Remotely Managing Hyper-V in a Workgroup Environment

blog.ropnop.com/remotely-managing-hyper-v-in-a-workgroup-environment

Background

A few weekends ago, I decided (because apparently I'm a masochist) that I was tired of the free version of ESXi running my home lab and decided to give Microsoft's Hyper-V Server 2016 a try.

I liked the idea that it was light weight, accessible with PowerShell, and there were lots of cool Github projects that made deploying test AD domains automatic.

I had played around with Hyper-V on my Windows laptop, and liked the GUI Hyper-V manager as well. If I could remotely manage my lab with that and PowerShell, I'd be happy.

Installing Hyper-V server from the ISO was easy, but little did I know I'd burn several days just trying to connect to it.

I followed the steps in these docs, but still never got it working:

- https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/manage/remotely-manage-hyper-v-hosts
- http://pc-addicts.com/remotely-manage-hyper-v-server-2012-core/

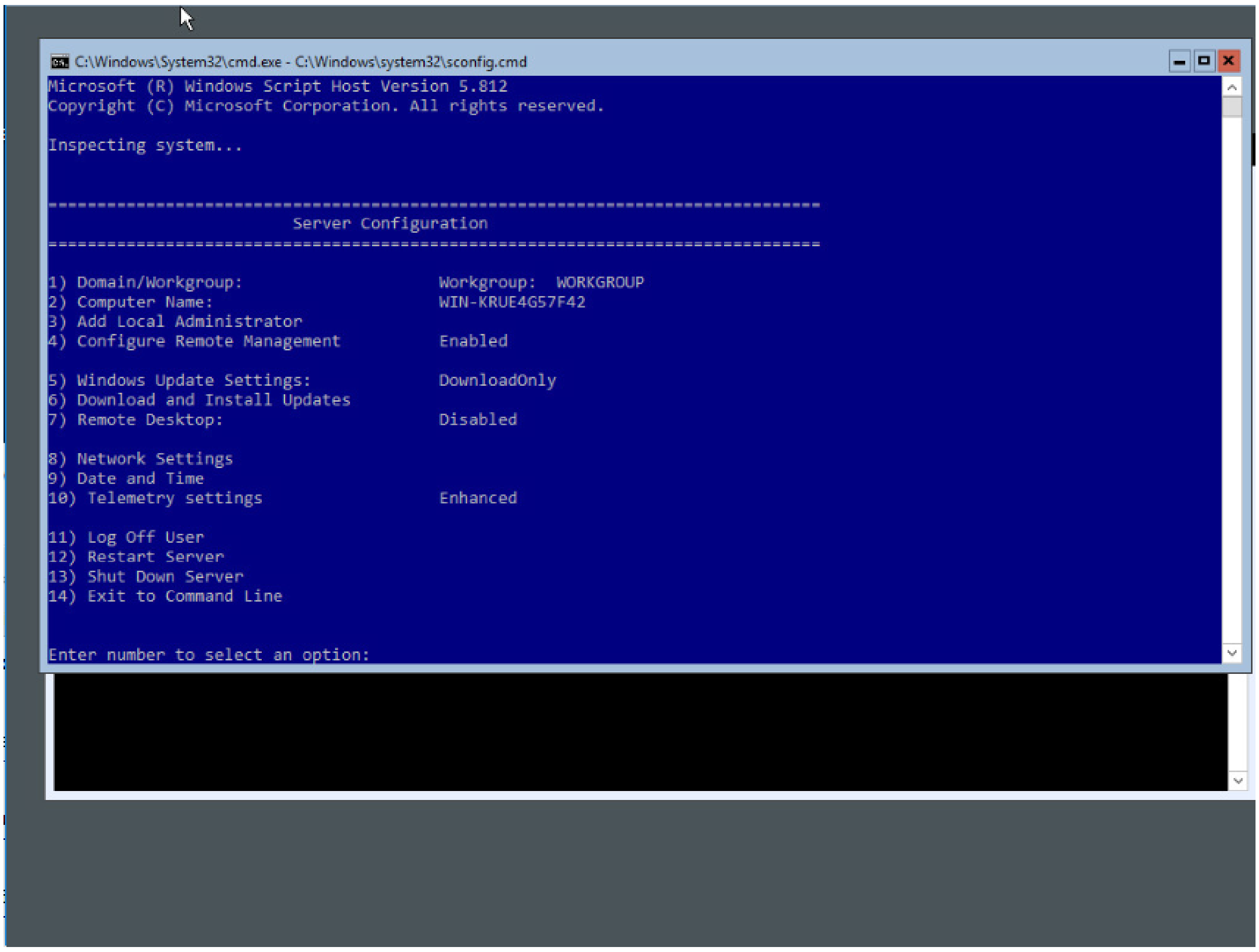
The crux of my problem is that Hyper-V Server is really intended to be run in an AD Domain environment. Remotely managing via PowerShell or the Hyper-V Manager I'm sure works seamlessly if AD is behind it handling authentication and authorization.

I don't have an AD Domain set up in my apartment, and had little desire to spin up a DC and manage a domain for personal use. So I set about trying to configure Hyper-V to play nice with a WORKGROUP. What I wanted to do was be able to remotely connect to and manage the server from my main Windows workstation. I also wanted to do the bare minimum amount of tweaking to my main workstation (i.e. not doing anything blatantly insecure like disabling firewalls, trusting wildcards, etc)

Below are the commands I found that work to get remote connections working.

Configuring the Server

After initial installation, and setting an Administrator password, you're presented with this minimalist screen:



```
C:\Windows\System32\cmd.exe - C:\Windows\system32\sconfig.cmd
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

Inspecting system...


===============================================================================
                         Server Configuration
===============================================================================

1) Domain/Workgroup:                    Workgroup:  WORKGROUP
2) Computer Name:                       WIN-KRUE4G57F42
3) Add Local Administrator
4) Configure Remote Management          Enabled

5) Windows Update Settings:             DownloadOnly
6) Download and Install Updates
7) Remote Desktop:                      Disabled

8) Network Settings
9) Date and Time
10) Telemetry settings                  Enhanced

11) Log Off User
12) Restart Server
13) Shut Down Server
14) Exit to Command Line


Enter number to select an option:
```

There’s a few things that need to be configured from here.

1. In “Configure Remote Management”, make sure Remote Management is enabled and change the setting to allow the server to be pinged
2. Enable Remote Desktop so you can remotely manage the server over RDP
3. Configuring a name. I chose `hyperv-server`. This requires a reboot.

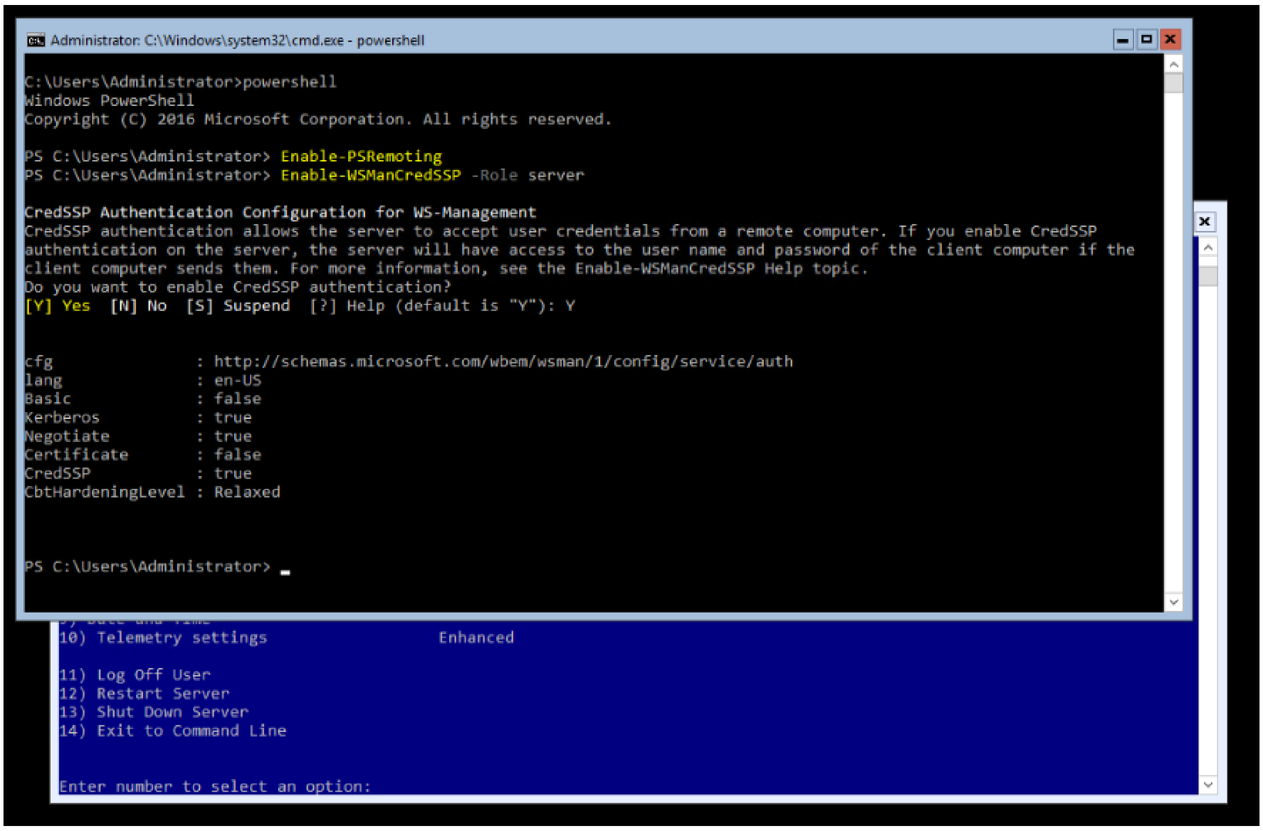
I also created a host entry for `hyperv-server.home.local` in my local DNS server so I could access it by hostname. Manually editing the `etc\hosts` file would work as well.

After these settings, I was able to RDP into the box by its hostname and using the credentials for `HYPERV-SERVER\Administrator`.

Server PowerShell Configuration

Behind the configuration menu is a command prompt, which lets you drop into PowerShell. From here, you need to configure the server to allow PS-Remoting and to allow CredSSP authentication to the server. Use the following commands:

```
Enable-PSRemoting
Enable-WSManCredSSP -Role
server
```



```
C:\Users\Administrator>powershell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> Enable-PSRemoting
PS C:\Users\Administrator> Enable-WSManCredSSP -Role server

CredSSP Authentication Configuration for WS-Management
CredSSP authentication allows the server to accept user credentials from a remote computer. If you enable CredSSP
authentication on the server, the server will have access to the user name and password of the client computer if the
client computer sends them. For more information, see the Enable-WSManCredSSP Help topic.
Do you want to enable CredSSP authentication?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): Y


cfg               : http://schemas.microsoft.com/wbem/wsman/1/config/service/auth
lang              : en-US
Basic             : false
Kerberos          : true
Negotiate         : true
Certificate       : false
CredSSP           : true
CbtHardeningLevel : Relaxed



PS C:\Users\Administrator> _
```

```
10) Telemetry settings                  Enhanced

11) Log Off User
12) Restart Server
13) Shut Down Server
14) Exit to Command Line


Enter number to select an option:
```

Once that's enabled, you're done configuring the server. Now came the tricky part of getting my client to communicate with it.

Client Configuration

I would be remotely managing the Hyper-V Server from my Windows 10 laptop. The first thing is to make sure that the connection profile is set to "Private" so that Windows Firewall allows the traffic (i.e. it won't work over a public network)

```
Get-NetAdapter|Get-NetConnectionProfile

Set-NetConnectionProfile -InterfaceAlias Ethernet -NetworkCategory
Private
```

The next step is to configure the client that it's okay to connect to the server by adding the server as a trusted host for the WS-Management Protocol.

To add the entry, you have to enable WinRM on the client. I did not want to enable WinRM on my laptop, but fortunately I discovered that stopping the service after adding the entry works. To add the entry:

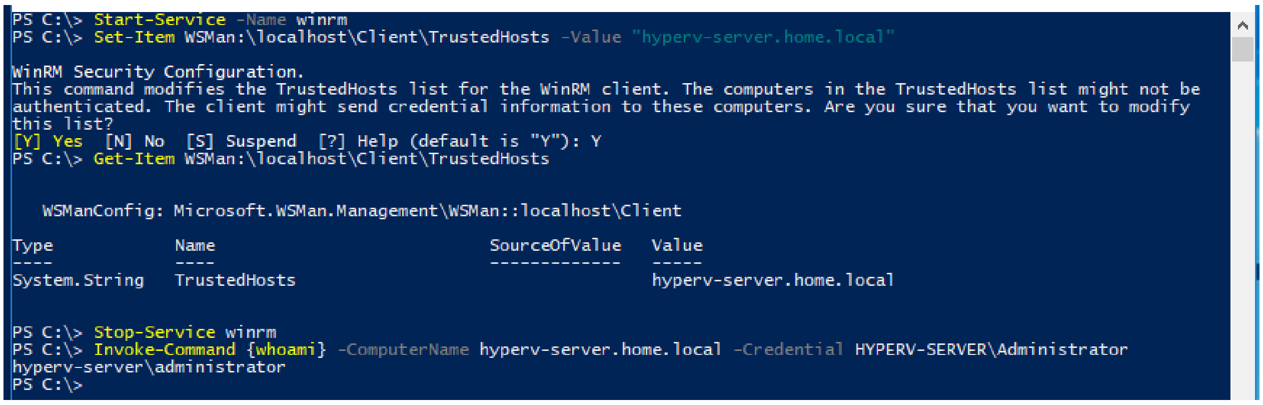
```
$ Start-Service -Name winrm
$ Set-Item WSMan:\localhost\Client\TrustedHosts -Value "hyperv-
server.home.local
$ Stop-Service -Name winrm
```

This should work with an IP address instead of hostname, but I didn’t test. YMMV

Now test out that you can authenticate to and remotely run commands from your workstation by testing `Invoke-Command`:

```
Invoke-Command {whoami} -ComputerName hyperv-server.home.local -Credential
HYPERV-SERVER\Administrator
```

It will prompt you for the local Administrator password and then remotely execute the `hostname` command:

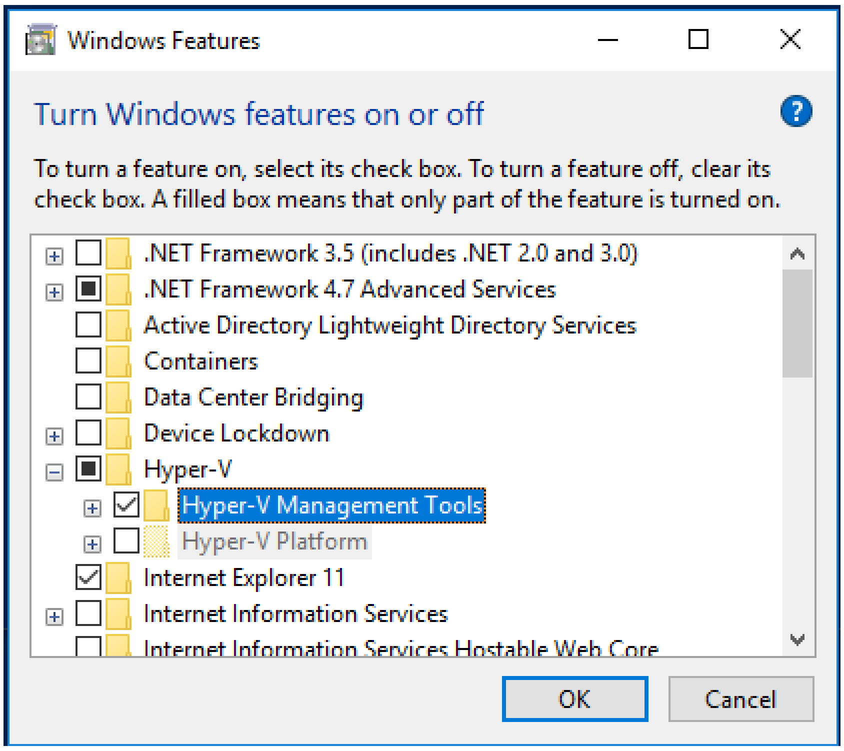


```
PS C:\> Start-Service -Name winrm
PS C:\> Set-Item WSMan:\localhost\Client\TrustedHosts -Value "hyperv-server.home.local"

WinRM Security Configuration.
This command modifies the TrustedHosts list for the WinRM client. The computers in the TrustedHosts list might not be
authenticated. The client might send credential information to these computers. Are you sure that you want to modify
this list?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): Y
PS C:\> Get-Item WSMan:\localhost\Client\TrustedHosts


   WSManConfig: Microsoft.WSMan.Management\WSMan::localhost\Client

Type            Name                           SourceOfValue   Value
----            ----                           -------------   -----
System.String   TrustedHosts                                   hyperv-server.home.local


PS C:\> Stop-Service winrm
PS C:\> Invoke-Command {whoami} -ComputerName hyperv-server.home.local -Credential HYPERV-SERVER\Administrator
hyperv-server\administrator
PS C:\>
```

If this works, you can now securely communicate between client and server and can finally start remotely managing Hyper-V.

Configuring Hyper-V Manager

If you haven’t already, make sure to enable `Hyper-V Management Tools` in Windows Features to get the Hyper-V Manager GUI. You don’t have to actually enable the Hyper-V Platform (I don’t since I run Virtualbox locally).

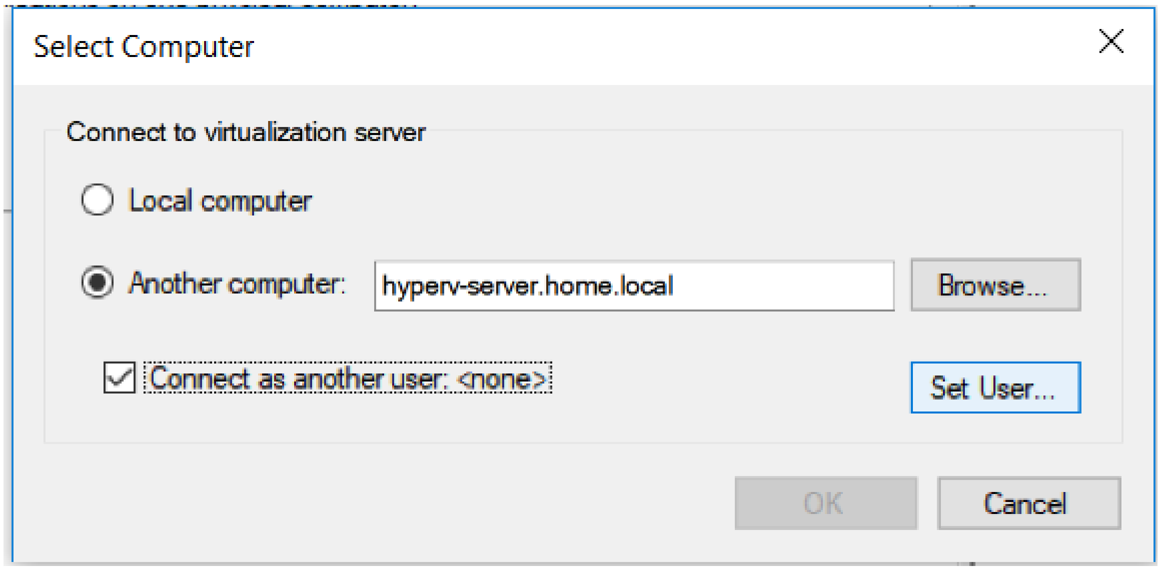

Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ☐ .NET Framework 3.5 (includes .NET 2.0 and 3.0)
- ■ .NET Framework 4.7 Advanced Services
- ☐ Active Directory Lightweight Directory Services
- ☐ Containers
- ☐ Data Center Bridging
- ☐ Device Lockdown
- ■ Hyper-V
  - ✓ Hyper-V Management Tools
  - ☐ Hyper-V Platform
- ✓ Internet Explorer 11
- ☐ Internet Information Services
- ☐ Internet Information Services Hostable Web Core

Open up Hyper-V Manager, right click and select “Connect to Server”. Enter the IP address or hostname of the Hyper-V server, and choose “Connect as another user”, entering the local Administrator credentials

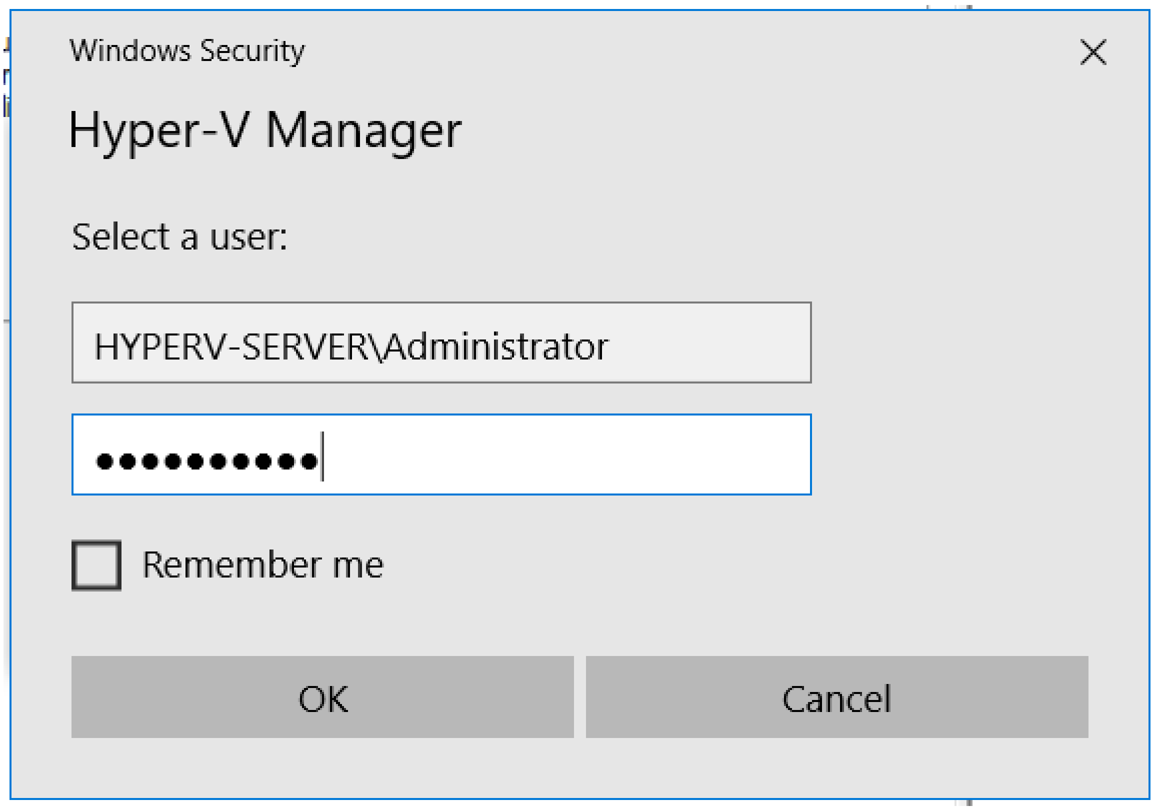

Select Computer

Connect to virtualization server

( ) Local computer

(●) Another computer: hyperv-server.home.local

✓ Connect as another user: <none>


Windows Security

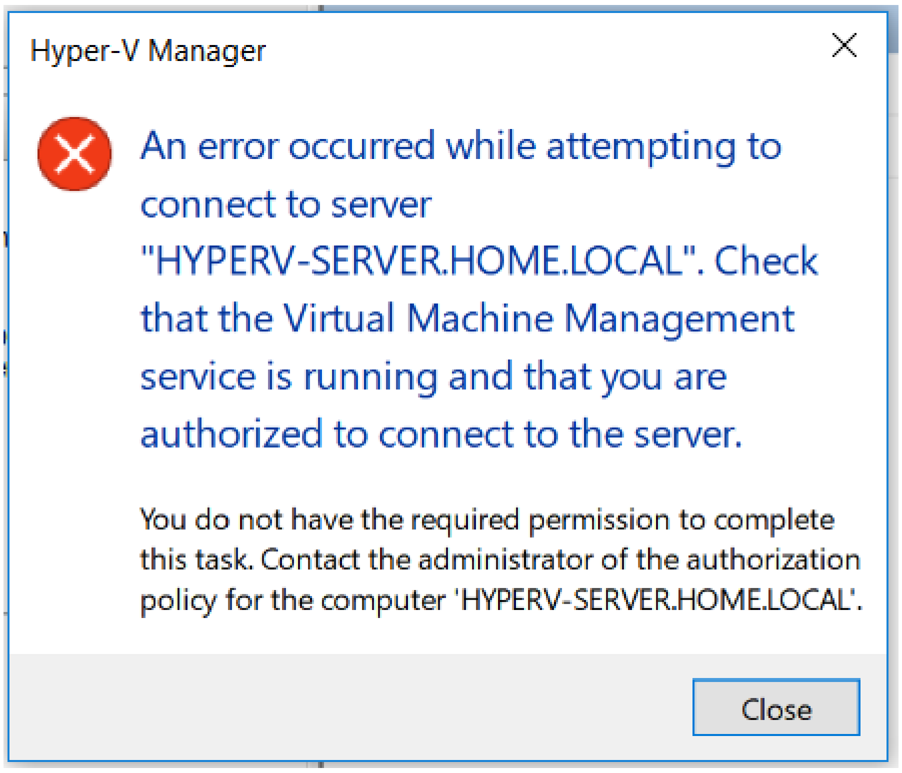
Hyper-V Manager

Select a user:

HYPERV-SERVER\Administrator

●●●●●●●●●●●

Now…you'd think this would totally work. Especially since we've already proved that WinRM and PS-Remoting work just fine. But hit 'OK' and this error pops up:


Hyper-V Manager

An error occurred while attempting to connect to server "HYPERV-SERVER.HOME.LOCAL". Check that the Virtual Machine Management service is running and that you are authorized to connect to the server.

You do not have the required permission to complete this task. Contact the administrator of the authorization policy for the computer 'HYPERV-SERVER.HOME.LOCAL'.

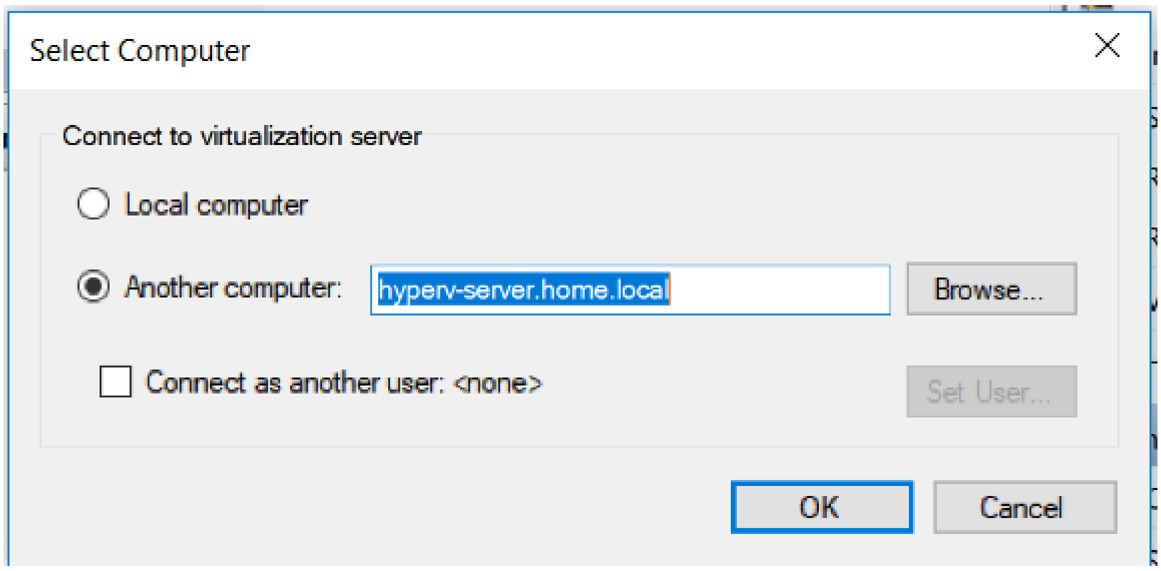
I have no idea why this is failing. This had me at a loss until I found a workaround.

Authorization Workaround

To get around the authorization error, you have to manually add the Hyper-V Server credential into memory using `cmdkey`. Exit out of Hyper-V server and enter the following at a command prompt on the client:

```
cmdkey /add:HYPERV-SERVER /user:Administrator /pass:
<REDACTED>
```

Now re-open Hyper-V Server and connect to the remote server again, but don't bother specifying another user since the creds we need are already in memory:


Select Computer

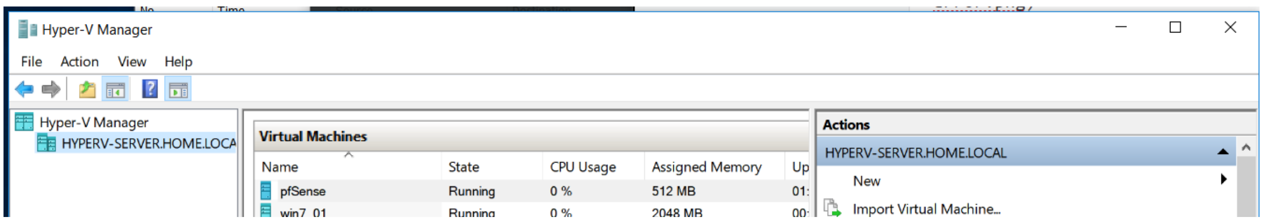
Connect to virtualization server

○ Local computer

● Another computer: hyperv-server.home.local

☐ Connect as another user: <none>

This finally works! You can connect and remotely manage the server:



Name	State	CPU Usage	Assigned Memory	Up
pfSense	Running	0 %	512 MB	01:
win7_01	Running	0 %	2048 MB	00:

As long as the credential remains in memory, you can remotely manage the server. Another possible workaround is to start Hyper-V Manager with `runas` and the correct local account:

```
runas /user:HYPERV-SERVER\Administrator /netonly "mmc
virtmgmt.msc"
```

You could create a shortcut to run this and be good to go.

tl;dr

On Hyper-V Server

- Enable Remote Management
- Enable Remote Desktop
- Set Computername
- Enable WSMan and PS-Remoting:

```
Enable-PSRemoting
Enable-WSManCredSSP -Role
server
```

On Managing Client

- Configure hosts or DNS entry for Hyper-V Server
- Ensure network profile is “Private”:

```
Set-NetConnectionProfile -InterfaceAlias Ethernet -NetworkCategory
Private
```

Temporarily start WinRM and add Hyper-V Server as Trusted Host:

```
Start-Service -Name winrm
Set-Item WSMan:\localhost\Client\TrustedHosts -Value "hyperv-
server.home.local"
Stop-Service -Name winrm
```

- Enable Hyper-V Management Tools in Windows Features
- Add Hyper-V Administrator credential with cmdkey:

```
cmdkey /add:HYPERV-SERVER /user:Administrator /pass:
<REDACTED>
```

Launch Hyper-V Manager and connect to server (don’t specify user)
or, launch Hyper-V Manager with `runas`:

```
runas /user:HYPERV-SERVER\Administrator /netonly "mmc
virtmgmt.msc"
```

Summary

After lots of mucking around, this is the bare-minimum working configuration I've found to remotely manage a Hyper-V server in a non-domain environment.

I'm still not 100% certain why Hyper-V Manager does not work with supplied credentials and the `cmdkey` or `runas` workaround is necessary. If anyone can shed some light on that, I would appreciate it!

Hope this helps someone out there -ropnop

See also

- ← Previous Post
- Next Post →