AI-03160 Hyper-V Networking Reset

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	@November 13, 2025 10:17 AM

I need to start fresh on my windows server 2016 hyper-v with Ubuntu server vm, what is the best method to do it without reinstalling the hyper-v. My problem I ran into is the virtual network inside hyper-v, no issues in Ubuntu

Perfect — thanks for clarifying. Since your issue is with **Hyper-V's virtual networking layer** (not Ubuntu itself), you don't need to reinstall Hyper-V or Windows Server. What you want is essentially a **network stack reset** inside Hyper-V. Here's a structured protocol you can follow:

🛠️ Step-by-Step Protocol: Reset Hyper-V Networking

1. Audit Existing Virtual Switches

- Open **Hyper-V Manager → Virtual Switch Manager**.
- Check for multiple or misconfigured switches (External, Internal, Private).
- Document current settings before making changes (important for reproducibility).

2. Remove Problematic Switches

- Delete any virtual switches that are not working or are duplicates.
- This clears out stale bindings that often cause connectivity issues.

3. Recreate a Clean External Switch

- Create a new **External Virtual Switch** bound to the correct physical NIC.

- Ensure “Allow management OS to share this network adapter” is checked if you want the host to use it too.
- Name it clearly (e.g., `vSwitch-External`).

4. Reset Network Adapter Bindings

- Go to **Control Panel → Network and Sharing → Change adapter settings**.
- Right-click your physical NIC → Properties.
- Ensure **Hyper-V Extensible Virtual Switch** is enabled only on the NIC used for the external switch.
- Disable it on NICs you don’t want Hyper-V to control.

5. Reattach VM Network Adapters

- In your Ubuntu VM settings, remove the old network adapter.
- Add a new adapter and connect it to the freshly created virtual switch.
- Boot the VM and verify connectivity (`ping`, `curl`, etc.).

6. Optional: Reset Host Networking Stack

If issues persist:

- Run in PowerShell (as Administrator):

  This resets all network adapters and bindings (including Hyper-V).
  ⚠️ It will drop all networking temporarily, so plan downtime.

  ```
  netcfg -d
  ```

⚖️ Best Practice for Future Fresh Starts

- **Checkpoint Baseline VM**: Once Ubuntu is working, take a Hyper-V checkpoint. Roll back here instead of rebuilding.
- **Document Switch Configs**: Keep a reproducible note of which NICs are bound to which switches.

- **Separate Management & VM Traffic**: If possible, dedicate one NIC for Hyper-V external switch and another for host management.

👉 Do you want me to draft a **PowerShell script** that automates the teardown and recreation of Hyper-V virtual switches, so you can reset networking in one reproducible step?

Here's a reproducible **PowerShell reset script** you can use to tear down and rebuild Hyper-V virtual switches cleanly on Windows Server 2016. This avoids reinstalling Hyper-V while giving you a fresh networking baseline:

Here's a **modular PowerShell protocol** you can use to reset Hyper-V networking and rebuild switches in a flexible way. Instead of hardcoding only an External switch, this script lets you choose between **External, Internal, or Private** depending on your project needs:

```
# Reset-HyperVNetworking-Modular.ps1
# Run as Administrator

Write-Host "Stopping all VMs..."
Get-VM | Stop-VM -Force

Write-Host "Removing existing virtual switches..."
Get-VMSwitch | ForEach-Object {
    Write-Host "Removing switch: $($_.Name)"
    Remove-VMSwitch -Name $_.Name -Force
}

# Prompt user for switch type
$choice = Read-Host "Enter switch type (External / Internal / Private)"

switch ($choice.ToLower()) {
    "external" {
        Write-Host "Listing physical adapters..."
        Get-NetAdapter | Where-Object { $_.Status -eq "Up" } | Format-Table -A
utoSize
```

```
        $nicName = Read-Host "Enter the physical NIC name to bind"
        $switchName = Read-Host "Enter name for new External switch"

        New-VMSwitch -Name $switchName -NetAdapterName $nicName -Allo
wManagementOS $true
        Write-Host "External switch '$switchName' created on NIC '$nicName'"
    }
    "internal" {
        $switchName = Read-Host "Enter name for new Internal switch"
        New-VMSwitch -Name $switchName -SwitchType Internal
        Write-Host "Internal switch '$switchName' created"
    }
    "private" {
        $switchName = Read-Host "Enter name for new Private switch"
        New-VMSwitch -Name $switchName -SwitchType Private
        Write-Host "Private switch '$switchName' created"
    }
    default {
        Write-Host "Invalid choice. Please run script again."
        exit
    }
}

# Reattach VM adapters to the new switch
$newSwitch = Get-VMSwitch | Sort-Object Id -Descending | Select-Object -Fi
rst 1
Write-Host "Reattaching VM adapters to switch: $($newSwitch.Name)"

Get-VM | ForEach-Object {
    Write-Host "Configuring VM: $($_.Name)"
    Set-VMNetworkAdapter -VMName $_.Name -SwitchName $newSwitch.Na
me
}
```

```
Write-Host "Networking reset complete. Restart VMs to apply changes."
```

🔑 How This Helps

- **Choice-driven**: You decide whether the VM network should reach the outside world (External), only the host (Internal), or only other VMs (Private).
- **Clean reset**: Removes all stale switches before creating new ones.
- **Reproducible**: Prompts ensure you document NIC names and switch types consistently.

⚖️ Best Practice

- Keep one **External switch** for internet access.
- Use **Internal** for host ↔ VM communication without exposing to LAN.
- Use **Private** for isolated VM labs.
- Document each baseline so you can repeat the setup quickly.

Would you like me to extend this protocol into a **decision matrix** (table format) that maps project scenarios — e.g., lab testing, production server, isolated sandbox — to the recommended switch type and configuration?

```
# Reset-HyperVNetworking.ps1
# Run as Administrator

Write-Host "Stopping all VMs..."
Get-VM | Stop-VM -Force

Write-Host "Removing existing virtual switches..."
Get-VMSwitch | ForEach-Object {
    Write-Host "Removing switch: $($_.Name)"
    Remove-VMSwitch -Name $_.Name -Force
}
```

```
Write-Host "Listing physical adapters..."
Get-NetAdapter | Where-Object { $_.Status -eq "Up" } | Format-Table -AutoSi
ze

# Replace with the name of your physical NIC
$nicName = "Ethernet"

Write-Host "Creating new external virtual switch..."
New-VMSwitch -Name "vSwitch-External" -NetAdapterName $nicName -Allo
wManagementOS $true

Write-Host "Reattaching VM adapters..."
Get-VM | ForEach-Object {
    Write-Host "Configuring VM: $($_.Name)"
    Set-VMNetworkAdapter -VMName $_.Name -SwitchName "vSwitch-Extern
al"
}

Write-Host "Networking reset complete. Restart VMs to apply changes."
```

🔑 How to Use

1. Save the script as `Reset-HyperVNetworking.ps1`.
2. Run it in **PowerShell (Admin)** on your Windows Server 2016 host.
3. Replace `$nicName = "Ethernet"` with the actual name of your physical NIC (check with `Get-NetAdapter`).
4. After execution, all VMs will be reattached to the new clean external switch.

⚖️ Why This Works

- **Removes stale switches** that often cause binding issues.
- **Recreates a single clean external switch** bound to the correct NIC.

- **Reattaches VM adapters automatically**, so you don’t need to manually edit each VM.

Would you like me to extend this into a **modular protocol** that lets you choose between External, Internal, or Private switches depending on your project needs, instead of hardcoding only External?