VMware Cloud Director Remote Access VPN Integration Guide

For Cloud and Service Providers

Romain Decker / Sachi Bhatt

Cloud Services Business Unit

November 2020

vmware

Table of Contents

Introduction

This white paper is intended for VCPP Cloud Providers who are interested in offering remote access VPN solutions in their multi-tenant environments managed by VMware Cloud Director (VCD) or VMware Cloud Director service (CDS). The content below describes the design and deployment procedures and clearly delineates the cloud provider actions from the those of the tenant, addressing both managed service and self-service offerings that are possible.

A virtual private network (VPN) provides traffic tunneling through an encrypted connection, preventing it from being seen or modified in transit. VMware NSX® Data Center for vSphere® includes a remote access VPN feature (SSL VPN-Plus) that allows remote users to connect securely to the private networks and applications in the organization virtual data center.

VMware NSX-T™ Data Center only supports site-to-site VPN. This limitation requires providers or tenants to select alternative solutions, including open source or commercial, depending on the desired mix of features and support. Example of such solutions are OpenVPN or Wireguard. This deployment guide will guide through the steps to enable tenant administrators to implement those solutions as an alternative to provide secured access in their organization virtual data centers backed by NSX-T Data Center.

The steps to enable the solutions are agnostic to the VPN solution used. In the proposed implementation, the components are deployed manually and managed separately from Cloud Director.

Note: NSX-V SSL VPN-Plus feature is not discussed in this document but is still supported for NSX-V backed organization VDCs.

Disclaimer

VMware does not endorse, recommend, or support any particular third-party utility.

This document aims not to select an alternative to NSX SSL VPN-Plus, but rather to guide through the steps required to provide a remote access VPN to an organization VDC in Cloud Director by using third-party software. As such, some topics are outside the scope of this document:

- All VPN server and client configuration permutations and implementation steps can vary based on multiple factors, such as operating system, security policies, and topology.
- User interface exists for both solutions, but their installation and usage are outside the scope of this document.

General knowledge of networking and security, as well as on VMware Cloud Director concepts, is also required.

Virtual Private Network

A virtual private network (VPN) extends a private network across a public network and enables users and applications to send and receive data across shared or public networks as if their computing devices were directly connected to the private network. Two main types of VPN exist: site-to-site VPN and remote access VPN.

- **Site-to-site VPN** – A site-to-site VPN (also known as a router to router VPN) is a connection between two or more networks using an encrypted tunnel. The primary usage is to connect networks in multiple physical locations where a dedicated, always-on, connection between the locations is required. This is typically set up as a permanent connection between networking equipment. VMware NSX-T™ Data Center supports IPSec VPN and Layer 2 VPN.
    - Internet Protocol Security (IPSec) VPN secures traffic flowing between two networks connected over a public network through IPSec gateways called endpoints.
    - With Layer 2 VPN (L2 VPN), you can extend Layer 2 networks across multiple sites on the same broadcast domain.
- **Remote access VPN** – A remote access VPN extends the perimeter of your network to the remote endpoints. Individual remote users can connect securely to private networks from any Internet-enabled location by using VPN client software. VPN clients are usually available for Windows, Mac, or Linux operating systems, or even smartphones (Android, iPhone), and allow users to use a VPN through their computers over a safe connection.

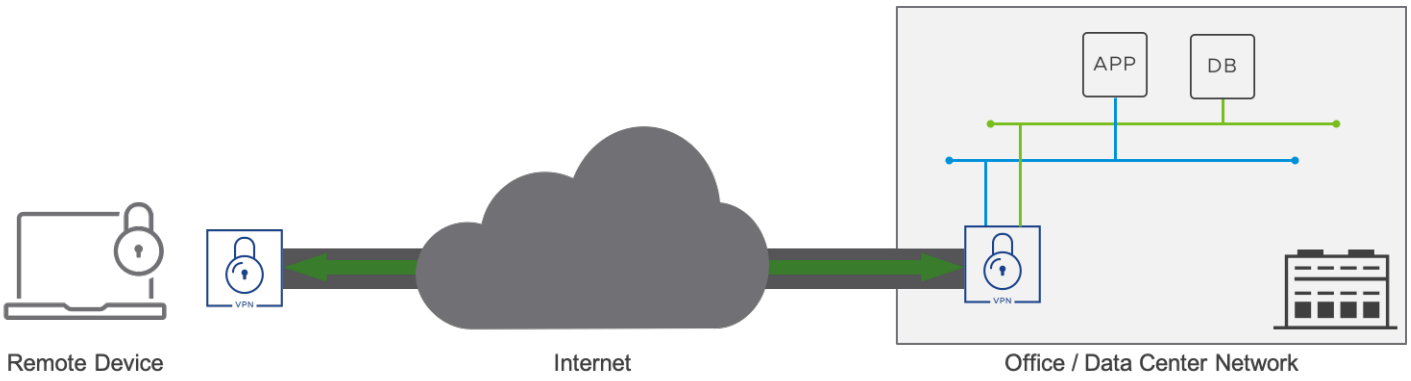

Figure 1 – Remote access VPN high-level concept

Remote Access VPN in VMware Cloud Provider Platform

VMware Cloud Director is a leading cloud service-delivery platform used by some of the world's most popular cloud providers to operate and manage successful cloud-service businesses. Using VMware Cloud Director, cloud providers deliver secure, efficient, and elastic cloud resources to thousands of enterprises and IT teams across the world.

VMware Cloud Director service is a software as a service implementation of VMware Cloud Director providing multi-tenancy support for VMware Cloud on AWS SDDC. VMware Cloud Director service is a unique and differentiated SaaS solution available to our VMware Cloud Providers via the Cloud Partner Navigator that helps them deliver new services, differentiate their offerings, increase their efficiency, expand globally, and onboard customers onto VMware Cloud on AWS.

While Cloud Director and Cloud Director service empower customers to deploy workloads in secured, isolated, and multi-tenant environments, customers still require access to their virtual machines for operations, maintenance, lifecycle, troubleshooting, and so on.

One approach could be to define different sets of firewall and NAT rules for required access ports (SSH, Remote Desktop, etc.), but the management overhead is growing with the number of virtual machines & services to access. Another option is to configure site-to-site VPN between on-premises networks and the NSX-T Data Center edge gateway in the organization virtual data center.

The approach described in this document is to use a remote access VPN solution, which has multiple benefits:

- Access securely virtual machines in an organization VDC without the use of a jump box from anywhere without exposing them to Internet.
- Limit the number of public IPs used for NAT'd environments.
- Limit the number of ports opened to an organization VDC.
- Allows the customer to continue using software that they are familiar with (no changes in tooling).

Topology

With the proposed implementation in this document, tenant admins can implement the VPN solution of their choice from the tenant portal, or cloud providers can provide this function as a managed service.

Both VMware Cloud Director and Cloud Director service are in the scope of this document. Although general concepts and the suggested topology for the VPN solution are similar, they will be treated separately throughout the document.

The VPN client can be an individual endpoint connecting from the Internet or reside within an on-premises datacenter. This guide assumes that outgoing ports are opened from the client to the VPN tunnel endpoint to be established.

The diagrams below are logical representations of the demonstrated solutions from this document. The actual implementation may differ, as Cloud Director offers a wide range of possibilities on how to be implemented.

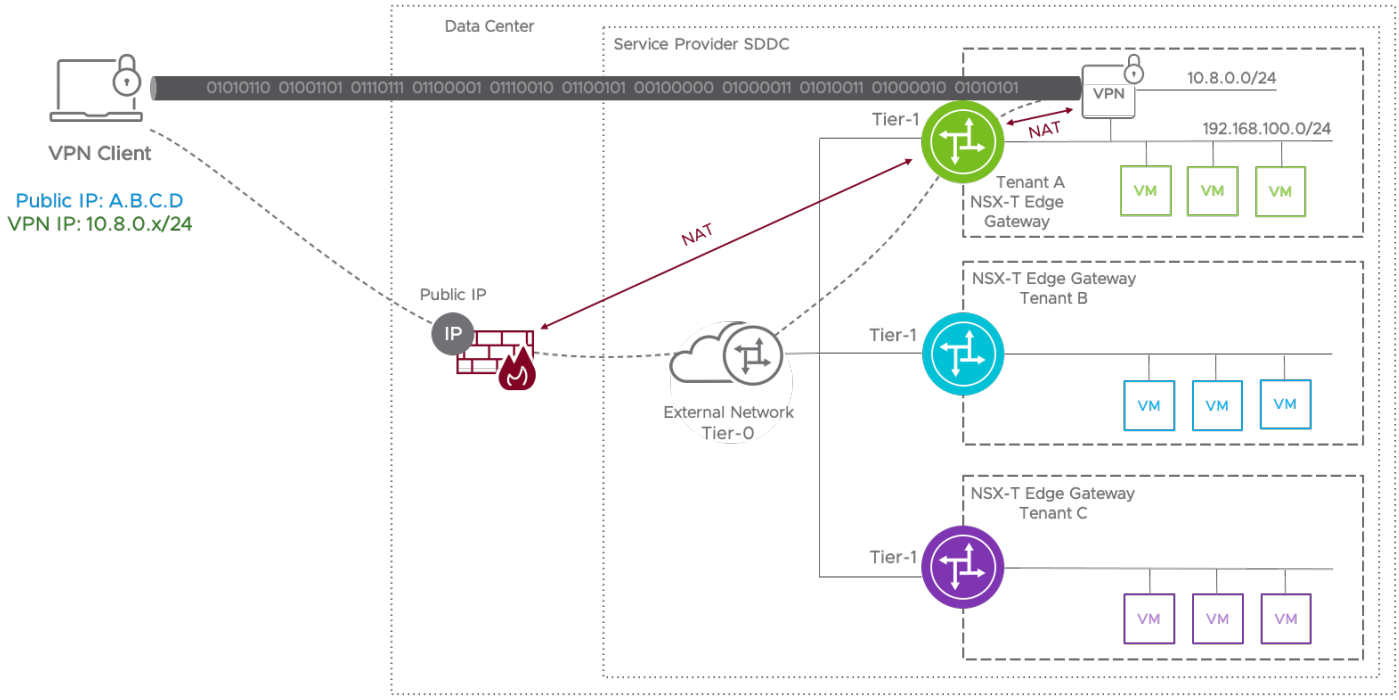

Figure 2 – Remote access VPN solution topology on VMware Cloud Director

VMware Cloud Director service networking and multi-tenancy architecture is described in detail in the Multi-Tenancy with VMware Cloud Director services white paper.

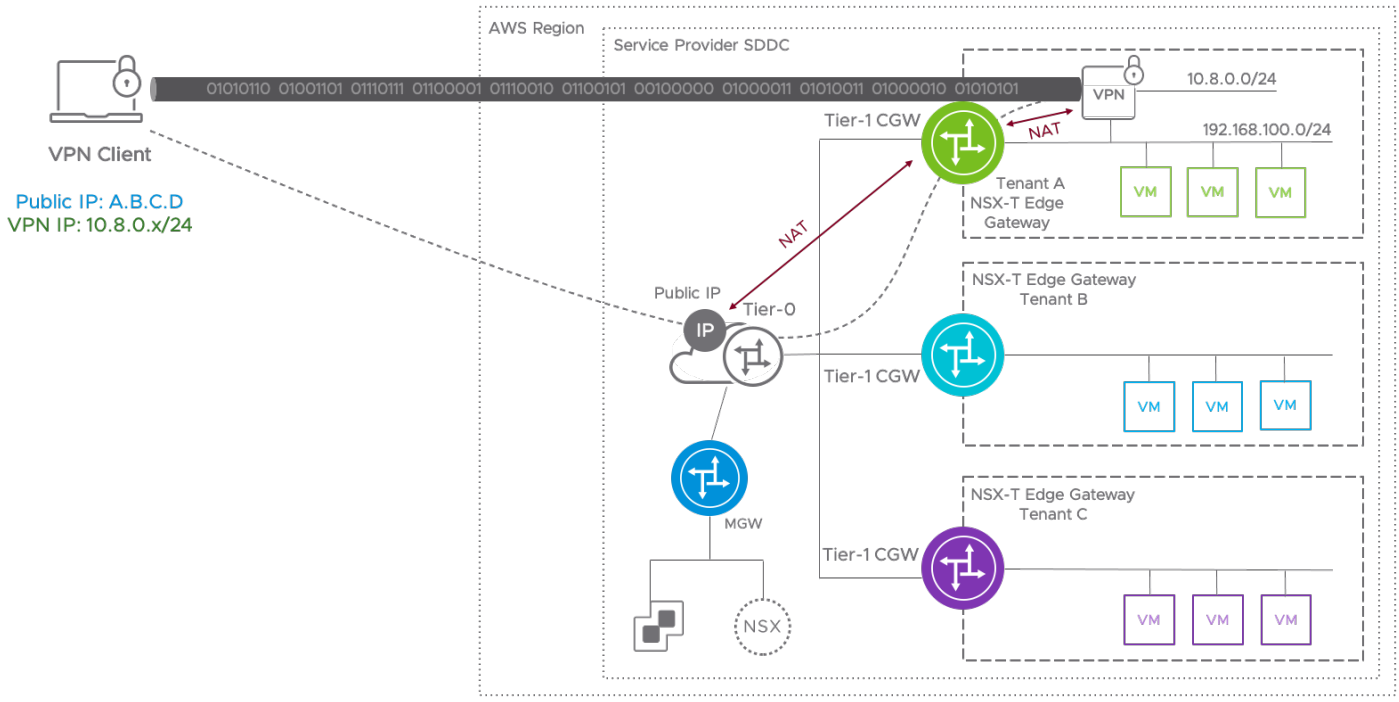

Figure 3 – Remote access VPN solution topology on Cloud Director service

Implementation Workflow

This section explains steps for common configuration tasks to be performed so that customers can implement a remote access VPN server on a virtual machine, which is either running on an on-premises Cloud Director instance or in Cloud Director service. Please note that the actual implementation may differ in your environment.

The following table identifies the high-level tasks required for the implementation of the solution, as well as their scope.

Task	Platform	Location	Description
Requirements	VCD / CDS	N/A	Confirm that the platform is in a healthy state before starting the implementation.
VPN server	VCD / CDS	Tenant portal	Deploy the virtual machine that will run the remote access solution on a routed organization VDC network.
Edge Gateway – NAT	VCD / CDS	Tenant portal	Configure NAT rules in the NSX-T edge gateway to map the traffic from the external network to the VPN server.
Edge Gateway – Firewall	VCD / CDS	Tenant portal	Configure the NSX-T edge gateway firewall to allow and secure the incoming VPN traffic.
Distributed Firewall	VCD / CDS	Tenant portal	(optional) If distributed firewall is enabled for the concerned virtual machines and networks, create rules to allow the VPN traffic.
Request a Public IP	CDS	VMC on AWS Console	Request a new public IP.
Internet NAT Rule	CDS	VMC on AWS Console	Create a new Internet NAT rule for customer's external IP.
Internet Firewall	VCD/CDS	N/A	(optional) If required, configure the additional firewall(s) (if existing) to allow incoming VPN traffic.
Remote Access Setup	VCD / CDS	N/A	Deploy and configure a remote access VPN solution.

Table 1 – High-level implementation plan

Common Steps

This section describes the initial steps that are common as NSX-T backs both platforms (VMware Cloud Director and Cloud Director service). The actions realized are launched from the tenant portal and provide an identical feature set across both platforms.

Both provider and tenant network administrators can configure the NSX-T Data Center edge gateway.

Create NSX-T Edge Gateway NAT Rules

NAT rules perform translation between a customer's allocated external network IP and an organization VDC network IP.

- A DNAT rule translates the IP address and, optionally, the port of packets received by an organization VDC network coming from an external network or from another organization VDC network.
- A SNAT rule translates the source IP address of packets sent from an organization VDC network out to an external network or to another organization VDC network.

Note: The external IP addresses must have been added to the NSX-T edge gateway interface on which you want to add the rules.

Configure the following:

- A DNAT rule to translate incoming traffic to the VPN server VM.
- A SNAT rule to translate outgoing traffic from the VPN server VM.

Customers can also specify the service for NAT rules per VPN service port used. In the example below, the VPN server was deployed with a 192.168.100.6 IP and configured with a 172.16.2.11 NAT IP on the external network.

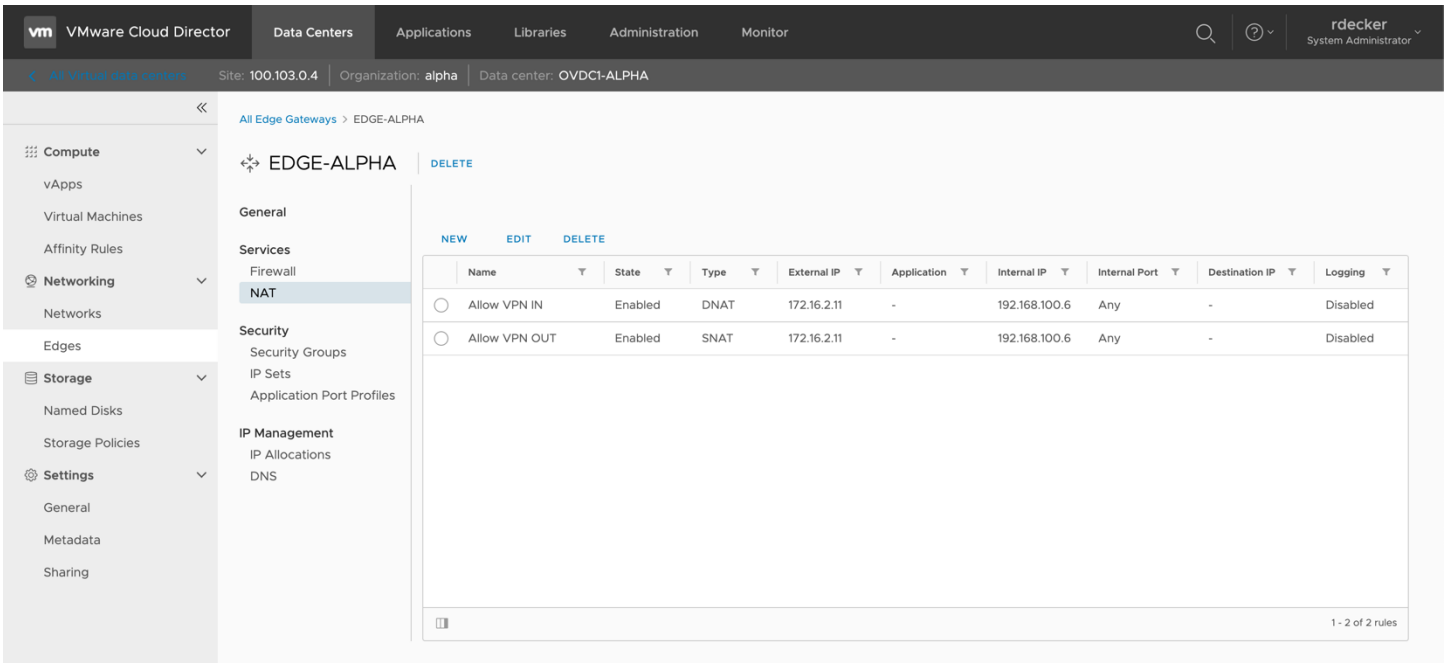

Figure 4 – NSX-T edge gateway NAT rules translating traffic from the external network to the VPN server

Create NSX-T Edge Gateway Firewall Rules

To secure the VPN server, the provider or tenant administrators can configure the NSX-T edge gateway firewall to allow only VPN and required application traffic while blocking any other traffic.

The following example configuration shows that incoming SSH and VPN traffic is permitted to VPN server VM, while HTTP(S) and DNS traffic is allowed from it. The VPN-Ports application port profile was created before the firewall rules and it contains the required ports (which may be different and customizable based on the chosen remote access VPN solution).

If VPN clients need to access workloads in different organization VDC networks connected to the same edge, additional firewall rules may be required. The usage of objects such as IP sets and/or security groups as a source or destination is recommended.

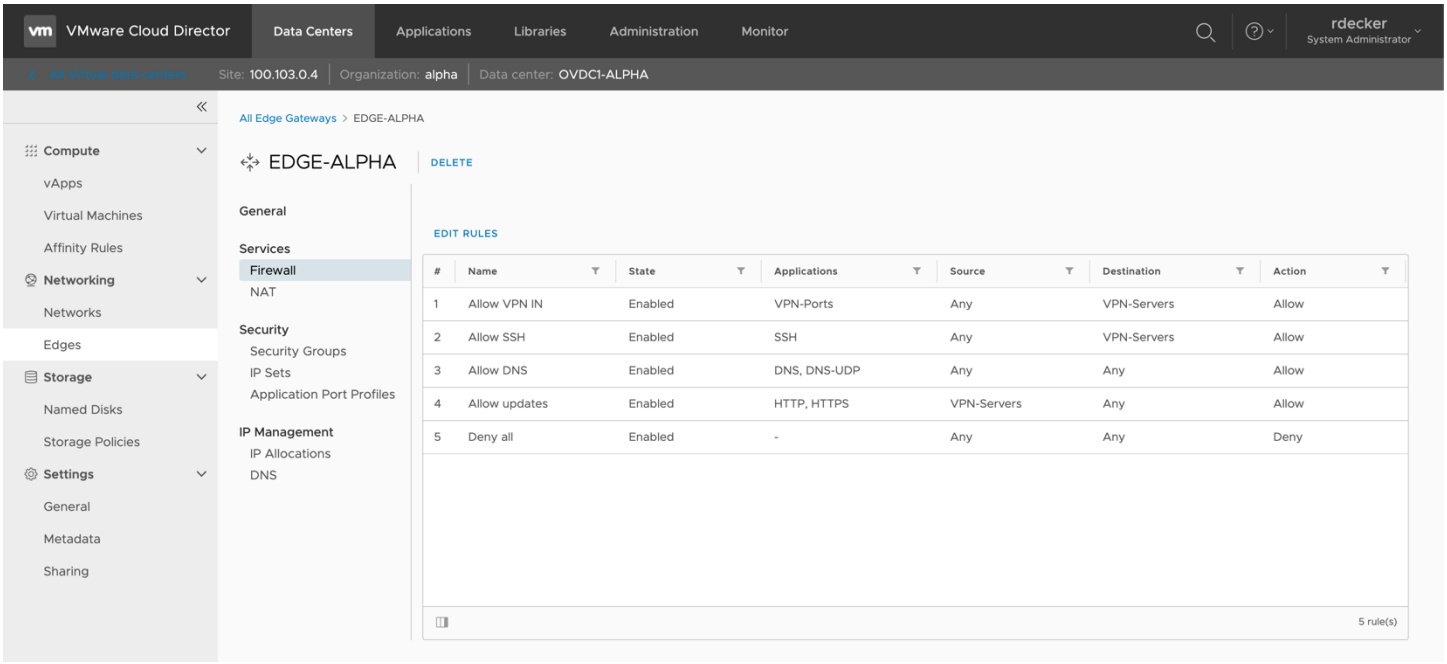

Figure 5 – NSX-T edge gateway firewall rules to allow VPN traffic

Cloud Director Service Additional Steps

VMware Cloud (VMC) on AWS allows providers to directly provision networking and security services up to the Internet. Using the VMware Cloud on AWS console, a provider administrator can request a public IP address to assign to the VPN server, as well as provision NAT rule as explained in later sections.

Request a Public IP Address from AWS

The provider can request a new IP address with one click from the "Networking & Security" section in the SDDC in the VMware Cloud on AWS console. VMware Cloud on AWS provisions the IP address from AWS.

A default SNAT public IP is available for all workloads to connect to the internet. However, for inbound access to the workloads from the Internet, additional public IP addresses are required.

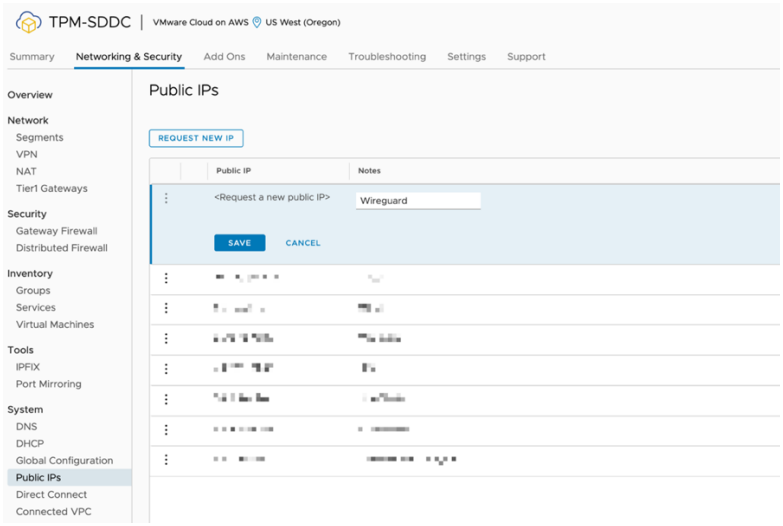

Figure 6 – Request a public IP address from VMware Cloud console

Create a NAT Rule for Inbound Access

The provider needs to map the requested public IP address to the customer's external IP to complete the inbound connectivity to the VM that will be leveraged as the VPN server. The goal is to provide internet connectivity up to the customer's external IP, which is used in NAT rules.

In the previous Create NSX-T Edge Gateway NAT section, we created SNAT and DNAT rules with the 172.16.2.11 external IP.

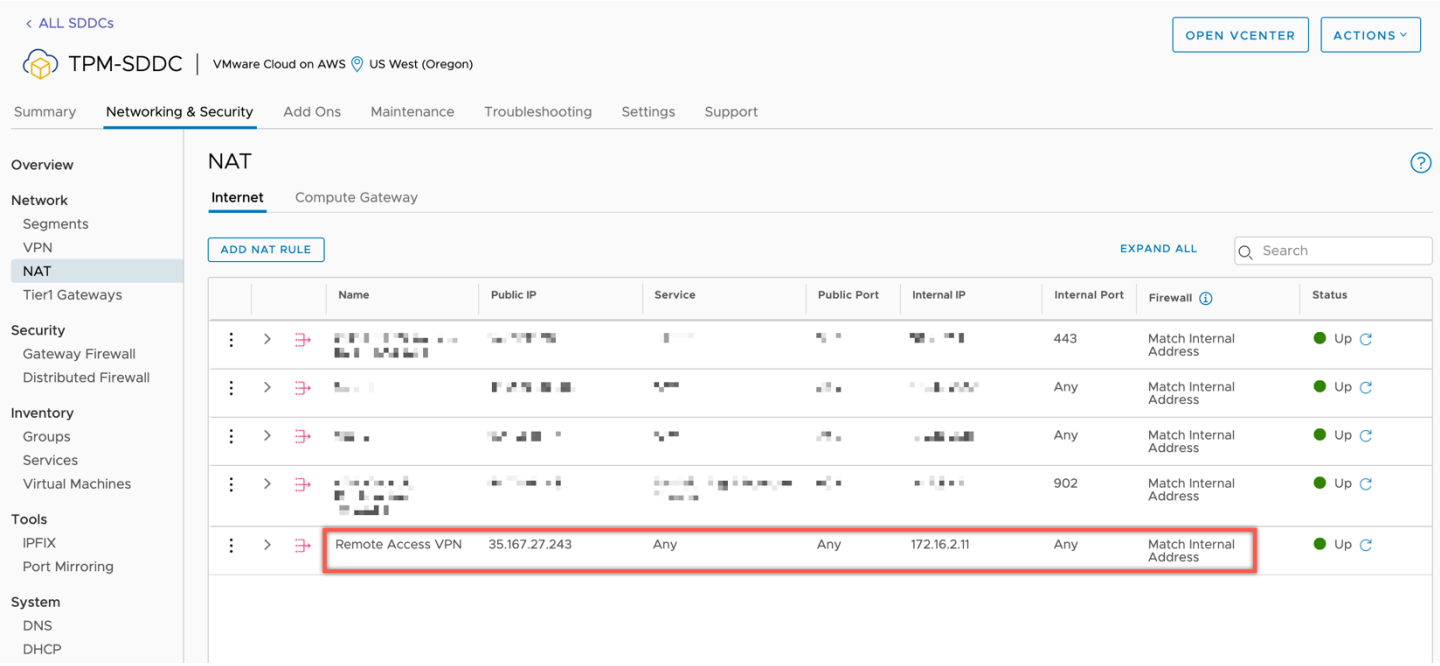

Figure 7 - Internet NAT rule with customer's external network IP and requested additional public IP address

VPN Server and Client Setup

Note: All steps below will be realized on Ubuntu.

Customers implementing remote access VPN must configure inbound and outbound connectivity. The actual VPN server can be added at the start of this process, or at the end. The document describes the latter.

At this point, customers can implement the remote access VPN solution of their choice on the VM acting as a VPN server.

Various open-source and commercial remote access VPN solutions are available on the market. For the purpose of this document, two open source solutions are provided as examples :

- **OpenVPN** – SSL VPN solution which implements OSI layer 2 or 3 secure network extension using the industry-standard SSL/TLS protocol, and that accommodates a wide range of configurations.
- **WireGuard** – WireGuard is a secure network tunnel, operating at OSI layer 3, and implemented as a kernel virtual network interface for Linux.

Both solutions support clients on a wide range of operating systems, including Linux, Windows, FreeBSD, OpenBSD, macOS, iOS, and Android. While those solutions are SSL VPN, they are not « clientless » SSL VPNs in the sense that some editors or vendors commonly state : VPN software clients must be installed on all client devices.

Setting up a VPN often entails linking together private subnets from different locations. The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets (codified in RFC 1918):

- 10.0.0.0 – 10.255.255.255 (10/8 prefix)
- 172.16.0.0 – 172.31.255.255 (172.16/12 prefix)

- 192.168.0.0 – 192.168.255.255 (192.168/16 prefix)

While addresses from these netblocks should normally be used in VPN configurations, it's important to select addresses that minimize the probability of IP address or subnet conflicts.

Some aspects of the server's networking configuration that are common to both solutions need to be tweaked so that they can correctly route traffic through the VPN. The first of these is IP forwarding, a method for determining where IP traffic should be routed. IP forwarding is the ability for an operating system to accept incoming network packets on one interface, recognize that they are not meant for the system itself, but that they should be passed on to another network, and then forwarded accordingly.

Adjust your server's default IP forwarding setting by uncommenting the `net.ipv4.ip_forward=1` line in the `/etc/sysctl.conf` file:

```
vmware@vpn-server:~$ sudo vim /etc/sysctl.conf
```

Save and exit the file, then run this command:

```
vmware@vpn-server:~$ sysctl -p
```

The alterations you've made to the sysctl.conf file should now have taken effect, and IP forwarding should now be enabled permanently.

Likewise, firewall configuration should be done on the virtual machine acting as a remote access VPN server : SSH port, as well as ports used by the VPN solutions (e.g., the default UDP/1194 port for OpenVPN, or a custom UDP/51820 port for WireGuard) can be opened using UFW (Uncomplicated Firewall) commands on Ubuntu :

```
vmware@vpn-server:~$ sudo ufw allow 22/tcp

vmware@vpn-server:~$ sudo ufw allow 1194/udp

vmware@vpn-server:~$ sudo ufw allow 51820/udp

vmware@vpn-server:~$ sudo ufw enable
```

You can check the status of UFW with this command:

```
vmware@vpn-server:~$ sudo ufw status verbose
```

As demonstrated solutions are configured with a single Ethernet connection and running in routed mode, they can be considered as a one-armed router. Thereby, we have to make sure that the VMs in the organization VDC networks to which we want to connect to using the VPN tunnel route all traffic from the tunnel network back to the VPN server.

A solution is to implement a one-to-many source NAT for connected VPN clients, known as IP masquerade in Linux. In this context, IP masquerade allows a set of VPN clients to access the organization VDC networks, as the client VPN IPs will be substituted with the interface IP from the VPN server.

To implement IP masquerading, the first step is to allow UFW to allow forwarded packets: open the `/etc/default/ufw` configuration and change `DEFAULT_FORWARD_POLICY="DROP"` to `DEFAULT_FORWARD_POLICY="ACCEPT"`.

Then, additional UFW rules for NAT and IP masquerading of connected clients need to be inserted towards the top of the `/etc/ufw/before.rules` file, as depicted below:

```
romain@openvpn:~$
romain@openvpn:~$ ip a | grep global
    inet 192.168.10.2/24 brd 192.168.10.255 scope global ens192
    inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
romain@openvpn:~$ sudo cat /etc/ufw/before.rules
#
# rules.before
#
# Rules that should be run before the ufw command line added rules. Custom
# rules should be added to one of these chains:
#   ufw-before-input
#   ufw-before-output
#   ufw-before-forward
#

# START VPN RULES
# NAT table rules
*nat
:POSTROUTING ACCEPT [0:0]
# Allow traffic from VPN client to ens192
-A POSTROUTING -s 10.8.0.0/24 -o ens192 -j MASQUERADE
COMMIT
# END VPN RULES

# Don't delete these required lines, otherwise there will be errors
*filter
:ufw-before-input - [0:0]
:ufw-before-output - [0:0]
:ufw-before-forward - [0:0]
:ufw-not-local - [0:0]
# End required lines
```

Figure 8 – NAT and IP masquerading using UFW

This will set the default policy for the POSTROUTING chain in the NAT table and masquerade any traffic coming from the VPN. Remember to replace the subnet and interface information `-A POSTROUTING` line above with your information.

Although the installation and configuration are different between OpenVPN and WireGuard, the implementation doesn't change from a Cloud Director and Cloud Director service aspect. As such, the OpenVPN and WireGuard implementation details that follow in the next sections are platform-agnostic.

OpenVPN

OpenVPN is both a VPN protocol and the code needed to implement that protocol. Initially released in 2001, it has become the most widely used VPN protocol thanks to its flexibility, reliability, and ability to work with NAT and firewalls. Open-source and considered to be very secure and reliable, OpenVPN code is published under the GNU General Public License (GPL) version 2.

OpenVPN is an SSL VPN which implements OSI layer 2 or 3 secure network extension using the industry-standard SSL/TLS protocol, supports flexible client authentication methods based on certificates, smart cards, and/or username/password credentials, and allows user or group-specific access control policies using firewall rules applied to the VPN virtual interface.

OpenVPN uses the OpenSSL library to provide encryption of both the data and control channels. OpenSSL can run over common network protocols (TCP and UDP) and supports a wide number of different cryptographic algorithms, which makes it agile. That is, the code can negotiate the use of different algorithms depending on circumstances. This makes OpenVPN very flexible but greatly increases the complexity of the code.

Crypto agility is the ability of a security system to switch between security protocols and encryption methods. This can be achieved without having to make any changes to the existing system in place.

The program is installed independently and configured by editing text files manually, rather than through a GUI-based wizard. The entire package consists of one binary for both client and server connections, an optional configuration file, and one or more key files depending on the authentication method used.

OpenVPN can be configured to provide either a routed or bridged VPN. Ethernet bridging essentially involves combining an ethernet interface with one or more virtual TAP interfaces and bridging them together under the umbrella of a single bridge interface.

Implement OpenVPN

The installation process is based on Ubuntu 18.04.5 LTS and OpenVPN 2.4.4. Documentation regarding other operating systems is available on the OpenVPN project website.

The first step in building an OpenVPN 2.x configuration is to establish a PKI (public key infrastructure). The PKI consists of:

- a separate certificate (also known as a public key) and private key for the server and each client, and
- a master Certificate Authority (CA) certificate and key that is used to sign each of the server and client certificates.

OpenVPN supports bidirectional authentication based on certificates, meaning that the client must authenticate the server certificate, and the server must authenticate the client certificate before mutual trust is established.

If such a public key infrastructure is not already available, it is recommended to create and manage it on a separate machine.

For PKI management, OpenVPN suggests the usage of easy-rsa, a CLI utility to build and manage a PKI CA. The steps describing how to set up your own Certificate Authority and generating certificates and keys are available in the OpenVPN documentation and won't be covered in this document.

From a networking aspect, the OpenVPN server uses port 1194 and the UDP protocol by default to accept client connections. If you need to use a different port because of restrictive network environments that your clients might be in, you can change the port and protocol options. Port 1194 is the official IANA assigned port number for OpenVPN, but any port number between 1 and 65535 will work if not already used on the server. Port 443 is a popular choice since it is usually allowed through firewall rules.

The configuration is stored in the `server.conf` file, and contains information such as the IP address and the port the service listens on, the service's cipher list, its service certificate, and so on.

It's best to use the OpenVPN sample configuration files as a starting point for your own configuration. These files can be found in :

- the **sample-config-files** directory of the OpenVPN source distribution
- the **sample-config-files** directory in **/usr/share/doc/packages/openvpn** or **/usr/share/doc/openvpn** if installed from an RPM or DEB package
- **Start Menu -> All Programs -> OpenVPN -> OpenVPN Sample Configuration Files** on Windows.

The sample server configuration file is an ideal starting point for an OpenVPN server configuration. It will create a VPN using a virtual **TUN** network interface (routed mode), will listen for client connections on **UDP port 1194**, and distribute virtual addresses to connecting clients from the **10.8.0.0/24** subnet.

Note: This IP range should be configured in accordance with your topology.

Before you use the sample configuration file, you should first edit the **ca**, **cert**, **key**, and **dh** (Diffie hellman) parameters to point to the files you generated in the PKI section above. At this point, the server configuration file is usable, and further customization will not be covered in this document.

As the server configuration is ready, you can start the OpenVPN service :

```
vmware@openvpn:~$ sudo systemctl start openvpn-server@server

vmware@openvpn:~$ sudo systemctl enable openvpn-server@server
```

The remaining steps concern the client setup : certificate and key pair generation, as well as the client configuration file creation.

Generating client certificates is very easy using easy-rsa. Run the following command from the PKI server :

```
vmware@openvpn-server:~/easy-rsa$ s./easyrsa build-client-full romain
```

```
romain@openvpn:~/easy-rsa$ ./easyrsa build-client-full romain

Note: using Easy-RSA configuration from: /home/romain/easy-rsa/vars
Using SSL: openssl OpenSSL 1.1.1  11 Sep 2018
Generating a RSA private key
.........................+++++
.............................+++++
writing new private key to '/home/romain/easy-rsa/pki/easy-rsa-6834.MkWSZQ/tmp.BSf0HT'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
Using configuration from /home/romain/easy-rsa/pki/easy-rsa-6834.MkWSZQ/tmp.vSUfY8
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'romain'
Certificate is to be certified until Jan 18 14:50:54 2023 GMT (825 days)

Write out database with 1 new entries
Data Base Updated
```

Figure 9 – Client certificates and keys generation with easy-rsa

The final step concerns the client configuration files: the sample client configuration file (**`client.conf`** on Linux/BSD/Unix or **`client.ovpn`** on Windows) mirrors the default directives set in the sample server configuration file.

- Like the server configuration file, first, edit the **ca**, **cert**, **key,** and **dh** parameters to point to the files you generated previously. Note that each client should have its own **certificate/key** pair. Only the **ca** file is universal across the OpenVPN server and all clients.
- Next, edit the **remote** directive to point to the hostname/IP address and port number of the OpenVPN server : use the public IP address that is NATed to the VPN server.

The OpenVPN client can be installed, the certificates were transferred to the device, and the configuration file was imported. In the screenshot below, a successful connection has been established, my client received a **10.8.0.6 IP**, and 2 routes were automatically installed by OpenVPN (**192.168.10.0/24** and **192.168.20.0/24**). Those routes represent organization VDC networks that I want to access.

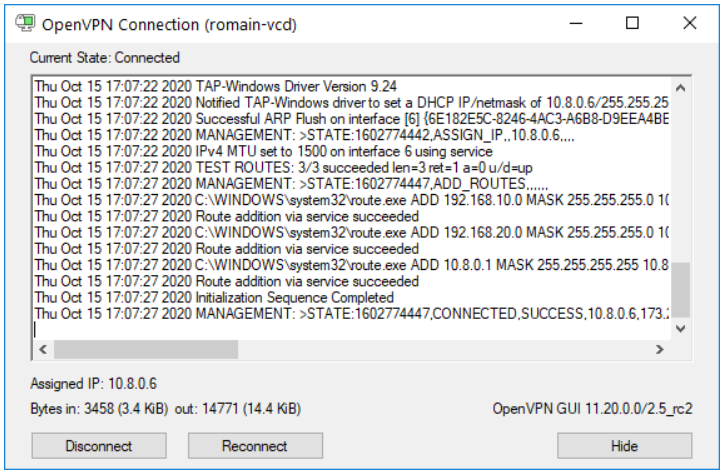

Figure 10 - Successful OpenVPN connection

WireGuard

WireGuard is a free open-source software application and communication protocol that implements VPN techniques to create secure point-to-point connections in routed or bridged configurations. It runs as a module inside the Linux kernel and aims to be easier to use and for better performance than the IPsec and OpenVPN tunneling protocols.

It is published under the GNU General Public License (GPL) version 2, just like OpenVPN. The Linux version of the software has reached a stable production release and was incorporated into the Linux kernel release in March 2020.

The WireGuard philosophy differs greatly from other alternatives when it comes to cryptographic algorithms. Whereas OpenVPN is crypto-agile, i.e., is flexible with the algorithms it uses, each WireGuard version uses one fixed set of algorithms (more details in the Protocol & Cryptography page from the WireGuard documentation). This leads to less complexity (and far less code), a smaller attack surface, and immunity to downgrade attacks. However,it will force all endpoints to upgrade to a latest version of WireGuard if a problem is discovered in any of the ciphers or protocols used in the current version.

Implement WireGuard

The installation process is based on Ubuntu 20.04.1 LTS and Wireguard 1.0. Documentation regarding the software installation as well as other operating systems is available on the WireGuard website.

From a networking aspect, WireGuard only works over UDP, and any port can be setup for the connection.

A set of CLI and configuration utilities come with the WireGuard package, such as `wg` or `wg-quick`:

- **`wg`** is the configuration utility for getting and setting the configuration of WireGuard tunnel interfaces. The interfaces themselves can be added and removed using **`ip-link,`** and their IP addresses and routing tables can be set using **`ip-address`** and **`ip-route`**.
- **`wg-quick`** is a script for easily bringing up a WireGuard interface quickly, and is suitable for most common use cases.

Once WireGuard is installed, start by adding a new interface:

```
vmware@wireguard:~$ sudo ip link add dev wg0 type wireguard
```

Then, assign an IP address; I selected 10.50.1.1/24 arbitrarily as this subnet was not used in the infrastructure:

```
vmware@wireguard:~$ sudo ip address add dev wg0 10.51.0.1/24
```

WireGuard requires base64-encoded public and private keys. These can be generated using the `wg` utility:

```
vmware@wireguard:~/keys$ umask 077

vmware@wireguard:~/keys$ wg genkey | tee privatekey | wg pubkey > publickey
```

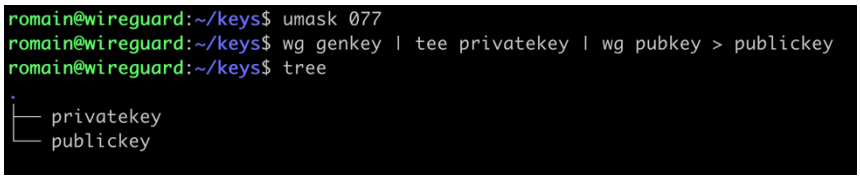

Figure 11 – WireGuard server keys generation

The server configuration can be done using the `wg` utility or by using a config file (`wg0.conf` in the example below). Create the `/etc/wireguard/wg0.conf` and add the following content:

```
[Interface]

PrivateKey = <redacted>

Address = 10.51.0.1/32

ListenPort = 51820

SaveConfig = true
```

The `interface` section represents the server configuration:

- `PrivateKey` — The server private key created previously.
- `Address` — Comma-separated list of private IP (v4 or v6) addresses (optionally with CIDR masks) of my VPN server.
- `ListenPort` — A 16-bit port for listening.
- `SaveConfig` — Automatically update the configuration file whenever a new peer is added while the service is running.

Start the WireGuard service and enable it to automatically restart on boot:

```
vmware@wireguard:$ sudo wg-quick up wg0

vmware@wireguard:$ sudo systemctl enable wg-quick@wg0
```

The `wg show` command allows to check if the VPN tunnel is running.

```
romain@wireguard:~$ sudo wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.51.0.1/32 dev wg0
[#] ip link set mtu 1420 up dev wg0
romain@wireguard:~$ sudo wg
interface: wg0
  public key: [redacted]
  private key: (hidden)
  listening port: 51820
```

Figure 12 – WireGuard server ready to accept client connection

The process for setting up a client is similar to setting up the server. When using Windows as your client's operating system, you can either import a client configuration file or create a new tunnel configuration with the **Add Tunnel** > **Add empty tunnel** button. By doing that, public and private keys are automatically generated, and the `interface` section is inserted in the file.

Complete the client configuration with the addition of the `peer` section:

```
[Peer]

PublicKey = <redacted>

AllowedIPs = 0.0.0.0/0

Endpoint = <vpn-server-public-ip:port>
```

- `PublicKey` — The server public key.
- `AllowedIPs` — A comma-separated list of IP (v4 or v6) addresses with CIDR masks from which incoming traffic for this peer is allowed and to which outgoing traffic for this peer is directed.
- `Endpoint` — The server public IP or hostname, followed by the port number.

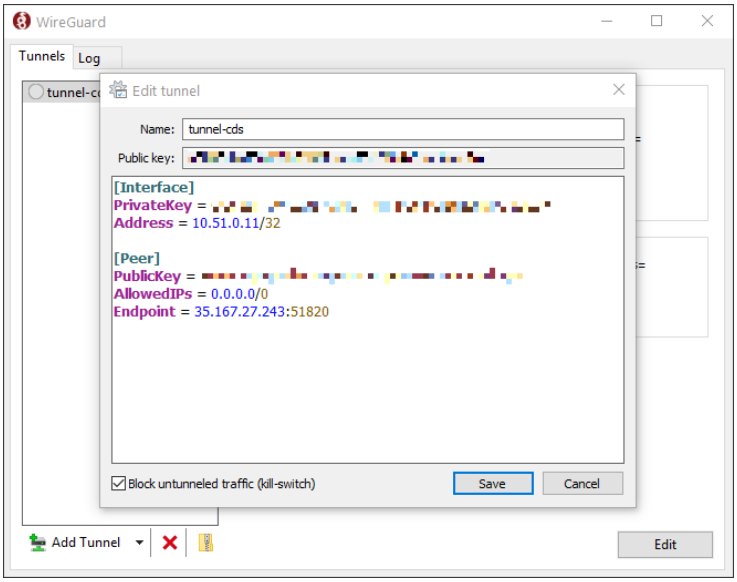

Figure 13 – New tunnel creation in the WireGuard Windows client

The final step is to add the peer information to the WireGuard server configuration. You can either directly edit the `/etc/wireguard/wg0.conf` file or use the command line.

```
vmware@wireguard:~$ sudo wg set wg0 peer <base64-client-public-key> allowed-ips 10.51.0.11/24
```

Verify the connection using `wg show`. Regardless of the method chosen to add the peer information to the server, there should be a `peer` section in the current WireGuard configuration.

Note: The peer section represents one client configuration; would multiple clients be required, additional peer sections would have to be added.

```
romain@wireguard:~/server$ sudo wg set wg0 peer [redacted]
allowed-ips 10.51.0.0/24
romain@wireguard:~/server$ sudo wg
interface: wg0
  public key: [redacted]
  private key: (hidden)
  listening port: 51820

peer: [redacted]
  allowed ips: 10.51.0.0/24
```

Figure 14 – Finalised WireGuard server configuration

Click Activate to connect the client and the server.

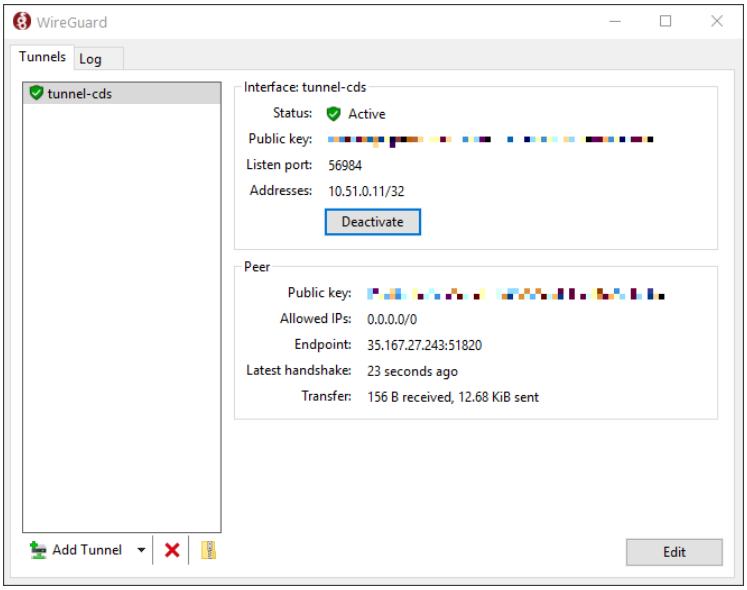

Figure 15 – WireGuard connection established

Conclusion

This deployment guide covered steps to facilitate customers to configure remote access VPN on VMware Cloud Director and Cloud Director service's tenant portal on VMware Cloud on AWS. These easy steps are agnostic to customer's selection of remote access VPN solution.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com.