SECURITY OF SYSTEMS AND NETWORKS

SYSTEMS AND NETWORK ENGINEERING 2022/23

Implementing OpenVPN, Cisco AnyConnect, and Custom VPN Solution with Comparison (+AD/LDAP authentication)

Author:
Hussein Al-Khalissi
Ammar Tutunchy

Instructor:
Nikola Zlatanov

December 5, 2022

Contents

List of Figures

Chapter 1

Introduction

With the rise of remote work and hybrid work environments, it is not easy to access remote resources safely over networks we do not trust, especially after the pandemic outbreak of COVID-19. Companies started utilizing remote work and hybrid work schedules. They accelerated their digital transformation strategies by migrating to the cloud and allowing employees easy remote access. There are many VPN vendors, and competition in this market is rising rapidly. In this study, the three most popular VPN options for medium-sized to large businesses and corporations will be looked at and compared. How easy is it to integrate with the infrastructure and company environment, allowing seamless user authentication via Active Directory and LDAP. We will compare the difficulty of the implementation, features, costs, integration, security, and availability. In addition, the more digitalized our lives become, as we rely more and more on technology in our daily lives, the greater the chance that our personal information may become compromised. From Internet Service Providers to companies gathering data to monetize and/or sell to hackers attempting to steal personal details for malicious purposes to companies that use such data for targeted advertising and data gathering, everyone and everything we interact with online may be tracking us. A virtual private network, or VPN, protects user data connections and hides where the user is. We can use VPN to browse the web without fear of being tracked. We will briefly discuss what a VPN is, how it functions, how to establish it, and the many modes and security features of VPNs. For the implementation part, we will be implementing three private networks across three separate regions (the first network in IRAQ—Erbil, the second network in Kazan—Innopolis, and the third network on AWS—Frankfurt) and connecting them using the provided solutions

and services. The network topology for the OpenVPN deployment we will be using is shown in Figure 1.

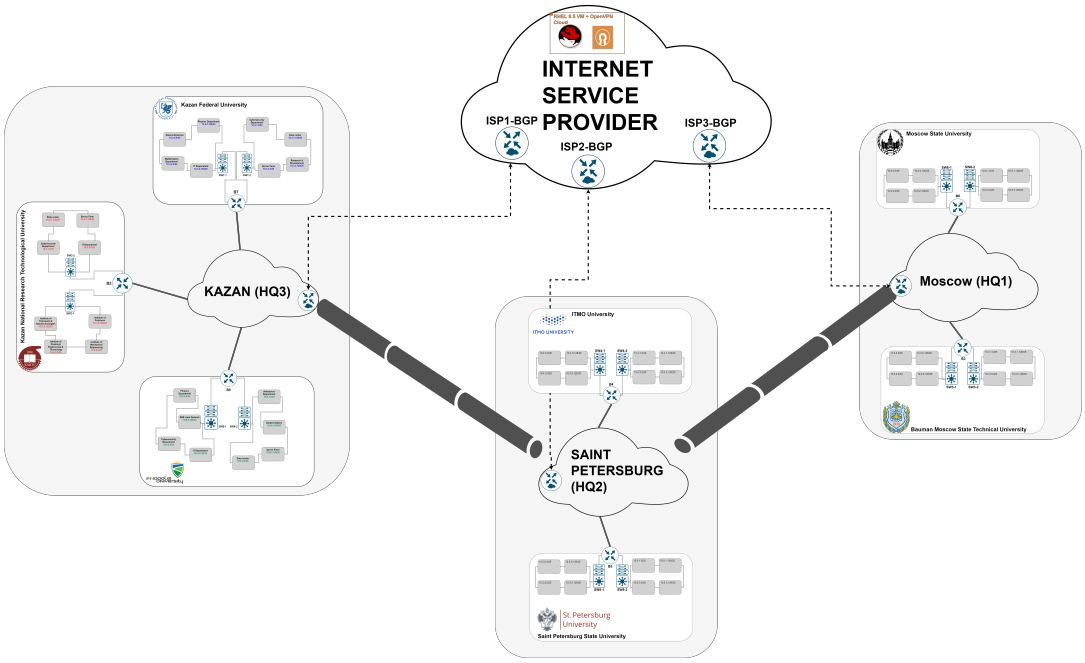

Figure 1: SSN Project Network Topology

The first city in our network will be Moscow; in this city, we will have two universities (Moscow State University and Bauman Moscow State Technical University) connecting to a central Cloud Services Router (CSRv1000) Cisco Router that will bridge the two universities with others in different cities. On the HQ1 Cloud Services Router, we will run a site-to-site IPsec tunnel to the Saint Petersburg (HQ2) CSRv1000 router. In Saint Petersburg, we will also have two universities connected (ITMO University and Saint Petersburg State University). We will set up a second site-to-site IPsec tunnel between Kazan universities and the main city router (HQ2) in St. Petersburg. In Kazan, there will be three universities this time (Innopolis University, Kazan National Research Technological University, and Kazan Federal University). Each of our cities' main city routers, CSRv1000, will be connected to a central Internet service provider for Russia, and the ISP will have three routers running in iBGP mode. In designing this network topology, employees of the seven universities distributed over Russia should be able to safely connect remotely to their university's internal network using Cisco AnyConnect. Cisco's site-to-site IPsec tunnel will be used for the site-to-site IPsec tunnel between cities, allowing universities in different cities to share their internal networks for research and development. As for the Internet Service Provider, we will implement an

RHEL8.5 server with an OpenVPN Network Connector instance running on it to bridge the whole network to the cloud. It will let us connect to the whole network from anywhere in the world.

1.1 Aims and Objectives

1.1.1 OpenVPN

- Implement OpenVPN WPC (Wide-area Private Cloud)
- Implement OpenVPN Network Connector (RHEL 8.5)
  - Remote Access
  - Site-to-site
  - Secure Internet Access
- Implement OpenVPN Host Connector (Windows 11, RHEL 8.5)
- Using Multiple Connectors to Increase Reliability of Remote Access
- Using Cyber Shield to Protect Users and Network
- Using AWS Private Hosted Zones with OpenVPN Cloud
- Securing Remote Access to AWS VPC
- Remote Access to Private Networks with Overlapping IP Addresses
- Zero Trust Application Access
- VPN with multiple VPN Egress locations
- Site-to-site Private Connectivity

1.1.2 Cisco AnyConnect Secure Mobility Client

- Cisco AnyConnect Secure Mobility Overview
- Understanding How AnyConnect Secure Mobility Works
- Supported Architectures

1.1.3 Custom VPN Solution on AWS EC2

- Algo VPN
- WireGuard
- Configuring AWS EC2 Instance

1.2 OpenVPN

1.2.1 What is OpenVPN

OpenVPN is considered one of the most popular VPN solution systems that use multiple protocols for establishing secured connected tunnels across a wide range of networks, allowing for easy and seamless remote access in two modes: routed and bridged. It can be deployed as either a client node or a server node, on-premises, or in the cloud. Users and peers can be verified using shared secret keys, certificates, or plaintext passwords. Suppose both the server and client are implemented in a server-client architecture. In that case, the server can establish connectivity with a certificate authority (CA) using digital signatures to issue authentication certificates for each connecting client. OpenVPN uses the OpenSSL library extensively in its software for encryption and the TLS protocol for establishing the tunnels. It offers a wide variety of security and administrative settings. A custom security protocol based on SSL/TLS that includes key exchange keeps it safe. It usually functions while passing across NATs and firewalls. This software, which was written and developed by James Yonan and made freely accessible, is distributed under the terms of the GNU General Public License, Version 2 (" GPLv2") (*OpenVPN*, 2022). We may also get a commercial license for running a business. OpenVPN utilizes the OpenSSL package for most of its data encryption and channel control. The protocol uses any ciphers in the OpenSSL library to encrypt and authenticate information. HMAC packet authentication offers an additional safety measure, which is referred to as the "HMAC Firewall" by James Yonan. If the hardware acceleration feature is enabled, encryption performance could be increased. TLS (Transport Layer Security) is also supported by mbed 2.3 (*Enterprise VPN*, 2021).

OpenVPN supports two authentication methods, which are TLS and pre-shared static keys. Before the tunnel is set up when using OpenVPN with a static key, a pre-shared key is made and sent to both peers. This static key has four separate keys: HMAC sends, HMAC receives, encrypts, and decrypts. Both hosts will utilize the same HMAC key and encryption/decryption key in static key mode by default. All four keys may be used separately by adding the direction parameter to -secret. A secure SSL session with mutual authentication is established in SSL/TLS mode. OpenSSL's RAND bytes function makes random bytes for encryption/decryption, and HMAC key exchange occurs after SSL/TLS authentication

is successful. In this mode, each peer uses its send HMAC, receive HMAC, encrypts the packet, and decrypts the packet keys. For key method 2, the TLS PRF function produces the keys from the random seed. Inclusion of the OpenSSL RAND bytes functions to create random bytes for use in the keys if key-method one is used. OpenVPN 1.5.0 added support for -key-method 2, and in OpenVPN 2.0, it will become the default. Once all the peers have their respective keys, the tunnel-forwarding process may begin. (*OpenVPN cryptographic layer*, 2019).

What follows is the structure of the encrypted packet:

- HMAC (explicit IV, encrypted envelope)
- Explicit IV
- Encrypted Envelope

Information carried in the body of a 64-bit sequence number, like an Internet Protocol packet or an Ethernet frame, is considered outside the encrypted envelope. This information is in the HMAC and the explicit IV. OpenSSL's RAND bytes seed a nonce-based PRNG that generates the IV for each packet. The EVP interface of OpenSSL can do HMAC, encryption, and decryption, and the user can choose the cipher, key size, and message digest for HMAC. We use Blowfish as the default encryption and SHA1 as the message digest. OpenSSL's EVP interface automatically takes care of PKCS#5 padding to an even multiple of block size. The use of CBC mode ciphers is suggested but not required. The -tls-auth directive lets the user create an HMAC key using a pre-shared passphrase to authenticate the packets that are part of the TLS handshake. This is a significant security improvement over plain TLS. The OpenSSL TLS implementation is safe from buffer overflow attacks because an attacker cannot start a TLS handshake without being able to make packets with the right HMAC signature (*OpenVPN cryptographic layer*, 2019).

1.2.2 OpenVPN WPC (Wide-area Private Cloud)

The cornerstone foundation and security efforts of OpenVPN are the OpenVPN Cloud, a safe and dependable multi-tenant virtualized NaaS (Network as a Service). Its cloud implementation includes cutting-edge security features like "Secure Access Service Edge"

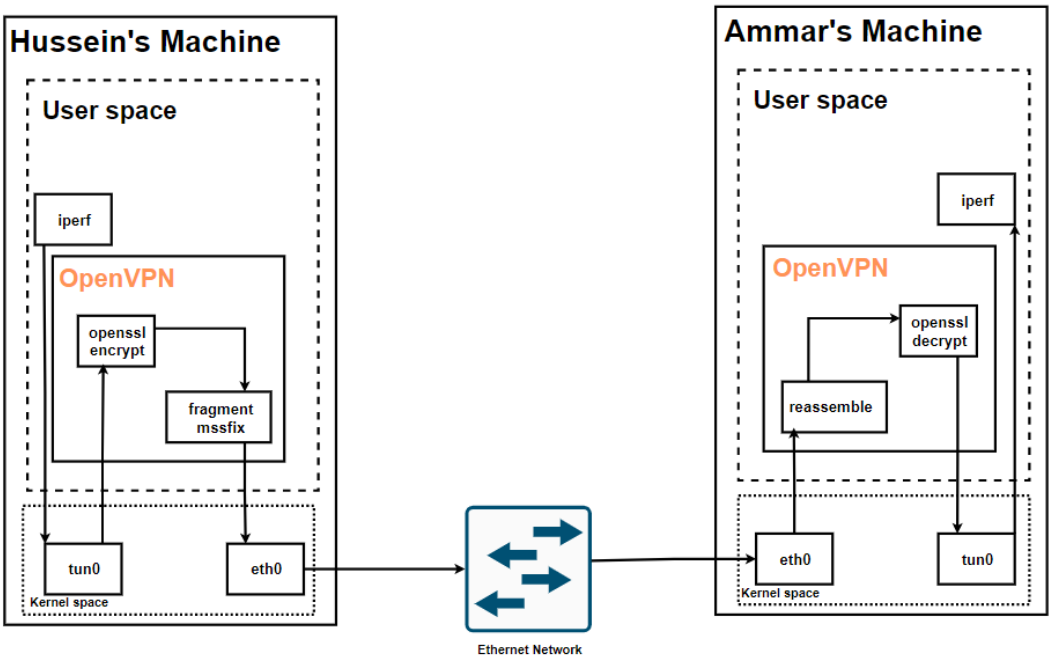

Figure 2: OpenVPN Protocol Overview

and "Zero Trust Network Access." Even though OpenVPN Cloud is a shared service, users get their virtual private network that works worldwide as soon as they sign up. An OpenVPN wide-area private cloud is a type of virtual wide network. The Cloud ID is given when we sign up, and it will be used to identify the WPC as a subdomain of openvpn.com. It will be located at [Cloud-ID].openvpn.com. Remember that it cannot be changed once a Cloud ID is selected.

The OpenVPN Cloud is a virtualization service delivered from the "cloud." Instead of a traditional network of hardware devices installed on-premise or on the provider's property, the OpenVPN Cloud uses OpenVPN Inc.'s management and scalability of software and cloud-native technologies. Connecting end client machines, servers, resources, and devices to the WPC is possible from any location with an internet connection because of the network's widespread points of presence. All WPC users can access their portals anytime by going to HTTPS://[Cloud-ID].openvpn.com. We can access the connection profile for the given WPC using the same URL and log in to the Connect Client program. As a result, the network serves a large geographical region. Even though we use the public internet to communicate, our data will stay safe thanks to the tunneling of the OpenVPN protocol and the fact that OpenVPN Cloud can support many tenants.

Chapter 2

Methodology

As stated in the introduction part of this report, we will be implementing the network architecture as shown in Figure 1. To begin with, we will use OpenVPN Wide-area Private Cloud to connect remotely to it and allow any user to access the private networks from anywhere in the world. We have created a temporary email account for this project: **sne22.innopolis@gmail.com**, which we will be operating for registering on OpenVPN and implementing the deployment in detail in this report. Finally, the TAs and instructor will have access to this deployment and its architecture.

We will begin the project implementation by navigating to **https://openvpn.net** website and creating an account using the test email provided for this project, as demonstrated in Figure 3. After the registration process, we will be introduced to two methods regarding our deployment scenario. Access Server is for self-hosting the VPN server, and it is for on-premise deployment. Figure 4 shows that OpenVPN Cloud is used to build a wide-area private network for our company, connect it to the cloud, and run it from the cloud. After choosing the WPC approach for our scenario, we will need to tell OpenVPN the name of our private cloud. OpenVPN will create a subdomain for our private cloud (see Figure 5). Later, we will register for a public domain name using GoDaddy and implement a temporary DNS forwarding record (302) for this subdomain. After successfully setting up the wide-area private cloud project, we will see the dashboard, as shown in Figure 6. With our free base subscription, we will be able to connect to the private cloud three times at the same time. Any additional connections will require upgrading the subscription to allow for five or more concurrent connections.

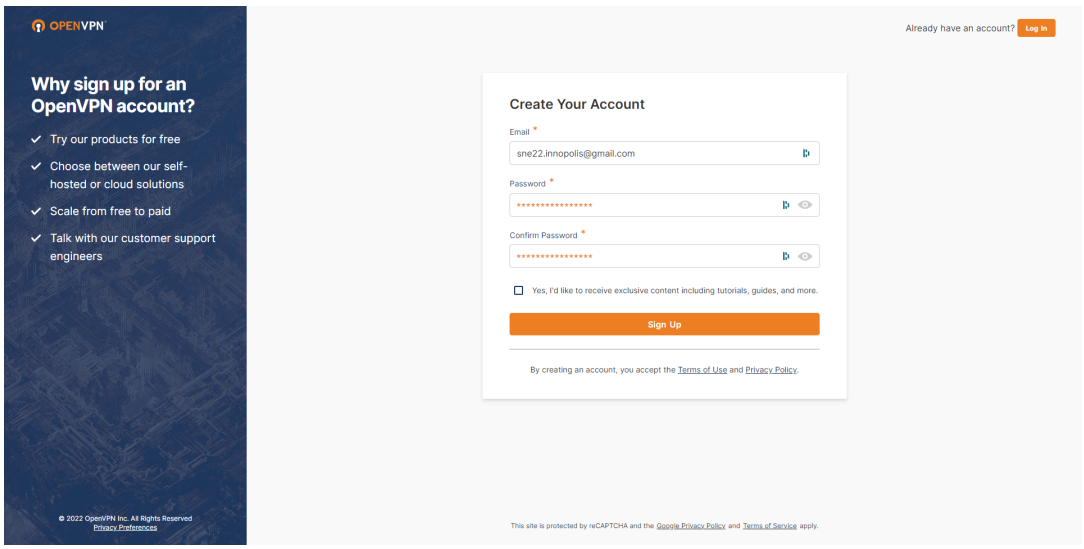

Figure 3: Registering for OpenVPN Account

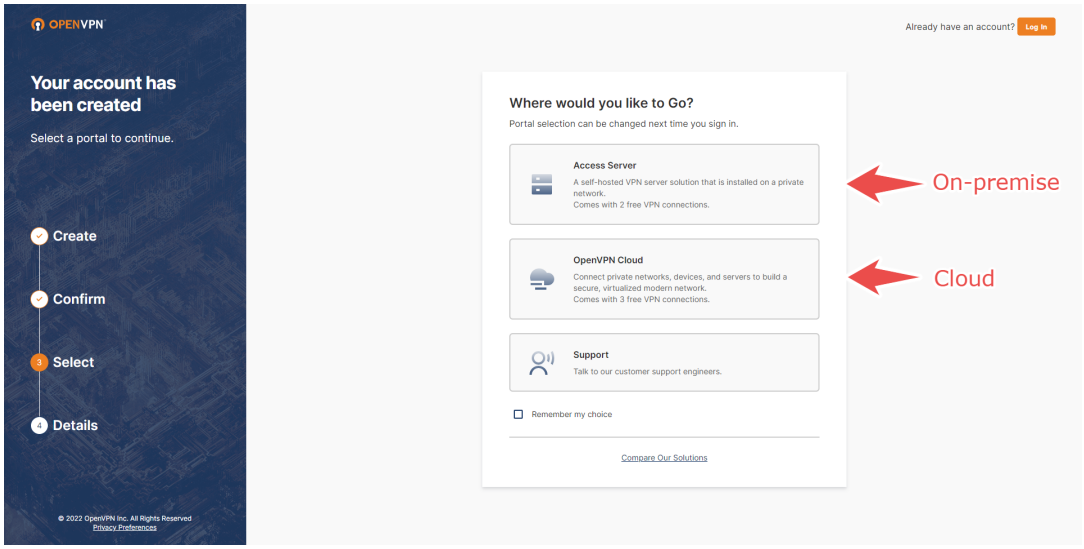

Figure 4: Choosing Deploymet Method for OpenVPN

Figure 1 shows the architecture of our on-premise network. To connect it to the private cloud, as shown in Figure 7, we will make a network connector point and choose the scenarios that apply to our implementation method. Figure 8 is a diagram that shows an overall view of the remote access idea and network scenario. On the left, users access SNE Wide-area Private Cloud from their computers using the OpenVPN Connect program and download the VPN profile from **https://sne.openvpn.com** portal. Users can establish an encrypted connection to the SNE Private Cloud region that is geographically nearest to them. The SSN Project Network Connectors, shown on the right, create a site-to-site encrypted VPN connection from the SSN private network to the Frankfurt OpenVPN region (the nearest

Figure 5: Creating Wide-area Private Cloud

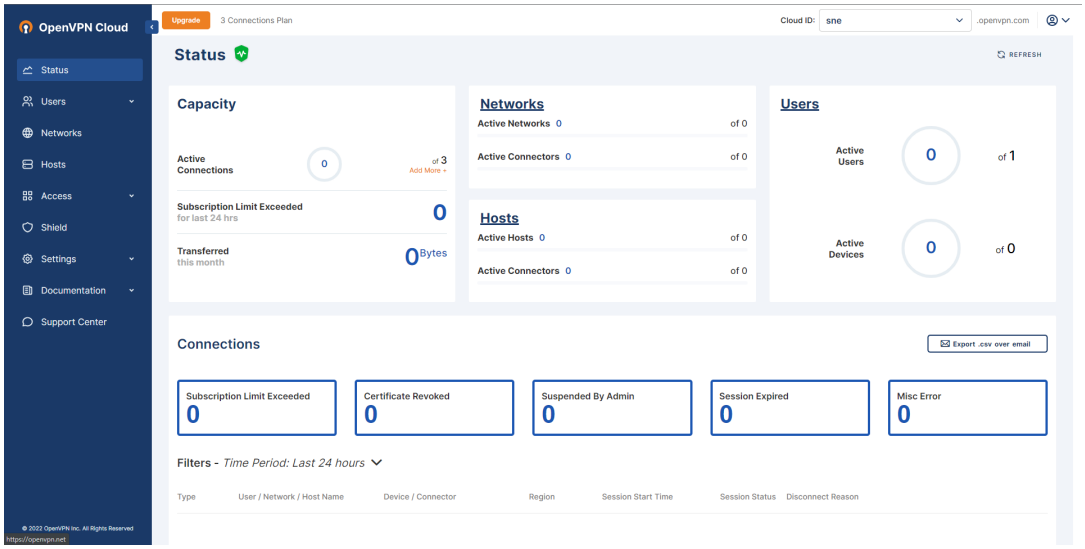

Figure 6: OpenVPN Cloud Dashboard - SNE Private Cloud

region) so that users may access the internal network over the cloud. It is how the setup process looks with split tunneling enabled. We can quickly get end users up and running with safe and dependable remote access.

Using a site-to-site network scenario, as shown in Figure 9, we can set up multiple network connectors with different private subnets (private IP addresses that overlap need to be set up differently). Furthermore, bridge them using OpenVPN cloud infrastructure by configuring routing for each site. Users in each site can access resources in other sites by routing through VPN tunnels using the WPC. This deployment uses a network architecture called "hub-and-spoke," and all traffic between sites will go around the OpenVPN cloud. We can configure

access control and shape traffic flow from the cloud. Through the OpenVPN Cloud, we can turn off split tunneling for end users and control how they connect to the internet. It will give us a holistic view of all the traffic coming in and out of our networks.

The OpenVPN Cloud management console could set up Secure Internet Access for all users and be the last point of network traffic for all users. We can turn on Egress to force all traffic through WPC or turn it off to find public resources using domain names or IP addresses. If we turn off the split tunnel, we can limit which users, networks, and hosts can connect to the internet.

Figure 10 depicts SNE WPC's naming convention and connection description. It is part of the deployment of our network connection. We will deploy three regional links for load balancing and high availability. Since Frankfurt has the lowest latency, it will house the top link. The second was sent to London, while the third was sent to Madrid. However, we will install the first connector for Frankfurt on an RHEL8.5 system during installation and show how it works. We can repeat the same process on other machines to have multiple network connectors' high availability and load-balancing feature.

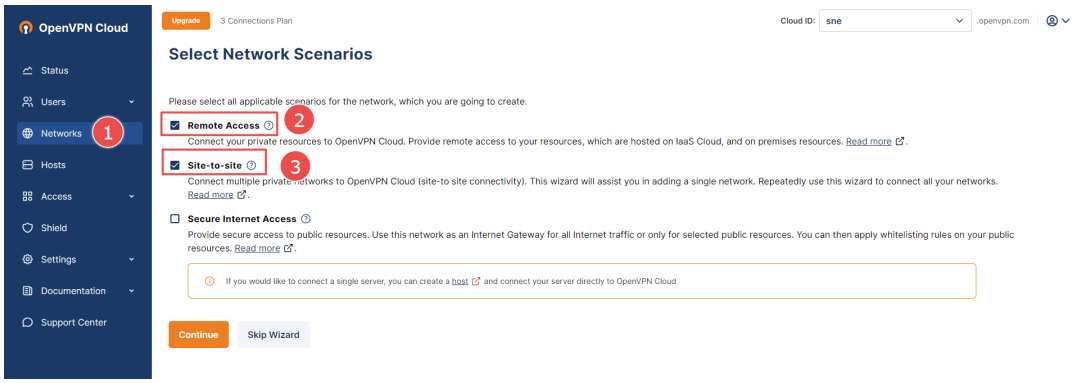

Figure 7: Creating Network Connector - Remote Access and Site-to-site

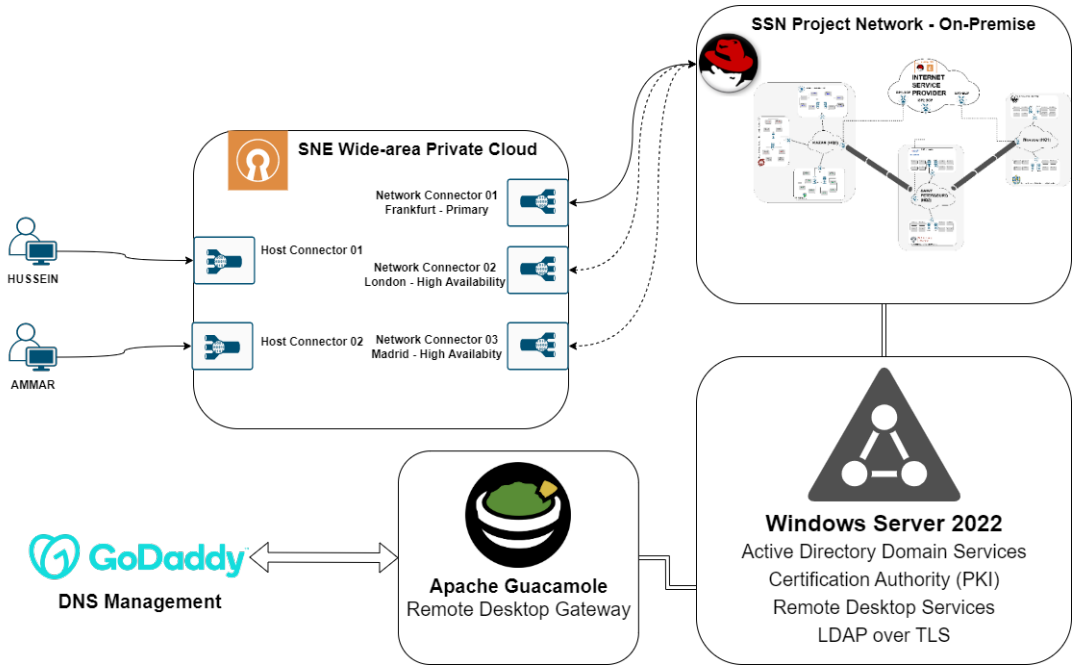

Figure 8: Remote Access VPN Network Scenario

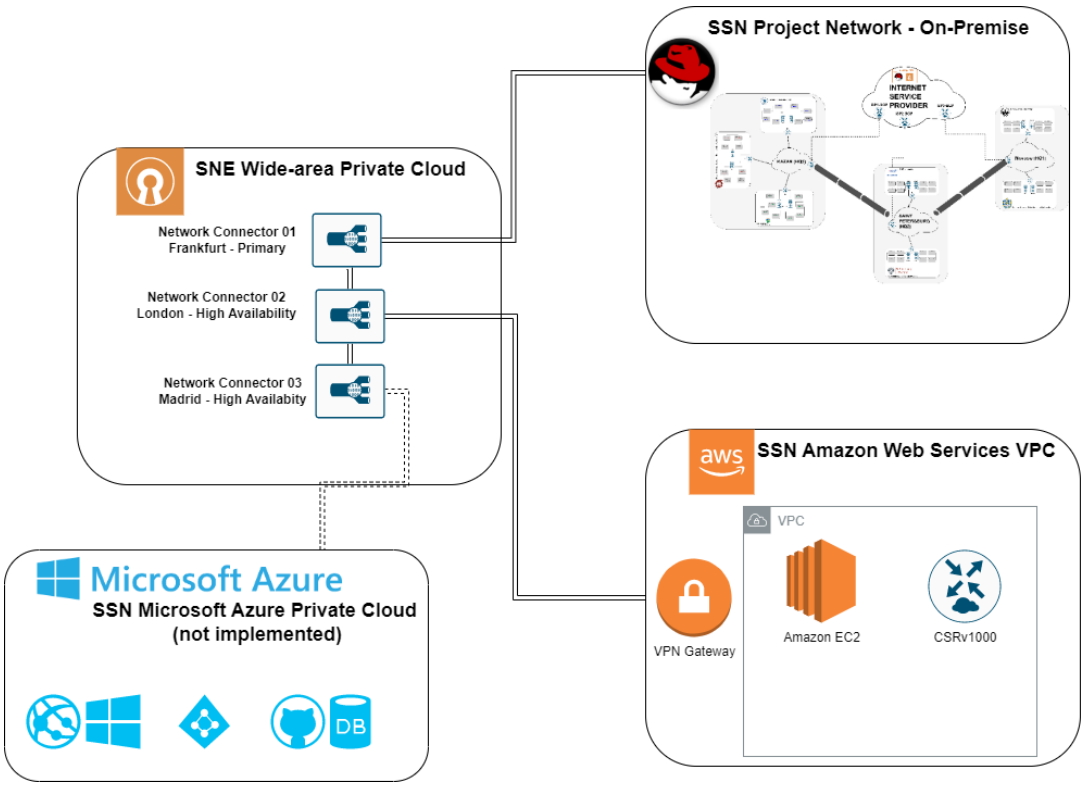

Figure 9: Site-to-site VPN Network Scenario

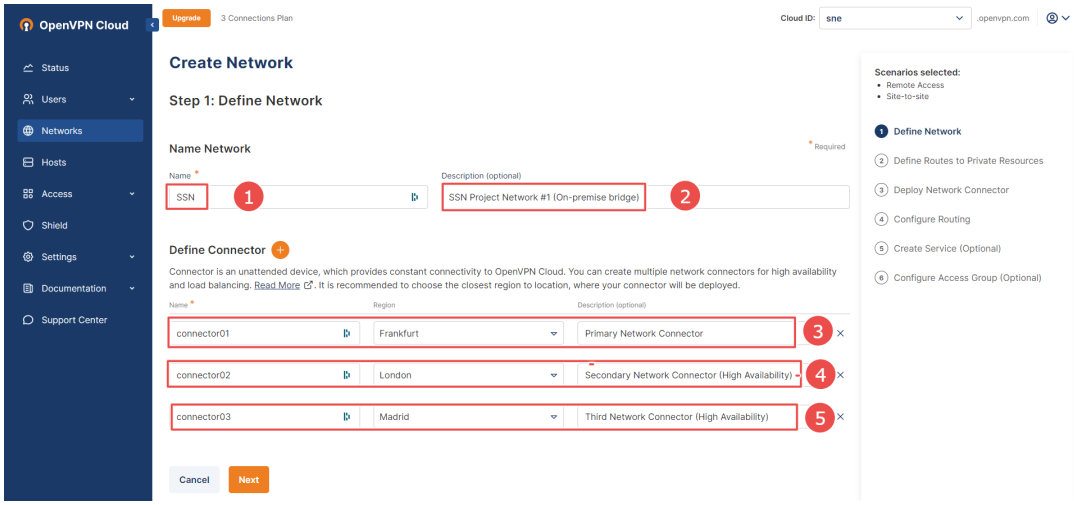

Figure 10: Defining Network Connectors for SNE WPC

Figures 11 and 12 show how our private network subnets (shown in Figure 1 of the SSN Project Network Architecture) are set up on the Network Connector. The OpenVPN Connect program will add these routes as static routes that point to the VPN virtual tunnel gateway. Figure 35 shows how we will have to set up these subnets by hand as static routes on the OpenVPN gateway on RHEL8.5. It will finish setting up our Network Connectors, and Figure 13 should show that they are ready to be used. Multiple distributions are supported to implement these network connectors. We can also deploy it on firewall appliances by extracting the CA certificate, client public key, private client key, and TLS-authentication unidirectional key from the Network Connector profile file **.ovpn** and connecting to the SNE WPC.

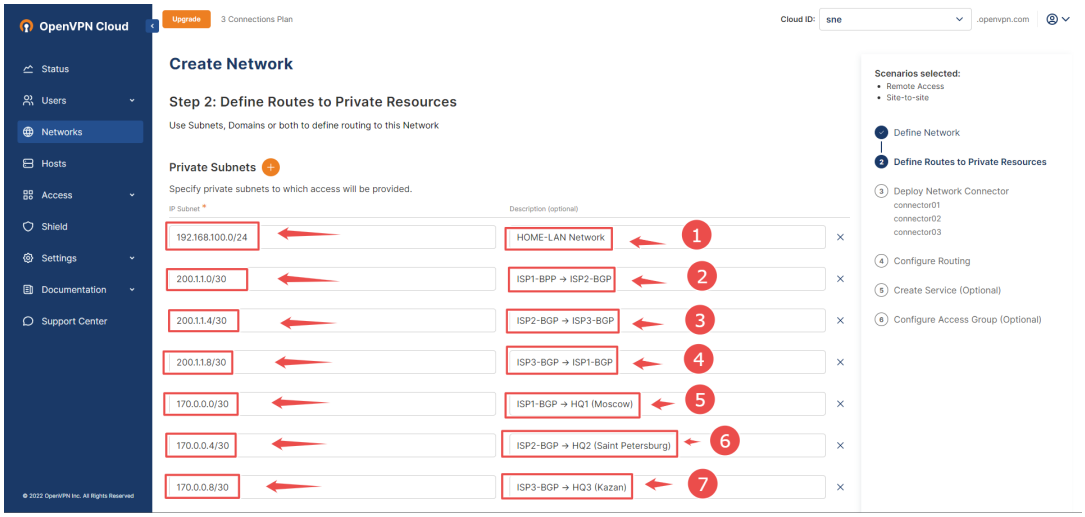

Figure 11: Defining Routes to Private Subnets (1)

Figure 12: Defining Routes to Private Subnets (2)

Figure 13 is the final step in configuring our network connector. We select the preferred operating system and distribution for deploying the connector on-premise. We can also deploy it on a public cloud provider such as Amazon Web Services, Microsoft Azure, or Google Cloud Platform. Then, we will receive either a script or an **.ovpn** profile containing all the configurations in the previously explained steps, ready for installation.

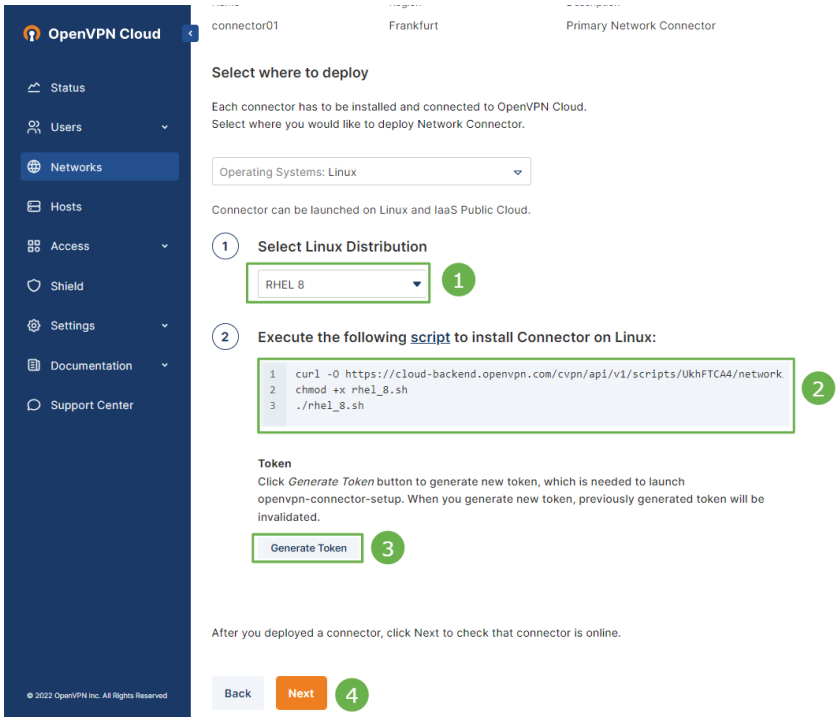

Figure 13: Deploy Network Connector on OS/Distribution

The Network Connector script will install the required dependencies by enabling EPEL, CodeReady-Builder, and dsommers/openvpn3 repositories on the system. It will deploy the python3-openvpn-connector-setup tool, which will be executed at the end of the script. The connector will also turn on the system's IPv4 and IPv6 forwarding and set up firewall rules to let traffic go through NAT. Figure 15 shows how the script is run, and Figure 16 shows the complicated commands.

We will need to generate a token in WPC and enter that token in the script to import the WPC connector profile, enable the OpenVPN systemd service, and start the service, as shown in Figure 17. This node will always be connected to the SNE WPC Network, which is hosted in the cloud. We will need to set up static routes for our devices in the network to point to the RHEL server as a gateway. This step needs to be set up so we can connect to the devices in our SSN network deployment. Figure 18 shows the systemd service and the

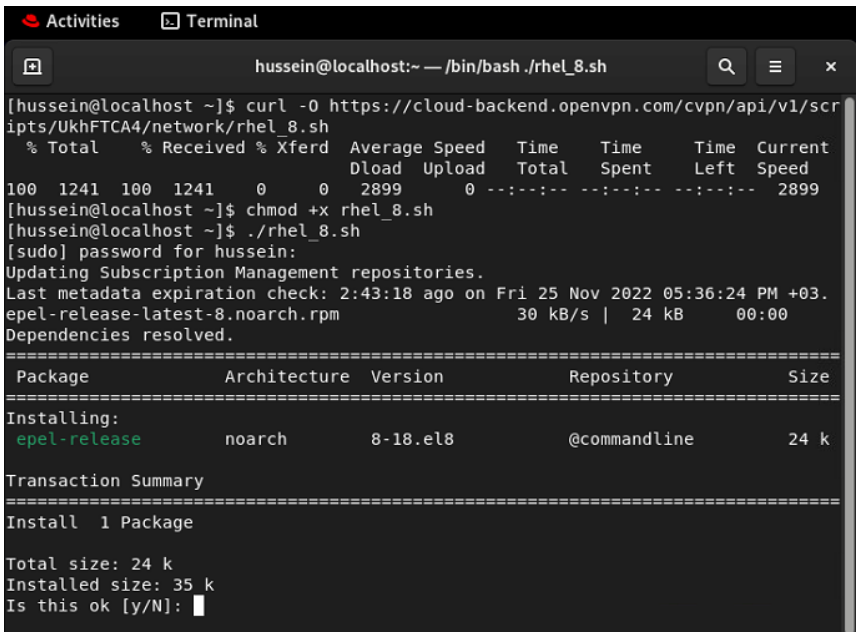

Figure 14: Deploy Network Connector on RHEL8.5

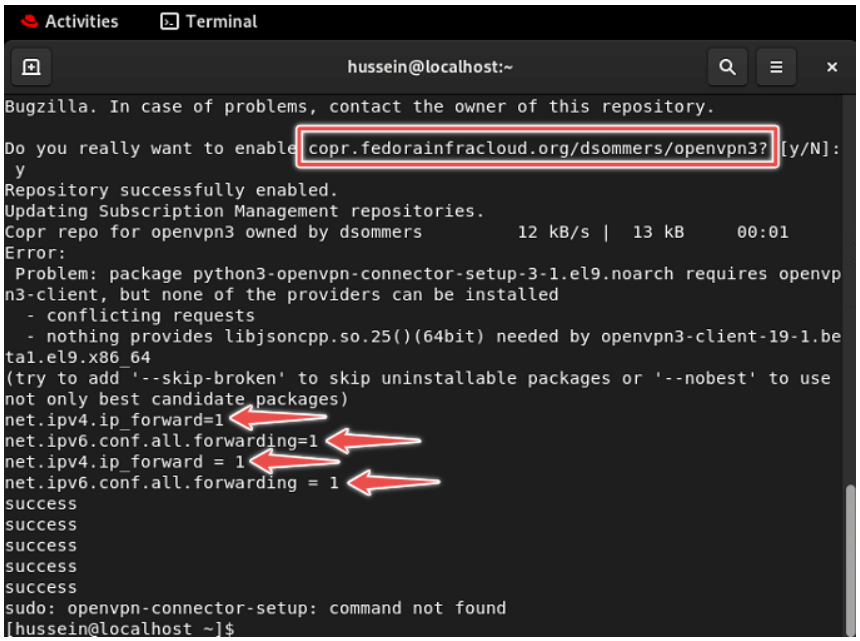

Figure 15: Network Connector Script Execution

name of the default connection profile. This shows that the deployment was successful.

OpenVPN will default to running in routed mode, as depicted in Figure 19. This mode will create a tun# virtual interface and route (NAT) traffic through the WPC subnet `100.96.0.0/11`. We can modify the source code of the OpenVPN connector client to allow for bridged mode deployment. We can verify that the connection status for the WPC node is up and running, as shown in Figure 20. We can start configuring routing from the

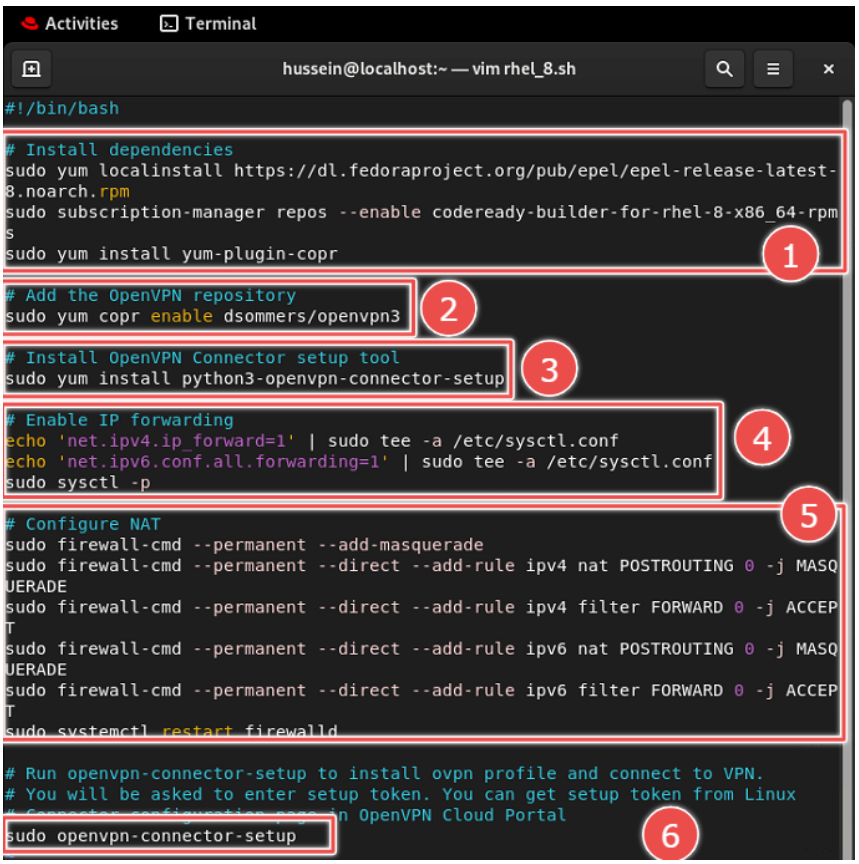

Figure 16: Network Connector Script Commands

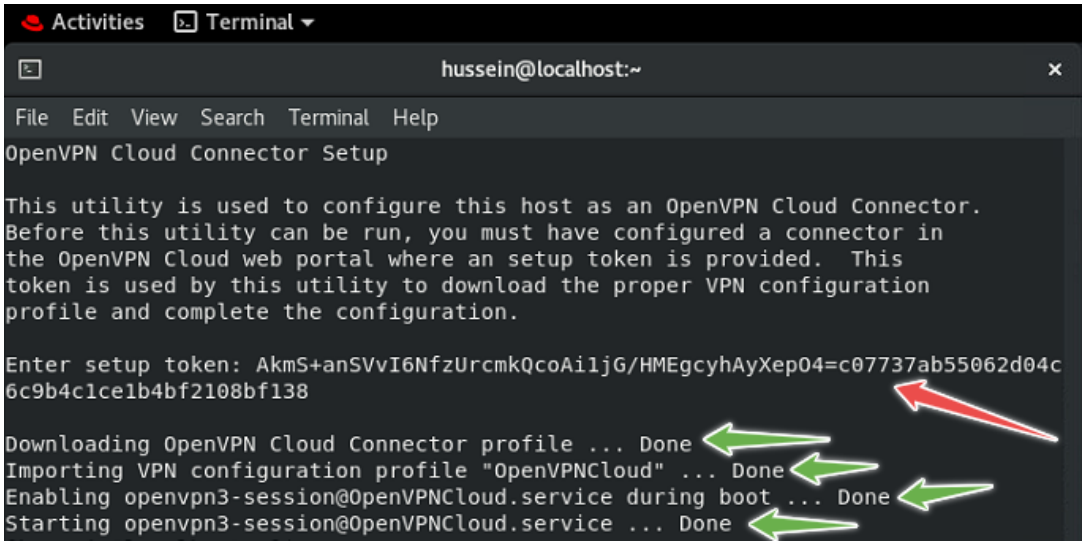

Figure 17: Generating Token for Importing OpenVPN Cloud Connector Profile

private network site (SSN on-premise network) to ensure devices from the site can route traffic to other sites and remote clients. Additionally, we could configure granular access groups by defining allocated services by domain name, IPv4 and IPv6 addresses, and ports (see Figures 22–24). We will try to cover this methodology in the report as best as possible.

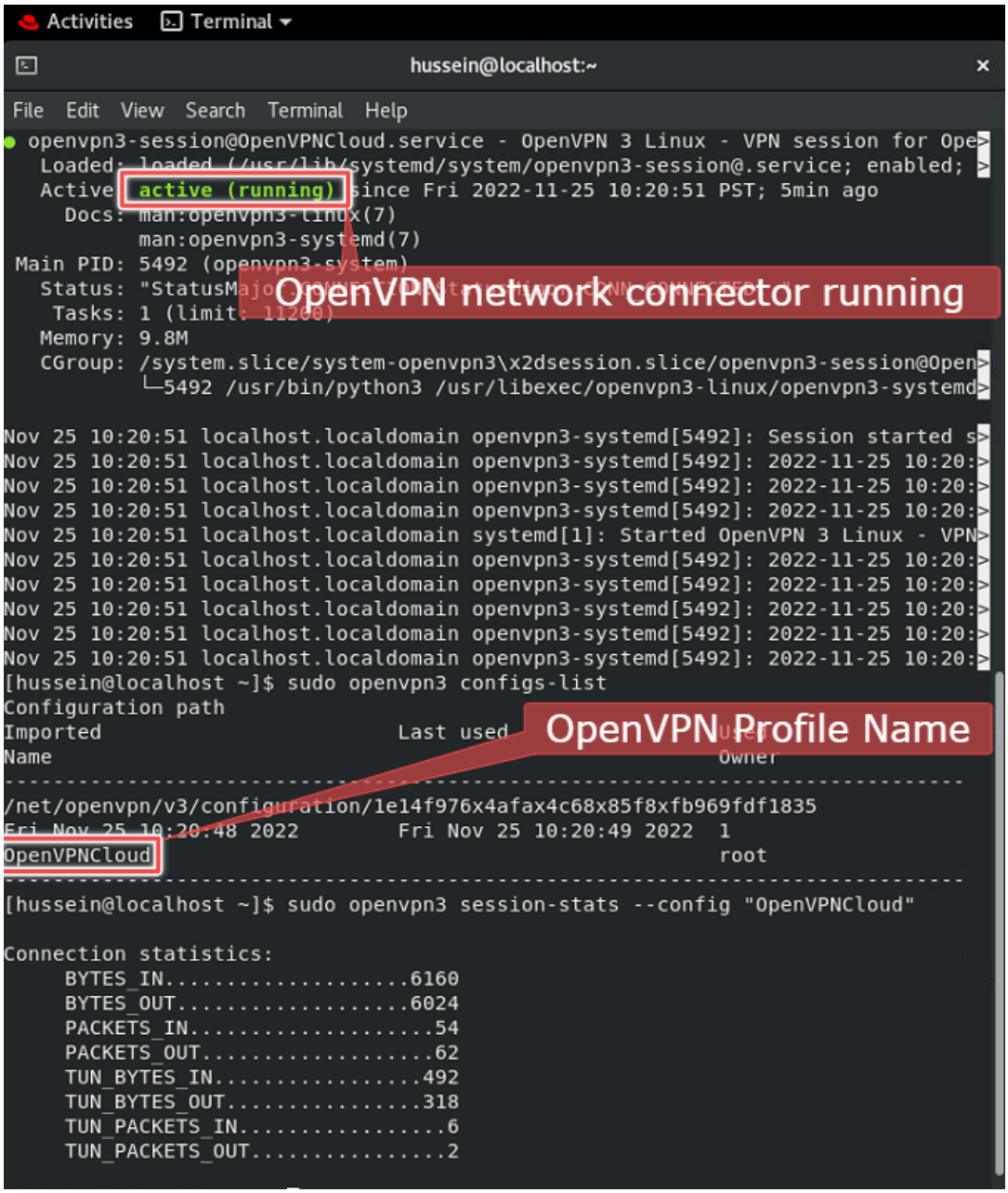

Figure 18: Network Connector Node Verification

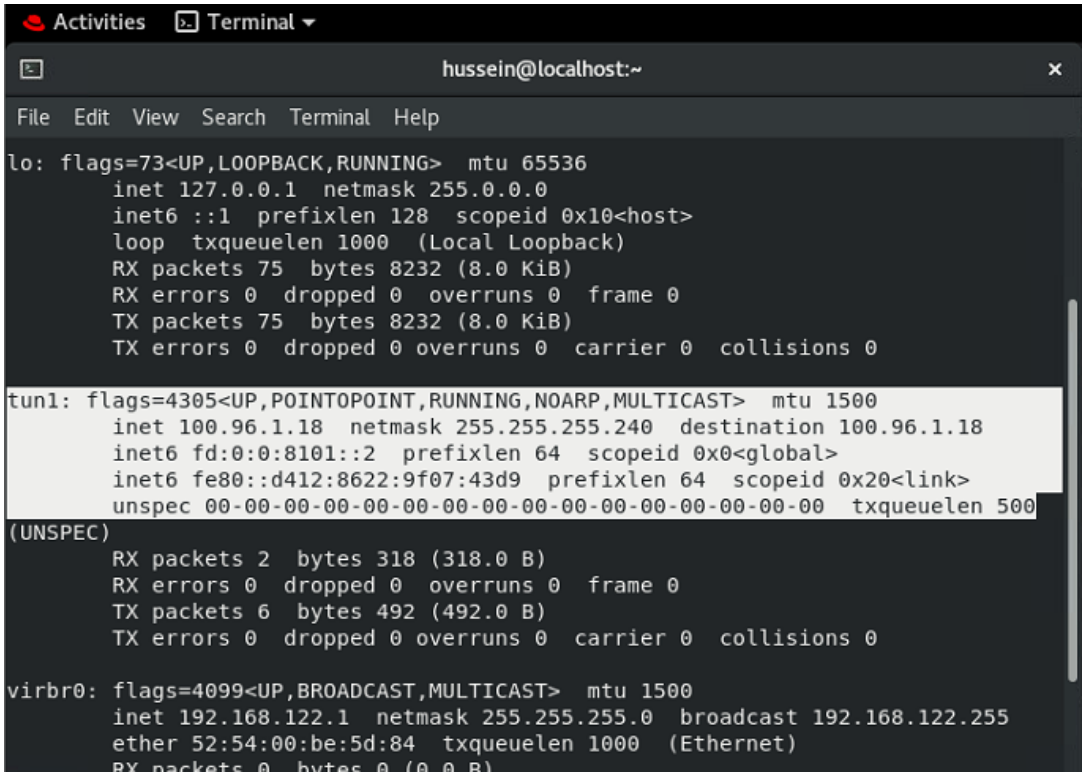

Figure 19: Routed mode with virtual tunnel interface - tun1

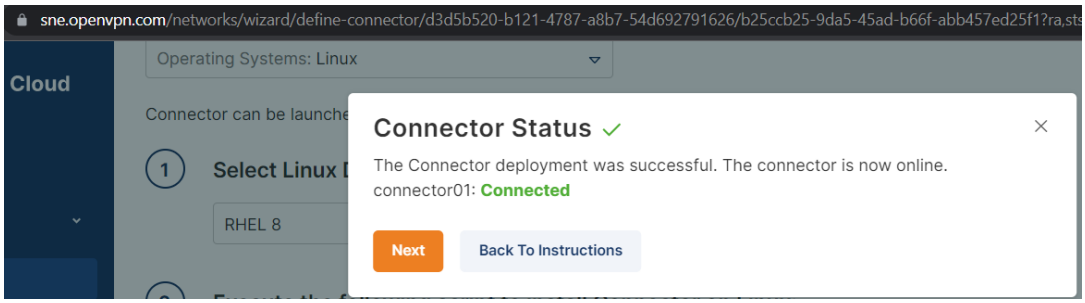

Figure 20: Network Connector Status in WPC

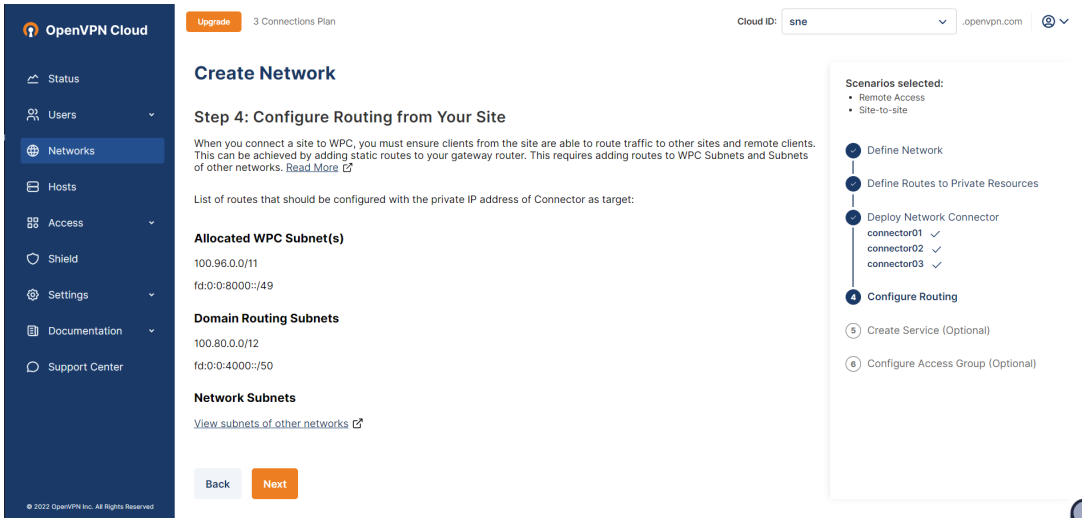

Figure 21: Allocated WPC and Domain Routing Subnets

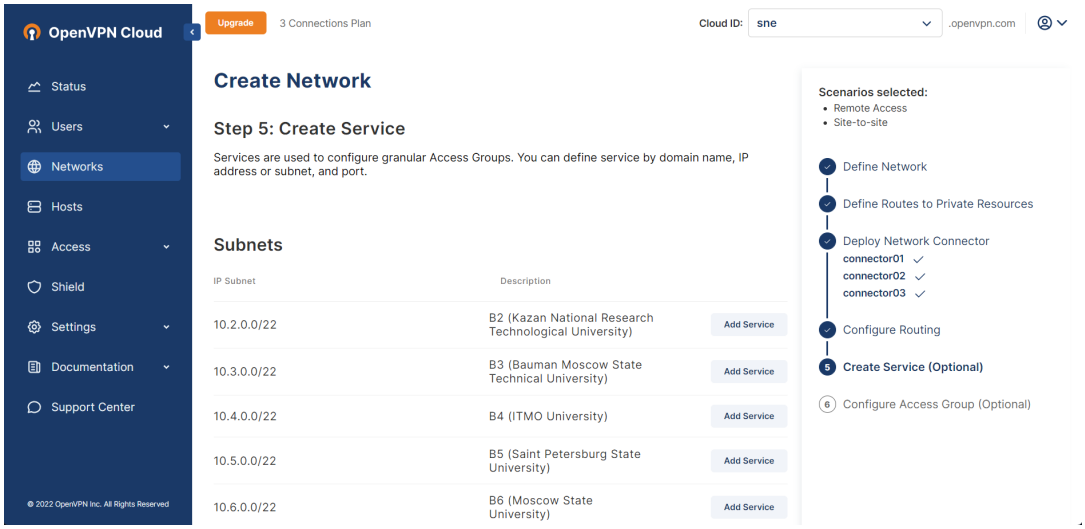

Figure 22: Create Service Definition

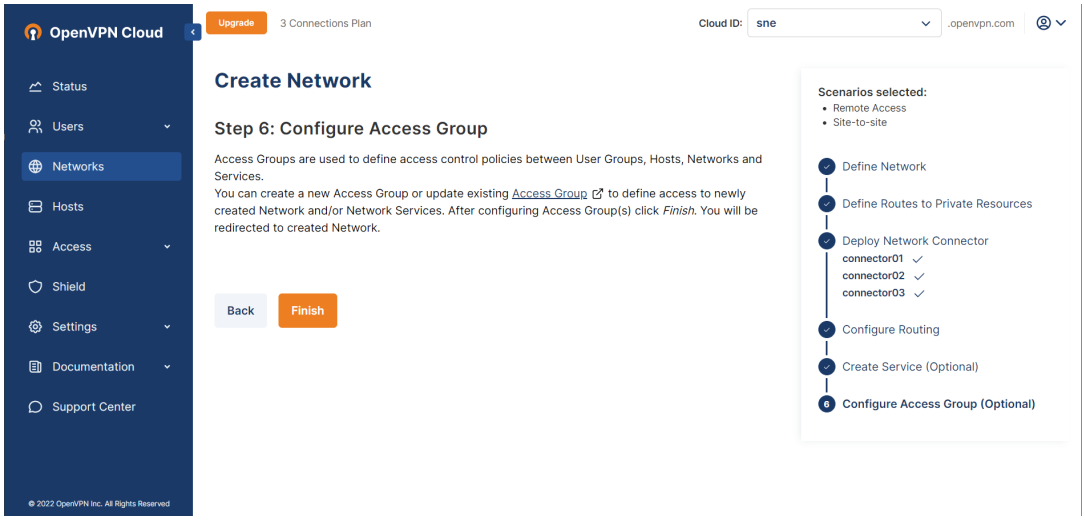

Figure 23: Granular Access Groups Definition

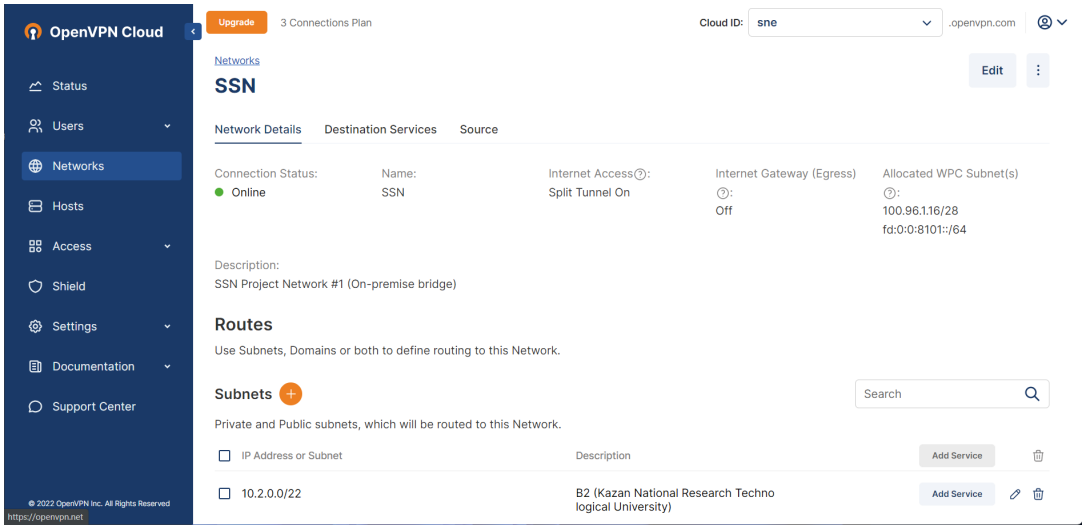

Figure 24: SSN Wide-area Private Cloud Network Online

2.1 Implementing Host Connector

To connect a host machine to SNE WPC, we will need to deploy a host connector to allow direct connectivity to the private cloud. We can do this by navigating the Hosts menu on the left panel. Figure-25. We will showcase how to set up a Windows 11 machine for remote access connectivity to the SNE WPC. Figure 26 illustrates the host machine definition in OpenVPN Cloud, and we can also define separate connectors for host networks. Multiple connector definitions are also permissible for high availability and load balancing. Once configured (Figure 27), we download the bundled profile connection **.ovpn** file and install the OpenVPN Connect application on the host machine. Import the profile configurations into the Connect application and establish a VPN tunnel session, as shown in Figures 28–30. The host machine will be allocated private IPv4 and IPv6 addresses from the assigned OpenVPN subnet in the portal, as illustrated in Figure 31.

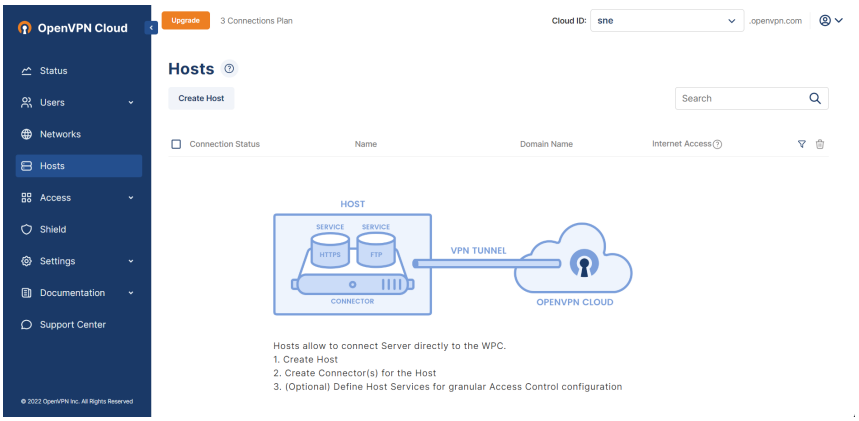

Figure 25: Create Host Connectors to SNE WPC

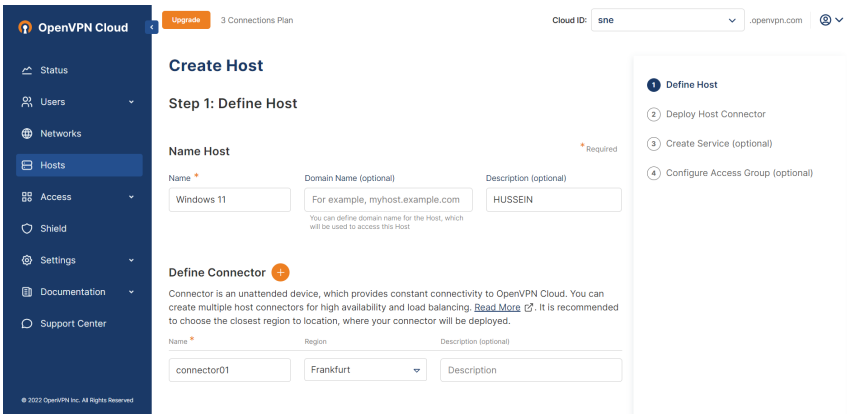

Figure 26: Define Host Name and Connectors to SNE WPC

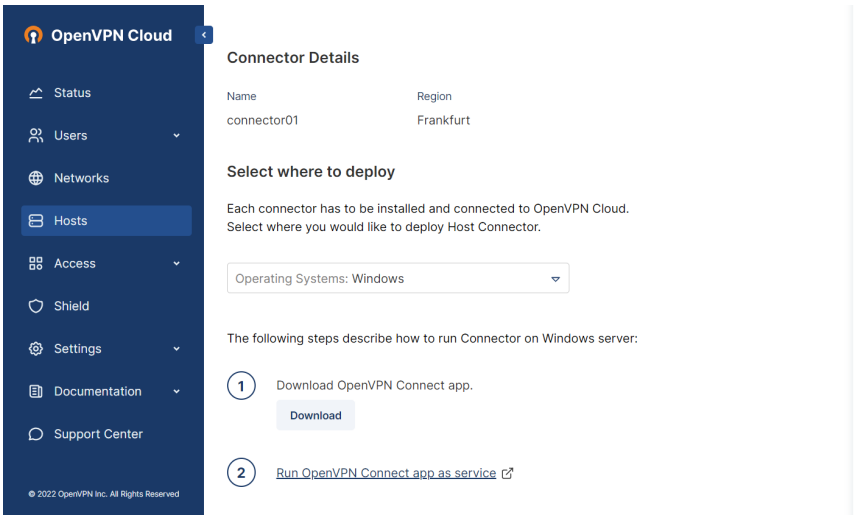

Figure 27: Host Connector OS and Bundled Connection Profile

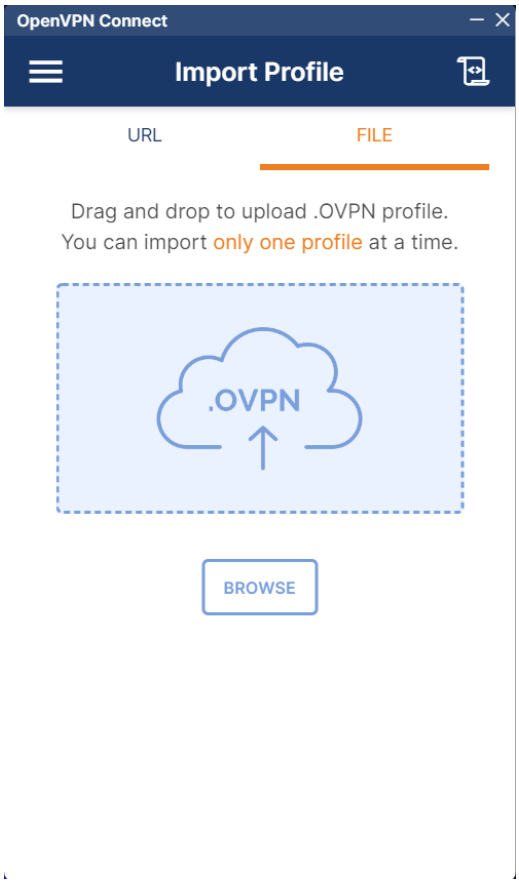

Figure 28: Import .OVPN Profile on OpenVPN Connect Program

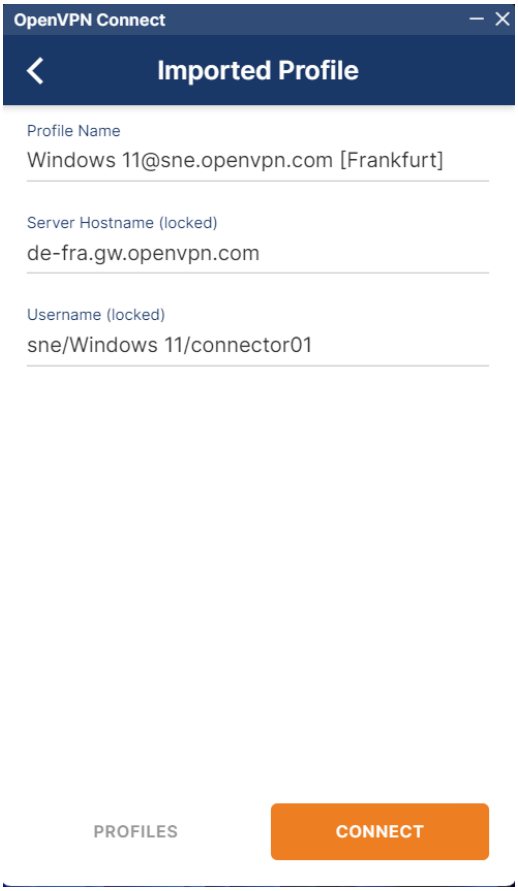

Figure 29: Imported .OVPN Profile Details

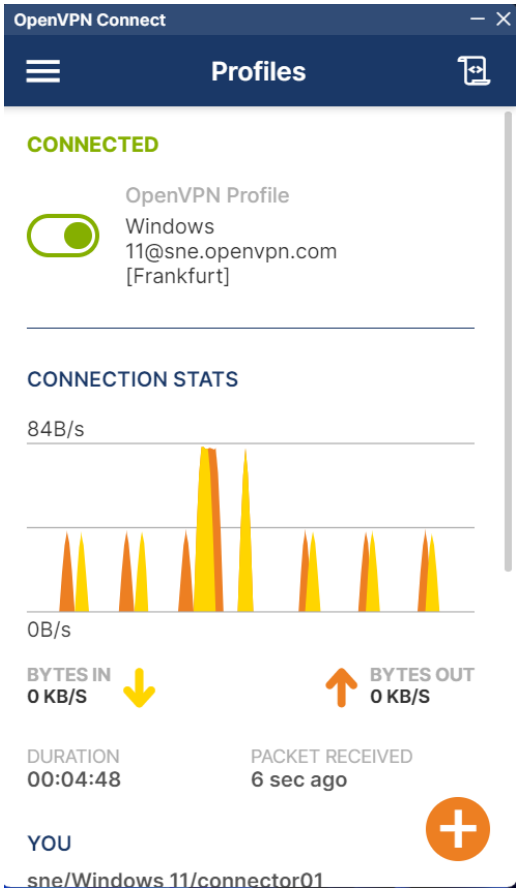

Figure 30: Host Connected to sne.openvpn.com WPC

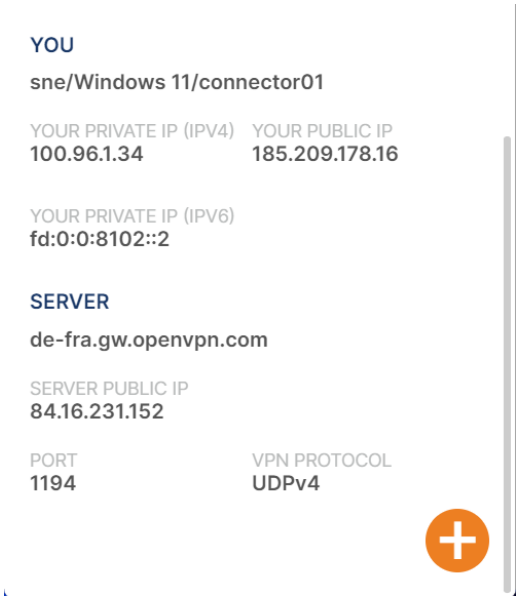

Figure 31: Host VPN Tunnel Details

2.2 Infrastructure Routing, SSHv2, and Automation

The OpenVPN Connect program will update the host machine's routing table and inject static routes with the virtual tunnel interface's next-hop address (TAP interface in bridged mode). To validate this deployment, we check the routing table on the Windows 11 host, as shown in Figure 32. In Figure-21, we can see that the network destinations for SSN on-premise network (Figure-1) routes are successfully injected with a gateway address of `100.96.1.33` corresponding to the SNE WPC subnet range of `100.96.0.0/11`. The OpenVPN Connect program can also inject IPv6 addresses; when we use an IPv6-only network in our deployment. As shown in Listing-1, we can examine the source code under the `openvpn/src/openvpn/route.c`.

Listing 2.1: OpenVPN IPv6 Routes Injection - Source Code

```
add_route_ipv6(struct route_ipv6 *r6, const struct tuntap *tt,
               unsigned int flags, const struct env_set *es,
               openvpn_net_ctx_t *ctx)
{
    bool status = false;
    const char *device = tt->actual_name;
    bool gateway_needed = false;

    if (!(r6->flags & RT_DEFINED) )
    {
        return;
    }

    struct argv argv = argv_new();
    struct gc_arena gc = gc_new();

#ifndef _WIN32
    if (r6->iface != NULL)              /* vpn server special route */
    {
```

```
        device = r6->iface;
        if (!IN6_IS_ADDR_UNSPECIFIED(&r6->gateway) )
        {
            gateway_needed = true;
        }
    }
#endif
.
.
.
//Rest of the Code is avaiable on GitHub!
//https://github.com/OpenVPN/openvpn/blob/master/src/openvpn/route.c
```

If we deploy multiple VPN endpoints on the network connector node, we can change the routes OpenVPN created with a bash script. This is an ideal setup for VPN failover, as we can integrate multiple vendor solutions into our network as a disaster recovery and business continuity planner. The script is included in Listing-2, and we can update the routes by passing them to the script using environment variables. SSN's on-premise network configuration represents a typical WAN configuration in which Windows machines share a network using the verb—192.168.100.190— Windows Server 2022 Active Directory Domain Services. Access to the OpenVPN Cloud will be facilitated by RHEL8.5 server routing, as it is not a platform-dependent deployment. The RHEL server will establish a tunnel to the SNE private cloud and transmit data between the SSN LAN network, the internet, and the SNE WPC. For the RHEL server to forward packets among its LAN ports, routing must be activated on the device, and NAT is allowed on the firewall.

Listing 2.2: bash script to update routes for vpn multi-endpoints

```
#!/bin/sh
nvram set wan_ipaddr=$ifconfig_local
nvram set wan_ifname=$dev
nvram set wan_default=$dev
killall upnp
```

```
upnp -D -W $dev
iptables -t nat -A POSTROUTING -o $dev -j MASQUERADE
for R in /tmp/resolv.conf /tmp/resolv.dnsmasq; do
mv $R $R~
for O in "$foreign_option_1" "$foreign_option_2"; do
P="$O"
par1=$(echo "$P" | cut -d " " -f1)
if [ "$p1" == "dhcp-option" ]; then
par2=$(echo "$P" | cut -d " " -f2)
par3=$(echo "$P" | cut -d " " -f3)
if [ "$p2" == "DNS" ] ; then
echo "nameserver $par3">>$R
fi
fi
done
```

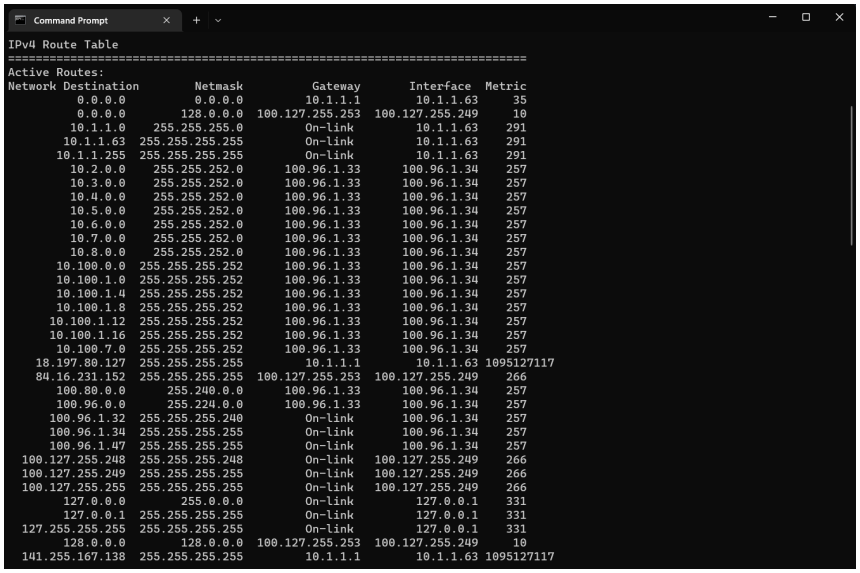

Figure 32: OpenVPN Host Routing Table

As our network deployment gets quite large, we will start preparing the Cisco network devices for remote management using SSH version 2. Then we can proceed and set up Ansible to enable Infrastructure-as-Code automation. Before configuring SSHv2 on the network devices, we will use Figure-33, which shows the verb—OpenSSL 3.0— hashing a plaintext pass-

word with a 4-letter salt keyword, to configure a Cisco type-5 secret password for additional security best practices. The salt must be a 4-letter keyword. Otherwise, Cisco devices will throw an exception for not matching their hashing algorithm implementation in their devices. **Note: newer Cisco IOS versions support larger salt lengths and SHA224,256,384,512 built-in hashing algorithms.**

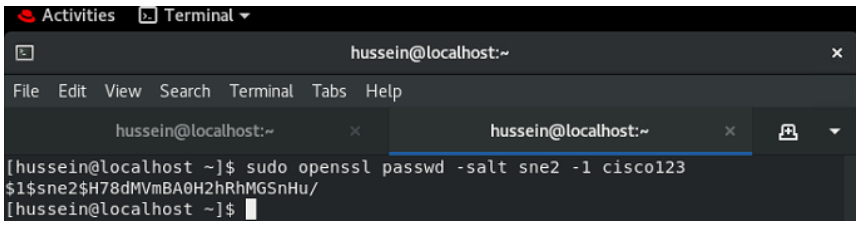

Figure 33: Hashing SSHv2 Type 5 Cisco's Password with OpenSSL

A sample configuration for SSHv2 is shown on the ISP1-BGP router in Figure 34. First, we set up the admin username with privilege 15 (later changed to privilege 1) and type 5 secret of the hashed password in Figure 33. It will prevent prying eyes from extracting the SSH password from the router's configuration file. Second, create the domain sne.openvpn.com, which will be required during the crypto RSA key generation process. Third, we generate the RSA keys with a modulus of 2048 bits (later changed from 1024 bits) and enable SSH version 2 only. Fourth, enter line virtual teletype 0 to 15 and enable local authorization through the local login command. Fifth, enable SSH only and disallow telnet on the virtual teletype lines. It should meet our Ansible deployment requirements in later steps.

```
ISP1-BGP>enable
ISP1-BGP#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ISP1-BGP(config)#username admin priv 15 secret 5 $1$sne2$H78dMVmBA0H2hRhMGSnHu/
ISP1-BGP(config)#ip domain-name sne.openvpn.com
ISP1-BGP(config)#crypto key generate rsa
The name for the keys will be: ISP1-BGP.sne.openvpn.com
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]

ISP1-BGP(config)#
*Mar  1 01:09:19.571: %SSH-5-ENABLED: SSH 1.99 has been enabled
ISP1-BGP(config)#ip ssh version 2
ISP1-BGP(config)#line vty 0 15
ISP1-BGP(config-line)#login local
ISP1-BGP(config-line)#trans
ISP1-BGP(config-line)#transport input ssh
ISP1-BGP(config-line)#end
ISP1-BGP#wr
Building configuration...
[OK]
ISP1-BGP#
*Mar  1 01:09:45.927: %SYS-5-CONFIG_I: Configured from console by console
ISP1-BGP#
```

Figure 34: Cisco's SSHv2 Configuration

We put static routes on the RHEL server to point toward the internal SSN network located within a virtual machine on the same subnet. These routes are set on the OpenVPN wide-area private cloud and will allow us to route traffic on the tunnel exit system. It will enable semi-routing on the server for the SSN Network architecture (Figure 1). (Figure 1). This is seen in Figure 35.

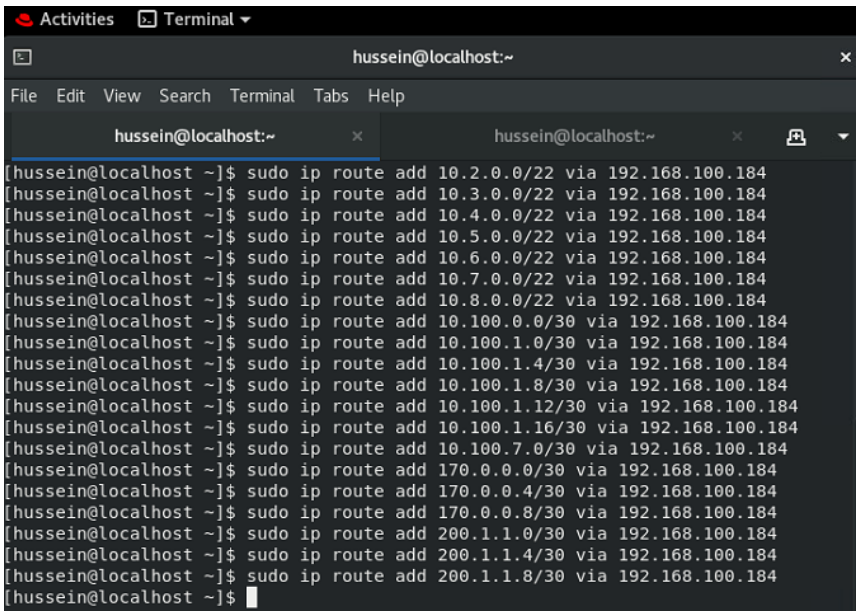

Figure 35: Static Routes Configuration for SSN Network

We check the updated configurations on the RHEL8.5 routing table, as shown in Figure 36. ISP1-BGP will be the gateway for the internal SSN network machines and devices. Any similar implementation should be matched with OpenVPN Wide-area Private Cloud subnets configuration. Otherwise, routing will not work as intended. To establish a default gateway in RedHat, network scripts read the file configuration located in /etc/sysconfig/network and any up-network interfaces. In our case, the SSN network subnets will use the ISP1-BGP gateway, which is different from the default gateway. Therefore, the GATEWAY directive may be used either globally or in interface-specific configuration files to define the default path for a chosen network subnet. The GATEWAYDEV directive is a system-wide alternative. The latter will take effect if we have multiple devices that meet the GATEWAY specification but only a single interface uses the GATEWAYDEV directive. If time permits, we could configure this alternative on an RHEL 9.1 system, as it offers more flexibility when working with IPv6 addresses and routing.

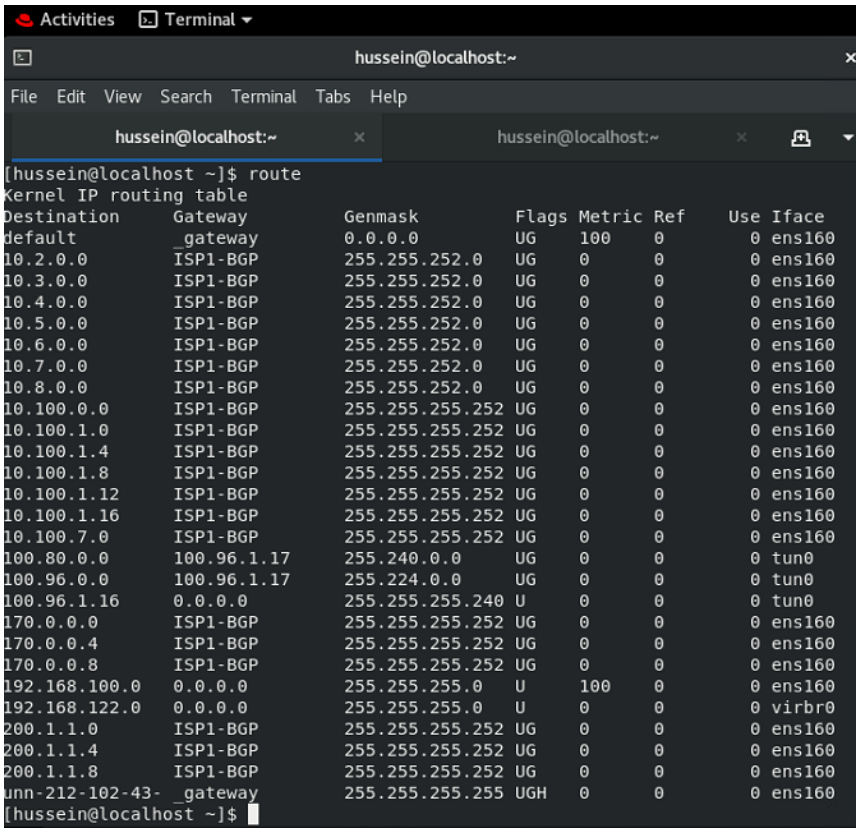

Figure 36: SNE WPC Network Connector - Routing Table

2.3 Active Directory Domain Services

We will install and configure Active Directory Domain Services on Windows Server 2022 using the data center 180-day evaluation license to configure directory services integration and authentication for our environment. Active Directory is a database that organizes data for networked items in a tree-like layout. Active Directory Domain Services (AD DS) is an example of a directory service that offers the means for storing and making accessible directory data to users and administrators of a network. User account data, including names, passwords, and more, are stored in AD DS and may be accessed by other authorized users on the same network (Justinha, n.d.).

Active Directory is considered a database that system administrators and end users may use to query against its records and access information about network objects stored on it. Using a structured data store creates a logical, hierarchical structure for directories. Information on AD objects could be found in this database by querying it, also called a directory. Servers, volumes, printers, users, and computer accounts on the network are all assets considered shared resources within the directory environment (Justinha, n.d.).

Active Directory incorporates security features such as login authentication, domain trust, federation services, certification authority, object-level access control, and role-based access control. Authorized users may get access to resources on the network with a single sign-on (SSO). System administrators can centrally manage directory data and organization as well. Even the most complicated network deployments can be easily managed with the help of policy-based administration, known as group policy objects (Justinha, n.d.).

Windows Server 2022 makes deploying and implementing Active Directory Domain Services (AD DS) easier and quicker. The installation procedure is mainly based on PowerShell, as it is integrated with Server Manager. Domain controllers may be utilized for an existing infrastructure with fewer hassles. As a result, deploying an AD environment is streamlined and expedited. We will deploy using a role-based or feature-based installation. It will allow us to specify which roles and features to deploy on a specific server instance. Furthermore, in the end, we could execute our desired state configurations or extract them as a PowerShell script. We have shown this in Figures 38–43. Figure 38 specifies the setup on a single server instance for demo purposes (Justinha, n.d.).

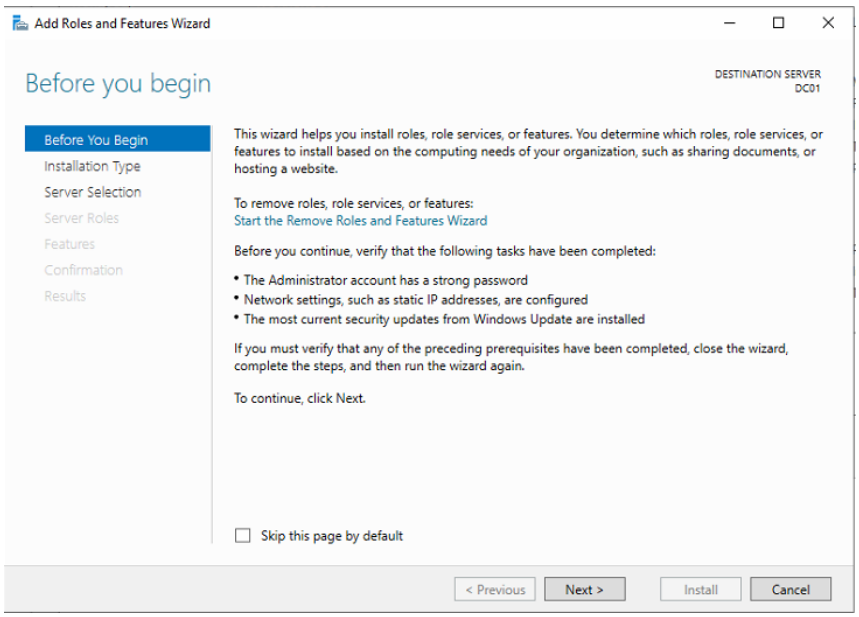

Figure 37: Active Directory Domain Services Deployment on Windows Server 2022

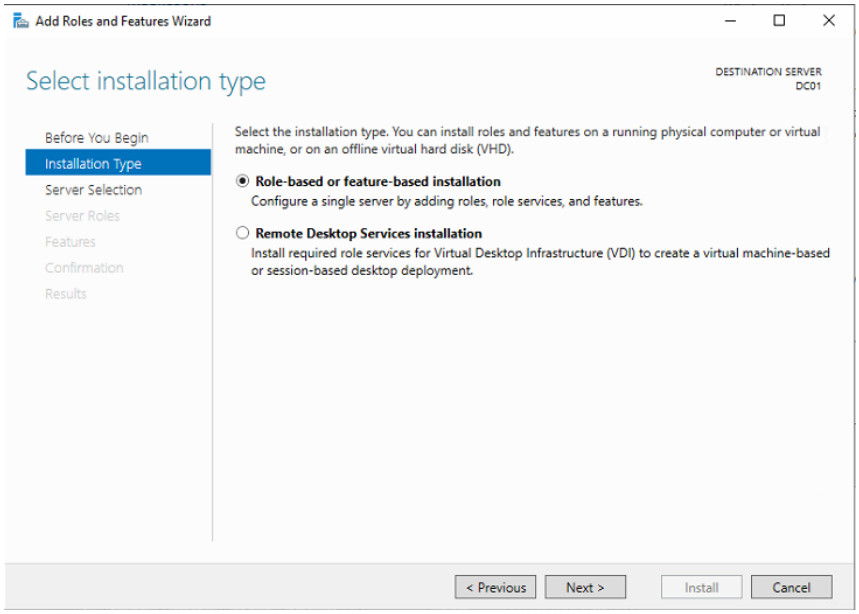

Figure 38: Configuring single server instance for Roles and Features

The next option to specify is the server or the virtual hard disk instance on which to install the AD DS roles and features. We can do an unattended installation of AD on an offline Windows Server hard disk. Furthermore, the same installation wizard used in Figure 39 can be installed in a Windows Clustering environment.

The AD DS dependencies shown in Figure 40 involve the ones shown here. The majority of the dependencies in this section are for GUI-Server deployment. We could decrease the size of the dependencies if our Windows Server deployment utilized the Core version (command-

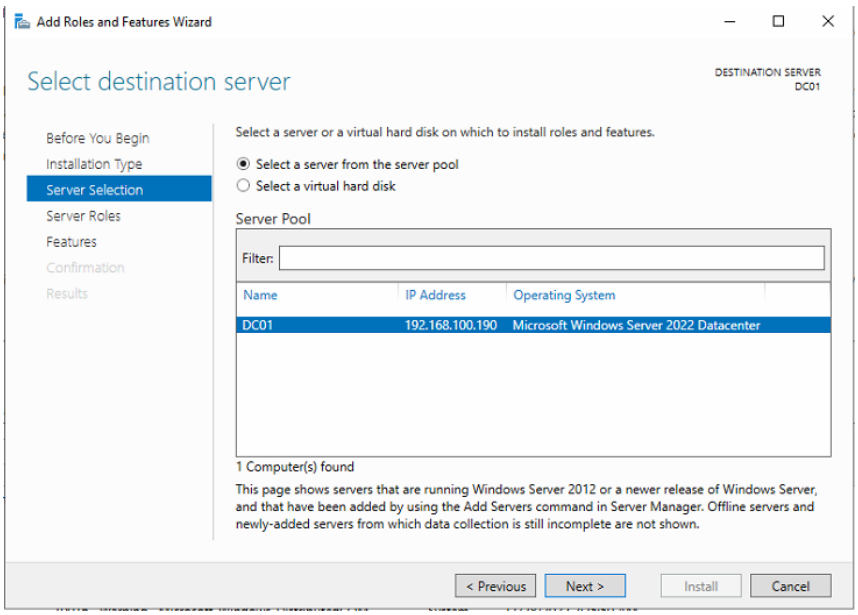

Figure 39: Select destination server or virtual hard disk

line only). As a result, we reduce the potential attack surface on our environment. It is usually done when the security team conducts server hardening steps.

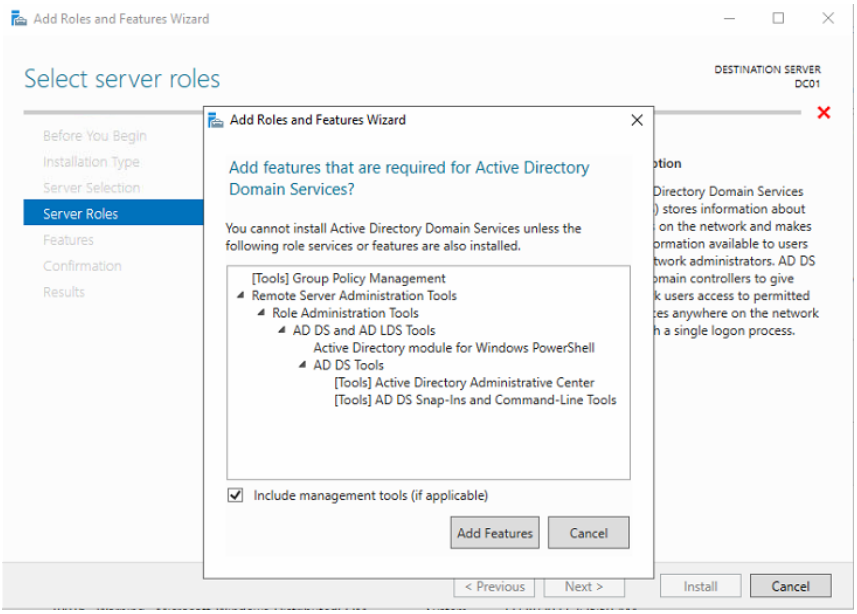

Figure 40: Active Directory Domain Services Dependencies

Group Policy Management Console (GPMC) will allow us to utilize scriptable interfaces to manage group policy. It is a convenient way to configure fine-grained access control in Active Directory environments. We will also incorporate this feature in our deployment (Dansimp, n.d.).

- Perform GPO operations such as backup, restoration, copy, and paste.
- Seek out the GPOs that have already been established.
- Generate printable and saveable HTML reports containing information such as the Resultant Set of Policies (RSoP).
- Collect RSoP information to analyze GPO interactions and fix group policy problems.
- We can import and copy GPOs between domains and forests if we create migration tables. Users, groups, machines, and Universal Naming Convention (UNC) paths referenced in the source GPO can be mapped to their new values in the destination GPO using files called migration tables.
- Make scriptable user interfaces for each GPMC command. We cannot utilize scripts to modify specific policy settings in a GPO.

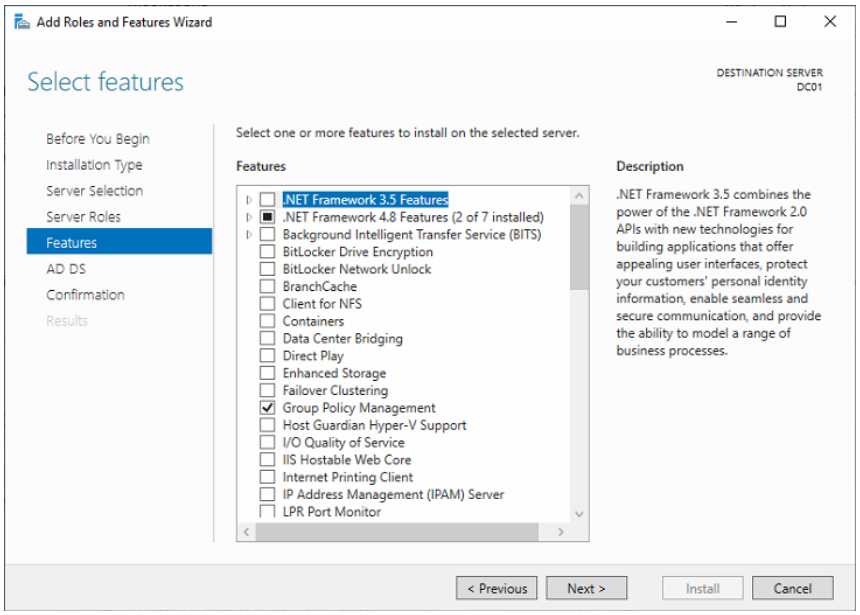

Figure 41: Select Group Policy Management Feature

We will deploy an Azure AD tenant infrastructure for our SSN Network deployment. Domain joins, group policies, LDAP, and Kerberos/NTLM authentication are some Azure Active Directory Domain Services (AD DS) domain services. We utilize these domain services in the cloud without worrying about setting up, maintaining, and updating domain controllers. Running legacy applications in the cloud that do not support contemporary authentication

techniques or do not want directory lookups to travel back to an on-premises AD DS environment constantly is possible with an Azure AD DS managed domain. Instead of managing the Active Directory Domain Services infrastructure in the cloud, we may specify a list and transfer those on-premises applications into a managed domain. Users could use their existing credentials to access the managed domain's services and applications. Use pre-existing user accounts and group memberships to restrict access to sensitive data. This is shown in Figure 42 for integrating an Azure Active Directory tenant with an on-premises deployment (Justinha, n.d.).

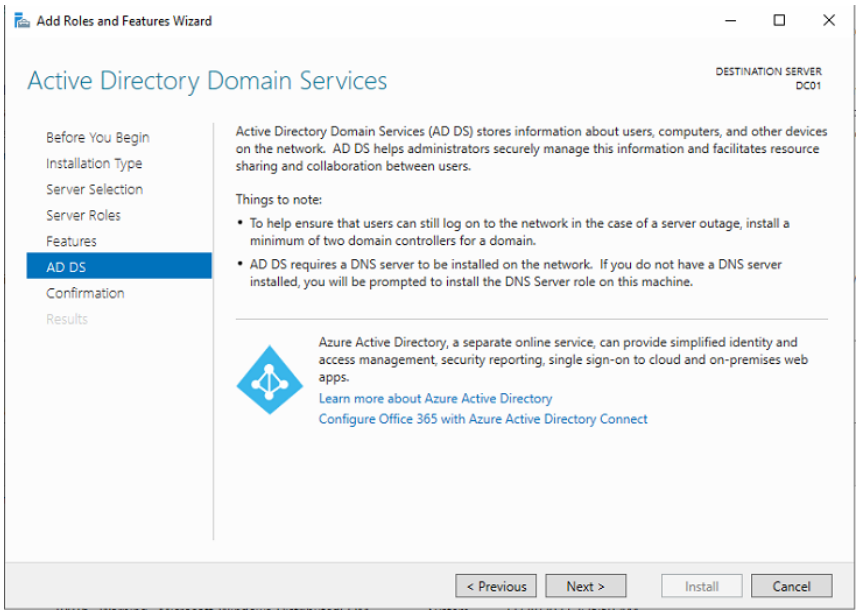

Figure 42: Azure Active Directory Connect

An overview of the selected roles and features is illustrated in Figure 43. As previously stated, the desired state configuration settings can be extracted to a PowerShell script and integrated with Ansible Automation Hub.

As shown in Figure 44, we should get a post-deployment configuration notice in the server manager telling us we need to set up the domain controller DC01 for AD DS. To configure this task, a Domain Admin or Enterprise Admin access level is required.

The initial step we will receive for configuring the domain controller for AD DS is to set up a new forest. We already have a forest and are configuring a second domain controller to make it redundant and more available. In that case, we could choose to "Add a domain controller to an existing domain" or "Add a new domain to an existing forest" in case of deploying a subdomain or secondary organizational domain. The root domain name for our

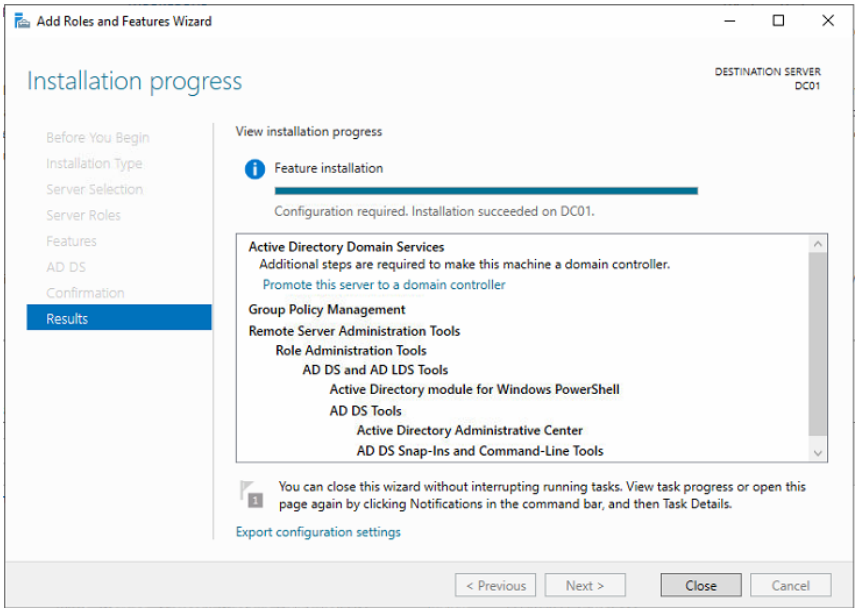

Figure 43: Installation progress

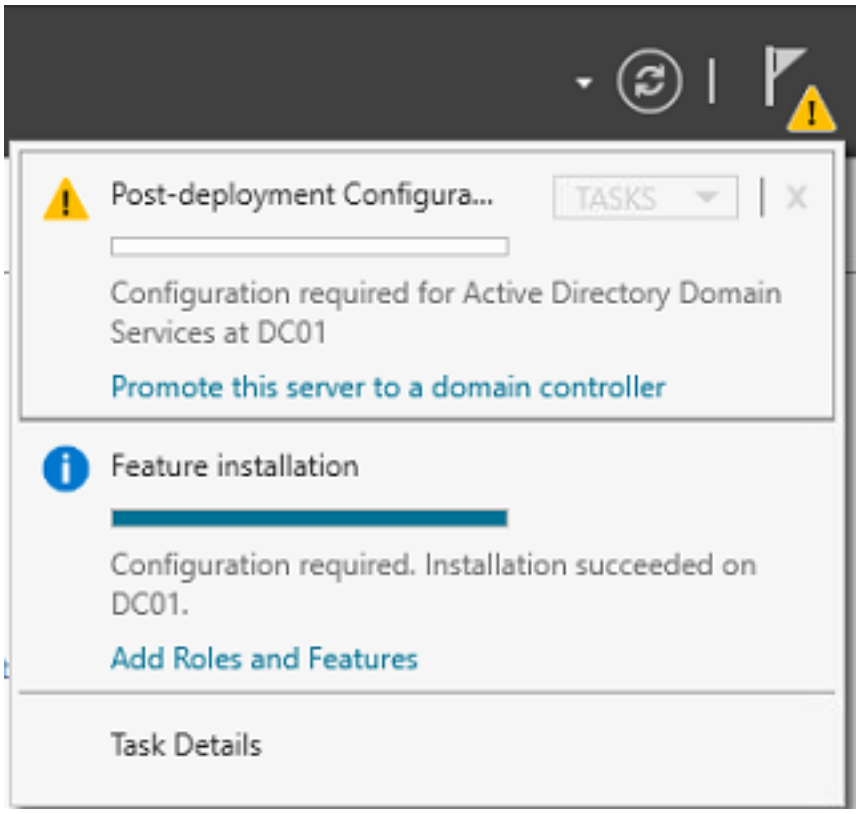

Figure 44: Post-deployment configuration for Domain Controller

deployment will be **innopolis.sne** as shown in Figure 45.

The features of a given Active Directory domain service for **innopolis.sne** domain and forest are determined and depend on their functional level. It also limits the versions of

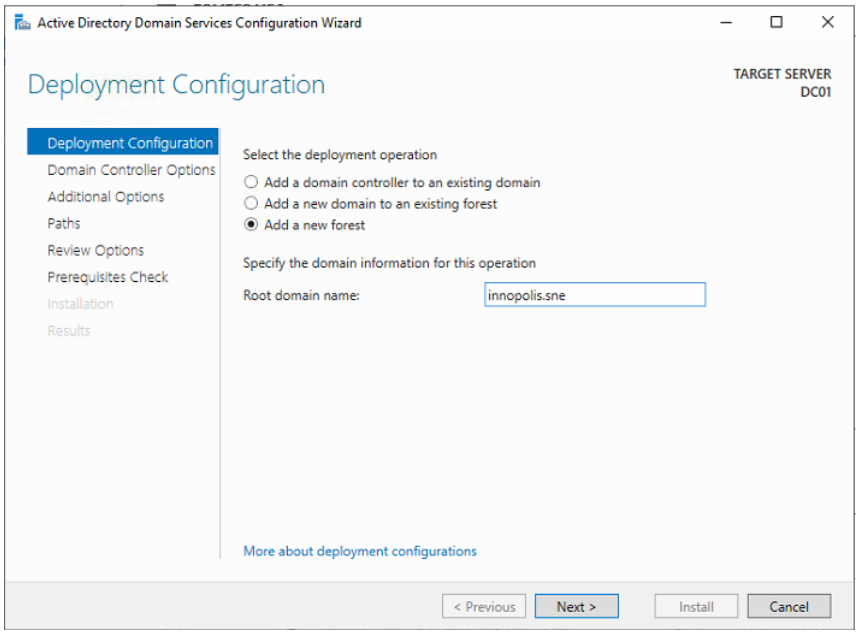

Figure 45: Deployment configuration - Add a new forest

Windows Server that the domain controllers in the forest may use. The functional level of a domain or forest determines the maximum capabilities of our infrastructure. Figure 46 depicts the configuration steps for both levels. As we are using Windows Server 2022, the best practice is to set it to Windows Server 2016. This means the minimum version we could use throughout the domain and forest is Windows Server 2016. Then, we set up the Directory Services Restore Mode password, a safe option for domain controllers that allows system administrators to repair or recover an Active Directory **NTDS.dit** (Dknappettmsft, n.d.).

- The **domain functional level** is set to Windows Server 2016
- The **forest functional level** is set to Windows Server 2016
- The configurable domain controller for innopolis.sne will include **DNS server** and **Global Catalog**; we cannot configure a read-only domain controller on the first domain we are creating.

DNS Options in Figure-47 are for creating delegation records in the parent Domain Name System (DNS) zone during AD DS installation. We could create the delegation if we have DNS installed on the current server. However, we will deploy it later and configure it manually.

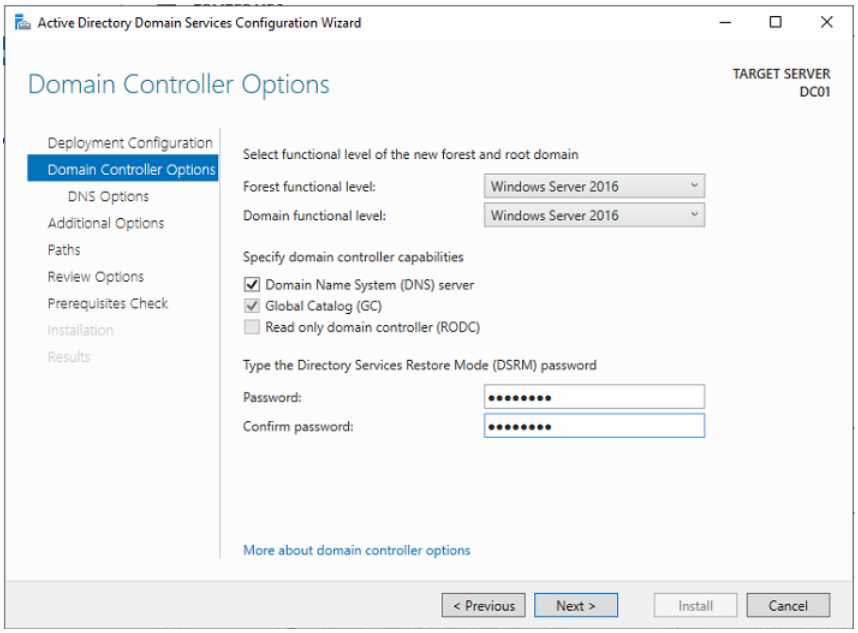

Figure 46: Domain Controller Options - forest and domain functional levels

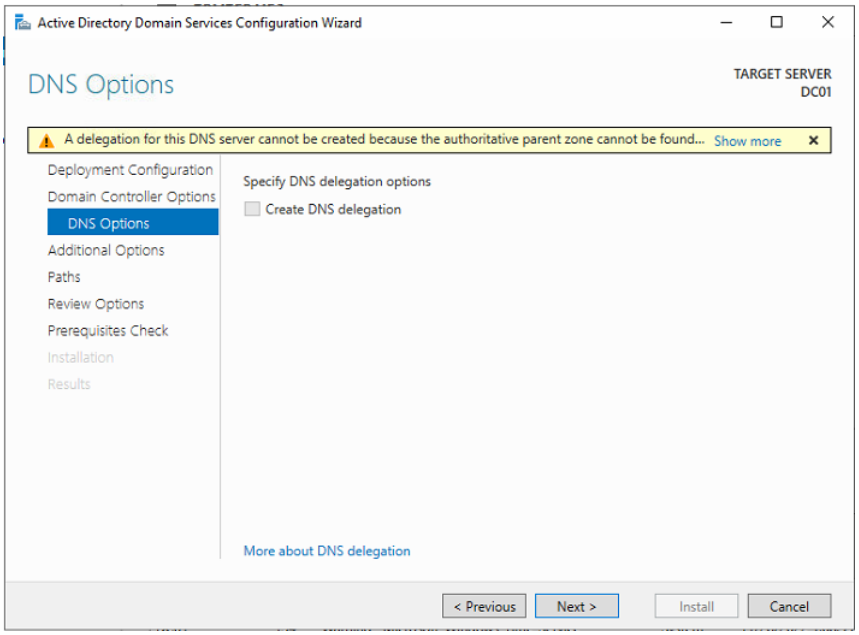

Figure 47: DNS Options - DNS delegation

Figure 48 allows for configuring the NetBIOS domain name for our domain. Most of the time, NetBIOS will be a subdomain of the DNS domain name. For our instance, it will be **INNOPOLIS**.

The Paths options page, Figure-49, enables us to override the folder location of the AD DS database; the default folder location will be under **C:\Windows\NTDS**. The other unique folder path to configure is for SYSVOL. The default location is under **C:\Windows\SYSVOL**.

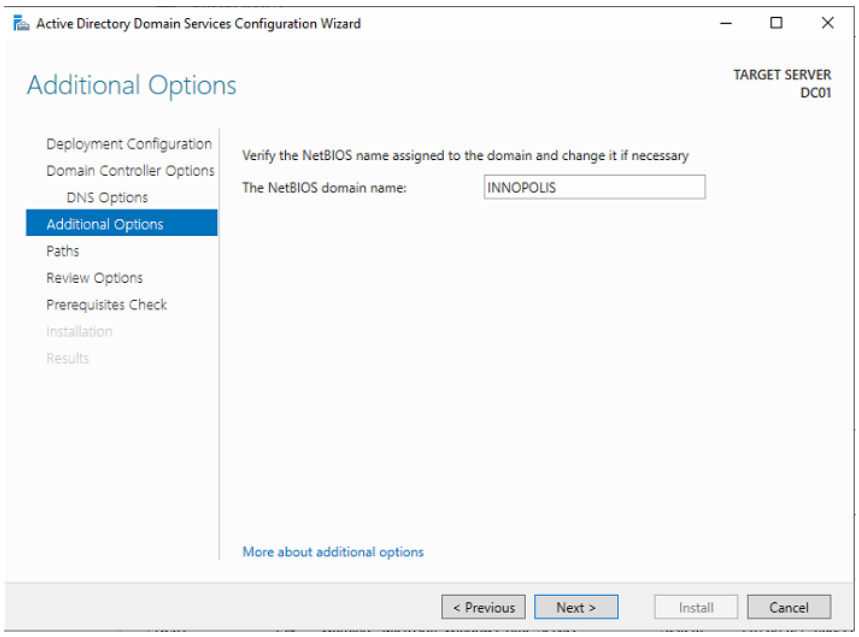

Figure 48: Additional Options - NetBIOS domain name

It is made up of a folder to store the following:

- **Group Policy templates (GPTs)**: which are replicated via SYSVOL. The container for it is also replicated through AD replication.
- **Scripts**: startup scripts for Group Policy objects.
- **Junction points**: These are like shortcuts which point to different directory structures in the system.

Figure 50 shows the prerequisites check, which allows us to validate the requirements before installing Active Directory Domain Services on our server. It is crucial for the user configuring this step to have sufficient privileges and be part of **Domain Admins** group.

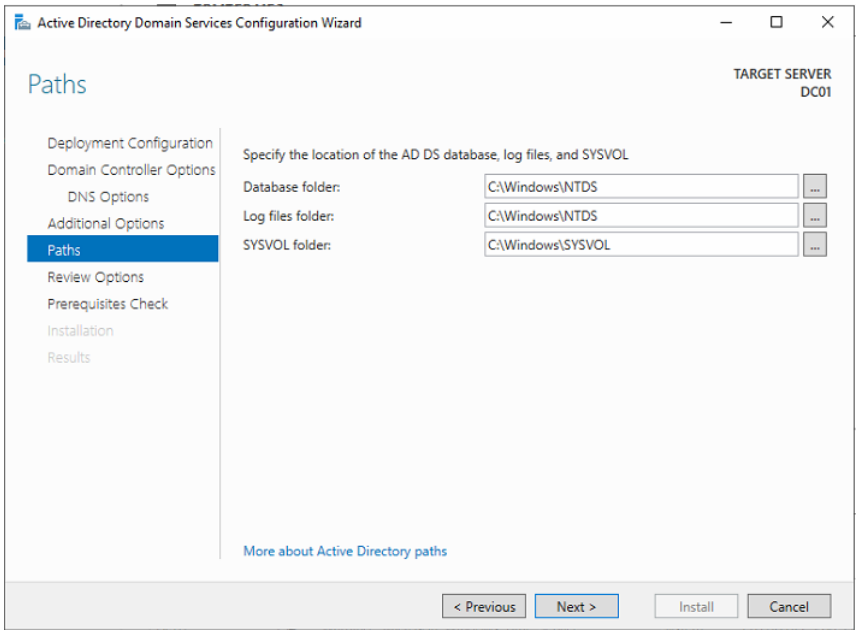

Figure 49: AD DS database, log files, and SYSVOL

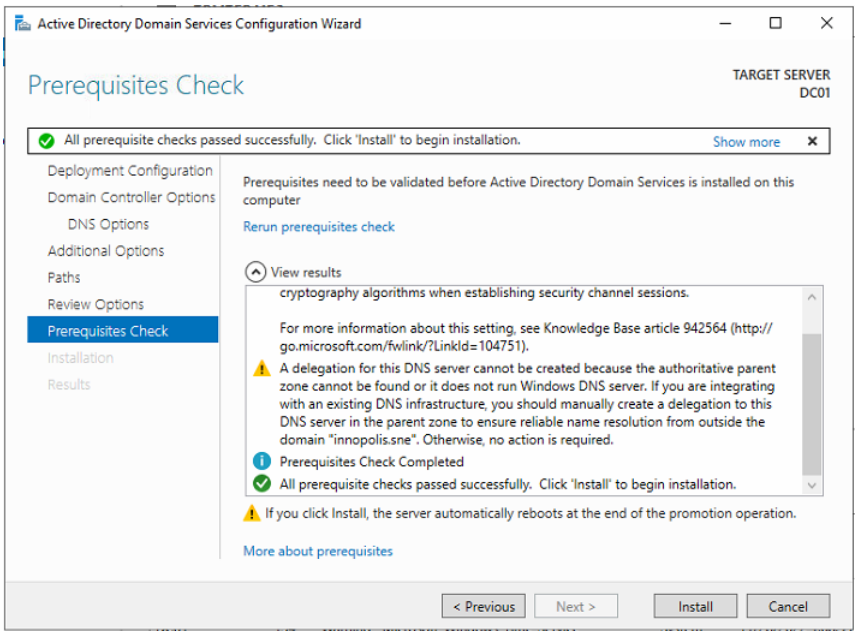

Figure 50: Prerequisites check

2.4 Remote Desktop Services

Remote Desktop Services is the ideal virtualization solution for business end users and system administrators. It is the best place to start when making solutions for any virtualization need. It supports the following virtualized environment configurations, but we will only use it for remote management via RDP:

- **Session-based virtualization**: We can utilize Windows Server processing power to setup a low-cost, multi-user environment for powering user's routine tasks.
- **VDI**: Utilize Windows client to deliver excellent performance, and software compatibility for the users.

In Figure 51, we will select the Remote Desktop Services installation to deploy the required role service for the Virtual Desktop Infrastructure (VDI) for creating session-based desktop deployments through RDP (ChristianMontoya, n.d.).

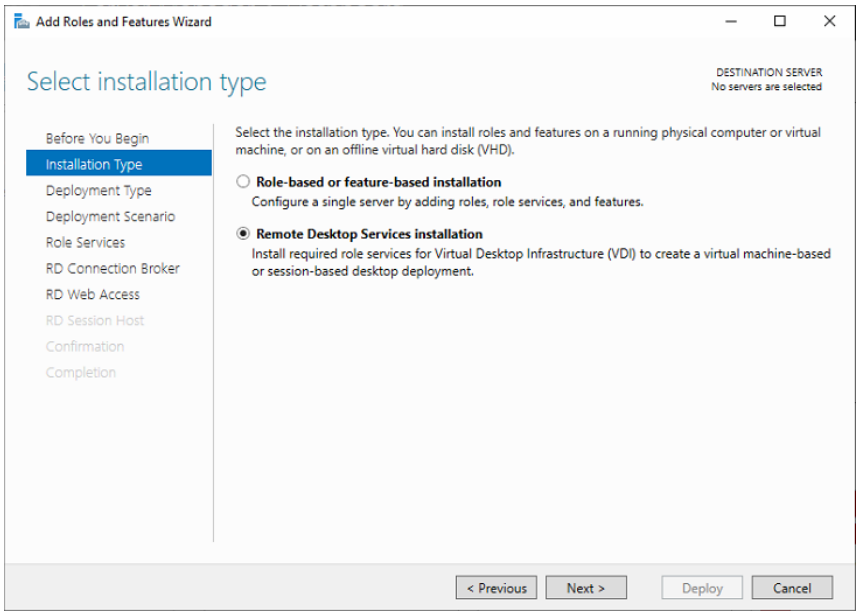

Figure 51: Remote Desktop Services installation type

If we choose the standard deployment option, we can deploy on our server without making a collection and publishing RemoteApp programs (see Figure 52).

Following that, we will choose the session-based desktop deployment to enable users to connect to session collections that include published session-based desktops for our server (see Figure 53).

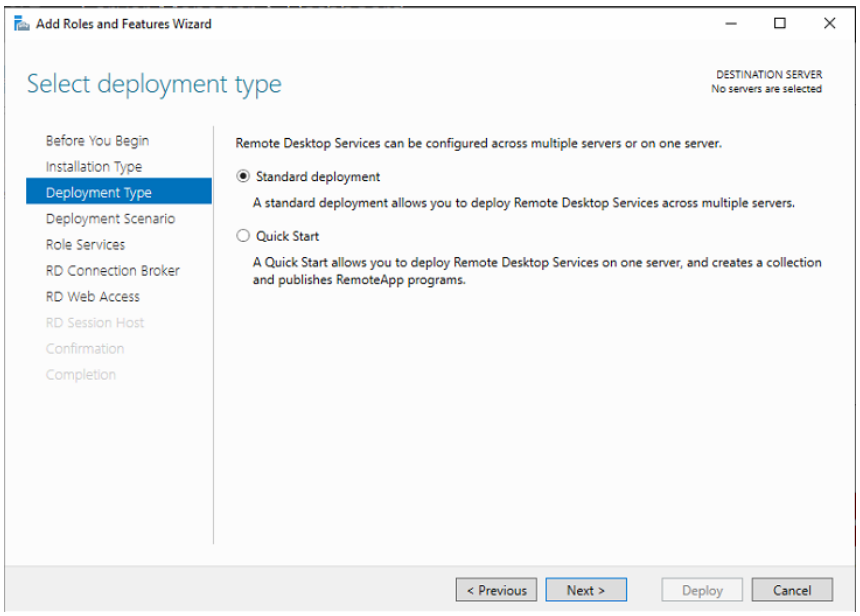

Figure 52: Remote Desktop Services - Standard deployment type

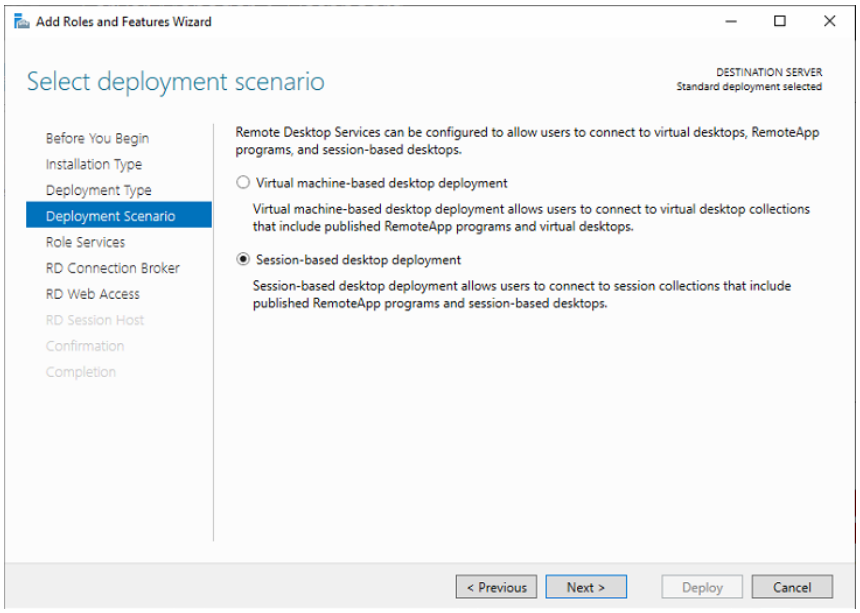

Figure 53: Remote Desktop Services - Standard deployment scenario

Our Windows Server 2022 will be installed and configured with Remote Desktop Services. The roles are **Remote Desktop Connection Broker**, **Remote Desktop Web Access**, **Remote Desktop Session Host** as shown in Figure-54-57.

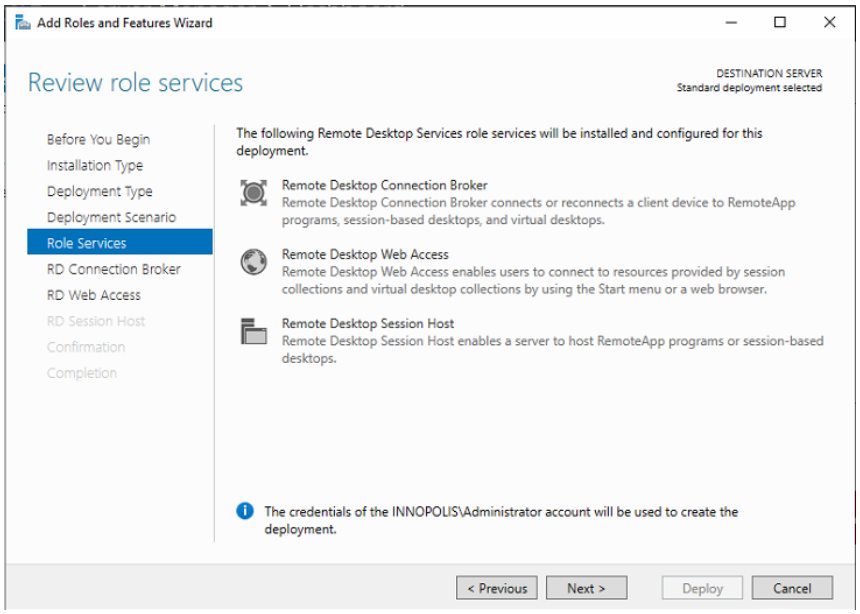

Figure 54: Remote Desktop Services - Role Services review

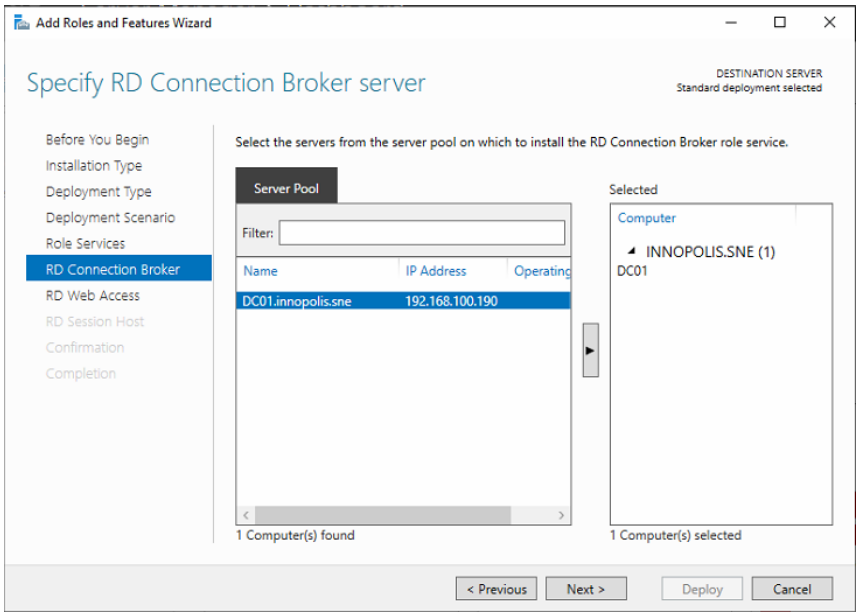

Figure 55: RD Connection Broker server

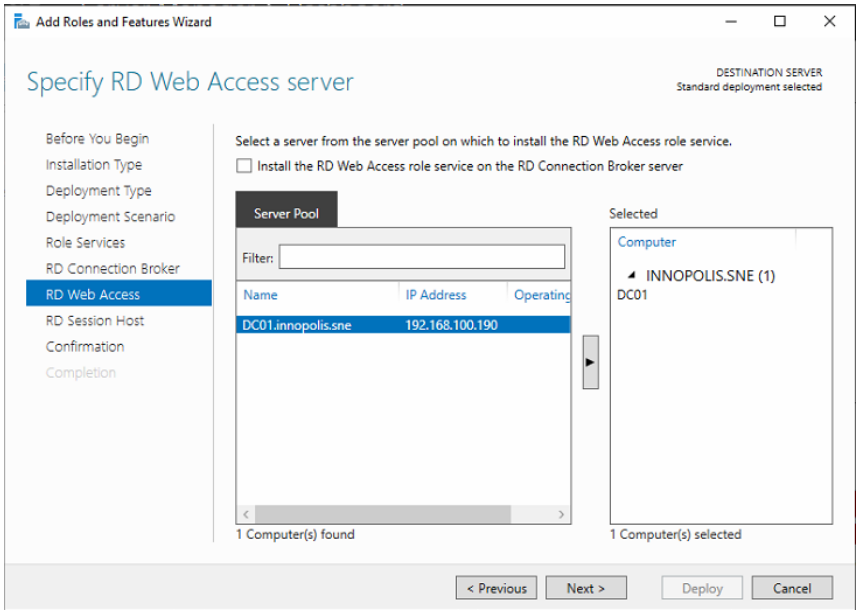

Figure 56: RD Web Access

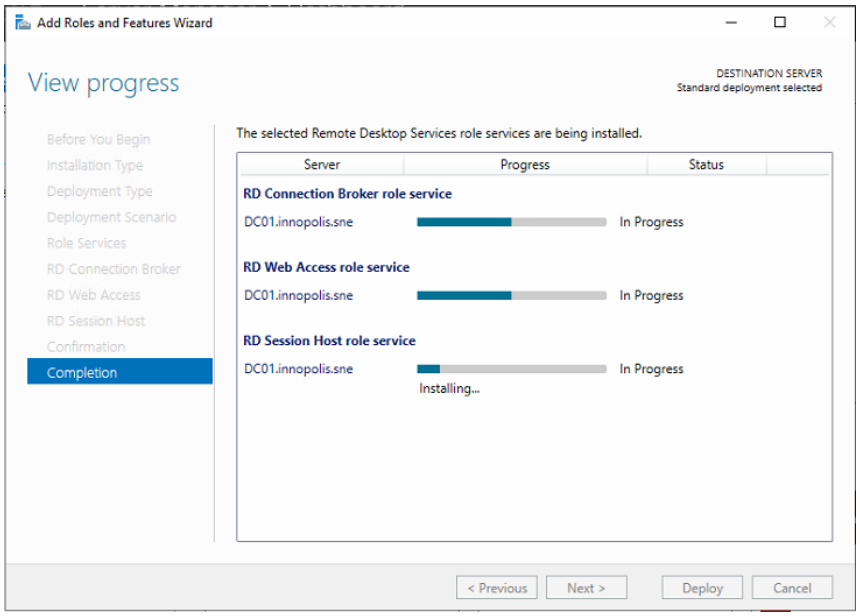

Figure 57: RDS Deployment Progress Track

2.5 Lightweight Directory Access Protocol - LDAP (389)

The Lightweight Directory Access Protocol (LDAP) is a way that an application protocol for talking to directory services is put into action. Through this service, the data can be sent to other nodes in the network. These nodes commonly utilize this protocol for authenticating users against the Active Directory database. As well used for access control and information discovery in enterprise applications (Janicericketts, n.d.). The components of the system are as follows:

- **User**: Browser-based access to LDAP-reliant software.
- **Web Browser:** The portal through which a user can reach the application external URL.
- **Virtual Network:** For the legacy application to use LDAP services, we set up a private network in Azure.
- **Legacy applications:** Any Azure VM or server workload that depends on LDAP and hass access to Active Directory Domain Services instance IPs through networking routes.
- **Azure AD:** Integerates with Azure Active Directory to synchronize user profiles with on-premises directories.
- **Azure AD Domain Services (AD DS):** Uses a one-way sync with Azure Active Directory to make it easy to access your organization's users accounts, groups, and credentials. The Active Directory Domain Services instance has a virtual network associated with it.
- **Azure AD Connect:** A device to link identities managed locally with Microsoft Azure Active Directory. You can set up the necessary components for the connection, such as the sync and sign in from Active Directory to Azure AD.
- **Active Directory:** On-premises directory service storing credentials such as user names, email addresses, and passwords.

(Janicericketts, n.d.)

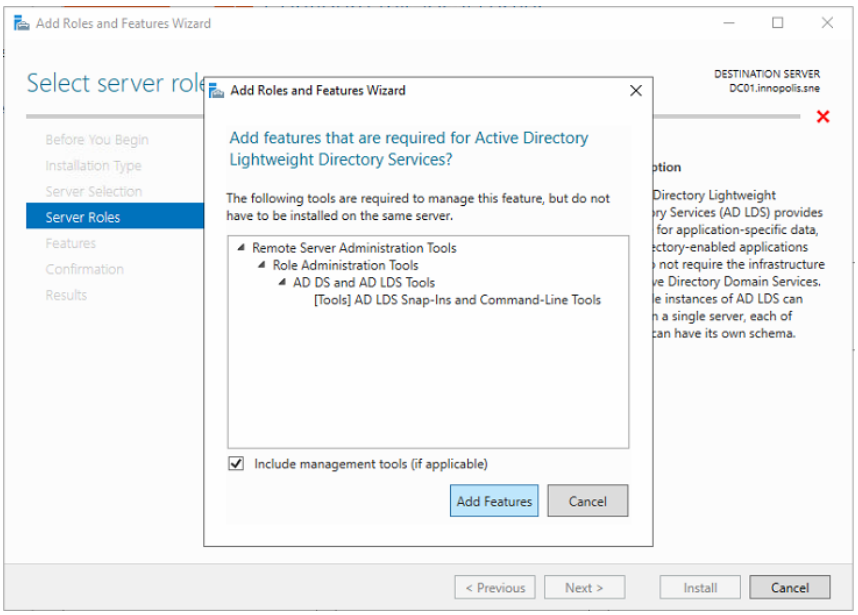

Figure 58: Lightweight Directory Services Roles and Features

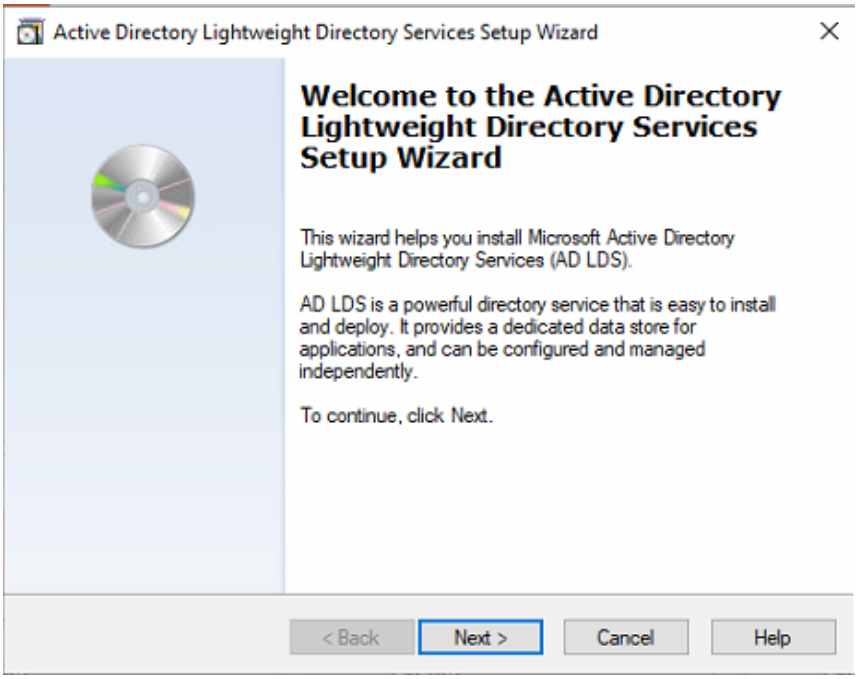

Figure 59: AD LDS Setup and Configuration Wizard

As shown in Figure 60, we will set up a unique LDAP instance on our Windows Server to meet our deployment requirements and allow OpenVPN users to sign in with their Active Directory accounts. It will create an instance with the default configurations and schema partitions that can replicate other existing instances.

In Figure 61, we are configuring the instance name (service name) that will be running on the server. The chosen name for our configuration will be **ssnldap**.

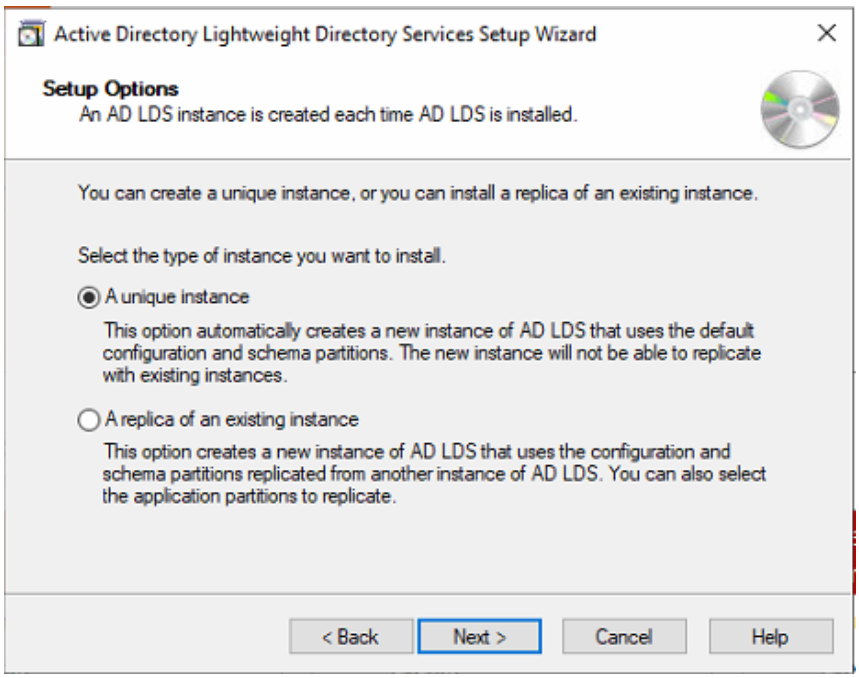

Figure 60: Creating AD LDS

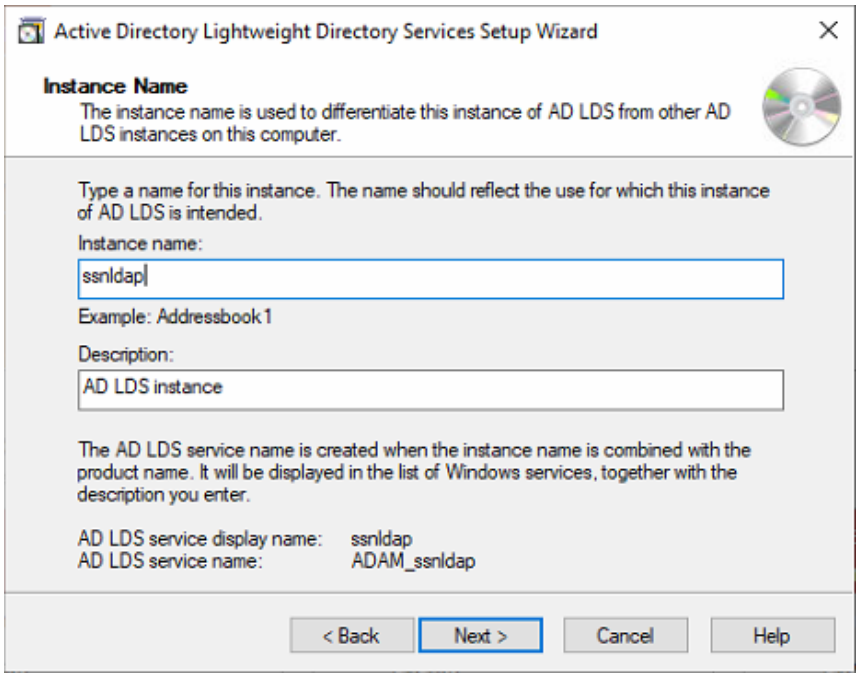

Figure 61: Configuring AD LDS Instance Name

In Figure-62, we are set to configure the plain LDAP port number **(5389)** later changed to **(389)**. Furthermore, LDAP over SSL\TLS using port **(1300)** later changed to **(636)** as per the default port numbers.

Figure 63 shows how we set up an application directory partition for our AD LDS (ssnldap) instance. This option is for apps like OpenVPN that do not create an app directory when

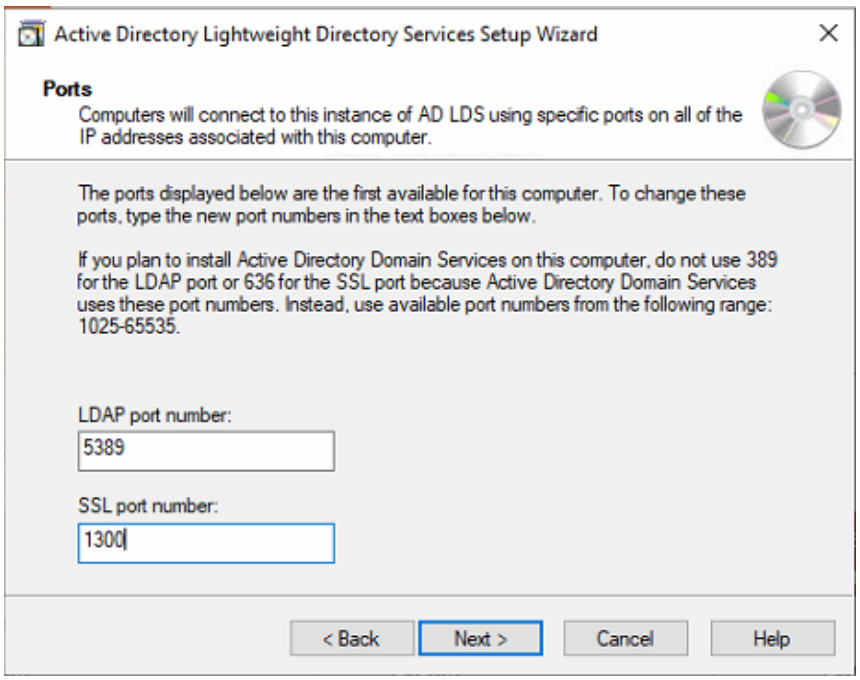

Figure 62: AD LDS Instance Port number - LDAP and LDAPS

installed. We are not putting the Access Server on the Windows Server. Instead, we are using the OpenVPN cloud and its network connector. The partition name we should be safe with for later is **OU=ssnldap, DC=innopolis, DC=sne**.

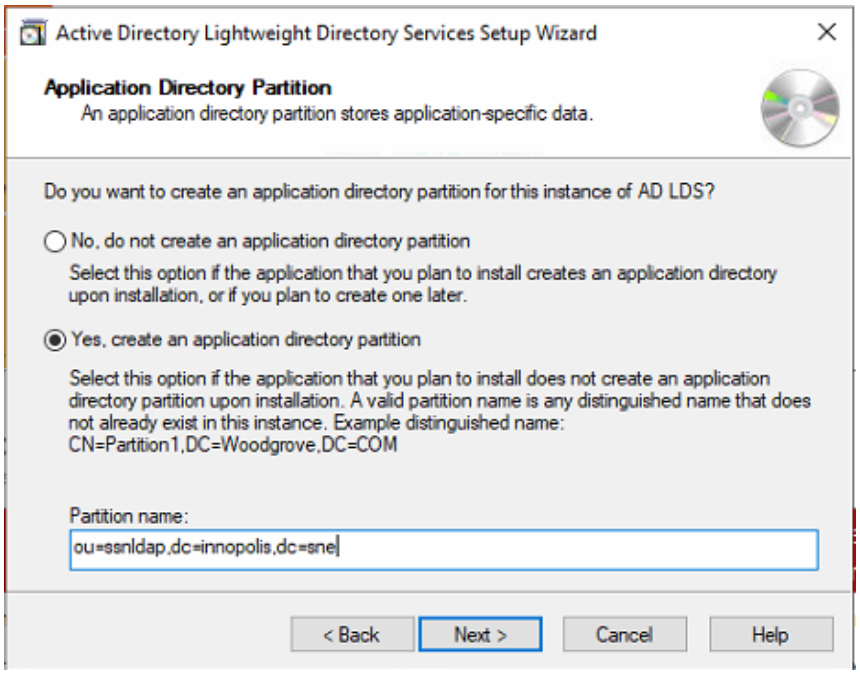

Figure 63: Active Directory Partition

Figure 64 shows how we leave the default folder paths and locations for data and recovery

files.

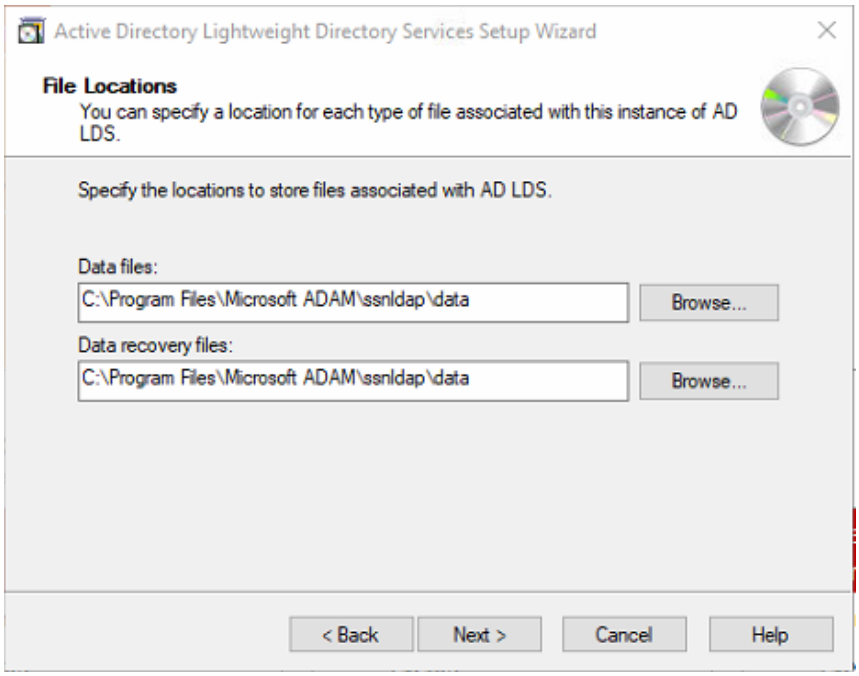

Figure 64: File Locations - Data files and Data recovery files

A good security measure is to add a new AD object (a user account) to run the LDAP service instance we are setting up. In Figure 65, we create a new user account named **LDAP Administrator** to use exclusively for the LDAP(s) service instance.

The assigned permissions and group membership for our created user account are shown in Figure 66. The best configuration would be to create a network service account and restrict that account's ability to log in interactively. However, as we are using a testing environment, not a production one, we will configure the ones in the figure for quick implementation.

We give the AD LDS service permissions for the selected account we created above (LDAP Administrator), as shown in Figure 67.

As for AD LDS Administrators' privileges for managing the LDAP instance, we will configure it as the currently logged-on user: **INNOPOLIS\Administrator** which is the user installing the current instance.

Directory information is sent in an LDIF file as a series of records, one record per object. In LDIF files, directories made with the Lightweight Directory Access Protocol (LDAP) and requests to change them are shown in a standard format for exchanging plain text data. In Figure 69, we import metadata from multiple .LDF files as shown (*What is AWS Directory Service? - AWS directory service*, n.d.).

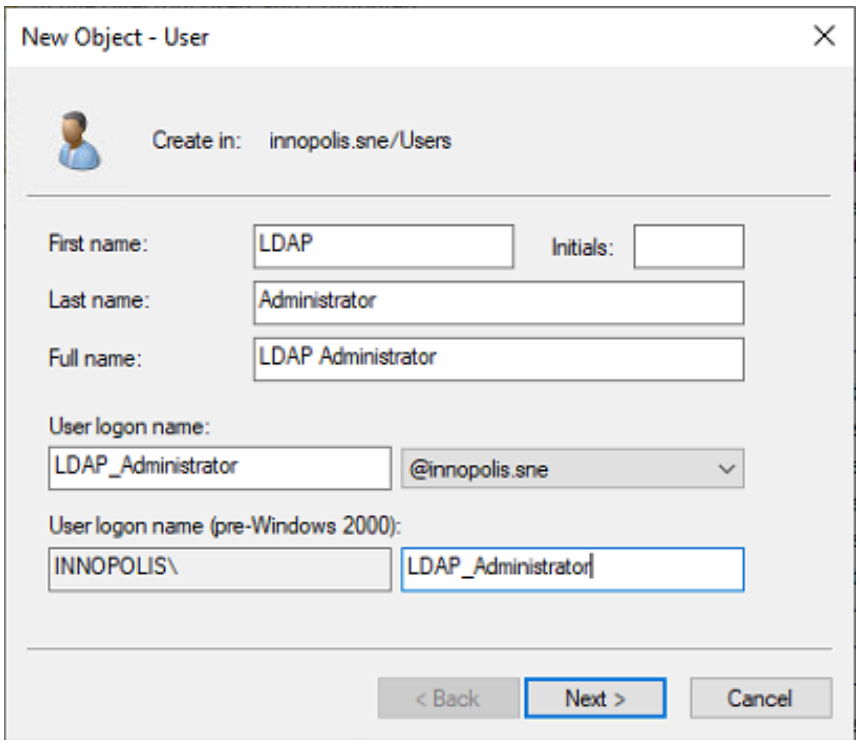

Figure 65: Creating AD Object User account - LDAP Administrator

Configuring user authentication via LDAP on the OpenVPN Cloud is relatively simple. In Figure-71, we set the **Bind DN** value to **CN=LDAP Administrator, CN=Users, DC=innopolis, DC=sne** and supply the password for the LDAP Administrator. **Base DN** is used as the starting point for querying the Active Directory, and we will set it to the value of **DC=innopolis, DC=sne**. The next value is configuring the username attribute when querying the Directory Services. In Active Directory, the attribute will have a value of **sAMAccountName**. If we work with a Linux-based Directory Service, the value will be **uid** in this example. In Figure 72, we configure the rest of the settings for the LDAP server. We can see we are establishing the connection over port **389**, and the SSL option is disabled. This is the initial configuration to test the LDAP deployment. We will eventually switch to the safer version of LDAPS, but first, we need to set up our public key infrastructure.

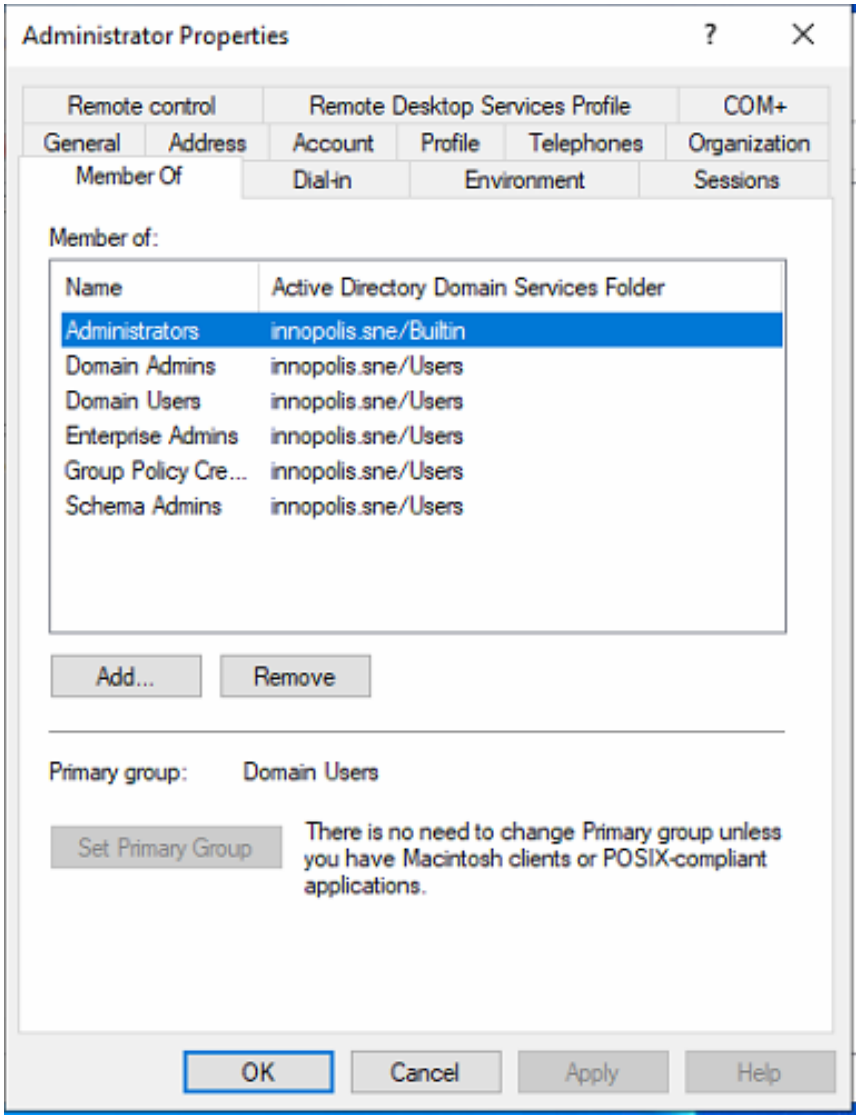

Figure 66: LDAP Administrator Group Membership

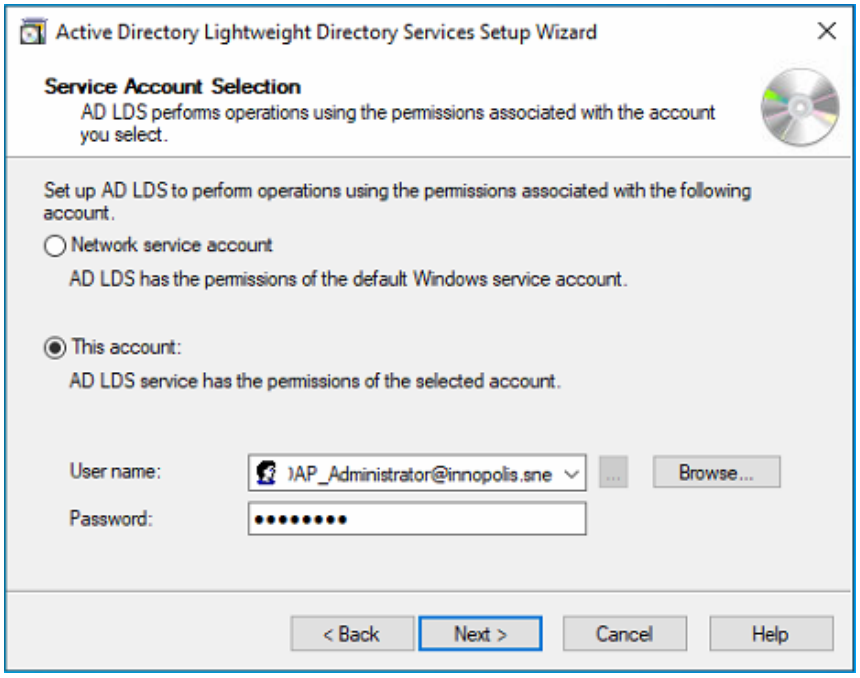

Figure 67: Service Account Selection - LDAP Administrator

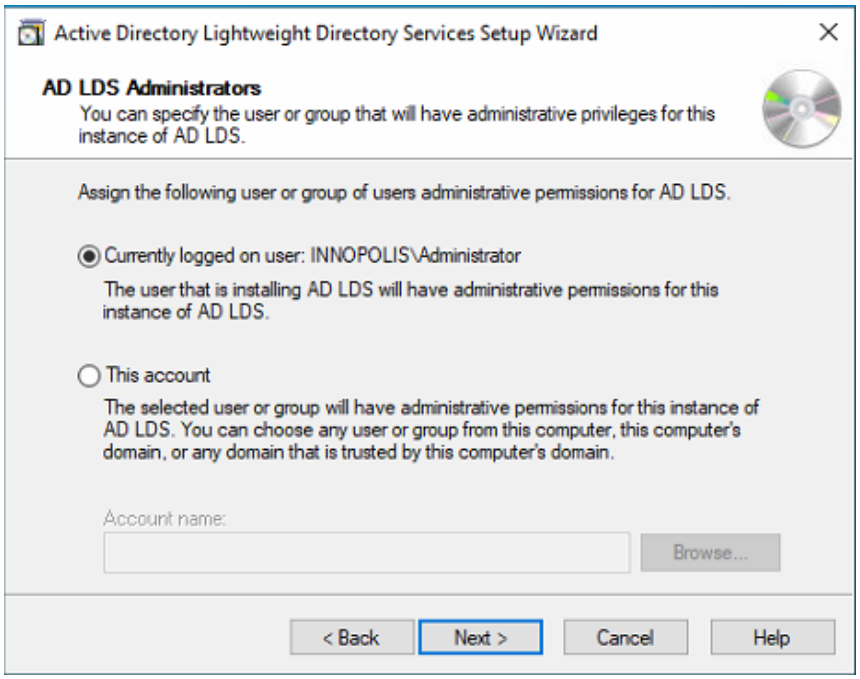

Figure 68: AD LDS Administrators Users and Group

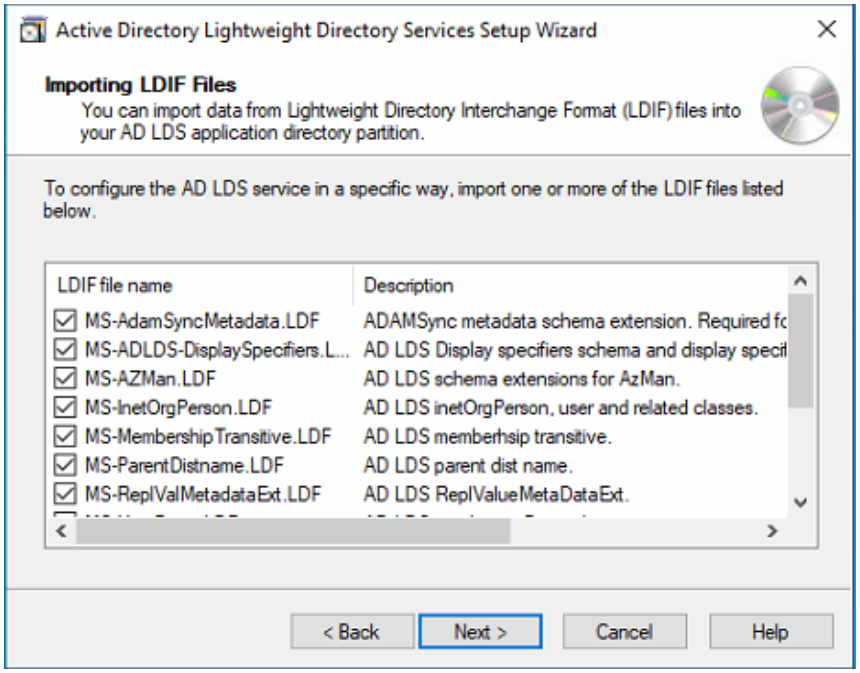

Figure 69: Importing LDIF Files

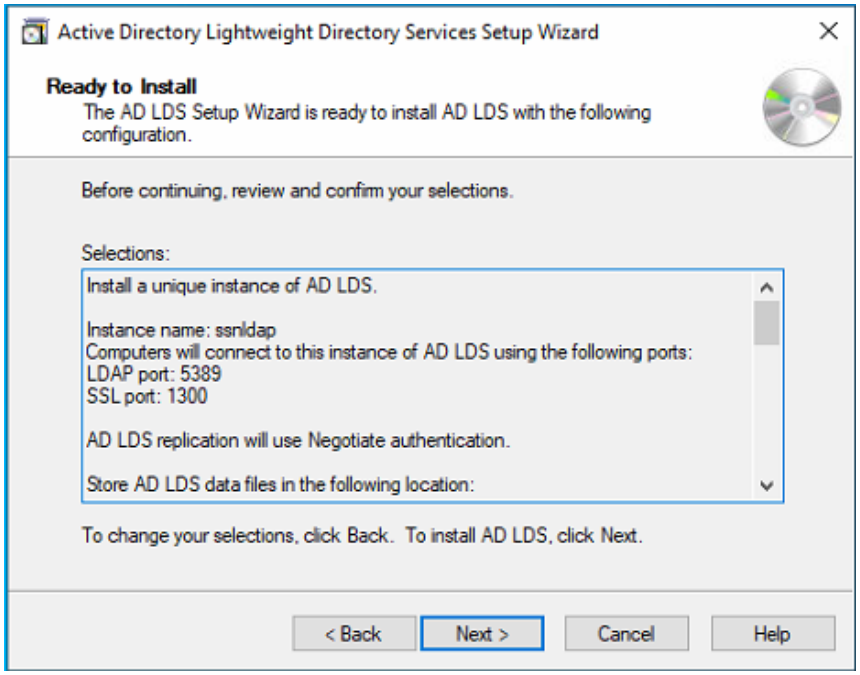

Figure 70: AD LDS Configurations Review - Ready to Install

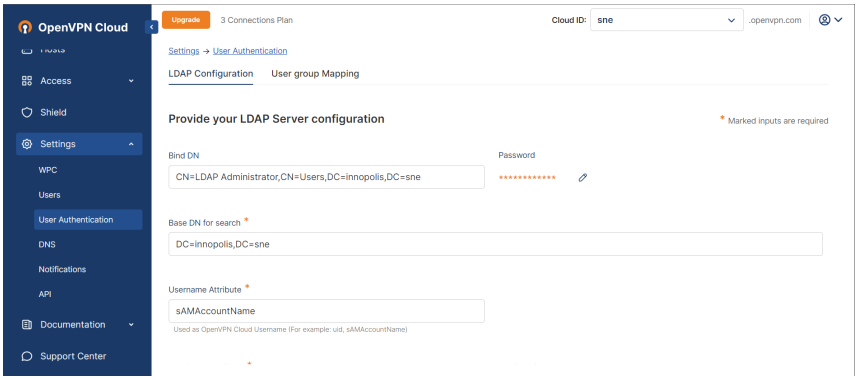

Figure 71: OpenVPN Cloud LDAP Options - Bind DN, Base DN, Username Attribute

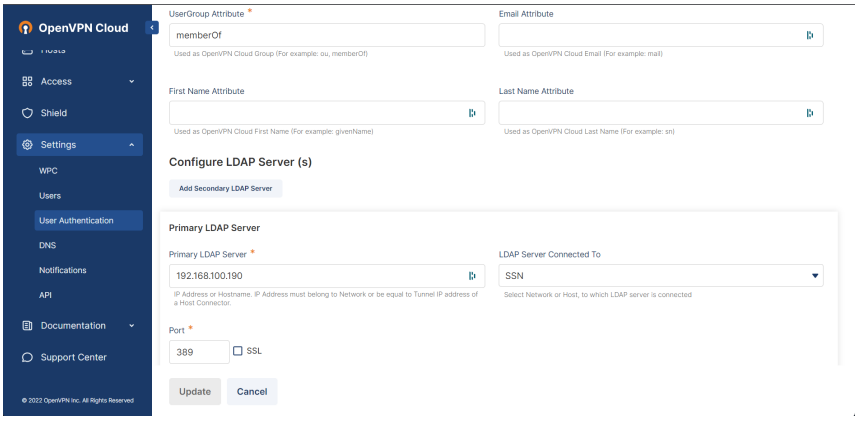

Figure 72: OpenVPN Cloud LDAP Server Settings

2.6 Lightweight Directory Access Protocol over TLS - LDAPS (636)

Active Directory Certificate Services - Public Key Infrastructure

In order to use the secured version of LDAP over TLS, we will need to deploy a public key infrastructure in our Active Directory environment. To deploy PKI, we need to configure Active Directory Certificate Services. AD CS provides software security solutions that use public keys in various ways to create and manage digital certificates. These digital certificates may be used to digitally sign and encrypt messages and files sent over the internet. Digital certificates made with AD CS could be used to check the accounts of computers, users, and devices. In addition, we could use the deployment for the following:

- **Encryption**: to ensure privacy in the first place.
- **Digital Signatures**: to guarantee honesty, which is a major advantage.
- **Network Authentication**: by using certificate-based account linking for computers, users, and other networked devices.

AD CS could also ensure that users, devices, and services are safe and secure by linking their unique ID to a secret key. We could control who has access to which certificates and safely revoke access from the certificate authority. Internet Protocol security (IPsec), Encrypting File System (EFS), smart card login, SSL/TLS, digital signatures, and Secure/Multipurpose Internet Mail Extensions (S/MIME) are only some of the applications supported by AD CS (*Active directory certificate services overview*, n.d.)

Selecting the required roles and services to deploy a public-key infrastructure involving the Certification Authority service is shown in Figure 74. It will be used to issue and manage certificates. The public key infrastructure environment can be formed by connecting multiple CAs.

Figure 75 shows that after installing AD CS successfully, we will need to use post-deployment configuration at DC01. We will use it to set up initial settings for the certification authority.

We will set up an enterprise CA to use Active Directory Domain Services to manage certificates so that we can say what kind of certification authority we are deploying. Standalone

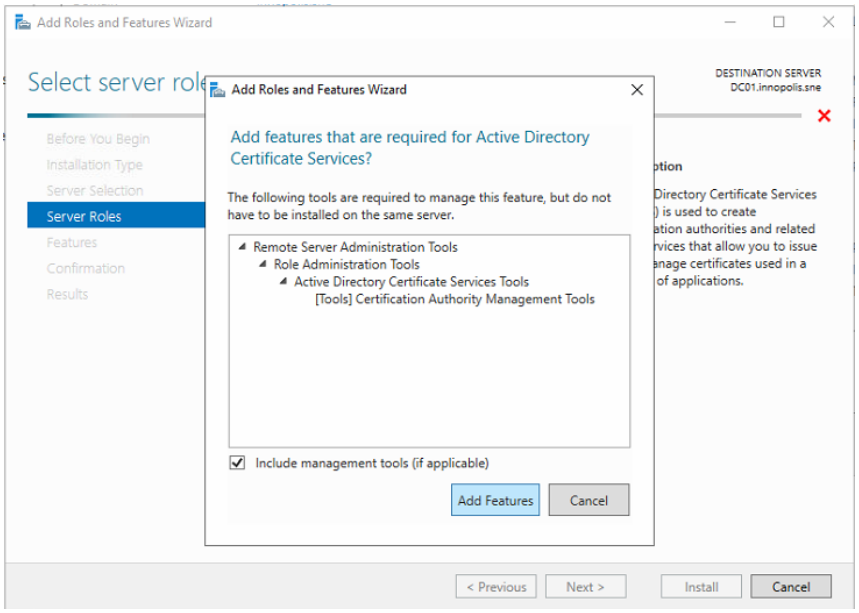

Figure 73: Active Directory Certificate Services Roles and Features

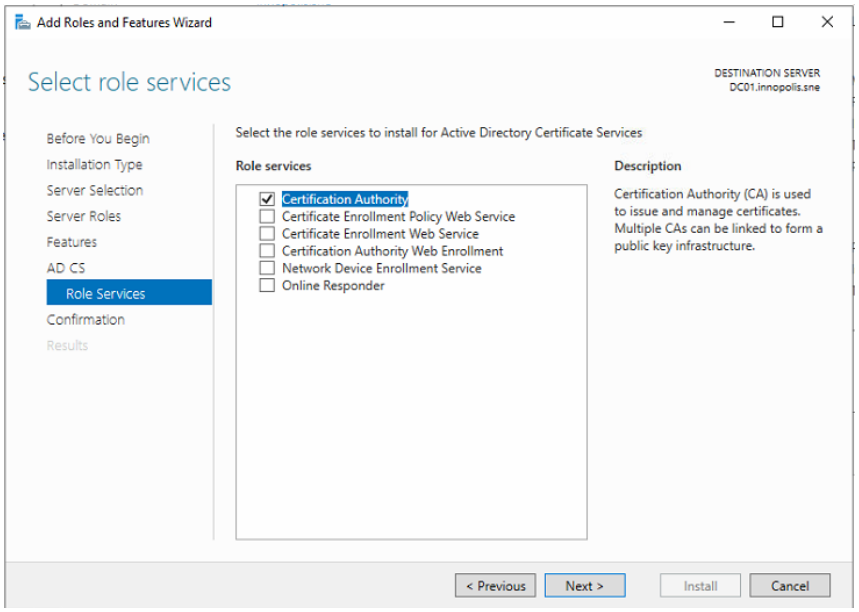

Figure 74: Certificate Authority Role Service - PKI

CAs do not use AD DS to issue or manage certificates. However, in large deployments, we deploy a root certification authority in the trust chain offline, generate the root certificate, sign the intermediary CAs, extract the private key to cold storage, and vault it. This is demonstrated in Figure 77.

On the other hand, a root CA will be the type of certification authority. This is often the first in a public key infrastructure hierarchy. A subordinate CA requires an established PKI hierarchy, as shown in Figure 78.

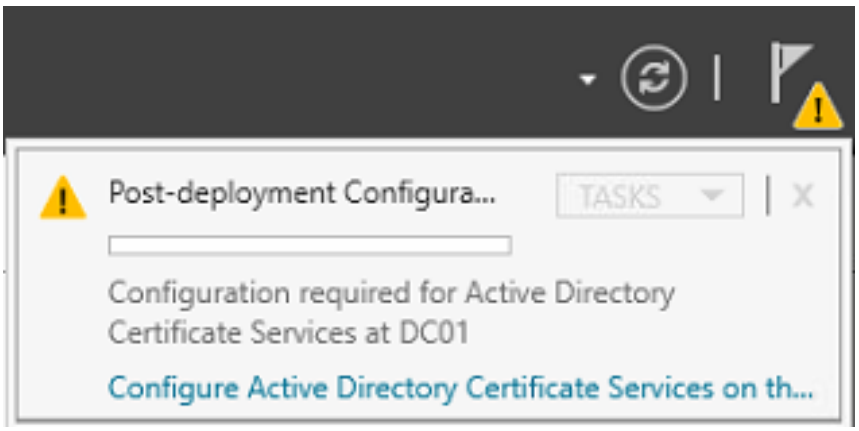

Figure 75: Post-deployment Configuration - AD CS

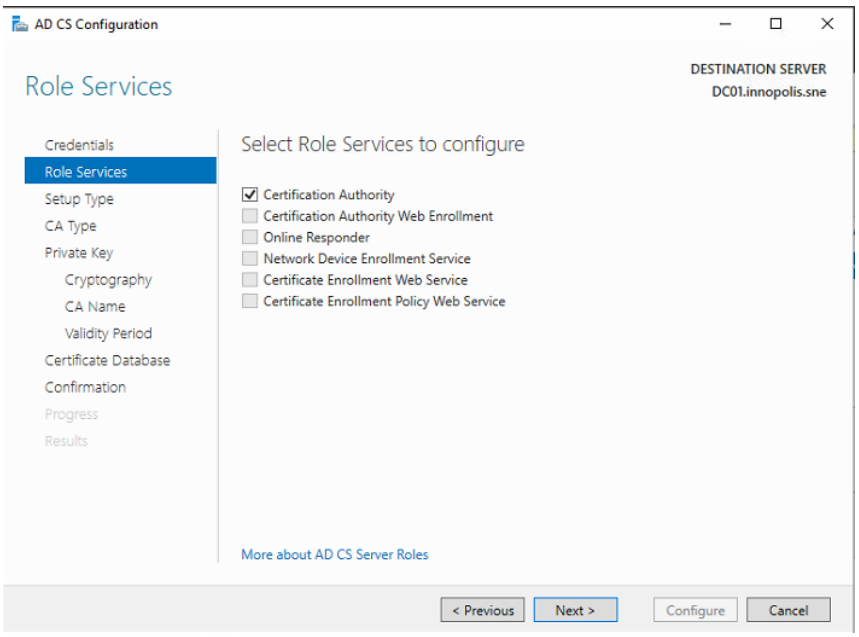

Figure 76: Role Services to Configure - Certification Authority

Then, we decide what private key to use to make certificates for users, computers, clients, etc., and give them to them. We will be creating a new private key on this server. In large deployments, it is best to generate a root CA's private key offline, then export it before connecting the machine to any network! The options for the configurations are illustrated in Figure 79.

For signing certificates, private key cryptography will use the cryptographic provider **RSA#Microsoft Software Key Storage Provider** and a hash algorithm of SHA256 with a key length of textbf2048 bits. Furthermore, when the CA accesses the private key, we will not allow the administrator to interact (see Figure 80).

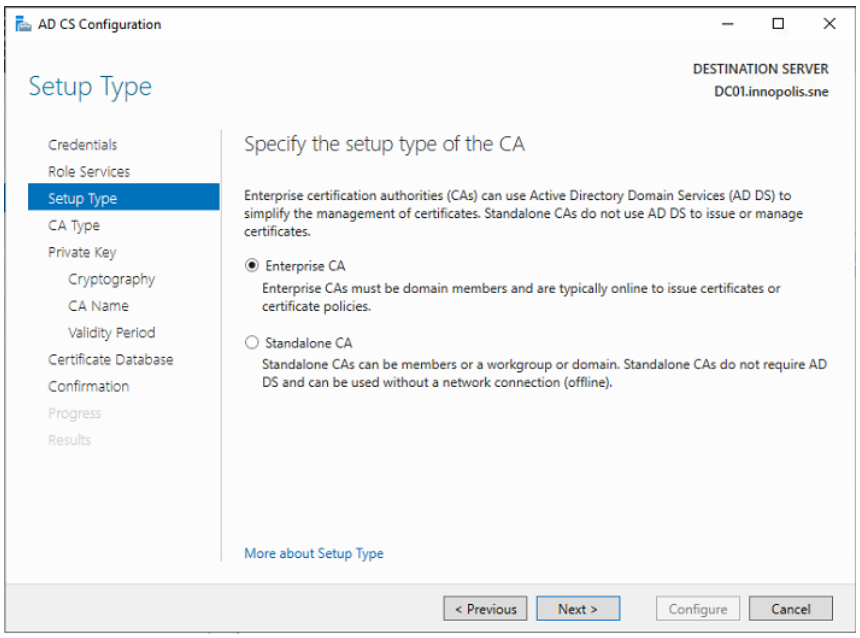

Figure 77: Setup type of Certification Authority - Enterprise CA

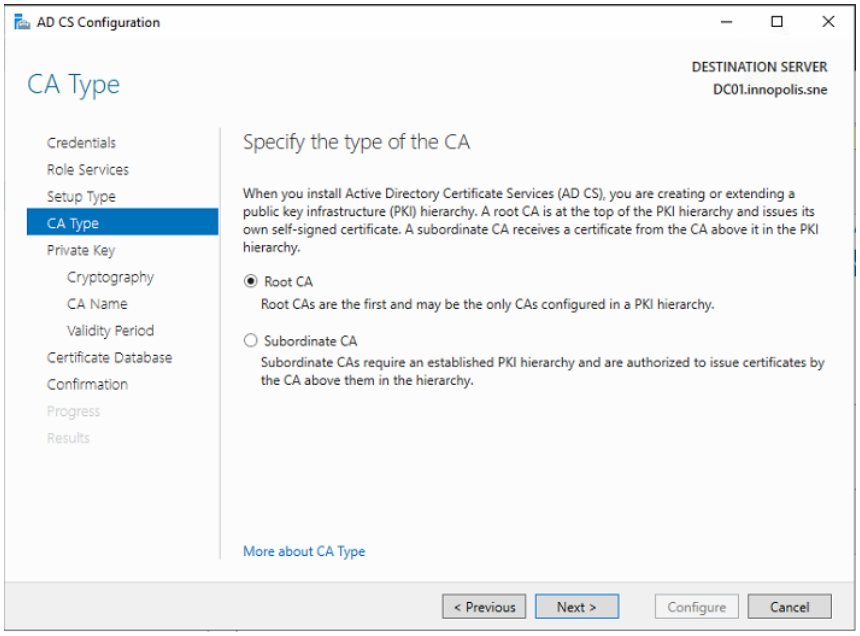

Figure 78: Specify the type of Certification Authority - Root CA

The configured common name will be **innopolis-DC01-CA** with a distinguished name suffix of **DC=innopolis, DC=sne** and a preview of distinguished name value **CN=innopolis-DC01-CA, DC=innopolis, DC=sne** as demonstrated in Figure-81. This name will be added to all the certificates issued by the certification authority. It will be easier to find if we tell the Active Directory database the common name for this certification authority.

The validity period for the root certificate generated by the certification authority will

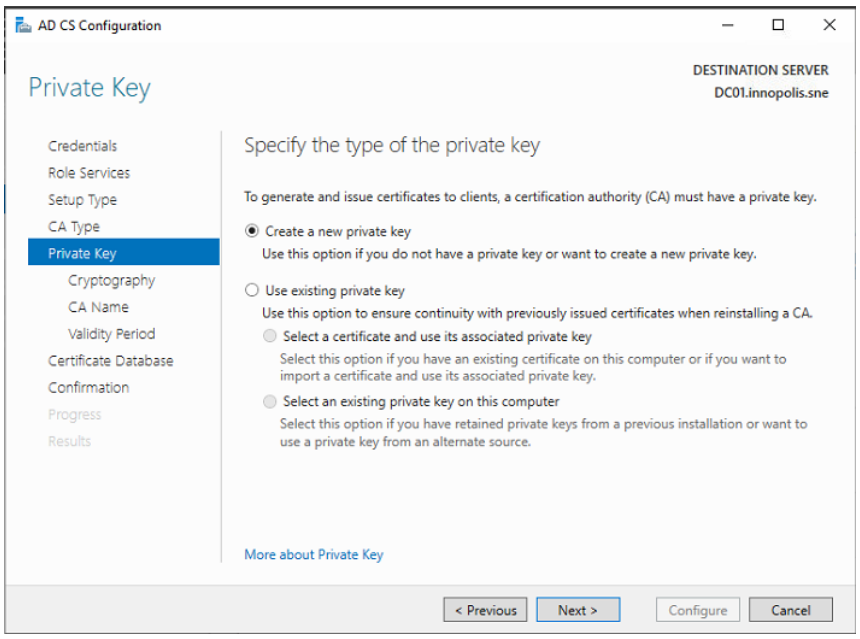

Figure 79: Create a new private key

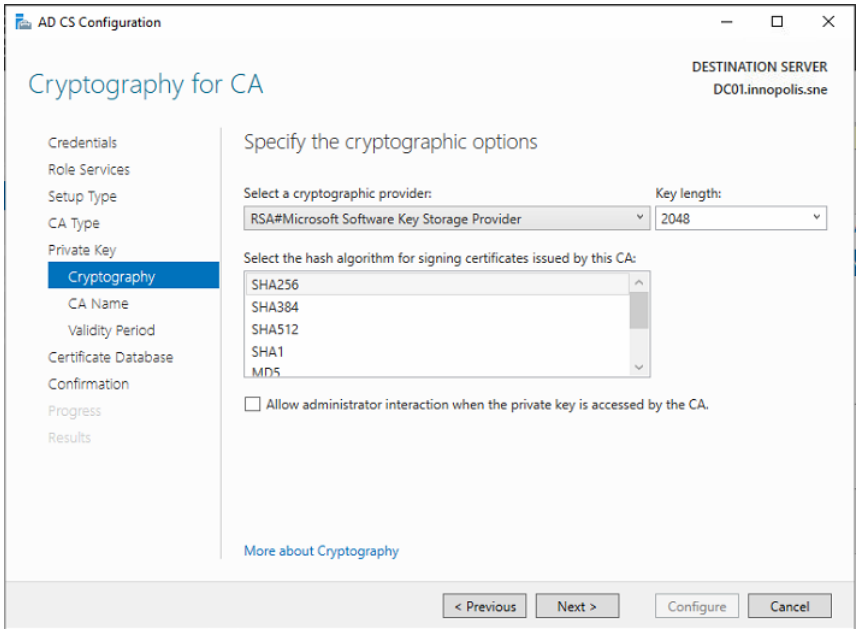

Figure 80: Private key cryptography - SHA256 2048-bit

have a default value of 5 years (see Figure 82). Although in complex PKI environments, this is set up with a value of +50 years.

We will leave the default path locations for the certificate database and log under **C:\Windows\system32\CertLog** as illustrated in Figure 83.

The final AD CS configuration overview and confirmation are shown in Figure 84. Figure 85 shows that the installation result of our configurations was successful. We can still

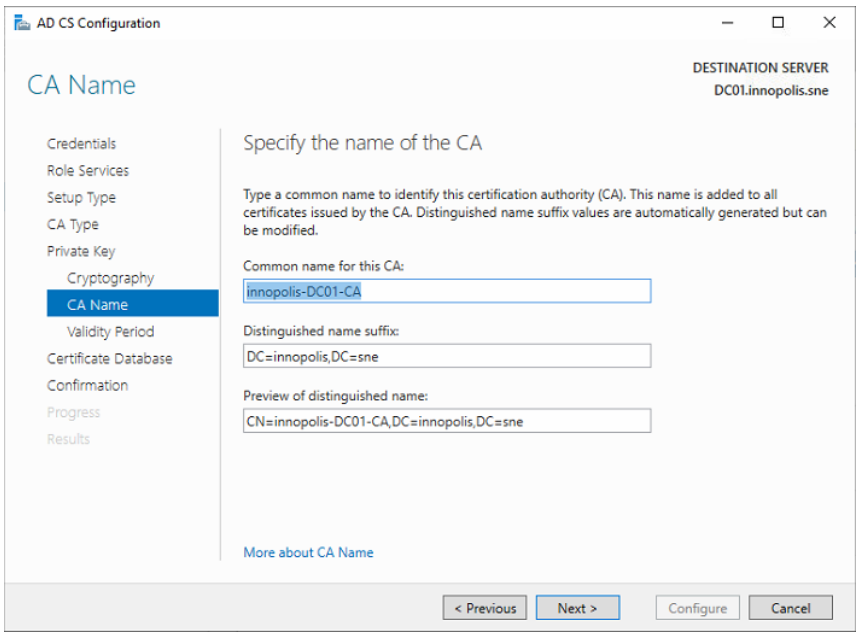

Figure 81: Certification Authority Common Name and Distinguished Name

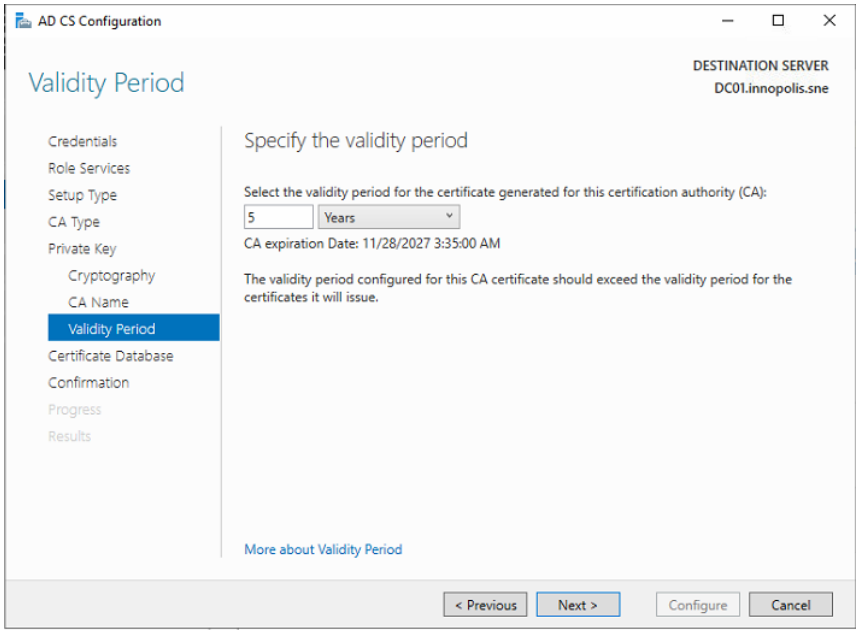

Figure 82: Private Key Validity Period

back out and modify the settings if needed. Otherwise, we will tear down the Certification Authority deployment and start over.

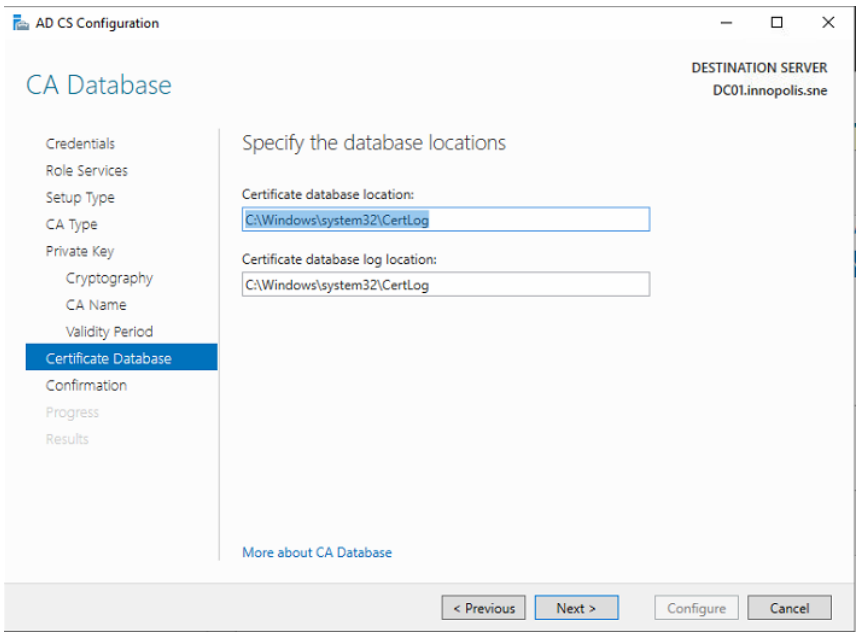

Figure 83: Certificate Authority Database location

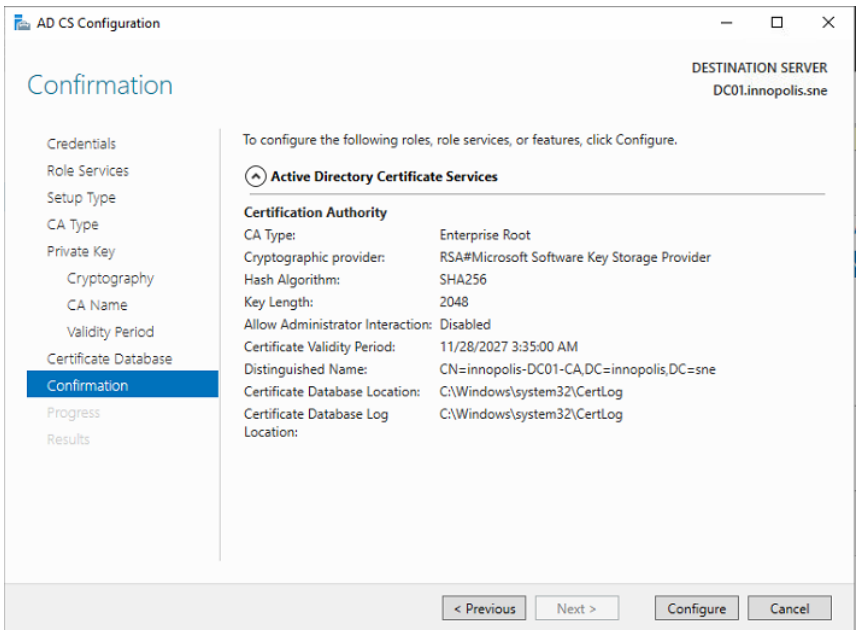

Figure 84: AD CS Configurations Confirmation

2.7 Kerberos Authentication with X.509 Certificate

Every client in a Kerberos authentication procedure will request permission data from the domain controller and then pass that information to the resource. We will use the Kerberos Authentication Template to create a template for OpenVPN LDAPS, generate a private key, and then export a certificate to OpenVPN Cloud to use while authenticating against the

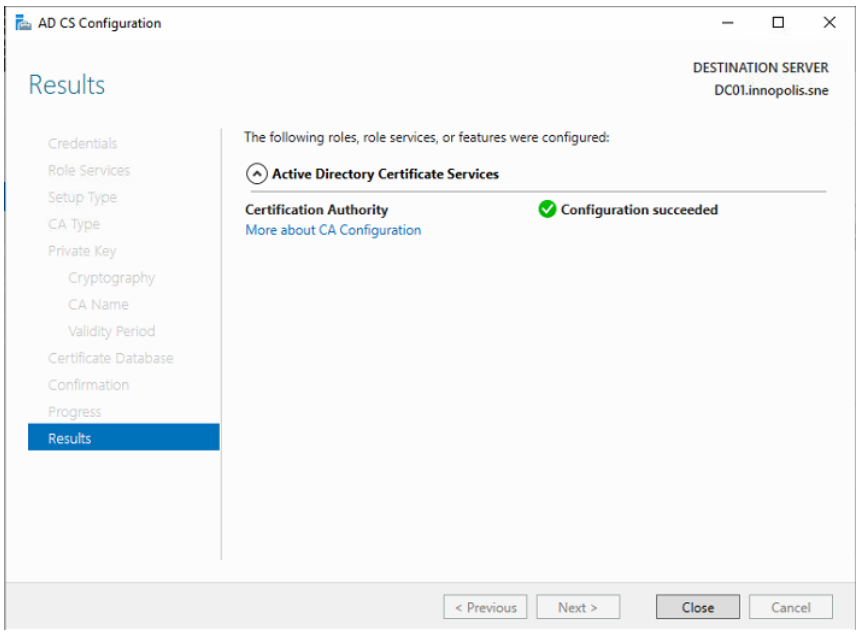

Figure 85: AD CS Deployment Results

Windows Server. It will allow us to transition from LDAP to LDAP over TLS. To start, we will duplicate the template shown in Figure-86, **Kerberos Authentication**, and configure its properties as shown in Figure-87 (*What's new in Kerberos Authentication*, n.d.).

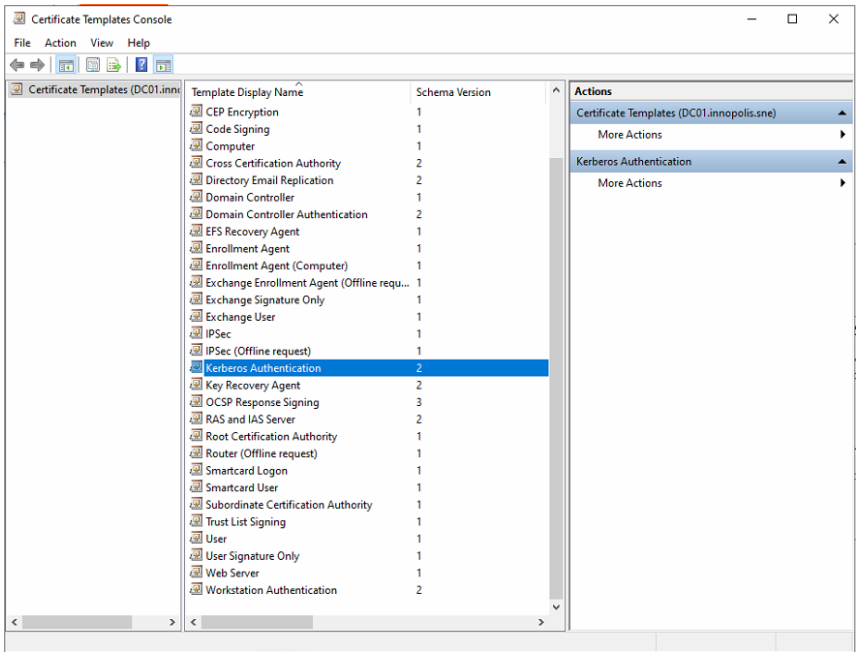

Figure 86: Kerberos Authentication - Certificate Templates Console

We will set the Template display name to **OpenVPN LDAPS** and the validity period to 1 year under the General tab of the new template properties. The exported certificate issued

to OpenVPN Cloud will have a validity period of 1 year per security best practices. Figure 87 shows that we will also publish the certificate in Active Directory.

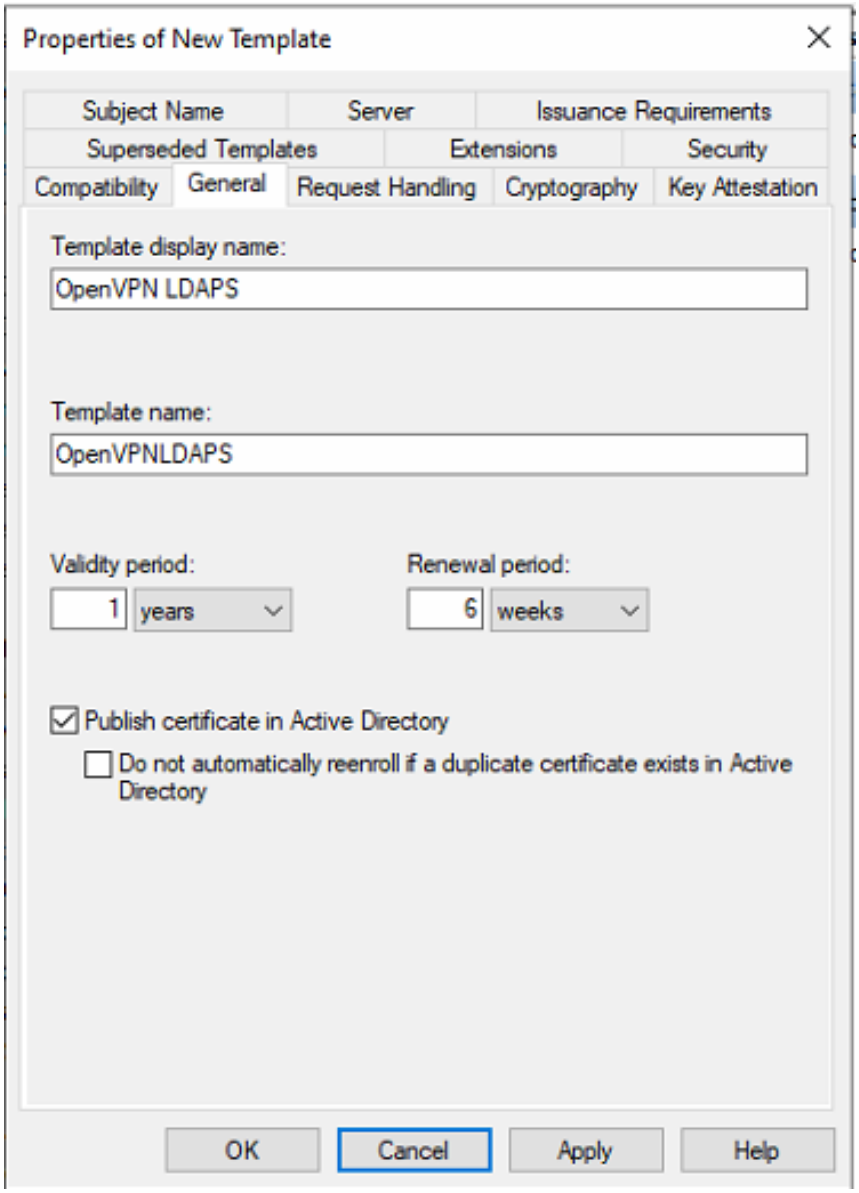

Figure 87: OpenVPN LDAPS Template Properties

Under the Request Handling tab, we will set the purpose to signature and encryption. Moreover, allow the private key to be exported, as we will need this option later. The settings are illustrated in Figure 88.

Under the subject name, we will include this information in an alternate subject name for the DNS name. Furthermore, change the selection to "Build from this Active Directory information." As shown in Figure 89, these settings should be adequate for the OpenVPN LDAPS template and ready for deployment.

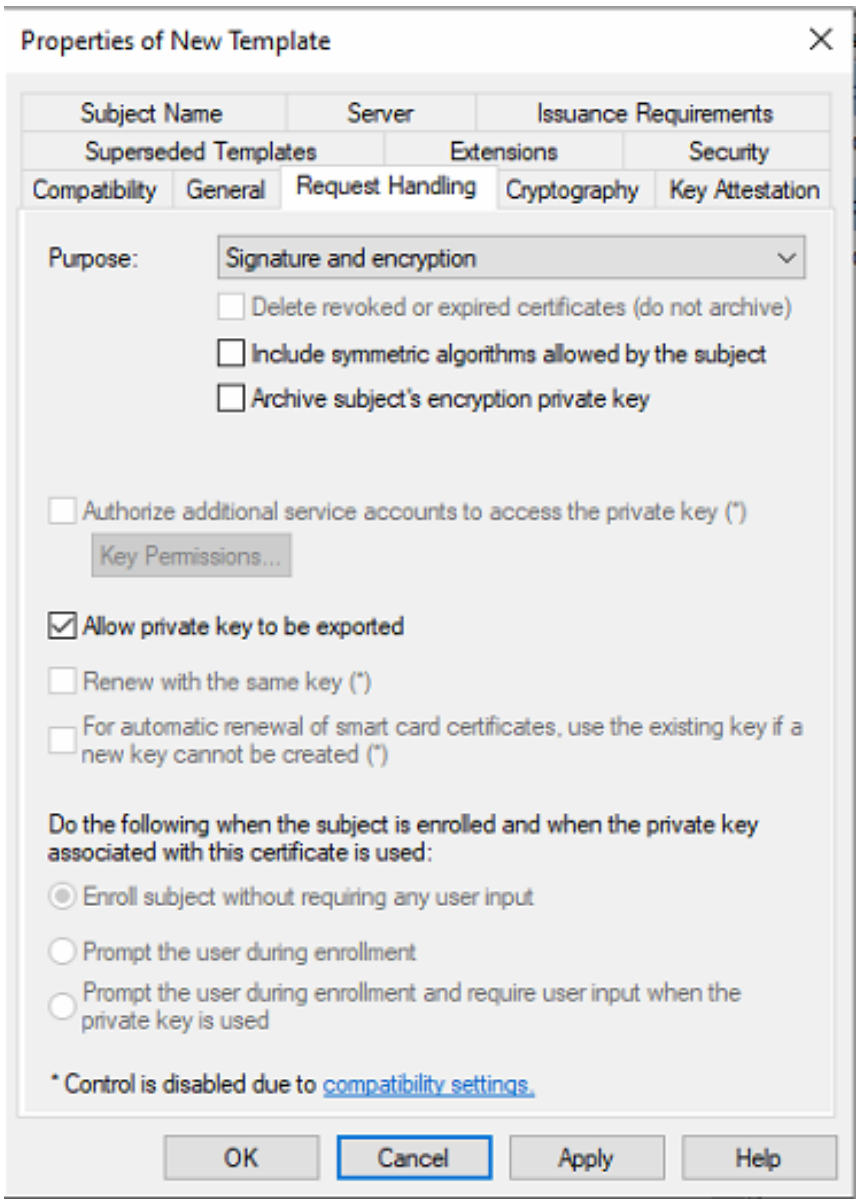

Figure 88: OpenVPN LDAPS Request Handling Properties

Once all is set, we are ready for the certificate template to be issued. Figure 90 shows us right-clicking on the certificate templates and selecting New —>Certificate Template to Issue. In Figure 91, we pick OpenVPN LDAPS, which will have the intended purposes of KDC authentication, smart card logon, and server authentication.

In Figure 92, we will manage the computer account certificate. Then, in Figure 93, select the local computer we are working on.

After that, the management console for the local window server will be shown in Figure 94. We navigate to Certificates->Personal->Certificates, right-click on it, and select All Tasks->Request New Certificate. It will initiate the certificate request wizard, as illustrated

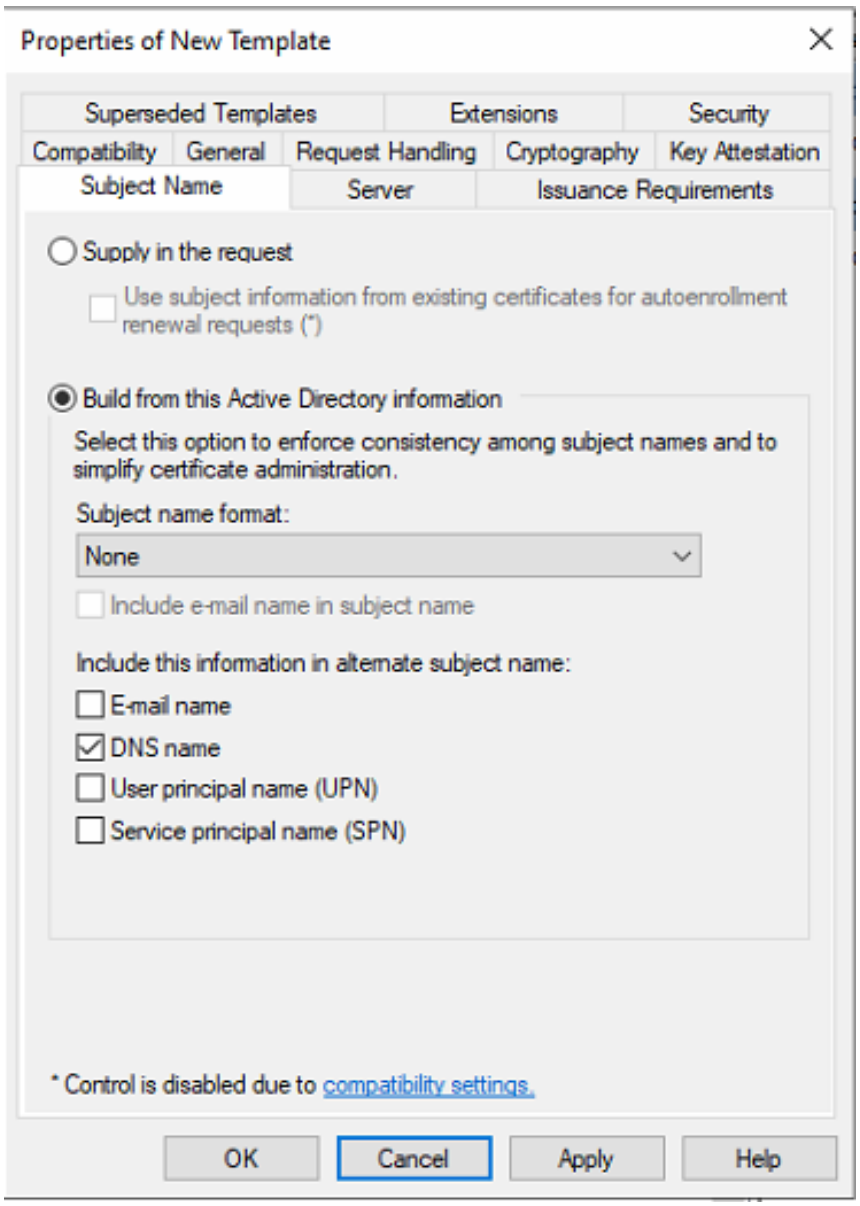

Figure 89: OpenVPN LDAPS New Template Subject Name

in Figure 95. We will need to enroll in a new certificate using the OpenVPN LDAPS template. Once issued (Figure 96), we right-click on the certificate and pick Export. This will initiate the Certificate Export Wizard for the private key, as demonstrated in Figure 97.

As for the certificate enrollment, we specify the Kerberos template we created in the previous steps and named it **OpenVPN LDAPS** as shown in Figure 95. We could also specify "show all templates" and select multiple certificate enrollment requests in this window.

When exporting the certificate format, we specify the formatting as **Base-64 encoded X.509 (.CER)** option, as shown in Figure-98. We will later use the OpenSSL library on the server to convert the format to **.PEM** as supported by the OpenVPN Cloud portal.

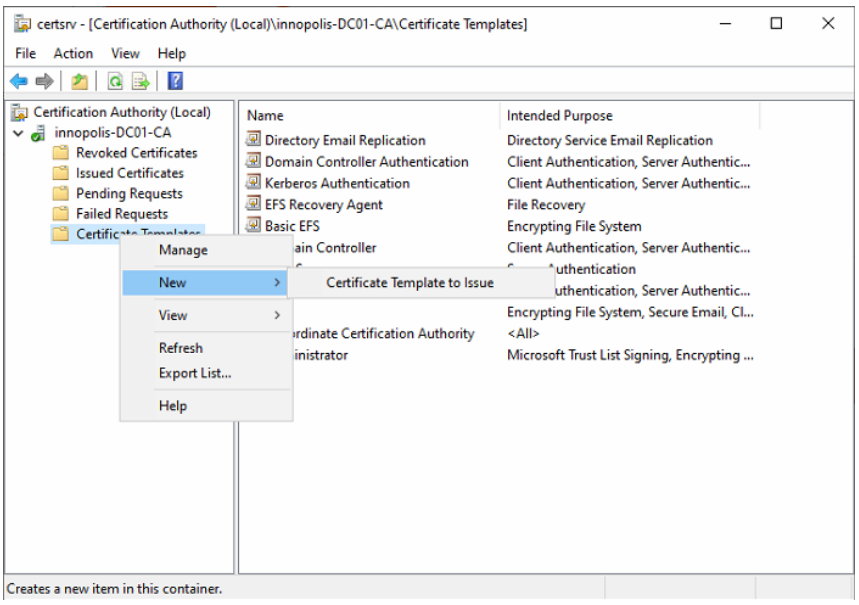

Figure 90: Issue New Certificate Template

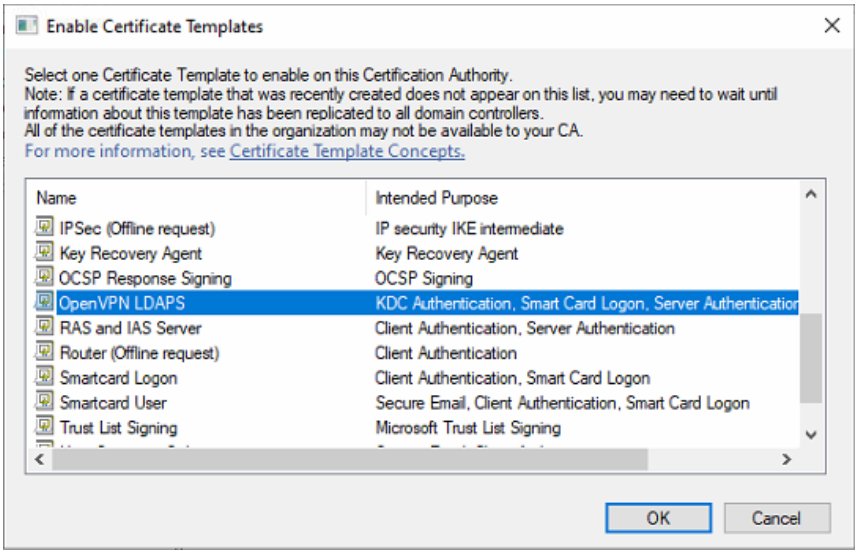

Figure 91: Enable Certificate Templates - OpenVPN LDAPS

Figure 99 shows that the Certificate Export Wizard is completed and ready for OpenSSL conversion in the next step.

Figure 100 shows that we downloaded the GnuWin32 OpenSSL library on Windows Server and converted the exported certificate format from .CER to .PEM as demonstrated.

On the OpenVPN Cloud, we changed the LDAP configuration to enable the SSL checkbox, changed the port number to 636 for LDAP over TLS, and uploaded the converted public certificate, as shown in Figure 101.

The uploaded certificate and the validation date of 1 year are shown in Figure 102.

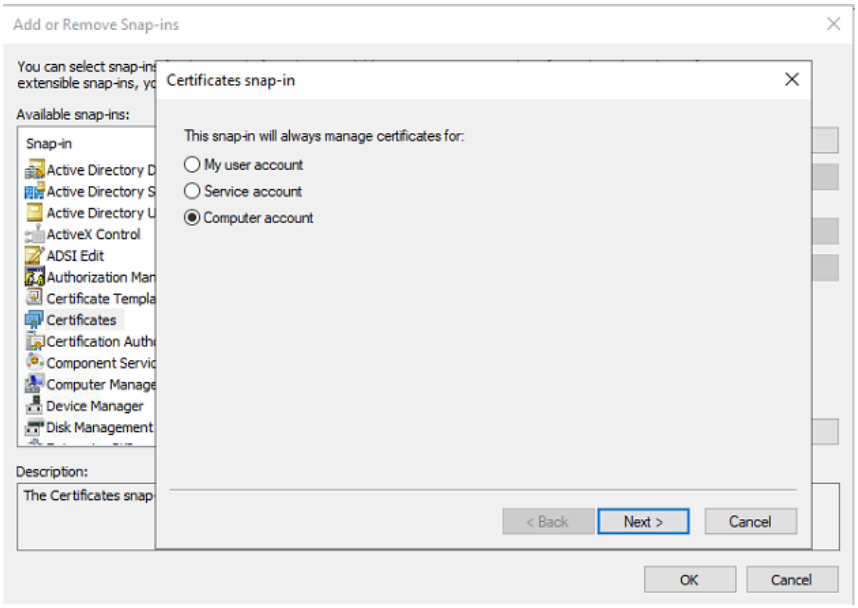

Figure 92: Certificates snap-in - Computer account

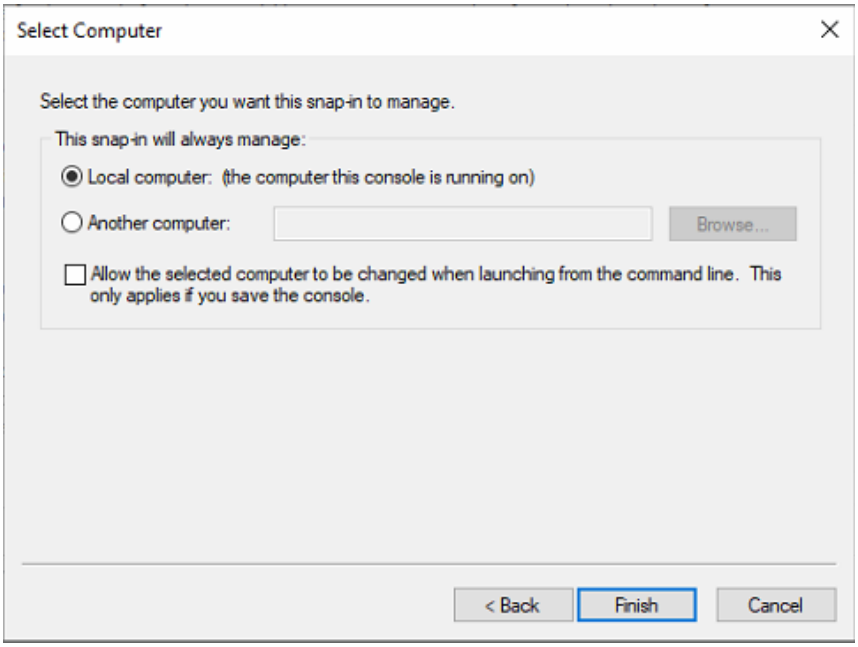

Figure 93: Select Computer to manage via snap-in

To revoke OpenVPN Cloud access, revoke the certificate from Active Directory Certificate Services, and OpenVPN will no longer be able to use AD authentication. OpenVPN Cloud Access will no longer work after one year if the certificate is not renewed.

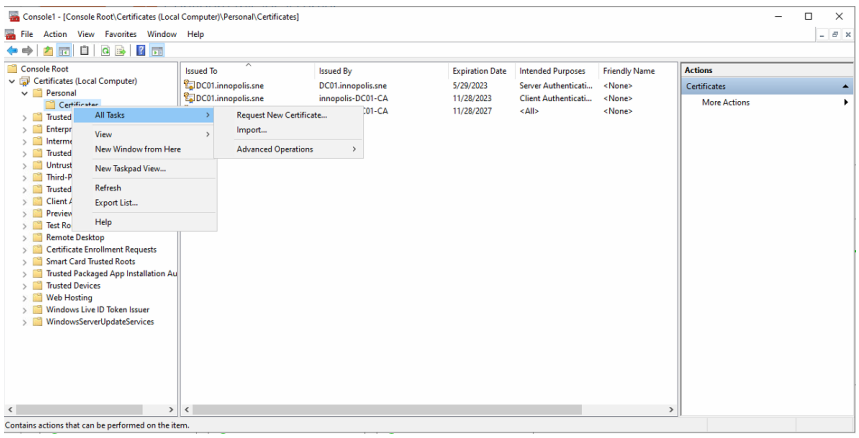

Figure 94: Microsoft Management Console - Certificates (Local Computer)

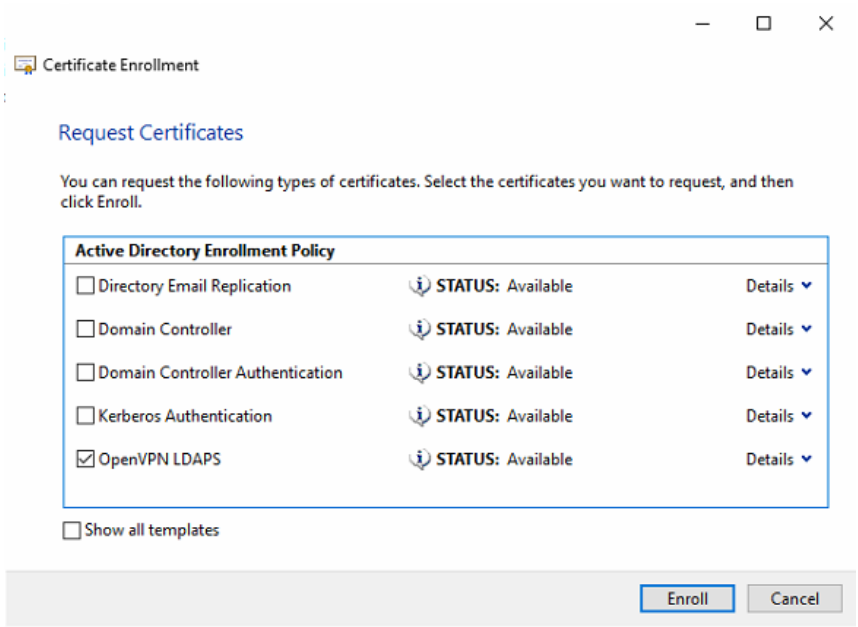

Figure 95: Certificate Enrollment - Request Certificates

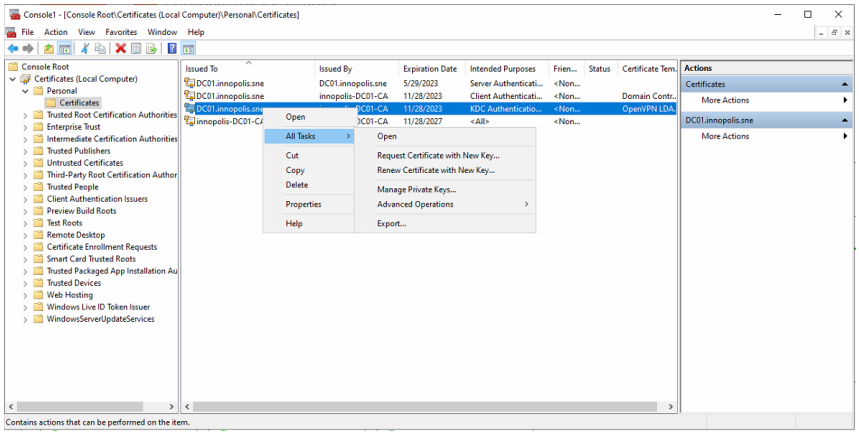

Figure 96: Export DC01.innopolis.sne KDC Certificate

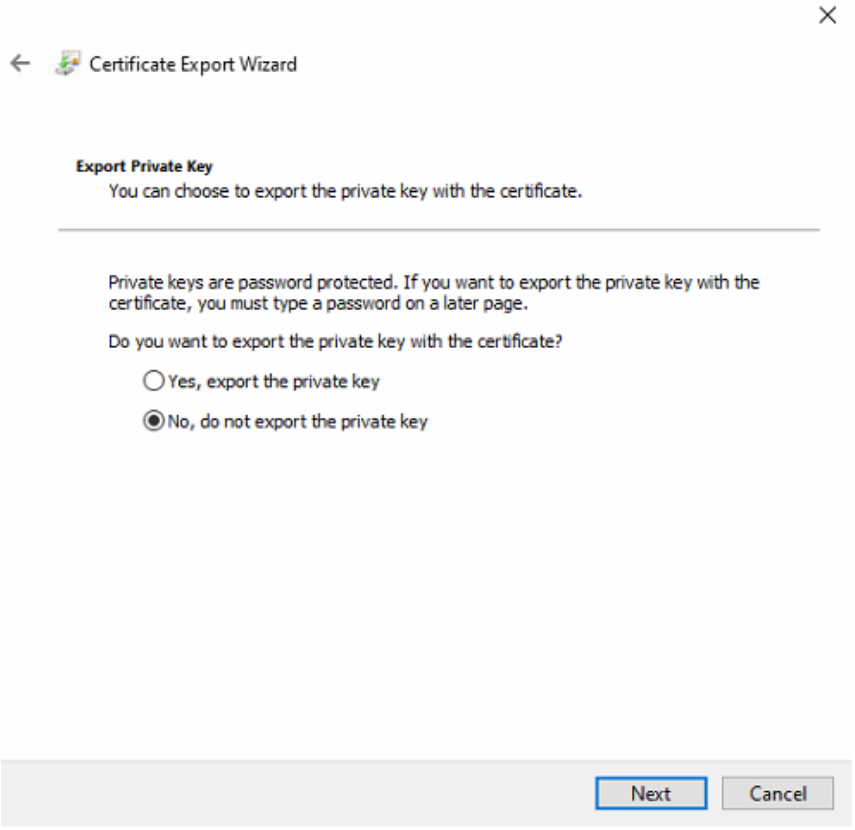

Figure 97: Private Key Export Wizard

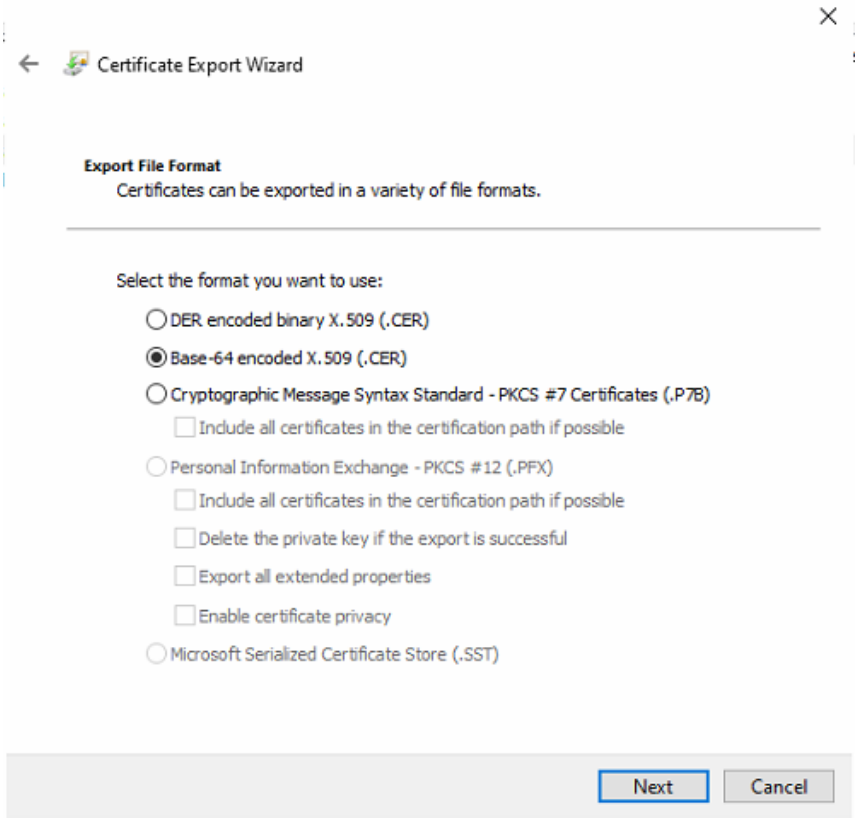

Figure 98: Certificate Export File Format - Base64 X.509

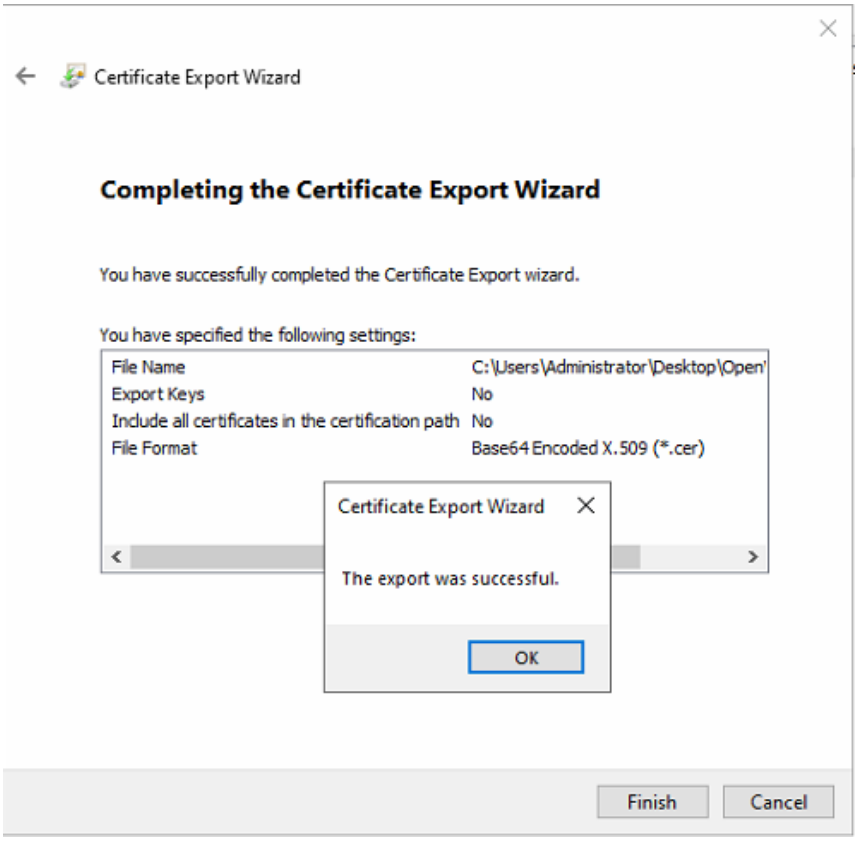

Figure 99: Completing the Certificate Export Wizard

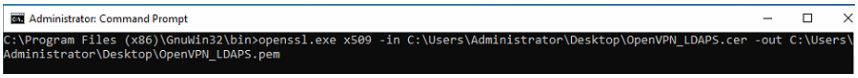

Figure 100: OpenSSL Certificate Conversion .CER to .PEM

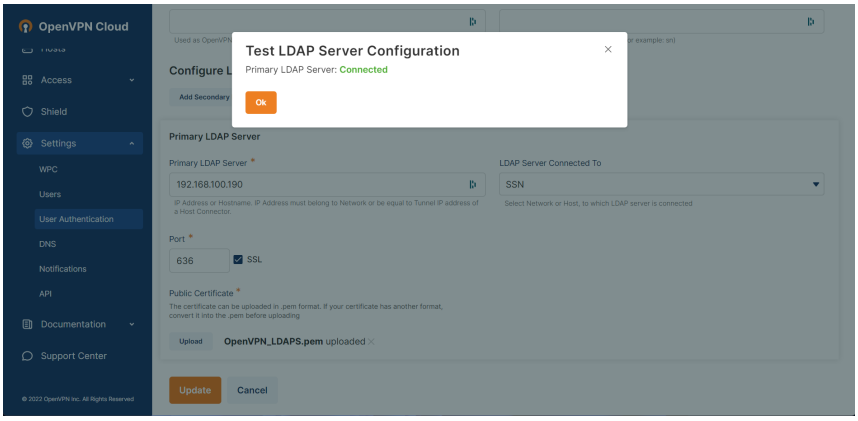

Figure 101: OpenVPN Test LDAPS Server Configuration

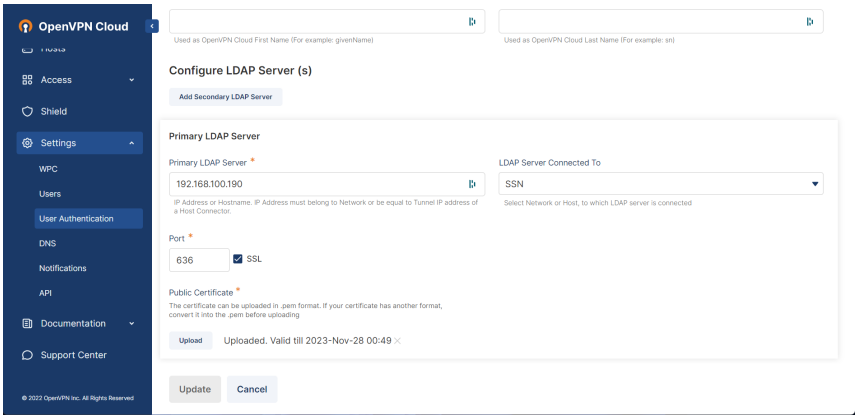

Figure 102: LDAPS Port and Public Certificate with Validation date

Chapter 3

Cisco AnyConnect Secure Mobility VPN

3.1 Brief Introduction to Cisco AnyConnect Secure Mobility

Internet access is no longer limited to a single physical place; rather, users and their devices may be found connecting from everywhere from the office to the living room to the airport to the local cafe. Users within the network were shielded from potential security issues, while those on the perimeter of the network had no acceptable use policy enforcement, weak protection against malware, and a greater chance of data loss. While it is in the best interest of business to provide workers and partners the fredom to do their job from any location using any device, it is equally crucial that company interests and assets be protected around the clock from any potential Internet-based dangers. However, when users or devices are not connected to the network, or when data is not routed via the security solutions, traditional network and content security solutions are rendered useless. Web Security appliance's web filtering services can be seamlessly integrated with Cisco's AnyConnect Secure Mobility, which can extend the network perimeter to distant endpoints. With Cisco AnyConnect Secure Mobility, IT managers can enforce strict security policies across the board and mobile users can enjoy a more streamlined, always-protected experience regardless of whatever platform they are on.

AnyConnect Secure Mobility is a collection of features across the following Cisco products:

- Cisco IronPort Web Security appliance (WSA)
- Cisco ASA 5500 series adaptive security appliance (ASA)
- Cisco AnyConnect client

The following are some of the ways in which Cisco AnyConnect Secure Mobility helps mobile workers:

- **Secure, continuous connection**: Remote access connection for AnyConnect Secure Mobility is provided by Cisco AnyConnect. Since both the person and the device must be verified and authroized before being granted access to the network, the connection is safe. Because AnyConnect may be set to remain active even while switching between networks, the connection will remain stable. Even while AnyConnect is constantly on, it is also adaptable enough to apply various regulations depending on location, enabling users to access the Internet in a "captive portal" scenario where they must agree to terms of service before doing so.
- **Security and policy enforcement persisted**: All users, especially those who are mobile or otherwise remotely located, may benefit from the Web Security appliance's context-aware rules, which include the enforcement of acceptable usage regulations and the prevention of malware infections. The adaptive security appliance authenticates the AnyConnect client and sends that information to the Web Security appliance, which then authenticates the user automatically before granting access to the website.

3.2 Recognizing the Functions of AnyConnect Secure Mobility

Control and security can now be extended into truly borderless networks with the help of Cisco AnyConnect Secure Mobility, a set of functionalities available across a variety of Cisco products. Together, the Web Security appliance, the adaptive security appliance, and the Cisco AnyConnect client form AnyConnect Secure Mobility. Users from off-premises and on-the-go can connect to the adaptive security appliance via VPN using the Cisco AnyConnect Secure VPN client. Web traffic from the adaptive security appliance is forwarded to the Web Security appliance, along with the user's IP address and login credentials. With the help

of the Web Security appliance, traffic is scanned for malicious content, acceptable usage standards are enforced, and the user is safeguarded from potential dangers. Any data that the adaptive security appliance determines is appropriate for the user will be sent to them.

Web trafffic is scanned only by the Web Security appliance, not the mobile device's client. Since certain mobile devices have limited processing power, this improves overall performance. Scan Internet traffic on the network, and you may update security patches and acceptable usage regulations instantly, rather than waiting days, weeks, or months to get the changes out to clients. Web requests are recorded and remote traffic is inspected using rules set up for remote access. The AnyConnect client may establish a VPN connection to an adaptive security device and then interact directly with the Web Security appliance, however this depends on how the Web Security appliance is set up.

3.3 Cisco AnyConnect VPN Architectures

Because of the diversity and complexity of enterprise network topologies, several different approaches exist for deploying the AnyConnect Secure Mobility solution. In addition to AnyConnect remote access connection, an adaptive security appliance, a Web security appliance, and, in many circumstances, a WCCP (Web Cache Communication Protocol) capable device are necessary for a successful Secure Mobility deployment. Design constraints, howevver, may force such designs to grow to accomodate new appliances and routers.

Client applications need not be aware of the existence of a proxy server on the network thanks to the WCCP router's ability to transparently reroute web traffic to the WSA. A WCCP router is necessary for the majority of the designs described in this work. Due to a restriction in the WCCP implementation on the adaptive security appliance, the WCCP router is often required in certain scenarios.

Cisco AnyConnect VPN Architecture Scenarios	
Architecture Scenario	Description
Single Subnet	Characteristics of this design approach include an enabled WCCP router running transparently to reroute all Web traffic to the Web Security appliance. There is just one subnet used by the WSA, ASA, and WCCP.
Mutliple Subnets	In this architecture scenario, multiple subnets are very similar to the single subnet architecture with one key difference: the WSA is located on a separate subnet from the ASA and the WCCP.
Explicit Forward Proxy	Client applications are set up to pass all web traffic to the Web Security appliance as a Web Proxy, making this architecture comparable to Single subnet architecture. WCCP is unnecessary.
Non-WCCP Router	This architecture is in case we do not have a WCCP enabled router, although we are using a router in this deployment. If we are using an ASA, then it would support web traffic reroute to the ASA.

IKE Phase 1

```
1.  !==================
2.  !IKE Phase I Configs - Tunnel1
3.  !====================
4.  crypto isakmp policy 200
5.   encryption aes 128
6.   authentication pre-share
7.   group 2
8.   lifetime 28800
9.   hash sha
10.  exit
11.  !
12.  crypto keyring keyring-vpn-01
13.   local-address FastEthernet 2/0
14.   pre-shared-key address

     3.122.9.69 key lqazxsw2

15.  exit
16.  !
17.  crypto isakmp profile isakmp-vpn-01
18.   local-address FastEthernet 2/0
19.   match identity address 3.122.9.69
20.   keyring keyring-vpn-01
21.  exit
```

IKE Phase 2 ISP1-BGP

```
1.  !==================
2.  !IKE Phase 2 Configs - Tunnel1
3.  !====================
4.  crypto ipsec transform-set

    ipsec-prop-vpn-01 esp-aes 128 esp-sha-hmac

5.   mode tunnel
6.  exit
7.  !
8.  crypto ipsec profile ipsec-vpn-01
9.   set pfs group2
10.  set security-association lifetime seconds 3600
11.  set transform-set ipsec-prop-vpn-01
12. exit
13. !
14. crypto ipsec df-bit clear
15. crypto isakmp keepalive 10 10 on-demand
16. crypto ipsec security-association replay window-size 128
17. crypto ipsec fragmentation before-encryption
18. !
```

Figure 103: Cisco IPsec Implemenation - IKEv2 Phase 1 and Phase 2

Chapter 4

Custom VPN Solution on AWS EC2

4.1 Implement our own VPN in Cloud (AWS)

With Internet service providers and government agencies closely monitoring user activity, online privacy has become an important issue of concern. To make matters worse, legislators are already working on data retention rules that would give ISPs the right to collect, keep, and sell your personal information to other parties. Since the entire privacy thing has come to light, many internet users have turned to virtual private networks (VPNs) in an effort to conceal their online identities. The good news is that it's not hard to set up your own cloud VPN server at home and shield your data from prying eyes. Why make your own virtual private network?

Building your own VPN may seem like a hard undertaking, but you do not need to be a oder to achieve it. You may need to learn some technical details, but the payoff is well worth the effort.

- **Cheap** With Amazon Web Services (AWS), we may set up our own personal VPN at no cost for the first year. When compared to the cost of premium VPN services, the cost of using a hosting provider like Linode is far more reasonable.

- **Disposable VPN** Our data will be sent from the Internet Service Provider to the cloud storage service. With a personal VPN, we may start up a brand-new VPN servver and establish a connection in a matter of minutes. At that point, we may terminate our instance it will be as if the VPN server never existed.

- **Increased personal discretion** Our privacy will increase because of the disposable VPN. We may also be giving up our information to the VPN service that might sell it or share it with other parties. We still should not put all our faith in Amazon and the like when it comes to data privacy, but at least we know they will not give out our hosting information to spammers and ads.

Setting up our own VPN in the cloud, though, will not let us get around geo-restrictions or censorship. However, if you have our own VPN, we can still encrypt and safeguard our data, which is more than enough for most of us. The fact that you will have control over our data and be able to delete it at whim is a huge plus (Timothy Joel Timothy, 2022).

4.2 Algo VPN

It is possible to build our own private network connection with the help of a number of different programs; nevertheless, Algo VPN is often considered to be the most reliable and user-friendly of them. It is a collection of scripts that let us establish a secure link to a remote server in the cloud. Algo VPN was built by the people at Trail of Bits, and it is supposed to be simple to use and at the same time giving maximum security. Algo's ability to be used as a disposable VPN is a major plus. However, there are alternative choices available, such as Streisand, which offers a Tor bridge integration and other privacy-enhancing tools. For this project, we will nevertheless continue with Algo VPN since is largely recognized as the finest and most secure. We will also need a cloud server to set up our VPN on.

Using Algo VPN, we will not have to worry about setting up an SSH connection to a server and running a series of convoluted command lines in order to install a VPN. Simply visit the site homepage and choose to Sign in into Amazon Web Services. Then go to Services ->IAM. The Users menu will be located on the left side of the screen. Select to add a new user. Create a login name and check the box labeled "Programmatic Access." Choose to directly attach existing policies. To look for administration directives, enter that word. Go to "Administrator Access" then proceed and download the CSV button found on the last screen. The file contains several numbers and access keys needed to start up Algo VPN. We may now proceed by selecting Close.

All we need is access to the command line on our machine to install Algo. Windows users

will need the Windows Subsystem for Linux in order to make use of Algo.

4.2.1 For Windows 10 users

- The Settings menu is where we need to be.
- Go to For Developers after updating and security.
- Activate the Pgorammer Mode by clicking the button.
- As soon as the last piece of software has been installed, go to the Control Panel and then the Programs menu.
- Windows controls that can be toggled with a click.
- The Windows Subsystem for Linux may be enabled by scrolling down and checking the box next to it; after that is done, click OK.
- Following application installation, Windows will restart. The Linux Bash shell has been installed, and it can be accessed by typing "Bash" in the search bar of the launcher. You only need to open it and respond to the few questions it asks. After that, Windows will download and install aditional applications.
- When everything is done, we will be brought to the prompt. To continue, please type the following.
- sudo apt-get update && sudo apt-get install build-essential libssl-dev python-dev python-virtualenv python-pip python-setuptools git -y
- git clone https://github.com/trailofbits/algo

Create a list of users by typing sudo vim config.cfg. This will launch the vim text editor. We can create new users by entering their names in the "users" section. If we plan on using the VPN on more than one device or sharing it with others, it is a good idea to compile a list of everyone who needs access. Next, we will save our work and close the program. We will have to initiate a rollout. To get things rolling, just enter ./algo. A few questions will be posed to us. In the provider field, enter 2 for Amazon EC2. After that, give the virtual private network the desired naming we plan on deploying, and select a server in any country.

In order to get the most out of the latency and VPN experience, it is best to choose a server that is near to us.

Then, access the CSV file we obtained from Amazon Web Services and copy the AWS Access Key and AWS Secret Key. The next step Algo will demand is VPN on Demand. We can select this option if we want for the VPN to connect immediately. Then, we should put yes for the security upgrades questionnaire. The next step is for Algo set up itself on the cloud. It will notify us when processing is finished. The last step is to link our devices to the VPN connection (Timothy Joel Timothy, 2022).

4.3 WireGuard

WireGuard is a VPN service that uses cutting-edge encryption technology. It is an attempt to avoid the tremendous hassle that is IPsec while yet providing the same or better speed. Its ultimate goal is to significantly outperform OpenVPN. WireGuard is a VPN that may be used in a wide variety of settings and configurations, from embedded interfaces to supercomputers. Although it was first published for the Linux kernel, it is currently compatible with and commonly deployable on a variety of other platforms. In spite of the fact it is still in the early stages of development, it has the potential to become the most secure and straightforward to implement VPN service available. WireGuards ultimate goal is to make up deployment as simple as SSH. Exchanging extremely basic public keys and establishes a VPN connection, and WireGuard will take care of the rest automatically and invisibly. Like Mosh, it can switch IP addresses dynamically. We will not have to handle anything involving connections, state, daemons, or internals. WireGuard provides a user interface that is both simple and effective. Among the cutting-edge cryptographic methods used by WireGuard are secure trusted constructions, HKDF, SipHash24, BLAKE2, ChaCha20, Poly1305, Curve25519, and the Noise protocol framework. It has been evaluated by cryptographers and makes cautious and fair decisions (Donenfeld, n.d.).

To configure our devices for Algo Virtual Private Network, we will need to create a profile or certificate for each device that will connect to the VPN. Though the specifics of the procedure wil change from device to device, all of the necessary files may be found in the "algo-master" subfolder of the "configs" folder. Windows VPN setup is more involved than

other platforms, but it is still possible.

Transfer the PEM, P12, and PS1 files from the config folder. To add the the PEM file contents to the Trusted Root certificate repository, just double-click on the PEM certificate. Then, we will execute the following powershell commands:

```
Set-ExecutionPolicy Unrestricted -Scope CurrentUser
```

Finally, we execute the PS1 file that was extracted from the config folder and the VPN service should begin automatically. Our information should be safe from now on.

Chapter 5

Results

Our research showed that OpenVPN is the best VPN solution because it is easy to set up and manage, has open-source code, costs less for hardware, and works with firewalls. Cisco AnyConnect is more for corporate and large enterprise customers because it has more features and is always available. Furthermore, the proactive defense it provides against threats, the strength of the network security it provides, and the ease of management it provides through its reliance on a single agent. Cisco AnyConnect safeguards all devices. It gives users access to the company network at all times, so they can do their jobs even when they are not in the office. It also gives a complete picture of what users and endpoints are doing and lets security measures be put in place effectively. Even though OpenVPN Cybershield has this feature, it is less potent than Cisco's security for virtual private networks. Cisco Umbrella offers businesses the convenience of bridging their ASA firewalls to the cloud. It allows us to use the same IP address up to thirty times globally. A custom VPN solution is more for tech-savvy people and projects that need specific software. In conclusion, Cisco AnyConnect is targeted at large companies. OpenVPN is geared towards small to medium-sized companies. Moreover, the custom VPN solution is geared toward project requirements, one-time use cases, enthusiasts, and new VPN technologies.

Chapter 6

Discussion

The perspective for future research regarding our implementation and testing of VPN solutions is to implement them natively on the cloud. We are interested in testing the different cloud service providers' VPN offerings, such as Microsoft Azure, IBM Cloud, Oracle Cloud, Amazon Web Services, and Google Cloud Compute. We also want to use our cryptographic algorithms to build a VPN solution from scratch for research purposes. We will keep working on this project and move the whole thing to scripts for Cisco AnyConnect and OpenVPN in Ansible Playbook. We will transfer the implementation to the Terraform script for the custom VPN solution and deploy it on the public cloud. Finally, publish the research outcome and scripts on GitHub.

References

Active directory certificate services overview (n.d.).

URL: *https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831740(v=ws.11)*

ChristianMontoya (n.d.), 'Welcome to remote desktop services in windows server 2016'.

URL: *https://learn.microsoft.com/en-us/windows-server/remote/remote-desktop-services/welcome-to-rds*

Dansimp (n.d.), 'Group policy, the group policy management console (gpmc), and internet explorer 11 (internet explorer 11 for it pros) - internet explorer'.

URL: *https://learn.microsoft.com/en-us/internet-explorer/ie11-deploy-guide/group-policy-and-group-policy-mgmt-console-ie11*

Dknappettmsft (n.d.), 'Active directory domain services functional levels in windows server'.

URL: *https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/active-directory-functional-levels*

Donenfeld, J. A. (n.d.), 'Fast, modern, secure vpn tunnel'.

URL: *https://www.wireguard.com/*

Enterprise VPN (2021).

URL: *https://openvpn.net/blog/enterprise-vpn/*

Janicericketts (n.d.), 'Ldap authentication with azure active directory - microsoft entra'.

URL: *https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/auth-ldap*

Justinha (n.d.), 'Overview of azure active directory domain services'.

URL: *https://learn.microsoft.com/en-us/azure/active-directory-domain-services/overview*

OpenVPN (2022).

URL: *https://en.wikipedia.org/wiki/OpenVPN*

OpenVPN cryptographic layer (2019).

URL: *https://openvpn.net/community-resources/openvpn-cryptographic-layer/*

Timothy Joel Timothy, J. (2022), 'How to make your own vpnfor free (updated 2022)'.

URL: *https://www.wizcase.com/blog/how-to-create-your-own-vpn-in-the-cloud/*

What is AWS Directory Service? - AWS directory service (n.d.).

URL: *https://docs.aws.amazon.com/directoryservice/latest/admin-guide/what¡s.html*

What's new in Kerberos Authentication (n.d.).

URL: *https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831747(v=ws.11)*