Configuration of WinRM

In this tutorial we will go through configuration of WinRM which is necessary for using WinRM connector It will cover configuration which we tested on multiple servers together with our connector. It cover just the basic stuff and if you want to study more about this topic you can use official documentation or 3rd party tutorials which will go deeper.

WinRM or Windows remote management, is a remote management protocol that uses Simple Object Access Protocol to interface with remote computers and servers, as well as Operating Systems and applications. WinRM is a command-line tool.

Check if Winrm is running

```
Test-WSMan
```

The output should be following: 

If you get some error then you need to do the quick default configuration

```
winrm quickconfig
```

Now execute the first command again and it should without error now.

Show current configuration

Display WinRM listener. It will show useful information about port, address, ... where WinRM is listening for incoming connections. After quick config you will probably see only one listener for HTTP.

```
winrm e winrm/config/listener
```

```
PS C:\Users\Administrator> winrm e winrm/config/listener
Listener
    Address = *
    Transport = HTTP
    Port = 5985
    Hostname
    Enabled = true
    URLPrefix = wsman
    CertificateThumbprint
    ListeningOn = 127.0.0.1, 172.31.255.181, ::1, fe80::5efe:172.31.255.181%13

Listener
    Address = *
    Transport = HTTPS
    Port = 5986
    Hostname = adradic2
    Enabled = true
    URLPrefix = wsman
    CertificateThumbprint = 906a790dac5d33c271cff8028f692354ce368028
    ListeningOn = 127.0.0.1, 172.31.255.181, ::1, fe80::5efe:172.31.255.181%13
```

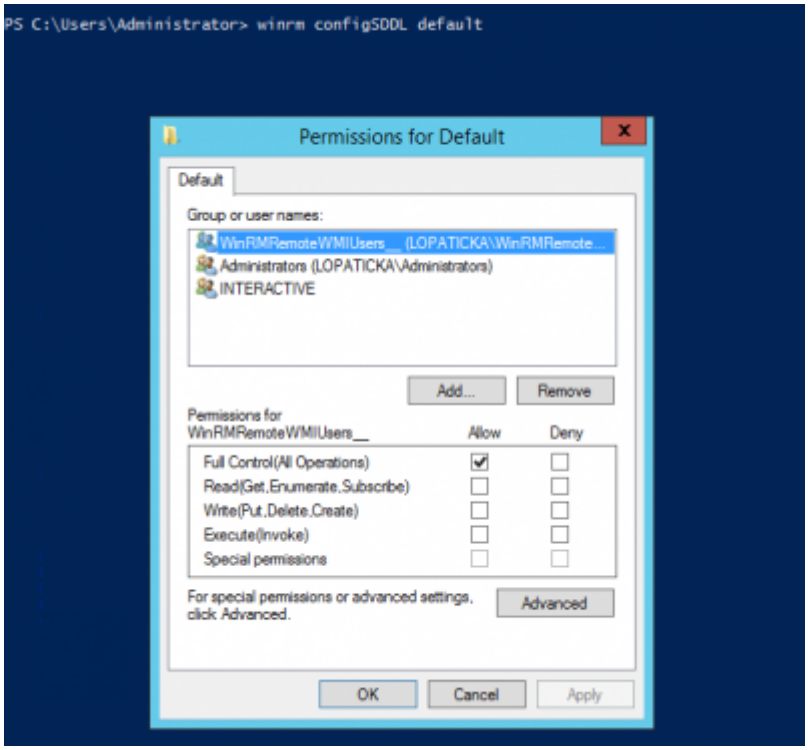
Display current winrm configuration

```
winrm get winrm/config
```

```
PS C:\Users\Administrator> winrm get winrm/config
Config
    MaxEnvelopeSizekb = 500
    MaxTimeoutms = 60000
    MaxBatchItems = 32000
    MaxProviderRequests = 4294967295
    Client
        NetworkDelayms = 5000
        URLPrefix = wsman
        AllowUnencrypted = false
        Auth
            Basic = true
            Digest = true
            Kerberos = true
            Negotiate = true
            Certificate = true
            CredSSP = false
        DefaultPorts
            HTTP = 5985
            HTTPS = 5986
        TrustedHosts
    Service
        RootSDDL = O:NSG:BAD:P(A;;GA;;;BA)(A;;GR;;;IU)(A;;GA;;;S-1-5-21-643154377-3231067919-152725009-1000)S:P(AU;FA;GA
;;;WD)(AU;SA;GXGW;;;WD)
        MaxConcurrentOperations = 4294967295
        MaxConcurrentOperationsPerUser = 1500
        EnumerationTimeoutms = 240000
        MaxConnections = 300
        MaxPacketRetrievalTimeSeconds = 120
        AllowUnencrypted = false
        Auth
            Basic = false
            Kerberos = true
            Negotiate = true
            Certificate = false
            CredSSP = true
            CbtHardeningLevel = Relaxed
        DefaultPorts
            HTTP = 5985
            HTTPS = 5986
        IPv4Filter = *
        IPv6Filter = *
        EnableCompatibilityHttpListener = false
        EnableCompatibilityHttpsListener = false
        CertificateThumbprint
        AllowRemoteAccess = true
    Winrs
        AllowRemoteShellAccess = true
        IdleTimeout = 7200000
        MaxConcurrentUsers = 10
        MaxShellRunTime = 2147483647
        MaxProcessesPerShell = 25
        MaxMemoryPerShellMB = 1024
        MaxShellsPerUser = 30
```

Show SDDL setting, this command will show dialog window

```
winrm configSDDL default
```



Authentications methods

	Type of user	Credential delegation	Message encryption
Basic	local	no	no
NTLM	local, domain	no	yes
Kerberos	domain	yes	yes
CredSSP	local, domain	yes	yes

You can configure trusted host which will be able to connect. If you don't want to specify this use

```
winrm set winrm/config/client '@{TrustedHosts="*"}'
```

We can use several methods for authentication.

- Basic - the second command will allow unencrypted data transfer, so it's not recommended to use it with HTTP. For some testing purpose it's ok.

```
winrm set winrm/config/service/auth '@{Basic="true"}'
winrm set winrm/config/service '@{AllowUnencrypted="true"}'
```

- NTLM

```
winrm set winrm/config/service/auth '@{Negotiate="true"}'
```

- Kerberos

```
winrm set winrm/config/service/auth '@{Kerberos="true"}'
```

- CredSSP - HTTPS must be enabled - see steps below: HTTPS setting

```
winrm set winrm/config/service/auth '@{CredSSP="true"}'
```

```
winrm set winrm/config/client/auth '@{CredSSP="true"}'
Enable-WSManCredSSP -Role Server
```

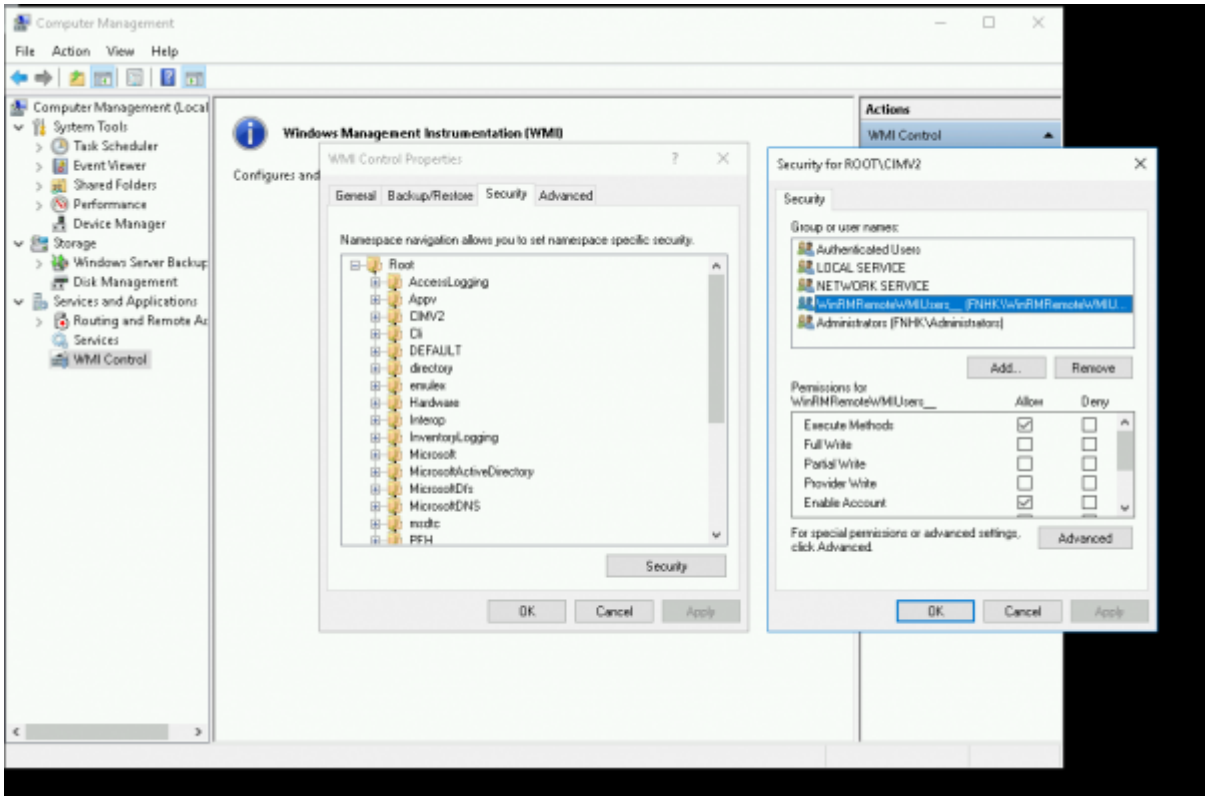
Permission configuration

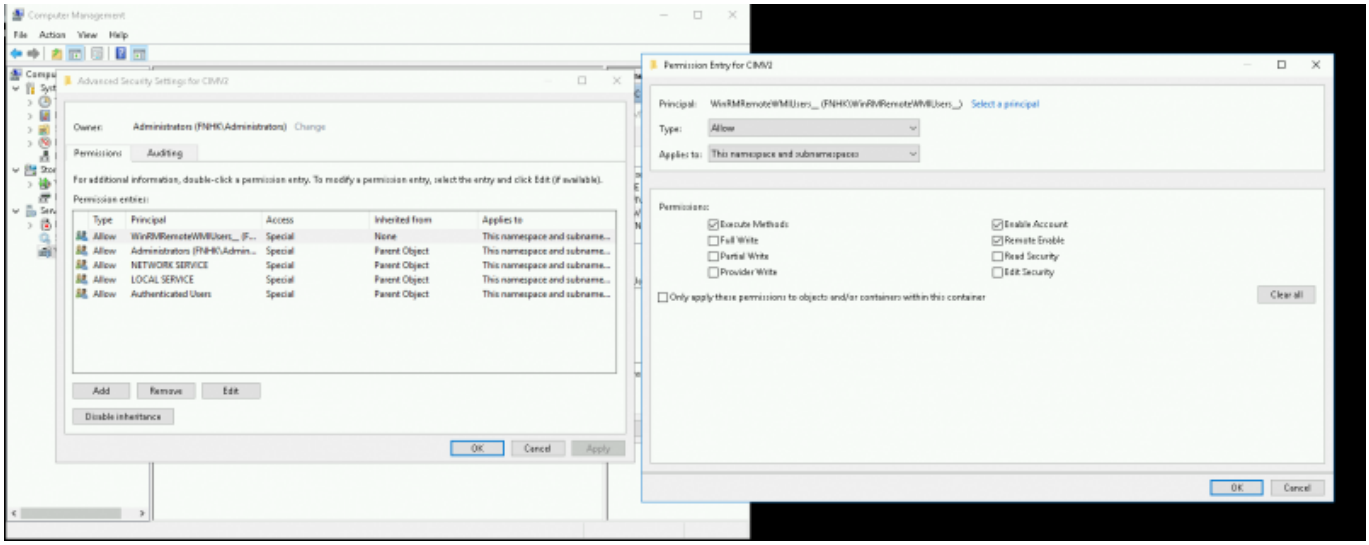
If you want to use user which is not admin then we need to do a more configuration. If you want to use admin user you should be ready to go even without it.

Now we need to set the right permissions. It's tested against NTLM, Kerberos and CredSSP auth It's tested with local user + group and with domain user + group. For the following steps you can use one of these groups WinRMRemoteWMIUsers__ or Remote Management Users It should work with both.

Assign user into group Set WMI access for group.

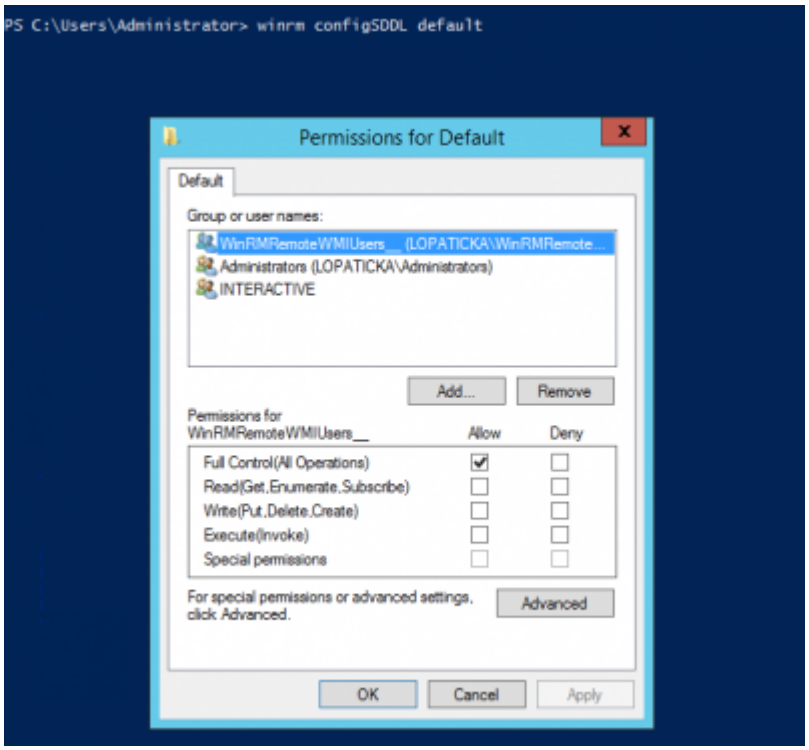
- Computer Management → Services and Application → right click WMI Control → Properties
- In new dialog window → tab Security → Root → CIMV2 and click button Security
- Next dialog window will appear - you need to add group (WinRMRemoteWMIUsers__ or Remote Management Users) here
- You need to select these options in the checkboxes - Execute Methods, Enable Account and Remote Enable
- Click on Advanced - select and edit group → Set "Applies to" This namespace and subnamespaces
- Confirm all changes in dialog windows and close them





Set SDDL

```
winrm configSDDL default
```



- Add group (WinRMRemoteWMIUsers__ or Remote Management Users) and give it Full Control
- Confirm changes Restart WinRM

```
Restart-Service winrm
```

Debugging

When you need to check if WinRM is ready for connection but you don't have access to the Windows server to check the configuration yourself use this tips.

Check if port is open and ready to connection, default ports are 5985 (HTTP) and 5986 (HTTPS): Linux

```
nc -vz HOST PORT
```

Windows

```
Test-WSMan -ComputerName HOST or Test-netConnection HOST -Port PORT
```

Now we know if we are able to connect to the WinRM port. In case the port is not accessible it can be probably blocked in firewall. Next we want to try to connect to WinRM. Install pywinrm follow only the first part of installation, we don't need to install connector server. Open terminal (Linux) or powershell (Windows)

```
> python>>> import winrm
>>> s = winrm.Session('[[http://HOST:5985/wsman|http://HOST:5985/wsman]]',
auth=('USER', 'PASS'), transport='ntlm')
>>> r = s.run_ps('Write-Host connection test OK')
>>> r
```

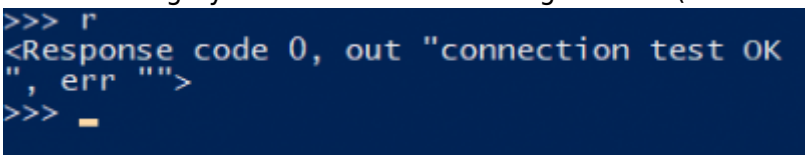
For connecting via HTTPS use this lane. The difference is in URL where we need to use https and port 5986. Then we are using one more argument where we specify path to trust store

```
>>> s = winrm.Session('[[https://HOST:5986/wsman|https://HOST:5986/wsman]]',
auth=(HOST, PASS), transport='ntlm', ca_trust_path='/etc/ssl/certs/CRT.pem')
```

Then, execute the winrm call. Followin call simply instructs the remote powershell to echo "connection test OK". If there some errors or warnings during the call, the python REPL will display them.

```
r = s.run_ps('Write-Host connection test OK')
```

The fact that there were some stacktraces printed does not necessarily mean the call failed. Now simply print the result by calling `r`. After executing `r`you should see something like this (note the "connection test OK" string is there):



Common issues

Specified credentials were rejected by the server

Can be caused by:

- wrong username or password
- user is not in correct user group on the Windows system

```
>>> r = s.run_ps('Write-Host connection test OK')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Python27\lib\site-packages\winrm\__init__.py", line 50, in run_ps
    rs = self.run_cmd('powershell -encodedcommand {0}'.format(encoded_ps))
  File "C:\Python27\lib\site-packages\winrm\__init__.py", line 37, in run_cmd
    shell_id = self.protocol.open_shell()
  File "C:\Python27\lib\site-packages\winrm\protocol.py", line 157, in open_shell
    res = self.send_message(xmltodict.unparse(req))
  File "C:\Python27\lib\site-packages\winrm\protocol.py", line 234, in send_message
    resp = self.transport.send_message(message)
  File "C:\Python27\lib\site-packages\winrm\transport.py", line 243, in send_message
    self.build_session()
  File "C:\Python27\lib\site-packages\winrm\transport.py", line 232, in build_session
    self.setup_encryption()
  File "C:\Python27\lib\site-packages\winrm\transport.py", line 238, in setup_encryption
    self._send_message_request(prepared_request, '')
  File "C:\Python27\lib\site-packages\winrm\transport.py", line 266, in _send_message_request
    raise InvalidCredentialsError("the specified credentials were rejected by the server")
winrm.exceptions.InvalidCredentialsError: the specified credentials were rejected by the server
```

Access denied 500

Can be caused by:

- wrong username or password
- WinRM SDDL is not configured

CredSSP handshake error

If you get this error when you trying to use CredSSP over HTTPS connection, the problem can be that there is configured certificate thumbprint directly in `winrm/config/service`.

```
class 'requests_credssp.exceptions.AuthenticationException'>("Server did not response with a CredSSP token after step Step 1. TLS Handshake - actual",)
```

Execute this command to delete `CertificateThumbprint` value from the `config/service`.

```
winrm set winrm/config/service '@{CertificateThumbprint=""}'
```

The configuration of certificate thumbprint in the Listener should remain there.

CredSSP Delegate credentials error

If you get this error when you are trying to use CredSSP over HTTPS connection. the problem can be that the server with WinRM has credential delegation turned off.

```
<class 'requests_credssp.exceptions.AuthenticationException'>("Server did not response with a CredSSP token after step Step 5. Delegate Credentials - actual",)
```

To turn the credentials delegation on. Open Group policy setting and navigate to Computer Configuration\Administrative Templates\System\Credentials Delegation.

The Allow Delegating Fresh Credentials (AllowFreshCredentials) policy setting must be enabled. If it's enabled validate if correct value (values) are added to this policy. The correct value is WSMAN/SPN of your server. For example

```
WSMAN/myComputer.myDomain.com
WSMAN/*.myDomain.com
```

You need to restart the computer after that.

x509 attribute parsing error

When calling WinRM over HTTPS, you can encounter following error:

```
Traceback (most recent call last):
  File "/usr/lib/python2.7/site-packages/OpenSSL/SSL.py", line 309, in
wrapper
  _lib.X509_up_ref(x509)
AttributeError: 'module' object has no attribute 'X509_up_ref'
```

This seems to be caused by older versions of the `cryptography` python library. Upgrading the library should solve the problem. Since this library is also used by some OS components, we recommend to upgrade it locally only for the user who runs python winrm scripts.

Requests using non-urllib3 backend

Please note this is **not** a fix to your situation. For more info, look at this Github issue.

This affects only `requests-ntlm` library and therefore only NTLM authentication. It does not seem to affect the overall function but the warning is at least an annoyance. When you see the warning:

```
/usr/lib/python2.7/site-packages/requests_ntlm/requests_ntlm.py:200:
NoCertificateRetrievedWarning: Requests is running with a non urllib3
backend, cannot retrieve server certificate for CBT
NoCertificateRetrievedWarning)
```

You can confirm the behavior by:

1. Installing `requests-ntlm` locally for the user.
2. Editing `~/.local/lib/python2.7/site-packages/requests_ntlm/requests_ntlm.py` and changing the import `from requests.packages.urllib3.response import HTTPResponse` to `from requests.packages.urllib3 import HTTPResponse`.
3. When running winrm script with NTLM, the warning should no longer pop up.

HTTPS certificate not trusted

Python, by default, uses its own certificate truststore located somewhere under `/usr/lib/python2.7/…`. If it cannot find it, it uses system-wide truststore provided by `ca-certificates`. However, you usually do not want to trust so many authorities. Also, your server usually have your certificates and that means you have to add your CA to the truststore. For debugging this (and WinRM at all) you can also use following script:

```
import os
# there, you can explicitly set path to your CA chain
# DO NOT put there server's certificate itself
os.environ["REQUESTS_CA_BUNDLE"_] = "/path/to/crt/chain.pem"
from winrm.protocol import Protocol
p = Protocol(
             endpoint='https://SERVER YOU WANT TO CONNECT TO:5986/wsman',
             transport='CHOOSE AUTHENTICATION METHOD:
basic,credssp,ntlm,kerberos',
             username='USERNAME OR USERNAME@DOMAIN', p
             assword='USER PASSWORD')
#server_cert_validation='ignore')
# put this into the Protocol object constructor to disable certificate
validation
shell_id = p.open_shell()
command_id = p.run_command(shell_id, 'ping', ['1.1.1.1'])
std_out, std_err, status_code = p.get_command_output(shell_id, command_id)
p.cleanup_command(shell_id, command_id)
p.close_shell(shell_id)
# this will output all that returned from the WinRM call
print "stdout",std_out
print "stderr",std_err
print "retcode",status_code
```

SDDL configuration - access denied

When you try to configure SDDL via command "winrm configSDDL default", after adding some group and clicking on "OK", you will see this error in command line:

```
access denied
Error number: -2147024891 0x80070005
```

This can be caused, because your user has no permission to change it.

For example if only local group "Administrators" had "full control" but for some reason someone remove it, you are not able to add the same group back or any other group back. The only solution is to edit registry.

Navigate to ComputerHkey_Local_MachineSoftwareMicrosoftWindowsCurrentVersionWSMANService

Set value for rootSDDL to

O:NSG:BAD:P(A;;GA;;;BA)(A;;GR;;;IU)(A;;GA;;;RM)S:P(AU;FA;GA;;;WD)(AU;SA;GXGW;;;WD)

After that when you open SDDL config "Administrators" group will be back again with full control permissions.

HTTPS support

The best case is to use HTTPS connection to connect to WinRM. To achieve this we need to do some more configuration on the server and on the client. We need to create HTTPS listener and for this we will need some certificate. In this tutorial we will cover setting up WinRM with self signed certificate. The configuration will be same if we want to use some other certificate, so if you already have certificate you can skip the part where we are generating one.

The tested way to generate self signed certificate on linux via tutorial which can be found here you should follow whole process except the part with finals steps because for our purpose we don't need to import it to browsers.

Now we have certificate which is imported in our windows server and now we can configure the HTTP listener

Create and export self signed certificate with powershell:

```
$pathToCertificate="C:\Users\Administrator.ZOO\Desktop\certificate"  ## Specify your preferred location for export
$hostname='ad.idstory.idm' #hostname your machine
$params = @{
    Subject = "CN=winrm.$hostname"
    DnsName = 'ad.idstory.idm'
    CertStoreLocation = 'Cert:\LocalMachine\My' #Certificate for WinRM, must be in stored in Local Computers
    KeyExportPolicy ='Exportable'
    KeySpec ='Signature'
    KeyLength ='2048'
    KeyAlgorithm = 'RSA'
    HashAlgorithm = 'SHA256'
}

$cert = New-SelfSignedCertificate @params
Export-Certificate -Cert $cert -FilePath "$pathToCertificate\$hostname.cer"
$mypwd = ConvertTo-SecureString -String "{myPassword}" -Force -AsPlainText
## Replace {myPassword}
Export-PfxCertificate -Cert $cert -FilePath "$pathToCertificate\$hostname.pfx" -Password $mypwd
```

List certificate in windows certificate storage:

```
Get-ChildItem -Path Cert:\LocalMachine\My -Recurse #List certificate stored in Local Computer, and copy certificate thumbrint
```

Configure WinRM listener with HTTPS certficate:

```
winrm create winrm/config/Listener?Address=*+Transport=HTTPS
'@{Hostname="HOSTNAME";CertificateThumbprint="THUMBPRINT"}'
for deleting
winrm delete winrm/config/Listener?Address=*+Transport=HTTPS
```

Create firewall rule for WinRM HTTPS:

```
New-NetFirewallRule -Displayname 'WinRM - Powershell remoting HTTPS-In' -
Name 'WinRM - Powershell remoting HTTPS-In' -Profile Any -LocalPort 5986 -
Protocol TCP
```

Restart WinRM

```
Restart-Service winrm
```

Next step is to validate if we can connect to HTTPS listener so follow instruction in section debug and validate if HTTPS port is accessible. Before we try to execute some powershell command via WinRM we need to import this certificate into client trust store and pass the path to this store as parameter - see debug section

Powershell 7 support

Install powershell 7:
https://learn.microsoft.com/en-us/powershell/scripting/install/installing-powershell-on-windows?view=powershell-7.4#installing-the-msi-package

Run app `C:\Program Files\PowerShell\7\pwsh.exe` and execute

```
Enable-PSRemoting
```