PowerShell SSH Client and Remoting Explained

lazyadmin.nl/powershell/powershell-ssh

July 15, 2021

Do you need to manage a remote server or network device? Then you are probably using SSH to connect and manage them. The SSH protocol allows you to connect securely to a remote device over an unsecured network (internet). To use SSH most people use an SSH client tool, like Putty. But did you know that **PowerShell has a built-in SSH Client**?

The PowerShell SSH client is enabled by default in Windows 10 1809 and higher. This means that you now easily can connect to any remote device from your favorite command-line tool.

We can also use SSH in PowerShell to manage another Windows machine remotely, but you will need to use PowerShell 6 or higher to run the SSH server.

PowerShell SSH Client

The most common way to use SSH in PowerShell is as an SSH Client. Assuming that you keep your Windows up-to-date you should have SSH enabled by default. You can simply check it by opening PowerShell and type the following command:

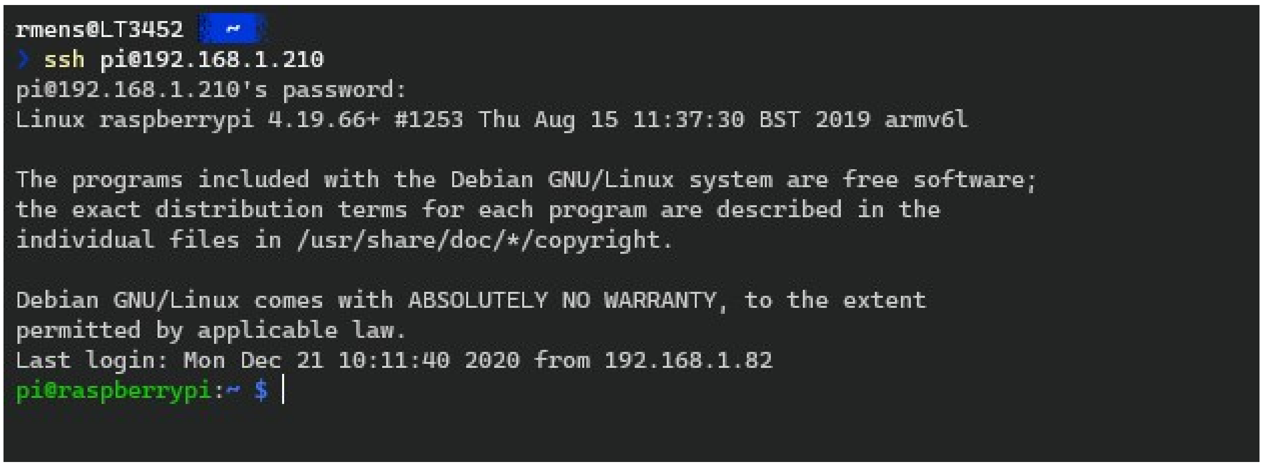
```
# type ssh and press enter
ssh
# Result:
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
[-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
[-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
[-i identity_file] [-J [user@]host[:port]] [-L address]
[-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
[-Q query_option] [-R address] [-S ctl_path] [-W host:port]
[-w local_tun[:remote_tun]] destination [command]
```

With PowerShell open you can connect to a remote server or network device with a single command:

```
ssh <username>@<host_ip_address>
# For example:
ssh pi@192.168.1.210
```

You can also use the hostname instead of the IP Address. If you don’t supply a username then your local user account will be used.

When you press enter your will need to enter the password of the SSH user. When you connect to a device for the first time you will need to accept the host’s key. Just press enter or type yes.



```
rmens@LT3452  ~
❯ ssh pi@192.168.1.210
pi@192.168.1.210's password:
Linux raspberrypi 4.19.66+ #1253 Thu Aug 15 11:37:30 BST 2019 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Dec 21 10:11:40 2020 from 192.168.1.82
pi@raspberrypi:~ $ |
```

If you have SSH running on a different port than the default port 22, then you can change the port number with the `-P` flag:

```
ssh <username>@<host_ip_address> -P <port_number>
# For example
ssh pi@192.168.1.210 -P 1022
```

PowerShell SSH Keygen

Some remote servers or services only allow SSH access with the use of an SSH key. You will need to generate a local SSH key and upload the key to the server (or services) so it can verify you and create a secure connection.

The SSH Key exists out of a private and public key. You will need to keep the private key on your local machine (and make sure that you keep it to yourself) and the public key is uploaded to the server.

PowerShell has a built-in SSH keygen that you can use to generate a new key. Type the following command to generate a key:

```
ssh-keygen -t Ed25519
# Result
Generating public/private Ed25519 key pair.
Enter file in which to save the key (C:\Users\lazyadmin/.ssh/id_ed25519): <press enter>
# It will ask for apassphrase, you can leave it empty - see below for more details
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
# Generated key
Your identification has been saved in C:\Users\lazyadmin/.ssh/id_ed25519.
Your public key has been saved in C:\Users\lazyadmin/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:1232131+ASdi123SscvBbwn9Qfxj0 lazyadmin@lab01
The key's randomart image is:
+--[ED25519 256]--+
| .o . |
|..o+ . |
|o..o . o |
|+..# o . |
|=.+ =...S. |
|.+.C.+. + E |
|. *o@OO O. . |
| = O*o . |
|. +++. |
+----[SHA256]-----+
```

The option -t Ed25519 is a newer algorithm that is used to create the key. On legacy systems, this may not be supported. You can then use the older RSA encryption system.

```
ssh-keygen -b 4096 -t RSA
```

The keys are stored in your user profile. The path is displayed in the output, but generally, they are saved in the following location:

```
%userprofile%\.ssh
```

SSH Keygen passphrase

During the creation of the ssh key, you will be asked to enter a passphrase or to leave it empty (default). The passphrase is used to encrypt the local key. This way you can securely store the local key. If anybody gets access to the key, then they can’t use it without the passphrase.

But before you use the passphrase there is one important thing you need to keep in mind. You will need to enter the passphrase every time you use the SSH key. Most of the time the SSH key is used in automation, like GitHub services for example. In those use-cases is the use of a passphrase is not an option.

PowerShell Remote Connection with SSH

We can also use SSH in PowerShell to establish a remote connection to another computer or server. This can be between Windows computers or Windows-Linux and vice versa.

Creating a PowerShell remoting session between two Windows machines was already possible with WinRM. But you can now also use SSH to create the connection. The advantage of SSH is that you can use it on Windows, Linux, and macOS.

You will need PowerShell 6 or higher to use SSH for remote connections between Windows and Linux machines. I am using PowerShell 7 in the examples below.

Install OpenSSH Server

The first step is to install the **OpenSSH server** on your Windows computer. As mentioned, the SSH client is now installed and enabled by default. But the SSH server isn't.

You can check if the OpenSSH server is installed with the cmdlet below:

```
Get-WindowsCapability -Online | ? Name -like 'OpenSSH*'
# Result:
Name : OpenSSH.Client~~~~0.0.1.0
State : Installed
Name : OpenSSH.Server~~~~0.0.1.0
State : NotPresent
```

As you can see is the OpenSSH server is not present. We can install/enable it with PowerShell:

```
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
```

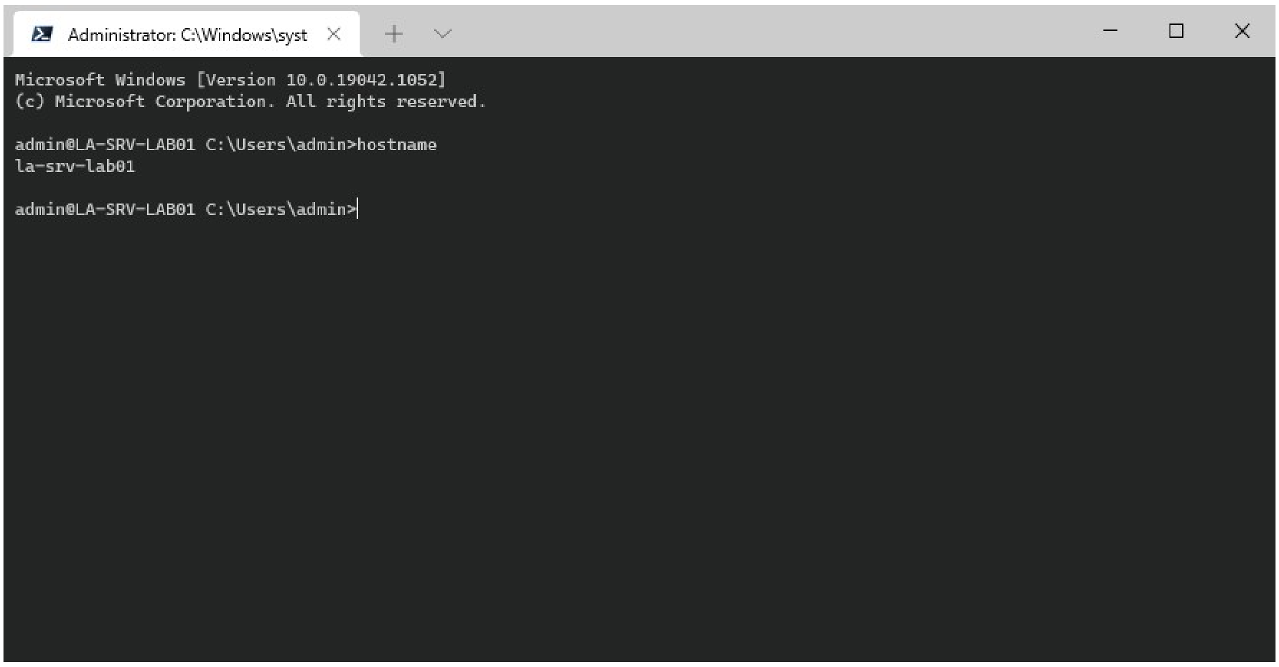
The next step is to start the OpenSSH server. Because it's a service we can set to start automatically:

```
# Start the sshd service
Start-Service sshd
# Set the service to automatic start
Set-Service sshd -StartupType Automatic
```

Configure the SSH Server

We need to edit the SSHD configuration file so we can authenticate with a password and use PowerShell on the remote machine.

If we don't configure the SSH server then we can only use basic commands on the remote machine (Windows Shell Prompt).



```
Microsoft Windows [Version 10.0.19042.1052]
(c) Microsoft Corporation. All rights reserved.

admin@LA-SRV-LAB01 C:\Users\admin>hostname
la-srv-lab01

admin@LA-SRV-LAB01 C:\Users\admin>
```

To fully use all PowerShell cmdlets we will need to configure the SSH Server to add PowerShell as a subsystem.

The SSHD configuration file is stored in the programdata folder. Because we already have PowerShell open, we will use it to open the configuration file:

# Change the directory to the programdata folder> ssh
cd $env:ProgramData\ssh
# Open the configuration file in notepad
PS C:\ProgramData\ssh> notepad .\sshd_config
In the configuration file **remove the #** before the line **PasswordAuthentication yes** and add the following line in the configuration file. *(the location doesn't really matter, but I added below the "# override default of no subsystems" part)*

Subsystem powershell c:/progra~1/powershell/7/pwsh.exe -sshs -NoLogo -NoProfile
Restart the SSHD service to load the new configuration file:

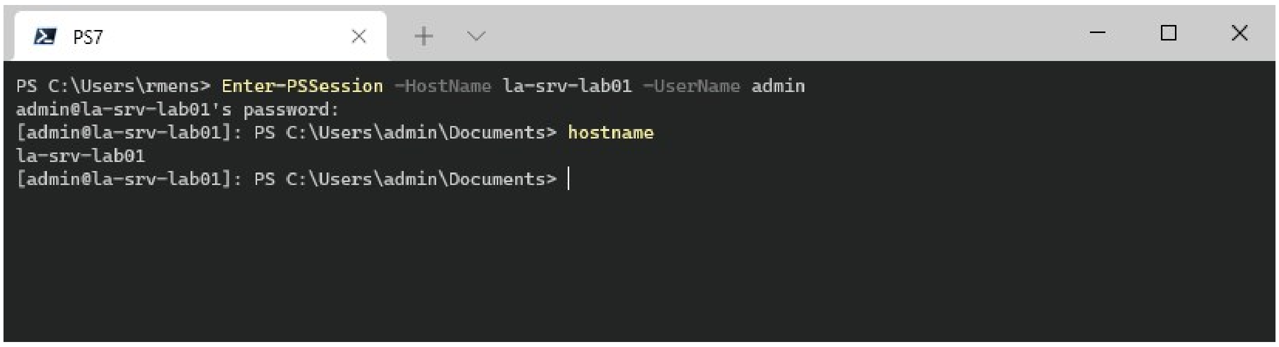
Restart-service sshd

SSH Remoting with PowerShell

With the SSH server installed on our server, we can now connect to it using PowerShell on our client. We can use the same cmdlet as with WinRM remoting, the difference is in the hostname vs computer name.

- New-PSSession
- Enter-PSSession
- Invoke-Command

If you use `New-PSSession -Computername` then the WinRM protocol will be used. When you use the parameter `-hostname` instead, then SSH will be used.



```
PS C:\Users\rmens> Enter-PSSession -HostName la-srv-lab01 -UserName admin
admin@la-srv-lab01's password:
[admin@la-srv-lab01]: PS C:\Users\admin\Documents> hostname
la-srv-lab01
[admin@la-srv-lab01]: PS C:\Users\admin\Documents> 
```

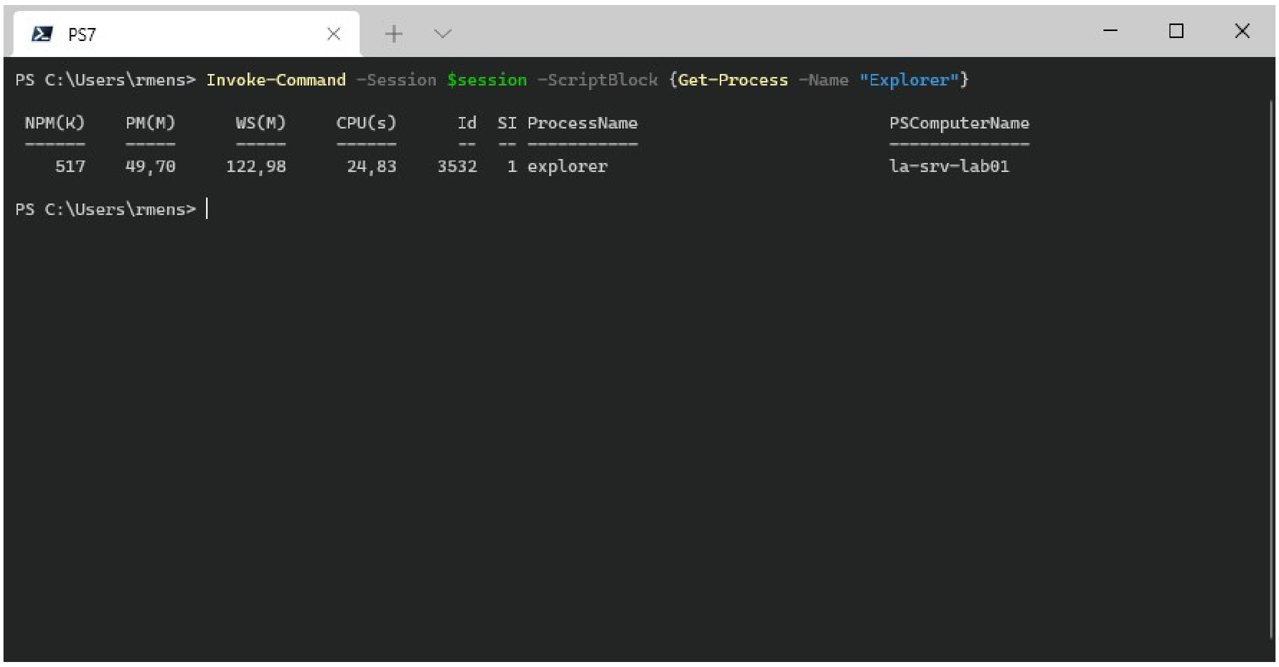
You can start a new PowerShell remoting session as followed:

# Directly open a remoting session
Enter-PSSession -HostName la-srv-lab01 -UserName admin
# Create a new session
$session = New-PSSession -HostName la-srv-lab01 -UserName admin
# Invoke a command on the remote machine

```
Invoke-Command -Session $session -ScriptBlock {Get-Process -Name "Explorer"}
# Result:
NPM(K) PM(M) WS(M) CPU(s) Id SI ProcessName PSComputerName
------ ----- ----- ------ -- -- ----------- -------------
517 49,78 122,99 24,83 3532 1 explorer la-srv-lab01
```

Also, note that we use the parameter `-Username` instead of `-Credentials` as with the WinRM method.

We can run remote PowerShell commands with the **Invoke-Command** cmdlet or open the PowerShell on the server with **Enter-PSSession**.

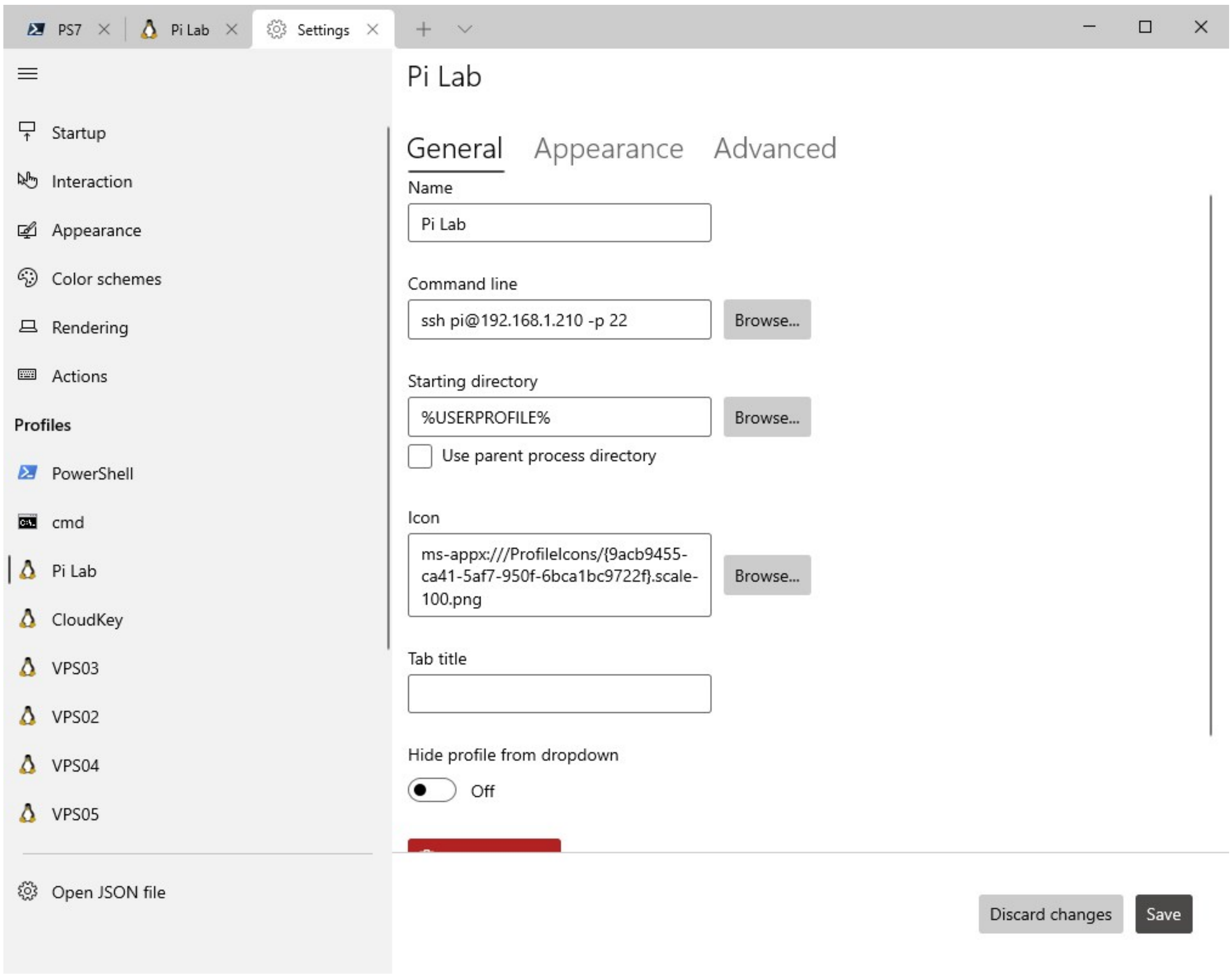


```
PS C:\Users\rmens> Invoke-Command -Session $session -ScriptBlock {Get-Process -Name "Explorer"}

 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName                    PSComputerName
 ------    -----      -----     ------      --  -- -----------                    --------------
    517    49,70     122,98      24,83    3532   1 explorer                       la-srv-lab01

PS C:\Users\rmens> |
```

Managing multiple SSH Connections

The advantage of tools like Putty is that you can manage and save your SSH connections. This allows you to easily reconnect to servers or network devices that you use often.

With Windows Terminal you can achieve the same thing. The new Windows Terminal allows you to easily store different connections and use different command line interfaces.



Pi Lab

Name: Pi Lab

Command line: ssh pi@192.168.1.210 -p 22

Starting directory: %USERPROFILE%

Use parent process directory

Icon: ms-appx:///ProfileIcons/{9acb9455-ca41-5af7-950f-6bca1bc9722f}.scale-100.png

Tab title:

Hide profile from dropdown: Off

If you want to know more about Winows Terminal then make sure you read this article where I explain all the details about this powerful terminal.

Wrapping Up

Using SSH with PowerShell is really convenient and eliminate the need for an extra tool (SSH Client). Personall I use PowerShell a lot, so when I need to check or do something on a server it’s super easy to quickly connect to it from my PowerShell window.

I hope you found this article useful, if you have any questions, then just drop a comment below.

Did you **Liked** this **Article**?
Get the latest articles like this **in your mailbox**
or share this article

I hate spam to, so you can unsubscribe at any time.