Proxmox Development Lab Setup Guide

Building Windows VM Templates for Automated Security Testing

Version 1 — April 2026

Cutaway Security, LLC
https://www.cutawaysecurity.com
https://github.com/cutaway-security

Table of Contents

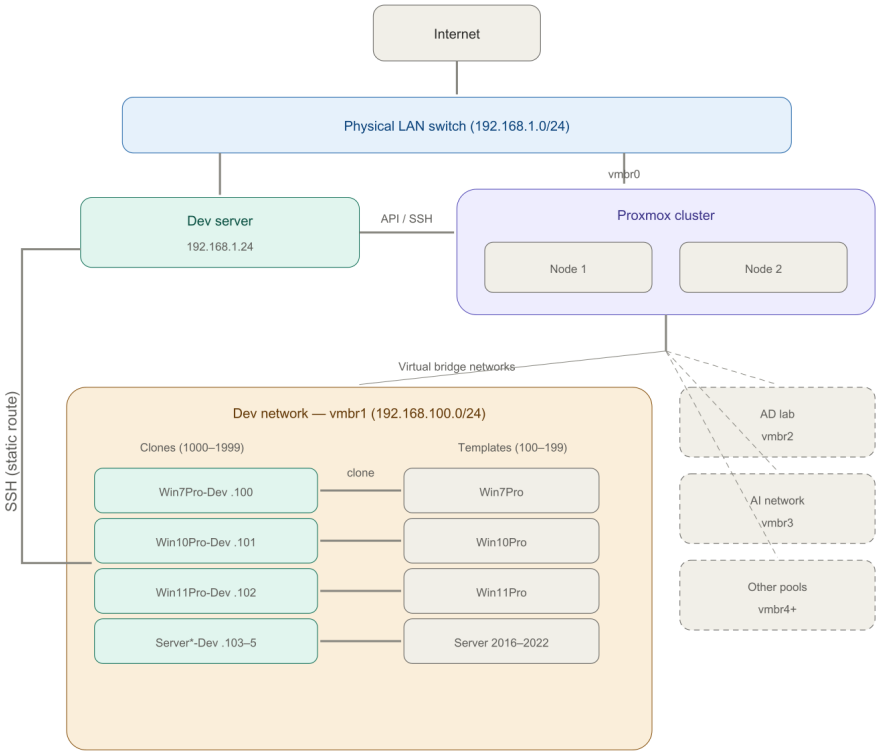
Overview

This guide documents the process of setting up a Proxmox-based development lab with Windows VM templates, isolated networking, and remote management via the Proxmox API and SSH. The lab supports multiple VM pools (development, Active Directory labs, AI workloads, etc.) with templates for rapid provisioning.

Architecture Summary

- **Physical LAN Switch**: 192.168.1.0/24 — connects all physical devices
- **Dev Server**: Linux workstation on the physical LAN used for development, Claude Code, and SSH-based testing against VMs
- **Proxmox Cluster**: Two-node cluster connected to the physical LAN via vmbr0, hosting multiple virtual bridge networks
- **Dev Network** (vmbr1): 192.168.100.0/24 — isolated VM network with NAT to internet

Network Architecture



Windows Editions Covered

- Windows 7 Pro (64-bit)
- Windows 10 Pro (64-bit)
- Windows 11 Pro (64-bit)
- Windows Server 2016 Datacenter (Desktop Experience)
- Windows Server 2019 Datacenter (Desktop Experience)
- Windows Server 2022 Datacenter (Desktop Experience)

---

1. Prerequisites

1.1 ISO Images

Upload the following ISOs to your Proxmox ISO storage library:

- Windows installation ISOs for each edition listed above
- VirtIO drivers ISO (`virtio-win.iso`)
- **Note**: For Windows 7, use `virtio-win-0.1.173.iso` — this is the last version with Win7 driver support that has been tested. Newer versions have dropped the `w7` driver subdirectories. Note that with this version, drivers must be installed manually via Device Manager rather than through the guest tools installer. Download from: `https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/archive-virtio/`
- Sysmon (latest version from Sysinternals, downloaded separately)
- **Note**: Windows 7 requires an older version of Sysmon. Version 10.42 is confirmed working. This version is no longer available from the Sysinternals website but can be downloaded via the Wayback Machine: `https://web.archive.org/web/20200214203827/https://download.sysinternals.com/files/Sysmon.zip`

1.2 Suggested VM ID Scheme

ID Range	Purpose
100–199	System templates
1000–1999	Development pool VMs
2000–2999	AD lab VMs
3000–3999	AI VMs

---

2. Proxmox Network Configuration

2.1 Create the Dev Network Bridge (vmbr1)

In the Proxmox GUI: **Datacenter** → **[Your Node]** → **Network** → **Create** → **Linux Bridge**

Setting	Value
Name	vmbr1
IPv4/CIDR	192.168.100.1/24
Autostart	Checked
Bridge Ports	*(leave blank)*
Comment	Dev network

Click **Apply Configuration** or reboot the node.

2.2 Enable NAT and IP Forwarding

The dev network is isolated from the physical LAN. To allow VMs on vmbr1 to reach the internet, enable NAT on the Proxmox host.

Edit `/etc/network/interfaces` on the Proxmox host and add the following lines to the vmbr1 block:

```
auto vmbr1
iface vmbr1 inet static
    address 192.168.100.1/24
    bridge-ports none
    bridge-stp off
    bridge-fd 0
    post-up echo 1 &gt; /proc/sys/net/ipv4/ip_forward
    post-up iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o vmbr0 -j MASQUERADE
    post-down iptables -t nat -D POSTROUTING -s 192.168.100.0/24 -o vmbr0 -j MASQUERADE
```

Apply changes:

```
ifreload -a
```

Verify:

```
# Should return 1
cat /proc/sys/net/ipv4/ip_forward

# Should show the MASQUERADE rule
iptables -t nat -L -n
```

2.3 Static Route on Dev Server

If the dev server (on the physical LAN at 192.168.1.0/24) needs bidirectional communication with VMs on the dev network, add a static route pointing the dev subnet to the Proxmox host.

Immediate (non-persistent):

```
sudo ip route add 192.168.100.0/24 via &lt;PROXMOX_HOST_VMBR0_IP&gt;
```

Persistent via NetworkManager GUI (Kubuntu):

1. Open Network Manager → select your wired connection → edit
2. Go to the **IPv4** tab → **Routes**
3. Add a route:

Address	Netmask	Gateway	Metric
192.168.100.0	255.255.255.0	*Proxmox host vmbr0 IP*	*(leave blank)*

4. Save and reconnect the interface

Persistent via netplan (if not using NetworkManager):

```
# /etc/netplan/*.yaml
network:
  ethernets:
    eno1:
      dhcp4: true
      routes:
        - to: 192.168.100.0/24
          via: &lt;PROXMOX_HOST_VMBR0_IP&gt;
```

Apply:

```
sudo netplan apply
```

2.4 VM Static IP Configuration

Configure static IPs on each VM individually. All VMs on the dev network use:

- **Subnet**: 192.168.100.0/24
- **Gateway**: 192.168.100.1
- **DNS**: 8.8.8.8 (the Proxmox host is not running a DNS server)

---

3. Create VM Pools

In the Proxmox GUI: **Datacenter** → **Permissions** → **Pools**

Create pools for each logical group:

Pool Name	Description
dev	Development systems
ad-lab-01	Active Directory lab
ai	AI/MCP/RAG workloads

Pools are logical groupings that allow batch permission management and provide a filter in the GUI. VMs can be reassigned between pools at any time via the VM's **Options** tab.

---

4. Create VM Templates

Templates are created by installing a Windows VM, configuring it, running Sysprep, and converting to a template. Clones are then made from the template and assigned to pools.

4.1 Create the VM via GUI

The following settings apply to **all editions except Windows 11** (see Section 4.2 for Win11-specific settings).

General Tab

Setting	Value
VM ID	Per the ID scheme above
Name	Descriptive template name (e.g., `win10-template`)

OS Tab

Setting	Value
ISO Image	Select the Windows ISO
Type	Microsoft Windows
Version	Select the matching version
Add additional drive for VirtIO	Checked — select your `virtio-win.iso`

System Tab

Setting	Value
Machine	q35
BIOS	OVMF (UEFI)
Add EFI Disk	Checked, storage: **local-lvm**
Pre-Enroll Keys	Unchecked (except Win11, see Section 4.2)
SCSI Controller	VirtIO SCSI Single
Qemu Agent	Checked

Disks Tab

Setting	Value
Bus/Device	SCSI
Storage	local-lvm
Disk Size	32–64 GB depending on edition
IO Thread	Checked

CPU Tab

Setting	Value
Type	host
Cores	2–4

Memory Tab

Setting	Value
Memory (MiB)	4096 minimum (8192 preferred for Win11/Server)

Network Tab

Setting	Value
Model	VirtIO (paravirtualized)
Bridge	vmbr0 (for template build; clones will use vmbr1)

4.2 Windows 11-Specific Settings

Windows 11 requires additional hardware for installation:

System Tab — add these to the settings above:

Setting	Value
Pre-Enroll Keys	**Checked** (enables Secure Boot)
Add TPM	**Checked**
TPM Storage	**local-lvm**
TPM Version	**v2.0**

BitLocker Prevention: Win11 may auto-enable BitLocker. **Immediately after reaching the desktop**, before anything else, open an elevated PowerShell and run:

```
reg add "HKLM\SYSTEM\CurrentControlSet\Control\BitLocker" /v PreventDeviceEncryption /t
    REG_DWORD /d 1 /f
manage-bde -status C:
```

If it shows encryption in progress:

```
manage-bde -off C:
```

Wait for decryption to complete (check with `manage-bde -status C:` until it shows "Fully Decrypted").

Offline Install (OOBE Bypass): Win11 requires a network connection during OOBE setup. To install offline, when you reach the network requirement screen, press **Shift+F10** to open a command prompt and run:

```
oobe\bypassnro
```

The system will reboot. When you get back to the network screen, click **I don't have internet** → **Continue with limited setup**. Complete the local account setup from there.

Test Clone Before Sysprep: Due to Win11's EFI/TPM/BitLocker complexity, perform a test clone before investing time in Sysprep. Shut down the VM, create a full clone, and verify it boots. If it fails with an EFI error (`winload.efi error 0xc00000f`), troubleshoot before proceeding. Delete the test clone after confirming. See Appendix B for details.

Sysprep Blocked Apps: Win11 may have pre-installed apps that block Sysprep. See Appendix C for the procedure to identify and remove blocking applications.

4.3 Windows Server-Specific Notes

Choose Desktop Experience: During the edition selection screen, select the **(Desktop Experience)** variant. Selecting the default installs Server Core (command-line only, no GUI).

IE Enhanced Security Configuration: After install, open **Server Manager** → **Local Server** → **IE Enhanced Security Configuration** and set both Administrators and Users to **Off**. This prevents IE from blocking all web access.

Administrator Account: Server editions keep the built-in Administrator account enabled after Sysprep, unlike workstation editions. No need to create a separate setup user.

Server 2016 — PowerShell TLS 1.2: PowerShell on Server 2016 defaults to TLS 1.0, which causes failures with Windows Update, NuGet, and other network operations. Apply the TLS 1.2 registry settings (see Appendix D) immediately after install, before running updates.

Disable Shutdown Event Tracker: Server editions prompt for a shutdown reason by default, which blocks remote and automated shutdowns. Disable via `gpedit.msc` → Computer Configuration → Administrative Templates → System → **Display Shutdown Event Tracker** → set to **Disabled**.

4.4 Windows 7-Specific Notes

Windows 7 requires additional steps due to its age.

During Installation: When the installer cannot find the hard drive, click **Load driver** → **Browse** → **VirtIO CD** → `vioscsi\w7\amd64` → select the driver → Next.

Network Adapter: The VirtIO network driver may not work on Win7. If the VM has no network after install, shut down the VM, change the NIC from VirtIO to **Intel E1000** in the Hardware tab, and restart.

Bootstrap TLS and Windows Update: A fresh Win7 install cannot reach Windows Update or browse the web due to outdated TLS support and root certificates. You must manually install prerequisite updates.

On a working machine, download these from the Microsoft Update Catalog (`catalog.update.microsoft.com`) — **get the x64 versions**:

1. **KB3020369** — April 2015 servicing stack update
2. **KB3125574** — Convenience rollup
3. **KB3140245** — TLS 1.2 support for WinHTTP

Transfer these to the VM. The VM cannot browse the web, so use one of these methods:

• **Python HTTP server** (fastest): On any machine on the same network, run `python3 -m http.server 8080` from the directory containing the files. On the Win7 VM, open IE and navigate to

`http://<host-ip>:8080/`. HTTP does not require TLS.

- **Custom ISO**: On the Proxmox host, place the files in a directory and build an ISO:

```
mkdir /tmp/win7-updates
# Copy .msu files into /tmp/win7-updates
genisoimage -o /var/lib/vz/template/iso/win7-updates.iso -J -r /tmp/win7-updates/
```

Attach the ISO as a CD-ROM drive to the VM.

Install in order, rebooting between each:

1. KB3020369 (x64) → reboot
2. KB3125574 (x64) → reboot
3. KB3140245 (x64) → reboot

After KB3140245, enable TLS 1.2 via registry:

```
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\WinHttp" /v
    DefaultSecureProtocols /t REG_DWORD /d 0x800 /f
reg add "HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Internet
    Settings\WinHttp" /v DefaultSecureProtocols /t REG_DWORD /d 0x800 /f
```

Reboot. Windows Update (`wuapp`) should now connect.

---

5. Post-Install Template Configuration Checklist

Perform these steps on each VM before converting to a template. All commands should be run from an **elevated command prompt (cmd.exe)** unless otherwise noted. PowerShell may be used for steps that do not involve Sysmon.

5.1 Install VirtIO Drivers and QEMU Guest Agent

Run `virtio-win-guest-tools.exe` from the mounted VirtIO ISO. This installs NIC, balloon, SCSI, and display drivers plus the QEMU Guest Agent in one step.

Verify the QEMU Guest Agent service is running:

```
Get-Service QEMU-GA
```

Win7 Note: With `virtio-win-0.1.173`, the guest tools installer may run silently without prompts and may not install all drivers. Open Device Manager and manually install drivers for any remaining unknown devices by browsing to the appropriate subdirectory on the VirtIO CD. The QEMU display adapter (`QEMUVGID`) can be skipped — the default display driver works fine for dev/test VMs.

5.2 Windows Update

Run one round of Windows Update. For Win7, ensure the TLS bootstrap steps from Section 4.4 are completed first.

5.3 Disable Windows Defender Real-Time Protection

For isolated test VMs only. Use Group Policy so the setting persists after Sysprep:

`gpedit.msc` → Computer Configuration → Administrative Templates → Windows Defender Antivirus → Real-time Protection → **Turn off real-time protection** → Enabled

5.4 Set Network Profile to Private

Proxmox virtual bridges cause Windows to treat the network as "unidentified" on each boot, prompting for network discovery settings and potentially blocking communications. To fix this, create a GPO startup script that forces the network profile to Private on every boot.

For Windows 10, Windows 11, and Server 2016/2019/2022, run the following from an elevated PowerShell:

```
$scriptDir = "C:\Windows\System32\GroupPolicy\Machine\Scripts\Startup"
New-Item -Path $scriptDir -ItemType Directory -Force
$script = @'
Start-Sleep -Seconds 5
Get-NetConnectionProfile | Set-NetConnectionProfile -NetworkCategory Private
'@
Set-Content -Path "$scriptDir\Set-NetworkPrivate.ps1" -Value $script

$regPath = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Group
    Policy\Scripts\Startup\0\0"
New-Item -Path $regPath -Force
Set-ItemProperty -Path $regPath -Name "Script" -Value
    "$scriptDir\Set-NetworkPrivate.ps1"
Set-ItemProperty -Path $regPath -Name "Parameters" -Value ""
Set-ItemProperty -Path $regPath -Name "IsPowershell" -Value 1 -Type DWord
Set-ItemProperty -Path $regPath -Name "ExecTime" -Value ([byte[]](0x00 * 8)) -Type
    Binary

$regParent = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Group
    Policy\Scripts\Startup\0"
Set-ItemProperty -Path $regParent -Name "GPO-ID" -Value "LocalGPO"
Set-ItemProperty -Path $regParent -Name "SOM-ID" -Value "Local"
Set-ItemProperty -Path $regParent -Name "FileSysPath" -Value
    "C:\Windows\System32\GroupPolicy\Machine"
Set-ItemProperty -Path $regParent -Name "DisplayName" -Value "Local Group Policy"
Set-ItemProperty -Path $regParent -Name "GPOName" -Value "Local Group Policy"
Set-ItemProperty -Path $regParent -Name "PSScriptOrder" -Value 1 -Type DWord

$statePath = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Group
    Policy\State\Machine\Scripts\Startup\0\0"
New-Item -Path $statePath -Force
Set-ItemProperty -Path $statePath -Name "Script" -Value
    "$scriptDir\Set-NetworkPrivate.ps1"
Set-ItemProperty -Path $statePath -Name "Parameters" -Value ""
Set-ItemProperty -Path $statePath -Name "ExecTime" -Value ([byte[]](0x00 * 8)) -Type
    Binary

$stateParent = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Group
    Policy\State\Machine\Scripts\Startup\0"
Set-ItemProperty -Path $stateParent -Name "GPO-ID" -Value "LocalGPO"
Set-ItemProperty -Path $stateParent -Name "SOM-ID" -Value "Local"
Set-ItemProperty -Path $stateParent -Name "FileSysPath" -Value
    "C:\Windows\System32\GroupPolicy\Machine"
Set-ItemProperty -Path $stateParent -Name "DisplayName" -Value "Local Group Policy"
Set-ItemProperty -Path $stateParent -Name "GPOName" -Value "Local Group Policy"
Set-ItemProperty -Path $stateParent -Name "PSScriptOrder" -Value 1 -Type DWord

gpupdate /force
```

For Windows 7, the `Get-NetConnectionProfile` cmdlet does not exist. Use the NLM COM object instead. The script content is different but the GPO registration commands are identical:

```
$scriptDir = "C:\Windows\System32\GroupPolicy\Machine\Scripts\Startup"
New-Item -Path $scriptDir -ItemType Directory -Force
$script = @'
Start-Sleep -Seconds 5
$nlm = [Activator]::CreateInstance(
    [Type]::GetTypeFromCLSID('DCB00C01-570F-4A9B-8D69-199FDBA5723B'))
$connections = $nlm.GetNetworkConnections()
foreach ($c in $connections) {
    $c.GetNetwork().SetCategory(1)
}
'@
Set-Content -Path "$scriptDir\Set-NetworkPrivate.ps1" -Value $script

$regPath = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Group
    Policy\Scripts\Startup\0\0"
New-Item -Path $regPath -Force
Set-ItemProperty -Path $regPath -Name "Script" -Value
    "$scriptDir\Set-NetworkPrivate.ps1"
Set-ItemProperty -Path $regPath -Name "Parameters" -Value ""
Set-ItemProperty -Path $regPath -Name "IsPowershell" -Value 1 -Type DWord
Set-ItemProperty -Path $regPath -Name "ExecTime" -Value ([byte[]](0x00 * 8)) -Type
    Binary

$regParent = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Group
    Policy\Scripts\Startup\0"
Set-ItemProperty -Path $regParent -Name "GPO-ID" -Value "LocalGPO"
Set-ItemProperty -Path $regParent -Name "SOM-ID" -Value "Local"
Set-ItemProperty -Path $regParent -Name "FileSysPath" -Value
    "C:\Windows\System32\GroupPolicy\Machine"
Set-ItemProperty -Path $regParent -Name "DisplayName" -Value "Local Group Policy"
Set-ItemProperty -Path $regParent -Name "GPOName" -Value "Local Group Policy"
Set-ItemProperty -Path $regParent -Name "PSScriptOrder" -Value 1 -Type DWord

$statePath = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Group
    Policy\State\Machine\Scripts\Startup\0\0"
New-Item -Path $statePath -Force
Set-ItemProperty -Path $statePath -Name "Script" -Value
    "$scriptDir\Set-NetworkPrivate.ps1"
Set-ItemProperty -Path $statePath -Name "Parameters" -Value ""
Set-ItemProperty -Path $statePath -Name "ExecTime" -Value ([byte[]](0x00 * 8)) -Type
    Binary

$stateParent = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Group
    Policy\State\Machine\Scripts\Startup\0"
Set-ItemProperty -Path $stateParent -Name "GPO-ID" -Value "LocalGPO"
Set-ItemProperty -Path $stateParent -Name "SOM-ID" -Value "Local"
Set-ItemProperty -Path $stateParent -Name "FileSysPath" -Value
    "C:\Windows\System32\GroupPolicy\Machine"
Set-ItemProperty -Path $stateParent -Name "DisplayName" -Value "Local Group Policy"
Set-ItemProperty -Path $stateParent -Name "GPOName" -Value "Local Group Policy"
Set-ItemProperty -Path $stateParent -Name "PSScriptOrder" -Value 1 -Type DWord

gpupdate /force
```

5.5 Install OpenSSH Server

Install via **Settings** → **Apps** → **Optional Features** → **Add a feature** → search for **OpenSSH Server** → Install.

Alternatively, via PowerShell:

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

Configure the service to start automatically and start it:

```
Set-Service sshd -Status Running -StartupType Automatic
```

Verify:

```
Get-Service | Where-Object ServiceName -Like "sshd*" | Format-List -Property
    DisplayName, ServiceName, Status, StartType
```

5.6 Install Sysmon

Sysmon is located in the user's Downloads directory. Install from an **elevated cmd.exe shell** (not PowerShell — PowerShell may not pass arguments correctly to Sysmon):

```
cd %USERPROFILE%\Downloads
Sysmon\Sysmon64.exe -accepteula -i
```

Verify the service is running and set to automatic:

```
Set-Service Sysmon64 -Status Running -StartupType Automatic
Get-Service | Where-Object ServiceName -Like "Sysmon*" | Format-List -Property
    DisplayName, ServiceName, Status, StartType
```

Note: Sysmon copies itself to `C:\Windows\Sysmon64.exe` during installation and runs the service from there. The original files in the Downloads directory are not needed after installation and can be deleted. You can verify the installed path with `sc qc Sysmon64`.

Important Notes:

- Do **not** install with a configuration file at template time. Configuration files will be applied during the testing process on cloned VMs.
- Downloaded files may be blocked by Windows. Before installation, unblock all files:

```
Get-ChildItem -Path "$env:USERPROFILE\Downloads\Sysmon" -Recurse | Unblock-File
```

- **Older OS Note**: The current version of Sysmon does not support all Windows versions. Older operating systems such as Windows 7 require an older version of Sysmon. Version 10.42 is confirmed working for Win7 (see Section 1.1 for download link).

5.7 Sysprep

Run Sysprep to generalize the image so clones get unique SIDs:

```
C:\Windows\System32\Sysprep\sysprep.exe /generalize /oobe /shutdown
```

Critical: After Sysprep shuts down the VM, do **not** boot it again. Booting triggers the OOBE setup, which regenerates the SID and undoes the generalization. Proceed directly to converting to a template.

If you accidentally boot the VM after Sysprep, let the OOBE complete (set up a temp user, skip through prompts), then re-run Sysprep and wait for shutdown.

If Sysprep fails with errors about applications that cannot be removed during generalization, see **Appendix C** for the procedure to identify and remove blocking applications. This is common on Windows 10 and 11.

---

6. Convert to Template and Clone

6.1 Convert to Template

After Sysprep has shut down the VM, in the Proxmox GUI:

Right-click the VM → Convert to template

This is irreversible — the VM becomes read-only and can only be cloned from.

Post-conversion:

- **Notes**: In the template's Notes field, record what's installed: VirtIO driver version, Sysmon version, Windows patch level, NIC type (VirtIO or E1000), and the date. You will forget.
- **Protection**: In **Options → Protection**, enable it to prevent accidental deletion.

6.2 Clone VMs from Templates

In the Proxmox GUI: **Right-click the template → Clone**

Setting	Value
Target VM ID	Per your numbering scheme
Name	Descriptive name (e.g., `dev-win10-01`)
Mode	**Full Clone**
Target Pool	Select the appropriate pool

Always use **Full Clone** for standalone systems. Linked clones share the template's disk and create a dependency — the template can never be deleted, and the clone cannot be moved to different storage.

6.3 Configure Cloned VMs

After cloning, before first boot:

1. **Network**: Change the network bridge from vmbr0 to **vmbr1** in the Hardware tab
2. **Boot the VM** and complete the Windows OOBE setup
3. **Set a static IP** on the dev network (192.168.100.0/24)
4. Assign a hostname

6.4 Updating Templates

Templates cannot be booted or modified. To update a template:

1. Clone the template (full clone)
2. Boot the clone and apply updates
3. Re-run Sysprep (`/generalize /oobe /shutdown`)
4. Convert the updated clone to a new template
5. Delete the old template

7. Proxmox API Configuration

The Proxmox API enables remote management of VMs — starting, stopping, and querying status — without logging into the GUI or SSH-ing to the host.

7.1 Create an API Token (Proxmox Host)

In the Proxmox GUI: **Datacenter** → **Permissions** → **API Tokens** → **Add**

Setting	Value
User	root@pam (or a dedicated service account)
Token ID	A descriptive name (e.g., `devlab`)
Privilege Separation	**Unchecked** (token inherits user permissions)

Save the token secret immediately — it is shown only once. The full token string is: `<user>!<token-id>=<secret>` (e.g., `root@pam!devlab=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`).

7.2 Test API Access from the Dev Server

```
# Check cluster status (200 = success, 401 = bad token, 403 = insufficient permissions)
curl -s -k -o /dev/null -w "%{http_code}\n" \
  -H 'Authorization: PVEAPIToken=root@pam!devlab=&lt;TOKEN_SECRET&gt;' \
  "https://&lt;PROXMOX_HOST_IP&gt;:8006/api2/json/cluster/status"
```

The `-k` flag skips certificate verification (the Proxmox self-signed cert). For production, configure a trusted certificate.

7.3 Common API Operations

Replace `<NODE>` with your Proxmox node name and `<VMID>` with the target VM ID.

Start a VM:

```
curl -s -k \
  -H 'Authorization: PVEAPIToken=root@pam!devlab=&lt;TOKEN_SECRET&gt;' \
  -X POST \
  "https://&lt;PROXMOX_HOST_IP&gt;:8006/api2/json/nodes/&lt;NODE&gt;/qemu/&lt;VMID&gt;/status/start"
```

Stop a VM (graceful shutdown):

```
curl -s -k \
  -H 'Authorization: PVEAPIToken=root@pam!devlab=&lt;TOKEN_SECRET&gt;' \
  -X POST \
  "https://&lt;PROXMOX_HOST_IP&gt;:8006/api2/json/nodes/&lt;NODE&gt;/qemu/&lt;VMID&gt;/status/shutdown"
```

Check VM status:

```
curl -s -k \
  -H 'Authorization: PVEAPIToken=root@pam!devlab=&lt;TOKEN_SECRET&gt;' \
  "https://&lt;PROXMOX_HOST_IP&gt;:8006/api2/json/nodes/&lt;NODE&gt;/qemu/&lt;VMID&gt;/status/current"
```

List all VMs:

```
curl -s -k \
  -H 'Authorization: PVEAPIToken=root@pam!devlab=&lt;TOKEN_SECRET&gt;' \
```

```
  "https://&lt;PROXMOX_HOST_IP&gt;:8006/api2/json/nodes/&lt;NODE&gt;/qemu"
```

7.4 Helper Script for VM Management

Save this on the dev server for convenience:

```
#!/bin/bash
# proxmox-vm.sh — Start, stop, and check Proxmox VMs
# Usage: ./proxmox-vm.sh &lt;start|stop|status&gt; &lt;VMID&gt;

PROXMOX_HOST="&lt;PROXMOX_HOST_IP&gt;"
PROXMOX_NODE="&lt;NODE&gt;"
TOKEN='PVEAPIToken=root@pam!devlab=&lt;TOKEN_SECRET&gt;'
BASE_URL="https://${PROXMOX_HOST}:8006/api2/json/nodes/${PROXMOX_NODE}/qemu"

ACTION=$1
VMID=$2

if [ -z "$ACTION" ] || [ -z "$VMID" ]; then
    echo "Usage: $0 &lt;start|stop|status&gt; &lt;VMID&gt;"
    exit 1
fi

case $ACTION in
    start)
        curl -s -k -H "Authorization: $TOKEN" -X POST "${BASE_URL}/${VMID}/status/start"
        echo ""
        echo "Starting VM ${VMID}..."
        ;;
    stop)
        curl -s -k -H "Authorization: $TOKEN" -X POST
    "${BASE_URL}/${VMID}/status/shutdown"
        echo ""
        echo "Shutting down VM ${VMID}..."
        ;;
    status)
        curl -s -k -H "Authorization: $TOKEN" "${BASE_URL}/${VMID}/status/current" |
    python3 -m json.tool
        ;;
    *)
        echo "Unknown action: $ACTION"
        echo "Usage: $0 &lt;start|stop|status&gt; &lt;VMID&gt;"
        exit 1
        ;;
esac
```

8. SSH Configuration

SSH is used from the dev server to execute commands on Windows VMs — testing Sysmon configurations, running scripts, etc.

8.1 Generate SSH Keys on the Dev Server

If you don't already have an SSH key pair:

```
ssh-keygen -t ed25519 -C "devlab"
```

8.2 Configure SSH Client on the Dev Server

Create or edit `~/.ssh/config` on the dev server to define named hosts and shared settings:

```
# Proxmox Host
Host proxmox
    HostName &lt;PROXMOX_HOST_IP&gt;
    User root
    IdentityFile ~/.ssh/id_ed25519
    StrictHostKeyChecking accept-new

# Dev VMs
Host Win7Pro-Dev
    HostName 192.168.100.100

Host Win10Pro-Dev
    HostName 192.168.100.101

Host Win11Pro-Dev
    HostName 192.168.100.102

Host WinServer2016-Dev
    HostName 192.168.100.103

Host WinServer2019-Dev
    HostName 192.168.100.104

Host WinServer2022-Dev
    HostName 192.168.100.105

# Shared settings for all dev VMs
Host *-Dev
    User Administrator
    IdentityFile ~/.ssh/id_ed25519
    StrictHostKeyChecking accept-new
    UserKnownHostsFile /dev/null
```

Notes on these settings:

- **Named hosts** allow you to use `ssh Win10Pro-Dev` instead of remembering IP addresses
- `StrictHostKeyChecking accept-new` accepts keys on first connection but warns if they change unexpectedly
- `UserKnownHostsFile /dev/null` for dev VMs prevents stale host key errors when clones are rebuilt with new keys after Sysprep. This is not set for the Proxmox host, which should retain its known key.
- `Host *-Dev` **wildcard** applies shared settings to all hosts ending in `-Dev`. Add new VMs with just a two-line entry (Host and HostName).

8.3 Copy SSH Key to Windows VMs

Windows OpenSSH uses a different authorized_keys location for administrators.

On each Windows VM, create the authorized keys file:

```
mkdir C:\ProgramData\ssh
```

Copy the contents of `~/.ssh/id_ed25519.pub` from the dev server into `C:\ProgramData\ssh\administrators_authorized_keys` on the Windows VM.

Set the correct permissions on the file (from an elevated PowerShell):

```
icacls "C:\ProgramData\ssh\administrators_authorized_keys" /inheritance:r /grant
    "Administrators:F" /grant "SYSTEM:F"
```

Verify the SSH config at `C:\ProgramData\ssh\sshd_config` contains (uncommented):

```
PubkeyAuthentication yes
AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authorized_keys
```

Restart the SSH service:

```
Restart-Service sshd
```

8.4 Test SSH from the Dev Server

```
ssh Win10Pro-Dev
```

You should connect without a password prompt.

8.5 SSH to Proxmox Host (Optional)

For VM management via `qm` commands instead of the API:

```
# Copy SSH key to Proxmox host
ssh-copy-id proxmox

# Start a VM
ssh proxmox "qm start &lt;VMID&gt;"

# Shutdown a VM and wait
ssh proxmox "qm shutdown &lt;VMID&gt; --timeout 120 &amp;&amp; qm wait &lt;VMID&gt; --timeout 120"
```

---

9. Automated VM Test Workflow

For sequential testing across multiple VMs (e.g., testing a Sysmon config across all Windows versions), use the following workflow. This is designed for resource-constrained hosts where only one VM should run at a time.

9.1 Sequential Test Script

```
#!/bin/bash
# test-all-vms.sh — Run a test script against each VM sequentially
# Usage: ./test-all-vms.sh &lt;script-to-run&gt;

PROXMOX_HOST="&lt;PROXMOX_HOST_IP&gt;"
PROXMOX_NODE="&lt;NODE&gt;"
TOKEN='PVEAPIToken=root@pam!devlab=&lt;TOKEN_SECRET&gt;'
BASE_URL="https://${PROXMOX_HOST}:8006/api2/json/nodes/${PROXMOX_NODE}/qemu"

TEST_SCRIPT=$1

if [ -z "$TEST_SCRIPT" ]; then
    echo "Usage: $0 &lt;script-to-run&gt;"
    exit 1
fi

# Define VMs: "VMID SSH_HOST DESCRIPTION"
VMS=(
    "1000 Win7Pro-Dev Win7-Pro"
    "1001 Win10Pro-Dev Win10-Pro"
    "1002 Win11Pro-Dev Win11-Pro"
    "1003 WinServer2016-Dev Server-2016"
    "1004 WinServer2019-Dev Server-2019"
    "1005 WinServer2022-Dev Server-2022"
)

wait_for_vm_ready() {
    local host=$1
    local max_attempts=60
    local attempt=0
    echo "  Waiting for SSH on ${host}..."
    while [ $attempt -lt $max_attempts ]; do
        if ssh -o ConnectTimeout=5 ${host} "echo ready" &amp;&gt;/dev/null; then
            echo "  VM is ready."
            return 0
        fi
        attempt=$((attempt + 1))
        sleep 5
    done
    echo "  ERROR: VM ${host} did not become ready."
    return 1
}

wait_for_vm_stopped() {
    local vmid=$1
    local max_attempts=60
    local attempt=0
    echo "  Waiting for VM ${vmid} to stop..."
    while [ $attempt -lt $max_attempts ]; do
        STATUS=$(curl -s -k -H "Authorization: $TOKEN" \
            "${BASE_URL}/${vmid}/status/current" | python3 -c "import sys,json;
    print(json.load(sys.stdin)['data']['status'])")
        if [ "$STATUS" = "stopped" ]; then
            echo "  VM ${vmid} is stopped."
            return 0
        fi
        attempt=$((attempt + 1))
        sleep 5
    done
    echo "  ERROR: VM ${vmid} did not stop in time."
    return 1
}
```

```
for VM_ENTRY in "${VMS[@]}"; do
    read -r VMID SSH_HOST DESC &lt;&lt;&lt; "$VM_ENTRY"
    echo "============================================="
    echo "Testing: ${DESC} (VM ${VMID} via ${SSH_HOST})"
    echo "============================================="

    # Start VM
    curl -s -k -H "Authorization: $TOKEN" -X POST "${BASE_URL}/${VMID}/status/start" &gt;
    /dev/null
    wait_for_vm_ready "$SSH_HOST" || continue

    # Run the test script
    echo "  Running test: ${TEST_SCRIPT}"
    ssh ${SSH_HOST} &lt; "$TEST_SCRIPT"
    echo "  Test complete."

    # Shutdown VM
    curl -s -k -H "Authorization: $TOKEN" -X POST "${BASE_URL}/${VMID}/status/shutdown"
    &gt; /dev/null
    wait_for_vm_stopped "$VMID"

    echo ""
done

echo "All tests complete."
```

9.2 Example: Deploy and Test a Sysmon Config

Create a test script (e.g., `deploy-sysmon-config.ps1`):

```
# deploy-sysmon-config.ps1 — Deploy a Sysmon config and verify
param()

# Copy the config file (assumes it was SCP'd to the VM first)
$configPath = "C:\Users\Administrator\sysmon-test-config.xml"

# Check current Sysmon schema version
$schemaLine = cmd /c "C:\Windows\Sysmon64.exe -s" | Select-String "SchemaVersion"
Write-Host "Sysmon Schema: $schemaLine"

# Apply the config
cmd /c "C:\Windows\Sysmon64.exe -c `"$configPath`""

# Verify rules loaded
cmd /c "C:\Windows\Sysmon64.exe -c"

# Check for events
Get-WinEvent -LogName "Microsoft-Windows-Sysmon/Operational" -MaxEvents 5 | Format-List
    TimeCreated, Id, Message
```

Run it:

```
# First SCP the config file to the VM
scp my-sysmon-config.xml Win10Pro-Dev:C:/Users/Administrator/sysmon-test-config.xml

# Then run the test script
ssh Win10Pro-Dev &lt; deploy-sysmon-config.ps1
```

9.3 Claude Code Integration

To use Claude Code for automated testing, include the following context in the project's `CLAUDE.md`:

```
## VM Test Environment
```

```
- Proxmox host: &lt;PROXMOX_HOST_IP&gt; (SSH as root or use API)
- Proxmox API token: stored in environment variable PROXMOX_TOKEN
- Only run ONE VM at a time — host resource constraint
- Workflow: start VM → wait for SSH → run tests → shutdown → next VM

### VM Inventory
| VMID | IP | OS |
|------|----|----|
| 1000 | 192.168.100.100 | Windows 7 Pro |
| 1001 | 192.168.100.101 | Windows 10 Pro |
| 1002 | 192.168.100.102 | Windows 11 Pro |
| 1003 | 192.168.100.103 | Server 2016 |
| 1004 | 192.168.100.104 | Server 2019 |
| 1005 | 192.168.100.105 | Server 2022 |

### SSH Access
- User: Administrator
- Auth: SSH key (no password)
- Sysmon location: C:\Windows\Sysmon64.exe (installed from Downloads, runs from
    C:\Windows)

### Sysmon Notes
- Install/config commands MUST be run via cmd.exe, not PowerShell
- To run cmd commands over SSH: cmd /c "command here"
- Install without config: Sysmon64.exe -accepteula -i
- Apply config separately: Sysmon64.exe -c config.xml
- Config schema version must match Sysmon version (check with: Sysmon64.exe -s)
```

Appendix A: Troubleshooting

Windows 7 — Cannot find hard drive during install

Load the VirtIO SCSI driver during setup: **Load driver** → **Browse** → **VirtIO CD** → `vioscsi\w7\amd64`

Windows 7 — Network not working with VirtIO NIC

Change the NIC to **Intel E1000** in the VM's Hardware tab.

Windows 7 — IE certificate errors / sites won't load

Fresh Win7 has outdated root certificates and defaults to TLS 1.0. Enable TLS 1.2 in IE: Internet Options → Advanced → Security → check **Use TLS 1.2**. See Section 4.4 for the full bootstrap process.

Windows 7 — Windows Update error 80072EFE

Windows Update cannot negotiate TLS 1.2. Install KB3020369, KB3125574, and KB3140245 manually, then apply the TLS registry keys. See Section 4.4.

Windows 11 — Does not meet minimum hardware requirements

Ensure TPM 2.0 and Secure Boot (Pre-Enroll Keys) are configured per Section 4.2. If issues persist, press Shift+F10 at the install screen and run:

```
setup.exe /product server
```

This bypasses the hardware compatibility checks while still installing the selected Win11 edition.

Windows Server — Installs to command prompt only (Server Core)

Reinstall and select the **(Desktop Experience)** edition variant.

Sysmon — Installs but no rules load

Sysmon — Silent failure when installing with config

Use cmd.exe instead of PowerShell. PowerShell may not pass arguments correctly to Sysmon. Install in two steps: first `Sysmon64.exe -accepteula -i`, then `Sysmon64.exe -c config.xml`.

Sysmon — Installer does not run / "not applicable"

Ensure the Sysmon executable and related files are not blocked by Windows. Unblock with:

```
Get-ChildItem -Path "&lt;sysmon-directory&gt;" -Recurse | Unblock-File
```

VMs on dev network cannot reach internet

Verify IP forwarding and NAT are configured on the Proxmox host. See Section 2.2. Ensure the VM's DNS is set to 8.8.8.8 (not the gateway).

Dev server cannot reach VMs on 192.168.100.0/24

Add a static route on the dev server pointing to the Proxmox host. See Section 2.3.

Appendix B: Windows 11 EFI Clone Boot Failure

If a Win11 clone fails to boot with `winload.efi error code 0xc00000f`, the EFI boot entry is referencing the wrong disk after cloning.

Fix procedure:

1. Attach the Win11 ISO and the VirtIO ISO to the clone
2. Boot from the ISO → **Repair your computer** → **Command Prompt**
3. Load the VirtIO SCSI driver so the repair environment can see the disk:

```
drvload D:\vioscsi\w11\amd64\vioscsi.inf
```

(Check drive letters with `wmic logicaldisk get caption` — the VirtIO ISO may be on a different letter)

4. Rebuild the boot configuration:

```
diskpart
list volume
select volume &lt;EFI_VOLUME_NUMBER&gt;
assign letter=S
exit

bcdboot C:\Windows /s S: /f UEFI
```

5. Remove the ISOs and reboot

If the disk is not visible even after loading the VirtIO driver, BitLocker likely encrypted the disk before Sysprep. The template must be rebuilt with BitLocker disabled immediately after install (see Section 4.2).

Appendix C: Sysprep Blocked Applications (Windows 10/11)

Windows 10 and 11 may have pre-installed applications that cannot be removed during the Sysprep generalization process. When Sysprep fails, it logs the blocking application in its log file.

Identifying the Blocking Application

Open the Sysprep log file:

```
C:\Windows\System32\Sysprep\Panther\setupact.txt
```

This file uses a column-delimited format: a timestamp (`YYYY-MM-DD HH:MM:SS`), followed by the entry type (`Info`, `Warning`, or `Error`), a hex error code in square brackets (e.g., `[0x0f0036]`), the process name (`SYSPRP`), and an error message.

Look for the **first** `Error` **entry** — this identifies the application blocking generalization. The error message contains the application name followed by a version number (e.g., `Microsoft.StartExperiencesApp_1.1.7.0_x64__2wekyb`).

Note: If the application noted is `Sysprep` itself, read the full error message carefully. This typically means the system needs a pending update or requires a reboot to apply updates before Sysprep can proceed.

Removing the Blocking Application

Extract the app name prefix from the error message (e.g., `StartExperiencesApp`) and run both commands, replacing `XXXX` with the app name:

```
# Remove for all users
Get-AppxPackage -AllUsers *XXXX* | Remove-AppxPackage

# Remove the provisioned package so it doesn't reinstall
Get-AppxProvisionedPackage -Online | Where-Object DisplayName -like "*XXXX*" |
    Remove-AppxProvisionedPackage -Online
```

Re-run Sysprep after each removal. You may need to repeat this process multiple times for different applications. Keep a record of removed applications as they may need to be reinstalled on deployed clones.

Appendix D: Windows Server 2016 — PowerShell TLS 1.2 Configuration

Windows Server 2016's PowerShell defaults to TLS 1.0 for .NET-based connections (e.g., `Invoke-WebRequest`, `Install-Module`, NuGet). Many endpoints now require TLS 1.2, causing silent failures or connection errors. Apply these registry settings to enable TLS 1.2 system-wide for .NET applications:

```
# 64-bit .NET Framework 4.x
Set-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319' -Name
    'SystemDefaultTlsVersions' -Value 1 -Type DWord
Set-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319' -Name
    'SchUseStrongCrypto' -Value 1 -Type DWord

# 32-bit .NET Framework 4.x (required for many background processes)
Set-ItemProperty -Path 'HKLM:\SOFTWARE\WOW6432Node\Microsoft\.NETFramework\v4.0.30319'
    -Name 'SystemDefaultTlsVersions' -Value 1 -Type DWord
Set-ItemProperty -Path 'HKLM:\SOFTWARE\WOW6432Node\Microsoft\.NETFramework\v4.0.30319'
    -Name 'SchUseStrongCrypto' -Value 1 -Type DWord
```

Reboot after applying. This should be done during the template build process before running Windows Update or installing any software that requires network access from PowerShell.