How to enable winrm in windows

Part 1 — Enable WinRM on HTTP (port 5985)

This is the easy part. Run **PowerShell as Administrator** on the Windows Server:

powershell

```
# One-shot configuration: starts the service, sets it to auto-start,
# creates the HTTP listener on 5985, and opens the firewall rule.
Enable-PSRemoting -Force
```

That's it for HTTP. Verify:

powershell

```
# Should show a Transport=HTTP listener on port 5985
winrm enumerate winrm/config/Listener

# Or with PowerShell cmdlets
Get-WSManInstance -ResourceURI winrm/config/listener -Enumerate
```

If the host is on a "Public" network profile, `Enable-PSRemoting` will refuse. Either change the profile to Private/Domain, or force it:

powershell

```
Enable-PSRemoting -Force -SkipNetworkProfileCheck
```

Part 2 — Enable WinRM on HTTPS (port 5986)

HTTPS requires a certificate. You have three options depending on your environment.

Option A — Self-signed certificate (lab / quick test)

powershell

```
# 1) Create a self-signed cert. Use the server's real FQDN.
$fqdn = [System.Net.Dns]::GetHostByName($env:COMPUTERNAME).HostName
$cert = New-SelfSignedCertificate `
    -DnsName $fqdn `
```

```
    -CertStoreLocation Cert:\LocalMachine\My `
    -KeyExportPolicy Exportable `
    -KeySpec Signature `
    -KeyLength 2048 `
    -KeyAlgorithm RSA `
    -HashAlgorithm SHA256 `
    -NotAfter (Get-Date).AddYears(5)

Write-Host "Thumbprint: $($cert.Thumbprint)"
Write-Host "FQDN:       $fqdn"

# 2) Create the HTTPS listener bound to that cert
New-Item -Path WSMan:\LocalHost\Listener `
    -Transport HTTPS `
    -Address * `
    -CertificateThumbPrint $cert.Thumbprint `
    -HostName $fqdn `
    -Force

# 3) Open the firewall for 5986
New-NetFirewallRule -DisplayName "WinRM HTTPS-In (TCP 5986)" `
    -Direction Inbound `
    -LocalPort 5986 `
    -Protocol TCP `
    -Action Allow `
    -Profile Domain,Private
```

Option B — Certificate from your enterprise CA (production)

If you have an Active Directory Certificate Services CA, request a Server Authentication cert first:

powershell

```
# Request from AD CS using a template that allows Server Authentication
# (the template name varies — common ones are "WebServer" or "Computer")
$req = Get-Certificate `
    -Template "WebServer" `
    -DnsName $fqdn `
    -SubjectName "CN=$fqdn" `
    -CertStoreLocation Cert:\LocalMachine\My
```

```
$cert = $req.Certificate

# Then create the listener the same way as Option A, step 2+3
New-Item -Path WSMan:\LocalHost\Listener `
    -Transport HTTPS `
    -Address * `
    -CertificateThumbPrint $cert.Thumbprint `
    -HostName $fqdn `
    -Force
```

Option C — Existing certificate (you already have one)

powershell

```
# Find an existing Server Auth cert for this host
$cert = Get-ChildItem Cert:\LocalMachine\My |
    Where-Object {
        $_.Subject -match $fqdn -and
        $_.EnhancedKeyUsageList.FriendlyName -contains "Server Authentication"
    } | Select-Object -First 1

# Use its thumbprint in step 2 of Option A
```

Part 3 — Verify both listeners

powershell

```
# List all listeners; you should see two — HTTP on 5985 and HTTPS on 5986
winrm enumerate winrm/config/Listener
```

Expected output (abbreviated):

```
Listener
    Address = *
    Transport = HTTP
    Port = 5985
    Enabled = true

Listener
    Address = *
    Transport = HTTPS
    Port = 5986
```

```
    Enabled = true
    CertificateThumbprint = A1B2C3D4...
```

Test from the **same server** first (loopback check):

powershell

```
# HTTP
Test-WSMan -ComputerName localhost

# HTTPS — use -UseSSL and skip CA check if self-signed
$opt = New-PSSessionOption -SkipCACheck -SkipCNCheck -SkipRevocationCheck
Test-WSMan -ComputerName $fqdn -UseSSL -SessionOption $opt
```

Then test **from another machine** on the network:

powershell

```
# From a different Windows box
Test-WSMan -ComputerName dc01.corp.local
Test-WSMan -ComputerName dc01.corp.local -UseSSL -SessionOption $opt
```

Or from your Linux box with the Python script you already have:

bash

```
python winrm_ad_test.py --host dc01.corp.local --user 'CORP\admin'
```

Part 4 — Hardening (recommended for production)

After both listeners work, lock things down:

powershell

```
# Disable unencrypted traffic (forces NTLM/Kerberos encryption or HTTPS)
Set-Item -Path WSMan:\LocalHost\Service\AllowUnencrypted -Value $false

# Disable Basic auth (use NTLM/Kerberos/CredSSP instead)
Set-Item -Path WSMan:\LocalHost\Service\Auth\Basic -Value $false

# If you want to require HTTPS only, remove the HTTP listener
Get-ChildItem WSMan:\LocalHost\Listener |
    Where-Object Keys -match "Transport=HTTP$" |
```

```
    Remove-Item -Recurse

# Optional: restrict who can connect (default is BUILTIN\Administrators)
Set-PSSessionConfiguration -Name Microsoft.PowerShell `
    -ShowSecurityDescriptorUI
```

For NTLM connections from non-domain Linux clients, you may also need to add the server to TrustedHosts on the **client**, not the server:

powershell

```
# On the Windows client (not needed for Linux pywinrm clients)
Set-Item WSMan:\LocalHost\Client\TrustedHosts -Value "dc01.corp.local" -Force
```

Part 5 — Rolling it out at scale via GPO

For multiple servers, do it once via Group Policy instead of per-host:

`Computer Configuration → Policies → Administrative Templates → Windows Components → Windows Remote Management (WinRM) → WinRM Service`

- **Allow remote server management through WinRM** → Enabled, IPv4/IPv6 filter = `*`
- **Allow Basic authentication** → Disabled
- **Allow unencrypted traffic** → Disabled

And in `Computer Configuration → Policies → Windows Settings → Security Settings → System Services`, set **Windows Remote Management (WS-Management)** to Automatic.

Firewall rule via GPO: `Computer Configuration → Policies → Windows Settings → Security Settings → Windows Defender Firewall with Advanced Security → Inbound Rules → New Rule → Predefined → "Windows Remote Management"`.

The HTTPS cert is the only piece GPO can't easily push — that needs either AD CS auto-enrollment (recommended) or a per-host script.

Common pitfalls

The most frequent issues, in order of how often I've seen them bite people: the firewall rule existing but being scoped to "Private only" while the NIC is on a "Public" profile; the cert's CN/SAN not matching the hostname clients actually use (cert says `dc01` but clients connect to `dc01.corp.local`); the HTTPS listener pointing at a thumbprint whose cert has expired or been removed; and `AllowUnencrypted=false` combined with Basic auth, which leaves no working auth path. If `Test-WSMan` fails, check those four before anything else.

Want me to put any of this into a single bootstrap PowerShell script you can drop onto a new server?

Revision #1
Created 19 May 2026 15:59:28 by Admin
Updated 19 May 2026 16:00:36 by Admin