

The Inside Playbook

([//www.ansible.com/blog](https://www.ansible.com/blog))

Connecting to a Windows Host

April 24, 2018 by Bianca Henderson (<https://www.ansible.com/blog/author/bianca-henderson>)



Welcome to the first installment of our Windows-specific Getting Started (</blog/topic/getting-started>) series!

Would you like to automate some of your Windows hosts with Red Hat Ansible Tower, but don't know how to set everything up? Are you worried that Red Hat Ansible Engine won't be able to communicate with your Windows servers without installing a bunch of

We use cookies on our websites to deliver our online services. Details about how we use cookies and how you may disable them are set out in our [Privacy Statement \(https://www.redhat.com/en/about/privacy-policy\)](https://www.redhat.com/en/about/privacy-policy).

By using this website you agree to our use of cookies.

extra software? Do you want to easily automate everyone's best friend, Clippy?

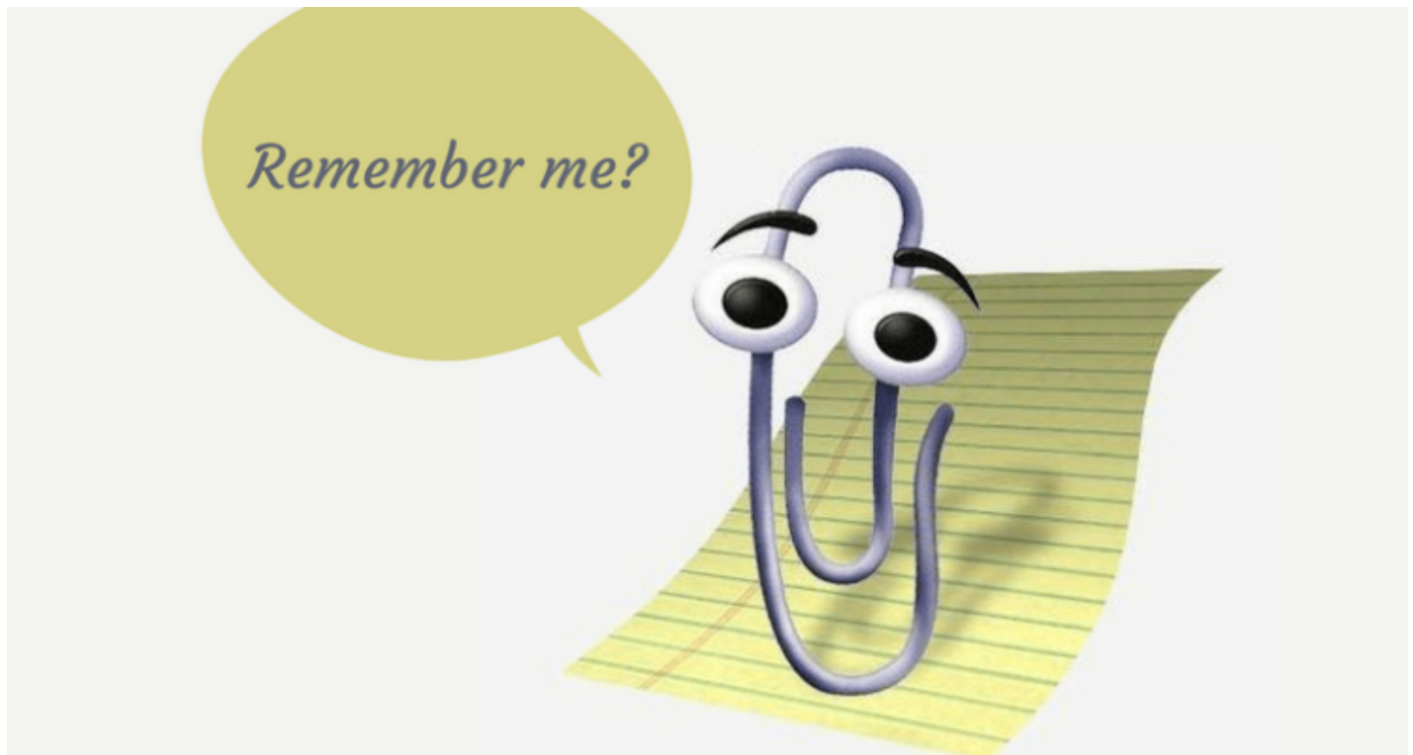


Image source: aguyiknow.com.au

We can't help with the last thing, but if you said yes to the other two questions, you've come to the right place. In this post, we'll walk you through all the steps you need to take in order to set up and connect to your Windows hosts with Ansible Engine.

Why Automate Windows Hosts?

A few of the many things you can do for your Windows hosts with Ansible Engine include:

- Starting, stopping and managing services
- Pushing and executing custom PowerShell scripts
- Managing packages with the Chocolatey package manager

In addition to connecting to and automating Windows hosts using local or domain users, you'll also be able to use `runas` to execute actions as the Administrator (the Windows alternative to Linux's `sudo` or `su`), so no privilege escalation ability is lost.

What's Required?

We use cookies on our websites to deliver our online services. Details about how we use cookies and how you may disable them are set out in our [Privacy Statement \(https://www.redhat.com/en/about/privacy-policy\)](https://www.redhat.com/en/about/privacy-policy).

Before we start, let's go over the basic requirements (http://docs.ansible.com/ansible/latest/user_guide/windows_setup.html#host-requirements). First, your control machine (where Ansible Engine will be executing your

chosen Windows modules from) needs to run Linux. Second, Windows support has been evolving rapidly, so make sure to use the newest possible version of Ansible Engine to get the latest features!

For the target hosts, you should be running at least Windows 7 SP1 or later or Windows Server 2008 SP1 or later. You don't want to be running something from the 90's like Windows NT, because this might happen:

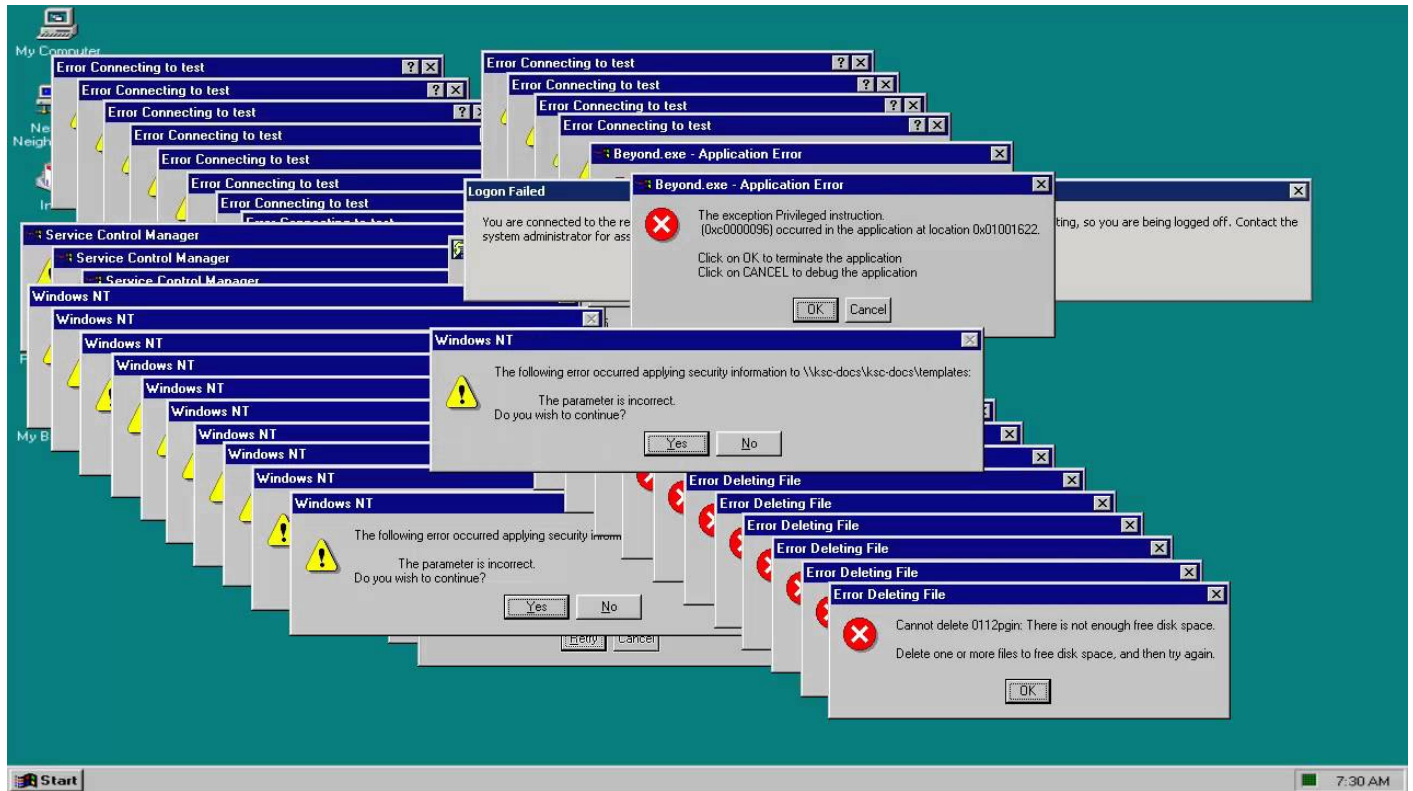


Image source: [youtube.com/watch?v=RoQxGYIwA4s](https://www.youtube.com/watch?v=RoQxGYIwA4s)

Lastly, since Ansible connects to Windows machines and runs PowerShell scripts by using Windows Remote Management ([https://msdn.microsoft.com/en-us/library/aa384291\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/aa384291(v=vs.85).aspx)) (WinRM) (as an alternative to SSH for Linux/Unix machines), a WinRM listener should be created and activated. The good news is, connecting to your Windows hosts can be done very easily and quickly using a script, which we'll discuss in the section below.

Step 1: Setting up WinRM

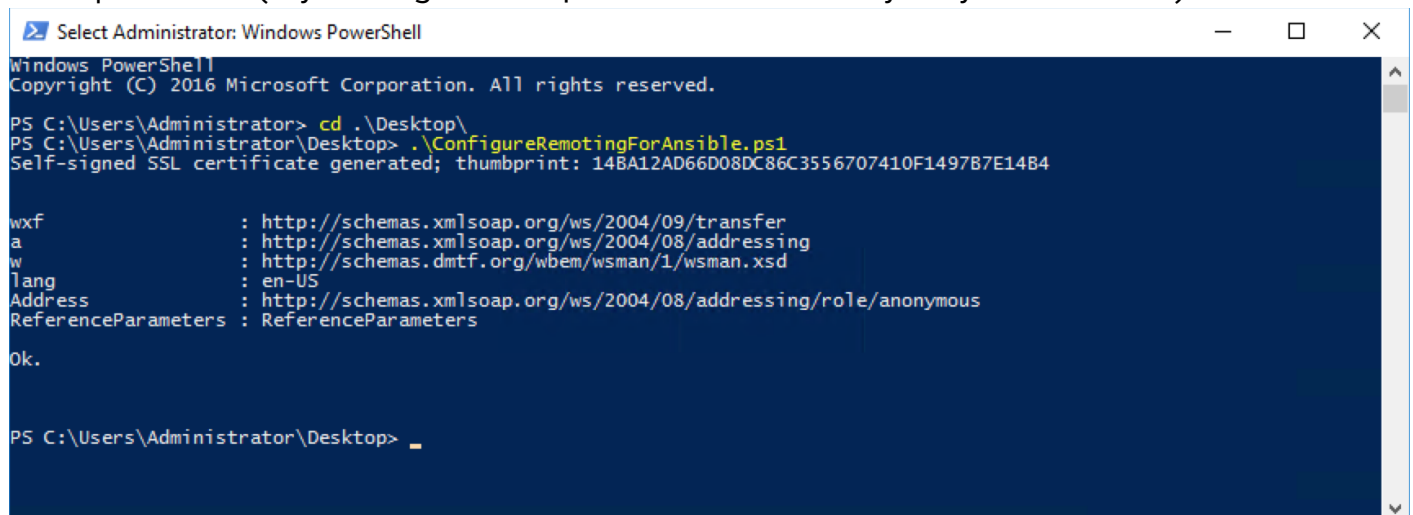
What's WinRM? It's a feature of Windows Vista and higher that lets administrators run management scripts remotely; it handles those connections by implementing the WS-Management Protocol based on Simple Object Access Protocol (SOAP) (<https://en.wikipedia.org/wiki/SOAP>). With WinRM, you can do cool stuff like

access, edit and update data from local and remote computers as a network administrator.

The reason WinRM is perfect for using with Ansible Engine is because you can obtain hardware data from WS-Management protocol implementations running on non-Windows operating systems (in this specific case, Linux). It's basically like a translator that allows different types of operating systems to work together.

So, how do we connect?

With most versions of Windows, WinRM ships in the box but isn't turned on by default. There's a Configure Remoting for Ansible (<https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/ConfigureRemotingForAnsible.ps1>) script you can run on the remote Windows machine (in a PowerShell console as an Admin) to turn on WinRM. To set up an https listener, build a self-signed cert and execute PowerShell commands, just run the script like in the example below (if you've got the .ps1 file stored locally on your machine):



```
Select Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> cd .\Desktop\
PS C:\Users\Administrator\Desktop> .\ConfigureRemotingForAnsible.ps1
Self-signed SSL certificate generated; thumbprint: 14BA12AD66D08DC86C3556707410F1497B7E14B4

wxsf      : http://schemas.xmlsoap.org/ws/2004/09/transfer
a         : http://schemas.xmlsoap.org/ws/2004/08/addressing
w         : http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd
lang      : en-US
Address   : http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters : ReferenceParameters

Ok.

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

Note: The win_psexec (http://docs.ansible.com/ansible/latest/modules/win_psexec_module.html) module will help you enable WinRM on multiple machines if you have lots of Windows hosts to set up in your environment.

For more information on WinRM and Ansible, check out the Windows Remote Management (http://docs.ansible.com/ansible/latest/user_guide/windows_winrm.html) documentation page.

Step 2: Install Pywinrm

We use cookies on our websites to deliver our online services. Details about how we use cookies and how you may disable them are set out in our [Privacy Statement \(https://www.redhat.com/en/about/privacy-policy\)](https://www.redhat.com/en/about/privacy-policy).

By using this website you agree to our use of cookies.

Since pywinrm dependencies aren't shipped with Ansible Engine (and these are necessary for using WinRM), make sure you install the pywinrm-related library on the machine that Ansible is installed on. The simplest method is to run `pip install pywinrm` in your Terminal.

Step 3: Set Up Your Inventory File Correctly

In order to connect to your Windows hosts properly, you need to make sure that you put in `ansible_connection=winrm` in the host vars (http://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#host-variables) section of your inventory file so that Ansible Engine doesn't just keep trying to connect to your Windows host via SSH.

Also, the WinRM connection plugin defaults to communicating via https, but it supports different modes like message-encrypted http. Since the "Configure Remoting for Ansible" script we ran earlier set things up with the self-signed cert, we need to tell Python, "Don't try to validate this certificate because it's not going to be from a valid CA." So in order to prevent an error, one more thing you need to put into the `host vars` section is: `ansible_winrm_server_cert_validation=ignore`

Just so you can see it in one place, here is an example host file (please note, some details for your particular environment will be different):

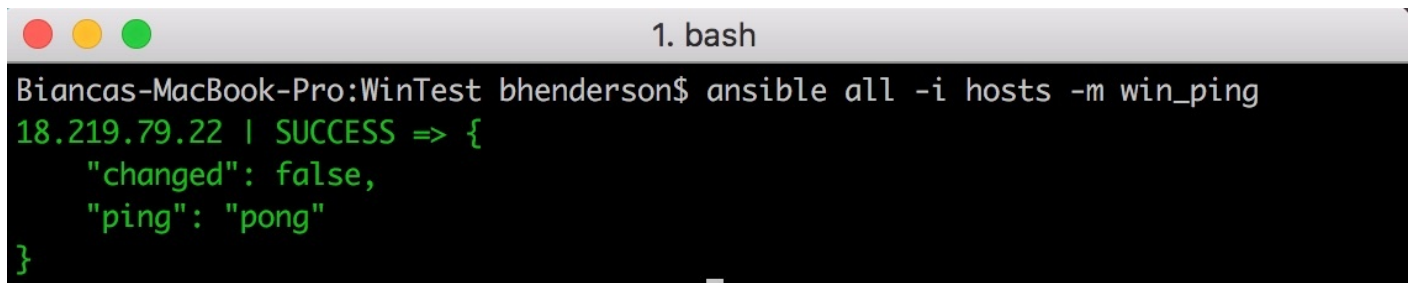
```
1  [win]
2  172.16.2.5
3  172.16.2.6
4
5  [win:vars]
6  ansible_user=vagrant
7  ansible_password=password
8  ansible_connection=winrm
9  ansible_winrm_server_cert_validation=ignore
```

Step 4: Test Connection

Let's check to see if everything is working. To do this, go to your control node's terminal and type `ansible [host_group_name_in_inventory_file] -i hosts -m win_ping`. Your output should look like this:

We use cookies on our websites to deliver our online services. Details about how we use cookies and how you may disable them are set out in our [Privacy Statement \(https://www.redhat.com/en/about/privacy-policy\)](https://www.redhat.com/en/about/privacy-policy).

By using this website you agree to our use of cookies.

A terminal window titled "1. bash" with a macOS-style title bar (red, yellow, green buttons). The prompt is "Biancas-MacBook-Pro:WinTest bhenderson\$". The command entered is "ansible all -i hosts -m win_ping". The output is in green text: "18.219.79.22 | SUCCESS => {" followed by "changed": false," on a new line, "ping": "pong" on a new line, and "}" on a new line.

```
1. bash
Biancas-MacBook-Pro:WinTest bhenderson$ ansible all -i hosts -m win_ping
18.219.79.22 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Note: The `win_` prefix on all of the Windows modules indicates that they are implemented in PowerShell and not Python.

Troubleshooting WinRM

Because WinRM can be configured in so many different ways, errors that seem Ansible Engine-related can actually be due to problems with host setup instead. Some examples of WinRM errors that you might see include an HTTP 401 or HTTP 500 error, timeout issues or a connection refusal. To get tips on how to solve these problems, visit the Common WinRM Issues

(http://docs.ansible.com/ansible/devel/user_guide/windows_setup.html#common-winrm-issues) section of our Windows Setup documentation page.

Conclusion

You should now be ready to automate your Windows hosts using Ansible, without the need to install a ton of additional software! Keep in mind, however, that even if you've followed the instructions above, some Windows modules have additional specifications (e.g., a newer OS or more recent PowerShell version). The best way to figure out if you're meeting the right requirements is to check the module-specific (http://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html) documentation pages.

For more in-depth information on how to use Ansible Engine to automate your Windows hosts, check out our Windows FAQ

(http://docs.ansible.com/ansible/latest/user_guide/windows_faq.html) and Windows Support (http://docs.ansible.com/ansible/latest/user_guide/windows.html) documentation page and stay tuned for more Windows-related blog posts!

Share:

We use cookies on our websites to deliver our online services. Details about how we use cookies and how you may disable them are set out in our [Privacy Statement \(https://www.redhat.com/en/about/privacy-policy\)](https://www.redhat.com/en/about/privacy-policy).

By using this website you agree to our use of cookies.