viettel
IDC

USER GUIDE

VIETTEL OPEN KUBERNETES SERVICE

FOR CUSTOMERS

TABLE OF CONTENTS

1. Service overview 6

1.1 Kubernetes & the Architecture of a Kubernetes Resource Cluster 6

1.1.1 Kubernetes & the Architecture of a Kubernetes Resource Cluster 6

1.1.3 Node (Worker Node) 8

1.1.4 Kubectl 8

1.1.5 Pod 8

1.1.6 Image 9

1.1.7 Deployment 9

1.1.8 ReplicaSet 9

1.1.9 Service 9

1.1.10 Labels 10

1.1.11 Taints 10

1.1.12 Worker Autoscale 11

1.2 Introduction to Viettel Open Kubernetes Service (VOKS) 12

1.2.1 What is VOKS? 12

1.2.2 Feature of VOKS 12

1.2.3 Payment Options 14

2. Cluster 15

2.1 What is Cluster? 15

2.2 Cluster in vOKS Service 15

2.3 Cluster Initialization 17

2.3.1 Initializing a Trial Cluster 17

2.3.2 Initializing a Saving Plan Cluster 19

2.4 Cluster Management 24

2.5 Upgrading Kubernetes Cluster Version 26

2.6 Best Practices for Upgrading Kubernetes Cluster Versions 27

2.7 Deleting a Cluster.....28

3. Access Management 29

3.1 Using the Identity Access Management (IAM) Feature 30

3.2 Service vOKS Role-Based Policies.....30

4. Resource Management 31

4.1 Creating a Node Group 31

4.1.1 Adding a Node Group to a Saving Plan Cluster 31

4.1.2 Adding a Node Group & Nodes for PAYG Service 33

4.2 Viewing Node Group Information & Nodes Inside 37

4.3 Editing Node Group Information 38

4.3.1 Editing the Node Group Name.....38

4.3.2 Editing Worker Autoscale Configuration for a Node Group 39

4.3.3 Editing Auto-Repair Configuration for a Node Group 41

4.4 Delete Node group.....42

5. Application Data Storage..... 43

5.1 Introduction to Volume & NFS Node 43

5.2 Types of Storage Connections on NFS Nodes 45

5.3 NFS Node Architecture.....45

5.4 Adding NFS Node Capacity 46

5.4.1 Adding NFS Node Capacity for Saving Plan Clusters 46

5.4.2 Adding NFS Node Capacity for PAYG Clusters.....48

5.5 Instructions for Configuring NFS Node Usage 48

6. Network Configuration 50

6.1 Private Connection to Other Services within the Same VPC Network.....50

6.2 Connecting to Viettel Open Virtual Private Cloud (vOVPC).....50

6.3 Connecting to Viettel Database Service (vDBS) 53

6.4 CNI.....55

6.5. Configure Viettel IDC's DNS 56

7. Deploying Applications on the Cluster 58

7.1 Setting Up the User-Side Control Environment 58

7.2 Instructions for Downloading the Kube-config File & Using the Kubernetes Dashboard.....61

7.2.1 Downloading the Kube-config File.....61

7.2.2 Using the Kubernetes Dashboard.....62

7.3 Tính năng Horizontal Pod Autoscaling 64

7.3.1 Overview.....64

7.3.2 Mechanism.....65

7.3.3 Instructions for Configuring HPA Metrics.....66

8. Managing Usage Costs 71

8.1 Checking the Uptime of Worker Nodes.....71

8.2 Viewing Billing History and Payment History.....72

8.3 Viewing and Downloading Cluster Action History 74

9. Monitor the Cluster 75

9.1 Using Prometheus, Grafana 75

9.1.1 Prometheus 76

9.1.2 Grafana.....77

9.2 Using Viettel Cloud Watch (coming soon).....78

10. Frequently Asked Questions (FAQ)..... 78

10.1 What is the basic differences between Container, Docker and Kubernetes? 78

10.2 What kind of service vOKS provide to customers? 78

10.3 What is the difference between Node Storage and Persistent Storage? 79

10.4 Is vOKS compatible with other containerized applications running on Kubernetes platform of other Cloud providers? 79

1. Service overview

1.1 Kubernetes & the Architecture of a Kubernetes Resource Cluster

1.1.1 Kubernetes & the Architecture of a Kubernetes Resource Cluster

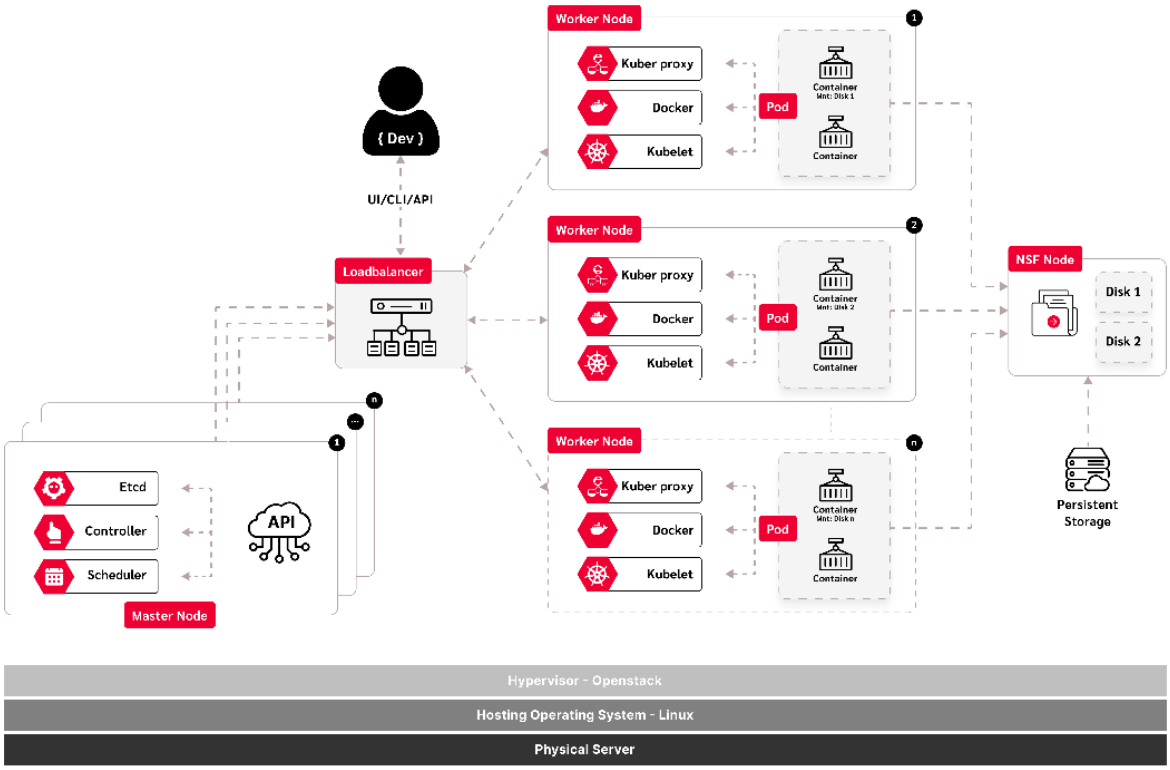
Kubernetes is an open-source platform with rapid adaptability and scalability for managing containerized applications and services. It facilitates the configuration and automation of application deployments.

The fundamental components of Kubernetes include:

- **Control Plane Components (Master Node):** These components are responsible for managing the overall state of the cluster.
- **Node Components (Worker Node):** These components run on every worker node, maintaining the running pods and providing a runtime environment for Kubernetes.
- **Addons:** Addons utilize Kubernetes resources (such as DaemonSet, Deployment, etc.) to extend Kubernetes functionality and deploy cluster features. Examples include DNS, Web UI (Dashboard), Container Resource Monitoring, and Cluster-level Logging.

The basic architecture of a Kubernetes resource cluster (Kubernetes Cluster) provided by Viettel IDC is as follows:

1.1.2 Control Plane (Master Node)



Control Plane (Master Node) is the server that controls Worker Nodes running Pods that host applications. The Master Node comprises four main components:

- Kube API Server: This component facilitates communication among other components. Developers interact with this API Server when deploying applications.
- Kube Scheduler: This component schedules deployments, deciding which Worker Node an application will be placed on to run.
- Kube Controler Manager: This component manages Worker Nodes, monitors their statuses (e.g., whether they are "alive" or "dead" via the Node Controller), handles application replication, and more.
- Etcd: This is Kubernetes' database, where all information about Kubernetes is persistently stored.

In addition to the four main components, the control plane can optionally integrate with cloud provider platforms through the cloud-controller-manager

1.1.3 Node (Worker Node)

Node (Worker Node) is a server that runs Pods containing applications. It consists of three main components:

- Container runtime: This component runs applications in the form of containers. Kubernetes supports various container runtimes, such as Docker, containerd, and CRI-O.
- Kubelet: This component communicates with the Kubernetes API Server and plays a role in managing containers.
- Kubernetes Service Proxy (tùy chọn): This component handles load balancing between applications.

Nodes are grouped into collections with identical configurations, referred to as Node Groups. Node Groups can be scaled up or down flexibly and divided into multiple groups within a Kubernetes resource cluster to meet application deployment needs. Users can configure Worker Autoscale by Node Group to optimize resource usage and application performance.

1.1.4 Kubectl

Kubectl is an agent running on each worker node within the Kubernetes cluster. Kubelet operates based on PodSpec. Kubelet retrieves a set of PodSpecs provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. Kubelet does not manage containers that were not created by Kubernetes.

1.1.5 Pod

A Pod is the smallest deployable computing unit that users can create and manage in Kubernetes. A Pod consists of one or more containers, sharing network resources and storage, along with a specification for how the containers should run. The contents of a Pod

are always co-located, scheduled together, and run within the same execution context. A Pod models a "logical host" dedicated to an application: it contains one or more application containers that are tightly coupled. In non-cloud environments, applications running on the same physical or virtual machine mirror how cloud applications run on the same logical host. In addition to application containers, a Pod can include init containers that run during the Pod's startup process.

1.1.6 Image

An Image represents the binary data encapsulating an application and all its dependencies. Images can run independently and make clearly defined assumptions about their runtime environment. Images are typically stored in centralized repositories, such as Docker Hub, which hosts official images for many popular applications like nginx, mysql, wordpress, redis, etc.

1.1.7 Deployment

A Deployment provides declarative updates for Pods and ReplicaSets. Users define the desired state in a Deployment, and the Deployment Controller adjusts the actual state to match the desired state at a controlled pace. Additionally, users can configure Deployments to create new ReplicaSets, delete existing Deployments, and reuse all their resources with new Deployments.

1.1.8 ReplicaSet

A ReplicaSet is designed to maintain a stable set of Pod replicas running at any given time. ReplicaSets are often used to ensure the availability of a group of Pods, allowing for easy scaling up or down as needed.

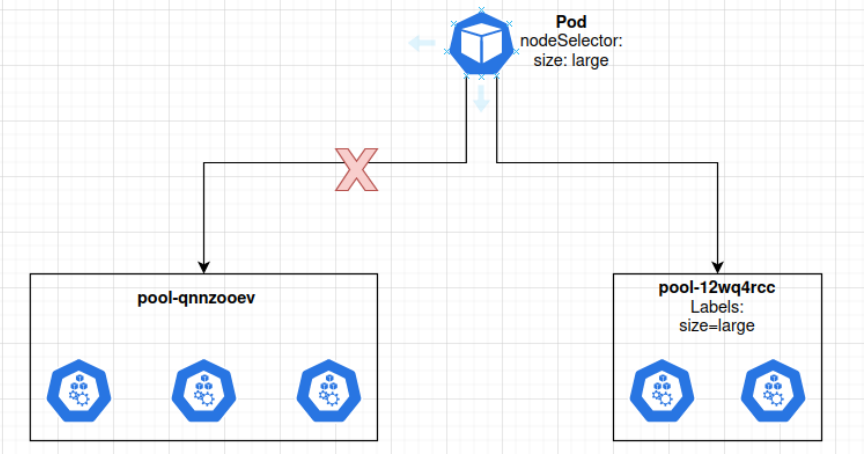
1.1.9 Service

Service is an abstraction for exposing an application running on a set of Pods as a network service. With Kubernetes, users do not need to modify their applications to use unfamiliar service discovery mechanisms. Kubernetes provides each Pod with its own IP

address and a unique DNS name for a group of Pods, and it can load balance traffic across these groups.

1.1.10 Labels

Labels are key/value pairs attached to objects, such as Pods. Labels are intended to identify meaningful and user-relevant attributes of objects. They can be used to organize and select subsets of objects (e.g., labeling objects by functionality such as frontend, backend, or by environments like dev, test, staging, production). Labels can be attached to objects at creation time and can be added or modified later. Each object can have a set of defined key/value labels, with each key being unique for a given object.



To use labels, the nodeSelector parameter is applied. The idea behind nodeSelector is to configure Pods to run only on nodes with specific labels.

For example, as shown in the diagram, Pods with a nodeSelector using the key-value pair **size:large** will be scheduled to a node pool labeled as **size=large**

1.1.11 Taints

Taint is a special type of label used to mark a node with specific characteristics, defined by a key-value pair and an action if a Pod does not meet the key-value condition.

Assigning a Taint to a node is like locking the node with a "lock" (consider the Taint as the lock). The lock is characterized by a label key=value and an associated action (called an **effect**) if no "key." The possible actions are:

- **NoSchedule:** Pods without the "key" will not be scheduled on this node.
- **NoExecute:** Pods without the "key" may be scheduled on this node but will not execute.
- **PreferNoSchedule:** Kubernetes will try to avoid scheduling Pods without tolerations on nodes with Taints.

Where there is a "lock" (Taint), there must also be a "key" (Toleration). To "unlock" (allow Pods to run on nodes with Taints), the Pod must declare **Tolerations**, which match the label key=value and action associated with the Taint.

Note: A node can have multiple Taints. In such cases, a Pod can be scheduled onto the node only if it has all the Tolerations corresponding to the node's Taints.

1.1.12 Worker Autoscale

Worker Autoscale is a feature that automatically scales Worker Node resources within a Node Group based on Kubernetes' core Cluster Autoscaler mechanism, with the following conditions:

- There are Pods in the cluster that cannot run due to insufficient resources, based on Horizontal Pod Autoscaling (HPA) with the Pods on the Nodes.
- There are Nodes in the cluster that have been underutilized for a long time, and their Pods can be placed on other existing Nodes.

For example, if users configure resource thresholds such as CPU usage exceeding 70% or memory usage exceeding 80% in a YAML file, the system will automatically create new Pods when the condition is met. If the newly created Pods lack sufficient resources to run

on the current Nodes, a new Worker Node will be created based on the specified Node Group configuration labeled as per the user's requirements.

If the Node Group is not labeled or shares the same label, Worker Nodes will be randomly added to one of the existing configurations. Users can monitor additional Worker Nodes on the management interface developed by Viettel IDC.

1.2 Introduction to Viettel Open Kubernetes Service (VOKS)

1.2.1 What is VOKS?

Viettel Open Kubernetes Service (vOKS) is a platform-as-a-service (PaaS) offering that provides software developers with a complete OS-level virtualized environment. It supports the development, testing, deployment, and distribution of applications on the Kubernetes platform.

vOKS is built on Viettel IDC's robust and stable OpenStack Public Cloud platform, enabling fully automated, fast, and convenient service provisioning for customers. vOKS complies with numerous domestic (G-Cloud) and international cloud service standards and certifications, including ISO 9001:2015 (Quality Management Systems), ISO 27001:2013 (Information Security Management Systems), and ISO 27017:2015 (Cloud Security Controls).

1.2.2 Feature of VOKS

VOKS delivers all the core features of the Kubernetes platform to customers, including:

- Service Discovery and Load Balancing: Kubernetes can expose a container using its DNS or unique IP address. If traffic to a container is high, Kubernetes can balance the load and distribute network traffic to ensure stable deployments.
- Storage Orchestration: Kubernetes enables automatic mounting of storage systems selected by developers, such as local storage or public cloud providers.

- Automated Rollouts and Rollbacks: Developers can describe the desired state for deployed containers, and Kubernetes will automatically adjust the actual state to match the desired state at a controlled rate. For instance, Kubernetes can automate creating new containers, deleting existing ones, and reallocating their resources.
- Automatic Packaging**:** Developers provide Kubernetes with a cluster of nodes to execute containerized tasks, specifying the required CPU and RAM for each container. Kubernetes schedules containers across nodes to maximize resource utilization.
- Self-Healing: Kubernetes restarts failed containers, replaces containers, deletes unresponsive containers, and ensures they are excluded from clients until they are ready.
- Configuration and Security Management: Kubernetes securely stores and manages sensitive information like passwords, OAuth tokens, and SSH keys. Developers can deploy and update application secrets and configurations without rebuilding container images or exposing secrets in stack configurations.

In addition to Kubernetes' native features, Viettel IDC enhances the platform with centralized management tools, providing users with more efficient cluster administration. Key features include:

Node Group Management: Developers can create, customize, and manage worker node groups for diverse infrastructure needs within the same Kubernetes cluster.

Centralized Node Administration: Manage Worker Nodes by labels, taints, and autoscaling configurations for Node Groups. Nodes can be added or removed quickly, and costs can be centrally monitored using Viettel IDC's self-service portal.

Version Upgrades: Support for the latest Kubernetes versions with continuous updates, ensuring improved performance, security, and access to new features.

Persistent Volume Storage with NFS Solution: Integration with an NFS Node for long-term data storage, enabling automatic configuration and data sharing across Nodes. Customers can add more storage as needed via the service management interface.

Pod Automation and Recovery: Kubernetes automatically creates new Pods, deletes old ones, and reallocates resources without downtime. It also restarts failed Pods and removes or replaces unresponsive ones.

Quick Installation of Monitoring Tools: Deploy monitoring tools such as Kubernetes Dashboard, Prometheus, and Grafana with a single click through Viettel IDC's service management interface.

Cluster Impact Control: Track all cluster changes using history logs, including Node Group creation, tool installation, and NFS Storage expansion.

Integration with Viettel CloudWatch (coming soon): Easily collect logs and monitor components like Pods, Volumes, and Nodes within Kubernetes using Viettel CloudWatch.

1.2.3 Payment Options

Users can choose between two payment methods:

- **Saving Plan:** This payment method allows users to register and prepay monthly, saving costs and fitting long-term, stable usage needs.
- **Pay As You Go (PAYG):** This method lets users use the service first and pay later. The costs for Worker Node resources and other components like Master Nodes, Persistent Volumes, IPs, and bandwidth are calculated hourly based on the creation and deletion time of the Worker Nodes and other components. Usage costs are consolidated and deducted daily from the customer's online account balance. The subscription package list is based on the number of resources used, leveraging the Worker Node Autoscale feature.

For detailed pricing for each payment method refer to the service information page.

2. Cluster

2.1 What is Cluster?

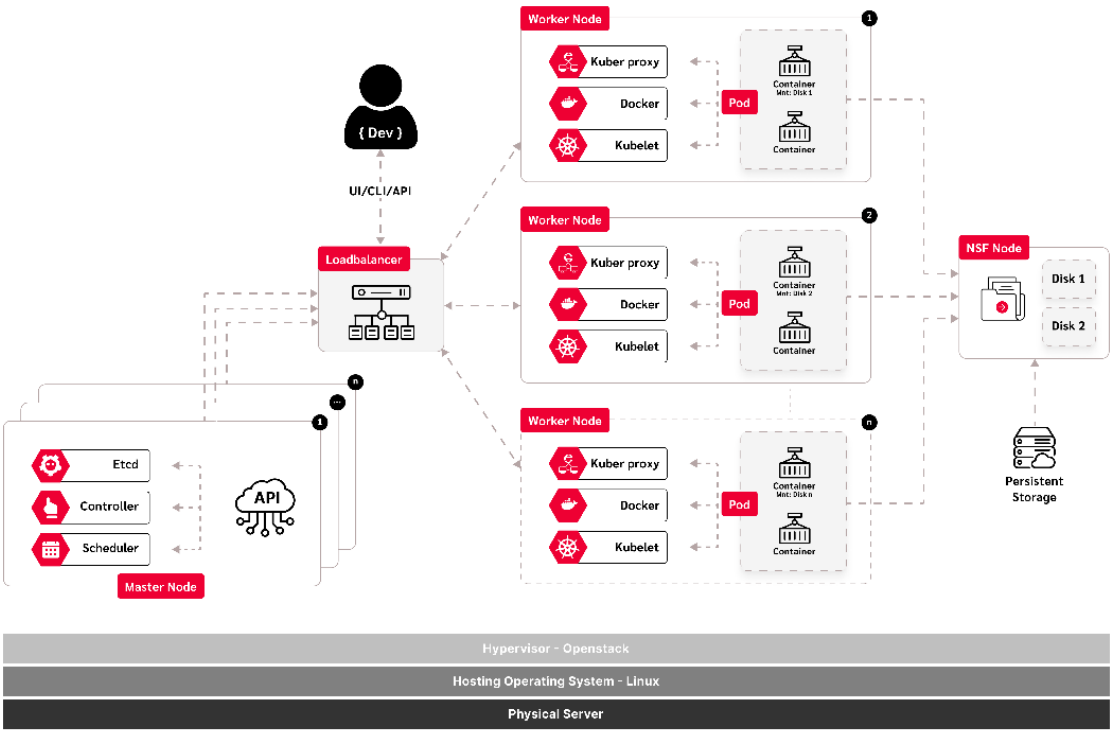
A Kubernetes Cluster (Cluster) comprises a Control plane (one or more Master Nodes); a collection of Worker Nodes (also called Nodes), running containerized applications. Each cluster requires at least one Worker Node to run Pod.

Worker Node(s) host the Pod, which are components of the application workload. Control plane manages the Worker Nodes and Pod within the cluster. In a production environment, typically runs on multiple machines, and the cluster often includes many Nodes to ensure fault tolerance and high availability.

2.2 Cluster in vOKS Service

The vOKS service provided by Viettel IDC includes a Kubernetes resource cluster comprising Master Nodes and Worker Node(s), with full support for features and components like Autoscale, service LB (Ingress Controller), version upgrades, Persistent Volumes, and more.

The basic architecture of a Kubernetes resource cluster offered by Viettel IDC is as follows:



Depending on the chosen payment method, a Kubernetes resource cluster can be initialized with varying component counts:

- With Saving Plan, a Kubernetes resource cluster includes:
    - 1 or 3 Master Nodes to ensure High Availability (HA) for the cluster
    - A minimum of 3 Worker Node in one Node group with user-selected configurations
    - NFS Storage capacity for Persistent Volume usage
    - 2 IPs for API Server and Kubernetes Dashboard.
    - Users can optionally purchase additional Node Groups with different configurations to meet specific service needs, as well as an Ingress Controller to display HTTP and HTTPS routes from outside the cluster to services within it.
- With PAYG, 01 a Kubernetes resource cluster includes:
    - 01 or 3 Master Nodes to ensure High Availability (HA) for the cluster.

- A minimum 01 Worker Node in 01 Node group with user-seleted configurations. Users can adjust the Worker Autoscale settings of Node Groups (min/max nodes) via Viettel IDC's management interface to meet node resource needs during application deployment.
- NFS Storage capacity for Persistent Volume usage
- 2 IPs for API Server and Kubernetes Dashboard.
- An Ingress Controller to display HTTP and HTTPS routes from outside the cluster to services within it.

2.3 Cluster Initialization

Users can select and initialize a cluster from the list of packages available on the service information page. To create a suitable cluster, users need to consider several options:

- Should they use a trial to experience the service or register for a paid plan upfront?
- Which payment method to choose: **Saving Plan** or **PAYG**?
- Which region to select for service usage: North or South?
- What configuration to choose for the application to be deployed on Kubernetes?
- Which Kubernetes version is appropriate for their purpose?

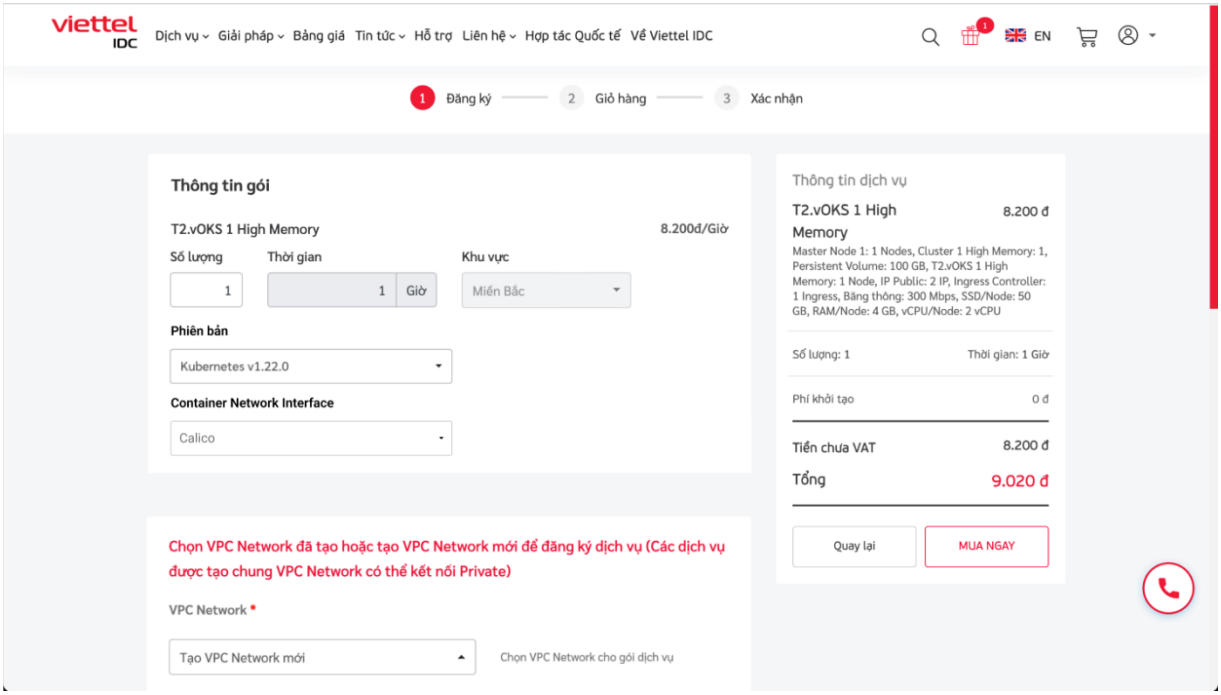
2.3.1 Initializing a Trial Cluster

Bước 1: On the Viettel Open Kubernetes Service (vOKS) information page, scroll down to the "Service Packages" section and click on **"Trial"** for the package you wish to try. The system will redirect to the **Trial Registration** screen.

Note: Users are allowed to trial **only one service package per month.** For special trial requests, please contact us for support.

Step 2: On the **Trial Registration** screen, users can:

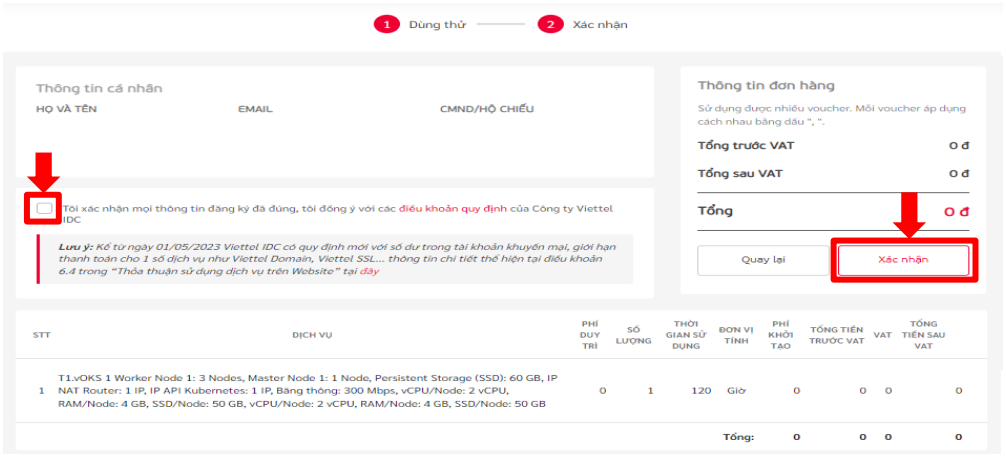
- Select the version of the Kubernetes resource cluster according to their needs.
- Choose to **Create a new VPC Network** or **Select an existing VPC Network** to connect with services like **Viettel Database Service (vDBS)** or **Viettel Open Virtual Private Cloud (vOVPC).**



Click "Buy Now" to proceed to the Order Confirmation step for the trial. If you wish to change the trial package, click "Back" to return to the previous step.

Step 3: On the **Confirmation** screen:

- Tick the checkbox to agree with the terms of service provided by Viettel IDC.
- Then, click **"Confirm"** to finalize the creation of the trial order.

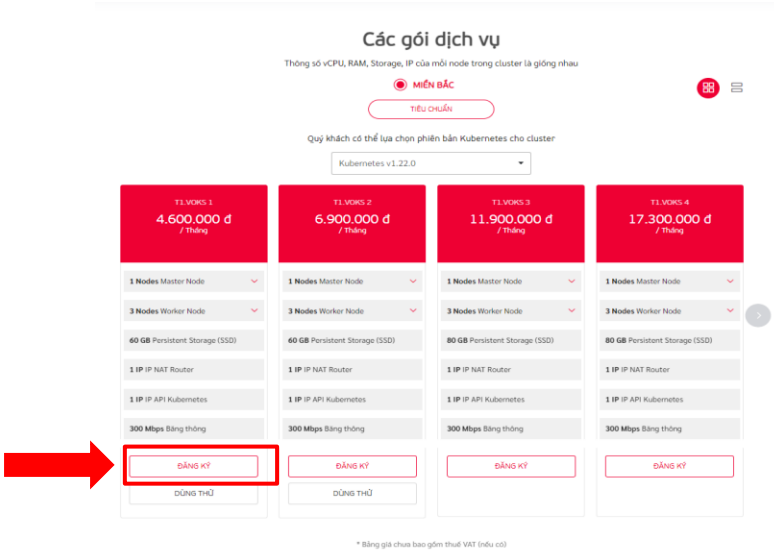


Bước 4: The system will display a notification indicating that the trial order has been successfully created. Users will receive an email containing details of the registered service package.

On the Viettel IDC homepage, users can click on the **Account** section in the top-right corner of the screen → click on **Service Management** to view the initialization status, detailed management information, and service usage.

2.3.2 Initializing a Saving Plan Cluster

Step 1: On the Viettel Open Kubernetes Service (vOKS) information page, scroll down to the Service Packages section, and click Register for the desired paid service package. The system will redirect to the Registration screen.



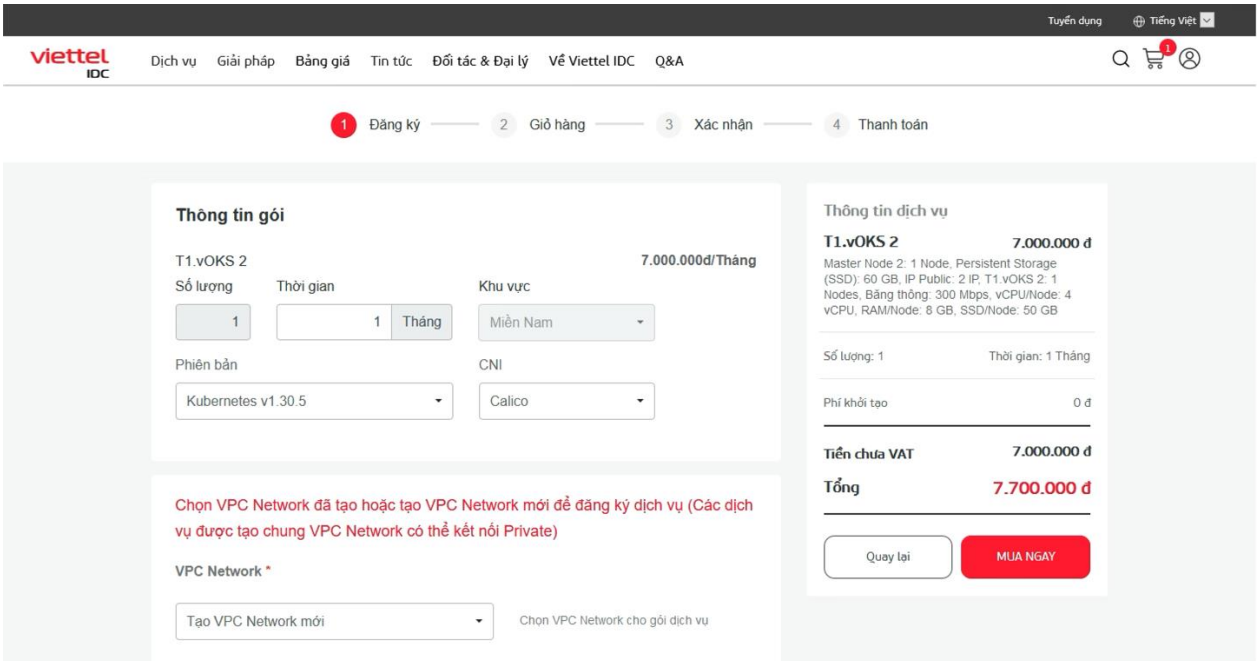
Step 2: On the **Registration** screen:

Users enter the **number of service packages**, **usage duration**, and **region**.

- Select the desired Kubernetes resource cluster version based on their needs.

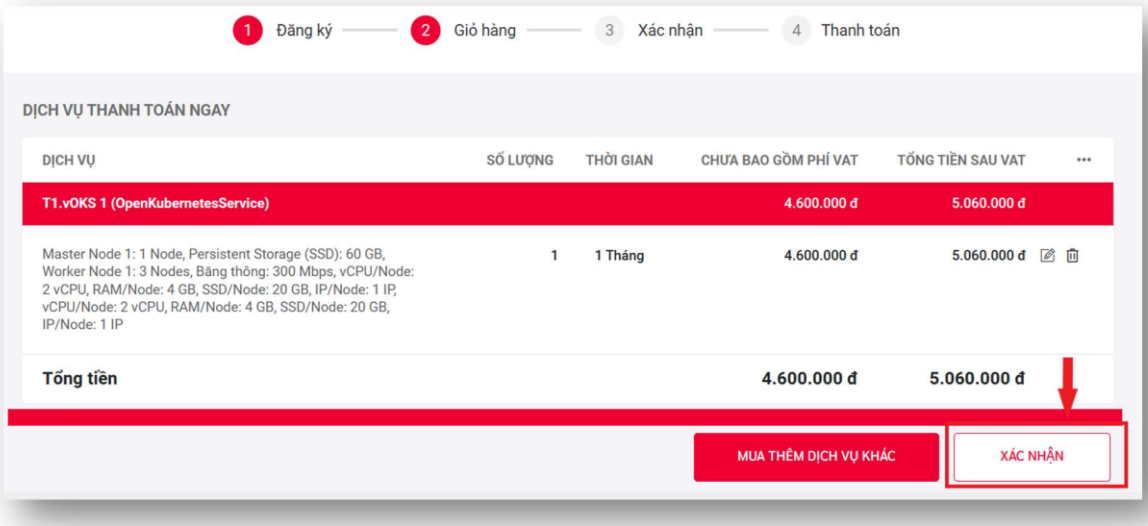
- Users can choose to **Create a new VPC Network** or **Select an existing VPC Network** to connect with services in the Viettel Cloud ecosystem (e.g., Viettel Database Service (vDBS), Viettel Open Virtual Private Cloud (vOVPC), etc.).
- The **Saving Plan** service package includes a minimum of **1 Base Node Group** with 3 Worker Nodes configured based on the package selected in Step 1. Users cannot remove this Node Group but can add additional Node Groups and Worker Nodes based on their usage needs.
- Users can optionally purchase additional add-ons:
    - **High Availability (HA):** Adds 2 additional Master Nodes and 1 Load Balancer to operate the cluster in a Multi-Master Node model, increasing service availability.

      Note: Once a Kubernetes Cluster is created in the Multi-Master Node model, it cannot be reverted to a Single Master Node model. The HA option cannot be added after the cluster has been created.
    - **Ingress Controller:** Users can purchase an Ingress Controller to configure HTTP Load Balancer routing for applications running on Kubernetes.
    - **Bandwidth:** Default bandwidth is 300 Mbps. Users can purchase additional bandwidth if needed



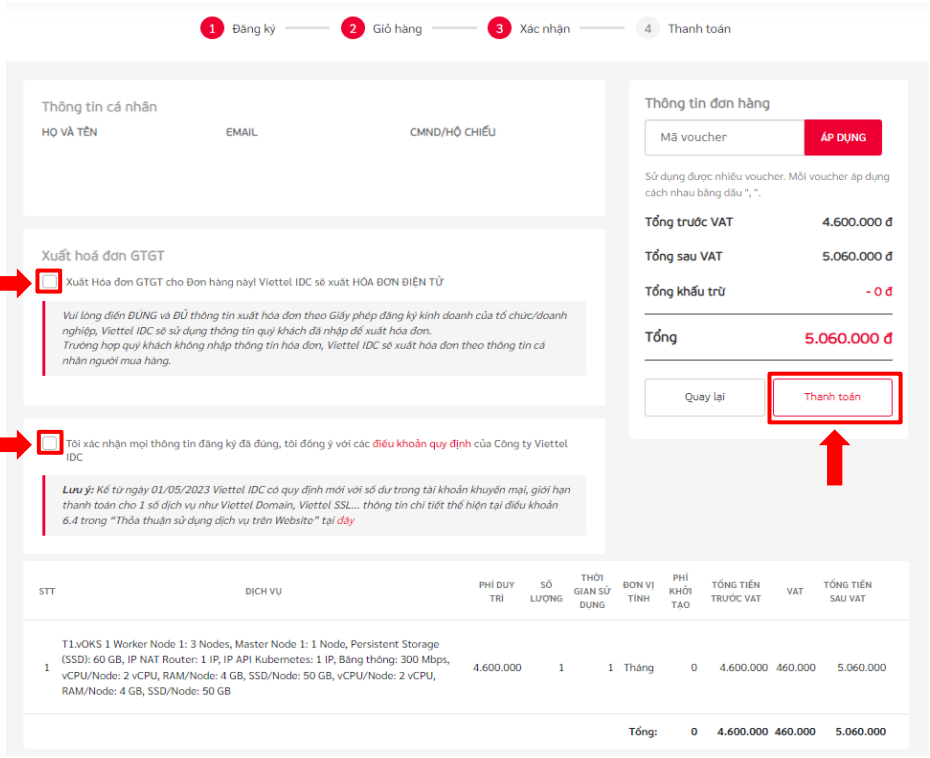
- Users click **"Buy Now"** to proceed to the **Cart** screen.

Step 3: On the **Cart** screen, users click **"Confirm"** to proceed to the **Order Confirmation** screen:



Step 4: On the Confirmation Screen:

- The user checks the "Issue VAT Invoice" box and fills in the relevant information to receive the invoice.
- The user checks the box to confirm agreement with Viettel IDC's terms of service.
- The user enters voucher information (promotion or discount codes) if applicable.
- The user clicks on "Pay" to proceed to the Payment screen for the order.



Step 5: On the Payment Screen:

- The user selects the appropriate payment method and completes the payment.

- The user clicks on "Confirm" to finalize the order for the paid subscription.

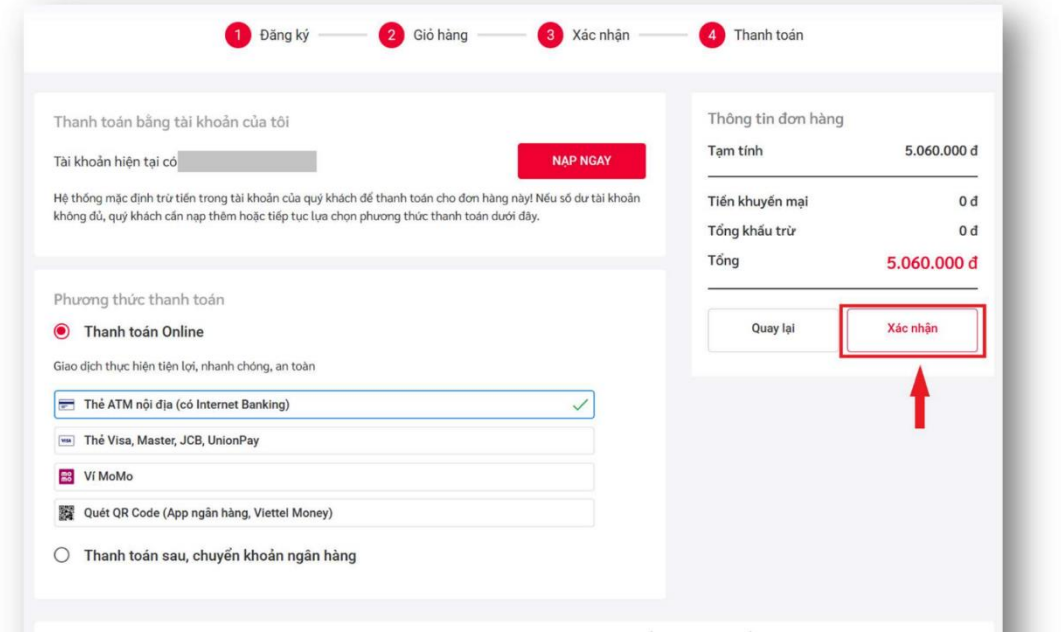
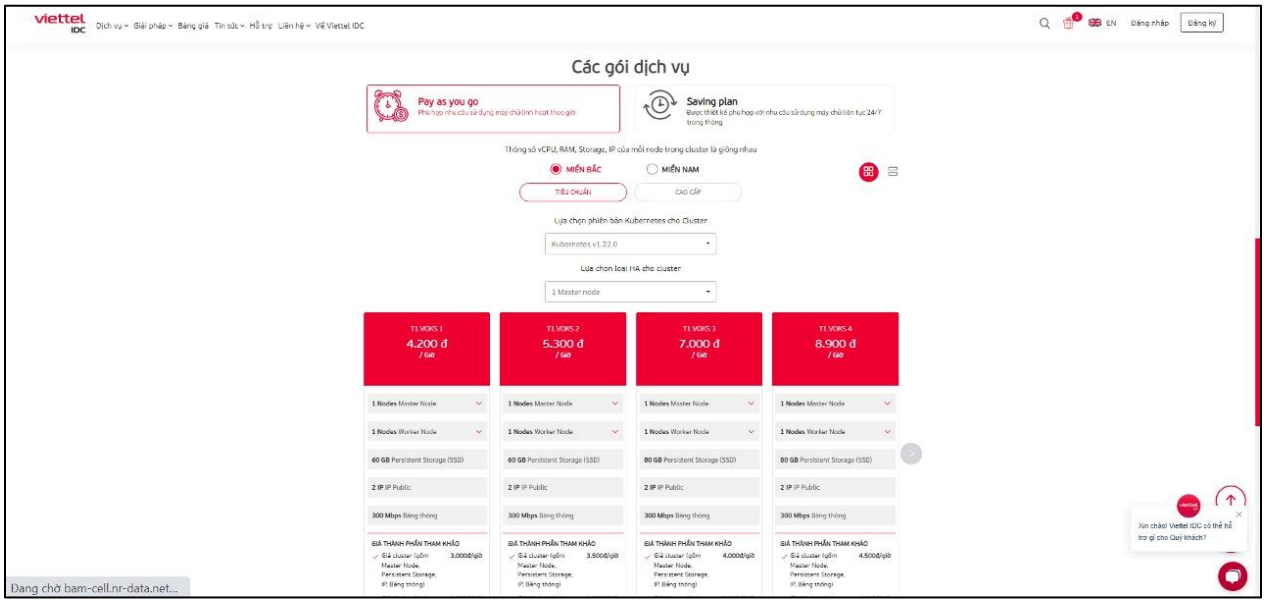


Image 3.3: Creating a Pay-as-you-go (PAYG) Cluster



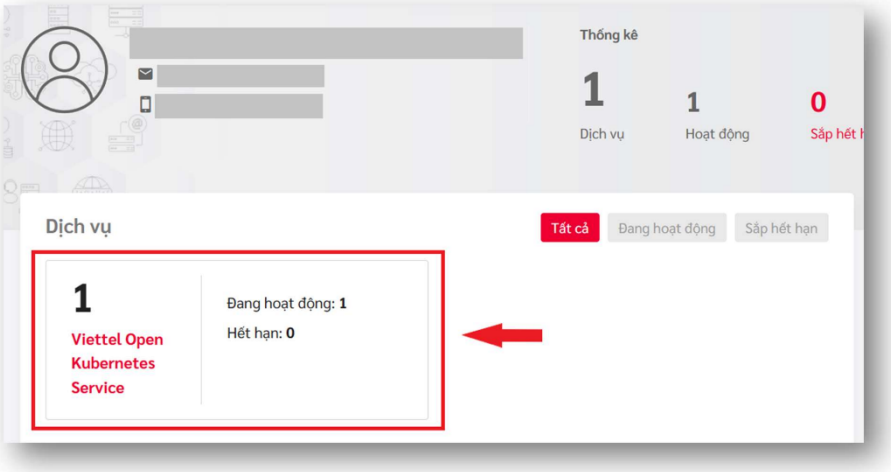
- **Pay-as-you-go (PAYG)** is a postpaid payment model where the costs of Worker Nodes and other components such as Master Nodes, Persistent Volumes, IPs, and

bandwidth are calculated hourly based on the creation and deletion times of these resources. The usage costs are consolidated and deducted daily from the customer's online account balance.

- Users register for the PAYG plan through a 4-step process similar to the Saving Plan cluster. Fees are calculated and deducted daily from the user's account on the website. Detailed instructions for checking the service fees for vOKS PAYG are provided in Section 3.2: Service Usage.

2.4 Cluster Management

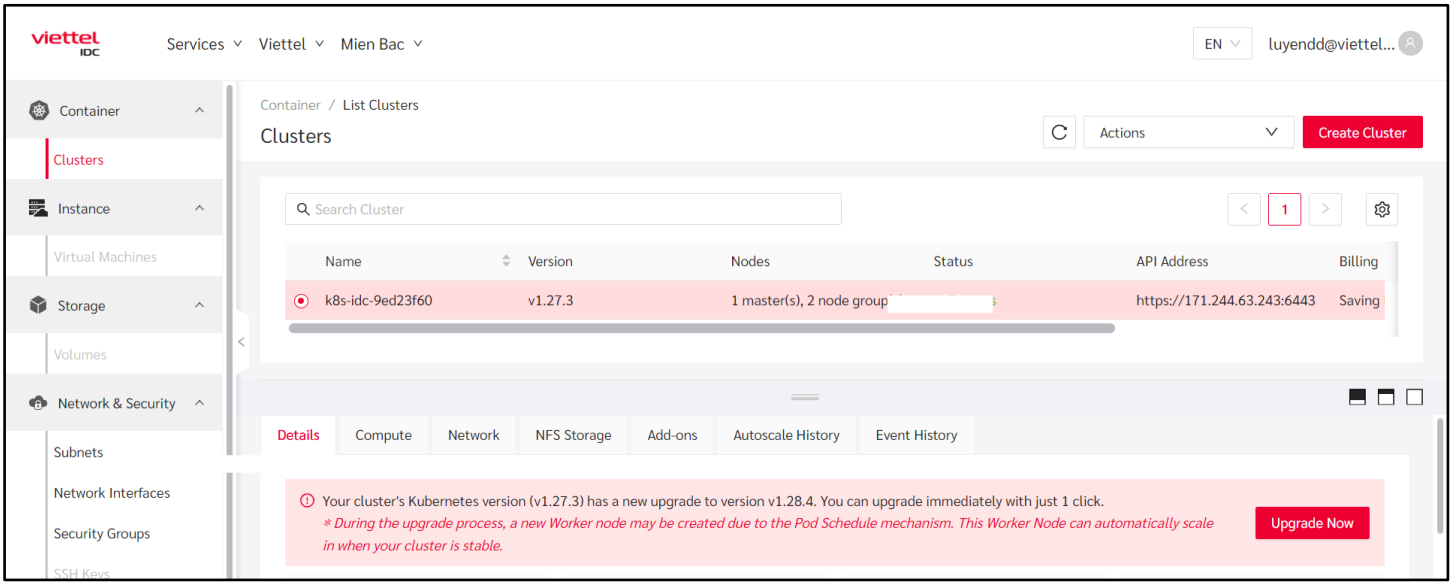
Step 1: On the Service Management screen, the user clicks on "Viettel Open Kubernetes Service" to access the Service Management portal (Cloud Management Platform – CMP).



Step 2: The user is redirected to the CMP (Cloud Management Platform) interface for service management.

- **List Cluster Section:** Displays a list of the user's existing Kubernetes (K8s) clusters.

- **Details Tab:** Provides detailed information about the K8s cluster, including cluster name, order ID, status, API address, number of Node Groups, billing method, VPC network name, and service start and end dates.
- **Compute Tab:** Displays detailed information about the nodes (Master, NFS) and Node Groups within the cluster.
- **Network Tab:** Displays network-related information, such as Security Groups.
- **NFS Storage Tab:** Displays information about NFS Storage – Persistent Storage.
- **Activity History Tab:** Shows the history of scale-out and scale-in events when nodes are added or removed using autoscale.
- **Add-ons Tab:** Allows users to install management and performance monitoring tools for the cluster, such as Dashboard, Prometheus, Grafana, and Viettel Cloud Watch.
- **Event History Tab:** Displays the history of interactions affecting the Kubernetes cluster.

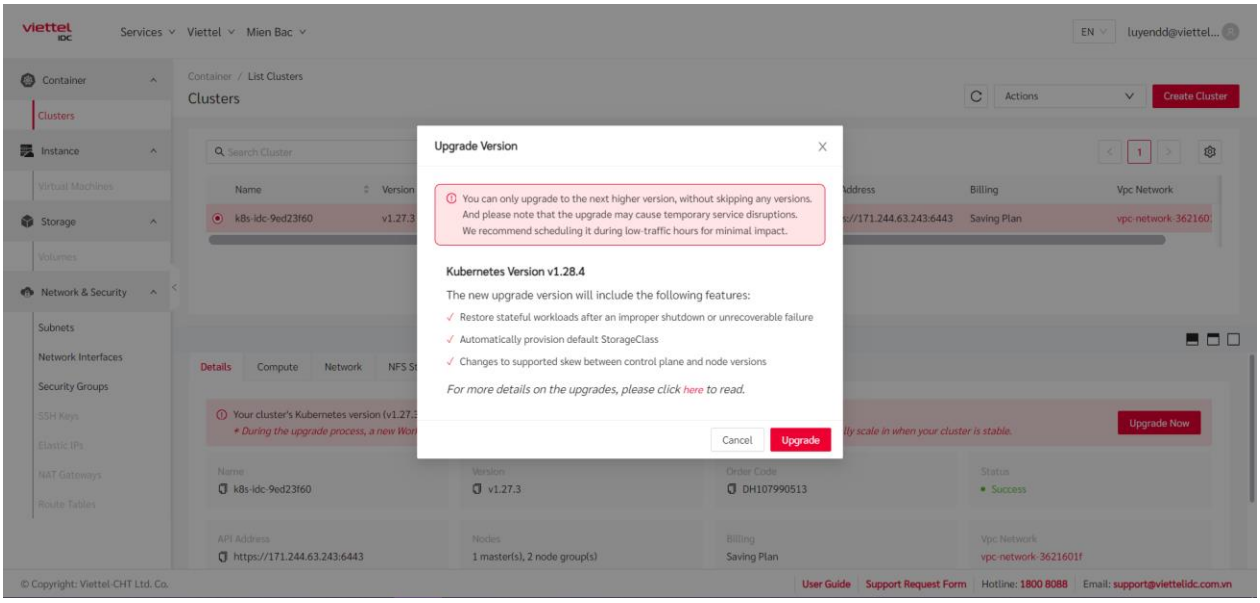


2.5 Upgrading Kubernetes Cluster Version

Viettel IDC supports users in upgrading Kubernetes clusters directly on the service management platform. Users can click "Upgrade Now" to perform the upgrade.

Note:

- Users can only upgrade to the next immediate higher version. For example, upgrading from version 1.27 to 1.28.
- The upgrade process may cause temporary service interruptions. Viettel IDC recommends scheduling the upgrade during off-peak hours to minimize downtime impacts.
- During the upgrade process, a new Worker Node may be created due to the Pod Scheduling mechanism. This Worker Node can automatically scale down once the Kubernetes cluster stabilizes.



2.6 Best Practices for Upgrading Kubernetes Cluster Versions

To ensure a seamless and satisfying experience when upgrading your cluster, here are some recommendations to consider throughout the upgrade process.

2.6.1. Preparation before Upgrading:

- Check the current version and target version to ensure they are within the supported upgrade range.
- If data is stored on nodes, back up the data to prevent potential data loss.
- Perform upgrades during periods of low system load (after business hours or on weekends).
- Before upgrading the production environment, test whether applications are compatible with the new K8S version in the development or UAT environment, particularly the APIs of the new K8S version.
- Back up the cluster configuration before upgrading: use CD tools or back up the manifest/Helm chart.
- Use an HA (High Availability) cluster with 3 master nodes and HA worker nodes (with at least 2 workers) to ensure zero downtime.
- Deploy applications using Workload Controllers (Deployment, ReplicaSet, DaemonSet) with replicas ≥ 2, avoiding the use of standalone Pods.

2.6.2. Performing the Upgrade:

- Ensure the cluster is always using the latest version.
- Avoid deploying applications that rely on local storage (e.g., emptyDir, hostPath). Instead, use CSI (Container Storage Interface)
- Upgrade sequentially through each version (e.g., 1.28 → 1.29 → 1.30).

- Viettel IDC recommends that customers upgrade to the latest version supported by Viettel IDC from the Kubernetes version list.

- During the upgrade process, Viettel IDC has designed resource sizing for applications to ensure sufficient resources (during the cluster upgrade, an additional worker node can be scaled as a buffer before upgrading the cluster version)..

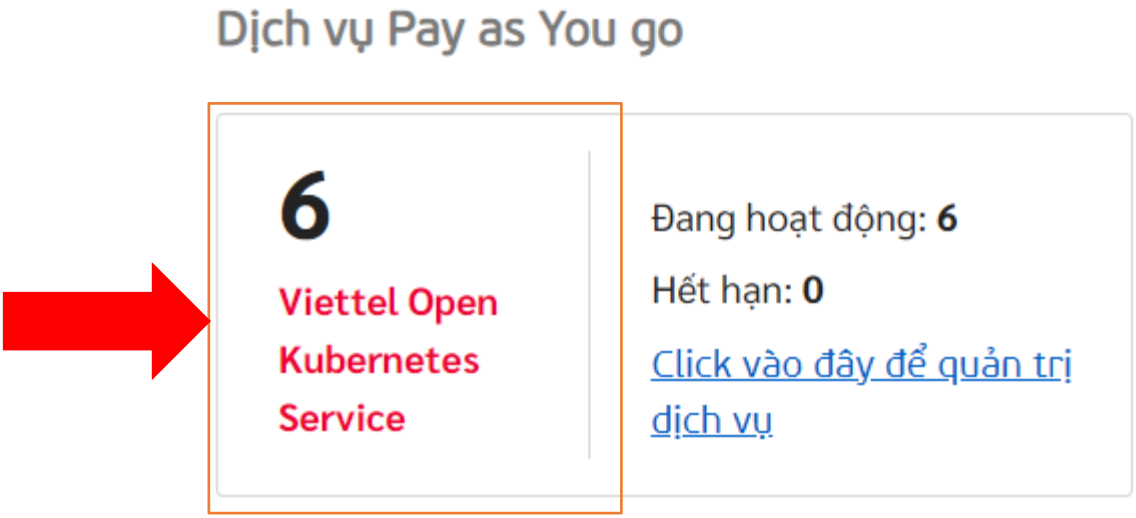
2.6.3. After the Upgrade:

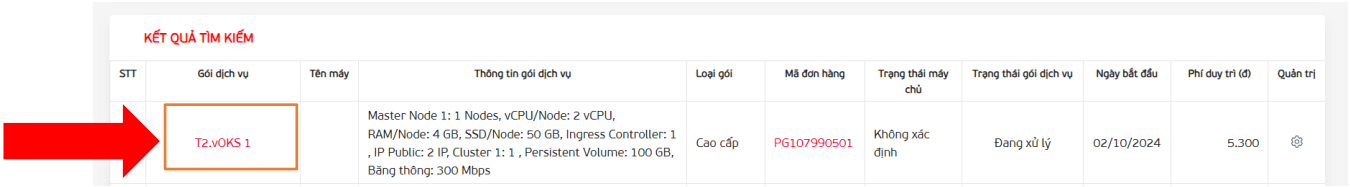
- Plan periodic cluster version upgrades to avoid using unsupported versions.
- Maintain monitoring and alert systems to track cluster performance and status after the upgrade.

2.7 Deleting a Cluster

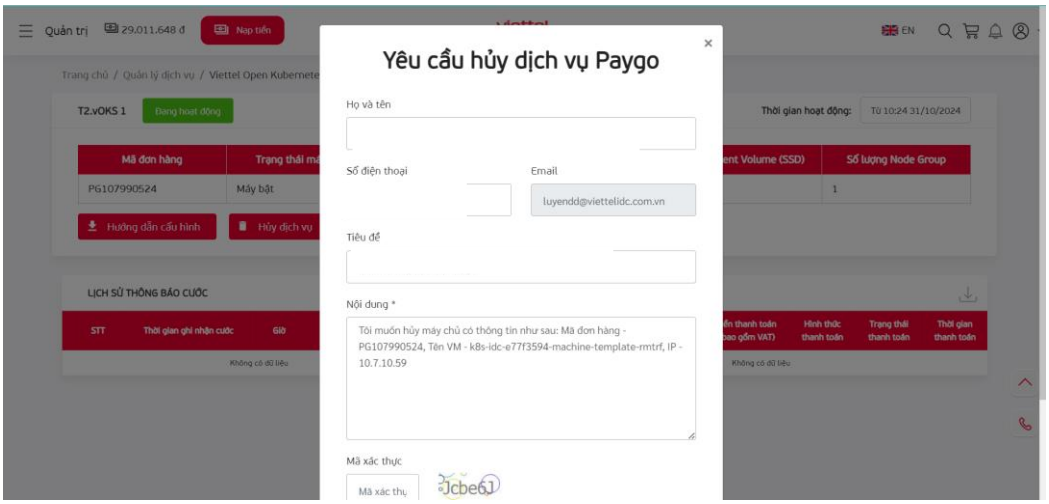
Currently, the vOKS service only supports cluster deletion requests via tickets for the PAYG model. For the Saving Plan model, users cannot delete clusters due to policies related to invoicing and prepaid services.

Step 1: On the Service Management screen, users click on "Viettel Open Kubernetes Service" to access the subscription management screen. Click on the subscription name to access the cluster deletion request page.





Step 2: The user clicks on "Submit Service Deletion Request" and fills out the form to allow Viettel IDC to contact and assist with the cluster deletion process.



3. Access Management

3.1 Using the Identity Access Management (IAM) Feature

Identity Access Management (IAM) is a feature developed and operated by Viettel IDC, allowing administrators to manage the access rights of other users through core IAM functionalities such as:

- **IAM User**
- **IAM Group**
- **IAM Policy**
- **IAM Session**
- **Monitor**
- **Account Setting**

To fully understand how to use all the IAM features provided by Viettel IDC, users can refer to the detailed guide [here].

3.2 Service vOKS Role-Based Policies

To support users in effective management, the vOKS service provides the following access control policies:

- **List:** Allows users to view information on the management interface.
- **Read:** Allows users to view detailed resource information.
- **Write:** Enables users to create, update, or delete resources.

With these permissions, users can interact with different components of the cluster on the management interface provided by Viettel, including:

- **Cluster**
- **Cluster Instance**
- **Network**

- **Add-ons**

4. Resource Management

A Kubernetes Node is a virtual machine containing resources to run applications encapsulated in containers. Each Node consists of the following core components:

- **Container Runtime:** Software responsible for running containers.
- **Kubelet:** Ensures that containers are running normally and linked to their assigned Pods.
- **Kube-proxy:** Maintains network rules to enable communication with Pods.

For additional details, refer to the **Nodes** section on the Kubernetes documentation page.

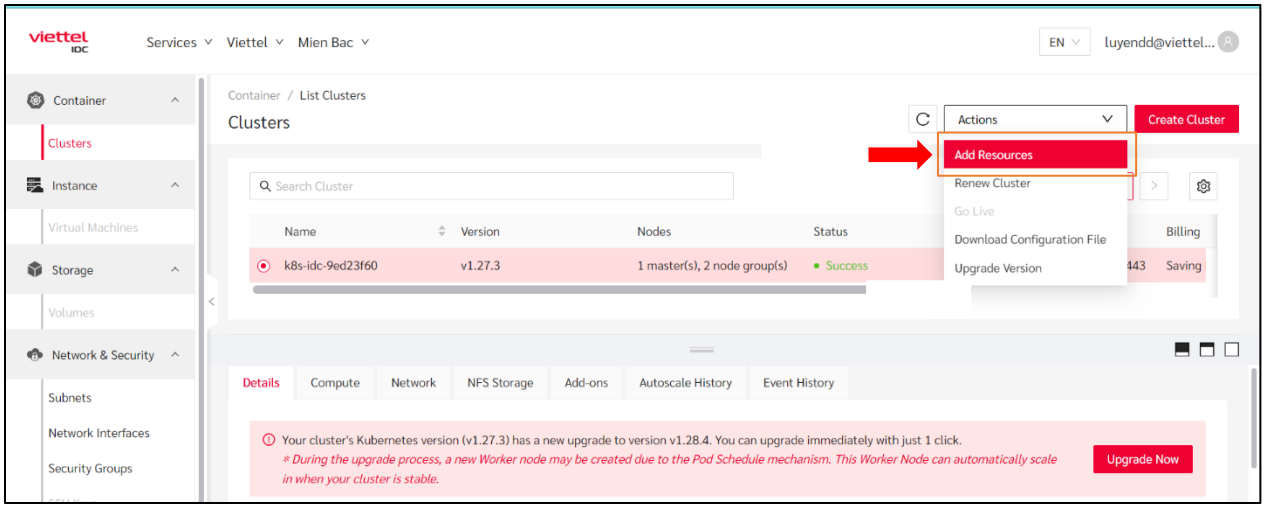
A **Node Group** is a collection of nodes with identical configurations, grouped to facilitate quick and easy configuration management. Node Groups can be scaled up or down flexibly and divided into multiple groups within a Kubernetes cluster to meet application deployment requirements.

Users can create additional Node Groups, view Node Group information, and adjust Worker Autoscale settings for each Node Group to optimize resource usage and application performance.

4.1 Creating a Node Group

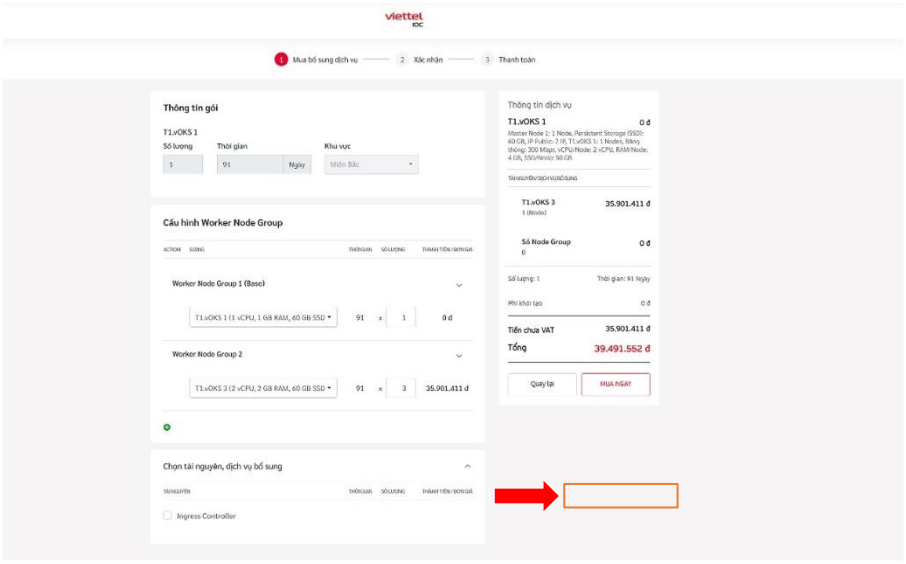
4.1.1 Adding a Node Group to a Saving Plan Cluster

Step 1: On the CMP interface, the user selects the cluster to which additional resources are to be added → then clicks **Action Add Resources**.



Step 2: The system redirects to the Add Resources screen, where the user can choose to add a new Node Group or purchase additional Worker Nodes for an existing Node Group:

- Click button ⊕ to add a new Node Group, adjust the sizing of the Node Group, and configure the default number of nodes in the Node Group.. The number of nodes can be modified based on the Autoscale feature of each Node Group.
- Select Worker Node, then input or drag and drop the number of Nodes to be added, and click Buy Now.



Step 3: Confirm the Order: Perform the same steps as when registering for a paid subscription.

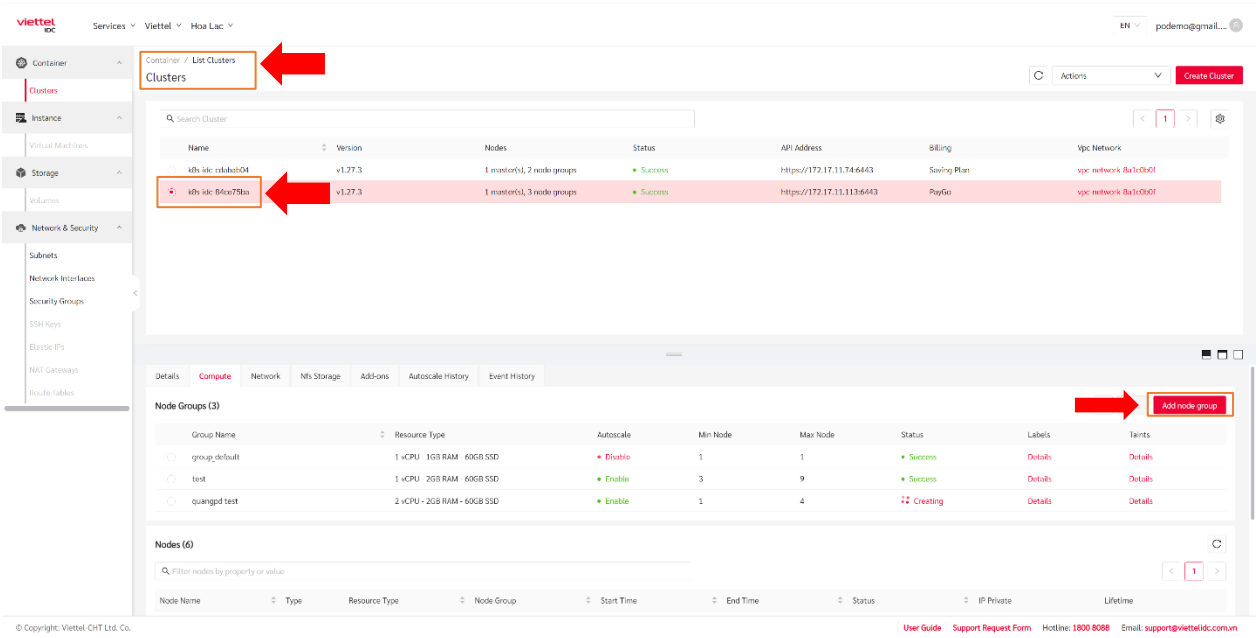
Step 4: Complete Payment: Perform the same steps as when registering for a paid subscription..

4.1.2 Adding a Node Group & Nodes for PAYG Service

When registering for the PAYG service, customers will receive one cluster with one Node Group for use. Customers can customize the min nodes and max nodes parameters to meet their autoscale configuration needs.

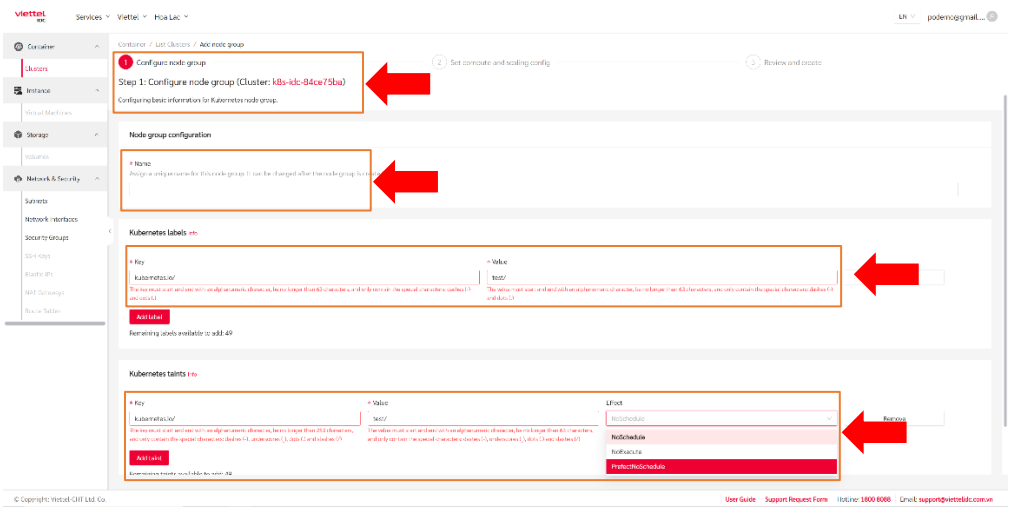
For adding additional Node Groups, customers should follow these steps:

Step 1: On the **List Clusters** screen, select the cluster where you want to add a Node Group and then go to the **Compute** tab.



Step 2: On the **Create Node Group** screen – Step 1: **Configure Node Group**, customers can:

- Enter a name for the Node Group as required.
- Fill in the desired **Labels** and **Taints** for the Node Group.
- Click **Next** to proceed.



Notes:

- The **Key** of Labels must:
  - Start and end with an alphanumeric character.
  - Be no longer than 63 characters.
  - Only contain two special characters: **period (.)** and **hyphen (-)**.
- The **Value** of Labels must:
  - Start and end with an alphanumeric character.
  - Be no longer than 63 characters.
  - Only contain two special characters: **period (.)** and **hyphen (-)**.
- The **Key** of Taints must:
  - Start and end with an alphanumeric character.
  - Be no longer than 253 characters.
  - Only contain four special characters: **period (.)**, **forward slash (/)**, **hyphen (-)**, and **underscore (_)**.

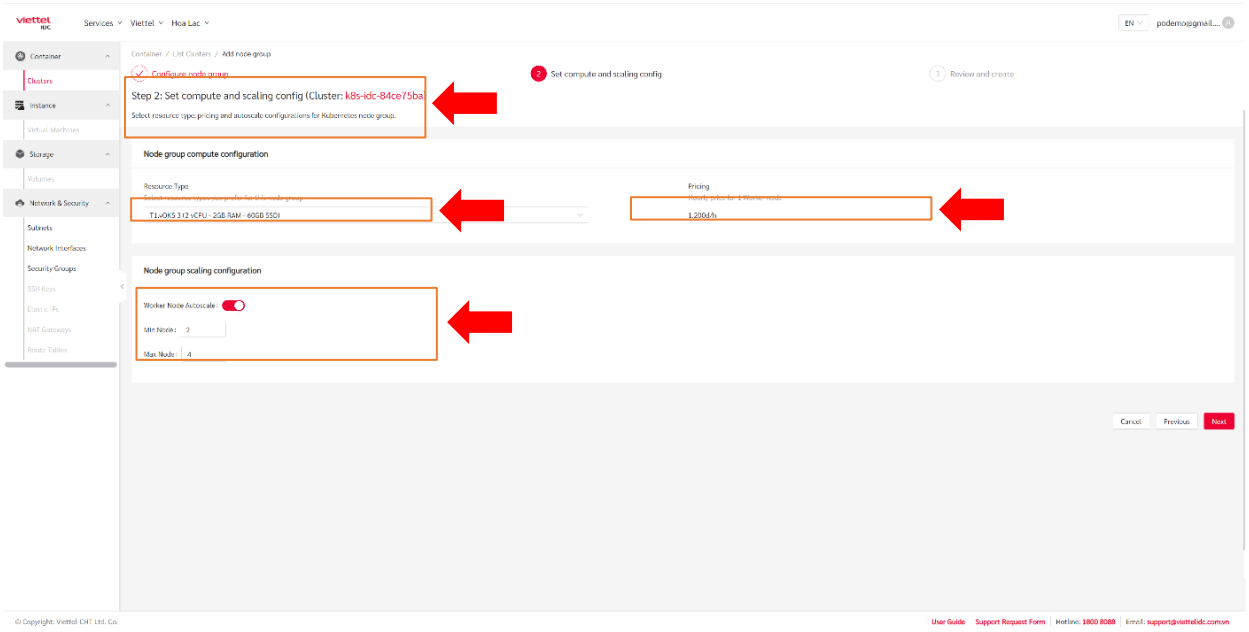
- The **Value** of Taints must:
    - Start and end with an alphanumeric character.
    - Be no longer than 63 characters.
    - Only contain four special characters: **period (.)**, **forward slash (/)**, **hyphen (-)**, and **underscore (_)**.

Effect options specify the behavior when Pods cannot tolerate the taint. Customers can choose one of the following:

1. **NoSchedule:** Pods without the required tolerations will not be scheduled on this Node.
2. **NoExecute:** Pods without the required tolerations will not be scheduled on this Node, and any Pods already running on the Node without the matching tolerations will be removed.
3. **PreferNoSchedule:** Kubernetes will avoid scheduling Pods without tolerations for the taint on this Node as much as possible.

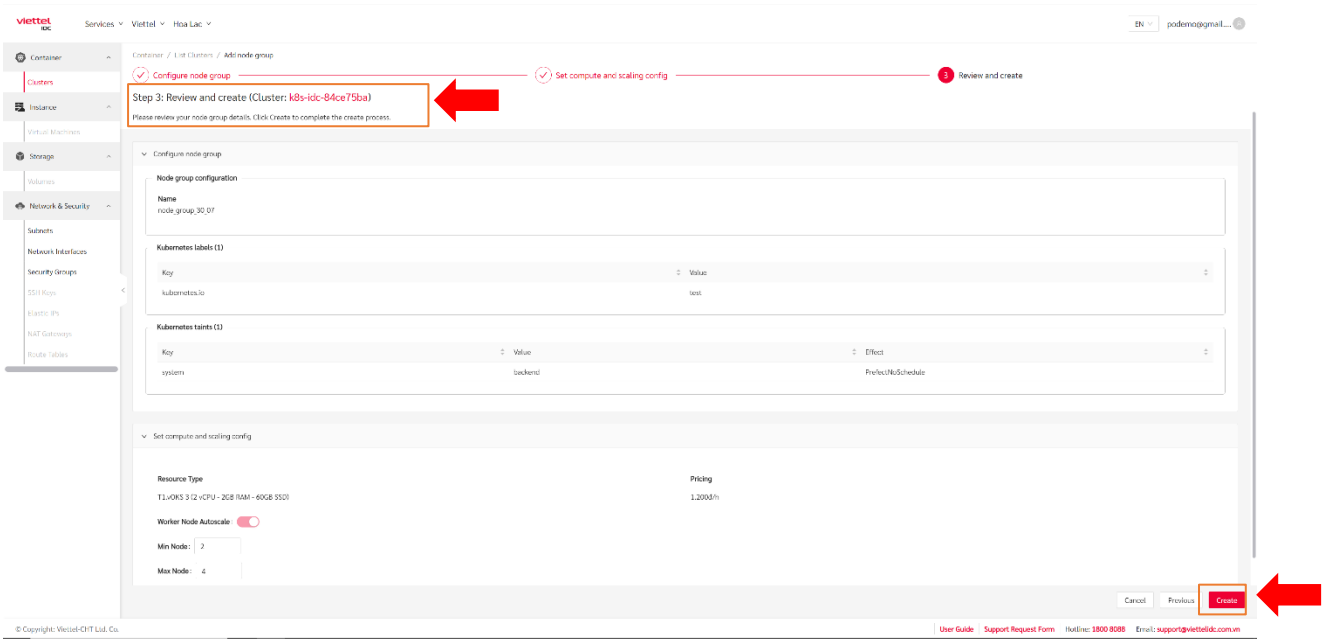
Step 3: On the **Create Node Group** screen – Step 2: **Set Compute & Scaling Config**, customers can:

- Select the configuration for the Node Group as needed.
- Enable or disable the **Worker Autoscale** feature.
- Enter the **minimum** and **maximum** number of nodes.
- Click **Next** to proceed.



Step 4: On the **Review Node Group Creation** screen – Step 3: **Review & Create,** customers can:

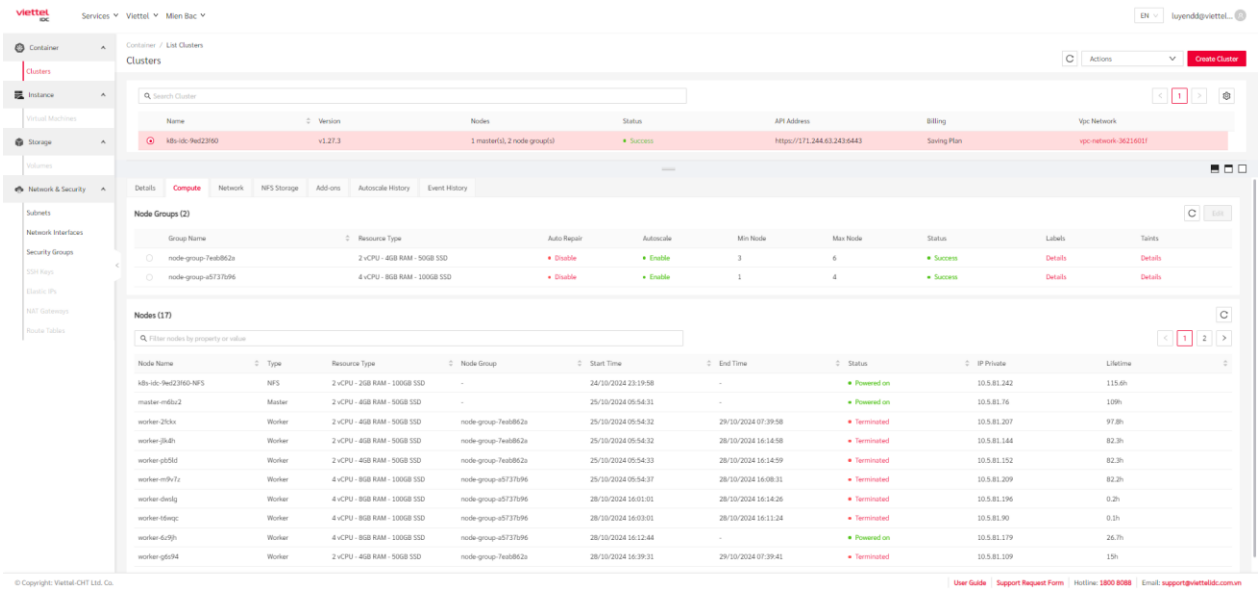
- Review the information for the Node Group to be created.
- Click **Create** to finalize and create the additional Node Group.



4.2 Viewing Node Group Information & Nodes Inside

Users can view information about Node Groups in the **Compute** tab, under the **Node Group** section. The Node Group table includes the Worker Nodes within the Node Group, while the Nodes table provides details on the number of nodes and specific information such as:

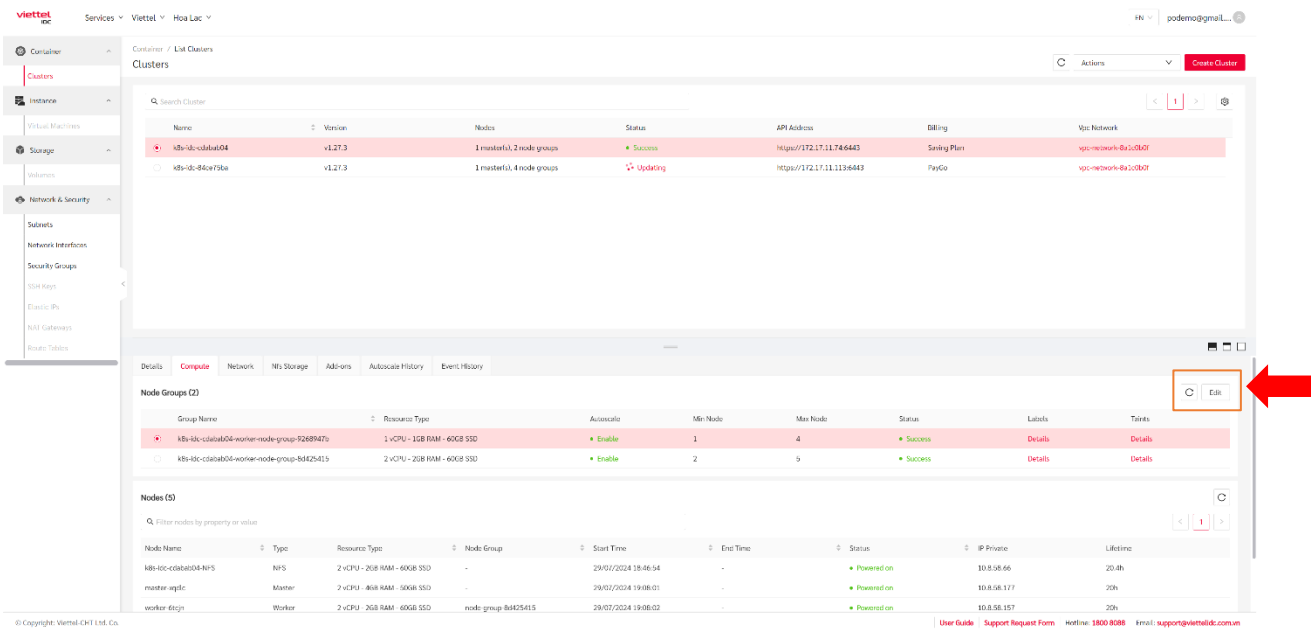
- **Node Name:** The name of the Node.
- **Type:** The type of Node, including three types: Master, Worker, and NFS Node (used for Persistent Volume).
- **Resource Type:** The configuration of the Node.
- **Node Group:** The configuration group to which the Node belongs.
- **Start Time:** The time the Node began operating.
- **End Time:** The time the Node stopped operating.
- **Status:** The status of the Node, which can be:
    - **Powered On:** The Node is active.
    - **Terminated:** The Node has stopped operating and cannot be restored.
- **IP Private:** The private IP address of the Node.
- **Lifetime:** The duration of the Node's operation, allowing customers to track operational time for PAYG billing.



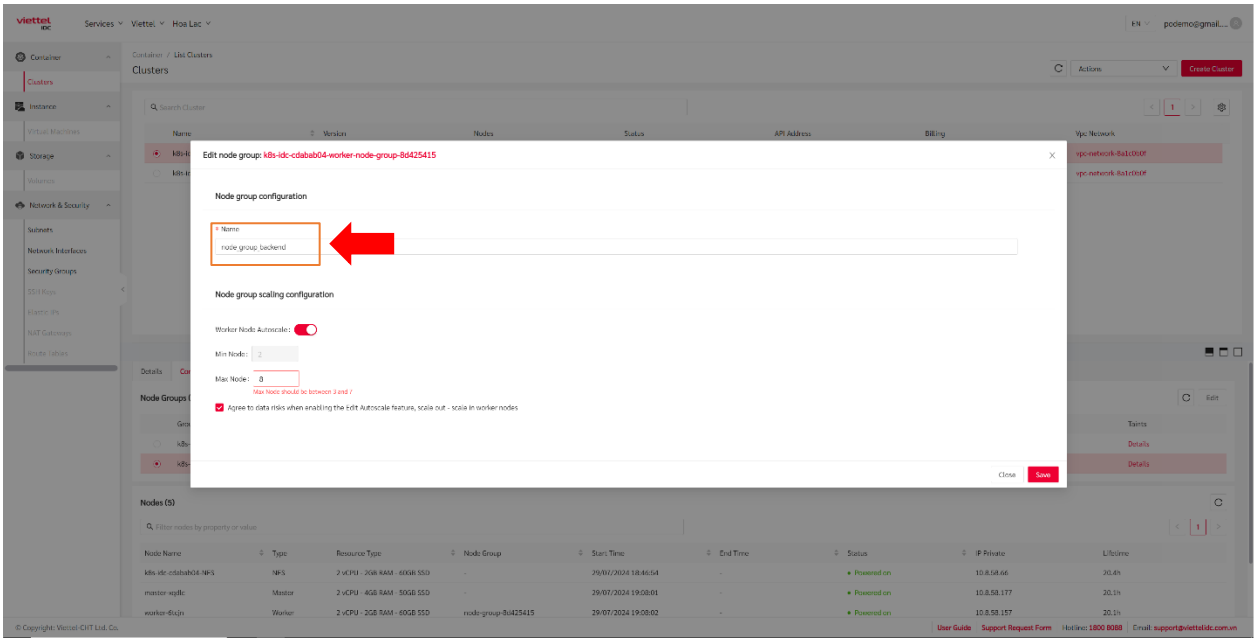
4.3 Editing Node Group Information

4.3.1 Editing the Node Group Name

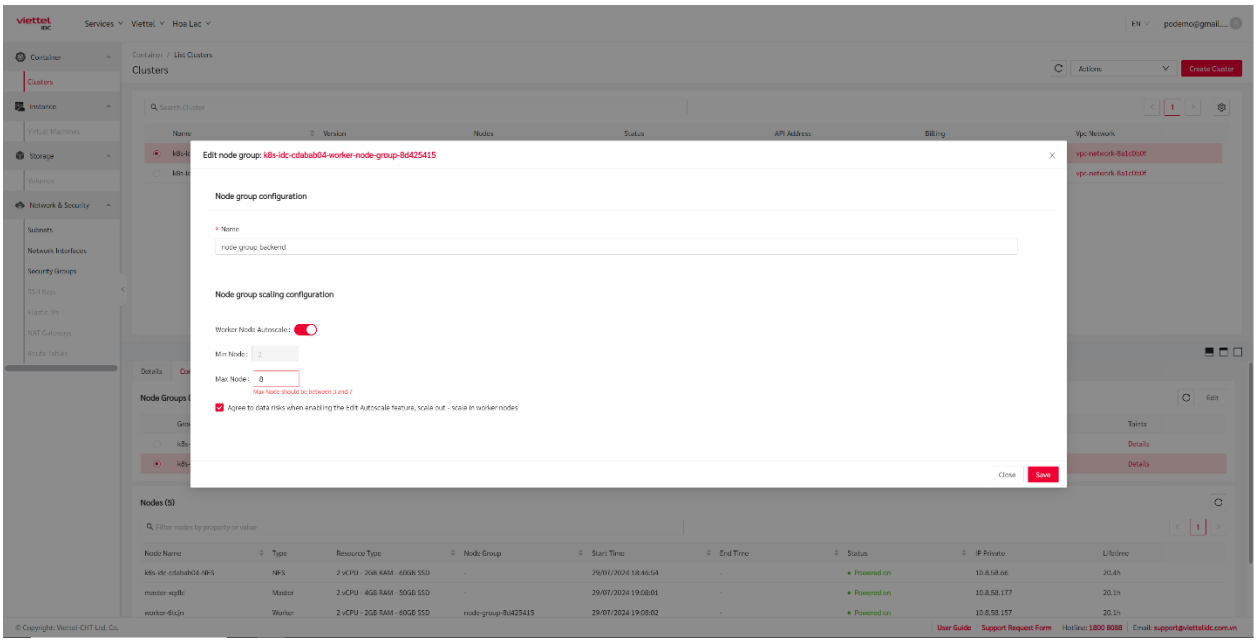
Step 1: The user selects the desired cluster, chooses the Node Group to rename, and clicks **Edit**.



Step 2: The user can update the name as desired and click **Save** to save the changes.



4.3.2 Editing Worker Autoscale Configuration for a Node Group



Users can configure the **Worker Autoscale** feature for each Node Group to meet resource demands. Viettel IDC currently supports Autoscale for both the **Saving Plan** and **PAYG** models with the following specifics:

- **For Saving Plan:** Users can configure up to 5 additional Worker Nodes beyond the minimum number of Worker Nodes purchased for a Node Group.
- **For PAYG:** Users can flexibly configure Autoscale for up to 10 Worker Nodes, allowing dynamic scaling to meet application resource needs.

Steps to Configure Worker Autoscale:

Step 1: Select the cluster, choose the Node Group to edit, and click **Edit**.

Step 2: Click **Enable**, then enter the desired **Min Node** and **Max Node** values to set resource thresholds for the cluster:

- **Min Node:** The number of nodes that remain after scaling in (reducing nodes).
- **Max Node:** The maximum number of nodes allowed during scaling out (increasing nodes).

Step 3: Agree to data risks by enabling the **Edit Worker Autoscale** feature and confirm acceptance of potential data risks when scaling out and scaling in Worker Nodes.

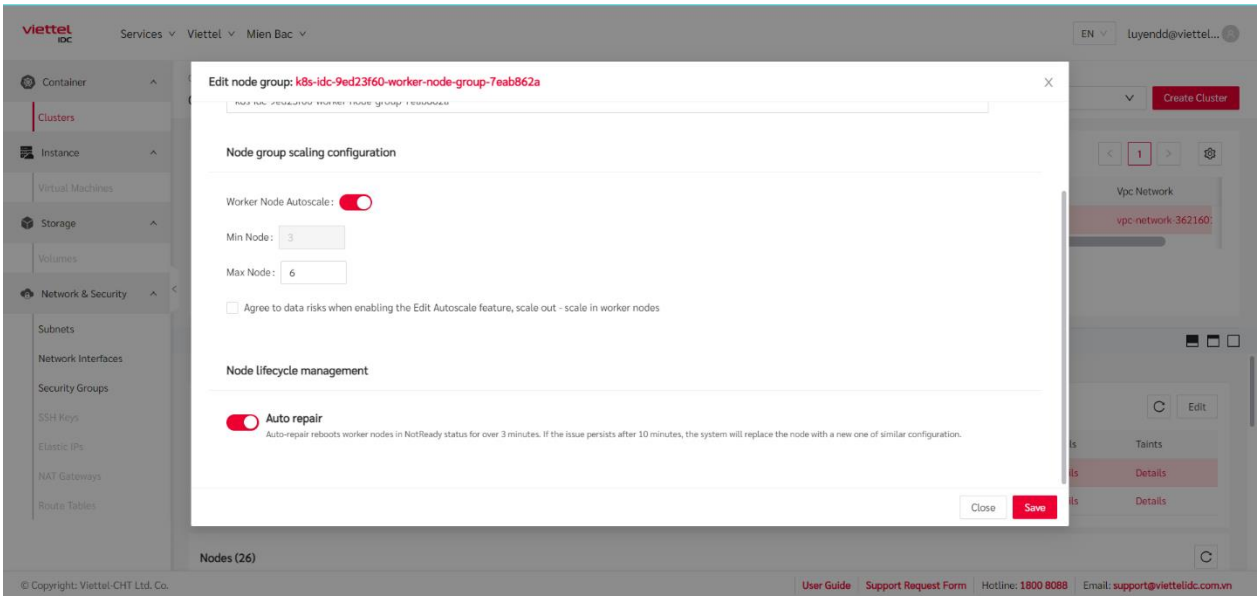
Step 4: Click **Update** to save the Worker Autoscale configuration.

Note:

When the cluster reaches its resource threshold and requires new Pods to run services, new Worker Nodes will be created within the range of **Min Node** to **Max Node**. If the number of nodes reaches the **Max Node** limit and the service still requires additional Pods, the system

will not create more Worker Nodes. In such cases, users must update the configuration to increase the **Max Node** limit for continued resource availability.

4.3.3 Editing Auto-Repair Configuration for a Node Group



In addition to the Worker Autoscale feature, Viettel IDC's vOKS service provides an **Auto-Repair** feature for Node Groups, offering several key benefits to ensure high availability and reliability for applications.

Key Features of Auto-Repair:

- **Automatic Fault Detection:** Kubernetes can automatically detect faulty or problematic nodes by monitoring their status. Indicators of faulty nodes include "NotReady" status, inability to ping the node, hardware failures, etc.
- **Automatic Node Restart:** When a node is identified as faulty, Kubernetes automatically restarts the node. Restarting can resolve temporary issues and restore the node to normal operation.
- **Reduced Manual Intervention:** Auto-repair minimizes the need for manual intervention by system administrators, saving time and effort.

- **Enhanced Operational Efficiency:** Auto-repair ensures nodes are always functioning correctly, improving the system's overall efficiency.

If a node cannot return to the "Ready" state after the auto-repair process, the system will replace the faulty "NotReady" node with a new one of similar configuration after 10 minutes.

By default, the Auto-Repair feature is **disabled** for all Node Groups. Users can enable or disable this feature for specific Worker Groups within the cluster.

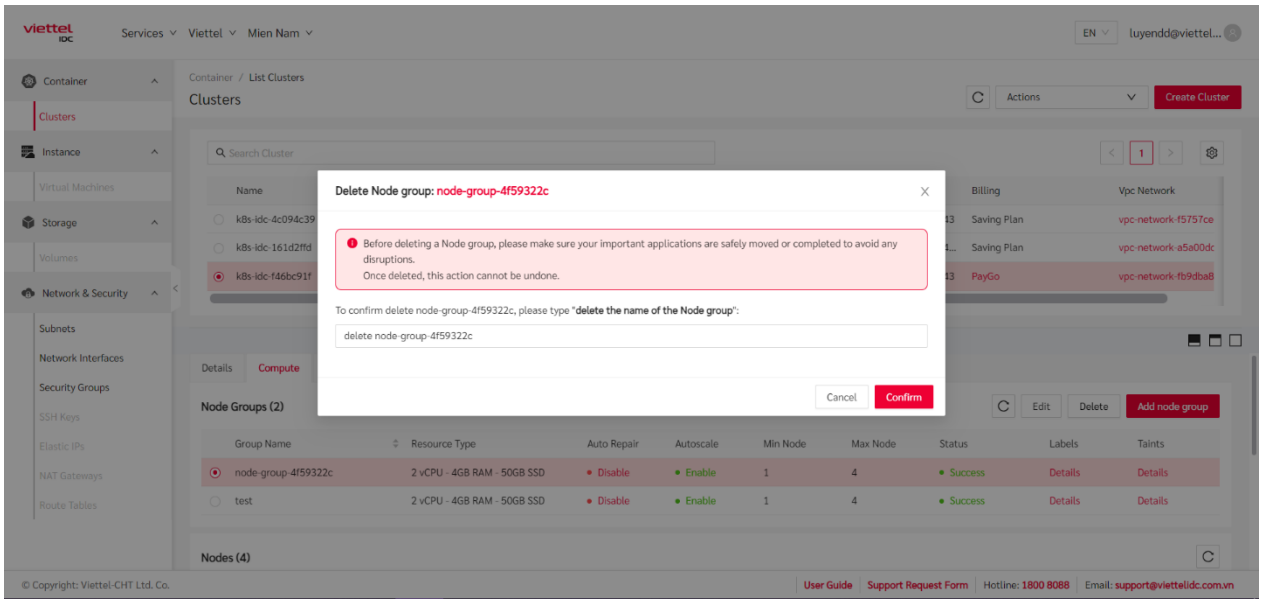
4.4 Delete Node group

For the **PAYG** payment model, users can easily adjust the number of Node Groups within the cluster by using the Node Group deletion feature.

Steps to Delete a Node Group:

Step 1: Select the cluster and the Node Group to delete, then click **Delete**.

Step 2: Enter the command in the format **delete** + **Node group name,** then click **Confirm** to delete the Node Group.



Notes:

- A cluster must have at least **one Node Group** remaining after deletion. To delete all Node Groups, users must delete the entire cluster via the ticketing process.
- Before deleting a Node Group, users should ensure that critical applications are safely migrated or completed to prevent service interruptions.
- Once a Node Group is deleted, this action **cannot be undone.**

5. Application Data Storage

5.1 Introduction to Volume & NFS Node

In Kubernetes, new Pods are created to replace failed, crashed, or terminated Pods. But what happens to the data stored in the old Pods? Can the new Pods retrieve the data from the old ones? **Volumes** are designed to address these issues.

Volumes are components associated with Pods. They are defined in the .yaml configuration file when Pods are initialized. Containers can mount data within themselves to volumes that belong to the same Pod.

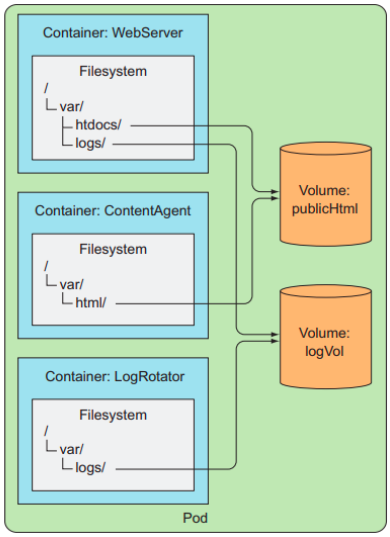


Diagram: Containers in a Pod mount to two volumes to share data with each other.

Types of volumes:

- EmptyDir
- HostPath
- GitRepo
- NFS
- Cloud volumes: gcePersistentDisk (Google Compute Engine Persistent Disk), awsElasticBlockStore (Amazon Web Services Elastic Block Store Volume), azureDisk (Microsoft Azure Disk Volume)
- Cinder, cephfs, iscsi, flocker, glusterfs, quobyte, rbd, flexVolume, sphereVolume, photonPersistentDisk, scaleIO
- ConfigMap, secret, downwardAPI
- PersistentVolumeClaim
- As shown, there are many types of volumes. However, the **vOKS service** focuses on providing a **Persistent Volume (PV)** solution via **NFS nodes**.
- vOKS can automatically add NFS nodes to the Kubernetes configuration when creating a new cluster. Administrators can use NFS nodes to implement shared

storage across multiple containers, enabling better deployment of stateful applications where data is consistently stored in fixed partitions.

- This section will describe NFS node management and provide an example of how to set up storage on NFS. Customers are expected to have a basic understanding of Kubernetes storage and NFS management.

5.2 Types of Storage Connections on NFS Nodes

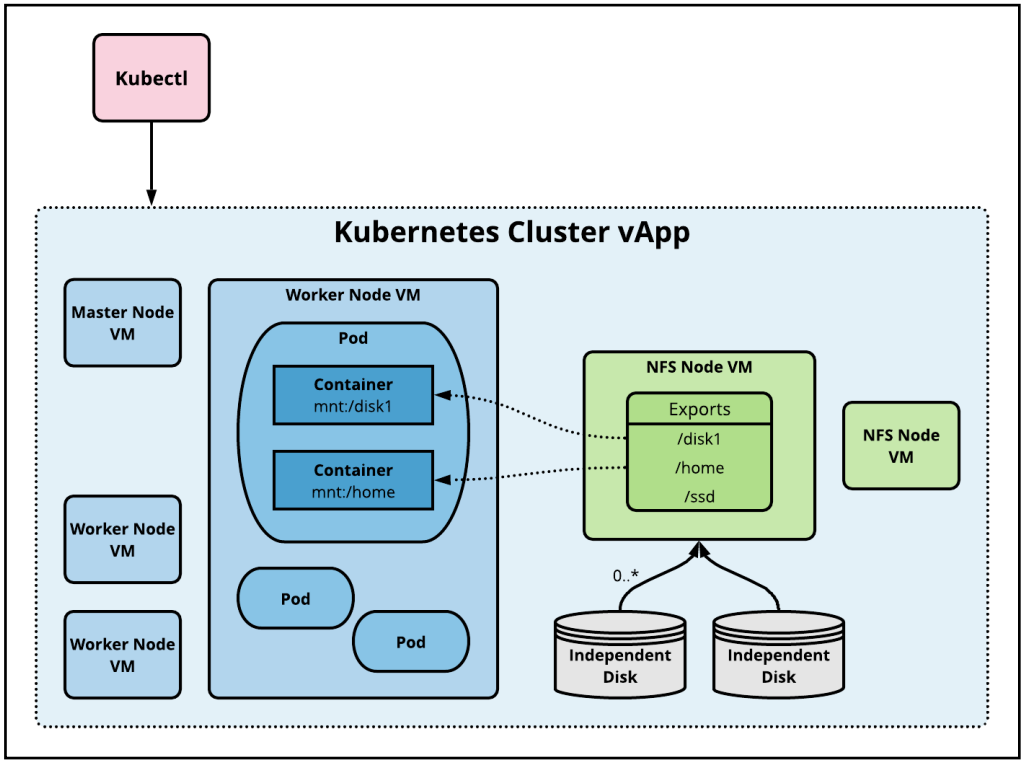
Static storage is pre-allocated by the cluster administrator. It contains details about the actual storage capacity available for use. These storage allocations exist within the Kubernetes API and are ready for use. Users can allocate static storage by creating a persistent volume claim (PVC) for the same storage partition or customizing it based on container requirements. vOKS supports pre-allocated static storage on NFS nodes.

Dynamic storage is not pre-allocated by the cluster administrator. Currently, vOKS does not support dynamic persistent storage provisioning.

5.3 NFS Node Architecture

An NFS node allows sharing one or more storage partitions across one or multiple cluster nodes. When a cluster is deleted, the data stored on the NFS node remains intact, and container connections to the storage partitions are disconnected. This means that data stored on an NFS node can be shared and used simultaneously across multiple clusters, even if a node in the cluster is deleted or fails. Furthermore, NFS supports multiple concurrent write sources.

Diagram: Deployment architecture for using an NFS node:

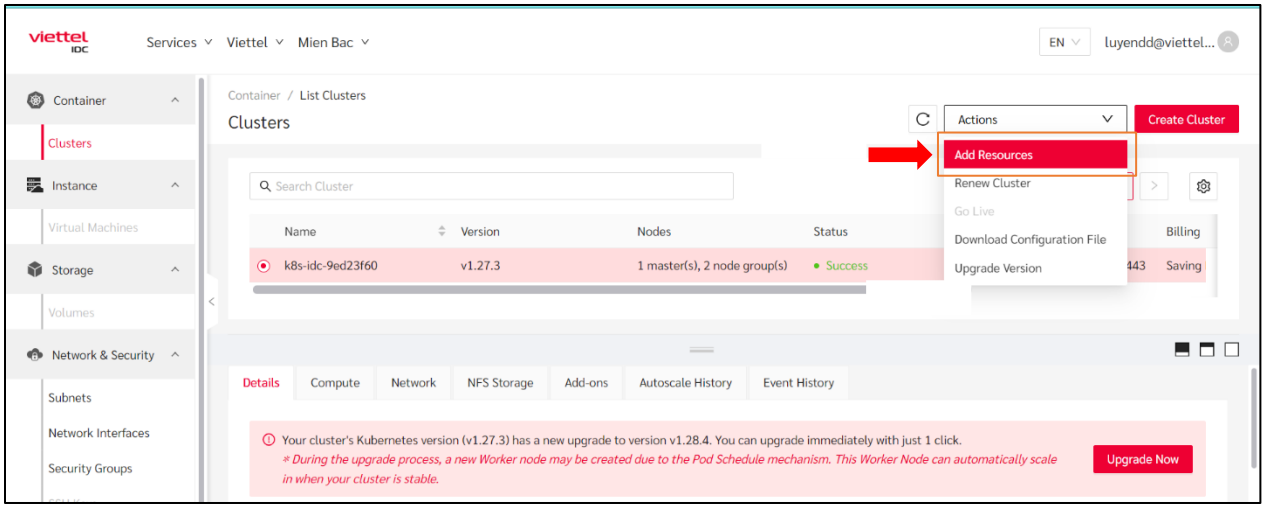


5.4 Adding NFS Node Capacity

Users can easily add capacity to NFS Nodes through the management interface developed by Viettel IDC.

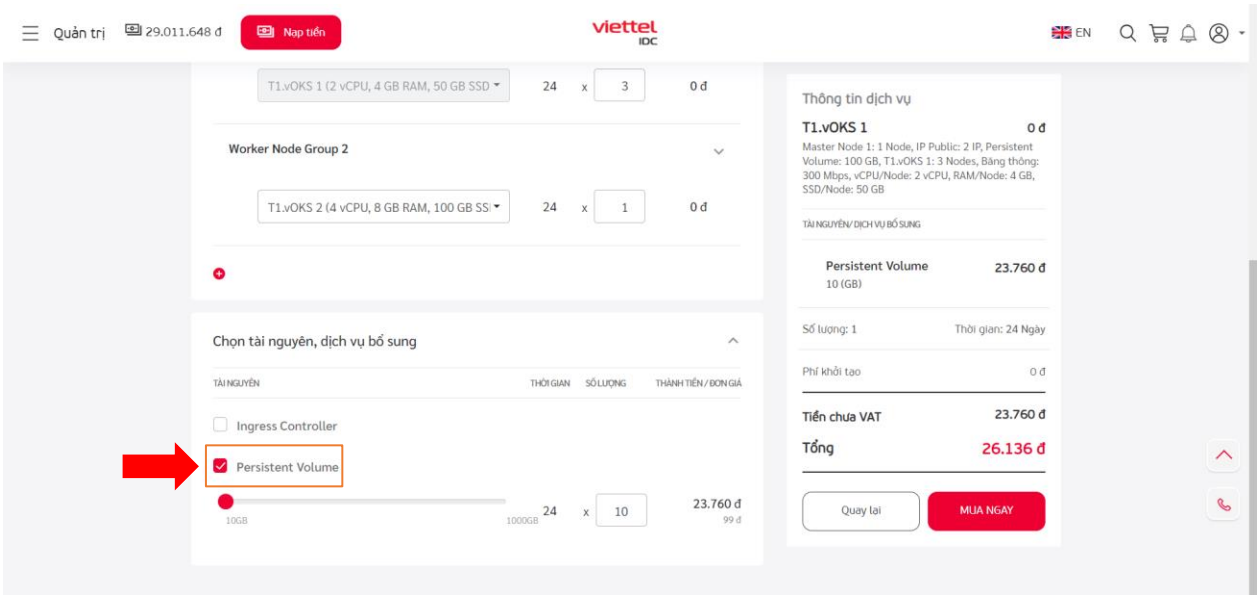
5.4.1 Adding NFS Node Capacity for Saving Plan Clusters

Step 1: On the CMP interface, the user selects the cluster for additional capacity → then clicks **Action Add Resources**.



Step 2: The system redirects to the **Add Resources** screen.

- The user selects **Persistent Volume** to add more NFS capacity.
- Input or drag and drop the desired capacity to purchase.
- Click **Buy Now**.



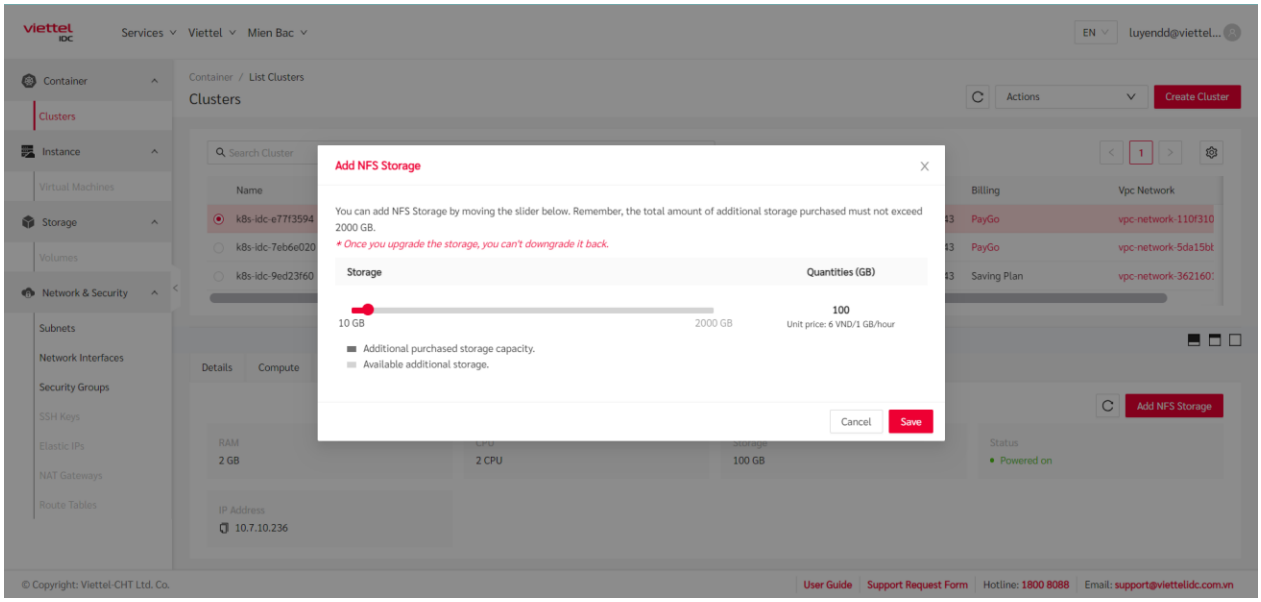
Step 3: Confirm the Order: Follow the same steps as when *creating a new Node Group*

Step 4: Complete Payment: Follow the same steps as when *creating a new Node Group*

5.4.2 Adding NFS Node Capacity for PAYG Clusters

To add more capacity to an NFS Node, customers should proceed as follows:

Step 1: On the **List Clusters** screen, select the cluster requiring additional capacity, then go to the **NFS Storage** tab.



Step 2: The user drags and drops the additional NFS Storage capacity required for the cluster. The resource usage details will be displayed on [the billing management screen.]

5.5 Instructions for Configuring NFS Node Usage

- **Creating an NFS Node:**

  The system will automatically create the NFS node, format the disk filesystem to the ext4 format, initialize the shared directory with a default mount point, and configure the system to automatically mount the shared directory into the /etc/fstab file in case the NFS node is restarted.

- **Configuring Storage Partitions on Kubernetes:**

  Currently, the NFS Node provides two volume partitions: **root disk** and **shared data storage**. The directory /data/mydata (default) will be pre-mounted with the shared data volume partition and shared across all NFS clients.

Note:

The path name /data/mydata must be accurately specified when creating persistent volumes to ensure the allocated resources in the NFS Node are utilized. If not, an error may occur when Kubernetes attempts to access the nodes in the cluster.

- **Step 1:** Install the helm-chart csi-driver-nfs from the client side:

```
helm repo add csi-driver-nfs https://raw.githubusercontent.com/kubernetes-csi/csi-driver-nfs/master/charts
helm install csi-driver-nfs csi-driver-nfs/csi-driver-nfs --namespace kube-system --version v4.6.0
```

- **Step 2:** Configure the default **StorageClass**.

StorageClass Configuration File (default-sc.yaml):

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-driver-nfs-default-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: 'true'
provisioner: nfs.csi.k8s.io
parameters:
  server: 10.14.53.175 # IP of nfs server
```

```
  share: /data/mydata
reclaimPolicy: Delete
mountOptions:
  - nfsvers=4.1
volumeBindingMode: Immediate
```

Create StorageClass:

```
kubectl create –f default-sc.yaml
```

Service Verification:

```
kubectl get sc -A -owide
kubectl get po -A
```

6. Network Configuration

6.1 Private Connection to Other Services within the Same VPC Network

When selecting to create a new VPC Network or using an existing VPC Network for the cluster during the cluster initialization step. Users can easily establish a private connection with other services within the Viettel IDC product ecosystem, such as vOKS and vOVPC. This offers key features like high-speed bandwidth, data and connection security, ensuring information safety and compliance with the highest security standards for IT systems.

6.2 Connecting to Viettel Open Virtual Private Cloud (vOVPC)

Case 1: vOVPC service VM uses a **Public Subnet**

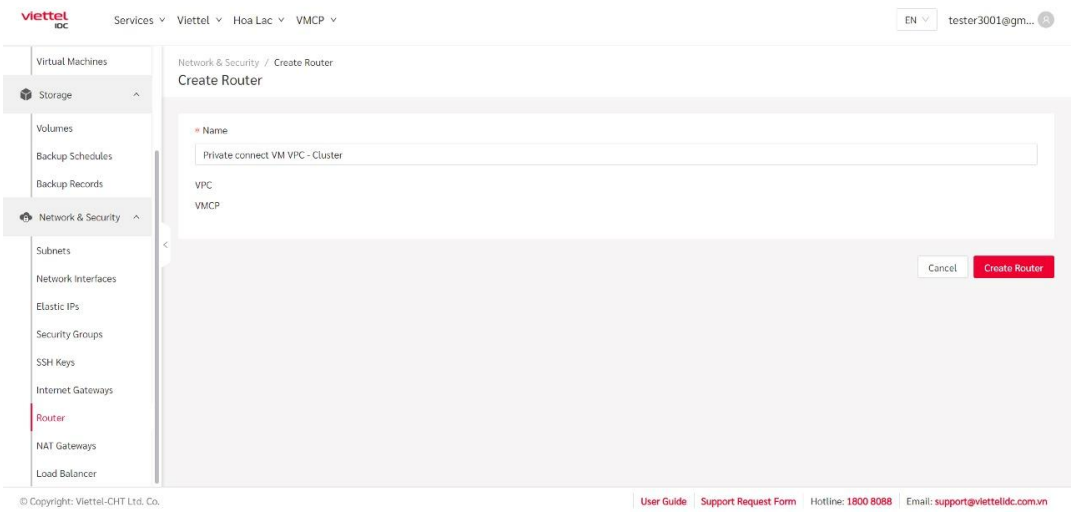
In this case, the vOKS Kubernetes cluster and the vOVPC service VM will automatically connect to each other by default.

Case 2: vOVPC service VM uses a **Private Subnet**

To enable interaction with the vOKS Kubernetes cluster, users need to follow these steps:

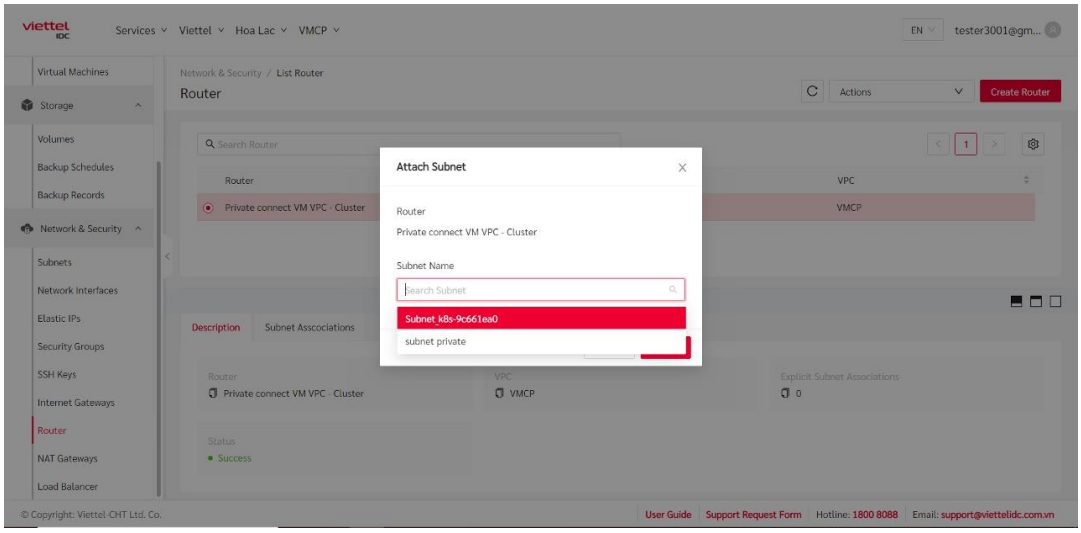
Step 1: Create a router in the **Network & Security** management interface

- Navigate to the vOVPC service management page → **Network & Security** tab → **Router** → select **Create Router** and enter a name for the router.



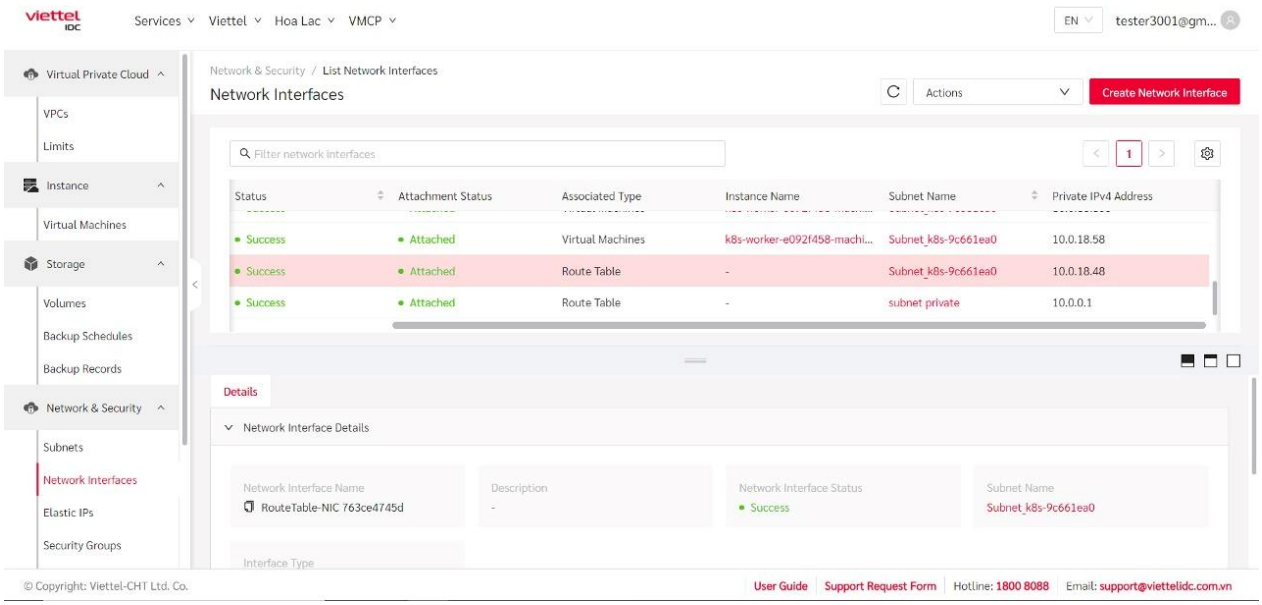
Step 2: Attach Two Subnets for the VM (X.X.X.0/24) and the vOKS Cluster (Y.Y.Y.0/24) to the Router

- Select the router you just created → click **Actions** → **Attach Subnet** → sequentially select the two corresponding subnets for the VM and the Kubernetes cluster.



Step 3: Retrieve the IPv4 Addresses of the Two Network Interfaces with the Type "Route Table" Corresponding to the Two Subnets

- Navigate to the vOVPC service management page → **Network & Security** tab → **Network Interface** → locate the two Network Interfaces with the type "Route Table" corresponding to the two subnets.



Eg: IPv4 Address of Subnet VM is 10.0.0.1

IPv4 Address of Cluster K8s is 10.0.18.48

Step 4: Add a Route for the vOVPC VM

Access the vOVPC VM you want to connect to the Kubernetes cluster and add the route:

ip r add Y.Y.Y.0/24 via x.x.x.x

Step 5: Add a Route for the Kubernetes Cluster

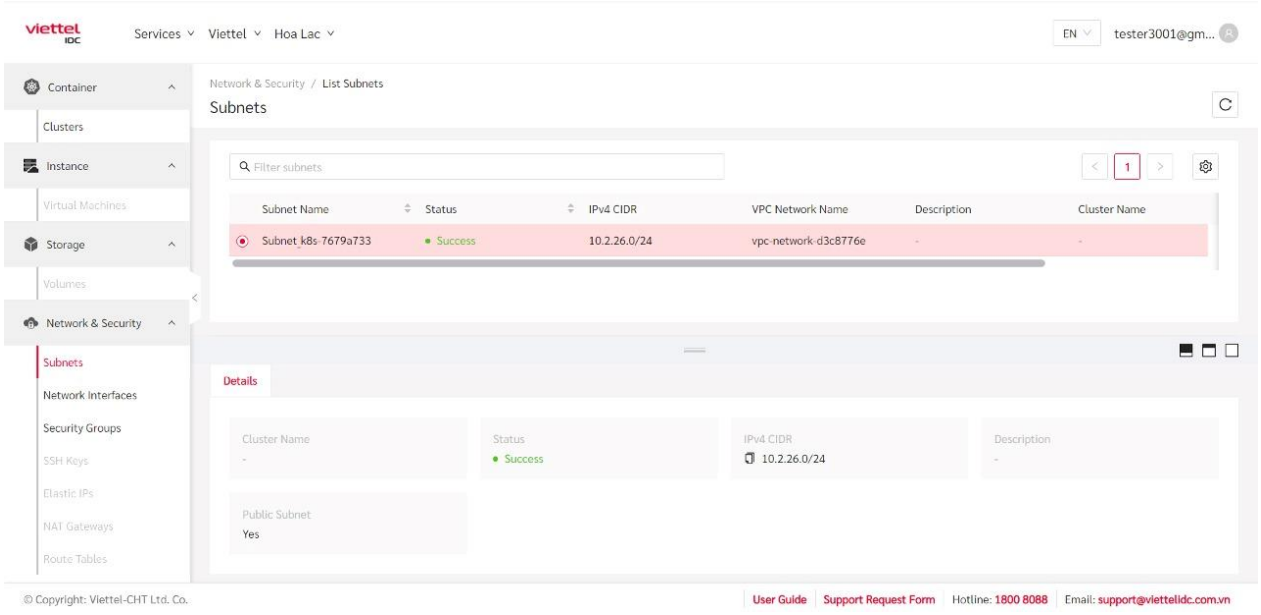
- Contact Viettel IDC via email at **support@viettelidc.com.vn** for assistance in configuring the network within the nodes.

6.3 Connecting to Viettel Database Service (vDBS)

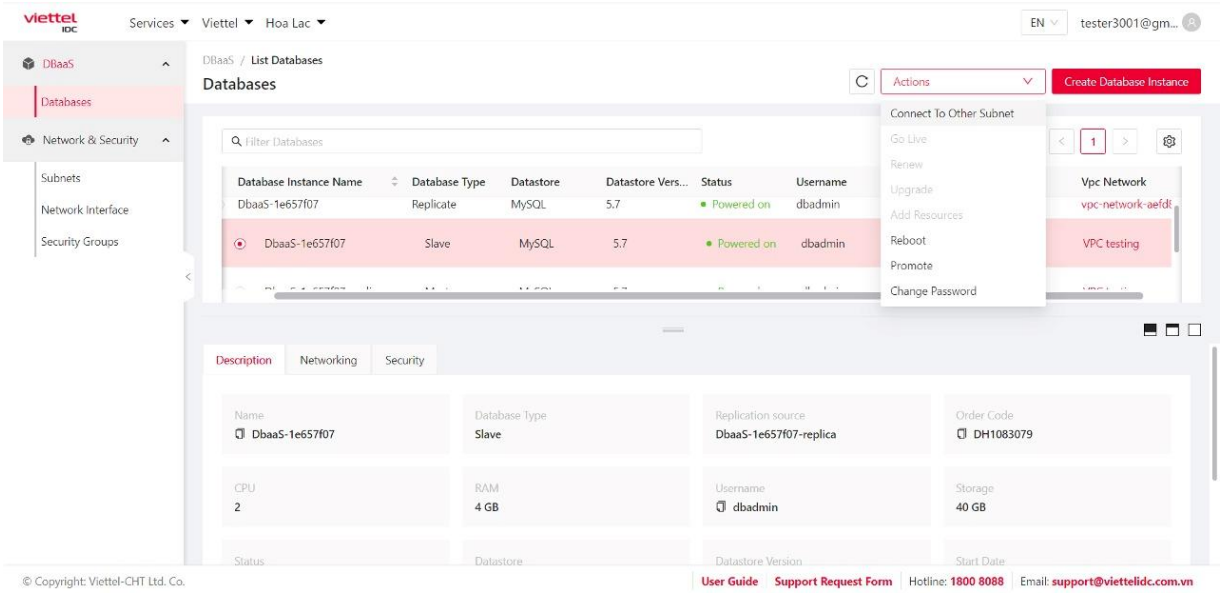
When initializing the vDBS service to connect with vOKS, users can select the same **VPC Network** to enable seamless private connectivity between the two services. The vDBS service, upon registration, is created in a private subnet to ensure the highest level of data server security. Users should follow these steps:

Step 1: View the Subnet Information of the vOKS Service

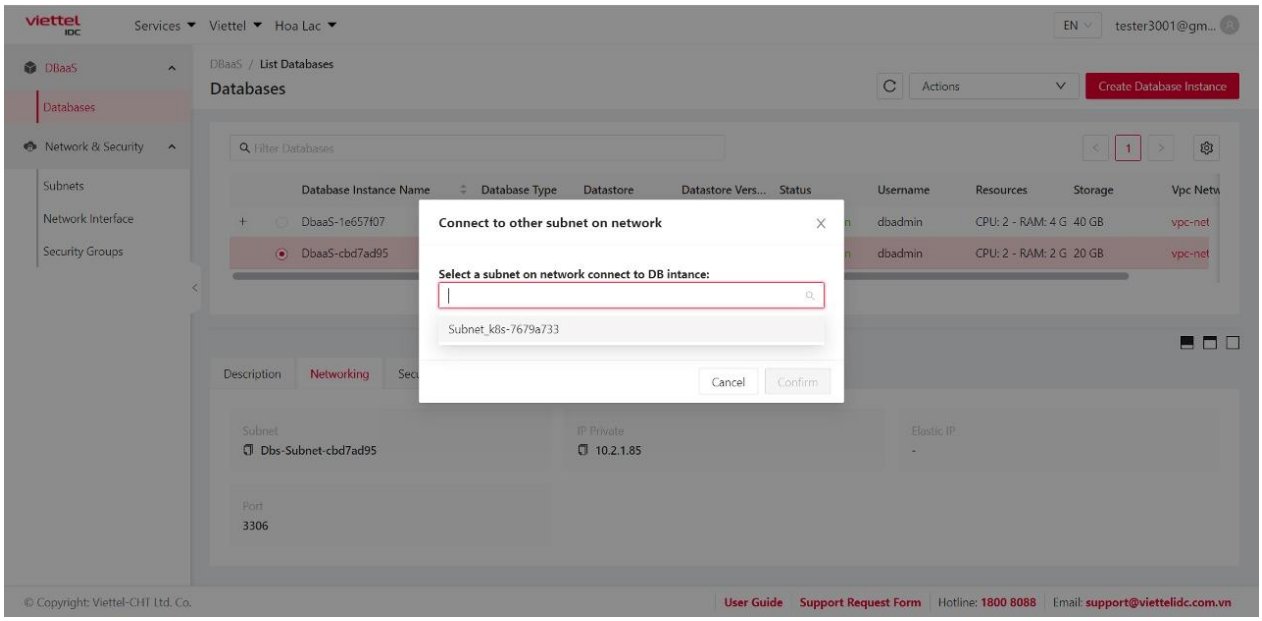
- For vOKS, users can view this in the **Network & Security** tab → **Subnet** → locate the corresponding cluster's subnet name.



Step 2: On the management interface of the vDBS service, locate the **Actions** button, click to display the menu → select **Connect To Other Subnet.**



Step 3: In the subnet selection window, choose the subnet of the vOKS service you want to connect to the vDBS service, then click **Update**.



When this happens, the system will create a Network Interface within the subnet range of the vOKS service that needs to connect to the vDBS service's database server. This Network Interface will be attached to the DB Instance, and Security Group rules will be created to allow the vOKS subnet to connect via the database's port.

6.4 CNI

6.4.1 Concept

CNI (Container Network Interface) is a standard toolkit that provides network connectivity for containers in a Kubernetes cluster. Simply put, CNI is an abstraction layer that enables Kubernetes to manage and configure the network for Pods (a group of containers sharing the same network) flexibly and efficiently.

6.4.2 Popular CNI Plugins

Several popular plugins are commonly used. Viettel IDC provides the following:

- **Calico:**

- Calico is a networking and security solution that allows workloads on Kubernetes, as well as non-Kubernetes or legacy workloads, to communicate seamlessly and securely.
- Uses direct routing without overlays, relying on BGP for performance optimization.
- Supports Network Policies to control traffic between Pods.
- Recommended for scenarios where ultra-high performance is not required.

- **Cilium:**
  - Cilium is open-source software designed to secure network connections between application services deployed on Linux container management platforms like Docker and Kubernetes transparently.
  - Leverages eBPF to optimize data transmission and security.
  - Simple and does not require extremely high performance, but supports advanced monitoring needs with tools like **Hubble**.

6.5. Configure Viettel IDC's DNS

6.5.1. Introduction

This document guides you on how to configure additional DNS on the Kubernetes cluster provided by Viettel IDC's vOKS service. Configuring additional Viettel IDC DNS helps the system resolve domain names quickly and accurately, ensuring stable connections to external services.

6.5.2. Benefits of using DNS from Viettel IDC

By default, CoreDNS uses Google's public DNS servers (8.8.8.8 and 8.8.4.4). However, using international DNS may encounter issues:

• Dependent on international transmission lines, affecting query speed during cable breaks or network congestion.

• Higher latency compared to domestic DNS due to traversing more network routes.

Viettel IDC provides DNS (115.84.181.20, 115.84.177.20) to ensure speed and stability:

• Increase domestic DNS query performance.

• Reduce latency by using closer DNS servers.

• Mitigate international network issues, ensuring smooth system operation.

6.5.3. Guide

Bước 1: Create file yaml Config Map mới config-coredns.yaml below:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns
  namespace: kube-system
data:
  Corefile: |
    .:53 {
        errors
        health {
          lameduck 5s
        }
        ready
        kubernetes cluster.local in-addr.arpa ip6.arpa {
          pods insecure
          fallthrough in-addr.arpa ip6.arpa
          ttl 30
        }
        prometheus :9153
        forward . 115.84.181.20 115.84.177.20 /etc/resolv.conf {
          max_concurrent 1000
        }
        cache 30
        loop
        reload
        loadbalance
    }
```

Note: Add Viettel IDC's DNS (115.84.181.20, 115.84.177.20) to the forward configuration of coredns.

Bước 2: Update new configuration

Use command: kubectl apply -f config-coredns.yaml

Bước 3: Restart pods coredns to receive new configuration

Use command: kubectl rollout restart deployment coredns -n kube-system

7. Deploying Applications on the Cluster

7.1 Setting Up the User-Side Control Environment

The Kubernetes command-line tool, **kubectl**, allows users to execute commands to control Kubernetes clusters. **kubectl** supports deploying applications, monitoring and managing resources, and viewing system logs.

To use it, users can install **kubectl** on various platforms. Detailed installation instructions for specific environments are provided below:

*** Installing kubectl on Windows**

- **Step 1:** Download **kubectl** from the installation link:
  https://storage.googleapis.com/kubernetes-release/release/v1.19.0/bin/windows/amd64/kubectl.exe
  To check the latest version of kubectl, visit:
  https://storage.googleapis.com/kubernetes-release/release/stable.txt

- **Step 2:** Add the **kubectl** file path to **PATH**:
Create a .kube folder in the current User directory.

Note: The logged-in user must have local administrator rights.

- **Method 1:** Log in to the Windows account with local administrator rights. Press Windows + R, type %userprofile%, and create the .kube folder here.

- Method 2: Log in to the Windows account with local administrator rights, open **Command Prompt**, and run the command: mkdir .kube

  Place the kubectl.exe file in the path: %userprofile%/.kube.

- Step 3: Verify the installed kubectl version:

```
kubectl version --client
kubectl cluster-info
```

```
PS C:\Users\QuangDA\.kube> .\kubectl.exe cluster-info
Kubernetes master is running at https://171.244.49.28:6443
KubeDNS is running at https://171.244.49.28:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
PS C:\Users\QuangDA\.kube> .\kubectl.exe get nodes
NAME        STATUS   ROLES    AGE     VERSION
mstr-obcy   Ready    master   3h56m   v1.17.2
node-gd6c   Ready    <none>   3h47m   v1.17.2
node-i8vk   Ready    <none>   3h46m   v1.17.2
node-ldd5   Ready    <none>   3h48m   v1.17.2
PS C:\Users\QuangDA\.kube>
```

- Step 4: Verify Successful Connection

Execute kubectl commands to interact with the Kubernetes cluster.

*** Installing kubectl on CentOS Linux**

- Step 1: Run the following command:

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_ 64
enabled=1 gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
```

```
EOF
```

- **Step 2:** Install the kubectl package:

```
yum install -y kubectl
```

- **Step 3:** Create a directory to store the configuration file:

```
mkdir ~/.kube
```

- **Step 4:** Verify the configuration file on the server

```
kubectl cluster-info
Kết quả:
Kubernetes master is running at https://master_ip:6443
KubeDNS is running at https://master_ip/api/v1/namespaces/kube-system/services/kube- dns:dns/proxy
```

- **Step 5:** Successful connection.

Begin executing commands using kubectl.

*** Installing kubectl on macOS**

- **Step 1:** Download the latest version using the **curl** command.

```
curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/darwin/amd64/kubectl"
```

To download a specific version, replace the variable $(curl –s https://storage.googleapis.com/kubernetes-release/release/stable.txt) with the desired version. Example: Download version v1.17.0 on macOS

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.17.0/bin/darwin/amd64/kubectl
```

- **Step 2:** Assign execute permissions to the kubectl binary.

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

- **Step 3:** Add the binary to the PATH environment variable.

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

- **Step 4:** Verify that the installed version is the latest one.

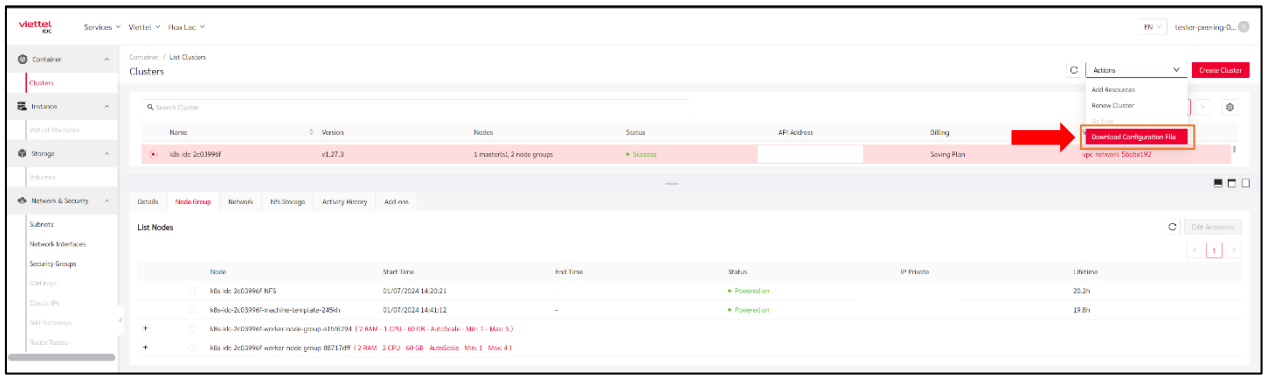
```
kubectl version
```

7.2 Instructions for Downloading the Kube-config File & Using the Kubernetes Dashboard

7.2.1 Downloading the Kube-config File

On the detailed information screen for a vOKS service package, the **"Package Actions"** section provides the following functions:

- **Download Kube-config File:** Download the file to access and configure settings for each Kubernetes (K8s) cluster;



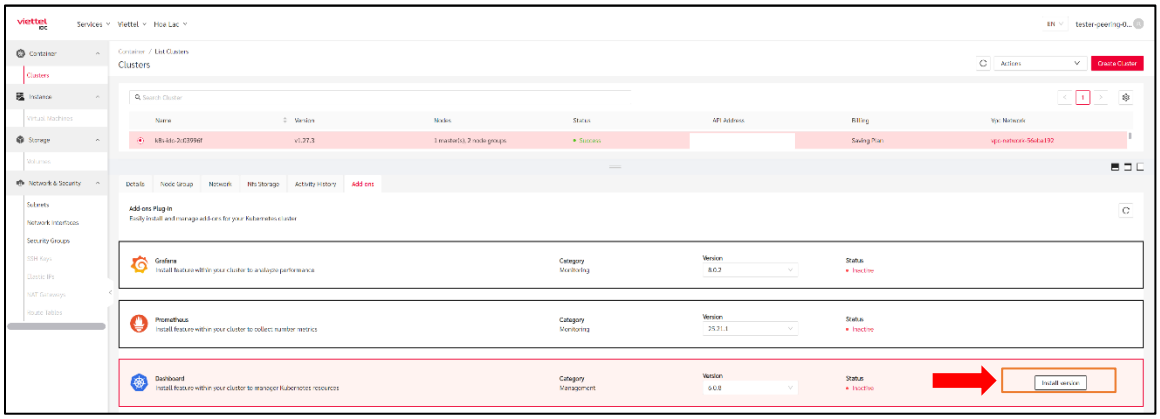
Note:

After downloading the file to your personal computer, rename it from download.config to config

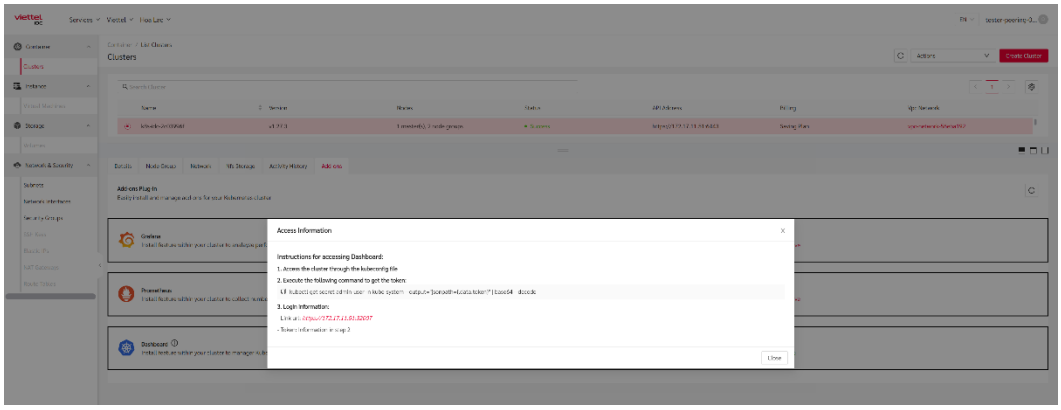
Place the config file in the appropriate environment-specific directory as follow: %userprofile%\.kube\config

7.2.2 Using the Kubernetes Dashboard

- **Access Kubernetes Dashboard:** Users can access the Kubernetes-native technical management interface for the cluster by installing the desired version;



After the Kubernetes Dashboard is installed, users can view access information and be redirected to the Kubernetes management page for the installed cluster. The versions of the Dashboard will be continuously updated in the Version column.



- To access the Dashboard, users need to retrieve a token by following these steps:

Step 1: Use kubectl to retrieve the token

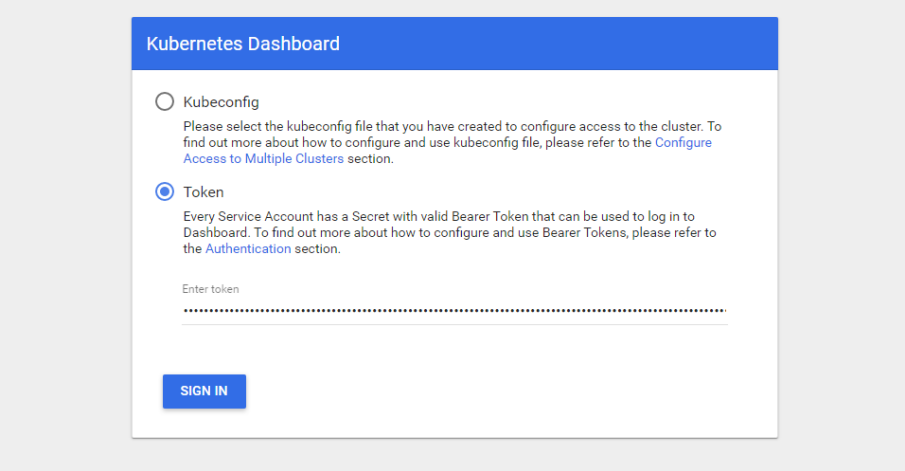
Copy the downloaded kubeconfig file from the instructions in the Download Kube-config File section and file kubectl.exe into the .kube directory created earlier.

Step 2: Retrieve the token using the appropriate command

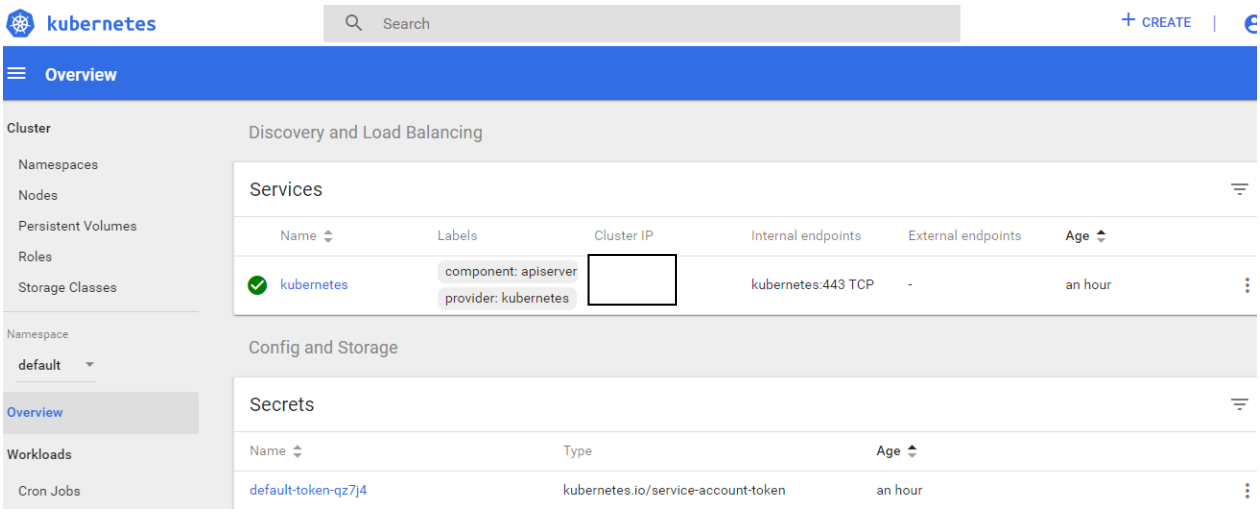
```
kubectl describe secrets admin-user -n kube-system
```



Step 3: Use the token retrieved in Step 2 to access the Kubernetes Dashboard.



Step 4: Log in to the Kubernetes Dashboard.



7.3 Tính năng Horizontal Pod Autoscaling

7.3.1 Overview

Horizontal Pod Autoscaling (HPA) is the process of increasing or decreasing the number of Pods in a node to a predefined count (horizontal scaling). This process is triggered by one or more events.

For example: On a Pod, when CPU usage exceeds 70% or Memory usage exceeds 80%, new Pods will be automatically created.

7.3.2 Mechanism

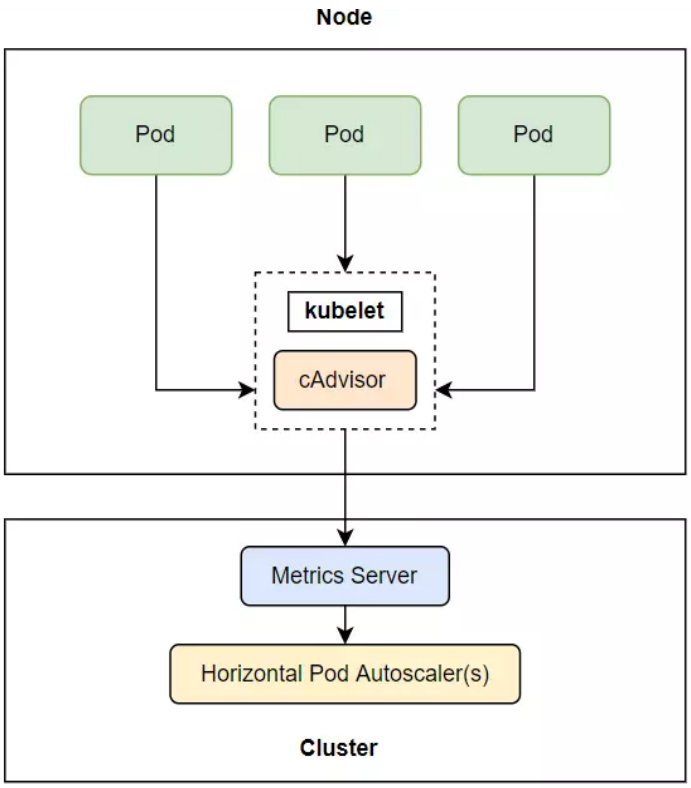
By default, HPA retrieves information from the **metrics-server** every 15 seconds to compare conditions.

- If the value of currentMetricValue / desiredMetricValue is less than or approximately equal to 1 (the difference is negligible), it continues listening to the **metrics-server**.
- If the value is less than 1, it will continue listening for 300 seconds. If the value remains below 1 during this time, it will proceed to scale down gradually.

Users can customize HPA scaling behavior (both scale up and scale down) through the **config behavior** (applicable for Kubernetes version 1.23 and above). Detailed instructions can be found here.

Additionally, users can configure HPA to scale up/down based on custom metrics or various types of metrics (applicable for Kubernetes version 1.23 and above). Detailed instructions can be found here.

To obtain monitoring data (CPU, Memory) as a basis for scaling Pods, users must first install the metrics-server. Instructions for installing the metrics-server can be found here:



7.3.3 Instructions for Configuring HPA Metrics

Method 1: Using a YAML File

Step 1: Run the command to deploy the **metrics-server**.

```
Kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

Step 2: Check the status of the metrics-server Pod using the appropriate command:

```
kubectl get deployment metrics-server -n kube-system
```

Step 3: Create the Horizontal Pod Autoscaler (HPA) configuration.

- **Condition 1:** Create the service-hpa.yml file with the condition:
    - **RAM > 80%**
    - Minimum replicas: **2**
    - Maximum replicas: **5**

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: test-app-autoscale
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: test-app-autoscale
  minReplicas: 2
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: memory
      target:
      type: Utilization
      averageUtilization: 80
```

=> Apply the service-hpa.yml file using the designated command:

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

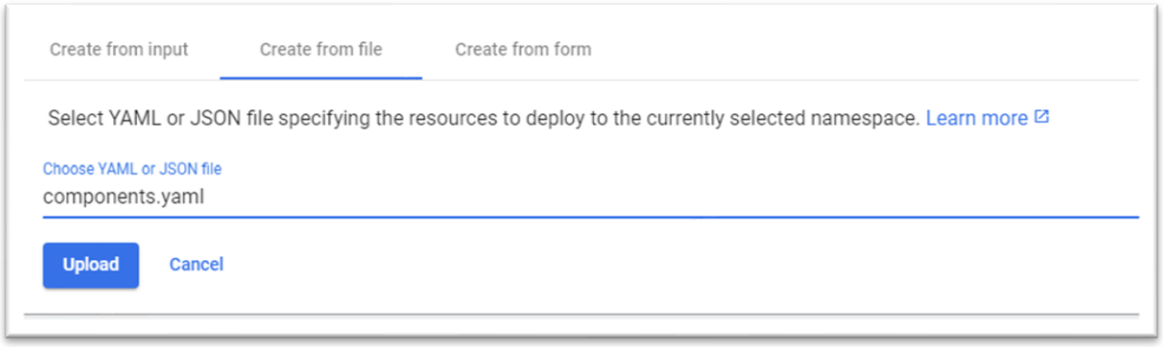
- **Condition 2:** Create the service-hpa.yml file with the condition:
    - **CPU > 70%**
    - Minimum replicas: **2**
    - Maximum replicas: **5**

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
 name: test-app-autoscale
spec:
 scaleTargetRef:
  apiVersion: apps/v1
  kind: Deployment
  name: test-app-autoscale
 minReplicas: 2
 maxReplicas: 5
 metrics:
 targetCPUUtilizationPercentage: 70
```

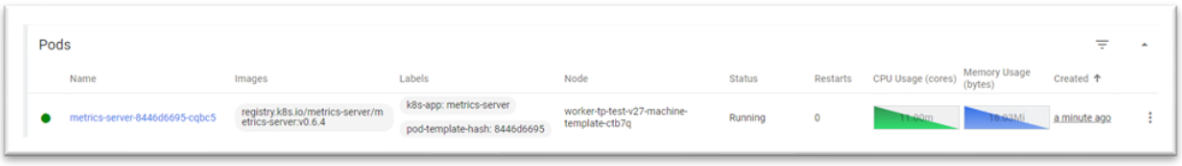
Method 2: Using Kubernetes Dashboard

Step 1: Deploy the **metrics-server**

- Download the **metrics-server** file to your system here
- After downloading, proceed to install the **metrics-server**

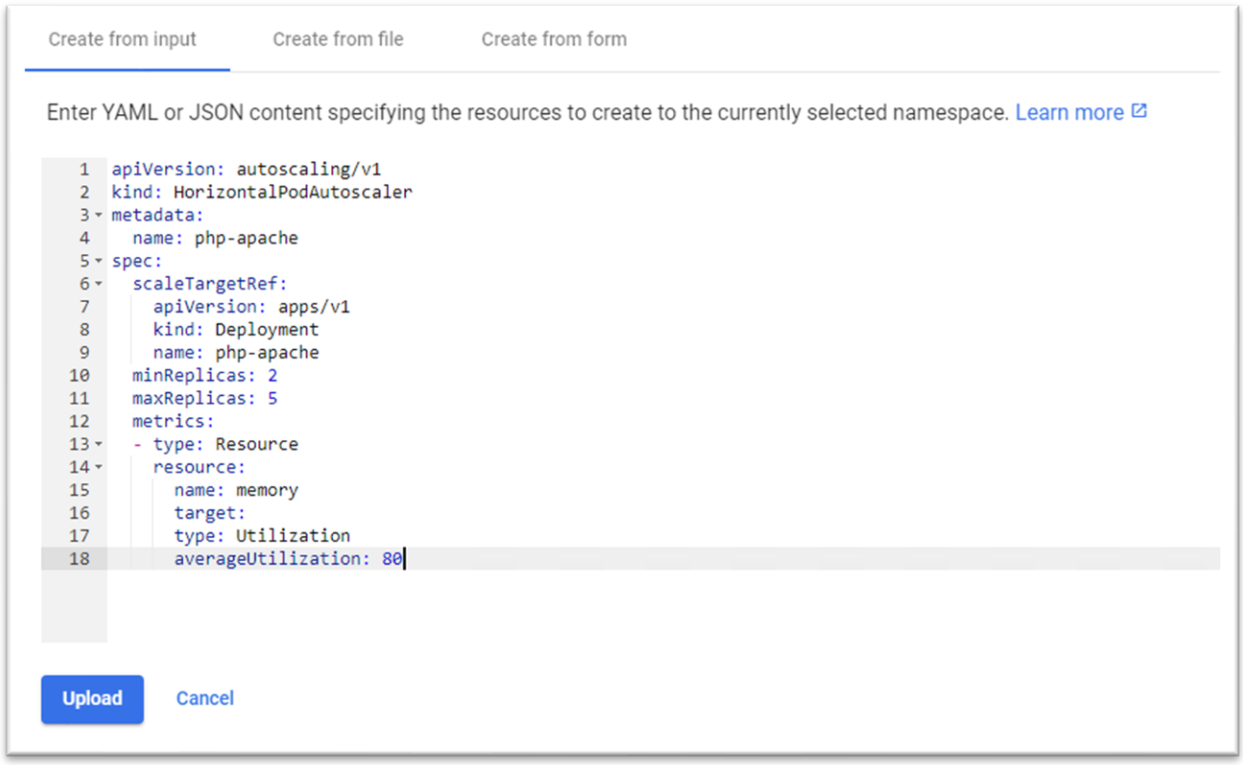


Step 2: Check the status of the **metrics-server** Pod to ensure it is running.

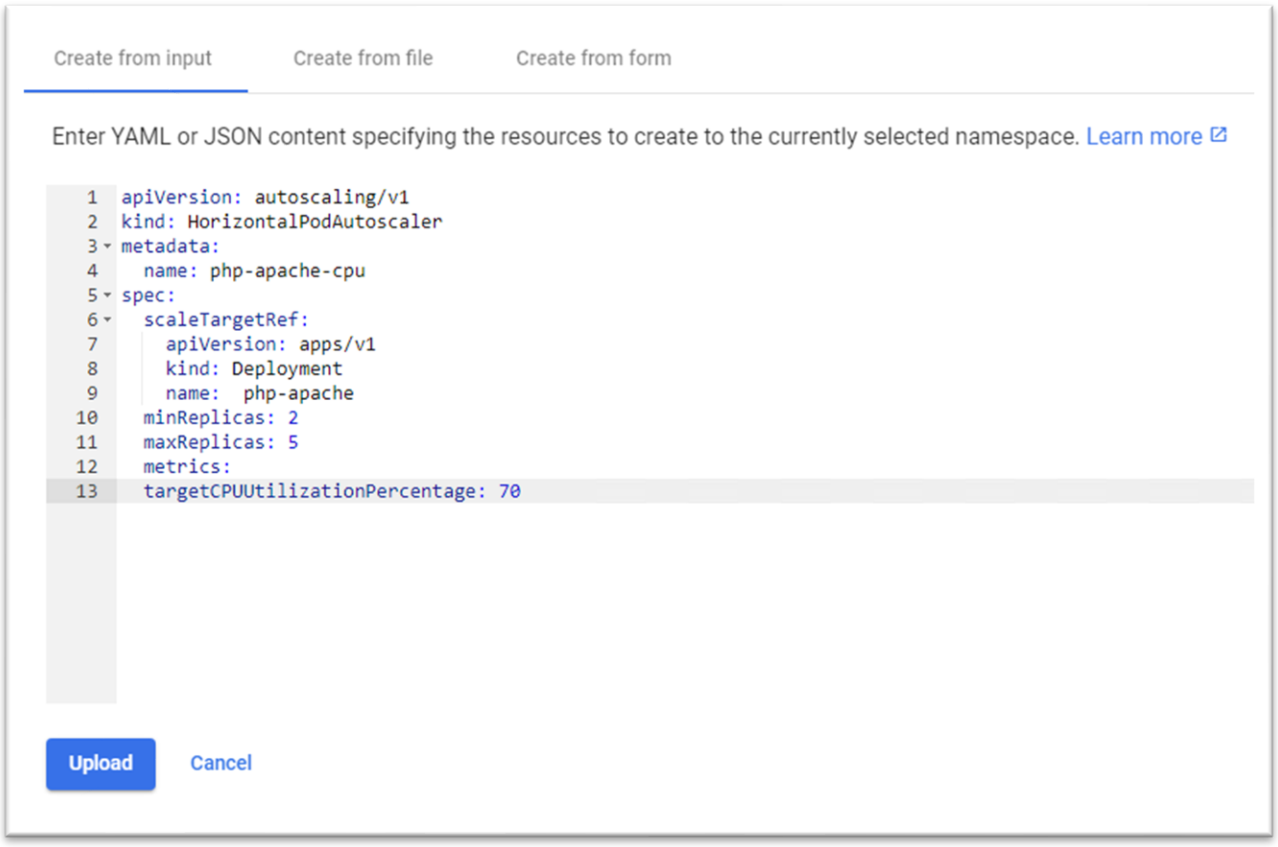


Step 3: Configure the HPA.

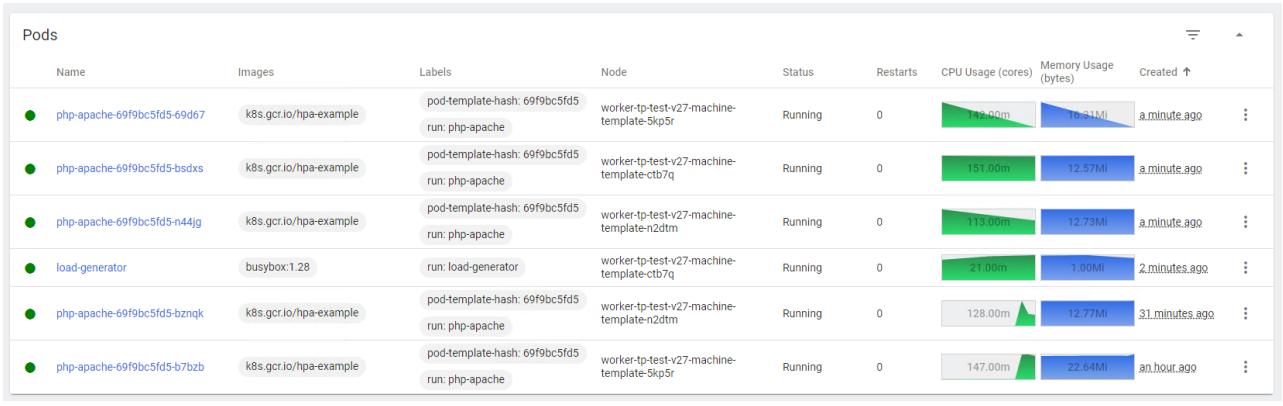
(+) Condition RAM > 80%



(+) Condition CPU > 70%

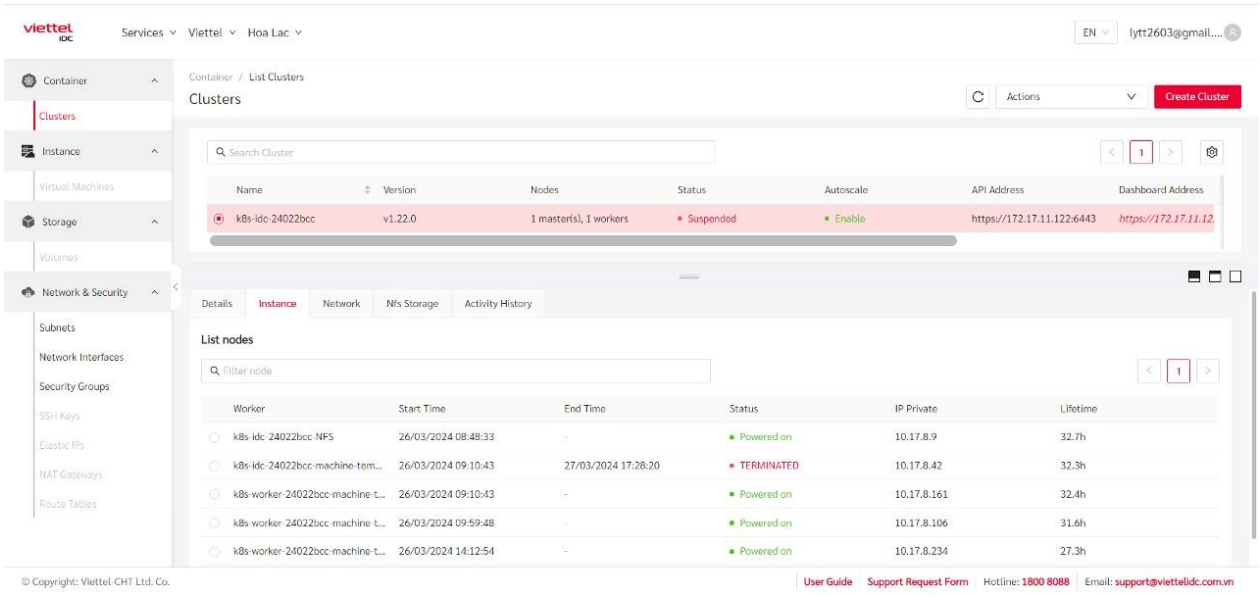


Step 4. Monitor the results of Pod scaling when the condition are met



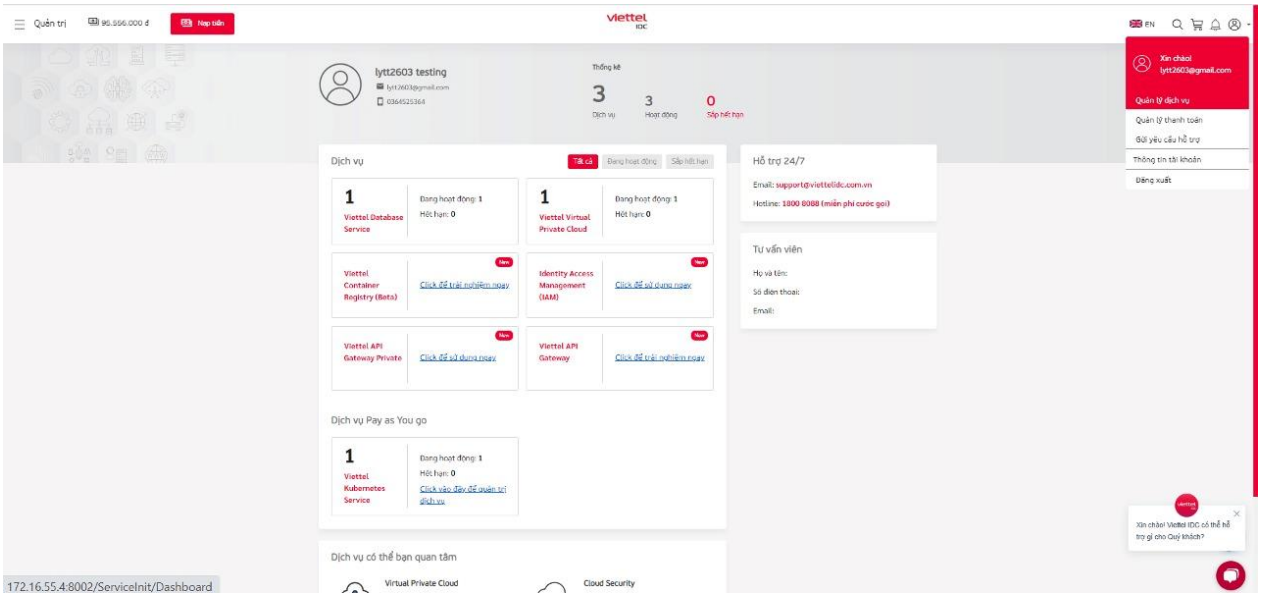
8. Managing Usage Costs

8.1 Checking the Uptime of Worker Nodes



Users can check the uptime and the creation or suspension history of Worker Nodes by selecting the cluster → navigating to the Instances tab to view the Lifetime column for each Worker Node.

8.2 Viewing Billing History and Payment History



Users can view billing history on the website by navigating to Service Management → selecting the Viettel Open Kubernetes Service → viewing the Billing History and Payment History, as follows:

Billing History:

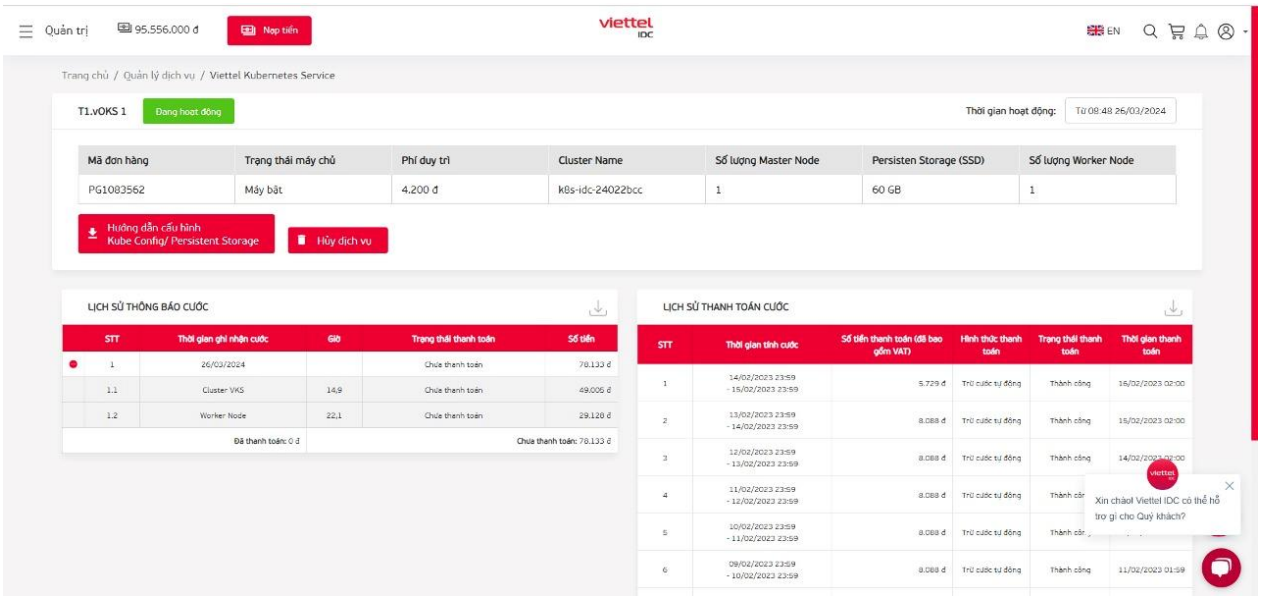
This includes daily records of usage fees for the Cluster and Worker Node Autoscale. Billing is calculated daily until the user no longer needs the service.

- Click the button ⊕ to expand and view detailed billing information, which includes two parts: **Cluster:** Covers costs for components such as Master Node, Persistent Storage, IP, and Bandwidth. **Worker Node:** Displays the usage fees based on the runtime of Worker Nodes. Users can cross-check and **verify the runtime with the uptime information on the Instances screen.**
- The total cost is calculated as the hourly usage fee of the cluster and Worker Nodes, multiplied by the runtime hours. The daily total cost for the vOKS PAYG service is the sum of the cluster and Worker Node fees.

Payment History:

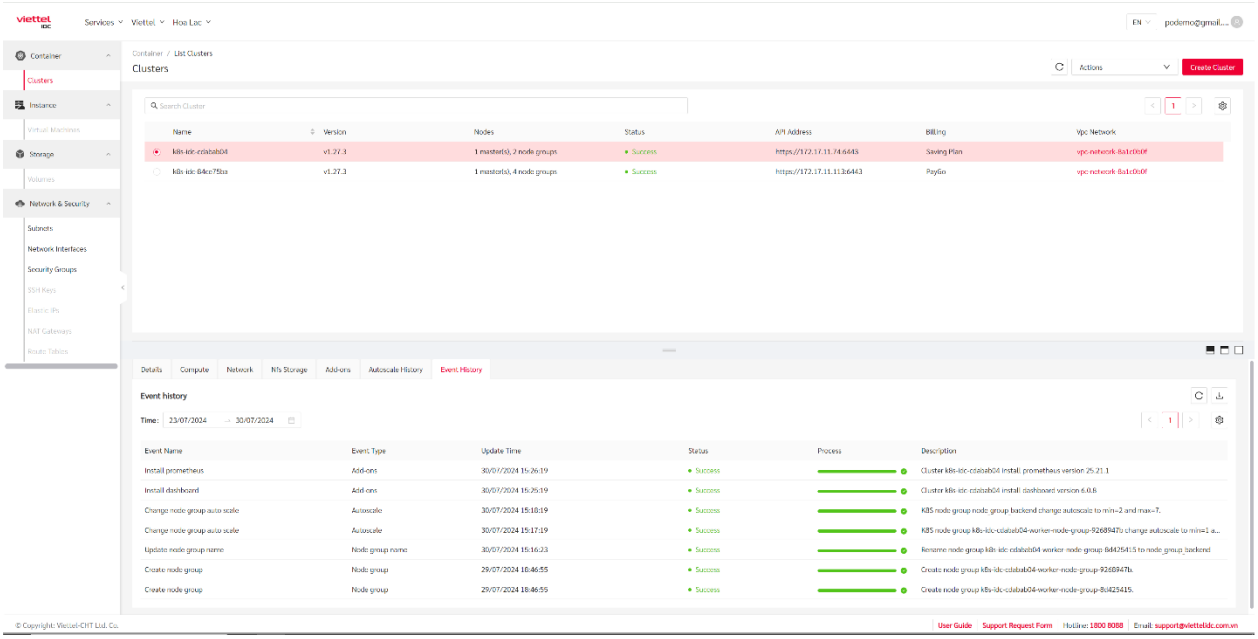
Users must pay usage fees daily and maintain sufficient funds to ensure uninterrupted service operation.

- If the service fee is overdue for one day, the service may be interrupted. Therefore, Viettel IDC recommends that users monitor daily vOKS service fees to control and forecast usage costs, avoiding system disruptions.



8.3 Viewing and Downloading Cluster Action History

Step 1: On the **List Clusters** screen, select the cluster where the Node Group was added and go to the **Event History** tab.



Step 2: The list of actions that users can perform with the cluster will display events from the past 7 days, including the following types of actions:

- Adding a Node Group, including purchasing additional Node Groups for Saving Plan clusters and creating new Node Groups for PAYG clusters.
- Editing the autoscale configuration of a Node Group.
- Viewing the history of installing and uninstalling add-ons such as Dashboard, Grafana, Prometheus for the cluster

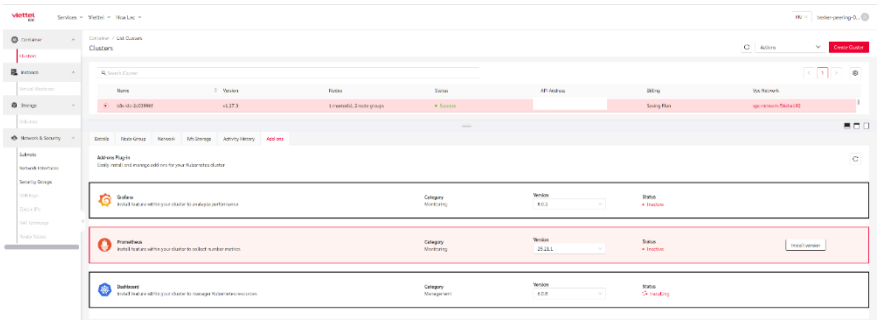
Step 3: Users can download the list of events that occurred with the cluster to monitor, manage, and control service resources.

9. Monitor the Cluster

9.1 Using Prometheus, Grafana

Prometheus and Grafana are open-source software solutions designed to address the needs of data collection and visualization for services, supporting monitoring and

observability in Kubernetes clusters. Viettel IDC supports users in installing these tools directly from the service management interface.

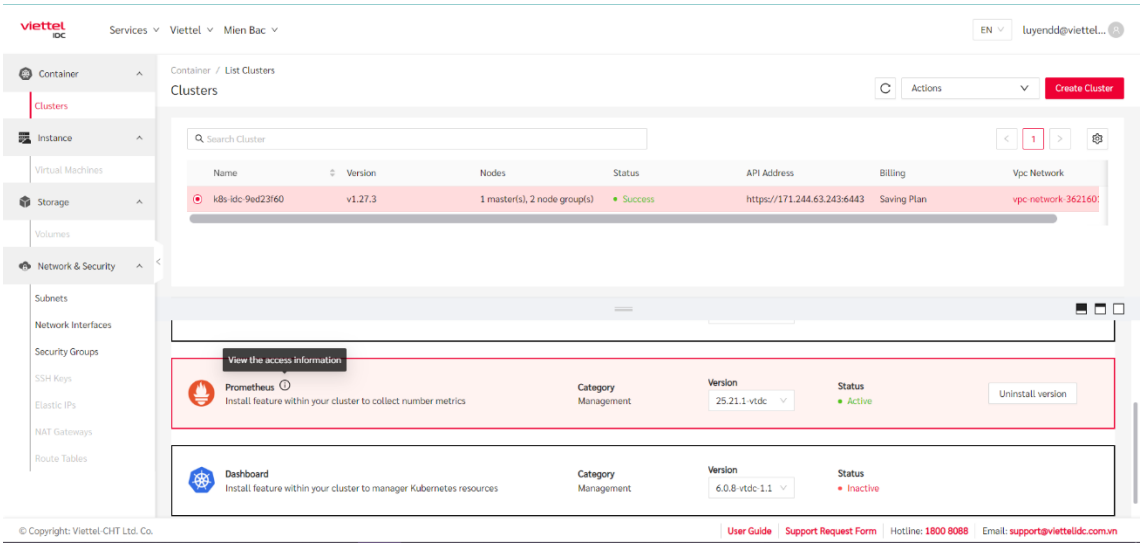


9.1.1 Prometheus

Prometheus is an open-source monitoring software that is becoming increasingly popular. It is highly powerful and particularly well-suited for monitoring services on Kubernetes, as it comes with many pre-built monitoring templates for open-source services, making deployment and configuration both quick and efficient.

In addition to Prometheus (acting as the monitoring system), an **Alert Manager** is also required to handle alerts, enabling administrators to detect issues as early as possible.

Users can install Prometheus and access its management interface directly from the service management page by clicking the designated icon ⓘ

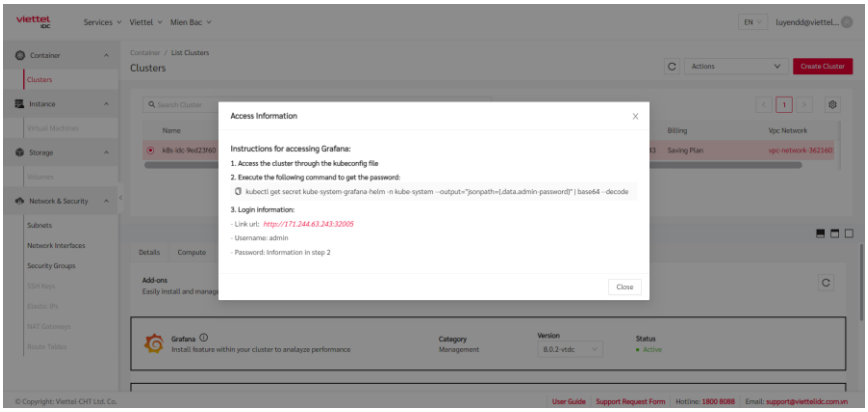


9.1.2 Grafana

Grafana is an open-source software used for analyzing and visually representing data. It allows for highly customizable dashboards that display data effectively, making it an invaluable tool for monitoring and analyzing time-series data. Grafana can source data from Prometheus, Graphite, or Elasticsearch, among others.

Users can also install and access Grafana information directly from the management interface by clicking the designated icon, similar to Prometheus.

Users can also install and view Grafana information directly from the management interface by clicking the designated icon ⓘ, similar to Prometheus.



9.2 Using Viettel Cloud Watch (coming soon)

10. Frequently Asked Questions (FAQ)

10.1 What is the basic differences between Container, Docker and Kubernetes?

Container is a standardized unit of software that contains the source code and dependencies needed to run an application or service.

Docker is a platform containing tools for “containerizing” an application, automating the deployment of a container application from one environment to another, ensuring that the application works stably and consistently.

Kubernetes is not responsible for creating containers by itself, but its key purpose is orchestrating, scheduling, and managing the state of containers when scaling deployment with the requirement to coordinate multiple containers working together to distribute distribute individual services on a physical or virtualized infrastructure.

10.2 What kind of service vOKS provide to customers?

vOKS provides customers with Kubernetes clusters with a choice of 01 Master Node or 03 Master Nodes with Loadbalancer to achieve high availability (HA). Also, the number of Worker Nodes can expand unlimitedly.

* Master Node: a VM containing all the logic components to coordinate the activities of the Worker Nodes and define, deploy, and manage the lifecycle of the containers running on the Worker Node.

* Worker Node: Containerized applications will deploy on Worker Nodes. In each vOKS service package, Viettel IDC provides at least 03 Worker Nodes in order to improve fault tolerance and availability when deploying applications.

* NFS Node - Persistent Storage: a NFS Server (Network File System) used as a protocol to connect containers running in Worker Node to an independent data storage area (Persistent Storage).

10.3 What is the difference between Node Storage and Persistent Storage?

Node Storage has a fixed capacity of 50 GB (SSD), is used to store data (cache form) of the container during operation. This data is only saved temporarily and will be deleted when the user restarts/reschedules the containers.

On the other hand, the container's data during operation is stored on its own storage area (Persistent Storage) and will be retained when the containers are restarted/rescheduled; this data can be transferred between containers.

10.4 Is vOKS compatible with other containerized applications running on Kubernetes platform of other Cloud providers?

Customers can easily use all existing plugins or tools developed and contributed by the Kubernetes community. Also, applications running on any standard Kubernetes environment on other Cloud infrastructure (AWS, Microsoft Azure, etc.) or On-Premise Data Center are compatible and easily migrated to vOKS.