**Introduction**

**HyperCX Overview**

REDrackX HyperCX Bento is a hyper-converged Cloud platform designed to provide Infrastructure as a Service (IaaS). REDrackX started as a Private Cloud provider, but nowadays, it also provides Public Cloud based on the same technology leveraged on HyperCX Private Clusters. Users can request REDrackX for private clouds, or register and start using resources from REDrackX own managed public cloud. Besides this, owners of a Private HyperCX Cluster can convert them into a Public Cluster without intervention from REDrackX.

**Note**

REDrackX HyperCX Bento is the newest HyperCX architecture, REDrackX no longer maintains the original HyperCX stack. In this documentation, the term HyperCX alone will always refer to HyperCX Bento.

Being a hyper-converged platform, a REDrackX HyperCX cluster integrates the following components:

· Network Subsystem: Provides connectivity to all cluster components and some workloads. Constitutes physical components as well as virtual components. These components can vary depending on the specific deployment. When REDrackX deploys in one of its data centers, REDrackX own managed switches will be used, but this is optional when deploying on-prem for a user. In the case of the firewall, something similar happens. By default, REDrackX recommends using its own virtual routers. These devices will provide connectivity to all the critical cluster components and the default virtual network that is created using private IPs. These virtual routers can be fully replaced by physical firewalls if desired, or use together to physical firewalls.

· Management Subsystem: HyperCX leverages an extended monitoring system tightly integrated to all its components. This system will send monitoring data to REDrackX central management servers, allowing REDrackX to monitor the health of every cluster constantly. Besides this, a local version of this monitoring system will be installed locally only on Bento clusters, accessible to the cluster's administrator to check advanced metrics not available through the orchestrator. This system offers:

* + Proactive and reactive monitoring.
  + Failure, accounting and performance monitoring.
  + Active and passive probes.
  + Optional email notifications.
* Storage Subsystem: A Software Defined Storage (SDS) solution deployed as a Storage Area Network (SAN) is used with the following characteristics:
  + Bare metal installation.
  + Replicated data.
  + Optimized performance for VMs and containers.
* Orchestration Subsystems: This subsystem provisions and manages resources on the cluster. It also handles more advanced features like High Availability.
* Compute Subsystems: HyperCX being a hyper-converged platform, shares the same cluster between workloads and the rest of the subsystems (storage, management, compute and orchestration). The compute subsystem offers KVM Virtual Machines and LXD containers. KVM Virtual Machines are mandatory and will be available on every HyperCX cluster. LXD containers are optional.

So far, we have mentioned HyperCX Bento and miniBento. HyperCX Bento is REDrackX flagship architecture designed to handle big and critical deployments. It is fully redundant, and the smallest configuration starts on three nodes. There are use cases where the smallest deployments are needed, like Edge Computing and development applications. HyperCX miniBento is more suited for these cases since it can be built on top of a single node (or more). To make it as lightweight as possible, High Availability is not supported on miniBento deployments. The local monitoring system is only available on HyperCX Bento configuration, although both Bento and miniBento will be remotely monitored. The last major difference is that HyperCX Bento is designed to be scaled out up to hundreds of nodes, but this is not the case for miniBento. Notice then, in this case, the limiting factor is not the technology, but the lack of redundancy. Since HyperCX stores all the workloads on a SAN, and there is no redundancy on miniBento if any server or any drive fails, most of the information can be lost or corrupted, so it is not feasible to scale-out miniBento instances since the more hardware running the more chances of failure. The following table summarizes the differences between both:

| **Feature** | **Bento** | **miniBento** |
| --- | --- | --- |
| HA | Yes | No |
| Minimum amount of nodes | 3 | 1 |
| Integrated Monitoring System | Yes | No |
| Remotely monitored | Yes | Yes |
| Scalable design | Yes | Partially |
|  |  |  |

For any Cloud Deployment model offered by REDrackX, the hardware can be installed on-prem (inside the user's facilities) and off-prem (hosted in one of REDrackX data centers). Each approach has advantages and disadvantages, some of those are summarized next:

| **On-prem** | **Off-prem** |
| --- | --- |
| Hardened security | Support |
| Low latency | Reliability |
| Less Internet BW required | Faster deployments |
|  |  |

Regarding the installation mode, two options are available: Full Stack and Software Only. The only difference is related to the hardware provider. A HyperCX cluster delivered by REDrackX, including the hardware, is a full-stack configuration. On the other hand, the user can provide the hardware, and REDrackX will build the cluster. It is called a software-only configuration. Full-stack configurations can be deployed inside REDrackX data centers (off-prem) or be shipped to different locations (on-prem). HyperCX software only clusters, on the other hand, can only be built off-prem, inside the user's facilities.

The following documentation is designed for HyperCX users. Most of this documentation, except for the **Public Cloud** section, is meant for administrators of a Private Cluster. This configuration exposes all of REDrackX HyperCX features, which are considerably more, hence why there is a single section for REDrackX Public Clouds.

If the goal is to quickly start spinning up virtual instances, you should go to the [Quick Start Guide](https://docs.virtalus.com/QuickStart/QuickStart)

# Quick Start guide

## Overview

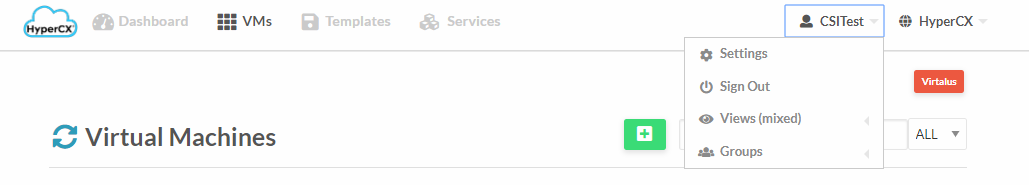
HyperCX can be set up with just a click of a button, and it’s that easy. Everything else has already been taken care of, although for full control over a Virtual Machine (VM), it’s images, and it’s context, we recommend understanding VM templates and images. But if you need a VM up and running with few or no modifications, you can follow this guide without worrying about that part.

**Note**

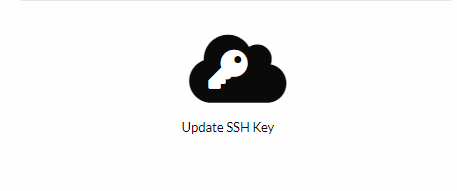
Some HyperCX clusters are only accessible from inside a VPN. If this is your case, make sure you are connected using the provided VPN configuration file. For detailed steps on how to connect to VPN [learn more](https://docs.virtalus.com/Misc/Others/#connect-to-openvpn-windows)

## Getting Started

1. (Optional) With HyperCX users can utilize password less SSH (Only for Linux), for windows a password must be provided when launching the VM. If you intend to launch a Windows VM you can skip this part. All a user needs to do is navigate to settings



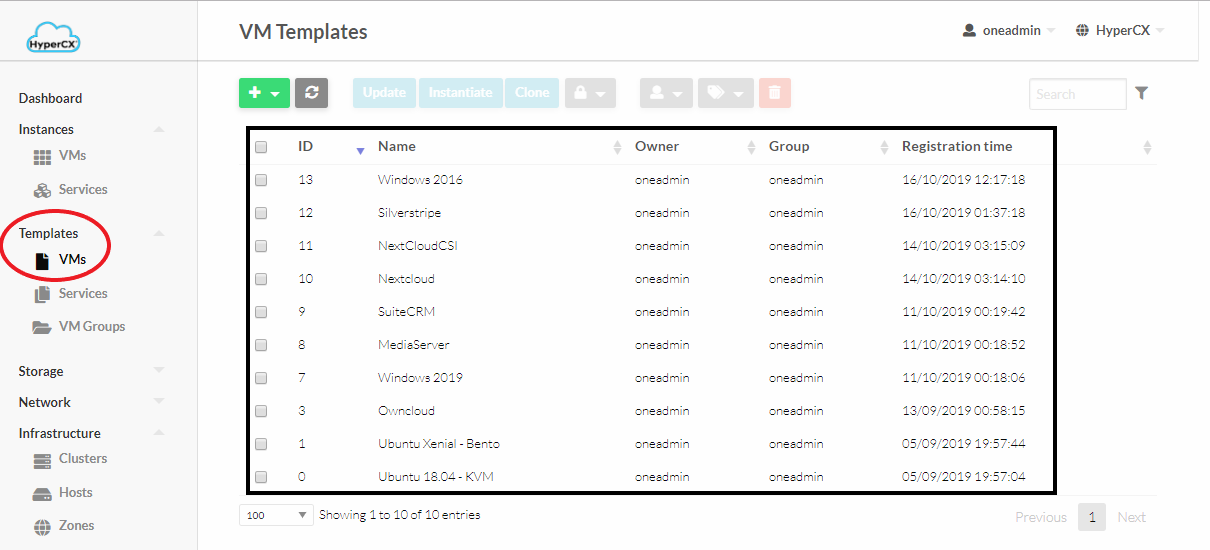
1. Then navigate to update SSH key



Add their public SSH key then click on update SSH key. From now on, this public key will be configured inside the **root** user for each new Linux VM or container deployed.

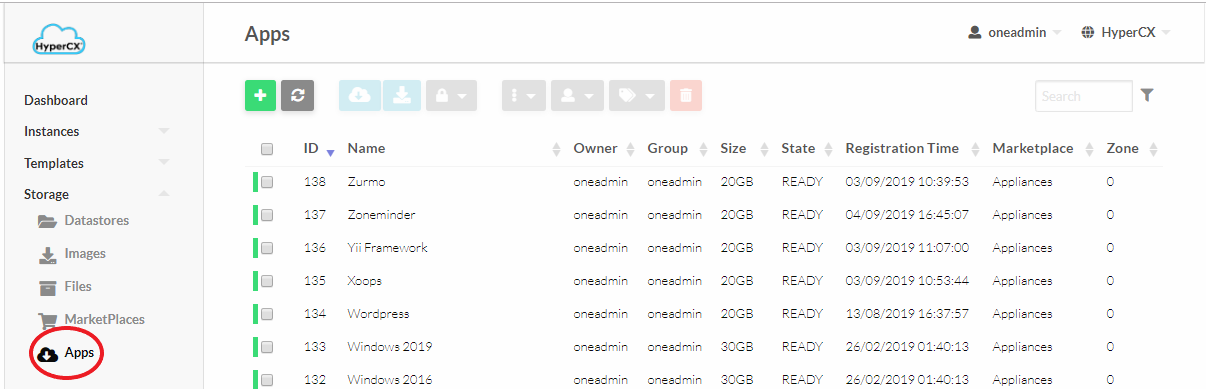
### Downloading a template to HyperCX

VMs are created from templates. If this is a new cluster, chances are the needed template will not be available (although some clusters come with some pre-downloaded templates, like in the case of Proof of Concept (PoC)). You can check if you’re template is already available by heading over to templates->VMs

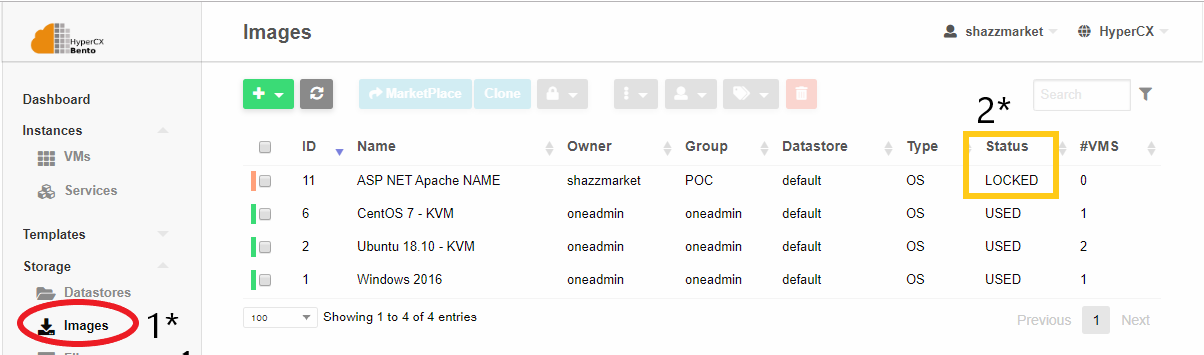


If your desired template is available you can head over to the next subchapter. If not, you will need to download the required template or appliance from one of REDrackX marketplaces.

1. Appliances can be accessed under Storage 🡪 Apps.



1. Once in the Apps UI select an existing appliance and click on the appliance to view its description.
2. Now click on https://docs.virtalus.com/images/QS6.png icon to start the download template creation process.. This process will download the OS image and create a new template.
3. Enter the template name, its Image name and click onhttps://docs.virtalus.com/images/QS7.png.
4. Wait for the Image to be downloaded. It’s status can be checked by accessing Storage -> Images



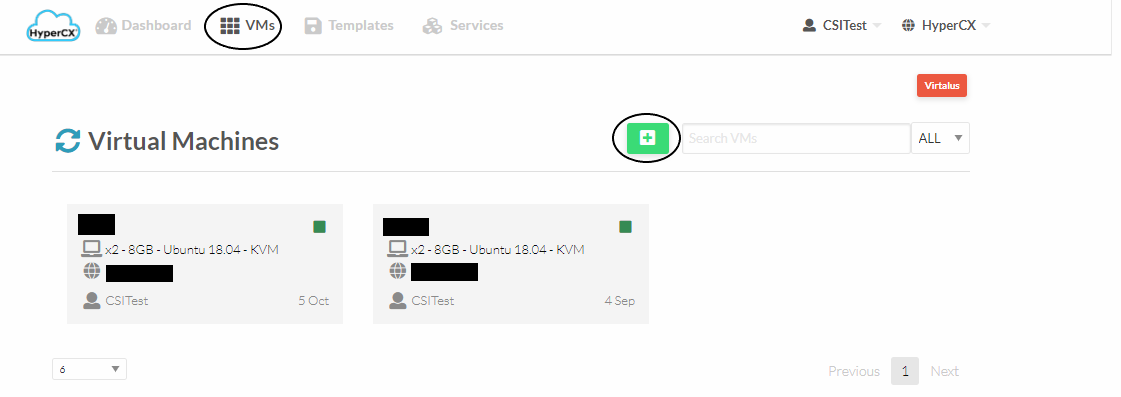
The status of the image will be LOCKED while the download is in progress. Once the image is downloaded its status is changed from LOCKED to READY, the new template will be ready to be instantiated.

[Click here](https://docs.virtalus.com/UserGuides/TemplateManagement/) to learn more about templates and images.

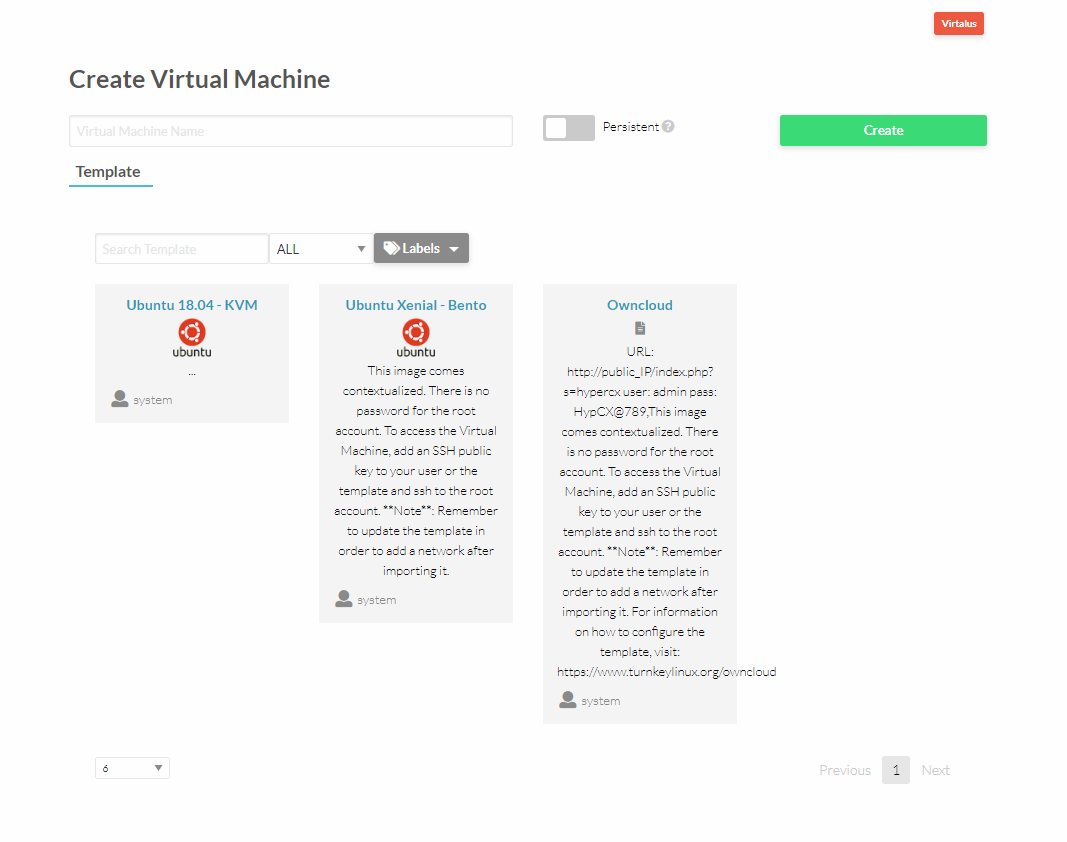
### Instantiating a VM

Instantiating a VM can be easily done by,

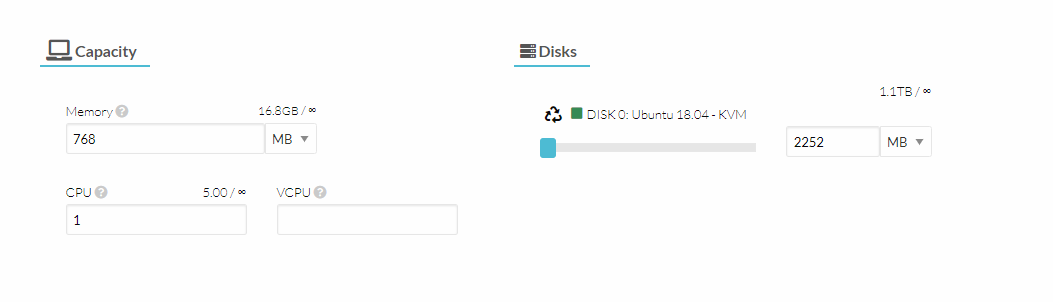
* Navigate to VMs and then clicking on the add button.



* Now select a predefined template that you want to use, it can be an OS or prebuilt image with everything preinstalled. Now enter a name for the virtual machine(This is optional and can be left blank)

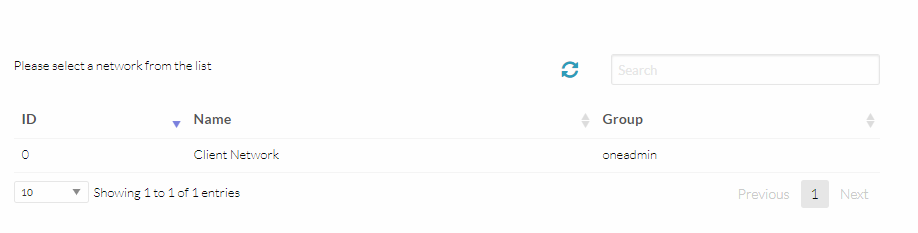


* Once the VM template is selected just specify capacity of the VM.



* Now carry on to the network part and attach the required network interface.





* Now scroll back up and click on the create button that’s all required to instantiate a VM.

### Accessing the VM

Determine the IP address of the VM by navigating to VMs and then under your newly created VM IP address will be mentioned.

#### Linux VM

Linux VMs can be accessed using password-less ssh (if the public key was previously defined). If a password were specified, this would be the root password, but ssh access using the root user and a password is not allowed for security reasons. If this is your case, you must use the VNC screen and either remove this restriction from the VM's SSH server or create a new user (recommended).

#### Windows VM

Remotely access the VM by using Remote Desktop Protocol (RDP) using the IP address determined earlier.

#### VNC

Users can also get a direct display of the VM using VNC by navigating to VMs and then clicking on the respective VM, now click on https://docs.virtalus.com/images/QS14.png

For more detailed information about how to manage VMs and it’s templates follow our documentation.

## FAQ

* Q: I don’t know the password of my windows or Linux machine`

A: Your virtual instance (VM o container) will only contain a password if this was specified when instantiating the instance. If not, there is no password associated with the instance. Windows VMs require a password; if not, they will not be able to access them later on via VNC or RDP. For Linux VMs, a password is optional, although public keys are recommended. If a password was set, it would be found on the VM configuration parameters.

# Disaster Recovery

## Overview

Disaster recovery (cloud DR) is a backup and restore strategy that involves storing and maintaining copies of electronic records in a cloud computing environment as a security measure. The goal of cloud DR is to provide an organization with a way to recover data and or implement failover in the event of a human-made or natural catastrophe.

## How it works?

Disaster recovery is leveraged by HyperCX in a way that user can continue working on their crucial data without even worrying about any failure that could disrupt their services. HyperCX at every scheduled day and time of the week backs up images of the VMs to remote clusters. If a disaster occurs VMs at the remote clusters are automatically deployed.

## Setup Backup for Disaster Recovery

Following are key components and definitions to understand how to implement Disaster recovery using HyperCX

**Remote Cluster**: This will serve as the backup cluster this cluster is on a remote location to prevent data loss.

**Present Cluster**: In this document present cluster refers to the primary cluster allocated to the client.

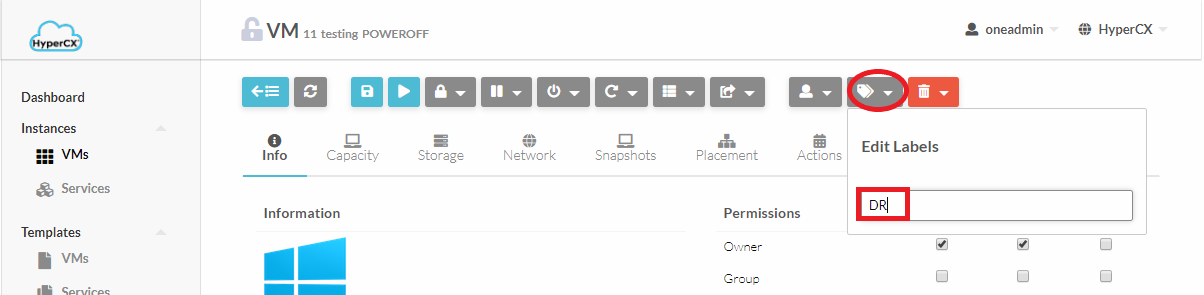
**Tags**: VMs that are enabled to backup for disaster recovery are tagged with “DR” (This tag may or may not be the same please contact support to verify it)

**Template**: On the remote cluster a VM template needs to be created in order for the cluster to be ready to be backed up.

### Getting Started

The VM that needs to be backed up should be set up first in order to achieve automated backups to the remote server, follow the below mentioned steps to successfully configure it.

1. Navigate to the VM that needs to be backed up and select it.
2. Now add a label “DR” to the VM.

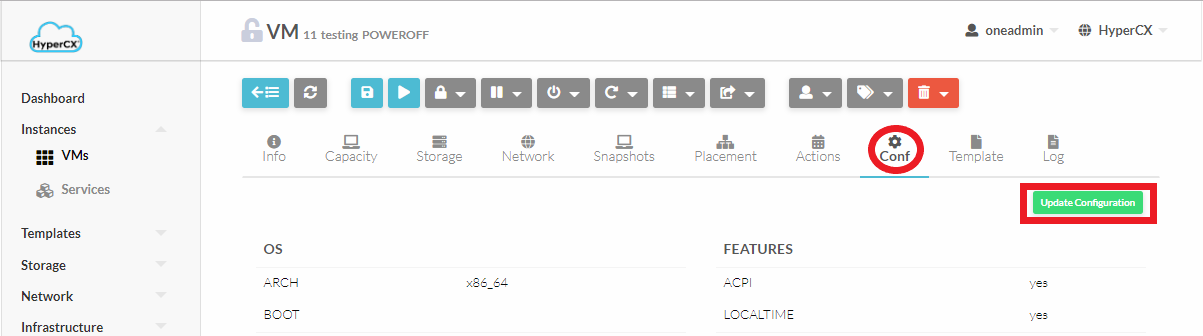


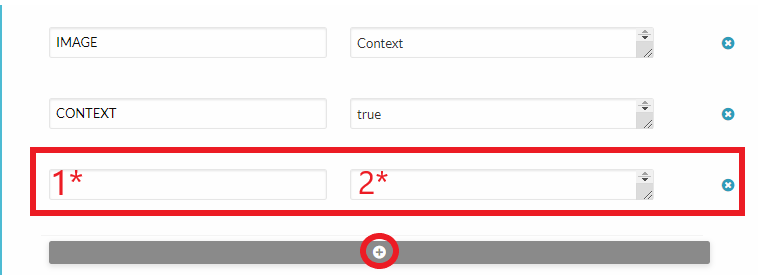
### Creating backup template

1. Now head over to the Remote Cluster’s portal and Login to that cluster
2. On the remote cluster, a template must be created for each VM that will leverage DR.
3. So create a new template by heading over to Templates-> VMs and click on (ADD button) and now click on create.
4. Set a unique name for this template (It will be used later on).
5. Navigate to Scheduling tab and under Placement Tab select “Select Clusters” now depending upon your Primary Cluster’s VM select either KVM or LXD cluster.
6. The configurations of this template should be the exact same as of your Present Cluster’s VM. Following configurations are mandatory rest is up to the user.
   1. VM Capacity (CPU, Memory, VCPU etc.)
   2. Network
   3. Storage should not be added as it will be automatically backed up by HyperCX from Present Cluster to Remote Cluster.

### Some final configurations..

1. Now head back to Present Cluster’s HyperCX portal and power off the VM that needs to be backed up.
2. Navigate to the Conf tab of the VMs and click on Update configurations.



1. Now navigate to Context tab and select Custom Vars and click on the add button(At the end of all context variable definitions)
2. Here we’re going to add a context variable with key “DR\_REMOTE\_TEMPLATE” and value will be the name of template (Step 6) we defined earlier at the remote cluster. If you’re already familiar with the key value pair configuration that HyperCX uses you can skip the following.
3. On the first input box (1\*) enter “DR\_REMOTE\_TEMPLATE”
4. On the second input box(2\*) enter the name of the template at the Remote Cluster

## Conclusion

HyperCX leverages delta transfers for backup (Only changes are backed up) this reduces bandwidth overhead, providing reliable backups.

## FAQs

Q: Remote cluster’s VM does not have the same network as of Primary Cluster

A: This outcome is expected as every network is unique to their geographical location(except for client network) so all a user needs to do is access the VM using the IP address that is allocated on the remote cluster.

# HyperCX Vault

## Overview

HyperCX vault is a backup solution that keeps versioning of VM’s images. HyperCX vault enables users to make incremental backups and manage those backups. If something goes wrong user can have a 90 day backup to recover from.

### How it works?

HyperCX vault is a backup server which resides on the same geographical region but on a different physical server so in case of any failure, user can revert back to their previous data and carry on with their important tasks with the VM.

### Setup HyperCX vault

To setup HyperCX vault all a user needs to do is add a tag to their important VMs so that HyperCX identifies those VMs and sequentially backup them. Following are the required steps to enable HyperCX vault

1. Navigate to the VM that needs to be backed up and select it.
2. Now add a label “Backup” to the VM.(The name of this label may or may not be “Backup” please consult REDrackX Support to confirm)

And that’s all you need to do to setup sequential backups.

### FAQs

**Q**: Is HyperCX vault an alternative for Disaster recovery?

**A**: No, HyperCX vault offers backup and restore features but in case of any uncontrollable and unavoidable circumstances (Natural disasters) all your important data might not be safe. HyperCX also offers disaster recovery which backs up user’s crucial data on a remote cluster.

# HyperCX NFV

## Overview

Network functions virtualization (NFV) is the process of decoupling network functions from hardware and running them on a software platform. Although it could be a complementary approach to software-defined networking (SDN) for network management, it is different. While both manage networks, they rely on different methods. While SDN separates the control and forwarding planes to offer a centralized view of the network, NFV primarily focuses on optimizing the network services themselves. It is a key part of cloud computing, data center networking, SD-WAN, and many others.

NFV began when service providers attempted to speed up the deployment of new network services to advance their revenue and growth plans. Developers found that proprietary hardware-based appliances limited their ability to achieve these goals. They looked to standard IT virtualization technologies and found that virtualizing network functions accelerated service dynamics and provisioning.

HyperCX NFV is designed to easily provide VNFs (Virtual Network Functions) to HyperCX cloud users. It is not designed to be a competitor of advanced, carrier-grade NFV architectures like MANO, or be feature-rich like software firewalls like PfSense. Instead, simplicity and ease of use are key design goals while maintaining good performance for small and medium cloud deployments.

HyperCX NFV has been pre-loaded with the following features:

* HA support using VRRP protocol through keepalived.
* Routing among connected Virtual Networks.
* VPN support leveraging OpenVPN.
* Support for masquerade on private networks.
* L4 and L7 load balancing features.
* Site to Site (S2S) VPN support leveraging StronSwang.

All of these features can be used at the same time on a single instance, or only one per HyperCX NFV instance. Next it will be shown how to deploy a basic NFV appliance and after that each feature will be described.

**Note**

HyperCX NFV is deployed and managed like a Virtual Router. In this document, the term Virtual Router or vRouter can also be used to refer to HyperCX NFV instances.

## Obtaining the NFV appliance

NFV appliances are available since HyperCX 10.5.2 and can easily be obtained on the marketplace. The appliance name on the marketplace is **HyperCX\_NFV** and the version can be seen on the details.

### Upgrading the NFV appliance

The version of the HyperCX NFV appliance that is being used on the cluster can be seen when login in to any VM instance via SSH or VNC. To check the version of the appliance in the marketplace, simply open the appliance under **Apps** tab and check the details.

If the market's version is newer, you might want to upgrade your current version. In order to do this, simply download the new appliance from the market and destroy your old local appliance (template + images). Note that you will not be able to destroy a template if VMs are running from it.

## Deploying an NFV appliance

Ease of use is one of the core principles around HyperCX NFV. An instance can be created in standalone mode (no support for HA) or HA mode. The chosen deployment strategy should be based on how critical these services will be. Note that HyperCX public clouds and HyperCX Bento clusters use redundancy on all the major components, providing full HA infrastructures. Still, a server failure (which is an isolated event since every hardware component is proactively monitored) will bring a downtime of 6-8 minutes to all the instances that were running on that server. This is the time that will take the system to detect the failure, validate it is not a false positive, and migrate all the instances.

HyperCX NFV appliances are deployed and managed from **Instances** --> **Virtual Routers**.

### Deploying in standalone mode

To deploy a single, non-HA, instance of HyperCX NFV, create a new **Virtual Router** appliance.

Leave in blank the **Keepalive** parameters since they only make sense for HA deployments.

Add all the networks that will require service from HyperCX NFV. Note that these networks will be routed among them. Like every HyperCX virtual instance, this appliance will use the gateway from the first NIC (or VNET) defined. This is very important when configuring the network interfaces. For example, if you intend that the internal (private) networks are routed to the internet (most common case), the first Virtual Network should use a public virtual network defined with a public IP and a gateway.

Select your HyperCX NFV template and set **1** under **Number of VM instances**.

### Deploying in HA mode

For HA deployment a few parameters can be leveraged, which made no sense on standalone instances:

* **Keepalive service ID**: Defines Keepalived **virtual\_router\_id** parameter. This parameter will be the same among all the VM instances of the appliance. If left blank, the vRouter ID will be used (recommended).
* **Keepalive password**: Defines a common password for all the instances to communicate via VRRP. If left blank, no password will be used and any other Keepalive instance will be able to join the cluster without authentication.

When selecting the networks, a floating IP will be needed on each NIC that will leverage HA. In order to do this, insert the IP under **Force IPv4** (or v6) and click **Floating IP**. This setting will provide an IP obtained from HyperCX's IPAM to each instance, and the defined IP will also be used on the master keepalived instance. A floating IP can be used on each different virtual network, but depending on the use case, not all the virtual networks actually need a floating IP. This will be covered in more detail later on.

If **Floating IP** is not selected, several instances will be deployed with a single IP on the specific NIC, no floating IP will be configured on that NIC so Active-Passive HA will not be supported on the selected interface. If no IP is specified but **Floating IP** option is checked, the floating IP will be automatically assigned from HyperCX internal IPAM.

Under **Number of VM instances**, select 2 or more. No more than 2 instances are needed or recommended.

### Getting information

After the virtual router is deployed, you can get information like the version number and all the enabled VNFs and their configuration by logging in to any VM instance. A message similar to this will appear:

\_ \_ \_\_\_\_\_\_ \_\_ \_ \_ \_\_\_\_\_\_\_ \_\_

| | | |\_ \_ \_ \_\_ \_\_\_ \_ \_\_ / \_\_\_\ \/ / | \ | | \_\_\_\ \ / /

| |\_| | | | | '\_ \ / \_ \ '\_\_| | \ / | \| | |\_ \ \ / /

| \_ | |\_| | |\_) | \_\_/ | | |\_\_\_ / \ | |\ | \_| \ V /

|\_| |\_|\\_\_, | .\_\_/ \\_\_\_|\_| \\_\_\_\_/\_/\\_\ |\_| \\_|\_| \\_/

|\_\_\_/|\_|

VERSION: 1.0

=========================VRRP========================

Interface ETH0 using floating IP:

Interface ETH1 using floating IP: 10.10.0.180

Keepalived ENABLED

=========================MASQUERADE========================

IPV4 MASQUERADE ENABLED VIA ETH0

IPV6 MASQUERADE DISABLED

=========================LOAD\_BALANCER========================

**LOAD** BALANCER ENABLED.

**Load** Balancer backend 1: 10.10.0.123

**Load** Balancer backend 2: 10.10.0.124

**Load** Balancer backend 3: 10.10.0.125

**Load** Balancer backend 4: 10.10.0.32

*--------------------------------------------------*

**LOAD** BALANCER **AUTHENTICATION** ENABLED

**Load** Balancer **monitoring** portal **URL**: **http**://server\_address:8989/stats

**Load** Balancer **monitoring** portal **user**: **admin**

**Load** Balancer **monitoring** portal **password**: **password**

*--------------------------------------------------*

=========================OPENVPN========================

OpenVPN ENABLED.

Created OpenVPN **account** **for** **user** user1

Created OpenVPN **account** **for** **user** user2

Detected internal network: 10.10.0.21 255.255.0.0

Detected internal network: 192.168.80.0 255.255.255.0

**Last** login: Tue Feb 4 23:45:14 2020 **from** 189.203.29.206

In addition, it is possible to obtain information in the **Info** section within the vm of **HyperCX NFV**. Information can be identified by **Service\_Name\_STATUS** and **Service\_Name\_INFO**.

## Virtual Network Functions

Next, every VNF supported by HyperCX NFV will be explained.

### Masquerade

This VNF provide Internet access to the internal Virtual Network. If the first Virtual Network defined on the virtual router uses a public IP, masquerade will be automatically enabled through this interface. No extra configuration is needed. HyperCX leverages **iptables** for the masquerade VNF.

#### Deployment considerations

Masquerade VNF will require that every VM uses the vRouter's IP as gateway. For HA deployments, a floating IP must be used on each internal network, but it is optional for the public network. The VMs inside the virtual networks will use the active vRouter's public IP as source IP for the masquerade.

### L4 Load Balancing

HyperCX uses HAProxy to provide load balancing features. This is used to balance all kind of applications, except those that use https, since it is not supported to handle the certificate. If this VNF is used, HAProxy will be configured with the following considerations:

* HAProxy will work in reverse-proxy mode, the backend servers see its IP address as their client address. This is easy to configure and manage since the backends will not need to configure the load balancer as gateway, and the client will not get a response from a different IP to which it made the request. This is sometimes annoying when the client's IP address is expected in server logs, so some monitoring features or sessions based on source IPs would be limited.
* To overcome the previous limitation, the well-known HTTP header "X-Forwarded-For" is added to all requests sent to the server. This header contains a value representing the client's IP address. Since this header is always appended at the end of the existing header list, the server must be configured to always use the last occurrence of this header only. This is something easy to implement on the backend application.
* Client requests will be maintained to a single server. This is useful for web applications that rely on sessions.
* Backends are monitored via **tcp-checks**. This means that, as long as the port is open and listening for connections, the backend will be marked as healthy and will receive requests.

#### Deployment considerations

Load Balancing VNF does not require that the backend VMs use the vRouter's IP as gateway.

There are two main strategies regarding the virtual networks that will be used:

* The vRouter will use at least two virtual networks. The frontend will listen into one while the backends are located on the second one. A good case scenario is to use a public IP on the first vNet and any IP on a private network where the backends are located. This will allow to publish a web app to the Internet without the need of an extra firewall performing NAT features.
* The vrouter will use a single virtual network. On this configuration, the frontend listens on the same virtual network where the backends are located. This is easier to setup and a common scenario could be a frontend located on Network X that connects to a few API servers located on the same network via the load balancer.

For HA deployments, a floating IP must be used on the frontend network and requests must be directed to the floating IP only. Configuring a floating IP on the backend's network is not necessary.

#### Configuration

Only two parameters are mandatory to enable the load balancing VNF. During instantiation, under **Custom attributes**. If both of the following attributes are filled, the load balancer will be enabled.

* **LB BACKENDS**: IPs of the backend servers separated by spaces. Set the IP of every backend server you intend to use separated by a space. Ex: 10.10.0.123 10.10.0.124 10.10.0.125
* **LB PORT**: Set the port that will be used by the backends. Only one port is supported at this time. This port will also be used by the frontend to listen for incoming requests.

#### Monitoring

HAProxy supports a monitoring web page that shows information of all the backends. This feature is disabled by default. In order to enable it, these two parameters must be filled:

* **LB USER**: Username for the load balancer's management portal
* **LB PASSWORD**: Password for the load balancer's management portal

After inserting the previous information, the management portal can be reached using the following URL:

http://$HyperCX\_NFV\_IP:8989/stats

### VPN

HyperCX NFV provides VPNs based on OpenVPN. Some considerations on this VPN are as follows:

* CA certificate is the same on all HyperCX NFV appliances. User's authentication is only performed via password via a custom authentication plugin developed by REDrackX.

#### Deployment considerations

VPN VNF will require that every VM uses the vRouter's IP as gateway. For HA deployments, a floating IP must be used on the public network and each internal network. This floating IP is the one that must be defined on the .ovpn configuration file.

#### Configuration

In order to enable the VPN VNF, a public network with a public IP must be set up as the first Virtual Network (just like the masquerade VNF). Besides of this, a configuration parameter is mandatory:

* **VPN CREDENTIALS**: User and password for each VPN user. This will configure all the users that will be able to use the VPN and their credentials. Each user will be configured with it's password separated by a colon in the format: "username:password". Several user/password pairs can be inserted separated by spaces. Ex: user1:pass1 user2:pass2 user3:pass3

After the appliance is running, you can get the .ovpn configuration files under **/root/vpn\_config\_files** on any instance. Alternatively, you can copy the following configuration file:

dev tun

cipher AES-256-CBC

persist-key

persist-tun

status openvpn-status.log

verb 3

**explicit**-exit-notify 1

client

auth-user-pass

auth-nocache

remote $PUBLIC\_IPV4\_ADDRESS 1194 udp

<ca>

-----BEGIN CERTIFICATE-----

MIIDQjCCAiqgAwIBAgIUKHe8HaisIp9iK7BoWrF3PrWET7MwDQYJKoZIhvcNAQEL

BQAwEzERMA8GA1UEAwwIQ2hhbmdlTWUwHhcNMjAwMjAzMDUxNTI1WhcNMzAwMTMx

MDUxNTI1WjATMREwDwYDVQQDDAhDaGFuZ2VNZTCCASIwDQYJKoZIhvcNAQEBBQAD

ggEPADCCAQoCggEBANZWy4vULsjfqsbUuj+S7oJ51EO1rhBC18cIa4ehgJegUGH7

Sz3BOE9uDVBy585pIwdta3KAkxV6rgdxEo23CfoUn+ibHoZtri3FUZJ+Rur5vyNF

jGj1AI8GDbAW7K11rhSvfLneNY+Ia/6/uJG+Wa28zmM1scC9u37PrWpFOXIfxCdY

tGe9sfkVx4yRbKNsNAZb14D6HEHqoK0F0dMqFnJeh6cERI6X77eq2QwPKqXzmqPg

PClkdohJnQ7Gg5Ac4LDUx8Zk/QUXWT/yPXC1NgAqVMGBiiXhG5nfgvef2W+WX7og

iFeU+RKXS1eJBru72mzEl2u5UJZP1VsAzsVmiykCAwEAAaOBjTCBijAdBgNVHQ4E

FgQUlq6eCqIWaGAEIrmbv+orFDm5hKMwTgYDVR0jBEcwRYAUlq6eCqIWaGAEIrmb

v+orFDm5hKOhF6QVMBMxETAPBgNVBAMMCENoYW5nZU1lghQod7wdqKwin2IrsGha

sXc+tYRPszAMBgNVHRMEBTADAQH/MAsGA1UdDwQEAwIBBjANBgkqhkiG9w0BAQsF

AAOCAQEAUQI/iQcCCHSkiQD9qUGozgsoHXMPPkm5xeDnjVXny+Ei3lCgrUJC/oqG

xPgja3WTODuOzUhFphRZQKqVxRgjVWRf8dOTJx3+KLVseYumFQ11yaZ/MBoTicMz

sjk9p8DqIrfU8x9cy720x7NQCaHvQlAushxnefXuzlkqWyeXmBJnPOM9hcxJc5/G

kPruTlKWmSBBg7qYEsy6CifBsVbN+6gZHwsgmkhyxQ/j7/h8t2gva4d11XWXtVWj

LyA0HCZw6iQSIowFnwWPFv+WckV4WWdC/ERlR95nBsPWCHzOsKtnkpEJjOHiBWh1

LzKPHae4zKnB4sNyRj/hEa5Xa/nNwg==

-----END CERTIFICATE-----

</ca>

Modify the public IPv4 address.

#### Hardening security

There are two basic options to harden the VPN security. In both cases, this requires that the user generates it's own PKI and copies some files to every vRouter instance.

##### Generating a PKI

This is a simple example to build a PKI. This does not need to be executed on any virtual router instance (although it is an option). The goal is to generate the required files and save them.

Download and decompress easy-rsa from [this link](https://github.com/OpenVPN/easy-rsa/releases/download/v3.0.6/EasyRSA-unix-v3.0.6.tgz).

Generate pki:

./easyrsa *--pki-dir=/etc/openvpn/pki --batch init-pki*

./easyrsa *--pki-dir=/etc/openvpn/pki --batch build-ca nopass*

./easyrsa *--pki-dir=/etc/openvpn/pki --batch gen-req HyperCX nopass*

./easyrsa *--pki-dir=/etc/openvpn/pki --batch sign-req server HyperCX*

openssl dhparam -out /etc/openvpn/dh2048.pem 2048

openvpn *--genkey --secret /etc/openvpn/ta.key*

##### Custom CA certificate

The easiest option would be to replace the CA certificate while keep relying on the user/password authentication. This will maintain all the server configurations and only the new CA certificate is required. Replace the server's **/etc/openvpn/pki/ca.crt** with the newly generated certificate. Client's configuration will also need to modify the included certificate.

##### Custom PKI and certificate based authentication

This is the recommended option, and requires the entire pki to be replaced. Also, the client's keys will need to be generated:

./easyrsa --pki-dir=/etc/openvpn/pki --batch gen-req client nopass

./easyrsa --pki-dir=/etc/openvpn/pki --batch sign-req client client

A new OpenVPN server and client configuration file are also required.

### S2S VPN

HyperCX NFV provides StrongSwan based Site-to-Site (S2S) VPNs. This VPN comes preconfigured and only needs a few parameters to be configured.

#### Deployment Considerations

* All the preconfigured parameters on the appliances must be the same as in the remote site for the VPN to work.
* For the VPN to be available, three parameters are required: **Remote Site Public IP**, **Remote Site Networks**, **Pre-shared Key**
* Each VM will be required to use the vRouter's IP as gateway.
* For HA deployments, a floating IP must be used on the public network and on each internal network.

#### Pre-configured Parameters

Since HyperCX NFV is based on its simplicity, there are some VPN parameters that are preconfigured as follows:

**Phase1:**

* Key Exchange version: **Auto**
* Authentication method: **Mutual PSK**
* Encryption algorithm: **aes256**
* Hash: **sha256**
* DH Group: **2(1024)**
* Lifetime (Seconds): **28800**

**Phase2:**

* Protocol: **ESP**
* Encryption algorithm: **aes256**
* Hash Algorithm: **sha256**
* PFS key group: **2(1024)**
* Lifetime: **3600**

#### Configuration

To enable S2S VPN, a public network with a public IP must be configured as the first Virtual Network (just like the OpenVPN VNF). In addition, three parameters are mandatory:

* **S2S REMOTE SITE IP**: Public IP of the remote site for the S2S VPN. This will be used as the edge point at the remote site.
* **S2S REMOTE NETWORKS**: Remote Networks shared by the S2S VPN. These are the networks from the remote site that will be able to access the local private networks and vice versa. Multiple networks separated by spaces can be inserted. Ex: 10.10.10.0/24 10.10.20.0/24
* **S2S PRE SHARED KEY**: Pre-shared key used by the S2S VPN. PSK is used as authentication method. This will be used to authenticate the VPN on each side. **Must be the same on both sites**.

There is another (optional) parameter that could be configured to monitor the status of the VPN, the **S2S HOST TO PING**. This option is for the Local Site to monitor the other edge, it must be a host at the Remote Site with the ping allowed.

### L7 Load Balancer

HyperCX NFV provides Layer 7 balancing capabilities based on NGINX. This is used to balance websites behind an https certificate. Here, NGINX will be configured with the following considerations:

* NGINX will work as reverse proxy, the backend servers see its IP as their client address. This is easy to configure and manage since the backends will not need to configure the load balancer as gateway, and the client will not get a response from a different IP to which it made the request. It is very similar to the other Load Balancer that HyperCX NFV provide.
* Client requests will be maintained to a single server.
* The SSL certificate will be maintained to a single server, the HyperCX NFV appliance. This avoids to keep the certificate in all the backends.
* Backends are monitored. This means that, as long as the backends is listening for connections, it will be marked as healthy and will receive requests.

#### Deployment Considerations

L7 Load Balancer does not require that the backends use the vRouter's IP as gateway. The considerations here are similar to the previously mentioned [L4 Load Balancing VNF](https://docs.virtalus.com/Advanced_Components/Hypercx_NFV/#l4-load-balancing).

**Note**

These VNF will automatically configure an SSL certificate for the HTTPS endpoint based on letsencrypt. For this to work, a DNS entry must be configured on your DNS servers pointing Site URL to the public IP these vrouter will have. You can force a specific IP from the address pool on the NIC options.

#### Configuration

Only two parameters are mandatory to enable the load balancer. During instantiation, under **Custom attributes**. If both of the following attributes are filled, the load balancer will be enabled.

* **L7 SITE NAME**: URL of the site that will be served by the L7 Load Balancer, the website you intend to use. Ex: demo.example.com.
* **L7 BACKENDS**: Set the IP of every backend server you intend to use separated by an space. Ex: 10.10.0.123 10.10.0.124 10.10.0.125

**Note**

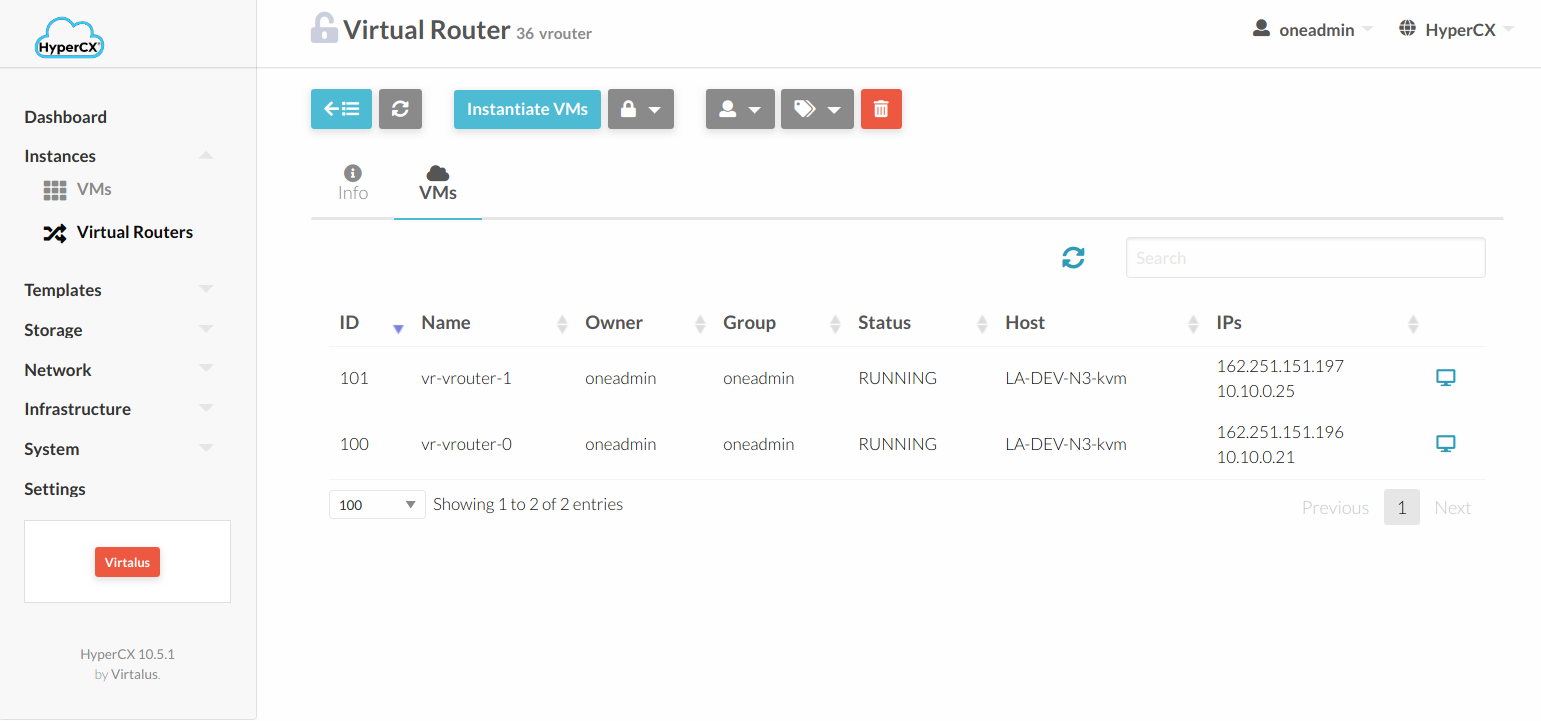
All http requests will be automatically redirected to https.

**Warning**

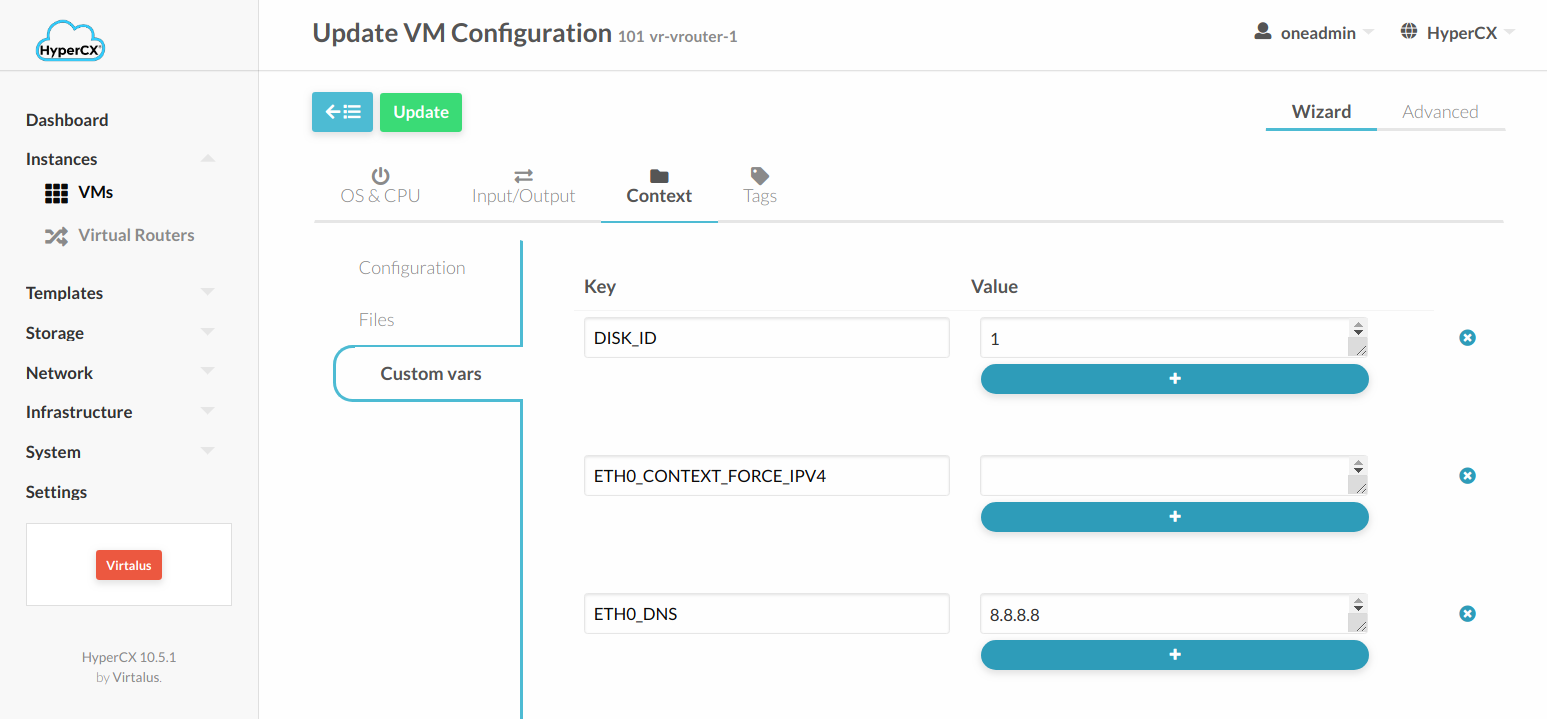
L7 Load Balancer is not compatible with **HyperCX NFV HA**. Works only with **standalone** deployments.

## Modify configurations

After the virtual router based on the HyperCX NFV template is deployed, any configuration from previous VNF must be configured directly on the VM instances. First, you need to identify all the VM instances that belong to the virtual router instance. This can be seen by selecting the virtual router and, inside the virtual router instance, clicking on the VMs tab.



To change the config in each VM it is necessary to go into the **Conf** tab from the VM and click **Update Configurations**. The required configurations can be found under **Context** --> **Custom vars**.

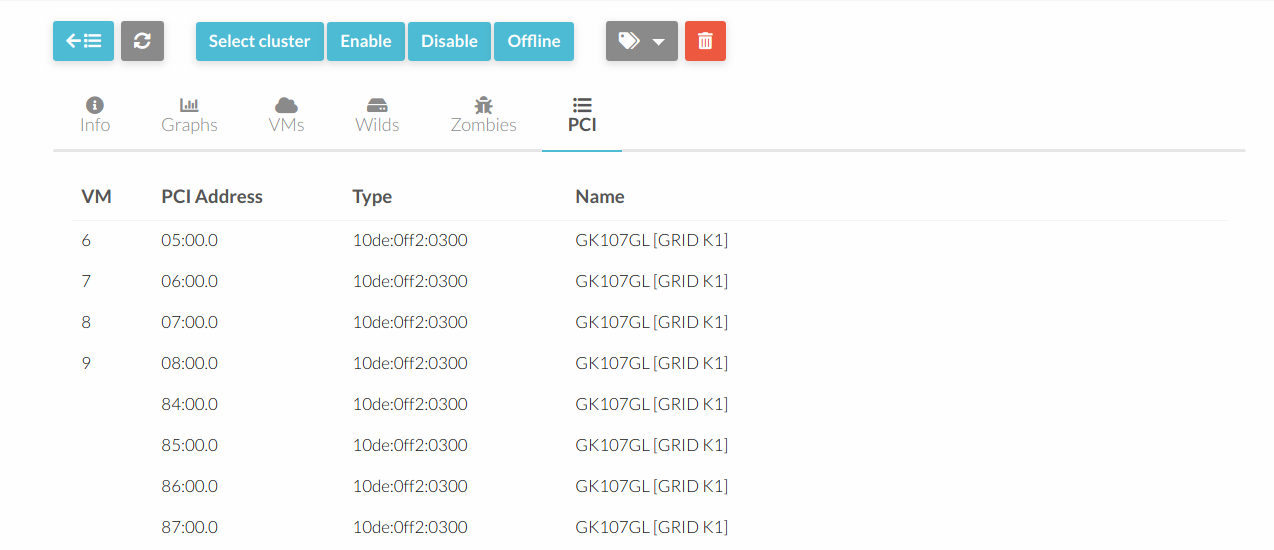


# HyperCX GPU

## Overview

Graphics Processing Units (GPUs) have become the tool choice in computationally demanding fields such as scientific computing and machine learning. GPU capable VMs enables every virtual machine to get the same GPU benefits as a physical desktop. Because work that was typically done by the CPU has been offloaded to the GPU, the user has a much better experience and more users can be supported, besides enabling additional fields.

GPU feature is available only on specific HyperCX clusters. For enabled clusters, one or several servers will contain GPU cards ready to be used. These can be seen by selecting the host under **Infrastructure --> Hosts** and selecting **PCI** tab.

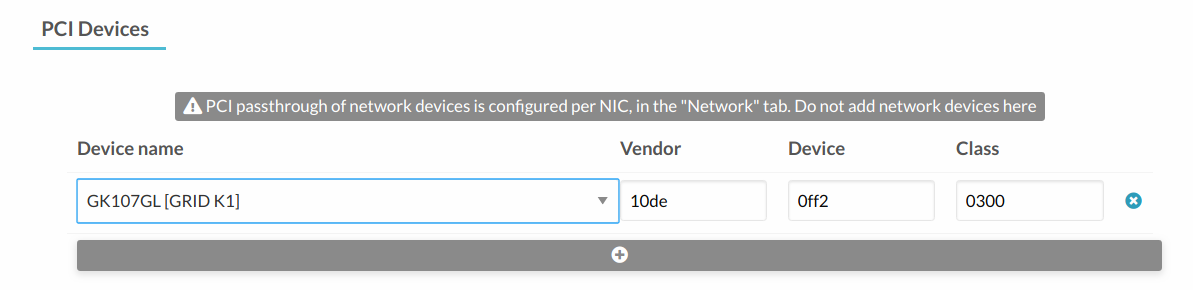


All the available GPU cards can be seen on this tab, and which VM is using a specific card, if any. Each card can only be assigned to a single VM.

## Adding a GPU card to a template

In order to use a GPU card on a VM, it must be configured on the template first, and then that template must be instantiated. HyperCX's scheduler will find a host with the specified GPU available and will only deploy the VM in that host. If no hosts match, an error will occur and the VM will stay in **PENDING** state. If it is manually deployed on a host (bypassing the scheduler) it will go into **FAILURE** state since the specified device it is not available on that host.

To add a GPU to a template, update the template, go to **Input/Output** tab and add the device under **PCI Devices**.

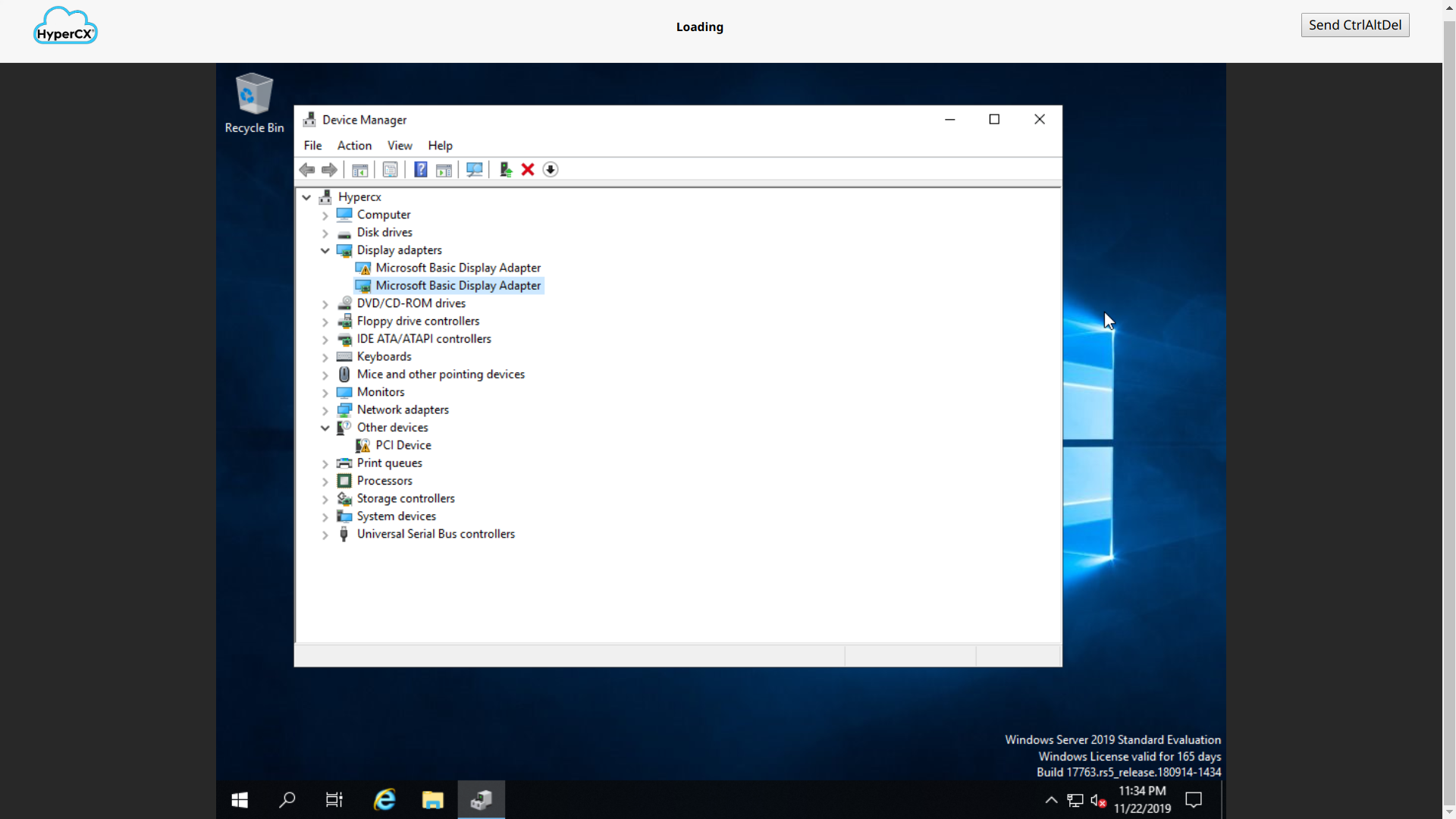


Note: The server might contain several cards from the same model of the same manufacturer. In this case, only one device will be shown on the template when adding the new PCI device. This is because Vendor, Device and Class match among all the available GPU cards in the server and it is the expected behavior. Once the device is selected, HyperCX will assign an unused card of the specified device model to each VM instantiated until the GPU cards run out.

## Guest OS configuration

After deploying the VM, assuming an appliance from the market was used, the GPU drivers will need to be installed. In this case, an example will be provided using Windows Server 2019 and Nvidia Grid K1.

Before installing the drivers, the guest OS should show something like this:



There are two video cards since once is the internal used by QEMU. This is the video card used to render the VNC connection. The second video card is the Nvidia Grid in this example. Drivers for the video card can be obtained from the manufacturer site. In this case, [Nvidia Drivers Download site.](https://www.nvidia.com/Download) Select **GRID** under Product Type, then **GRID Series** and **GRID K1**. Select your Operating System, download and install.

After a successful installation you will be ready to use the GPU card.

