ADMIN:DOCS

Windows Server 2025 Core: Installation und Administration über SSH und PowerShell

Windows Server 2025 Core unter Proxmox VE 9 ohne Desktop-GUI betreiben: Von der VM-Erstellung und SConfig-Erstkonfiguration über OpenSSH bis zur PowerShell-Verwaltung.

Kategorie: Windows Server **Veröffentlicht:** 10.11.2026

Windows-Server binden auf einem Virtualisierungs-Host wie Proxmox VE 9 mit einer vollständigen Desktop-Oberfläche (Desktop Experience) erhebliche Ressourcen: Mehrere Gigabyte zusätzlicher Arbeitsspeicher, dauerhafte CPU-Grundlast für Grafik- und Fenstermanager-Subsysteme sowie eine unnötig große virtuelle Festplatte für Bibliotheken, die im Headless-Betrieb unberührt bleiben. Hinzu kommt eine breitere Angriffsfläche, die regelmäßig zu umfangreicheren monatlichen Patch-Zyklen und entsprechenden Neustarts führt.

Windows Server 2025 Core verzichtet auf den Windows-Explorer und alle lokalen grafischen Verwaltungswerkzeuge. Das System startet direkt in eine minimale textbasierte Konsole. Für heterogene Umgebungen bedeutet das: Ein Windows-Server lässt sich mit denselben betrieblichen Mustern betreiben wie eine Linux-Instanz – headless, ressourcenschonend und über standardisierte Netzwerkprotokolle steuerbar.

Im Zusammenspiel mit Proxmox VE 9 und Linux-basierten Administrations-Clients schließt OpenSSH unter Windows unter Windows Server 2025 die bisherige Lücke zwischen beiden Welten. Der SSH-Server ist in dieser Version standardmäßig im Betriebssystem integriert und erfordert keine komplexen WinRM-HTTPS-Zertifikatsketten mehr. Der Fernzugriff erfolgt direkt über den gewohnten OpenSSH-Client aus der Linux-Kommandozeile in eine native Windows-PowerShell-Sitzung.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-installation-und-administration

Das Ziel dieser Bereitstellung ist eine schlanke, überprüfbare Server-Core-Instanz auf Proxmox VE 9

Initial über SConfig netzwerkseitig eingerichtet, mit persistentem OpenSSH-Dienst abgesichert und anschließend ausschließlich über PowerShell aus dem Terminal verwaltet.

> **Voraussetzungen und Testumgebung:** Eine reproduzierbare Bereitstellung von Windows Server Core setzt ein klares Verständnis der Zielgruppe, der zugrunde liegenden Hypervisor-Hardware und der Unterschiede zwischen Mindestanforderungen und praxistauglichen Laborressourcen voraus.

Zielgruppe und Vorkenntnisse

Dieser Artikel richtet sich an Administratoren, die bereits mit grundlegenden Aufgaben des Systembetriebs vertraut sind:

- Verständnis von Betriebssystemkonzepten, Prozessen, Benutzerkonten und Dienstberechtigungen
- Praktischer Umgang mit IP-Netzwerken, Subnetzen, Routing und Namensauflösung
- Vertrautheit mit Virtualisierungsplattformen wie Proxmox VE oder KVM-basiertem Hypervisor- und VM-Management
- Grundlegende Bedienung einer administrativen Befehlszeile

Erfahrungen mit Linux und dem OpenSSH-Client erleichtern den Einstieg bei der Fernverwaltung, sind jedoch nicht zwingend erforderlich. Spezifische Vorkenntnisse in Windows Server Core oder der PowerShell-Syntax werden nicht vorausgesetzt; alle administrativen Schritte und Cmdlets werden an den jeweiligen Arbeitsabschnitten hergeleitet und überprüft.

Systemanforderungen und Labor-Empfehlungen

Microsoft definiert für Windows Server 2025 offizielle Mindestanforderungen, die für den reinen Start des Kernels genügen, für einen stabilen Betrieb unter Hypervisoren jedoch nicht als Maßstab dienen sollten:

Offizielle Mindestanforderung

Prozessor
: 1,4 GHz x64 (x86-64-v2/v3)

Arbeitsspeicher
: 2 GiB RAM (Server Core)

Massenspeicher
: 32 GiB Festplatte

Firmware & Typ
: UEFI 2.3.1c (Secure Boot)

Netzwerk
: 1 Gbit/s Ethernet

Gast-Tools
: Nicht spezifiziert

Empfohlene Laborkonfiguration (PVE)

Prozessor
: 2 bis 4 vCPUs (Typ host)

Arbeitsspeicher
: 4 bis 8 GiB RAM

Massenspeicher
: 64 GiB (VirtIO-SCSI)

Firmware & Typ
: OVMF (UEFI), Maschinentyp Q35

Netzwerk
: VirtIO (paravirtualisiert)

Gast-Tools
: QEMU Guest Agent (qemu-ga)

Gemeinsam: 64-Bit-Architektur, lauffähig als KVM-Gast unter Proxmox VE 9

Die Wahl des Storage-Controllers und des Netzwerkadapters hat unmittelbare Auswirkungen auf die Installation: Während paravirtualisierte VirtIO-Geräte den geringsten Overhead und den höchsten Datendurchsatz bieten, bringt das Standard-Installationsmedium von Windows Server keine passenden VirtIO-Treiber mit. Diese müssen während des Setups über die Treiber-ISO `virtio-win` nachgeladen werden.

Dokumentierte Testkonfiguration

Die im weiteren Verlauf beschriebenen Arbeitsschritte und Befehlsausgaben basieren auf einer real durchgeführten Installation auf einem Proxmox-VE-Host. Um Treiberabhängigkeiten während der anfänglichen Setup-Phase bewusst zu minimieren, kam hierbei eine Konfiguration mit emulierter SATA- und Intel-Hardware zum Einsatz:

Komponente	Konfiguration
Hypervisor	Proxmox VE 9
Gastbetriebssystem	Windows Server 2025 Standard Evaluation (x64, Deutsch)
Installationsvariante	Server Core (ohne Desktop Experience)
Maschinentyp & Firmware	Q35, OVMF (UEFI) mit EFI-Disk
Prozessor	6 vCPU, Typ `host`
Arbeitsspeicher	14.096 MiB
Systemlaufwerk	150 GiB, Schnittstelle: SATA
Netzwerkadapter	Intel E1000
Treiber-Medium	virtio-win-0.1.302 (für spätere Gast-Tools)
Administrationsclient	Linux-Workstation mit OpenSSH-Client

> □□ **Hardware-Emulation vs. Paravirtualisierung:** Der Einsatz von SATA und Intel E1000 erlaubt eine Installation ohne Unterbrechung, da Windows Server 2025 Standardtreiber für beide Komponenten mitbringt. In produktiven Umgebungen auf

> Proxmox VE wird jedoch der Einsatz von VirtIO-SCSI und VirtIO-Netzwerk empfohlen, um Emulations-Overhead zu vermeiden. Die manuelle Treibereinbindung wird im Installationskapitel transparent gegenübergestellt.

Die zugewiesenen 14 GiB Arbeitsspeicher und 6 vCPUs spiegeln die Kapazitäten des vorhandenen Testsystems wider und stellen keine Mindestvoraussetzung dar. Spezifische Host-Anpassungen (wie SMBIOS-Modifikationen oder feste CPU-Affinitäten) sind für den regulären Betrieb nicht erforderlich und werden bewusst ausgelassen.

Netzwerktopologie und Port-Matrix für die Administration

In heterogenen Netzwerken, in denen Linux-Workstations, Proxmox-Cluster und Windows-Server über Firewalls und VLANs getrennt sind, müssen die administrativen Ports exakt definiert werden. Auf Windows Server Core entfallen browserbasierte Konfigurationsschnittstellen; der gesamte Datenverkehr läuft über standardisierte Netzwerkprotokolle:

Port	Protokoll	Dienst / Zweck	Notwendig für
22	TCP	OpenSSH Server (sshd)	Primäre Fernverwaltung aus dem Linux-Terminal und Ansible
53	UDP/TCP	Domain Name System (DNS)	Namensauflösung im lokalen Netz und Active Directory
88	UDP/TCP	Kerberos v5	Domänen-Authentifizierung bei Active-Directory-Beitritt
123	UDP	Network Time Protocol (W32Time)	Zeitsynchronisation zur Vermeidung von Kerberos-Drifts
135	TCP	RPC Endpoint Mapper	Notwendig für Remoteverwaltung über RSAT und WMI
389 / 636	TCP	LDAP / LDAPS	Active Directory Benutzer- und Gruppenabfragen
445	TCP	Server Message Block (SMB)	Dateiübertragungen, Skriptverteilung und Named Pipes
3389	TCP/UDP	Remote Desktop Protocol (RDP)	Optionale Textkonsolen-Sitzung über RDP-Clients

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-installation-und-administration

Port	Protokoll	Dienst / Zweck	Notwendig für
5985	TCP	WinRM HTTP	PowerShell Remoting (Nachrichtenverschlüsselung via Kerberos/NTLM; Transport ohne TLS)
5986	TCP	WinRM HTTPS	PowerShell Remoting mit TLS-Transportverschlüsselung und Serverzertifikat
8006	TCP	Proxmox VE Web-GUI	Host-Management und noVNC-Konsolenfenster

Speicherplatz- und Partitionsplanung im Detail

Im Vergleich zu einer Vollinstallation mit Desktop Experience (die nach der Installation mindestens 24 bis 28 GiB belegt) benötigt ein frisches Windows Server 2025 Core System signifikant weniger Speicherplatz:

- **Windows-Basissystem (`C:\Windows`):** ca. 11 bis 13 GiB.
- **Component Store (`C:\Windows\WinSxS`):** ca. 5 bis 7 GiB für Systemkomponenten und Patch-Backups.
- **Virtueller Arbeitsspeicher (`pagefile.sys`):** Standardmäßig dynamisch dimensioniert (entspricht bei 8 GiB RAM meist ca. 2 bis 4 GiB Auslagerungsdatei).
- **Crash-Dump-Reservierung (`MEMORY.DMP`):** Bei Kernel-Dumps bis zur vollen Größe des physischen RAMs (bei 14 GiB RAM ca. 14 GiB maximal reservierbarer Speicher).
- **Puffer für monatliche Qualitäts-Updates:** Microsoft-Update-Pakete entpacken während der Installation temporäre Archive von 4 bis 8 GiB Größe.

Für eine isolierte Systempartition unter Proxmox VE empfiehlt sich daher eine Mindestkapazität von **60 bis 80 GiB**. Dies bietet ausreichend Puffer für mehrjährige Update-Zyklen und Treiber-Backups, ohne dass das Dateisystem manuell bereinigt werden muss.

Windows Server Core verstehen

Windows Server Core ist keine eigenständige Windows-Edition mit separater Lizenzierung, sondern eine minimale Installationsoption von Windows Server 2025 Standard und Datacenter. Das Betriebssystem verzichtet vollständig auf die klassische Desktop-Oberfläche: Es gibt keinen

Windows Explorer, keinen Desktop, keine Taskleiste und kein lokales Startmenü. Beim Systemstart initialisiert der Kernel lediglich das Subsystem für Konsolenanwendungen und öffnet eine textbasierte Eingabeaufforderung.

> **Praxisorientierte Arbeitsweise:** Auf Server Core steht dir nach dem Booten ausschließlich eine Shell zur Verfügung. Schließt du versehentlich das Konsolenfenster, bleibt der Bildschirm schwarz. Über die Tastenkombination `Strg` + `Umschalt` + `Esc` (oder `Strg` + `Alt` + `Entf` in Proxmox noVNC) rufst du den Task-Manager auf und startest über *Datei → Neuen Task ausführen* mit `powershell.exe` oder `cmd.exe` jederzeit eine neue Konsole.

Architektur und Unterschiede zur Desktop Experience

In herkömmlichen Windows-Server-Installationen mit Desktop Experience ist der grafische Software-Stack tief mit dem Gesamtsystem verzahnt. Bibliotheken für Fensterverwaltung, Schriftarten-Rendering, Themes und grafische Benutzeroberflächen verbleiben permanent auf dem Dateisystem und werden beim Booten teilweise direkt in den Arbeitsspeicher geladen.

Server Core entfernt diese Komponenten fast vollständig. Binärdateien für `explorer.exe`, `mmc.exe` und zugehörige grafische Assistenten existieren auf dem System nicht. Das hat drei wesentliche administrative Konsequenzen:

- **Reduzierte Angriffsfläche:** Sicherheitslücken in Windows-Komponenten betreffen häufig clientnahe Bibliotheken wie Grafik-Engines, Web-Rendering-Module oder Shell-Erweiterungen. Auf Server Core sind diese Bibliotheken gar nicht erst vorhanden.
- **Geringerer Ressourcenbedarf:** Der Verzicht auf die Desktop-Grafik spart dauerhaft Arbeitsspeicher und I/O-Last auf dem Hypervisor. Für reine Hintergrunddienste steht die zugewiesene Host-Hardware ohne GUI-Overhead zur Verfügung.
- **Geringere Update-Frequenz:** Da monatliche kumulative Sicherheitsupdates für den grafischen Desktop-Stack entfallen oder spürbar kompakter ausfallen, sinkt die Zahl der Patches, die einen sofortigen Neustart des Servers erzwingen.

Desktop Experience

Lokale Benutzeroberfläche
: Windows Explorer, Startmenü, Taskleiste

Grafische Werkzeuge
: Lokale MMC-Snap-ins und Assistenten

Ressourcenbedarf
: Höhere RAM- und I/O-Grundlast

Wartungsaufwand
: Höhere Patch-Frequenz für GUI-Pakete

Angriffsfläche
: Umfassend durch Desktop-Bibliotheken

Server Core

Lokale Benutzeroberfläche
: Reine Textkonsole (cmd.exe / powershell.exe)

Grafische Werkzeuge
: Ausschließlich über Remote-Verwaltung nutzbar

Ressourcenbedarf
: Geringer Speicher-Footprint

Wartungsaufwand
: Weniger Update-Pakete, seltener Neustarts

Angriffsfläche
: Reduziert auf Systemkern und Serverdienste

Gemeinsam: Identischer Windows Server 2025 Kernel, identische Treiberarchitektur und Win32/CIM-APIs

Vollständige Rollen-Kompatibilitätsmatrix für Windows Server 2025 Core

Eine der wichtigsten Planungsfragen vor der Bereitstellung von Server Core betrifft die Unterstützung von Serverrollen. Nahezu alle klassischen Rechenzentrums-Dienste laufen auf Server Core out-of-the-box. Bestimmte Rollen, die zwingend lokale Endbenutzer-Sitzungen oder

Legacy-GUI-Schnittstellen erfordern, sind hingegen ausgeschlossen:

Server-Rolle	Status auf Server Core	Feature-Name (PowerShell)	Verwaltungsmethode
Active Directory Domain Services	✅ Vollständig	`AD-Domain-Services`	PowerShell (`ActiveDirectory`), Remote-RSAT
Active Directory Certificate Services	✅ Vollständig	`AD-Certificate`	PowerShell (`ADCSAdministration`), `certutil.exe`
Active Directory Federation Services	✅ Vollständig	`ADFS-Federation`	PowerShell (`ADFS`)
Active Directory Lightweight Directory Services	✅ Vollständig	`ADLDS`	PowerShell, Remote-RSAT
Active Directory Rights Management	✅ Vollständig	`ADRMS`	PowerShell, Remote-Webkonsole
DHCP-Server	✅ Vollständig	`DHCP`	PowerShell (`DhcpServer`), Remote-RSAT
DNS-Server	✅ Vollständig	`DNS`	PowerShell (`DnsServer`), `dnscmd.exe`
Datei- und Speicherdienste	✅ Vollständig	`File-Services`, `FS-FileServer`	PowerShell (`Storage`, `SmbShare`)
Webserver (IIS)	✅ Vollständig	`Web-Server`	PowerShell (`WebAdministration`), AppCmd
Hyper-V	✅ Vollständig	`Hyper-V`	PowerShell (`Hyper-V`), Hyper-V-Manager remote
Druck- und Dokumentdienste	✅ Vollständig	`Print-Server`	PowerShell (`PrintManagement`)
Windows Server Update Services (WSUS)	✅ Vollständig	`UpdateServices`	PowerShell (`UpdateServices`), SQL/WID
Remotedesktopdienste: Verbindungsbroker	✅ Vollständig	`RDS-Connection-Broker`	PowerShell (`RemoteDesktop`)

Server-Rolle	Status auf Server Core	Feature-Name (PowerShell)	Verwaltungsmethode
Remotedesktopdienste: Gateway	✅ Vollständig	`RDS-Gateway`	PowerShell, Remote-RSAT
Remotedesktopdienste: Web Access	✅ Vollständig	`RDS-Web-Access`	PowerShell, Web-Interface
Remotedesktopdienste: Session Host	❌ Nicht unterstützt	Nicht verfügbar	Zwingend Desktop Experience erforderlich
Netzwerkrichtlinien- und Zugriffsdienste (NPS)	⚠ Eingeschränkt	`NPAS`	Ausschließlich über Remote-RSAT (keine lokale GUI)
Windows Deployment Services (WDS)	⚠ Eingeschränkt	`WDS`	`wdsutil.exe`, Remote-RSAT
Faxserver	❌ Nicht unterstützt	Nicht verfügbar	Veraltet, erfordert Desktop Experience

> ⚠️ **Kein nachträglicher Wechsel mehr:** Seit Windows Server 2016 lässt sich die Benutzeroberfläche nach der Installation nicht mehr beliebig hinzufügen oder entfernen. Die Entscheidung für Server Core muss vor der Installation feststehen. Ein Wechsel erfordert eine Neuinstallation des Gastbetriebssystems.

Lokale und entfernte Administrationswerkzeuge

Ohne lokale Benutzeroberfläche stützt sich der administrative Betrieb auf eine Kombination aus textbasierten Menüs, Kommandozeilen und Remote-Protokollen:

- **SConfig (`sconfig.exe`):** Das integrierte Werkzeug für die Server-Core-Erstkonfiguration ist ein menügeführtes Konsolenprogramm, das unter Windows Server 2025 auf PowerShell basiert. Es dient der schnellen Grundeinrichtung direkt an der Hypervisor-Konsole (wie noVNC in Proxmox VE), um Hostname, Netzwerkkarten, Domänenzugehörigkeit und Windows-Update-Richtlinien ohne komplexe Befehlsketten zu setzen.

- **Windows PowerShell:** Die objektorientierte Befehlsumgebung ist das primäre Werkzeug für alle administrativen Aufgaben. Anstelle von unstrukturiertem Text verarbeitet PowerShell strukturierte .NET-Objekte über Cmdlets und die Pipeline. Lokale Konfigurationen, Dienste, Firewallregeln und Rollen werden über PowerShell gesteuert.
- **WinRM (Windows Remote Management):** Microsofts traditionelles Fernverwaltungsprotokoll basiert auf dem SOAP- und HTTP/HTTPS-basierten WS-Management-Standard (Standard-TCP-Ports 5985 und 5986). WinRM ist die technische Grundlage für Remote PowerShell (`Enter-PSSession -ComputerName`), das Windows Admin Center (WAC) sowie entfernte MMC-Snap-ins von anderen Windows-Systemen aus.
- **OpenSSH:** Der plattformoffene Standard für verschlüsselte Terminalsitzungen über Port 22. OpenSSH kapselt die Verbindung in einen standardisierten SSH-Tunnel und übergibt die interaktive Kontrolle an eine System-Shell (standardmäßig CMD oder PowerShell). Es ermöglicht den direkten Zugriff von Linux- und macOS-Clients ohne Windows-spezifische Authentifizierungsprotokolle wie Kerberos oder NTLM über HTTP.

> **Warum SSH die bessere Wahl für heterogene Netze ist:** WinRM verschlüsselt Nutzdaten bei Kerberos- oder NTLM-Authentifizierung auf Nachrichtenebene auch über HTTP (Port 5985). Außerhalb von Active-Directory-Domänen erfordert die verlässliche Serverauthentifizierung jedoch entweder eigene TLS-Zertifikate für HTTPS (Port 5986) oder die Konfiguration der lokalen `TrustedHosts`-Liste. `TrustedHosts` verifiziert jedoch weder die Identität des Zielsystems noch bietet es eigenständige Verschlüsselung. OpenSSH nutzt stattdessen den in Linux-Umgebungen etablierten Standard aus Host-Keys für die Serververifikation und schlüsselbasierter Authentifizierung, wie du ihn bereits aus der Linux-Systemhärtung kennst.

Die Trennlinie: SConfig-Remoteverwaltung versus OpenSSH

Ein häufiges Missverständnis bei der Ersteinrichtung von Server Core betrifft den Menüpunkt 4 in SConfig: *Remoteverwaltung konfigurieren*.

Wird dieser Punkt auf *Aktiviert* gesetzt, schaltet Windows Server ausschließlich die Firewall-Ausnahmen und Listener für WinRM sowie Schnittstellen für WMI und den Remotedesktop-Dienst frei (sofern RDP zusätzlich aktiviert wird).

> ⚠ **Häufige Stolperfalle:** Das Aktivieren der Remoteverwaltung in SConfig berührt den OpenSSH-Server (`sshd`) in keiner Weise. Die Windows-Firewall öffnet durch SConfig nicht den TCP-Port 22, und der SSH-Dienst wird weder gestartet noch auf automatischen Start gesetzt. Wer nach der SConfig-Konfiguration die Verbindung trennt, kann die Maschine per SSH nicht erreichen.

Diese funktionale Trennung bedeutet in der Praxis:

- OpenSSH ist unter Windows Server 2025 ein eigenständiger Systemdienst.
- Dienststatus, Starttyp und Firewallfreigaben für Port 22 müssen unabhängig von SConfig eingerichtet werden.
- Für Administratoren an Linux-Workstations ist die SConfig-Option 4 optional, während die dedizierte PowerShell-Konfiguration von `sshd` die zwingende Voraussetzung für den Fernzugriff darstellt.

Die Dual-Stack-Verwaltungsarchitektur: OpenSSH und Windows Admin Center (WAC)

Im professionellen Rechenzentrumsbetrieb schließt der Verzicht auf die lokale Desktop Experience grafische Übersichten keineswegs aus. Moderne Windows-Server-Architekturen setzen auf ein praxistaugliches Dual-Stack-Modell:

1. **Terminal- und Automatisierungsebene (OpenSSH):**

Dient der direkten Ad-hoc-Administration, schnellen Fehlerbehebung, Shell-Skripterstellung und der Einbindung in plattformübergreifende Automatisierungswerkzeuge wie Ansible, Terraform oder CI/CD-Runner. Über Port 22 greifst du von Linux-Clients ohne Abhängigkeiten zu Windows-RPC-Ports oder NTLM-Zertifikaten direkt auf die native PowerShell 5.1 zu.

1. **Webbasierte Managementebene (Windows Admin Center):**

Microsofts modernes browserbasiertes Verwaltungswerkzeug (Windows Admin Center, WAC) läuft entweder auf einem zentralen Gateway-Server oder auf einer separaten Administrator-Workstation. WAC kommuniziert über WinRM (Port 5985) und PowerShell Remoting im Hintergrund mit dem Core-Server und stellt im Browser eine vollwertige HTML5-Oberfläche

bereit:

- Visuelle Echtzeit-Leistungsdiagramme für CPU, RAM und Datenträger-IOPS
- Zertifikatsverwaltung und Firewall-Konfiguration im Webbrowser
- Verwaltung von Speicher-Pools und Festplattenzuweisungen
- Durchsuchbare Ereignisanzeige mit strukturiertem Export

Dieses Modell vereint die Vorteile beider Welten: Das eigentliche Serversystem bleibt frei von ressourcenhungrigem GUI-Overhead und potenziellen Desktop-Sicherheitslücken, während Administratoren je nach Aufgabe flexibel zwischen der schnellen SSH-Kommandozeile und einer modernen Web-Oberfläche wechseln können.

Virtuelle Maschine unter Proxmox VE 9 erstellen

Nachdem die architektonischen Eigenheiten von Server Core geklärt sind, folgt die praktische Bereitstellung des Gastsystems auf dem Hypervisor. Anstelle der grafischen Proxmox-Weboberfläche erfolgt die VM-Erstellung hier konsistent über die Shell des Proxmox-VE-Hosts. Das Kommandozeilenwerkzeug `qm` (QEMU/KVM Virtual Machine Manager) ermöglicht die vollständig reproduzierbare, skriptfähige Definition aller Hardwareparameter.

Die Einrichtung auf der Befehlszeile gliedert sich in folgende aufeinander aufbauende Schritte:

1. **Speicherorte & ISOs prüfen** Windows- und VirtIO-Images auf dem Proxmox-Storage verifizieren
2. **VM-Basis per qm anlegen** Maschinentyp Q35, UEFI (OVMF), Prozessor und RAM definieren
3. **Datenträger & EFI konfigurieren** EFI-Disk, Systemlaufwerk (SATA oder VirtIO-SCSI) und Boot-Order setzen
4. **Optionale Komponenten & Start** QEMU Guest Agent aktivieren, vTPM anbinden und Konsole öffnen

Vorbereitung der Installationsmedien

Vor der VM-Erstellung müssen die beiden ISO-Dateien auf dem Proxmox-Storage bereitliegen: das Installationsmedium von Windows Server 2025 sowie die aktuelle Treiber-ISO nach den offiziellen VirtIO-Best-Practices. Den genauen Speicherpfad und die Verfügbarkeit der Images überprüfst du auf der Proxmox-Shell mit dem Storage-Manager `pvesm`:

```
pvesm list local --content iso
```

Die Ausgabe listet die Volumennamen der abgelegten Images auf, beispielsweise `local:iso/Windows_Server_2025_Evaluation_x64.iso` und `local:iso/virtio-win-0.1.302.iso`. Diese Bezeichner werden den virtuellen CD/DVD-Laufwerken übergeben.

VM-Erstellung mit dem CLI-Werkzeug qm

Unter Proxmox VE 9 steuert der Befehl `qm create` die vollständige Allokation der KVM-Ressourcen. Für moderne Windows-Gäste sind Maschinentyp, Firmware und CPU-Eigenschaften entscheidend.

Der folgende Befehl legt die Basiskonfiguration für eine neue virtuelle Maschine mit der Beispiel-ID `104` an:

```
qm create 104 \
  --name "WIN-CORE01" \
  --ostype win11 \
  --machine q35 \
  --bios ovmf \
  --cpu host \
  --cores 4 \
  --sockets 1 \
```

```
    --memory 8192 \
    --agent enabled=1
```

> **Bedeutung des OS-Typs `win11`:** Proxmox VE nutzt das Flag `--ostype win11` als Sammelbezeichnung für alle modernen 64-Bit-Windows-Systeme ab Windows 11 sowie Windows Server 2022 und 2025. Dadurch aktiviert QEMU intern spezifische Hyper-V-Enlightenments (wie synic, stimer und relaxed), die den Virtualisierungs-Overhead für den Windows-Kernel spürbar reduzieren.

Die gesetzten Parameter im Detail:

- `--machine q35`: Emuliert den modernen PCIe-Chipsatz Q35 anstelle des veralteten PCI-Busses `i440fx`.
- `--bios ovmf`: Schaltet von legacy SeaBIOS auf UEFI um. Windows Server 2025 setzt für Secure Boot und moderne Partitionierungsstrukturen (GPT) zwingend auf UEFI.
- `--cpu host`: Reicht die Prozessor-Flags des Proxmox-Hosts direkt durch.

> ⚠ **Leistungsfalle kvm64:** Belässt du den Standardwert `kvm64`, emuliert KVM einen veralteten Prozessor ohne moderne Befehlssatzerweiterungen wie AVX, AVX2, AES-NI und SSE4.2. Dies führt unter Windows Server 2025 zu gravierenden Leistungseinbußen bei kryptografischen Operationen (wie SSH und TLS) und kann im schlimmsten Fall Fehler während des Setups verursachen.

- `--agent enabled=1`: Weist Proxmox an, den QEMU Guest Agent zu erwarten. Dadurch kann der Hypervisor später den Status überwachen, IP-Adressen im Dashboard anzeigen und konsistente Backups via Proxmox Backup Server (PBS) durchführen.

EFI-Datenträger und virtuelles TPM anlegen

Da UEFI (`ovmf`) als Firmware definiert wurde, benötigt die VM einen separaten EFI-Datenträger zur Speicherung von NVRAM-Variablen (wie Boot-Einträgen und Secure-Boot-Schlüsseln):

```
qm set 104 --efidisk0 local-zfs:1,efitype=4m,pre-enrolled-keys=1
```

Der Parameter `pre-enrolled-keys=1` schreibt die standardmäßigen Microsoft-Zertifikate für Secure Boot direkt in den EFI-Speicher.

Falls du für spätere Härtungsmaßnahmen BitLocker oder Virtualization-based Security (VBS) evaluieren möchtest, kannst du optional ein virtualisiertes TPM 2.0 hinzufügen:

```
qm set 104 --tpmstate0 local-zfs:1,version=v2.0
```

Festplatten-Controller und Speicherlaufwerk konfigurieren

An dieser Stelle entscheidet sich die Wahl des Speicher-Controllers:

- **Variante A: VirtIO-SCSI (Best Practice für Performance)**

Paravirtualisierter Controller mit minimalem I/O-Overhead. Erfordert das Einbinden von Treibern während der Windows-Installation:

```
qm set 104 \
  --scsihw virtio-scsi-single \
  --scsi0 local-zfs:64,discard=on,ssd=1
```

`discard=on` erlaubt die Rückgabe freigegebener Blöcke via TRIM an das Storage, `ssd=1` signalisiert Windows, dass es sich um Solid-State-Speicher handelt.

- **Variante B: SATA (Einfachste Bereitstellung ohne Zusatztreiber)**

Entspricht der Konfiguration unserer dokumentierten Test-VM. Windows Server 2025 erkennt das Laufwerk ohne manuelle Treiberinstallation:

```
qm set 104 --sata0 local-zfs:150
```

Netzwerkadapter, Installationsmedien und Boot-Reihenfolge

Im letzten Schritt verknüpfst du das Netzwerk, hängst die ISO-Images in die virtuellen CD/DVD-Laufwerke ein und definierst die Boot-Reihenfolge:

- **Netzwerk:** Für maximale Performance empfiehlt sich VirtIO (`--net0 virtio,bridge=vmbr0`). In unserer Test-VM kam zur Vermeidung von Treiber-Abhängigkeiten beim Setup die Intel-E1000-Emulation zum Einsatz:

```
qm set 104 --net0 e1000,bridge=vmbr0
```

- **ISO-Laufwerke einhängen:**

```
qm set 104 \
  --ide2 local:iso/Windows_Server_2025_Evaluation_x64.iso,media=cdrom \
  --ide0 local:iso/virtio-win-0.1.302.iso,media=cdrom
```

> 💡 **VirtIO-Treiber-ISO immer mitführen:** Selbst wenn du dich wie in unserer Test-VM für SATA und Intel E1000 entscheidest, lohnt sich das permanente Einhängen der `virtio-win`-ISO an `ide0`. Dadurch stehen die Installationsdateien für den QEMU Guest Agent (`qemu-ga-x86_64.msi`) und spätere VirtIO-Treiber nach dem Booten sofort zur Verfügung.

- **Boot-Reihenfolge auf das Windows-Installationsmedium setzen:**

```
qm set 104 --boot order=ide2
```

Vollständige Referenzkonfiguration

Um die Konfiguration in einem einzigen Schritt reproduzierbar auszuführen, fasst der folgende Shell-Befehl alle Parameter zusammen. Passe lediglich die VM-ID (`104`), den Ziel-Storage (`local-zfs`) und die genauen Dateinamen deiner ISO-Images an:

```
qm create 104 \
  --name "WIN-CORE01" \
  --ostype win11 \
  --machine q35 \
  --bios ovmf \
  --efidisk0 local-zfs:1,efitype=4m,pre-enrolled-keys=1 \
  --cpu host \
  --cores 4 \
  --sockets 1 \
  --memory 8192 \
  --agent enabled=1 \
  --sata0 local-zfs:150 \
  --net0 e1000,bridge=vmbr0 \
  --ide2 local:iso/Windows_Server_2025_Evaluation_x64.iso,media=cdrom \
```

```
    --ide0 local:iso/virtio-win-0.1.302.iso,media=cdrom \
    --boot order=ide2
```

Das Ergebnis lässt sich direkt in der zugehörigen Proxmox-Konfigurationsdatei überprüfen:

```
# /etc/pve/qemu-server/104.conf
agent: 1
bios: ovmf
boot: order=ide2
cores: 4
cpu: host
efidisk0: local-zfs:vm-104-disk-0,efitype=4m,pre-enrolled-keys=1
ide0: local:iso/virtio-win-0.1.302.iso,media=cdrom
ide2: local:iso/Windows_Server_2025_Evaluation_x64.iso,media=cdrom
machine: q35
memory: 8192
name: WIN-CORE01
net0: e1000,bridge=vmbr0
ostype: win11
sata0: local-zfs:vm-104-disk-1,size=150G
sockets: 1
```

> **Konfigurationsdateien im Proxmox-Cluster:** Proxmox VE speichert Maschinendefinitionen im clusterweiten Dateisystem unter `/etc/pve/qemu-server/<VMID>.conf`. Wenn du Parameter wie Arbeitsspeicher oder CPU-Kerne später anpassen möchtest, kannst du die Werte entweder über `qm set` überschreiben oder die Datei direkt auf dem Host bearbeiten.

Erweiterte Hypervisor-Optimierungen für den Produktionsbetrieb

Für Testumgebungen genügt die Standardkonfiguration. In Produktions-Clustern mit hohen I/O-Lasten oder virtualisierten Datenbankrollen entscheiden zusätzliche KVM- und QEMU-Parameter

über Latenz und Stabilität.

1. KVM Hyper-V Enlightenments im Detail:

Beim Setzen von `--ostype win11` schaltet Proxmox VE spezifische paravirtualisierte Schnittstellen frei, die der Windows-Kernel nativ unterstützt. Diese sogenannten Enlightenments gaukeln Windows vor, auf Microsoft Hyper-V zu laufen, wodurch kostspielige VM-Exits auf dem Host minimiert werden:

- `hv_relaxed`: Verhindert Abstürze und Timeouts des Gast-Betriebssystems bei kurzzeitiger CPU-Drosselung oder Host-Überbuchung.
- `hv_spinlocks`: Optimiert das Warten auf Kernel-Sperren bei Mehrprozessorsystemen.
- `hv_vapic`: Beschleunigt den Zugriff auf den virtuellen Advanced Programmable Interrupt Controller (APIC).
- `hv_time` & `hv_synic`: Stellen synthetische Timer- und Interrupt-Architekturen bereit, die den CPU-Overhead bei periodischen Systemintervallen drastisch senken.

2. Storage-Tuning: AIO-Backend, Caching und IO-Threads:

Wird anstelle von SATA der paravirtualisierte SCSI-Controller `virtio-scsi-single` verwendet, lassen sich Durchsatz und IOPS über drei Schlüsselparameter optimieren:

```
qm set 104 \
  --scsihw virtio-scsi-single \
  --scsi0 local-zfs:64,discard=on,ssd=1,iothread=1,cache=none,aio=io_uring
```

- `iothread=1`: Weist dem Controller einen dedizierten Host-Thread für I/O-Operationen zu. Dadurch blockieren intensive Schreibvorgänge der virtuellen Festplatte nicht den QEMU-Hauptprozess, der für CPU-Emulation und Timer zuständig ist.
- `cache=none`: Schaltet die Pufferung im Host-Betriebssystem-Cache ab (O_DIRECT). Bei ZFS-Pools verhindert dies eine doppelte Speicherbelegung (Host-ZFS-ARC versus Windows File System Cache) und garantiert, dass Schreibvorgänge transaktionssicher direkt auf dem

Storage landen.

- `aio=io_uring`: Nutzt das moderne Linux-Kernel-Interface `io_uring` für asynchrone Ein-/Ausgaben, was insbesondere auf NVMe-basierten Speicher-Pools signifikant niedrigere Latenzen gegenüber klassischem `threads` erzielt.

3. vCPU-Topologie und NUMA-Knoten:

Auf physischen Servern mit mehreren CPU-Sockeln oder Prozessoren mit AMD-Chiplet-Architektur (EPYC) sollte die virtuelle Maschine die reale NUMA-Topologie (Non-Uniform Memory Access) abbilden:

```
# NUMA-Emulation aktivieren und Sockets/Cores definieren
qm set 104 --numa 1 --sockets 1 --cores 4
```

Ohne aktiviertes NUMA behandelt Windows den zugewiesenen Arbeitsspeicher als einheitlichen Block. Bei speicherintensiven Diensten (wie Active Directory oder SQL Server) führt dies dazu, dass Speicherzugriffe unwissentlich über langsamere Inter-Socket-Verbindungen (wie AMD Infinity Fabric oder Intel UPI) geroutet werden.

4. Netzwerk-Härtung und VLAN-Segmentierung:

Im Rechenzentrum gehört der Administrationszugang eines Core-Servers in ein separates Management-VLAN. Proxmox VE erlaubt die direkte Zuweisung eines 802.1Q-VLAN-Tags an der virtuellen Netzwerkkarte, ohne dass im Gastbetriebssystem zusätzliche Treiber oder virtuelle Adapter konfiguriert werden müssen:

```
# VM-Netzwerkkarte an VLAN 20 (Management) binden und Proxmox-Firewall aktivieren
qm set 104 --net0 virtio,bridge=vmbr0,tag=20,firewall=1
```

Automatisierte VM-Erstellung per Shell-Skript

Um die Bereitstellung neuer Core-Instanzen reproduzierbar und fehlerfrei zu gestalten, empfiehlt sich ein Bash-Skript auf dem Proxmox-Host. Das folgende Skript prüft vorab, ob die gewählte VM-ID frei ist, validiert das Vorhandensein der erforderlichen ISO-Dateien auf dem Storage und legt die Maschine vollautomatisiert an:

```
#!/usr/bin/env bash
# /root/deploy-wincore-vm.sh
# Automatisiertes Anlegen einer Windows Server 2025 Core VM unter Proxmox VE 9

set -euo pipefail

# Konfigurationsvariablen
VM_ID="${1:-104}"
VM_NAME="${2:-WIN-CORE01}"
STORAGE_POOL="${3:-local-zfs}"
DISK_SIZE="${4:-150G}"
RAM_MB="${5:-8192}"
CPU_CORES="${6:-4}"
BRIDGE="${7:-vmbr0}"

WIN_ISO="local:iso/Windows_Server_2025_Evaluation_x64.iso"
VIRTIO_ISO="local:iso/virtio-win-0.1.302.iso"

echo "=== Windows Server 2025 Core VM Deployment ==="
echo "VM-ID:     ${VM_ID}"
echo "Hostname:  ${VM_NAME}"
echo "Storage:   ${STORAGE_POOL} (${DISK_SIZE})"
echo "Ressourcen: ${CPU_CORES} Cores, ${RAM_MB} MB RAM"

# 1. Prüfen, ob VM-ID bereits vergeben ist
if qm status "${VM_ID}" &>/dev/null; then
    echo "FEHLER: VM-ID ${VM_ID} existiert bereits auf diesem Node!" >&2
    exit 1
fi

# 2. Prüfen, ob die Installations-ISOs auf dem Storage verfügbar sind
for iso in "${WIN_ISO}" "${VIRTIO_ISO}"; do
```

```
    vol_id="${iso##*:}"
    storage_name="${iso%%:*}"
    if ! pvesm list "${storage_name}" | grep -q "${vol_id}"; then
        echo "FEHLER: Image '${iso}' wurde auf Storage '${storage_name}' nicht gefunden!"
>&2
        exit 1
    fi
done

echo "[+] ISO-Dateien verifiziert."

# 3. VM-Basiskonfiguration erstellen
echo "[+] Erstelle virtuelle Maschine..."
qm create "${VM_ID}" \
    --name "${VM_NAME}" \
    --ostype win11 \
    --machine q35 \
    --bios ovmf \
    --cpu host \
    --cores "${CPU_CORES}" \
    --sockets 1 \
    --memory "${RAM_MB}" \
    --numa 1 \
    --agent enabled=1

# 4. EFI-Disk und Boot-Medium anbinden
echo "[+] Konfiguriere EFI-Disk und TPM..."
qm set "${VM_ID}" --efidisk0 "${STORAGE_POOL}:1,efitype=4m,pre-enrolled-keys=1"
qm set "${VM_ID}" --tpmstate0 "${STORAGE_POOL}:1,version=v2.0"

# 5. Speicherlaufwerk und optische Medien einbinden
echo "[+] Binde Festplatte und Installationsmedien an..."
qm set "${VM_ID}" \
    --sata0 "${STORAGE_POOL}:${DISK_SIZE}" \
    --net0 "e1000,bridge=${BRIDGE},firewall=1" \
    --ide2 "${WIN_ISO},media=cdrom" \
    --ide0 "${VIRTIO_ISO},media=cdrom" \
    --boot order=ide2

echo "=== Bereitstellung erfolgreich abgeschlossen ==="
echo "Startbefehl: qm start ${VM_ID}"
```

Speichere das Skript auf dem Proxmox-Knoten, mache es ausführbar (`chmod +x /root/deploy-wincore-vm.sh`) und starte die Bereitstellung mit:

```
/root/deploy-wincore-vm.sh 104 "WIN-CORE01" local-zfs 150G 8192 4 vmbr0
```

Proxmox-Snapshot- und Backup-Strategien über die CLI

Bevor das Betriebssystem installiert oder größere Wartungsarbeiten durchgeführt werden, bietet Proxmox VE auf Shell-Ebene mächtige Werkzeuge für Snapshots und Datensicherungen:

1. ZFS-gestützte Snapshots vor kritischen Änderungen:

Über den Befehl `qm snapshot` lässt sich der Zustand der virtuellen Maschine innerhalb von Sekunden einfrieren:

```
# Snapshot vor dem ersten Start oder vor Rollen-Upgrades erstellen
qm snapshot 104 "clean-hw-state" --description "Frisch angelegte VM-Hardware vor
Betriebssystem-Setup"

# Vorhandene Snapshots der VM auflisten
qm listsnapshot 104

# Bei Fehlkonfigurationen blitzschnell auf den Snapshot zurückspringen
qm rollback 104 "clean-hw-state"
```

2. VSS-integrierte Backups mit vzdump:

Wird später der QEMU Guest Agent im Gastsystem ausgeführt, erzeugt Proxmox über `vzdump` garantierte anwendungskonsistente Datensicherungen:

```
# Snapshot-Backup auf Proxmox Backup Server oder NFS-Storage initiieren
vzdump 104 --storage local --mode snapshot --compress zstd --mailnotification failure
```

Mit diesem Schritt ist die VM-Konfiguration auf der Proxmox-Shell vollständig abgeschlossen. Die Instanz kann nun mit `qm start 104` gestartet werden.

Windows Server 2025 Core installieren

Nachdem die virtuelle Maschine auf dem Proxmox-Host angelegt ist, beginnt die eigentliche Betriebssysteminstallation. Der gesamte Vorgang wird über die interaktive noVNC-Konsole der Proxmox-Weboberfläche begleitet, da das Gastsystem in dieser Phase noch über keine Netzwerkdienste oder Fernzugänge verfügt.

Der Installationsablauf gliedert sich in folgende Phasen:

1. **Initialer Bootvorgang** VM starten, noVNC-Konsole öffnen und Tastendruck für CD-Boot abfangen
2. **Editionsauswahl** Windows Server 2025 Standard (Server Core) als Zielversion definieren
3. **Datenträgererkennung** SATA sofort verfügbar (oder VirtIO-SCSI-Treiber von ISO laden)
4. **Dateikopie & Erster Reboot** Automatische Installation und Systemstart in das neue Systemlaufwerk
5. **Kennwortvergabe & Login** Administratorpasswort festlegen und initialer Einstieg in SConfig

Start der virtuellen Maschine und Boot-Phase

Starte die virtuelle Maschine auf der Proxmox-Shell oder über die Web-GUI:

```
qm start 104
```

Öffne unmittelbar danach in der Proxmox-Weboberfläche die **Konsole** der VM (noVNC).

> **Boot-Prompt in noVNC rechtzeitig bestätigen:** Unmittelbar nach der Initialisierung der OVMF-Firmware erscheint die Meldung *„Press any key to boot from CD or DVD...“*. Klicke mit der Maus sofort in das noVNC-Fenster und drücke eine beliebige Taste (beispielsweise die Leertaste). Erfolgt innerhalb weniger Sekunden keine Eingabe, überspringt das UEFI das Installationsmedium und landet in der *UEFI Interactive Shell*. In diesem Fall musst du die VM über das Proxmox-Menü zurücksetzen (*Reset*).

Sobald das Setup geladen ist, begrüßt dich der grafische Windows-Setup-Assistent. Bestätige Sprache (*Deutsch*), Zeit- und Währungsformat sowie die Tastaturbelegung und klicke auf *Weiter* sowie anschließend auf *Jetzt installieren*.

Auswahl der Server-Core-Installation

Nach Aufforderung zur Eingabe eines Product Keys (im Evaluierungs-Setup wählst du *„Ich habe keinen Product Key“*) präsentiert das Installationsmedium die verfügbaren Editionen.

An dieser Stelle fällt die verbindliche Vorentscheidung über den System-Footprint:

- **Windows Server 2025 Standard** *(dies ist die Server-Core-Variante)*
- Windows Server 2025 Standard (Desktopdarstellung / Desktop Experience)
- Windows Server 2025 Datacenter
- Windows Server 2025 Datacenter (Desktopdarstellung)

> ⚠ **Editionsbezeichnung im Setup:** Microsoft bezeichnet die Core-Variante im Setup-Menü schlicht als *Windows Server 2025 Standard* – ohne den Zusatz „Server Core“. Der grafische Desktop wird explizit durch den Zusatz *(Desktopdarstellung)* gekennzeichnet. Wähle den ersten Eintrag, um Server Core zu installieren.

Akzeptiere die Lizenzbedingungen und wähle im darauffolgenden Dialog die Installationsart: **Benutzerdefiniert: Nur Windows installieren (für fortgeschrittene Benutzer)**.

Datenträgererkennung: SATA versus VirtIO-SCSI

Im Dialog *„Wo möchten Sie Windows installieren?“* unterscheidet sich der Ablauf je nach zuvor gewähltem Storage-Controller:

- **SATA-Installation (unsere dokumentierte Test-VM):**

Windows Server 2025 bringt standardmäßig AHCI-Treiber mit. Das 150-GiB-Laufwerk erscheint sofort als *„Nicht zugewiesener Speicherplatz auf Datenträger 0“*. Es ist kein manuelles Nachladen erforderlich. Klicke direkt auf *Weiter*.

- **VirtIO-SCSI-Installation (Labor-Empfehlung):**

Die Datenträgerliste bleibt zunächst vollständig leer, da Windows KVM-Speichercontroller nicht nativ unterstützt. Gehe wie folgt vor:

1. Klicke auf **Treiber laden** und anschließend auf **Durchsuchen**.
2. Navigiere auf das zweite virtuelle CD-Laufwerk, in dem die `virtio-win`-ISO eingehängt ist.
3. Öffne den Ordner `vioscsi` → `2k25` → `amd64`.
4. Bestätige die Auswahl des Treibers *Red Hat VirtIO SCSI pass-through controller*.
5. Nach dem Ladevorgang wird der paravirtualisierte Datenträger in der Liste angezeigt.

> □□ **Automatische GPT-Partitionierung:** Beim Klick auf *Weiter* auf dem unpartitionierten Datenträger partitioniert das Windows-Setup die Festplatte automatisch nach UEFI-Standards: Es entstehen eine EFI-Systempartition (FAT32, ca. 100 MiB), eine Microsoft

> Reserved Partition (MSR, 16 MiB) sowie die primäre Systempartition (NTFS), auf die die Windows-Dateien entpackt werden.

Das Setup kopiert nun die Systemabbilder, installiert Funktionen und startet die virtuelle Maschine nach wenigen Minuten automatisch neu.

Administratorpasswort festlegen und Erstanmeldung

Nach dem ersten Neustart bootet die VM selbstständig von der virtuellen Festplatte. Anstelle einer Desktop-Begrüßung erscheint auf Server Core eine textbasierte Eingabeaufforderung mit einem blauen Konsolen-Dialog:

```
Das Kennwort des Benutzers muss vor der ersten Anmeldung geändert werden.
```

1. Bestätige mit der `Eingabetaste` auf dem vorausgewählten Feld `<OK>`.
2. Gib das neue Kennwort für das lokale Konto `Administrator` ein und wiederhole es in der zweiten Zeile.
3. Bestätige mit `Eingabetaste`.

> ⚠ **Tastaturlayout in der noVNC-Konsole beachten:** Je nach Browsereinstellung und Proxmox-Knotenkonfiguration kann die noVNC-Konsole vor dem ersten vollständigen Gaststart auf das US-Tastaturlayout (QWERTY) zurückfallen. Achte bei der Kennwortvergabe besonders auf vertauschte Tasten (`Z` und `Y`) sowie Sonderzeichen, um spätere Anmeldefehler beim SSH-Zugriff zu vermeiden.

Windows bestätigt die Änderung mit der Meldung *„Das Kennwort wurde geändert“*. Nach einem weiteren Druck auf die `Eingabetaste` meldet Windows den Administrator an.

Auf dem Bildschirm öffnet sich ein schwarzes Terminalfenster (`cmd.exe`), und nach wenigen Sekunden startet im Vordergrund automatisch das blaue Menü von **SConfig**. Das System befindet sich nun im betriebsbereiten Grundzustand.

Vollautomatisierte Bereitstellung: Unattended Setup mit autounattend.xml

In größeren Rechenzentren oder Testlaboren ist das manuelle Durchklicken des Setup-Assistenten über noVNC unwirtschaftlich. Windows Server 2025 unterstützt wie seine Vorgänger die vollautomatisierte Installation über eine Antwortdatei namens `autounattend.xml` .

Befindet sich diese Datei im Stammverzeichnis eines erkannten Datenträgers (beispielsweise auf einem zweiten virtuellen CD-ROM-Laufwerk oder einem virtuellen USB-Stick), überspringt das Windows-Setup alle interaktiven Abfragen, partitioniert die Festplatte selbstständig, wählt die Core-Edition aus, vergibt das Administratorkennwort und aktiviert auf Wunsch sofort den OpenSSH-Dienst.

1. Struktur der Antwortdatei autounattend.xml:

Die folgende XML-Konfiguration ist für eine vollautomatische Bereitstellung von Windows Server 2025 Standard (Core) auf UEFI-Systemen optimiert:

```
<?xml version="1.0" encoding="utf-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
  <!-- Phase 1: Windows Preinstallation Environment (WinPE) -->
  <settings pass="windowsPE">
    <component name="Microsoft-Windows-International-Core-WinPE"
processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral"
versionScope="nonSxS">
      <SetupUILanguage>
        <UILanguage>de-DE</UILanguage>
      </SetupUILanguage>
      <InputLocale>de-DE</InputLocale>
      <SystemLocale>de-DE</SystemLocale>
      <UILanguage>de-DE</UILanguage>
```

```
      <UserLocale>de-DE</UserLocale>
    </component>
    <component name="Microsoft-Windows-Setup" processorArchitecture="amd64"
publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
      <DiskConfiguration>
        <Disk wcm:action="add"
xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State">
          <DiskID>0</DiskID>
          <WillWipeDisk>true</WillWipeDisk>
          <CreatePartitions>
            <!-- 1. EFI-Systempartition (FAT32, 100 MB) -->
            <CreatePartition wcm:action="add">
              <Order>1</Order>
              <Type>EFI</Type>
              <Size>100</Size>
            </CreatePartition>
            <!-- 2. Microsoft Reserved Partition (MSR, 16 MB) -->
            <CreatePartition wcm:action="add">
              <Order>2</Order>
              <Type>MSR</Type>
              <Size>16</Size>
            </CreatePartition>
            <!-- 3. Betriebssystempartition (Rest der Festplatte) -->
            <CreatePartition wcm:action="add">
              <Order>3</Order>
              <Type>Primary</Type>
              <Extend>true</Extend>
            </CreatePartition>
          </CreatePartitions>
          <ModifyPartitions>
            <ModifyPartition wcm:action="add">
              <Order>1</Order>
              <PartitionID>1</PartitionID>
              <Format>FAT32</Format>
              <Label>System</Label>
            </ModifyPartition>
            <ModifyPartition wcm:action="add">
              <Order>2</Order>
              <PartitionID>2</PartitionID>
            </ModifyPartition>
            <ModifyPartition wcm:action="add">
              <Order>3</Order>
              <PartitionID>3</PartitionID>
              <Format>NTFS</Format>
```

```
                <Label>Windows</Label>
                <Letter>C</Letter>
              </ModifyPartition>
            </ModifyPartitions>
          </Disk>
        </DiskConfiguration>
        <ImageInstall>
          <OSImage>
            <InstallFrom>
              <MetaData wcm:action="add">
                <Key>/IMAGE/NAME</Key>
                <!-- Exakte Bezeichnung für Server Core in Windows Server 2025 -->
                <Value>Windows Server 2025 Standard</Value>
              </MetaData>
            </InstallFrom>
            <InstallTo>
              <DiskID>0</DiskID>
              <PartitionID>3</PartitionID>
            </InstallTo>
          </OSImage>
        </ImageInstall>
        <UserData>
          <AcceptEula>true</AcceptEula>
        </UserData>
      </component>
    </settings>

    <!-- Phase 2: Allgemeine Systemeinstellungen -->
    <settings pass="specialize">
      <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="amd64"
  publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
        <ComputerName>WIN-CORE01</ComputerName>
        <TimeZone>W. Europe Standard Time</TimeZone>
      </component>
    </settings>

    <!-- Phase 3: Out-of-Box Experience (OOBE) & Erststart-Befehle -->
    <settings pass="oobeSystem">
      <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="amd64"
  publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
        <UserAccounts>
          <AdministratorPassword>
            <Value>Start12345!</Value>
            <PlainText>true</PlainText>
```

```
        </AdministratorPassword>
      </UserAccounts>
      <!-- Befehle, die beim ersten automatischen Systemstart ausgeführt werden -->
      <FirstLogonCommands>
        <!-- OpenSSH Server Capability installieren und Dienst starten -->
        <SynchronousCommand wcm:action="add">
          <Order>1</Order>
          <Description>OpenSSH Server installieren</Description>
          <CommandLine>powershell.exe -Command "Add-WindowsCapability -Online -Name
OpenSSH.Server~~~~0.0.1.0"</CommandLine>
        </SynchronousCommand>
        <SynchronousCommand wcm:action="add">
          <Order>2</Order>
          <Description>OpenSSH Dienst aktivieren und starten</Description>
          <CommandLine>powershell.exe -Command "Set-Service -Name sshd -StartupType
Automatic; Start-Service -Name sshd"</CommandLine>
        </SynchronousCommand>
        <SynchronousCommand wcm:action="add">
          <Order>3</Order>
          <Description>SConfig Autostart deaktivieren</Description>
          <CommandLine>powershell.exe -Command "Set-SConfig -AutoLaunch
$false"</CommandLine>
        </SynchronousCommand>
      </FirstLogonCommands>
    </component>
  </settings>
</unattend>
```

 Sicherheitsrisiko bei Klartextkennwörtern in der Antwortdatei: Bei `<PlainText>true</PlainText>` wird das Kennwort im unverschlüsselten Klartext in der XML-Datei hinterlegt. Jeder Benutzer oder Prozess mit Leserechten auf das ISO-Image, den virtuellen Datenträger oder die Datei `autounattend.xml` kann diese Zugangsdaten unmittelbar einsehen. Ein nachträglicher Kennwortwechsel nach der ersten Anmeldung beseitigt dieses Risiko nicht rückwirkend, da die Datei auf dem Installationsmedium verbleibt. Der Wert `Start12345!` dient ausschließlich als nicht produktiv verwendbarer Beispielplatzhalter. Für reale Bereitstellungen müssen individuelle temporäre Zugangsdaten über kontrollierte Provisionierungskanäle übergeben und umgehend rotiert werden. Die von Microsoft dokumentierte Option `<PlainText>false</PlainText>`

> blendet den Wert lediglich unkenntlich aus, bietet jedoch keine kryptografisch sichere Verschlüsselung.

2. Antwortdatei als ISO-Image auf dem Proxmox-Host verpacken:

Damit Windows PE die `autounattend.xml` beim Booten erkennt, erstellst du auf der Proxmox-Shell ein minimales ISO-Image:

```
# Arbeitsverzeichnis erstellen und XML-Datei ablegen
mkdir -p /tmp/unattend
cat <<'EOF' > /tmp/unattend/autounattend.xml
<!-- Hier den obigen XML-Inhalt einfügen -->
EOF

# ISO-Image im lokalen Proxmox-Template-Speicher generieren
genisoimage -o /var/lib/vz/template/iso/unattend-wincore.iso -J -r /tmp/unattend/
```

3. Antwort-ISO in die VM einhängen und Zero-Touch-Installation starten:

Binde das generierte ISO an das freie Laufwerk `ide1` der virtuellen Maschine:

```
qm set 104 --ide1 local:iso/unattend-wincore.iso,media=cdrom
```

Wird die VM nun mit `qm start 104` gestartet, liest das Windows-Setup die Antwortdatei automatisch ein. Ohne eine einzige Tastatureingabe im noVNC-Fenster durchläuft das System die Partitionierung, installiert Windows Server 2025 Core, richtet das Administratorkonto ein und startet den OpenSSH-Dienst. Nach wenigen Minuten ist die Instanz direkt per SSH von deinem Linux-Terminal aus erreichbar.

Treiberinstallation und Hardware-Prüfung ohne Geräte-Manager (pnputil)

Auf einem Server Core steht der grafische Geräte-Manager (`devmgmt.msc`) nicht zur Verfügung. Um nach der Installation zu prüfen, ob alle virtuellen KVM-Geräte (wie Speichercontroller, PCI-Brücken und Netzwerkkarten) fehlerfrei initialisiert wurden, nutzt du die PowerShell und das integrierte Microsoft-Werkzeug `pnputil.exe` .

1. Unbekannte oder fehlerhafte Geräte ermitteln:

Mit `Get-PnpDevice` fragst du alle Hardwarekomponenten ab, deren Treiberstatus von `OK` abweicht:

```
Get-PnpDevice | Where-Object Status -ne 'OK' | Format-Table -Property InstanceId, Class, FriendlyName, Status
```

Wurden beim Setup Standard-Hardwarekomponenten verwendet oder fehlen noch spezifische VirtIO-Treiber, werden die entsprechenden Geräte hier als `Error` oder `Unknown` geführt.

2. Alle Treiber von der VirtIO-ISO in einem Schritt installieren:

Befindet sich die `virtio-win` -Treiber-ISO im virtuellen CD-Laufwerk (im Beispiel Laufwerk `D:`), installierst du sämtliche paravirtualisierten Treiber für Speicher, Netzwerk, Ballooning und serielle Schnittstellen mit einem einzigen rekursiven Befehl:

```
pnputil.exe /add-driver D:\*.inf /subdirs /install
```

`pnputil.exe` scannt alle Unterordner, signiert die Treiber im lokalen Windows Driver Store und bindet sie unmittelbar an passende Hardwarekomponenten an.

3. Hardware-Scan erzwingen und VirtIO-Geräte verifizieren:

Erzwinge anschließend die erneute Erkennung des PCI-Busses und überprüfe die aktiven VirtIO-Treiber:

```
# PCI-Bus neu einlesen
pnputil.exe /scan-devices

# Erkannte VirtIO- und QEMU-Treiber prüfen
Get-PnpDevice -Class Net, SCSIAdapter, System |
    Where-Object { $_.FriendlyName -match 'Red Hat|VirtIO|QEMU' } |
    Format-Table -Property Class, FriendlyName, Status
```

Erstkonfiguration mit SConfig

Unmittelbar nach der ersten Anmeldung öffnet Server Core das menügeführte Konsolenprogramm **SConfig** (`sconfig.cmd`). Das Werkzeug wurde in modernen Windows-Server-Versionen vollständig auf Basis von PowerShell neu implementiert und bündelt die elementaren Administrationsschritte, die vor dem regulären Netzwerkbetrieb zwingend abgeschlossen sein müssen.

Die Navigation erfolgt rein numerisch: Du gibst die Ziffer der gewünschten Option ein, bestätigst mit der `Eingabetaste` und folgst den Textabfragen.

Das SConfig-Hauptmenü im Überblick

Das Konsolenmenü von Windows Server 2025 gliedert die Erstkonfiguration in 15 strukturierte Optionen:

Ziffer	Menüoption	Typischer Status / Funktion
1	Domäne/Arbeitsgruppe	`Arbeitsgruppe: WORKGROUP` (oder Domänenbeitritt)

Ziffer	Menüoption	Typischer Status / Funktion
2	Computername	`WIN-CORE01` (NetBIOS- und DNS-Hostname)
3	Lokalen Administrator hinzufügen	Zusätzliche lokale Notfall-Benutzerkonten anlegen
4	Remoteverwaltung	`Aktiviert` (WinRM, Server Manager und WAC-Schnittstellen)
5	Einstellung aktualisieren	`Nur Herunterladen` (oder Automatisch / Manuell)
6	Updates installieren	Qualität- und Sicherheits-Updates direkt suchen und einspielen
7	Remotedesktop	`Deaktiviert` (oder Aktiviert mit NLA)
8	Netzwerkeinstellungen	IP-Adressen, Subnetzmasken, Gateways und DNS-Server pflegen
9	Datum und Uhrzeit	Zeitzone und NTP-Konfiguration anpassen
10	Diagnosedateneinstellung	`Erforderlich` (Telemetriestufe)
11	Windows-Aktivierung	Lizenzstatus abfragen oder Product Key hinterlegen
12	Benutzer abmelden	Aktuelle Sitzung beenden
13	Server neu starten	Geordneten Betriebssystem-Reboot ausführen
14	Server herunterfahren	Gastsystem sauber beenden und ausschalten
15	In Befehlszeile (PowerShell) beenden	SConfig schließen und in die native PowerShell wechseln

> **Autostart von SConfig steuern:** Standardmäßig startet SConfig nach jeder interaktiven Konsolenanmeldung automatisch im Vordergrund. Möchtest du nach Abschluss der Grundeinrichtung direkt in der regulären PowerShell-Konsole landen, kannst du diesen Autostart über den PowerShell-Befehl `Set-SConfig -AutoLaunch $false` dauerhaft deaktivieren (mit `$true` wird er wieder aktiviert). SConfig selbst lässt sich jederzeit manuell über den Befehl `sconfig` aufrufen.

Computername und Arbeitsgruppe anpassen

Ab Werk generiert das Windows-Setup einen zufälligen Computernamen wie `WIN-7K2M9LPQ3R1`. Für einen geordneten Rechenzentrumsbetrieb vergibst du hier die im Labor geplante Kennung:

1. Wähle im Hauptmenü die Ziffer `2` (*Computername*) und drücke `Enter`.
2. Gib den neuen Computernamen ein (in unserer dokumentierten Test-VM: `WIN-CORE01`).
3. SConfig fragt, ob das System sofort neu gestartet werden soll. Wähle hier zunächst **`Nein (N)`**, um die Netzwerkeinstellungen ohne zwischenzeitliche Wartezeit im selben Durchgang abzuschließen.

Die Option `1` (*Domäne/Arbeitsgruppe*) belässt du für diesen Guide standardmäßig auf der Arbeitsgruppe `WORKGROUP`.

Netzwerkeinstellungen: Statische IPv4-Adresse und DNS konfigurieren

Standardmäßig bezieht Windows Server Core IP-Adressen dynamisch über DHCP. Für einen Serverbetrieb ist eine feste Adressierung zwingend erforderlich.

1. Wähle im Hauptmenü die Option `8` (*Netzwerkeinstellungen*).
2. SConfig listet alle erkannten Netzwerkadapter mit ihrem Index, dem Schnittstellennamen und der aktuellen IP-Konfiguration auf. Bei Verwendung der Intel E1000 oder VirtIO-Netzwerkkarte wird der Adapter typischerweise mit dem Index `1` geführt.
3. Gib den entsprechenden **Index des Adapters** ein (z. B. `1`) und bestätige mit `Enter`.

Nun öffnet sich das Untermenü für den ausgewählten Netzwerkadapter:

- **Statische IP-Adresse vergeben:**

1. Wähle die Option `1` (*Netzwerkadapteradresse festlegen*).
2. Auf die Frage nach statischer Adresse oder DHCP wählst du `S` (*Statisch*).
3. Gib die statische IP-Adresse ein. Verwende für Labore neutrale Adressbereiche, beispielsweise: `192.0.2.10`
4. Gib die Subnetzmaske an (beispielsweise `255.255.255.0` für ein `/24`-Netz).
5. Gib das Standard-Gateway deines Subnetzes an: `192.0.2.1`

- **DNS-Server konfigurieren:**

1. Wähle die Option `2` (*DNS-Server festlegen*).

2. Gib die primäre IPv4-Adresse deines internen oder vorgelagerten DNS-Resolvers ein (beispielsweise `192.0.2.1`).
3. Die Abfrage nach einem sekundären DNS-Server kannst du entweder mit einem zweiten Resolver belegen oder mit `Enter` überspringen.

- **Zurück zum Hauptmenü:** Wähle die Option `4`, um zur Menü-Startseite zurückzukehren.

Remoteverwaltung, Remotedesktop und Windows Updates

Vor dem Abschluss der Erstkonfiguration konfigurierst du die verbleibenden Basisdienste:

- **Option 4 (Remoteverwaltung):** Stelle sicher, dass die Remoteverwaltung auf *Aktiviert* steht. Dadurch richtet Windows die Listener für WinRM ein, was für administrative Abfragen anderer Windows-Clients oder des Windows Admin Centers hilfreich ist.
- **Option 7 (Remotedesktop):** Auf Server Core kann RDP bei Bedarf aktiviert werden (Auswahl `E` für *Aktivieren*, gefolgt von Option `1` für *Nur Clients mit Authentifizierung auf Netzwerkebene / NLA*).

> □□ **Remotedesktop ohne Explorer-Shell:** Aktivierst du RDP auf Windows Server Core, öffnet sich bei einer Verbindung über einen RDP-Client (wie FreeRDP, Remmina oder mstsc) keine grafische Desktop-Oberfläche. Du landest in einem bildschirmfüllenden, schwarzen Fenster, das exakt ein Konsolenfenster (`cmd.exe`) anzeigt. Für Linux-Administratoren bietet OpenSSH über das Terminal eine deutlich performantere und flexiblere Verbindung.

- **Option 5 & 6 (Windows Updates):** Setze das Update-Verhalten unter Option 5 idealerweise auf *Manuell* (`M`) oder *Nur Download* (`D`), um zu verhindern, dass die virtuelle Maschine während späterer Konfigurationsschritte unkontrolliert Hintergrund-Downloads startet. Über Option 6 kannst du bei bestehender Internetverbindung sofort nach aktuellen Qualitäts- und Sicherheitsupdates suchen.
- **Option 11 (Windows-Aktivierung):** Hier lässt sich der aktuelle Aktivierungsstatus überprüfen. Eine Evaluation-Version läuft ab Installation für 180 Tage ohne Eingabe eines Lizenzschlüssels.

Geordneter Neustart und Übergang zur Shell

Nachdem der Computername geändert und die IP-Konfiguration abgeschlossen ist, muss das System neu gestartet werden, damit alle Netzwerkstacks, NetBIOS-Namen und Windows-Sicherheits-Identifier (SID) sauber initialisiert werden.

1. Wähle im SConfig-Hauptmenü die Ziffer `13` (*Server neu starten*).
2. Bestätige die Sicherheitsabfrage mit `Ja` **(** `J` **)**.

> ⚠ **Neustart zwingend vor der SSH-Einrichtung:** Überspringe diesen Neustart nicht. Wenn du den SSH-Server vor der Namensänderung installierst oder startest, generiert der Dienst Host-Schlüssel und Firewallregeln unter dem temporären Zufallsnamen. Der Neustart stellt sicher, dass das System künftig unter `WIN-CORE01` im Subnetz erreichbar ist.

Nach dem Reboot meldest du dich erneut über die noVNC-Konsole als `Administrator` mit deinem gesetzten Kennwort an. Sobald SConfig wieder auf dem Bildschirm erscheint, wählst du:

`15` (*In Befehlszeile (PowerShell) beenden*)

SConfig schließt sich, und du befindest dich in der nativen PowerShell-Eingabeaufforderung.

SConfig versus native PowerShell: Die vollständige Referenzmatrix

Für Administratoren, die Routineaufgaben skripten oder Konfigurationsmanagement-Werkzeuge (wie Ansible) einsetzen, ist das interaktive SConfig-Menü oft ein Zwischenschritt. Hinter jedem Menüpunkt von SConfig steht ein nativer PowerShell-Befehl, ein WMI/CIM-Aufruf oder ein Registrierungsschlüssel.

Die folgende Matrix stellt alle 15 Optionen von SConfig dem jeweiligen nativen PowerShell-Befehl gegenüber:

Nr.	SConfig-Option	Nativer PowerShell-Befehl / API
1	Domäne/Arbeitsgruppe	`Add-Computer -WorkgroupName "WORKGROUP"` bzw. `-DomainName "corp.example.com"`

Nr.	SConfig-Option	Nativer PowerShell-Befehl / API
2	Computername	`Rename-Computer -NewName "WIN-CORE01" -Force -Restart`
3	Lokalen Administrator hinzufügen	`New-LocalUser -Name "secadmin" -Password $Password; Add-LocalGroupM -Group "Administratoren" -Member "secadmin"`
4	Remoteverwaltung	`Configure-SMRemoting.exe -enable` oder `Enable-PSRemoting -Force`
5	Windows Update-Einstellungen	`(New-Object -ComObject "Microsoft.Update.AutoUpdate").Settings.NotificationLevel = 2`
6	Updates installieren	`Install-WindowsUpdate -MicrosoftUpdate -AcceptAll -AutoReboot`
7	Remotedesktop (RDP)	`Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server' -Name 'fDenyTSConnections' -Value 0`
8	Netzwerkeinstellungen	`New-NetIPAddress -InterfaceAlias 'Ethernet0' -IPAddress 192.0.2.10 PrefixLength 24 -DefaultGateway 192.0.2.1`
9	Datum und Uhrzeit	`Set-TimeZone -Id "W. Europe Standard Time"; w32tm /resync`
10	Diagnosedateneinstellung	`Set-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\DataCollection& -Name 'AllowTelemetry' -Value 1`
11	Windows-Aktivierung	`slmgr.vbs /ipk <ProductKey>; slmgr.vbs /ato`
12	Benutzer abmelden	`logoff.exe`
13	Server neu starten	`Restart-Computer -Force`
14	Server herunterfahren	`Stop-Computer -Force`
15	In Befehlszeile beenden	`exit`

Automatisierte Erstkonfiguration per PowerShell-Skript

Anstelle der manuellen Menüführung in SConfig lässt sich die gesamte Erstkonfiguration in einem einzigen, fehlergeprüften PowerShell-Skript bündeln. Dies ist der empfohlene Weg, wenn mehrere Server-Core-Instanzen identisch aufgesetzt werden sollen.

□□ Praktisches Beispiel: Post-Deployment-Skript Initialize-ServerCore.ps1

Führe das folgende Skript direkt nach dem ersten Login an der PowerShell-Konsole aus:

```
# Initialize-ServerCore.ps1
# Vollständige Headless-Erstkonfiguration für Windows Server 2025 Core

[CmdletBinding()]
param (
    [string]$NewComputerName = "WIN-CORE01",
    [string]$InterfaceAlias = "Ethernet0",
    [string]$StaticIP = "192.0.2.10",
    [int]$PrefixLength = 24,
    [string]$Gateway = "192.0.2.1",
    [string[]]$DnsServers = @("192.0.2.1", "1.1.1.1"),
    [string]$TimeZoneId = "W. Europe Standard Time"
)

Write-Host "=== Starte automatisierte Erstkonfiguration ===" -ForegroundColor Cyan

# 1. Zeitzone konfigurieren
Write-Host "[1/6] Setze Zeitzone auf $TimeZoneId..." -ForegroundColor Yellow
Set-TimeZone -Id $TimeZoneId

# 2. SConfig-Autostart deaktivieren
Write-Host "[2/6] Deaktiviere automatischen SConfig-Start..." -ForegroundColor Yellow
Set-SConfig -AutoLaunch $false

# 3. Statische IP-Adresse und Gateway konfigurieren
Write-Host "[3/6] Konfiguriere Netzwerkadapter $InterfaceAlias
($StaticIP/$PrefixLength)..." -ForegroundColor Yellow
# Eventuell vorhandene alte Adressen bereinigen
Get-NetIPAddress -InterfaceAlias $InterfaceAlias -AddressFamily IPv4 -ErrorAction
SilentlyContinue |
    Remove-NetIPAddress -Confirm:$false

New-NetIPAddress -InterfaceAlias $InterfaceAlias -IPAddress $StaticIP -PrefixLength
$PrefixLength -DefaultGateway $Gateway

# 4. DNS-Server zuweisen
Write-Host "[4/6] Trage DNS-Resolver ein..." -ForegroundColor Yellow
Set-DnsClientServerAddress -InterfaceAlias $InterfaceAlias -ServerAddresses $DnsServers
```

```
# 5. Telemetrie auf Minimum (Erforderlich / Wert 1) setzen
Write-Host "[5/6] Setze Diagnosedateneinstellung auf Minimum..." -ForegroundColor Yellow
Set-ItemProperty -Path
'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\DataCollection' -Name
'AllowTelemetry' -Value 1 -Type DWord

# 6. Computername setzen und Neustart vorbereiten
$CurrentName = $env:COMPUTERNAME
if ($CurrentName -ne $NewComputerName) {
    Write-Host "[6/6] Benenne System von '$CurrentName' in '$NewComputerName' um..." -
ForegroundColor Yellow
    Rename-Computer -NewName $NewComputerName -Force
    Write-Host "=== Erstkonfiguration abgeschlossen. System startet in 5 Sekunden neu...
===" -ForegroundColor Green
    Start-Sleep -Seconds 5
    Restart-Computer -Force
} else {
    Write-Host "=== Erstkonfiguration abgeschlossen. Kein Neustart erforderlich. ===" -
ForegroundColor Green
}
```

Nach Ausführung des Skripts startet der Server automatisch neu. Danach ist der Hostname, die feste Netzwerkadresse und die Autostart-Einstellung aktiv.

Sprache, Region und Tastaturlayout per PowerShell konfigurieren

Auf Windows Server Core steht die bekannte Systemsteuerung (*„Region und Sprache“*) nicht zur Verfügung. Wurde das System versehentlich mit einem englischen Tastaturlayout oder abweichenden Formateinstellungen installiert, lässt sich das Lokalisierungsmodell direkt über das Modul `International` anpassen.

1. Aktuelle Spracheinstellungen abfragen:

```
Get-WinUserLanguageList | Format-Table -Property LanguageTag, Autonym, EnglishName
```

```
Get-Culture | Format-Table -Property Name, DisplayName
Get-WinSystemLocale | Format-Table -Property Name, DisplayName
```

2. Deutsches Tastaturlayout und Regionseinstellungen erzwingen:

Um die deutsche Tastaturbelegung (QWERTZ) und das deutsche Datums- und Währungsformat für alle Benutzer und Systemdienste festzulegen:

```
# Neue Sprachliste mit deutschem Layout initialisieren
$LangList = New-WinUserLanguageList -Language "de-DE"
Set-WinUserLanguageList -LanguageList $LangList -Force

# Systemgebietsschema (für Nicht-Unicode-Programme) auf Deutsch setzen
Set-WinSystemLocale -SystemLocale "de-DE"

# Kultur- und Formatierungsstandards (Datum, Zahlen, Währung) anpassen
Set-Culture -CultureInfo "de-DE"

# Geografischen Standort auf Deutschland (GeoID 94) setzen
Set-WinHomeLocation -GeoId 94
```

Nach der Neuanmeldung oder einem Neustart ist das deutsche Tastaturlayout auch in noVNC- und RDP-Konsolenfenstern dauerhaft aktiv. Nun sind alle Voraussetzungen geschaffen, um OpenSSH einzurichten.

OpenSSH Server installieren und konfigurieren

Nachdem SConfig über Option 15 beendet wurde, liegt die Kontrolle bei der Windows PowerShell an der noVNC-Konsole. Um den Server künftig ohne Hypervisor-Konsole aus dem Linux-Terminal zu verwalten, richten wir nun den integrierten OpenSSH-Dienst ein.

Im Gegensatz zu WinRM, das auf HTTP/SOAP aufbaut und außerhalb von Active-Directory-Domänen komplexe HTTPS-Zertifikate für die gegenseitige Authentifizierung oder `TrustedHosts`-

Ausnahmen ohne Identitätsprüfung verlangt, arbeitet OpenSSH nach dem von Linux bekannten Prinzip: verschlüsselte Transportkanäle über Port 22, Host-Key-Verifikation und native Unterstützung für moderne SSH-Schlüssel.

Der Bereitstellungsablauf gliedert sich in folgende Schritte:

1. **Status prüfen** OpenSSH-Capability und Dienstzustand in PowerShell abfragen
2. **Dienst & Autostart aktivieren** Starttyp auf Automatic setzen und sshd-Dienst starten
3. **Firewall & Profil verifizieren** Regeln für Port 22 prüfen und Netzwerkprofil evaluieren
4. **Fernzugriff von Linux etablieren** Verbindung per ssh herstellen und PowerShell als Default-Shell setzen

Verfügbarkeit und Prüfung des Installationsstatus

Unter Windows Server 2025 ist OpenSSH Server standardmäßig im Betriebssystem-Image integriert. Ob das Feature aktiv und einsatzbereit ist, prüfst du über das Cmdlet `Get-WindowsCapability`:

```
Get-WindowsCapability -Online -Name OpenSSH.Server*
```

Liefert die Ausgabe `State : Installed`, ist das Paket vollständig vorhanden.

Sollte das Feature auf einem minimal angepassten Image wider Erwarten als `NotPresent` gemeldet werden, installierst du es mit folgendem PowerShell-Befehl nach:

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

Dienststart, Autostart und Statusprüfung

Aus Sicherheitsgründen wird der Dienst `sshd` nach der Betriebssysteminstallation nicht automatisch ausgeführt. Er befindet sich im Status `Stopped` mit dem Starttyp `Manual` oder `Disabled`.

1. Starttyp auf automatisch festlegen:

```
Set-Service -Name sshd -StartupType Automatic
```

2. OpenSSH-Dienst starten:

```
Start-Service -Name sshd
```

3. Dienstzustand und Socket überprüfen:

Verifiziere, dass der Prozess aktiv läuft und der TCP-Port 22 lokal lauscht:

```
Get-Service -Name sshd | Format-Table -Property Name, DisplayName, Status, StartType
```

Erwartete Ausgabe:

```
Name Status  StartType DisplayName
---- ------  --------- -----------
sshd Running Automatic OpenSSH SSH Server
```

Prüfe zusätzlich, ob der Socket an Port 22 gebunden ist:

```
Get-NetTCPConnection -LocalPort 22 -State Listen | Format-Table -Property LocalAddress,
LocalPort, State
```

> **Host-Schlüssel werden beim Erststart erzeugt:** Beim allerersten Start des Dienstes `sshd` generiert Windows OpenSSH automatisch die serverspezifischen Host-Keys (RSA, ECDSA und Ed25519) und legt sie im geschützten Verzeichnis `C:\ProgramData\ssh\` ab.

Firewallregeln und Windows-Netzwerkprofile

Damit eingehende SSH-Verbindungen von anderen Rechnern nicht an der Windows Defender Firewall scheitern, müssen die Firewallregeln und das zugewiesene Netzwerkprofil zusammenpassen.

Windows unterscheidet drei Sicherheitszonen für Netzwerkschnittstellen:

- **`Public` (Öffentlich):** Standardprofil für neu erkannte Adapter. Blockiert fast alle eingehenden Verbindungsanfragen und deaktiviert die Netzwerkerkennung.

- `Private` **(Privat):** Für vertrauenswürdige interne Netzwerke hinter einem Router oder einer Firewall. Erlaubt administrative Freigaben und ICMP-Echoforderungen (Ping).
- `DomainAuthenticated`**:** Greift automatisch, sobald der Server einer Active-Directory-Domäne beitritt und den Domänencontroller kontaktieren kann.

Prüfe das aktuelle Profil deines Netzwerkadapters:

```
Get-NetConnectionProfile
```

> ⚠ **Sicherheitsfolgen des Profilwechsels:** Ein pauschaler Wechsel von `Public` auf `Private` öffnet neben SSH auch andere vordefinierte Windows-Dienste (wie Dateifreigaben oder NetBIOS) für das gesamte Subnetz. Ändere das Profil nur dann auf `Private`, wenn sich der Server in einem isolierten Administrations- oder Labor-VLAN befindet. In geteilten Netzwerken belässt du das Profil auf `Public` und passt stattdessen gezielt die SSH-Firewallregel an.

Für isolierte Testumgebungen stellst du das Profil über die PowerShell wie folgt auf privat um:

```
Set-NetConnectionProfile -InterfaceAlias "Ethernet" -NetworkCategory Private
```

Prüfe, für welche Profile die OpenSSH-Firewallregel freigeschaltet ist:

```
Get-NetFirewallRule -Name *OpenSSH-Server* | Format-Table -Property Name, DisplayName,
Enabled, Profile, Action
```

Sollte die Regel `OpenSSH-Server-In-TCP` nur für `Private` gelten, du den Adapter aber auf `Public` betreiben möchtest, bindest du die Regel gezielt an alle Profile:

```
Set-NetFirewallRule -Name "OpenSSH-Server-In-TCP" -Profile Any -Enabled True
```

SSH-Zugriff von einem Linux-System herstellen

Der administrative Zugriff erfolgt nun von deiner Linux-Workstation aus über den Standard-OpenSSH-Client. Du benötigst die zuvor in SConfig vergebene IP-Adresse (in unserem Beispiel `192.0.2.10`):

```
ssh Administrator@192.0.2.10
```

1. **Host-Key bestätigen:**

Beim ersten Verbindungsaufbau meldet der SSH-Client den noch unbekannten Fingerprint des Windows-Servers. Bestätige die Vertrauensstellung mit `yes`.

1. **Kennwort eingeben:**

Gib das während der Windows-Installation vergebene Kennwort des Benutzers `Administrator` ein.

Nach erfolgreicher Authentifizierung öffnet sich die Windows-Kommandozeile. Du bist nun direkt über SSH mit Windows Server 2025 Core verbunden – noVNC wird für die weitere Administration nicht mehr benötigt.

PowerShell als Standard-Shell konfigurieren

Standardmäßig startet Windows OpenSSH eine klassische Eingabeaufforderung (`cmd.exe`). Im Terminal siehst du den Prompt:

```
C:\Users\Administrator>
```

Zwar kannst du jederzeit durch die Eingabe von `powershell` in die PowerShell wechseln, für eine konsistente Fernverwaltung ist es jedoch ergonomischer, wenn SSH direkt eine PowerShell-Sitzung bereitstellt.

Führe folgenden Befehl auf der Windows-Maschine (entweder noch in noVNC oder bereits in der laufenden SSH-Sitzung) aus, um die Standard-Shell in der Registrierung auf Windows PowerShell festzulegen:

```
New-ItemProperty -Path "HKLM:\SOFTWARE\OpenSSH" `
  -Name DefaultShell `
  -Value "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" `
  -PropertyType String -Force
```

Trennst du die SSH-Verbindung mit `exit` und meldest dich erneut von Linux aus an, begrüßt dich sofort der gewohnte PowerShell-Prompt:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

PS C:\Users\Administrator>
```

Schlüsselbasierte Authentifizierung für Administratoren

Die Kennwort-Authentifizierung ist für die Ersteinrichtung zweckmäßig, im professionellen Betrieb jedoch fehleranfällig und ungeeignet für Automatisierungen mit Ansible oder Shell-Skripten.

Unter Windows OpenSSH existiert für Mitglieder der lokalen Administratorgruppe eine wichtige Sicherheitsarchitektur, die sich grundlegend von Standard-Linux-Systemen unterscheidet:

- **Reguläre Windows-Benutzer:** Verwenden wie unter Linux die Datei `C:\Users\<Benutzername>\.ssh\authorized_keys`.
- **Lokale Administratoren:** Windows ignoriert die Datei im Benutzerprofil und verlangt zwingend die globale Datei:

`C:\ProgramData\ssh\administrators_authorized_keys`

> **Rechtefalle bei administrators_authorized_keys:** Der SSH-Dienst verweigert die schlüsselbasierte Anmeldung strikt, wenn die Datei `administrators_authorized_keys` Berechtigungen für unberechtigte Konten enthält. Ausschließlich die Gruppe `SYSTEM` und die lokalen `Administratoren` dürfen Lese- und Schreibzugriff besitzen; alle vererbten Berechtigungen müssen entfernt werden.

Praktisches Beispiel: SSH-Public-Key von Linux hinterlegen

Kopiere den öffentlichen Schlüssel deines Linux-Clients (z. B. `~/.ssh/id_ed25519.pub`) auf den Windows-Server. In einer bestehenden PowerShell-Sitzung richtest du die Datei mit den erforderlichen Windows-Zugriffssteuerungslisten (ACLs) wie folgt ein:

```
# Public-Key eintragen (Beispiel-Key durch deinen echten Schlüssel ersetzen)
$PublicKey = "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIG... user@linux-workstation"
$AuthKeysPath = "$env:ProgramData\ssh\administrators_authorized_keys"

# Datei anlegen und Schlüssel schreiben
Set-Content -Path $AuthKeysPath -Value $PublicKey -Encoding Ascii

# Vererbung deaktivieren und Rechte auf SYSTEM und Administrators beschränken
icacls.exe $AuthKeysPath /inheritance:r /grant "NT AUTHORITY\SYSTEM:(F)" /grant
"BUILTIN\Administrators:(F)"
```

Anschließend gelingt der Verbindungsaufbau von deiner Linux-Workstation ohne Kennwortabfrage:

```
ssh -i ~/.ssh/id_ed25519 Administrator@192.0.2.10
```

Damit steht dir ein vollständig abgesicherter, schlüsselbasierter Zugang zur Verfügung.

Konfiguration und Härtung des SSH-Daemons (sshd_config)

Die zentrale Konfigurationsdatei des OpenSSH-Dienstes liegt unter Windows im geschützten Verzeichnis `C:\ProgramData\ssh\sshd_config`. Nach erfolgreicher Einrichtung des Schlüsselpaars sollte die Konfiguration gehärtet werden, um Brute-Force-Angriffe und schwache kryptografische Verfahren auszuschließen.

1. Kennwort-Authentifizierung vollständig deaktivieren:

Um sicherzustellen, dass Anmeldungen ausschließlich mit autorisierten SSH-Schlüsseln möglich sind, bearbeitest du die Datei `C:\ProgramData\ssh\sshd_config`. Unter PowerShell lässt sich die Datei wie folgt modifizieren:

```
$ConfigPath = "$env:ProgramData\ssh\sshd_config"
$Config = Get-Content $ConfigPath

# PasswordAuthentication auf 'no' setzen
$Config = $Config -replace '^#?PasswordAuthentication\s+.*', 'PasswordAuthentication no'

# PubkeyAuthentication explizit auf 'yes' forcieren
$Config = $Config -replace '^#?PubkeyAuthentication\s+.*', 'PubkeyAuthentication yes'

# LogLevel auf VERBOSE für forensische Auditierung setzen
$Config = $Config -replace '^#?LogLevel\s+.*', 'LogLevel VERBOSE'

# Änderungen zurückschreiben
Set-Content -Path $ConfigPath -Value $Config -Encoding Ascii
```

2. Benutzerkreis auf autorisierte Administratoren beschränken:

Verhindere, dass andere lokale oder später über Domänen beigetretene Konten eine SSH-Sitzung initiieren können, indem du die Direktive `AllowUsers` am Ende der Konfigurationsdatei anhängst:

```
Add-Content -Path $ConfigPath -Value "`nAllowUsers Administrator secadmin"
```

3. Kryptografische Verfahren härten (Ciphers und KEX):

Windows Server 2025 unterstützt moderne kryptografische Schlüsselaustauschverfahren, einschließlich Post-Quantum-Hybriden (`sntrup761`). Veraltete Verfahren wie Diffie-Hellman-Group1 oder SHA-1-basierte MACs sollten explizit ausgeschlossen werden.

Ergänze die `sshd_config` um folgende gehärtete Direktiven:

```
# Gehärtete KEX- und Verschlüsselungsverfahren für OpenSSH unter Windows Server 2025
KexAlgorithms sntrup761x25519-sha512@openssh.com,curve25519-sha256,diffie-hellman-group16-sha512,diffie-hellman-group18-sha512
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com
```

4. Syntaxprüfung und Dienstneustart:

Vor dem Neustart des Dienstes sollte die Konfigurationsdatei stets auf Syntaxfehler überprüft werden, um ein Aussperren zu verhindern:

```
# Syntaxprüfung der Konfiguration
& "C:\Windows\System32\OpenSSH\sshd.exe" -t

# Bei fehlerfreier Rückgabe den Dienst neu starten
Restart-Service sshd
```

> **Bestehende SSH-Sitzung offen halten:** Schließe deine aktive SSH-Sitzung während der Härtung niemals vor einem Test. Öffne auf deinem Linux-Client ein zweites Terminalfenster und teste den Verbindungsaufbau: `ssh -i ~/.ssh/id_ed25519 Administrator@192.0.2.10`. Erst wenn der Login im zweiten Terminal fehlerfrei gelingt, ist die Härtung sicher abgeschlossen.

SSH-Verbindungsanalyse und Fehlersuche im Windows Event Log

Unter Windows Server schreibt der OpenSSH-Dienst betriebliche Meldungen und fehlgeschlagene Authentifizierungsversuche in ein eigenes dediziertes Windows-

Ereignisprotokoll: `OpenSSH/Operational`.

Tritt beim Verbindungsaufbau ein Problem auf (beispielsweise wegen fehlerhafter Berechtigungen auf `administrators_authorized_keys`), findest du die Ursache nicht in Textdateien unter `/var/log`, sondern fragst das Protokoll über PowerShell ab:

□□ Praktisches Beispiel: OpenSSH-Fehlerprotokolle in Echtzeit auswerten

```
# Die letzten 20 Ereignisse des OpenSSH-Servers abrufen
Get-WinEvent -LogName "OpenSSH/Operational" -MaxEvents 20 |
    Select-Object TimeCreated, Id, LevelDisplayName, Message |
    Format-Table -Wrap
```

Häufige Fehlermuster und deren Ursache:

- **Event-ID 4 (Warning / Error):** *„Authentication refused: bad ownership or modes for file...“*

$\rightarrow$ Berechtigungen der Datei `administrators_authorized_keys` sind fehlerhaft. Wiederhole den im vorherigen Abschnitt beschriebenen `icacls`-Befehl, um fremde Konten zu entfernen.

- **Event-ID 130 (Information):** *„Accepted key RSA/ED25519...“*

$\rightarrow$ Erfolgreicher schlüsselbasierter Verbindungsaufbau mit Angabe des Fingerprints.

- **Event-ID 131 (Information):** *„Failed publickey for Administrator...“*

$\rightarrow$ Der vom SSH-Client präsentierte öffentliche Schlüssel ist nicht in `administrators_authorized_keys` hinterlegt oder der Schlüsseltyp wurde in der Konfiguration blockiert.

Sichere Dateiübertragung mit SFTP und SCP

Ein gravierender Nachteil von WinRM ist die umständliche Übertragung größerer Binärdateien (wie Treiberpakete, Installer oder ISO-Images). Mit OpenSSH steht auf Windows Server 2025 automatisch ein vollwertiges SFTP- und SCP-Subsystem zur Verfügung.

Der SFTP-Serverdienst wird durch die Binärdatei `sftp-server.exe` im OpenSSH-Verzeichnis bereitgestellt und ist in der Standard-`sshd_config` bereits vorkonfiguriert:

```
Subsystem sftp sftp-server.exe
```

Dateien von der Linux-Workstation auf den Server kopieren (SCP):

Das Kopieren von Konfigurationsdateien oder Skripten auf den Windows-Server erfolgt mit dem vertrauten `scp`-Befehl. Pfade auf dem Windows-System werden mit Vorwärts-Slashes (`/`) angegeben:

```
# Einzelnes Skript auf das Windows-Laufwerk C: übertragen
scp -i ~/.ssh/id_ed25519 setup-app.ps1 Administrator@192.0.2.10:C:/Scripts/

# Verzeichnis rekursiv übertragen
scp -i ~/.ssh/id_ed25519 -r ./install-packages/ Administrator@192.0.2.10:C:/Setup/
```

Interaktive Dateiübertragung via SFTP:

Für komplexere Dateioperationen öffnest du eine interaktive SFTP-Sitzung:

```
sftp -i ~/.ssh/id_ed25519 Administrator@192.0.2.10
```

Innerhalb der SFTP-Shell navigierst du auf dem Windows-Dateisystem:

```
Connected to 192.0.2.10.
sftp> pwd
Remote path: /C:/Users/Administrator
sftp> cd C:/Shares/Deploy
sftp> put package.zip
Uploading package.zip to /C:/Shares/Deploy/package.zip
package.zip                                   100%  142MB  48.2MB/s   00:02
sftp> bye
```

SSH-Tunneling und Port-Forwarding für administrative Dienste

Auf einem Server Core sollten interne Webdienste oder Verwaltungsschnittstellen nicht unverschlüsselt auf allen Netzwerkschnittstellen lauschen. Mit SSH-Port-Forwarding (Local Port Forwarding) tunnelst du interne Ports sicher über die verschlüsselte SSH-Verbindung auf deine lokale Linux-Workstation.

□□ Praktisches Beispiel: Internen Webdienst sicher über SSH tunneln

Läuft auf dem Server ein interner Webdienst oder ein Verwaltungsportal auf Port `80` oder `443`, das nur an `127.0.0.1` gebunden ist, leitest du den Port über deine SSH-Sitzung weiter:

```
# Leitet Port 8080 der lokalen Linux-Maschine auf Port 80 von WIN-CORE01 weiter
```

```
ssh -i ~/.ssh/id_ed25519 -L 8080:localhost:80 Administrator@192.0.2.10
```

Solange diese SSH-Verbindung aktiv ist, öffnest du auf deiner Linux-Workstation im Webbrowser die Adresse `http://localhost:8080` und greifst direkt auf den internen Webserver des Windows-Systems zu – ohne dass Port 80 in der Windows-Firewall nach außen geöffnet werden muss.

Windows Server Core mit PowerShell administrieren

Sobald die SSH-Verbindung aufgebaut ist oder du dich an der physischen bzw. virtuellen Konsole anmeldest, arbeitest du direkt in der PowerShell-Eingabeaufforderung. Auf Windows Server Core ist die Shell kein nachgelagertes Hilfswerkzeug, sondern die primäre Schnittstelle für Diagnose, Konfiguration, Speicherverwaltung und Betrieb sämtlicher Systemdienste.

Windows PowerShell 5.1 und PowerShell 7 im Vergleich

Windows Server 2025 liefert standardmäßig die integrierte **Windows PowerShell 5.1** aus. Diese Version basiert auf dem klassischen .NET Framework 4.8 und ist fest mit dem Betriebssystem sowie allen administrativen WMI/CIM-Modulen verzahnt.

Parallel dazu existiert **PowerShell 7** (ehemals PowerShell Core), das auf modernem .NET (Core) basiert, plattformübergreifend arbeitet und kontinuierlich weiterentwickelt wird.

Windows PowerShell 5.1

Basis
: .NET Framework 4.8

Installation
: Vorinstalliert im Betriebssystem

Modulkompatibilität
: Vollständige Abdeckung aller Server-Rollen

Typischer Einsatz
: Native Systemadministration out-of-the-box

PowerShell 7 (Core)

Basis
: .NET 8 / 9

Installation
: Optionales Paket (MSI oder winget)

Modulkompatibilität
: Sehr hoch (teils via WinPS-Kompatibilitätslayer)

Typischer Einsatz
: DevOps, moderne CI/CD-Pipelines, Cloud

Gemeinsam: Identische Syntax, Cmdlet-Struktur und Objekt-Pipeline

Für die grundlegende Systemadministration von Server Core genügt die vorinstallierte Windows PowerShell 5.1 vollständig. Alle in dieser Anleitung behandelten Befehle und Module laufen nativ ohne Nachinstallation zusätzlicher Frameworks.

Das Objektmodell der Pipeline verstehen

Der entscheidende Unterschied zwischen Unix-Shells (wie Bash oder Zsh) und der PowerShell liegt in der Datenverarbeitung: Während klassische Shells unstrukturierte Byteströme (Text) übergeben, die mit `grep`, `awk` oder `cut` geparst werden müssen, transportiert die PowerShell-Pipeline **strukturierte .NET-Objekte**.

Jedes Objekt besitzt typisierte Eigenschaften (*Properties*) und Methoden (*Methods*). Filter- und Sortieroperationen greifen direkt auf diese Werte zu, ohne dass Textausgaben zerlegt werden müssen.

Wichtige Kern-Cmdlets für den Datenfluss:

- `Where-Object` (Alias `?`): Filtert Objekte anhand logischer Bedingungen.
- `Select-Object` (Alias `select`): Wählt spezifische Eigenschaften aus oder begrenzt die Anzahl der Treffer (`-First`, `-Last`).
- `Sort-Object` (Alias `sort`): Sortiert Objekte nach einer oder mehreren Eigenschaften auf- oder absteigend.
- `Measure-Object` (Alias `measure`): Berechnet Anzahl, Summe, Durchschnitt, Minimal- und Maximalwerte von Eigenschaften.
- `Format-Table` (Alias `ft`) und `Format-List` (Alias `fl`): Bereiten Objekte für die menschliche Lesbarkeit auf dem Bildschirm auf.

> ⚠ **Formatierungs-Regel für die Pipeline:** Die Cmdlets `Format-Table` und `Format-List` zerstören das ursprüngliche .NET-Objekt und erzeugen stattdessen interne Steuerungsstrukturen für die Bildschirmausgabe. Verwende `Format-*` daher ausnahmslos als allerletzten Befehl am Ende einer Befehlskette und niemals vor nachfolgenden Filter- oder Verarbeitungsstufen.

□□ Praktisches Beispiel: Pipeline-Filterung versus Textausgabe

Der Unterschied wird bei der Abfrage von Diensten deutlich. Statt einer unstrukturierten Textsuche prüfst du direkt die Status-Eigenschaft des Objekts:

```
# Objektbasiert filtern und nur relevante Spalten ausgeben
Get-Service | Where-Object Status -eq 'Running' | Select-Object -Property Name,
DisplayName, StartType | Sort-Object Name
```

Erwartete tabellarische Ausgabe:

```
Name     DisplayName                          StartType
----     -----------                          ---------
BFE      Base Filtering Engine                Automatic
Dhcp     DHCP Client                          Automatic
EventLog Windows Event Log                    Automatic
LanmanServer Server                           Automatic
sshd     OpenSSH SSH Server                   Automatic
w32time  Windows Time                         Automatic
WinRM    Windows Remote Management (WS-Man)   Automatic
```

System- und Netzwerkinspektion

Auf einem Headless-System müssen grundlegende Systemparameter schnell und zuverlässig abgefragt werden können.

Systemstatus und Uptime:

Das Cmdlet `Get-ComputerInfo` liefert eine umfassende Zusammenstellung von Hardware-, Betriebssystem- und BIOS-Informationen. Da der Befehl sehr viele Daten ausliest, empfiehlt sich die gezielte Eingrenzung auf relevante Eigenschaften:

```
Get-ComputerInfo | Select-Object -Property CsName, WindowsProductName, WindowsVersion,
OsUptime, OsLastBootUpTime, CsTotalPhysicalMemory
```

Für eine schnelle Abfrage der Systemstartzeit und des freien Arbeitsspeichers eignet sich die WMI/CIM-Schnittstelle über `Get-CimInstance`:

```
Get-CimInstance -ClassName Win32_OperatingSystem | Select-Object -Property Caption,
LastBootUpTime, @{Name='FreeRAM_MB'; Expression={[math]::Round($_.FreePhysicalMemory /
1KB, 0)}}
```

Netzwerkkonfiguration analysieren:

Für die Diagnose der Netzwerkverbindungen stehen im Modul `NetTCPIP` spezialisierte Cmdlets bereit:

```
# Übersicht über Schnittstellen und IP-Adressen
Get-NetIPConfiguration

# Detaillierte Liste aller zugewiesenen IPv4-Adressen
Get-NetIPAddress -AddressFamily IPv4 | Format-Table -Property InterfaceAlias, IPAddress,
PrefixLength, InterfaceIndex

# Routing-Tabelle für IPv4 auslesen
Get-NetRoute -AddressFamily IPv4 | Where-Object DestinationPrefix -ne 'ff00::/8' | Sort-
Object InterfaceIndex
```

Zur Überprüfung entfernter Ports und Netzwerkrouten nutzt du `Test-NetConnection` (das moderne Äquivalent zu `ping` und `telnet`):

```
# Port-Prüfung zu einem Gateway oder DNS-Server
Test-NetConnection -ComputerName 192.0.2.1 -Port 53
```

Typische Ausgabe:

```
ComputerName     : 192.0.2.1
RemoteAddress    : 192.0.2.1
RemotePort       : 53
InterfaceAlias   : Ethernet0
SourceAddress    : 192.0.2.10
TcpTestSucceeded : True
```

Netzwerkkarten-Teaming (NIC Teaming) für Hochverfügbarkeit:

Werden der virtuellen Maschine in Proxmox VE mehrere Netzwerkkarten auf unterschiedlichen Bridges zugewiesen, lassen sich diese auf Server Core ohne Drittanbieter-Tools zu einem ausfallsicheren Verbund (*NIC Team*) bündeln:

```
# Switch-unabhängiges Teaming mit dynamischem Lastenausgleich anlegen
New-NetLbfoTeam -Name "Team01" `
  -TeamMembers "Ethernet0", "Ethernet1" `
  -TeamingMode SwitchIndependent `
  -LoadBalancingAlgorithm Dynamic

# Status des aktiven Teams überprüfen
Get-NetLbfoTeam | Format-Table -Property Name, Members, TeamingMode, Status
```

Umgebungsvariablen und System-PATH dauerhaft anpassen:

Auf einem Headless-System entfällt der bekannte Systemeinstellungs-Dialog für Umgebungsvariablen. Änderungen an `$env:PATH` in einer laufenden PowerShell gelten nur flüchtig für den aktuellen Prozess. Um Systempfade (beispielsweise für eigene Skriptverzeichnisse wie `C:\Scripts`) dauerhaft auf Maschinenebene zu verankern, nutzt du die .NET-Klasse `System.Environment`:

```
# Aktuellen systemweiten PATH abrufen
$CurrentPath = [System.Environment]::GetEnvironmentVariable("Path", "Machine")

# Prüfen und neues Verzeichnis anhängen, falls noch nicht vorhanden
if ($CurrentPath -notlike "*C:\Scripts*") {
    $NewPath = "$CurrentPath;C:\Scripts"
    [System.Environment]::SetEnvironmentVariable("Path", $NewPath, "Machine")
}

# Dauerhafte globale Umgebungsvariable definieren
[System.Environment]::SetEnvironmentVariable("ADMIN_ENV", "Production", "Machine")
```

Speicher- und Partitionsverwaltung ohne diskmgmt.msc

In einer virtualisierten Umgebung wie Proxmox VE gehört das Vergrößern bestehender Festplatten oder das Hinzufügen zusätzlicher Datenträger zum Standardalltag. Ohne grafische Datenträgerverwaltung (`diskmgmt.msc`) steuerst du den gesamten Speicherstack über das PowerShell-Modul `Storage` .

Festplatten und Partitionen inventarisieren:

Verschaffe dir zunächst einen Überblick über die physischen bzw. virtuellen Datenträger und deren Partitionierungsstil (GPT oder MBR):

```
Get-Disk | Format-Table Number, FriendlyName, OperationalStatus, Size, PartitionStyle
```

Die Ausgabe zeigt die erkannten Festplatten:

```
Number FriendlyName         OperationalStatus      Size PartitionStyle
------ ------------         -----------------      ---- --------------
     0 QEMU HARDDISK        OK                   60 GB  GPT
```

Praktisches Beispiel: Virtuelle Systemfestplatte im laufenden Betrieb vergrößern

Erhöhst du auf dem Proxmox-Host die Festplattengröße der virtuellen Maschine (beispielsweise via `qm resize 104 scsi0 +20G`), wird der zusätzliche Speicherplatz unter Windows zunächst als unzugeordneter Bereich am Ende des Datenträgers geführt.

Um die Partition und das NTFS-Dateisystem ohne Neustart auf die maximale Größe zu erweitern, gehst du in drei Schritten vor:

```
# 1. Datenträger neu einlesen, um die Größenänderung des Hypervisors zu erkennen
Update-Disk -Number 0

# 2. Maximal unterstützte Größe für Partition 3 (C:) ermitteln
$MaxSize = (Get-PartitionSupportedSize -DiskNumber 0 -PartitionNumber 3).SizeMax

# 3. Partition auf das Maximum erweitern
Resize-Partition -DiskNumber 0 -PartitionNumber 3 -Size $MaxSize
```

Prüfe anschließend die Kapazität des Volumes:

```
Get-Volume -DriveLetter C | Format-Table DriveLetter, FileSystemLabel, FileSystem, Size,
SizeRemaining
```

Zusätzliche Datenfestplatte initialisieren und formatieren:

Fügst du der virtuellen Maschine eine zweite Festplatte für Nutzdaten hinzu (z. B. Datenträger `1`), richtest du diese mit einer einzigen PowerShell-Pipeline vollständig ein:

```
# Datenträger initialisieren, GPT-Partition erstellen und mit NTFS formatieren
Initialize-Disk -Number 1 -PartitionStyle GPT -PassThru |
    New-Partition -AssignDriveLetter -UseMaximumSize |
    Format-Volume -FileSystem NTFS -NewFileSystemLabel "Data" -Confirm:$false
```

Speicherpools und Software-Spiegelung (Storage Spaces):

Werden der virtuellen Maschine mehrere Festplatten für ausfallsichere Nutzdaten zugewiesen (beispielsweise Datenträger `1` und `2`), lassen sich diese ohne Drittanbieter-Tools zu einem gespiegelten Speicherpool (*Mirror Storage Space*) zusammenfassen:

```
# 1. Unpartitionierte Festplatten ermitteln, die für Pools bereitstehen
$Disks = Get-PhysicalDisk -CanPool $true

# 2. Neuen Speicherpool "DataPool" anlegen
$Pool = New-StoragePool -FriendlyName "DataPool" `
  -StorageSubsystemFriendlyName "Windows Storage*" `
  -PhysicalDisks $Disks

# 3. Gespiegeltes virtuelles Laufwerk (RAID-1-Äquivalent) im Pool erstellen
$VDisk = New-VirtualDisk -StoragePoolFriendlyName "DataPool" `
  -FriendlyName "MirrorVolume" `
  -ResiliencySettingName Mirror `
  -UseMaximumSize

# 4. Virtuellen Datenträger initialisieren, partitionieren und formatieren
$VDisk | Get-Disk | Initialize-Disk -PartitionStyle GPT -PassThru |
```

```
    New-Partition -AssignDriveLetter -UseMaximumSize |
    Format-Volume -FileSystem NTFS -NewFileSystemLabel "MirrorData" -Confirm:$false
```

Lokale Benutzer und Gruppen verwalten

Auf Windows Server Core steht das Snap-in für lokale Benutzer und Gruppen (`lusrmgr.msc`) nicht lokal zur Verfügung. Die Benutzerverwaltung erfolgt über das Modul `Microsoft.PowerShell.LocalAccounts` .

Lokale Konten auflisten und prüfen:

```
Get-LocalUser | Format-Table Name, Enabled, PasswordRequired, PasswordLastSet, LastLogon
```

Neues lokales Administratorkonto anlegen:

Für administrative Notfälle empfiehlt sich ein dediziertes lokales Benutzerkonto neben dem standardmäßigen Administrator:

```
# Sicheres Kennwort abfragen
$Password = Read-Host -AsSecureString "Kennwort für neuen Administrator eingeben"

# Benutzer anlegen
New-LocalUser -Name "secadmin" -Password $Password -FullName "Secondary Admin" -
Description "Notfall-Administrator"

# Benutzer zur lokalen Administratorengruppe hinzufügen
Add-LocalGroupMember -Group "Administratoren" -Member "secadmin"
```

Kennwortrichtlinie und Kontensperrung steuern:

Dienstkonten oder Notfallbenutzer erfordern häufig feste Richtlinieneinstellungen:

```
# Kennwortablauf für das Konto deaktivieren
Set-LocalUser -Name "secadmin" -PasswordNeverExpires $true

# Gesperrtes Benutzerkonto nach fehlgeschlagenen Anmeldeversuchen entsperren
Unlock-LocalUser -Name "secadmin"
```

Lokale Sicherheitsrichtlinien und Kontensperrung ohne secpol.msc (secedit):

Auf Server Core steht das grafische Snap-in für lokale Sicherheitsrichtlinien (`secpol.msc`) nicht zur Verfügung. Um Schwellenwerte für Kontensperrungen bei Brute-Force-Angriffen (beispielsweise Sperrung nach 5 Fehlversuchen für 30 Minuten) oder Kennwortkomplexitätsvorgaben zu erzwingen, nutzt du das integrierte Werkzeug `secedit.exe`:

```
# 1. Aktuelle lokale Sicherheitsrichtlinie in eine Zwischendatei exportieren
secedit /export /cfg C:\secpol.inf
```

Passe die Richtlinienwerte per PowerShell-Textverarbeitung an:

```
(Get-Content C:\secpol.inf) `
    -replace '^LockoutBadCount\s*=.*', 'LockoutBadCount = 5' `
    -replace '^ResetLockoutCount\s*=.*', 'ResetLockoutCount = 30' `
    -replace '^LockoutDuration\s*=.*', 'LockoutDuration = 30' `
    -replace '^PasswordComplexity\s*=.*', 'PasswordComplexity = 1' |
    Set-Content C:\secpol.inf -Encoding Ascii
```

Wende die modifizierte Sicherheitsdatenbank an:

```
# 2. Richtlinien in die lokale Sicherheitsdatenbank schreiben
secedit /configure /db C:\Windows\security\local.sdb /cfg C:\secpol.inf /areas
SECURITYPOLICY

# 3. Gruppenrichtlinien sofort aktualisieren und temporäre Datei entfernen
gpupdate /force
del C:\secpol.inf
```

NTP und Zeitsynchronisation konfigurieren

In virtualisierten Umgebungen kann die Systemuhrzeit durch CPU-Überbuchung oder Host-Migrationen leicht driften. Ein exakter Zeitabgleich ist jedoch zwingende Voraussetzung für Kerberos-Authentifizierung, Active Directory und die Korrelation von Protokolldateien.

Der Windows-Zeitgeberdienst wird über das Befehlszeilenwerkzeug `w32tm.exe` gesteuert.

Aktuellen Zeitstatus abfragen:

```
w32tm /query /status
```

Typische Statusausgabe:

```
Leap Indicator: 0(no warning)
Stratum: 2 (secondary reference - syncd by (S)NTP)
```

```
Precision: -23 (119.209ns per tick)
Root Delay: 0.0152588s
Root Dispersion: 0.0315482s
ReferenceId: 0xC0000201 (source IP:  192.0.2.1)
Last Successful Sync Time: 10.11.2026 10:15:32
Source: 192.0.2.1,0x8
Poll Interval: 6 (64s)
```

Externe NTP-Zeitserver festlegen:

Soll der Server seine Uhrzeit mit einem lokalen Gateway oder öffentlichen Zeitservern abgleichen, konfigurierst du die Peer-Liste wie folgt:

```
w32tm /config /manualpeerlist:"0.de.pool.ntp.org,0x8 1.de.pool.ntp.org,0x8"
/syncfromflags:manual /reliable:yes /update
```

Erzwinge anschließend den sofortigen Neuabgleich:

```
w32tm /resync /rediscover
```

Dienste und Prozesse steuern

Auf Server Core werden Hintergrunddienste ohne grafisches Verwaltungswerkzeug (`services.msc`) direkt über das Modul `Microsoft.PowerShell.Management` verwaltet.

Dienste abfragen und modifizieren:

Dienste lassen sich anhand ihres Namens, Status oder Starttyps filtern und im laufenden Betrieb neu starten oder konfigurieren:

```
# Alle automatisch startenden Dienste anzeigen, die aktuell gestoppt sind
Get-Service | Where-Object { $_.StartType -eq 'Automatic' -and $_.Status -eq 'Stopped' } |
Format-Table Name, DisplayName

# Dienst neu starten (z. B. DNS-Client oder SSH-Daemon)
Restart-Service -Name sshd -Force

# Starttyp eines Dienstes ändern (z. B. Windows Time Service)
Set-Service -Name w32time -StartupType Automatic
Start-Service -Name w32time
```

Prozesse analysieren und beenden:

Ohne den klassischen Windows Task-Manager (`taskmgr.exe`) analysierst du Ressourcenengpässe über `Get-Process` :

```
# Die 10 speicherintensivsten Prozesse auflisten
Get-Process | Sort-Object WorkingSet64 -Descending | Select-Object -First 10 -Property Id,
ProcessName, CPU, @{Name='WorkingSet_MB'; Expression={[math]::Round($_.WorkingSet64 / 1MB,
2)}} | Format-Table -AutoSize
```

Ausgabebeispiel:

```
  Id ProcessName  CPU WorkingSet_MB
  -- -----------  --- -------------
2148 svchost    12,45        184,32
1604 sshd        1,12         42,18
2892 powershell  4,88         98,64
```

```
  824 lsass       3,21         36,50
```

Sollte ein hängender Prozess beendet werden müssen, geschieht dies zielgerichtet über seine Prozess-ID (PID):

```
# Prozess sicher über PID stoppen
Stop-Process -Id 4328 -Force
```

Rollen und Features verwalten

Windows Server Core unterstützt nahezu alle klassischen Infrastrukturrollen (Active Directory Domain Services, DNS, DHCP, Webserver/IIS, Hyper-V, Datei- und Speicherdienste). Die Verwaltung erfolgt über das Modul `ServerManager`.

Verfügbare Rollen ermitteln:

```
# Alle installierten Rollen und Features auf dem Server anzeigen
Get-WindowsFeature | Where-Object Installed -eq $true | Format-Table Name, DisplayName

# Nach einem spezifischen Feature suchen (z. B. DNS-Server)
Get-WindowsFeature -Name *DNS*
```

Rollen installieren und entfernen:

Die Installation erfolgt über `Install-WindowsFeature`. Bei Rollen, die zusätzliche Management-Werkzeuge oder Subfeatures benötigen, empfiehlt sich das Flag `-IncludeManagementTools`:

```
# DNS-Server-Rolle installieren
Install-WindowsFeature -Name DNS -IncludeManagementTools

# Installation überprüfen
Get-WindowsFeature -Name DNS
```

Soll eine Rolle deinstalliert werden, nutzt du `Uninstall-WindowsFeature`:

```
Uninstall-WindowsFeature -Name DNS
```

> **Unterschied zwischen WindowsFeatures und WindowsCapabilities:** Rollen und Infrastruktur-Features wie Web-Server, Active Directory oder DNS werden über das Modul `ServerManager` mit `Install-WindowsFeature` verwaltet. Optionale Betriebssystemkomponenten und Werkzeuge (wie der OpenSSH-Server oder RSAT-Erweiterungen) werden hingegen als Capabilities über `Get-WindowsCapability` und `Add-WindowsCapability` bereitgestellt.

Domänenbeitritt per PowerShell durchführen:

Soll der Server Core einer bestehenden Active-Directory-Domäne beitreten, geschieht dies vollständig über das Cmdlet `Add-Computer`. Vorab muss sichergestellt sein, dass der Server die Domäne über seinen DNS-Resolver auflösen kann:

```
# 1. Namensauflösung zum Domain Controller testen
Resolve-DnsName -Name "corp.example.com"

# 2. Domänenbeitritt mit Administrator-Anmeldeinformationen initiieren
$DomainCred = Get-Credential
Add-Computer -DomainName "corp.example.com" -Credential $DomainCred -Restart -Force
```

Nach dem automatischen Neustart meldest du dich künftig mit Domänenkonten an (z. B. `ssh Administrator@corp.example.com@192.0.2.10` oder `ssh -l DOMAIN\Administrator 192.0.2.10`).

DNS-Server-Rolle vollständig konfigurieren:

Nach der Installation der Rolle `DNS` steht das PowerShell-Modul `DnsServer` zur Verfügung. Eine primäre Forward-Lookup-Zone für ein internes Labornetzwerk wird wie folgt eingerichtet:

```
# 1. Primäre Forward-Lookup-Zone erstellen
Add-DnsServerPrimaryZone -Name "lab.internal" -ZoneFile "lab.internal.dns"

# 2. Statische A-Records und CNAME-Einträge hinzufügen
Add-DnsServerResourceRecordA -ZoneName "lab.internal" -Name "gateway" -IPv4Address
"192.0.2.1"
Add-DnsServerResourceRecordA -ZoneName "lab.internal" -Name "pve-node01" -IPv4Address
"192.0.2.2"
Add-DnsServerResourceRecordCName -ZoneName "lab.internal" -Name "dns" -HostNameAlias
"gateway.lab.internal."

# 3. Externe DNS-Weiterleitungen (Forwarder) für Internet-Anfragen setzen
Set-DnsServerForwarder -IPAddress "1.1.1.1", "8.8.8.8"

# 4. Zonenkonfiguration und Records verifizieren
Get-DnsServerResourceRecord -ZoneName "lab.internal" | Format-Table -Property HostName,
RecordType, RecordData
```

Dateiserver und SMB-Freigaben ohne GUI bereitstellen:

Für Datei- und Speicherrollen auf Server Core installierst du das Feature `FS-FileServer` und steuerst Freigaben über das Modul `SmbShare`:

```
# Dateiserver-Funktion aktivieren
Install-WindowsFeature -Name FS-FileServer

# Freigabeverzeichnis auf Datenlaufwerk anlegen
New-Item -Path "D:\Shares\Deploy" -ItemType Directory -Force

# SMB-Freigabe mit expliziten Berechtigungen erstellen
New-SmbShare -Name "Deploy" `
  -Path "D:\Shares\Deploy" `
  -FullAccess "BUILTIN\Administrators" `
  -ChangeAccess "NT AUTHORITY\Authenticated Users" `
  -Description "Zentrales Installations- und Skript-Repository"

# Aktive SMB-Freigaben und verbundene Sessions überwachen
Get-SmbShare | Format-Table -Property Name, Path, Description
Get-SmbSession | Format-Table -Property SessionId, ClientComputerName, ClientUserName,
NumOpens
```

Dateisystem-Berechtigungen (NTFS ACLs) ohne Explorer-Sicherheitsdialog:

Auf Server Core entfällt der bekannte Reiter *„Sicherheit“* im Windows Explorer. Zugriffskontrolllisten (Access Control Lists, ACLs) werden direkt über die PowerShell-Objekte `System.Security.AccessControl` gesteuert:

```
# 1. Bestehende ACL des Zielverzeichnisses abrufen
$Path = "D:\Shares\Deploy"
$Acl = Get-Acl -Path $Path
```

```
# 2. Vererbung deaktivieren und bestehende Rechte in explizite Berechtigungen umwandeln
$Acl.SetAccessRuleProtection($true, $true)

# 3. Neue Zugriffsregel für Administratoren (Vollzugriff, vererblich auf Unterordner und
Dateien) definieren
$AdminRule = New-Object System.Security.AccessControl.FileSystemAccessRule(
    "BUILTIN\Administratoren",
    "FullControl",
    "ContainerInherit,ObjectInherit",
    "None",
    "Allow"
)
$Acl.AddAccessRule($AdminRule)

# 4. Leserechte für Authentifizierte Benutzer hinzufügen
$UserRule = New-Object System.Security.AccessControl.FileSystemAccessRule(
    "NT AUTHORITY\Authentifizierte Benutzer",
    "ReadAndExecute",
    "ContainerInherit,ObjectInherit",
    "None",
    "Allow"
)
$Acl.AddAccessRule($UserRule)

# 5. Modifizierte Zugriffssteuerungsliste anwenden
Set-Acl -Path $Path -AclObject $Acl

# 6. Berechtigungen verifizieren
(Get-Acl -Path $Path).Access | Format-Table -Property IdentityReference, FileSystemRights,
AccessControlType, IsInherited
```

Zertifikatsverwaltung auf Server Core ohne certmgr.msc

Auf einem Headless-System steht kein lokales Zertifikats-Snap-in (`certmgr.msc` oder `certlm.msc`) bereit. Die Verwaltung von SSL/TLS-Zertifikaten (etwa für interne Webserver, LDAP-Sicherheit oder Skript-Signaturen) erfolgt über das PowerShell-Laufwerk `Cert:` und das Befehlszeilenwerkzeug `certutil.exe`.

Lokale Zertifikatsspeicher inspizieren:

Der Windows-Zertifikatsspeicher ist in der PowerShell wie ein Dateisystem über das PSDrive `Cert:` eingebunden:

```
# Eigene Zertifikate des lokalen Computers auflisten
Get-ChildItem -Path "Cert:\LocalMachine\My" | Format-Table -Property Subject, Thumbprint,
NotAfter

# Stammzertifikate vertrauenswürdiger Zertifizierungsstellen anzeigen
Get-ChildItem -Path "Cert:\LocalMachine\Root" | Select-Object -First 10 -Property Subject,
Thumbprint
```

Selbstsigniertes Zertifikat für Testzwecke generieren:

Für die Verschlüsselung interner Testdienste lässt sich ein Zertifikat mit 2048-Bit-RSA-Schlüssel und 2 Jahren Gültigkeit erstellen:

```
$Cert = New-SelfSignedCertificate `
  -DnsName "win-core01.lab.internal" `
  -CertStoreLocation "Cert:\LocalMachine\My" `
  -NotAfter (Get-Date).AddYears(2) `
  -KeyLength 2048 `
  -FriendlyName "WIN-CORE01 Testzertifikat"

# Erstellten Fingerabdruck (Thumbprint) ausgeben
$Cert.Thumbprint
```

Zertifikat einer internen Unternehmens-CA importieren:

Wird im Netzwerk eine eigene Active-Directory-Zertifizierungsstelle betrieben, importierst du deren Root-Zertifikat mit administrativen Rechten in den vertrauenswürdigen Speicher:

```
Import-Certificate -FilePath "C:\Certs\InternalRootCA.crt" -CertStoreLocation
"Cert:\LocalMachine\Root"
```

Zertifikatsabläufe proaktiv prüfen:

Um Dienstunterbrechungen durch abgelaufene Zertifikate zu verhindern, prüfst du regelmäßig Zertifikate, deren Gültigkeit in weniger als 30 Tagen endet:

```
Get-ChildItem -Path "Cert:\LocalMachine\My" |
    Where-Object { $_.NotAfter -lt (Get-Date).AddDays(30) } |
    Format-Table -Property Subject, NotAfter, Thumbprint
```

Ereignisprotokolle analysieren mit Get-WinEvent

Auf einem Server ohne grafische Oberfläche steht die Microsoft Management Console (`eventvwr.msc`) lokal nicht zur Verfügung. Das Cmdlet `Get-WinEvent` ist das primäre Werkzeug zur forensischen Analyse und Fehlerdiagnose.

> **Performance-Regel für Ereignisprotokolle:** Vermeide es, Ereignisse unbedarft abzurufen und erst anschließend über die Pipeline mit `Where-Object` zu filtern (`Get-WinEvent -LogName System | Where-Object ...`). Auf Produktivsystemen mit Millionen Einträgen führt dies zu massiver CPU- und I/O-Last. Verwende stets den Parameter `-FilterHashtable` , damit die Filterung bereits auf Kernel-Ebene durch die Windows-Event-Engine erfolgt.

Praktisches Beispiel: Fehler und Warnungen im Systemprotokoll filtern

Um alle Fehler (`Level = 2`) und Warnungen (`Level = 3`) der vergangenen 24 Stunden aus dem System-Log zu extrahieren, definierst du eine strukturierte Filter-Hashtabelle:

```
# Filter für die letzten 24 Stunden definieren
$StartTime = (Get-Date).AddDays(-1)
$Filter = @{
    LogName   = System
    Level     = 2, 3       # 2 = Error, 3 = Warning
    StartTime = $StartTime
}

# Ereignisse abrufen und formatiert ausgeben
Get-WinEvent -FilterHashtable $Filter -MaxEvents 50 | Format-Table TimeCreated, Id,
ProviderName, Message -AutoSize
```

□□ Praktisches Beispiel: Fehlgeschlagene Anmeldungen im Sicherheitslog finden

Zur Sicherheitsüberwachung lassen sich fehlgeschlagene Anmeldeversuche (Event-ID 4625) gezielt isolieren:

```
$SecurityFilter = @{
    LogName = Security
    Id      = 4625
}

# Die jüngsten 10 fehlgeschlagenen Anmeldeversuche prüfen
Get-WinEvent -FilterHashtable $SecurityFilter -MaxEvents 10 | Select-Object TimeCreated,
Id, Message | Format-List
```

Webserver (IIS) auf Server Core bereitstellen

Die Bereitstellung des Microsoft Internet Information Services (IIS) auf Windows Server Core ist ein Paradebeispiel für einen ressourcenschonenden Produktionsdienst: Ohne grafische Verwaltungsoberfläche verbraucht der Webserver im Leerlauf weniger als 80 MiB RAM.

1. IIS-Webserver-Rolle installieren:

```
# Web-Server (IIS) und grundlegende Verwaltungsmodule installieren
Install-WindowsFeature -Name Web-Server, Web-Mgmt-Tools

# Dienst- und Feature-Status prüfen
Get-WindowsFeature -Name Web-Server
Get-Service -Name W3SVC
```

2. Dedizierten Application Pool und Website anlegen:

Statt der Standard-Website (`Default Web Site`) empfiehlt sich ein isolierter Anwendungspool mit eigenem Dokumentenverzeichnis:

```
# WebAdministration-Modul importieren
Import-Module WebAdministration

# 1. Dokumentenverzeichnis anlegen
$WebRoot = "C:\inetpub\docs"
New-Item -Path $WebRoot -ItemType Directory -Force | Out-Null
Set-Content -Path "$WebRoot\index.html" -Value "<h1>AdminDocs Windows Server 2025
Core</h1><p>Status: Online</p>"

# 2. Neuen isolierten Application Pool erstellen
New-WebAppPool -Name "DocsPool"
Set-ItemProperty -Path "IIS:\AppPools\DocsPool" -Name "managedRuntimeVersion" -Value ""
```

```
# 3. Neue Website auf Port 80 binden
New-WebSite -Name "DocsApp" -Port 80 -PhysicalPath $WebRoot -ApplicationPool "DocsPool" -
Force
```

3. Firewall-Freigabe für HTTP und HTTPS aktivieren:

Damit externe Clients den Webserver erreichen können, erstellst du explizite Firewall-Regeln:

```
# Eingehenden HTTP-Verkehr (Port 80) erlauben
New-NetFirewallRule -Name "IIS-HTTP-In" `
  -DisplayName "Webserver HTTP (Port 80)" `
  -Direction Inbound `
  -Protocol TCP `
  -LocalPort 80 `
  -Action Allow

# Eingehenden HTTPS-Verkehr (Port 443) erlauben
New-NetFirewallRule -Name "IIS-HTTPS-In" `
  -DisplayName "Webserver HTTPS (Port 443)" `
  -Direction Inbound `
  -Protocol TCP `
  -LocalPort 443 `
  -Action Allow
```

Headless Netzwerk-Traces und Paket-Captures ohne Wireshark

Auf einem Headless-System ohne grafische Oberfläche steht Wireshark lokal nicht zur Verfügung. Um sporadische Netzwerkprobleme, blockierte Kerberos-Tickets oder abbrechende SMB-Sitzungen dennoch bis auf Paketebene zu analysieren, nutzt Windows Server das integrierte Befehlszeilenwerkzeug `netsh trace`.

Das Werkzeug schreibt Rohpakete und Windows-Kernel-Netzwerkereignisse direkt in eine ETL-Protokolldatei (*Event Trace Log*):

□□ Praktisches Beispiel: Ringpuffer-Paketaufzeichnung starten

```
# Ringpuffer-Mitschnitt starten (begrenzt auf 256 MB, überschreibt älteste Pakete)
netsh trace start capture=yes tracefile=C:\trace.etl maxsize=256 filemode=circular
overwrite=yes
```

Sobald die Erfassung läuft, reproduzierst du den Fehler auf der Netzwerkverbindung. Beende den Mitschnitt anschließend:

```
# Paketmitschnitt stoppen
netsh trace stop
```

Die erzeugte Datei `C:\trace.etl` enthält alle aufgezeichneten Ethernet-Frames.

Um die Pakete auf deiner Linux-Workstation mit Wireshark zu analysieren, kopierst du die Datei per SCP auf deinen Client:

```
scp -i ~/.ssh/id_ed25519 Administrator@192.0.2.10:C:/trace.etl ~/trace.etl
```

Unter Linux lassen sich Microsoft-ETL-Dateien mit dem Open-Source-Werkzeug `etl2pcapng` in Standard-PCAPNG-Dateien konvertieren und anschließend direkt in Wireshark oder `tshark` öffnen:

```
etl2pcapng ~/trace.etl ~/trace.pcapng
wireshark ~/trace.pcapng
```

Mit dieser Methode diagnostizierst du selbst tiefste Netzwerk- und Authentifizierungsprobleme ohne GUI-Abhängigkeiten.

Mit dieser objektbasierten Arbeitsweise führst du auf Server Core sämtliche Administrations- und Analyseaufgaben mindestens genauso schnell und reproduzierbar durch wie auf Systemen mit vollständiger Desktopoberfläche.

Updates, Wartung und grundlegende Absicherung

Nach der Ersteinrichtung und Bereitstellung verlagert sich der Fokus auf den stabilen Dauerbetrieb. Auf einem Server ohne grafische Oberfläche erfordern Patch-Management, Virenschutz, Netzwerkfilterung, Speicheroptimierung und konsistente Datensicherung automatisierbare Abläufe.

Patch-Management und Windows Update

Windows Server Core besitzt kein grafisches Windows-Update-Fenster. Für das Einspielen von Sicherheits- und Qualitäts-Updates stehen dir zwei praxiserprobte Wege zur Verfügung.

Interaktive Aktualisierung über SConfig:

Für gelegentliche manuelle Wartungen auf der Textkonsole ist Option `6) Updates installieren` in SConfig der schnellste Weg:

```
sconfig
```

Nach Auswahl von Option 6 wählst du, ob nur empfohlene oder alle Updates gesucht werden sollen. SConfig listet die gefundenen KB-Pakete nummeriert auf und ermöglicht die gezielte oder vollständige Installation samt automatischem Neustart.

Skriptbasierte Aktualisierung mit PSWindowsUpdate:

In standardisierten Umgebungen und automatisierten Wartungsfenstern empfiehlt sich das PowerShell-Modul `PSWindowsUpdate`. Da PowerShellGet auf einem frischen Server Core zunächst den Paket-Provider benötigt, initialisierst du die Umgebung wie folgt:

```
# NuGet-Provider und PSWindowsUpdate-Modul installieren
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force
Install-Module -Name PSWindowsUpdate -Force
```

Nach dem Import des Moduls steuerst du den gesamten Update-Prozess skriptbasiert:

```
# Verfügbare Updates auflisten, ohne sie zu installieren
Get-WindowsUpdate

# Alle ausstehenden Updates herunterladen, installieren und bei Bedarf neu starten
Install-WindowsUpdate -MicrosoftUpdate -AcceptAll -AutoReboot
```

> **Wartungsfenster planen:** Der Parameter `-AutoReboot` startet den Server ohne Rückfrage neu, sobald ein Patch dies erfordert. Führe diesen Befehl auf Produktivsystemen ausschließlich innerhalb freigegebener Wartungszeitfenster aus.

Installierte Updates und Patch-Historie verifizieren:

Um nach einer Wartung zu überprüfen, welche Qualitätsupdates installiert wurden, fragst du den Hotfix-Bestand ab:

```
Get-HotFix | Sort-Object InstalledOn -Descending | Format-Table -Property HotFixID,
Description, InstalledBy, InstalledOn
```

Die tatsächlich installierten Updates und deren Zeitstempel hängen vom jeweiligen Veröffentlichungs- und Patch-Stand des Systems ab. Das folgende schematische Beispiel verdeutlicht das tabellarische Ausgabeformat:

```
HotFixID   Description     InstalledBy          InstalledOn
--------   -----------     -----------          -----------
KBxxxxxxx  Security Update NT AUTHORITY\SYSTEM  ...
KBxxxxxxx  Update          NT AUTHORITY\SYSTEM  ...
```

Microsoft Defender Antivirus auf Server Core

Auf Windows Server 2025 Core ist Microsoft Defender Antivirus standardmäßig aktiv und vollständig integriert. Die Überwachung und Steuerung erfolgt über das Modul `Defender`.

Schutzstatus prüfen:

Mit `Get-MpComputerStatus` fragst du den Zustand der Antiviren-Engine, den Status des Echtzeitschutzes und das Alter der Signaturdefinitionen ab:

```
Get-MpComputerStatus | Select-Object -Property AntivirusEnabled,
RealTimeProtectionEnabled, AntivirusSignatureVersion, AntivirusSignatureLastUpdated
```

Signaturen aktualisieren und Scans ausführen:

Signaturen lassen sich jederzeit manuell aktualisieren, gefolgt von einer schnellen oder vollständigen Systemüberprüfung:

```
# Signaturdefinitionen sofort aktualisieren
Update-MpSignature

# Schnellen Systemscan im Hintergrund starten
Start-MpScan -ScanType QuickScan
```

Ausschlüsse für Datenbank- und Speicherpfade definieren:

Auf Servern beeinträchtigt das Echtzeit-Scannen von Transaktionsprotokollen oder virtuellen Festplattendateien die I/O-Leistung massiv. Ausschlüsse werden per PowerShell definiert:

```
# Pfadausschluss für Datenbankverzeichnisse hinzufügen
Add-MpPreference -ExclusionPath "C:\Data\Database"

# Ausschluss für spezifische Dateiendungen (z. B. virtuelle Disks)
Add-MpPreference -ExclusionExtension ".vhdx"

# Gesetzte Ausschlüsse verifizieren
Get-MpPreference | Select-Object -ExpandProperty ExclusionPath
```

Erkannte Bedrohungen auslesen:

Wurden verdächtige Dateien oder Schadsoftware blockiert, gibt `Get-MpThreatDetection` Auskunft über den Fundort und die ergriffenen Quarantänemaßnahmen:

```
Get-MpThreatDetection | Select-Object -Property InitialDetectionTime, ThreatName,
Resources
```

Windows Firewall härten und Zugriffe beschränken

Die integrierte Windows Defender Firewall ist auf Server Core standardmäßig für alle Profile (*Domain*, *Private*, *Public*) aktiv und blockiert unaufgeforderte eingehende Verbindungen.

Profilstatus überprüfen:

```
Get-NetFirewallProfile | Format-Table -Property Name, Enabled, DefaultInboundAction,
DefaultOutboundAction
```

In einer gehärteten Serverumgebung sollte `DefaultInboundAction` ausnahmslos auf `Block` stehen.

SSH-Port auf Administrationsnetze beschränken:

Standardmäßig erlaubt die bei der OpenSSH-Einrichtung erstellte Firewall-Regel Verbindungen von beliebigen IP-Adressen. In Produktivumgebungen solltest du den Zugriff zwingend auf dein dediziertes Administrations- oder Management-Subnetz einschränken:

```
# SSH-Zugriff strikt auf das Subnetz 192.0.2.0/24 begrenzen
Set-NetFirewallRule -Name 'OpenSSH-Server-In-TCP' -RemoteAddress '192.0.2.0/24'

# Änderung verifizieren
Get-NetFirewallRule -Name 'OpenSSH-Server-In-TCP' | Get-NetFirewallAddressFilter | Format-
List RemoteAddress
```

Protokollierung abgewiesener Pakete aktivieren:

Zur Fehleranalyse bei geblockten Netzwerkverbindungen aktivierst du das Firewall-Logging:

```
# Verworfene Pakete im Domain- und Private-Profil protokollieren
Set-NetFirewallProfile -Profile Domain, Private -LogBlocked True -LogFileName
"$env:SystemRoot\System32\LogFiles\Firewall\pfirewall.log"
```

Systemhärtung nach CIS-Sicherheitsstandards

Für Server im Rechenzentrumsbetrieb empfiehlt das Center for Internet Security (CIS) die Deaktivierung veralteter Namensauflösungs- und Authentifizierungsprotokolle, die anfällig für Man-in-the-Middle-Angriffe (wie Responder oder NTLM-Relaying) sind:

1. LLMNR (Link-Local Multicast Name Resolution) deaktivieren:

LLMNR sendet Multicast-Anfragen im lokalen Subnetz, wenn DNS-Anfragen fehlschlagen – ein beliebtes Einfallstor für Angreifer. Die Deaktivierung erfolgt über die Registrierung:

```
$DnsPolicyPath = "HKLM:\SOFTWARE\Policies\Microsoft\Windows NT\DNSClient"
if (-not (Test-Path $DnsPolicyPath)) { New-Item -Path $DnsPolicyPath -Force | Out-Null }
Set-ItemProperty -Path $DnsPolicyPath -Name "EnableMulticast" -Value 0 -Type DWord
```

2. NetBIOS über TCP/IP auf allen Netzwerkadaptern abschalten:

NetBIOS über Port 137/138 ist veraltet und sollte auf reinen DNS-gestützten Server-Core-Systemen deaktiviert werden:

```
$Adapters = Get-CimInstance -ClassName Win32_NetworkAdapterConfiguration | Where-Object
IPEnabled
foreach ($adapter in $Adapters) {
    # 2 = NetBIOS über TCP/IP deaktivieren
    Invoke-CimMethod -InputObject $adapter -MethodName SetTcpipNetbios -Arguments
@{TcpipNetbiosOptions = 2} | Out-Null
}
```

3. NTLMv1 verbieten und NTLMv2 erzwingen:

Verhindere, dass Clients oder Dienste unsichere NTLMv1-Hashes aushandeln:

```
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Lsa" `
  -Name "LmCompatibilityLevel" -Value 5 -Type DWord
```

4. WDigest-Klartext-Zwischenspeicherung im LSASS-Speicher unterbinden:

Verhindert, dass Windows Benutzerkennwörter im Klartext im Speicher des Local Security Authority Subsystem Service (`lsass.exe`) vorhält:

```
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest"
`
  -Name "UseLogonCredential" -Value 0 -Type DWord
```

QEMU Guest Agent und VSS-Snapshots für Proxmox VE

Beim Betrieb von Windows Server Core auf Proxmox VE 9 ist die Installation des **QEMU Guest Agent** (`qemu-ga`) ein unverzichtbarer Schritt für zuverlässige Datensicherungen und Host-Interaktionen.

Bedeutung für anwendungskonsistente Backups:

Wird eine virtuelle Maschine auf Proxmox VE (beispielsweise über einen Proxmox Backup Server) gesichert, erzeugt der Hypervisor standardmäßig einen Snapshot auf Blockebene. Ohne Gast-Agent ist dieses Backup lediglich *crash-konsistent* – Daten im Dateisystem-Cache oder in Transaktionsprotokollen können unvollständig sein.

Der QEMU Guest Agent bindet sich über einen internen VSS-Provider (*Volume Shadow Copy Service*) in das Windows-Betriebssystem ein. Während der Snapshot-Erstellung friert der Agent das Dateisystem und alle VSS-kompatiblen Dienste (wie Active Directory, Microsoft SQL Server oder Dateidienste) für wenige Millisekunden kontrolliert ein (*fsfreeze*). Das Ergebnis ist ein garantiert *anwendungskonsistentes* Backup.

□□ Praktisches Beispiel: QEMU Guest Agent auf Server Core installieren

Binde die `virtio-win.iso` im Proxmox-Webinterface oder per CLI als virtuelles CD-ROM-Laufwerk an (im Beispiel Laufwerksbuchstabe `D:`). Führe in deiner PowerShell-Sitzung folgende Schritte aus:

```
# MSI-Paket des Guest Agent geräuschlos installieren
Start-Process -FilePath "msiexec.exe" -ArgumentList "/i D:\guest-agent\qemu-ga-x86_64.msi
/quiet /norestart" -Wait

# Dienst-Starttyp auf Automatisch setzen und Dienst starten
Set-Service -Name "QEMU-GA" -StartupType Automatic
Start-Service -Name "QEMU-GA"

# Status des Agenten prüfen
Get-Service -Name "QEMU-GA" | Format-Table Name, Status, StartType
```

Damit der Proxmox-Host den installierten Agenten nutzt, aktivierst du die Option in der Konfiguration der VM auf dem Proxmox-Host:

```
qm set 104 -agent 1
```

Nach wenigen Sekunden meldet Proxmox VE die genauen IP-Adressen und den Betriebssystemstatus des Core-Servers direkt in der Web-Oberfläche und im Backup-Task.

VSS-Writer-Zustand nach Backups prüfen:

Sollten Hypervisor-Backups fehlschlagen, liegt die Ursache häufig in einem hängenden VSS-Writer. Den Zustand aller registrierten Komponenten prüfst du mit `vssadmin`:

```
vssadmin list writers
```

Alle Writer (wie `System Writer`, `ASR Writer`, `WMI Writer`) müssen sich im Zustand `[1] Stabil` (`State: [1] Stable`) ohne Fehler befinden:

```
Writer-Name: "System Writer"
   Writer-ID: {e81062d3-1f76-4d37-a4b7-8331db428a5e}
   Writer-Instanz-ID: {4a5b6c7d-8e9f-0a1b-2c3d-4e5f6a7b8c9d}
   Status: [1] Stabil
   Letzter Fehler: Kein Fehler
```

Thin-Provisioning und Trim/Discard optimieren

Wird die virtuelle Maschine auf Proxmox VE auf ZFS, Ceph oder LVM-Thin betrieben, wächst die zugewiesene Image-Datei bei Schreibvorgängen dynamisch an. Werden jedoch Dateien innerhalb von Windows gelöscht, erfährt der Hypervisor davon standardmäßig nichts – der Speicherplatz bleibt belegt.

Voraussetzung für die automatische Speicherfreigabe ist, dass am Proxmox-SCSI-Controller die Option `discard=on` aktiv ist.

Mit dem Cmdlet `Optimize-Volume` sendest du manuelle TRIM/UNMAP-Befehle an den Hypervisor, um gelöschte Blöcke auf Host-Ebene wieder freizugeben:

```
# Dateisystem auf Laufwerk C: bereinigen und ungenutzte Blöcke an den Hypervisor freigeben
Optimize-Volume -DriveLetter C -Defrag -Verbose
```

Die Ausgabe bestätigt die Speicherfreigabe:

```
AUSFÜHRLICH: Ein erneutes Trimmen für "Windows (C:)" wird gestartet...
AUSFÜHRLICH: Erneutes Trimmen: 100% abgeschlossen.
AUSFÜHRLICH: Nachbereitung für "Windows (C:)" wird ausgeführt...
```

Geplante Aufgaben und automatisierte Wartung ohne taskschd.msc

Auf einem Server ohne grafische Oberfläche steht die Aufgabenplanung (`taskschd.msc`) nicht lokal bereit. Regelmäßige Wartungsarbeiten wie Festplatten-Trim, Defender-Aktualisierungen und Protokollbereinigung werden über das PowerShell-Modul `ScheduledTasks` automatisiert.

□□ Praktisches Beispiel: Vollständiges Wartungsskript Maintain-ServerCore.ps1

Erstelle ein zentrales Wartungsskript unter `C:\Scripts\Maintain-ServerCore.ps1`:

```
# Maintain-ServerCore.ps1
# Zyklische Wartungsroutine für Windows Server 2025 Core

$LogPath = "C:\Scripts\Logs"
if (-not (Test-Path $LogPath)) { New-Item -Path $LogPath -ItemType Directory -Force | Out-
Null }
$LogFile = "$LogPath\Maintenance_$(Get-Date -Format 'yyyy-MM-dd').log"

function Write-Log {
    param ([string]$Message)
    $Timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    "[$Timestamp] $Message" | Tee-Object -FilePath $LogFile -Append
}

Write-Log "=== Starte Server Core Wartungsroutine ==="

# 1. Antiviren-Signaturen aktualisieren und Schnellscan starten
Write-Log "[1/4] Aktualisiere Microsoft Defender Signaturen..."
Update-MpSignature
```

```
Write-Log "[1/4] Starte QuickScan..."
Start-MpScan -ScanType QuickScan

# 2. Festplatten-Optimierung und Hypervisor-Trim durchführen
Write-Log "[2/4] Führe TRIM/UNMAP auf Systemlaufwerk C: aus..."
Optimize-Volume -DriveLetter C -Defrag -Verbose 4>&1 | ForEach-Object { Write-Log "  $_" }

# 3. Temporäre Systemdateien bereinigen (Dateien älter als 7 Tage)
Write-Log "[3/4] Bereinige veraltete temporäre Dateien..."
$TempDirs = @("$env:SystemRoot\Temp", "$env:TEMP")
$CutOffDate = (Get-Date).AddDays(-7)

foreach ($dir in $TempDirs) {
    if (Test-Path $dir) {
        Get-ChildItem -Path $dir -Recurse -File -ErrorAction SilentlyContinue |
            Where-Object { $_.LastWriteTime -lt $CutOffDate } |
            Remove-Item -Force -ErrorAction SilentlyContinue
    }
}

# 4. Status der VSS-Writer validieren
Write-Log "[4/4] Überprüfe Volume Shadow Copy Writer..."
$FailedWriters = vssadmin list writers | Select-String -Pattern "Status: \[[^1]\]"
if ($FailedWriters) {
    Write-Log "WARNUNG: Einer oder mehrere VSS-Writer befinden sich in einem
Fehlerzustand!"
} else {
    Write-Log "Alle VSS-Writer befinden sich im Zustand [1] Stabil."
}

Write-Log "=== Wartungsroutine erfolgreich abgeschlossen ==="
```

Wartungsskript als wöchentliche Aufgabe registrieren:

Registriere das Skript so, dass es jeden Sonntag um 03:00 Uhr nachts mit `SYSTEM`-Privilegien ausgeführt wird:

```
# Aktion, Trigger und Prinzipal definieren
$Action = New-ScheduledTaskAction -Execute "powershell.exe" `
  -Argument "-NoProfile -ExecutionPolicy Bypass -File C:\Scripts\Maintain-ServerCore.ps1"

$Trigger = New-ScheduledTaskTrigger -Weekly -DaysOfWeek Sunday -At 03:00

$Principal = New-ScheduledTaskPrincipal -UserId "NT AUTHORITY\SYSTEM" -LogonType
ServiceAccount -RunLevel Highest

# Geplante Aufgabe registrieren
Register-ScheduledTask -TaskName "ServerCore-Maintenance" `
  -Action $Action `
  -Trigger $Trigger `
  -Principal $Principal `
  -Description "Wöchentliche Wartungsroutine (Defender, TRIM, VSS, Temp)"

# Registrierte Aufgabe überprüfen
Get-ScheduledTask -TaskName "ServerCore-Maintenance"
```

Performance-Monitoring und Kapazitätsplanung mit Get-Counter

Auf Server Core fehlen der grafische Performance-Monitor (`perfmon.exe`) und der Ressourcenmonitor (`resmon.exe`). Das offizielle PowerShell-Cmdlet zur Abfrage von Leistungsindikatoren ist `Get-Counter` .

Live-Werte für CPU, RAM und Festplatten-Latenz abrufen:

```
# Wichtige Systemindikatoren in einer Abfrage bündeln
$Counters = @(
    "\Processor(_Total)\% Processor Time",
    "\Memory\Available MBytes",
    "\PhysicalDisk(_Total)\% Disk Time",
    "\PhysicalDisk(_Total)\Current Disk Queue Length",
```

```
    "\Network Interface(*)\Bytes Total/sec"
)

Get-Counter -Counter $Counters | Select-Object -ExpandProperty CounterSamples |
    Format-Table -Property Path, CookedValue -AutoSize
```

Ausgabebeispiel:

```
Path                                                    CookedValue
----                                                    -----------
\\win-core01\processor(_total)\% processor time                3,12
\\win-core01\memory\available mbytes                       6842,00
\\win-core01\physicaldisk(_total)\% disk time                 0,45
\\win-core01\physicaldisk(_total)\current disk queue length   0,00
\\win-core01\network interface(ethernet0)\bytes total/sec    1248,50
```

Kontinuierliches Performance-Logging in eine CSV-Datei:

Für Latenzanalysen oder Lasttests kannst du Indikatoren über einen definierten Zeitraum aufzeichnen:

```
# 60 Messungen im Abstand von 5 Sekunden aufzeichnen und als CSV speichern
Get-Counter -Counter "\Processor(_Total)\% Processor Time", "\Memory\Available MBytes" `
  -SampleInterval 5 -MaxSamples 60 |
  Export-Counter -Path "C:\Scripts\Logs\Performance_$(Get-Date -Format 'yyyy-MM-
dd_HHmm').csv" -FileFormat CSV
```

Zentrale WSUS- und Gruppenrichtlinien ohne gpedit.msc steuern

Auf einem Standalone-Core-Server oder vor dem Active-Directory-Beitritt steht der lokale Gruppenrichtlinien-Editor (`gpedit.msc`) nicht zur Verfügung. Richtlinien für Windows Update werden direkt über die Windows-Registrierung im Pfad `HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate` verwaltet:

```
$WsusPath = "HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate"
$AuPath   = "$WsusPath\AU"

# Registrierungsschlüssel anlegen, falls noch nicht vorhanden
if (-not (Test-Path $WsusPath)) { New-Item -Path $WsusPath -Force | Out-Null }
if (-not (Test-Path $AuPath))   { New-Item -Path $AuPath -Force | Out-Null }

# Internen WSUS-Server definieren
Set-ItemProperty -Path $WsusPath -Name "WUServer" -Value
"http://wsus.corp.example.com:8530"
Set-ItemProperty -Path $WsusPath -Name "WUStatusServer" -Value
"http://wsus.corp.example.com:8530"

# Automatische Updates aktivieren: 4 = Automatisch herunterladen und laut Zeitplan
installieren
Set-ItemProperty -Path $AuPath -Name "AUOptions" -Value 4 -Type DWord

# Installation jeden Samstag um 04:00 Uhr
Set-ItemProperty -Path $AuPath -Name "ScheduledInstallDay" -Value 7 -Type DWord
Set-ItemProperty -Path $AuPath -Name "ScheduledInstallTime" -Value 4 -Type DWord

# Windows Update Agent neu initialisieren
Restart-Service -Name wuauserv
```

WinSxS-Speicherbereinigung mit DISM (Component Store Cleanup)

Nach mehreren monatlichen Update-Zyklen wächst der Windows-Komponentenspeicher (`C:\Windows\WinSxS`) kontinuierlich an, da Windows ältere Versionen ersetzter Systemkomponenten für eventuelle Rollbacks vorhält. Auf Server Core lässt sich dieser Speicherplatz mit dem *Deployment Image Servicing and Management* Werkzeug (`dism.exe`)

freigeben:

1. Bereinigungsbedarf des Komponentenspeichers analysieren:

```
dism.exe /Online /Cleanup-Image /AnalyzeComponentStore
```

Die Ausgabe zeigt die reale Größe des Ordners `WinSxS`, den Umfang geteilter Systemkomponenten und gibt eine Empfehlung ab, ob eine Bereinigung ratsam ist (*„Component Store Cleanup Recommended : Yes“*).

2. Veraltete Update-Versionen bereinigen:

```
dism.exe /Online /Cleanup-Image /StartComponentCleanup
```

3. Maximale Speicherfreigabe mit Basiszustands-Reset (/ResetBase):

Soll der maximale Speicherplatz auf der virtuellen Festplatte freigegeben werden, nutzt du den Schalter `/ResetBase`. Dieser entfernt alle vorherigen Update-Versionen dauerhaft und spart typischerweise 3 bis 6 GiB Speicherplatz:

```
dism.exe /Online /Cleanup-Image /StartComponentCleanup /ResetBase
```

> ⚠ **Auswirkung von /ResetBase beachten:** Nach der Ausführung von `/ResetBase` können die aktuell installierten Qualitäts- und Sicherheitsupdates nicht mehr einzeln deinstalliert werden. Führe diesen Schritt nur auf Systemen durch, die nach einem Patchday bereits mehrere Tage stabil und fehlerfrei im Produktionsbetrieb gelaufen sind.

Systemdateiprüfung und Image-Reparatur mit SFC und DISM

Treten nach Kernel-Updates, unerwarteten Stromausfällen des Hypervisors oder Festplatten-E/A-Fehlern unerklärliche Dienstabstürze auf, verifizierst du die Unversehrtheit des Windows-Dateisystems mit dem System File Checker (`sfc.exe`) und DISM.

1. Integrität geschützter Systemdateien prüfen:

```
sfc.exe /scannow
```

Findet SFC beschädigte Bibliotheken, ersetzt das Werkzeug diese automatisch aus dem lokalen Komponentenspeicher (`WinSxS`).

2. Beschädigungen im Windows-Image analysieren und reparieren:

Sollte der Komponentenspeicher selbst beschädigt sein, repariert DISM das Basis-Image online über Microsoft Windows Update:

```
# Status und Reparaturfähigkeit des Images prüfen
dism.exe /Online /Cleanup-Image /ScanHealth
dism.exe /Online /Cleanup-Image /CheckHealth

# Automatische Reparatur über Windows Update initiieren
```

```
dism.exe /Online /Cleanup-Image /RestoreHealth
```

3. Offline-Reparatur über die Windows Server 2025 Installations-ISO:

Befindet sich der Server in einem isolierten Netzwerk ohne Internetzugriff, nutzt du die gemountete Installations-ISO (im Beispiel Laufwerk `D:`) als verlässliche Referenzquelle für intakte Systemdateien:

```
dism.exe /Online /Cleanup-Image /RestoreHealth /Source:wim:D:\sources\install.wim:1
/LimitAccess
```

Der Schalter `/LimitAccess` verhindert, dass DISM vergeblich versucht, Microsoft Update über das Internet zu kontaktieren.

Speicherplatz-Audits der Ordnergrößen in PowerShell:

Auf Server Core fehlen grafische Festplattenanalysatoren wie TreeSize. Um schnell zu ermitteln, welche Verzeichnisse auf dem Systemlaufwerk den meisten Speicherplatz belegen:

```
Get-ChildItem -Path "C:\" -Directory -Force | ForEach-Object {
    $DirName = $_.Name
    $Size = (Get-ChildItem -Path $_.FullName -Recurse -File -Force -ErrorAction
SilentlyContinue |
             Measure-Object -Property Length -Sum).Sum
    [PSCustomObject]@{
        Verzeichnis = $DirName
        Groesse_GB  = [math]::Round($Size / 1GB, 2)
    }
} | Sort-Object Groesse_GB -Descending | Format-Table -AutoSize
```

Geordnete Wartungsneustarts

Nach Patch-Installationen oder Treiber-Aktualisierungen muss der Server kontrolliert neu gestartet werden. Verwende dafür entweder das PowerShell-Cmdlet `Restart-Computer` oder das klassische `shutdown.exe`:

```
# Sofortiger kontrollierter Neustart über PowerShell
Restart-Computer -Force

# Oder zeitverzögerter Neustart mit Warnung für aktive Benutzersitzungen
shutdown.exe /r /t 60 /c "Geplanter Wartungsneustart nach Patch-Installation"
```

Soll der Server stattdessen geordnet heruntergefahren und ausgeschaltet werden, nutzt du:

```
Stop-Computer -Force
```

Befehlsreferenz für die tägliche Administration

Für den schnellen Zugriff im Betriebsalltag fasst die folgende Referenz die wichtigsten Befehle für Windows Server Core zusammen. Alle Befehle lassen sich direkt über eine SSH-Sitzung oder die lokale PowerShell-Konsole ausführen:

SConfig aufrufen Startet das menügeführte Konsolen-Konfigurationsmenü für Rechnername, Domäne, Netzwerk und Updates

$ `sconfig`

SConfig Autostart deaktivieren Verhindert das automatische Starten von SConfig bei der Administrator-Anmeldung

```
$ Set-SConfig -AutoLaunch $false
```

Netzwerkadapter und IP-Adressen anzeigen Listet alle physischen und virtuellen Netzwerkadapter samt zugewiesenen IP-Adressen auf

```
$ Get-NetIPConfiguration
```

Statische IPv4-Adresse konfigurieren Weist dem primären Netzwerkadapter eine feste IP-Adresse, Präfixlänge und Standard-Gateway zu

```
$ New-NetIPAddress -InterfaceAlias "Ethernet0" -IPAddress 192.0.2.10 -PrefixLength 24 -DefaultGateway 192.0.2.1
```

DNS-Server konfigurieren Definiert primäre und sekundäre DNS-Server für die Namensauflösung

```
$ Set-DnsClientServerAddress -InterfaceAlias "Ethernet0" -ServerAddresses ("192.0.2.1", "1.1.1.1")
```

Netzwerk- und Portverbindung testen Prüft die TCP-Erreichbarkeit entfernter Dienste (z. B. DNS oder Gateways)

```
$ Test-NetConnection -ComputerName 192.0.2.1 -Port 53
```

OpenSSH-Server Autostart und Dienststart Aktiviert den automatischen Start des SSH-Daemons und startet den Hintergrunddienst

```
$ Set-Service -Name sshd -StartupType Automatic; Start-Service -Name sshd
```

PowerShell als SSH-Standardshell konfigurieren Setzt die PowerShell 5.1 anstelle der cmd.exe als Login-Shell für eingehende SSH-Sitzungen

```
$ New-ItemProperty -Path "HKLM:\SOFTWARE\OpenSSH" -Name DefaultShell -Value "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -PropertyType String -Force
```

Rechte für administrators_authorized_keys setzen Entzieht vererbte Rechte und beschränkt den Zugriff strikt auf SYSTEM und Administratoren

```
$ icacls.exe $env:ProgramData\ssh\administrators_authorized_keys /inheritance:r /grant "NT AUTHORITY\SYSTEM:(F)" /grant "BUILTIN\Administrators:(F)"
```

SSH-Firewallregel auf Administrationsnetz beschränken Begrenzt eingehende SSH-Verbindungen auf ein definiertes Administrations-Subnetz

```
$ Set-NetFirewallRule -Name "OpenSSH-Server-In-TCP" -RemoteAddress "192.0.2.0/24"
```

Erkannte Festplatten inventarisieren Gibt physische und virtuelle Datenträger samt Partitionsstil und Gesamtkapazität aus

```
$ Get-Disk | Format-Table Number, FriendlyName, OperationalStatus, Size, PartitionStyle
```

Systempartition C: im laufenden Betrieb erweitern Liest Datenträger nach Hypervisor-Resize neu ein und vergrößert Partition 3 auf das Maximum

```
$ Update-Disk -Number 0; Resize-Partition -DiskNumber 0 -PartitionNumber 3 -Size ((Get-PartitionSupportedSize -DiskNumber 0 -PartitionNumber 3).SizeMax)
```

Zusätzliche Festplatte initialisieren und formatieren Richtet einen neuen Datenträger mit GPT und NTFS-Dateisystem in einer Pipeline ein

```
$ Initialize-Disk -Number 1 -PartitionStyle GPT -PassThru | New-Partition -AssignDriveLetter -UseMaximumSize | Format-Volume -FileSystem NTFS -NewFileSystemLabel "Data" -Confirm:$false
```

Speicherplatz freigeben (Trim/Discard) Sendet TRIM-Befehle an den Hypervisor zur Speicherfreigabe auf ZFS-, Ceph- oder Thin-Pools

```
$ Optimize-Volume -DriveLetter C -Defrag -Verbose
```

Laufende Hintergrunddienste abfragen Listet alle aktuell aktiven Dienste auf dem Server tabellarisch auf

```
$ Get-Service | Where-Object Status -eq 'Running' | Format-Table Name, DisplayName, StartType
```

Dienst kontrolliert neu starten Erzwingt den Neustart eines Hintergrunddienstes (z. B. DNS-Client oder SSH-Daemon)

$ `Restart-Service -Name sshd -Force`

Top 10 speicherintensive Prozesse ermitteln Sortiert aktive Prozesse absteigend nach ihrer Arbeitsspeicherbelegung im WorkingSet

$ `Get-Process | Sort-Object WorkingSet64 -Descending | Select-Object -First 10 -Property Id, ProcessName, CPU, @{Name='RAM_MB';Expression={[math]::Round($_.WorkingSet64/1MB,2)}} | Format-Table`

Hängenden Prozess beenden Stoppt einen blockierten oder fehlerhaften Prozess zielgerichtet über seine Prozess-ID

$ `Stop-Process -Id 4328 -Force`

Lokale Benutzerkonten anzeigen Gibt alle lokalen Konten samt Aktivierungsstatus und letzter Anmeldung aus

$ `Get-LocalUser | Format-Table Name, Enabled, PasswordRequired, PasswordLastSet, LastLogon`

Neuen lokalen Administrator anlegen Erstellt ein lokales Benutzerkonto und fügt es der Administratorengruppe hinzu

$ `New-LocalUser -Name "secadmin" -Password $Password; Add-LocalGroupMember -Group "Administratoren" -Member "secadmin"`

Rollen und Features auflisten Zeigt alle auf dem Server installierten Rollen und Windows-Features an

$ `Get-WindowsFeature | Where-Object Installed -eq $true | Format-Table Name, DisplayName`

Windows-Rolle mit Verwaltungswerkzeugen installieren Installiert eine Serverrolle (z. B. DNS-Server) samt zugehörigen Verwaltungs-Cmdlets

$ `Install-WindowsFeature -Name DNS -IncludeManagementTools`

Windows-Updates suchen und installierenLädt alle ausstehenden Qualitäts- und Sicherheitsupdates herunter und startet bei Bedarf neu

$ `Install-WindowsUpdate -MicrosoftUpdate -AcceptAll -AutoReboot`

Installierte Hotfixes und Patches verifizieren Gibt die installierten Windows-Updates absteigend nach Installationsdatum aus

$ `Get-HotFix | Sort-Object InstalledOn -Descending | Format-Table HotFixID, Description, InstalledOn`

Microsoft Defender Signaturen aktualisieren Fragt die neuesten Antiviren-Definitionen online ab und wendet sie unmittelbar an

$ `Update-MpSignature`

Schnellen Systemscan startenFührt eine schnelle Hintergrundprüfung wichtiger Betriebssystembereiche durch

$ `Start-MpScan -ScanType QuickScan`

Ausschluss für Datenbankpfade im Defender definieren Verhindert I/O-Leistungseinbußen durch den Echtzeitschutz bei geschäftskritischen Datenpfaden

$ `Add-MpPreference -ExclusionPath "C:\Data\Database"`

Status der VSS-Writer prüfen Validiert den Zustand aller Schattenkopie-Komponenten für konsistente Hypervisor-Backups

$ `vssadmin list writers`

Status der Zeitsynchronisation prüfen Gibt Stratum, Server-Quelle, Präzision und Drift des Windows-Zeitgeberdienstes aus

$ `w32tm /query /status`

Zeitsynchronisation sofort erzwingen Veranlasst den sofortigen Neuabgleich der Systemzeit mit konfigurierten NTP-Zeitservern

$ `w32tm /resync /rediscover`

VirtIO-Treiber rekursiv von ISO installieren Bindet alle paravirtualisierten KVM-Treiber im Driver Store ein und aktualisiert Geräte

```
$ pnputil.exe /add-driver D:\*.inf /subdirs /install
```

Hardware- und Treiberstatus überprüfen Listet alle Hardwarekomponenten auf, deren Treiberstatus von OK abweicht

```
$ Get-PnpDevice | Where-Object Status -ne 'OK'
```

Netzwerkkarten-Teaming anlegenBündelt mehrere Adapter zu einem redundanten Verbund ohne Switch-Abhängigkeit

```
$ New-NetLbfoTeam -Name "Team01" -TeamMembers "Ethernet0", "Ethernet1" -TeamingMode SwitchIndependent -LoadBalancingAlgorithm Dynamic
```

System-PATH dauerhaft erweitern Trägt ein Skriptverzeichnis persistent auf Maschinenebene in die Windows-Umgebungsvariable ein

```
$ [System.Environment]::SetEnvironmentVariable("Path", "$([System.Environment]::GetEnvironmentVariable('Path','Machine'));C:\Scripts", "Machine")
```

Storage Pool und Mirror-Datenträger erstellen Fasst unpartitionierte Festplatten zu einem fehlertoleranten Software-Spiegel zusammen

```
$ New-StoragePool -FriendlyName "DataPool" -StorageSubsystemFriendlyName "Windows Storage*" -PhysicalDisks (Get-PhysicalDisk -CanPool $true); New-VirtualDisk -StoragePoolFriendlyName "DataPool" -FriendlyName "MirrorVolume" -ResiliencySettingName Mirror -UseMaximumSize
```

IIS Webserver und Tools installieren Aktiviert den Webserver-Dienst samt zugehörigen PowerShell-Modulen

```
$ Install-WindowsFeature -Name Web-Server, Web-Mgmt-Tools
```

Neue IIS-Website und Application Pool erstellen Richtet eine isolierte Webanwendung auf Port 80 mit eigenem Dokumentenpfad ein

```
$ New-WebAppPool -Name "AppPool"; New-WebSite -Name "Site" -Port 80 -PhysicalPath "C:\inetpub\site" -ApplicationPool "AppPool"
```

Selbstsigniertes Zertifikat generieren Erstellt ein 2048-Bit-RSA-Zertifikat für Testdienste im lokalen Zertifikatsspeicher

```
$ New-SelfSignedCertificate -DnsName "win-core01.lab.internal" -CertStoreLocation "Cert:\LocalMachine\My" -NotAfter (Get-Date).AddYears(2)
```

Stammzertifikat importieren Importiert das Zertifikat einer internen CA in den Speicher vertrauenswürdiger Stammzertifizierungsstellen

```
$ Import-Certificate -FilePath "C:\Certs\InternalCA.crt" -CertStoreLocation "Cert:\LocalMachine\Root"
```

Dateisystem-Rechte (NTFS ACL) anwenden Schreibt explizite Zugriffsregeln für Benutzer und Gruppen auf ein Verzeichnis

```
$ Set-Acl -Path "D:\Shares\Deploy" -AclObject $Acl
```

Active Directory Domänenbeitritt Tritt einer Active-Directory-Domäne bei und startet den Server unmittelbar neu

```
$ Add-Computer -DomainName "corp.example.com" -Credential (Get-Credential) -Restart -Force
```

Primäre DNS-Zone erstellen Richtet eine neue Forward-Lookup-Zone auf dem lokalen DNS-Server ein

```
$ Add-DnsServerPrimaryZone -Name "lab.internal" -ZoneFile "lab.internal.dns"
```

Statischen DNS-A-Record eintragenFügt der angegebenen DNS-Zone einen festen IPv4-Adresseintrag hinzu

```
$ Add-DnsServerResourceRecordA -ZoneName "lab.internal" -Name "gateway" -IPv4Address "192.0.2.1"
```

SMB-Freigabe konfigurieren Erstellt eine Netzwerkfreigabe mit vollen Zugriffsrechten für lokale Administratoren

```
$ New-SmbShare -Name "Deploy" -Path "D:\Shares\Deploy" -FullAccess "BUILTIN\Administrators"
```

Wöchentliche Wartungsaufgabe planen Registriert ein PowerShell-Wartungsskript als Scheduled Task mit SYSTEM-Rechten

```
$ Register-ScheduledTask -TaskName "Maintenance" -Action (New-ScheduledTaskAction -Execute "powershell.exe" -Argument "-File C:\Scripts\Maintain-ServerCore.ps1") -Trigger (New-ScheduledTaskTrigger -Weekly -DaysOfWeek Sunday -At 03:00) -Principal (New-ScheduledTaskPrincipal -UserId "NT AUTHORITY\SYSTEM" -RunLevel Highest)
```

Performance-Counter live abfragen Liest aktuelle Systemauslastung für CPU, freien Speicher und Disk-Queue ab

```
$ Get-Counter -Counter "\Processor(_Total)\% Processor Time", "\Memory\Available MBytes"
```

Performance-Logging in CSV exportieren Zeichnet Leistungsindikatoren über ein Intervall auf und speichert sie als CSV

```
$ Get-Counter -Counter "\Processor(_Total)\% Processor Time" -SampleInterval 5 -MaxSamples 60 | Export-Counter -Path "C:\perf.csv" -FileFormat CSV
```

LLMNR-Namensauflösung deaktivieren Schaltet Link-Local Multicast Name Resolution zur Härtung gegen Man-in-the-Middle-Angriffe ab

```
$ Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows NT\DNSClient" -Name "EnableMulticast" -Value 0 -Type DWord
```

NetBIOS über TCP/IP abschalten Deaktiviert NetBIOS auf allen netzwerkfähigen Schnittstellen zur Reduzierung von Broadcasts

```
$ Get-CimInstance Win32_NetworkAdapterConfiguration | Where-Object IPEnabled | Invoke-CimMethod -MethodName SetTcpipNetbios -Arguments @{TcpipNetbiosOptions=2}
```

NTLMv1 verbieten und NTLMv2 erzwingen Konfiguriert die LSA-Authentifizierungsebene auf den sichersten Wert

```
$ Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Lsa" -Name "LmCompatibilityLevel" -Value 5 -Type DWord
```

WinSxS Komponentenspeicher bereinigen und Basis zurücksetzen Entfernt ersetzte Update-Versionen permanent zur Freigabe von bis zu 6 GiB Speicher

$ `dism.exe /Online /Cleanup-Image /StartComponentCleanup /ResetBase`

Netzwerk-Paketmitschnitt starten Startet einen Ringpuffer-Mitschnitt des Netzwerkverkehrs in eine Microsoft-ETL-Datei

$ `netsh trace start capture=yes tracefile=C:\trace.etl maxsize=256 filemode=circular overwrite=yes`

Netzwerk-Paketmitschnitt stoppen Beendet die aktive Paketerfassung und schließt die ETL-Datei

$ `netsh trace stop`

Datei per SCP von Linux auf Windows übertragen Kopiert Skripte oder Pakete über SSH direkt in das Windows-Dateisystem

$ `scp -i ~/.ssh/id_ed25519 setup.ps1 Administrator@192.0.2.10:C:/Scripts/`

Internen Dienst per SSH-Tunnel weiterleiten Leitet einen lokalen Port der Linux-Workstation sicher auf den internen Port des Windows-Servers weiter

$ `ssh -i ~/.ssh/id_ed25519 -L 8080:localhost:80 Administrator@192.0.2.10`

Deutsches Tastaturlayout und Region erzwingen Setzt Sprachliste, Tastaturbelegung und Systemgebietsschema auf Deutsch (QWERTZ)

$ `Set-WinUserLanguageList (New-WinUserLanguageList "de-DE") -Force; Set-WinSystemLocale "de-DE"; Set-Culture "de-DE"`

Proxmox VM-Snapshot vor Wartung erstellen Friert den Zustand der VM auf dem Hypervisor vor Konfigurationsänderungen ein

$ `qm snapshot 104 "pre-upgrade" --description "Snapshot vor Patchday"`

Proxmox VM-Snapshot wiederherstellen Setzt die virtuelle Maschine auf der Proxmox-Shell auf einen gespeicherten Snapshot zurück

$ `qm rollback 104 "pre-upgrade"`

Proxmox VSS-Backup ausführen Erzeugt über vzdump und QEMU Guest Agent ein anwendungskonsistentes Snapshot-Backup

```
$ vzdump 104 --storage local --mode snapshot --compress zstd
```

Benutzerkonto entsperren Hebt die Kontensperre eines lokalen Benutzers nach wiederholten Fehlversuchen auf

```
$ Unlock-LocalUser -Name "Administrator"
```

VSS-Schattenspeicher vergrößern Erweitert das Speicherlimit für Schattenkopien auf dem Systemlaufwerk

```
$ vssadmin resize shadowstorage /for=C: /on=C: /maxsize=25%
```

DNS-Client-Cache leeren Löscht veraltete Namensauflösungs-Einträge aus dem lokalen Resolver-Puffer

```
$ Clear-DnsClientCache
```

Systemdateien mit SFC überprüfen Scannt und repariert beschädigte geschützte Betriebssystemdateien

```
$ sfc.exe /scannow
```

Komponentenspeicher online mit DISM reparieren Stellt beschädigte Windows-Image-Dateien über Windows Update wieder her

```
$ dism.exe /Online /Cleanup-Image /RestoreHealth
```

Systemstartzeit und Uptime abfragen Ermittelt den exakten Zeitpunkt des letzten Bootvorgangs über CIM

```
$ Get-CimInstance Win32_OperatingSystem | Select-Object LastBootUpTime
```

Papierkorb aller Laufwerke leeren Gibt gelöschte Dateien aus dem Papierkorb ohne Bestätigungsabfrage frei

```
$ Clear-RecycleBin -Force -Confirm:$false
```

Aktive lauschende TCP-Ports auflisten Listet alle lokalen Netzwerkports im Status Listen samt Prozess-ID auf

$ `Get-NetTCPConnection -State Listen | Format-Table LocalAddress, LocalPort, OwningProcess`

Firewall-Regel für benutzerdefinierten Port anlegen Erstellt eine eingehende Freigabe für einen benutzerdefinierten TCP-Dienst

$ `New-NetFirewallRule -Name "App-TCP-In" -DisplayName "App Service" -Direction Inbound -Protocol TCP -LocalPort 8080 -Action Allow`

Veralteten SSH-Hostkey auf Linux-Client entfernenLöscht den gespeicherten Serverschlüssel nach Neuinstallation oder Key-Rotation aus known_hosts

$ `ssh-keygen -R 192.0.2.10`

WinRM TrustedHosts für Standalone-Server setzen Erlaubt Remote-Verbindungen zu Standalone-Servern ohne Serverzertifikats-Prüfung außerhalb von Domänen

$ `Set-Item WSMan:\localhost\Client\TrustedHosts -Value "192.0.2.10" -Force`

DNS-Verbindung zu Microsoft Update testen Prüft die Namensauflösung der Windows-Update-Endpunkte bei Fehlern wie 0x8024402C

$ `Resolve-DnsName update.microsoft.com`

Windows Update Cache zurücksetzen Bereinigt korrumpierte Download-Archive und startet den Update-Dienst neu

$ `Stop-Service wuauserv; Remove-Item C:\Windows\SoftwareDistribution\Download\* -Recurse -Force; Start-Service wuauserv`

Datenträger online schalten und Schreibschutz entfernen Behebt den Offline-Status neuer virtueller Festplatten oder SAN-LUNs

$ `Set-Disk -Number 1 -IsOffline $false; Set-Disk -Number 1 -IsReadOnly $false`

VSS-Dienst und QEMU Guest Agent neu starten Setzt hängende Volume Shadow Copy Komponenten bei Backup-Fehlern zurück

```
$ Restart-Service vss, QEMU-GA -Force
```

Server kontrolliert neu starten Führt einen sofortigen, kontrollierten Neustart des Systems über die PowerShell durch

```
$ Restart-Computer -Force
```

Server geordnet herunterfahren Fährt das Betriebssystem kontrolliert herunter und schaltet die virtuelle Maschine aus

```
$ Stop-Computer -Force
```

Notfallzugriff und Wiederherstellung über die Proxmox noVNC-Konsole

Sollte eine fehlerhafte Netzwerkkonfiguration, eine blockierende Firewallregel oder ein Syntaxfehler in der SSH-Konfiguration den Remote-Zugriff vollständig trennen, ist der physische oder virtuelle Konsolenanschluss der einzige verbleibende Rettungsanker.

Gehe bei einem Verbindungsverlust strukturiert in folgenden Schritten vor:

1. **noVNC-Konsole im Proxmox-Dashboard öffnen:**

Navigiere zu deiner virtuellen Maschine (`104`) und öffne über das Menü die **Konsole** (noVNC).

1. **Anmeldedialog aktivieren:**

Klicke oben links im noVNC-Menü auf den Button **`Send Ctrl-Alt-Del`** (oder nutze die Tastenkombination, sofern vom Browser nicht abgefangen), um den gesperrten Bildschirm zu aktivieren.

1. **Lokale Anmeldung durchführen:**

Melde dich mit dem lokalen Administratorkonto und dem bei der Ersteinrichtung definierten Kennwort an.

1. **Notfall-Shell öffnen:**

Bleibt der Bildschirm schwarz, drücke `Strg` + `Umschalt` + `Esc`, um den Task-Manager zu öffnen. Wähle *Datei → Neuen Task ausführen*, gib `powershell.exe` ein und aktiviere die Checkbox *„Diesen Task mit Administratorrechten erstellen“*. **5. Netzwerk- und Firewall-Reset durchführen:**

Setze die Firewall im Notfall auf Standardwerte zurück, um blockierende Filterregeln zu eliminieren:

```
netsh advfirewall reset
```

Starte anschließend das menügeführte SConfig-Werkzeug, um die Netzwerkadapter wieder auf DHCP oder bekannte statische Parameter zu setzen:

```
sconfig
```

6. SSH-Dienst prüfen:

Überprüfe in der PowerShell, ob der SSH-Daemon fehlerfrei startet:

```
Restart-Service sshd
Get-NetTCPConnection -LocalPort 22 -State Listen
```

Sobald Port 22 lokal wieder lauscht und die Firewall zurückgesetzt ist, gelingt die Anmeldung von deiner Linux-Workstation wieder wie gewohnt über das Terminal.

Weiterführende Ressourcen

Die folgenden Dokumentationen, Standards und internen AdminDocs-Module vertiefen die behandelten Administrations- und Virtualisierungskonzepte:

Microsoft Learn: Neuerungen in Windows Server 2025 Offizielle Übersicht über Plattform-Architektur, Sicherheitsfunktionen und Rollen in Windows Server 2025

Microsoft Learn: Windows Server Core verwalten Offizielle Microsoft-Dokumentation zu Verwaltungsmethoden, Rollenunterstützung und Wartung von Windows Server Core

Microsoft Learn: SConfig-Konfigurationsreferenz Vollständige Dokumentation und Parameter für das menügeführte Serverkonfigurationstool

Microsoft Learn: OpenSSH für Windows Installations- und Konfigurationsleitfaden für den nativen Windows-OpenSSH-Server

Microsoft Learn: PowerShell-Architektur und Skripterstellung Offizielle Dokumentation zu PowerShell-Pipelines, WMI/CIM-Steuerung und Modulverwaltung

Proxmox VE Administration Guide: QEMU/KVM Offizieller Hypervisor-Leitfaden zu virtuellen Maschinen, Controllern, vTPM und Hardware-Emulation

Proxmox Wiki: Windows Guest Best Practices Offizielle Dokumentation zur Konfiguration von Windows Server 2025 mit VirtIO-Treibern und Guest Agent

Proxmox VE: Installation und erste VM Schritt-für-Schritt-Anleitung zur Hypervisor-Einrichtung, Speicheranbindung und Netzwerkkonfiguration

Linux Administration: Virtualisierung und VM-Management Tiefer Einblick in KVM-, QEMU- und VirtIO-Architektur für virtualisierte Serverumgebungen

Linux Administration: Systemsicherheit und Härtung Sicherheitsstrategien, Firewall-Architektur und SSH-Härtung im Rechenzentrumsbetrieb

Fazit

Operative Einsatzmatrix und architektonische Bewertung

Die Wahl von Windows Server 2025 Core ist eine strategische Architekturentscheidung, die über die isolierte Bereitstellung einer einzelnen virtuellen Maschine hinausgeht. Für die operative Betriebsplanung im Rechenzentrum liefert der Headless-Betrieb klare Entscheidungskriterien, wann der Core-Einsatz zwingend geboten ist – und wann die klassische Desktop Experience weiterhin unvermeidbar bleibt.

Wann Server Core die bevorzugte Wahl ist:

- **Infrastruktur-Basisdienste:** Active Directory Domain Services (AD DS), DNS, DHCP, Zertifikatsdienste (AD CS) sowie Datei- und Speicherdienste laufen auf Server Core mit maximaler Stabilität, minimalem Ressourcenbedarf und ohne unnötige Wartungsneustarts durch GUI-Patchdays.
- **Heterogene Multiplattform-Umgebungen:** Durch den nativen OpenSSH-Server lässt sich Windows Server 2025 nahtlos in bestehende Linux-Automatisierungen (z. B. Ansible-Playbooks, Terraform oder CI/CD-Pipelines) integrieren, ohne dass proprietäre RPC-Ports oder komplexe WinRM-Zertifikatsketten über Netzwerkgrenzen exponiert werden müssen.
- **Dichte Virtualisierungs-Cluster:** Auf Hypervisoren wie Proxmox VE 9 ermöglicht der geringe Speicher-Overhead (unter 1,5 GiB RAM im Leerlauf) eine spürbar höhere VM-Dichte pro Host-Knoten bei gleichzeitig schnellen Snapshot- und Backup-Zyklen.

Operative Ausschlusskriterien (Wann Desktop Experience erforderlich ist):

- **Benutzerinteraktive Workloads:** Server Core kann prinzipbedingt nicht als Remote Desktop Session Host (RDSH / Terminalserver) für Endanwender betrieben werden.
- **Software-Inkompatibilitäten:** Bestimmte Drittanbieter-Anwendungen (proprietäre Datenbanksysteme, veraltete Backup-Agenten oder Branchensoftware) setzen für Konfigurationsdialoge oder Lizenzierungsroutinen zwingend Windows-GDI- und Explorer-Bibliotheken voraus.
- **Spezialisierte Verwaltungsrollen:** Einzelne Server-Rollen wie der Fax-Server oder Legacy-Dienste, die ausschließlich über lokale grafische Snap-ins ohne Remote-Unterstützung konfiguriert werden können, erfordern weiterhin die vollständige Desktop-Umgebung.

Administrative Qualifikation und Notfallbetrieb:

Der Wechsel zu Server Core verlagert den Aufwand von der Host-Ressourcenebene in das Betriebsteam: Ohne disziplinierte PowerShell-Kompetenz und eine dokumentierte Befehlsreferenz gerät die Fehleranalyse im Störfall ins Stocken.

Sollte eine restriktive Firewall-Regel oder eine fehlerhafte Netzwerkkonfiguration den SSH-Zugriff unterbrechen, existiert auf Server Core keine grafische Benutzeroberfläche zur schnellen Fehlerbehebung. Der primäre Lebensretter ist in solchen Szenarien der Out-of-Band-Zugriff über

die Hypervisor-Konsole (Proxmox noVNC): Über das Konsolenfenster anmelden, mit `sconfig` die Netzwerkeinstellungen auf einen bekannten Status zurücksetzen oder fehlerhafte Adapterkonfigurationen direkt in der lokalen PowerShell bereinigen.

Wer diese operativen Rahmenbedingungen berücksichtigt, erhält mit Windows Server 2025 Core auf Proxmox VE 9 ein extrem stabiles, wartungsarmes und vollständig automatisierbares Fundament für moderne Unternehmensnetzwerke.