 10.111.0.10: seq=3 ttl=62 time=0.463 ms
--- 10.111.0.10 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.463/1.200/3.327 ms
```

Storage

•

Working with Container Storage Interface (CSI)

•

What is CSI?

- Container Storage Interface (CSI) is a standard interface that allows containers to interact with different storage systems. CSI provides a set of common APIs that containers can use to access and manage data, regardless of the underlying storage system.

Integrate with Container Storage Interface (CSI)

•

To integrate CSI with Kubernetes cluster, follow these steps:

Prepare

- Create a Kubernetes cluster on VNGCloud, or use an existing cluster. Note: make sure you have downloaded the cluster configuration file once the cluster has been successfully initialized and accessed your cluster.

Create Service Account and install VNGCloud BlockStorage CSI Driver



Attention:

- When you initialize the Cluster according to the instructions above, if you have not enabled the **Enable BlockStore Persistent Disk CSI Driver** option , by default we will not pre-install this plugin into your Cluster. You need to manually create Service Account and install VNGCloud BlockStorage CSI Driver according to the instructions below. If you have enabled the **Enable BlockStore Persistent Disk CSI Driver** option , we have pre-installed this plugin into your Cluster, skip the Service Account Initialization step, install VNGCloud BlockStorage CSI Driver and continue following the instructions from now on. Deploy a Workload.
- **VNGCloud BlockStorage CSI Driver** only supports attaching volumes to a single node (VM) throughout the life of that volume. If you have a need for ReadWriteMany, you may consider using the NFS CSI Driver, as it allows multiple nodes to Read and Write on the same volume at the same time. This is very useful for applications that need to share data between multiple pods or services in Kubernetes.

- ✓ Create Service Account and install VNGCloud BlockStorage CSI Driver

Initialize Service Account

- Create or use a **service account** created on IAM and attach policy: **vServerFullAccess**. To create a service account, go here [and](#) follow these steps:
 - Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account
 - Find and select **Policy: vServerFullAccess**, then click " **Create a Service Account** " to create a Service Account, Policy: vLBFullAccess and Policy: vServerFullAccess are created by VNG Cloud, you cannot delete these policies.
 - After successful creation, you need to save **the Client_ID** and **Secret_Key** of the Service Account to perform the next step.

Install VNGCloud BlockStorage CSI Driver

- Install Helm version 3.0 or higher. Refer to <https://helm.sh/docs/intro/install/> for instructions on how to install.
- Add this repo to your cluster via the command:

```
helm repo add vks-helm-charts https://vngcloud.github.io/vks
helm repo update
```

- Replace your K8S cluster's ClientID, Client Secret, and ClusterID information and continue running:

Copy

```
helm install vngcloud-blockstorage-csi-driver vks-helm-chart
--replace --namespace kube-system \
--set vngcloudAccessSecret.keyId=${VNGCLOUD_CLIENT_ID} \
--set vngcloudAccessSecret.accessKey=${VNGCLOUD_CLIENT_SEC
--set vngcloudAccessSecret.vksClusterId=${VNGCLOUD_VKS_CLU
```

- After the installation is complete, check the status of vngcloud-blockstorage-csi-driver pods:

```
kubectl get pods -n kube-system | grep vngcloud-csi-
```

- For example, in the image below you have successfully installed vngcloud-blockstorage-csi-driver:

NAME	READY	STATUS
vngcloud-csi-controller-56bd7b85f-ctpns	7/7	Running
vngcloud-csi-controller-56bd7b85f-npp9n	7/7	Running
vngcloud-csi-node-c8r2w	3/3	Running

Deploy a Workload

The following is a guide for you to deploy the nginx service on Kubernetes.

Step 1 : Create Deployment for Nginx app.

- Create **nginx-service.yaml** file with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.19.1
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- Deploy This deployment equals:

```
kubectl apply -f nginx-service.yaml
```

Step 2: Check the Deployment and Service information just deployed

- Run the following command to test **Deployment**

```
kubectl get svc,deploy,pod -owide
```

- If the results are returned as below, it means you have deployed Deployment successfully.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	F	
service/kubernetes	ClusterIP	10.96.0.1	<none>	4	
service/nginx-app	NodePort	10.96.215.192	<none>	3	
service/nginx-service	LoadBalancer	10.96.179.221	<pending>	8	
NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CON
deployment.apps/nginx-app	1/1	1	1	16s	ngi
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx-app-7f45b65946-t7d7k	1/1	Running	0	16s	17

Create Persistent Volume

- Create a **persistent-volume.yaml** file with the following content:

Copy

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-expansion-storage-class          # [1] The StorageClass name
  provisioner: bs.csi.vngcloud.vn           # The VNG-CLOUD provisioner
parameters:
  type: vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018 # The volume type
  allowVolumeExpansion: true                  # MUST set this parameter
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-expansion-pvc                    # [2] The PVC name,
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi                         # [3] The PVC size,
  storageClassName: my-expansion-storage-class # [4] The StorageClass name
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx                             # [5] The Pod name,
spec:
  containers:
    - image: nginx
      imagePullPolicy: IfNotPresent
      name: nginx
      ports:
        - containerPort: 80
          protocol: TCP
      volumeMounts:
        - mountPath: /var/lib/www/html
          name: my-volume-name                # MUST be the same as the PVC name
  volumes:
    - name: my-volume-name                  # [6] The volume name
  persistentVolumeClaim:
    claimName: my-expansion-pvc            # MUST be the same as the PVC name
    readOnly: false

```

- Run the following command to deploy a Pod using a Persistent Volume

Copy

```
kubectl apply -f persistent-volume.yaml
```

At this time, the vServer system will automatically create a Volume corresponding to the yaml file above, for example:

The screenshot shows the VNG Cloud BlockStore Volumes page. The left sidebar includes sections for vServer, Route tables, Peerings, Bandwidths, Interconnects, Network ACL, Servers, Server Groups, SSH Keys, System Images, Flavors, BlockStore (Volumes, Volume Types, Images, Backups, Snapshots), Load Balancing, Load Balancers, LB Packages, and Certificates. The main content area is titled 'Volume' and defines it as a resource that goes with a virtual server, responsible for storing business data. It shows a table of volumes with columns: Name, Status, Type, IOPS, Size, Attachment, Multi-Attach, and Action. There are two volumes listed:

Name	Status	Type	IOPS	Size	Attachment	Multi-Attach	Action
vol-2f295ad8-98c4-4043-af28-98637704bb59 pvc-14456f4a-e9e9-435d-a94f-5a2e820954e9 21/04/2024 21:16:57	IN USE	SSD	200	20	ins-4ae9966b-29a1-4c01-b6eb-a543b1d2abf4	No	⋮
vol-4403a56a-11fb-470c-bf5e-56d6f134c5c1 ng-3f06013a-f6a5-47ba-a51f-be5e9c2b10a7-eceat boot_volume 19/04/2024 18:27:26	IN USE	SSD	3000	20	ins-4ae9966b-29a1-4c01-b6eb-a543b1d2abf4	No	⋮

Total: 7 Show: 10

Create Snapshots

Snapshot is a low-cost, convenient and effective data backup method and can be used to create images, restore data and distribute copies of data. If you are a new user who has never used the Snapshot service, you will need to Activate Snapshot Service before you can create a Snapshot for your Persistent Volume.

Activate Snapshot Service

To be able to create Snapshots, you need to perform Activate Snapshot Service. You will not be charged for activating the snapshot service. After you create snapshots, costs will be calculated based on the storage capacity and storage time of these snapshots. Follow these steps to enable the Snapshot service:

Step 1: Visit <https://hcm-3.console.vngcloud.vn/vserver/block-store/snapshot/overview>

Step 2: Select Activate Snapshot Service .

For example:

The screenshot shows the VNG Cloud console interface. On the left, there is a sidebar with various service categories like vServer, BlockStore, and Snapshots. Under Snapshots, the 'Snapshot' option is selected and has a 'DISABLE' status indicator. The main content area is titled 'Snapshot' and contains a 'Billing Note' section. Below it, a message says 'You have no Snapshot'. To the right of this message is an orange button labeled 'Activate Snapshot Service'. A red arrow points to this button. At the bottom of the page, there is a footer with copyright information and links to Terms, Privacy Policy, and various compliance logos.

Install VNGCloud Snapshot Controller

- Install Helm version 3.0 or higher. Refer to <https://helm.sh/docs/intro/install/> for instructions on how to install.
- Add this repo to your cluster via the command:

```
helm repo add vks-helm-charts https://vngcloud.github.io/vks-helm-charts
helm repo update
```

- Continue running:

```
helm install vngcloud-snapshot-controller vks-helm-charts/vngcloud-snapshot-controller
--replace --namespace kube-system
```

- After the installation is complete, check the status of vngcloud-blockstorage-csi-driver pods:

```
kubectl get pods -n kube-system | grep snapshot-controller
```

For example, in the image below you have successfully installed vngcloud-snapshot-controller:

NAME	READY	STATUS
snapshot-controller-7fdd984f89-745tg	0/1	ContainerCreating
snapshot-controller-7fdd984f89-k94wq	0/1	ContainerCreating

Create a snapshot.yaml file with the following content

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: my-snapshot-storage-class  # [2] The name of the volume snapshot
driver: bs.csi.vngcloud.vn
deletionPolicy: Delete
parameters:
  force-create: "false"
---

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: my-snapshot-pvc  # [4] The name of the snapshot, CAN be changed
spec:
  volumeSnapshotClassName: my-snapshot-storage-class  # MUST match with [2]
  source:
    persistentVolumeClaimName: my-expansion-pvc  # MUST match with [3]

```

- Run the following command to deploy Volume Snapshot

```
kubectl apply -f snapshot.yaml
```

Check the newly created PVC and Snapshot

- After applying the file successfully, you can check the service and pvc list via:

```
kubectl get sc,pvc,pod -owide
```

NAME	PROVISIONER				
storageclass.storage.k8s.io/my-expansion-storage-class	bs.csi.vngc				
storageclass.storage.k8s.io/sc-iops-200-retain (default)	bs.csi.vngc				
NAME	STATUS	VOLUME			
persistentvolumeclaim/my-expansion-pvc	Bound	pvc-14456f4a-ee9e-43			
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx	1/1	Running	0	10m	17
pod/nginx-app-7f45b65946-t7d7k	1/1	Running	0	94m	17

Change the IOPS parameters of the newly created Persistent Volume

To change the IOPS parameters of the newly created Persistent Volume, follow these steps:

Step 1: Run the command below to list the PVCs in your Cluster

```
kubectl get persistentvolumes
```

Step 2: Edit the PVC YAML file according to the command

```
kubectl edit pvc my-expansion-pvc
```

- If you have not edited the IOPS of the Persistent Volume before, when you run the above command, add an annotation `bs.csi.vngcloud.vn/volume-type: "volume-type-id"`. For example, below I am changing the Persistent Volume IOPS from 200 (Volume type id = `vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018`) to 1000 (Volume type id = `vtype-85b39362-a360-4bbb-9afa-a36a40cea748`)

Copy

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    bs.csi.vngcloud.vn/volume-type: "vtype-85b39362-a360-4bbb-9afa-a36
      kubectl.kubernetes.io/last-applied-configuration: |
        {"apiVersion":"v1","kind":"PersistentVolumeClaim","metadata":>{"a
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: bs.csi.vngcloud.vn
    volume.kubernetes.io/storage-provisioner: bs.csi.vngcloud.vn
  creationTimestamp: "2024-04-21T14:16:53Z"
  finalizers:
    - kubernetes.io/pvc-protection
  name: my-expansion-pvc
  namespace: default
  resourceVersion: "11041591"
  uid: 14456f4a-ee9e-435d-a94f-5a2e820954e9
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: my-expansion-storage-class
  volumeMode: Filesystem
  volumeName: pvc-14456f4a-ee9e-435d-a94f-5a2e820954e9
status:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 20Gi
  phase: Bound

```

- If you have edited the IOPS of the Persistent Volume before, when you run the above command, your yaml file will already have the annotation `bs.csi.vngcloud.vn/volume-type: "volume-type-id"` . Now, edit this annotation to the Volume type id with the IOPS you desire.

Change the Disk Volume of the newly created Persistent Volume

To change the Disk Volume of the newly created Persistent Volume, run the following command:

For example, initially the PVC created was 20 Gi in size, now I will increase it to 30 Gi

```
kubectl patch pvc my-expansion-pvc -p '{"spec":{"resources":{"requests":{}}
```

 **Attention:**

- You can only increase Disk Volume but cannot reduce Disk Volume size.

Restore Persistent Volume from Snapshot

To restore Persistent Volume from Snapshot, follow these steps:

- Create file **restore-volume.yaml** with the following content:

Copy

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-restore-pvc # The name of the PVC, CAN be changed
spec:
  storageClassName: my-expansion-storage-class
  dataSource:
    name: my-snapshot-pvc # MUST match with [4] from the section 5.2
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

CSI Limitation

•

Security Group

Security Group acts as a firewall to help you control traffic going in and out of the server (VM). On the VKS system, to ensure the cluster operates safely and effectively, default Security Groups are set up to allow necessary access for the cluster's internal operations. Automatically creating a Security Group simplifies the cluster deployment process and ensures that the cluster is protected from the start. Specifically, when you initialize a Cluster, we will automatically create several Security Groups with the following parameters:

The default security group is automatically created for all Clusters

For each Cluster created in the VKS system, we will automatically create a Security Group. This security group will include:

- Inbound:

Protocol	Ether type	Port range	Source	Meaning
TCP	IPv4	30000-32767	CIDR of the VPC you use for the Cluster.	Security group rule used for TCP Node Port Services
UDP	IPv4	30000-32767	CIDR of the VPC you use for the Cluster.	Security group rule used for UDP Node Port Services
TCP	IPv4	10250	External IP of Load Balancer used for Cluster.	Security group rule used for Kubelet API control-plane
TCP	IPv4	10250	CIDR of the VPC you use for the Cluster.	Security group rule used for

				Kubelet API control-plane
TCP	IPv4	179	CIDR of the VPC you use for the Cluster.	Security group rule used for Kubelet API control-plane
4	IPv4	1-65535	CIDR of the VPC you use for the Cluster.	Security group rule used for Calico IP-in-IP
TCP	IPv4	5473	CIDR of the VPC you use for the Cluster.	Security group rule used for Calico Typha

- Outbound

Protocol	Ether type	Port range	Destination	Meaning
ANY	IPv4	0-65535	0.0.0.0/0	Default rule of all Security groups
ANY	IPv6	0-65535	::/0	Default rule of all Security groups

Security group is automatically created by VNGCLOUD Controller Manager

When you use VNGCloud Controller Manager to integrate Network Load Balancer with Cluster on VKS system, we will automatically create a Security Group. This security group will include:

- Inbound:

Protocol	Ether type	Port range	Source
TCP, UDP or ICMP	IPv4	Port of Service	Subnet Mask of the Subnet you

use for the Cluster.

- Outbound:

Protocol	Ether type	Port range	Destination	Meaning
ANY	IPv4	0-65535	0.0.0.0/0	Default rule of all Security groups
ANY	IPv6	0-65535	::/0	Default rule of all Security groups

Security group is automatically created by VNGCLOUD Ingress Controller

When you use VNGCloud Ingress Controller to integrate Application Load Balancer with Cluster on VKS system, we will automatically create a Security Group. This security group will include:

- Inbound:

Protocol	Ether type	Port range	Source
TCP	IPv4	Port of Service	Subnet Mask of the Subnet you use for the Cluster.

- Outbound:

Protocol	Ether type	Port range	Destination	Meaning
ANY	IPv4	0-65535	0.0.0.0/0	Default rule of all Security groups
ANY	IPv6	0-65535	::/0	Default rule of all Security groups



Attention:

- Default Security Groups are set up to meet the basic security needs of the cluster. If you edit or delete the Security Groups created for the cluster, it may result in connectivity and access issues between nodes in the cluster or the cluster may not function correctly or may not even start. To ensure the stability and security of the cluster, the system will automatically reset Security Groups to default settings after every fixed period of time.

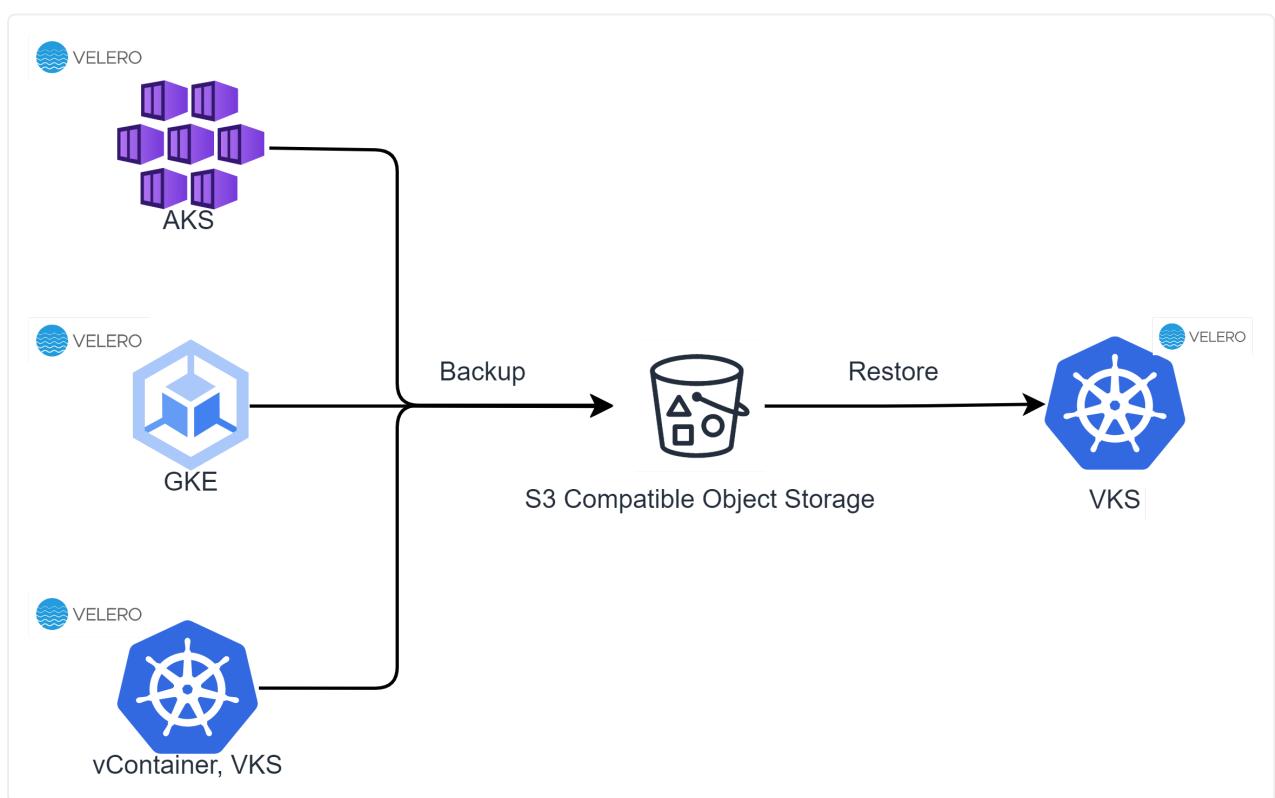
Migration

•

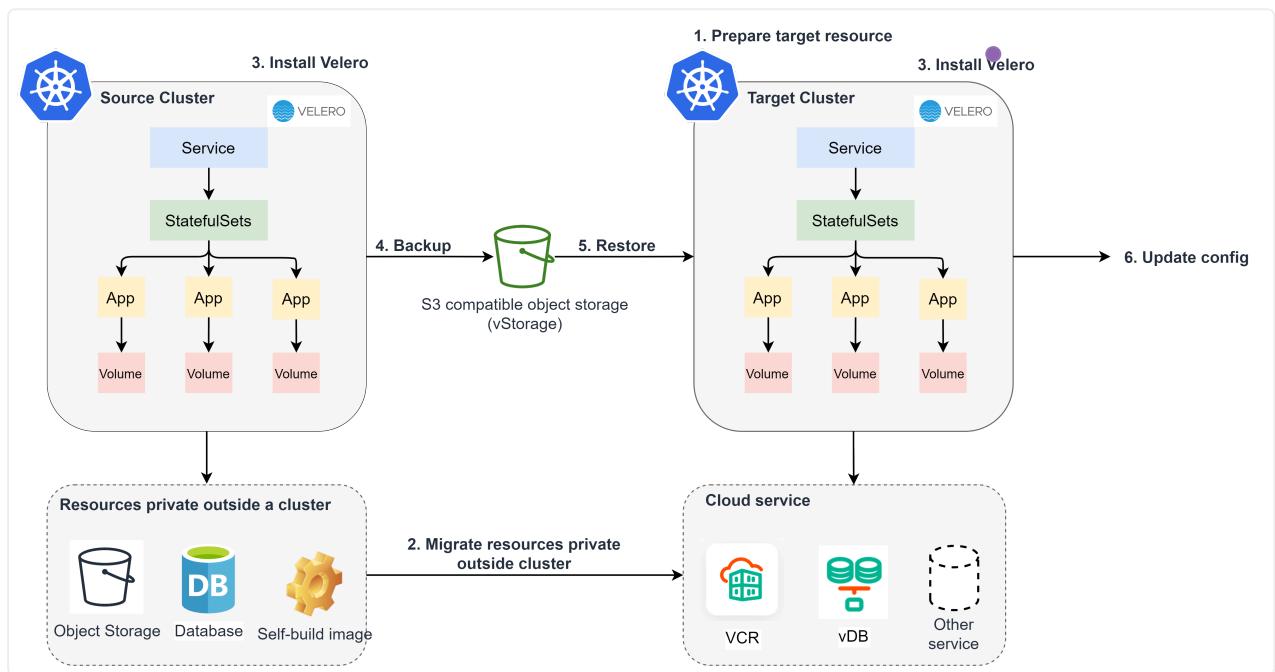
Overview

Migration from one cluster to another is the process of moving data, applications, and services from one set of servers to another. The purpose of this is often to upgrade the system, increase availability and fault tolerance, or to scale the system.

Model



Implementation steps



Specifically:

- **Step 1: Prepare target cluster (Prepare target resource)**: on the VKS system, you need to initialize a Cluster according to the instructions [here](#). Make sure that the destination cluster's configuration is the same as the source cluster's configuration.
- **Step 2 [Optional] :** If your cluster has private resources such as images, databases, storage... Now, before starting to migrate, you need to **proactively** migrate these resources yourself.
- **Step 3 :** Install Velero on both source and destination clusters (Install Velero tool): after you have migrated private resources outside the cluster, you can use the migration tool to backup and restore. Restore the application on the source cluster and target cluster.
- **Step 4 :** Backup: To back up resources, use the Velero tool to create a backup object in the source cluster. Velero will perform queries, package the data, and upload them to an S3 Compatible Object Storage.
- **Step 5 :** Restore: During the restore process at the destination cluster, Velero will download backup data to the new cluster and redeploy resources based on the JSON file.
- **Step 6 [Optional] :** Update resource config: After the target cluster's resources are deployed properly, you can **switch traffic** for your service. After confirming that all services are running properly, you can delete the source cluster.

Below are detailed instructions for common cases when you migrate workloads from one Cluster to another. You can refer to and follow the instructions at:

- [Migrate Cluster from VKS to VKS](#)
- [Migrate Cluster from vContainer to VKS](#)
- [Migrate Cluster from another platform to VKS](#)

Migrate Cluster from VKS to VKS

To migrate a Cluster from the VKS system to the VKS system, follow the steps in this document.

Prerequisites

- **Perform download helper bash script and grant execute permission for this file ([velero_helper.sh](#))**
- (Optional) Deploy some services to check the correctness of the migration.
Suppose, at the source Cluster, I have deployed an nginx service as follows:
 - Deployment files:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: mynamespace
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    name: web
  selector:
    app: nginx
  type: NodePort
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
  namespace: mynamespace
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
  spec:
    containers:
    - name: nginx
      image: nginx
      ports:
      - containerPort: 80
        name: web
      volumeMounts:
      - name: disk-ssd
        mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: disk-ssd
      namespace: mynamespace
  spec:
    accessModes: [ "ReadWriteOnce" ]
    storageClassName: ssd-3000
    resources:
```

```
requests:  
  storage: 40Gi
```

Copy

```
kubectl exec -n mynamespace -it web-0 bash  
cd /usr/share/nginx/html  
echo -e "<html>\n<head>\n  <title>MyVNGCloud</title>\n</head>\n<body>\n  <h1>Hello, MyVNGCloud!</h1>\n</body>\n</html>" > index.html
```

- Now, when you access Node's Public IP, you will see "Hello, MyVNGCloud".

Prepare target cluster (Prepare target resource)

On the VKS system, you need to initialize a Cluster according to the instructions [here](#). Make sure that the destination cluster's configuration is the same as the source cluster's configuration.



Attention:

For the migration to be successful, on the target Cluster, you need to ensure the following requirements:

- The amount of resources needed such as number of nodes, node instance configuration,...
- Node labels and node taints are the same as the old cluster.
- Corresponding or alternative Storage Class.

[Optional] Migrate private resources outside cluster

Migrating private resources outside cluster (moving private resources outside the cluster) is the process of moving private resources outside the source Cluster to a place that the destination Cluster can use. For example, you may have private resources such as images, databases, etc. Now, before starting to migrate, you need to migrate these resources yourself. For example, if you need:

- Migrate Container Images: you can migrate images to VNGCloud Container Registry through instructions [here](#).
- Migrate Databases: you can use **Relational Database Service (RDS)** and **Object Storage Service (OBS)** depending on your needs. After the migration is complete, remember to reconfigure the database for your applications on VKS Cluster.
- Migrate Storage: you can use vServer's **NFS Server**.

 **Attention:**

- After you migrate resources outside the Cluster, you need to ensure the target Cluster can connect to these migrated resources.

Install Velero on both source and destination clusters (Install Velero tool)

After you have migrated private resources outside the cluster, you can use the migration tool to backup and restore the application on the source cluster and target cluster.

- Create a **vStorage Project, Container** to receive the cluster's backup data according to instructions [here](#).
- Create an S3 key corresponding to this vStorage Project according to the instructions [here](#).

For example, I have initialized a vStorage Project, Container with the following information: Region: HCM03, Container: mycontainer, Endpoint: <https://hcm03.vstorage.vngcloud.vn>.

On both Clusters (source and target)

- Create file **credentials-velero** with the following content:

```
[default]
aws_access_key_id=-----
aws_secret_access_key=-----
```

- Install Velero CLI:

```
curl -OL https://github.com/vmware-tanzu/velero/releases/download/v1.1
tar -xvf velero-v1.13.2-linux-amd64.tar.gz
cp velero-v1.13.2-linux-amd64/velero /usr/local/bin
```

- Install Velero on your 2 clusters with the command:

```
velero install --provider aws \
--plugins velero/velero-plugin-for-aws:v1.9.0,velero/velero-plugin-f
--secret-file ./credentials-velero \
--bucket ----- \
--backup-location-config region=hcm03,s3ForcePathStyle="true",s3Url=
--use-node-agent \
--features=EnableCSI
```

```
velero client config set features=EnableCSI
```

Attention:

- When you perform a cluster migration from VKS to VKS, we recommend that you use **Snapshot** to migrate your Volume from the source cluster to the destination cluster.
- Install the **VNGCloud Snapshot Controller** plugin on 2 clusters with the command:
Copy

```
helm repo add vks-helm-charts https://vngcloud.github.io/vks-helm-char
helm repo update
helm install vngcloud-snapshot-controller vks-helm-charts/vngcloud-sna
--replace --namespace kube-system
```

At the source Cluster

- Annotate the Persistent Volumes that need to be backed up. By default, Velero will not backup volume. You can run the command below to annotate backup of all volumes.

```
./velero_helper.sh mark_volume -c
```

- Additionally, you can mark not to backup system resources with the following command:

```
./velero_helper.sh mark_exclude -c
```

- Apply the file below to create the default VolumeSnapshotClass:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: vngcloud-vsclass
  labels:
    velero.io/csi-volumesnapshot-class: "true"
driver: bs.csi.vngcloud.vn
deletionPolicy: Delete

# user can choose the VolumeSnapshotClass by setting annotation velero
# user can choose the VolumeSnapshotClass by setting annotation velero
```

- Perform backup according to the syntax:

```
velero backup create vks-full-backup \
  --exclude-namespaces velero \
  --include-cluster-resources=true \
  --wait
```

```
velero backup describe vks-full-backup --details
```

At the destination Cluster

- Perform restore according to the command:

```
velero restore create --from-backup vks-full-backup \  
--exclude-resources="MutatingWebhookConfiguration,ValidatingWebhook
```

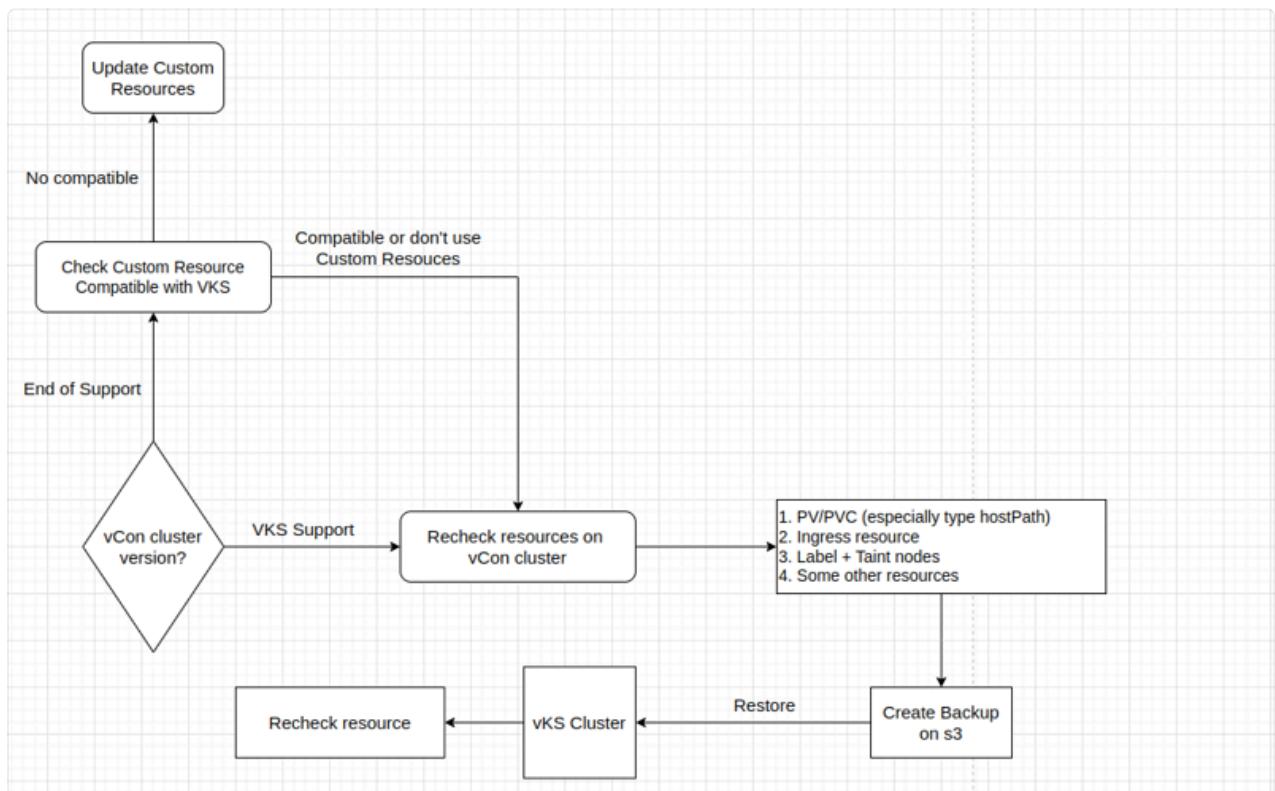
```
velero restore create --from-backup vks-full-backup
```

Migration Cluster from vContainer to VKS

•

Tools used: Velero + vStorage (s3)

1. Process



Process of migrating from vContainer to VKS (Customer + VNG Cloud)

Step 1: Evaluate the current version of the vContainer cluster and the corresponding vKS cluster to migrate. At this point, there will be 2 cases (step 2 and step 3)

Step 2: If the vContainer version is lower than the versions currently supported by vKS, then the Custom Resources (CRDs) need to be reviewed for compatibility with the new kubernetes versions.

- If compatible with the new vKS version, continue with step 3.

- If not compatible, need to manually update and reconfigure CRD as well as related Applications.

Step 3: If the vContainer version is supported by vKS (1.27, 1.28 and 1.29), then you need to check the resources on vContainer before backing up. You need to pay attention to the following types of resources:

- **PV/PVC:** Velero does not support backup with hostPath type, only Local type is supported.
- **Ingress resources :** Ingress resources managed by **container-ingress-nginx-controller** will not work after migration.
- **Label and Taint nodes :** Velero does not re-attach Labels and Taints to nodes in the vKS

Step 4 : Backup resources on vContainer cluster

Step 5 : Restore resources on vKS cluster

Step 6 : Perform testing and adjustments

2. Important Note:

1. Mapping StorageClass on vKS cluster

Check the existing StorageClasses on vContainer and create corresponding StorageClasses on vKS. Then perform 1:1 mapping of StorageClasses between vContainer and vKS cluster.

Example ConfigMap file in **Step 3.**

2. Use PersistentVolume as hostPath

Velero does not support hostPath volume backup, only local.

For clusters using PV type hostPath, it is necessary to convert to local type as follows:

Note: Need to delete and redeploy the Application using PV type hostPath

Case 1: Type Local supports hostPath path

For example, PV hostPath is configured to mount at the /opt/data folder, convert to Local type and mount back to Pods, according to the following form:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: manual
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  local:
    path: "/opt/data"
  nodeAffinity:
    required:
      nodeSelectorTerms:
```

```
- matchExpressions:
  - key: kubernetes.io/hostname
    operator: Exists
  ...
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Case 2: Type Local does not support the original hostPath path

When creating PV/PVC according to Case 1, Pod cannot mount PVC and an error appears

MountVolume.NewMounter initialization failed for volume "pvc" : path "/mnt/data" does not exist

Copy the data to a new folder (eg /var, /opt, /tmp, ...) and repeat Case 1, then mount the PVC into the Pod as usual:

```
cp -R /mnt/data /var
```

3. Detailed implementation steps

Step 1: Install Velero on both clusters (vContainer and vKS)

Create a vStorage project, Container and corresponding S3 key to store backup data

On both clusters:

- Create a **credentials-velero** file with the following content:

```
[default]
aws_access_key_id=_----- # <= Adjust here
aws_secret_access_key=_----- # <= Adjust here
```

- Install Velero CLI

```
curl -OL https://github.com/vmware-tanzu/velero/releases/download/v1.14.
tar -xvf velero-v1.14.1-linux-amd64.tar.gz
cp velero-v1.14.1 -linux-amd64/velero /usr/local/bin
```

- Install Velero on 2 kubernetes clusters

```
velero install \  
  --provider aws \  
  --plugins velero/velero-plugin-for-aws:v1.9.0 \  
  --use-node-agent \  
  --use-volume-snapshots=false \  
  --secret-file ./credentials-velero \  
  --bucket _____\# <= Adjust here \  
  --backup-location-config region=hcm03,s3ForcePathStyle="true",s3Url=h
```

Step 2: Perform backup on vContainer Cluster

- Annotate the Persistent Volumes to be backed up

```
./velero_helper.sh mark_volume --confirm
```

- Annotate non-backup resources of kube-system

```
./velero_helper.sh mark_exclude --confirm
```

- Annotate other resources (not marked in the velero_helper.sh file), such as CSI, Ingress Controller, or other resources that you do not want to migrate (note that you need to label all resources of objects that do not need to be backed up).
 - For example, an application includes DaemonSet, Deployment, Pod, ... then it is necessary to mark labels for all those resources.

```
# Thêm label velero.io/exclude-from-backup=true cho từng resources

## Với Cinder CSI

kubectl -n kube-system label StatefulSet/csi-cinder-controllerplugin velero.io/exclude-from-backup=true
kubec kubectl -n kube-system label DaemonSet/csi-cinder-nodeplugin velero.io/exclude-from-backup=true

## Với vcontainer-ingress-nginx-controller

kubectl -n kube-system label Deployment/ vcontainer-ingress-nginx-control velero.io/exclude-from-backup=true
kubectl -n kube-system label Deployment/ vcontainer-ingress-nginx-default velero.io/exclude-from-backup=true
```

- Perform check and attach Labels and Taints on vContainer nodes to vKS nodes before restore

```
# Kiểm tra các Label nodes
./velero_helper.sh check_node_label
# Kiểm tra các Taint nodes
./velero_helper.sh check_node_taint
```

- Create 2 backups for Cluster resources and Namespace resources according to the syntax

```
# Tạo cluster resource backup
velero backup create vcontainer-cluster --include-namespaces "" --includ

# Tạo cluster namespace backup
velero backup create vcontainer-namespace --exclude-namespaces velero --w

# Xóa các bản backup (nếu cần)
velero backup delete vcontainer-namespace vcontainer-cluster --confirm
```

- View created backups and details of backed up resources (note the **STATUS** of the backup)

```
# List các bản backup được tạo
velero get backup

# Chi tiết của một bản backup

velero backup describe <bk-name> --details

# Xem logs quá trình backup

velero backup logs <bk-name>
```

Step 3: Perform Restore on VKS Cluster

- If in the vContainer cluster using CSI is **cinder.csi.openstack.org**, need to perform StorageClass mapping between 2 clusters vContainer and vKS
 - Mapping **csi-sc-cinderplugin-nvme-5000** (vContainer) and **vngcloud-nvme-5000-delete** (vKS), similarly for other StorageClasses
 - In vKS, it is necessary to create corresponding StorageClasses.

Create **sc-mapping.yaml** file and apply on VKS cluster

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: change-storage-class-config
  namespace: velero
  labels:
    velero.io/plugin-config: ""
    velero.io/change-storage-class: RestoreItemAction
data:
  csi-sc-cinderplugin-nvme-5000: vngcloud-nvme-5000-delete
  #-----old_storage_class-----: -----new_storage_class----- # <=

```

- Add permissions for Velero to restore data PersistentVolume

Create **add-permission.yaml** file and apply on VKS cluster

```

apiVersion: v1

kind: ConfigMap

metadata:

  name: fs-restore-action-config

  namespace: velero

  labels:

    velero.io/plugin-config: ""
    velero.io/pod-volume-restore: RestoreItemAction

data:

  secCtx: |

    capabilities:

      drop: []

      add: []

    allowPrivilegeEscalation: false

    readOnlyRootFilesystem: true

    runAsUser: 0

    runAsGroup: 0

```

- Perform restore in order
 - Note, for each restore, you need to check whether the restore process was successful or not before continuing to execute other commands.

```

# Kiểm tra restore

velero get restore

velero describe restore <restore-name> --details

```

```
velero restore create --item-operation-timeout 1m --from-backup vcontainer
```

```
velero restore create --item-operation-timeout 1m --from-backup vcontainer
```

```
velero restore create --item-operation-timeout 1m --from-backup vcontainer
```

- In case of migrating to VKS and still using vcontainer-ingress-controller, it is necessary to change the Service type to LoadBalancer

```
kubectl patch service -n kube-system vcontainer-ingress-nginx-controller
```

Migrate Cluster from another platform to VKS

•

To migrate a Cluster from the Cloud Provider or On-premise system to the VKS system, follow the steps in this document.

Prerequisites

- **Perform download helper bash script and grant execute permission for this file ([velero_helper.sh](#))**
- (Optional) Deploy some services to check the correctness of the migration. Suppose, at the source Cluster, I have deployed an nginx service as follows:
 - Deployment files:

```

apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: mynamespace
  labels:
    app: nginx
spec:
  ports:
    - port: 80
      name: web
  selector:
    app: nginx
  type: NodePort
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
  namespace: mynamespace
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
              name: web
          volumeMounts:
            - name: disk-ssd
              mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
    - metadata:
        name: disk-ssd
        namespace: mynamespace
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: standard-rwo
      resources:

```

```
requests:  
  storage: 40Gi
```

```
kubectl exec -n mynamespace -it web-0 bash  
cd /usr/share/nginx/html  
echo -e "<html>\n<head>\n  <title>MyVNGCloud</title>\n</head>\n<body>\n
```

- Now, when you access Node's Public IP, you will see "Hello, MyVNGCloud".

Prepare target cluster (Prepare target resource)

On the VKS system, you need to initialize a Cluster according to the instructions [here](#). Make sure that the destination cluster's configuration is the same as the source cluster's configuration.

 **Attention:**

For the migration to be successful, on the target Cluster, you need to ensure the following requirements:

- The amount of resources needed such as number of nodes, node instance configuration,...
- Node labels and node taints are the same as the old cluster.
- Corresponding or alternative Storage Class.

[Optional] Migrate private resources outside cluster

Migrating private resources outside cluster (moving private resources outside the cluster) is the process of moving private resources outside the source Cluster to a place that the destination Cluster can use. For example, you may have private resources such as images, databases, etc. Now, before starting to migrate, you need to migrate these resources yourself. For example, if you need:

- Migrate Container Images: you can migrate images to VNGCloud Container Registry through instructions [here](#).
- Migrate Databases: you can use **Relational Database Service (RDS)** and **Object Storage Service (OBS)** depending on your needs. After the migration is complete, remember to reconfigure the database for your applications on VKS Cluster.
- Migrate Storage: you can use vServer's **NFS Server**.

 **Attention:**

- After you migrate resources outside the Cluster, you need to ensure the target Cluster can connect to these migrated resources.

Install Velero on both source and destination clusters (Install Velero tool)

After you have migrated private resources outside the cluster, you can use the migration tool to backup and restore the application on the source cluster and target cluster.

- Create a **vStorage Project, Container** to receive the cluster's backup data according to instructions [here](#).
- Create an S3 key corresponding to this vStorage Project according to the instructions [here](#).

For example, I have initialized a vStorage Project, Container with the following information: Region: HCM03, Container: mycontainer, Endpoint: <https://hcm03.vstorage.vngcloud.vn>.

On both Clusters (source and target)

- Create file **credentials-velero** with the following content:

```
[default]
aws_access_key_id=-----
aws_secret_access_key=-----
```

- Install Velero CLI:

```
curl -OL https://github.com/vmware-tanzu/velero/releases/download/v1.1
tar -xvf velero-v1.13.2-linux-amd64.tar.gz
cp velero-v1.13.2-linux-amd64/velero /usr/local/bin
```

- Install Velero on your 2 clusters with the command:

```
velero install \
--provider aws \
--plugins velero/velero-plugin-for-aws:v1.9.0 \
--use-node-agent \
--use-volume-snapshots=false \
--secret-file ./credentials-velero \
--bucket ----- \
--backup-location-config region=hcm03,s3ForcePathStyle="true",s3Ur
```

For Clusters on Amazon Elastic Kubernetes Service (EKS)

At the source Cluster

- Annotate the Persistent Volumes that need to be backed up. By default, Velero will not backup volume. You can run the command below to annotate backup of all volumes.

```
./velero_helper.sh mark_volume -c
```

- Additionally, you can mark not to backup system resources with the following command:

```
./velero_helper.sh mark_exclude -c
```

- Perform backup according to the syntax:

```
velero backup create eks-cluster --include-namespaces "" \  
--include-cluster-resources=true \  
--wait
```

```
velero backup create eks-namespace --exclude-namespaces velero \  
--wait
```

 **Attention:**

- You must create 2 backup versions for Cluster Resource and Namespace Resource.

At the destination Cluster

- Create a Storage Class mapping file between source and destination Cluster:

```
apiVersion: v1  
kind: ConfigMap  
metadata:  
  name: change-storage-class-config  
  namespace: velero  
  labels:  
    velero.io/plugin-config: ""  
    velero.io/change-storage-class: RestoreItemAction  
data:  
  old_storage_class: new_storage_class #  
  old_storage_class: new_storage_class #
```

- Perform restore according to the command:

```
velero restore create --item-operation-timeout 1m --from-backup eks-cl  
--exclude-resources="MutatingWebhookConfiguration,ValidatingWebhoc
```

```
velero restore create --item-operation-timeout 1m --from-backup eks-na
```

```
velero restore create --item-operation-timeout 1m --from-backup eks-cl
```

For Cluster on Google Kubernetes Engine (GKE)

At the source Cluster

- Annotate the Persistent Volumes and label resources that need to be excluded from the backup

```
./velero_helper.sh mark_volume -c
```

- Additionally, you can mark not to backup system resources with the following command:

```
./velero_helper.sh mark_exclude -c
```

- Perform backup according to the syntax:

```
velero backup create gke-cluster --include-namespaces "" \  
--include-cluster-resources=true \  
--wait
```

```
velero backup create gke-namespace --exclude-namespaces velero \  
--wait
```

Attention:

- You must create 2 backup versions for Cluster Resource and Namespace Resource.

At the destination Cluster

- Create a Storage Class mapping file between source and destination Cluster:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: change-storage-class-config
  namespace: velero
  labels:
    velero.io/plugin-config: ""
    velero.io/change-storage-class: RestoreItemAction
data:
  old_storage_class: new_storage_class #
  old_storage_class: new_storage_class #
```

- Perform restore according to the command:

```
velero restore create --item-operation-timeout 1m --from-backup gke-cl
--exclude-resources="MutatingWebhookConfiguration,ValidatingWebhookConfiguration"
```

```
velero restore create --item-operation-timeout 1m --from-backup gke-na
```

```
velero restore create --item-operation-timeout 1m --from-backup gke-cl
```



Attention:

- Google Kubernetes Engine (GKE) does not allow daemonset deployment on all nodes. However, Velero only needs to deploy the daemonset on the node with the PV mount. The solution to this problem is that you can adjust the taint and toleration of the daemonset to only deploy it on the node with the PV mount.
- You can change the default resource request `cpu:500m` and `mem:512M` in the installation step or make adjustments when deploying the yaml.

Migrate Limitation

•

When using Velero to migrate Cluster to Cluster, you can add the following options.

Mark the Volumes you want to backup and unnecessary resources

To mark the Volumes you want to backup and unnecessary resources, first you need to download the bash helper script we provide and perform grand execute permission. You can see details of the tab file at: [velero_helper.sh](#)

1. Convert hostPath Volume to Persistent Volume to be able to perform backup

Since Velero does not support backing up hostPath Volume, you need to convert hostPath Volume to Persistent Volume according to the following instructions:

- To list hostPath Volumes in use:

```
./helper.sh check_hostPath
```

2. Mark Persistent Volume to include in backup

All data Persistent Volumes are stored on vStorage. Need to add annotation for all pods using PV with volume name:

```
backup.velero.io/backup-volumes=volume1, volume2
```

- Or you can automatically find volumes by:

```
./helper.sh mark_volume
```

3. Mark resources in exclude in backup

Because VKS operates under the Fully Managed Control Plane mechanism, you do not need to backup resources such as: `calico`, `kube-dns`, `kube-scheduler`, `kube-apiserver`, ... In addition, vContainer resources such as: `magnitude-auto-healer`, `cluster-autoscaler`, `csi-cinder`, ... will also be ignored.

- Identify resources that do not need backup via the command:

```
./helper.sh mark_exclude
```

4. Check label and taint of node

When performing a migration, it is possible that the resources in the source Cluster are using labels and taints. You need to ensure these important labels and taints exist in the target Cluster.

- Check lable and taint via command:

```
./helper.sh check_node_label  
./helper.sh check_node_taint
```

5. Mapping Storage Class

- If your Storage Class is different between the source Cluster and the destination Cluster, you need to transfer the Storage Class between the two clusters. For example:

- At the source Cluster, you have the following 2 Storage Classes:

```
@ kubectl get sc  
NAME           PROVISIONER      RECLAIMPOLICY  
sc-iops-200-retain (default) csi.vngcloud.vn Retain  
sc-ssd-10000-delete (default) csi.vngcloud.vn Delete
```

- You can create a mapping file with content like the example below to convert 2 storage classes from the source Cluster into 2 storage classes at the destination Cluster. This file must be applied at the target Cluster before you run the backup command:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: change-storage-class-config
  namespace: velero
  labels:
    velero.io/plugin-config: ""
    velero.io/change-storage-class: RestoreItemAction
data:
  sc-iops-200-retain: ssd-200
  sc-ssd-10000-delete: ssd-10000
```

Working VKS with Terraform •

What is Terraform?

Terraform is an open source infrastructure as code tool that allows users to manage their infrastructure easily and efficiently across different cloud platforms, such as VNG Cloud, AWS, Google Cloud and Azure. Terraform Server refers to the instance of the Terraform engine running on a specific server or machine. This is where infrastructure code is written and executed, allowing users to create, modify, and destroy resources on the cloud platform.

Terraform itself does not have a graphical user interface, instead users interact with it using a command line interface. Terraform requires a cloud provider account and key to be configured along with a Terraform configuration file to execute the infrastructure as code. Additionally, Terraform can operate in clustered environments where multiple users can collaborate on the same infrastructure codebase, making it a powerful and flexible tool for infrastructure management.

Implementation steps

To initialize a Kubernetes Cluster using Terraform, you need to perform the following steps:

1. **Access the IAM Portal here** , create a Service Account with [VKS Full Access](#) authority . Specifically, at the IAM site, you can:
 - Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account.
 - Find and select **Policy: VKSFullAccess** then click " **Create a Service Account** " to create a Service Account, **Policy: VKSFullAccess** is created by VNG Cloud, you cannot delete these policies.

- After successful creation, you need to save **the Client_ID** and **Secret_Key** of the Service Account to perform the next step.
- 2. Access the VKS Portal here , Activate [the](#) VKS service on the Overview tab.**
Please wait until we successfully create your VKS account.
 - 3. Install Terraform:**
 - Download and install Terraform for your operating system from <https://developer.hashicorp.com/terraform/install> .
 - 4. Initialize Terraform configuration:**
 - Create a file `variable.tf` and declare Service Account information in this file.
 - Create a file `main.tf` and define the Kubernetes Cluster resources you want to create.

For example:

- The file `variable.tf`: you need to replace the Client ID and Client Secret created in step 1 in this file.

```
variable "client_id" {
  type = string
  default = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
variable "client_secret" {
  type = string
  default = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

-
- On the **main.tf** file , you need to be able to add resources to create a Cluster/ Node Group:
 - Create independent Cluster my-vks-cluster and Node Group my-nodegroup:

```

resource "vngcloud_vks_cluster" "primary" {
  name      = "my-cluster"
  cidr      = "172.16.0.0/16"
  vpc_id    = "net-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  subnet_id = "sub-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}

resource "vngcloud_vks_cluster_node_group" "primary" {
  name= "my-nodegroup"
  ssh_key_id= "ssh-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  cluster_id= vngcloud_vks_cluster.primary.id
}

```

- Create Cluster with Default Node Group

```

resource "vngcloud_vks_cluster" "primary" {
  name      = "my-cluster"
  cidr      = "172.16.0.0/16"
  vpc_id    = "net-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  subnet_id = "sub-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  node_group {
    name= "my-nodegroup"
    ssh_key_id= "ssh-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  }
}

```

Attention:

- We recommend that you create and manage Clusters and Node Groups as separate resources, as in the example below. This allows you to add or remove Node Groups without recreating the entire Cluster. If you declare Node Group Default directly in the vngcloud_vks_cluster resource, you cannot delete them without recreating the Cluster itself.
- In the main.tf file, to successfully create a cluster with a node group, you must enter information in the following 4 fields:

```

vpc_id    = "net-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
subnet_id = "sub-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
ssh_key_id= "ssh-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

```

Example 1:

Below is the main.tf file I used to initialize the Cluster with the following parameters:

- Cluster name: cluster-demo
- K8S Version: v1.29.1
- Mode: Public Cluster and Public Node Group
- Node Group name: nodegroup1
- Initial Node: 3
- Turn on AutoScaling: scale from 0 to 5 nodes

```

terraform {
  required_providers {
    vngcloud = {
      source  = "vngcloud/vngcloud"
      version = "1.2.2"
    }
  }
}

provider "vngcloud" {
  token_url      = "https://iamapis.vngcloud.vn/accounts-api/v2/auth/to
  client_id      = var.client_id
  client_secret   = var.client_secret
  vserver_base_url = "https://hcm-3.api.vngcloud.vn/vserver/vserver-gatew
  vlb_base_url    = "https://hcm-3.api.vngcloud.vn/vserver/vlb-gateway"
}

resource "vngcloud_vks_cluster" "primary" {
  name      = "cluster-demo"
  description = "Cluster create via terraform"
  version   = "v1.29.1"
  cidr      = "172.16.0.0/16"
  enable_private_cluster = false
  network_type = "CALICO"
  vpc_id     = "net-70ef12d4-d619-43fc-88f0-1c1511683123"
  subnet_id  = "sub-0725ef54-a32e-404c-96f2-34745239c123"
  enabled_load_balancer_plugin = true
  enabled_block_store_csi_plugin = true
}

resource "vngcloud_vks_cluster_node_group" "primary" {
  cluster_id = vngcloud_vks_cluster.primary.id
  name = "nodegroup1"
  num_nodes = 3
  auto_scale_config {
    min_size = 0
    max_size = 5
  }
  upgrade_config {
    strategy = "SURGE"
    max_surge = 1
    max_unavailable = 0
  }
  image_id = "img-108b3a77-ab58-4000-9b3e-190d0b4b07fc"
  flavor_id = "flav-9e88cfb4-ec31-4ad4-8ba5-243459f6d123"
  disk_size = 50
  disk_type = "vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018"
  enable_private_nodes = false
}

```

```
--  
  ssh_key_id= "ssh-f923c53c-cba7-4131-9f86-175d04ae2123"  
  security_groups = ["secg-faf05344-fbd6-4f10-80a2-cda08d15ba5e"]  
  labels = {  
    "test" = "terraform"  
  }  
  taint {  
    key      = "key1"  
    value    = "value1"  
    effect   = "PreferNoSchedule"  
  }  
}  
}
```

Example 2

Below is the main.tf file I used to initialize the Cluster with the following parameters:

- Cluster name: my-cluster
- K8S Version: v1.29.1
- Mode: Public Cluster and Private Node Group
- Node Group name: my-nodegroup
- Initial Node: 3 nodes
- Turn on AutoScaling: scale from 0 to 5 nodes

First, apply the main file according to the following structure:

```

terraform {
  required_providers {
    vngcloud = {
      source  = "vngcloud/vngcloud"
      version = "1.2.2"
    }
  }
}

provider "vngcloud" {
  token_url      = "https://iamapis.vngcloud.vn/accounts-api/v2/auth/to
  client_id      = var.client_id
  client_secret   = var.client_secret
  vserver_base_url = "https://hcm-3.api.vngcloud.vn/vserver/vserver-gatew
  vlb_base_url    = "https://hcm-3.api.vngcloud.vn/vserver/vlb-gateway"
}

resource "vngcloud_vks_cluster" "primary" {
  name      = "my-cluster"
  description = "VNGCLOUD uses terraform"
  version = "v1.29.1"
  cidr      = "172.16.0.0/16"
  enable_private_cluster = false
  network_type = "CALICO"
  vpc_id     = "net-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"
  subnet_id  = "sub-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"
  enabled_load_balancer_plugin = true
  enabled_block_store_csi_plugin = true
}

resource "vngcloud_vks_cluster_node_group" "primary" {
  cluster_id= vngcloud_vks_cluster.primary.id
  name= "my-nodegroup"
  num_nodes = 3
  auto_scale_config {
    min_size = 0
    max_size = 5
  }
  upgrade_config {
    strategy = "SURGE"
    max_surge = 1
    max_unavailable = 0
  }
  image_id = "img-108b3a77-ab58-4000-9b3e-190d0b4b07fc"
  flavor_id = "flav-9e88cfb4-ec31-4ad4-8ba5-243459f6dc4b"
  disk_size = 20
  disk_type = "vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018"
  enable_private_nodes = true
}

```

```
--  
ssh_key_id= "ssh-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"  
labels = {  
    "mylabel" = "vngcloud"  
}  
taint {  
    key     = "mykey"  
    value   = "myvalue"  
    effect = "PreferNoSchedule"  
}  
}
```

Then, if you need to add Whitelist IP for Control Plane, add this field to the main.tf file and reapply this file:

```

terraform {
  required_providers {
    vngcloud = {
      source  = "vngcloud/vngcloud"
      version = "1.2.2"
    }
  }
}

provider "vngcloud" {
  token_url      = "https://iamapis.vngcloud.vn/accounts-api/v2/auth/to
  client_id      = var.client_id
  client_secret   = var.client_secret
  vserver_base_url = "https://hcm-3.api.vngcloud.vn/vserver/vserver-gatew
  vlb_base_url    = "https://hcm-3.api.vngcloud.vn/vserver/vlb-gateway"
}

resource "vngcloud_vks_cluster" "primary" {
  name      = "my-cluster"
  description = "VNGCLOUD uses terraform"
  version = "v1.29.1"
  cidr      = "172.16.0.0/16"
  white_list_node_cidr = "172.25.32.1/16"
  enable_private_cluster = false
  network_type = "CALICO"
  vpc_id     = "net-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"
  subnet_id  = "sub-xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx"
  enabled_load_balancer_plugin = true
  enabled_block_store_csi_plugin = true
}

resource "vngcloud_vks_cluster_node_group" "primary" {
  cluster_id= vngcloud_vks_cluster.primary.id
  name= "my-nodegroup"
  num_nodes = 3
  auto_scale_config {
    min_size = 0
    max_size = 5
  }
  upgrade_config {
    strategy = "SURGE"
    max_surge = 1
    max_unavailable = 0
  }
  image_id = "img-108b3a77-ab58-4000-9b3e-190d0b4b07fc"
  flavor_id = "flav-9e88cfb4-ec31-4ad4-8ba5-243459f6dc4b"
  disk_size = 20
  disk_type = "vtvpe-61c3fc5b-f4e9-45b4-8957-8aa7b6029018"
}

```

```
enable_private_nodes = true
ssh_key_id= "ssh-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx"
labels = {
  "mylabel" = "vngcloud"
}
taint {
  key      = "mykey"
  value    = "myvalue"
  effect   = "PreferNoSchedule"
}
}
```

Attention:

- To get the image_id you want to use, you can access VKS Portal, select System Image menu and get the ID you want or get this information [here](#) .
- To get the flavor_id you want to use for your Node group, please get the ID [here](#) .

Launch Terraform command

- After completing the above information, run the command below:

```
terraform init
```

- Then, to see the changes that will be applied to the resources that terraform is managing, you can run:

```
terraform plan
```

- Finally, you choose to run the command line:

```
terraform apply
```

- Select **YES** to initiate Cluster and Node Group via Terraform

Check the newly created Cluster on the VNG Cloud Portal interface

After successfully initializing Terraform, you can go to VKS Portal to view the newly created Cluster information.

See more about how to use Terraform to work with VKS [here](#).

Monitoring

•

Metrics

You can install vMonitor Platform Metric Agent into your Kubernetes Cluster to collect and push metrics to the vMonitor Platform site, then use the features at vMonitor Platform to centrally manage resources and monitor unusual activity of your Kubernetes Cluster. .

Install Metric Agent using Helm

Preparation steps before installation

1. Check that you have the Metric Quota and that your quota has not reached the limit. If you do not have it, you need to buy the Metric Quota [here](#) .

2. Create a Service Account and attach policy: vMonitorMetricPush to have enough rights to push Metric to vMonitor

To create a service account, go [here](#) , then perform the following steps:

- Select " **Create a Service Account** ", enter a name for the Service Account and click **Next Step** to assign permissions to the Service Account
- Find and select **Policy: vMonitorMetricPush**, then click " **Create a Service Account** " to create a Service Account, Policy: vMonitorMetricPush created by VNG Cloud only contains the correct permission to push metrics to the system.
- After successful creation, you need to save **the Client_ID** and **Secret_Key** to perform the next step.

Install helm on Debian/Ubuntu server

1. You need to install Helm on a server **with kubeconfig containing enough permissions** to interact with the Kubernetes Cluster.

- Check permissions with **kubectl command :**

Kube checks permission

```
# Lệnh dùng để kiểm tra quyền tương tác tất cả resource tại namespace def
kubectl auth can-i '*' '*'
# Lệnh dùng để kiểm tra quyền tương tác tất cả resource tại namespace chỉ
kubectl auth can-i -n <namespace_chỉ định> '*' '*'
# Lệnh dùng để kiểm tra quyền tạo clusterrole và clusterrolebinding
kubectl auth can-i create clusterrole
kubectl auth can-i create clusterrolebinding
```

- If the result of the above commands is **YES**, then you have enough rights.

2. Proceed to install Helm

- Execute the following commands:

```
curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /us
sudo apt-get install apt-transport-https --yes
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyring
sudo apt-get update
sudo apt-get install helm
```

- Check that Helm has been installed successfully

```
helm version
# Kết quả mong muốn của command
# version.BuildInfo{Version:"v3.11.0", GitCommit:"472c5736ab01133de504a82
```

Install Helm in other operating systems

Refer to the installation instructions here: [Install Helm Through Package Managers](#)

3. Some additional notes about Helm: (see details at: [helm docs](#))

- Helm will use kubeconfig to interact with the cluster, by default helm will use config in the path: "~/.kube/config"
- If we need to change the path to kubeconfig we can use 2 ways:
 - For each helm command, add the --kubeconfig option: for example, helm install --kubeconfig <path_to_kubeconfig>
 - Declare environment variable: KUBECONFIG

Install Metric Agent

By default, when installing vMonitor Platform Metric Agent, there will be 2 components:

- **Deployment** Agent kube-state-metrics: collect metrics for each resource of k8s cluster (pod, daemonset, deployment, replicaset,)
- **Daemonset** Agent: collect metrics for each k8s cluster node (CPU, Memory usage, ...)

1. Add Helm vMonitor Platform Repo

```
helm repo add vmonitor-platform https://vngcloud.github.io/helm-charts-vm  
helm repo update
```

2. Install charts

- Check and delete related resources before installation to avoid conflicts

```
# Get Clusterrole vmonitor metric agent  
kubectl get clusterrole | grep vmonitor-metric-agent  
# Thực hiện xóa resource nếu có tồn tại  
kubectl delete clusterrole vmonitor-metric-agent
```

- Install at **default namespace** (add flag **-n <namespace_specified>** to install agent at another namespace)

```
helm install vmonitor-metric-agent vmonitor-platform/vmonitor-metric-agen  
--set vmonitor.iamClientID=<YOUR_CLIENT_ID_XXXXXXXXXXXXXXXXXXXX> \  
--set vmonitor.iamClientSecret=<YOUR_CLIENT_SECRET_XXXXXXXXXXXXXXXXXXXX> \  
--set vmonitor.clusterName=<CLUSTER_NAME>
```

- <YOUR_CLIENT_ID_XXXXXXXXXXXXXXXXXXXX>, <YOUR_CLIENT_SECRET_XXXXXXXXXXXXXXXXXXXX>: Service account information created in the preparation step
- <CLUSTER_NAME>: This information will be used to filter hosts of k8s cluster in case there are many clusters to monitor
- Check that the agent installation was successful

```
# Chạy command và đảm bảo output là các pods ở trạng thái running  
kubectl get pod | grep "vmonitor-metric-agent"
```

- If the pods agent is not in running state, use the corresponding command to check for errors

```
# Chạy command nếu pod ở trạng thái Pending  
kubectl describe pod <vmonitor-metric-agent-node-name>  
# Chạy command nếu pod ở trạng thái CrashLoopBackOff and Error  
kubectl logs <vmonitor-metric-agent-node-name>  
# Sau đó kiểm tra logs xuất hiện tại agent
```

After the installation is complete, kubernetes tracking metrics have been pushed to the vMonitor Platform site, you can proceed to vMonitor to draw dashboards and widgets.

Uninstall Metric Agent

Execute the following command to delete the installed related k8s resources:

```
helm uninstall vmonitor-metric-agent
```

Metric Agent installation does not use kube-state-metrics

- Install at **default namespace** (add flag **-n <namespace_specified>** to install agent at another namespace)

```
helm install vmonitor-metric-agent vmonitor-platform/vmonitor-metric-agen  
--set vmonitor.iamClientID=<YOUR_CLIENT_IDXXXXXXXXXXXXXXXXXX> \  
--set vmonitor.iamClientSecret=<YOUR_CLIENT_SECRETXXXXXXXXXXXXXX> \  
--set vmonitor.clusterName=<CLUSTER_NAME> \  
--set vmonitor.kubeStateMetricsEnabled=false \  
--set kubeStateMetricsAgent.enabled=false
```

Charging Fee

•

For the Procuracy, **the Managed Control Plane cost is completely free**. You only need to pay for other resources that you actually use, including:

- All nodes present in the Cluster (VM). Details on how to calculate the price of vServer can be found [here](#).
- Load Balancer is integrated into your Cluster. Details on how to calculate the price of Load Balancer can be found [here](#).
- Persistent Volume, Snapshot integrated into your Cluster. Details on how to calculate Volume prices can be found [here](#).



Attention:

- To ensure your Cluster operates stably, we have automatically set up Auto-renew for all resources on your Cluster. Before the expiration date of the resources, make sure your credit balance is enough for the system to auto-renew successfully.

Reference

•

Kubernetes versions

•

Currently, the VKS system is providing you with 3 Kubernetes versions including:

Version	Timeline	Attachment
Vesion 1.29.1	2025-02-28	https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.29.md#v1291
Version 1.28.8	2024-10-28	https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.28.md#v1288
Version 1.27.12	2024-06-28	https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.27.md#v12712

Node Flavors

•

Below is a list of Flavors currently supported by VKS:

- General Code A

Flavor Name	CPU	Memory	Flavor ID
a-general-2×4	2	4	flav-305a67bf-1825-4e3f-bc72-d41afa160d93
a-general-4×8	4	8	flav-9e88cfb4-ec31-4ad4-8ba5-243459f6dc4b
a-general-8×16	8	16	flav-2fec3902-4b71-489c-a294-19bad1df3a70
a-general-12×24	12	24	flav-ea60643c-36f0-4fd2-be8d-e42198d6320e
a-general-12×24-n10	12	24	flav-0db2d13d-530b-4312-b2be-cdb3b80c73e4
a-general-8×16-n10	8	16	flav-a04a8e9c-8def-4fde-9bfc-2ba738576463
a-general-16×32-n10	16	32	flav-65701e64-471f-4747-a837-500651b4c67d
a1-standard-2×8	2	8	flav-edd3a4bd-ce5a-4b72-9323-468d58615b68
a1-standard-4×16	4	16	flav-4cb926dc-63be-4edc-8a21-4a66d963b45b

a1-standard-8×32	8	32	flav-b834f30b-8dad-4d7e-8b71-952824bfef23
a1-standard-16×64	16	64	flav-7f6e15db-191c-4c89-80da-70c74ab5f786
a1-standard-32×128	32	128	flav-67617e54-48a5-4aed-9213-4560dc0cd9c1
a1-standard-48×192	48	192	flav-778bfd0f-ad5a-4c83-b44a-7a1f2f855240
a1-standard-8×32-n10	8	32	flav-f31884f4-e8b9-4f9a-9de5-41307afa5960
a1-standard-16×64-n10	16	64	flav-c398e0a9-a153-4826-8bd5-4d715d3e5032
a1-standard-32×128-n10	32	128	flav-2ac60541-046e-4761-b40c-990012597e09
a1-standard-48×192-n10	48	192	flav-bde154c4-52c9-4e92-9495-7fd7af3080c8
a1-highmem-2×16	2	16	flav-7615377f-909c-4e1d-9512-bc3b639a3777
a1-highmem-4×32	4	32	flav-dc274c6a-ef9a-4aff-9966-adecfac3731f
a1-highmem-8×64	8	64	flav-3e3eb1c2-9f38-47f3-b2a1-878ce28c160b
a1-highmem-16×128	16	128	flav-618a1e2c-c3eb-48e8-af13-17661ce8e37c

a1-highmem-32×256	32	256	flav-9f9421c3-8956-43fb-a9c5-2563d5316fb9
a1-highmem-8×64-n10	8	64	flav-f51b4591-7e2d-4a2f-93ce-7a0b2a4827db
a1-highmem-16×128-n10	16	128	flav-f9e6b4a4-a802-4f86-9adb-fcb746fe62e6
a1-highmem-32×256-n10	32	256	flav-3f52df69-638b-4363-836c-4b8ff9622080
a1-highcpu-4×4	4	4	flav-3bb2088e-5c17-4780-93c8-edea2b5bc1d6
a1-highcpu-8×8	8	8	flav-4c4cb707-3e85-4a17-8891-485985b10c0a
a1-highcpu-16×16	16	16	flav-67489c98-a620-466a-9449-5f08dfdeb3
a1-highcpu-32×32-n10	32	32	flav-5f84e226-833b-4ab8-a797-7653dae9a764
a1-highcpu-16×16-n10	16	16	flav-9ef49395-9628-4a50-94fd-56c3aeeef5065
a1-highcpu-32×64-n10	32	64	flav-bf77e8d1-8ec4-42e1-aca4-bb4a703b67d8

- General Code S

		•
eci.ins.s-general-4×8	4	
eci.ins.s-general-8×16	8	
eci.ins.s-general-16×32	16	
s-general-2×4	2	
s-general-4×8	4	

s-general-8×16	8
s-general-16×32	16
s-general-16×32-n10	16
s-general-8×16-n10	8
s1-standard-2×8	2
s1-standard-4×16	4

		•
s1-standard-8×32	8	
s1-standard-12×48	12	
s1-standard-16×64	16	
s1-standard-32×128	32	
s1-standard-64×256-n10	64	
s1-standard-48×192-n10	48	

		•
s1-standard-32×128-n10	32	
s1-standard-16×64-n10	16	
s1-standard-8×32-n10	8	
s1-highmem-2×16	2	
s1-highmem-4×32	4	
s1-highmem-8×64	8	

		•
s1-highmem-16×128	16	
s1-highmem-32×256	32	
s1-highmem-32×256-n10	32	
s1-highmem-16×128-n10	16	
s1-highmem-8×64-n10	8	
s1-highcpu-4×4	4	

		•
s1-highcpu-8×8	8	
s1-highcpu-32×32	32	
s1-highcpu-32×64-n10	32	
s1-highcpu-16×16-n10	16	
s1-highcpu-8×8-n10	8	

eci.ins.s1-standard-2×8	● 2
eci.ins.s1-standard-4×16	4
eci.ins.s1-standard-8×32	8
eci.ins.s1-standard-16×64	16
eci.ins.s1-standard-32×128	32
s1-highcpu-16×16	16

		•
s1-highcpux2-32×64	32	
eci.ins.s1-highmem-2×16	2	
eci.ins.s1-highmem-4×32	4	
eci.ins.s1-highmem-8×64	8	
eci.ins.s1-highmem-16×128	16	
eci.ins.s1-highmem-32×256	32	

eci.ins.s1-highcpu-4×4	4
eci.ins.s1-highcpu-8×8	8
eci.ins.s1-highcpu-16×16	16

- GPU Code G

Flavor Name	CPU	Memory
g1-standard-4×16-1rtx2080ti	4	16
g1-standard-8×32-1rtx2080ti	8	32
g1-standard-8×32-2rtx2080ti	8	32
g1-standard-16×64-2rtx2080ti	16	64
g1-standard-16×64-4rtx2080ti	16	64
g1-standard-32×128-8rtx2080ti	32	128

g1-standard-8×64-1rtx2080ti	8	●	64
g1-standard-8×64-2rtx2080ti	8		64

- GPU Code RTX4090

g2-standard-32×128-4rtx4090	32	128	4	flav- 79d5ef 91bd- 442b- 9919- f7d7dd! 761a
g2-standard-16×128-2rtx4090	16	128	2	flav- a9bfd4 -0b9f- 481a- a6ce- 8c924c 08ec3
g2-standard-16×64-2rtx4090	16	64	2	flav- fc2c0e0 -8ecb- 4039- aa98- 11bccc9 2d35
g2-standard-32×64-2rtx4090	32	64	2	flav- 6b11c1c 2631- 4ce8- 8184- 1f94859 18cb
g2-standard-16×64-1rtx4090	16	64	1	flav- 2a2286 -503d- 4e0e- 8cdf-

b99447
865d3

System Image

•

Below is a list of System Images that the VKS system currently supports:

Kubernetes version	Image name	Image ID
v1.27.12	1_Ubuntu-22.kube_v1-27-12	img-36ee0a61-863d-4d40-a768-9b41269b8a62
v1.28.8	1_Ubuntu-22.kube_v1-28-8	img-983d55cf-9b5b-44cf-aa72-23f3b25d43ce
v1.29.1	1_Ubuntu-22.kube_v1-29-1	img-108b3a77-ab58-4000-9b3e-190d0b4b07fc