Windows SSH

- SSH Setup
  - Default Shell Configuration
- Ansible Configuration
- SSH Authentication
  - Key Authentication
  - GSSAPI Authentication
  - Password Authentication

On newer Windows versions, you can use SSH to connect to a Windows host. This is an alternative connection option to `WinRM <windows_winrm>`.

> While Ansible could use the SSH connection plugin with Windows nodes since Ansible 2.8, official support was only added in version 2.18.

SSH Setup

Microsoft provides an OpenSSH implementation with Windows since Windows Server 2019 as a Windows capability. It can also be installed through an upstream package under Win32-OpenSSH. Ansible officially only supports the OpenSSH implementation shipped with Windows, not the upstream package. The OpenSSH version must be version `7.9.0.0` at a minimum. This effectively means official support starts with Windows Server 2022 because Server 2019 ships with version `7.7.2.1`. Using older Windows versions or the upstream package might work but is not supported.

To install the OpenSSH feature on Windows Server 2022 and later, use the following PowerShell command:

```
Get-WindowsCapability -Name OpenSSH.Server* -Online |
    Add-WindowsCapability -Online
Set-Service -Name sshd -StartupType Automatic -Status Running
```

```
$firewallParams = @{
    Name        = 'sshd-Server-In-TCP'
    DisplayName = 'Inbound rule for OpenSSH Server (sshd) on TCP port 22'
    Action      = 'Allow'
    Direction   = 'Inbound'
    Enabled     = 'True'  # This is not a boolean but an enum
    Profile     = 'Any'
    Protocol    = 'TCP'
    LocalPort   = 22
}
New-NetFirewallRule @firewallParams

$shellParams = @{
    Path         = 'HKLM:\SOFTWARE\OpenSSH'
    Name         = 'DefaultShell'
    Value        = 'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe'
    PropertyType = 'String'
    Force        = $true
}
New-ItemProperty @shellParams
```

Default Shell Configuration

By default, OpenSSH on Windows uses `cmd.exe` as the default shell. While Ansible can work with this default shell it is recommended to change this to `powershell.exe` as it is better tested and should be faster than having `cmd.exe` as the default. To change the default shell you can use the following PowerShell script:

```
# Set default to powershell.exe
$shellParams = @{
    Path         = 'HKLM:\SOFTWARE\OpenSSH'
    Name         = 'DefaultShell'
    Value        = 'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe'
    PropertyType = 'String'
    Force        = $true
}
New-ItemProperty @shellParams
```

```
# Set default back to cmd.exe
Remove-ItemProperty -Path HKLM:\SOFTWARE\OpenSSH -Name DefaultShell
```

The new default shell setting will apply to the next SSH connection, there is no need to restart the `sshd` service. You can also use Ansible to configure the default shell:

```
- name: set the default shell to PowerShell
  ansible.windows.win_regedit:
    path: HKLM:\SOFTWARE\OpenSSH
    name: DefaultShell
    data: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
    type: string
    state: present

- name: reset SSH connection after shell change
  ansible.builtin.meta: reset_connection

- name: set the default shell to cmd
  ansible.windows.win_regedit:
    path: HKLM:\SOFTWARE\OpenSSH
    name: DefaultShell
    state: absent

- name: reset SSH connection after shell change
  ansible.builtin.meta: reset_connection
```

The `meta: reset_connection` is important to ensure the subsequent tasks will use the new default shell.

Ansible Configuration

To configure Ansible to use SSH for Windows hosts, you must set two connection variables:

- set `ansible_connection` to `ssh`
- set `ansible_shell_type` to `powershell` or `cmd`

The `ansible_shell_type` variable should reflect the `DefaultShell` configured on the Windows host. Other SSH options as documented under the `ssh <ssh_connection>` can also be set for the Windows host.

SSH Authentication

Win32-OpenSSH authentication with Windows is similar to SSH authentication on Unix/Linux hosts. While there are many authentication methods that can be used there are typically three used on Windows:

Option	Local Accounts	Active Directory Accounts	Credential Delegation
Key	Yes	Yes	No
GSSAPI	No	Yes	Yes
Password	Yes	Yes	Yes

In most cases it is recommended to use key or GSSAPI authentication over password authentication.

Key Authentication

SSH key authentication on Windows works in the same way as SSH key authentication for POSIX nodes. You can generate a key pair using the `ssh-keygen` command and add the public key to the `authorized_keys` file in the user's profile directory. The private key should be kept secure and not shared.

One difference is that the `authorized_keys` file for admin users is not located in the `.ssh` folder in the user's profile directory but in `C:\ProgramData\ssh\administrators_authorized_keys`. It is possible to change the location of the `authorized_keys` file for admin users back to the user profile directory by removing, or commenting, the lines in `C:\ProgramData\ssh\sshd_config` and restarting the `sshd` service.

```
Match Group administrators
    AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authorized_keys
```

SSH keys work with both local and domain accounts but suffer from the double-hop issue. This means that when using SSH key authentication with Ansible, the remote session will not have access to user credentials and will fail when attempting to access a network resource. To work around this problem, you can use `become <become>` on the task with the credentials of the user that needs access to the remote resource.

GSSAPI Authentication

GSSAPI authentication will use Kerberos to authenticate the user with the Windows host. To use GSSAPI authentication with Ansible, the Windows server must be configured to allow GSSAPI authentication by editing the `C:\ProgramData\ssh\sshd_config` file. Either add in the following line or edit the existing line:

```
GSSAPIAuthentication yes
```

Once edited restart the `sshd` service with `Restart-Service -Name sshd`.

On the Ansible control node, you need to have Kerberos installed and configured with the domain the Windows host is a member of. How to set this up and configure is outside the scope of this document. Once the Kerberos realm is configured you can use the `kinit` command to get a ticket for the user you are connecting with and `klist` to verify what tickets are available:

```
> kinit username@REALM.COM
Password for username@REALM.COM

> klist
Ticket cache: KCM:1000
Default principal: username@REALM.COM

Valid starting     Expires            Service principal
29/08/24 13:54:51  29/08/24 23:54:51  krbtgt/REALM.COM@REALM.COM
        renew until 05/09/24 13:54:48
```

Once you have a valid ticket you can use the `ansible_user` hostvar to specify the UPN username and Ansible will automatically use the Kerberos ticket for that user when using SSH.

It is also possible to enable unconstrained delegation through GSSAPI authentication to have the Windows node access network resources. For GSSAPI delegation to work the ticket retrieved by `kinit` must be forwardable and `ssh` must be called with the `-o GSSAPIDelegateCredentials=yes` option. To retrieve a forwardable ticket either use the `-f` flag with `kinit` or add `forwardable = true` under `[libdefaults]` in the `/etc/krb5.conf` file.

```
> kinit -f username@REALM.COM
Password for username@REALM.COM

# -f will show the ticket flags, we want to see F
> klist -f
Ticket cache: KCM:1000
Default principal: username@REALM.COM

Valid starting     Expires            Service principal
```

```
29/08/24 13:54:51  29/08/24 23:54:51  krbtgt/REALM.COM@REALM.COM
        renew until 05/09/24 13:54:48, Flags: FRIA
```

The `GSSAPIDelegateCredentials=yes` option can either be set in the `~/.ssh/config` file or as a hostvar variable in the inventory:

```
ansible_ssh_common_args: -o GSSAPIDelegateCredentials=yes
```

Unlike the `psrp` or `winrm` connection plugins, the SSH connection plugin cannot get a Kerberos TGT ticket when provided with an explicit username and password. This means that the user must have a valid Kerberos ticket before running the playbook.

See `windows_winrm_kerberos` for more information on how to configure, use, and troubleshoot Kerberos authentication.

Password Authentication

Password authentication is the least secure method of authentication and is not recommended. However, it is possible to use password authentication with Windows SSH. To use password authentication with Ansible, set the `ansible_password` variable in the inventory file or in the playbook. Using password authentication requires the `sshpass` package to be installed on the Ansible control node.

Password authentication works like WinRM CredSSP authentication where the username and password is given to the Windows host and it will perform unconstrained delegation to access network resources.

□□□ #2
□ □□□ □□□ 2026-10-31 14:39:58 UTC
□ □□□ □□□ 2026-10-31 14:39:59 UTC