

VKS

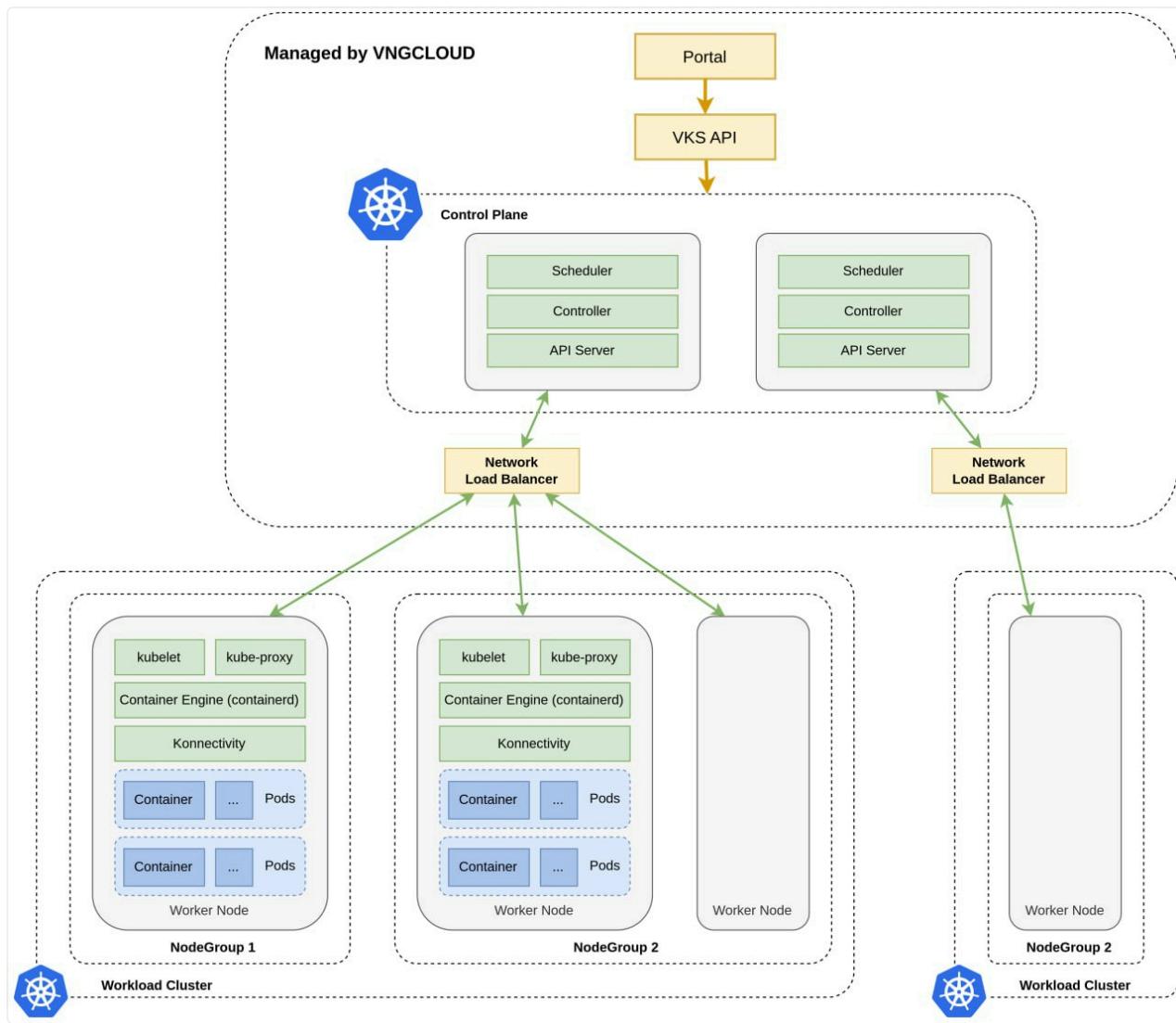
VKS (VNGCloud Kubernetes Service) là một dịch vụ được quản lý trên VNGCloud giúp bạn đơn giản hóa quá trình triển khai và quản lý các ứng dụng dựa trên container. Kubernetes là một nền tảng mã nguồn mở được phát triển bởi Google, được sử dụng rộng rãi để quản lý và triển khai các ứng dụng container trên môi trường phân tán.

VKS Demo Video - Giải Pháp Quản Lý Kubernetes Toàn Diện Của VNG Cloud



VKS là gì?

VKS (VNGCloud Kubernetes Service) là một dịch vụ được quản lý trên VNGCloud giúp bạn đơn giản hóa quá trình triển khai và quản lý các ứng dụng dựa trên container. Kubernetes là một nền tảng mã nguồn mở được phát triển bởi Google, được sử dụng rộng rãi để quản lý và triển khai các ứng dụng container trên môi trường phân tán.



Những điểm nổi bật của VKS

- Quản lý Control Plane hoàn toàn tự động (Fully Managed control plane):** VKS sẽ giải phóng bạn khỏi gánh nặng quản lý Control Plane của Kubernetes, giúp bạn tập trung vào việc phát triển ứng dụng.

- **Hỗ trợ các phiên bản Kubernetes mới nhất:** VKS luôn cập nhật những phiên bản Kubernetes mới nhất (minor version từ 1.27, 1.28, 1.29) để đảm bảo bạn luôn tận dụng được những tính năng tiên tiến nhất.
- **Kubernetes Networking:** VKS tích hợp Calico CNI, mang lại tính hiệu quả và bảo mật cao.
- **Upgrade seamlessly:** VKS hỗ trợ nâng cấp giữa các phiên bản Kubernetes một cách dễ dàng và nhanh chóng, giúp bạn luôn cập nhật những cải tiến mới nhất.
- **Scaling & Healing Automatically:** VKS tự động mở rộng Node group khi cần thiết và tự động sửa lỗi khi node gặp vấn đề, giúp bạn tiết kiệm thời gian và công sức quản lý.
- **Giảm chi phí và nâng cao độ tin cậy:** VKS triển khai Control Plane của Kubernetes ở chế độ sẵn sàng cao và hoàn toàn miễn phí, giúp bạn tiết kiệm chi phí và nâng cao độ tin cậy cho hệ thống.
- **Tích hợp Blockstore Native (Container Storage Interface - CSI):** VKS cho phép bạn quản lý Blockstore thông qua YAML của Kubernetes, cung cấp lưu trữ bền vững cho container và hỗ trợ các tính năng quan trọng như thay đổi kích thước, thay đổi IOPS và snapshot volume.
- **Tích hợp Load Balancer (Network Load Balancer, Application Load Balancer) thông qua các driver được tích hợp sẵn như VNGCloud Controller Manager, VNGCloud Ingress Controller:** VKS cung cấp khả năng quản lý NLB/ALB thông qua YAML của Kubernetes, giúp bạn dễ dàng expose Service trong Kubernetes ra bên ngoài.
- **Nâng cao bảo mật:** VKS cho phép bạn tạo Private Node Group với chỉ Private IP và kiểm soát quyền truy cập vào cluster thông qua tính năng Whitelist IP, đảm bảo an toàn cho hệ thống của bạn.

Ngoài ra, VKS còn có các ưu điểm sau:

- Dễ sử dụng: VKS cung cấp giao diện đơn giản và dễ sử dụng.
- Chi phí hợp lý: VKS cung cấp mức giá cạnh tranh cho các dịch vụ của mình.

Region

Hiện tại, trên VKS, chúng tôi đang cung cấp cho bạn 2 cơ sở hạ tầng riêng biệt được đặt tại Hà Nội và Hồ Chí Minh. Bạn có thể lựa chọn sử dụng VKS trên mỗi region tùy thuộc vào vị trí và nhu cầu thực tế của bạn. Đối với 2 farm HCM03, HAN01, các thông số cụ thể cho mỗi region được chúng tôi cung cấp như sau:

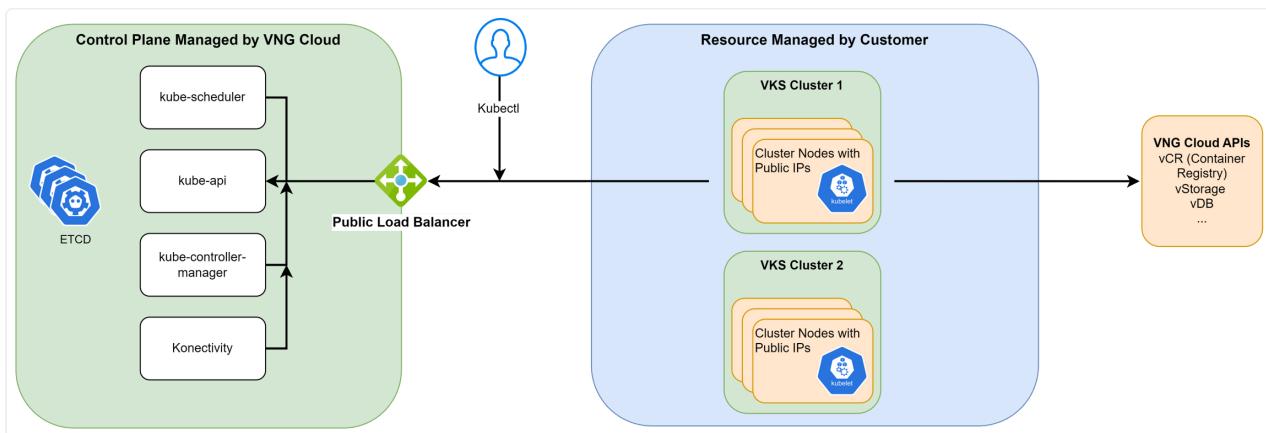
Farm	Domain
HAN01	vnsite.com
HCM03	vnsite.com

HCM03	https://vks.console.vngcloud.vn
HAN01	https://vks-han-1.console.vngcloud.vn

Mô hình hoạt động

Bên dưới là các concepts hiện tại VKS đang cung cấp cho bạn:

1. Public Cluster



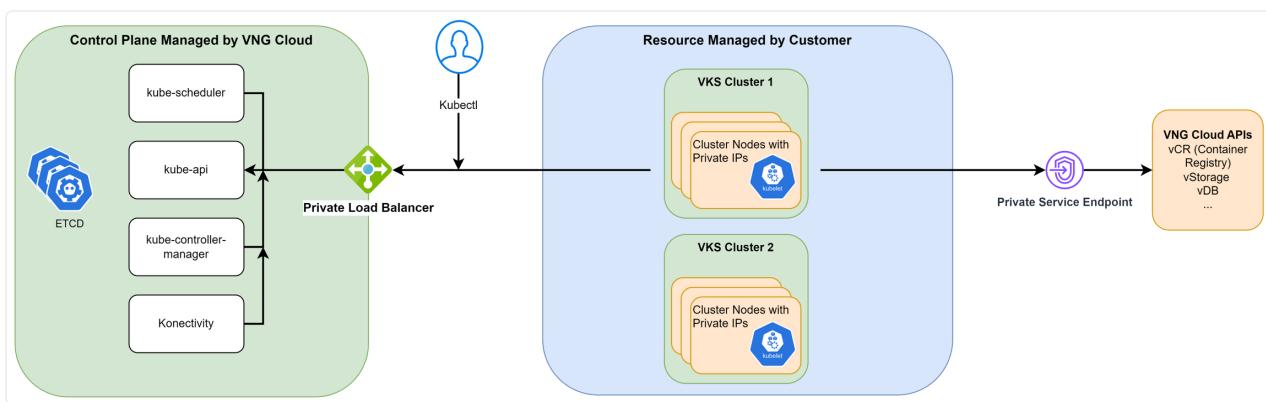
Khi bạn khởi tạo một **Public Cluster** với **Public Node Group**, hệ thống VKS sẽ:

- Tạo VM có Floating IP (tức có IP Public). Lúc này các VM (Node) này có thể join trực tiếp vào cụm K8S thông qua Public IP này. Bằng cách sử dụng Public Cluster và Public Node Group, bạn có thể dễ dàng tạo các cụm Kubernetes và thực hiện expose service mà không cần sử dụng Load Balancer. Việc này sẽ góp phần tiết kiệm chi phí cho cụm của bạn.

Khi bạn khởi tạo một **Public Cluster** với **Private Node Group**, hệ thống VKS sẽ:

- Tạo VM không có Floating IP (tức không có IP Public). Lúc này các VM (Node) này không thể join trực tiếp vào cụm K8S. Để các VM này có thể join vào cụm K8S, bạn cần phải sử dụng một NAT Gateway (**NATGW**). **NATGW** hoạt động như một trạm chuyển tiếp, cho phép các VM kết nối với cụm K8S mà không cần IP Public. Với VNG Cloud, chúng tôi khuyến cáo bạn sử dụng Pfsense hoặc Palo Alto như một NATGW cho Cluster của bạn. Pfsense sẽ giúp bạn quản lý lưu lượng mạng đến và đi (inbound và outbound traffic) một cách hiệu quả, đảm bảo an ninh mạng và quản lý truy cập. Bên cạnh đó, việc sử dụng Private Node Group sẽ giúp bạn kiểm soát các ứng dụng trong cụm được bảo mật hơn, cụ thể bạn có thể thực hiện giới hạn quyền truy cập control plane thông qua tính năng Whitelist IP.

2. Private Cluster



Khi bạn khởi tạo một **Private Cluster** với **Public/ Private Node Group**, hệ thống VKS sẽ:

- Để nâng cao bảo mật cho cluster của bạn, chúng tôi đã cho ra mắt mô hình private cluster. Tính năng Private Cluster giúp cho cụm K8S của bạn được bảo mật nhất có thể, mọi kết nối hoàn toàn là private từ kết nối giữa nodes tới control plane, kết nối từ client tới control plane, hay kết nối từ nodes tới các sản phẩm dịch vụ khác trong VNG Cloud như: vStorage, vCR, vMonitor, VNGCloud APIs,...Private Cluster là lựa chọn lý tưởng cho **các dịch vụ yêu cầu kiểm soát truy cập chặt chẽ, đảm bảo tuân thủ các quy định về bảo mật và quyền riêng tư dữ liệu**.

3. So sánh giữa việc sử dụng Public Cluster và Private Cluster

Dưới đây là bảng so sánh giữa việc tạo và sử dụng Public Cluster và Private Cluster trên hệ thống VKS:

Tiêu chí	Public Cluster	Private Cluster
Kết nối	Sử dụng địa chỉ Public IP để giao tiếp giữa nodes và control plane, giữa client và control plane, giữa nodes và các dịch vụ khác trong VNG Cloud.	Sử dụng địa chỉ Private IP để giao tiếp giữa nodes và control plane, giữa client và control plane, giữa nodes và các dịch vụ khác trong VNG Cloud.
Bảo mật	Bảo mật trung bình do các kết nối sử dụng Public IP.	Bảo mật cao hơn với tất cả kết nối đều private và giới hạn truy cập.
Quản lý truy cập	Khó kiểm soát hơn, có thể quản lý truy cập thông qua tính năng Whitelist	Kiểm soát truy cập chặt chẽ, mọi kết nối đều nằm trong mạng private của VNG Cloud,

		từ đó giảm thiểu nguy cơ từ các cuộc tấn công mạng từ bên ngoài.
Khả năng mở rộng (AutoScaling)	Dễ dàng mở rộng thông qua tính năng Auto Scaling .	Dễ dàng mở rộng thông qua tính năng Auto Scaling .
Khả năng tự hồi phục (AutoHealing)	Tự động phát hiện lỗi và khởi động lại node (Auto Healing)	Tự động phát hiện lỗi và khởi động lại node (Auto Healing)
Khả năng truy cập từ bên ngoài	Dễ dàng truy cập từ bất kỳ đâu với internet.	Truy cập từ bên ngoài phải qua các giải pháp bảo mật khác.
Cấu hình và triển khai	Đơn giản hơn do không yêu cầu thiết lập mạng nội bộ.	Phức tạp hơn, yêu cầu cấu hình mạng private và bảo mật.
Chi phí	Thường thấp hơn do không cần thiết lập cơ sở hạ tầng bảo mật phức tạp.	Chi phí cao hơn do yêu cầu thêm các thành phần bảo mật và quản lý. Cụ thể, khi sử dụng private cluster, bạn cần chi trả chi phí cho 4 private service endpoint được tạo tự động để kết nối tới các dịch vụ trên VNG Cloud.
Tính linh hoạt	Cao, dễ dàng thay đổi và truy cập các dịch vụ.	Linh hoạt hơn trong các ứng dụng yêu cầu bảo mật, nhưng ít linh hoạt hơn cho các ứng dụng yêu cầu truy cập từ bên ngoài.

Do đó:

- **Public Cluster:** Phù hợp cho các ứng dụng không yêu cầu bảo mật cao và cần sự linh hoạt, truy cập từ nhiều địa điểm. Dễ dàng triển khai và quản lý nhưng có rủi ro bảo mật cao hơn.
- **Private Cluster:** Phù hợp cho các ứng dụng yêu cầu bảo mật cao, tuân thủ nghiêm ngặt các quy định về bảo mật và quyền riêng tư. Mang lại kết nối ổn định và bảo mật, nhưng yêu cầu cấu hình và quản lý phức tạp hơn, cũng như chi phí cao hơn.

Thông báo và cập nhật

Release notes

Dec 5, 2024

VKS (VNGCloud Kubernetes Service) vừa ra mắt bản cập nhật mới nhất, mang đến nhiều tính năng mới cho người dùng. Dưới đây là những điểm nổi bật của bản cập nhật:

Tính năng mới:

- **Force-Upgrade, Auto-Upgrade:** Tự động nâng cấp phiên bản Kubernetes cho cluster/node group theo lịch trình hoặc khi phiên bản hiện tại sắp hết hạn.. Chi tiết tham khảo thêm tại [đây](#).
-

Oct 23, 2024

VKS (VNGCloud Kubernetes Service) vừa ra mắt bản cập nhật mới nhất, mang đến nhiều cải tiến cho người dùng. Dưới đây là những điểm nổi bật của bản cập nhật:

Cải tiến:

- **Hỗ trợ POC/ Stop POC cho Cluster:** Người dùng giờ đây có thể thực hiện POC/ Stop POC cho các tài nguyên trên VKS như Server, Volume, Load Balancer, Endpoint. Tính năng này mang đến sự linh hoạt cao cho người dùng khi muốn trải nghiệm với VKS. Chi tiết tham khảo thêm tại [đây](#).
 - **Nâng cấp Plugin VNGCloud BlockStorage CSI Driver:** Các lỗi đã được phát hiện trong các phiên bản trước đã được khắc phục, giúp hệ thống hoạt động mượt mà và tin cậy hơn.
 - **Tự do lựa chọn/ chỉnh sửa cấu hình có/ không sử dụng plugin VNGCloud Controller Manager, Plugin VNGCloud Ingress Controller trên cụm VKS hiện có:** Khả năng tùy chỉnh cấu hình plugin cho phép người dùng tối ưu hóa cụm VKS theo nhu cầu cụ thể của mình. Điều này giúp tăng tính linh hoạt và đáp ứng các yêu cầu đặc biệt của từng ứng dụng.
 - **Ngoài ra,** trong bản cập nhật này, chúng tôi cũng đã khắc phục một số lỗi nhỏ để mang đến trải nghiệm người dùng tốt hơn.
-

Oct 03, 2024

VKS (VNGCloud Kubernetes Service) vừa ra mắt bản cập nhật mới nhất, mang đến nhiều tính năng và cải tiến cho người dùng. Dưới đây là những điểm nổi bật của bản cập nhật:

Triển khai Region mới:

- Bên cạnh Region HCM03, VKS hiện đã hỗ trợ thêm Region HAN01. Việc bổ sung này giúp khách hàng có thêm lựa chọn trong việc triển khai ứng dụng, đặc biệt hữu ích cho các doanh nghiệp có yêu cầu về vị trí đặt dữ liệu.

Tính năng mới:

- Network Type: Cilium Overlay, Cilium VPC Native Routing:** Ngoài Calico Overlay, bản phát hành lần này chúng tôi đã bổ sung hai loại mạng mới: Cilium Overlay và Cilium VPC Native Routing. Cilium Overlay cho phép bạn xây dựng mạng overlay linh hoạt, trong khi Cilium VPC Native Routing tích hợp chặt chẽ với VPC của VNG Cloud, giúp tối ưu hiệu suất và bảo mật cho ứng dụng của bạn. Chi tiết tham khảo thêm tại [đây](#).

Cải tiến:

- Multiple Subnet cho Cluster trên VKS:** VKS giờ đây hỗ trợ sử dụng nhiều subnet cho một cluster. Điều này cho phép bạn cấu hình từng node group trong cluster nằm ở các subnet khác nhau trong cùng một VPC, tối ưu hóa việc phân bổ tài nguyên và quản lý mạng.
- Chỉnh sửa Labels/Taints trên cụm VKS hiện có:** Với khả năng chỉnh sửa trực tiếp Labels/Taints trên cụm VKS đã triển khai, bạn có thể điều khiển việc lên lịch Pod, áp dụng các chính sách khác nhau cho các Node Group, và tùy chỉnh quy tắc lựa chọn node cho các ứng dụng. Điều này giúp quản lý và phân loại tài nguyên hiệu quả hơn.
- Enable/Disable lựa chọn sử dụng Private Service Endpoint:** Trước đây, khi tạo private cluster trên VKS, việc tạo private service endpoint là bắt buộc. Giờ đây, bạn có thể dễ dàng bật/tắt tính năng này, cho phép các dịch vụ trong cụm VKS giao tiếp qua địa chỉ IP nội bộ, tăng cường bảo mật và giảm thiểu rủi ro tấn công từ bên ngoài.
- Enable/Disable lựa chọn mã hóa Volume:** Tính năng mã hóa Volume cho phép bạn bảo vệ dữ liệu nhạy cảm được lưu trữ trong các Persistent Volume của cụm VKS. Việc này đảm bảo an toàn dữ liệu và tuân thủ các quy định về bảo vệ thông tin. Giờ đây, bạn có thể bật/tắt tính năng mã hóa cho từng Volume theo nhu cầu.

Aug 28, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều tính năng mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Tính năng mới:

- **Private Cluster:** Trước đây, các public cluster trên VKS đang sử dụng địa chỉ Public IP để giao tiếp giữa nodes và control plane. Để nâng cao bảo mật cho cluster của bạn, chúng tôi đã cho ra mắt mô hình private cluster. Tính năng Private Cluster giúp cho cụm K8S của bạn được bảo mật nhất có thể, mọi kết nối hoàn toàn là private từ kết nối giữa nodes tới control plane, kết nối từ client tới control plane, hay kết nối từ nodes tới các sản phẩm dịch vụ khác trong VNG Cloud như: vStorage, vCR, vMonitor, VNGCloud APIs,...Private Cluster là lựa chọn lý tưởng cho **các dịch vụ yêu cầu kiểm soát truy cập chặt chẽ, đảm bảo tuân thủ các quy định về bảo mật và quyền riêng tư dữ liệu.** Chi tiết 2 mô hình hoạt động của Cluster, bạn có thể tham khảo thêm tại [đây](#) và tham khảo các bước khởi tạo một private Cluster tại [đây](#).

Aug 26, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Cải tiến:

- **Kubernetes Version:** VKS đã bổ sung các image mới nhằm tối ưu hóa size, feature, network,...so với các image cũ. Việc xây dựng các image này cũng nhằm phục vụ cho cả hai loại cluster là Public và Private mà VKS sắp ra mắt. Cụ thể, trong bản phát hành này, chúng tôi đã bổ sung thêm các image sau:
 - 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) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Cải tiến:

- **Event History:** VKS đã bổ sung thêm các sự kiện của **Auto Scaling** và **Auto Healing**. Giờ đây, với Event History, bạn có thể theo dõi từng thay đổi xảy ra trong Cluster, từ việc tự động điều chỉnh quy mô (Auto Scaling) cho đến việc tự động phục hồi (Auto Healing). Các event này giúp tăng cường khả năng giám sát và quản lý cụm Kubernetes của bạn.

Aug 01, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều tính năng và cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Tính năng mới:

- **Giám sát tài nguyên VKS:** Người dùng có thể theo dõi trực tiếp tình trạng hoạt động của Cluster, Node, tình trạng sử dụng CPU, RAM, Memory,... của Node thông qua các dashboard trực quan. Để hiển thị dữ liệu lên dashboard, người dùng cần cài đặt `vmonitor-metric-agent` vào cụm mong muốn thực hiện giám sát. Chi tiết tham khảo thêm tại [đây](#).

July 25, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Cải tiến:

- **Nâng cấp quản lý VKS thông qua Terraform:** Người dùng có thể **đồng thời** điều chỉnh số lượng node (Number of nodes) và thay đổi số lượng node cho autoscale (Minimum/Maximum node Autoscale) ngay trong quá trình chỉnh sửa cấu hình. Với khả năng điều chỉnh nhiều thông số cùng lúc, việc quản lý cụm Kubernetes trở nên linh hoạt và thuận tiện hơn. Chi tiết tham khảo thêm ví dụ tại [đây](#).
-

July 23, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Cải tiến:

- **Nâng cấp Plugin VNGCloud Controller Manager, Plugin VNGCloud Ingress Controller:** Các lỗi đã được phát hiện trong các phiên bản trước đã được khắc phục, giúp hệ thống hoạt động mượt mà và tin cậy hơn.
-

July 18, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Cải tiến:

- **Nâng cấp Plugin VNGCloud BlockStorage CSI Driver:** Các lỗi đã được phát hiện trong các phiên bản trước đã được khắc phục, giúp hệ thống hoạt động mượt mà và tin cậy hơn.
-

July 17, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều cải tiến cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Cải tiến:

- **Private Node Group:** MTU cho các Node thuộc Private Node Group đã được cập nhật lên 1450. Điều này giúp cải thiện hiệu suất mạng cho các ứng dụng chạy trong Private Node Group.
 - **Number of nodes và AutoScale:** Giờ đây bạn có thể chỉnh sửa cả hai thuộc tính này trong cùng một API. Điều này giúp đơn giản hóa việc quản lý Cluster của bạn.
-

July 02, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều tính năng và cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Tính năng mới:

- **Hỗ trợ Stop POC cho Cluster:** Người dùng giờ đây có thể thực hiện Stop POC cho toàn bộ các tài nguyên đang được POC trên một Cluster một cách dễ dàng, thay vì phải thực hiện Stop POC riêng lẻ cho từng resource. Điều này giúp tiết kiệm thời gian và công sức khi chuyển Cluster từ tài nguyên thử nghiệm sang tài nguyên thật. Chi tiết tham khảo thêm tại [đây](#).

Cải tiến:

- **Trạng thái Node Group:** Bổ sung trạng thái "**Degraded**" để người dùng có thể theo dõi tình trạng hoạt động của Node Group một cách chính xác hơn. Trạng thái này sẽ hiển thị khi số node hoạt động ít hơn số replica thực tế.
 - **Timeout cho Cluster và Node Group:** Đã thêm thời gian chờ (timeout) cho việc tạo Cluster và Node Group, cải tiến này đảm bảo VKS vận hành mượt mà và hiệu quả, đồng thời cung cấp thông tin rõ ràng và kịp thời cho người dùng. Timeout cho việc tạo Cluster là **1 giờ** và cho Node Group là **3 giờ**. Nếu sau khoảng thời gian này mà Cluster hoặc Node Group của bạn chưa được tạo thành công, chúng tôi sẽ cập nhật trạng thái chúng về ERROR. Lúc này, bạn có thể thực hiện xóa và tạo Cluster, Node Group khác thay thế.
 - **KubeConfig Access:** Việc truy cập vào tệp tin KubeConfig giờ chỉ được phép khi Cluster đã Active. Cải tiến này giúp người dùng tránh các lỗi cấu hình khi sử dụng Terraform để tự động hóa việc triển khai Kubernetes.
-

June 27, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Cải tiến:

- Nâng cấp Plugin VNGCloud Controller Manager, Plugin VNGCloud Ingress Controller:**
Các lỗi đã được phát hiện trong các phiên bản trước đã được khắc phục, giúp hệ thống hoạt động mượt mà và tin cậy hơn.

June 19, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Cải tiến:

- Nâng cấp tính năng thiết lập kích cỡ PVC (Persistent Volume Claim Size):** Người dùng giờ đây có thể chỉ định kích cỡ tối thiểu cho ổ đĩa CSI là **1GB** thay vì kích cỡ tối thiểu là 20GB như trước đây. Chi tiết bạn có thể tham khảo thêm tại [Volume](#) và [Integrate with Container Storage Interface](#).
- Thay đổi Storage Class mặc định sử dụng cho Cluster:** thay đổi mặc định từ ổ đĩa loại SSD - IOPS 200 thành mặc định ổ đĩa loại SSD - IOPS 3000.
- Nâng cấp Plugin VNGCloud Controller Manager, Plugin VNGCloud Ingress Controller:** cải tiến plugin giúp tránh trùng lặp việc đặt tên Load Balancer.

Chú ý:

Do Storage Class mặc định cũ đã được chúng tôi xóa khỏi hệ thống, nếu bạn muốn tiếp tục sử dụng và thực hiện resize storage class này, bạn có thể:

- Tạo Storage Class có tên sc-iops-200-retain với Volume Type mà bạn mong muốn.
- Resize Storage Class thông qua lệnh:

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

Chi tiết tham khảo thêm tại [Integrate with Container Storage Interface](#).

June 12, 2024

VKS (VNGCloud Kubernetes Service) giới thiệu bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều tính năng và cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Tính năng mới:

- **Hỗ trợ người dùng làm việc với VKS thông qua Terraform:** Người dùng có thể dễ dàng khởi tạo Cluster và Node Group trong VKS bằng Terraform. Chi tiết tham khảo thêm tại [đây](#).

Cải tiến:

- **Nâng cấp Plugin VNGCloud Controller Manager:** Bổ sung Annotation để cấu hình Load Balancer hỗ trợ Proxy Protocol. Chi tiết tham khảo thêm tại [đây](#).

May 30, 2024

Chúng tôi vô cùng trân trọng thông báo, bản release chính thức (**General Availability**) của dịch vụ VNGCloud Kubernetes Service đã có sẵn. Với bản release chính thức này, ngoài các tính năng mà chúng tôi đã cung cấp trên các bản release trước đó, phiên bản này sẽ mang đến nhiều tính năng và cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Tính năng mới:

- **Re-activate :** VKS cho phép bạn thực hiện yêu cầu hệ thống tự động khởi tạo lại IAM Service Account mặc định khi bạn xóa nhầm hoặc thay đổi thông tin IAM Service Account mặc định đã khởi tạo trước đó. IAM Service Account mặc định là IAM Service Account được hệ thống VKS tự động khởi tạo khi bạn bắt đầu làm việc với VKS, chúng tôi sẽ sử dụng IAM Service Account này để khởi tạo các resource cho Cluster của bạn.

- **Event History:** VKS sẽ thực hiện hiển thị lịch sử các sự kiện xảy ra khi người dùng làm việc với Cluster hoặc từng Node Group. Đây sẽ là một cách giúp bạn có thể giám sát các hoạt động xảy ra với Cluster của bạn, từ đó hạn chế các hoạt động bất thường xảy ra.
- **Volume:** VKS đã tích hợp hiển thị danh sách Volume tại Resource Tab, giúp bạn dễ dàng quản lý các Volume đang được attach vào Cluster của bạn.
- **Load Balancer:** VKS đã tích hợp hiển thị danh sách Load Balancer tại Resource Tab, giúp bạn dễ dàng quản lý các Load Balancer đang được sử dụng cho Cluster của bạn.

Cải tiến:

- **Hiệu năng:** VKS đã thực hiện tối ưu hiệu năng khi khởi tạo Cluster. Cụ thể, tại bản Alpha, thời gian từ khi bắt đầu khởi tạo Cluster (với Default Node Group) tới khi Cluster chuyển trạng thái **ACTIVE** vào khoảng 04:00s tới 04:30s. Hiện tại thời gian này đã được chúng tôi tối ưu về **02:30s tới 03:00s** tùy thuộc vào từng Cluster và từng thời điểm mà bạn khởi tạo.
- **Garbage collection of unused containers and images:** VKS sẽ tự động xóa các image không được sử dụng khi disk chạm mức giới hạn sử dụng (tỷ lệ usage/ quota >= 85%).
- **Ngoài ra**, bản GA này được chúng tôi cải thiện một vài vấn đề khác như:
 - Thay đổi cách đặt tên **Node Name** giúp bạn dễ dàng sử dụng và quản lý Cluster của bạn. Cụ thể, tên Node Name sẽ có thêm thông tin **Cluster Name, Node Group Name**.
 - Xóa **User Builder** trên User's Worker Node.
 - Thay đổi cơ chế **SSH** từ Port 22 qua Port 234.

Nếu bạn gặp bất kỳ vấn đề với bản phát hành chính thức này, vui lòng liên hệ với bộ phận hỗ trợ của VKS để được trợ giúp.

May 03, 2024

Bản cập nhật mới nhất cho VKS đã có sẵn, mang đến nhiều tính năng và cải tiến mới cho người dùng. Dưới đây là chi tiết về bản cập nhật:

Tính năng mới:

- **Hỗ trợ tính năng Whitelist:** VKS cho phép tạo Private Node Group với chỉ Private IP đồng thời cho phép IP nào kết nối tới Cluster thông qua tính năng Whitelist IP. Chi tiết tham khảo thêm tại [Whitelist](#).

Cài tiến:

- **Tối ưu hóa hệ thống:** Giúp hệ thống hoạt động trơn tru và hiệu quả hơn.
- **Sửa lỗi:** Khắc phục một số lỗi nhỏ để mang đến trải nghiệm người dùng tốt hơn.

Nếu bạn gặp bất kỳ vấn đề nào sau khi cập nhật, vui lòng liên hệ với bộ phận hỗ trợ của VKS để được trợ giúp.

April 17, 2024

Chúng tôi vô cùng giới thiệu bản cập nhật mới cho dịch vụ VKS, mang đến cho bạn trải nghiệm quản lý Kubernetes mạnh mẽ và hiệu quả hơn bao giờ hết!

Điểm nổi bật:

- **Quản lý Control Plane hoàn toàn tự động (Fully Managed control plane):** VKS sẽ giải phóng bạn khỏi gánh nặng quản lý Control Plane của Kubernetes, giúp bạn tập trung vào việc phát triển ứng dụng.
- **Hỗ trợ các phiên bản Kubernetes mới nhất:** VKS luôn cập nhật những phiên bản Kubernetes mới nhất (minor version từ 1.27, 1.28, 1.29) để đảm bảo bạn luôn tận dụng được những tính năng tiên tiến nhất.
- **Kubernetes Networking:** VKS tích hợp Calico CNI, mang lại tính hiệu quả và bảo mật cao.
- **Upgrade seamlessly:** VKS hỗ trợ nâng cấp giữa các phiên bản Kubernetes một cách dễ dàng và nhanh chóng, giúp bạn luôn cập nhật những cải tiến mới nhất.
- **Scaling & Healing Automatically:** VKS tự động mở rộng Node group khi cần thiết và tự động sửa lỗi khi node gặp vấn đề, giúp bạn tiết kiệm thời gian và công sức quản lý.
- **Giảm chi phí và nâng cao độ tin cậy:** VKS triển khai Control Plane của Kubernetes ở chế độ sẵn sàng cao và hoàn toàn miễn phí, giúp bạn tiết kiệm chi phí và nâng cao độ tin cậy cho hệ thống.
- **Tích hợp Blockstore Native (Container Storage Interface - CSI):** VKS cho phép bạn quản lý Blockstore thông qua YAML của Kubernetes, cung cấp lưu trữ bền vững cho container và hỗ trợ các tính năng quan trọng như thay đổi kích thước, thay đổi IOPS và snapshot volume.
- **Tích hợp Load Balancer (Network Load Balancer, Application Load Balancer) thông qua các driver được tích hợp sẵn như VNGCloud Controller Manager, VNGCloud**

Ingress Controller: VKS cung cấp khả năng quản lý NLB/ALB thông qua YAML của Kubernetes, giúp bạn dễ dàng expose Service trong Kubernetes ra bên ngoài.

- **Nâng cao bảo mật:** VKS cho phép bạn tạo Private Node Group với chỉ Private IP và kiểm soát quyền truy cập vào cluster thông qua tính năng Whitelist IP, đảm bảo an toàn cho hệ thống của bạn.

Với những tính năng đột phá này, VKS hứa hẹn sẽ mang đến cho bạn trải nghiệm quản lý Kubernetes hoàn toàn mới, giúp bạn tối ưu hóa hiệu quả và tiết kiệm chi phí!

Hãy liên hệ với chúng tôi để được tư vấn và hỗ trợ thêm!

Bắt đầu với VKS

Bạn có thể sử dụng các hướng dẫn sau đây để bắt đầu làm việc với VKS. Trong quá trình sử dụng, nếu gặp vấn đề gì với dịch vụ, bạn hãy vui lòng liên hệ với VNG Cloud qua email support@vngcloud.vn hoặc hotline **19001549**.

Hướng dẫn cài đặt và cấu hình công cụ kubectl trong Kubernetes

Điều kiện cần

Bạn cần phải sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster. Ví dụ, một client v1.2 nên được hoạt động với master v1.1, v1.2 và v1.3. Sử dụng phiên bản mới nhất của kubectl giúp tránh được các vấn đề không lường trước được.

Cài đặt kubectl trên Linux

Cài đặt kubectl binary với curl trên Linux

Bước 1: Tải về phiên bản mới nhất với câu lệnh:

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

Để tải về phiên bản cụ thể, hãy thay thế phần `$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)` trong câu lệnh với một phiên bản cụ thể.

Ví dụ, để tải về phiên bản v1.17.0 trên Linux, hãy chạy lệnh:

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

Bước 2: Tạo kubectl binary thực thi qua lệnh:

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

Bước 3: Đưa bản binary vào biến môi trường PATH của bạn qua lệnh:

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

Bước 4: Kiểm tra chắc chắn rằng phiên bản bạn cài là mới nhất qua lệnh:

```
kubectl version
```

Cài đặt kubectl sử dụng trình quản lý gói

- Với các hệ điều hành CentOS, RHEL hoặc Fedora, bạn có thể chạy lệnh:

```
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.listsu
```

Cài đặt kubectl với snap

Nếu bạn đang sử dụng Ubuntu hoặc distro Linux khác hỗ trợ trình quản lý gói [snap](#), thì kubectl đã có sẵn trong [snap](#).

Bước 1: Chuyển sang user snap và thực thi câu lệnh cài đặt:

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

Bước 2: Kiểm tra phiên bản bạn vừa cài là mới nhất:

```
kubectl version
```

Cài đặt kubectl trên macOS

Cài đặt kubectl binary với curl trên macOS

Bước 1: Tải về phiên bản mới nhất:

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

Để tải về phiên bản cụ thể, hãy thay thế phần `$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)` trong câu lệnh với phiên bản cụ thể.

Ví dụ, để tải về phiên bản v1.17.0 trên macOS, hãy chạy lệnh:

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

Bước 2: Tạo kubectl binary thực thi qua lệnh:

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

Bước 3: Đưa bản binary vào biến môi trường PATH của bạn qua lệnh:

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

Bước 4: Kiểm tra chắc chắn rằng phiên bản bạn cài là mới nhất qua lệnh:

```
kubectl version
```

Cài đặt với Homebrew trên macOS

Nếu bạn đang trên macOS và sử dụng trình quản lý gói [Homebrew](#), bạn có thể cài đặt kubectl với Homebrew.

Bước 1: Chạy câu lệnh cài đặt:

```
brew install kubernetes-cli
```

hoặc lệnh:

```
brew install kubectl
```

Bước 2: Kiểm tra chắc chắn rằng phiên bản bạn cài là mới nhất:

```
kubectl version
```

Cài đặt với Macports trên macOS

Nếu bạn đang trên macOS và sử dụng trình quản lý gói [Macports](#), bạn có thể cài đặt kubectl với Macports.

Bước 1: Chạy câu lệnh cài đặt:

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

Bước 2: Kiểm tra chắc chắn rằng phiên bản bạn cài là mới nhất:

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

Cài đặt kubectl trên Windows

Cài đặt kubectl binary với curl trên Windows

Bước 1: Tải về phiên bản mới nhất v1.17.0 từ [đường dẫn này](#). Hoặc nếu bạn đã cài đặt `curl`, hãy sử dụng câu lệnh sau:

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

Để tìm ra phiên bản ổn định mới nhất, hãy xem <https://storage.googleapis.com/kubernetes-release/release/stable.txt>.

Bước 2: Đưa bản binary vào biến môi trường PATH của bạn.

Bước 3: Kiểm tra chắc chắn phiên bản `kubectl` giống với bản đã tải về:

```
kubectl version
```

Ghi chú: [Docker Desktop cho Windows](#) thêm phiên bản `kubectl` riêng của nó vào PATH. Nếu bạn đã cài đặt Docker Desktop trước đây, bạn có thể cần đặt đường dẫn PATH của bạn trước khi bản cài đặt Docker Desktop thêm 1 PATH vào hoặc loại bỏ `kubectl` của Docker Desktop.

Cài đặt với Powershell từ PSGallery

Nếu bạn đang ở trên Windows và sử dụng trình quản lý gói [Powershell Gallery](#), bạn có thể cài đặt và cập nhật `kubectl` với Powershell.

Bước 1: Thực thi các câu lệnh cài đặt sau (hãy đảm bảo bạn tự định nghĩa `DownloadLocation`):

```
Install-Script -Name install-kubectl -Scope CurrentUser -Forceinstall-kubectl.ps
```

Ghi chú: Nếu bạn không định nghĩa `DownloadLocation`, `kubectl` sẽ được cài đặt ở thư mục temp của user.

Bản cài đặt sẽ tạo ra `$HOME/.kube` và hướng dẫn để tạo ra file cấu hình

Bước 1: Kiểm tra chắc chắn rằng phiên bản bạn cài là mới nhất:

```
kubectl version
```

Ghi chú: Cập nhật của bản cài đặt sẽ được thực hiện khi chạy lại các câu lệnh từ bước 1.

Cài đặt trên Windows sử dụng Chocolatey hoặc Scoop

Bước 1: Để cài đặt `kubectl` trên Windows bạn có thể sử dụng trình quản lý gói [Chocolatey](#) hoặc bộ cài đặt câu lệnh [Scoop](#).

- Nếu bạn dùng Choco

```
choco install kubernetes-cli
```

- Nếu bạn dùng Scoop

```
scoop install kubectl
```

Bước 2: Kiểm tra chắc chắn rằng phiên bản bạn cài là mới nhất:

```
kubectl version
```

Bước 3: Di chuyển tới thư mục home của bạn:

```
cd %USERPROFILE%
```

Bước 4: Tạo thư mục `.kube`:

```
mkdir .kube
```

Bước 5: Di chuyển tới thư mục `.kube` bạn vừa mới tạo:

```
cd .kube
```

Bước 6: Cấu hình kubectl để sử dụng Kubernetes cluster từ xa:

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

Ghi chú: Chỉnh sửa file cấu hình với trình soạn thảo văn bản, như là Notepad.

Tải xuống từ một phần của Google Cloud SDK

Bạn có thể cài đặt kubectl từ một phần của Google Cloud SDK.

Bước 1: Cài đặt [Google Cloud SDK](#).

Bước 2: Thực thi câu lệnh cài đặt `kubectl`:

```
gcloud components install kubectl
```

Bước 3: Kiểm tra chắc chắn rằng phiên bản bạn cài là mới nhất:

```
kubectl version
```

Xác minh cấu hình kubectl

- Để kubectl tìm kiếm và truy cập Kubernetes cluster, nó cần một kubeconfig file, được tự động tạo ra khi bạn tạo mới một cluster sử dụng [kube-up.sh](#) hoặc triển khai thành công một Minikube cluster. Mặc định thì cấu hình của kubectl được xác định tại `~/ .kube/config`.
- Kiểm tra kubectl được cấu hình đúng bằng việc xem trạng thái của cluster:
`kubectl cluster-info`

```
kubectl cluster-info
```

- Nếu bạn trông thấy một URL phản hồi, thì kubectl đã được cấu hình đúng để truy cập vào cluster của bạn.
- Nếu bạn trông thấy một tin nhắn tương tự bên dưới, thì kubectl chưa được cấu hình đúng hoặc chưa thể kết nối với Kubernetes cluster.

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

Ví dụ như, nếu bạn đang định chạy một Kubernetes cluster trên laptop của bạn (locally), bạn sẽ cần một công cụ như Minikube được cài trước đó và chạy lại các câu lệnh bên trên.

Nếu kubectl cluster-info trả về url nhưng bạn không thể truy cập vào cluster của bạn, thì hãy kiểm tra nó đã được cấu hình đúng hay chưa, bằng cách:

```
kubectl cluster-info dump
```

Các lựa chọn cấu hình kubectl

Kích hoạt shell completion

- kubectl cung cấp autocomplete hỗ trợ cho Bash and Zsh, giúp bạn giảm thiểu việc phải gõ nhiều câu lệnh.
 - Bên dưới đây là các bước để thiết lập autocomplete cho Bash (bao gồm sự khác nhau giữa Linux và macOS) và Zsh.
-

Bash trên Linux

Giới thiệu

- Kubelet completion script cho Bash được tạo ra với câu lệnh `kubectl completion bash`. Sau khi script được tạo ra, bạn cần source (thực thi) script đó để kích hoạt tính năng autocomplete.
- Tuy nhiên, completion script phụ thuộc vào [bash-completion](#), nên bạn phải cài đặt bash-completion trước đó (kiểm tra bash-completion tồn tại với câu lệnh `type _init_completion`).

Cài đặt bash-completion

1. bash-completion được cung cấp bởi nhiều trình quản lý gói (xem tại [đây](#)). Bạn có thể cài đặt với lệnh `apt-get install bash-completion` hoặc `yum install bash-completion`.
2. Các lệnh trên tạo ra `/usr/share/bash-completion/bash_completion`, đây là script chính của bash-completion. Tùy thuộc vào trình quản lý gói của bạn, mà bạn phải source (thực thi) file này trong file `~/.bashrc`.
3. Để tìm ra file này, reload lại shell hiện tại của bạn và chạy lệnh `type _init_completion`. Nếu thành công, bạn đã thiết lập xong, không thì hãy thêm đoạn sau vào file `~/.bashrc` của bạn:

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

4. Reload lại shell của bạn và xác nhận bash-completion được cài đặt đúng bằng lệnh `type _init_completion`.

Kích hoạt kubectl autocomplete

Bây giờ bạn cần đảm bảo rằng kubectl completion script được sourced trên tất cả các session của shell. Có 2 cách để làm việc này:

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

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

- Thêm script vào thư mục `/etc/bash_completion.d` :

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

- Nếu bạn có một alias cho kubectl, bạn có thể thêm một shell completion nữa cho alias đó:

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

Ghi chú: bash-completion sources tất cả completion scripts trong `/etc/bash_completion.d`.

Các cách trên đều hiệu quả tương đương nhau. Sau khi reload lại shell, kubectl autocompletion sẽ làm việc.

Bash trên macOS

Giới thiệu

Kubectl completion script trên Bash được tạo ra bởi `kubectl completion bash`. Source script này sẽ kích hoạt tính năng kubectl completion.

Tuy nhiên, kubectl completion script phụ thuộc vào [bash-completion](#) mà bạn cài trước đó.

Cảnh báo: Có 2 phiên bản của bash-completion là v1 và v2. V1 dành cho Bash 3.2 (Bash mặc định trên macOS), và v2 dành cho Bash 4.1+. Kubectl completion script **không làm việc** phù hợp với bash-completion v1 và Bash 3.2. Nó tương thích với **bash-completion v2 và Bash 4.1+**. Vì vậy, để sử dụng kubectl completion một cách chính xác trên macOS thì bạn phải cài đặt Bash 4.1+ ([hướng dẫn](#)). Hướng dẫn tiếp theo giả định rằng bạn đang sử dụng Bash 4.1+ (nghĩa là, bất kỳ phiên bản Bash nào từ 4.1 trở lên).

Cài đặt bash-completion

Ghi chú: Như đã đề cập, những hướng dẫn này giả định bạn đang sử dụng Bash 4.1+, có nghĩa rằng bạn sẽ cài đặt bash-completion v2 (trái ngược với Bash 3.2 và bash-completion v1, trong trường hợp đó, kubectl completion sẽ không hoạt động).

1. Bạn có thể kiểm tra bash-completion v2 đã cài đặt trước đó chưa với lệnh `type _init_completion`. Nếu chưa, bạn có thể cài đặt nó với Homebrew:

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

2. Từ đầu ra của lệnh này, hãy thêm đoạn sau vào file `~/.bashrc` của bạn:
`export BASH_COMPLETION_COMPAT_DIR="/usr/local/etc/bash_completion.d"`

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

Tải lại shell của bạn và xác minh rằng bash-completion v2 được cài đặt chính xác chưa bằng lệnh `type _init_completion`.

Kích hoạt kubectl autocompletion

Bây giờ bạn phải đảm bảo rằng kubectl completion script đã được sourced trong tất cả các phiên shell của bạn. Có nhiều cách để đạt được điều này:

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

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

- Thêm completion script vào thư mục `/usr/local/etc/bash_completion.d` :

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

- Nếu bạn có alias cho kubectl, bạn có thể mở rộng shell completion để làm việc với alias đó:

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

- Nếu bạn đã cài kubectl với Homebrew (như đã giới thiệu [bên trên](#)) thì kubectl completion script sẽ có trong `/usr/local/etc/bash_completion.d/kubectl`. Trong trường hợp này thì bạn không cần làm gì cả.

Ghi chú: Cài đặt theo cách Homebrew đã source tất cả các tệp trong thư mục `BASH_COMPLETION_COMPAT_DIR`, đó là lý do tại sao hai phương thức sau hoạt động.

Trong mọi trường hợp, sau khi tải lại shell của bạn, kubectl completion sẽ hoạt động.

Zsh

- Kubectl completion script cho Zsh được tạo ra với lệnh `kubectl completion zsh`. Source completion script trong shell của bạn sẽ kích hoạt kubectl autocompletion. Để nó hoạt động cho tất cả các shell, thêm dòng sau vào file `~/.zshrc` :

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

- Nếu bạn có alias cho kubectl, bạn có thể mở rộng shell completion để làm việc với alias đó:

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

Sau khi tải lại shell, kubectl autocompletion sẽ hoạt động.

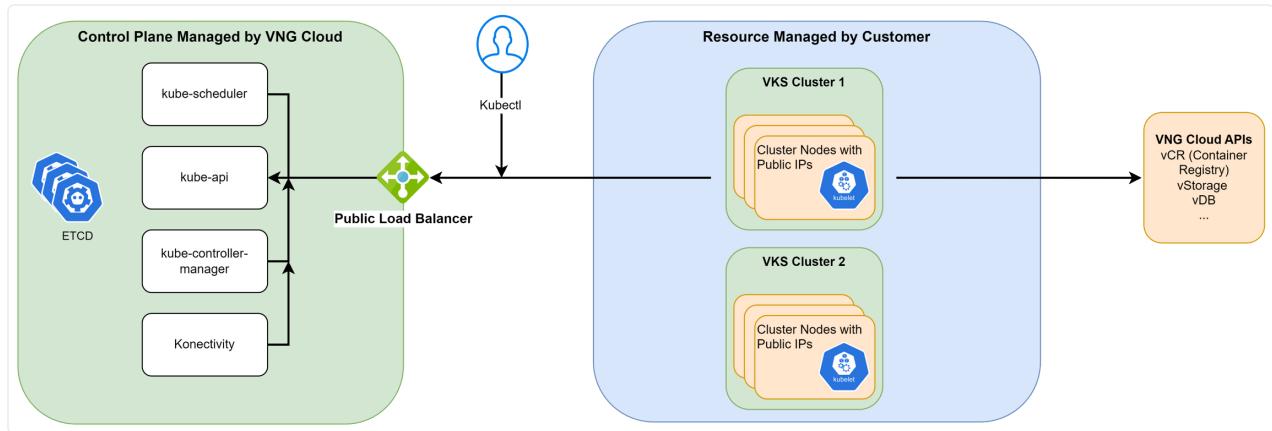
- Nếu bạn nhận được lỗi `complete:13: command not found: compdef`, thêm dòng sau vào đầu file `~/.zshrc` :

```
autoload -Uz compinit
compinit
```

Chi tiết tham khảo thêm tại kubernetes.io

Khởi tạo một Public Cluster

Model



Khi bạn khởi tạo một **Public Cluster** với **Public Node Group**, hệ thống VKS sẽ:

- Tạo VM có Floating IP (tức có IP Public). Lúc này các VM (Node) này có thể join trực tiếp vào cụm K8S thông qua Public IP này. Bằng cách sử dụng Public Cluster và Public Node Group, bạn có thể dễ dàng tạo các cụm Kubernetes và thực hiện expose service mà không cần sử dụng Load Balancer. Việc này sẽ góp phần tiết kiệm chi phí cho cụm của bạn.

Khi bạn khởi tạo một **Public Cluster** với **Private Node Group**, hệ thống VKS sẽ:

- Tạo VM không có Floating IP (tức không có IP Public). Lúc này các VM (Node) này không thể join trực tiếp vào cụm K8S. Để các VM này có thể join vào cụm K8S, bạn cần phải sử dụng một NAT Gateway (**NATGW**). **NATGW** hoạt động như một trạm chuyển tiếp, cho phép các VM kết nối với cụm K8S mà không cần IP Public. Với VNG Cloud, chúng tôi khuyến cáo bạn sử dụng Pfsense hoặc Palo Alto như một NATGW cho Cluster của bạn. Pfsense sẽ giúp bạn quản lý lưu lượng mạng đến và đi (inbound và outbound traffic) một cách hiệu quả, đảm bảo an ninh mạng và quản lý truy cập. Bên cạnh đó, việc sử dụng Private Node Group sẽ giúp bạn kiểm soát các ứng dụng trong cụm được bảo mật hơn, cụ thể bạn có thể thực hiện giới hạn quyền truy cập control plane thông qua tính năng Whitelist IP.

Khởi tạo một Public Cluster với Public Node Group

Điều kiện cần

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn tại [đây](#).
- Có ít nhất 1 **SSH key** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).
- Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.

Khởi tạo Cluster

Cluster trong Kubernetes là một tập hợp gồm một hoặc nhiều máy ảo (VM) được kết nối lại với nhau để chạy các ứng dụng được đóng gói dạng container. Cluster cung cấp một môi trường thống nhất để triển khai, quản lý và vận hành các container trên quy mô lớn.

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn, bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Cluster và Node Group của bạn tại Cluster Configuration, Default Node Group Configuration, Plugin. **Mặc định chúng tôi sẽ khởi tạo cho bạn một Public Cluster với Public Node Group.**

Bước 5: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Kết nối và kiểm tra thông tin Cluster vừa tạo

Sau khi Cluster được khởi tạo thành công, bạn có thể thực hiện kết nối và kiểm tra thông tin Cluster vừa tạo theo các bước:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/k8s-cluster>

Bước 2: Danh sách Cluster được hiển thị, chọn biểu tượng và chọn **Download Config File** để thực hiện tải xuống file kubeconfig. File này sẽ giúp bạn có toàn quyền truy cập vào Cluster của bạn.

Bước 3: Đổi tên file này thành config và lưu nó vào thư mục **~/.kube/config**

Bước 4: Thực hiện kiểm tra Cluster thông qua lệnh:

- Chạy câu lệnh sau đây để kiểm tra **node**

```
kubectl get nodes
```

- Nếu kết quả trả về như bên dưới tức là bạn Cluster của bạn được khởi tạo thành công với 3 node như bên dưới.

NAME	STATUS	ROLES	AGE	VERSI0
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready	<none>	50m	v1.28.
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready	<none>	23m	v1.28.
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready	<none>	28m	v1.28.

Deploy một Workload

Sau đây là hướng dẫn để bạn deploy service nginx trên Kubernetes.

Bước 1: Tạo Deployment và Service cho Nginx app

- Tạo file **nginx-service.yaml** với nội dung sau:

```
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 Deployment này bằng lệnh:

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

Bước 2: Kiểm tra thông tin Deployment, Service trước khi expose ra Internet.

- Chạy câu lệnh sau đây để kiểm tra **Deployment**

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

- Nếu kết quả trả về như bên dưới tức là bạn đã deploy service nginx thành công.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	2d4h
service/nginx-service	ClusterIP	10.96.178.229	<none>	80/TCP	74s
NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS
deployment.apps/nginx-app	1/1	1	1	74s	nginx
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx-app-7f45b65946-5pcvz	1/1	Running	0	74s	172.16.24.20

Expose Nginx Service ra Internet

- Chạy câu lệnh sau đây để expose nginx-service ra internet:

```
kubectl expose deployment nginx-app --type=NodePort --port=30080 --target-port=8
```

- Nếu kết quả trả về như bên dưới tức là bạn đã expose Service ra Internet thành công.

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>	30080:31289/TCP	
service/nginx-service	ClusterIP	10.96.178.229	<none>	80/TCP	
NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS
deployment.apps/nginx-app	1/1	1	1	2m43s	nginx
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx-app-7f45b65946-5pcvz	1/1	Running	0	2m43s	172.16.24.

Để truy cập vào app nginx vừa export, bạn có thể sử dụng URL với định dạng:

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

Trong đó node_ip có thể là địa chỉ node_port của bất kỳ node nào trong cluster. Bạn có thể lấy thông tin External IP của Node tại giao diện vServer. Cụ thể truy cập tại <https://hcm-3.console.vngcloud.vn/vserver/v-server/cloud-server>.

Ví dụ, bên dưới tôi đã truy cập thành công vào app nginx với địa chỉ :

<http://61.28.231.65:31007/>

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.



Nếu bạn muốn expose service này thông qua vLB Layer4, vLB Layer7, vui lòng tham khảo tại:

- [Expose một service thông qua vLB Layer4](#)
- [Expose một service thông qua vLB Layer7](#)

Khởi tạo một Public Cluster với Private Node Group

Điều kiện cần

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn tại [đây](#).
- Có ít nhất 1 **SSH** key đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).
- Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.

 **Chú ý:**

- **Để đảm bảo các VM trong các NodeGroup trên subnet có thể đi outbound ra internet và kết nối được tới Control Plane thì bạn bắt buộc phải thiết lập NAT Gateway. Cụ thể tham khảo thêm tại mục bên dưới.**

Khởi tạo Palo Alto hoặc Pfsense thay thế cho NAT Gateway

 **Chú ý:**

- Để được hỗ trợ tốt nhất khi sử dụng Palo Alto hoặc Pfsense, vui lòng liên hệ với đội ngũ chuyên viên của chúng tôi qua Hotline **1900 1549** hoặc email support@vngcloud.vn.

Hoặc bạn có thể chọn sử dụng Palo Alto hoặc Pfsense để làm việc với Private Node Group theo hướng dẫn tại:

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

Khởi tạo Route Table

Sau khi Palo Alto, Pfsense được khởi tạo thành công, bạn cần tạo một Route table để kết nối tới các mạng khác nhau. Cụ thể thực hiện theo các bước sau để tạo Route table:

Bước 1: Truy cập vào <https://hcm-3.console.vngcloud.vn/vserver/network/route-table>

Bước 2: Tại thanh menu điều hướng, chọn **Tab Network/ Route table**.

Bước 3: Chọn **Create Route table**.

Bước 4: Nhập tên mô tả cho Route table. Tên Route table có thể bao gồm các chữ cái (a-z, A-Z, 0-9, '_', '-'). Độ dài dữ liệu đầu vào nằm trong khoảng từ 5 đến 50. Nó không được bao gồm khoảng trắng ở đầu hoặc ở cuối.

Bước 5: Chọn **VPC** cho Route table của bạn, nếu chưa có VPC cần tạo mới một VPC theo hướng dẫn tại [Trang VPC](#). **VPC sử dụng để thiết lập Route table phải là VPC được chọn sử dụng cho Palo Alto hoặc Pfsense và Cluster của bạn.**

Bước 6: Chọn **Create** để tạo mới Route table.

Bước 7: Chọn tại Route table vừa tạo sau đó chọn **Edit Routes**.

Bước 8: Tại phần thêm mới **Route** hãy nhập vào các thông tin:

- Đối với Destination, hãy nhập **Destination CIDR là 0.0.0.0/0**
- Đối với Target, hãy nhập **Target CIDR là địa chỉ IP Network Interface của Palo Alto hoặc Pfsense tương ứng**.

Khởi tạo Cluster

Cluster trong Kubernetes là một tập hợp gồm một hoặc nhiều máy ảo (VM) được kết nối lại với nhau để chạy các ứng dụng được đóng gói dạng container. Cluster cung cấp một môi trường thống nhất để triển khai, quản lý và vận hành các container trên quy mô lớn.

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn, bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Cluster và Node Group của bạn tại Cluster Configuration, Default Node Group Configuration, Plugin. **Mặc định chúng tôi sẽ khởi tạo cho bạn một Public Cluster với Public Node Group. Bạn cần thay đổi lựa chọn của bạn thành Private Node Group.**

Bước 5: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Kết nối và kiểm tra thông tin Cluster vừa tạo

Sau khi Cluster được khởi tạo thành công, bạn có thể thực hiện kết nối và kiểm tra thông tin Cluster vừa tạo theo các bước:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/k8s-cluster>

Bước 2: Danh sách Cluster được hiển thị, chọn biểu tượng và chọn **Download config file** để thực hiện tải xuống file kubeconfig. File này sẽ giúp bạn có toàn quyền truy cập vào Cluster của bạn.

Bước 3: Đổi tên file này thành config và lưu nó vào thư mục `~/.kube/config`

Bước 4: Thực hiện kiểm tra Cluster thông qua lệnh:

- Chạy câu lệnh sau đây để kiểm tra **node**

```
kubectl get nodes
```

- Nếu kết quả trả về như bên dưới tức là bạn Cluster của bạn được khởi tạo thành công với 3 node như bên dưới.

NAME	STATUS	ROLES	AGE	VERSIO
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready	<none>	50m	v1.28.
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready	<none>	23m	v1.28.
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready	<none>	28m	v1.28.

Deploy một Workload

Sau đây là hướng dẫn để bạn deploy service nginx trên Kubernetes.

Bước 1: Tạo Deployment cho Nginx app.

- Tạo file **nginx-service-lb4.yaml** với nội dung sau:

```

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 Deployment này bằng lệnh:

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

Bước 2: Kiểm tra thông tin Deployment, Service trước khi expose ra Internet.

- Chạy câu lệnh sau đây để kiểm tra **Deployment**

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

- Nếu kết quả trả về như bên dưới tức là bạn đã deploy service nginx thành công.

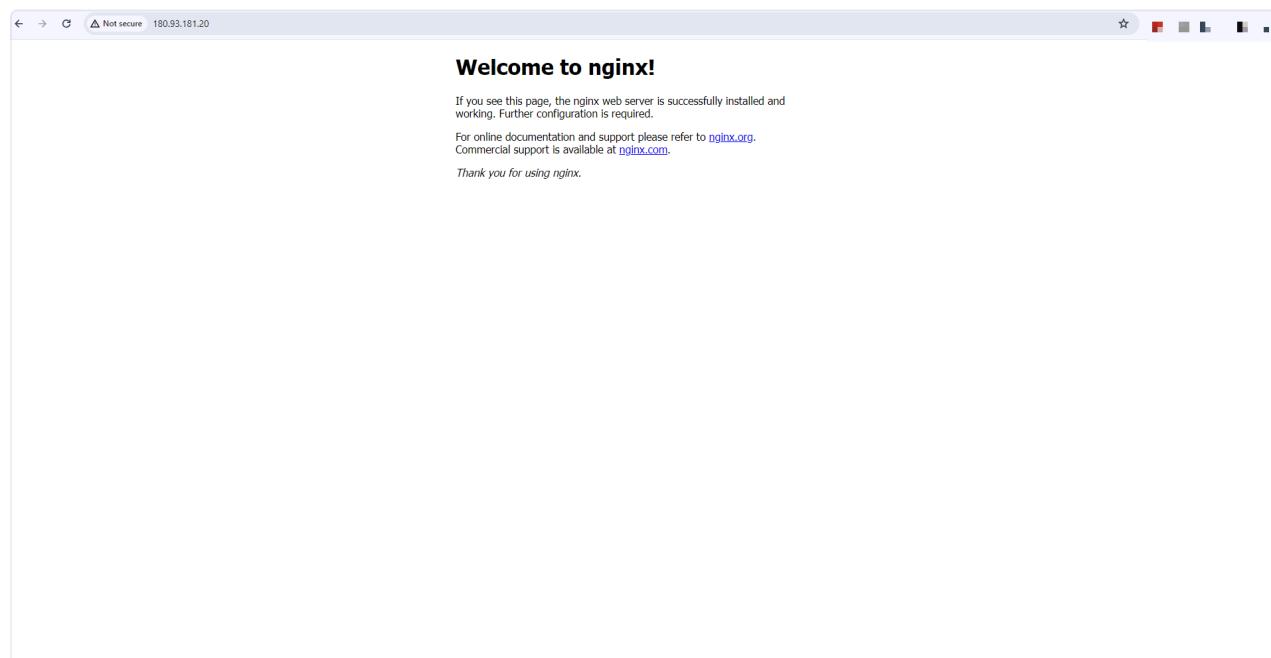
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	
service/nginx-app	NodePort	10.96.215.192	<none>	30080:31	
service/nginx-service	LoadBalancer	10.96.179.221	<pending>	80:32624	
NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINER
deployment.apps/nginx-app	1/1	1	1	2m16s	nginx
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx-app-7f45b65946-t7d7k	1/1	Running	0	2m16s	172.16.

Bước 3: Để truy cập vào app nginx vừa export, bạn có thể sử dụng URL với định dạng:

http://Endpoint/

Bạn có thể lấy thông tin Public Endpoint của Load Balancer tại giao diện vLB. Cụ thể truy cập tại <https://hcm-3.console.vngcloud.vn/vserver/load-balancer/vlb/>

Ví dụ, bên dưới tôi đã truy cập thành công vào app nginx với địa chỉ : <http://180.93.181.20/>



Palo Alto as a NAT Gateway

Sử dụng hướng dẫn bên dưới để làm việc với Private Node group thông qua Palo Alto.

Điều kiện cần

Để có thể sử dụng Palo Alto làm NAT Gateway cho Cluster trên hệ thống VKS, bạn cần có:

- Một **server (VM) Windows** đã được khởi tạo trên hệ thống **vServer** với cấu hình như sau:

Item	Cấu hình
Flavor	2x4
Volume	20 GB
VPC	10.76.0.0/16
Subnet	10.76.0.4/24
Network Interface 1	10.76.0.3

- Một **server (VM) Palo Alto** được khởi tạo trên hệ thống **vMarketPlace** theo hướng dẫn bên dưới với cấu hình như sau:

Item	Cấu hình
Flavor	2x8
Volume	60 GB
VPC	10.76.0.0/16
Network Interface 1	10.76.255.4
Network Interface 2	10.76.0.4

Khởi tạo Palo Alto

Bước 1: Truy cập vào <https://marketplace.console.vngcloud.vn/>

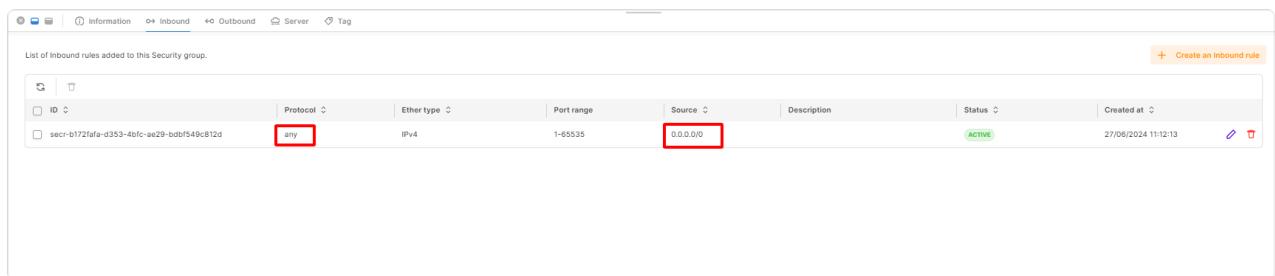
Bước 2: Tại màn hình chính, thực hiện tìm kiếm **Palo Alto**, tại dịch vụ **Palo Alto**, chọn **Launch**.

Bước 3: Lúc này, bạn cần thiết lập cấu hình cho **Palo Alto**. Cụ thể, bạn có thể chọn **Volume**, **IOPS**, **Network**, **Security Group** mong muốn. **Bạn cần lựa chọn VPC và Subnet giống với VPC và Subnet mà bạn lựa chọn sử dụng cho Cluster của bạn.** Ngoài ra bạn cũng cần chọn Một Server Group đã tồn tại hoặc chọn **Dedicated SOFT ANTI AFFINITY group** để chúng tôi tự động tạo một server group mới.

Bước 4: Tiến hành thanh toán như các tài nguyên bình thường trên VNG Cloud.

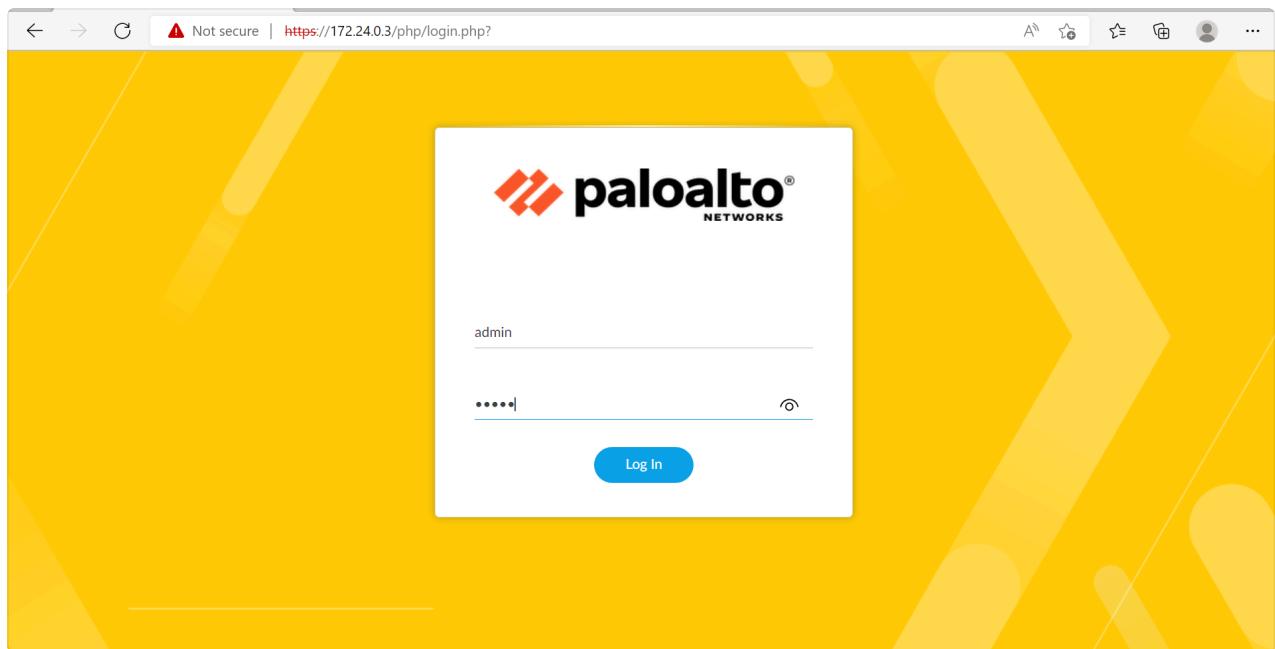
Cấu hình thông số cho Palo Alto

Bước 1: Sau khi khởi tạo Palo Alto từ vMarketPlace theo hướng dẫn bên trên, bạn có thể truy cập vào giao diện vServer tại [đây](#) để kiểm tra server chạy Palo Alto đã được khởi tạo xong chưa. **Tiếp theo, bạn mở rule Any trên Security Group cho server Palo Alto vừa tạo. Việc mở rule Any trên Security Group sẽ cho phép tất cả lưu lượng truy cập đến server Palo Alto.**



Bước 2: Sau khi server chạy Palo Alto được khởi tạo thành công. Để vào GUI của Palo Alto bạn cần có 1 vServer chạy Windows. Sau đó bạn truy cập vào bằng IP Internal Interface với tên đăng nhập và mật khẩu mặc định là: **admin/admin**

Lưu ý: Về phần Network của vServer Windows để truy cập vào GUI của Palo Alto. Bạn cần tạo cùng VPC và sử dụng subnet khác với subnet có priority là 1 khi khởi tạo Palo Alto



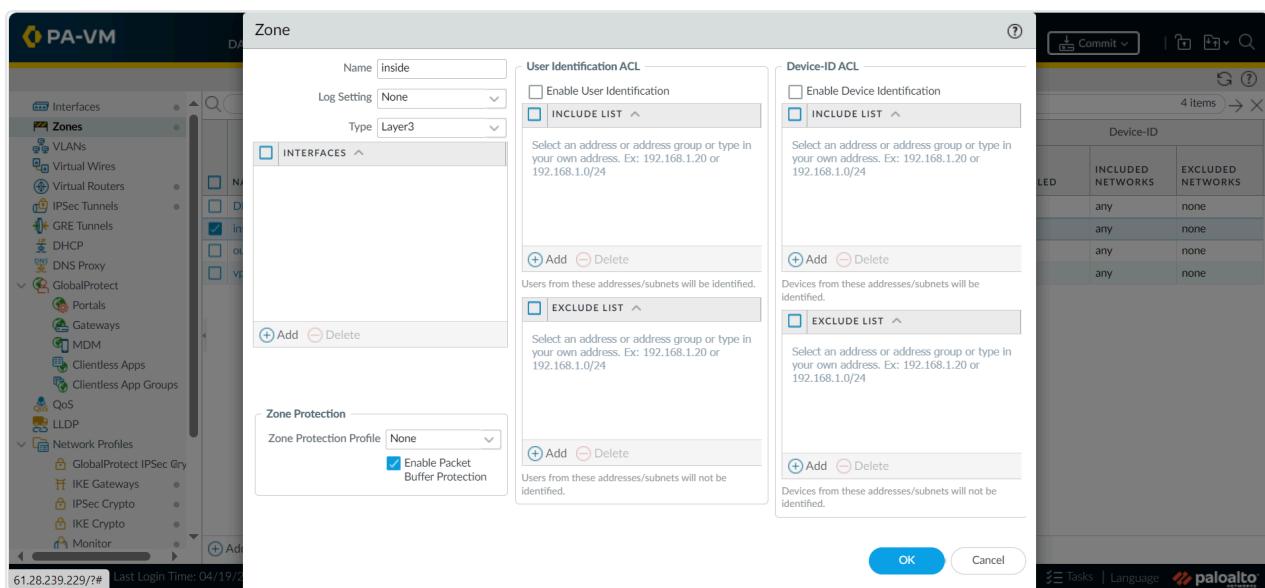
Bước 3: Sau khi đăng nhập xong, bạn cần thực hiện thay đổi mật khẩu lần đầu. Hãy nhập mật khẩu mới theo mong muốn của bạn.

Bước 4: Bạn cần tiến hành khởi tạo 1 Zone Inside và 1 Zone Outside the hướng dẫn bên dưới:

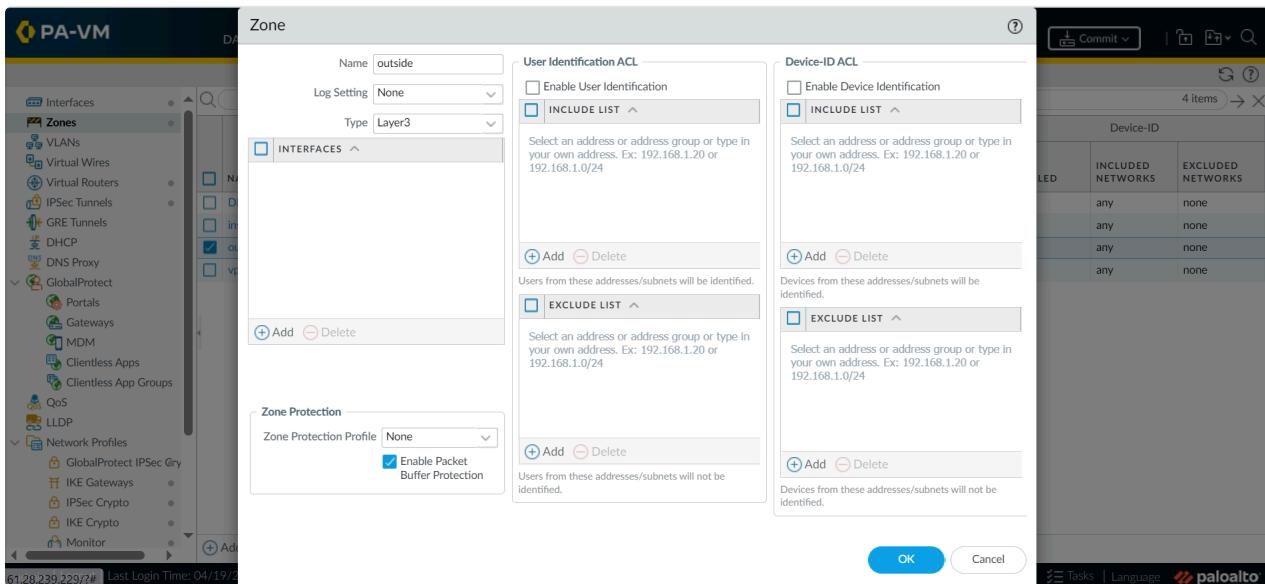
- Chọn bút Add

Name	Type	INTERFACE / VIRTUAL SYSTEMS	ZONE PROTECTION PROFILE	PACKET BUFFER PROTECTION	LOG SETTING	User-ID	Device-ID	INCLUDED NETWORKS	EXCLUDED NETWORKS	ENABLED	INCLUDED NETWORKS	EXCLUDED NETWORKS

- Đặt tên cho **Zone: Inside** sau đó chọn **OK**

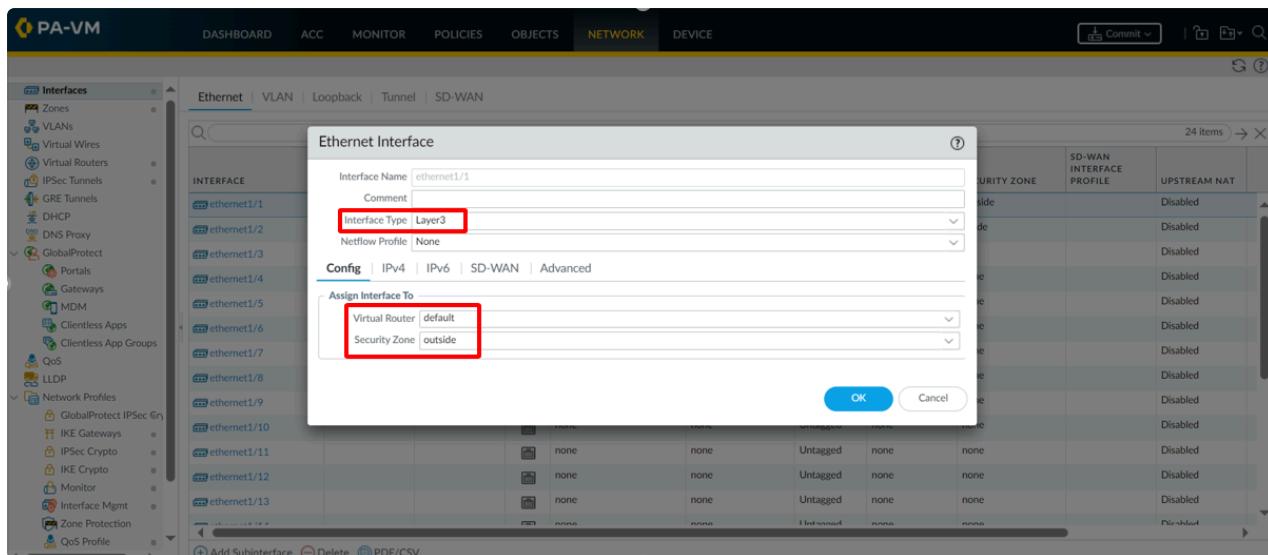


- Làm tương tự đối với Zone Outside

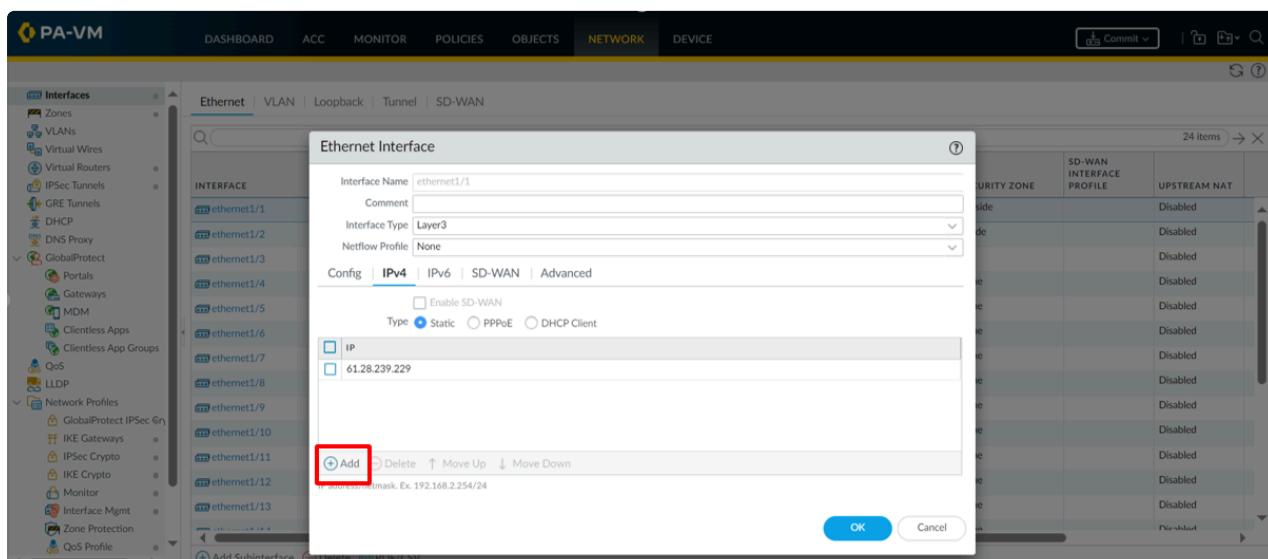


Bước 5: Cấu hình cho External Interface

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



- Chuyển sang Tab **IPv4** và chọn **Add** để nhập **Static IP** cho **External Interface**



- Để lấy thông tin IP này bạn vào phần **Network Interface** của **Palo Alto** để xem thông tin

Detail information
Details of your server while it is in use

INTERNAL INTERFACE

ID	VPC ID	Subnet ID	Fixed IP	Floating IP Association
net-in-fcc60993-3643-4447-a8f6-e19ff622dbc6	net-a4aeef856-7c16-4557-96c9-2851262dbebf	sub-e08ae230-4040-41d3-9d57-7b8e88fc302c	10.76.255.4	
net-in-fbf0a9bf-ec2f-4cc0-9169-6045a3befa7d	net-a4aeef856-7c16-4557-96c9-2851262dbebf	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

- Chuyển sang tab **Advanced**, ở phần **MTU** bạn cần chỉnh thành **1400**

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**

Bước 6: Thực hiện cấu hình tương tự cho các Internal Interface

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

- Tại tab **IPv4**: bạn tiến hành thiết lập **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

- Chuyển sang tab **Advanced**, ở phần **MTU** bạn chỉnh thành 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**

Bước 7: Tạo static route

- Vào phần Network → Virtual Routers → Chọn default → Chuyển sang mục Static Routes

PA-VM

DASHBOARD ACC MONITOR POLICIES OBJECTS NETWORK DEVICE

Virtual Router - default

Router Settings

Static Routes

Redistribution Profile

RIP OSPF OSPFv3 BGP Multicast

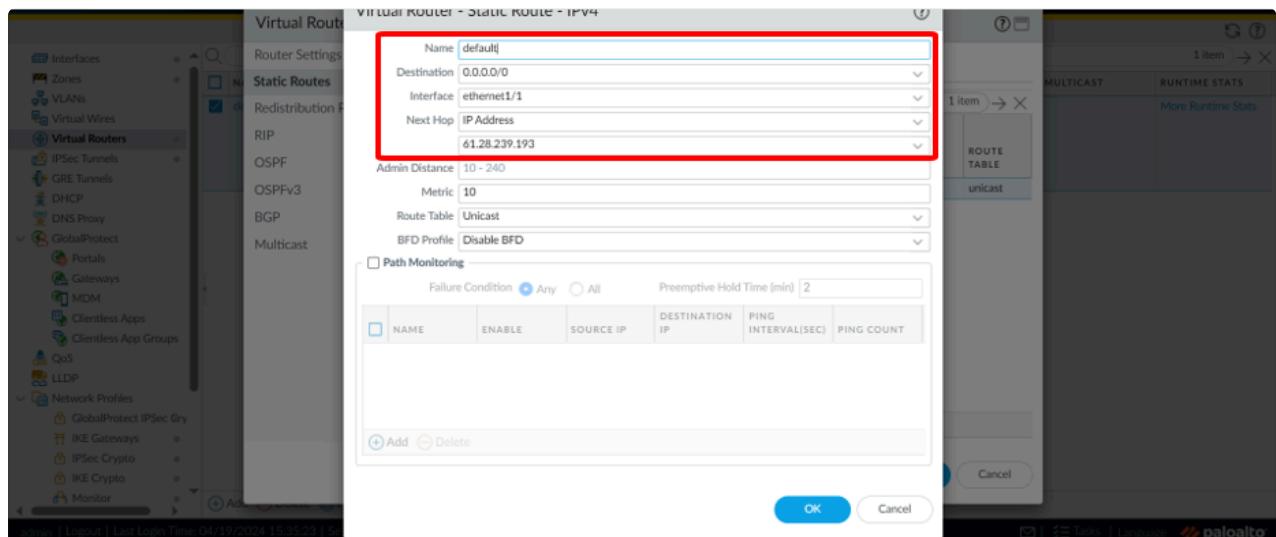
IPv4 | IPv6

	NAME	DESTINA...	INTERFACE	TYPE	VALUE	ADMIN DISTANCE	METRIC	BFD	ROUTE TABLE
0 items → X									

+ Add **Delete** **Clone**

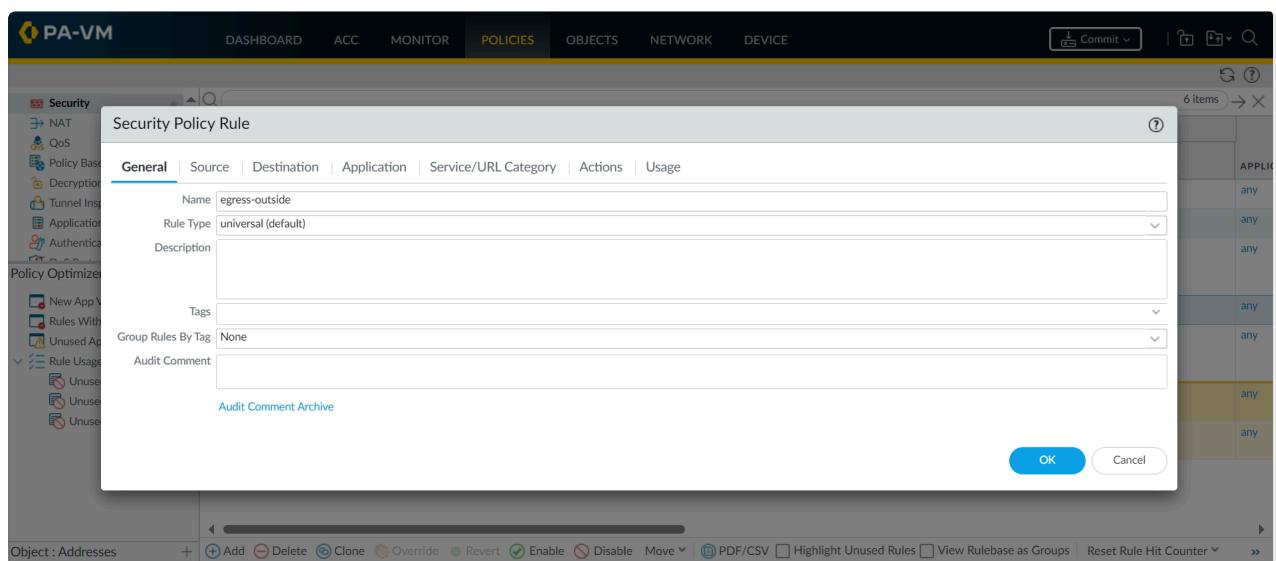
OK Cancel

- Thực hiện tạo 1 route như hình bên dưới

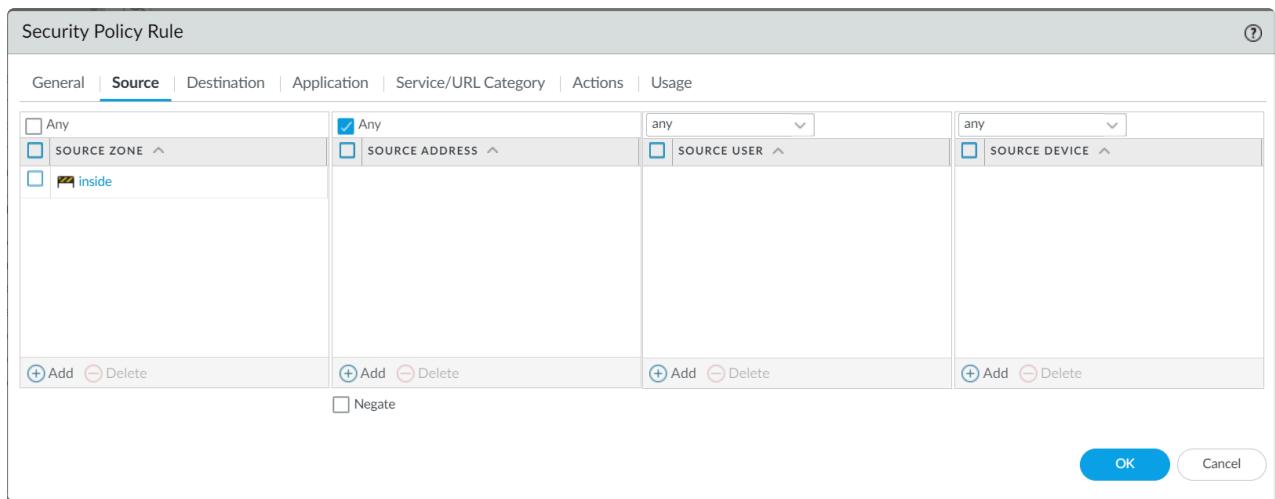


Bước 8: Tạo Security Policy Rule

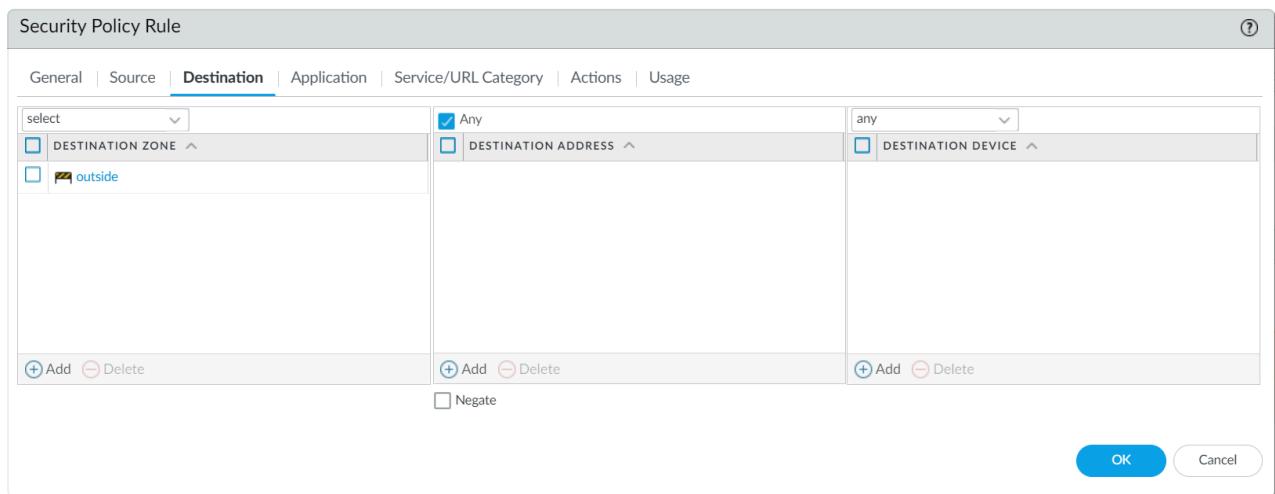
- Vào phần Policies → Security → Add
- Tại tab General, bạn cần đặt tên cho rule



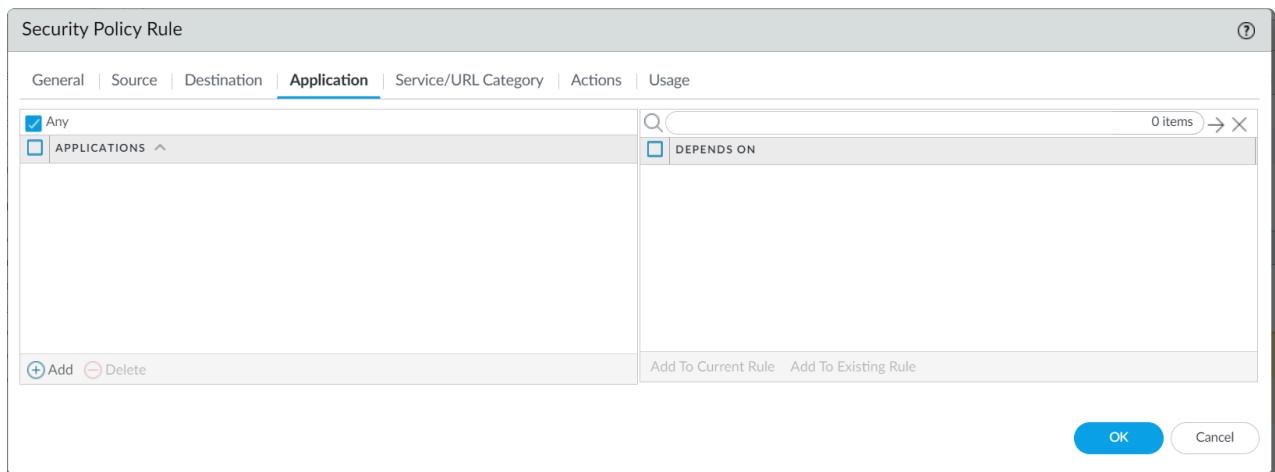
- Tại tab Source, thiết lập các thông tin như **Source Zone, Source Address, Source User, Source Device**



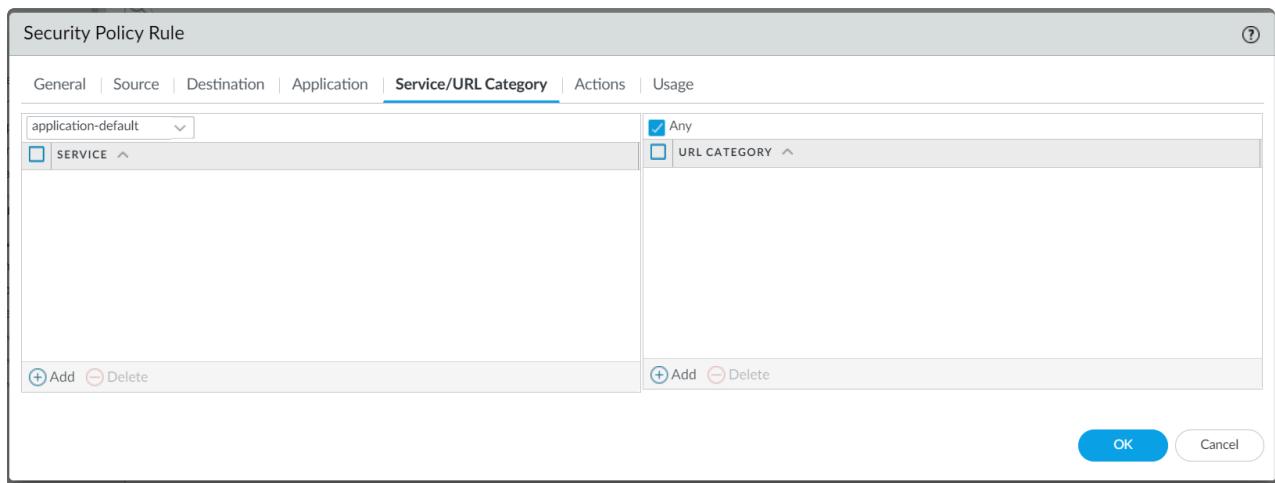
- Tại tab **Destination**, thiết lập các thông tin như **Destination Zone, Destination Address, Destination Device**



- Tại tab **Application**, thiết lập các thông tin như **Application, Depend On**



- Tại tab **Service/URL Category**, thiết lập các thông tin như **Service, URL Category**

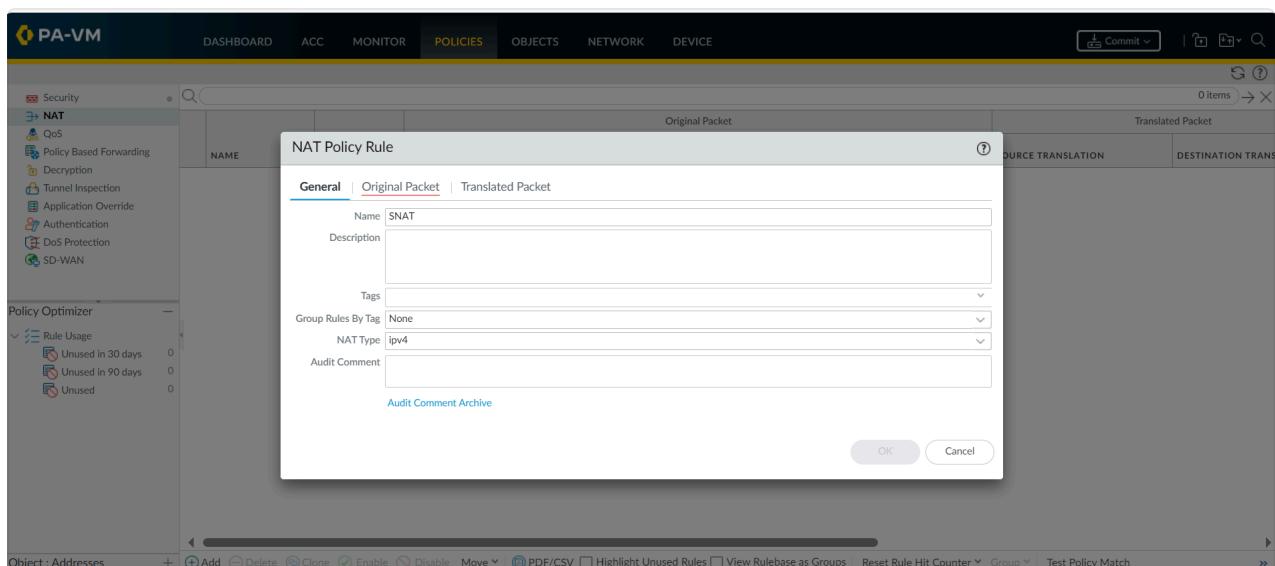


- Tại tab **Actions**, thiết lập các thông tin như **Action, Log, Profile, Other Settings**

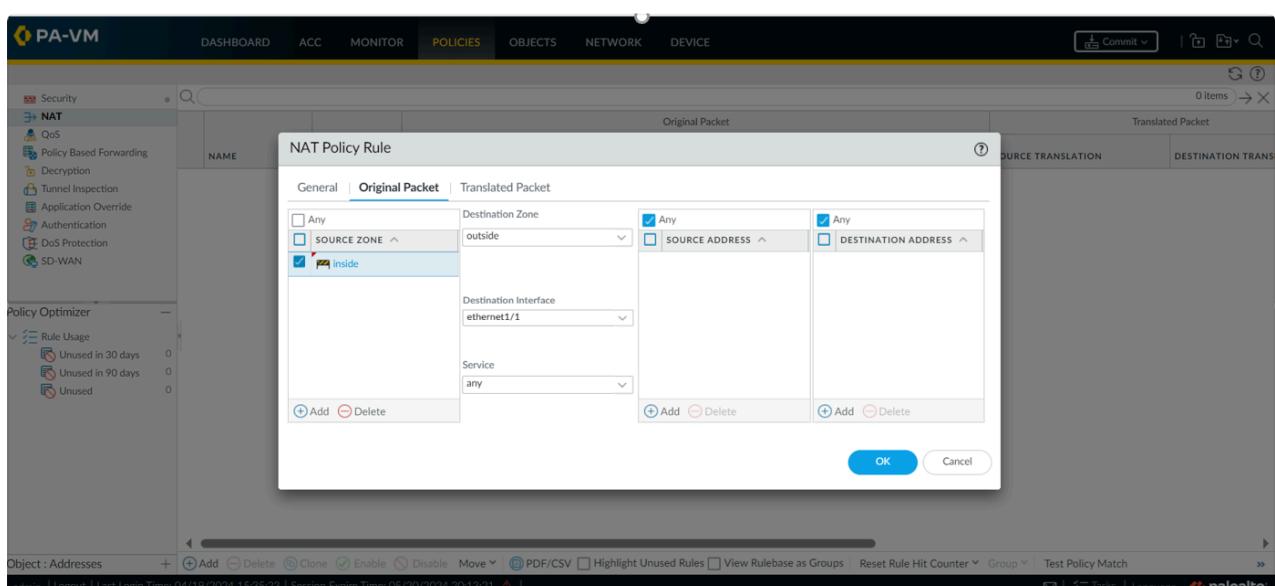
Bước 9: Tạo rule NAT để các VM có thể đi ra Internet

- Vào phần **Policies** → **NAT** → **Add**

- Tại tab **General** đặt tên cho **NAT rule**

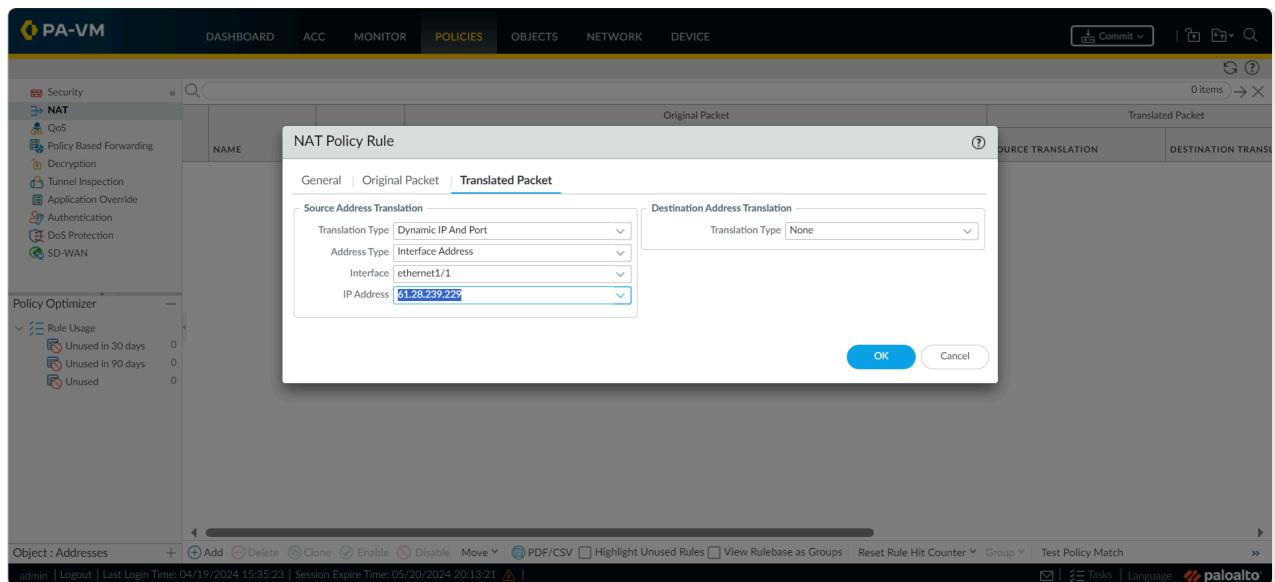


- Tại tab **Original Packet** chọn **Source Zone, Destination Zone, Destination Interface, Service, Source Address, Destination Address**

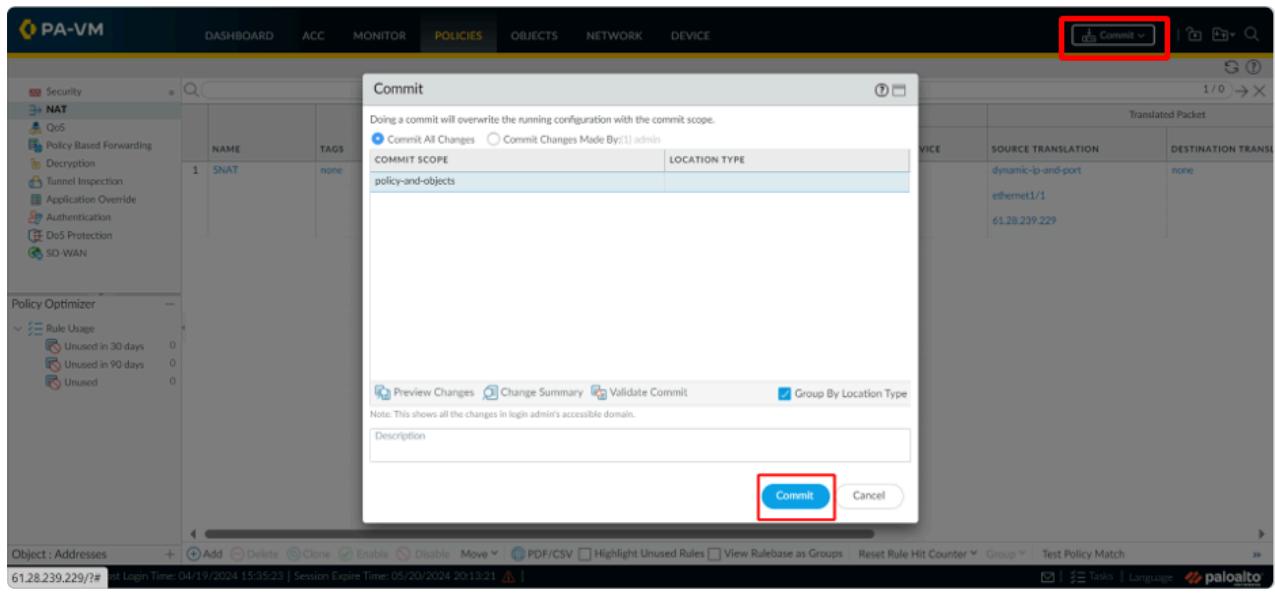


- Tạo tab **Translated Packet** thực hiện cấu hình như hình bên dưới

Lưu ý: Cần thay đổi **IP Address** thành địa chỉ **Static IP** mà bạn đã cấu hình ở bước 6



Bước 10: Tiến hành Commit



Khởi tạo Route Table

Sau khi Palo Alto được khởi tạo và cấu hình thành công, bạn cần tạo một Route table để kết nối tới các mạng khác nhau. Cụ thể thực hiện theo các bước sau để tạo Route table:

Bước 1: Truy cập vào <https://hcm-3.console.vngcloud.vn/vserver/network/route-table>

Bước 2: Tại thanh menu điều hướng, chọn Tab Network/ Route table.

Bước 3: Chọn Create Route table.

Bước 4: Nhập tên mô tả cho Route table. Tên Route table có thể bao gồm các chữ cái (a-z, A-Z, 0-9, '_', '-'). Độ dài dữ liệu đầu vào nằm trong khoảng từ 5 đến 50. Nó không được bao gồm khoảng trắng ở đầu hoặc ở cuối.

Bước 5: Chọn **VPC** cho Route table của bạn, nếu chưa có VPC cần tạo mới một VPC theo hướng dẫn tại [Trang VPC](#). **VPC sử dụng để thiết lập Route table phải là VPC được chọn sử dụng cho Palo Alto và Cluster của bạn.**

Bước 6: Chọn **Create** để tạo mới Route table.

Bước 7: Chọn tại Route table vừa tạo sau đó chọn **Edit Routes**.

Bước 8: Tại phần thêm mới **Route** hãy nhập vào các thông tin:

- Đối với Destination, hãy nhập **Destination CIDR là 0.0.0.0/0**
- Đối với Target, hãy nhập **Target CIDR là địa chỉ IP Network Interface 2 của Palo Alto.**

Ví dụ:

The screenshot shows the AWS Route Table management interface. At the top, there is a table with columns: Name, VPC, Status, Created at, and Action. One row is visible, showing 'rt-7f0108bb-8ab3-4e24-9e49-4ed513412229' as the name, 'net-a4aef856-7c16-4557-96c9-2851262dbefb' as the VPC, 'ACTIVE' as the status, '24/04/2024 14:03:11' as the creation date, and a delete icon in the Action column. Below this table is a section titled 'List of Route attached to this Route Table.' It contains a table with columns: Destination and Target. One row is shown with '0.0.0.0/0' as the Destination and '10.76.0.4' as the Target. A red box highlights the 'Target' value '10.76.0.4' with the text 'IP Network Interface 2' overlaid.

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	

List of Route attached to this Route Table.

Destination	Target
0.0.0.0/0	10.76.0.4

Kiểm tra kết nối

- Tiến hành ping 8.8.8.8 hoặc 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

Sử dụng hướng dẫn bên dưới để làm việc với Private Node group thông qua Pfsense

Điều kiện cần

Để có thể sử dụng Pfsense làm NAT Gateway cho Cluster trên hệ thống VKS, bạn cần có:

- Một **server (VM) Pfsense** được khởi tạo trên hệ thống **vMarketPlace** theo hướng dẫn bên dưới với cấu hình như sau:

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

Khởi tạo Pfsense

Bước 1: Truy cập vào <https://marketplace.console.vngcloud.vn/>

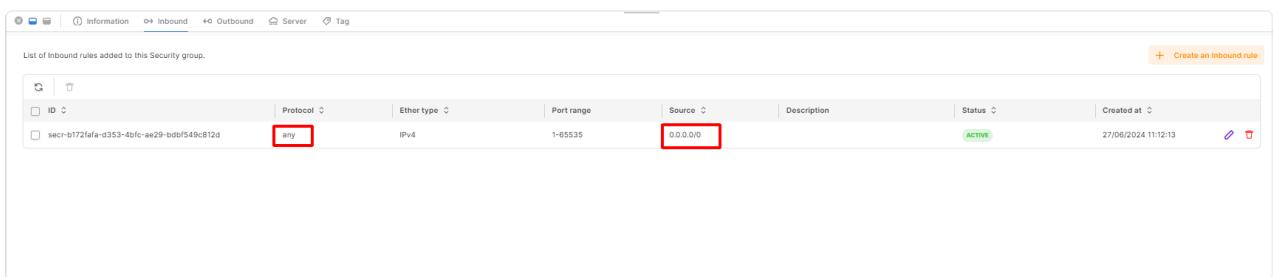
Bước 2: Tại màn hình chính, thực hiện tìm kiếm **Pfsense**, tại dịch vụ **Pfsense**, chọn **Launch**.

Bước 3: Lúc này, bạn cần thiết lập cấu hình cho **Pfsense**. Cụ thể, bạn có thể chọn **Volume**, **IOPS**, **Network**, **Security Group** mong muốn. **Bạn cần lựa chọn VPC và Subnet giống với VPC và Subnet mà bạn lựa chọn sử dụng cho Cluster của bạn.** Ngoài ra bạn cũng cần chọn Một Server Group đã tồn tại hoặc chọn **Dedicated SOFT ANTI AFFINITY group** để chúng tôi tự động tạo một server group mới.

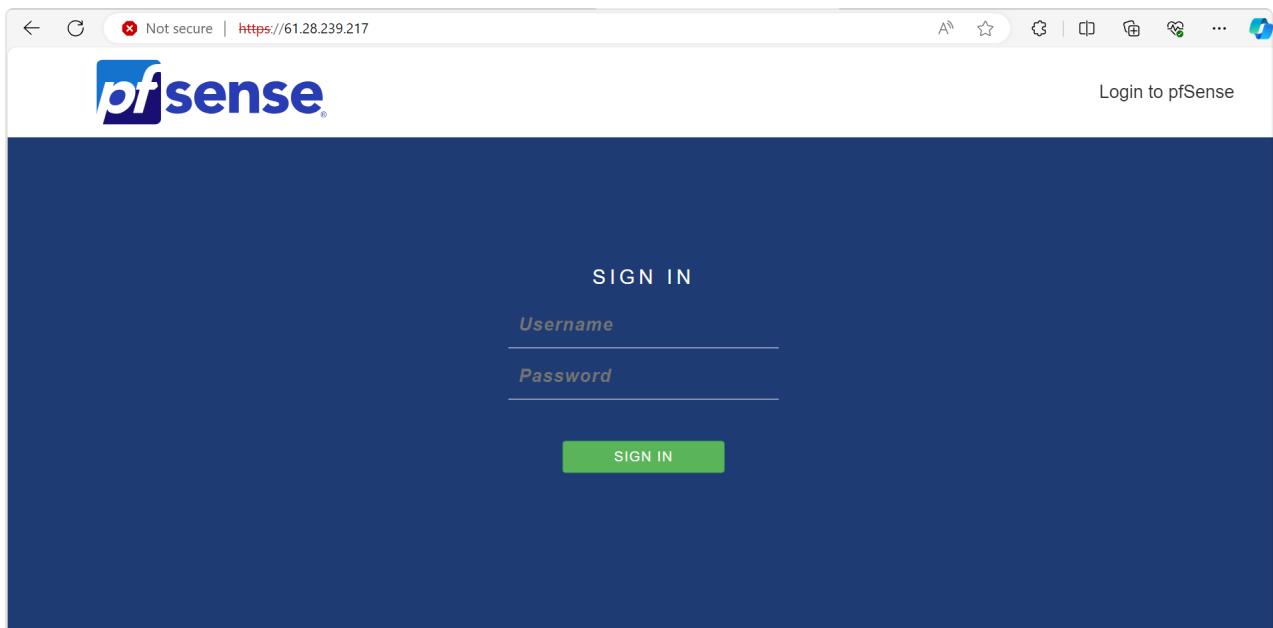
Bước 4: Tiến hành thanh toán như các tài nguyên bình thường trên VNG Cloud.

Cấu hình thông số cho Pfsense

Bước 1: Sau khi khởi tạo Pfsense từ vMarketPlace theo hướng dẫn bên trên, bạn có thể truy cập vào giao diện vServer tại [đây](#) để kiểm tra server chạy Pfsense đã được khởi tạo xong chưa. **Tiếp theo, bạn mở rule Any trên Security Group cho server Pfsense vừa tạo. Việc mở rule Any trên Security Group sẽ cho phép tất cả lưu lượng truy cập đến server Pfsense.**



Bước 2: Sau khi server chạy Pfsense được khởi tạo thành công. Để vào GUI của Pfsense, bạn cần sử dụng địa chỉ IP của External Interface đăng nhập với Tên đăng nhập và mật khẩu mặc định là **admin/pfsense**.



- Để lấy thông tin IP này, bạn vào phần **Network Interface** của **Pfsense** để xem thông tin

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-a4aeef856-7c16-4557-96c9-2851262dbebf	sub-e08ae230-4040-41d3-9d57-7b8e88fc302c	10.76.255.4	
net-in-fb0a9bf-ec2f-4cc0-9169-6045a3befa7d	net-a4aeef856-7c16-4557-96c9-2851262dbebf	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

Bước 3: Mở rule trên firewall

- Tiến hành **Add rule**

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.](#)

Firewall / Rules / WAN

Floating **WAN**

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.

Add Add Delete Toggle Copy Save Separator

- Bạn có thể mở rule như bên dưới để truy cập vào GUI bằng **External Interface**.

Chú ý:

- Bạn nên giới hạn lại Range IP được phép kết nối tới GUI Pfsense để hạn chế user được phép truy cập vào GUI Pfsense

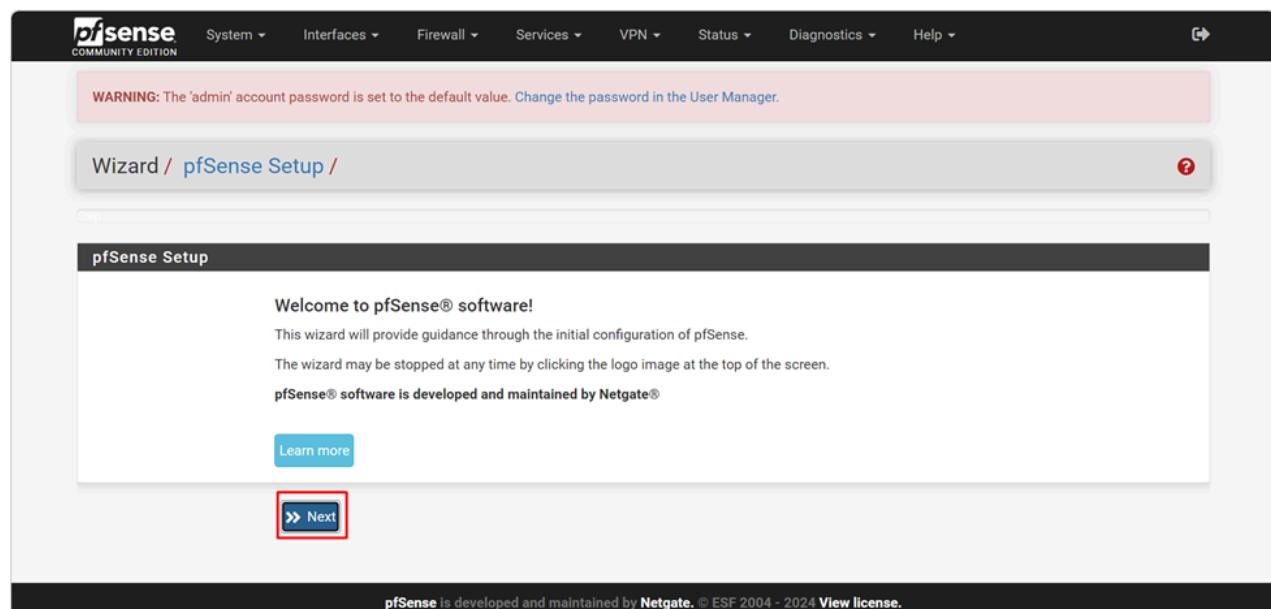
Firewall / Rules / Edit

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	

- Chọn **Save**
- Sau đó chọn **Apply change**

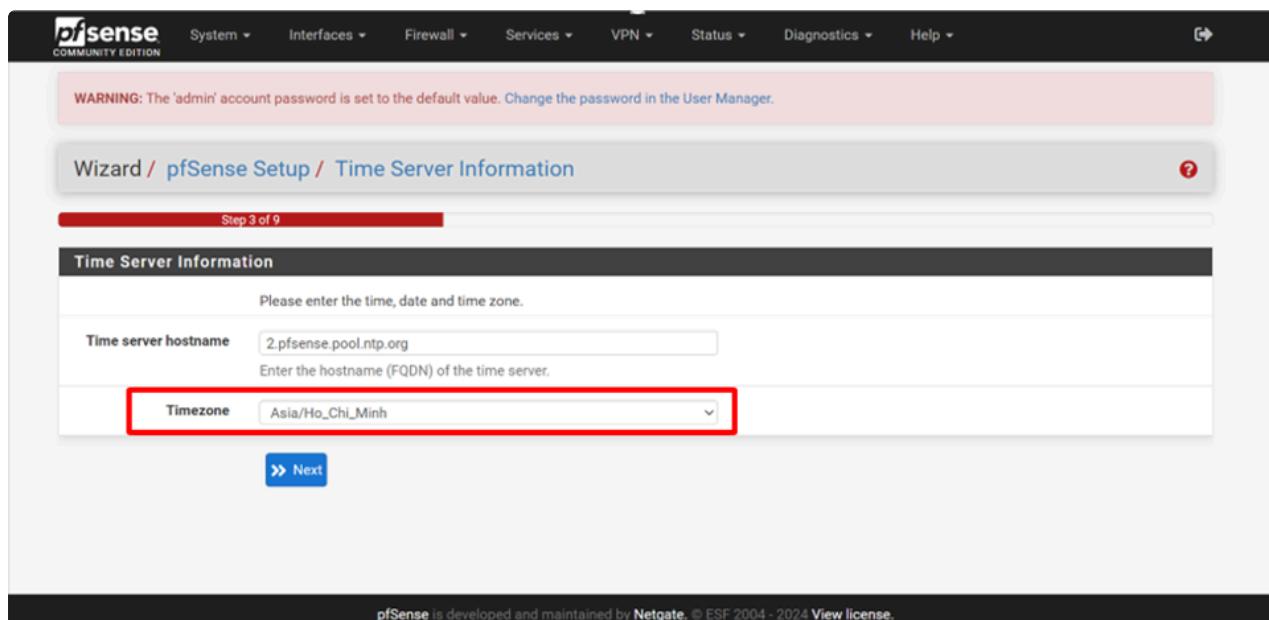
Bước 4: Tiến hành General Setup, bạn vui lòng thực hiện như bên dưới



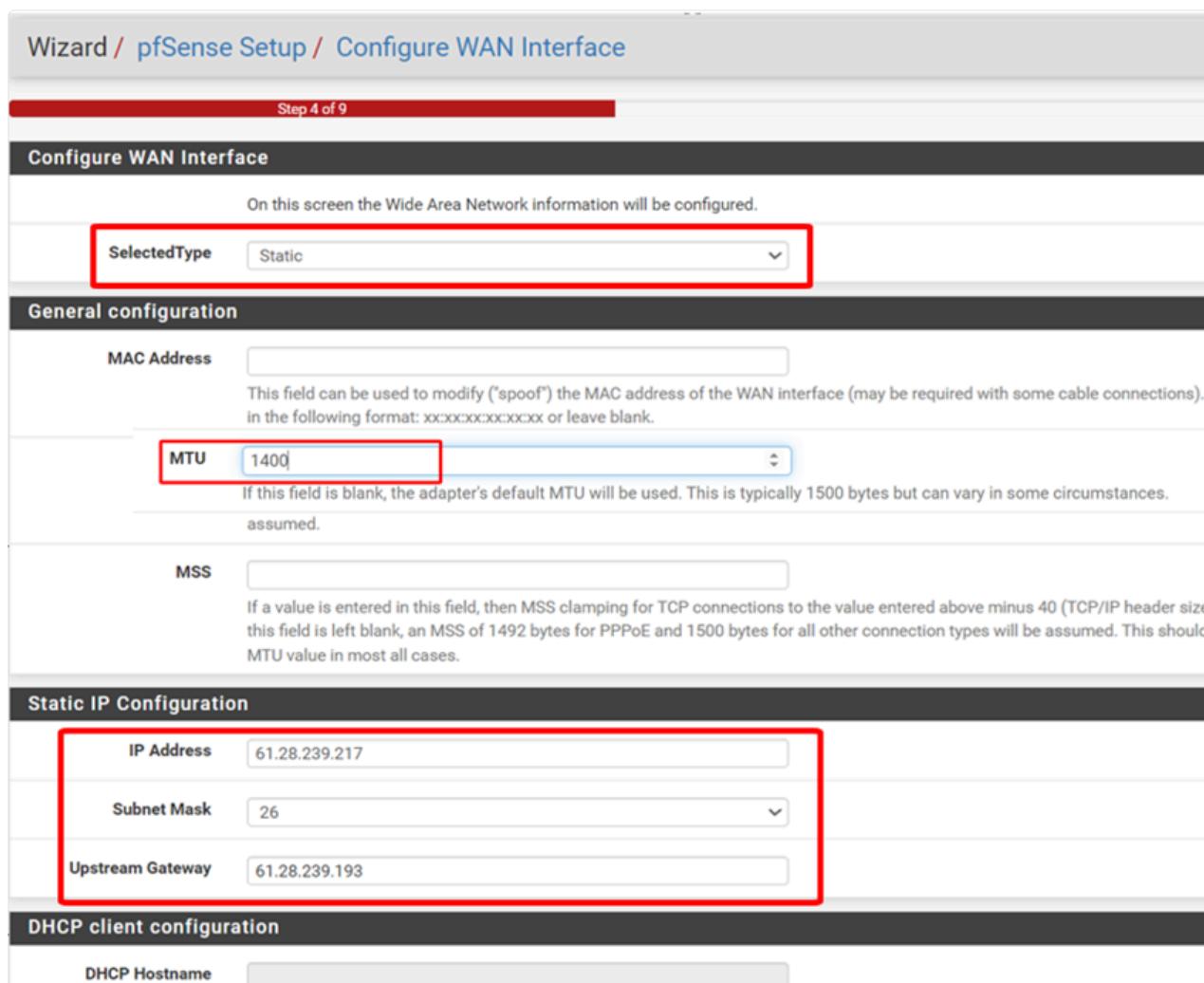
The screenshot shows the pfSense Setup Wizard at Step 1 of 9. At the top, there is a warning message: "WARNING: The 'admin' account password is set to the default value. Change the password in the User Manager." Below this, the title is "Wizard / pfSense Setup / Netgate® Global Support is available 24/7". A sub-header says "Step 1 of 9". The main content area is titled "Netgate® Global Support is available 24/7" and contains text about the support team's qualifications and availability. It also lists reasons why companies choose Netgate support. A "Learn more" button is present, and a "» Next" button is at the bottom.

This screenshot shows the "General Parameters" step of the pfSense Setup Wizard. It includes fields for Hostname (set to "pfSense"), Domain (set to "home.arpa"), and DNS server configuration. The "Primary DNS Server" field (containing "8.8.8.8") and the "Secondary DNS Server" field (containing "8.8.4.4") are both highlighted with a red box. An "Override DNS" checkbox is checked. A note at the bottom explains the behavior of the DNS resolver. A "» Next" button is at the bottom.

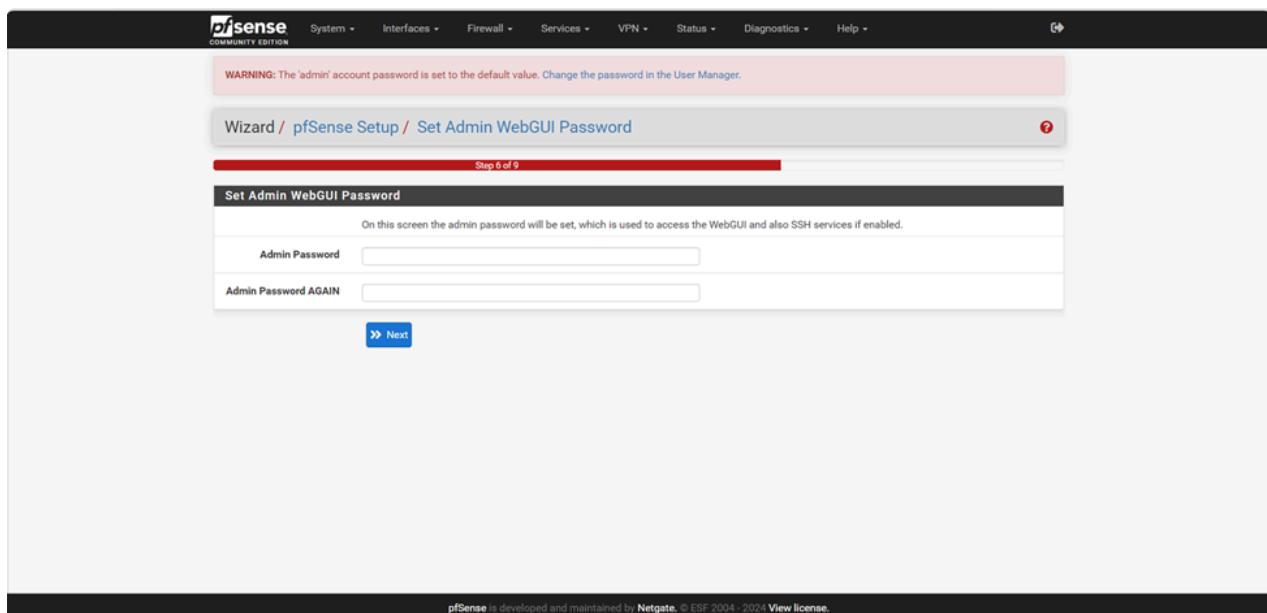
On this screen the general pfSense parameters will be set.	
Hostname	pfSense
Name of the firewall host, without domain part.	
Examples: pfsense, firewall, edgefw	
Domain	home.arpa
Domain name for the firewall.	
Examples: home.arpa, example.com	
Do not end the domain name with '.local' as the final part (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	



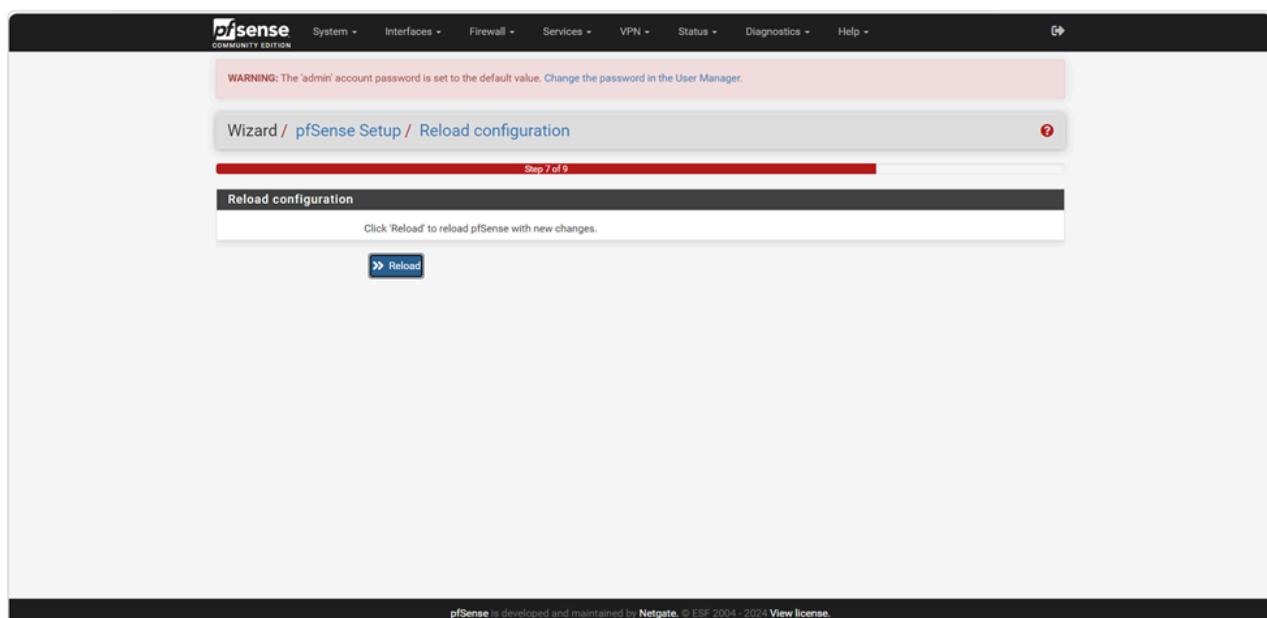
- Cấu hình cho **WAN Interface**



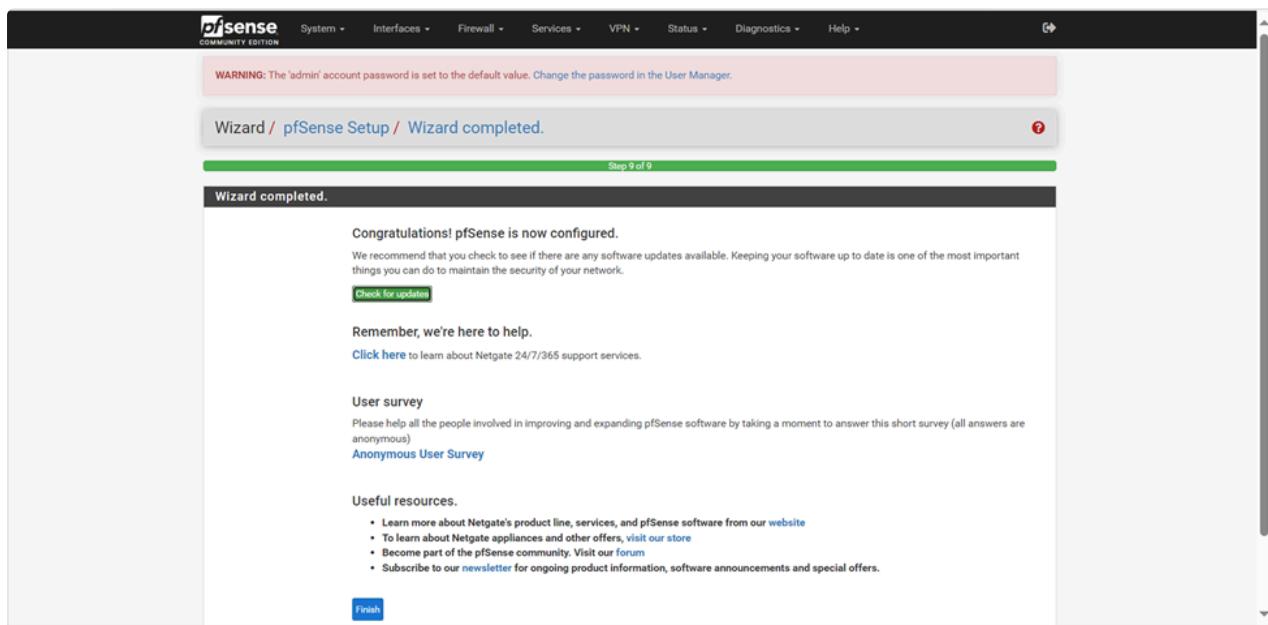
- Thay đổi **password** vào GUI



- Tiến hành **reload**



- Đã hoàn thành **General Setup**



Bước 5: Cấu hình Interface LAN

- Vào phần **Interfaces → Assignments** để gắn thêm **Interface LAN**

The screenshot shows the pfSense web interface. At the top, the navigation bar includes 'System', 'Interfaces', 'Firewall', 'Services', 'VPN', 'Status', 'Diagnostics', and 'Help'. The 'Interfaces' menu is open, with 'Assignments' highlighted by a red box. Below the menu, the status bar says 'WARNING: The 'admin' account password is set to the default value. Change the password in the User Manager.' The main content area shows 'Status / Dashboard' with a 'WAN' section. To the right, there's a 'Netgate Services And Support' sidebar with sections for 'Contract type' (Community Support, Community Support Only), 'NETGATE AND pfSense COMMUNITY SUPPORT RESOURCES', and a note about purchasing TAC support. The URL in the address bar is 'https://61.28.239.217/#'.

- Nhấn vào **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)

+ Add

Save

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.

- Sau đó nhấn vào **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

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](#).

- Vào phần **Interfaces → Assignments** để tiến hành **enable LAN Interface**

The screenshot shows the pfSense interface under the 'Interfaces' menu. The 'LAN' tab is active. It displays two network ports: 'WAN' (vtnet0) and 'LAN' (vtnet1). The 'Save' button is highlighted with a red box.

- Bạn thực hiện cấu hình như bên dưới

The screenshot shows the 'General Configuration' page for the LAN interface. The 'Enable' checkbox is checked and highlighted with a red box. Other configuration options shown include:

- Description:** LAN
- IPv4 Configuration Type:** Static IPv4 (highlighted with a red box)
- IPv6 Configuration Type:** None
- MAC Address:** XX:XX:XX:XX:XX:XX
- MTU:** 1400 (highlighted with a red box)
- MSS:** [empty]

- Cấu hình IP cho LAN

The screenshot shows the 'Static IPv4 Configuration' page. The 'IPv4 Address' field contains '10.3.0.3' and is highlighted with a red box. The 'Subnet Mask' field contains '/24' and is also highlighted with a red box. A green button labeled '+ Add a new gateway' is highlighted with a red box.

- Sau đó tiến hành **Add a new gateway**: tiến hành nhập **Gateway cho 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"/>	

- Để lấy thông tin IP này, bạn vào mục **Network Interface** của server Pfsense để xem thông tin:

Detail information
Details of your server while it is in use

INTERNAL INTERFACE				
<input type="checkbox"/> ID	VPC ID	Subnet ID	Fixed IP	Floating IP Association
<input type="checkbox"/> net-in-fcc60093-3643-4447-a0f6-e19ff022dbc6	net-a4aef856-7c16-4557-96c9-2851262dbefb	sub-e08ae230-4040-41d3-9d57-7b8e88fc302c	10.76.255.4	
<input type="checkbox"/> net-in-fbf0a8bf-ec2f-4cc0-9169-6045a3bea7d	net-a4aef856-7c16-4557-96c9-2851262dbefb	sub-3add2e0e-abcb-4ccb-89de-5f77a8eb4581	10.76.0.4	

- Tiến hành **Save** lại

Static IPv4 Configuration

IPv4 Address	<input type="text" value="10.3.0.3"/>	/ 24
IPv4 Upstream gateway	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>		

Bước 6: Xem lại thông tin cấu hình

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 sections for Contract type (Community Support, Community Support Only), NETGATE AND pfSense COMMUNITY SUPPORT RESOURCES, and a note about purchasing support. It also lists links for Upgrade Your Support, Community Support Resources, Netgate Global Support FAQ, Official pfSense Training by Netgate, and Netgate Professional Services. A red box highlights the 'Interfaces' section, which shows two entries: WAN (10Gbase-T <full-duplex>, IP 61.28.239.217) and LAN (10Gbase-T <full-duplex>, IP 10.3.0.3).

Bước 7: Mở rule đi ra Internet cho interface LAN

The screenshot shows the pfSense Firewall / Rules / LAN page. At the top, there are tabs for Floating, WAN, and LAN, with LAN selected. Below is a table titled 'Rules (Drag to Change Order)' with columns for States, Protocol, Source, Port, Destination, Port, Gateway, Queue, Schedule, Description, and Actions. Three rules are listed: one for Anti-Lockout Rule (0/0 B, */*, LAN Address, 443, 80, */*, *), and two default allow rules for LAN (0/76 B and 0/0 B, IPv4/IPv6, LAN net, *, *, *, *, none, Default allow LAN to any rule and Default allow LAN IPv6 to any rule). At the bottom, there are buttons for Add, Delete, Toggle, Copy, Save, and Separator. The 'Add' button is highlighted with a red box.

- Tại source bạn chọn dải IP được phép đi ra 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					

Bước 8: Cấu hình NAT để các vServer có thể đi ra được Internet

- Vào mục Firewall → NAT

The screenshot shows the pfSense Firewall interface. The top navigation bar includes 'System', 'Interfaces', 'Firewall', 'Services', 'VPN', 'Status', 'Diagnostics', and 'Help'. A warning message 'WARNING: The "admin" account password is set to the default value. Please change the password in the User Manager.' is displayed. The 'Firewall' menu is open, showing options like 'Aliases', 'NAT' (which is highlighted with a red box), 'Rules', 'Schedules', 'Traffic Shaper', and 'Virtual IPs'. Below the menu, there's a 'Status / Dashboard' section with 'System Information' and a 'Netgate Services And Support' sidebar. The 'System Information' table contains details such as Name (pfSense.home.arpa), User (admin@116.110.40.116), System (KVM Guest), BIOS (Vendor: SeaBIOS, Version: 1.11.0-2.e17, Release Date: Tue Apr 1 2014), Version (2.7.0-RELEASE (amd64)), CPU Type (Intel(R) Xeon(R) Silver 4310 CPU @ 2.10GHz), Hardware crypto (Inactive), Kernel PTI (Disabled), MDS Mitigation (Inactive), Uptime (00 Hour 08 Minutes 25 Seconds), and Current date/time (Sun Apr 21 12:43:44 +07 2024). The 'Netgate Services And Support' sidebar provides information about Netgate support contracts and resources.

- Chọn mode NAT sau đó tiến hành cấu hình NAT

Firewall / NAT / Outbound

Port Forward 1:1 **Outbound** NPt

Outbound NAT Mode

Mode	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
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

- Nhấn vào **Add** để thêm rule

Mappings

Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description	Actions
									Add

Automatic Rules

Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description
✓ 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

i

- Chọn **source, destination NAT**

Firewall / NAT / Outbound / Edit

Edit Advanced Outbound NAT Entry

Disabled	<input type="checkbox"/> Disable this rule						
Do not NAT	<input type="checkbox"/> 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	Type	Source network for the outbound NAT mapping.	Port or Range
Destination	Any		/	24	Type	Destination network for the outbound NAT mapping.	Port or Range
<input type="checkbox"/> Not Invert the sense of the destination match.							

Khởi tạo Route Table

Sau khi Pfsense được khởi tạo và cấu hình thành công, bạn cần tạo một Route table để kết nối tới các mạng khác nhau. Cụ thể thực hiện theo các bước sau để tạo Route table:

Bước 1: Truy cập vào <https://hcm-3.console.vngcloud.vn/vserver/network/route-table>

Bước 2: Tại thanh menu điều hướng, chọn **Tab Network/ Route table**.

Bước 3: Chọn **Create Route table**.

Bước 4: Nhập tên mô tả cho Route table. Tên Route table có thể bao gồm các chữ cái (a-z, A-Z, 0-9, '_', '-'). Độ dài dữ liệu đầu vào nằm trong khoảng từ 5 đến 50. Nó không được bao gồm khoảng trắng ở đầu hoặc ở cuối.

Bước 5: Chọn **VPC** cho Route table của bạn, nếu chưa có VPC cần tạo mới một VPC theo hướng dẫn tại [Trang VPC](#). **VPC sử dụng để thiết lập Route table phải là VPC được chọn sử dụng cho Pfsense và Cluster của bạn.**

Bước 6: Chọn **Create** để tạo mới Route table.

Bước 7: Chọn tại Route table vừa tạo sau đó chọn **Edit Routes**.

Bước 8: Tại phần thêm mới **Route** hãy nhập vào các thông tin:

- Đối với Destination, hãy nhập **Destination CIDR là 0.0.0.0/0**
- Đối với Target, hãy nhập **Target CIDR là địa chỉ IP Network Interface 2 của Pfsense**.

Ví dụ:

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

List of Route attached to this Route Table.

Destination *	Target *
0.0.0.0/0	172.24.0.3

Add a route

Kiểm tra kết nối

Tiến hành ping google.com hoặc 8.8.8.8 để kiểm tra

- Trước khi **Enable NAT** server không ra được 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
```

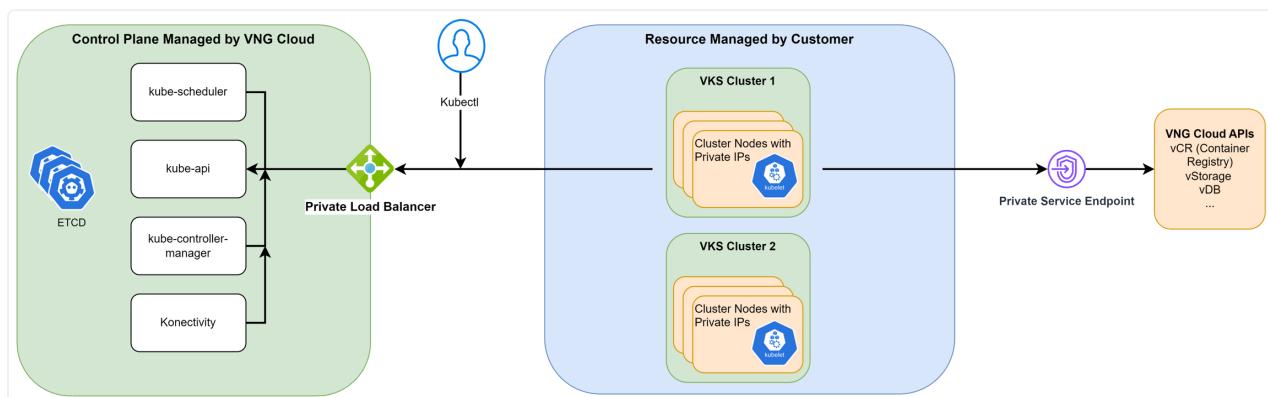
- Sau khi **cấu hình NAT** tiến hành ping 8.8.8.8 để kiểm tra

```
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
```

Khởi tạo một Private Cluster

Trước đây, các public cluster trên VKS đang sử dụng địa chỉ Public IP để giao tiếp giữa nodes và control plane. Để nâng cao bảo mật cho cluster của bạn, chúng tôi đã cho ra mắt mô hình private cluster. Tính năng Private Cluster giúp cho cụm K8S của bạn được bảo mật nhất có thể, mọi kết nối hoàn toàn là private từ kết nối giữa nodes tới control plane, kết nối từ client tới control plane, hay kết nối từ nodes tới các sản phẩm dịch vụ khác trong VNG Cloud như: vStorage, vCR, vMonitor, VNGCloud APIs,...Private Cluster là lựa chọn lý tưởng cho **các dịch vụ yêu cầu kiểm soát truy cập chặt chẽ, đảm bảo tuân thủ các quy định về bảo mật và quyền riêng tư dữ liệu.**

Model



Thành phần

- **Control plane:** Được quản lý bởi VNG Cloud, chịu trách nhiệm điều phối và quản lý toàn bộ cluster.
- **Nodes:** Các Node trong Cluster khi được tạo ra sẽ chỉ có internal IP và không thể đi ra public internet. Nếu muốn node truy cập được internet, bạn cần sử dụng NAT Gateway. Chi tiết tham khảo thêm tại [đây](#).
- **Private Load Balancer:** Được quản lý bởi VNG Cloud, chịu trách nhiệm giúp các Private Node giao tiếp với Control Plane.
- **Private Service Endpoint:** Khi bạn tạo một private cluster, hệ thống tự động tạo 4 endpoints giúp kết nối với các dịch vụ khác trên VNG Cloud bao gồm:
 - **Endpoint** để kết nối tới dịch vụ **IAM** (Endpoint Name: vks-iam-endpoint-...)
 - **Endpoint** để kết nối tới dịch vụ **vCR** (Endpoint Name: vks-vcr-endpoint-...)
 - **Endpoint** để kết nối tới dịch vụ **vServer** (Endpoint Name: vks-vserver-endpoint-...)

- Endpoint để kết nối tới dịch vụ vStorage (Endpoint Name: vks-vstorage-endpoint-...)

Bạn có thể xem thông tin 4 private service endpoint thông qua portal vServer theo đường dẫn tại [đây](#).

Name	Status	Description	VPC	Endpoint IP	Service Name	Action
eng-bcc284ec-b5f2-4e66-a8b4-9f6e296cb777 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-b390-4454-8126-3b038fc259f9 demo_vpc	10.7.8.6	vcr	⋮
eng-5d58eaf-e75f-4e07-94c7-5b7f8e9f752d 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-b390-4454-8126-3b038fc259f9 demo_vpc	10.7.8.5	vserver.hcm03	⋮
eng-757076-6ba2-4332-bbec-ef806884deb9 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-b390-4454-8126-3b038fc259f9 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-b390-4454-8126-3b038fc259f9 demo_vpc	10.7.8.3	iam	⋮

(i) Warning:

- Không xóa Private Service Endpoint:** Để đảm bảo hoạt động ổn định của cluster, bạn không nên xóa 4 service endpoint đã được tạo sẵn. Nếu vô tình xóa hoặc chỉnh sửa 4 endpoint này, trong vòng tối đa 5 phút, hệ thống sẽ tự động tạo lại nhưng có thể gây gián đoạn đến các dịch vụ đang chạy. Lúc này, do service endpoint tạo lại có thể đã thay đổi Endpoint IP so với ban đầu nên để cluster hoạt động được, bạn cần thực hiện thêm Endpoint IP một cách thủ công cho những server đang chạy trước đó thông qua câu lệnh:

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

Ví dụ, nếu bạn xóa private service endpoint của vCR thì bạn cần add host qua lệnh:

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

- Tái sử dụng Private Service Endpoint:** Các service endpoint có thể được nhiều private cluster cùng sử dụng. Khi các private cluster chung VPC thì chúng tôi sẽ tái sử dụng chúng cho các cluster này.
- Xóa Private Service Endpoint tự động:** Khi bạn xóa cluster, nếu không còn cluster nào tái sử dụng các service endpoint này, hệ thống sẽ tự động xóa chúng.

- **Chi phí sử dụng Private Service Endpoint:** Việc sử dụng private cluster sẽ phát sinh thêm chi phí cho 4 private service endpoint, nhưng nó mang lại nhiều lợi ích về bảo mật cho dự án của bạn. Bạn hãy cân nhắc kỹ lưỡng các yếu tố để đưa ra quyết định sử dụng public hay private cho cluster của mình.

Điều kiện cần

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn tại [đây](#).
- Có ít nhất 1 **SSH** key đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).
- Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.

Khởi tạo Cluster

Cluster trong Kubernetes là một tập hợp gồm một hoặc nhiều máy ảo (VM) được kết nối lại với nhau để chạy các ứng dụng được đóng gói dạng container. Cluster cung cấp một môi trường thống nhất để triển khai, quản lý và vận hành các container trên quy mô lớn.

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn, bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Cluster và Node Group của bạn tại Cluster Configuration, Default Node Group Configuration, Plugin.

Bước 5: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Warning:

- Một Cluster có thể có **nhiều Node Group**, mỗi Node Group có thể hoạt động ở mode Public/ Private tùy theo nhu cầu của bạn.
- Do Cluster của bạn được khởi tạo ở chế độ **Private**, để có thể truy cập vào **kube-api** của Control Plane thì bạn cần **đang trong VPC** mà bạn chọn sử dụng cho Cluster của bạn.

Kết nối và kiểm tra thông tin Cluster vừa tạo

Sau khi Cluster được khởi tạo thành công, bạn có thể thực hiện kết nối và kiểm tra thông tin Cluster vừa tạo theo các bước:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/k8s-cluster>

Bước 2: Danh sách Cluster được hiển thị, chọn biểu tượng **Download** và chọn **Download config file** để thực hiện tải xuống file kubeconfig. File này sẽ giúp bạn có toàn quyền truy

cập vào Cluster của bạn.

Bước 3: Đổi tên file này thành config và lưu nó vào thư mục `~/.kube/config`

Bước 4: Do Cluster của bạn được khởi tạo ở chế độ Private, để có thể truy cập vào kube-api, bạn cần đứng trong VPC mà bạn đã chọn sử dụng cho Cluster của bạn. Ví dụ, khi bạn không đứng ở VPC và thực hiện get nodes, kết quả sẽ hiển thị như sau:

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

Trong ví dụ bên dưới tôi sẽ đứng tại một server có VPC cùng với VPC được sử dụng cho Cluster. Bạn có thể thực hiện SSH vào server theo hướng dẫn tại [đây](#). Sau khi SSH vào server, hãy cài đặt kubectl theo hướng dẫn tại [đây](#).

- Ví dụ, tôi đang sử dụng một server linux để thực hiện get nodes, tôi có thể cài đặt kubectl qua lệnh:

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

- Sau đó, tôi kiểm tra kubectl qua lệnh:

```
kubectl version
```

- Tạo thư mục `.kube` qua lệnh:

```
mkdir -p .kube
```

- Sau đó, bạn hãy nhập file kubeconfig qua lệnh:

```
vim .kube/config
```

- Sau đó, bạn nhập `:wq` để lưu file kubeconfig và thoát vim.
- Chạy câu lệnh sau đây để kiểm tra **cluster**

```
kubectl get svc
```

- Bạn sẽ thấy kết quả trả về tương tự sau:

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

- Chạy câu lệnh sau đây để kiểm tra **node**

```
kubectl get nodes
```

- Nếu kết quả trả về như bên dưới tức là bạn Cluster của bạn được khởi tạo thành công với 1 node như bên dưới.

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

Sử dụng Docker để Pull/Push image

Do Private Cluster chỉ có thể kết nối private tới hệ thống vContainer Registry (vCR) và không thể kết nối ra các Container Registry khác ngoài internet, nên bạn cần pull/ push image về vCR để sử dụng theo hướng dẫn sau đây:

Bước 1: Cài đặt Docker

- Thực hiện cài đặt docker theo hướng dẫn tại [đây](#).

Bước 2: Khởi tạo Public Repository và Repository User trên vContainer Registry Portal:

- Đăng nhập portal vCR tại đường dẫn: <https://vcr.console.vngcloud.vn/list>
- Thực hiện khởi tạo Repository và Repository User theo hướng dẫn tại [đây](#). Ví dụ trong hình dưới, tôi đã khởi tạo demo_repo với demo_user có thể pull/ push image:

The screenshot shows the VNG Cloud Repository interface. At the top, there are tabs for Container Registry, Repository, Repository user, and Resource Billing. The Repository tab is selected. Below the tabs, there's a search bar and a 'Create a repository' button. The main area displays two sections: 'Repositories' (containing 4 items) and 'Images' (containing 4 items). A red box highlights the first repository entry in the 'Repositories' section, which is named 'repo_53461-repo_demo' and has a PUBLIC access level.

This screenshot shows the detailed view of the repository 'repo_53461-repo_demo'. The top navigation bar includes links for 'Images', 'Repository User' (which is highlighted), and 'History'. The 'Repository User' section lists a single user: '53461-user_demo' with 'Status' set to 'ACTIVE' and 'Permission' set to 'PULL AND PUSH'. A red box highlights this user entry.

Warning:

- Nếu bạn mong muốn khởi tạo Private Repository, để pull image từ Private Repository, bạn cần tạo secret key theo hướng dẫn tại [đây](#).

Bước 3: Thực hiện pull image nginx về theo lệnh:

```
docker pull nginx:latest
```

Bước 4: Thực hiện login vào vCR qua lệnh:

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

- Ví dụ, câu lệnh dưới tôi sử dụng để login vào repo demo:

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

Bước 5: Gán tag cho image nginx

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

- Ví dụ, câu lệnh dưới tôi sử dụng để gán tag cho image nginx:

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

Bước 6: Thực hiện push image lên repo qua lệnh:

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

- Ví dụ, câu lệnh dưới tôi sử dụng để push image lên demo_repo:

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

Deploy một Workload

Sau đây là hướng dẫn để bạn deploy service nginx và expose service này qua Network Load Balancer

Bước 1: Thực hiện tạo file nginx-service-lb4.yaml qua lệnh:

```
vi nginx.yaml
```

Sau đó, bạn hãy nhập nội dung cho file này như sau: bạn cần thay image bằng đường dẫn image lưu trên vCR mà bạn đã push ở bước bên trên:

```

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

```

- Nhập :**wq** để lưu tệp tin này.
- Deploy Deployment này bằng lệnh:

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

Bước 2: Kiểm tra thông tin Deployment, Service trước khi expose ra Internet.

- Chạy câu lệnh sau đây để kiểm tra **Deployment**

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

- Nếu kết quả trả về như bên dưới tức là bạn đã deploy service nginx thành công.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)		
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP		
service/nginx-service	LoadBalancer	10.96.81.236	116.118.88.236	80:32333/		
NAME		READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS
deployment.apps/nginx-app		1/1	1	1	3m32s	nginx
NAME		READY	STATUS	RESTARTS	AGE	IP
pod/nginx-app-56bbc8fdd8-4pz68		1/1	Running	0	3m32s	172.16.4.2

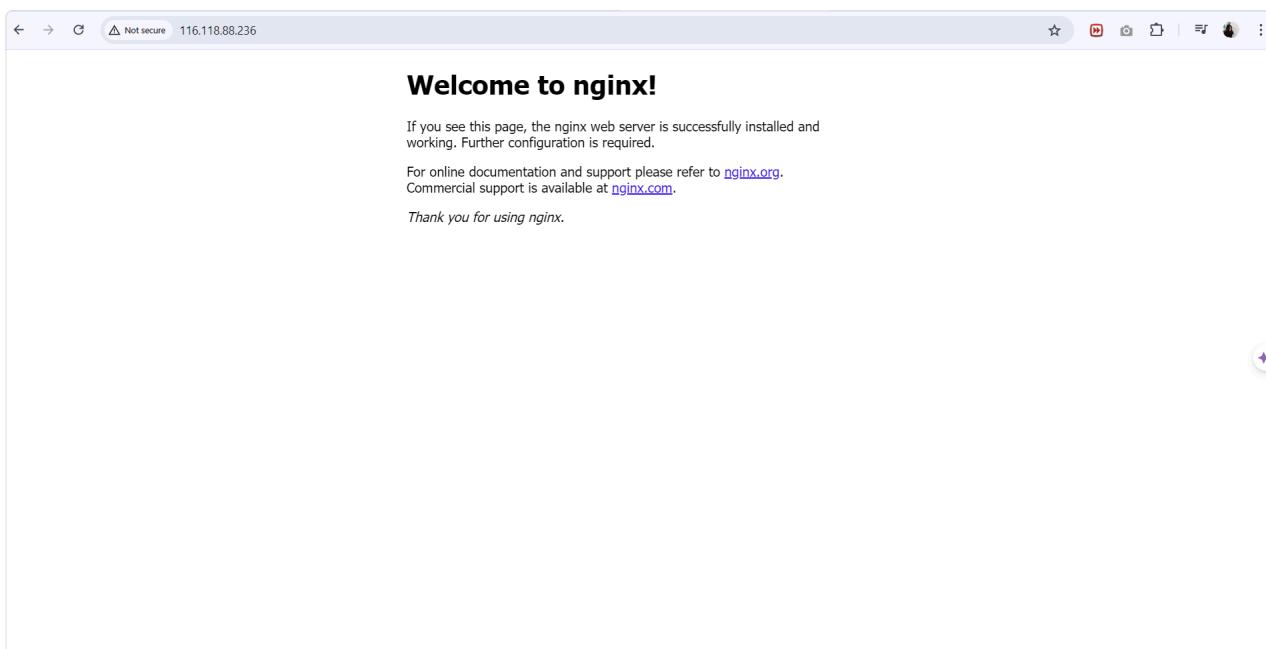
Lúc này, hệ thống vLB sẽ khởi tạo một Network Load Balancer, bạn có thể xem thông tin LB này thông qua portal vLB tại [đây](#).

Bước 3: Để truy cập vào app nginx vừa export, bạn có thể sử dụng Endpoint của Load Balancer URL với định dạng:

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

Bạn có thể lấy thông tin Public Endpoint của Load Balancer tại giao diện vLB. Cụ thể truy cập tại <https://hcm-3.console.vngcloud.vn/vserver/load-balancer/vlb/>

Ví dụ, bên dưới tôi đã truy cập thành công vào app nginx với địa chỉ : <http://116.118.88.236/>



Một vài chú ý khác:

- Bên trên là ví dụ hướng dẫn bạn thực hiện expose một service thông qua vLB Layer 4, bạn có thể thực hiện expose service thông qua vLB Layer 7 theo hướng dẫn tại [đây](#).
- Để đảm bảo private cluster hoạt động hiệu quả, chúng tôi đã tự động thêm Subnet mà bạn chọn sử dụng cho Cluster vào danh sách Whitelist của cụm. Bạn có thể sử dụng tính năng Whitelist để giới hạn các Subnet trong VPC có quyền truy cập vào kube-api. Chi tiết cách sử dụng tính năng Whitelist vui lòng tham khảo tại [đây](#).

Expose một service thông qua vLB Layer4

Điều kiện cần

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn tại [đây](#).
- Có ít nhất 1 **SSH key** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).
- Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.

Khởi tạo Cluster

Cluster trong Kubernetes là một tập hợp gồm một hoặc nhiều máy ảo (VM) được kết nối lại với nhau để chạy các ứng dụng được đóng gói dạng container. Cluster cung cấp một môi trường thống nhất để triển khai, quản lý và vận hành các container trên quy mô lớn.

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn, bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Cluster và Node Group của bạn tại Cluster Configuration, Default Node Group Configuration, Plugin. **Khi bạn chọn bật option , mặc định chúng tôi sẽ cài sẵn plugin này vào Cluster của bạn.**

Bước 5: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Kết nối và kiểm tra thông tin Cluster vừa tạo

Sau khi Cluster được khởi tạo thành công, bạn có thể thực hiện kết nối và kiểm tra thông tin Cluster vừa tạo theo các bước:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/k8s-cluster>

Bước 2: Danh sách Cluster được hiển thị, chọn biểu tượng và chọn **Download Config File** để thực hiện tải xuống file kubeconfig. File này sẽ giúp bạn có toàn quyền truy cập vào Cluster của bạn.

Bước 3: Đổi tên file này thành config và lưu nó vào thư mục **~/.kube/config**

Bước 4: Thực hiện kiểm tra Cluster thông qua lệnh:

- Chạy câu lệnh sau đây để kiểm tra **node**

```
kubectl get nodes
```

- Nếu kết quả trả về như bên dưới tức là bạn Cluster của bạn được khởi tạo thành công với 3 node như bên dưới.

NAME	STATUS	ROLES	AGE	VERSI0
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready	<none>	50m	v1.28.
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready	<none>	23m	v1.28.
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready	<none>	28m	v1.28.

Khởi tạo Service Account và cài đặt VNGCloud Controller Manager

 **Chú ý:**

Khi bạn thực hiện khởi tạo Cluster theo hướng dẫn bên trên, nếu bạn chưa bật option **Enable vLB Native Integration Driver**, mặc định chúng tôi sẽ không cài sẵn plugin này vào Cluster của bạn. Bạn cần tự thực hiện Khởi tạo Service Account và cài đặt VNGCloud Controller Manager theo hướng dẫn bên dưới. Nếu bạn đã bật option **Enable vLB Native Integration Driver**, thì chúng tôi đã cài sẵn plugin này vào Cluster của bạn, hãy bỏ qua bước Khởi tạo Service Account, cài đặt VNGCloud Controller Manager và tiếp tục thực hiện theo hướng dẫn kể từ Deploy một Workload.

- ✓ Hướng dẫn khởi tạo Service Account và cài đặt VNGCloud Controller Manager

Khởi tạo Service Account

- Khởi tạo hoặc sử dụng một **service account** đã tạo trên IAM và gắn policy: **vLBFULLAccess**, **vServerFullAccess**. Để tạo service account bạn truy cập tại [đây](#) và thực hiện theo các bước sau:
 - Chọn "**Create a Service Account**", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account
 - Tìm và chọn **Policy: vLBFULLAccess** và **Policy: vServerFullAccess**, sau đó nhấn "**Create a Service Account**" để tạo Service Account, Policy: vLBFULLAccess và Policy: vServerFullAccess do VNG Cloud tạo ra, bạn không thể xóa các policy này.
 - Sau khi tạo thành công bạn cần phải lưu lại **Client_ID** và **Secret_Key** của Service Account để thực hiện bước tiếp theo.
- Gỡ cài đặt cloud-controller-manager

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

- Bên cạnh đó, bạn có thể xóa Service Account đang sử dụng cho cloud-controller-manager vừa gỡ

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

Cài đặt VNGCloud Controller Manager

- Cài đặt Helm phiên bản từ 3.0 trở lên. Tham khảo tại <https://helm.sh/docs/intro/install/> để biết cách cài đặt.
- Thêm repo này vào cluster của bạn qua lệnh:

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

- Thay thế thông tin ClientID, Client Secret và ClusterID của cụm K8S của bạn và tiếp tục chạy:

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

- Sau khi việc cài đặt hoàn tất, thực hiện kiểm tra trạng thái của vngcloud-Integrate-controller pods:

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

Ví dụ như ảnh bên dưới là bạn đã cài đặt thành công vngcloud-controller-manager:

NAME	READY	STATUS	RESTARTS
vngcloud-controller-manager-8864c754c-bqhvvz	1/1	Running	5 (91s)

Deploy một Workload

Sau đây là hướng dẫn để bạn deploy service nginx trên Kubernetes.

Bước 1: Tạo Deployment, Service cho Nginx app.

- Tạo file **nginx-service-lb4.yaml** với nội dung sau:

```

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 Service này bằng lệnh:

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

Bước 2: Kiểm tra thông tin Deployment, Service vừa deploy

- Chạy câu lệnh sau đây để kiểm tra **Deployment**

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

- Nếu kết quả trả về như bên dưới tức là bạn đã deploy Deployment thành công.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)			
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP			
service/nginx-service	LoadBalancer	10.96.74.154	<pending>	80:31623/TCP			
NAME		READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS	
deployment.apps/nginx-app	nginx	0/1	1	0	2s		
NAME		READY	STATUS		RESTARTS	AGE	IP
pod/nginx-app-7f45b65946-bmrcf		0/1	ContainerCreating	0	2s	<n	

Lúc này, hệ thống vLB sẽ tự động tạo một LB tương ứng cho nginx app đã deployment, ví dụ:

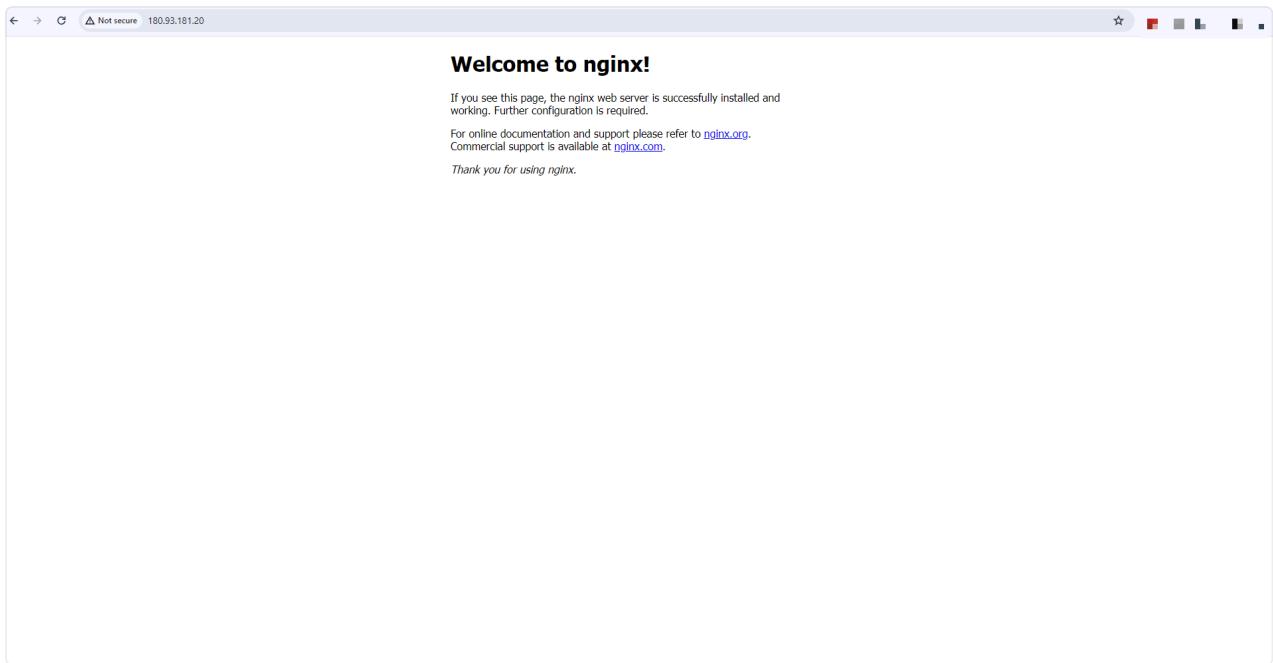
The screenshot shows the VNG Cloud interface for managing Load Balancers. On the left, there's a sidebar with navigation links like vServer, Region HCM03, Create a Load Balancer, and various network-related options. The main content area is titled 'Load Balancer' and contains a table listing a single load balancer entry. The table columns include Name, Status, End point, Schema, Type, Package, Created at, and Action. The listed entry is 'lb-6ea77772-0ff8-442c-81b7-f03e44dde742 vks-k8s-6d2acb-default-nginx-serv-998c5' with an 'ACTIVE' status, endpoint '180.93.181.20', schema 'Internet', type 'Network', package 'NLB_Small', and created at '21/04/2024 19:52:22'. There are also buttons for creating a new load balancer and a search bar.

Bước 3: Để truy cập vào app nginx vừa export, bạn có thể sử dụng URL với định dạng:

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

Bạn có thể lấy thông tin Public Endpoint của Load Balancer tại giao diện vLB. Cụ thể truy cập tại <https://hcm-3.console.vngcloud.vn/vserver/load-balancer/vlb/>

Ví dụ, bên dưới tôi đã truy cập thành công vào app nginx với địa chỉ : <http://180.93.181.20/>



Bạn có thể xem thêm về ALB tại [Working with Network load balancing \(NLB\)](#).

(i) Chú ý:

- Việc thay đổi tên hoặc kích thước (Rename, Resize) tài nguyên Load Balancer trên vServer Portal có thể gây ra sự không tương thích với tài nguyên trên Kubernetes Cluster. Điều này có thể dẫn đến việc các tài nguyên không hoạt động trên Cluster, hoặc tài nguyên bị đồng bộ lại hoặc thông tin tài nguyên giữa vServer Portal và Cluster không khớp. Để ngăn chặn vấn đề này, hãy sử dụng `kubectl` để quản lý tài nguyên của Cluster.

Preserve Source IP khi sử dụng vLB Layer4 và Nginx Ingress Controller

Preserve Source IP khi sử dụng vLB Layer 4 và Nginx Ingress Controller trong Kubernetes là quá trình duy trì địa chỉ IP gốc của client khi traffic được chuyển tiếp qua load balancer và vào cụm Kubernetes. Điều này rất quan trọng trong một số trường hợp khi bạn cần các thông tin chi tiết về kết nối của client, chẳng hạn như địa chỉ IP gốc và port gốc của client, để có thể thực hiện các quyết định xử lý traffic hoặc logging chính xác. Bên dưới là hướng dẫn cụ thể của chúng tôi để giúp bạn có thể thực hiện usecase này.

Điều kiện cần

- Bạn đã thực hiện khởi tạo Cluster trên hệ thống VKS theo các hướng dẫn tại [đây](#) và trên cụm của bạn đã được cài đặt **VNGCloud Controller Manager** với appversion từ **v0.2.1** trở lên. Nếu appversion của bạn thấp hơn version tiêu chuẩn này, bạn có thể thực hiện upgrade theo các hướng dẫn sau:
 - Đầu tiên, bạn cần lấy release name của **vngcloud-controller-manager** đã cài trên cụm của bạn:

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

- Sau đó, bạn hãy thực hiện upgrade lên version mới nhất thông qua lệnh:

```
helm upgrade vngcloud-controller-manager-1716448250 oci://vcr.vngcloud.vn/81--namespace kube-system
```
- Tiếp theo, bạn cần thực hiện cài đặt nginx-ingress-controller theo lệnh:

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

Cấu hình ConfigMap cho Nginx Ingress Controller

- Thêm vào ConfigMap của Nginx Ingress Controller các thiết lập để kích hoạt proxy protocol thông qua lệnh:

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

- Nếu bạn không sử dụng `cert-manager`, bạn cần thêm đoạn code sau vào tệp tin ConfigMap:

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

- Nếu bạn có sử dụng `cert-manager`, bạn cần thêm đoạn code sau vào tệp tin ConfigMap:

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

Cấu hình vLB Layer 4

- Tiếp theo, bạn cần cấu hình vLB Layer4 cho phép sử dụng proxy protocol cho service Load Balancer Nginx. Giá trị truyền vào là danh sách các service name trong Load Balancer sử dụng Proxy Protocol.

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

- Cuối cùng, bạn hãy thực hiện kiểm tra NLB trên vLB Portal cho tới khi các Load Balancer này được ACTIVE với đầy đủ listener, pool.

Cách sử dụng

- Giả sử, bạn có một service prometheus-node-exporter với port 9100 trong namespace default, bạn có thể apply yaml sau để có thể truy cập thông qua NLB

```
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
```

- Sau đó tôi sử dụng IP 103.245.252.75 để curl vào host kkk.example.com như sau:

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

- Kết quả log ghi nhận được đã có thông tin Client IP này như hình:

```
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" "-"
```

Expose một service thông qua vLB Layer7

Điều kiện cần

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn tại [đây](#).
- Có ít nhất 1 **SSH key** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).
- Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.

Khởi tạo Cluster

Cluster trong Kubernetes là một tập hợp gồm một hoặc nhiều máy ảo (VM) được kết nối lại với nhau để chạy các ứng dụng được đóng gói dạng container. Cluster cung cấp một môi trường thống nhất để triển khai, quản lý và vận hành các container trên quy mô lớn.

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn, bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Cluster và Node Group của bạn tại Cluster Configuration, Default Node Group Configuration, Plugin. **Khi bạn chọn bật option , mặc định chúng tôi sẽ cài sẵn plugin này vào Cluster của bạn.**

Bước 5: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Kết nối và kiểm tra thông tin Cluster vừa tạo

Sau khi Cluster được khởi tạo thành công, bạn có thể thực hiện kết nối và kiểm tra thông tin Cluster vừa tạo theo các bước:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/k8s-cluster>

Bước 2: Danh sách Cluster được hiển thị, chọn biểu tượng và chọn **Download Config File** để thực hiện tải xuống file kubeconfig. File này sẽ giúp bạn có toàn quyền truy cập vào Cluster của bạn.

Bước 3: Đổi tên file này thành config và lưu nó vào thư mục **~/.kube/config**

Bước 4: Thực hiện kiểm tra Cluster thông qua lệnh:

- Chạy câu lệnh sau đây để kiểm tra **node**

```
kubectl get nodes
```

- Nếu kết quả trả về như bên dưới tức là bạn Cluster của bạn được khởi tạo thành công với 3 node như bên dưới.

NAME	STATUS	ROLES	AGE	VERSIO
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready	<none>	50m	v1.28.
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready	<none>	23m	v1.28.
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready	<none>	28m	v1.28.

Khởi tạo Service Account và cài đặt VNGCloud Ingress Controller

 **Chú ý:**

Khi bạn thực hiện khởi tạo Cluster theo hướng dẫn bên trên, nếu bạn chưa bật option **Enable vLB Native Integration Driver**, mặc định chúng tôi sẽ không cài sẵn plugin này vào Cluster của bạn. Bạn cần tự thực hiện Khởi tạo Service Account và cài đặt VNGCloud Ingress Controller theo hướng dẫn bên dưới. Nếu bạn đã bật option **Enable vLB Native Integration Driver**, thì chúng tôi đã cài sẵn plugin này vào Cluster của bạn, hãy bỏ qua bước Khởi tạo Service Account, cài đặt VNGCloud Ingress Controller và tiếp tục thực hiện theo hướng dẫn kể từ Deploy một Workload.

- ✓ Khởi tạo Service Account và cài đặt VNGCloud Ingress Controller

Khởi tạo Service Account

- Khởi tạo hoặc sử dụng một **service account** đã tạo trên IAM và gắn policy: **vLBFullAccess**, **vServerFullAccess**. Để tạo service account bạn truy cập tại [đây](#) và thực hiện theo các bước sau:
 - Chọn "**Create a Service Account**", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account
 - Tìm và chọn **Policy: vLBFullAccess** và **Policy: vServerFullAccess**, sau đó nhấn "**Create a Service Account**" để tạo Service Account, Policy: vLBFullAccess và Policy: vServerFullAccess do VNG Cloud tạo ra, bạn không thể xóa các policy này.
 - Sau khi tạo thành công bạn cần phải lưu lại **Client_ID** và **Secret_Key** của Service Account để thực hiện bước tiếp theo.

Cài đặt VNGCloud Ingress Controller

- Cài đặt Helm phiên bản từ 3.0 trở lên. Tham khảo tại <https://helm.sh/docs/intro/install/> để biết cách cài đặt.
- Thêm repo này vào cluster của bạn qua lệnh:

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

- Thay thế thông tin ClientID, Client Secret và ClusterID của cụm K8S của bạn và tiếp tục chạy:

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

- Sau khi việc cài đặt hoàn tất, thực hiện kiểm tra trạng thái của vngcloud-ingress-controller pods:

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

Ví dụ như ảnh bên dưới là bạn đã cài đặt thành công vngcloud-controller-manager:

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

Deploy một Workload

Sau đây là hướng dẫn để bạn deploy service nginx trên Kubernetes.

Bước 1: Tạo Deployment cho Nginx app.

- Tạo file **nginx-service-lb7.yaml** với nội dung sau:

```

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 Deployment này bằng lệnh:

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

Bước 2: Kiểm tra thông tin Deployment, Service vừa deploy

- Chạy câu lệnh sau đây để kiểm tra **Deployment**

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

- Nếu kết quả trả về như bên dưới tức là bạn đã deploy Deployment thành công.

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/TCP		
NAME		READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS
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	172.16.54.

Tạo Ingress Resource

- Tạo file **nginx-ingress.yaml** với nội dung sau:

```

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

```

- Chạy câu lệnh sau đây để triển khai Ingress

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

Lúc này, hệ thống vLB sẽ tự động tạo một LB tương ứng với Ingress resource bên trên, ví dụ:

(i) Chú ý:

- Hiện tại Ingress chỉ hỗ trợ duy nhất TLS port 443 và là điểm kết thúc cho TLS (TLS termination). TLS Secret phải chứa các trường với tên key là tls.crt và tls.key, đây chính là certificate và private key để sử dụng cho TLS. Nếu bạn muốn sử dụng Certificate cho một host, hãy thực hiện tải lên Certificate theo hướng dẫn tại [Upload a certificate] và sử dụng chúng như một annotation. Ví dụ:

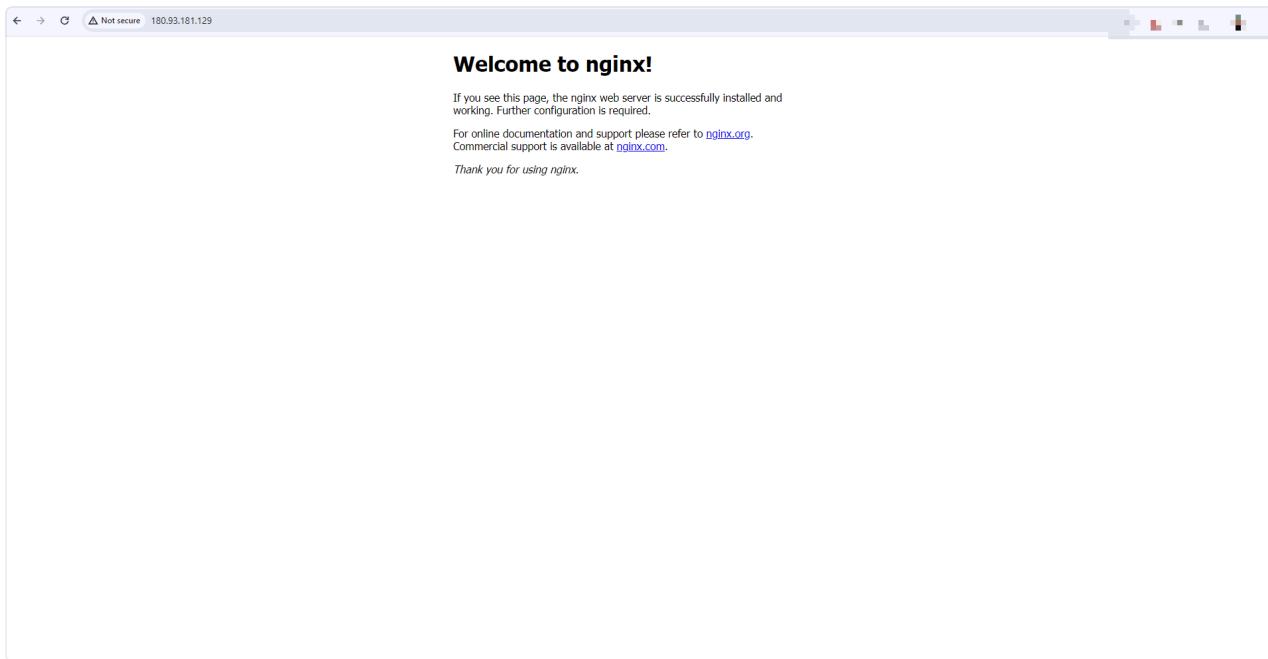
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    # kubernetes.io/ingress.class: "vngcloud" # this annotation is deprecated
    vks.vngcloud.vn/load-balancer-id: "lb-6cdea8fd-4589-410e-933f-c3bc46"
    vks.vngcloud.vn/certificate-ids: "secret-a6d20ec6-f3e5-499a-981b-b14"
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
```

Để truy cập vào app nginx, bạn có thể sử dụng Endpoint của Load Balancer mà hệ thống đã tạo.

http://Endpoint/

Bạn có thể lấy thông tin Public Endpoint của Load Balancer tại giao diện vLB. Cụ thể truy cập tại

Ví dụ, bên dưới tôi đã truy cập thành công vào app nginx với địa chỉ : <http://180.93.181.129/>



Bạn có thể xem thêm về ALB tại [Working with Application Load Balancer \(ALB\)](#).

(i) Chú ý:

- Việc thay đổi tên hoặc kích thước (Rename, Resize) tài nguyên Load Balancer trên vServer Portal có thể gây ra sự không tương thích với tài nguyên trên Kubernetes Cluster. Điều này có thể dẫn đến việc các tài nguyên không hoạt động trên Cluster, hoặc tài nguyên bị đồng bộ lại hoặc thông tin tài nguyên giữa vServer Portal và Cluster không khớp. Để ngăn chặn vấn đề này, hãy sử dụng `kubectl` để quản lý tài nguyên của Cluster.

Integrate with Container Storage Interface (CSI)

Điều kiện cần

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn tại [đây](#).
- Có ít nhất 1 **SSH** key đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).
- Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.

Khởi tạo Cluster

Cluster trong Kubernetes là một tập hợp gồm một hoặc nhiều máy ảo (VM) được kết nối lại với nhau để chạy các ứng dụng được đóng gói dạng container. Cluster cung cấp một môi trường thống nhất để triển khai, quản lý và vận hành các container trên quy mô lớn.

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn, bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Cluster và Node Group của bạn tại Cluster Configuration, Default Node Group Configuration, Plugin. Khi bạn chọn bật option **Enable vLB Native Integration Driver**, mặc định chúng tôi sẽ cài sẵn plugin này vào Cluster của bạn.

Bước 5: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Kết nối và kiểm tra thông tin Cluster vừa tạo

Sau khi Cluster được khởi tạo thành công, bạn có thể thực hiện kết nối và kiểm tra thông tin Cluster vừa tạo theo các bước:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/k8s-cluster>

Bước 2: Danh sách Cluster được hiển thị, chọn **Download Config File** để thực hiện tải xuống file kubeconfig. File này sẽ giúp bạn có toàn quyền truy cập vào Cluster của bạn.

Bước 3: Đổi tên file này thành config và lưu nó vào thư mục `~/.kube/config`

Bước 4: Thực hiện kiểm tra Cluster thông qua lệnh:

- Chạy câu lệnh sau đây để kiểm tra node

```
kubectl get nodes
```

- Nếu kết quả trả về như bên dưới tức là bạn Cluster của bạn được khởi tạo thành công với 3 node như bên dưới.

NAME	STATUS	ROLES	AGE	VERSI
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready	<none>	50m	v1.28.
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready	<none>	23m	v1.28.
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready	<none>	28m	v1.28.

Khởi tạo Service Account và cài đặt VNGCloud BlockStorage CSI Driver

 **Chú ý:**

- Khi bạn thực hiện khởi tạo Cluster theo hướng dẫn bên trên, nếu bạn chưa bật option **Enable BlockStore Persistent Disk CSI Driver**, mặc định chúng tôi sẽ không cài sẵn plugin này vào Cluster của bạn. Bạn cần tự thực hiện Khởi tạo Service Account và cài đặt VNGCloud BlockStorage CSI Driver theo hướng dẫn bên dưới. Nếu bạn đã bật option **Enable BlockStore Persistent Disk CSI Driver**, thì chúng tôi đã cài sẵn plugin này vào Cluster của bạn, hãy bỏ qua bước Khởi tạo Service Account, cài đặt VNGCloud BlockStorage CSI Driver và tiếp tục thực hiện theo hướng dẫn kể từ Deploy một Workload.
- **VNGCloud BlockStorage CSI Driver** chỉ hỗ trợ attach volume với một node (VM) duy nhất trong suốt vòng đời của volume đó. Nếu bạn có nhu cầu **ReadWriteMany**, bạn có thể cân nhắc sử dụng NFS CSI Driver, vì nó cho phép nhiều nodes có thể Read và Write trên cùng một volume cùng một lúc. Điều này rất hữu ích cho các ứng dụng cần chia sẻ dữ liệu giữa nhiều pods hoặc services trong Kubernetes.

- ✓ Khởi tạo Service Account và cài đặt VNGCloud BlockStorage CSI Driver

Khởi tạo Service Account

- Khởi tạo hoặc sử dụng một **service account** đã tạo trên IAM và gắn policy: **vServerFullAccess**. Để tạo service account bạn truy cập tại [đây](#) và thực hiện theo các bước sau:
 - Chọn "**Create a Service Account**", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account
 - Tìm và chọn **Policy: vServerFullAccess**, sau đó nhấn "**Create a Service Account**" để tạo Service Account, Policy: vLBFullAccess và Policy: vServerFullAccess do VNG Cloud tạo ra, bạn không thể xóa các policy này.
 - Sau khi tạo thành công bạn cần phải lưu lại **Client_ID** và **Secret_Key** của Service Account để thực hiện bước tiếp theo.

Cài đặt VNGCloud BlockStorage CSI Driver

- Cài đặt Helm phiên bản từ 3.0 trở lên. Tham khảo tại <https://helm.sh/docs/intro/install/> để biết cách cài đặt.
- Thêm repo này vào cluster của bạn qua lệnh:

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

- Thay thế thông tin ClientID, Client Secret và ClusterID của cụm K8S của bạn và tiếp tục chạy:

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

- Sau khi việc cài đặt hoàn tất, thực hiện kiểm tra trạng thái của vngcloud-blockstorage-csi-driver pods:

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

Ví dụ như ảnh bên dưới là bạn đã cài đặt thành công vngcloud-controller-manager:

NAME	READY	STATUS	RESTARTS
vngcloud-csi-controller-56bd7b85f-ctpns	7/7	Running	6 (2d)
vngcloud-csi-controller-56bd7b85f-npp9n	7/7	Running	2 (2d)
vngcloud-csi-node-c8r2w	3/3	Running	0

Deploy một Workload

Sau đây là hướng dẫn để bạn deploy service nginx trên Kubernetes.

Bước 1: Tạo Deployment cho Nginx app.

- Tạo file **nginx-service.yaml** với nội dung sau:

```

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 Deployment này bằng lệnh:

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

Bước 2: Kiểm tra thông tin Deployment, Service vừa deploy

- Chạy câu lệnh sau đây để kiểm tra **Deployment**

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

- Nếu kết quả trả về như bên dưới tức là bạn đã deploy Deployment thành công.

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>	30080:31289		
service/nginx-service	LoadBalancer	10.96.179.221	<pending>	80:32624/TC		
NAME		READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS
deployment.apps/nginx-app		1/1	1	1	16s	nginx
NAME		READY	STATUS	RESTARTS	AGE	IP
pod/nginx-app-7f45b65946-t7d7k		1/1	Running	0	16s	172.16.24.20

Tạo Persistent Volume

- Tạo file **persistent-volume.yaml** với nội dung sau:

```

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 CSI driver
parameters:
  type: vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018    # The volume type UUID
allowVolumeExpansion: true                  # MUST set this value to true
---

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

apiVersion: v1
kind: Pod
metadata:
  name: nginx                            # [5] The Pod name, CAN be changed
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 [6]
  volumes:
    - name: my-volume-name                 # [6] The volume name, CAN be changed
      persistentVolumeClaim:
        claimName: my-expansion-pvc         # MUST be the same as [2]
        readOnly: false

```

- Chạy câu lệnh sau đây để triển khai Ingress

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

Lúc này, hệ thống vServer sẽ tự động tạo một Volume tương ứng với file yaml bên trên, ví dụ:

The screenshot shows the VNG Cloud vServer Block Store Volumes page. On the left sidebar, under the 'vServer' section, 'Volumes' is selected. The main area displays a table of volumes with the following columns: Name, Status, Type, IOPS, Size, Attachment, Multi-Attach, and Action. Two volumes are listed:

Name	Status	Type	IOPS	Size	Attachment	Multi-Attach	Action
vol-2f95e9d-86c4-4043-a728-99637704b050	IN USE	SSD	200	20	ins-4ae9966b-29a1-4c01-b6eb-a543b1d2abf4	No	⋮
ng-3f06013a-f6a5-47ba-a51f-bc5e9c2b10a7-ecea1 boot_volume	IN USE	SSD	3000	20	ins-4ae9966b-29a1-4c01-b6eb-a543b1d2abf4	No	⋮

At the bottom of the table, it says 'Total: 7'. The footer contains copyright information for VNG Data IT JSC and links for Terms, Privacy Policy, and various certifications.

Tạo Snapshot

Snapshot là phương pháp sao lưu giữ liệu với chi phí thấp, thuận tiện và hiệu quả và có thể được sử dụng để tạo image, phục hồi dữ liệu và phân phối các bản sao dữ liệu. Nếu bạn là người dùng mới chưa từng sử dụng dịch vụ Snapshot, bạn cần thực hiện Activate Snapshot Service (Kích hoạt dịch vụ Snapshot) trước khi có thể tạo Snapshot cho Persistent Volume của bạn.

Activate Snapshot Service

Để có thể tạo Snapshot, bạn cần thực hiện Activate Snapshot Service. Bạn sẽ không bị tính phí khi kích hoạt dịch vụ snapshot. Sau khi bạn tạo snapshot, chi phí sẽ được tính dựa trên dung lượng lưu trữ và thời gian lưu trữ của các bản snapshot này. Thực hiện theo các bước sau đây để kích hoạt dịch vụ Snapshot:

Bước 1: Truy cập vào <https://hcm-3.console.vngcloud.vn/vserver/block-store/snapshot/overview>

Bước 2: Chọn Activate Snapshot Service.

Ví dụ:

Billing Note

You are not charged for activating the snapshot service.

After you create snapshots, the snapshots are billed based on the amount of storage space they consume and the amount of time for which they are stored. Snapshots are billed per region every hour on the hour (7.7 VND/ 1GB/ 1 hour). Pay attention to fees incurred in the future.

We recommend that you delete snapshots that are no longer needed on a regular basis. [Click here for more information about snapshot pricing.](#)

You have no Snapshot

VNG Cloud snapshot service allows you to create snapshots for all disk categories. Snapshots are a low-cost, convenient, and efficient method to back up data and can be used to create images, implement disaster recovery plans, and distribute data copies.

[Activate Snapshot Service](#)

[What can I do with Snapshots?](#)

Cài đặt VNGCloud Snapshot Controller

- Cài đặt Helm phiên bản từ 3.0 trở lên. Tham khảo tại <https://helm.sh/docs/intro/install/> để biết cách cài đặt.
- Thêm repo này vào cluster của bạn qua lệnh:

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

- Tiếp tục chạy:

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

- Sau khi việc cài đặt hoàn tất, thực hiện kiểm tra trạng thái của vngcloud-blockstorage-csi-driver pods:

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

Ví dụ như ảnh bên dưới là bạn đã cài đặt thành công vngcloud-controller-manager:

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

Tạo file `snapshot.yaml` với nội dung sau:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: my-snapshot-storage-class  # [2] The name of the volume snapshot class,
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]
```

- Chạy câu lệnh sau đây để triển khai Ingress

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

Kiểm tra PVC và Snapshot vừa tạo

- Sau khi apply tập tin thành công, bạn có thể kiểm tra danh sách service, pvc thông qua:

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

NAME	PROVISIONER				
storageclass.storage.k8s.io/my-expansion-storage-class	bs.csi.vngcloud.vn				
storageclass.storage.k8s.io/sc-iops-200-retain (default)	bs.csi.vngcloud.vn				
NAME	STATUS	VOLUME			
persistentvolumeclaim/my-expansion-pvc	Bound	pvc-14456f4a-ee9e-435d-a94f-5a			
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx	1/1	Running	0	10m	172.16.24.20
pod/nginx-app-7f45b65946-t7d7k	1/1	Running	0	94m	172.16.24.20

Thay đổi thông số IOPS của Persistent Volume vừa tạo

Để thay đổi thông số IOPS của Persistent Volume vừa tạo, hãy thực hiện theo các bước sau đây:

Bước 1: Chạy lệnh bên dưới để liệt kê các PVC trong Cluster của bạn

```
kubectl get persistentvolumes
```

Bước 2: Chỉnh sửa tệp tin YAML của PVC theo lệnh

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

- Nếu bạn chưa chỉnh sửa IOPS của Persistent Volume lần nào trước đó, khi bạn chạy lệnh trên, bạn hãy thêm 1 annotation bs.csi.vngcloud.vn/volume-type: "volume-type-id" . Ví dụ: bên dưới tôi đang thay đổi IOPS của Persistent Volume từ 200 (Volume type id = vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018) lên 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-a36a40cea748"
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"PersistentVolumeClaim","metadata":{"annotations":{},"name":"my-expansion-pvc","namespace":"default","resourceVersion":"11041591","uid":"14456f4a-ee9e-435d-a94f-5a2e820954e9}}
    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

```

- Nếu bạn đã chỉnh sửa IOPS của Persistent Volume lần nào trước đó, khi bạn chạy lệnh trên, tệp tin yaml của bạn đã có sẵn annotation `bs.csi.vngcloud.vn/volume-type: "volume-type-id"`. Lúc này, hãy chỉnh sửa annotation này về Volume type id có IOPS mà bạn mong muốn.

Thay đổi Disk Volume của Persistent Volume vừa tạo

Để thay đổi Disk Volume của Persistent Volume vừa tạo, hãy thực hiện chạy lệnh sau:

Ví dụ: ban đầu PVC được tạo có kích cỡ 20 Gi, hiện tại tôi sẽ tăng nó lên 30Gi

```
kubectl patch pvc my-expansion-pvc -p '{"spec": {"resources": {"requests": {"storage": "20Gi"}, "limits": {"storage": "20Gi"}}, "volumeClaimTemplate": {"spec": {"storageClassName": "my-expansion-storage-class", "dataSource": {"name": "my-snapshot-pvc", "kind": "VolumeSnapshot", "apiGroup": "snapshot.storage.k8s.io"}, "accessModes": ["ReadWriteOnce"], "resources": {"storage": "20Gi"}}, "metadata": {"name": "my-expansion-pvc"}}}
```

 **Chú ý:**

- Bạn chỉ có thể thực hiện tăng Disk Volume mà không thể thực hiện giảm kích thước Disk Volume này.

Restore Persistent Volume từ Snapshot

Để khôi phục Persistent Volume từ Snapshot, bạn hãy thực hiện theo các bước sau:

- Tạo file **restore-volume.yaml** với nội dung sau:

```
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
```

Khởi tạo một Cluster thông qua ví POC

Tài nguyên POC sinh ra nhằm mục đích hỗ trợ người dùng có thể trải nghiệm dịch vụ tại VNG Cloud một cách tốt nhất.

Điều kiện sử dụng tài nguyên POC:

- **Đối tượng:** Người dùng trả trước được cấp ví POC
- **Nguồn tiền:** [Ví POC](#)
- **Tài nguyên:** Tất cả các tài nguyên được áp dụng POC
- **Thời gian sử dụng:** Tùy thuộc vào thời hạn ví POC được cấp.

Điều kiện cần

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn tại [đây](#).
- Có ít nhất 1 **SSH** key đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).
- Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.

Khởi tạo Cluster

Cluster trong Kubernetes là một tập hợp gồm một hoặc nhiều máy ảo (VM) được kết nối lại với nhau để chạy các ứng dụng được đóng gói dạng container. Cluster cung cấp một môi trường thống nhất để triển khai, quản lý và vận hành các container trên quy mô lớn.

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn, bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Cluster và Node Group của bạn tại Cluster Configuration, Default Node Group Configuration, Plugin. **Mặc định chúng tôi sẽ khởi tạo cho bạn một Public Cluster với Public Node Group.**

Bước 5: Chọn **POC** và chọn tiếp **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Chú ý:

- Khi bạn khởi tạo Cluster và chọn sử dụng ví POC, chúng tôi đã tự động tạo Control Plane, Node, Volume và Private Service Endpoint (nếu bạn chọn sử dụng) thông qua ví POC. Đối với các tài nguyên khác như
 - PVC:** khi thực hiện khởi tạo qua yaml, bạn vui lòng thêm tham số `isPOC: "true"` vào file yaml này. Tham khảo ví dụ bên dưới.
 - LoadBalancer:** khi thực hiện khởi tạo qua yaml, bạn vui lòng thêm `annotation vks.vngcloud.vn/is-poc: "true"` vào file yaml này. Tham khảo ví dụ bên dưới.

- Do các resource **Load Balancer** và **PVC** được quản lý thông qua YAML, sau khi Stop POC, nếu trong file YAML của bạn vẫn có tham số `isPOC : true` hoặc `is-poc : true`, trong trường hợp bạn xóa Load Balancer từ Portal vLB và xóa tham số `load-balancer-id` trong yaml, lúc này hệ thống sẽ tự động tạo lại các resource này thông qua ví POC. Để tạo Load Balancer và PVC khác bằng tiền thật, vui lòng thay đổi tham số `isPOC` thành `false`. (`isPOC : false` hoặc `is-poc : false`). Chúng tôi khuyến cáo bạn nên thực hiện điều chỉnh tham số này trước khi thực hiện Stop POC cho Cluster của bạn.

Kết nối và kiểm tra thông tin Cluster vừa tạo

Sau khi Cluster được khởi tạo thành công, bạn có thể thực hiện kết nối và kiểm tra thông tin Cluster vừa tạo theo các bước:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/k8s-cluster>

Bước 2: Danh sách Cluster được hiển thị, chọn biểu tượng **Download** và chọn **Download Config File** để thực hiện tải xuống file kubeconfig. File này sẽ giúp bạn có toàn quyền truy cập vào Cluster của bạn.

Bước 3: Đổi tên file này thành config và lưu nó vào thư mục `~/.kube/config`

Bước 4: Thực hiện kiểm tra Cluster thông qua lệnh:

- Chạy câu lệnh sau đây để kiểm tra **node**

```
kubectl get nodes
```

- Nếu kết quả trả về như bên dưới tức là bạn Cluster của bạn được khởi tạo thành công với 3 node như bên dưới.

NAME	STATUS	ROLES	AGE	VERSIO
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-834b7	Ready	<none>	50m	v1.28.
ng-0e10592c-e70e-404d-a4e8-5e3b80f805e4-cf652	Ready	<none>	23m	v1.28.
ng-0f4ed631-1252-49f7-8dfc-386fa0b2d29b-a8ef0	Ready	<none>	28m	v1.28.

Deploy Workload và expose service thông qua vLB Layer 4 hoặc vLB Layer 7

Sau đây là hướng dẫn để bạn deploy 2 workload và expose chúng qua Load Balancer Layer 4 và Load Balancer Layer 7 trên Kubernetes.

Bước 1: Tạo Deployment, Service cho Nginx app.

- Tạo file **nginx-service.yaml** với nội dung sau:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      unique-label: app1
  template:
    metadata:
      labels:
        unique-label: app1
        same-label: vngcloud
    spec:
      containers:
        - name: nginx-deployment
          image: nginx
          ports:
            - containerPort: 80
              name: http
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-http-server
spec:
  replicas: 2
  selector:
    matchLabels:
      app: python-http-server
  template:
    metadata:
      labels:
        app: python-http-server
        same-label: vngcloud
    spec:
      containers:
        - name: python-http-server
          image: python:3.9-slim
          command: ["python", "-m", "http.server", "8080"]
          ports:
            - containerPort: 8080
              name: http
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: default
  annotations:

```

```

vks.vngcloud.vn/is-poc: "true"      # Tham số chỉ định Load Balancer được tạ
spec:
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http-server
  selector:
    same-label: vngcloud
  type: LoadBalancer

---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: webapp-02
  namespace: default
  annotations:
    vks.vngcloud.vn/is-poc: "true"      # Tham số chỉ định Load Balancer được tạ
spec:
  ingressClassName: vngcloud
  defaultBackend:
    service:
      name: nginx-service
      port:
        name: http-server

```

- Deploy Service này bằng lệnh:

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

- Tiếp theo, bạn có thể thực hiện kiểm tra Deployment qua lệnh:

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

Tạo Persistent Volume

- Tạo file **persistent-volume.yaml** với nội dung sau:

```

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 CSI driver
parameters:
  type: vtype-xxxxxxxxxxxxxxxxxxxxxxxxxxxxx   # The volume type UUID
  isPOC: "true"                                # Tham số chỉ định Volume
allowVolumeExpansion: true                   # MUST set this value to true
---

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

```

```

apiVersion: v1
kind: Pod
metadata:
  name: nginx                                # [5] The Pod name, CAN be changed
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 [6]
  volumes:
    - name: my-volume-name                    # [6] The volume name, CAN be changed
  persistentVolumeClaim:
    claimName: my-expansion-pvc            # MUST be the same as [2]
    readOnly: false

```

- Chạy câu lệnh sau đây để triển khai Ingress

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

Tạo Snapshot

Đối với loại resource **Snapshot**, bạn không thể chỉ định snapshot sử dụng ví POC từ VKS.

Để thực hiện tạo Snapshot qua ví POC, tại **vServer Portal**, vui lòng chọn **Activate Snapshot**

Snapshot, sau đó tại màn hình **Checkout**, vui lòng chọn sử dụng ví **POC**. Lúc này **tất cả các resource snapshot của bạn sẽ được tạo qua ví POC**. Do đó, việc stop POC cần được bạn thực hiện thông qua **vConsole** hoặc **vServer Portal**. Tham khảo thêm hình bên dưới.

The screenshot shows the VNG Cloud vServer Portal interface. On the left, there's a sidebar with various cloud services like vServer, Network, and Server. The main content area is titled 'Snapshot' with a status of 'DISABLE'. It contains a 'Billing Note' section stating that snapshots are low-cost, convenient, and efficient for backup and disaster recovery. Below this is a note about regular snapshot deletion. A prominent red arrow points to a large orange button labeled 'Activate Snapshot Service'.

The screenshot shows the 'Checkout' page for a 'snapshot-HCM-03'. It lists the resource as 'vstorage - snapshot snapshot-HCM-03' with a price of '0 VND'. At the bottom, there's a 'Coupon' input field, an 'Apply' button, and a 'Holding (1 items) 0 VND' section. To the right of this holding section is a checkbox labeled 'Pay with POC wallet' with the note 'Available: 956,000,600 credits'. A red arrow points to this checkbox.

Cài đặt VNGCloud Snapshot Controller

- Cài đặt Helm phiên bản từ 3.0 trở lên. Tham khảo tại <https://helm.sh/docs/intro/install/> để biết cách cài đặt.
- Thêm repo này vào cluster của bạn qua lệnh:

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

- Tiếp tục chạy:

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

- Sau khi việc cài đặt hoàn tất, thực hiện kiểm tra trạng thái của vngcloud-blockstorage-csi-driver pods:

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

Ví dụ như ảnh bên dưới là bạn đã cài đặt thành công vngcloud-controller-manager:

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

Tạo file snapshot.yaml với nội dung sau:

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: my-snapshot-storage-class # [2] The name of the volume snapshot class,
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]

```

- Chạy câu lệnh sau đây để triển khai Ingress

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

Kiểm tra PVC và Snapshot vừa tạo

- Sau khi apply tập tin thành công, bạn có thể kiểm tra danh sách service, pvc thông qua:

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

NAME	PROVISIONER				
storageclass.storage.k8s.io/my-expansion-storage-class	bs.csi.vngcloud.vn				
storageclass.storage.k8s.io/sc-iops-200-retain (default)	bs.csi.vngcloud.vn				
NAME	STATUS	VOLUME			
persistentvolumeclaim/my-expansion-pvc	Bound	pvc-14456f4a-ee9e-435d-a94f-5a			
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginxx	1/1	Running	0	10m	172.16.24.20
pod/nginxx-app-7f45b65946-t7d7k	1/1	Running	0	94m	172.16.24.20

Thay đổi thông số IOPS của Persistent Volume vừa tạo

Để thay đổi thông số IOPS của Persistent Volume vừa tạo, hãy thực hiện theo các bước sau đây:

Bước 1: Chạy lệnh bên dưới để liệt kê các PVC trong Cluster của bạn

```
kubectl get persistentvolumes
```

Bước 2: Chỉnh sửa tệp tin YAML của PVC theo lệnh

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

- Nếu bạn chưa chỉnh sửa IOPS của Persistent Volume lần nào trước đó, khi bạn chạy lệnh trên, bạn hãy thêm 1 annotation bs.csi.vngcloud.vn/volume-type: "volume-type-id" . Ví dụ: bên dưới tôi đang thay đổi IOPS của Persistent Volume từ 200 (Volume type id = vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018) lên 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-a36a40cea748"
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"PersistentVolumeClaim","metadata":{"annotations":{},"name":null,"namespace":null,"resourceVersion":null,"uid":null}}
    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

```

- Nếu bạn đã chỉnh sửa IOPS của Persistent Volume lần nào trước đó, khi bạn chạy lệnh trên, tệp tin yaml của bạn đã có sẵn annotation `bs.csi.vngcloud.vn/volume-type: "volume-type-id"`. Lúc này, hãy chỉnh sửa annotation này về Volume type id có IOPS mà bạn mong muốn.

Thay đổi Disk Volume của Persistent Volume vừa tạo

Để thay đổi Disk Volume của Persistent Volume vừa tạo, hãy thực hiện chạy lệnh sau:

Ví dụ: ban đầu PVC được tạo có kích cỡ 20 Gi, hiện tại tôi sẽ tăng nó lên 30Gi

```
kubectl patch pvc my-expansion-pvc -p '{"spec": {"resources": {"requests": {"storage": "20Gi"}, "limits": {"storage": "20Gi"}}, "volumeClaimTemplate": {"spec": {"volumeName": "my-expansion-pvc", "storageClassName": "my-expansion-storage-class", "accessModes": [{"mode": "ReadWriteOnce"}], "resources": {"requests": {"storage": "20Gi"}, "limits": {"storage": "20Gi"}}, "dataSource": {"name": "my-snapshot-pvc", "kind": "VolumeSnapshot", "apiGroup": "snapshot.storage.k8s.io"}}}}'
```

 **Chú ý:**

- Bạn chỉ có thể thực hiện tăng Disk Volume mà không thể thực hiện giảm kích thước Disk Volume này.

Restore Persistent Volume từ Snapshot

Để khôi phục Persistent Volume từ Snapshot, bạn hãy thực hiện theo các bước sau:

- Tạo file **restore-volume.yaml** với nội dung sau:

```
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
```

Sau một thời gian trải nghiệm VKS, nếu bạn muốn chuyển các tài nguyên POC này thành tài nguyên thật, vui lòng thực hiện theo các bước theo hướng dẫn tại [đây](#).

Sử dụng Terraform để khởi tạo Cluster và Node Group

Tổng quan

Terraform là một công cụ mã nguồn mở được sử dụng để tự động hóa việc cung cấp và quản lý cơ sở hạ tầng như máy ảo, mạng, lưu trữ và Kubernetes.

Với Terraform, bạn có thể mô tả cơ sở hạ tầng mong muốn bằng mã, sau đó Terraform sẽ thực hiện các thao tác cần thiết để tạo hoặc cập nhật cơ sở hạ tầng cho phù hợp với mô tả của bạn.

Khởi tạo Cluster và Node Group

Để khởi tạo một Cluster Kubernetes bằng Terraform, bạn cần thực hiện các bước sau:

1. **Truy cập IAM Portal** tại [đây](#), thực hiện tạo Service Account với quyền hạn **VKS Full Access**. Cụ thể, tại trang IAM, bạn có thể:
 - Chọn "Create a Service Account", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account.
 - Tìm và chọn **Policy: VKSFullAccess** sau đó nhấn "**Create a Service Account**" để tạo Service Account, **Policy: VKSFullAccess** do VNG Cloud tạo ra, bạn không thể xóa các policy này.
 - Sau khi tạo thành công bạn cần phải lưu lại **Client_ID** và **Secret_Key** của Service Account để thực hiện bước tiếp theo.
2. **Truy cập VKS Portal** tại [đây](#), **thực hiện Activate** dịch vụ VKS ở tab **Overview**. Hãy chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn.
3. **Cài đặt Terraform:**
 - Tải xuống và cài đặt Terraform cho hệ điều hành của bạn từ <https://developer.hashicorp.com/terraform/install>.
4. **Khởi tạo cấu hình Terraform:**
 - Tạo tệp `variable.tf` và khai báo thông tin Service Account trong file này.
 - Tạo tệp `main.tf` và định nghĩa các tài nguyên Kubernetes Cluster mà bạn muốn tạo.

Ví dụ:

- Tệp `variable.tf`: bạn cần thay thế Client ID và Client Secret đã khởi tạo ở bước 1 ở file này.

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

- Tệp `main.tf`: trong ví dụ này tôi thực hiện khởi tạo một Cluster và một Node Group có thông tin sau:
 - Tên Cluster: my-cluster
 - K8S Version: v1.28.8
 - Mode: Public Cluster và Public Node Group
 - Tên Node Group: my-nodegroup
 - Bật AutoScaling: scale từ 0 tới 5 nodes

Chú ý:

- Trong file `main.tf`, để khởi tạo một cluster với một node group, bạn bắt buộc cần truyền vào các thông số sau:

```
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"
```

- File `main.tf` mẫu giúp bạn tạo Cluster và Node Group theo cấu hình bên trên:

```

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

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

5. Khởi tạo Terraform:

- Chạy lệnh `terraform init`. Lệnh này sẽ tải xuống các plugin cần thiết và khởi tạo trạng thái Terraform.

6. Áp dụng cấu hình Terraform:

- Chạy lệnh `terraform apply`. Lệnh này sẽ tạo Cluster Kubernetes theo mô tả trong tệp `main.tf`.

Tham khảo thêm về cách sử dụng Terraform để làm việc với VKS tại [đây](#).

Khởi tạo và làm việc với NVIDIA GPU Node Group

Tổng quan

- [NVIDIA GPU Operator](#) là một operator giúp đơn giản hóa việc triển khai và quản lý các node GPU trong các cụm Kubernetes. Nó cung cấp một tập hợp các tài nguyên tùy chỉnh và bộ điều khiển của Kubernetes cùng làm việc để tự động hóa quản lý tài nguyên GPU trong cụm Kubernetes.
 - Trong hướng dẫn này, chúng tôi sẽ hướng dẫn bạn:
 - Khởi tạo một node group với NVIDIA GPU
 - Cài đặt NVIDIA GPU Operator.
 - Triển khai một GPU Workload.
 - Thiết lập GPU Sharing.
 - Giám sát hoạt động GPU resources.
 - Autoscale GPU resources.
-

Khởi tạo một node group với NVIDIA GPU

- Thực hiện khởi tạo một Cluster với ít nhất một node group sử dụng NVIDIA GPU theo hướng dẫn tại [đây](#) nếu bạn muốn khởi tạo public node group hoặc tại [đây](#) nếu bạn muốn khởi tạo private node group.
- Ngoài ra, bạn cần đảm bảo đã cài đặt `kubectl` và `helm` tại thiết bị của bạn.
- Bên cạnh đó, bạn cũng có thể cài đặt các công cụ và thư viện khác mà bạn có thể sử dụng để giám sát và quản lý tài nguyên Kubernetes của mình:
 - `kubectl-view-allocations` plugin sử dụng để giám sát tài nguyên Kubernetes của bạn. Để biết thêm thông tin, vui lòng tham khảo thêm tại [kubectl-view-allocations](#).
- Bạn có thể thực hiện kiểm tra các cài đặt bên trên thông qua lệnh:

```
# Check kubectl CLI version
kubectl version

# Check Helm version
helm version

# Check kubectl-view-allocations version
kubectl-view-allocations --version
```

```
root@kubeconfig:/# kubectl version
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.29.1
root@kubeconfig:/#
root@kubeconfig:/# helm version
version.BuildInfo{Version:"v3.15.1", GitCommit:"e211f2aa62992bd72586b395de50979e31231829", GitTreeState:"clean", GoVersion:"go1.22.3"}
root@kubeconfig:/#
root@kubeconfig:/# kubectl-view-allocations --version
kubectl-view-allocations 0.19.2
root@kubeconfig:/#
```

- Trên Cluster vừa tạo, thực hiện kiểm tra node trong node group của bạn qua lệnh:

```
kubectl get nodes -owide
```

```
root@kubeconfig:/# kubectl get nodes -owide
NAME           STATUS   ROLES      AGE    VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION   CONTAINER-RUNTIME
vks-cuongdm3-grindelwald-rtx-2080ti-15c88   Ready    <none>    4h32m   v1.29.1   10.84.0.3   <none>        Ubuntu 22.04.4 LTS   5.15.0-101-generic   containerd://1.7.13
root@kubeconfig:/#
```

Cài đặt NVIDIA GPU Operator

- Nội dung bên dưới chúng tôi sẽ trực tiếp hướng dẫn bạn cài đặt NVIDIA GPU Operator, để biết thêm thông tin về plugin này, vui lòng tham khảo thêm tại [NVIDIA GPU Operator Documentation](#).
- Trên Cluster bạn đã tạo ở bước trên, thực hiện chạy lệnh:

```
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
```

```
root@kubeconfig:/# 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
Pulled: vcr.vngcloud.vn/81-vks-public/vks-helm-charts/gpu-operator:v24.3.0
Digest: sha256:402a8d7194b0b3c2ffa467c8fde58344303b3df200b0302b0bccd95c56202397
NAME: nvidia-gpu-operator
LAST DEPLOYED: Wed Jun 12 08:48:57 2024
NAMESPACE: gpu-operator
STATUS: deployed
REVISION: 1
TEST SUITE: None
root@kubeconfig:/#
```

- Hệ thống mất khoảng 5 - 10 phút để thực hiện cài đặt operator này, bạn hãy đợi tới khi việc cài đặt hoàn thành. Trong thời gian này, bạn có thể kiểm tra tất cả các pods trong namespace `gpu-operator` đang chạy thông qua lệnh:

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

NAME	READY	STATUS	RESTARTS	AGE
gpu-feature-discovery-gn8pn	1/1	Running	0	6m25s
gpu-operator-7cc57d576f-qcsdq	1/1	Running	0	7m
nvidia-container-toolkit-daemonset-pq9zk	1/1	Running	0	6m25s
nvidia-cuda-validator-smxth	0/1	Completed	0	52s
nvidia-dcgm-exporter-qzsk8	1/1	Running	0	6m25s
nvidia-device-plugin-daemonset-gncld	1/1	Running	0	6m25s
nvidia-driver-daemonset-w58gn	1/1	Running	0	6m41s
nvidia-gpu-operator-node-feature-discovery-gc-844bfbd5bd-q65pf	1/1	Running	0	7m
nvidia-gpu-operator-node-feature-discovery-master-6bb87956t6qb2	1/1	Running	0	7m
nvidia-gpu-operator-node-feature-discovery-worker-8dwjm	1/1	Running	0	7m
nvidia-operator-validator-d889g	1/1	Running	0	6m25s

- Operator sẽ gán label `nvidia.com/gpu` cho node trong node group của bạn, label này được NVIDIA GPU Operator sử dụng để identify nodes, bạn cũng có thể sử dụng label này để filter những node đang có NVIDIA GPU. Bạn có thể kiểm tra các node được gán nhãn này qua lệnh:

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

Ví dụ, đối với kết quả bên dưới, node trong cụm có label `nvidia.com/gpu`, có nghĩa là node đó có GPU. Các label này cũng cho biết nút này đang sử dụng 1 card GPU RTX 2080Ti, số lượng GPU có sẵn, Bộ nhớ GPU và các thông tin khác.

```
root@kubeconfig:/# kubectl get node -o json | jq '.items[].metadata.labels' | grep "nvidia.com"
{
    "nvidia.com/cuda.driver-version.full": "550.54.15",
    "nvidia.com/cuda.driver-version.major": "550",
    "nvidia.com/cuda.driver-version.minor": "54",
    "nvidia.com/cuda.driver-version.revision": "15",
    "nvidia.com/cuda.driver.major": "550",
    "nvidia.com/cuda.driver.minor": "54",
    "nvidia.com/cuda.driver.rev": "15",
    "nvidia.com/cuda.runtime.version.full": "12.4",
    "nvidia.com/cuda.runtime.version.major": "12",
    "nvidia.com/cuda.runtime.version.minor": "4",
    "nvidia.com/cuda.runtime.major": "12",
    "nvidia.com/cuda.runtime.minor": "4",
    "nvidia.com/gfd.timestamp": "1718182539",
    "nvidia.com/gpu-driver-upgrade-state": "upgrade-done",
    "nvidia.com/gpu.compute.major": "7",
    "nvidia.com/gpu.compute.minor": "5",
    "nvidia.com/gpu.count": "1",
    "nvidia.com/gpu.deploy.container-toolkit": "true",
    "nvidia.com/gpu.deploy.dcm": "true",
    "nvidia.com/gpu.deploy.dcm-exporter": "true",
    "nvidia.com/gpu.deploy.device-plugin": "true",
    "nvidia.com/gpu.deploy.driver": "true",
    "nvidia.com/gpu.deploy.gpu-feature-discovery": "true",
    "nvidia.com/gpu.deploy.node-status-exporter": "true",
    "nvidia.com/gpu.deploy.nvsm": "",
    "nvidia.com/gpu.deploy.operator-validator": "true",
    "nvidia.com/gpu.family": "turing",
    "nvidia.com/gpu.machine": "OpenStack-Compute",
    "nvidia.com/gpu.memory": "11264",
    "nvidia.com/gpu.present": "true",
    "nvidia.com/gpu.product": "NVIDIA-GeForce-RTX-2080-Ti",
    "nvidia.com/gpu.replicas": "1",
    "nvidia.com/gpu.sharing-strategy": "none",
    "nvidia.com/mig.capable": "false",
    "nvidia.com/mig.strategy": "single",
    "nvidia.com/mps.capable": "false",
    "nvidia.com/mps.strategy": "single"
}
root@kubeconfig:/#
```

- Trên pod `nvidia-device-plugin-daemonset` trong namespace `gpu-operator`, bạn có thể chạy lệnh `nvidia-smi` để kiểm tra thông tin GPU trên node:

```
POD_NAME=$(kubectl -n gpu-operator get pods -l app=nvidia-device-plugin-daemonset -o yaml | grep name | awk '{print $1}')
kubectl -n gpu-operator exec -it $POD_NAME -- nvidia-smi
```

```
root@kubeconfig:/# POD_NAME=$(kubectl -n gpu-operator get pods -l app=nvidia-device-plugin-daemonset -o yaml | grep name | awk '{print $1}')
root@kubeconfig:/# kubectl -n gpu-operator exec -it $POD_NAME -- nvidia-smi
Defaulted container "nvidia-device-plugin" out of: nvidia-device-plugin, toolkit-validation (init)
Wed Jun 12 09:26:01 2024
+-----+
| NVIDIA-SMI 550.54.15      Driver Version: 550.54.15     CUDA Version: 12.4 |
+-----+
| GPU Name      Persistence-M | Bus-Id      Disp.A  | Volatile Uncorr. ECC | | | | |
| Fan Temp  Perf  Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
|          |          |          |          |          |          | MIG M. |
+-----+
| 0  NVIDIA GeForce RTX 2080 Ti  On   00000000:00:05.0 Off | N/A       | 0%      Default |
| 27% 36C   P8    18W / 250W | 0MB / 11264MiB |          |          | N/A       |
+-----+
MIG does not support on
RTX-2080Ti model
+-----+
| Processes:                               GPU Memory |
| GPU  GI  CI      PID  Type  Process name        Usage  |
| ID  ID          ID           |
+-----+
| No running processes found
+-----+
root@kubeconfig:/#
```

Triển khai một GPU workload

Deploy Cuda VectorAdd Test

- Bạn có thể thực hiện deploy `cuda-vectoradd-test` như một workload mẫu trên cluster của bạn. Cụ thể bạn có thể tham khảo thêm về workload mẫu này tại [cuda-vectoradd-test.yaml](#).
- Thực hiện chạy các lệnh bên dưới để deploy workload và kiểm tra các pods đã được khởi chạy:

```
# Apply the manifest
kubectl apply -f \
https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/main/nvidia-gpu/manifest/cuda-vectoradd-test.yaml

# 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
```



The screenshot shows a terminal window with a light gray background and a dark gray header bar. The header bar has three colored dots (red, yellow, green) on the left and a search icon on the right. The main area of the terminal shows the following command-line session:

```
root@kubeconfig:/# kubectl apply -f \
https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/main/nvidia-gpu/manifest/cuda-vectoradd-test.yaml
pod/cuda-vectoradd created
root@kubeconfig:/#
root@kubeconfig:/# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
cuda-vectoradd  0/1     Completed   0          56s
root@kubeconfig:/#
root@kubeconfig:/# kubectl logs cuda-vectoradd
[Vector addition of 50000 elements]
Copy input data from the host memory to the CUDA device
CUDA kernel launch with 196 blocks of 256 threads
Copy output data from the CUDA device to the host memory
Test PASSED
Done
root@kubeconfig:/#
```

Deploy TensorFlow Test

- Ngoài Cuda VectorAdd Test, bạn cũng có thể deploy TensorFlow như một workload trên Cluster của bạn. Cụ thể bạn có thể tham khảo về workload mẫu này tại [tensorflow-gpu.yaml](#).
- Thực hiện chạy các lệnh bên dưới để deploy workload và kiểm tra các pods đã được khởi chạy:

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

# 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-name> -- 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
```

```
root@kubeconfig:/# kubectl apply -f https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/main/nvidia-gpu/manifest/tensorflow-gpu.yaml
deployment.apps/tensorflow-gpu created
root@kubeconfig:/#
root@kubeconfig:/# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
tensorflow-gpu-55c85c4f57-sjhqb   1/1     Running   0          5s
root@kubeconfig:/#
root@kubeconfig:/# kubectl -n gpu-operator exec -it nvidia-driver-daemonset-w58gm -- nvidia-smi
Thu Jun 13 03:55:47 2024
+-----+-----+-----+-----+-----+
| NVIDIA-SMI 550.54.15      Driver Version: 550.54.15     CUDA Version: 12.4      |
| GPU  Name        Persistence-M  Bus-Id      Disp.A  Volatile Uncorr. ECC  |
| Fan  Temp  Perf  Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. | MIG M. |
|-----+-----+-----+-----+-----+
| 0  NVIDIA GeForce RTX 2080 Ti   On   00000000:00:05.0 Off   N/A       N/A      |
| 27%  41C   P2    64W / 250W | 9955MB / 11264MB | 0%      Default   N/A      |
+-----+-----+-----+-----+-----+
+-----+
| Processes:                               GPU Memory |
| GPU  GI CI   PID  Type  Process name        Usage  |
| ID  ID
+-----+
| 0  N/A N/A 883101  C  python               9952MB |
+-----+
root@kubeconfig:/# kubectl logs tensorflow-gpu-55c85c4f57-sjhqb --tail 20
2024-06-13 03:55:52.496448: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-06-13 03:55:55.789904: I external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-13 03:55:55.802275: I external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-13 03:55:55.802632: I external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-13 03:55:55.803819: I external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-13 03:55:55.804211: I external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-13 03:55:55.804495: I external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-13 03:55:56.114405: I external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-13 03:55:56.114641: I external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-13 03:55:56.114840: I external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-13 03:55:56.114995: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1928] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 9796 MB memory: -> device 0, name: NVIDIA GeForce RTX 2080 Ti, pci bus id: 0000:00:05.0, compute capability: 7.5
root@kubeconfig:/#
```

Thiết lập 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:
- Chiến lược GPU Sharing cho phép nhiều containers có thể sử dụng chung một node GPU nhằm mục đích tiết kiệm chi phí của bạn. Bảng sau đây tóm tắt sự khác biệt giữa các chiến lược GPU Sharing được NVIDIA hỗ trợ:

Sharing mode
Multi-instance GPU (MIG)

Supported by VKS



Workload isolation level
Best

Pros

- Processes are executed in parallel
- Full isolation (dedicated memory and compute resources)

Cons

- Supported by fewer GPU models (only Ampere or more recent architectures)
- Coarse-grained control over memory and compute resources

Suitable for these workloads

- Recommended for workloads running in parallel and that need certain resiliency and QoS. For example, when running AI inference workloads, multi-instance GPU multi-instance GPU allows multiple inference queries to run simultaneously for quick responses, without slowing each other down.

Sharing mode
GPU Time-slicing

Supported by VKS



Workload isolation level
None

Pros

- Processes are executed concurrently
- Supported by older GPU architectures (Pascal or newer)

Cons

- No resource limits
- No memory isolation
- Lower performance due to context-switching overhead

Suitable for these workloads

- 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 GPU access might not be necessary, for example, when multiple users test or prototype workloads.
- Workloads that use time-sharing need to tolerate certain performance and latency compromises.

Sharing mode
Multi-process server (MPS)

Supported by VKS



Workload isolation level
Medium

Pros

- Processes are executed parallel
- Fine-grained control over memory and compute resources allocation

Cons

- No error isolation and memory protection

Suitable for these workloads

- Recommended for batch processing 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 MPS is optimal for cooperative processes acting as a single application. For example, MPI jobs with inter-MPI rank parallelism. With these jobs, each small CUDA process (typically MPI ranks) can run concurrently on the GPU to fully saturate the whole GPU.
- Workloads that use CUDA MPS need to tolerate the [memory protection and error containment limitations](#).

GPU time-slicing

- **GPU time-slicing** là kỹ thuật chia sẻ tài nguyên GPU giữa nhiều tiến trình hoặc pod trong Kubernetes bằng cách phân bổ thời gian sử dụng GPU cho từng tiến trình theo chu kỳ.

Configure GPU time-slicing

- Để sử dụng GPU time-slicing cho cluster của bạn, bạn cần tạo tệp tin `ConfigMap` theo mẫu bên dưới để định nghĩa cấu hình time-slicing. Trong đó:
 - `replicas` : field chỉ định số lượng pods có thể share chung GPU, tối đa bạn có thể thiết lập 48 pods share chung một GPU.
 - `name`: mặc định `nvidia.com/gpu` - label sử dụng để filter node có GPU.
 - `minStrategy`: mặc định `none` do GPU hiện tại chưa hỗ trợ MIG.

```

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

```

- Hoặc bạn có thể chạy câu lệnh bên dưới để apply cấu hình trên cho tất cả các node trong cluster của bạn có label `nvidia.com/gpu`.

```

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

```

- Sau đó, sử dụng lệnh `kubectl patch` để patch ClusterPolicy và bật GPU time-slicing. Lệnh patch chỉ hoạt động khi `ClusterPolicy` có trạng thái `Ready`.

```

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

# 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}}}'"

```

```

root@kubeconfig:/# kubectl -n gpu-operator create -f \
https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/main/nvidia-gpu/manifest/time-slicing-config-all.yaml
configmap/gpu-sharing-config created
root@kubeconfig:/#
root@kubeconfig:/# kubectl -n gpu-operator get cm
NAME                                     DATA   AGE
default-gpu-clients                      1      22h
default-mig-parted-config                 1      22h
gpu-sharing-config                         1      9s
kube-root-ca.crt
nvidia-container-toolkit-entrypoint       1      22h
nvidia-device-plugin-entrypoint          1      22h
nvidia-gpu-operator-node-feature-discovery-master-conf 1      22h
nvidia-gpu-operator-node-feature-discovery-topology-updater-conf 1      22h
nvidia-gpu-operator-node-feature-discovery-worker-conf    1      22h
nvidia-mig-manager-entrypoint            1      22h
root@kubeconfig:/#
root@kubeconfig:/# kubectl patch clusterpolicies.nvidia.com/cluster-policy \
-n gpu-operator --type merge \
-p '{"spec": {"devicePlugin": {"config": {"name": "gpu-sharing-config", "default": "any"}}}}'
clusterpolicy.nvidia.com/cluster-policy patched (no change)
root@kubeconfig:/#
root@kubeconfig:/# kubectl patch clusterpolicies.nvidia.com/cluster-policy \
-n gpu-operator --type merge \
-p '{"spec": {"dcgmExporter": {"enabled": false}}}'
clusterpolicy.nvidia.com/cluster-policy patched (no change)
root@kubeconfig:/#
root@kubeconfig:/# kubectl get clusterpolicy
NAME        STATUS   AGE
cluster-policy   ready   2024-06-12T08:48:59Z
root@kubeconfig:/#
root@kubeconfig:/# kubectl -n gpu-operator get pods
NAME                           READY   STATUS    RESTARTS   AGE
gpu-feature-discovery-7nmbj   2/2     Running   0          8m59s
gpu-operator-7cc57d576f-qcsdq 1/1     Running   0          22h
nvidia-container-toolkit-daemonset-pq9zk 1/1     Running   0          22h
nvidia-cuda-validator-smxth  0/1     Completed  0          22h
nvidia-device-plugin-daemonset-4d669 2/2     Running   0          8m59s
nvidia-driver-daemonset-w58gn  1/1     Running   0          22h
nvidia-gpu-operator-node-feature-discovery-gc-844bfbd5bd-q65pf 1/1     Running   0          22h
nvidia-gpu-operator-node-feature-discovery-master-6bb87956t6qb2 1/1     Running   0          22h
nvidia-gpu-operator-node-feature-discovery-worker-8dwjn   1/1     Running   0          22h
nvidia-operator-validator-d889g  1/1     Running   0          22h
root@kubeconfig:/#

```

The ConfigMap for GPU sharing mode

These two pods are restarted when you patch the ClusterPolicy resource

Verify GPU time-slicing

- Bây giờ, bạn có thể deploy một application có 5 pods sử dụng GPU bằng cách apply yaml theo mẫu [time-slicing-verification.yaml](#). Bởi vì cấu hình config map thiết lập bên trên đang có replica = 4 nên pod thứ 5 sẽ có trạng thái Pending state.
- Lần lượt deploy và verify GPU time-slicing của bạn qua các lệnh:

```

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

# 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

```

```

root@kubeconfig:/# kubectl apply -f \
  https://raw.githubusercontent.com/nvidia/kubernetes-sample-apps/main/nvidia-gpu/manifest/time-slicing-verification.yaml
deployment.apps/time-slicing-verification created
root@kubeconfig:/#
root@kubeconfig:/# kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
time-slicing-verification-5849747c99-2xgq4   1/1     Running   0          19s
time-slicing-verification-5849747c99-lq75b   1/1     Running   0          19s
time-slicing-verification-5849747c99-n7qw4   0/1     Pending   0          19s
time-slicing-verification-5849747c99-tlqsg   1/1     Running   0          19s
time-slicing-verification-5849747c99-tlqsg   1/1     Running   0          19s
root@kubeconfig:/#
root@kubeconfig:/# kubectl logs time-slicing-verification-5849747c99-2xgq4 --tail 10
CUDA kernel launch with 196 blocks of 256 threads
Copy output data from the CUDA device to the host memory
Test PASSED
Done
[Vector addition of 50000 elements]
Copy input data from the host memory to the CUDA device
CUDA kernel launch with 196 blocks of 256 threads
Copy output data from the CUDA device to the host memory
Test PASSED
Done
root@kubeconfig:/# kubectl events | grep "FailedScheduling"
2m48s (x2 over 2m49s) Warning FailedScheduling Pod/time-slicing-verification-5849747c99-n7qw4 0/1 nodes are available: 1 Insufficient nvidia.com/gpu. preemption: 0/1
nodes are available: 1 No preemption victims found for incoming pod.
root@kubeconfig:/#
root@kubeconfig:/#

```

This pod CAN NOT be scheduled because of only 4 replicas to share

Multi-process server (MPS)

- **NVIDIA Multi-Process Server (MPS)** là một giải pháp thay thế và tương thích nhị phân cho giao diện lập trình ứng dụng CUDA (API). Kiến trúc runtime của MPS được thiết kế để cho phép các ứng dụng CUDA đa quy trình hợp tác, thường là các tác vụ MPI, tận dụng khả năng Hyper-Q trên các GPU NVIDIA mới nhất (Kepler và sau đó). Hyper-Q cho phép các kernel CUDA được xử lý đồng thời trên cùng một GPU, điều này có thể mang lại lợi ích về hiệu suất khi dung lượng tính toán GPU không được sử dụng đầy đủ bởi một quy trình ứng dụng duy nhất. Bạn có thể tham khảo thêm về MPS tại [Multi-Process Service \(MPS\)](#).

Configure MPS

- Để sử dụng MPS cho cluster của bạn, bạn cần tạo tệp tin `ConfigMap` theo mẫu bên dưới để định nghĩa cấu hình MPS. Trong đó:
 - `replicas`: field chỉ định số lượng pods có thể share chung GPU.
 - `name`: mặc định `nvidia.com/gpu` - label sử dụng để filter node có GPU.
 - `minStrategy`: mặc định `none` do GPU hiện tại chưa hỗ trợ MIG.

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: gpu-sharing-config
data:
  any-mps: |-  

    version: v1  

    flags:  

      migStrategy: none          # MIG strategy is not used, this field SHOULD  

    sharing:  

      mps:                      # Enable MPS for the GPU  

      resources:  

        - name: nvidia.com/gpu   # Only apply for the node with the node.status  

          replicas: 4            # Allow 4 pods to share the GPU

```

- Hoặc bạn có thể chạy câu lệnh bên dưới để apply cấu hình trên cho tất cả các node trong cluster của bạn có label `nvidia.com/gpu`.

```

# 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/main/nvnic

```

- Sau đó, sử dụng lệnh `kubectl patch` để patch ClusterPolicy và bật GPU MPS. Lệnh patch chỉ hoạt động khi `ClusterPolicy` có trạng thái `Ready`.

```

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

# 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

```

Verify MPS

- Bây giờ, bạn có thể deploy một application có 5 pods sử dụng GPU bằng cách apply yaml theo mẫu [mps-verification.yaml](#). Bởi vì cấu hình config map thiết lập bên trên đang có replica = 4 nên pod thứ 5 sẽ có trạng thái `Pending` state.
- Lần lượt deploy và verify GPU MPS của bạn qua các lệnh:

```

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

```

```

root@kubeconfig:~# kubectl apply -f \
https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/main/nvidia-gpu/manifest/mps-verification.yaml
job.batch/nbody-sample created
root@kubeconfig:~#
root@kubeconfig:~# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
nbody-sample-4d7sq  1/1    Running   0          9s
nbody-sample-7k6ln  1/1    Running   0          9s
nbody-sample-dshfF  1/1    Running   0          9s
nbody-sample-f2b72  0/1    Pending   0          9s
nbody-sample-q9qbd  1/1    Running   0          9s
root@kubeconfig:~#
root@kubeconfig:~# kubectl logs -l job-name=nbody-sample
> Windowed mode
> Simulation data stored in video memory
> Single precision floating point simulation
> 1 Devices used for simulation
GPU Device 0: "Turing" with compute capability 7.5

> Compute 7.5 CUDA device: [NVIDIA GeForce RTX 2080 Ti]
16384 bodies, total time for 5000 iterations: 11379.522 ms
= 117.947 billion interactions per second
= 2358.934 single-precision GFLOP/s at 20 flops per interaction
> Windowed mode
> Simulation data stored in video memory
> Single precision floating point simulation
> 1 Devices used for simulation
GPU Device 0: "Turing" with compute capability 7.5

> Compute 7.5 CUDA device: [NVIDIA GeForce RTX 2080 Ti]
16384 bodies, total time for 5000 iterations: 11330.421 ms
= 118.458 billion interactions per second
= 2369.157 single-precision GFLOP/s at 20 flops per interaction
> Windowed mode
> Simulation data stored in video memory
> Single precision floating point simulation
> 1 Devices used for simulation
GPU Device 0: "Turing" with compute capability 7.5

> Compute 7.5 CUDA device: [NVIDIA GeForce RTX 2080 Ti]
16384 bodies, total time for 5000 iterations: 11317.587 ms
= 118.592 billion interactions per second
= 2371.844 single-precision GFLOP/s at 20 flops per interaction
> Windowed mode
> Simulation data stored in video memory

```

Chú ý:

- Trên một node GPU chỉ có thể sử dụng một trong hai chiến lược chia sẻ GPU là Time Slicing hoặc MPS, không thể sử dụng đồng thời cả hai. Lý do là vì cả hai chiến lược đều sử dụng cùng một tài nguyên GPU vật lý, và việc sử dụng cả hai đồng thời sẽ dẫn đến xung đột và hiệu suất không mong muốn.
- Tuy nhiên, nếu bạn có một node group gồm nhiều node GPU, bạn có thể sử dụng hai chiến lược khác nhau trên hai node riêng biệt. Ví dụ: bạn có thể sử dụng Time Slicing trên một node để chia sẻ GPU và sử dụng MPS trên node còn lại để chia sẻ GPU cho các ứng dụng khác. Chi tiết tham khảo mục bên dưới.

Applying Multiple Node-Specific Configurations

- Nếu bạn có một node group gồm nhiều node GPU, bạn có thể sử dụng hai chiến lược khác nhau trên hai node riêng biệt. Ví dụ: bạn có thể sử dụng Time Slicing trên một node để chia sẻ GPU và sử dụng MPS trên node còn lại để chia sẻ GPU cho các ứng dụng khác. Ví dụ, tôi đã tạo 2 node group trong một Cluster với các thông số như sau:
 - NodeGroup 1 có instance GPU RTX 2080Ti.
 - NodeGroup 2 có instance GPU RTX 4090.

Hiện tại, bạn mong muốn:

- NodeGroup 1 có instance GPU RTX 2080Ti sẽ chạy 4 pods sharing GPU sử dụng time-slicing.
- NodeGroup 2 có instance GPU RTX 4090 sẽ chạy 8 pods sharing GPU sử dụng MPS.

Configure Multiple Node-Specific Configurations

- Để sử dụng GPU time-slicing, MPS cho cluster của bạn, bạn cần tạo tệp tin `ConfigMap` theo mẫu bên dưới để định nghĩa cấu hình time-slicing, MPS mong muốn.

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

- Hoặc bạn có thể chạy câu lệnh bên dưới để apply cấu hình trên cho tất cả các node trong cluster của bạn có label `nvidia.com/gpu` .

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

- Sau đó, sử dụng lệnh `kubectl patch` để patch ClusterPolicy và bật GPU time-slicing, MPS. Lệnh patch chỉ hoạt động khi `ClusterPolicy` có trạng thái `Ready` .

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



```
# 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
```



```
root@kubeconfig:/# kubectl -n gpu-operator create -f \
https://raw.githubusercontent.com/vngcloud/kubernetes-sample-apps/main/nvidia-gpu/manifest/multiple-gpu-sharing.yaml
configmap/gpu-multi-sharing-config created
root@kubeconfig:/#
root@kubeconfig:/# kubectl patch clusterpolicies.nvidia.com/cluster-policy \
-n gpu-operator --type merge \
-p '{"spec": {"devicePlugin": {"config": {"name": "gpu-multi-sharing-config"}}}}'
clusterpolicy.nvidia.com/cluster-policy patched
root@kubeconfig:/#
root@kubeconfig:/# kubectl patch clusterpolicies.nvidia.com/cluster-policy \
-n gpu-operator --type merge \
-p '{"spec": {"dcgmExporter": {"enabled": false}}}'
clusterpolicy.nvidia.com/cluster-policy patched (no change)
root@kubeconfig:/#
root@kubeconfig:/# kubectl get clusterpolicy
NAME      STATUS   AGE
cluster-policy ready   2024-06-12T08:48:59Z
The ClusterPolicy status MUST be ready.
Otherwise you CANNOT use NVIDIA GPU to do anything.
```

- Bây giờ, bạn cần thêm label cho node với tên mà bạn chỉ định tại trong file `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-2080ti
kubectl label node <node-name> nvidia.com/device-plugin.config=rtx-4090
```

```
root@kubeconfig:/# kubectl get nodes
NAME           STATUS   ROLES   AGE    VERSION
vks-cuongdm3-grindelwald-rtx-2080ti-15c88  Ready    <none>  30h   v1.29.1
vks-cuongdm3-grindelwald-rtx-4090-0ebbd  Ready    <none>  20m   v1.29.1
root@kubeconfig:/# kubectl label node vks-cuongdm3-grindelwald-rtx-2080ti-15c88 nvidia.com/device-plugin.config=rtx-2080ti
node/vks-cuongdm3-grindelwald-rtx-2080ti-15c88 labeled
root@kubeconfig:/#
root@kubeconfig:/# kubectl label node vks-cuongdm3-grindelwald-rtx-4090-0ebbd nvidia.com/device-plugin.config=rtx-4090
node/vks-cuongdm3-grindelwald-rtx-4090-0ebbd labeled
root@kubeconfig:/#
```

Verify Multiple Node-Specific Configurations

- Bên dưới đây là câu lệnh chúng tôi hướng dẫn bạn training MNIST model trong TensorFlow sử dụng 2 node GPU RTX 2080Ti và RTX 4090. Node RTX 2080Ti sẽ được share chung cho 4 pods sử dụng Time-slicing và node RTX 4090 sẽ được share chung cho 8 pods sử dụng MPS. Tham khảo thêm về TensorFlow tại [tensorflow-mnist-sample.yaml](#).

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

# 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 terminal window shows the following commands and their outputs:

```

root@kubeconfig:/# kubectl apply -f \
https://github.com/vngcloud/kubernetes-sample-apps/raw/main/nvidia-gpu/manifest/tensorflow-mnist-sample.yaml
deployment.apps/tensorflow-mnist created
root@kubeconfig:/#
root@kubeconfig:/# kubectl get pods -owide
NAME          READY   STATUS    RESTARTS   AGE   IP
tensorflow-mnist-c98bcc99c-44cjg  1/1    Running   0          82s   172.17.193.89
tensorflow-mnist-c98bcc99c-5zdnq  1/1    Running   0          82s   172.17.193.87
tensorflow-mnist-c98bcc99c-6j9qj  1/1    Running   0          82s   172.17.193.85
tensorflow-mnist-c98bcc99c-8b5zj  1/1    Running   0          82s   172.17.193.86
tensorflow-mnist-c98bcc99c-cztvx  1/1    Running   0          82s   172.17.227.50
tensorflow-mnist-c98bcc99c-fdst5  1/1    Running   0          82s   172.17.227.49
tensorflow-mnist-c98bcc99c-p6qw9  1/1    Running   0          82s   172.17.193.84
tensorflow-mnist-c98bcc99c-jtqq7  1/1    Running   0          82s   172.17.193.83
tensorflow-mnist-c98bcc99c-jxs68  1/1    Running   0          82s   172.17.193.82
tensorflow-mnist-c98bcc99c-trc6w  1/1    Running   0          82s   172.17.193.88
root@kubeconfig:/#
root@kubeconfig:/# kubectl logs tensorflow-mnist-c98bcc99c-jtqq7 --tail 20
WARNING:tensorflow:From tensorflow-sample-code/tfjob/docker/mnist/main.py:120: softmax_cross_entropy_with_logits (from tensorflow.python.ops.nn_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.
See tf.nn.softmax_cross_entropy_with_logits_v2.

2024-06-14 02:21:00.022580: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 AVX512F FMA
2024-06-14 02:21:00.232359: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:895] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2024-06-14 02:21:00.232888: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1105] Found device 0 with properties:
name: NVIDIA GeForce RTX 4090 major: 8 minor: 9 memoryClockRate(GHz): 2.52
pciBusID: 0000:00:05.0
totalMemory: 23.64GiB freeMemory: 2.94GiB
2024-06-14 02:21:00.232910: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1195] Creating TensorFlow device (/device:GPU:0) -> (device: 0, name: NVIDIA GeForce RTX 4090, p
root@kubeconfig:/#

```

The GPU resource visualization chart shows two nodes (vks) each with four GPUs (RTX 4090-0ebbd). The GPUs are represented by colored squares (blue, orange, red, green).

Giám sát hoạt động GPU Resources

- Giám sát tài nguyên GPU NVIDIA trong cụm Kubernetes là điều cần thiết để đảm bảo hiệu suất tối ưu, sử dụng tài nguyên hiệu quả và giải quyết vấn đề một cách chủ động. Sau đây là hướng dẫn của chúng tôi về việc thiết lập và sử dụng Prometheus và NVIDIA Data Center GPU Manager (DCGM) để giám sát tài nguyên GPU trong môi trường Kubernetes.
- Đầu tiên, bạn cần cài đặt **Prometheus Stack** và **Prometheus Adapter** để tích hợp với Kubernetes API server qua lệnh:

```

# 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-stack \
  --version 60.0.2 \
  --set prometheus.prometheusSpec.serviceMonitorSelectorNilUsesHelmValues=false

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

```

```

root@kubeconfig:/# helm install --wait prometheus-stack \
--namespace prometheus --create-namespace \
oci://vcr.vngcloud.vn/81-vks-public/vks-helm-charts/kube-prometheus-stack \
--version 60.0.2 \
--set prometheus.prometheusSpec.serviceMonitorSelectorNilUsesHelmValues=false
Pulled: vcr.vngcloud.vn/81-vks-public/vks-helm-charts/kube-prometheus-stack:60.0.2
Digest: sha256:b429527b2df70b7ea8264e435495f0c2283d2d46465e9166c0fb7b20dc26d34e
NAME: prometheus-stack
LAST DEPLOYED: Fri Jun 14 02:50:03 2024
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace prometheus get pods -l "release=prometheus-stack"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
root@kubeconfig:/# helm install --wait prometheus-service=(kubectl get svc -n prometheus -l app=kube-prometheus-stack-prometheus -ojsonpath='{range .items[*]}{.metadata.name}{"\n"}{end}')
helm install --wait prometheus-adapter \
--namespace prometheus --create-namespace \
oci://vcr.vngcloud.vn/81-vks-public/vks-helm-charts/prometheus-adapter \
--version 4.10.0 \
--set prometheus.url=http://$(prometheus_service).prometheus.svc.cluster.local
Pulled: vcr.vngcloud.vn/81-vks-public/vks-helm-charts/prometheus-adapter:v4.10.0
Digest: sha256:6e6636831cbdb8e9c7db07654a33e2e4298d21cf71bf4afdd5b52d6ec555736d
NAME: prometheus-adapter
LAST DEPLOYED: Fri Jun 14 02:58:39 2024
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
prometheus-adapter has been deployed.
In a few minutes you should be able to list metrics using the following command(s):

  kubectl get --raw /apis/custom.metrics.k8s.io/v1beta1
root@kubeconfig:/#

```

- Sau khi cài đặt thành công, chạy lệnh bên dưới để kiểm tra các resource Prometheus đang chạy:

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

```

root@kubeconfig:/# kubectl -n prometheus get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/alertmanager-prometheus-stack-kube-prom-alertmanager-0   2/2     Running   0          15m
pod/prometheus-adapter-556f4bcd8b-nrc7r   1/1     Running   0          8m22s
pod/prometheus-prometheus-stack-kube-prom-prometheus-0   2/2     Running   0          15m
pod/prometheus-stack-grafana-6c988fc844-8rxcq   3/3     Running   0          16m
pod/prometheus-stack-kube-prom-operator-7c6d69d64c-7k6zj   1/1     Running   0          16m
pod/prometheus-stack-kube-state-metrics-58575d877f-mxqg7   1/1     Running   0          16m
pod/prometheus-stack-prometheus-node-exporter-9p6z7   1/1     Running   0          16m
pod/prometheus-stack-prometheus-node-exporter-b8c65   1/1     Running   0          16m

NAME                           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)           AGE
service/alertmanager-operated   ClusterIP   <none>        9093/TCP,9094/TCP,9094/UDP   15m
service/prometheus-adapter     ClusterIP   10.96.194.7   <none>        443/TCP          8m23s
service/prometheus-operated   ClusterIP   <none>        9090/TCP          15m
service/prometheus-stack-grafana   ClusterIP   10.96.198.21   <none>        80/TCP          16m
service/prometheus-stack-kube-prom-alertmanager   ClusterIP   10.96.86.127   <none>        9093/TCP,8080/TCP   16m
service/prometheus-stack-kube-prom-operator   ClusterIP   10.96.35.125   <none>        443/TCP          16m
service/prometheus-stack-kube-prom-prometheus   ClusterIP   10.96.77.128   <none>        9090/TCP,8080/TCP   16m
service/prometheus-stack-kube-state-metrics   ClusterIP   10.96.172.235   <none>        8080/TCP          16m
service/prometheus-stack-prometheus-node-exporter   ClusterIP   10.96.42.179   <none>        9100/TCP          16m

NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-stack-prometheus-node-exporter   2         2         2         2             2           kubernetes.io/os=linux   16m

NAME              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-adapter   1/1     1           1           8m23s
deployment.apps/prometheus-stack-grafana   1/1     1           1           16m
deployment.apps/prometheus-stack-kube-prom-operator   1/1     1           1           16m
deployment.apps/prometheus-stack-kube-state-metrics   1/1     1           1           16m

NAME              DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-adapter-556f4bcd8b   1         1         1         8m23s
replicaset.apps/prometheus-stack-grafana-6c988fc844   1         1         1         16m
replicaset.apps/prometheus-stack-kube-prom-operator-7c6d69d64c   1         1         1         16m
replicaset.apps/prometheus-stack-kube-state-metrics-58575d877f   1         1         1         16m

NAME              READY   AGE
statefulset.apps/alertmanager-prometheus-stack-kube-prom-alertmanager   1/1     15m
statefulset.apps/prometheus-stack-kube-prom-prometheus   1/1     15m
root@kubeconfig:/#

```

- Tiếp theo, bạn cần chạy lệnh bên dưới để enable DCGM exporter để giám sát GPU resources trên cluster của bạn:

```

# 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 you MUST
(about 1-3 mins) for the DCGM exporter to be ready

```

kubectl get --raw /apis/custom.metrics.k8s.io/v1beta1 | jq -r . | grep DCGM

```

```

root@kubeconfig:/# kubectl patch clusterpolicies.nvidia.com/cluster-policy \
-n gpu-operator --type merge \
-p '{"spec": {"dcgmExporter": {"enabled": true}}}'
clusterpolicy.nvidia.com/cluster-policy patched (no change)
root@kubeconfig:/#
root@kubeconfig:/# kubectl get --raw /apis/custom.metrics.k8s.io/v1beta1 | jq -r . | grep DCGM
{
  "name": "services/DCGM_FI_DEV_GPU_UTIL",
  "name": "pods/DCGM_FI_DEV_SM_CLOCK",
  "name": "services/DCGM_FI_DEV_XID_ERRORS",
  "name": "jobs.batch/DCGM_FI_DEV_TOTAL_ENERGY_CONSUMPTION",
  "name": "jobs.batch/DCGM_FI_DEV_PCIE_REPLAY_COUNTER",
  "name": "pods/DCGM_FI_DEV_VGPU_LICENSE_STATUS",
  "name": "services/DCGM_FI_DEV_POWER_USAGE",
  "name": "namespaces/DCGM_FI_DEV_DEC_UTIL",
  "name": "jobs.batch/DCGM_FI_DEV_XID_ERRORS",
  "name": "jobs.batch/DCGM_FI_DEV_POWER_USAGE",
  "name": "services/DCGM_FI_DEV_MEMORY_TEMP",
  "name": "pods/DCGM_FI_DEV_PCIE_REPLAY_COUNTER",
  "name": "namespaces/DCGM_FI_DEV_MEM_COPY_UTIL",
  "name": "services/DCGM_FI_DEV_MEM_CLOCK",
  "name": "services/DCGM_FI_DEV_FB_USED",
  "name": "services/DCGM_FI_DEV_GPU_TEMP",
  "name": "namespaces/DCGM_FI_DEV_XID_ERRORS",
  "name": "jobs.batch/DCGM_FI_DEV_SM_CLOCK",
  "name": "namespaces/DCGM_FI_DEV_FB_FREE",
  "name": "jobs.batch/DCGM_FI_DEV_MEM_COPY_UTIL",
  "name": "pods/DCGM_FI_DEV_MEM_CLOCK",
  "name": "services/DCGM_FI_DEV_FB_FREE",
  "name": "namespaces/DCGM_FI_DEV_GPU_UTIL",
  "name": "pods/DCGM_FI_DEV_DEC_UTIL",
  "name": "services/DCGM_FI_DEV_NVLINK_BANDWIDTH_TOTAL",
  "name": "namespaces/DCGM_FI_DEV_ROW_REMAP_FAILURE",
  "name": "jobs.batch/DCGM_FI_DEV_FB_FREE",
  "name": "services/DCGM_FI_DEV_MEM_COPY_UTIL",
  "name": "jobs.batch/DCGM_FI_DEV_MEMORY_TEMP",
  "name": "jobs.batch/DCGM_FI_DEV_VGPU_LICENSE_STATUS",
  "name": "jobs.batch/DCGM_FI_DEV_UNCORRECTABLE_REMAPPED_ROWS",
  "name": "namespaces/DCGM_FI_DEV_POWER_USAGE",
  "name": "pods/DCGM_FI_DEV_POWER_USAGE",
  "name": "pods/DCGM_FI_DEV_MEMORY_TEMP",
  "name": "jobs.batch/DCGM_FI_DEV_FB_USED",
  "name": "namespaces/DCGM_FI_DEV_ENC_UTIL",
}

```

- Bây giờ, bạn hãy chạy lệnh bên dưới để chuyển Prometheus Adapter tới máy localhost của bạn và sau đó kiểm tra các metrics đã thu thập được thông qua <http://localhost:9090>

```

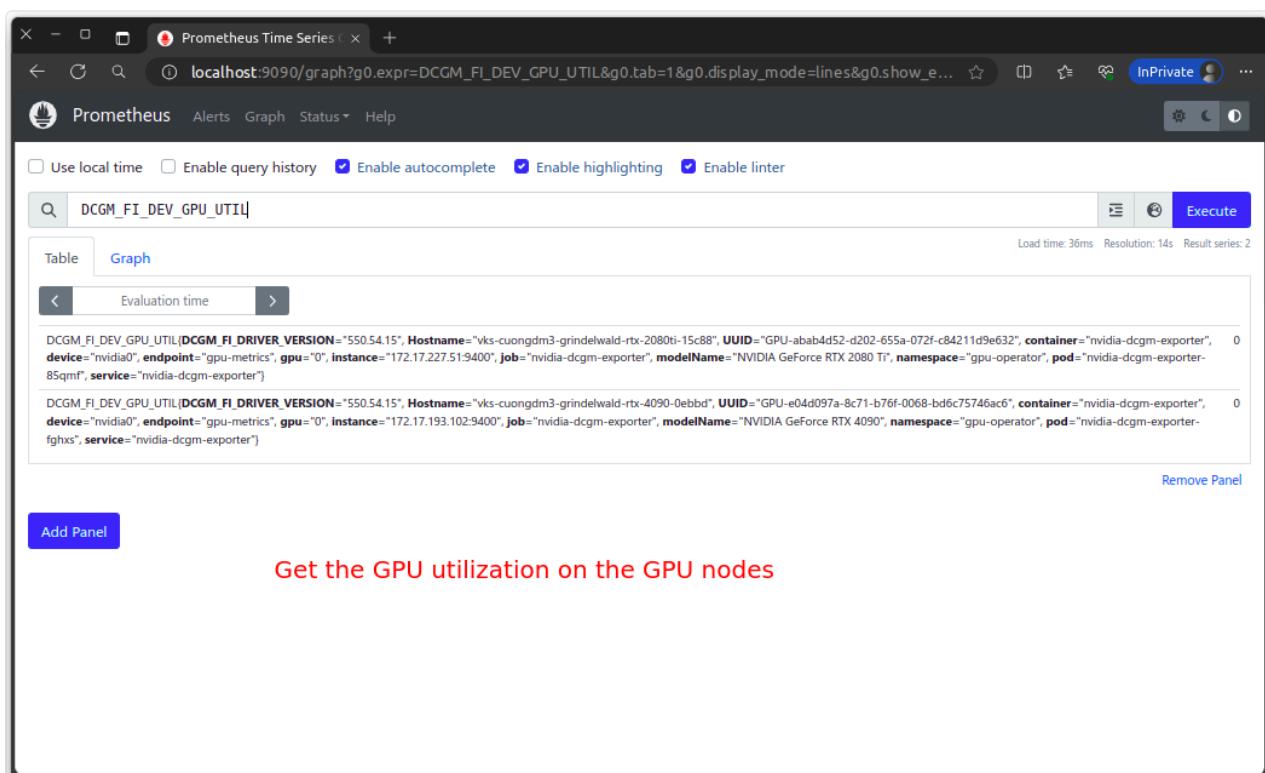
# Forward the Prometheus Adapter to your local machine
kubectl -n prometheus \
port-forward svc/prometheus-stack-kube-prom-prometheus 9090:9090

```

```

root@kubeconfig:/# kubectl -n prometheus port-forward svc/prometheus-stack-kube-prom-prometheus 9090:9090
Forwarding from 127.0.0.1:9090 -> 9090
Forwarding from [::1]:9090 -> 9090

```



- Bên dưới là danh sách một vài GPU Metrics được sử dụng thường xuyên. Tham khảo danh sách GPU metric đầy đủ tại [Field Identifiers](#).

- **Bảng 1: Usage**

Metric Name	Metric Type	Unit	Description
DCGM_FI_DEV_GPU_UTIL	Gauge	Percentage	GPU usage.
DCGM_FI_DEV_MEM_COPY_UTIL	Gauge	Percentage	Memory usage.
DCGM_FI_DEV_ENC_UTIL	Gauge	Percentage	Encoder usage.
DCGM_FI_DEV_DEC_UTIL	Gauge	Percentage	Decoder usage.

- **Bảng 2: Memory**

Metric Name	Metric Type	Unit	Description
DCGM_FI_DEV_FB_FREE	Gauge	MB	Number of remaining frame buffers. The frame buffer is called VRAM.
DCGM_FI_DEV_FB_USED	Gauge	MB	Number of used frame buffers. The value is the

same as the value of memory-usage in the nvidia-smi command.

- **Bảng 3:** Temperature and power

Metric Name	Metric Type	Unit	Description
DCGM_FI_DEV_GPU_TEMP	Gauge	°C	Current GPU temperature of the device.
DCGM_FI_DEV_POWER_USAGE	Gauge	W	Power usage of the device.

Autoscaling GPU Resources

- Để enable autoscaling GPU Resource, bạn cần:
 - Bật tính năng **Autoscale** cho GPU Nodegroups theo hướng dẫn tại [đây](#).
 - Cài đặt **Keda** thông qua Helm chart trên Cluster của bạn qua lệnh:

```
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
```



```

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
          metricName: engine_active
          query: sum(DCGM_FI_DEV_GPU_UTIL) / count(DCGM_FI_DEV_GPU_UTIL) / 100
          threshold: '0.5' # Scale the GPU Nodegroup when the GPU usage is greater than 50%
      - type: prometheus
        metadata: # prometheus-stack-kube-prom-prometheus
          serverAddress: http://prometheus-stack-kube-prom-prometheus.prometheus
          metricName: engine_active
          query: sum(DCGM_FI_DEV_MEM_COPY_UTIL) / count(DCGM_FI_DEV_MEM_COPY_UTIL) / 100
          threshold: '0.5' # Scale the GPU Nodegroup when the GPU memory usage is greater than 50%

```

- Cấu hình này điều chỉnh số lượng GPU trong Nodegroup dựa trên GPU usage và memory usage. Trong đó:
 - Field `query` chỉ định truy vấn để lấy số liệu từ Prometheus.
 - Field `threshold` chỉ định giá trị ngưỡng để điều chỉnh số lượng GPU trong Nodegroup.
 - Field `minReplicaCount` và `maxReplicaCount` chỉ định số lượng tối thiểu và tối đa mà Nodegroup GPU có thể điều chỉnh đến.
- 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%.
- Thực hiện apply tệp tin [scaling-app.yaml](#) để generate resources sử dụng verify cho tính năng autoscaling:

```

kubectl apply -f \
https://github.com/vngcloud/kubernetes-sample-apps/raw/main/nvidia-gpu/mani

```

- Tiếp tục thực hiện apply tệp tin [scale-gpu.yaml](#) để tạo `ScaleObject` cho deployment của bạn:

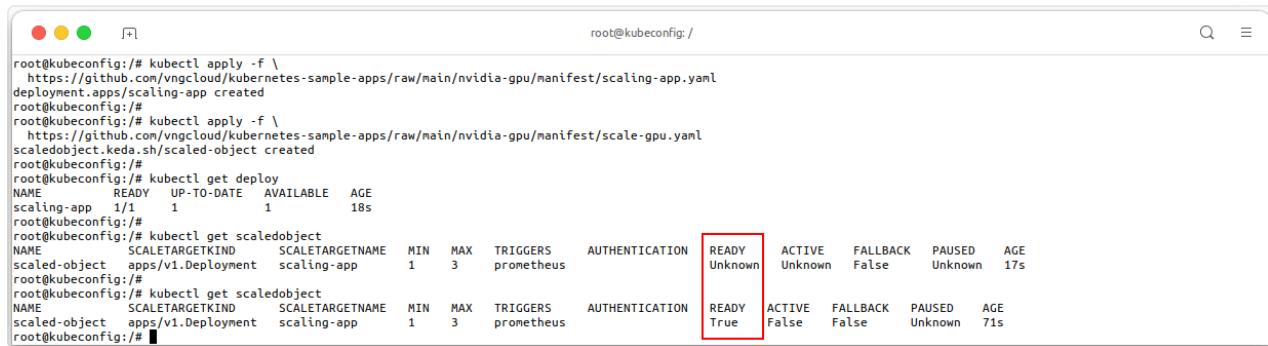
```

kubectl apply -f \
https://github.com/vngcloud/kubernetes-sample-apps/raw/main/nvidia-gpu/mani

kubectl get deploy

# Check the ScaledObject
kubectl get scaledobject

```



```

root@kubeconfig:~# kubectl apply -f \
https://github.com/vngcloud/kubernetes-sample-apps/raw/main/nvidia-gpu/manifest/scaling-app.yaml
deployment.apps/scaling-app created
root@kubeconfig:~#
root@kubeconfig:~# kubectl apply -f \
https://github.com/vngcloud/kubernetes-sample-apps/raw/main/nvidia-gpu/manifest/scale-gpu.yaml
scaledobject.keda.sh/scaled-object created
root@kubeconfig:~#
root@kubeconfig:~# kubectl get deploy
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
scaling-app   1/1     1            1           18s
root@kubeconfig:~#
root@kubeconfig:~# kubectl get scaledobject
NAME          SCALETARGETKIND   SCALETARGETNAME   MIN   MAX   TRIGGERS   AUTHENTICATION   READY
scaled-object   apps/v1.Deployment   scaling-app   1     3     prometheus   Unknown        Unknown
root@kubeconfig:~#
root@kubeconfig:~# kubectl get scaledobject
NAME          SCALETARGETKIND   SCALETARGETNAME   MIN   MAX   TRIGGERS   AUTHENTICATION   READY
scaled-object   apps/v1.Deployment   scaling-app   1     3     prometheus   Unknown        Unknown
root@kubeconfig:~#

```

- Khi trạng thái **ScaledObject Ready** là **True**, GPU Nodegroup được scale dựa trên GPU usage.

Clusters

Cluster trong Kubernetes là một tập hợp gồm một hoặc nhiều máy ảo (VM) được kết nối lại với nhau để chạy các ứng dụng được đóng gói dạng container. Cluster cung cấp một môi trường thống nhất để triển khai, quản lý và vận hành các container trên quy mô lớn.

Điều kiện cần:

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn tại [đây](#).
- Có ít nhất 1 **SSH key** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).
- Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.

Khởi tạo Cluster

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**.

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn, bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Cluster và Node Group của bạn tại Cluster Configuration, Default Node Group Configuration, Plugin.

- Cluster Configuration:
 - Cluster Information:

- **Cluster Name:** Tên cho Cluster của bạn. Tên chỉ có thể chứa các ký tự chữ và số (a-z, A-Z, 0-9, '_', '-'). Độ dài dữ liệu đầu vào của bạn phải từ 5 đến 50. Tên phải là duy nhất trong Khu vực và tài khoản VNG Cloud mà bạn đang tạo Cluster.
- **Kubernetes Version:** Phiên bản Kubernetes sẽ sử dụng cho Cluster của bạn. Chúng tôi khuyên bạn nên chọn phiên bản mới nhất, trừ khi bạn cần phiên bản cũ hơn.
- **Description:** Nhập vào thông tin bạn muốn ghi chú cho Cluster nhằm tạo dấu hiệu riêng cho việc quản lý chúng dễ dàng hơn trong tương lai.
- Network Setting:
 - **Network type:** lựa chọn loại network mà bạn mong muốn sử dụng cho Cluster của bạn. Hiện tại, VKS cung cấp cho bạn chọn sử dụng 1 trong 3 loại network Calico Overlay, Cilium Overlay, Cilium VPC Native Routing.
 - Đối với loại network Calico Overlay, Cilium Overlay, **Encapsulation Mode** được tự chọn mặc định bởi hệ thống và bạn không thể thay đổi chúng, tuy nhiên bạn có thể nhập lại thông số **Calico CIDR**: Dải mạng ảo mà các pod sẽ sử dụng (lưu ý IP phải là riêng tư và có thể chọn theo các tùy chọn sau (10.0.0.0 - 10.255.0.0 / 172.16.0.0 - 172.24.0.0 / 192.168.0.0)
 - Đối với loại network Cilium VPC Native Routing:
 - **Default Pod IP range** là dải địa chỉ IP thứ cấp được sử dụng cho các pod. Nó được gọi là **Secondary IP range** vì nó không trùng với dải IP chính của node (Primary IP range). Các pod trong Cluster sẽ được gán IP từ dải này. Bạn cần lựa chọn ít nhất 1 dải Secondary IP range đã tạo từ vServer.
 - **Node CIDR mask size:** Kích thước của CIDR dành cho các node. Thông số này cho biết mỗi node sẽ được gán bao nhiêu địa chỉ IP từ dải pod IP range. Kích thước này cần được chọn sao cho đảm bảo có đủ địa chỉ IP cho tất cả các pod trên mỗi node. Bạn có thể tham khảo bảng bên dưới để hiểu các tính số lượng IP có thể sử dụng để cấp phát cho node, pod trong cluster của bạn. Bạn cần lựa chọn một giá trị trong danh sách mà chúng tôi cung cấp phù hợp với nhu cầu của bạn.
 - **VPC:** Chọn một VPC hiện có đáp ứng các yêu cầu của K8S để tạo Cluster của bạn. Trước khi chọn một VPC, chúng tôi khuyên bạn nên làm quen với tất cả các yêu cầu và cân nhắc trong VPC cũng như các yêu cầu và cân nhắc về Subnet. Bạn không thể thay đổi VPC nào bạn muốn sử dụng sau khi tạo Cluster. Nếu không có VPC nào được liệt kê, trước tiên bạn cần tạo một VPC. Để biết thêm thông tin, hãy xem [Tạo VPC](#).
 - **Subnet:** Theo mặc định, tất cả các mạng con khả dụng trong VPC được chỉ định trong trường trước đó sẽ được chọn ngẫu nhiên ở thứ tự đầu tiên, bạn có thể chọn lại Subnet khác, tuy nhiên chỉ được chọn duy nhất 1.
- Default Node group Configuration:

- Node Group Information:
 - **Node Group Name:** Tên gợi nhớ cho Node Group của bạn.
 - **Number of nodes:** Nhập vào số lượng Worker node cho Cluster của bạn, lưu ý số lượng node cần lớn hơn hoặc bằng 1 và nhỏ hơn hoặc bằng 100.
- Node Group Automation Setting:
 - **Auto Healing:** Mặc định chúng tôi sẽ bật tính năng HA trong Cluster của bạn. Khi có node hoặc pod bị lỗi, Kubernetes sẽ tự động khởi động lại hoặc tạo mới pod để thay thế, đảm bảo ứng dụng của bạn luôn hoạt động mà không bị gián đoạn.
 - **Auto Scaling:** Bật tính năng tự động mở rộng trong Cluster của bạn. Auto scaling giúp tự động điều chỉnh số lượng pod (đơn vị triển khai ứng dụng) dựa trên nhu cầu sử dụng thực tế, tránh tình trạng lãng phí tài nguyên khi nhu cầu thấp hoặc quá tải khi nhu cầu cao.
 - **Minimum node:** số node tối thiểu mà Cluster cần có.
 - **Maximum node:** số node tối đa mà Cluster có thể scale tới.
 - Node Group upgrade strategy: chiến lược upgrade Node Group. Khi bạn thiết lập **Node Group Upgrade Strategy** thông qua phương thức **Surge upgrade** cho một Node Group trong VKS, hệ thống VKS sẽ cập nhật tuần tự để nâng cấp các node, theo thứ tự không xác định.
 - **Max surge:** giới hạn số lượng node được nâng cấp đồng thời (số lượng node mới (surge) có thể được tạo ra cùng một lúc). Mặc định **Max surge = 1** - chỉ nâng cấp một node tại một thời điểm. với maxUnavailable
 - **Max unavailable:** giới hạn số lượng node không thể truy cập được trong quá trình nâng cấp (số lượng node hiện tại có thể bị gián đoạn cùng một lúc). Mặc định **Max unavailable = 0** - đảm bảo tất cả các node đều có thể truy cập được trong quá trình nâng cấp.
- Node Group Setting:
 - **Image:** mặc định chúng tôi cung cấp 1 loại Image là Ubuntu with containerd.
 - **Instance type:** chọn loại phiên bản cấu hình phù hợp cho Worker node theo nhu cầu sử dụng của bạn.
- Node Group Volume Setting: **Cấu hình Boot Volume** – Các thông số được cài đặt mặc định bởi hệ thống giúp tối ưu cho Cluster của bạn
- Node Group Network Setting: Bạn có thể lựa chọn **Public Node Group** hoặc **Private Node Group** tùy theo nhu cầu sử dụng Cluster của bạn.
- Node Group Security Setting: Bạn có thể chọn **Security Group** và **SSH Key** cho Node Group của bạn.

- Node Group Metadata Setting: Bạn có thể nhập **Metadata** tương ứng cho Node Group.
- Plugin
 - **Enable BlockStore Persistent Disk CSI Driver:** bật để chúng tôi tự động cài đặt CSI Controller trên Cluster của bạn.
 - **Enable vLB Native Integration Driver:** bật để chúng tôi tự động cài đặt LB Controller trên Cluster của bạn.

Bước 5: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Tải xuống tệp tin Kube Config

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn menu **Kubernetes Cluster**.

Bước 3: Tại **Cluster** đã tạo thành công, chọn biểu tượng **và** chọn **Download Config File**.

Bước 4: File config sẽ được lưu về máy, lúc này bạn có thể sử dụng `Kubectl` để quản lý Cluster của bạn trên thiết bị cá nhân của mình.

Xóa một Cluster

 **Chú ý:**

Khi không còn nhu cầu sử dụng Kubernetes Cluster, bạn nên xóa các tài nguyên được liên kết với cụm đó để không phải chịu bất kỳ chi phí không cần thiết nào.

Khi xoá Kubernetes Cluster các tài nguyên sau sẽ bị xóa:

- Control Plane Resource của Cluster.
- Tất cả các node có trong Cluster (VM)
- Tất cả các Pod nào đang chạy trên các node.

- Security Group mặc định tạo cho Cluster đó.
- Load Balancer mặc định tạo cho Cluster đó.
- ETCD.

Hệ thống có thể không xóa các tài nguyên sau:

- Load Balancer được integrated vào Cluster bởi bạn.
- Persistent Volume được integrated vào Cluster bởi bạn.

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

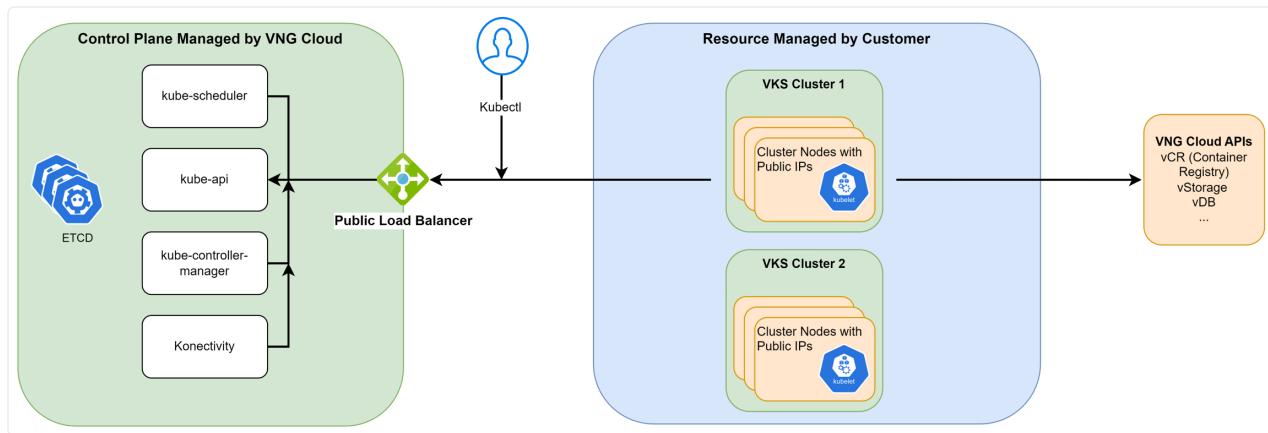
Bước 2: Tại màn hình **Overview**, chọn menu **Kubernetes Cluster**.

Bước 3: Tại **Cluster** đã tạo thành công, chọn cluster muốn xóa và chọn **Delete**

Bước 4: Chọn **Delete** để xóa hoàn toàn Cluster của bạn.

Public Cluster và Private Cluster

1. Public Cluster



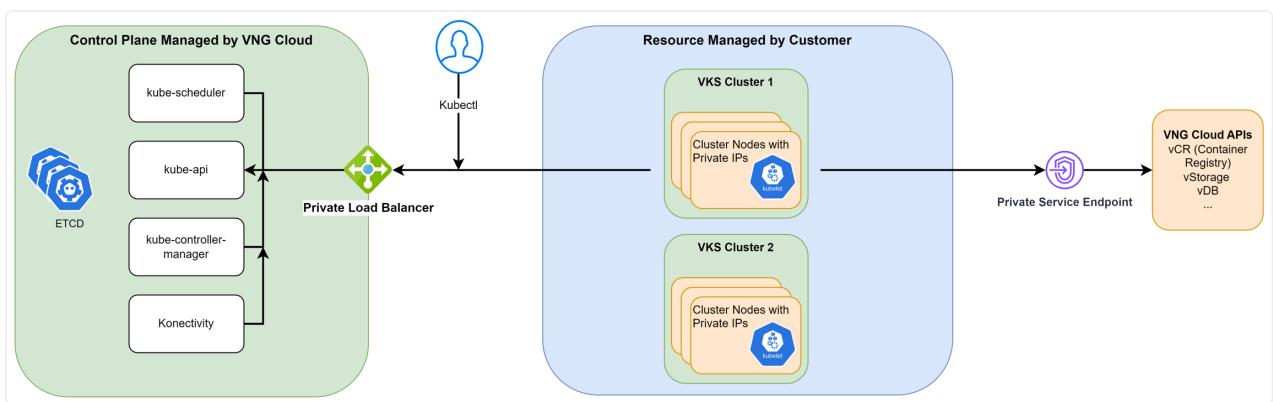
Khi bạn khởi tạo một **Public Cluster** với **Public Node Group**, hệ thống VKS sẽ:

- Tạo VM có Floating IP (tức có IP Public). Lúc này các VM (Node) này có thể join trực tiếp vào cụm K8S thông qua Public IP này. Bằng cách sử dụng Public Cluster và Public Node Group, bạn có thể dễ dàng tạo các cụm Kubernetes và thực hiện expose service mà không cần sử dụng Load Balancer. Việc này sẽ góp phần tiết kiệm chi phí cho cụm của bạn.

Khi bạn khởi tạo một **Public Cluster** với **Private Node Group**, hệ thống VKS sẽ:

- Tạo VM không có Floating IP (tức không có IP Public). Lúc này các VM (Node) này không thể join trực tiếp vào cụm K8S. Để các VM này có thể join vào cụm K8S, bạn cần phải sử dụng một NAT Gateway (**NATGW**). **NATGW** hoạt động như một trạm chuyển tiếp, cho phép các VM kết nối với cụm K8S mà không cần IP Public. Với VNG Cloud, chúng tôi khuyến cáo bạn sử dụng Pfsense hoặc Palo Alto như một NATGW cho Cluster của bạn. Pfsense sẽ giúp bạn quản lý lưu lượng mạng đến và đi (inbound và outbound traffic) một cách hiệu quả, đảm bảo an ninh mạng và quản lý truy cập. Bên cạnh đó, việc sử dụng Private Node Group sẽ giúp bạn kiểm soát các ứng dụng trong cụm được bảo mật hơn, cụ thể bạn có thể thực hiện giới hạn quyền truy cập control plane thông qua tính năng Whitelist IP.

2. Private Cluster



Khi bạn khởi tạo một **Public Cluster** với **Public/ Private Node Group**, hệ thống VKS sẽ:

- Để nâng cao bảo mật cho cluster của bạn, chúng tôi đã cho ra mắt mô hình private cluster. Tính năng Private Cluster giúp cho cụm K8S của bạn được bảo mật nhất có thể, mọi kết nối hoàn toàn là private từ kết nối giữa nodes tới control plane, kết nối từ client tới control plane, hay kết nối từ nodes tới các sản phẩm dịch vụ khác trong VNG Cloud như: vStorage, vCR, vMonitor, VNGCloud APIs,...Private Cluster là lựa chọn lý tưởng cho **các dịch vụ yêu cầu kiểm soát truy cập chặt chẽ, đảm bảo tuân thủ các quy định về bảo mật và quyền riêng tư dữ liệu**.

Upgrading Control Plane Version

Hiện tại, hệ thống VKS của chúng tôi đã hỗ trợ bạn nâng cấp Control Plane Version, bạn có thể:

- Nâng cấp **Minor Version** mới hơn (ví dụ: 1.24 lên 1.25)
- Nâng cấp **Patch Version** mới hơn (ví dụ: 1.24.2-VKS.100 lên 1.24.5-VKS.200)

Để thực hiện nâng cấp phiên bản Control Plane, bạn có thể thực hiện theo hướng dẫn sau:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn menu **Kubernetes Cluster**.

Bước 3: Chọn biểu tượng và chọn **Upgrade control plane version** để thực hiện nâng cấp version control plane.

Bước 4: Bạn có thể lựa chọn phiên bản mới cho control plane. Phiên bản mới cần hợp lệ và tương thích với phiên bản hiện tại của cluster. Cụ thể: bạn có thể chọn:

- Nâng cấp **Minor Version** mới hơn (ví dụ: 1.24 lên 1.25)
- Nâng cấp **Patch Version** mới hơn (ví dụ: 1.24.2-VKS.100 lên 1.24.5-VKS.200)

Bước 4: Hệ thống VKS sẽ thực hiện nâng cấp các thành phần **Control Plane** của **Cluster** lên phiên bản mới. Sau khi việc nâng cấp hoàn tất, trạng thái Cluster trở về **ACTIVE**.

Chú ý:

- Việc nâng cấp Control Plane Version là không bắt buộc và độc lập với việc nâng cấp Node Group Version. Tuy nhiên Control Plane Version và Node Group Version trong cùng một Cluster không được lệch quá 1 minor version. Bên cạnh đó, hệ thống VKS tự động nâng cấp Control Plane Version khi phiên bản K8S Version hiện tại đang sử dụng cho Cluster của bạn quá thời hạn được nhà cung cấp hỗ trợ.
- Trong quá trình nâng cấp Control Plane Version, bạn không thể thực hiện các hành động khác trên Cluster của bạn.
- Bên dưới là một vài lưu ý trước, trong và sau quá trình nâng cấp, vui lòng tham khảo thêm:

Trước khi thực hiện:

- Sao lưu dữ liệu: Nên sao lưu dữ liệu của cluster trước khi nâng cấp để đảm bảo an toàn trong trường hợp nâng cấp thất bại.
- Kiểm tra phiên bản hiện tại: Truy cập Releases để tham khảo danh sách các phiên bản được hỗ trợ. Chọn phiên bản mới hợp lệ và tương thích với phiên bản hiện tại của cluster.
- Đảm bảo tính sẵn sàng của cluster: Cluster phải đang ở trạng thái hoạt động (ACTIVE) và tất cả các node phải HEALTHY.
- Ngừng các tác vụ đang chạy: Ngừng các tác vụ đang chạy trên cluster để tránh ảnh hưởng đến quá trình nâng cấp.

Trong khi thực hiện:

- Theo dõi trạng thái cluster: Theo dõi trạng thái cluster trong quá trình nâng cấp. Trạng thái cluster sẽ chuyển sang UPDATING và sau khi hoàn tất sẽ trở về ACTIVE.
- Kiểm tra nhật ký hệ thống: Kiểm tra nhật ký hệ thống để phát hiện bất kỳ lỗi hoặc cảnh báo nào trong quá trình nâng cấp.

Sau khi thực hiện:

- Kiểm tra tính sẵn sàng của cluster: Xác nhận rằng cluster đã được nâng cấp thành công và tất cả các node đang hoạt động bình thường.
- Kiểm tra các ứng dụng: Kiểm tra các ứng dụng đang chạy trên cluster để đảm bảo chúng hoạt động bình thường sau khi nâng cấp.

Lưu ý:

- Việc nâng cấp Control Plane Version có thể mất một khoảng thời gian tùy thuộc vào kích thước và độ phức tạp của cluster.
- Trong một số trường hợp hiếm gặp, việc nâng cấp Control Plane Version có thể thất bại. Nếu điều này xảy ra, hệ thống VKS sẽ tự động rollback cluster về phiên bản hiện tại.

Whitelist

Tổng quan

Tính năng Whitelist IP trên chế độ Private Node Group của VKS cho phép bạn chỉ cho phép các địa chỉ IP cụ thể kết nối đến Cluster của bạn. Điều này giúp tăng cường bảo mật cho các ứng dụng và dữ liệu nhạy cảm bằng cách hạn chế truy cập từ các nguồn không xác định.

Lợi ích

- Tăng cường bảo mật:** Whitelist IP giúp bảo vệ dữ liệu và ứng dụng của bạn khỏi các mối đe dọa tiềm ẩn trên mạng công cộng, chẳng hạn như tấn công mạng và vi phạm dữ liệu.
- Giảm thiểu rủi ro:** Bằng cách hạn chế truy cập vào các node nhạy cảm, Whitelist IP giúp giảm thiểu rủi ro lây lan vi phạm dữ liệu sang các phần khác của mạng của bạn.
- Kiểm soát tốt hơn:** Whitelist IP cho phép bạn kiểm soát chặt chẽ quyền truy cập vào các node của mình, đảm bảo chỉ những người dùng và ứng dụng được ủy quyền mới có thể truy cập.

i **Khuyến nghị về Sử dụng Whitelist trong Các Mô Hình Cluster:**

1. Public Cluster Chỉ Bao Gồm Public Node Group

- Khuyến nghị:** Không khuyến khích sử dụng whitelist.
- Nếu bạn có nhu cầu sử dụng Whitelist IP vì security, vui lòng allow danh sách IP Range Public của vServer theo danh sách sau:

```
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 Bao Gồm Private Node Group Đi Qua NAT Gateway (Pfsense, PaloAlto)

- **Khuyến nghị:** Có thể sử dụng tính năng whitelist.
- Cần thực hiện Whitelist thêm IP của NAT Gateway.

3. Private Cluster Bao Gồm Public Node Group hoặc Private Node Group

- **Khuyến nghị:** Có thể sử dụng tính năng whitelist.

Chỉnh sửa Whitelist

Để sử dụng tính năng Whitelist IP trên chế độ Private Node Group, bạn cần thực hiện các bước sau:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn menu **Kubernetes Cluster**.

Bước 3: Chọn biểu tượng **Action** và chọn **Edit Whitelist** hoặc chọn biểu tượng **Edit** khi xem chi tiết một Cluster để thực hiện thêm Whitelist cho Cluster của bạn.

Bước 4: Lúc này, màn hình **Edit Whitelist** hiển thị, bạn có thể nhập địa chỉ IP mà bạn muốn cho phép truy cập vào Cluster sau đó chọn **Add**.

Bước 5: Lặp lại bước 4 nếu bạn muốn thêm nhiều **Whitelist IP** cho Cluster của bạn. Bạn cũng có thể chọn **Delete** để xóa Whitelist IP mà bạn đã thêm trước đó.

Bước 6: Chọn **Save** để lưu thông tin hoặc **Cancel** để hủy bỏ việc lưu các thông số này.

POC và Stop POC

Trước khi tìm hiểu cách Stop POC cho tài nguyên của bạn trên VKS, bạn nên hiểu rõ các khái niệm cũng các hành động mà bạn có thể thao tác đối với tài nguyên POC. Chi tiết tham khảo thêm [tại đây](#).

Khởi tạo tài nguyên VKS thông qua ví POC

Để khởi tạo Cluster thông qua ví POC, hãy thực hiện các bước như sau;

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn, bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Cluster và Node Group của bạn tại Cluster Configuration, Default Node Group Configuration, Plugin. **Mặc định chúng tôi sẽ khởi tạo cho bạn một Public Cluster với Public Node Group.**

Bước 5: Chọn POC như hình bên dưới

Bước 6: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 7: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Chú ý:

- Khi bạn khởi tạo Cluster và chọn sử dụng ví POC, chúng tôi đã tự động tạo Control Plane, Node, Volume và Private Service Endpoint (nếu bạn chọn sử dụng) thông qua ví POC. Đối với các tài nguyên khác như
 - PVC:** khi thực hiện khởi tạo qua yaml, bạn vui lòng thêm tham số `isPOC: "true"` vào file yaml này. Tham khảo ví dụ bên dưới.
 - LoadBalancer:** khi thực hiện khởi tạo qua yaml, bạn vui lòng thêm annotation `vks.vngcloud.vn/is-poc: "true"` vào file yaml này. Tham khảo ví dụ bên dưới.
- Do các resource **Load Balancer** và **PVC** được quản lý thông qua YAML, sau khi Stop POC, nếu trong file YAML của bạn vẫn có tham số `isPOC : true` hoặc `is-poc : true`, trong trường hợp bạn xóa Load Balancer từ Portal vLB và xóa tham số `load-balancer-id` trong yaml, lúc này hệ thống sẽ tự động tạo lại các resource này thông qua ví POC. Để tạo Load Balancer và PVC khác bằng tiền thật, vui lòng thay đổi tham số `isPOC` thành `false`. (`isPOC : false` hoặc `is-poc : false`). Chúng tôi khuyến cáo bạn nên thực hiện điều chỉnh tham số này trước khi thực hiện Stop POC cho Cluster của bạn.

Bên dưới là yaml mẫu để tạo **Load Balancer** thông qua số dư ví POC:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      unique-label: app1
  template:
    metadata:
      labels:
        unique-label: app1
        same-label: vngcloud
    spec:
      containers:
        - name: nginx-deployment
          image: nginx
          ports:
            - containerPort: 80
              name: http
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-http-server
spec:
  replicas: 2
  selector:
    matchLabels:
      app: python-http-server
  template:
    metadata:
      labels:
        app: python-http-server
        same-label: vngcloud
    spec:
      containers:
        - name: python-http-server
          image: python:3.9-slim
          command: ["python", "-m", "http.server", "8080"]
          ports:
            - containerPort: 8080
              name: http
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: default
  annotations:

```

```

vks.vngcloud.vn/is-poc: "true"
spec:
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http-server
  selector:
    same-label: vngcloud
  type: LoadBalancer

---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: webapp-02
  namespace: default
  annotations:
    vks.vngcloud.vn/is-poc: "true"
spec:
  ingressClassName: vngcloud
  defaultBackend:
    service:
      name: nginx-service
    port:
      name: http-server

```

Và đây là yaml mẫu để tạo **PVC** thông qua số dư ví POC:

```

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 CSI driver
parameters:
  type: vtype-xxxxxxxxxxxxxxxxxxxxxxxxxxxxx   # The volume type UUID
  isPOC: "true"
allowVolumeExpansion: true                   # MUST set this value to true
---

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

```

```

apiVersion: v1
kind: Pod
metadata:
  name: nginx                                # [5] The Pod name, CAN be changed
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 [6]
  volumes:
    - name: my-volume-name                    # [6] The volume name, CAN be changed
  persistentVolumeClaim:
    claimName: my-expansion-pvc             # MUST be the same as [2]
    readOnly: false

```

Stop POC

Trước khi hết hạn sử dụng ví POC cho Cluster, bạn có hai lựa chọn chính:

- Xóa Cluster đang POC này và tạo lại Cluster bình thường khác.
- Thực hiện Stop POC để gia hạn Cluster đang POC thành Cluster bình thường.

(i) Chủ ý:

Để thực hiện Stop POC cho một Cluster và các thành phần liên quan của Cluster, bạn cần thực hiện:

- **Bước 1:** Thực hiện **cập nhật tham số** `isPOC : true` hoặc `is-poc : true` về `isPOC : false` hoặc `is-poc : false` trong các yaml cho Load Balancer, PVC nếu có.
- **Bước 2:** Thực hiện **Stop POC** cho Cluster thông qua nút **Stop POC** trên **VKS Portal** theo hướng dẫn bên dưới.
- **Bước 3:** Thực hiện **Thanh toán cho các tài nguyên thông qua tiền thật**.

Để tiếp tục sử dụng tài nguyên vừa dừng POC như một tài nguyên bình thường (với mục đích giữ nguyên cấu hình), người dùng có thể thực hiện:

Bước 1: Truy cập vào [VKS Portal](#), chọn Cluster mà bạn muốn Stop POC.

Bước 2: Chọn nút **Stop POC** phía trên góc phải màn hình.

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' detail page. At the top, it shows the cluster status as 'ACTIVE' and was created on '03/07/2024 11:23:18'. Below this, there are four summary boxes: 'Total Node Groups' (1), 'Total Nodes' (2), 'Healthy/Active Nodes' (1 green box), and 'Unhealthy Nodes' (1 yellow box). To the right of these boxes is a red arrow pointing to a blue 'Stop POC' button. Further down, there are sections for 'General Information' and 'Detail Information', each with several configuration details. At the bottom, there are links for 'Node group', 'Volume', 'Load balancer', and 'Event history', along with a 'Add Node Group' button.

Bước 3: Lúc này, màn hình hiển thị danh sách tất cả các Server và Volume (bao gồm cả Boot Volume và PVC mà bạn attach vào node trong Cluster của bạn), Load Balancer, Endpoint thuộc Cluster đang có trạng thái POC. Bạn có thể kiểm tra thông tin sau đó chọn Stop POC

The screenshot shows the UNIS CLOUD interface for managing a Kubernetes cluster. The main page displays general information about the cluster, including its ID, version, and node groups. Below this, there are tabs for Node group, Volume, Load balancer, and Endpoint. The Endpoint tab is currently active, showing a list of endpoints associated with the cluster. A modal dialog titled 'Stop Resource POC' is open over the main content. This dialog contains a message stating that stopping POC will stop POC of its resources. It lists several resources under the 'Servers' tab, with a red box highlighting the 'Servers' tab itself. At the bottom of the dialog, there is a blue 'Stop POC' button, which is also highlighted with a red arrow.

This screenshot shows another view of the UNIS CLOUD interface for a Kubernetes cluster named 'demo-cluster'. The layout is similar to the previous one, with a main cluster overview and an active 'Endpoint' tab. A modal dialog titled 'Stop Resource POC' is displayed, listing resources under the 'Endpoint' tab. A red box highlights the 'Endpoint' tab, and a red arrow points to the 'Stop POC' button at the bottom of the dialog.

Stop Resource POC

POC is stopped for this cluster, however, we found some resources that are still in POC. Check these resources and stop POC for them.

Resources associated with this Cluster that has POC stopped previously from vServer or from another cluster will not be listed below.

After checking, click Stop POC to be directed to the Payment page to proceed.

Servers Volumes Snapshot Load balancer Endpoint

LB name	ID
vks-k8s-b075dd-default-nginx-svcs-bf78f	lb-dc361979-e5e5-4d4e-ab92-612c36ecb3fb
vks-k8s-b075dd-default-webapp-02-a91a3	lb-33725755-55ea-421b-bb61-5fd31ca4891

(x) Cancel **Stop POC**

General Information
Main information of your cluster

ID
Version
Description
Created at
Whitelist
BlockStore Persistent Disk
vLB Native Integration

Detail Information
Details of your cluster while it is in use

Node group **Volume** **Load balancer** **Endpoint** **Event history** **Monitor**

Add Node Group

© 2024, VI Data IT JSC Golden King Building, No. 15 Nguyen Luong Bang Street Tax code: 0304851362
support@vngcloud.vn - 190007549 Tan Phu, District 7, HCMC, Viet Nam Dept. of Planning and Investment of HCMC on 20/02/2024

Terms Privacy Policy ISO ISO ISO SLA PCI DSS

Bước 4: Tiến hành thanh toán tài nguyên bằng tiền thật, bạn có thể lựa chọn **Chu kỳ sử dụng mong muốn, bật tắt Tự động gia hạn, nhập Coupon** nếu có và chọn **Continue** để thực hiện Thanh toán tài nguyên

Check out > Order Summary

ITEMS TO EXCHANGE POC
Confirm information of the following item then click Continue to view your Order summary

Resource	Unit Price	Period	Price	
vserver - instance ins-7f347e01-524c-4328-9505-bf7866f72817	555,500 VND / month	<input checked="" type="checkbox"/> Auto-renew	1 month	555,500 VND
vserver - instance ins-8dbe56e5-cb65-4785-9114-9bae712c94a1	555,500 VND / month	<input checked="" type="checkbox"/> Auto-renew	1 month	555,500 VND
vserver - instance ins-91f92d3e-d2e7-400a-917a-4c64c12414ea	555,500 VND / month	<input checked="" type="checkbox"/> Auto-renew	1 month	555,500 VND
vserver - endpoint enp-4113a42c-9f22-4c19-9982-30f88073f9d4	391,820 VND / month	<input checked="" type="checkbox"/> Auto-renew	1 month	391,820 VND
vserver - endpoint enp-0e75b2c-0eb5-45fd-8af0-ac25ea49cf72	391,820 VND / month	<input checked="" type="checkbox"/> Auto-renew	1 month	391,820 VND
vserver - endpoint enp-7e739011-2505-4f22-b28a-d46b4ea506f6	391,820 VND / month	<input checked="" type="checkbox"/> Auto-renew	1 month	391,820 VND
vserver - endpoint enp-59c0ee1e-5a64-4ef2-ae73-5370ff085f04	391,820 VND / month	<input checked="" type="checkbox"/> Auto-renew	1 month	391,820 VND

Coupon Enter the code **Apply**

Total (7 items) **3,233,780 VND** (Saved 3,233,780 VND) **CONTINUE**

© 2022, VI Data Information Technology JSC Helios Building, Quang Trung Software City Tax code: 0304851362
sales@vngcloud.vn - 190007549 Tan Chanh Hiep Ward, District 12, HCMC Department of Planning and Investment of HCMC

Terms & Conditions ISO ISO ISO SLA PCI DSS

Bước 5: Thực hiện thanh toán bằng số dư credit hoặc qua các hình thức thanh toán khác nếu có.

Order Summary

7 ITEMS

Resource	Period	Price
vserver - instance ins-7f347e01-524c-4328-9505-bf7866f72817	1 month Until 22/11/2024 17:26	555,500 VND
vserver - instance ins-8dbe56e5-cb65-4785-9114-9bae712c94a1	1 month Until 22/11/2024 17:26	555,500 VND
vserver - instance ins-9ff92d3e-d2d7-400a-917a-4c64c12414ea	1 month Until 22/11/2024 17:26	555,500 VND
vserver - endpoint emp-4113a42c-9f22-4c19-9982-30f88073f9d4	1 month Until 22/11/2024 17:26	391,820 VND
vserver - endpoint emp-de075b2c-0ab5-45fd-8af8-ac25ea49cf72	1 month Until 22/11/2024 17:26	391,820 VND
vserver - endpoint emp-7e739011-2505-4f22-b28a-d46b4ea506f6	1 month Until 22/11/2024 17:26	391,820 VND
vserver - endpoint emp-59c0ee1e-5ae4-4ef2-ae73-5370ff085f04	1 month Until 22/11/2024 17:26	391,820 VND

ORDER SUMMARY

Original price	6,467,560 VND
Item discount	3,233,780 VND
Total	3,233,780 VND

I agree with the Terms & Conditions

Use 3,233,780 credits

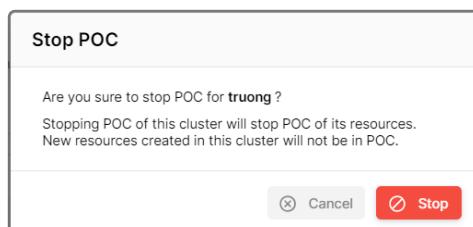
You'll pay for this order with 3,233,780 credits

PAY

© 2022, Vi Na Data Information Technology JSC
sales@vngcloud.vn - 19001549 Helios Building, Quang Trung Software City
Tax code: 0304851362
Tan Chanh Hiep Ward, District 12, HCMC
Department of Planning and Investment of HCMC

Terms & Conditions ISO 9001 ISO 27001 ISO 22381 SLA PCI DSS

Để đảm bảo VKS hoạt động chính xác, việc thực hiện Stop POC cần được tiến hành trên VKS Portal thay vì thực hiện riêng lẻ trên vServer Portal hoặc vConsole. Nếu bạn đã thực hiện stop POC riêng lẻ cho từng resource trên vServer Portal trước đó, bạn vẫn **cần thực hiện Stop POC** cho Cluster tại VKS Portal, lúc này, màn hình sẽ hiển thị như sau. Bạn hãy nhấn **Stop** để tắt lựa chọn POC cho Cluster của bạn.



(i) Chú ý:

- Sau khi stop POC trên VKS, nút "**Stop POC**" sẽ tiếp tục hiển thị **nếu chúng tôi thấy vẫn còn resource chưa được Stop POC** sau khi thực hiện trên VKS Portal. Bạn có thể tiếp tục chọn và thực hiện **Stop POC** cho đến khi tất cả các resource được chuyển về resource thật.
- Đối với loại resource Snapshot**, bạn không thể chỉ định snapshot sử dụng ví POC từ VKS. Để thực hiện tạo Snapshot qua ví POC, tại **vServer Portal**, vui lòng chọn **Activate Snapshot**, sau đó tại màn hình **Checkout**, vui lòng chọn sử dụng ví **POC**. Lúc này **tất cả các resource snapshot của bạn sẽ được tạo qua ví POC**. Do đó, việc stop POC cần được bạn thực hiện thông qua **vConsole** hoặc **vServer Portal**. Tham khảo thêm hình bên dưới.

VNG CLOUD

Search VNG Cloud services

vServer / BlockStore / Snapshot

Snapshot DISABLE

Snapshots are a low-cost, convenient, and efficient method to back up data and can be used to create images, implement disaster recovery plans, and distribute data copies.

Billing Note

You are not charged for activating the snapshot service.

After you create snapshots, the snapshots are billed based on the amount of storage space they consume and the amount of time for which they are stored. Snapshots are billed per region every hour on the hour (7.7 VND/ 1GB/ 1 hour). Pay attention to fees incurred in the future.

We recommend that you delete snapshots that are no longer needed on a regular basis. [Click here](#) for more information about snapshot pricing.

You have no Snapshot

VNG Cloud snapshot service allows you to create snapshots for all disk categories. Snapshots are a low-cost, convenient, and efficient method to back up data and can be used to create images, implement disaster recovery plans, and distribute data copies.

Activate Snapshot Service

[What can I do with Snapshots? →](#)

© 2024, Vi Na Data IT JSC Golden King Building, No. 15 Nguyen Luong Bang Street Tax code: 0304851362
Dept. of Planning and Investment of HCMC on 26/02/2007

Terms Privacy Policy ISO ISO ISO ISO SLA PCI DSS

VNG CLOUD

← Cancel and go back

Check out > Order summary

RESOURCES TO BUY NEW

Confirm information of the following item then click **Continue** to view your Order summary.

Resource	Price
vstorage - snapshot snapshot-HCM-03	0 VND

Coupon **Apply**

Holding (1 items) 0 VND (1) **Pay with POC wallet** Available: 956,000,600 credits

CONTINUE

© 2022, Vi Na Data Information Technology JSC Helios Building, Quang Trung Software City Tax code: 0304851362
sales@vngcloud.vn | 1800 1549 Tan Chanh Hiep Ward, District 12, HCMC Department of Planning and Investment of HCMC

Terms & Conditions ISO ISO ISO ISO SLA PCI DSS

Node Groups

Node Group là một khái niệm quan trọng trong Kubernetes, dùng để quản lý nhóm các **node** (VM) có cùng chung cấu hình trong một cluster. Đối với một Node Group, bạn có thể:

Tạo một Node Group

Để khởi tạo một Node Group, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại Cluster đã khởi tạo trước đó, hãy chọn **Create a Node group.**

Bước 3: Tại màn hình khởi tạo Node Group, chúng tôi đã thiết lập thông tin cho Node Group của bạn. Bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Node Group của bạn tại:

- Node Group Information:
 - **Node Group Name:** Tên gợi nhớ cho Node Group của bạn.
 - **Number of nodes:** Nhập vào số lượng Worker node cho Cluster của bạn, lưu ý số lượng node cần lớn hơn hoặc bằng 1 và nhỏ hơn hoặc bằng 100.
- Node Group Automation Setting:
 - **Auto Healing:** Mặc định chúng tôi sẽ bật tính năng HA trong Cluster của bạn. Khi có node hoặc pod bị lỗi, Kubernetes sẽ tự động khởi động lại hoặc tạo mới pod để thay thế, đảm bảo ứng dụng của bạn luôn hoạt động mà không bị gián đoạn.
 - **Auto Scaling:** Bật tính năng tự động mở rộng trong Cluster của bạn. Auto scaling giúp tự động điều chỉnh số lượng pod (đơn vị triển khai ứng dụng) dựa trên nhu cầu sử dụng thực tế, tránh tình trạng lãng phí tài nguyên khi nhu cầu thấp hoặc quá tải khi nhu cầu cao.
 - **Minimum node:** số node tối thiểu mà Cluster cần có.
 - **Maximum node:** số node tối đa mà Cluster có thể scale tới.
 - **Node Group upgrade strategy:** chiến lược upgrade Node Group. Khi bạn thiết lập **Node Group Upgrade Strategy** thông qua phương thức **Surge upgrade** cho một Node Group trong VKS, hệ thống VKS sẽ cập nhật tuần tự để nâng cấp các node, theo thứ tự không xác định.
 - **Max surge:** giới hạn số lượng node được nâng cấp đồng thời (số lượng node mới (surge) có thể được tạo ra cùng một lúc). Mặc định **Max surge = 1** - chỉ nâng cấp một node tại một thời điểm. với maxUnavailable

- **Max unavailable:** giới hạn số lượng node không thể truy cập được trong quá trình nâng cấp (số lượng node hiện tại có thể bị gián đoạn cùng một lúc). Mặc định **Max unavailable = 0** - đảm bảo tất cả các node đều có thể truy cập được trong quá trình nâng cấp.
- Node Group Setting:
 - **Image:** mặc định chúng tôi cung cấp 1 loại Image là Ubuntu with containerd.
 - **Instance type:** chọn loại phiên bản cấu hình phù hợp cho Worker node theo nhu cầu sử dụng của bạn.
- Node Group Volume Setting: **Cấu hình Boot Volume** – Các thông số được cài đặt mặc định bởi hệ thống giúp tối ưu cho Cluster của bạn
- Node Group Network Setting: Bạn có thể lựa chọn **Public Node Group** hoặc **Private Node Group** tùy theo nhu cầu sử dụng Cluster của bạn.
- Node Group Security Setting: Bạn có thể chọn **Security Group và SSH Key** cho Node Group của bạn.
- Node Group Metadata Setting: Bạn có thể nhập **Metadata** tương ứng cho Node Group.

Bước 5: Chọn **Create Node Group**. Hãy chờ vài phút để chúng tôi khởi tạo Node Group của bạn, trạng thái của Node Group lúc này là **Creating**.

Bước 6: Khi trạng thái **Node Group** là **Active**, bạn có thể xem thông tin Node Group bằng cách chọn vào **Node Group Name** tại màn hình chính.

Chỉnh sửa một Node Group

Đối với **Node Group**, bạn có thể chỉnh sửa các thông số: **Number of Nodes**, **Auto Scaling**, **Upgrade Strategy**, **Security Group** trong từng lần chỉnh sửa riêng biệt. Cụ thể, bạn có thể thực hiện theo các bước sau đây:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại Cluster đã khởi tạo trước đó, hãy chọn **vào Cluster bạn muốn chỉnh sửa Node group**.

Bước 3: Tại màn hình chứa danh sách Node Group đang có, tại Node Group bạn muốn chỉnh sửa chọn một trong các phương án:

- Tính năng **Resize**: bạn có thể thay đổi

- Number of nodes: Nhập vào số lượng Worker node cho Cluster của bạn, lưu ý số lượng node cần lớn hơn hoặc bằng 1 và nhỏ hơn hoặc bằng 100.
- Tính năng **Edit Auto Scaling**: bạn có thể thay đổi
 - Auto Scaling: Bật tính năng tự động mở rộng trong Cluster của bạn. Auto scaling giúp tự động điều chỉnh số lượng pod (đơn vị triển khai ứng dụng) dựa trên nhu cầu sử dụng thực tế, tránh tình trạng lãng phí tài nguyên khi nhu cầu thấp hoặc quá tải khi nhu cầu cao.
 - Minimum node: số node tối thiểu mà Cluster cần có.
 - Maximum node: số node tối đa mà Cluster có thể scale tới.
- Tính năng **Edit Upgrade Stratety**: bạn có thể thay đổi
 - Node Group upgrade stratety: chiến lược upgrade Node Group. Khi bạn thiết lập Node Group Upgrade Strategy thông qua phương thức Surge upgrade cho một Node Group trong VKS, hệ thống VKS sẽ cập nhật tuần tự để nâng cấp các node, theo thứ tự không xác định.
 - Max surge: giới hạn số lượng node được nâng cấp đồng thời (số lượng node mới (surge) có thể được tạo ra cùng một lúc). Mặc định Max surge = 1 - chỉ nâng cấp một node tại một thời điểm. với maxUnavailable
 - Max unavailable: giới hạn số lượng node không thể truy cập được trong quá trình nâng cấp (số lượng node hiện tại có thể bị gián đoạn cùng một lúc). Mặc định Max unavailable = 0 - đảm bảo tất cả các node đều có thể truy cập được trong quá trình nâng cấp.
- Tính năng **Edit Security Group**: bạn có thể thay đổi
 - Node Group Security Setting: Bạn có thể chọn Security Group và SSH Key cho Node Group của bạn.

Xóa một Node Group

Chú ý:

Khi không còn nhu cầu sử dụng Node Group, bạn hãy thực hiện xóa chúng để tiết kiệm chi phí. Khi xoá Node Group, các tài nguyên sau sẽ bị xóa:

- Tất cả các node có trong Node Group (VM)

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn menu **Kubernetes Cluster**.

Bước 3: Tại **Cluster** đã tạo thành công, chọn Node Group bạn muốn xóa và chọn **Delete**.

Bước 4: Chọn **Delete** để xóa hoàn toàn **Node Group** của bạn.

Auto Healing

Tổng quan

Trên hệ thống VKS, tính năng Auto Healing được apply cho mỗi Node Group và luôn luôn ở trạng thái bật. Việc bật tính năng tự phục hồi (self-healing) trong Kubernetes mang lại nhiều lợi ích quan trọng, giúp đảm bảo tính sẵn sàng và độ tin cậy cao cho ứng dụng của bạn.

Tính năng Auto Healing có những điểm nổi bật sau:

- Tự động phát hiện lỗi:** Kubernetes có thể tự động phát hiện các node bị lỗi hoặc gặp sự cố thông qua việc theo dõi trạng thái của các node. Một số dấu hiệu cho thấy node bị lỗi bao gồm: node báo cáo trạng thái "NotReady", node không thể ping được, node bị lỗi phần cứng, v.v.
- Tự động khởi động lại node:** Khi một node bị phát hiện lỗi, Kubernetes sẽ tự động khởi động lại node. Việc khởi động lại node có thể giúp khắc phục các lỗi tạm thời và đưa node trở lại trạng thái hoạt động bình thường.
- Giảm thiểu sự can thiệp thủ công:** Auto Healing giúp giảm thiểu sự can thiệp thủ công của người quản trị viên hệ thống, tiết kiệm thời gian và công sức.
- Cải thiện hiệu quả hoạt động:** Auto Healing giúp cải thiện hiệu quả hoạt động của hệ thống bằng cách đảm bảo rằng các node luôn hoạt động bình thường.

Cơ chế hoạt động

Cơ chế Auto Healing: hệ thống VKS thực hiện kích hoạt auto healing khi

- Node báo cáo trạng thái **NotReady** trong các lần kiểm tra liên tiếp trong khoảng thời gian **10 phút**.

Nếu thỏa mãn điều kiện trên, hệ thống sẽ ngay lập tức thực hiện auto healing. Quá trình này được thực hiện theo 2 bước:

- Bước 1:** Hệ thống VKS thực hiện drain node, tức là di chuyển tất cả các pod đang chạy trên node NotReady này sang các node khác trong node group trước khi gỡ bỏ node đó khỏi node group.
- Bước 2:** Hệ thống sẽ tạo lại node mới với cấu hình đã được thiết lập trên node group và thực hiện join node này vào cụm. Nếu sau khi khởi động lại, node vẫn báo cáo trạng

thái "NotReady", hệ thống sẽ tiếp tục khởi động lại node cho đến khi node trở lại trạng thái hoạt động bình thường.

 **Chú ý:**

- Khi hệ thống thực hiện Auto Healing, việc tạo ra node mới có thể gặp lỗi nếu bạn không có đủ credit hoặc bạn đã hết quota để tạo VM trên hệ thống vServer. Lúc này, mỗi 30 phút thì hệ thống sẽ thực hiện khởi động lại node cho đến khi node trở lại trạng thái hoạt động bình thường. Để tránh gặp lỗi bên trên, bạn cần:
 - **Đảm bảo bạn có đủ credit:** Nếu bạn là người dùng trả trước, hãy nạp thêm credit vào tài khoản của bạn.
 - **Yêu cầu tăng quota:** Bạn có thể yêu cầu tăng quota cho tài khoản của mình tại [đây](#).

Bật Auto Healing

Hiện tại, tính năng Auto Healing được apply cho mỗi Node Group và luôn luôn ở trạng thái **bật**. Bạn không cần thao tác bật thủ công khi khởi tạo Cluster cũng như Node Group.

Auto Scaling

Tổng quan

Auto Scaling cho Cluster là một tính năng trong Kubernetes cho phép tự động điều chỉnh kích thước của cụm (Cluster) cụ thể là số lượng các node trong cụm để đáp ứng nhu cầu sử dụng.

Tính năng Auto Scaling có những điểm nổi bật sau:

- Tối ưu hóa hiệu suất:** Auto Scaling cho phép cụm tự động mở rộng tài nguyên khi có nhu cầu. Khi khối lượng công việc cao hơn, cụm sẽ tự động tạo thêm các node để đảm bảo các ứng dụng hoạt động với hiệu suất tốt nhất.
- Tiết kiệm chi phí:** Auto Scaling cho phép cụm tự động giảm tài nguyên khi không cần thiết. Nếu khối lượng công việc giảm đi, cụm sẽ tự động thu hồi các tài nguyên không sử dụng để tiết kiệm chi phí.
- Đảm bảo tính sẵn sàng:** Auto Scaling giúp đảm bảo rằng cụm có sẵn để đáp ứng nhu cầu sử dụng và tránh tình trạng quá tải hoặc thiếu tài nguyên.
- Tự động phục hồi:** Auto Scaling giúp tự động phục hồi từ các sự cố hoặc lỗi bằng cách tạo ra các node mới để thay thế các node bị hỏng.

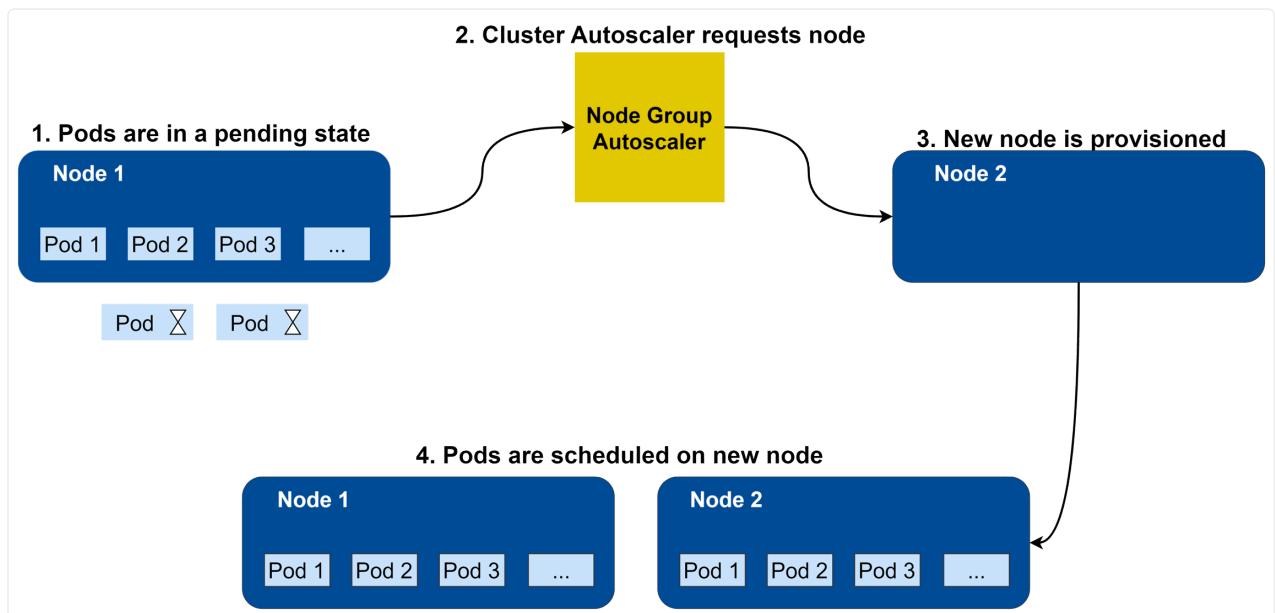
Khi triển khai các ứng dụng trong môi trường cloud, việc sử dụng tính năng Auto Scaling giúp tối ưu hóa việc sử dụng tài nguyên, cải thiện tính sẵn sàng và hiệu suất của ứng dụng, và giúp quản lý cụm trở nên dễ dàng và hiệu quả hơn.

Cơ chế hoạt động

Cơ chế Scale up: hệ thống VKS thực hiện scale up khi

- Các pods không thể scheduling trên bất kỳ node hiện tại nào vì lý do thiếu resource.
- Việc tăng thêm 1 node giống với cấu hình node group hiện tại là hữu ích và có thể xử lý vấn đề thiếu resource này.

Minh họa:



Nếu thỏa mãn 2 điều kiện trên, hệ thống sẽ tăng số node (một hoặc nhiều nodes) để đáp ứng toàn bộ pods đang unscheduling. Quá trình này sẽ được thực hiện ngay lập tức theo 2 bước:

- **Bước 1:** Hệ thống VKS tạo node mới theo cấu hình node group hiện tại.
- **Bước 2:** Hệ thống VKS sẽ deploy các pods đang unscheduling này lên các node mới.

(i) **Chú ý:**

- Khi hệ thống thực hiện Auto Scaling, việc tạo ra node mới có thể gặp lỗi nếu bạn không có đủ credit hoặc bạn đã hết quota để tạo VM trên hệ thống vServer. Để tránh gặp lỗi bên trên, bạn cần:
 - **Đảm bảo bạn có đủ credit:** Nếu bạn là người dùng trả trước, hãy nạp thêm credit vào tài khoản của bạn.
 - **Yêu cầu tăng quota:** Bạn có thể yêu cầu tăng quota cho tài khoản của mình tại [đây](#).

Cơ chế Scale down: hệ thống VKS thực hiện scale down khi

- Một hoặc nhiều node có tải thấp liên tục trong một khoảng thời gian. Cụ thể node có utilization (độ khả dụng) bao gồm cả request CPU và memory của pod thấp ở mức < 50%.
- Tất cả các pod hiện tại của node đó, có thể được di chuyển qua node khác mà không gặp vấn đề gì.

Nếu thỏa mãn 2 điều kiện trên, mặc định là trong khoảng 10 phút, node đó sẽ bị xóa đi khỏi Cluster. Quá trình xóa này sẽ bao gồm 3 bước:

- **Bước 1:** Hệ thống VKS sẽ đánh dấu là node đó là unschedulable.
 - **Bước 2:** Hệ thống di chuyển (move) toàn bộ pod qua node khác.
 - **Bước 3:** Sau khi di chuyển tất cả các pod qua node khác thành công, hệ thống VKS sẽ xóa node được đánh dấu.
-

Bật Auto Scaling

Trên hệ thống VKS, bạn có thể bật Auto Scaling khi:

- **Khởi tạo một Cluster**
- **Khởi tạo một Node Group**
- **Chỉnh sửa một Node Group**

Để bật Auto Scaling cho Kubernetes Cluster của bạn vui lòng bật lựa chọn **Enable Auto Scaling**, khi bật lựa chọn này bạn cần nhập::

- **Minimum node:** số node tối thiểu mà Cluster phải có.
- **Maximum node:** số node tối đa mà Cluster có thể scale tới.
- **Node Group upgrade strategy:** chiến lược upgrade Node Group. Khi bạn thiết lập **Node Group Upgrade Strategy** thông qua phương thức **Surge upgrade** cho một Node Group trong VKS, hệ thống VKS sẽ cập nhật tuần tự để nâng cấp các node, theo thứ tự không xác định.
 - **Max surge:** giới hạn số lượng node được nâng cấp đồng thời (số lượng node mới (surge) có thể được tạo ra cùng một lúc). Mặc định **Max surge = 1** - chỉ nâng cấp một node tại một thời điểm.
 - **Max unavailable:** giới hạn số lượng node không thể truy cập được trong quá trình nâng cấp (số lượng node hiện tại có thể bị gián đoạn cùng một lúc). Mặc định **Max unavailable = 0** - đảm bảo tất cả các node đều có thể truy cập được trong quá trình nâng cấp.

Ví dụ như hình bên dưới: tôi đã khởi tạo một Node Group với:

- **Number of nodes: 3 nodes**
- **Minimum node: 1 nodes**

- **Maximum node: 5 nodes**

Lúc này:

- Nếu có một hoặc nhiều pod không thể scheduling trên bất kỳ node trên cụm vì lý do thiếu resource và việc thêm 1 node giống với cấu hình node group này có thể xử lý vấn đề thì hệ thống sẽ thực hiện scale up. Với thiết lập này thì hệ thống có thể scale up tối đa lên tới **5 node**.
- Nếu có 1 node có low utilization (khả dụng) thấp ở mức **< 50%** và tất cả các pod của node đó có thể scheduling trên node khác thì hệ thống sẽ thực hiện scale up. Với thiết lập này thì hệ thống có thể scale down xuống mức tối thiểu là **1 node**.

Upgrading Node Group Version

Hiện tại, hệ thống VKS của chúng tôi đã hỗ trợ bạn nâng cấp Node Group Version, bạn có thể nâng cấp Node Group Version lên:

- **Control Plane Version** (Ví dụ nâng cấp từ 1.24 (Node Group version hiện tại) lên 1.25 (Control Plane Version hiện tại), nhưng không thể nâng cấp lên các phiên bản khác.

Để thực hiện nâng cấp phiên bản Node Group Version, bạn có thể thực hiện theo hướng dẫn sau:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn menu **Kubernetes Cluster**. Chọn vào một **Cluster** mà bạn muốn nâng cấp **Node Group Version**.

Bước 3: Chọn biểu tượng và chọn **Upgrade Node Group version** để thực hiện nâng cấp version node group.

Bước 4: Bạn có thể lựa chọn phiên bản mới cho tất cả các Node Group. Phiên bản mới cần hợp lệ và tương thích với phiên bản hiện tại của cluster. Cụ thể: bạn có thể chọn:

- Nâng cấp Node Group sao cho về cùng version với Control Plane Version (ví dụ: 1.24 lên 1.25)

Bước 5: Hệ thống VKS sẽ thực hiện nâng cấp tất cả các Node Group lên version của Control Plane. Sau khi việc nâng cấp hoàn tất, trạng thái Node Group trở về **ACTIVE**.

Chú ý:

- Việc nâng cấp Node Group Version là không bắt buộc và độc lập với việc nâng cấp Control Plane Version. Tuy nhiên tất cả các Node Group trong một Cluster sẽ được nâng cấp trong cùng một lần, cũng như Control Plane Version và Node Group Version trong cùng một Cluster không được lệch quá 1 minor version. Bên cạnh đó, hệ thống VKS tự động nâng cấp Node Group Version khi phiên bản K8S Version hiện tại đang sử dụng cho Cluster của bạn quá thời hạn được nhà cung cấp hỗ trợ.
- Trong quá trình nâng cấp Node Group Version, bạn không thể thực hiện các hành động khác trên Node Group của bạn.

- Bên dưới là một vài lưu ý trước, trong và sau quá trình nâng cấp, vui lòng tham khảo thêm:

Trước khi thực hiện:

- Kiểm tra phiên bản hiện tại: Truy cập Releases để tham khảo danh sách các phiên bản được hỗ trợ. Chọn phiên bản mới hợp lệ và tương thích với phiên bản hiện tại của cluster.
- Đảm bảo tính sẵn sàng của Node Group: Node Group phải đang ở trạng thái hoạt động (ACTIVE) và tất cả các node phải HEALTHY.
- Ngừng các tác vụ đang chạy: Ngừng các tác vụ đang chạy trên cluster để tránh ảnh hưởng đến quá trình nâng cấp.

Trong khi thực hiện:

- Theo dõi trạng thái Node Group: Theo dõi trạng thái Node Group trong quá trình nâng cấp. Trạng thái Node Group sẽ chuyển sang UPDATING và sau khi hoàn tất sẽ trở về ACTIVE.
- Kiểm tra nhật ký hệ thống: Kiểm tra nhật ký hệ thống để phát hiện bất kỳ lỗi hoặc cảnh báo nào trong quá trình nâng cấp.

Sau khi thực hiện:

- Kiểm tra tính sẵn sàng của Node Group: Xác nhận rằng Node Group đã được nâng cấp thành công và tất cả các node đang hoạt động bình thường.
- Kiểm tra các ứng dụng: Kiểm tra các ứng dụng đang chạy trên cluster để đảm bảo chúng hoạt động bình thường sau khi nâng cấp.

Lưu ý:

- Việc nâng cấp Node Group Version có thể mất một khoảng thời gian tùy thuộc vào kích thước và độ phức tạp của Node Group.
- Trong một số trường hợp hiếm gặp, việc nâng cấp Node Group Version có thể thất bại. Nếu điều này xảy ra, hệ thống VKS sẽ tự động rollback cluster về phiên bản hiện tại.

Ngoài ra, bạn cần chú ý thêm trường thông tin:

- Node Group upgrade strategy: chiến lược upgrade Node Group. Khi bạn thiết lập Node Group Upgrade Strategy thông qua phương thức Surge upgrade cho một Node Group trong VKS, hệ thống VKS sẽ cập nhật tuần tự để nâng cấp các node, theo thứ tự không xác định.

- Max surge: giới hạn số lượng node được nâng cấp đồng thời (số lượng node mới (surge) có thể được tạo ra cùng một lúc). Mặc định Max surge = 1 - chỉ nâng cấp một node tại một thời điểm. với maxUnavailable
- Max unavailable: giới hạn số lượng node không thể truy cập được trong quá trình nâng cấp (số lượng node hiện tại có thể bị gián đoạn cùng một lúc). Mặc định Max unavailable = 0 - đảm bảo tất cả các node đều có thể truy cập được trong quá trình nâng cấp.

Label và Taint

Label

Label là một tính năng quan trọng trong Kubernetes, được sử dụng để tổ chức và quản lý các đối tượng một cách hiệu quả. Bạn có thể gán các cặp key-value cho các đối tượng Kubernetes như Pod, Node, Service, Deployment, v.v. Cụ thể:

- **Mỗi Label là một cặp key-value:** Key (khoá) là một chuỗi ký tự dùng để xác định tên của label. Value (giá trị) là một chuỗi ký tự tùy chọn, cung cấp thông tin chi tiết về label.
- **Key và value phải tuân theo các quy tắc đặt tên:** Key và value không được chứa dấu khoảng trắng, ký tự đặc biệt ngoài (-, _, .).
- Label có thể được sử dụng cho nhiều mục đích khác nhau, bao gồm:
 - Phân loại các đối tượng dựa trên các tiêu chí như environment, version, status, v.v.
 - Theo dõi và quản lý các đối tượng trong cụm Kubernetes.

Ví dụ:

- `app: nginx` - Label này cho biết đối tượng có liên quan đến ứng dụng Nginx.
- `environment: production` - Label này cho biết đối tượng thuộc về môi trường production.
- `version: 1.7.2` - Label này cho biết đối tượng có liên quan đến phiên bản 1.7.2.

Tạo Label

Để tạo Label cho một Node Group, bạn hãy thực hiện theo hướng dẫn sau:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại Cluster đã khởi tạo trước đó, hãy chọn **Create a Node group**.

Bước 3: Tại màn hình khởi tạo Node Group, chúng tôi đã thiết lập thông tin cho Node Group của bạn. Bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Node Group của bạn. Tại mục **Node Group Metadata Setting**, bạn cần:

- Nhập key cho label của bạn. Key phải bắt đầu và kết thúc bằng chữ hoặc số và bao gồm các ký tự a-z, A-Z, 0-9, -, _, . tối đa 253 ký tự. Ngoài ra bạn có thể nhập key là một DNS subdomain ví dụ: <example.com/my-app>

- Nhập value cho key tương ứng này.

Bước 5: Chọn **Create Node Group**. Hãy chờ vài phút để chúng tôi khởi tạo Node Group của bạn, trạng thái của Node Group lúc này là **Creating**.

Bước 6: Khi trạng thái **Node Group** là **Active**, bạn có thể xem thông tin Node Group bằng cách chọn vào **Node Group Name** tại màn hình chính.

Hoặc bạn có thể tạo Label thông qua kubectl theo câu lệnh:

```
kubectl label nodes my-node1 disktype=ssd
```

Bạn có thể kiểm tra lại label vừa tạo qua lệnh:

```
kubectl get nodes --show-labels
```

Ví dụ kết quả cho lệnh này sẽ như sau:

NAME	STATUS	ROLES	AGE	VERSION	LABELS
worker0	Ready	<none>	1d	v1.13.0	...,disktype=ssd,kubernetes...
worker1	Ready	<none>	1d	v1.13.0	...,kubernetes.io/hostname=w...
worker2	Ready	<none>	1d	v1.13.0	...,kubernetes.io/hostname=w...

Sử dụng Label với nodeSelector

nodeSelector là một tham số được sử dụng trong PodSpec để chỉ định rằng Pod chỉ nên được lên lịch trên các Node có label cụ thể. Điều này hữu ích khi bạn muốn chạy Pod trên các Node có tài nguyên hoặc thuộc tính cụ thể.

- Tạo tệp tin **my-pod.yaml** chứa nội dung như sau:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  nodeSelector:
    disktype: ssd
    region: hcm03
```

Trong ví dụ này, Pod `my-pod` chỉ được lên lịch trên các Node có label `disktype: ssd` và `region: hcm03`.

- Triển khai Pod trên Cluster của bạn:

```
kubectl -f apply my-pod.yaml
```

Taint

Taint là một tính năng quan trọng trong Kubernetes, đóng vai trò như một cơ chế để đánh dấu các Node và kiểm soát việc lên lịch Pod trên những Node đó. Khác với Label thông thường, Taint được sử dụng để chỉ định các thuộc tính đặc biệt của Node và thực thi các hành động cụ thể khi Pod không đáp ứng các điều kiện được xác định bởi Taint. Cụ thể:

Cụ thể:

- **Mỗi Taint bao gồm:**
 - Key (khoá) là một chuỗi ký tự dùng để xác định tên của taint.
 - Value (giá trị) là một chuỗi ký tự tùy chọn, cung cấp thông tin chi tiết về taint.
 - Effect:
 - **NoSchedule:** Ngăn Pod không có Toleration tương ứng được lên lịch trên Node.
 - **NoExecute:** Cho phép Pod được lên lịch trên Node nhưng Pod sẽ không được thực thi.
 - **PreferNoSchedule:** Kubernetes sẽ cố gắng ưu tiên không lên lịch Pod lên Node có Taint này.
- **Key và value phải tuân theo các quy tắc đặt tên:** Key và value không được chứa dấu khoảng trắng, ký tự đặc biệt ngoài (-, _, .).
- **Toleration:** Để Pod có thể được lên lịch và chạy trên Node có Taint, Pod cần có Toleration tương ứng. Toleration được khai báo trong PodSpec bằng cách sử dụng `tolerations` field. Ví dụ:

```
tolerations: - key: node.role.kubernetes.io/master effect: NoSchedule
```

- **Mối quan hệ giữa Taint và Toleration:** Khi Kubernetes lên lịch Pod, Kubernetes sẽ so khớp các Taint của Node với các Toleration của Pod. Pod chỉ được lên lịch trên Node nếu có Toleration cho tất cả các Taint của Node đó.

Ví dụ:

- node.role.kubernetes.io/master:NoSchedule - ngăn các Pod thông thường được chạy trên Node này.

Tạo Taint

Để tạo Taint cho một Node Group, bạn hãy thực hiện theo hướng dẫn sau:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại Cluster đã khởi tạo trước đó, hãy chọn **Create a Node group.**

Bước 3: Tại màn hình khởi tạo Node Group, chúng tôi đã thiết lập thông tin cho Node Group của bạn. Bạn có thể giữ các giá trị mặc định này hoặc điều chỉnh các thông số mong muốn cho Node Group của bạn. Tại mục **Node Group Metadata Setting**, bạn cần:

- Nhập key cho taint của bạn. Key phải bắt đầu và kết thúc bằng chữ hoặc số và bao gồm các ký tự a-z, A-Z, 0-9, -, _, . tối đa 253 ký tự. Ngoài ra bạn có thể nhập key là một DNS subdomain ví dụ: <example.com/my-app>
- Nhập value cho key tương ứng này.
- Chọn 1 trong 3 loại effect: **NoSchedule**, **NoExecute**, **PreferNoSchedule**.

Bước 5: Chọn **Create Node Group**. Hãy chờ vài phút để chúng tôi khởi tạo Node Group của bạn, trạng thái của Node Group lúc này là **Creating**.

Bước 6: Khi trạng thái **Node Group** là **Active**, bạn có thể xem thông tin Node Group bằng cách chọn vào **Node Group Name** tại màn hình chính.

Hoặc bạn có thể tạo Taint thông qua kubectl theo câu lệnh:

```
kubectl taint node my-node node.role.kubernetes.io/master:NoSchedule.
```

Ví dụ sử dụng Taint:

Giả sử bạn có một Node `master` được sử dụng cho mục đích quản lý và bạn muốn ngăn các Pod thông thường được chạy trên Node này. Bạn có thể sử dụng Taint như sau:

```
kubectl taint node my-master node.role.kubernetes.io/master:NoSchedule
```

Để Pod có thể chạy trên Node `master`, Pod cần có Toleration tương ứng:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  tolerations:
  - key: node.role.kubernetes.io/master
    effect: NoSchedule
```

Network

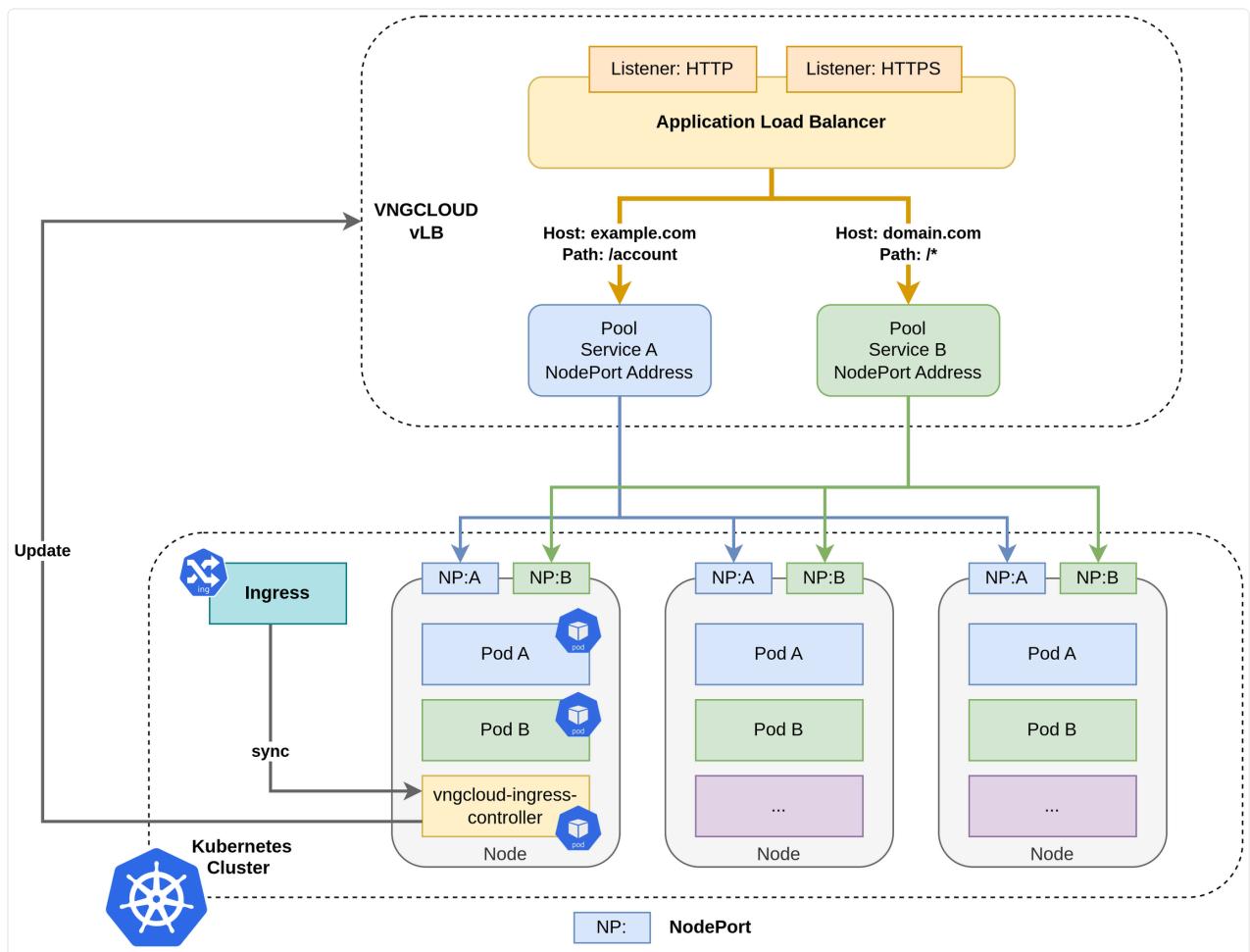
Làm việc với Application Load Balancer (ALB)

Tổng quan

ALB là gì?

- Application Load Balancer (ALB) là một công cụ trong hạ tầng mạng và máy chủ được sử dụng để phân phối lưu lượng mạng đến nhiều máy chủ hoặc máy ảo để cải thiện hiệu suất và khả năng sẵn sàng của các ứng dụng. ALB hoạt động ở tầng ứng dụng, cho phép phân phối lưu lượng dựa trên nhiều yếu tố như loại yêu cầu, trạng thái của máy chủ và thuật toán phân phối tải. ALB cung cấp khả năng route nâng cao, cho phép điều hướng lưu lượng dựa trên các Host hay Path Header. Nó cũng hỗ trợ tính năng duy trì phiên, giúp duy trì liên tục phiên của người dùng đối với cùng một máy chủ. Điều này rất hữu ích cho các ứng dụng yêu cầu sự nhất quán trong quá trình tương tác của người dùng. Để biết thêm thông tin chi tiết về ALB, vui lòng tham khảo tại [How it works (ALB)]

Mô hình triển khai



Ngoài các thành phần cơ bản của một cụm K8S và một ALB mà bạn đã biết, trong mô hình này chúng tôi sử dụng:

- **Ingress:** là tài nguyên trong Kubernetes được cấu hình để cho các Services có thể truy cập được từ bên ngoài k8s cluster thông qua URL, ngoài ra cũng có thể cân bằng tải traffic, hỗ trợ kết nối SSL/TLS và cung cấp virtual hosting dựa theo tên. Một Ingress không expose tùy tiện các protocol ngoài HTTP và HTTPS. **Ingress** đóng vai trò như một điểm vào duy nhất cho các request HTTP và HTTPS từ bên ngoài cluster đến các service bên trong. Việc route traffic được kiểm soát bởi các quy tắc (rule) được định nghĩa trong tài nguyên Ingress (Ingress Yaml File). Một Ingress được quản lý bởi **VNGCloud Ingress Controller:** là một ứng dụng chạy trong cluster và quản lý các tài nguyên Ingress dựa trên Ingress Yaml File do khách hàng định nghĩa

Ingress for an Application Load Balancer

Để cho tài nguyên Ingress (Ingress Yaml file) có thể hoạt động được, cluster phải có 1 VNGCloud Ingress Controller đang chạy. Không giống các loại Controller khác được chạy như là 1 phần của **kube-controller-manager**. VNGCloud Ingress Controller không được tự động khởi động cùng với cluster. Hãy làm theo hướng dẫn sau đây để cài đặt VNGCloud Ingress Controller cũng như làm việc với Ingress Yaml file.

Chuẩn bị

- Tạo một Kubernetes cluster trên VNGCloud, hoặc sử dụng một cluster đã có. Lưu ý: đảm bảo bạn đã tải xuống cluster configuration file sau khi cluster được khởi tạo thành công và truy cập vào cluster của bạn.
- Khởi tạo hoặc sử dụng một **service account** đã tạo trên IAM và gắn policy: **vLBFULLAccess**, **vServerFullAccess**. Để tạo service account bạn truy cập tại [đây](#) và thực hiện theo các bước sau:
 - Chọn "**Create a Service Account**", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account
 - Tìm và chọn **Policy: vLBFULLAccess** và **Policy: vServerFullAccess**, sau đó nhấn "**Create a Service Account**" để tạo Service Account, Policy: vLBFULLAccess và Policy: vServerFullAccess do VNG Cloud tạo ra, bạn không thể xóa các policy này.
 - Sau khi tạo thành công bạn cần lưu lại **Client_ID** và **Secret_Key** của Service Account để thực hiện bước tiếp theo.
- Thay đổi thông tin **Security Group** để cho phép các ALB có thể kết nối được tới các Node trong Node Group của bạn. Bạn cần thay đổi chúng trên vServer Portal khi:
 - Security Group được gắn vào **Cluster/Node Group** của bạn **khác với thông số mặc định** mà chúng tôi đã tạo.
 - Bạn cần **thay đổi mức độ bảo mật** cho Cluster của mình hoặc bạn cần **mở thêm cổng** cho các dịch vụ cụ thể hoạt động trên Cluster. Chi tiết tham khảo tại [đây](#).

Khởi tạo Service Account và cài đặt VNGCloud Ingress Controller



Chú ý:

Khi bạn thực hiện khởi tạo Cluster theo hướng dẫn bên trên, nếu bạn chưa bật option **Enable vLB Native Integration Driver**, mặc định chúng tôi sẽ không cài sẵn plugin này vào Cluster của bạn. Bạn cần tự thực hiện Khởi tạo Service Account và cài đặt VNGCloud Ingress Controller theo hướng dẫn bên dưới. Nếu bạn đã bật option **Enable vLB Native Integration Driver**, thì chúng tôi đã cài sẵn plugin này vào Cluster của bạn, hãy bỏ qua bước Khởi tạo Service Account, cài đặt VNGCloud Ingress Controller và tiếp tục thực hiện theo hướng dẫn kể từ Deploy một Workload.

- ✓ Khởi tạo Service Account và cài đặt VNGCloud Ingress Controller

Khởi tạo Service Account

- Khởi tạo hoặc sử dụng một **service account** đã tạo trên IAM và gắn policy: **vLBFullAccess**, **vServerFullAccess**. Để tạo service account bạn truy cập tại [đây](#) và thực hiện theo các bước sau:
 - Chọn "**Create a Service Account**", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account
 - Tìm và chọn **Policy: vLBFullAccess** và **Policy: vServerFullAccess**, sau đó nhấn "**Create a Service Account**" để tạo Service Account, Policy: vLBFullAccess và Policy: vServerFullAccess do VNG Cloud tạo ra, bạn không thể xóa các policy này.
 - Sau khi tạo thành công bạn cần phải lưu lại **Client_ID** và **Secret_Key** của Service Account để thực hiện bước tiếp theo.

Cài đặt VNGCloud Ingress Controller

- Cài đặt Helm phiên bản từ 3.0 trở lên. Tham khảo tại <https://helm.sh/docs/intro/install/> để biết cách cài đặt.
- Thêm repo này vào cluster của bạn qua lệnh:

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

- Thay thế thông tin ClientID, Client Secret và ClusterID của cụm K8S của bạn và tiếp tục chạy:

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

- Sau khi việc cài đặt hoàn tất, thực hiện kiểm tra trạng thái của vngcloud-ingress-controller pods:

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

Ví dụ như ảnh bên dưới là bạn đã cài đặt thành công vngcloud-controller-manager:

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

Deploy một Workload

Sau đây là hướng dẫn để bạn deploy service nginx trên Kubernetes.

Bước 1: Tạo Deployment cho Nginx app.

- Tạo file **nginx-service-lb7.yaml** với nội dung sau:

```

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 Deployment này bằng lệnh:

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

Bước 2: Kiểm tra thông tin Deployment, Service vừa deploy

- Chạy câu lệnh sau đây để kiểm tra **Deployment**

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

- Nếu kết quả trả về như bên dưới tức là bạn đã deploy Deployment thành công.

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/TCP		
NAME		READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS
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	172.16.54.

Tạo Ingress Resource

- 1.Nếu bạn chưa có sẵn một Application Load Balancer nào đã khởi tạo trước đó trên hệ thống vLB.

Lúc này, khi tạo một Ingress, bạn hãy để trống thông tin Load Balancer ID tại annotation vks.vngcloud.vn/load-balancer-id.

- Ví dụ, giả sử bạn đã deployment một service có tên nginx-service. Lúc này, bạn có thể tạo file **nginx-ingress.yaml** như sau:

```
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
```

- Chạy câu lệnh sau đây để triển khai Ingress

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

Sau khi bạn đã thực hiện triển khai Ingress , Chúng tôi sẽ tự động tạo 1 ALB trên cluster của bạn. ALB này sẽ hiển thị trên vLB Portal, chi tiết truy cập tại [đây](#). ALB này sẽ có thông tin mặc định:

Thành phần	Số lượng	Thuộc tính
ALB Package	1	VNG ALB_Small
Listener	2	<ul style="list-style-type: none">1 listener với protocol HTTP và port 801 listener với protocol HTTPS và port 443
Pool	1	<ul style="list-style-type: none">1 pool default protocol HTTP và algorithm ROUND ROBIN
Health Check	1	<ul style="list-style-type: none">Sử dụng TCP để health check các member.

Ví dụ:

The screenshot shows the VNG Cloud interface for managing Load Balancers. On the left, there's a sidebar with various services like VServer, BlockStore, Volumes, Images, Backups, Snapshots, Load Balancing, LB Packages, Certificates, Container, Kubernetes Clusters, Persistent Volumes, Container Registry, AutoScale, and Billing. The 'Load Balancing' section is expanded, showing 'Load Balancers' under it. The main area displays a table of load balancers with the following data:

Name	Status	End point	Schema	Type	Package	Created at	Action
lb-1fcabeb35-15af-4ba7-9e83-0f3209638fd9 vks-k8s-6d2acb-default-nginx-ingr-897e5	ACTIVE	180.93.181.129	Internet	Application	ALB_Small	21/04/2024 20:08:28	⋮
lb-6ea77772-0ff8-442c-81b7-f03e44de742 vks-k8s-6d2acb-default-nginx-serv-998c5	ACTIVE	180.93.181.20	Internet	Network	NLB_Small	21/04/2024 19:52:22	⋮

Total: 2

Show: 10

Chú ý:

- Hiện tại Ingress chỉ hỗ trợ duy nhất TLS port 443 và là điểm kết thúc cho TLS (TLS termination). TLS Secret phải chứa các trường với tên key là tls.crt và tls.key, đây chính là certificate và private key để sử dụng cho TLS. Nếu bạn muốn sử dụng Certificate cho một host, hãy thực hiện tải lên Certificate theo

hướng dẫn tại [Upload a certificate] và sử dụng chúng như một annotation. Ví dụ:

```
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-b14"
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.Nếu bạn đã có sẵn một Application Load Balancer đã khởi tạo trước đó trên hệ thống vLB và bạn muốn tái sử dụng ALB cho cluster của bạn.

Lúc này, khi tạo một Ingress, bạn hãy nhập thông tin Load Balancer ID vào annotation **vks.vngcloud.vn/load-balancer-id**. Ví dụ, trong trường hợp này tôi đã tái sử dụng ALB có 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 wi
    vks.vngcloud.vn/load-balancer-id: "lb-2b9d8974-3760-4d60-8203-9671f229fb96"
    vks.vngcloud.vn/certificate-ids: "secret-a6d20ec6-f3e5-499a-981b-b1484e340ce
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

```

- Sau khi bạn đã thực hiện tạo ingress theo hướng dẫn tại [Ingress for an Application Load Balancer](#). Nếu:
 - ALB của bạn đang có sẵn 2 listener trong đó:
 - 1 listener có cấu hình protocol HTTP và port 80
 - 1 listener có cấu hình protocol HTTPS và port 443 thì chúng tôi sẽ sử dụng 2 listener này.
 - ALB của bạn chưa có một trong hai hoặc cả 2 listener có cấu hình trên, chúng tôi sẽ tự động khởi tạo chúng.

(i) Chú ý:

Nếu ALB của bạn có:

- 1 listener có cấu hình protocol HTTP và port 443
- Hoặc 1 listener có cấu hình protocol HTTPS và port 80

thì khi tạo Ingress sẽ xảy ra lỗi. Lúc này, bạn cần chỉnh sửa lại thông tin listener hợp lệ trên hệ thống vLB và thực hiện tạo lại ingress.

3. Sau khi tạo ingress thành công với một ALB, bạn có thể thực hiện

- Chỉnh sửa cấu hình ingress của bạn theo hướng dẫn cụ thể tại [Configure for an Application Load Balancer](#).
- Hoặc bạn có thể thêm/ sửa/ xóa policy trong ALB của bạn bằng các chỉnh sửa các thông số sau trong tài nguyên ingress (Ingress Yaml file). Ví dụ như bên dưới, tôi đã thực hiện thiết lập **2 rule** như sau:

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    vks.vngcloud.vn/certificate-ids: "secret-58542bfb-f410-4095-9e1c-34cd39995c6
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

```

- Cũng giống như các tài nguyên Kubernetes khác, Ingress có cấu trúc gồm các trường thông tin như sau:
 - **apiVersion:** Phiên bản API cho Ingress.
 - **kind:** Loại tài nguyên, trong trường hợp này là "Ingress".
 - **ingressClassName:** bạn cần chỉ định giá trị field này là "vngcloud" để sử dụng vngcloud-ingress-controller.
 - **metadata:** Thông tin mô tả Ingress, bao gồm tên, annotations.
 - **spec:** Cấu hình Ingress, bao gồm các rule route traffic theo điều kiện của các incoming request. Tài nguyên Ingress chỉ hỗ trợ các rule để điều hướng HTTP

traffic.

Để biết thông tin chung về cách làm việc với tài nguyên Ingress (Ingress Yaml file), hãy xem tại [Configure for an Application Load Balancer]).

Kiểm tra, chỉnh sửa Ingress resource đã tạo

- Sau khi tạo ingress thành công, bạn có thể xem danh sách ingress qua lệnh

```
kubectl get ingress
```

Ví dụ, bên dưới chúng tôi đã tạo thành công nginx-ingress:

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
nginx-ingress	vngcloud	*	180.93.181.129	80	103m

- Hoặc xem chi tiết một ingress bằng cách

```
kubectl describe ingress nginx-ingress
```

Ví dụ, bên dưới là thông tin chi tiết của nginx-ingress mà tôi đã tạo:

```
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-933f-c3bc4
Events:    <none>
```

- Để cập nhật nginx-ingress hiện có, ta có thể thực hiện bằng cách cập nhật Ingress Yaml file như sau:

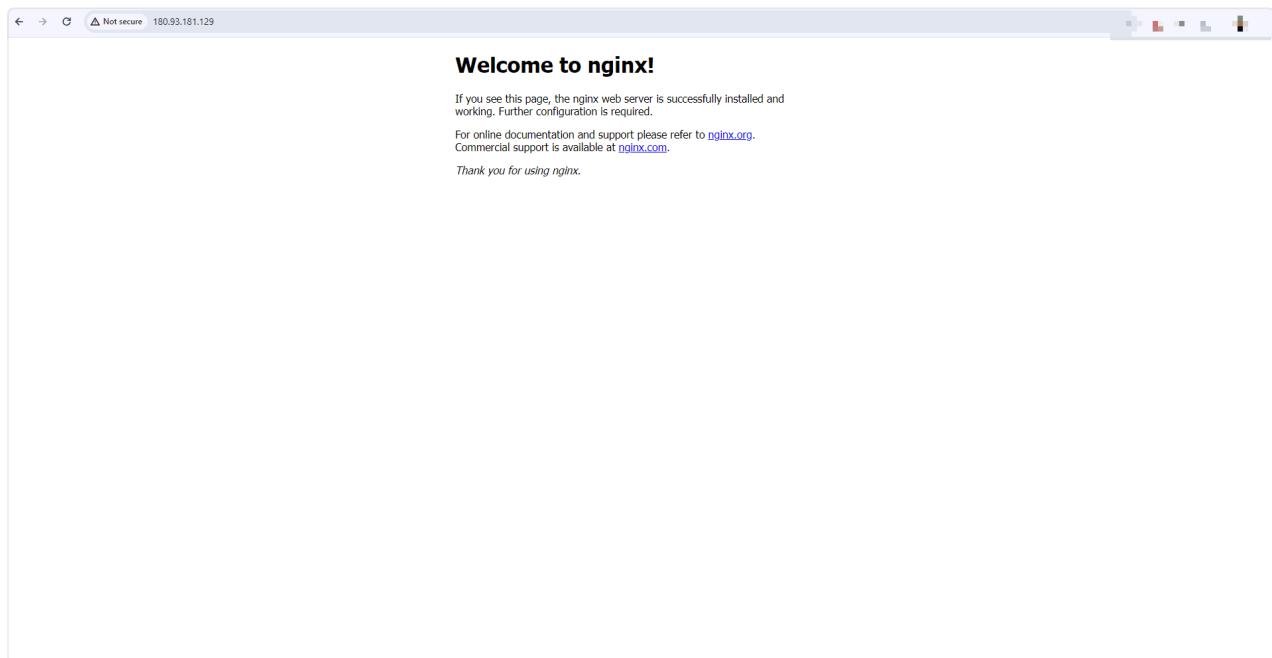
```
kubectl edit ingress nginx-ingress
```

Để truy cập vào app nginx, bạn có thể sử dụng Endpoint của Load Balancer mà hệ thống đã tạo.

http://Endpoint/

Bạn có thể lấy thông tin Public Endpoint của Load Balancer tại giao diện vLB. Cụ thể truy cập tại

Ví dụ, bên dưới tôi đã truy cập thành công vào app nginx với địa chỉ : <http://180.93.181.129/>



Cấu hình cho một Application Load Balancer

Tại trang [Ingress for an Application Load Balancer], chúng tôi đã hướng dẫn bạn cách thực hiện cài đặt Ingress Controller và tạo ingress thông qua Ingress Yaml file. Sau đây là chi tiết các ý nghĩa các thông tin bạn có thể thiết lập cho một Ingress

Annotation

Sử dụng các annotation dưới đây khi thực thiense tạo ingress để tuỳ chỉnh Load Balancer phù hợp với nhu cầu của bạn:

Annotation	Bắt buộc/ Không bắt buộc	Ý nghĩa
vks.vngcloud.vn/loa d-balancer-id	Không bắt buộc	<ul style="list-style-type: none">Nếu bạn chưa có sẵn một Application Load Balancer đã khởi tạo trước đó trên hệ thống vLB. Lúc này, khi tạo một Ingress, bạn hãy để trống thông tin này. Sau khi bạn đã thực hiện triển khai Ingress theo hướng dẫn tại Ingress for an Application Load Balancer. Chúng tôi sẽ tự động tạo 1 ALB trên cluster của bạn. ALB này sẽ hiển thị trên vLB Portal, chi tiết truy cập tại đâyNếu bạn đã có sẵn một Application Load Balancer đã khởi tạo trước đó trên hệ thống vLB và bạn muốn tái sử dụng ALB cho cluster của bạn. Lúc này, khi tạo một Ingress, bạn hãy nhập thông tin Load Balancer ID vào annotation này. Sau khi bạn đã thực hiện tạo Ingress theo hướng dẫn tại Ingress for an Application Load Balancer. Nếu:<ul style="list-style-type: none">ALB của bạn đang có sẵn 2 listener trong đó:<ul style="list-style-type: none">1 listener có cấu hình protocol HTTP và port 801 listener có cấu hình protocol HTTPS và port

443 thì chúng tôi sẽ sử dụng 2 listener này.

- ALB của bạn chưa có một trong hai hoặc cả 2 listener có cấu hình trên, chúng tôi sẽ tự động khởi tạo chúng.

Chú ý:

Nếu ALB của bạn có:

- 1 listener có cấu hình protocol HTTP và port 443
- Hoặc 1 listener có cấu hình protocol HTTPS và portal 80

thì khi tạo Ingress sẽ xảy ra lỗi. Lúc này, bạn cần chỉnh sửa lại thông tin listener hợp lệ trên hệ thống vLB và thực hiện tạo lại ingress.

vks.vngcloud.vn/load-balancer-name

Không bắt buộc

- Annotation

vks.vngcloud.vn/load-balancer-name

 sẽ được sử dụng nếu bạn **không** sử dụng annotation

load-balancer-id

.
- Annotation

vks.vngcloud.vn/load-balancer-name

 chỉ có ý nghĩa khi bạn tạo mới một Ingress resource. Sau khi Ingress resource được tạo thành công, annotation này **sẽ tự động bị xóa**. Việc sử dụng annotation này sau khi Ingress resource được tạo sẽ **không có tác dụng**.
- Khi bạn sử dụng annotation này, nếu bạn chưa có sẵn một **Application Load Balancer** đã khởi tạo trước đó trên hệ thống vLB. Chúng tôi sẽ tự động tạo 1 ALB trên cluster của bạn. ALB này sẽ hiển thị trên vLB Portal, chi tiết truy cập tại [đây](#)
- Nếu bạn đã có sẵn một **Application Load Balancer** đã khởi tạo trước đó trên hệ thống vLB và bạn muốn tái sử dụng ALB cho cluster của bạn. Lúc này, bạn hãy

		nhập thông tin Load Balancer Name vào annotation này.
vks.vngcloud.vn/package-id	Không bắt buộc	<ul style="list-style-type: none"> Nếu bạn không nhập thông tin này thì mặc định chúng tôi sẽ sử dụng cấu hình ALB Small. Nếu bạn đã có sẵn host vLB đang ACTIVE và bạn muốn tích hợp host này vô cụm K8S của bạn, vui lòng bỏ qua trường thông tin này.
vks.vngcloud.vn/tags	Không bắt buộc	<ul style="list-style-type: none"> Tag được gắn thêm cho ALB của bạn.
vks.vngcloud.vn/scheme	Không bắt buộc	<ul style="list-style-type: none"> Mặc định internet-facing, bạn có thể đổi thành internal tùy theo nhu cầu sử dụng.
vks.vngcloud.vn/security-groups	Không bắt buộc	<ul style="list-style-type: none"> Mặc định sẽ tạo một security group default theo Cluster của bạn.
vks.vngcloud.vn/ inbound-cidrs	Không bắt buộc	<ul style="list-style-type: none"> Mặc định All CIDR: 0.0.0.0/0
vks.vngcloud.vn/healthy-threshold-count	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 3
vks.vngcloud.vn/unhealthy-threshold-count	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 3
vks.vngcloud.vn/healthcheck-interval-seconds	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 30
vks.vngcloud.vn/healthcheck-timeout-seconds	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 5
vks.vngcloud.vn/healthcheck-protocol	Không bắt buộc	<ul style="list-style-type: none"> Mặc định TCP. Người dùng có thể chọn một trong các giá trị TCP / HTTP
vks.vngcloud.vn/healthcheck-http-method	Không bắt buộc	<ul style="list-style-type: none"> Mặc định GET. Người dùng có thể chọn một trong các giá trị GET / POST / PUT
vks.vngcloud.vn/healthcheck-path	Không bắt buộc	<ul style="list-style-type: none"> Mặc định /

vks.vngcloud.vn/he althcheck-http-version	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 1.0. Người dùng có thể chọn một trong các giá trị 1.0, 1.1
vks.vngcloud.vn/he althcheck-http-domain-name	Không bắt buộc	<ul style="list-style-type: none"> Mặc định trống
vks.vngcloud.vn/he althcheck-port	Không bắt buộc	<ul style="list-style-type: none"> Mặc định traffic port
vks.vngcloud.vn/success-codes	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 200
vks.vngcloud.vn/idle-timeout-client	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 50
vks.vngcloud.vn/idle-timeout-member	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 50
vks.vngcloud.vn/idle-timeout-connection	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 5
vks.vngcloud.vn/pool-algorithm	Không bắt buộc	<ul style="list-style-type: none"> Mặc định ROUND_ROBIN. Người dùng có thể chọn một trong các giá trị ROUND_ROBIN / LEAST_CONNECTIONS / SOURCE_IP
vks.vngcloud.vn/enable-sticky-session	Không bắt buộc	<ul style="list-style-type: none"> Mặc định false.
vks.vngcloud.vn/enable-tls-encryption	Không bắt buộc	<ul style="list-style-type: none"> Mặc định false
vks.vngcloud.vn/target-node-labels	Không bắt buộc	<ul style="list-style-type: none"> Mặc định trống
vks.vngcloud.vn/certificate-ids	Không bắt buộc	<ul style="list-style-type: none"> Mặc định trống

IngressClassName

Các Ingress được cài đặt bởi các VNGCloud Ingress Controller sẽ có thông tin `IngressClassName = "vngcloud"`. Bạn không được thay đổi thông tin này.

DefaultBackend

- Một Ingress không có rule nào sẽ gửi tất cả traffic đến 1 service default backend mặc định duy nhất hoặc nếu không có *host* và *path* nào khớp với HTTP request trong Ingress Yaml file thì traffic sẽ được route đến service default backend. Ví dụ như bên dưới, chúng tôi đang cấu hình mặc định nếu request không thỏa mãn rule nào trong Ingress yaml file thì sẽ đi vào service name: example-svc-1 với port number 8080

```
defaultBackend:  
  service:  
    name: example-svc-1  
    port:  
      number: 8080
```

TLS

Bạn có thể bảo mật Ingress bằng cách chỉ định 1 Secret có chứa TLS key và certificate. Hiện tại Ingress chỉ hỗ trợ duy nhất TLS port 443 và là điểm kết thúc cho TLS (TLS termination). TLS Secret phải chứa các trường với tên key là tls.crt và tls.key, đây chính là certificate và private key để sử dụng cho TLS. Cụ thể, bạn cần chỉ định:

- Host: các host được chỉ định sẽ dùng cert.
- SecretName: tên secret chứa cert.

Path types

Mỗi path (đường dẫn) trong Ingress có một pathType (loại đường dẫn) tương ứng. Có ba pathType được hỗ trợ:

- Exact: Khớp với đường dẫn URL một cách chính xác tuyệt đối và phân biệt chữ hoa chữ thường.
- Prefix: Khớp dựa trên tiền tố của đường dẫn URL được phân tách bởi dấu /. Việc so khớp (match) là có phân biệt chữ hoa thường và được thực hiện trên từng thành phần của đường dẫn URL. Một thành phần của đường dẫn URL chính là 1 label được phân tách bằng dấu phân cách / trong đường dẫn URL (Nghĩa là đường dẫn URL có thể bao gồm nhiều cấp phân tách nhau bởi dấu /, mỗi chuỗi đứng giữa 2 dấu / chính là 1 label, mỗi label chính là 1 thành phần của đường dẫn URL). Một request URL được xem như là

khớp với 1 trường path (được cấu hình trong đặc tả Ingress) khi toàn bộ giá trị của path (có thể gồm nhiều thành phần phân tách bằng dấu /) khớp với các label đầu tiên (tính từ bên trái của đường dẫn URL). Ví dụ /example1/path1 khớp với /example1/path1/path2, nhưng không khớp với /example1/path1path2

Ví dụ cụ thể:

Path type	Path(s)	Request path(s)	Có match hay không?
Exact	/example1	/example1	Có
Exact	/example1	/host1	Không
Exact	/example1	/example1/	Không
Exact	/example1/	/example1	Không
Prefix	/	(all paths)	Có
Prefix	/example1	/example1 , /example1/	Có
Prefix	/example1/	/example1 , /example1/	Có
Prefix	/example1/host1	/example1/host1	Không
Prefix	/example1/host1	/example1/host1	Có
Prefix	/example1/host1/	/example1/host1	Có
Prefix	/example1/host1	/example1/host1/	Có
Prefix	/example1/host1	/example1/host1/c cc	Có
Prefix	/example1/host1	/example1/host1xy z	Không
Prefix	/ , /example1	/example1/ccc	Có
Prefix	/ , /example1 , /example1/host1	/example1/host1	Có
Prefix	/ , /example1 , /example1/host1	/ccc	Có
Prefix	/example1	/ccc	Không

- Trong một số trường hợp, nhiều path bên trong Ingress sẽ khớp với đường dẫn của request URL. Trong những trường hợp đó, quyền ưu tiên sẽ được trao cho path khớp dài nhất đầu tiên. Nếu hai path vẫn có độ dài khớp bằng nhau thì quyền ưu tiên sẽ theo thứ tự rule được tạo trên Ingress Yaml file.
-

Ingress rule

Mỗi HTTP rule chứa các thông tin sau:

- 1 host tùy chọn.** Nếu không có host (ta có thể hiểu là 1 tên miền) nào được chỉ định nên rule sẽ được áp dụng cho tất cả HTTP traffic đi vào (inbound) địa chỉ IP đã được chỉ định. Nếu có chỉ định host (ví dụ example1.com) thì rule chỉ áp dụng cho host đó thôi.
- 1 danh sách các đường dẫn (path)** (ví dụ /example1/host1), mỗi path có 1 service backend gắn liền với nó được định nghĩa bởi Service Name và Port Number. Cả host và path phải khớp với nội dung của incoming request trước khi bộ cân bằng tải điều hướng traffic đến Services mong muốn.
- 1 backend** là tổ hợp của tên Services và Port Number. HTTP và HTTPS request đi đến Ingress và có URL khớp với host và path của rule sẽ được gửi đến danh sách các backend.

Ví dụ: Host có khớp với Host header theo bảng:

Host	Host header	Có match hay không?
*. example1.com	example2.example1.com	Có
*. example1.com	baz.example2.example1.com	Không
*. example1.com	example1.com	Không

Giới hạn và hạn chế ALB

Giới hạn

Một vài lưu ý về giới hạn của việc ingress an ALB vào một cluster:

- Một cluster có thể có nhiều Ingress, mỗi Ingress có chứa thông tin tài nguyên cho một ALB duy nhất.
 - Một ALB có thể được sử dụng chung cho nhiều Ingress. Từ đó một ALB có thể được sử dụng chung cho nhiều cluster nhưng phải đảm bảo các cluster này có chung **Subnet**.
 - Một ALB có thể bao gồm nhiều listener, nhiều pool, nhiều policy. Các giới hạn về số lượng listener, số lượng pool, số lượng policy vui lòng tham khảo tại [Hạn mức tài nguyên]
-

Hạn chế

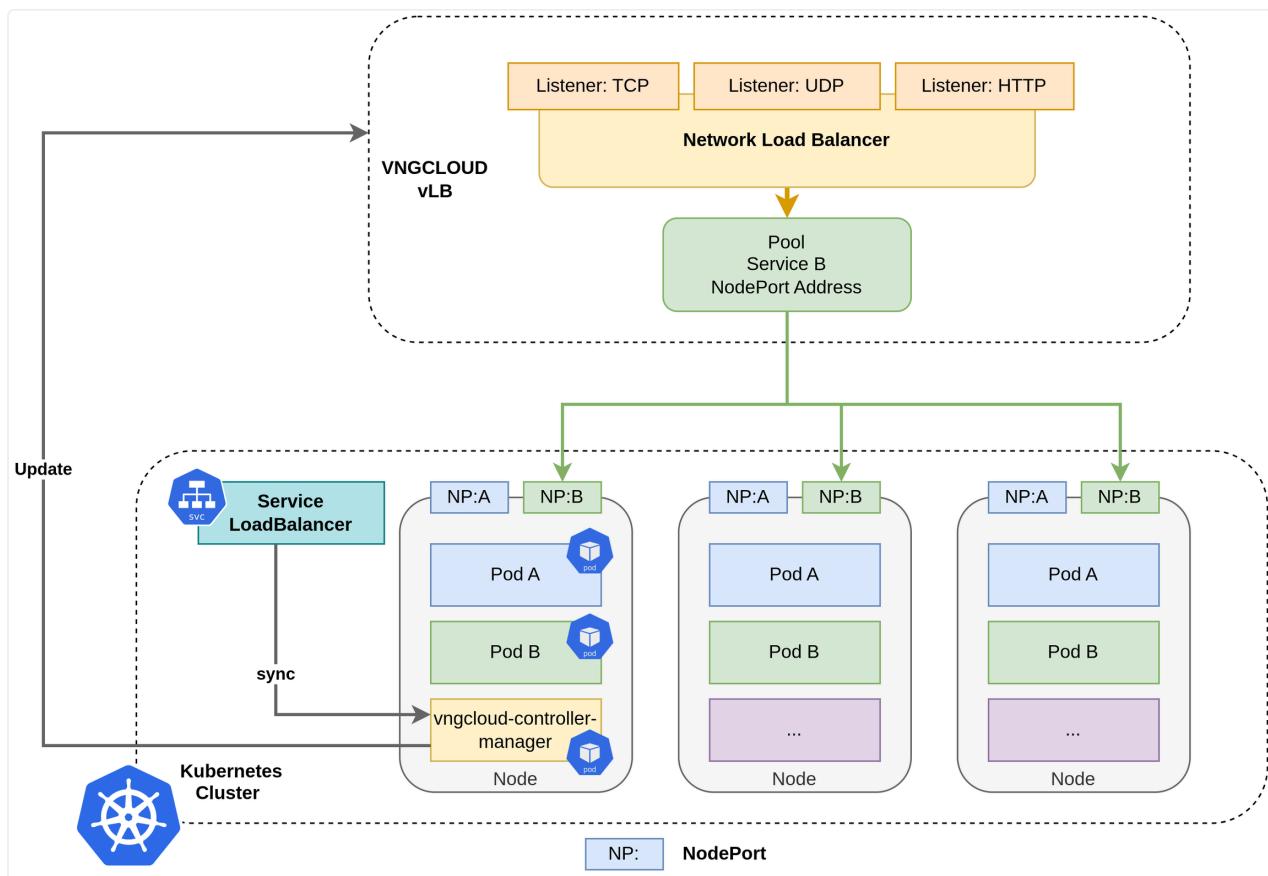
- Ingress controller manager cần phải cấu hình các annotation để chỉ định các thuộc tính của ALB, như protocol, port,... Các annotation này có thể khác nhau tùy theo nhà cung cấp dịch vụ đám mây, hiện tại với **vngcloud ingress controller** đang cung cấp danh sách annotation tham khảo tại [Cấu hình cho một Application Load Balancer](#). Chúng tôi sẽ nâng cấp thêm phần này trong các phiên bản release kế tiếp.
- Hiện tại, vLB chưa hỗ trợ tính năng strip path ở Load Balancer Layer 7, chúng tôi sẽ sớm tích hợp tính năng này trong tương lai.
- Việc thay đổi tên hoặc kích thước (Rename, Resize) tài nguyên Load Balancer trên vServer Portal có thể gây ra sự không tương thích với tài nguyên trên Kubernetes Cluster. Điều này có thể dẫn đến việc các tài nguyên không hoạt động trên Cluster, hoặc tài nguyên bị đồng bộ lại hoặc thông tin tài nguyên giữa vServer Portal và Cluster không khớp. Để ngăn chặn vấn đề này, hãy sử dụng `kubectl` để quản lý tài nguyên của Cluster.

Làm việc với Network load balancing (NLB)

NLB là gì?

- **Network Load Balancer (NLB)** là một bộ cân bằng tải được cung cấp bởi VNGCloud giúp phân phối lưu lượng truy cập mạng đến nhiều máy chủ back-end (backend servers) trong một nhóm máy tính (instance group). NLB hoạt động ở layer 4 của mô hình OSI, giúp cân bằng tải dựa trên địa chỉ IP và cổng TCP/UDP. Để biết thêm thông tin chi tiết về NLB, vui lòng tham khảo tại [How it works (NLB)]

Mô hình triển khai



- **vngcloud-controller-manager:** VNG Cloud Controller Manager là một bộ điều khiển chạy trên các cụm Kubernetes được triển khai trên VNG Cloud. Nó chịu trách nhiệm cho việc quản lý các tài nguyên VNG Cloud cho các cụm Kubernetes, bao gồm:
 - **Tạo và quản lý Network Load Balancer (NLB)** cho các Service Kubernetes có service type = Load Balancer.

Integrate with Network Load Balancer

Để tích hợp một Network Load Balancer với một Kubernetes cluster, bạn có thể sử dụng một Service với type là [LoadBalancer](#). Khi bạn tạo một Service như vậy, VNGCloud Controller Manager sẽ tự động một NLB để chuyển tiếp lưu lượng đến các pod trên node của bạn. Bạn cũng có thể sử dụng các annotation để tùy chỉnh các thuộc tính của Network Load Balancer, như port, protocol,...

Chuẩn bị

- Tạo một Kubernetes cluster trên VNGCloud, hoặc sử dụng một cluster đã có. Lưu ý: đảm bảo bạn đã tải xuống cluster configuration file sau khi cluster được khởi tạo thành công và truy cập vào cluster của bạn.
- Khởi tạo hoặc sử dụng một **service account** đã tạo trên IAM và gắn policy: **vLBFULLAccess**, **vServerFullAccess**. Để tạo service account bạn truy cập tại [đây](#) và thực hiện theo các bước sau:
 - Chọn "**Create a Service Account**", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account
 - Tìm và chọn **Policy: vLBFULLAccess** và **Policy: vServerFullAccess**, sau đó nhấn "**Create a Service Account**" để tạo Service Account, Policy: vLBFULLAccess và Policy: vServerFullAccess do VNG Cloud tạo ra, bạn không thể xóa các policy này.
 - Sau khi tạo thành công bạn cần phải lưu lại **Client_ID** và **Secret_Key** của Service Account để thực hiện bước tiếp theo.

Khởi tạo Service Account và cài đặt VNGCloud Controller Manager

 **Chú ý:**

Khi bạn thực hiện khởi tạo Cluster theo hướng dẫn bên trên, nếu bạn chưa bật option **Enable vLB Native Integration Driver**, mặc định chúng tôi sẽ không cài sẵn plugin này vào Cluster của bạn. Bạn cần tự thực hiện Khởi tạo Service Account và cài đặt VNGCloud Controller Manager theo hướng dẫn bên dưới. Nếu bạn đã bật option **Enable vLB Native Integration Driver**, thì chúng tôi đã cài sẵn plugin này vào Cluster của bạn, hãy bỏ qua bước Khởi tạo Service Account, cài đặt

VNGCloud Controller Manager và tiếp tục thực hiện theo hướng dẫn kề từ Deploy một Workload.

- ✓ Hướng dẫn khởi tạo Service Account và cài đặt VNGCloud Controller Manager

Khởi tạo Service Account

- Khởi tạo hoặc sử dụng một **service account** đã tạo trên IAM và gắn policy: **vLBFULLAccess**, **vServerFullAccess**. Để tạo service account bạn truy cập tại [đây](#) và thực hiện theo các bước sau:
 - Chọn "**Create a Service Account**", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account
 - Tìm và chọn **Policy: vLBFULLAccess** và **Policy: vServerFullAccess**, sau đó nhấn "**Create a Service Account**" để tạo Service Account, Policy: vLBFULLAccess và Policy: vServerFullAccess do VNG Cloud tạo ra, bạn không thể xóa các policy này.
 - Sau khi tạo thành công bạn cần phải lưu lại **Client_ID** và **Secret_Key** của Service Account để thực hiện bước tiếp theo.
- Gỡ cài đặt cloud-controller-manager

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

- Bên cạnh đó, bạn có thể xóa Service Account đang sử dụng cho cloud-controller-manager vừa gỡ

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

Cài đặt VNGCloud Controller Manager

- Cài đặt Helm phiên bản từ 3.0 trở lên. Tham khảo tại <https://helm.sh/docs/intro/install/> để biết cách cài đặt.
- Thêm repo này vào cluster của bạn qua lệnh:

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

- Thay thế thông tin ClientID, Client Secret và ClusterID của cụm K8S của bạn và tiếp tục chạy:

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

- Sau khi việc cài đặt hoàn tất, thực hiện kiểm tra trạng thái của vngcloud-Integrate-controller pods:

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

Ví dụ như ảnh bên dưới là bạn đã cài đặt thành công vngcloud-controller-manager:

NAME	READY	STATUS	RESTARTS
vngcloud-controller-manager-8864c754c-bqhvv	1/1	Running	5 (91s)

Deploy một Workload

1.Nếu bạn chưa có sẵn một Network Load Balancer đã khởi tạo trước đó trên hệ thống vLB.

Lúc này, bạn cần thực hiện:

Bước 1: Tạo Deployment, Service cho Nginx app.

- Tạo file **nginx-service-lb4.yaml** với nội dung sau:

```

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

```

- Hoặc sử dụng file mẫu sau đây để deploy HTTP Apache Service với Internal LoadBalancer cho phép truy cập nội bộ trên cổng 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 to crea
spec:
  selector:
    app: apache2
  type: LoadBalancer                           # MUST set like this to crea
  ports:
    - name: http
      protocol: TCP
      port: 8080                                # CAN be accessed via this p
      targetPort: 80

```

- Hoặc tập tin YAML mẫu để tạo Deployment và Service cho ứng dụng máy chủ UDP trong một cụm Kubernetes:

```

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.Nếu bạn đã có sẵn một Network Load Balancer đã khởi tạo trước đó trên hệ thống vLB và bạn muốn tái sử dụng NLB cho cluster của bạn.

Lúc này, bạn hãy nhập thông tin Load Balancer ID vào annotation **vks.vngcloud.vn/load-balancer-id**. Ví dụ dưới đây là tập tin YAML mẫu để triển khai Nginx với External LoadBalancer sử dụng vngcloud-controller-manager để tự động expose dịch vụ tới internet bằng bộ cân bằng tài L4 sử dụng một NLB có sẵn với 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" # ID
    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.Sau khi một NLB mới đã được chúng tôi tự động khởi tạo, lúc này bạn có thể thực hiện

- Chỉnh sửa cấu hình NLB của bạn theo hướng dẫn cụ thể tại [Configure for a Network Load Balancer](#). Ví dụ như bên dưới, tôi đã thực hiện chỉnh sửa protocol và port như sau:

```

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-b382-37982c4d98af"
spec:
  selector:
    app: apache2
  type: LoadBalancer
  ports:
    - name: http
      protocol: TCP
      port: 8000
      targetPort: 80

```

- Cũng giống như các tài nguyên Kubernetes khác, **vngcloud-controller-manager** có cấu trúc gồm các trường thông tin như sau:
 - **apiVersion:** Phiên bản API cho Ingress.
 - **kind:** Loại tài nguyên, trong trường hợp này là "Service".
 - **metadata:** Thông tin mô tả Ingress, bao gồm tên, annotations.
 - **spec:** Cấu hình điều kiện của các incoming request.

Để biết thông tin chung về cách làm việc với **vngcloud-controller-manager**, hãy xem tại [Configure for a Network Load Balancer]

- Deploy Service này bằng lệnh:

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

Kiểm tra thông tin Deployment, Service vừa deploy

- Chạy câu lệnh sau đây để kiểm tra **Deployment**

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

- Nếu kết quả trả về như bên dưới tức là bạn đã deploy Deployment thành công.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	
service/nginx-service	LoadBalancer	10.96.74.154	<pending>	80:31623/TCP	
NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS
deployment.apps/nginx-app	0/1	1	0	2s	nginx
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx-app-7f45b65946-bmrcf	0/1	ContainerCreating	0	2s	<n

Lúc này, hệ thống vLB sẽ tự động tạo một LB tương ứng cho nginx app đã deployment, ví dụ:

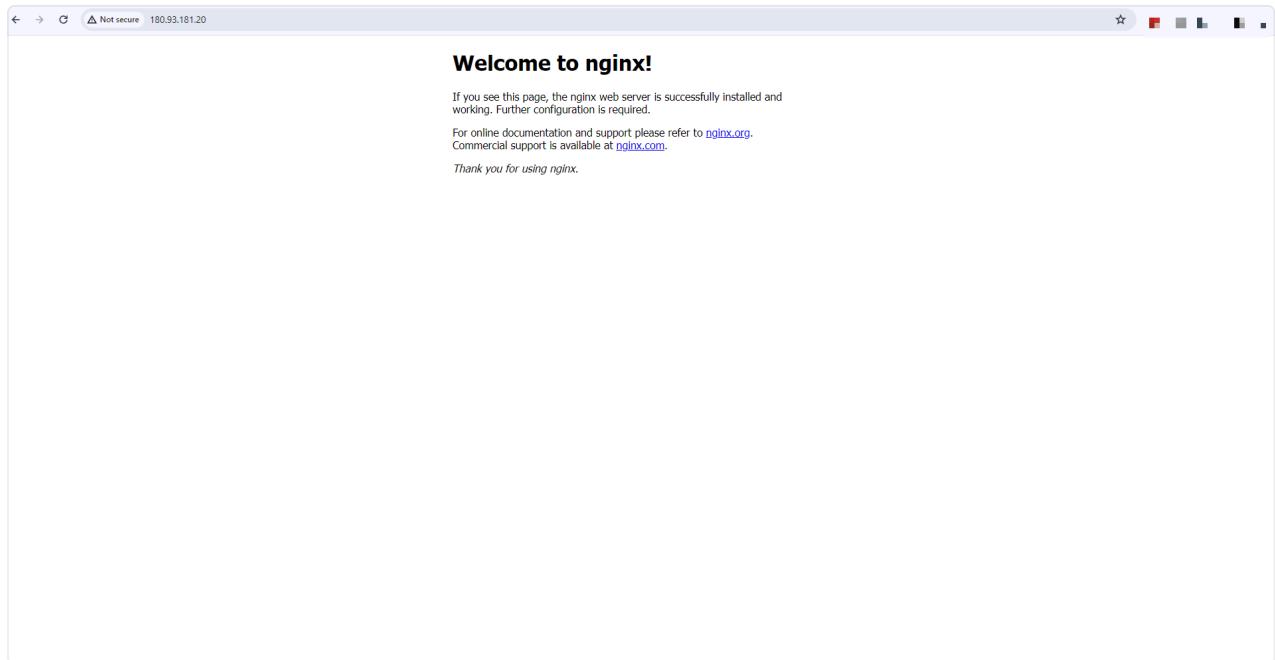
The screenshot shows the VNG Cloud vServer Load Balancer interface. On the left, there's a sidebar with navigation links like Overview, Limit, Network, VPCs, Floating IPs, etc. The main content area is titled 'Load Balancer' and contains a table with one row. The table columns are: Name, Status, End point, Schema, Type, Package, Created at, and Action. The single row shows: Name 'lb-6ea7772-0ff-442c-81b7-f03e44d9e742', Status 'ACTIVE', End point '180.93.181.20', Schema 'Internet', Type 'Network', Package 'NLB_Small', Created at '21/04/2024 19:52:22', and Action '...'. At the bottom of the table, it says 'Total: 1' and 'Show: 10'.

Bước 3: Để truy cập vào app nginx vừa export, bạn có thể sử dụng URL với định dạng:

http://Endpoint/

Bạn có thể lấy thông tin Public Endpoint của Load Balancer tại giao diện vLB. Cụ thể truy cập tại <https://hcm-3.console.vngcloud.vn/vserver/load-balancer/vlb/>

Ví dụ, bên dưới tôi đã truy cập thành công vào app nginx với địa chỉ : <http://180.93.181.20/>



Cấu hình cho một Network Load Balancer

Tại trang [Integrate with Network Load Balancer], chúng tôi đã hướng dẫn bạn cách thực hiện cài đặt VNGCloud Controller Manager, tạo và apply yaml file. Sau đây là chi tiết các ý nghĩa các thông tin bạn có thể thiết lập trong yaml file:

Annotation

Sử dụng các annotation dưới đây để tuỳ chỉnh Load Balancer phù hợp với nhu cầu của bạn:

Annotation	Bắt buộc/ Không bắt buộc	Ý nghĩa
vks.vngcloud.vn/load-balancer-id	Không bắt buộc	<ul style="list-style-type: none">Nếu bạn chưa có sẵn một Network Load Balancer đã khởi tạo trước đó trên hệ thống vLB. Chúng tôi sẽ tự động tạo 1 NLB trên cluster của bạn. NLB này sẽ hiển thị trên vLB Portal, chi tiết truy cập tại đây.Nếu bạn đã có sẵn một Network Load Balancer đã khởi tạo trước đó trên hệ thống vLB và bạn muốn tái sử dụng NLB cho cluster của bạn. Lúc này, bạn hãy nhập thông tin Load Balancer ID vào annotation này.
vks.vngcloud.vn/load-balancer-name	Không bắt buộc	<ul style="list-style-type: none">Annotation <code>vks.vngcloud.vn/load-balancer-name</code> sẽ được sử dụng nếu bạn không sử dụng annotation <code>load-balancer-id</code>.Annotation <code>vks.vngcloud.vn/load-balancer-name</code> chỉ có ý nghĩa khi bạn tạo mới một load balancer. Sau khi load balancer được tạo thành công, annotation này sẽ tự động bị xóa. Việc sử dụng annotation này sau khi load balancer được tạo sẽ không có tác dụng.

		<ul style="list-style-type: none"> Khi bạn sử dụng annotation này, nếu bạn chưa có sẵn một Network Load Balancer đã khởi tạo trước đó trên hệ thống vLB. Chúng tôi sẽ tự động tạo 1 NLB trên cluster của bạn. ALB này sẽ hiển thị trên vLB Portal, chi tiết truy cập tại đây Nếu bạn đã có sẵn một Network Load Balancer đã khởi tạo trước đó trên hệ thống vLB và bạn muốn tái sử dụng NLB cho cluster của bạn. Lúc này, bạn hãy nhập thông tin Load Balancer Name vào annotation này.
vks.vngcloud.vn/package-id	Không bắt buộc	<ul style="list-style-type: none"> Nếu bạn không nhập thông tin này thì mặc định chúng tôi sẽ sử dụng cấu hình NLB Small. Nếu bạn đã có sẵn host vLB đang ACTIVE và bạn muốn tích hợp host này vô cụm K8S của bạn, vui lòng bỏ qua trường thông tin này.
vks.vngcloud.vn/tags	Không bắt buộc	<ul style="list-style-type: none"> Tag được gắn thêm cho NLB của bạn.
vks.vngcloud.vn/scheme	Không bắt buộc	<ul style="list-style-type: none"> Mặc định internet-facing, bạn có thể đổi thành internal tùy nhu cầu sử dụng.
vks.vngcloud.vn/security-groups	Không bắt buộc	<ul style="list-style-type: none"> Mặc định sẽ tạo một security group default theo Cluster của bạn.
vks.vngcloud.vn/ inbound-cidrs	Không bắt buộc	<ul style="list-style-type: none"> Mặc định All CIDR: 0.0.0.0/0
vks.vngcloud.vn/healthy-threshold-count	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 3
vks.vngcloud.vn/unhealthy-threshold-count	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 3
vks.vngcloud.vn/healthcheck-interval-seconds	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 30

vks.vngcloud.vn/he althcheck-timeout-seconds	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 5
vks.vngcloud.vn/he althcheck-protocol	Không bắt buộc	<ul style="list-style-type: none"> Mặc định TCP. Người dùng có thể chọn một trong các giá trị TCP/ HTTP/ HTTPS/ PING-UDP
vks.vngcloud.vn/he althcheck-http-method	Không bắt buộc	<ul style="list-style-type: none"> Mặc định GET. Người dùng có thể chọn một trong các giá trị GET / POST / PUT
vks.vngcloud.vn/he althcheck-path	Không bắt buộc	<ul style="list-style-type: none"> Mặc định /
vks.vngcloud.vn/he althcheck-http-version	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 1.0. Người dùng có thể chọn một trong các giá trị 1.0, 1.1
vks.vngcloud.vn/he althcheck-http-domain-name	Không bắt buộc	<ul style="list-style-type: none"> Mặc định trống
vks.vngcloud.vn/he althcheck-port	Không bắt buộc	<ul style="list-style-type: none"> Mặc định traffic port
vks.vngcloud.vn/success-codes	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 200
vks.vngcloud.vn/idle-timeout-client	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 50
vks.vngcloud.vn/idle-timeout-member	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 50
vks.vngcloud.vn/idle-timeout-connection	Không bắt buộc	<ul style="list-style-type: none"> Mặc định 5
vks.vngcloud.vn/pool-algorithm	Không bắt buộc	<ul style="list-style-type: none"> Mặc định ROUND_ROBIN. Người dùng có thể chọn một trong các giá trị ROUND_ROBIN / LEAST_CONNECTIONS / SOURCE_IP
vks.vngcloud.vn/target-node-labels	Không bắt buộc	<ul style="list-style-type: none"> Mặc định trống

vks.vngcloud.vn/enable-proxy-protocol

Không bắt buộc

- Mặc định trống. Người dùng chỉ định danh sách các service name trong Load Balancer mà Proxy Protocol sẽ được áp dụng.

Giới hạn và hạn chế NLB

Giới hạn

Một vài lưu ý về giới hạn của việc integrate a NLB vào một cluster:

- Một NLB có thể được sử dụng chung cho nhiều cluster nhưng phải đảm bảo các cluster này có chung **Subnet**.
 - Một NLB có thể bao gồm nhiều listener, nhiều pool, nhiều policy. Các giới hạn về số lượng listener, số lượng pool, số lượng policy vui lòng tham khảo tại [Hạn mức tài nguyên](#).
-

Hạn chế

- Việc thay đổi tên hoặc kích thước (Rename, Resize) tài nguyên Load Balancer trên vServer Portal có thể gây ra sự không tương thích với tài nguyên trên Kubernetes Cluster. Điều này có thể dẫn đến việc các tài nguyên không hoạt động trên Cluster, hoặc tài nguyên bị đồng bộ lại hoặc thông tin tài nguyên giữa vServer Portal và Cluster không khớp. Để ngăn chặn vấn đề này, hãy sử dụng `kubectl` để quản lý tài nguyên của Cluster.

CNI

Tổng quan

CNI (Container Network Interface) là một bộ công cụ tiêu chuẩn cung cấp khả năng kết nối mạng cho các container trong một cụm Kubernetes. Nói một cách đơn giản, CNI là một lớp trừu tượng, giúp Kubernetes quản lý và cấu hình mạng cho các pod (một tập hợp các container chia sẻ cùng một mạng) một cách linh hoạt và hiệu quả.

CNI hoạt động như thế nào?

Khi bạn tạo một pod mới, Kubernetes sẽ gọi đến CNI để tạo một mạng interface cho pod đó. Plugin CNI sẽ thực hiện các tác vụ sau:

- **Cấp phát địa chỉ IP:** Gán một địa chỉ IP duy nhất cho pod.
- **Cấu hình routing:** Thiết lập các quy tắc routing cho phép giao tiếp giữa các pod,...

Bên cạnh đó, các kết nối hoạt động như sau:

- **Kết nối trong cùng một VPC:** Các node trong cùng một VPC sẽ kết nối trực tiếp với nhau.
- **Kết nối giữa các VPC khác nhau:** Sử dụng VPC Peering để kết nối các node giữa các VPC khác nhau.
- **Kết nối tới hạ tầng bên ngoài:** Sử dụng các giải pháp kết nối mạng như VPN site-to-site hoặc Direct Connect để kết nối từ các node trong VPC tới các hạ tầng bên ngoài (On Cloud, On-premise).

Điều này giúp duy trì một hạ tầng mạng liên tục, linh hoạt và bảo mật trong môi trường multi-cloud hoặc hybrid-cloud.

So sánh giữa các plugin CNI

Hiện tại, VKS đang cung cấp 3 plugin CNI phổ biến là Calico Overlay, Cilium Overlay, Cilium VPC Native Routing. Trong đó:

- **Calico Overlay:** Sử dụng mô hình overlay thông qua tunneling (**IP-in-IP**). Tương thích với nhiều hạ tầng nhưng hiệu suất có thể bị ảnh hưởng bởi **overhead** từ tunnel.
- **Cilium Overlay:** Cũng sử dụng mô hình overlay nhưng có sự tích hợp mạnh mẽ với **eBPF**, giúp cải thiện hiệu suất, bảo mật, và khả năng mở rộng.
- **Cilium VPC Native Routing:** Sử dụng **eBPF** và **không cần overlay**, tận dụng khả năng routing của hạ tầng VPC, mang lại hiệu suất và khả năng mở rộng tốt nhất.

Khi nào nên sử dụng Calico Overlay: đơn giản trong việc sử dụng, không yêu cầu hiệu suất quá cao.

Khi nào nên sử dụng Cilium Overlay: đơn giản trong việc sử dụng, không yêu cầu hiệu suất quá cao tuy nhiên có nhu cầu monitor chuyên sâu (Hubble).

Khi nào nên sử dụng Cilium VPC Native Routing: yêu cầu hiệu suất cao, khả năng **kết nối với các hệ thống bên ngoài** dễ dàng và có nhu cầu monitor chuyên sâu (Hubble).

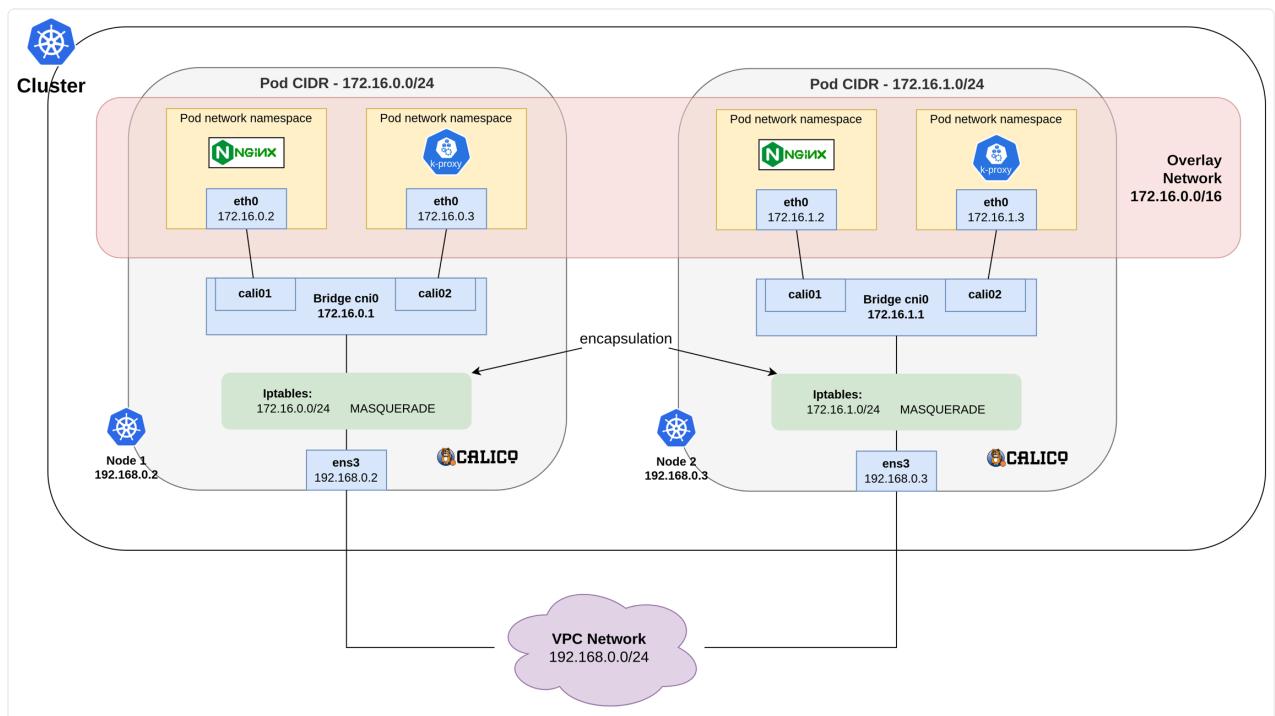
Sử dụng CNI Calico Overlay

Tổng quan

CNI Calico Overlay trong VKS là một loại **overlay network** sử dụng giao thức **IP-in-IP encapsulation** tạo ra một mạng overlay. Điều này cho phép các pod giao tiếp với nhau mà không cần thay đổi cấu hình mạng vật lý bên dưới. Các pod sẽ nhận địa chỉ IP từ dải địa chỉ IP được cấu hình cho Calico, thường là khác với địa chỉ IP của VPC hoặc subnet của bạn.

Model

Trên VKS, **Calico Overlay** hoạt động theo mô hình sau:



Trong đó:

- Các pod trên mỗi node giao tiếp với nhau thông qua **cali interface** và **bridge cni0**.
- Khi các pod cần giao tiếp với các pod trên các node khác, gói tin sẽ được gói (encapsulated) vào trong gói overlay và gửi qua **mạng vật lý (VPC Network)**.
- **Calico** trên mỗi node chịu trách nhiệm thực hiện gói hóa (encapsulation) và giải gói (decapsulation) để các pod có thể liên lạc xuyên qua các node khác nhau.

Điều kiện cần

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn tại [đây](#).
- Có ít nhất 1 **SSH key** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).
- Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.

Khởi tạo một Cluster sử dụng Calico Overlay

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**.

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn. Để sử dụng **Calico Overlay** cho **Cluster** của bạn, vui lòng chọn:

- **Network type: Calico Overlay**

Field	Ý nghĩa	Ví dụ minh họa
VPC	Dải địa chỉ IP mà các node của Cluster sẽ sử dụng để giao tiếp.	Trong hình, chúng tôi lựa chọn VPC có IP range là 10.111.0.0/16 , tương ứng với 65536 IP

Subnet	Dải địa chỉ IP nhỏ hơn thuộc VPC. Mỗi node trong Cluster sẽ được gán một IP từ Subnet này. Subnet phải nằm trong dải IP của VPC đã chọn.	Trong hình, chúng tôi lựa chọn Subnet có Primary IP range là 10.111.0.0/24 , tương ứng với 256 IP
IP-IP encapsulation mode	Chế độ IP-IP encapsulation trong VKS là Always	Trong hình, chúng tôi lựa chọn chế độ Always để luôn encapsulate các gói tin.
CIDR	Dải mạng ảo mà các pod sẽ sử dụng	Trong hình, chúng tôi lựa chọn dải mạng ảo là 172.16.0.0/16 . Các pod sẽ lấy IP từ dải IP này.

The screenshot shows the 'Network setting' section of the VNG Cloud Kubernetes cluster creation interface. It includes the following fields:

- Network type:** Calico Overlay (selected)
- IPIP encapsulation mode:** Always
- CIDR:** 172.16.0.0 /16
- VPC:** (10.111.0.0/16) (selected)
- Subnet:** subnet1 (10.111.0.0/24) (selected)
- Whitelist IP:** 0.0.0.0 /0

On the right side, there are summary sections for 'Cluster configuration' and 'Default Node group configuration'. The 'Cluster configuration' section shows details like Cluster Name: cluster-a7f9b556-471, Kubernetes Version: v1.29.1-vks.1724605200, and Network setting: Public, Calico Overlay, Always. The 'Default Node group configuration' section shows Node group name: nodegroup-29c72 and Number of nodes: 3.

(i) Chú ý:

- Chỉ một loại networktype:** Trong một cluster, bạn chỉ có thể sử dụng một trong ba loại networktype: Calico Overlay, Cilium Overlay, hoặc Cilium VPC Native Routing
- Multiple subnet cho một cluster:** VKS hỗ trợ việc sử dụng nhiều subnet cho một cluster. Điều này cho phép bạn cấu hình mỗi node group trong cluster nằm ở các subnet khác nhau trong cùng một VPC, giúp tối ưu hóa việc phân bổ tài nguyên và quản lý mạng.

Bước 5: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Deploy một Workload

Bên dưới là hướng dẫn triển khai một deployment nginx và kiểm tra việc phân chia IP cho các pod được triển khai trong cluster của bạn.

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/k8s-cluster>

Bước 2: Danh sách Cluster được hiển thị, chọn biểu tượng **Download** và chọn **Download Config File** để thực hiện tải xuống file kubeconfig. File này sẽ giúp bạn có toàn quyền truy cập vào Cluster của bạn.

Bước 3: Đổi tên file này thành config và lưu nó vào thư mục **~/.kube/config**

Bước 4: Thực hiện kiểm tra Cluster thông qua lệnh:

- Chạy câu lệnh sau đây để kiểm tra **node**

```
kubectl get nodes
```

- Nếu kết quả trả về như bên dưới tức là bạn Cluster của bạn được khởi tạo thành công với 5 node:

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

- Tiếp tục thực hiện chạy lệnh sau đây để kiểm tra các **pod** đã được triển khai trên namespace kube-system của bạn:

```
kubectl get pods -A
```

- Nếu kết quả trả về như bên dưới tức là các **pods** hỗ trợ chạy **Calico Overlay** đã running:

NAMESPACE	NAME	READY	STATUS	R
kube-system	calico-kube-controllers-868b574465-wbxlx	1/1	Running	0
kube-system	calico-node-65ql2	1/1	Running	0
kube-system	calico-node-d9hc7	1/1	Running	0
kube-system	calico-node-gp2s7	1/1	Running	0
kube-system	calico-node-hgk86	1/1	Running	0
kube-system	calico-node-vj9ts	1/1	Running	0
kube-system	calico-typa-74d79bf5f6-zzdn9	1/1	Running	0
kube-system	coredns-1727334072-86776977c9-19tcp	1/1	Running	0
kube-system	coredns-1727334072-86776977c9-xqcn9	1/1	Running	0
kube-system	konnectivity-agent-bj7wc	1/1	Running	0
kube-system	konnectivity-agent-fnm7j	1/1	Running	0
kube-system	konnectivity-agent-gvnbl	1/1	Running	0
kube-system	konnectivity-agent-jj764	1/1	Running	0
kube-system	konnectivity-agent-vgmwf	1/1	Running	0
kube-system	kube-proxy-8r85m	1/1	Running	0
kube-system	kube-proxy-bddf5	1/1	Running	0
kube-system	kube-proxy-kwskl	1/1	Running	0
kube-system	kube-proxy-zv6m4	1/1	Running	0
kube-system	kube-proxy-zw65v	1/1	Running	0
kube-system	vngcloud-controller-manager-67cf7f868c-jc66k	1/1	Running	0
kube-system	vngcloud-csi-controller-746b67bcb8-dn5d7	7/7	Running	0
kube-system	vngcloud-csi-controller-746b67bcb8-hqm24	7/7	Running	0
kube-system	vngcloud-csi-node-24nlb	3/3	Running	0
kube-system	vngcloud-csi-node-fgxpath	3/3	Running	0
kube-system	vngcloud-csi-node-q9npf	3/3	Running	0
kube-system	vngcloud-csi-node-tw5sv	3/3	Running	0
kube-system	vngcloud-csi-node-z2sk9	3/3	Running	0
kube-system	vngcloud-ingress-controller-0	1/1	Running	0

Bước 2: Triển khai nginx trên cluster vừa khởi tạo:

- Thực hiện khởi tạo tệp tin **nginx-deployment.yaml** với nội dung tương tự bên dưới:

```
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
```

- Thực hiện triển khai deployment này qua lệnh:

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

Bước 3: Kiểm tra các pod nginx đã được triển khai và địa chỉ IP được gán cho mỗi pod

- Thực hiện kiểm tra các **pod** qua lệnh:

```
kubectl get pods -o wide
```

- Bạn có thể quan sát bên dưới, các **pod nginx** được gán các IP 172.16.x.x thỏa mãn điều kiện **Calico CIDR 172.16.0.0/16** mà chúng tôi đã chỉ định bên trên:

NAME	READY	STATUS	RESTARTS	AGE	IP
nginx-app-7c79c4bf97-2xbwd	1/1	Running	0	49s	172.16.197.136
nginx-app-7c79c4bf97-5hcds	1/1	Running	0	49s	172.16.197.137
nginx-app-7c79c4bf97-5hgwp	1/1	Running	0	49s	172.16.197.138
nginx-app-7c79c4bf97-5l79h	1/1	Running	0	49s	172.16.83.4
nginx-app-7c79c4bf97-5q2f4	1/1	Running	0	49s	172.16.226.196
nginx-app-7c79c4bf97-5szc6	1/1	Running	0	49s	172.16.83.6
nginx-app-7c79c4bf97-9272q	1/1	Running	0	49s	172.16.167.71
nginx-app-7c79c4bf97-cgwrj	1/1	Running	0	49s	172.16.67.195
nginx-app-7c79c4bf97-fhlg4	1/1	Running	0	49s	172.16.167.70
nginx-app-7c79c4bf97-fj865	1/1	Running	0	49s	172.16.83.3
nginx-app-7c79c4bf97-gh6hj	1/1	Running	0	49s	172.16.167.69
nginx-app-7c79c4bf97-hx2rn	1/1	Running	0	49s	172.16.83.5
nginx-app-7c79c4bf97-jv26j	1/1	Running	0	49s	172.16.167.68
nginx-app-7c79c4bf97-km7p4	1/1	Running	0	49s	172.16.226.198
nginx-app-7c79c4bf97-lrh2r	1/1	Running	0	49s	172.16.167.67
nginx-app-7c79c4bf97-lvj6g	1/1	Running	0	49s	172.16.67.196
nginx-app-7c79c4bf97-nhhdk	1/1	Running	0	49s	172.16.226.197
nginx-app-7c79c4bf97-qr2lm	1/1	Running	0	49s	172.16.67.198
nginx-app-7c79c4bf97-x4ztb	1/1	Running	0	49s	172.16.226.195
nginx-app-7c79c4bf97-xrqwx	1/1	Running	0	49s	172.16.67.197

- Bạn cũng có thể thực hiện xem mô tả chi tiết mỗi pod để kiểm tra thông tin pod này qua lệnh:

```
kubectl describe pod nginx-app-7c79c4bf97-2xbwd
```

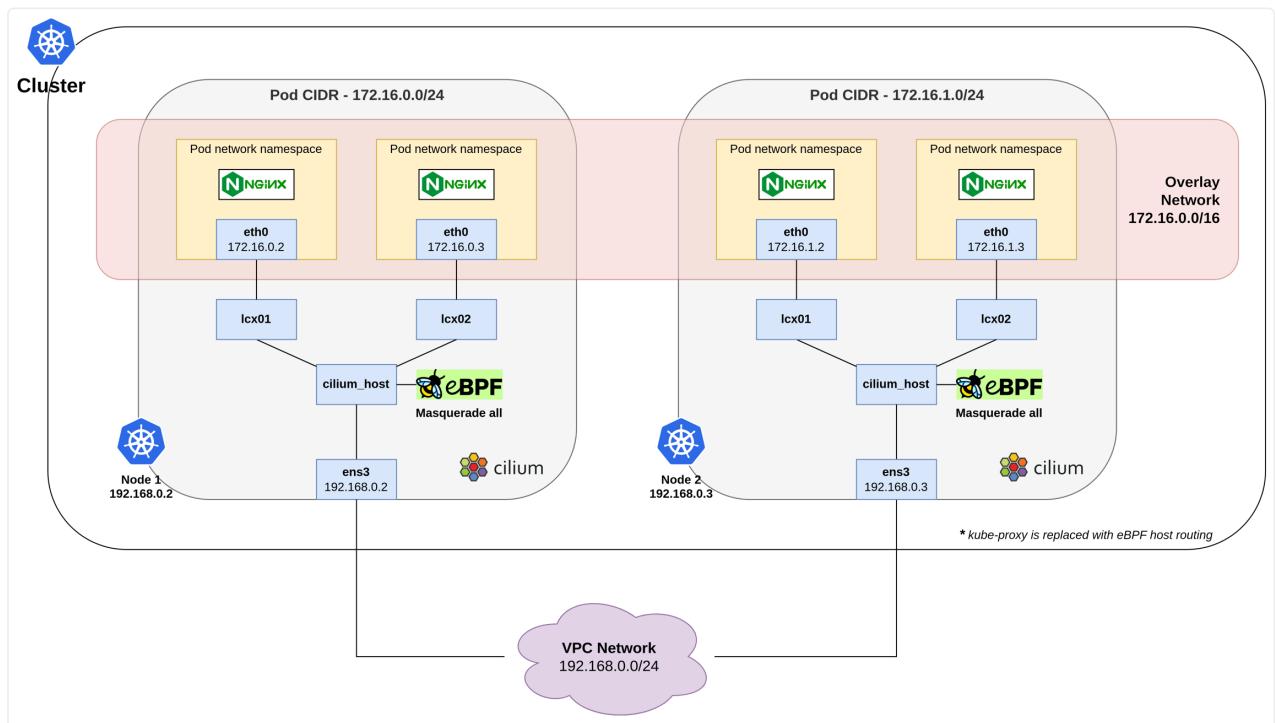
Sử dụng CNI Cilium Overlay

Tổng quan

CNI Cilium Overlay trong VKS là một loại **overlay network** sử dụng **eBPF (extended Berkeley Packet Filter)** để tăng cường hiệu suất và bảo mật mạng. Nếu so sánh với các giải pháp như **Calico Overlay**, Cilium mang lại hiệu suất cao hơn nhờ khả năng xử lý lưu lượng trực tiếp trong kernel bằng eBPF. Ngoài ra, với Calico thường sử dụng **iptables** để quản lý lưu lượng, trong khi Cilium với eBPF có thể xử lý các chính sách mạng và các hành vi ứng dụng cụ thể (Layer 7).

Model

Trên VKS, **Cilium Overlay** hoạt động theo mô hình sau:



Trong đó:

- **Pod (eth0) → Ixc01/Ixc02:** Các Pod giao tiếp thông qua mạng ảo được tạo bởi Cilium.

- **Ixc01/Ixc02 → cilium_host:** Các gói tin từ các Pod được chuyển đến `cilium_host`, là lớp trung gian giữa mạng của Pod và mạng vật lý.
 - **cilium_host → ens3:** Sau khi được xử lý bởi Cilium (và eBPF), các gói tin được gửi đến mạng vật lý thông qua `ens3`.
 - **ens3 → VPC Network:** Cuối cùng, các gói tin được truyền qua mạng vật lý để đến các node khác hoặc ra khỏi cluster.
-

Điều kiện cần

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn tại [đây](#).
 - Có ít nhất 1 **SSH key** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).
 - Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.
-

Khởi tạo một Cluster sử dụng Cilium Overlay

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**.

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn. Để sử dụng **Cilium Overlay** cho **Cluster** của bạn, vui lòng chọn:

- **Network type:** Cilium Overlay

Field	Ý nghĩa	Ví dụ minh họa
VPC	Dải địa chỉ IP mà các node của Cluster sẽ sử dụng để giao tiếp.	Trong hình, chúng tôi lựa chọn VPC có IP range là 10.111.0.0/16 , tương ứng với 65536 IP
Subnet	Dải địa chỉ IP nhỏ hơn thuộc VPC. Mỗi node trong Cluster sẽ được gán một IP từ Subnet này. Subnet phải nằm trong dải IP của VPC đã chọn.	Trong hình, chúng tôi lựa chọn Subnet có Primary IP range là 10.111.0.0/24 , tương ứng với 256 IP
IP-IP encapsulation mode	Chế độ IP-IP encapsulation trong VKS là Always	Trong hình, chúng tôi lựa chọn chế độ Always để luôn encapsulate các gói tin.
CIDR	Dải mạng ảo mà các pod sẽ sử dụng	Trong hình, chúng tôi lựa chọn dải mạng ảo là 172.16.0.0/16 . Các pod sẽ lấy IP từ dải IP này.

The screenshot shows the VNG Cloud interface for creating a Kubernetes cluster. The 'Network setting' section is highlighted with a red box. It includes fields for Network type (Cilium Overlay), IP-IP encapsulation mode (Always), CIDR (172.16.0.0/16), VPC (10.111.0.0/16), and Subnet (subnet1 (10.111.0.0/24)). Below this, a 'Whitelist IP' field contains '0.0.0.0 /0'. To the right, the 'Summary' tab displays cluster configuration details like Cluster Name (cluster-02), Kubernetes Version (v1.29.1-vks1724605200), and Default Node group configuration (nodegroup-7fb09). A total cost of 4,356,000 VND is shown at the bottom.

(i) Chú ý:

- Chỉ một loại networktype:** Trong một cluster, bạn chỉ có thể sử dụng một trong ba loại networktype: Calico Overlay, Cilium Overlay, hoặc Cilium VPC Native Routing

- **Multiple subnet cho một cluster:** VKS hỗ trợ việc sử dụng nhiều subnet cho một cluster. Điều này cho phép bạn cấu hình mỗi node group trong cluster nằm ở các subnet khác nhau trong cùng một VPC, giúp tối ưu hóa việc phân bổ tài nguyên và quản lý mạng.

Bước 5: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Deploy một Workload

Bên dưới là hướng dẫn triển khai một deployment nginx và kiểm tra việc phân chia IP cho các pod được triển khai trong cluster của bạn.

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/k8s-cluster>

Bước 2: Danh sách Cluster được hiển thị, chọn biểu tượng **Download** và chọn **Download Config File** để thực hiện tải xuống file kubeconfig. File này sẽ giúp bạn có toàn quyền truy cập vào Cluster của bạn.

Bước 3: Đổi tên file này thành config và lưu nó vào thư mục **~/.kube/config**

Bước 4: Thực hiện kiểm tra Cluster thông qua lệnh:

- Chạy câu lệnh sau đây để kiểm tra **node**

```
kubectl get nodes
```

- Nếu kết quả trả về như bên dưới tức là bạn Cluster của bạn được khởi tạo thành công với 3 node:

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

- Tiếp tục thực hiện chạy lệnh sau đây để kiểm tra các **pod** đã được triển khai trên namespace kube-system của bạn:

```
kubectl get pods -A
```

- Nếu kết quả trả về như bên dưới tức là các **pods** hỗ trợ chạy **Cilium Overlay** đã được running:

NAMESPACE	NAME	READY	STATUS	R
kube-system	cilium-8xtwz	1/1	Running	1
kube-system	cilium-cpxvv	1/1	Running	0
kube-system	cilium-envoy-b95pg	1/1	Running	0
kube-system	cilium-envoy-dx8qg	1/1	Running	0
kube-system	cilium-envoy-sqdn8	1/1	Running	0
kube-system	cilium-operator-75b8c6f6d4-7x4f6	1/1	Running	0
kube-system	cilium-operator-75b8c6f6d4-k7j45	1/1	Running	0
kube-system	cilium-zs2cm	1/1	Running	1
kube-system	coredns-1727408780-5fcf89468-7hmvp	1/1	Running	0
kube-system	coredns-1727408780-5fcf89468-v9nbd	1/1	Running	0
kube-system	hubble-relay-8899f8cdc-976zf	1/1	Running	1
kube-system	hubble-ui-574c5bb99b-gg7jx	2/2	Running	0
kube-system	konnectivity-agent-46nvd	1/1	Running	0
kube-system	konnectivity-agent-qhq4m	1/1	Running	0
kube-system	konnectivity-agent-xs7bq	1/1	Running	0
kube-system	vngcloud-controller-manager-7c47d64584-z8827	1/1	Running	0
kube-system	vngcloud-csi-controller-848f68f46-2hkxl	7/7	Running	2
kube-system	vngcloud-csi-controller-848f68f46-bkvkg	7/7	Running	2
kube-system	vngcloud-csi-node-8rxbx	3/3	Running	2
kube-system	vngcloud-csi-node-mxknq	3/3	Running	3
kube-system	vngcloud-csi-node-tfrsp	3/3	Running	2
kube-system	vngcloud-ingress-controller-0	1/1	Running	1

Bước 2: Triển khai nginx trên cluster vừa khởi tạo:

- Thực hiện khởi tạo tệp tin **nginx-deployment.yaml** với nội dung tương tự bên dưới:

```
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
```

- Thực hiện triển khai deployment này qua lệnh:

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

Bước 3: Kiểm tra các pod nginx đã được triển khai và địa chỉ IP được gán cho mỗi pod

- Thực hiện kiểm tra các pod qua lệnh:

```
kubectl get pods -o wide
```

- Bạn có thể quan sát bên dưới, các **pod nginx** được gán các IP 172.16.x.x thỏa mãn điều kiện **Cilium CIDR 172.16.0.0/16** mà chúng tôi đã chỉ định bên trên:

NAME	READY	STATUS	RESTARTS	AGE	IP	N
nginx-app-7c79c4bf97-4lcbn	1/1	Running	0	83s	172.16.0.6	v
nginx-app-7c79c4bf97-669z9	1/1	Running	0	83s	172.16.1.115	v
nginx-app-7c79c4bf97-7hqp5	1/1	Running	0	83s	172.16.1.32	v
nginx-app-7c79c4bf97-8fjhm	1/1	Running	0	83s	172.16.2.51	v
nginx-app-7c79c4bf97-8xmfm	1/1	Running	0	83s	172.16.0.100	v
nginx-app-7c79c4bf97-9b4px	1/1	Running	0	83s	172.16.2.19	v
nginx-app-7c79c4bf97-b7vlg	1/1	Running	0	83s	172.16.1.128	v
nginx-app-7c79c4bf97-bc6r4	1/1	Running	0	83s	172.16.0.160	v
nginx-app-7c79c4bf97-f1kz5	1/1	Running	0	83s	172.16.2.253	v
nginx-app-7c79c4bf97-k55j6	1/1	Running	0	83s	172.16.2.76	v
nginx-app-7c79c4bf97-19p8p	1/1	Running	0	83s	172.16.2.187	v
nginx-app-7c79c4bf97-1lnfq	1/1	Running	0	83s	172.16.1.30	v
nginx-app-7c79c4bf97-mg9t8	1/1	Running	0	83s	172.16.0.221	v
nginx-app-7c79c4bf97-mlh7g	1/1	Running	0	83s	172.16.2.191	v
nginx-app-7c79c4bf97-n946h	1/1	Running	0	83s	172.16.1.82	v
nginx-app-7c79c4bf97-p9k42	1/1	Running	0	83s	172.16.0.112	v
nginx-app-7c79c4bf97-s14b8	1/1	Running	0	83s	172.16.1.22	v
nginx-app-7c79c4bf97-tdtjc	1/1	Running	0	83s	172.16.2.109	v
nginx-app-7c79c4bf97-zwxps	1/1	Running	0	83s	172.16.2.209	v
nginx-app-7c79c4bf97-zxx87	1/1	Running	0	83s	172.16.0.212	v

- Bạn cũng có thể thực hiện xem mô tả chi tiết mỗi pod để kiểm tra thông tin pod này qua lệnh:

```
kubectl describe pod nginx-app-7c79c4bf97-4lcbn
```

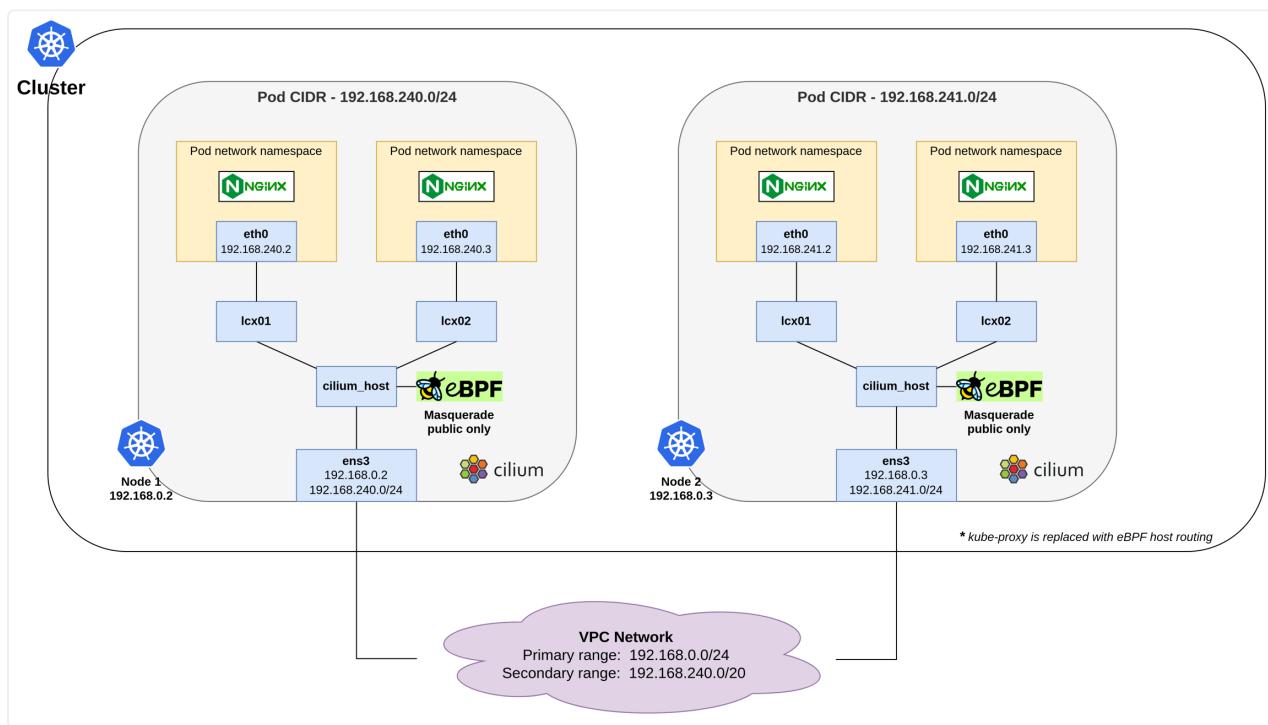
Sử dụng CNI Cilium VPC Native Routing

Tổng quan

CNI (Container Network Interface) Cilium VPC Native Routing là một cơ chế giúp Kubernetes quản lý mạng mà không cần sử dụng overlay networks. Thay vì dùng các lớp mạng ảo, **CNI Cilium VPC Native Routing** tận dụng khả năng routing trực tiếp từ VPC (Virtual Private Cloud) của các nhà cung cấp dịch vụ đám mây để tối ưu hóa việc truyền dữ liệu giữa các node và pod trong cụm Kubernetes.

Model

Trên VKS, **CNI (Container Network Interface) Cilium VPC Native Routing** hoạt động theo mô hình sau:



Trong đó:

- Mỗi **Node** có một dải địa chỉ IP riêng cho các pod (Pod CIDR). Các pod trong mỗi node sử dụng địa chỉ từ CIDR này và giao tiếp qua mạng ảo.

- **Cilium** và **eBPF** thực hiện quản lý mạng cho tất cả các pod trên mỗi node, bao gồm việc xử lý lưu lượng đi từ pod này đến pod khác, hoặc từ node này sang node khác. Khi cần, eBPF thực hiện **masquerading** để ẩn địa chỉ IP nội bộ của pod khi giao tiếp với mạng ngoài.
 - **Cilium** đảm bảo rằng các pod có thể giao tiếp với nhau cả bên trong cùng node và giữa các node khác nhau.
-

Điều kiện cần

Để có thể khởi tạo một **Cluster** và **Deploy** một **Workload**, bạn cần:

- Có ít nhất 1 **VPC** và 1 **Subnet** đang ở trạng thái **ACTIVE**. Nếu bạn chưa có VPC, Subnet nào, vui lòng khởi tạo VPC, Subnet theo hướng dẫn bên dưới:
 - **Bước 1:** Truy cập vào trang chủ vServer tại link <https://hcm-3.console.vngcloud.vn/vserver>
 - **Bước 2:** Chọn menu **VPCs** ở menu bên trái màn hình.
 - **Bước 3:** Tại đây, nếu bạn chưa có VPC nào, vui lòng chọn **Create VPC** bằng cách nhập VPC name và định nghĩa dãy **CIDR/16** mong muốn.
 - **Bước 4:** Sau khi đã có ít nhất 1 VPC, để tạo subnet, bạn cần chọn **View Detail** để mở rộng bảng điều khiển ở phía dưới, trong đó có mục **Subnet**.
 - **Bước 5:** Tại mục **Subnet**, chọn **Add Subnet**. Lúc này, bạn cần nhập:
 - **Subnet name:** tên gợi nhớ của subnet
 - **Primary CIDR:** Đây là dải địa chỉ IP chính của subnet. Mọi địa chỉ IP nội bộ của các máy ảo (VM) trong subnet này sẽ được lấy từ dải địa chỉ này. Giả sử, nếu bạn đặt Primary CIDR là 10.1.0.0/24, các địa chỉ IP của các VM sẽ nằm trong khoảng từ 10.1.0.1 đến 10.1.0.254.
 - **Secondary CIDR:** Đây là dải địa chỉ IP phụ, được sử dụng để cung cấp thêm địa chỉ IP hoặc để phân chia các dịch vụ khác nhau trong cùng một subnet. Mỗi Node có một dải địa chỉ IP riêng cho các pod (Pod CIDR). Các pod trong mỗi node sử dụng địa chỉ từ CIDR này và giao tiếp qua mạng ảo.

The screenshot shows the VNG Cloud interface for managing VPCs. On the left sidebar, under the 'Network' section, 'VPCs' is selected. In the main content area, there is a table of existing VPCs with columns for Name, Status, CIDR, DHCP, Route Table, Created at, and Action. A red arrow points to the 'net-2ade157b-c5cf-42a8-9639-1aac1c61321d' row. Below the table, a sub-section titled 'List of Subnets added to this VPC.' shows two subnets: 'subnet1' and 'subnet2'. A red arrow points to the '+ Add subnet' button.

This screenshot shows the 'Create subnet' dialog box overlaid on the VPC management interface. The dialog 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 '10.111.160.0/20' and 'cidr-2' with CIDR '10.111.128.0/20'. Both entries have a red circle with a minus sign next to them, indicating they are currently disabled or invalid. A red box highlights the 'Secondary CIDR' section. The background shows the list of subnets with the same two subnets from the previous screenshot.



Chú ý:

- Các dải địa chỉ IP của **Primary CIDR** và **Secondary CIDR** không được trùng lặp. Điều này có nghĩa là dải địa chỉ của **Secondary CIDR** phải nằm ngoài phạm vi của **Primary CIDR** và ngược lại. Giả sử, nếu Primary CIDR là 10.1.0.0/24, thì Secondary CIDR không thể là 10.1.0.0/20 vì nó nằm trong phạm vi của Primary CIDR. Thay vào đó, bạn có thể sử dụng một dải địa chỉ khác như 10.1.16.0/20.
- Có ít nhất 1 SSH key đang ở trạng thái **ACTIVE**. Nếu bạn chưa có SSH key nào, vui lòng khởi tạo SSH key theo hướng dẫn tại [đây](#).

- Đã cài đặt và cấu hình **kubectl** trên thiết bị của bạn. vui lòng tham khảo tại [đây](#) nếu bạn chưa rõ cách cài đặt và sử dụng kubectl. Ngoài ra, bạn không nên sử dụng phiên bản kubectl quá cũ, chúng tôi khuyến cáo bạn nên sử dụng phiên bản kubectl sai lệch không quá một phiên bản với version của cluster.

(i) Chú ý:

- Khi bạn sử dụng loại network Cilium Native Routing cho cluster của bạn, bạn cần thực hiện cấu hình các **Security Groups** cho phép các kết nối cần thiết. Ví dụ, khi bạn chạy một NGINX pod trên một node, bạn phải cho phép traffic trên port 80 để đảm bảo các requests từ các node khác có thể kết nối tới. Việc cấu hình này chỉ bắt buộc nếu bạn sử dụng Cilium Native Routing, đối với Calico Overlay và Cilium Overlay thì việc cấu hình Security Groups này là không cần thiết.

Khởi tạo một Cluster sử dụng CNI Cilium VPC Native Routing

Để khởi tạo một Cluster, hãy làm theo các bước bên dưới:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn **Activate**.

Bước 3: Chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn. Sau khi Activate thành công, bạn hãy chọn **Create a Cluster**.

Bước 4: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group** cho bạn. Để sử dụng **CNI Cilium VPC Native Routing** cho **Cluster** của bạn, vui lòng chọn:

- Network type:** Cilium VPC Native Routing và các thông số khác như sau:

Field	Ý nghĩa	Ví dụ minh họa
VPC	Dải địa chỉ IP mà các node của Cluster sẽ sử dụng để giao tiếp.	Trong hình, chúng tôi lựa chọn VPC có IP range là

		10.111.0.0/16 , tương ứng với 65536 IP
Subnet	Dải địa chỉ IP nhỏ hơn thuộc VPC. Mỗi node trong Cluster sẽ được gán một IP từ Subnet này. Subnet phải nằm trong dải IP của VPC đã chọn.	Trong hình, chúng tôi lựa chọn Subnet có Primary IP range là 10.111.0.0/24 , tương ứng với 256 IP
Default Pod IP range	Đây là dải địa chỉ IP thứ cấp được sử dụng cho các pod. Nó được gọi là Secondary IP range vì nó không trùng với dải IP chính của node (Primary IP range). Các pod trong Cluster sẽ được gán IP từ dải này.	Trong hình, chúng tôi lựa chọn Secondary IP range là 10.111.160.0/20 - Tương ứng với 4096 IP cho các pod
Node CIDR mask size	Kích thước của CIDR dành cho các node. Thông số này cho biết mỗi node sẽ được gán bao nhiêu địa chỉ IP từ dải pod IP range. Kích thước này cần được chọn sao cho đảm bảo có đủ địa chỉ IP cho tất cả các pod trên mỗi node. Bạn có thể tham khảo bảng bên dưới để hiểu các tính số lượng IP có thể sử dụng để cấp phát cho node, pod trong cluster của bạn.	Trong hình, chúng tôi lựa chọn Node CIDR mask size là /25 - Mỗi node sẽ có 128 địa chỉ IP , phù hợp với số lượng pod bạn mong muốn chạy trên một node.

Các tính toán số lượng IP cho pod và node:

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 dải Secondary IP range:

$$\text{Số node có thể tạo} = \frac{\text{Số IP trong dải Secondary IP range}}{\text{Số IP mỗi node có thể sử dụng từ Node CIDR}}$$

3. Số lượng pod thực tế có thể tạo trên mỗi node:

$$\text{Số lượng pod thực tế có thể tạo} = \frac{\text{Số lượng IP khả dụng trên mỗi node}}{2}$$

Giả sử, khi khởi tạo cluster, tôi lựa chọn:

- **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:** Các giá trị có thể chọn từ **/24** đến **/26**.

Node CIDR mask size	Số lượng IP cho mỗi node	Số lượng node có thể tạo trong dài /20 (4096 IP)	Số lượng IP phân bổ cho pod trên mỗi node	Số lượng pod thực tế có thể tạo
/24	256	16	256	128
/25	128	32	128	64
/26	64	64	64	32

Chú ý:

- Chỉ một loại networktype:** Trong một cluster, bạn chỉ có thể sử dụng một trong ba loại networktype: Calico Overlay, Cilium Overlay, hoặc Cilium VPC Native Routing
- Multiple subnet cho một cluster:** VKS hỗ trợ việc sử dụng nhiều subnet cho một cluster. Điều này cho phép bạn cấu hình mỗi node group trong cluster nằm ở các subnet khác nhau trong cùng một VPC, giúp tối ưu hóa việc phân bổ tài nguyên và quản lý mạng.
- Cilium VPC Native Routing và Secondary IP Range:** Khi sử dụng Cilium VPC Native Routing cho một cluster, bạn có thể sử dụng nhiều Secondary IP Range. Tuy nhiên, mỗi Secondary IP Range chỉ có thể được sử dụng bởi một cluster duy nhất. Điều này giúp tránh xung đột địa chỉ IP và đảm bảo tính nhất quán trong quản lý mạng.
- Khi không đủ địa chỉ IP trong **Node CIDR range** hoặc **Secondary IP range** để tạo thêm node, cụ thể:

- Nếu bạn **không thể sử dụng Node mới do** hết dải địa chỉ IP trong **Secondary IP range**. Lúc này, các node mới vẫn sẽ được tạo và được join vào cụm nhưng bạn không thể sử dụng chúng. Các pod được yêu cầu khởi chạy trên node mới này sẽ bị kẹt trong trạng thái "**ContainerCreating**" do không thể tìm thấy node phù hợp để triển khai. Lúc này, bạn cần tạo node group mới với secondary range IP chưa được sử dụng trên cluster nào.

Bước 5: Chọn **Create Kubernetes cluster**. Hãy chờ vài phút để chúng tôi khởi tạo Cluster của bạn, trạng thái của Cluster lúc này là **Creating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, bạn có thể xem thông tin Cluster, thông tin Node Group bằng cách chọn vào Cluster Name tại cột **Name**.

Deploy một Workload

Bên dưới là hướng dẫn triển khai một deployment nginx và kiểm tra việc phân chia IP cho các pod được triển khai trong cluster của bạn.

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/k8s-cluster>

Bước 2: Danh sách Cluster được hiển thị, chọn biểu tượng **Download** và chọn **Download Config File** để thực hiện tải xuống file kubeconfig. File này sẽ giúp bạn có toàn quyền truy cập vào Cluster của bạn.

Bước 3: Đổi tên file này thành config và lưu nó vào thư mục **~/.kube/config**

Bước 4: Thực hiện kiểm tra Cluster thông qua lệnh:

- Chạy câu lệnh sau đây để kiểm tra **node**

```
kubectl get nodes
```

- Nếu kết quả trả về như bên dưới tức là bạn Cluster của bạn được khởi tạo thành công với 3 node:

NAME	STATUS	ROLES	AGE	VERSION
vks-cluster-democilium-nodegroup-558f4-39206	Ready	<none>	5m35s	v1.28.8
vks-cluster-democilium-nodegroup-558f4-63344	Ready	<none>	5m45s	v1.28.8
vks-cluster-democilium-nodegroup-558f4-e6e4d	Ready	<none>	6m24s	v1.28.8

- Tiếp tục thực hiện chạy lệnh sau đây để kiểm tra các **pod** đã được triển khai trên namespace `kube-system` của bạn:

```
kubectl get pods -A
```

- Nếu kết quả trả về như bên dưới tức là trạng thái các **pod**s hỗ trợ chạy Cilium VPC Native đều thành công (**Running**):

NAMESPACE	NAME	READY	STATUS	RE
kube-system	cilium-envoy-2g221	1/1	Running	0
kube-system	cilium-envoy-h9mjb	1/1	Running	0
kube-system	cilium-envoy-ngz89	1/1	Running	0
kube-system	cilium-ft98g	1/1	Running	1
kube-system	cilium-operator-5fc5c56c4c-6616d	1/1	Running	0
kube-system	cilium-operator-5fc5c56c4c-qfnw2	1/1	Running	0
kube-system	cilium-rfrr7	1/1	Running	1
kube-system	cilium-xmlq5	1/1	Running	1
kube-system	coredns-1727334052-85db76748b-fpmfr	1/1	Running	0
kube-system	coredns-1727334052-85db76748b-jqv79	1/1	Running	0
kube-system	hubble-relay-8578649fdb-bgzzz	1/1	Running	1
kube-system	hubble-ui-574c5bb99b-g7l6c	2/2	Running	0
kube-system	konnectivity-agent-hmf2x	1/1	Running	0
kube-system	konnectivity-agent-q69n2	1/1	Running	0
kube-system	konnectivity-agent-wgqbw	1/1	Running	0
kube-system	vngcloud-controller-manager-d4d4f7b84-m65nb	1/1	Running	0
kube-system	vngcloud-csi-controller-565c55dbcc-88pt4	7/7	Running	8
kube-system	vngcloud-csi-controller-565c55dbcc-v22q4	7/7	Running	8
kube-system	vngcloud-csi-node-665r2	3/3	Running	3
kube-system	vngcloud-csi-node-8x542	3/3	Running	3
kube-system	vngcloud-csi-node-gx7zd	3/3	Running	2
kube-system	vngcloud-ingress-controller-0	1/1	Running	1

Bước 2: Triển khai nginx trên cluster vừa khởi tạo:

- Thực hiện khởi tạo tệp tin **nginx-deployment.yaml** với nội dung tương tự bên dưới:

```
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
```

- Thực hiện triển khai deployment này qua lệnh:

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

Bước 3: Kiểm tra các pod nginx đã được triển khai và địa chỉ IP được gán cho mỗi pod

- Thực hiện kiểm tra các pod qua lệnh:

```
kubectl get pods -o wide
```

- Bạn có thể quan sát bên dưới, các **pod nginx** được gán các IP 10.111.16x.x thỏa mãn điều kiện **Secondary IP range** và **Node CIDR mask size** mà chúng tôi đã chỉ định bên trên:

NAME	READY	STATUS	RESTARTS	AGE	IP
nginx-app-7c79c4bf97-6v88s	1/1	Running	0	31s	10.111.161.53
nginx-app-7c79c4bf97-754m7	1/1	Running	0	31s	10.111.161.1
nginx-app-7c79c4bf97-9tjw7	1/1	Running	0	31s	10.111.160.212
nginx-app-7c79c4bf97-c6vx7	1/1	Running	0	31s	10.111.160.46
nginx-app-7c79c4bf97-c7nch	1/1	Running	0	31s	10.111.161.3
nginx-app-7c79c4bf97-cggfq	1/1	Running	0	31s	10.111.161.74
nginx-app-7c79c4bf97-cz4xc	1/1	Running	0	31s	10.111.160.115
nginx-app-7c79c4bf97-d84rb	1/1	Running	0	31s	10.111.160.152
nginx-app-7c79c4bf97-dbmt7	1/1	Running	0	31s	10.111.160.184
nginx-app-7c79c4bf97-gtx8b	1/1	Running	0	31s	10.111.161.57
nginx-app-7c79c4bf97-km7tx	1/1	Running	0	31s	10.111.160.94
nginx-app-7c79c4bf97-lmk7c	1/1	Running	0	31s	10.111.161.26
nginx-app-7c79c4bf97-mc24h	1/1	Running	0	31s	10.111.160.98
nginx-app-7c79c4bf97-n4zvf	1/1	Running	0	31s	10.111.160.204
nginx-app-7c79c4bf97-n84tc	1/1	Running	0	31s	10.111.161.106
nginx-app-7c79c4bf97-qtjjx	1/1	Running	0	31s	10.111.160.32
nginx-app-7c79c4bf97-rp4bt	1/1	Running	0	31s	10.111.160.202
nginx-app-7c79c4bf97-sk7tf	1/1	Running	0	31s	10.111.160.196
nginx-app-7c79c4bf97-x8jxm	1/1	Running	0	31s	10.111.160.135
nginx-app-7c79c4bf97-zlstg	1/1	Running	0	31s	10.111.160.121

- Bạn cũng có thể thực hiện xem mô tả chi tiết mỗi pod để kiểm tra thông tin pod này qua lệnh:

```
kubectl describe pod nginx-app-7c79c4bf97-6v88s
```

Bước 4: Bạn có thể thực hiện một vài bước để kiểm tra chuyên sâu việc hoạt động của Cilium. Cụ thể:

- Đầu tiên, bạn cần cài đặt Cilium CLI theo hướng dẫn tại [đây](#).
- Sau khi cài đặt Cilium CLI, thực hiện kiểm tra **trạng thái** của Cilium trong cluster của bạn qua lệnh:

```
cilium status wait
```

- Nếu kết quả hiển thị như bên dưới tức là **Cilium** đang **hoạt động đúng và đầy đủ**:

```

  /--\
 /--\_\_/\--\ Cilium:          OK
 \_\_/\--\_\_/\ Operator:        OK
 /--\_\_/\--\ Envoy DaemonSet: OK
 \_\_/\--\_\_/\ Hubble Relay:   OK
 \_\_/\             ClusterMesh:    disabled

DaemonSet           cilium-envoy      Desired: 3, Ready: 3/3, Available: 3/3
Deployment          hubble-relay     Desired: 1, Ready: 1/1, Available: 1/1
Deployment          hubble-ui       Desired: 1, Ready: 1/1, Available: 1/1
Deployment          cilium-operator Desired: 2, Ready: 2/2, Available: 2/2
DaemonSet           cilium          Desired: 3, Ready: 3/3, Available: 3/3
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/cilium/c
  cilium-envoy    vcr.vngcloud.vn/81-vks-public/cilium/c
  hubble-relay    vcr.vngcloud.vn/81-vks-public/cilium/h
  hubble-ui       vcr.vngcloud.vn/81-vks-public/cilium/h
  hubble-ui       vcr.vngcloud.vn/81-vks-public/cilium/h
  cilium-operator vcr.vngcloud.vn/81-vks-public/cilium/o

```

Bước 5: Bạn có thể thực hiện healthy check kiểm tra Cilium trong cluster của bạn

- Chạy lệnh sau để thực hiện healthy check:

```
kubectl -n kube-system exec ds/cilium -- cilium-health status --probe
```

- Kết quả healthy check mong muốn sẽ như sau:

```

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

```

Ngoài ra, bạn cũng có thể thực hiện thêm các bài kiểm tra kết nối End-to-End hoặc kiểm tra Network performance theo hướng dẫn tại [End-To-End Connectivity Testing](#) hoặc [Network Performance Test](#).

Bước 6: Kiểm tra kết nối giữa các Pod

- Thực hiện kiểm tra kết nối giữa các pod, đảm bảo rằng các **pod có thể giao tiếp qua địa chỉ IP của VPC mà không cần qua overlay networks**. Ví dụ bên dưới tôi thực hiện ping từ pod nginx-app-7c79c4bf97-6v88s có địa chỉ IP: 10.111.161.53 tới một server trong cùng VPC có địa chỉ IP: 10.111.0.10:

```
kubectl exec -it nginx-app-7c79c4bf97-6v88s -- ping 10.111.0.10
```

- Nếu kết quả như sau tức là kết nối đã thông:

```
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

Làm việc với Container Storage Interface (CSI)

CSI là gì?

- Container Storage Interface (CSI) là một giao diện tiêu chuẩn cho phép các container tương tác với các hệ thống lưu trữ khác nhau. CSI cung cấp một tập hợp các API chung mà các container có thể sử dụng để truy cập và quản lý dữ liệu, bất kể hệ thống lưu trữ cơ bản là gì.

Integrate with Container Storage Interface (CSI)

Để integrate CSI với Kubernetes cluser, hãy làm theo các bước sau đây:

Chuẩn bị

- Tạo một Kubernetes cluster trên VNGCloud, hoặc sử dụng một cluster đã có. Lưu ý: đảm bảo bạn đã tải xuống cluster configuration file sau khi cluster được khởi tạo thành công và truy cập vào cluster của bạn.

Khởi tạo Service Account và cài đặt VNGCloud BlockStorage CSI Driver

Chú ý:

- Khi bạn thực hiện khởi tạo Cluster theo hướng dẫn bên trên, nếu bạn chưa bật option **Enable BlockStore Persistent Disk CSI Driver**, mặc định chúng tôi sẽ không cài sẵn plugin này vào Cluster của bạn. Bạn cần tự thực hiện Khởi tạo Service Account và cài đặt VNGCloud BlockStorage CSI Driver theo hướng dẫn bên dưới. Nếu bạn đã bật option **Enable BlockStore Persistent Disk CSI Driver**, thì chúng tôi đã cài sẵn plugin này vào Cluster của bạn, hãy bỏ qua bước Khởi tạo Service Account, cài đặt VNGCloud BlockStorage CSI Driver và tiếp tục thực hiện theo hướng dẫn kể từ Deploy một Workload.
- VNGCloud BlockStorage CSI Driver** chỉ hỗ trợ attach volume với một node (VM) duy nhất trong suốt vòng đời của volume đó. Nếu bạn có nhu cầu ReadWriteMany, bạn có thể cân nhắc sử dụng NFS CSI Driver, vì nó cho phép nhiều nodes có thể Read và Write trên cùng một volume cùng một lúc. Điều này rất hữu ích cho các ứng dụng cần chia sẻ dữ liệu giữa nhiều pods hoặc services trong Kubernetes.

- Khởi tạo Service Account và cài đặt VNGCloud BlockStorage CSI Driver

Khởi tạo Service Account

- Khởi tạo hoặc sử dụng một **service account** đã tạo trên IAM và gắn policy: **vServerFullAccess**. Để tạo service account bạn truy cập tại [đây](#) và thực hiện theo các bước sau:

- Chọn "**Create a Service Account**", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account
- Tìm và chọn **Policy: vServerFullAccess**, sau đó nhấn "**Create a Service Account**" để tạo Service Account, Policy: vLBFullAccess và Policy: vServerFullAccess do VNG Cloud tạo ra, bạn không thể xóa các policy này.
- Sau khi tạo thành công bạn cần phải lưu lại **Client_ID** và **Secret_Key** của Service Account để thực hiện bước tiếp theo.

Cài đặt VNGCloud BlockStorage CSI Driver

- Cài đặt Helm phiên bản từ 3.0 trở lên. Tham khảo tại <https://helm.sh/docs/intro/install/> để biết cách cài đặt.
- Thêm repo này vào cluster của bạn qua lệnh:

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

- Thay thế thông tin ClientID, Client Secret và ClusterID của cụm K8S của bạn và tiếp tục chạy:

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

- Sau khi việc cài đặt hoàn tất, thực hiện kiểm tra trạng thái của vngcloud-blockstorage-csi-driver pods:

```
kubectl get pods -n kube-system | grep vngcloud-csi-
```

- Ví dụ như ảnh bên dưới là bạn đã cài đặt thành công vngcloud-blockstorage-csi-driver:

NAME	READY	STATUS	RE
vngcloud-csi-controller-56bd7b85f-ctpns	7/7	Running	6
vngcloud-csi-controller-56bd7b85f-npp9n	7/7	Running	2
vngcloud-csi-node-c8r2w	3/3	Running	0

Deploy một Workload

Sau đây là hướng dẫn để bạn deploy service nginx trên Kubernetes.

Bước 1: Tạo Deployment cho Nginx app.

- Tạo file **nginx-service.yaml** với nội dung sau:

```
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 Deployment này bằng lệnh:

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

Bước 2: Kiểm tra thông tin Deployment, Service vừa deploy

- Chạy câu lệnh sau đây để kiểm tra **Deployment**

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

- Nếu kết quả trả về như bên dưới tức là bạn đã deploy Deployment thành công.

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>	30080:31	
service/nginx-service	LoadBalancer	10.96.179.221	<pending>	80:32624	
NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS
deployment.apps/nginx-app	1/1	1	1	16s	nginx
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx-app-7f45b65946-t7d7k	1/1	Running	0	16s	172.16.24

Tạo Persistent Volume

- Tạo file **persistent-volume.yaml** với nội dung sau:

```

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 dri
parameters:
  type: vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018 # The volume type UUID
allowVolumeExpansion: true                   # MUST set this value to true
---

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

apiVersion: v1
kind: Pod
metadata:
  name: nginx                             # [5] The Pod name, CAN be
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 [6]
  volumes:
    - name: my-volume-name                  # [6] The volume name, CAN be
  persistentVolumeClaim:
    claimName: my-expansion-pvc            # MUST be the same as [2]
    readOnly: false

```

- Chạy câu lệnh sau đây để triển khai 1 Pod sử dụng Persistent Volume

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

Lúc này, hệ thống vServer sẽ tự động tạo một Volume tương ứng với file yaml bên trên, ví dụ:

Tạo Snapshot

Snapshot là phương pháp sao lưu giữ liệu với chi phí thấp, thuận tiện và hiệu quả và có thể được sử dụng để tạo image, phục hồi dữ liệu và phân phối các bản sao dữ liệu. Nếu bạn là người dùng mới chưa từng sử dụng dịch vụ Snapshot, bạn cần thực hiện Activate Snapshot Service (Kích hoạt dịch vụ Snapshot) trước khi có thể tạo Snapshot cho Persistent Volume của bạn.

Activate Snapshot Service

Để có thể tạo Snapshot, bạn cần thực hiện Activate Snapshot Service. Bạn sẽ không bị tính phí khi kích hoạt dịch vụ snapshot. Sau khi bạn tạo snapshot, chi phí sẽ được tính dựa trên dung lượng lưu trữ và thời gian lưu trữ của các bản snapshot này. Thực hiện theo các bước sau đây để kích hoạt dịch vụ Snapshot:

Bước 1: Truy cập vào <https://hcm-3.console.vngcloud.vn/vserver/block-store/snapshot/overview>

Bước 2: Chọn **Activate Snapshot Service.**

Ví dụ:

The screenshot shows the VNGCloud console interface for managing snapshots. On the left, there's a sidebar with various service icons like vServer, BlockStore, and Container. Under BlockStore, 'Solutions' is expanded, showing 'Snapshots'. The main content area is titled 'Snapshot' with a status of 'DISABLE'. It contains a 'Billing Note' section stating that users are not charged for activating the service. Below this is a note about billing based on storage space and time. A large button labeled 'Activate Snapshot Service' is prominently displayed, with a red arrow pointing directly at it. To the right of the button is a link to 'What can I do with Snapshots?'. At the bottom of the page, there's some footer text and links to Terms, Privacy Policy, and various compliance logos.

Cài đặt VNGCloud Snapshot Controller

- Cài đặt Helm phiên bản từ 3.0 trở lên. Tham khảo tại <https://helm.sh/docs/intro/install/> để biết cách cài đặt.
- Thêm repo này vào cluster của bạn qua lệnh:

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

- Tiếp tục chạy:

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

- Sau khi việc cài đặt hoàn tất, thực hiện kiểm tra trạng thái của vngcloud-blockstorage-csi-driver pods:

```
kubectl get pods -n kube-system | grep snapshot-controller
```

Ví dụ như ảnh bên dưới là bạn đã cài đặt thành công vngcloud-snapshot-controller:

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

Tạo file snapshot.yaml với nội dung sau

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: my-snapshot-storage-class # [2] The name of the volume snapshot class,
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]

```

- Chạy câu lệnh sau đây để triển khai Volume Snapshot

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

Kiểm tra PVC và Snapshot vừa tạo

- Sau khi apply tập tin thành công, bạn có thể kiểm tra danh sách service, pvc thông qua:

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

NAME	PROVISIONER				
storageclass.storage.k8s.io/my-expansion-storage-class	bs.csi.vngcloud.vn				
storageclass.storage.k8s.io/sc-iops-200-retain (default)	bs.csi.vngcloud.vn				
NAME	STATUS	VOLUME			
persistentvolumeclaim/my-expansion-pvc	Bound	pvc-14456f4a-ee9e-435d-a94f			
NAME	READY	STATUS	RESTARTS	AGE	IP
pod/nginx	1/1	Running	0	10m	172.16.24
pod/nginx-app-7f45b65946-t7d7k	1/1	Running	0	94m	172.16.24

Thay đổi thông số IOPS của Persistent Volume vừa tạo

Để thay đổi thông số IOPS của Persistent Volume vừa tạo, hãy thực hiện theo các bước sau đây:

Bước 1: Chạy lệnh bên dưới để liệt kê các PVC trong Cluster của bạn

```
kubectl get persistentvolumes
```

Bước 2: Chỉnh sửa tệp tin YAML của PVC theo lệnh

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

- Nếu bạn chưa chỉnh sửa IOPS của Persistent Volume lần nào trước đó, khi bạn chạy lệnh trên, bạn hãy thêm 1 annotation bs.csi.vngcloud.vn/volume-type: "volume-type-id" . Ví dụ: bên dưới tôi đang thay đổi IOPS của Persistent Volume từ 200 (Volume type id = vtype-61c3fc5b-f4e9-45b4-8957-8aa7b6029018) lên 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-a36a40cea5"
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"PersistentVolumeClaim","metadata":{"annotations":{},"name":"my-expansion-pvc","namespace":"default","resourceVersion":"11041591","uid":"14456f4a-ee9e-435d-a94f-5a2e820954e9}}
    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

```

- Nếu bạn đã chỉnh sửa IOPS của Persistent Volume lần nào trước đó, khi bạn chạy lệnh trên, tệp tin yaml của bạn đã có sẵn annotation `bs.csi.vngcloud.vn/volume-type: "volume-type-id"`. Lúc này, hãy chỉnh sửa annotation này về Volume type id có IOPS mà bạn mong muốn.

Thay đổi Disk Volume của Persistent Volume vừa tạo

Để thay đổi Disk Volume của Persistent Volume vừa tạo, hãy thực hiện chạy lệnh sau:

Ví dụ: ban đầu PVC được tạo có kích cỡ 20 Gi, hiện tại tôi sẽ tăng nó lên 30Gi

```
kubectl patch pvc my-expansion-pvc -p '{"spec":{"resources":{"requests":{"storage": "30Gi"}}}}
```

 **Chú ý:**

- Bạn chỉ có thể thực hiện tăng Disk Volume mà không thể thực hiện giảm kích thước Disk Volume này.

Restore Persistent Volume từ Snapshot

Để khôi phục Persistent Volume từ Snapshot, bạn hãy thực hiện theo các bước sau:

- Tạo file **restore-volume.yaml** với nội dung sau:

```
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
```

Giới hạn và hạn chế CSI

Security Group

Security Group đóng vai trò như một firewall giúp bạn kiểm soát lưu lượng truy cập ra vào máy chủ (VM). Trên hệ thống VKS, để đảm bảo cluster hoạt động an toàn và hiệu quả, các Security Group mặc định được thiết lập để cho phép các truy cập cần thiết cho hoạt động nội bộ của cluster. Việc tự động tạo Security Group giúp đơn giản hóa quá trình triển khai cluster và đảm bảo rằng cluster được bảo vệ ngay từ đầu. Cụ thể, khi bạn thực hiện khởi tạo một Cluster, chúng tôi sẽ tự động khởi tạo một vài Security Group với các thông số sau:

Security group mặc định được tạo tự động cho tất cả Cluster

Mỗi Cluster được tạo ra trong hệ thống VKS, chúng tôi sẽ tự động tạo một Security Group. Security group này sẽ bao gồm:

- Inbound:

Protocol	Ether type	Port range	Source	Ý nghĩa
TCP	IPv4	30000-32767	CIDR của VPC mà bạn sử dụng cho Cluster.	Security group rule sử dụng cho TCP Node Port Services
UDP	IPv4	30000-32767	CIDR của VPC mà bạn sử dụng cho Cluster.	Security group rule sử dụng cho UDP Node Port Services
TCP	IPv4	10250	External IP của Load Balancer sử dụng cho Cluster.	Security group rule sử dụng cho Kubelet API control-plane
TCP	IPv4	10250	CIDR của VPC mà bạn sử dụng cho Cluster.	Security group rule sử dụng cho Kubelet API control-plane
TCP	IPv4	179	CIDR của VPC mà bạn sử dụng cho Cluster.	Security group rule sử dụng cho Kubelet API control-plane

4	IPv4	1-65535	CIDR của VPC mà bạn sử dụng cho Cluster.	Security group rule sử dụng cho Calico IP-in-IP
TCP	IPv4	5473	CIDR của VPC mà bạn sử dụng cho Cluster.	Security group rule sử dụng cho Calico Typha

- Outbound

Protocol	Ether type	Port range	Destination	Ý nghĩa
ANY	IPv4	0-65535	0.0.0.0/0	Rule mặc định của tất cả Security group
ANY	IPv6	0-65535	::/0	Rule mặc định của tất cả Security group

Security group được tạo tự động bởi VNGCLOUD Controller Manager

Khi bạn sử dụng VNGCloud Controller Manager để tích hợp Network Load Balancer với Cluster trên hệ thống VKS, chúng tôi sẽ tự động tạo một Security Group. Security group này sẽ bao gồm:

- Inbound:

Protocol	Ether type	Port range	Source
TCP, UDP hoặc ICMP	IPv4	Port của Service	Subnet Mask của Subnet mà bạn sử dụng cho Cluster.

- Outbound:

Protocol	Ether type	Port range	Destination	Ý nghĩa
ANY	IPv4	0-65535	0.0.0.0/0	Rule mặc định của tất cả Security group

ANY	IPv6	0-65535	::/0	Rule mặc định của tất cả Security group
-----	------	---------	------	---

Security group được tạo tự động bởi VNGCLOUD Ingress Controller

Khi bạn sử dụng VNGCloud Ingress Controller để tích hợp Application Load Balancer với Cluster trên hệ thống VKS, chúng tôi sẽ tự động tạo một Security Group. Security group này sẽ bao gồm:

- Inbound:

Protocol	Ether type	Port range	Source
TCP	IPv4	Port của Service	Subnet Mask của Subnet mà bạn sử dụng cho Cluster.

- Outbound:

Protocol	Ether type	Port range	Destination	Ý nghĩa
ANY	IPv4	0-65535	0.0.0.0/0	Rule mặc định của tất cả Security group
ANY	IPv6	0-65535	::/0	Rule mặc định của tất cả Security group

(i) Chú ý:

- Các Security Group mặc định được thiết lập để đáp ứng các nhu cầu bảo mật cơ bản của cluster. Nếu bạn sửa hoặc xóa các Security Group được tạo sẵn cho cluster, có thể dẫn đến các vấn đề về kết nối và truy cập giữa các node trong cluster hoặc cluster có thể không hoạt động chính xác hoặc thậm chí không thể khởi động được. Để đảm bảo tính ổn định và bảo mật của cluster, hệ thống sẽ tự động reset các Security Group về cài đặt mặc định sau mỗi khoảng thời gian cố định.

Upgrade Kubernetes Version

Phiên bản hỗ trợ Kubernetes

Hiện tại, hệ thống VKS đang cung cấp cho bạn 6 Kubernetes version bao gồm:

Version	Ngày hết hạn	Thao khác thêm
Vesion 1.29.1	2025-02-28	https://hub.kubes/kubes/1mas/HANG/CI/ELO/1.29.1291
Version 1.28.8	2024-10-28	https://hub.kubes/kubes/1mas/HANG/CI/ELO/1.28.1288
Version 1.27.12	2024-06-28	https://hub.kubes/kubes/1mas/HANG/CI/ELO/1.27.1271
Version 1.29.1-vks.1724605200	2025-02-28	
Version 1.28.8-vks.1724605200	2024-10-28	

Version 1.27.12-vks.1724605200

2024-06-28

Manually Upgrade

Upgrading Control Plane Version

Hiện tại, hệ thống VKS của chúng tôi đã hỗ trợ bạn nâng cấp Control Plane Version, bạn có thể:

- Nâng cấp **Minor Version** mới hơn (ví dụ: 1.24 lên 1.25)
- Nâng cấp **Patch Version** mới hơn (ví dụ: 1.24.2-VKS.100 lên 1.24.5-VKS.200)

Để thực hiện nâng cấp phiên bản Control Plane, bạn có thể thực hiện theo hướng dẫn sau:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn menu **Kubernetes Cluster**.

Bước 3: Chọn biểu tượng và chọn **Upgrade control plane version** để thực hiện nâng cấp version control plane.

Bước 4: Bạn có thể lựa chọn phiên bản mới cho control plane. Phiên bản mới cần hợp lệ và tương thích với phiên bản hiện tại của cluster. Cụ thể: bạn có thể chọn:

- Nâng cấp Minor Version mới hơn (ví dụ: 1.24 lên 1.25)
- Nâng cấp Patch Version mới hơn (ví dụ: 1.24.2-VKS.100 lên 1.24.5-VKS.200)

Bước 4: Hệ thống VKS sẽ thực hiện nâng cấp các thành phần Control Plane của Cluster lên phiên bản mới. Sau khi việc nâng cấp hoàn tất, trạng thái (status) Cluster trở về **ACTIVE**.

 **Chú ý:**

- Việc nâng cấp Control Plane Version là không bắt buộc và độc lập với việc nâng cấp Node Group Version. Tuy nhiên Control Plane Version và Node Group Version trong cùng một Cluster không được lệch quá 1 minor version. Bên cạnh đó, hệ thống VKS tự động nâng cấp Control Plane Version khi phiên bản Kubernetes Version hiện tại đang sử dụng cho Cluster của bạn quá thời hạn được nhà cung cấp hỗ trợ.

- Trong quá trình nâng cấp Control Plane Version, bạn không thể thực hiện các hành động khác trên Cluster của bạn.
- Bên dưới là một vài lưu ý trước, trong và sau quá trình nâng cấp, vui lòng tham khảo thêm:

Trước khi thực hiện:

- Sao lưu dữ liệu: Nên sao lưu dữ liệu của cluster trước khi nâng cấp để đảm bảo an toàn trong trường hợp nâng cấp thất bại.
- Kiểm tra phiên bản hiện tại: Truy cập Releases để tham khảo danh sách các phiên bản được hỗ trợ. Chọn phiên bản mới hợp lệ và tương thích với phiên bản hiện tại của cluster.
- Đảm bảo tính sẵn sàng của cluster: Cluster phải đang ở trạng thái (status) hoạt động (ACTIVE) và tất cả các node phải HEALTHY.
- Ngừng các tác vụ đang chạy: Ngừng các tác vụ đang chạy trên cluster để tránh ảnh hưởng đến quá trình nâng cấp.

Trong khi thực hiện:

- Theo dõi trạng thái (status) cluster: Theo dõi trạng thái (status) cluster trong quá trình nâng cấp. trạng thái (status) cluster sẽ chuyển sang UPDATING và sau khi hoàn tất sẽ trở về ACTIVE.
- Kiểm tra logs hệ thống: Kiểm tra logs hệ thống để phát hiện bất kỳ lỗi hoặc cảnh báo nào trong quá trình nâng cấp.

Sau khi thực hiện:

- Kiểm tra tính sẵn sàng của cluster: Xác nhận rằng cluster đã được nâng cấp thành công và tất cả các node đang hoạt động bình thường.
- Kiểm tra các ứng dụng: Kiểm tra các ứng dụng đang chạy trên cluster để đảm bảo chúng hoạt động bình thường sau khi nâng cấp.

Lưu ý:

- Việc nâng cấp Control Plane Version có thể mất một khoảng thời gian tùy thuộc vào kích thước và độ phức tạp của cluster.
- Trong một số trường hợp hiếm gặp, việc nâng cấp Control Plane Version có thể thất bại. Nếu điều này xảy ra, hệ thống VKS sẽ tự động rollback cluster về phiên bản hiện tại.

Upgrading Node Group Version

Hiện tại, hệ thống VKS của chúng tôi đã hỗ trợ bạn nâng cấp Node Group Version, bạn có thể nâng cấp Node Group Version lên:

- **Control Plane Version** (Ví dụ nâng cấp từ 1.24 (Node Group version hiện tại) lên 1.25 (Control Plane Version hiện tại), nhưng không thể nâng cấp lên các phiên bản khác.

Để thực hiện nâng cấp phiên bản Node Group Version, bạn có thể thực hiện theo hướng dẫn sau:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Tại màn hình **Overview**, chọn menu **Kubernetes Cluster**. Chọn vào một **Cluster** mà bạn muốn nâng cấp **Node Group Version**.

Bước 3: Chọn biểu tượng và chọn **Upgrade Node Group Version** để thực hiện nâng cấp version node group.

Bước 4: Bạn có thể lựa chọn phiên bản mới cho tất cả các Node Group. Phiên bản mới cần hợp lệ và tương thích với phiên bản hiện tại của cluster. Cụ thể: bạn có thể chọn:

- Nâng cấp Node Group sao cho về cùng version với Control Plane Version (ví dụ: 1.24 lên 1.25)

Bước 5: Hệ thống VKS sẽ thực hiện nâng cấp tất cả các Node Group lên version của Control Plane. Sau khi việc nâng cấp hoàn tất, trạng thái (status) Node Group trở về **ACTIVE**.

Chú ý:

- Việc nâng cấp Node Group Version là không bắt buộc và độc lập với việc nâng cấp Control Plane Version. Tuy nhiên tất cả các Node Group trong một Cluster sẽ được nâng cấp trong cùng một lần, cũng như Control Plane Version và Node Group Version trong cùng một Cluster không được lệch quá 1 minor version. Bên cạnh đó, hệ thống VKS tự động nâng cấp Node Group Version khi phiên bản Kubernetes Version hiện tại đang sử dụng cho Cluster của bạn quá thời hạn được nhà cung cấp hỗ trợ.
- Trong quá trình nâng cấp Node Group Version, bạn không thể thực hiện các hành động khác trên Node Group của bạn.

- Bên dưới là một vài lưu ý trước, trong và sau quá trình nâng cấp, vui lòng tham khảo thêm:

Trước khi thực hiện:

- Kiểm tra phiên bản hiện tại: Truy cập Releases để tham khảo danh sách các phiên bản được hỗ trợ. Chọn phiên bản mới hợp lệ và tương thích với phiên bản hiện tại của cluster.
- Đảm bảo tính sẵn sàng của Node Group: Node Group phải đang ở trạng thái (status) hoạt động (ACTIVE) và tất cả các node phải HEALTHY.
- Ngừng các tác vụ đang chạy: Ngừng các tác vụ đang chạy trên cluster để tránh ảnh hưởng đến quá trình nâng cấp.

Trong khi thực hiện:

- Theo dõi trạng thái (status) Node Group: Theo dõi trạng thái (status) Node Group trong quá trình nâng cấp. trạng thái (status) Node Group sẽ chuyển sang UPDATING và sau khi hoàn tất sẽ trở về ACTIVE.
- Kiểm tra logs hệ thống: Kiểm tra logs hệ thống để phát hiện bất kỳ lỗi hoặc cảnh báo nào trong quá trình nâng cấp.

Sau khi thực hiện:

- Kiểm tra tính sẵn sàng của Node Group: Xác nhận rằng Node Group đã được nâng cấp thành công và tất cả các node đang hoạt động bình thường.
- Kiểm tra các ứng dụng: Kiểm tra các ứng dụng đang chạy trên cluster để đảm bảo chúng hoạt động bình thường sau khi nâng cấp.

Lưu ý:

- Việc nâng cấp Node Group Version có thể mất một khoảng thời gian tùy thuộc vào kích thước và độ phức tạp của Node Group.
- Trong một số trường hợp hiếm gặp, việc nâng cấp Node Group Version có thể thất bại. Nếu điều này xảy ra, hệ thống VKS sẽ tự động rollback cluster về phiên bản hiện tại.

Automatically Upgrade

Tổng quan

Automatically Upgrade Cluster trên VKS là quá trình hệ thống VKS tự động nâng cấp cluster theo upgrade window mà bạn chỉ định, hoặc tự động nâng cấp nhằm cải thiện hiệu suất, bảo mật và khả năng tương thích của hệ thống. Hiện tại, VKS cung cấp 2 loại tự động nâng cấp bao gồm:

1. Required Upgrades (Nâng cấp bắt buộc)

- **Mục đích:** Các nâng cấp bắt buộc nhằm bảo vệ cụm khỏi các rủi ro liên quan đến bảo mật, ổn định và việc sử dụng các phiên bản Kubernetes không còn được hỗ trợ.
- **Cụ thể:** hệ thống VKS sẽ thực hiện upgrade cluster trong các trường hợp:
 - **End-of-Life (EOL) Upgrades:** Cập nhật Cluster lên các phiên bản mới hơn khi phiên bản hiện tại của Cluster sắp hết hạn.
 - **Security Upgrades:** Vá các lỗ hổng bảo mật, đảm bảo an toàn dữ liệu.
 - **Stability Upgrades:** Sửa các lỗi nghiêm trọng ảnh hưởng đến sự ổn định của hệ thống.
- **Thời gian thực hiện:**
 - Các nâng cấp này sẽ được hệ thống VKS tự động thực hiện **sau 8 giờ tối vào bất kỳ ngày nào** nếu cần, nhằm giảm thiểu ảnh hưởng đến bạn. Chúng tôi sẽ thông báo cho bạn về các nâng cấp này trong thời gian sớm nhất.

2. Regular Upgrades (Nâng cấp thông thường)

- **Mục đích:** Các nâng cấp định kỳ theo **upgrade window** mà bạn chỉ định nhằm giúp cluster của bạn luôn cập nhật với các phiên bản mới nhất, bao gồm cả **minor version** và **patch version**.
- **Cụ thể:**
 - **Minor Version Upgrades:** Cập nhật các tính năng và API mới. Ví dụ, nếu cluster hiện tại đang sử dụng phiên bản 1.28.2, hệ thống sẽ tự động nâng cấp lên phiên bản 1.29.6.
 - **Patch Version Upgrades:** Vá lỗi nhỏ và cải thiện hiệu suất. Ví dụ, nếu cluster hiện tại đang sử dụng phiên bản 1.29.1, hệ thống sẽ tự động nâng cấp lên phiên bản 1.29.2.
- **Thời gian thực hiện:**

- Bạn có thể tự cấu hình lịch trình nâng cấp thông qua **VKS Portal** theo hướng dẫn dưới đây. Sau khi bạn chọn lịch auto-upgrade qua VKS Portal, hệ thống sẽ **bắt đầu thực hiện nâng cấp sau ít nhất 6 ngày** kể từ ngày hiện tại. Thời gian này giúp bạn có đủ thời gian để chuẩn bị và kiểm tra trước khi nâng cấp diễn ra.
- Hệ thống sẽ **lặp lại cho các tuần kế tiếp** kể từ lần nâng cấp trước đó và tuân theo các ngày đã được bạn chọn.
- **Trước mỗi lần nâng cấp**, hệ thống sẽ gửi một **email thông báo** cho bạn trước 6 ngày tính tới thời điểm chạy auto-upgrade thực tế. Trong email, chúng tôi sẽ nêu rõ thời gian cụ thể mà việc auto-upgrade sẽ diễn ra.
- Nếu **chọn nhiều ngày trong tuần** (như Thứ Hai, Thứ Năm), hệ thống sẽ tính toán chu kỳ nâng cấp cho các ngày đã chọn, không ảnh hưởng đến các ngày khác.

Vui lòng tham khảo 2 ví dụ bên dưới để hiểu rõ cách hệ thống VKS thực hiện chạy Regular Upgrade.

- Trường hợp 1: Bạn chọn lịch chạy là Thứ Ba hằng tuần vào lúc 08:00 PM

*Giả sử **Hôm nay là Thứ Hai (02/12/2024)***

- *Bạn chọn auto-upgrade vào **Thứ Ba, 08:00 PM**.*
- *Lịch trình hoạt động của hệ thống sẽ như sau:*
 - *Ngày 04/12/2024 (Thứ Tư): Gửi email thông báo lịch nâng cấp lần 1.*
 - *Ngày 10/12/2024 (Thứ Ba, 08:00 PM): Thực hiện auto-upgrade lần 1 thành công.*
 - *Ngày 11/12/2024 (Thứ Tư): Gửi email thông báo lịch nâng cấp lần 2.*
 - *Ngày 17/12/2024 (Thứ Ba, 08:00 PM): Thực hiện auto-upgrade lần 2 thành công.*
 - *Ngày 18/12/2024 (Thứ Tư): Gửi email thông báo lịch nâng cấp lần 3.*
 - *Ngày 24/12/2024 (Thứ Ba, 08:00 PM): Thực hiện auto-upgrade lần 3 thành công, và tiếp tục lặp lại.*

- Trường hợp 2: Bạn chọn lịch chạy Auto-upgrade là Thứ Hai và Thứ Năm hằng tuần vào lúc 12:00 PM

*Giả sử **Hôm nay là Thứ Ba (03/12/2024)***

- *Bạn chọn auto-upgrade vào **Thứ Hai và Thứ Năm, 12:00 PM**.*
- *Lịch trình hoạt động của hệ thống sẽ như sau:*

Lịch Thứ Năm (12:00 PM):

- **Ngày 06/12/2024 (Thứ Sáu):** Gửi email thông báo lịch nâng cấp **lần 1**.
- **Ngày 12/12/2024 (Thứ Năm, 12:00 PM):** Thực hiện **auto-upgrade lần 1 thành công..**
- **Ngày 13/12/2024 (Thứ Sáu):** Gửi email thông báo lịch nâng cấp **lần 2**.
- **Ngày 19/12/2024 (Thứ Năm, 12:00 PM):** Thực hiện **auto-upgrade lần 2 thành công..**
- **Ngày 20/12/2024 (Thứ Sáu):** Gửi email thông báo lịch nâng cấp **lần 3**.
- **Ngày 26/12/2024 (Thứ Năm, 12:00 PM):** Thực hiện **auto-upgrade lần 3 thành công, và tiếp tục lặp lại.**

Lịch Thứ Hai(12:00 PM):

- **Ngày 10/12/2024 (Thứ Ba):** Gửi email thông báo lịch nâng cấp **lần 1**.
- **Ngày 16/12/2024 (Thứ Hai, 12:00 PM):** Thực hiện **auto-upgrade lần 1 thành công..**
- **Ngày 17/12/2024 (Thứ Ba):** Gửi email thông báo lịch nâng cấp **lần 2**.
- **Ngày 23/12/2024 (Thứ Hai, 12:00 PM):** Thực hiện **auto-upgrade lần 2 thành công..**
- **Ngày 24/12/2024 (Thứ Ba):** Gửi email thông báo lịch nâng cấp **lần 3**.
- **Ngày 30/12/2024 (Thứ Hai, 12:00 PM):** Thực hiện **auto-upgrade lần 3 thành công, và tiếp tục lặp lại.**

(i) Chú ý:

- Hệ thống VKS **sẽ cố gắng thực hiện nâng cấp** theo lịch trình mà bạn đã cấu hình qua **VKS Portal**. Tuy nhiên, tùy thuộc vào tải của hệ thống, **một số lần nâng cấp có thể bị hoãn** hoặc không thực hiện đúng như lịch. Khi đó, hệ thống **sẽ tự động chuyển lịch nâng cấp sang thời điểm thích hợp tiếp theo**, chính là chu kỳ lặp lại trong tuần tiếp theo.

Các bước thực hiện

Bên dưới là hướng dẫn thực hiện cập nhật Upgrade Policy trên hệ thống VKS:

Bước 1: Truy cập vào <https://vks.console.vngcloud.vn/overview>

Bước 2: Chọn **Create a Cluster**.

Bước 3: Tại màn hình khởi tạo Cluster, chúng tôi đã thiết lập thông tin cho Cluster và một **Default Node Group**. Tại mục Cluster Upgrade policy, bạn có thể chọn **Enable/ Disable Automatic upgrade**. Trong đó:

- **Enable Automatic upgrade**, lúc này:
 - Các nâng cấp **Required Upgrades** sẽ được thực hiện tự động sau 8 PM bất kỳ ngày nào bởi hệ thống VKS khi cần.
 - Các nâng cấp **Regular Upgrades** sẽ được thực hiện theo lịch trình do bạn thiết lập.
- **Disable Automatic upgrade**, lúc này:
 - Chỉ các nâng cấp **Required Upgrades** mới được thực hiện tự động sau 8 PM bất kỳ ngày nào bởi hệ thống VKS khi cần.

Bước 4: Nếu bạn chọn **Enable Automatic upgrade**, vui lòng chọn thời gian hệ thống có thể thực hiện nâng cấp. Cụ thể, bạn cần:

- **Chọn một hoặc nhiều ngày** trong tuần mà hệ thống VKS có thể thực hiện auto-upgrade (ví dụ: Monday, Tuesday, ...).
- **Chọn một mốc thời gian** cụ thể mà bạn mong muốn hệ thống VKS thực hiện auto-upgrade (ví dụ: 20:00 (08:00 PM - theo múi giờ UTC+07:00)

 **Cluster Upgrade policy**

Configure the how and when **Regular upgrades** will be installed for your Cluster.

 VKS maintenance of your cluster through required upgrades and regular upgrades.

- Required upgrades including **End-of-Life (EOL) upgrades, Security upgrades and Stability upgrades** will be performed **after 8 PM any day if needed** to ensure that clusters remain within supported versions, protecting them from critical issues that could affect data security.
- Regular upgrades including **other Minor and Patch Version upgrades** can be configured below.

Enable Automatic upgrade
When this option is disabled, only **Required upgrades** are automatically installed.

UPGRADE WINDOW
We will automatically install regular upgrades during a four-hour window after the time set.

Day *

Monday Tuesday Wednesday Thursday Friday Saturday

Select when the day on which the maintenance will run.

Time (time zone: UTC+07:00) *

20:00 

Select the time maintenance will start on the selected days.

Bước 5: Chọn **Create Kubernetes cluster/ Update Cluster**. Hãy chờ vài phút để chúng tôi khởi tạo/ chỉnh sửa Cluster của bạn, trạng thái của Cluster lúc này là **Creating/ Updating**.

Bước 6: Khi trạng thái **Cluster** là **Active**, tức là hệ thống VKS đã hoàn thành việc tạo/ chỉnh sửa cluster này.

Quy tắc hoạt động của hệ thống

Required Upgrade

Trên hệ thống VKS, required upgrades bao gồm **End-of-Life (EOL) Upgrades, Security Upgrades, Stability Upgrades**. Required Upgrades sẽ được thực hiện **sau 8:00 PM** vào bất kỳ ngày nào cần thiết. Những nâng cấp này không tuân theo lịch trình cố định mà được kích hoạt khi hệ thống phát hiện cần đảm bảo:

- Cluster đang sử dụng phiên bản Kubernetes vẫn nằm trong danh sách được hỗ trợ.
- Tránh các vấn đề nghiêm trọng có thể gây rủi ro đến bảo mật hoặc dữ liệu.

Trước khi thực hiện Required Upgrade, hệ thống sẽ gửi email thông báo chi tiết cho bạn, bao gồm:

- Thông tin Cluster/ Node Group dự kiến được nâng cấp.
- Phiên bản hiện tại và Phiên bản nâng cấp dự kiến.
- Ngày và giờ dự kiến thực hiện nâng cấp,...

Sau khi nâng cấp hoàn tất, hệ thống sẽ gửi email xác nhận tình trạng cluster và các thay đổi liên quan.

Riêng đối với trường hợp Force Upgrade, khi phát hiện ra Kubernetes version của bạn sắp hết hạn hỗ trợ. Chúng tôi sẽ gửi email thông báo cho bạn trước **60 ngày, 30 ngày, 7 ngày và 1 ngày**. Trong thời gian này, bạn có thể thực hiện nâng cấp thủ công phiên bản Kubernetes theo hướng dẫn tại [đây](#). Sau thời gian này, nếu bạn không thực hiện nâng cấp thủ công, chúng tôi sẽ tiến hành force upgrade lên phiên bản Kubernetes được hỗ trợ gần nhất. Tùy thuộc vào workload của bạn mà quá trình nâng cấp có thể gây gián đoạn dịch vụ trong khoảng thời gian khác nhau. Vui lòng tham khảo mục bên dưới để biết chi tiết.

Regular Upgrades

Regular Upgrades bao gồm các nâng cấp **Minor** và **Patch** nhằm cải thiện hiệu năng, tính năng mới, và sửa các lỗi nhỏ trong hệ thống Kubernetes. Bạn có thể tự cấu hình lịch Regular Upgrades thông qua **VKS Portal**. Hệ thống sẽ cố gắng thực hiện nâng cấp trong cluster theo ngày, giờ mà bạn đã chỉ định.

Trước khi thực hiện Regular Upgrade, hệ thống sẽ gửi email thông báo chi tiết cho bạn, bao gồm:

- Thông tin Cluster/ Node Group dự kiến được nâng cấp.
- Phiên bản hiện tại và Phiên bản nâng cấp dự kiến.
- Ngày và giờ dự kiến thực hiện nâng cấp,...

Sau khi nâng cấp hoàn tất, hệ thống sẽ gửi email xác nhận tình trạng cluster và các thay đổi liên quan.

Trước thời điểm hệ thống thực hiện tự động nâng cấp, nếu bạn thực hiện thay đổi Upgrade Window hoặc thực hiện nâng cấp thủ công, hệ thống sẽ bỏ qua lần nâng cấp tự động này. Trong trường hợp đó, bạn có thể bỏ qua email thông báo là chúng tôi gửi này. Úy thuộc vào

workload của bạn mà quá trình nâng cấp có thể gây gián đoạn dịch vụ trong khoảng thời gian khác nhau. Vui lòng tham khảo mục bên dưới để biết chi tiết.

Tối Ưu Hóa Quá Trình Upgrade và Giảm Thiểu Downtime Khi Upgrade Kubernetes Version

Nâng cấp Kubernetes cluster là một bước quan trọng để duy trì tính ổn định, bảo mật và hiệu suất của hệ thống. Tuy nhiên, quá trình này có thể dẫn đến downtime nếu không được thực hiện đúng cách. Sau đây là chi tiết các nguyên nhân phổ biến gây gián đoạn, đồng thời cung cấp các giải pháp giúp bạn giảm thiểu rủi ro trong quá trình nâng cấp.

Khi nâng cấp cluster, hai thành phần chính được nâng cấp:

- **Control Plane (Kubernetes API Server):** Thành phần này được thay thế bằng một phiên bản mới. Trong quá trình này, API server sẽ không khả dụng trong vài phút, nhưng workload sẽ không bị ảnh hưởng.
- **Worker Nodes:** Các node này được nâng cấp theo cơ chế **rolling update**, từng node một trong mỗi node group.

Quy trình nâng cấp node group:

- **Bước 1:** Hệ thống VKS thực hiện xác định các node cần nâng cấp.
- **Bước 2:** Hệ thống VKS thực hiện drain node, tức là di chuyển tất cả các pod đang chạy trên node cũ chưa nâng cấp này sang các node khác.
- **Bước 3:** Hệ thống sẽ tạo lại node mới với cấu hình đã được thiết lập trên node group và thực hiện join node này vào cụm. Nếu sau khi khởi động lại, node vẫn báo cáo trạng thái "NotReady", hệ thống sẽ tiếp tục khởi động lại node cho đến khi node trở lại trạng thái hoạt động bình thường.

Nếu **surge upgrades** được kích hoạt, hệ thống sẽ tạo tối đa 10 node mới trước khi bắt đầu nâng cấp các node hiện tại. Điều này giúp đảm bảo workload có đủ tài nguyên chạy trong suốt quá trình nâng cấp.

Các Nguyên Nhân Gây Downtime Hoặc Việc Upgrade Không Thành Công và Cách Giải Quyết

- **Pods Không Đủ Replicas Hoặc Không Được Triển Khai Qua Deployment/StatefulSet:** Nếu pod không có đủ replicas hoặc không được triển khai với Deployment/StatefulSet, khi node chứa pod này bị nâng cấp, dịch vụ sẽ ngừng hoạt động. **Giải pháp:**
 - Đảm bảo các pod quan trọng luôn có số lượng replicas ≥ 2 .
 - Sử dụng Deployment hoặc StatefulSet để tự động quản lý pods.
- **Pod Disruption Budget (PDB) Cấu Hình Quá Nhỏ:** PDB giới hạn số lượng pod có thể bị gián đoạn tại một thời điểm. Nếu cấu hình quá nhỏ, quá trình nâng cấp có thể bị treo. **Giải pháp:**
 - Cấu hình PDB phù hợp với yêu cầu của ứng dụng (ví dụ: cho phép ít nhất 1 pod luôn hoạt động).
- **Persistent Volume (PV) Không Được Cấu Hình Với ReadWriteMany:** PV với chế độ `ReadWriteOnce` chỉ có thể được gắn vào một node duy nhất. Khi node này nâng cấp, PV cần được di chuyển, gây downtime. **Giải pháp:**
 - Sử dụng chế độ `ReadWriteMany` nếu ứng dụng yêu cầu dữ liệu khả dụng trên nhiều node.
- **Pods Thiếu Liveness và Readiness Probes:** Nếu không có probes, Kubernetes không thể xác định trạng thái của pod, dẫn đến routing sai hoặc downtime. **Giải pháp:**
 - Cấu hình **Liveness Probes** để kiểm tra pod có hoạt động đúng cách.
 - Cấu hình **Readiness Probes** để đảm bảo chỉ route traffic đến các pod đã sẵn sàng.
- **Workload Mất Cân Bằng (Affinity/Anti-Affinity):** Quy tắc Affinity/Anti-Affinity quá chặt chẽ có thể khiến Kubernetes không tìm được đủ node để khởi động lại pod. **Giải pháp:**
 - Điều chỉnh các quy tắc Affinity/Anti-Affinity sao cho hợp lý.
 - Kết hợp với **surge upgrades** để tăng thêm tài nguyên tạm thời. **Surge Upgrades** là một tính năng giúp tạo các node mới trước khi nâng cấp các node hiện tại. Điều này đảm bảo workload có tài nguyên dự phòng, giúp tăng tốc quá trình nâng cấp và hạn chế downtime.

Ngoài ra, để việc upgrade diễn ra thành công, bạn cũng cần xem xét tới 2 vấn đề sau:

- **Billing Problem:** Khi thực hiện upgrade, hệ thống VKS sẽ thực hiện drain node cũ, tạo node mới trước khi thực hiện xóa node cũ, do đó, tại thời điểm hệ thống thực hiện upgrade, chi phí của cluster sẽ tạm thời tăng. Nếu số dư credit không đủ, hệ thống không thể tạo node mới, khiến quá trình nâng cấp bị gián đoạn. **Giải pháp:**
 - Theo dõi thông báo lịch **upgrade qua email và chuẩn bị số dư credit** đủ để hệ thống thực hiện nâng cấp.
- **Resource Quota Problem:** Trong trường hợp số lượng node mà bạn có thể tạo do vượt quá quota hiện tại, việc nâng cấp Kubernetes có thể bị gián đoạn vì không thể tạo thêm node mới. Điều này đặc biệt quan trọng khi sử dụng **Surge Upgrade**, nơi hệ thống cần

tạo thêm các node mới trước khi xóa các node cũ để đảm bảo tính liên tục của dịch vụ.

Giải pháp:

- Theo dõi thông báo lịch **upgrade qua email** và thực hiện yêu cầu nâng cấp quota nếu có.

Trong trường hợp bạn muốn chủ động kiểm soát việc nâng cấp Kubernetes Cluster (manually upgrade) và không muốn hệ thống tự động thực hiện nâng cấp, bạn có thể tắt tính năng **Regular Upgrade** bằng cách bỏ chọn **Enable Automatic Upgrade**.



Enable Automatic upgrade

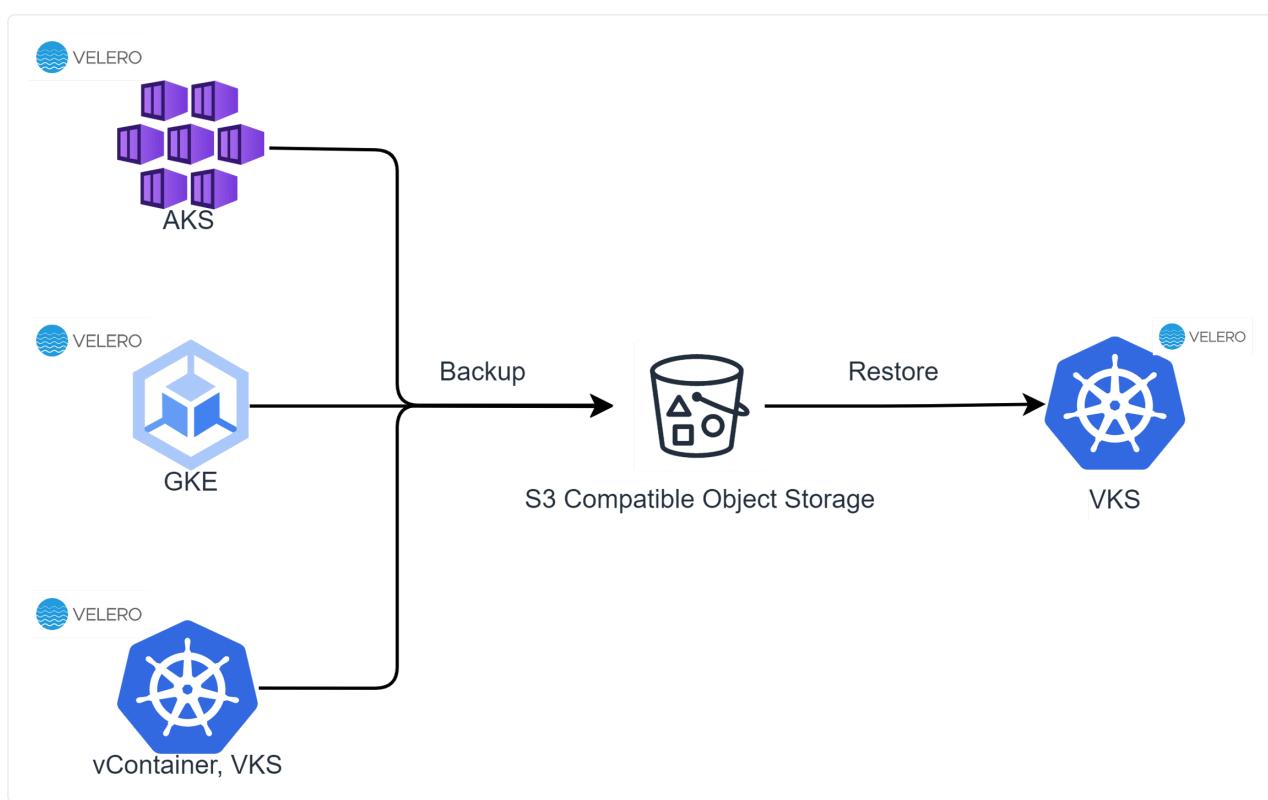
When this option is disabled, only Required upgrades are automatically installed.

Migration

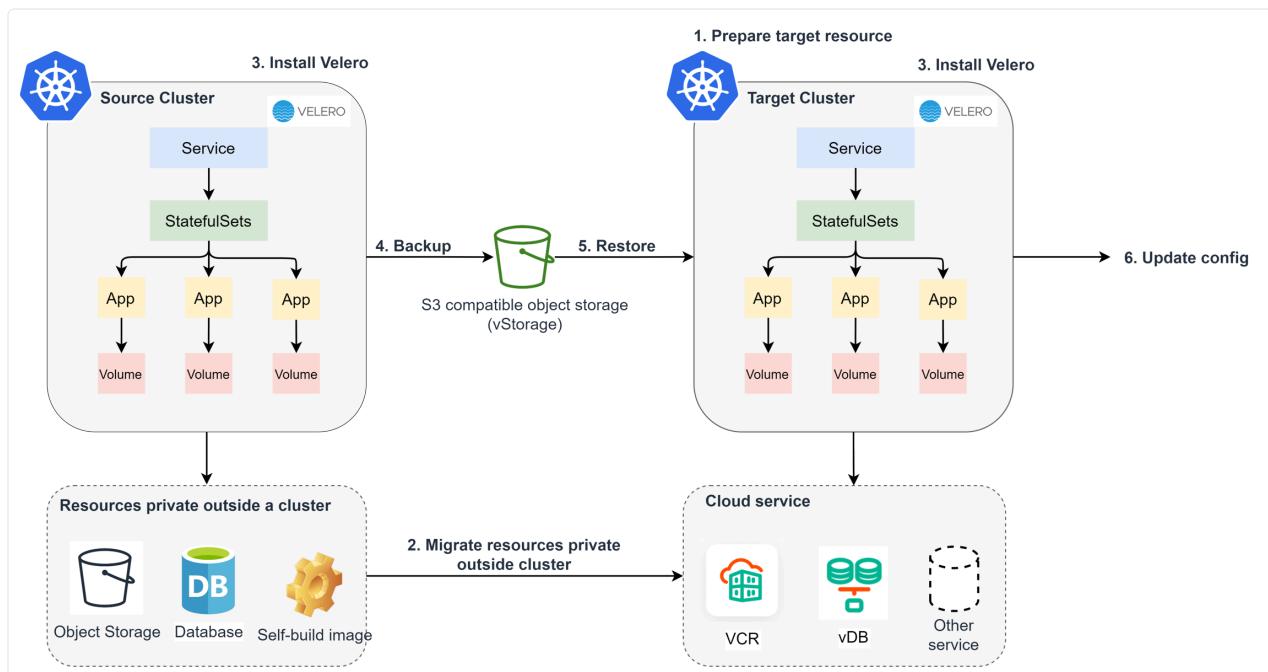
Tổng quan

Migration từ một cluster sang một cluster là quá trình chuyển dữ liệu, ứng dụng, và các dịch vụ từ một nhóm máy chủ này sang một nhóm máy chủ khác. Mục đích của việc này thường là để nâng cấp hệ thống, tăng cường tính sẵn sàng và khả năng chịu lỗi, hoặc để mở rộng quy mô hệ thống.

Mô hình tổng quan



Các bước thực hiện



Cụ thể:

- **Bước 1:** Chuẩn bị cluster đích (Prepared target resource): trên hệ thống VKS, bạn cần thực hiện khởi tạo một Cluster theo hướng dẫn tại [đây](#). Đảm bảo rằng cấu hình của cluster đích giống với cấu hình của cluster nguồn.
- **Bước 2 [Optional]:** Nếu cluster của bạn có các tài nguyên riêng tư như image, database, storage... Lúc này, trước khi bắt đầu migrate, bạn cần **chủ động tự thực hiện** việc migrate các tài nguyên này.
- **Bước 3:** Cài đặt Velero trên cả 2 cluster nguồn và cluster đích(Install Velero tool): sau khi bạn đã thực hiện migrate các tài nguyên private ngoài cluster, bạn có thể sử dụng công cụ migration để sao lưu (backup) và khôi phục (restore) application trên cluster nguồn và cluster đích.
- **Bước 4:** Sao lưu (Backup): Để sao lưu tài nguyên, hãy sử dụng công cụ Velero để tạo đối tượng sao lưu trong cluster nguồn. Velero sẽ thực hiện truy vấn, đóng gói dữ liệu và tải chúng lên một S3 Compatible Object Storage.
- **Bước 5:** Khôi phục (Restore): Trong quá trình khôi phục tại cluster đích, Velero sẽ thực hiện tải dữ liệu sao lưu xuống cụm mới và triển khai lại tài nguyên dựa trên tệp JSON.
- **Bước 6 [Optional]:** Update resource config: Sau khi tài nguyên của cluster đích được triển khai đúng cách, bạn có thể thực hiện **switch traffic** cho dịch vụ của bạn. Sau khi xác nhận rằng tất cả các dịch vụ đều chạy bình thường, bạn có thể thực hiện xóa cluster nguồn.

Bên dưới là hướng dẫn chi tiết các trường hợp phổ biến khi bạn thực hiện migrate workload từ một Cluster sang một Cluster khác, bạn có thể tham khảo và làm theo hướng dẫn tại:

- [Migrate Cluster from VKS to VKS](#)
- [Migrate Cluster from vContainer to VKS](#)
- [Migrate Cluster from another platform to VKS](#)

Migrate Cluster từ VKS tới VKS

Để migrate một Cluster từ hệ thống VKS tới hệ thống VKS, hãy thực hiện theo các bước theo tài liệu này.

Điều kiện cần

- **Thực hiện tải xuống helper bash script và grant execute permission cho file này ([velero_helper.sh](#))**
- (Optional) Triển khai một vài service để kiểm tra tính đúng đắn của việc migrate. Giả sử, tại Cluster nguồn, tôi đã triển khai một service nginx như sau:
 - File triển khai:

```

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

```

```
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>" | tee index.html
```

- Lúc này, khi bạn truy cập vào Public IP của Node, bạn sẽ thấy "Hello, MyVNGCloud".

Chuẩn bị cluster đích (Prepare target resource)

Trên hệ thống VKS, bạn cần thực hiện khởi tạo một Cluster theo hướng dẫn tại [đây](#). Đảm bảo rằng cấu hình của cluster đích giống với cấu hình của cluster nguồn.

Chú ý:

Để việc migrate thành công, trên Cluster đích, bạn cần đảm bảo các yêu cầu sau:

- Lượng resource cần thiết như số lượng node, cấu hình instance của node,...
- Node labels và node taints giống cluster cũ.
- Storage Class tương ứng hoặc thay thế.

[Optional] Migrate resources private outside cluster

Migrating resources private outside cluster (di chuyển tài nguyên riêng tư bên ngoài cụm) là quá trình di chuyển tài nguyên riêng tư nằm ngoài Cluster nguồn sang một nơi mà Cluster đích có thể sử dụng. Ví dụ, bạn có thể có những tài nguyên riêng tư như image, database,... Lúc này, trước khi bắt đầu migrate, bạn cần tự thực hiện việc migrate các tài nguyên này. Ví dụ, nếu bạn cần:

- Migrate Container Images: bạn có thể migrate image tới VNGCloud Container Registry thông qua hướng dẫn tại [đây](#).
- Migrate Databases: bạn có thể sử dụng **Relational Database Service (RDS)** và **Object Storage Service (OBS)** tùy theo nhu cầu sử dụng của bạn. Sau khi việc migration hoàn tất, hãy config lại database cho applications của bạn trên VKS Cluster.
- Migrate Storage: bạn có thể sử dụng **NFS Server** của vServer.

(i) Chú ý:

- Sau khi bạn thực hiện migrate các resource ngoài Cluster, bạn cần đảm bảo Cluster đích kết nối được tới các resource đã migrate này.

Cài đặt Velero trên cả 2 cluster nguồn và cluster đích (Install Velero tool)

Sau khi bạn đã thực hiện migrate các tài nguyên private ngoài cluster, bạn có thể sử dụng công cụ migration để sao lưu (backup) và khôi phục (restore) application trên cluster nguồn và cluster đích.

- Tạo một **vStorage Project, Container** làm nơi nhận dữ liệu backup của cụm theo hướng dẫn tại [đây](#).
- Khởi tạo S3 key tương ứng với vStorage Project này theo hướng dẫn tại [đây](#).

Ví dụ, tôi đã khởi tạo một vStorage Project, Container có thông tin sau: Region: HCM03, Container: mycontainer, Endpoint: <https://hcm03.vstorage.vngcloud.vn>.

Trên cả 2 Cluster (source and target)

- Tạo file **credentials-velero** với nội dung sau:

```
[default]
aws_access_key_id=-----
aws_secret_access_key=-----
```

- Cài đặt Velero CLI:

```
curl -OL https://github.com/vmware-tanzu/velero/releases/download/v1.13.2/velero-v1.13.2-linux-amd64.tar.gz
tar -xvf velero-v1.13.2-linux-amd64.tar.gz
cp velero-v1.13.2-linux-amd64/velero /usr/local/bin
```

- Cài đặt Velero trên 2 cụm của bạn theo lệnh:

```
velero install --provider aws \
--plugins velero/velero-plugin-for-aws:v1.9.0,velero/velero-plugin-for-csi:
--secret-file ./credentials-velero \
--bucket _____ \
--backup-location-config region=hcm03,s3ForcePathStyle="true",s3Url=https://
--use-node-agent \
--features=EnableCSI

velero client config set features=EnableCSI
```

Chú ý:

- Khi bạn thực hiện migrate cluster từ VKS tới VKS, chúng tôi khuyến cáo bạn sử dụng **Snapshot** để migrate Volume của bạn từ cluster nguồn qua cluster đích.
- Cài đặt plugin **VNGCloud Snapshot Controller** trên 2 cluster theo lệnh:

```
helm repo add vks-helm-charts https://vngcloud.github.io/vks-helm-charts
helm repo update
helm install vngcloud-snapshot-controller vks-helm-charts/vngcloud-snapshot-c
--replace --namespace kube-system
```

Tại Cluster nguồn

- Annotate các Persistent Volume cần backup. Mặc định velero sẽ không backup volume. Bạn có thể chạy lệnh dưới để annotate backup tất cả volume.

```
./velero_helper.sh mark_volume -c
```

- Ngoài ra, bạn có thể đánh dấu không backup các resource của system bằng lệnh sau:

```
./velero_helper.sh mark_exclude -c
```

- Thực hiện apply file bên dưới để tạo 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.io/csi
# user can choose the VolumeSnapshotClass by setting annotation velero.io/csi
```

- Thực hiện backup theo cú pháp:

```
velero backup create vks-full-backup \
--exclude-namespaces velero \
--include-cluster-resources=true \
--wait
```

```
velero backup describe vks-full-backup --details
```

Tại Cluster đích

- Thực hiện restore theo lệnh:

```
velero restore create --from-backup vks-full-backup \
--exclude-resources="MutatingWebhookConfiguration,ValidatingWebhookConfig
```

```
velero restore create --from-backup vks-full-backup
```

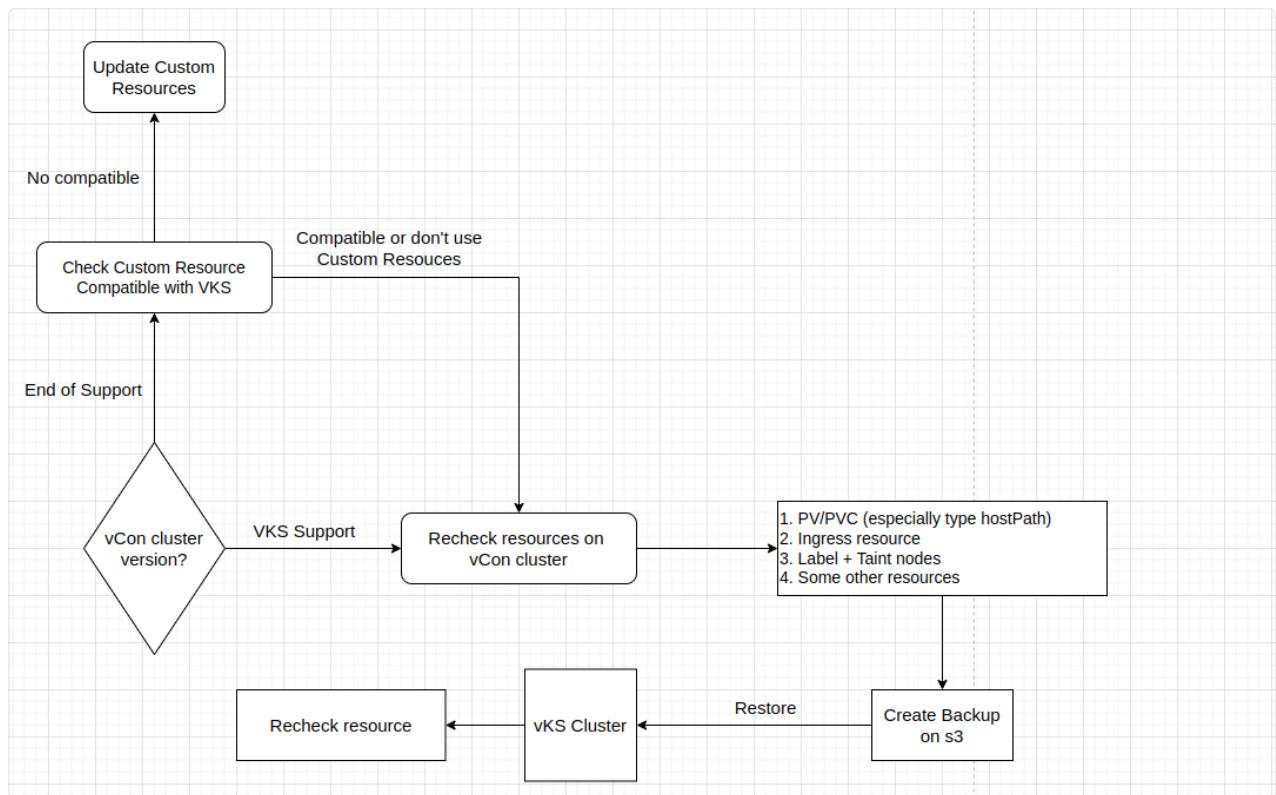
Migration Cluster từ vContainer tới VKS

Để migrate một Cluster từ hệ thống vContainer tới hệ thống VKS, hãy thực hiện theo các bước theo tài liệu này.

Điều kiện cần

- **Thực hiện tải xuống helper bash script và grant execute permission cho file này ([velero_helper.sh](#))**

Quy trình thực hiện



Quy trình thực hiện migrate từ vContainer sang VKS (Khách hàng + VNG Cloud)

Step 1: Đánh giá version hiện tại của vContainer cluster và tương ứng với vKS cluster sẽ migrate. Lúc này sẽ xảy ra 2 trường hợp (step 2 và step 3)

Step 2: Nếu version vContainer thấp hơn so với các version đang được hỗ trợ bởi vKS, thì các Custom Resources (CRD) cần được xem xét mức độ tương thích với các version kubernetes mới.

- Nếu tương thích với version vKS mới, tiếp tục thực hiện step 3
- Nếu không tương thích, cần manual update và cấu hình lại CRD cũng như các Applications liên quan

Step 3: Nếu version vContainer được hỗ trợ bởi vKS (1.27, 1.28 và 1.29), thì cần kiểm tra các resources trên vContainer trước khi backup. Cần lưu ý với các loại resources sau:

- **PV/PVC:** Velero không hỗ trợ backup với type hostPath, chỉ hỗ trợ type Local.
- **Ingress resources:** Ingress resources được quản lý bởi **container-ingress-nginx-controller** sau khi migrate sang sẽ không hoạt động được.
- **Label và Taint node:** Velero không thực hiện gắn lại các Label và Taint cho các nodes ở vKS

Step 4: Thực hiện backup các resources trên vContainer cluster

Step 5: Thực hiện restore các resources trên vKS cluster

Step 6: Thực hiện kiểm tra và những điều chỉnh

Lưu ý quan trọng:

1. Mapping StorageClass trên vKS cluster

Kiểm tra các StorageClass hiện có trên vContainer và tạo các StorageClass tương ứng trên vKS. Sau đó thực hiện mapping 1:1 StorageClass giữa vContainer và vKS cluster.

Ví dụ file ConfigMap ở **Bước 3**.

2. Sử dụng PersistentVolume dạng hostPath

Velero không hỗ trợ backup volume dạng hostPath, chỉ hỗ trợ dạng local.

Đối với các cluster đang sử dụng PV type hostPath, cần thực hiện chuyển sang dạng local như sau:

Lưu ý: Cần thực hiện xóa và deploy lại Application đang sử dụng PV type hostPath

Case 1: Type Local hỗ trợ đường dẫn hostPath

Ví dụ PV hostPath đang cấu hình mount tại folder /opt/data, thực hiện convert sang type Local và mount vào lại Pods, theo mẫu sau:

```
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 không hỗ trợ đường dẫn hostPath gốc

Khi thực hiện tạo các PV/PVC theo Case 1, Pod không mount được PVC và xuất hiện lỗi

MountVolume.NewMounter initialization failed for volume "pvc" : path "/mnt/data" does not exist

Thực hiện copy data sang một folder mới (ví dụ /var, /opt, /tmp, ...) và thực hiện lại theo Case 1, sau đó mount PVC vào Pod như thường:

```
cp -R /mnt/data /var
```

Các bước thực hiện chi tiết

Bước 1: Cài đặt Velero trên cả 2 cluster (vContainer và vKS)

Tạo một project vStorage, Container và S3 key tương ứng để làm nơi lưu trữ dữ liệu backup

Trên cả 2 cluster:

- Tạo file **credentials-velero** với nội dung sau:

```
[default]
aws_access_key_id=----- # <= Adjust here
aws_secret_access_key=----- # <= Adjust here
```

- Cài đặt Velero CLI

```
curl -OL https://github.com/vmware-tanzu/velero/releases/download/v1.14.1/velero-v1.14.1-linux-amd64.tar.gz
tar -xvf velero-v1.14.1-linux-amd64.tar.gz
cp velero-v1.14.1 -linux-amd64/velero /usr/local/bin
```

- Cài đặt Velero trên 2 cluster kubernetes

```
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=https://
```

Bước 2: Thực hiện backup trên Cluster vContainer

Đối với

- Annotate các Persistent Volume cần backup

```
./velero_helper.sh mark_volume --confirm
```

- Annotate các resource không backup của kube-system

```
./velero_helper.sh mark_exclude --confirm
```

- Annotate các resource khác (không được mark trong file velero_helper.sh), như CSI, Ingress Controller, hoặc những resources khác không muốn migrate (lưu ý cần mark label hết toàn bộ resources của objects không cần backup).

- Ví dụ như một application bao gồm DaemonSet, Deployment, Pod, ... thì cần mark label cho toàn bộ resources đó

```
# 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
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-controller velero.io/exclude-from-backup=true
kubectl -n kube-system label Deployment/vcontainer-ingress-nginx-default-backend velero.io/exclude-from-backup=true
```

- Thực hiện kiểm tra và gắn các Labels and Taint trên nodes vContainer vào các nodes vKS trước khi restore

```
# Kiểm tra các Label nodes
./velero_helper.sh check_node_label
# Kiểm tra các Taint nodes
./velero_helper.sh check_node_taint
```

- Tạo 2 bản backup cho Cluster resources và Namespace resource theo cú pháp

```
# Tạo cluster resource backup
velero backup create vcontainer-cluster --include-namespaces "" --include-cluster-scopes --namespace=vcontainer-cluster

# Tạo cluster namespace backup
velero backup create vcontainer-namespace --exclude-namespaces velero --wait

# Xóa các bản backup (nếu cần)
velero backup delete vcontainer-namespace vcontainer-cluster --confirm
```

- Xem các bản backup đã được tạo và details các resources được backup (chú ý **STATUS** của 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>
```

Bước 3: Thực hiện Restore trên Cluster VKS

- Nếu ở cluster vContainer sử dụng CSI là **cinder.csi.openstack.org**, cần thực hiện mapping StorageClass giữa 2 cluster vContainer và vKS
 - Mapping **csi-sc-cinderplugin-nvme-5000** (vContainer) và **vngcloud-nvme-5000-delete** (vKS), tương tự đối với các StorageClass khác
 - Ở vKS cần tạo các StorageClass tương ứng

Tạo file **sc-mapping.yaml** và apply trên cluster VKS

```
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 here
```

- Thực hiện thêm permissions cho Velero được quyền restore data PersistentVolume

Tạo file **add-permission.yaml** và apply trên cluster VKS

```
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
```

- Thực hiện restore theo thứ tự
 - Lưu ý, với mỗi lần thực hiện restore, cần kiểm tra quá trình restore đã thành công hay chưa rồi mới tiếp tục thực hiện các command khác

```
# Kiểm tra restore
velero get restore
velero describe restore <restore-name> --details
```

```
velero restore create --item-operation-timeout 1m --from-backup vcontainer-clust
```

```
velero restore create --item-operation-timeout 1m --from-backup vcontainer-names
```

```
velero restore create --item-operation-timeout 1m --from-backup vcontainer-clust
```

- Trường hợp migrate qua VKS vẫn sử dụng vcontainer-ingress-controller, thì cần thực hiện đổi type Service thành LoadBalancer

```
kubectl patch service -n kube-system vcontainer-ingress-nginx-controller -p '{"s
```

Migrate Cluster từ các platform khác tới VKS

Để migrate một Cluster từ hệ thống Cloud Provider hoặc On-premise tới hệ thống VKS, hãy thực hiện theo các bước theo tài liệu này.

Điều kiện cần

- **Thực hiện tải xuống helper bash script và grant execute permission cho file này ([velero_helper.sh](#))**
- (Optional) Triển khai một vài service để kiểm tra tính đúng đắn của việc migrate. Giả sử, tại Cluster nguồn, tôi đã triển khai một service nginx như sau:
 - File triển khai:

```

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  <h1>
```

- Lúc này, khi bạn truy cập vào Public IP của Node, bạn sẽ thấy "Hello, MyVNGCloud".

Chuẩn bị cluster đích (Prepare target resource)

Trên hệ thống VKS, bạn cần thực hiện khởi tạo một Cluster theo hướng dẫn tại [đây](#). Đảm bảo rằng cấu hình của cluster đích giống với cấu hình của cluster nguồn.

 **Chú ý:**

Để việc migrate thành công, trên Cluster đích, bạn cần đảm bảo các yêu cầu sau:

- Lượng resource cần thiết như số lượng node, cấu hình instance của node,...
- Node labels và node taints giống cluster cũ.
- Storage Class tương ứng hoặc thay thế.

[Optional] Migrate resources private outside cluster

Migrating resources private outside cluster (di chuyển tài nguyên riêng tư bên ngoài cụm) là quá trình di chuyển tài nguyên riêng tư nằm ngoài Cluster nguồn sang một nơi mà Cluster đích có thể sử dụng. Ví dụ, bạn có thể có những tài nguyên riêng tư như image, database,... Lúc này, trước khi bắt đầu migrate, bạn cần tự thực hiện việc migrate các tài nguyên này. Ví dụ, nếu bạn cần:

- Migrate Container Images: bạn có thể migrate image tới VNGCloud Container Registry thông qua hướng dẫn tại [đây](#).
- Migrate Databases: bạn có thể sử dụng **Relational Database Service (RDS)** và **Object Storage Service (OBS)** tùy theo nhu cầu sử dụng của bạn. Sau khi việc migration hoàn tất, hãy config lại database cho applications của bạn trên VKS Cluster.
- Migrate Storage: bạn có thể sử dụng **NFS Server** của vServer.

(i) Chú ý:

- Sau khi bạn thực hiện migrate các resource ngoài Cluster, bạn cần đảm bảo Cluster đích kết nối được tới các resource đã migrate này.

Cài đặt Velero trên cả 2 cluster nguồn và cluster đích (Install Velero tool)

Sau khi bạn đã thực hiện migrate các tài nguyên private ngoài cluster, bạn có thể sử dụng công cụ migration để sao lưu (backup) và khôi phục (restore) application trên cluster nguồn và cluster đích.

- Tạo một **vStorage Project, Container** làm nơi nhận dữ liệu backup của cụm theo hướng dẫn tại [đây](#).
- Khởi tạo S3 key tương ứng với vStorage Project này theo hướng dẫn tại [đây](#).

Ví dụ, tôi đã khởi tạo một vStorage Project, Container có thông tin sau: Region: HCM03, Container: mycontainer, Endpoint: <https://hcm03.vstorage.vngcloud.vn>.

Trên cả 2 Cluster (source and target)

- Tạo file **credentials-velero** với nội dung sau:

```
[default]
aws_access_key_id=-----
aws_secret_access_key=-----
```

- Cài đặt Velero CLI:

```
curl -OL https://github.com/vmware-tanzu/velero/releases/download/v1.13.2/velero-v1.13.2-linux-amd64.tar.gz
tar -xvf velero-v1.13.2-linux-amd64.tar.gz
cp velero-v1.13.2-linux-amd64/velero /usr/local/bin
```

- Cài đặt Velero trên 2 cụm của bạn theo lệnh:

```
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",s3Url=https://s3.hcm03.vnaws.com
```

Đối với Cluster trên Amazon Elastic Kubernetes Service (EKS)

Tại Cluster nguồn

- Annotate các Persistent Volume cần backup. Mặc định velero sẽ không backup volume. Bạn có thể chạy lệnh dưới để annotate backup tất cả volume.

```
./velero_helper.sh mark_volume -c
```

- Ngoài ra, bạn có thể đánh dấu không backup các resource của system bằng lệnh sau:

```
./velero_helper.sh mark_exclude -c
```

- Thực hiện backup theo cú pháp:

```
velero backup create eks-cluster --include-namespaces "" \
    --include-cluster-resources=true \
    --wait
```

```
velero backup create eks-namespace --exclude-namespaces velero \
    --wait
```

Chú ý:

- Bạn phải tạo 2 phiên bản backup cho Cluster Resource và Namespace Resource.

Tại Cluster đích

- Tạo file mapping Storage Class giữa Cluster nguồn và đích:

```
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----- # <= Adjustment
  -----old_storage_class-----: -----new_storage_class----- # <= Adjustment
```

- Thực hiện restore theo lệnh:

```
velero restore create --item-operation-timeout 1m --from-backup eks-cluster \
--exclude-resources="MutatingWebhookConfiguration,ValidatingWebhookConfig"
velero restore create --item-operation-timeout 1m --from-backup eks-namespace
velero restore create --item-operation-timeout 1m --from-backup eks-cluster
```

Đối với Cluster trên Google Kubernetes Engine (GKE)

Tại Cluster nguồn

- Annotate các Persistent Volume và lable resource cần loại trừ khỏi bản backup

```
./velero_helper.sh mark_volume -c
```

- Ngoài ra, bạn có thể đánh dấu không backup các resource của system bằng lệnh sau:

```
./velero_helper.sh mark_exclude -c
```

- Thực hiện backup theo cú pháp:

```
velero backup create gke-cluster --include-namespaces "" \
--include-cluster-resources=true \
--wait
```

```
velero backup create gke-namespace --exclude-namespaces velero \
--wait
```

(i) Chú ý:

- Bạn phải tạo 2 phiên bản backup cho Cluster Resource và Namespace Resource.

Tại Cluster đích

- Tạo file mapping Storage Class giữa Cluster nguồn và đích:

```
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 # <= Adjustment
  old_storage_class: new_storage_class # <= Adjustment
```

- Thực hiện restore theo lệnh:

```
velero restore create --item-operation-timeout 1m --from-backup gke-cluster \
  --exclude-resources="MutatingWebhookConfiguration,ValidatingWebhookConfig"
```

```
velero restore create --item-operation-timeout 1m --from-backup gke-namespace
```

```
velero restore create --item-operation-timeout 1m --from-backup gke-cluster
```

(i) Chú ý:

- Google Kubernetes Engine (GKE) không cho phép triển khai daemonset trên tất cả các node. Tuy nhiên, Velero chỉ cần triển khai daemonset trên node có mount PV. Giải pháp cho vấn đề này là bạn có thể điều chỉnh taint và toleration của daemonset để chỉ triển khai nó trên node có mount PV.
- Bạn có thể thay đổi yêu cầu tài nguyên mặc định `cpu:500m` và `mem:512M` trong bước cài đặt hoặc điều chỉnh khi triển khai yaml.

Giới hạn và hạn chế

Khi sử dụng Velero để migrate Cluster to Cluster, bạn có thể thêm các tùy chọn sau.

Đánh dấu các Volume bạn muốn backup và các resource không cần thiết

Để đánh dấu các Volume bạn muốn backup và các resource không cần thiết, đầu tiên bạn cần tải xuống đoạn helper bash script được chung tôi cung cấp sẵn và thực hiện grand execute permission. Chi tiết file mẫu bạn có thể xem tại: [velero_helper.sh](#)

1. Chuyển hostPath Volume thành Persistent Volume để có thể thực hiện backup

Do Velero không hỗ trợ sao lưu hostPath Volume, bạn cần phải chuyển hostPath Volume thành Persistent Volume theo hướng dẫn sau đây:

- Để list các hostPath Volume đang sử dụng:

```
./helper.sh check_hostPath
```

2. Mark Persistent Volume to include in backup

Tất cả data Persistent Volumes được lưu trữ trên vStorage. Cần thêm annotation cho tất cả pod dùng PV với volume name: `backup.velero.io/backup-volumes=volume1,volume2`

- Hoặc có thể tự động tìm các volume bằng cách:

```
./helper.sh mark_volume
```

3. Mark resource in exclude in backup

Do VKS hoạt động theo cơ chế Fully Managed Control Plane, nên bạn không cần backup các resource như: `calico`, `kube-dns`, `kube-scheduler`, `kube-apiserver`,... Ngoài ra, các resource của vContainer như là: `magnum-auto-healer`, `cluster-autoscaler`, `csi-cinder`,... cũng sẽ được bỏ qua.

- Đánh dấu resource không cần backup thông qua lệnh:

```
./helper.sh mark_exclude
```

4. Check label and taint of node

Khi thực hiện migrate, có thể tài nguyên trong Cluster nguồn đang sử dụng label và taint. Bạn cần đảm bảo các label và taint quan trọng này tồn tại trong Cluster đích.

- Kiểm tra label và taint thông qua lệnh:

```
./helper.sh check_node_label
./helper.sh check_node_taint
```

5. Mapping Storage Class

- Nếu Storage Class của bạn khác nhau giữa Cluster nguồn và Cluster đích, bạn cần chuyển Storage Class giữa 2 cụm. Ví dụ:

- Tại Cluster nguồn, bạn đang có 2 Storage Class sau:

@ kubectl get sc			
NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBO
sc-iops-200-retain (default)	csi.vngcloud.vn	Retain	Immedia
sc-ssd-10000-delete (default)	csi.vngcloud.vn	Delete	Immedia

- Bạn có thể tạo file mapping chứa nội dung như ví dụ bên dưới để thực hiện chuyển đổi 2 storage class từ Cluster nguồn thành 2 storage class tại Cluster đích. File này phải được apply tại Cluster đích trước khi bạn chạy lệnh backup:

```
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
```

Sử dụng VKS với Terraform

Terraform là gì?

Terraform là một cơ sở hạ tầng nguồn mở dưới dạng công cụ mã cho phép người dùng quản lý cơ sở hạ tầng của họ một cách dễ dàng và hiệu quả trên các nền tảng đám mây khác nhau, chẳng hạn như VNG Cloud, AWS, Google Cloud và Azure. Máy chủ Terraform đề cập đến phiên bản của công cụ Terraform đang chạy trên một máy chủ hoặc máy cù thể. Đây là nơi mã cơ sở hạ tầng được viết và thực thi, cho phép người dùng tạo, sửa đổi và hủy tài nguyên trên nền tảng đám mây.

Bản thân Terraform không có giao diện người dùng đồ họa, thay vào đó người dùng tương tác với nó bằng giao diện dòng lệnh. Terraform yêu cầu tài khoản và khóa của nhà cung cấp đám mây được định cấu hình cùng với tệp cấu hình Terraform để thực thi cơ sở hạ tầng dưới dạng mã. Ngoài ra, Terraform có thể hoạt động trong môi trường nhóm nơi nhiều người dùng có thể cộng tác trên cùng một cơ sở mã cơ sở hạ tầng, khiến nó trở thành một công cụ mạnh mẽ và linh hoạt để quản lý cơ sở hạ tầng đám mây.

Các bước thực hiện

Để khởi tạo một Cluster Kubernetes bằng Terraform, bạn cần thực hiện các bước sau:

1. **Truy cập IAM Portal** tại [đây](#), thực hiện tạo Service Account với quyền hạn **VKS Full Access**. Cụ thể, tại trang IAM, bạn có thể:
 - Chọn "**Create a Service Account**", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account.
 - Tìm và chọn **Policy: VKSFullAccess** sau đó nhấn "**Create a Service Account**" để tạo Service Account, **Policy: VKSFullAccess** do VNG Cloud tạo ra, bạn không thể xóa các policy này.
 - Sau khi tạo thành công bạn cần phải lưu lại **Client_ID** và **Secret_Key** của Service Account để thực hiện bước tiếp theo.
2. **Truy cập VKS Portal** tại [đây](#), **thực hiện Activate** dịch vụ VKS ở tab **Overview**. Hãy chờ đợi tới khi chúng tôi khởi tạo thành công tài khoản VKS của bạn.
3. **Cài đặt Terraform:**
 - Tải xuống và cài đặt Terraform cho hệ điều hành của bạn từ <https://developer.hashicorp.com/terraform/install>.

4. Khởi tạo cấu hình Terraform:

- Tạo tệp `variable.tf` và khai báo thông tin Service Account trong file này.
- Tạo tệp `main.tf` và định nghĩa các tài nguyên Kubernetes Cluster mà bạn muốn tạo.

Ví dụ:

- Tệp `variable.tf`: bạn cần thay thế Client ID và Client Secret đã khởi tạo ở bước 1 ở file này.

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

-
- Trên file `main.tf`, bạn cần có thể thêm resource để tạo Cluster/ Node Group:
 - Tạo Cluster my-vks-cluster và Node Group my-nodegroup độc lập:

```
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  
}
```

- Tạo Cluster với 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"
  }
}

```

 **Chú ý:**

- Chúng tôi khuyên bạn nên tạo và quản lý các Cluster, Node Group dưới dạng resource riêng biệt, như trong ví dụ bên dưới. Điều này cho phép bạn thêm hoặc xóa các Node Group mà không cần tạo lại toàn bộ Cluster. Nếu bạn khai báo trực tiếp Node Group Default trong tài nguyên vngcloud_vks_cluster, bạn không thể xóa chúng mà không tạo lại chính Cluster đó.
- Trong file main.tf, để khởi tạo một cluster với một node group thành công, bạn bắt buộc cần nhập thông tin của 4 field sau:

```

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"

```

Các ví dụ tham khảo

Example Usage 1 - Create a Cluster with Network type CALICO OVERLAY and a Node Group with AutoScale Mode

```

terraform {
  required_providers {
    vngcloud = {
      source  = "vngcloud/vngcloud"
      version = "1.2.7"
    }
  }
}

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 Usage 2 - Create a Cluster with Network type CILIUM OVERLAY and a Node Group with AutoScale Mode

```

terraform {
  required_providers {
    vngcloud = {
      source  = "vngcloud/vngcloud"
      version = "1.2.7"
    }
  }
}

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 = "CILIUM_OVERLAY"
  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 Usage 3 - Create a Cluster with Network type CILIUM VPC Native Routing and a Node Group with AutoScale Mode

```

terraform {
  required_providers {
    vngcloud = {
      source  = "vngcloud/vngcloud"
      version = "1.2.7"
    }
  }
}

resource "vngcloud_vks_cluster" "primary" {
  name        = "cluster-demo"
  description = "Cluster create via terraform"
  version     = "v1.29.1"
  enable_private_cluster = false
  network_type = "CILIUM_NATIVE_ROUTING"
  vpc_id      = "net-70ef12d4-d619-43fc-88f0-1c1511683123"
  subnet_id   = "sub-0725ef54-a32e-404c-96f2-34745239c123"
  secondary_subnets = ["10.200.27.0/24", "10.200.28.0/24"]
  node_netmask_size = 25
  enable_service_endpoint = false
  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"
  subnet_id = "sub-cddd7ffa-be05-4698-9b3d-794e1adfcfce"
  secondary_subnets = ["10.200.27.0/24", "10.200.28.0/24"]
  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"
  }
}

```

```
}
```

 **Chú ý:**

- Để lấy image_id bạn mong muốn sử dụng, bạn có thể truy cập vào VKS Portal, chọn menu System Image và lấy ID mà bạn mong muốn hoặc lấy thông tin này tại [đây](#).
- Để lấy flavor_id bạn mong muốn sử dụng cho Node group của bạn, vui lòng lấy ID tại [đây](#).

Khởi chạy Terraform command

- Sau khi hoàn tất các thông tin trên, thực hiện chạy lệnh bên dưới:

```
terraform init
```

- Sau đó, bạn để xem những thay đổi sẽ được áp dụng trên những resource mà terraform đang quản lý bạn có thể chạy:

```
terraform plan
```

- Cuối cùng bạn chọn chạy dòng lệnh:

```
terraform apply
```

- Chọn **YES** để thực hiện việc khởi tạo Cluster, Node Group thông qua Terraform

Kiểm tra Cluster vừa tạo trên giao diện VNG Cloud Portal

Sau khi khởi tạo thành công Terraform, bạn có thể lên VKS Portal để xem thông tin Cluster vừa tạo.

Tham khảo thêm về cách sử dụng Terraform để làm việc với VKS tại [đây](#).

Một số lưu ý khi sử dụng VKS với Terraform:

Khi sử dụng **Terraform** để khởi tạo **Cluster** và **Node Group** trên hệ thống VKS, nếu bạn thay đổi một trong các field sau, hệ thống sẽ tự động xóa Node Group/ Cluster và thực hiện khởi tạo lại Node Group/ Cluster theo thông số mới tương ứng. Việc xóa sẽ được thực hiện trước khi tạo Node Group/ Cluster mới.

- Đối với resource `vngcloud_vks_cluster`, các field khi bạn thay đổi hệ thống sẽ xóa Cluster và tạo lại bao gồm:

- `name`
- `description`
- `enable_private_cluster`
- `network_type`
- `vpc_id`
- `subnet_id`
- `cidr`
- `enabled_load_balancer_plugin`
- `enabled_block_store_csi_plugin`
- `node_group`
- `secondary_subnets`
- `node_netmask_size`

- Đối với resource `vngcloud_vks_cluster_node_group`, các field khi bạn thay đổi hệ thống sẽ xóa Cluster và tạo lại bao gồm:

- `cluster_id`
- `name`
- `flavor_id`
- `disk_size`
- `disk_type`
- `enable_private_nodes`
- `ssh_key_id`
- `secondary_subnets`
- `enabled_encryption_volume`
- `subnet_id`

Để chỉ định hệ thống tạo cluster/node group mới rồi mới thực hiện xóa cluster/ node group cũ, bạn có thể thêm số `lifecycle { create_before_destroy = true }` vào file main.tf của bạn. Cụ thể:

- Đối với resource `vngcloud_vks_cluster`

```
resource "vngcloud_vks_cluster" "example" {  
    # ...  
  
    lifecycle {  
        create_before_destroy = true  
    }  
}
```

- Đối với resource `vngcloud_vks_cluster_node_group`

```
resource "vngcloud_vks_cluster_node_group" "example" {  
    # ...  
  
    lifecycle {  
        create_before_destroy = true  
    }  
}
```

Giám sát

Metrics

Bạn có thể cài đặt vMonitor Platform Metric Agent vào Kubernetes Cluster để thu thập và đẩy metric về vMonitor Platform site, sau đó sử dụng các tính năng tại vMonitor Platform để quản lý tập trung tài nguyên và theo dõi hoạt động bất thường của Kubernetes Cluster.

Cài đặt Metric Agent sử dụng Helm

Các bước chuẩn bị trước khi cài đặt

1. Kiểm tra bạn đã có Metric Quota và quota chưa chạm mức giới hạn, nếu chưa có bạn cần thực hiện mua Quota Metric tại [đây](#).

2. Tạo Service Account và gắn policy: vMonitorMetricPush để có đủ quyền đẩy Metric về vMonitor

Để tạo service account bạn truy cập tại [đây](#), sau đó thực hiện các bước sau:

- Chọn "Create a Service Account", điền tên cho Service Account và nhấn **Next Step** để gắn quyền cho Service Account
- Tìm và chọn **Policy: vMonitorMetricPush**, sau đó nhấn "Create a Service Account" để tạo Service Account, Policy: vMonitorMetricPush do VNG Cloud tạo ra chỉ chứa chính xác quyền đẩy metric về hệ thống
- Sau khi tạo thành công bạn cần lưu lại **Client_ID** và **Secret_Key** để thực hiện bước tiếp theo.

Cài đặt helm tại Debian/Ubuntu server

1. Bạn cần cài đặt Helm trên server **có kubeconfig chứa đủ quyền** để tương tác Kubernetes Cluster.

- Kiểm tra quyền bằng **command kubectl**:

Kube check permission

```
# Lệnh dùng để kiểm tra quyền tương tác tất cả resource tại namespace default
kubectl auth can-i '*' '*'
# Lệnh dùng để kiểm tra quyền tương tác tất cả resource tại namespace chỉ định,
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
```

- Nếu kết quả các command trên là **YES** thì bạn đã đủ quyền.

2. Tiến hành cài đặt Helm

- Thực hiện những command sau:

```
curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share
sudo apt-get install apt-transport-https --yes
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.
sudo apt-get update
sudo apt-get install helm
```

- Kiểm tra Helm đã cài đặt thành công

```
helm version
# Kết quả mong muốn của command
# version.BuildInfo{Version:"v3.11.0", GitCommit:"472c5736ab01133de504a826bd9ee1
```

Cài đặt Helm tại các hệ điều hành khác

Tham khảo hướng dẫn cài đặt tại đây: [Install Helm Through Package Managers](#)

3. Một số lưu ý thêm về Helm: (tham khảo chi tiết tại: [helm docs](#))

- Helm sẽ dùng kubeconfig để tương tác với cluster, mặc định helm sẽ dùng config ở đường dẫn: "~/.kube/config"
- Nếu cần thay đổi đường dẫn tới kubeconfig ta có thể sử dụng 2 cách:
 - Với mỗi command helm ta thêm option --kubeconfig: ví dụ helm install --kubeconfig <path_to_kubeconfig>
 - Khai báo biến môi trường: KUBECONFIG

Cài đặt Metric Agent

Mặc định khi cài đặt vMonitor Platform Metric Agent sẽ có 2 thành phần:

- **Deployment** Agent kube-state-metrics: thu thập metric từng resource của k8s cluster (pod, daemonset, deployment, replicaset,)
- **Daemonset** Agent: thu thập metrics từng node k8s cluster (CPU, Memory usage, ...)

1. Thêm Helm vMonitor Platform Repo

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

2. Cài đặt chart

- Kiểm tra và xóa các resource liên quan trước khi cài đặt để tránh đụng độ

```
# 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
```

- Cài đặt tại **namespace default** (thêm cờ **-n <namespace_chỉ định>** để cài đặt agent tại namespace khác)

```
helm install vmonitor-metric-agent vmonitor-platform/vmonitor-metric-agent \
--set vmonitor.iamClientID=<YOUR_CLIENT_ID_XXXXXXXXXXXXXXXXXXXX> \
--set vmonitor.iamClientSecret=<YOUR_CLIENT_SECRET_XXXXXXXXXXXXXXXX> \
--set vmonitor.clusterName=<CLUSTER_NAME>
```

- <YOUR_CLIENT_ID_XXXXXXXXXXXXXXXXXXXX>, <YOUR_CLIENT_SECRET_XXXXXXXXXXXXXXXX>: Thông tin service account đã tạo ở bước chuẩn bị
- <CLUSTER_NAME>: Thông tin này sẽ dùng để lọc các host của k8s cluster trong trường hợp có nhiều cluster để theo dõi
- Kiểm tra việc install agent thành công

```
# Chạy command và đảm bảo output là các pods ở trạng thái running
kubectl get pod | grep "vmonitor-metric-agent"
```

- Nếu các pods agent không ở trạng thái running, sử dụng command tương ứng để kiểm tra lỗi

```
# 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
```

Sau khi cài đặt hoàn tất, metric theo dõi kubernetes đã được đẩy về vMonitor Platform site , bạn có thể tiến hành lên vMonitor để vẽ các dashboard, widget.

Gỡ cài đặt Metric Agent

Thực hiện command sau để xóa các k8s resources liên quan đã cài đặt:

```
helm uninstall vmonitor-metric-agent
```

Cài đặt Metric Agent không sử dụng kube-state-metrics

- Cài đặt tại **namespace default** (thêm cờ **-n <namespace_chỉ định>** để cài đặt agent tại namespace khác)

```
helm install vmonitor-metric-agent vmonitor-platform/vmonitor-metric-agent \  
--set vmonitor.iamClientID=<YOUR_CLIENT_IDXXXXXXXXXXXXXXXXXXXX> \  
--set vmonitor.iamClientSecret=<YOUR_CLIENT_SECRETXXXXXXXXXXXXXXXX> \  
--set vmonitor.clusterName=<CLUSTER_NAME> \  
--set vmonitor.kubeStateMetricsEnabled=false \  
--set kubeStateMetricsAgent.enabled=false
```

Cách tính giá

Đối với VKS chi phí **Managed Control Plane** là hoàn toàn miễn phí. Bạn chỉ cần chi trả cho các resource khác mà thực tế bạn sử dụng bao gồm:

- Đối với **Public Cluster**, bạn cần chi trả cho:
 - Tất cả các node có trong Cluster (VM). Chi tiết cách tính giá của vServer bạn có thể tham khảo thêm tại [đây](#).
 - Load Balancer được integrated vào Cluster của bạn. Chi tiết cách tính giá của Load Balancer bạn có thể tham khảo thêm tại [đây](#).
 - Persistent Volume, Snapshot được integrated vào Cluster của bạn. Chi tiết cách tính giá của Volume bạn có thể tham khảo tại [đây](#).
- Đối với **Private Cluster**, bạn cần chi trả cho:
 - Tất cả các node có trong Cluster (VM). Chi tiết cách tính giá của vServer bạn có thể tham khảo thêm tại [đây](#).
 - Load Balancer được integrated vào Cluster của bạn. Chi tiết cách tính giá của Load Balancer bạn có thể tham khảo thêm tại [đây](#).
 - Persistent Volume, Snapshot được integrated vào Cluster của bạn. Chi tiết cách tính giá của Volume bạn có thể tham khảo tại [đây](#).
 - Ngoài ra, khi bạn tạo một private cluster, hệ thống tự động tạo 4 endpoints giúp kết nối với các dịch vụ khác trên VNG Cloud bao gồm:
 - **Endpoint** để kết nối tới dịch vụ **IAM** (Endpoint Name: vks-iam-endpoint-...)
 - **Endpoint** để kết nối tới dịch vụ **vCR** (Endpoint Name: vks-vcr-endpoint-...)
 - **Endpoint** để kết nối tới dịch vụ **vServer** (Endpoint Name: vks-vserver-endpoint-...)
 - **Endpoint** để kết nối tới dịch vụ **vStorage** (Endpoint Name: vks-vstorage-endpoint-...)

Việc sử dụng private cluster sẽ phát sinh thêm chi phí cho 4 private service endpoint này, nhưng nó mang lại nhiều lợi ích về bảo mật cho dự án của bạn. Bạn hãy cân nhắc kỹ lưỡng các yếu tố để đưa ra quyết định sử dụng public hay private cho cluster của mình.

 **Chú ý:**

- Để đảm bảo Cluster của bạn hoạt động ổn định, chúng tôi đã tự động thiết lập Auto-renew cho tất cả các resource trên Cluster của bạn. Trước ngày hết hạn của các resource, hãy đảm bảo số dư credit của bạn đủ để hệ thống có thể thực hiện Auto-renew thành công.

Tham khảo thêm

Danh sách Flavor đang hỗ trợ

Bên dưới là danh sách các Flavor đang hỗ trợ bởi 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-56c3aee5065
a1-highcpu-32×64-n10	32	64	flav-bf77e8d1-8ec4-42e1-aca4-bb4a703b67d8

- General Code S

eci.ins.s-general-2×4	2	4	flav-152e00ea-5412-44c5-868f-0b30e37e4f90
eci.ins.s-general-4×8	4	8	flav-f5e0201d-606f-4480-be72-de11f46d48ff
eci.ins.s-general-8×16	8	16	flav-aece130a-2a09-4d0d-b5eb-0c79589f1547
eci.ins.s-general-16×32	16	32	flav-4b949be1-e3ef-4d73-8ce4-cd264fee86df
s-general-2×4	2	4	flav-8066e9ff-5d80-4e8f-aeae-9e8a934bfc44
s-general-4×8	4	8	flav-3929c073-9da9-486f-a96f-9282dbb8d83f
s-general-8×16	8	16	flav-578e2030-154e-45db-ac5a-72c7c7f70e47
s-general-16×32	16	32	flav-4b686b68-a141-4a1a-a8f6-e54f710706bf
s-general-16×32-n10	16	32	flav-903a8d4a-066a-449f-b537-2fd889d79de4
s-general-8×16-n10	8	16	flav-65870741-34cd-4d78-89f5-63c4354ccfd8
s1-standard-2×8	2	8	flav-198daa13-4829-43b4-9e7e-445906388ffb
s1-standard-4×16	4	16	flav-2d84e9ce-ad64-4e1d-a623-b11326f229b4
s1-standard-8×32	8	32	flav-80681f16-3202-47c5-8387-011285199e64
s1-standard-12×48	12	48	flav-54007569-0943-47b7-adaf-7ab672bd96a5
s1-standard-16×64	16	64	flav-eab7d53c-f711-40fe-90fc-2430d529742f
s1-standard-32×128	32	128	flav-b0f73b54-06b3-4743-9e11-734d0fab0a37
s1-standard-64×256-n10	64	256	flav-e55076b2-6c75-446f-a35c-51c70e6344f6

s1-standard-48×192-n10	48	192	flav-3519fdf6-7387-4ff4-9323-3c0d4b3ca431
s1-standard-32×128-n10	32	128	flav-1271e7a6-c15b-4002-9aba-077d81126902
s1-standard-16×64-n10	16	64	flav-59f8bb49-1ef1-4425-be92-414eb80465d8
s1-standard-8×32-n10	8	32	flav-3c0a65d2-775f-4c15-826e-3f41ff8c7900
s1-higmem-2×16	2	16	flav-e3ded003-043f-4fdf-8d2f-2d98fba7bb9f
s1-higmem-4×32	4	32	flav-23385d8e-1abe-4897-acf5-f2dc25f9d04a
s1-higmem-8×64	8	64	flav-bcae1721-a43c-4752-a3da-9f53ee33b297
s1-higmem-16×128	16	128	flav-6af6448-48e8-4f3e-a7e7-12f0fc357564
s1-higmem-32×256	32	256	flav-c0986513-5863-456a-8e85-5fb217e37934
s1-higmem-32×256-n10	32	256	flav-1df93b8c-a39f-4298-ad0a-6cacd0ba86e4
s1-higmem-16×128-n10	16	128	flav-3bc652a4-99a6-4b1c-8676-dc8a35ef3a70
s1-higmem-8×64-n10	8	64	flav-34bc77ca-3887-4a52-a068-ac0160be7cf
s1-highcpu-4×4	4	4	flav-6d08591d-292f-4220-9ac2-6db0448c78ba
s1-highcpu-8×8	8	8	flav-9ef5b92c-40fc-46a3-b8d0-81f6d602c997
s1-highcpu-32×32	32	32	flav-2b9063fc-799f-4dc6-ad25-6f279b212785
s1-highcpu-32×64-n10	32	64	flav-f50dddce-3ab9-426f-8fa1-0d4f84b4e4db
s1-highcpu-16×16-n10	16	16	flav-5da432a8-8b26-4db4-bcb3-14236e3e1521

s1-highcpu-8×8-n10	8	8	flav-bc0a9310-dbef-4206-9d3e-44c4780a222a
eci.ins.s1-standard-2×8	2	8	flav-afef7157-38ac-49e0-8329-327f1559733e
eci.ins.s1-standard-4×16	4	16	flav-bb0bc0db-2710-44a1-933e-2b9e1fb99d6e
eci.ins.s1-standard-8×32	8	32	flav-d83e3d9d-83a6-4935-aca7-e7891a415a51
eci.ins.s1-standard-16×64	16	64	flav-4ee7c1db-ad75-4b0a-9f8a-3e43f0b27df3
eci.ins.s1-standard-32×128	32	128	flav-12f11d02-ab75-4157-8a14-f7be6f14095e
s1-highcpu-16×16	16	16	flav-a3be1208-a778-4cc5-9777-30d09fc41179
s1-highcpux2-32×64	32	64	flav-be97bcad-9b37-4570-ad06-ae7d9786e4fb
eci.ins.s1-highmem-2×16	2	16	flav-49528db1-71d2-452f-bf75-e48c7ae0922f
eci.ins.s1-highmem-4×32	4	32	flav-a3b67f38-dd45-47e3-8755-273b6b1e506f
eci.ins.s1-highmem-8×64	8	64	flav-54a945d5-a811-4c55-a22f-6b3338223794
eci.ins.s1-highmem-16×128	16	128	flav-bf0dc4ef-3262-4730-a1fb-14eb27ba1fdb
eci.ins.s1-highmem-32×256	32	256	flav-50d0f16b-cc73-4301-ae6e-07ef3b538a6c
eci.ins.s1-highcpu-4×4	4	4	flav-4988e043-2cee-4e64-bf8b-e97171a66045
eci.ins.s1-highcpu-8×8	8	8	flav-e87c9a15-e1bc-4ee1-a25d-389dbadc4325
eci.ins.s1-highcpu-16×16	16	16	flav-4d67306c-6fd2-4363-85ac-98e4319d7eb6

- GPU Code G

Flavor Name	CP U	Memory	CPU	Flavor ID
g1-standard-4×16-1rtx2080ti	4	16	1	flav-ae002cb3-41ac-417f-ad40-fb0a
g1-standard-8×32-1rtx2080ti	8	32	1	flav-f73d71b9-e269-47e0-af78-3be6
g1-standard-8×32-2rtx2080ti	8	32	2	flav-c8313a57-b53a-4970-b4a0-634
g1-standard-16×64-2rtx2080ti	16	64	2	flav-c8f87213-8a35-47d3-aec1-29ab
g1-standard-16×64-4rtx2080ti	16	64	4	flav-3bd899f0-0725-4a78-a5a1-9d0
g1-standard-32×128-8rtx2080ti	32	128	8	flav-c370e9ab-5c74-4e86-982b-0d9
g1-standard-8×64-1rtx2080ti	8	64	1	flav-b9febd61-07fd-46eb-9ead-f084
g1-standard-8×64-2rtx2080ti	8	64	2	flav-52a090f3-5a64-48c3-a47e-4e1

- GPU Code RTX4090

g2-standard-32×128-4rtx4090	32	128	4	flav-79d5efbf-91bd-442b-9919-f7d7dd59761a
g2-standard-16×128-2rtx4090	16	128	2	flav-a9bfd432-0b9f-481a-a6ce-8c924dd08ec3
g2-standard-16×64-2rtx4090	16	64	2	flav-fc2c0e0b-8ecb-4039-aa98-11bccc982d35
g2-standard-32×64-2rtx4090	32	64	2	flav-6b11c1c0-2631-4ce8-8184-1f94859518cb
g2-standard-16×64-1rtx4090	16	64	1	flav-2a2286db-503d-4e0e-8cdf-b994475865d3

Danh sách System Image đang hỗ trợ

Bên dưới là danh sách System Image mà hệ thống VKS đang hỗ trợ:

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