Remote Access VPN with FreeRADIUS - Debian

-Setup

-Public subnet: 10.10.10.0/24
Debian server: 10.10.10.1/24
Windows client: 10.10.10.10/24
Debian client: 10.10.10.100/24
-Private subnet: 192.168.100.0/24
Debian server: 192.168.100.100/2
-VPN private IP pool: 192.168.3.0/24

-Install freeradius, strongswan and it's plugins

apt install freeradius strongswan strongswan-pki libcharon-extra-plugins

-Create a CA and create the certs

-Create the directories

mkdir -p ~/pki/{cacerts,certs,private}
chmod 700 ~/pki

-Generate the key

pki --gen --type rsa --size 4096 --outform pem > ~/pki/private/ca-key.pem

-Sign the root cert

pki --self --ca --lifetime 3650 --in ~/pki/private/ca-key.pem --type rsa --dn "CN=vpn.wsc2022.kr" --outform pem > ~/pki/cacerts/ca-cert.pem

-Generate a private key for the VPN server

pki --gen --type rsa --size 4096 --outform pem > ~/pki/private/server-key.pem

-Create and sign the VPN server cert. If you use the DNS name of the server in the CN and SAN fields you'll only need one SAN field

pki --pub --in ~/pki/private/server-key.pem --type rsa | pki --issue --lifetime 1825 --cacert ~/pki/cacerts/ca-cert.pem --cakey ~/pki/private/ca-key.pem --dn „CN=10.10.10.1" --san @10.10.10.1 --san 10.10.10.1 --flag serverAuth --flag ikeIntermediate --outform pem > ~/pki/certs/server-cert.pem

-Copy the certs and keys to /etc/ipsec.d and /etc/freeradius/3.0

cp -r ~/pki/* /etc/ipsec.d
cp -r ~/pki/* /etc/freeradius/3.0

-Configure Strongswan-

-Edit /etc/ipsec.conf

```
config setup
        uniqueids = no
        charondebug = „ike 1, knl 1, cfg 0"
conn ikev2-vpn
        auto = add
        compress = no
         type = tunnel
        keyexchange = ikev2
        fragmentation = yes
        forceencaps = yes
        dpdaction = clear
        dpddelay = 300s
        rekey = no
        left = %any
        leftid = 10.10.10.1 #you can use a domain name aswell: leftid = @domain.tld
```

leftcert = server-cert.pem
leftsendcert = always
leftsubnet = 192.168.100.0/24
right = %any
rightid = %any
rightauth = eap-radius
rightsourceip = 192.168.3.0/24
rightdns = 8.8.8.8,8.8.4.4
rightsendcert = never
eap_identity = %identity
ike = aes128-sha1-modp1024!
esp = aes128-sha1!

```
  GNU nano 5.4                                  /etc/ipsec.conf *
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup
        uniqueids = no
        charondebug = "ike 1, knl 1, cfg 0"
# Add connections here.
conn ikev2-vpn
        auto = add
        compress = no
        type = tunnel
        keyexchange = ikev2
        fragmentation = yes
        forceencaps = yes
        dpdaction = clear
        dpddelay = 300s
        rekey = no
        left = %any
        leftid = 10.10.10.1
        leftcert = server-cert.pem
        leftsendcert = always
        leftsubnet = 192.168.100.0/24
        right = %any
        rightid = %any
        rightauth = eap-radius
        rightsourceip = 192.168.3.0/24
        rightdns = 8.8.8.8,8.8.4.4
        rightsendcert = never
        eap_identity = %any
        ike = aes128-sha1-modp1024!
        esp = aes128-sha1!
```

-Edit /etc/ipsec.secrets

: RSA „server-key.pem”

```
  GNU nano 5.4                                    /etc/ipsec.secrets
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.

: RSA "server-key.pem"
```

-Edit /etc/strongswan.conf

-In the plugins module define the eap-radius module and the radius server itself

```
  GNU nano 5.4                                /etc/strongswan.conf
# strongswan.conf - strongSwan configuration file
#
# Refer to the strongswan.conf(5) manpage for details
#
# Configuration changes should be made in the included files

charon {
        load_modular = yes
        plugins {
                include strongswan.d/charon/*.conf
                eap-radius {
                        servers {
                                server-a {
                                        #"server-a" is just a name for the RADIUS server
                                        # it can be anything as far as I know
                                        address = 10.10.10.1
                                        secret = supersecret
                                }
                        }
                }
        }
}
```

-Configure FreeRADIUS-

-Edit /etc/freeradius/3.0/clients.conf

-Define the VPN server as a RADIUS client at the end of the file

client 10.10.10.1 {
    secret = <random_secret>
    shortname = vpn
}

```
  GNU nano 5.4                          /etc/freeradius/3.0/clients.conf
#  the same as above, but they are nested inside of a section.
#
#  You can have as many per-socket client lists as you have "listen"
#  sections, or you can re-use a list among multiple "listen" sections.
#
#  Un-comment this section, and edit a "listen" section to add:
#  "clients = per_socket_clients".  That IP address/port combination
#  will then accept ONLY the clients listed in this section.
#
#clients per_socket_clients {
#       client socket_client {
#               ipaddr = 192.0.2.4
#               secret = testing123
#       }
#}

client 10.10.10.1 {
        secret = supersecret
        shortname = vpn
}
```

-Edit /etc/freeradius/3.0/users

-Define the users that can authenticate

"<username>" Cleartext-Password := "<userpassword>"

```
  GNU nano 5.4                                /etc/freeradius/3.0/users
# # Last default: shell on the local terminal server.
# #
# DEFAULT
#       Service-Type = Administrative-User


# On no match, the user is denied access.


############################################################
# You should add test accounts to the TOP of this file!    #
# See the example user "bob" above.                        #
############################################################

"bob"   Cleartext-Password := "bob"
"alice" Cleartext-Password := "alice"
```

-Edit /etc/freeradius/3.0/mods-enabled/mschap

-Uncomment the following lines and change their values to "yes"

use_mppe = yes

require_encryption = yes

require_strong = yes

```
  GNU nano 5.4                        /etc/freeradius/3.0/mods-enabled/mschap
#  This module supports MS-CHAP and MS-CHAPv2 authentication.
#  It also enforces the SMB-Account-Ctrl attribute.
#
mschap {
        #
        #  If you are using /etc/smbpasswd, see the 'passwd'
        #  module for an example of how to use /etc/smbpasswd
        #

        #
        #  If use_mppe is not set to no mschap, will
        #  add MS-CHAP-MPPE-Keys for MS-CHAPv1 and
        #  MS-MPPE-Recv-Key/MS-MPPE-Send-Key for MS-CHAPv2
        #
        use_mppe = yes

        #
        #  If MPPE is enabled, require_encryption makes
        #  encryption moderate
        #
        require_encryption = yes

        #
        #  require_strong always requires 128 bit key
        #  encryption
        #
        require_strong = yes
```

-Edit /etc/freeradius/3.0/mods-enabled/eap

-In the eap module set default_eap_type to peap

```
  GNU nano 5.4                        /etc/freeradius/3.0/mods-enabled/eap
#  Whatever you do, do NOT set 'Auth-Type := EAP'.  The server
#  is smart enough to figure this out on its own.  The most
#  common side effect of setting 'Auth-Type := EAP' is that the
#  users then cannot use ANY other authentication method.
#
eap {
        #  Invoke the default supported EAP type when
        #  EAP-Identity response is received.
        #
        #  The incoming EAP messages DO NOT specify which EAP
        #  type they will be using, so it MUST be set here.
        #
        #  For now, only one default EAP type may be used at a time.
        #
        #  If the EAP-Type attribute is set by another module,
        #  then that EAP type takes precedence over the
        #  default type configured here.
        #
        default_eap_type = peap
```

-In the tls-config tls-common module

-comment the private_key_password line if you don't need a password for the private key

-set the private_key_file, certificate_file and ca_file
-uncomment this line: random = /dev/urandom

private_key_file = /etc/freeradius/3.0/private/server-key.pem
certificate_file = /etc/freeradius/3.0/certs/server-cert.pem
ca_file = /etc/freeradius/3.0/cacerts/ca-cert.pem

```
 GNU nano 5.4                    /etc/freeradius/3.0/mods-enabled/eap
        #  authenticate via EAP-TLS!  This is likely not what you want.
        #
        tls-config tls-common {
                #private_key_password =
                private_key_file = /etc/freeradius/3.0/private/server-key.pem
```

```
 GNU nano 5.4                    /etc/freeradius/3.0/mods-enabled/eap
                #
                certificate_file = /etc/freeradius/3.0/certs/server-cert.pem

                #  Trusted Root CA list
                #
                #  This file can contain multiple CA certificates.
                #  ALL of the CA's in this list will be trusted to
                #  issue client certificates for authentication.
                #
                #  In general, you should use self-signed
                #  certificates for 802.1x (EAP) authentication.
                #  In that case, this CA file should contain
                #  *one* CA certificate.
                #
                ca_file = /etc/freeradius/3.0/cacerts/ca-cert.pem
```

```
 GNU nano 5.4                    /etc/freeradius/3.0/mods-enabled/eap

                 #  If your system doesn't have /dev/urandom,
                 #  you will need to create this file, and
                 #  periodically change its contents.
                 #
                 #  For security reasons, FreeRADIUS doesn't
                 #  write to files in its configuration
                 #  directory.
                 #
                 random_file = /dev/urandom
```

-In the peap module set the default_eap_type to mschapv2

```
 GNU nano 5.4                          /etc/freeradius/3.0/mods-enabled/eap
        #  which is separate from the one for the non-tunneled
        #  EAP module.  Inside of the TLS/PEAP tunnel, we
        #  recommend using EAP-MS-CHAPv2.
        #
        peap {
                #  Which tls-config section the TLS negotiation parameters
                #  are in - see EAP-TLS above for an explanation.
                #
                #  In the case that an old configuration from FreeRADIUS
                #  v2.x is being used, all the options of the tls-config
                #  section may also appear instead in the 'tls' section
                #  above. If that is done, the tls= option here (and in
                #  tls above) MUST be commented out.
                #
                tls = tls-common

                #  The tunneled EAP session needs a default
                #  EAP type which is separate from the one for
                #  the non-tunneled EAP module.  Inside of the
                #  PEAP tunnel, we recommend using MS-CHAPv2,
                #  as that is the default type supported by
                #  Windows clients.
                #
                default_eap_type = mschapv2
```

-Enable packet forwarding in /etc/sysctl.conf

-Uncomment the following lines

net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
net.ipv4.conf.all.accept.redirects = 0
net.ipv4.conf.all.send.redirects = 0

-Enable the changes with: sysctl -p

```
root@debvpn:~# sysctl -p
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.send_redirects = 0
root@debvpn:~#
```

-Restart the strongswan and freeradius

systemctl restart freeradius
systemctl restart strongswan-starter

-To check active connections

ipsec status

```
root@debvpn:~# ipsec status
Security Associations (2 up, 0 connecting):
   ikev2-vpn[2]: ESTABLISHED 2 minutes ago, 10.10.10.1[10.10.10.1]...10.10.10.100[bob]
   ikev2-vpn{2}:  INSTALLED, TUNNEL, reqid 2, ESP in UDP SPIs: c045f69d_i c991dcb4_o
   ikev2-vpn{2}:   192.168.100.0/24 === 192.168.3.2/32
   ikev2-vpn[1]: ESTABLISHED 3 minutes ago, 10.10.10.1[10.10.10.1]...10.10.10.10[10.10.10.10]
   ikev2-vpn{1}:  INSTALLED, TUNNEL, reqid 1, ESP in UDP SPIs: ce708cc6_i 32184764_o
   ikev2-vpn{1}:   192.168.100.0/24 === 192.168.3.1/32
root@debvpn:~# _
```

-Configuring the Debian client-

-Install strongswan on the client aswell

apt install strongswan libcharon-extra-plugins

-Copy the CA certificate from the server to /etc/ipsec.d/cacerts

-To ensure the VPN only runs on demand, disable it from running automatically

systemctl disable –now strongswan-starter

-Edit the /etc/ipsec.secrets file

<username> : EAP „<password>”

```
  GNU nano 5.4                    /etc/ipsec.secrets
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.

bob : EAP "bob"
```

-Edit the /etc/ipsec.conf

conn ikev2-rw

right = 10.10.10.1 #You can use domain name
rightid = 10.10.10.1 #You can use domain name
rightsubnet = 0.0.0.0/0
rightauth = pubkey
leftsourceip = %config
leftid = <username> #Enter a username from /etc/ipsec.secrets
leftauth = eap-mschapv2
eap_identity = %identity
auto = start
ike = aes128-sha1-modp1024! #Needs to be same as it’s on the server
esp = aes128-sha1! #Needs to be the same as it’s on the server

```
  GNU nano 5.4                              /etc/ipsec.conf
conn ikev2-rw
        right = 10.10.10.1
        rightid = 10.10.10.1
        rightsubnet = 0.0.0.0/0
        rightauth = pubkey
        leftsourceip = %config
        leftid = bob
        leftauth = eap-mschapv2
        eap_identity = %identity
        auto = start
        ike = aes128-sha1-modp1024!
        esp = aes128-sha1!
```

-To connect to the VPN

systemctl start strongswan-starter

-To disconnect

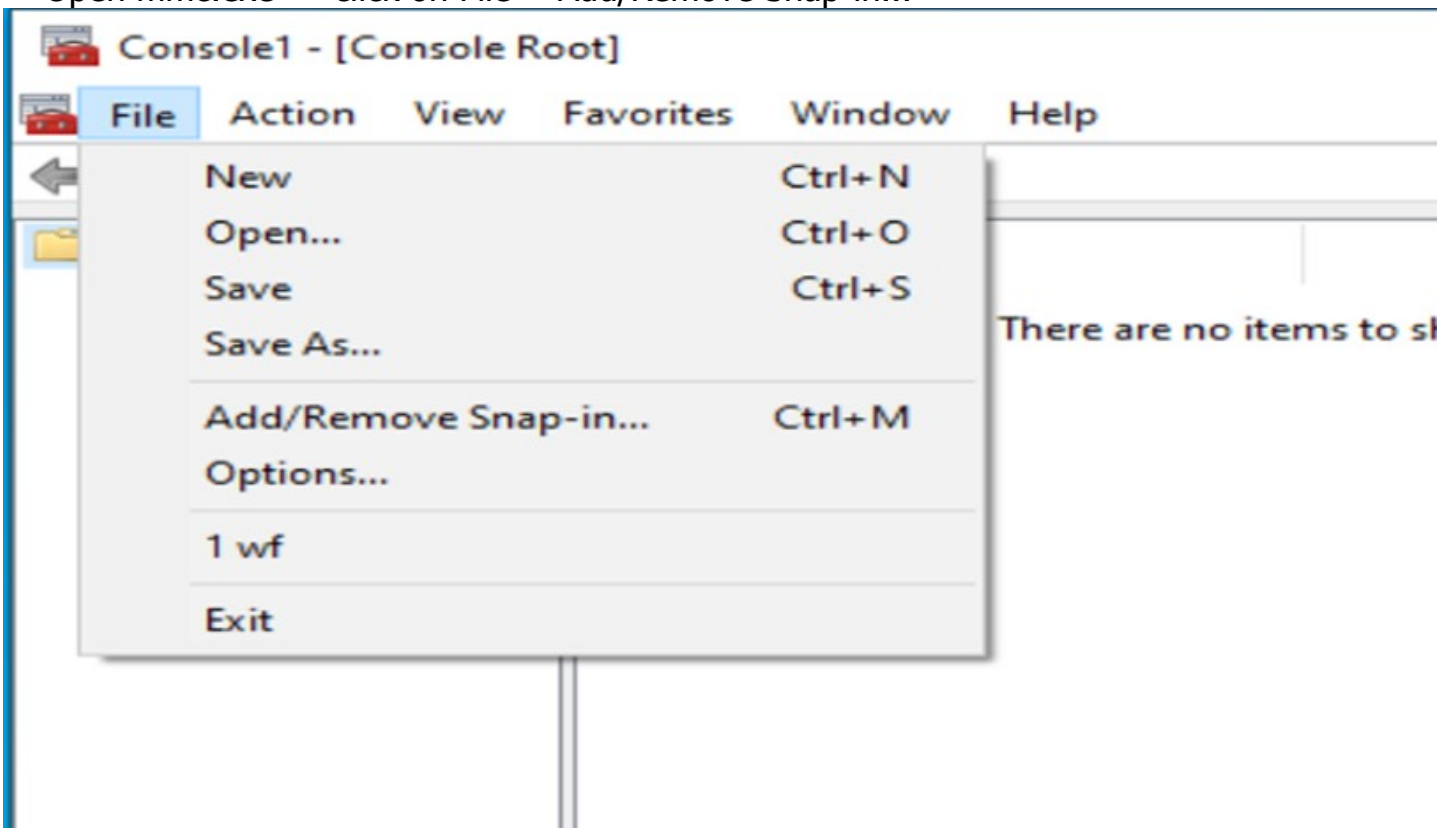
systemctl stop strongswan-starter

-Configuring the Windows client-

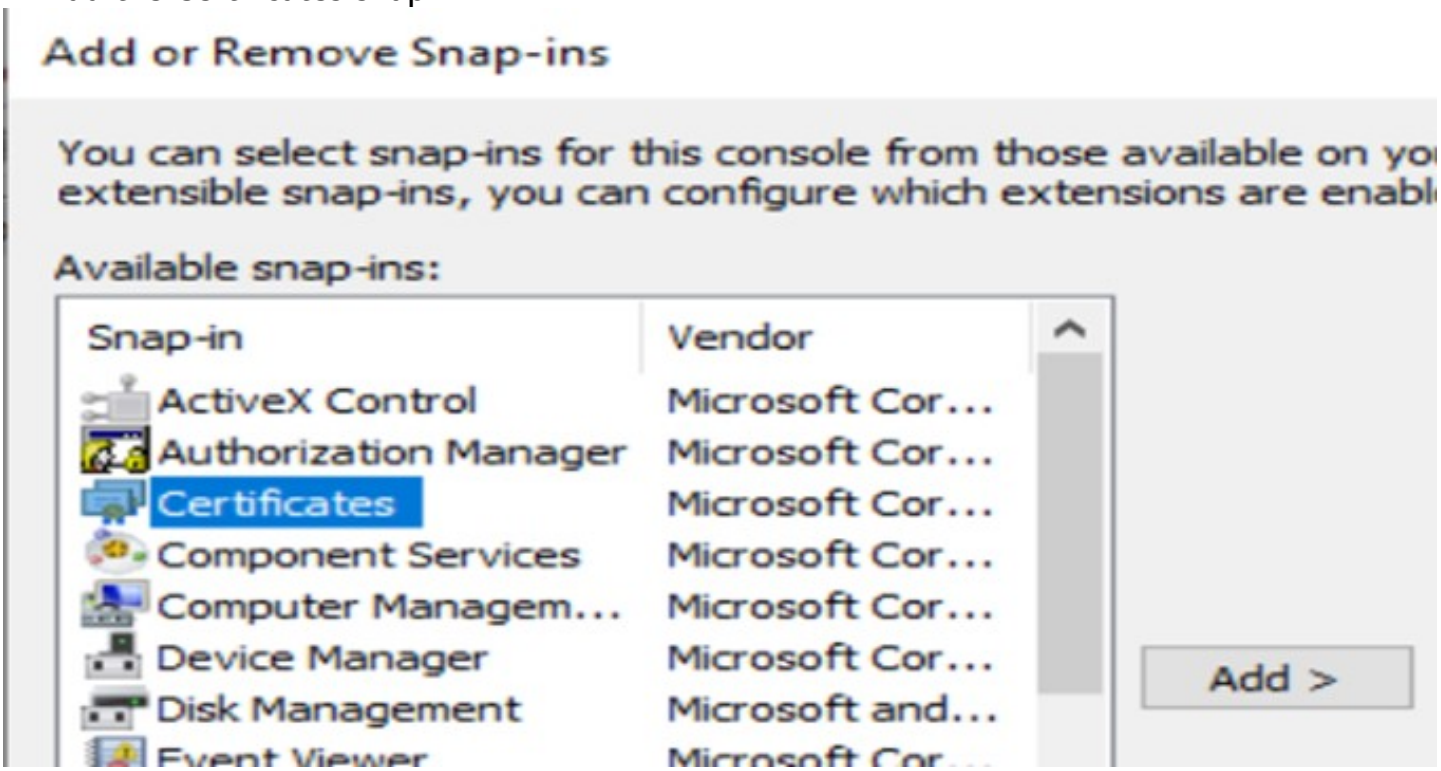
-Copy the CA certificate from the server

-Import the root cert

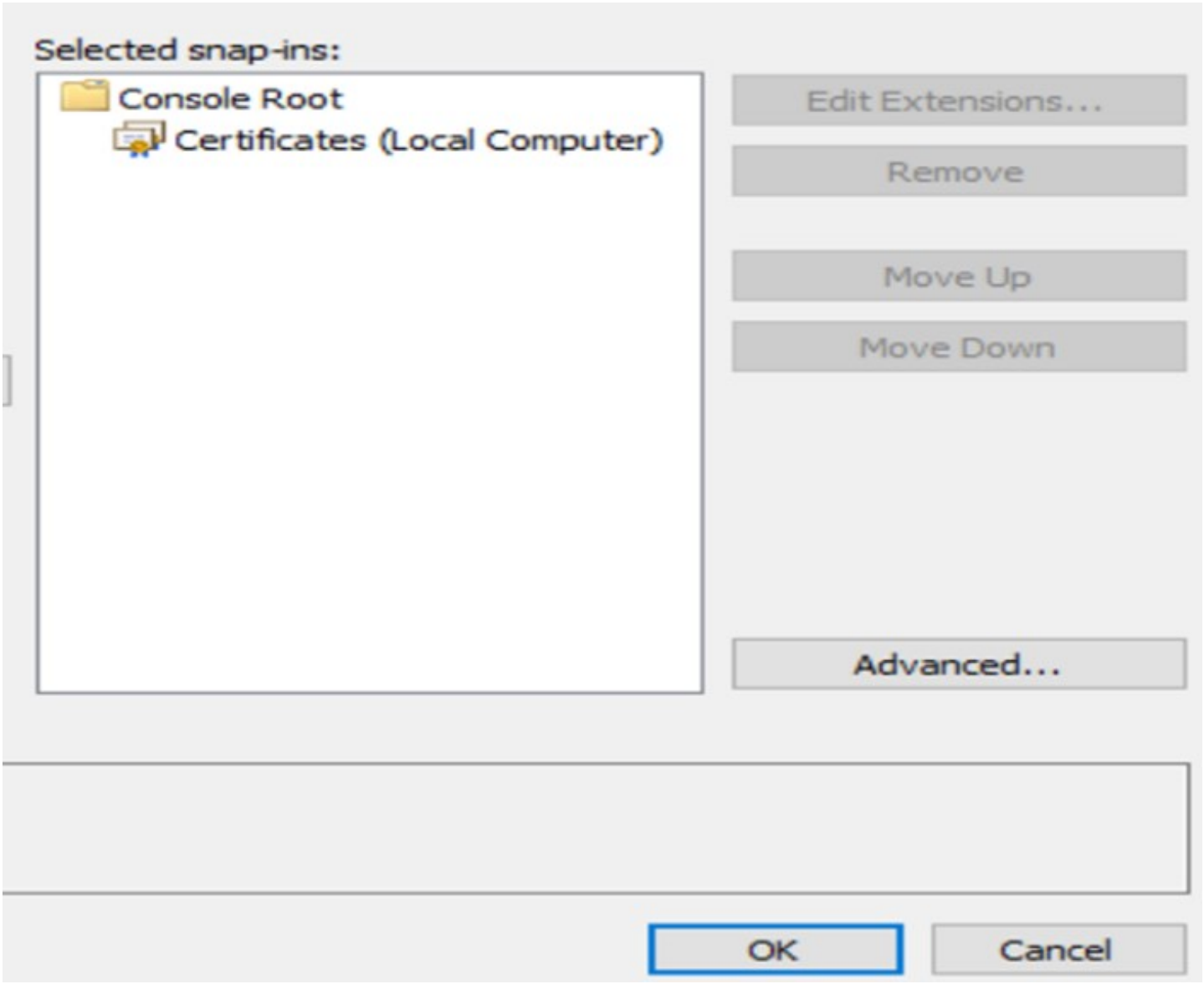
-Open mmc.exe -> click on File > Add/Remove Snap-in...



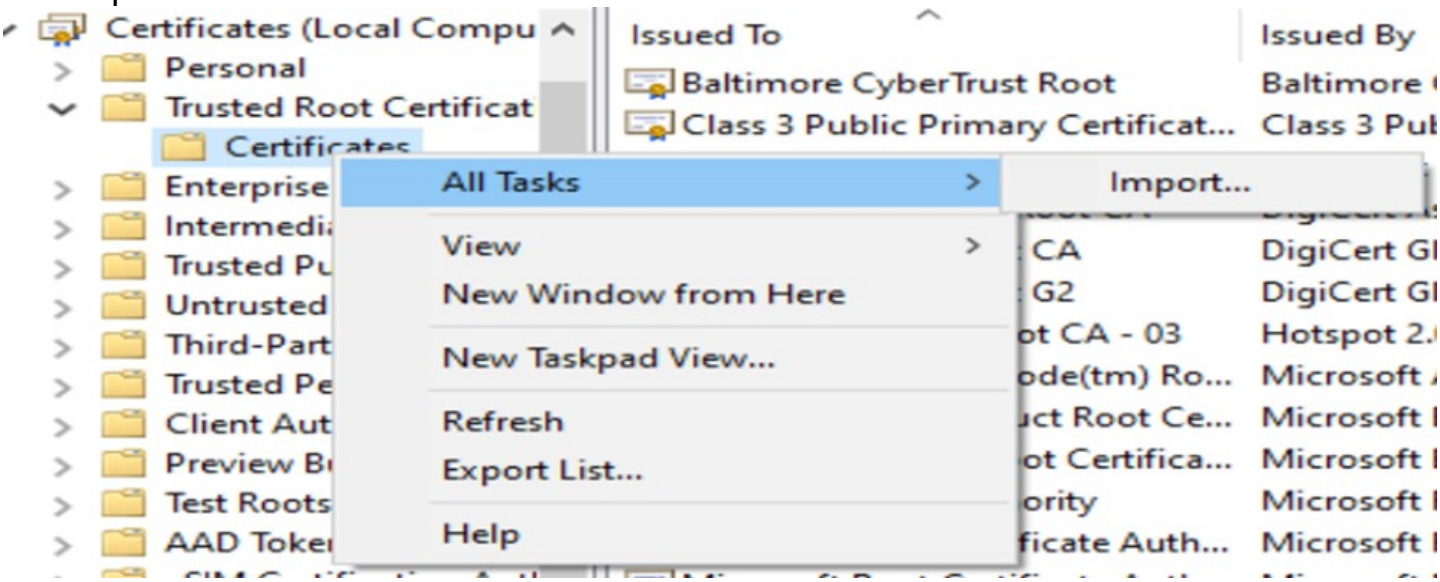
-Add the Certificates snap-in



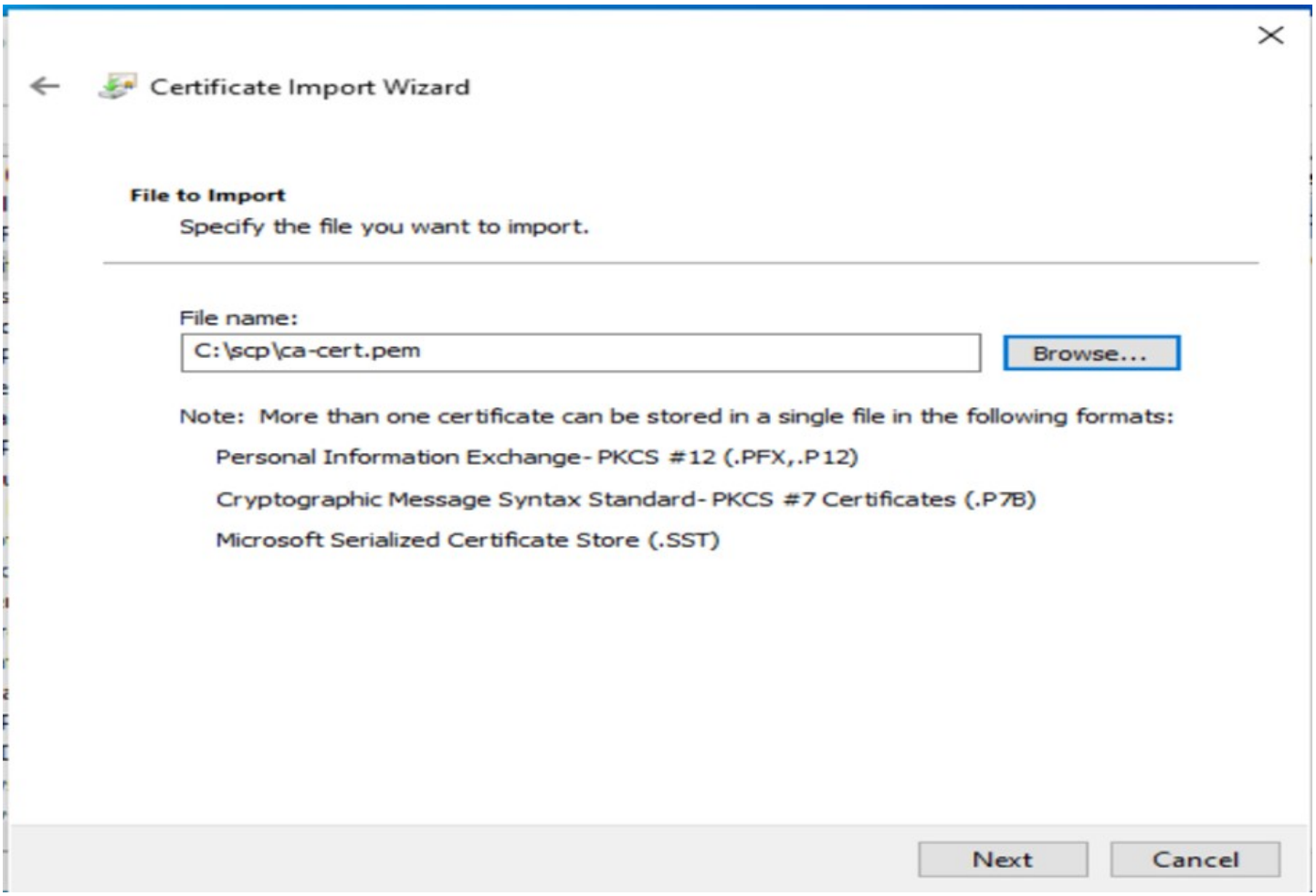
-Select Computer Account, then Local Computer, then OK



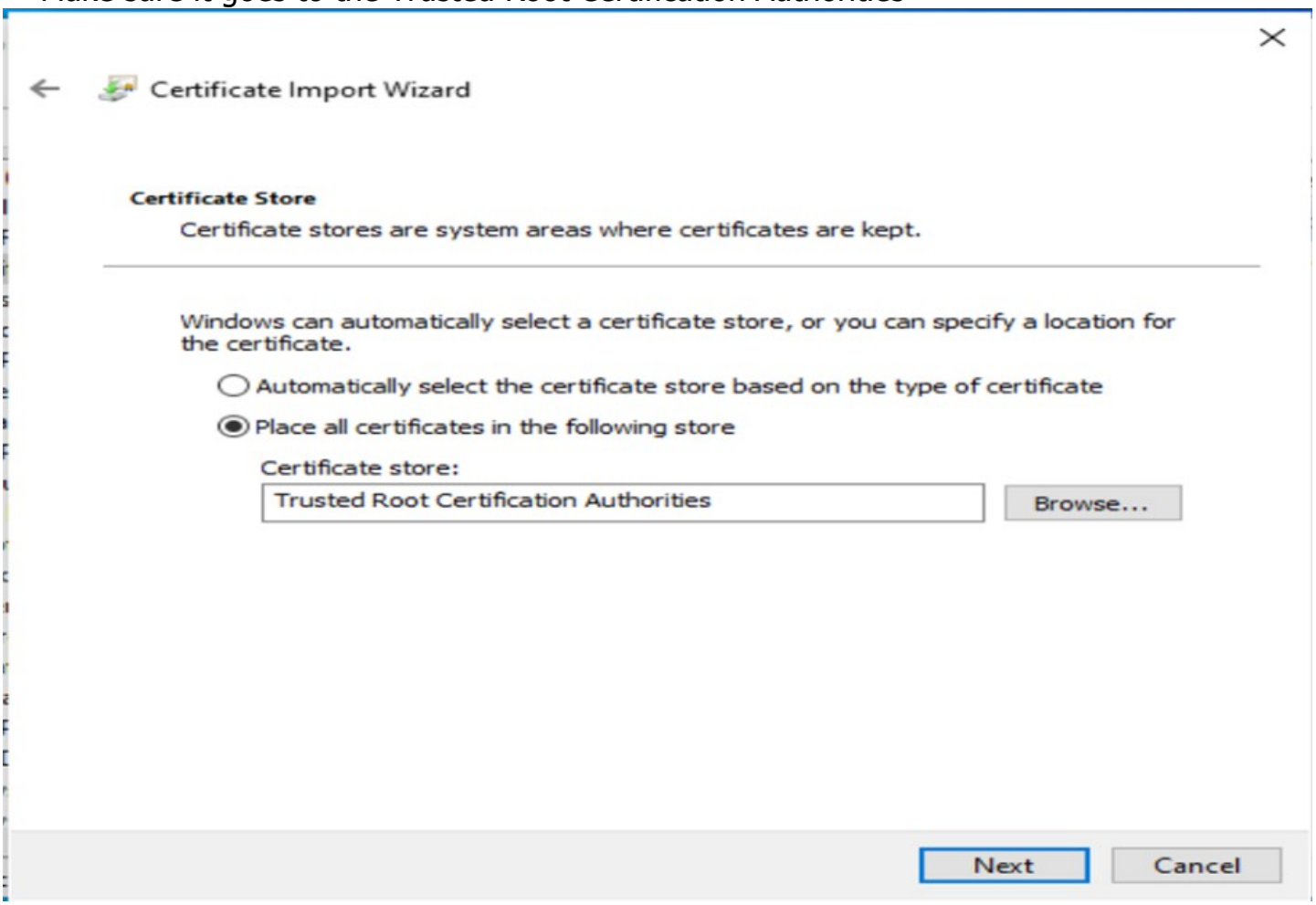
-Open Trusted Root Certification Authorities and right click on Certificates, then click on All tasks > Import



-Specify the cert you copied over

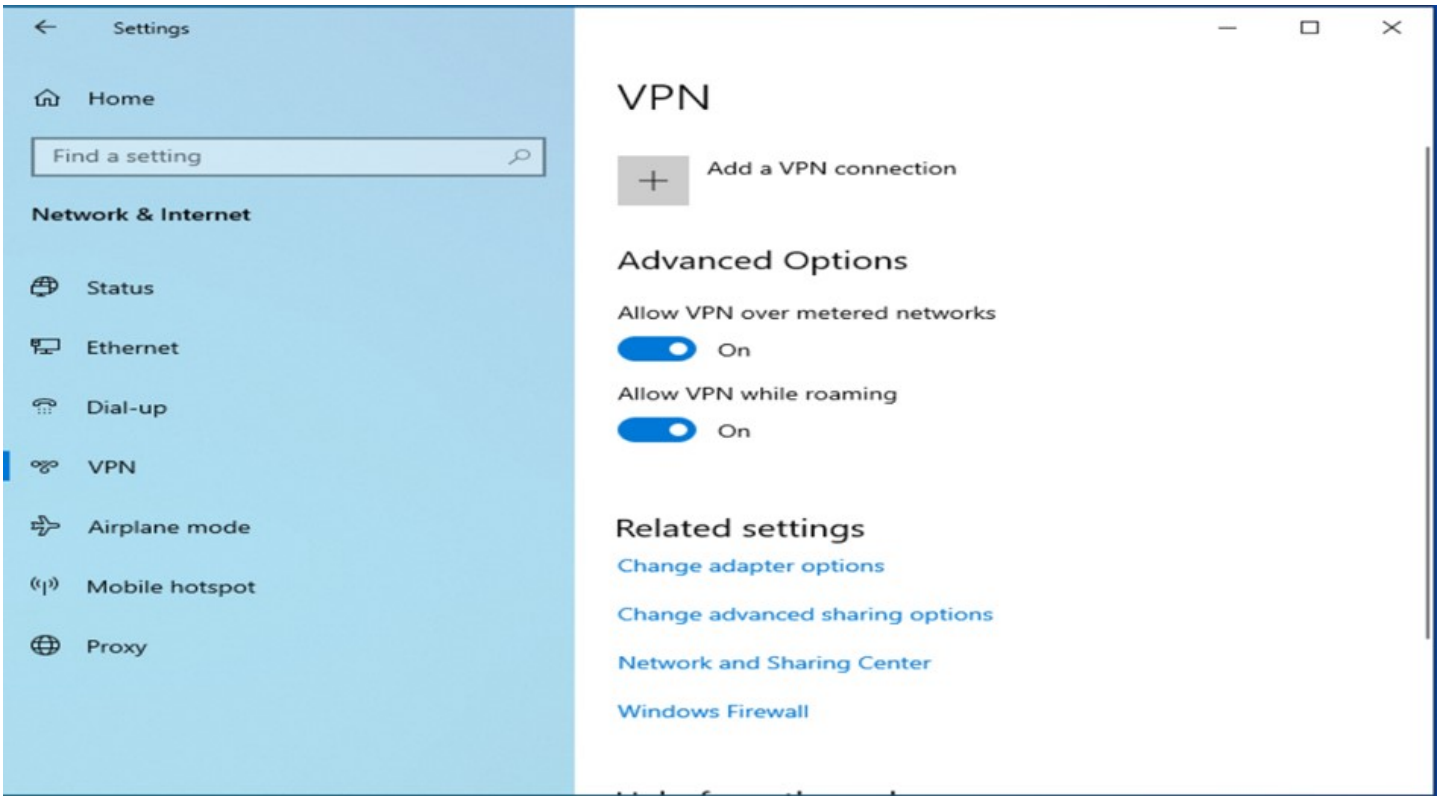


-Make sure it goes to the Trusted Root Certification Authorities

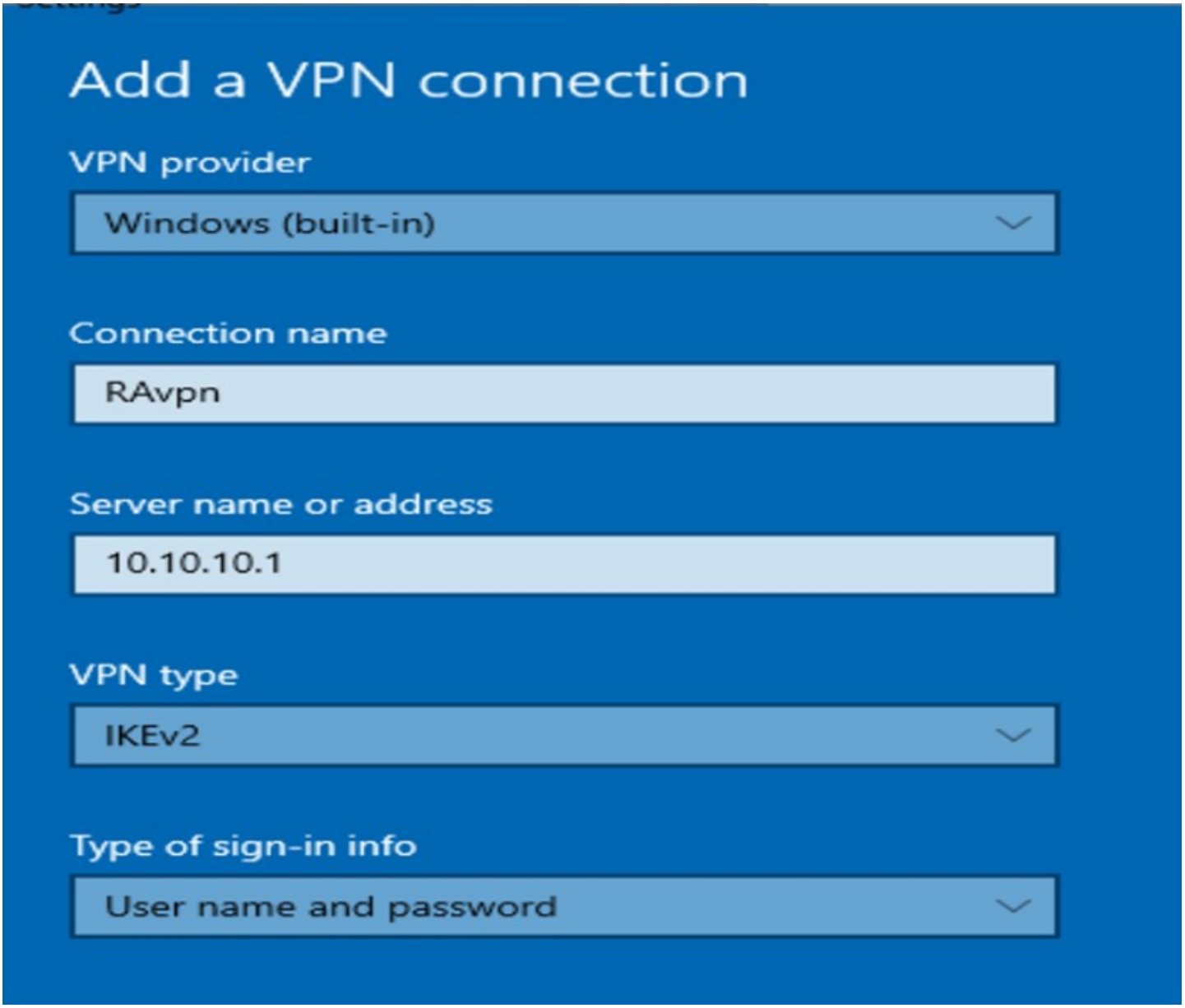


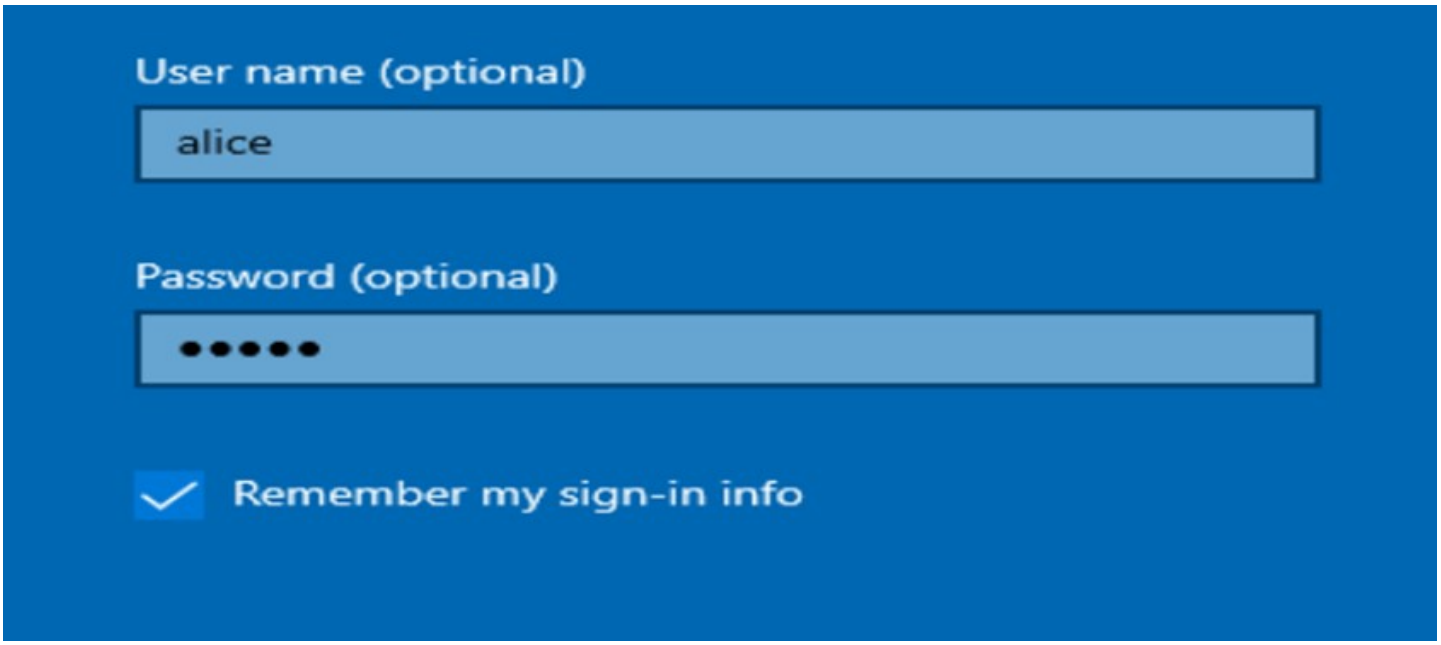
-Add a new VPN connection

-Go to Settings > VPN > Add a VPN connection

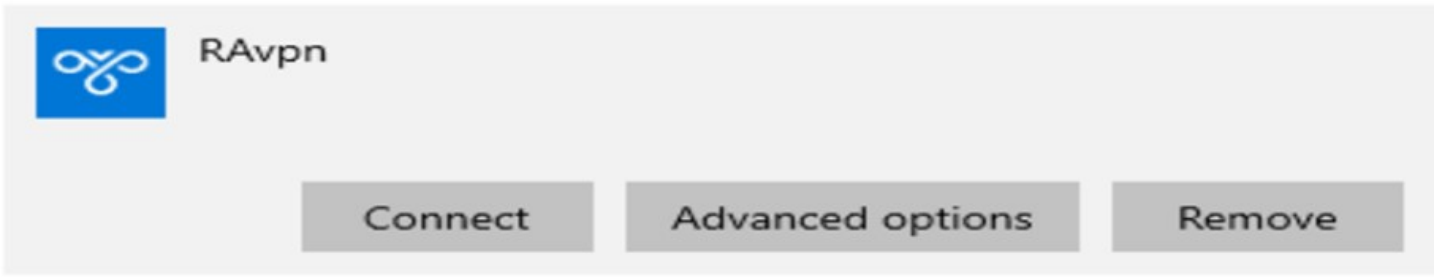


-Fill out the fields





-Click on connect on the new VPN connection



-If everything works you should be able to ping stuff in 192.168.100.0/24

```
C:\Users\LocalAdmin>ping 192.168.100.100

Pinging 192.168.100.100 with 32 bytes of data:
Reply from 192.168.100.100: bytes=32 time=1ms TTL=64
Reply from 192.168.100.100: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.100.100:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

```
bob@debClient:~$ ping 192.168.100.100
PING 192.168.100.100 (192.168.100.100) 56(84) bytes of data.
64 bytes from 192.168.100.100: icmp_seq=1 ttl=64 time=1.24 ms
64 bytes from 192.168.100.100: icmp_seq=2 ttl=64 time=0.808 ms
64 bytes from 192.168.100.100: icmp_seq=3 ttl=64 time=0.712 ms
64 bytes from 192.168.100.100: icmp_seq=4 ttl=64 time=0.966 ms
^C
--- 192.168.100.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3008ms
rtt min/avg/max/mdev = 0.712/0.930/1.237/0.198 ms
bob@debClient:~$
```