RDS-Farmen im Systemhaus standardisieren: jede Farm gleich gut



WISSEN

Rollen, Sammlungen, Lizenzierung und Betrieb der Remotedesktopdienste im Überblick.

› **Remote Desktop Services**

BERATUNG

Einen eigenen Farmstandard für dein Systemhaus entwerfen und an einer Kundenfarm erproben.

› **Beratung zu Remote Desktop Services**

SCHULUNG

Workshop Farmbau: Standardfarm per Skript bauen, prüfen, abnehmen und zurückbauen.

› **Schulung Remote Desktop Services**

Du willst die RDS-Farm standardisieren, weil dein Systemhaus inzwischen zwanzig Terminalserver-Farmen betreut und keine davon der anderen gleicht. Bei Kunde A heißt der Broker TS-BROKER, bei Kunde B SRV-RDS-NEU-2, bei Kunde C steht er auf dem Domänencontroller, weil das „damals schneller ging“. Jede Störung beginnt mit einer Stadtführung durch fremde Architektur, und jeder neue Kollege braucht pro Kunde eine eigene Einarbeitung. Das ist kein Betrieb, das ist Denkmalpflege.

Dieser Beitrag zeigt dir, wie du als Systemhaus einen Farmstandard aufbaust, der über alle Kunden trägt: Namenskonventionen, drei Standardarchitekturen von klein bis hochverfügbar, Baseline-Gruppenrichtlinien, ein Abnahmeprotokoll, eine einheitliche Dokumentation und ein Verfahren, mit dem du Session Hosts ergänzt, ohne dass die Farm auseinanderläuft. Dazu kommt ein Vier-Augen-Prinzip, das mit Trockenlauf und Skriptvorschau auskommt statt mit Formularen. Die Grundlagen zu Connection Broker, RDSH, Gateway und Lizenzierung findest du in der Übersicht zu Remote Desktop Services.

FAKTEN · Das Wichtigste in Kürze

Ein Farmstandard besteht aus wenigen, festen Bausteinen: Namen, Architekturgrößen, Baseline-GPOs, Installationsskripte, Abnahmeprotokoll, Dokumentationsvorlage und Änderungsverfahren.

Drei Größen reichen für die meisten Kunden: S (Rollen zusammengelegt), M (Rollen getrennt, Session Hosts redundant) und L (Connection Broker hochverfügbar mit SQL-Datenbank, Gateways hinter einem Load Balancer).

Die Bereitstellung der Remotedesktopdienste hat keinen eingebauten Sollzustand. Gleichheit entsteht

nur, wenn du sie per Skript herstellst und per Vergleich nachweist.

Lesen kommt vor Schreiben: Jede Bestandsfarm prüfst du zuerst mit Get-RD*-Cmdlets, CIM, GroupPolicy und Get-WinEvent gegen den Standard.

Feste Termine im Standard: Der erweiterte Support für Windows Server 2016 endet am 12.01.2027, die Übergangsfrist ohne Lizenzserver beträgt 120 Tage, Benutzer-CALs werden technisch nicht erzwungen.

Ausgangslage: zwanzig Kunden, zwanzig Farmen, zwanzig Wahrheiten

Im Tagesgeschäft eines Systemhauses zeigt sich fehlende Standardisierung nicht als großer Knall, sondern als Dauerrauschen. Tickets dauern länger, als sie müssten. Die Bereitschaft telefoniert mit dem Kollegen, der die Farm vor vier Jahren gebaut hat, und der Kollege hat inzwischen gekündigt, Elternzeit oder schlicht vergessen, warum RDSH03 einen eigenen Druckertreiber hat. Die Anwender merken davon nur, dass „der Terminalserver mal wieder spinnt“. Das tut er nicht. Er ist nur anders als die anderen.

Was du im Systemhaus erlebst	Was dahinter steckt
Jede Störung beginnt mit „Wie ist das hier eigentlich gebaut?“	Keine einheitliche Architektur, keine Vorlage für die Doku
Neue Session Hosts verhalten sich anders als die alten	Installation von Hand, Reihenfolge und Patchstand zufällig
Bei Kunde X dauert die Anmeldung doppelt so lange	Gewachsene GPOs statt Baseline, Loopback mal so, mal so
Gateway-Zertifikat läuft ab, keiner war gewarnt	Kein Abnahmepunkt für Zertifikate, kein Fristenkalender
Kalkulation für Erweiterungen ist Bauchgefühl	Kein Standardbaustein, also jedes Mal Neuplanung
Einarbeitung pro Kunde statt pro Technologie	Jede Farm ist ein Unikat mit eigenem Wissen
Nach der Änderung geht etwas kaputt, Rückweg unklar	Kein Skript, kein Protokoll, kein Rückbau

AUS DER PRAXIS · Ein Maschinenbauer mit 300 Benutzern und drei Brokern, die keiner wollte

Bei einem Maschinenbauer mit 300 Benutzern fand die Bestandsaufnahme drei Bereitstellungen: eine für die Verwaltung, eine für die Konstruktion und eine, die ein früherer Dienstleister „zum Testen“ angelegt hatte und in der seitdem die Buchhaltung arbeitete. Drei Broker, drei Zertifikate mit drei Ablaufdaten, zwei Lizenzmodi. Die Konsolidierung auf eine Farm der Größe L dauerte weniger lange als die Diskussion darüber, wer die Testfarm eigentlich bezahlt hatte.

Warum RDS-Farmen ohne Standard auseinanderlaufen

Der technische Grund ist unspektakulär: Eine RDS-Bereitstellung kennt keinen Sollzustand. Der Connection Broker speichert, welche Server welche Rollen haben, welche Sammlungen es gibt, welche Zertifikate gesetzt sind und welcher Lizenzmodus gilt. Ob RDSH05 dieselben Updates, dieselbe FSLogix-Version und dieselben Anwendungen hat wie RDSH01, interessiert ihn nicht. Er verteilt Sitzungen fröhlich auf alle Hosts, die in der Sammlung stehen und neue Verbindungen erlauben.

Dazu kommen drei Verstärker. Erstens ist der Server-Manager ein Klickwerkzeug: Was dort gebaut wird, hinterlässt kein Skript und kein Protokoll, nur ein Ergebnis. Zweitens wirken Gruppenrichtlinien je nach OU, Vererbung, Sicherheitsfilter und Loopback-Modus, und jede Farm bekommt im Laufe der Jahre ihre eigene Sammlung von „nur kurz“-GPOs. Drittens wachsen Farmen in Etappen: Der fünfte Session Host kommt zwei Jahre nach den ersten vier, auf einem anderen Build, installiert von jemand anderem. So entsteht Konfigurationsdrift, und zwar nicht aus Nachlässigkeit, sondern aus Physik. Wie du Drift innerhalb einer Farm erkennst, beschreibt Session Hosts verhalten sich unterschiedlich: Konfigurationsdrift erkennen.

WICHTIG · Standard heißt nicht gleich groß, sondern gleich gebaut

Ein Standard zwingt keinen Zehn-Personen-Kunden in eine hochverfügbare Farm. Er sorgt dafür, dass eine kleine Farm bei jedem Kunden gleich klein ist: gleiche Namen, gleiche Rollenverteilung, gleiche GPOs, gleiche Abnahme. Unterschiede sind erlaubt, aber nur als dokumentierte Parameter, nie als Überraschung.

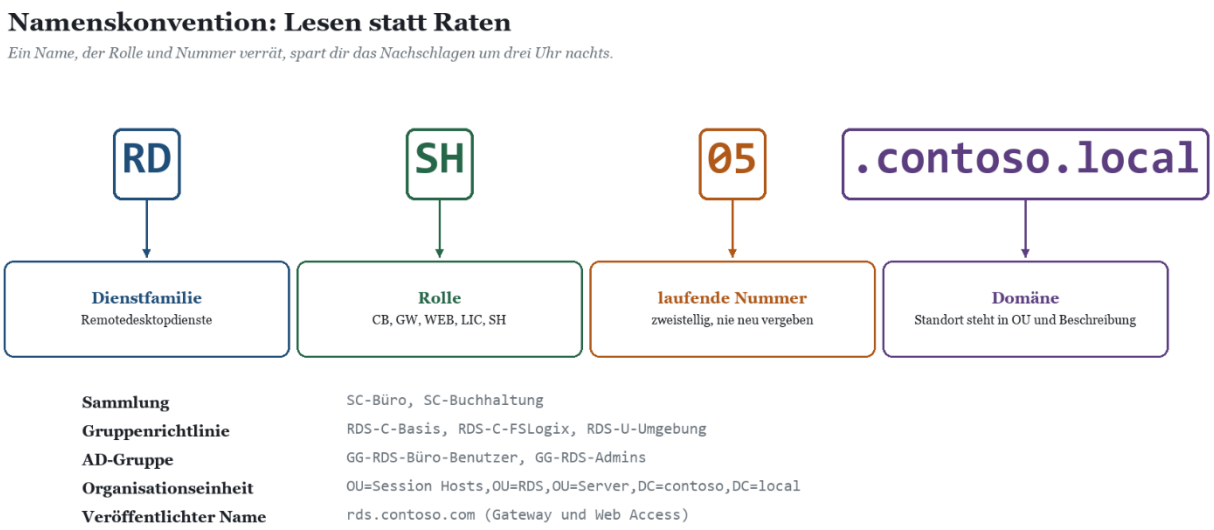
RDS-Farm standardisieren: die Bausteine des Systemhaus-Standards

Ein brauchbarer Standard passt auf wenige Seiten und besteht aus Bausteinen, die du einzeln versionierst. Die folgende Übersicht ist die Kurzfassung, die Abschnitte danach gehen ins Detail.

Standardbaustein	Nutzen
Namenskonvention	Jeder Techniker erkennt Rolle und Zweck eines Servers, einer GPO oder Gruppe ohne Doku
Drei Standardarchitekturen (S, M, L)	Angebot, Kalkulation und Aufbau aus der Schublade statt Neuplanung
Baseline-GPOs	Gleiche Sitzungslimits, Lizenzierung, Profile und Härtung bei jedem Kunden
Installationsskripte aus Bausteinen	Wiederholbarer Aufbau, gleiche Reihenfolge, kein Vergessen
Trockenlauf und Skriptvorschau	Vier-Augen-Prüfung vor der Änderung statt Fehleranalyse danach
Abnahmeprotokoll	Nachweis gegenüber dem Kunden, dass die Farm dem Standard entspricht
Dokumentationsvorlage	Jede Farmdoku hat dieselben Kapitel, jeder findet alles an derselben Stelle
Referenzhost und Soll-Ist-Vergleich	Erweiterung ohne Drift, neue Hosts nachweislich gleich
Rückbauverfahren	Jede Änderung hat einen geplanten Rückweg
Versionierter Standard	Klar, welche Farm auf welchem Stand ist und was ein Update bedeutet

Namenskonventionen, die nachts um drei noch funktionieren

Die beste Namenskonvention ist langweilig. Sie verrät die Rolle, hat eine laufende Nummer und verzichtet auf Kreativität. RDSH05 ist ein Session Host, RDCB01 ein Connection Broker, RDGW01 ein Gateway. Der Standort gehört nicht zwingend in den Namen; bei Kunden mit Standorten in Frankfurt, Köln und Hamburg hältst du ihn sauberer in OU-Struktur, Active-Directory-Beschreibung und Dokumentation fest. Wichtig ist nur, dass die Regel bei allen Kunden gleich ist.



Skizze: Aufbau eines Servernamens und weitere Namensregeln des Standards, Beispiel Contoso

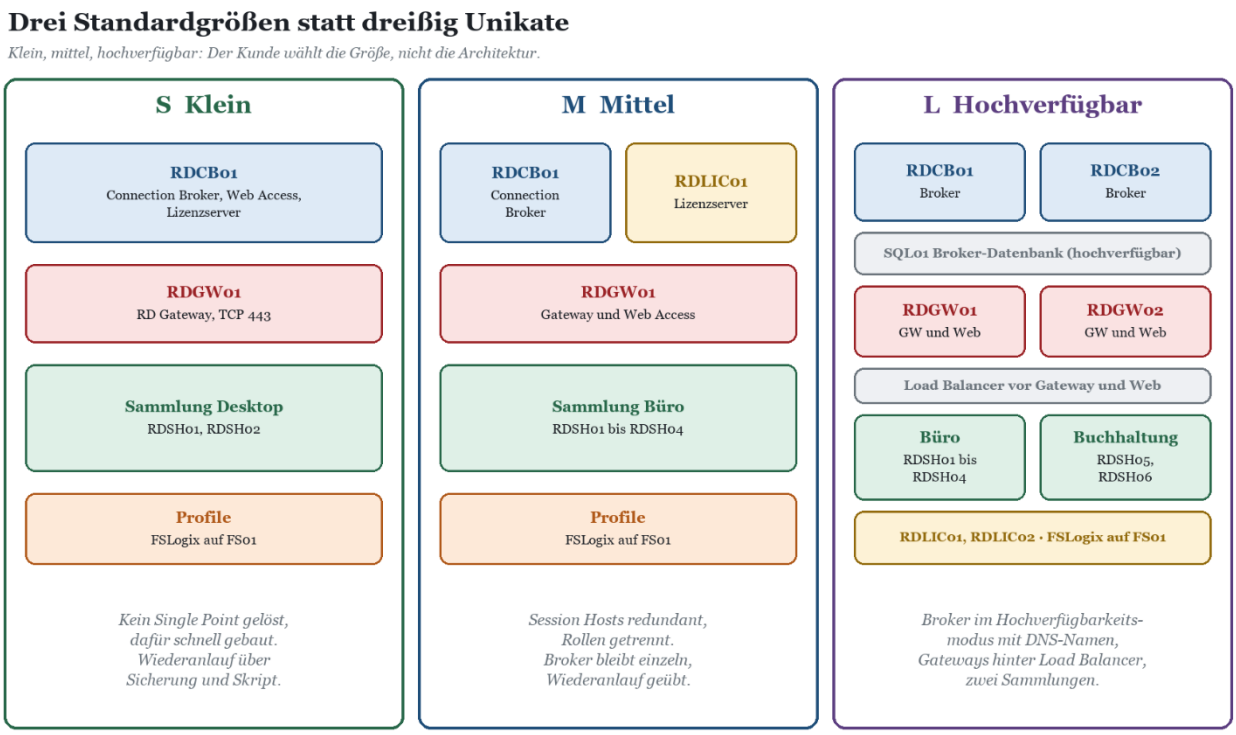
Objekt	Regel	Beispiel
Connection Broker	RDCB + Nummer	RDCB01, RDCB02
Gateway und Web Access	RDGW + Nummer, Web Access in M und L mit auf dem Gateway	RDGW01
Lizenzserver	RDLIC + Nummer, in S auf dem Broker	RDLIC01
Session Host	RDSH + Nummer, Nummern nie neu vergeben	RDSH01 bis RDSH06
Sammlung	SC- + Zweck	SC-Büro, SC-Buchhaltung
Gruppenrichtlinie	RDS- + C (Computer) oder U (Benutzer) + Zweck	RDS-C-Basis, RDS-U-Umgebung
Gruppen	GG-RDS- + Sammlung + Rolle	GG-RDS-Büro-Benutzer
Veröffentlichter Name	ein Name für Gateway und Web Access	rds.contoso.com
Broker-DNS-Name (L)	eigener Name für den Broker-Verbund	rdcb.contoso.local

TIPP · Nummern sind Einwegware

Wird RDSH03 ersetzt, heißt der Nachfolger RDSH07 und nicht wieder RDSH03. Sonst verwechselst du in Protokollen, Ereignissen und Tickets zwei verschiedene Maschinen, und die Fehleranalyse wird zur Archäologie mit Zeitreise.

Drei Standardarchitekturen: klein, mittel, hochverfügbar

Die Microsoft-Dokumentation erlaubt es, die Rollen Connection Broker, Web Access, Gateway und Lizenzierung auf wenigen Servern zusammenzufassen, und beschreibt für die Hochverfügbarkeit des Brokers eine gemeinsame Datenbank, etwa eine SQL-Server-Verfügbarkeitsgruppe oder Azure SQL Database, plus einen DNS-Namen für den Broker-Verbund. Aus diesen Möglichkeiten machst du drei feste Größen. Mehr braucht ein Systemhaus selten, weniger reicht fast nie.



Skizze: Die drei Standardgrößen einer RDS-Farm im Systemhaus, Beispiel contoso.local (vereinfacht)

Merkmal	S Klein	M Mittel	L Hochverfügbar
Connection Broker	RDCB01, mit Web Access und Lizenzserver	RDCB01 allein	RDCB01 und RDCB02 im HA-Modus
Broker-Datenbank	lokal (Standard)	lokal (Standard)	SQL-Server hochverfügbar oder Azure SQL
Gateway und Web Access	RDGW01	RDGW01	RDGW01 und RDGW02 hinter Load Balancer
Lizenzserver	auf RDCB01	RDLIC01	RDLIC01 und RDLIC02
Session Hosts	2	3 bis 6	ab 4, mehrere Sammlungen
Profile	FSLogix auf FS01	FSLogix auf FS01	FSLogix auf hochverfügbarer Freigabe
Ausfall Broker	Wiederaufbau per Skript	Wiederaufbau per Skript	zweiter Broker übernimmt
Typischer Kunde	Büro mit einem Standort	Mittelstand, ein bis zwei Standorte	mehrere Standorte, Schichtbetrieb

Die Anzahl der Session Hosts ist ein Startwert und kein Versprechen. Wie viele Benutzer ein Host verträgt, hängt an den Anwendungen und gehört per Messung belegt, siehe Wie viele Benutzer pro Terminalserver? Lasttest statt Daumenregel. Für die Größe L brauchst du einen Load Balancer vor Gateway und Web Access, zum Beispiel einen Kemp LoadMaster, und für die Broker-Hochverfügbarkeit das Vorgehen aus RDS Connection Broker hochverfügbar machen. Die Einzelheiten zur Datenbank stehen in der Hochverfügbarkeit des Connection Brokers in der Microsoft-Dokumentation.

WARNUNG · Kein Broker auf dem Domänencontroller, auch nicht in Größe S

Rollen der Remotedesktopdienste auf einem Domänencontroller sind die Abkürzung, die später am meisten kostet: andere Sicherheitsanforderungen, keine sauberen Rechte, keine saubere Migration.

Größe S heißt „wenige Server“, nicht „alles auf einen“.

Baseline-GPOs: einmal richtig, dann überall gleich

Gruppenrichtlinien sind der Ort, an dem Standards am leisesten sterben. Deshalb definierst du einen festen Satz Baseline-GPOs, exportierst sie mit Backup-GPO, legst sie versioniert ab und importierst sie bei jedem Kunden mit Import-GPO. Kundenspezifisches kommt in eine eigene, klar benannte GPO, nie in die Baseline. Die Richtlinien für die Remotedesktopdienste findest du in der Gruppenrichtlinienverwaltung unter Computerkonfiguration › Richtlinien › Administrative Vorlagen › Windows-Komponenten › Remotedesktopdienste › Remotedesktopsitzungs-Host. Wie Loopback dabei sauber funktioniert, erklärt Gruppenrichtlinien für Terminalserver: Best Practice mit Loopback.

Baseline-GPO	Inhalt	Verknüpft mit
RDS-C-Basis	Loopback (Zusammenführen), Sitzungszeitlimits, Lizenzmodus und Lizenzserver	OU Session Hosts
RDS-C-FSLogix	Profilcontainer, Pfad, Größe, Ausschlüsse	OU Session Hosts
RDS-C-Härtung	Gerätekomponenten-Umleitung, Verschlüsselung, NLA	OU Session Hosts
RDS-C-Infrastruktur	Firewall, Remoteverwaltung, Updates	OU RDS-Infrastruktur
RDS-U-Umgebung	Benutzereinstellungen über Loopback	OU Session Hosts
RDS-C-Kunde-<Zweck>	alles Kundenspezifische, einzeln dokumentiert	nach Bedarf

Was in die Härtungs-GPO gehört, hat eine eigene Checkliste in RDS-Farm härten: Sicherheits-Checkliste für Terminalserver.

Abnahmeprotokoll: Was nicht geprüft ist, ist nicht fertig

Das Abnahmeprotokoll ist der Moment, in dem aus „sollte gehen“ ein „ist geprüft“ wird. Es enthält feste Prüfpunkte mit Methode und Sollwert, ist bei jedem Kunden gleich und wird unterschrieben, idealerweise von zwei Personen: einer, die gebaut hat, und einer, die nicht gebaut hat.

Prüfpunkt	Methode	Sollwert
Rollen und Server	Get-RDServer	alle Rollen laut Größe, keine Fremdserver
Sammlungen	Get-RDSessionCollection, Get-RDSessionHost	Hosts laut Plan, alle erlauben Verbindungen
Zertifikate	Get-RDCertificate	alle vier Rollen vertrauenswürdig, Ablauf > 90 Tage
Lizenzierung	Get-RDLicenseConfiguration	Modus laut Vertrag, Lizenzserver eingetragen
Broker-HA (nur L)	Get-RDConnectionBrokerHighAvailability	beide Broker aktiv, DNS-Name gesetzt
Gleichheit der Hosts	Get-CimInstance, Updates, Softwareliste	Build und Updates identisch zum Referenzhost
Gruppenrichtlinien	Get-GPInheritance, Ergebnis je Host	Baseline-GPOs verknüpft, keine Fremd-GPOs
Erreichbarkeit	Test-NetConnection rds.contoso.com -Port 443	erreichbar von intern und extern
Ereignisse	Get-WinEvent seit Inbetriebnahme	keine Fehler ohne Erklärung
Anmeldung	Testkonto je Sammlung	Desktop oder RemoteApp startet, Profil

		wird angelegt

Dokumentation: eine Vorlage für alle Kunden

Die Dokumentation folgt bei jedem Kunden derselben Gliederung: Architektur, Rollen, Zertifikate mit Ablauf, Sammlungen, Gruppenrichtlinien, Profile, Lizenzierung, Software je Host, Betriebshandbuch, Änderungsprotokoll. Dazu kommt im Systemhaus ein Kapitel, das oft fehlt: die Abweichungen vom Standard mit Begründung und Ablaufdatum. Was in die einzelnen Kapitel gehört, steht ausführlich in RDS-Farm dokumentieren: was in eine gute Farmdokumentation gehört.

Vorgehen: Bestandsfarm gegen den Standard prüfen, Schritt für Schritt

Bevor du eine bestehende Kundenfarm auf den Standard hebst, liest du sie aus. Die folgenden Befehle sind ausschließlich lesend. Du führst sie auf dem Connection Broker oder einem Verwaltungsserver mit installiertem Modul RemoteDesktop aus, mit einem Konto, das auf den Servern lesen darf. Die Beispiele verwenden contoso.local.

WICHTIG · Erst lesen, dann vergleichen, dann ändern

Keine Änderung, bevor der Ist-Stand gesichert und mit dem Standard abgeglichen ist. Sonst hast du hinterher zwei Probleme: das alte und deins.

Schritt 1: Rollen, Sammlungen und Broker auslesen

Im Server-Manager findest du dasselbe unter Remotedesktopdienste › Übersicht › Bereitstellungsübersicht. Per PowerShell ist es wiederholbar und vergleichbar:

```
Import-Module RemoteDesktop
$cb = 'rdcb01.contoso.local'

Get-RDServer -ConnectionBroker $cb | Sort-Object Server |
    Format-Table Server, @{n='Rollen'; e={$_.Roles -join ', '}}

Get-RDSessionCollection -ConnectionBroker $cb
Get-RDSessionHost -CollectionName 'SC-Büro' -ConnectionBroker $cb

# nur in Größe L
Get-RDConnectionBrokerHighAvailability -ConnectionBroker $cb
```

Abgleich mit dem Standard: Stimmen die Rollen mit der Größe überein? Gibt es Server, die nicht nach der Konvention heißen? Ein kleines Regex genügt:

```
Get-RDServer -ConnectionBroker $cb |
    Where-Object { ($_.Server -split '\.')[0] -notmatch '^RD(CB|GW|LIC|SH)\d{2}$' } |
    Select-Object Server, Roles
```

Schritt 2: Zertifikate, Lizenzierung und Sammlungseinstellungen

```
Get-RDCertificate -ConnectionBroker $cb |
    Format-Table Role, Level, Subject, ExpiresOn

Get-RDLicenseConfiguration -ConnectionBroker $cb |
    Format-List Mode, LicenseServer

Get-RDSessionCollectionConfiguration -CollectionName 'SC-Büro' -ConnectionBroker $cb |
    Format-List CollectionName, *SessionLimitMin, AutomaticReconnectionEnabled
```

```
Get-RDSessionCollectionConfiguration -CollectionName 'SC-Büro' -UserGroup -ConnectionBroker $cb
```

FAKTEN · Lizenzierung im Standard festhalten

Ohne erreichbaren Lizenzserver arbeitet eine Farm nur während der Übergangsfrist von 120 Tagen.

Benutzer-CALs werden technisch nicht erzwungen. Der gezählte Bestand gehört deshalb ins Abnahmeprotokoll und in die Dokumentation, nicht nur in den Ordner mit den Rechnungen.

Welches CAL-Modell zu welchem Kunden passt, klärt RDS-CALs pro Benutzer oder pro Gerät: wie viele brauchst du wirklich?.

Schritt 3: Gleichheit der Session Hosts per CIM prüfen

```
$hosts = (Get-RDSessionHost -CollectionName 'SC-Büro' -ConnectionBroker $cb).SessionHost

Get-CimInstance -ClassName Win32_OperatingSystem -ComputerName $hosts |
    Select-Object PSComputerName, Caption, Version, BuildNumber, LastBootUpTime

# Updates je Host, dann Abweichungen gegen den Referenzhost RDSH01
$upd = Get-CimInstance -ClassName Win32_QuickFixEngineering -ComputerName $hosts
$ref = ($upd | Where-Object PSComputerName -like 'RDSH01*').HotFixID
foreach ($h in $hosts) {
    $ist = ($upd | Where-Object PSComputerName -eq $h).HotFixID
    Compare-Object $ref $ist | Select-Object @{n='Host';e={$h}}, InputObject, SideIndicator
}
```

Dasselbe Muster funktioniert für die FSLogix-Version, die installierte Software aus den Uninstall-Schlüsseln der Registry oder die Dienste. Der Vergleich gegen einen Referenzhost ist das Herzstück jedes Standards: Er beantwortet die Frage „gleich oder nicht?“ mit einer Liste statt mit einem Gefühl.

Schritt 4: Gruppenrichtlinien gegen die Baseline prüfen

In der Gruppenrichtlinienverwaltung siehst du unter der OU der Session Hosts im Reiter „Gruppenrichtlinienvererbung“, welche GPOs in welcher Reihenfolge wirken. Per Modul GroupPolicy geht es vergleichbar:

```
Import-Module GroupPolicy
$ou = 'OU=Session Hosts,OU=RDS,OU=Server,DC=contoso,DC=local'

(Get-GPInheritance -Target $ou).InheritedGpoLinks |
    Select-Object Order, DisplayName, Enabled, Enforced

# Abweichungen: GPOs, die nicht zur Baseline gehören
$baseline = 'RDS-C-Basis','RDS-C-FSLogix','RDS-C-Härtung','RDS-U-Umgebung'
(Get-GPInheritance -Target $ou).InheritedGpoLinks |
    Where-Object { $_.DisplayName -notin $baseline -and $_.DisplayName -notlike 'RDS-C-Kunde-*' }

# Bericht für die Doku
Get-GPOReport -Name 'RDS-C-Basis' -ReportType Html -Path C:\Abnahme\RDS-C-Basis.html
```

Schritt 5: Ereignisse der letzten Tage sichten

```
$log = 'Microsoft-Windows-TerminalServices-LocalSessionManager/Operational'
Invoke-Command -ComputerName $hosts -ScriptBlock {
    Get-WinEvent -FilterHashtable @{ LogName = $using:log; Level = 2,3;
                                     StartTime = (Get-Date).AddDays(-7) } -ErrorAction
SilentlyContinue
} | Group-Object PSComputerName, Id | Sort-Object Count -Descending | Select-Object -First 20 Count,
Name
```

Häufen sich Fehler auf einem einzigen Host, ist das fast immer ein Hinweis auf Drift. Was die einzelnen Ereignisse bedeuten, ist ein Thema für sich; für die Standardisierung reicht die Frage, ob alle Hosts dasselbe Muster zeigen.

Schritt 6: Ist-Stand sichern

```
$ziel = "C:\Abnahme\Contoso\$(Get-Date -Format yyyy-MM-dd)"
New-Item -ItemType Directory -Path $ziel -Force | Out-Null
Get-RDServer -ConnectionBroker $cb | Export-Clixml "$ziel\Server.xml"
Get-RDCertificate -ConnectionBroker $cb | Export-Clixml "$ziel\Zertifikate.xml"
Backup-GPO -All -Path $ziel   # Sicherung der GPOs vor jeder Änderung
```

Damit hast du einen datierten Ausgangspunkt. Jede spätere Änderung vergleichst du gegen genau diesen Stand, und jede GPO lässt sich mit Restore-GPO auf ihn zurücksetzen.

Lösung: Standard ausrollen, Farm erweitern, sauber zurückbauen

Skripte aus Bausteinen statt Klickstrecken

Eine Standardfarm baust du nicht im Server-Manager, sondern per Skript aus wiederverwendbaren Bausteinen: Voraussetzungen prüfen, Bereitstellung anlegen, Gateway und Lizenzserver aufnehmen, Lizenzmodus setzen, Sammlung anlegen, Baseline-GPOs importieren. Jeder Baustein ist ein eigenes, versioniertes Stück PowerShell. Die Parameter je Kunde stehen getrennt davon. Die komplette Installationsfolge zeigt RDS-Farm per PowerShell installieren: Schritt für Schritt; hier geht es um das Prinzip, das daraus einen Standard macht.

```
param(
    [ValidateSet('S','M','L')] [string] $Stufe = 'M',
    [switch] $Scharf      # ohne -Scharf nur Trockenlauf
)
$cb = 'rdcb01.contoso.local'

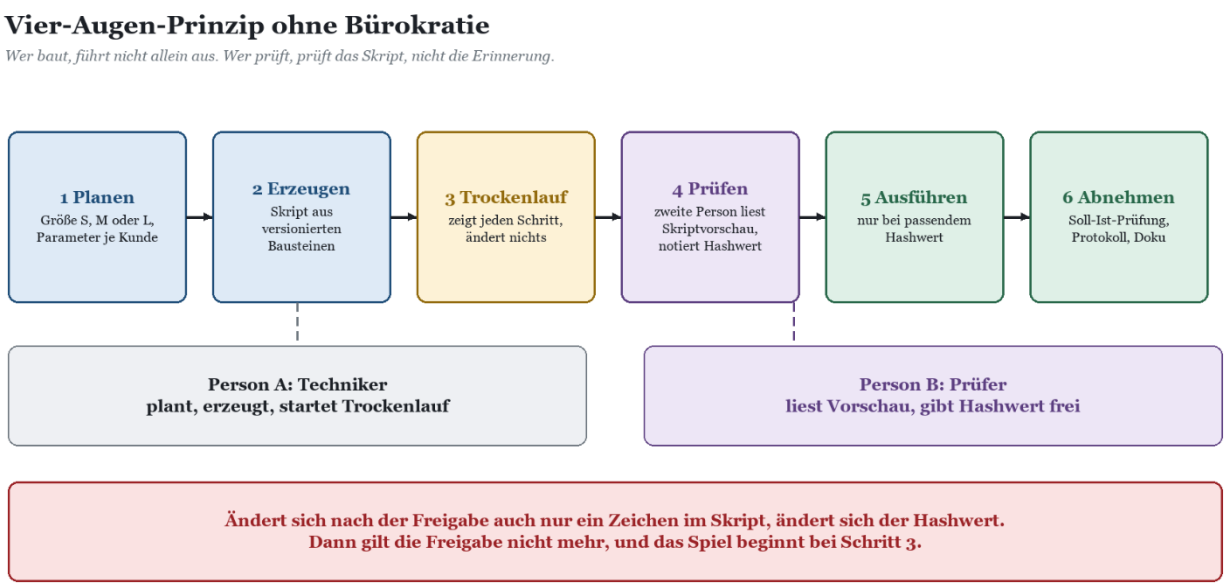
function Invoke-Schritt([string]$Text, [scriptblock]$Aktion) {
    if ($Scharf) { Write-Host "AUSFÜHREN: $Text"; & $Aktion }
    else         { Write-Host "TROCKENLAUF: $Text" }
}

Invoke-Schritt 'Bereitstellung anlegen' {
    New-RDSessionDeployment -ConnectionBroker $cb -WebAccessServer 'rdgw01.contoso.local' `
        -SessionHost 'rdsh01.contoso.local','rdsh02.contoso.local' }
Invoke-Schritt 'Lizenzserver und Modus setzen' {
    Add-RDServer -Server 'rdlic01.contoso.local' -Role RDS-LICENSING -ConnectionBroker $cb
    Set-RDLicenseConfiguration -LicenseServer 'rdlic01.contoso.local' -Mode PerUser `
        -ConnectionBroker $cb -Force }
Invoke-Schritt 'Sammlung SC-Büro anlegen' {
    New-RDSessionCollection -CollectionName 'SC-Büro' -ConnectionBroker $cb `
        -SessionHost 'rdsh01.contoso.local','rdsh02.contoso.local' }
```

Der Trockenlauf ist hier bewusst selbst gebaut: Die Cmdlets des Moduls RemoteDesktop bieten nicht durchgängig einen Parameter -WhatIf. Ein eigener Schalter, der ohne ausdrückliche Freigabe nur beschreibt, was passieren würde, ist zuverlässiger als die Hoffnung, dass jedes Cmdlet ihn unterstützt.

Vier-Augen-Prinzip über Trockenlauf und Skriptvorschau

Im Systemhaus heißt Vier-Augen-Prinzip oft: Einer macht, der andere hat theoretisch zugestimmt. Praktischer ist ein Verfahren, bei dem die zweite Person genau das Skript prüft, das später läuft, und bei dem sich das nachweisen lässt. Dafür genügt ein Hashwert.



Skizze: Freigabeablauf mit Trockenlauf, Skriptvorschau und Hashwert (Beispiel)

```
# Person A: Trockenlauf und Hashwert
.\Farm-Contoso.ps1 -Stufe M          # nur Ausgabe, keine Änderung
Get-FileHash .\Farm-Contoso.ps1 -Algorithm SHA256 | Select-Object Hash

# Person B: liest Skript und Trockenlauf, trägt den Hash ins Ticket ein

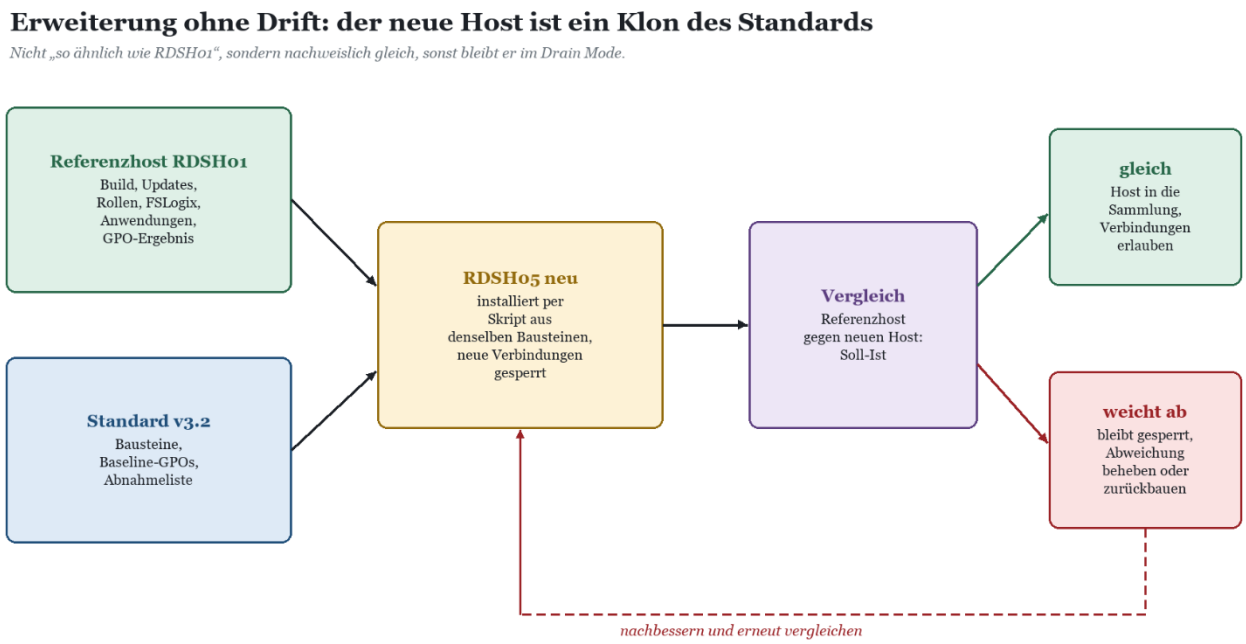
# Person A: vor dem Ausführen gegen den freigegebenen Hash prüfen
$frei = '<Hash aus dem Ticket>'
if ((Get-FileHash .\Farm-Contoso.ps1 -Algorithm SHA256).Hash -ne $frei) {
    throw 'Skript wurde nach der Freigabe verändert.' }
Start-Transcript -Path C:\Abnahme\Contoso\Lauf.txt
.\Farm-Contoso.ps1 -Stufe M -Scharf
Stop-Transcript
```

TIPP · Die zweite Person prüft Ergebnis und Rückweg

Gute Prüffragen für die Skriptvorschau: Passen Hostnamen und Sammlung zum Plan? Gibt es für jeden Abschnitt einen Rückbau? Wird irgendwo neu gestartet, und ist das mit dem Kunden abgestimmt? Wer nur auf Syntax schaut, prüft den Compiler, nicht die Änderung.

Erweiterung ohne Drift: neuen Session Host hinzufügen

Die häufigste Änderung im Systemhausalltag ist der zusätzliche Session Host. Genau hier entsteht Drift, wenn der neue Host „so ähnlich“ installiert wird. Im Standard läuft es immer gleich: Host per Skript aus denselben Bausteinen aufbauen, in die Sammlung aufnehmen, sofort für neue Verbindungen sperren, gegen den Referenzhost vergleichen, erst dann freigeben.



Skizze: Session Host nach Standard hinzufügen und vor der Freigabe gegen den Referenzhost prüfen

```
$neu = 'rdsh05.contoso.local'
Add-RDServer -Server $neu -Role RDS-RD-SERVER -ConnectionBroker $cb
Add-RDSessionHost -CollectionName 'SC-Büro' -SessionHost $neu -ConnectionBroker $cb
Set-RDSessionHost -SessionHost $neu -NewConnectionAllowed No -ConnectionBroker $cb

# Vergleich gegen den Referenzhost wie in Schritt 3, danach freigeben:
Set-RDSessionHost -SessionHost $neu -NewConnectionAllowed Yes -ConnectionBroker $cb
```

Was beim Hinzufügen sonst noch schiefgehen kann, etwa Neustarts während der Rolleninstallation oder fehlende Zertifikate, beschreibt Session Host zur bestehenden RDS-Farm hinzufügen, ohne Überraschungen. Wie du einen Host umgekehrt sauber leerlaufen lässt, steht in Session Host sauber außer Betrieb nehmen: Drain Mode richtig nutzen.

Rollback: jeder Abschnitt hat einen Rückweg

Zu jedem Baustein gehört ein Gegenstück. Für den neuen Session Host sieht der Rückbau so aus:

```
Set-RDSessionHost -SessionHost $neu -NewConnectionAllowed No -ConnectionBroker $cb
# warten, bis keine Sitzungen mehr laufen:
Get-RDUserSession -ConnectionBroker $cb | Where-Object HostServer -eq $neu
Remove-RDSessionHost -SessionHost $neu -ConnectionBroker $cb
Remove-RDServer -Server $neu -Role RDS-RD-SERVER -ConnectionBroker $cb

# GPO-Änderungen zurücksetzen aus der Sicherung von Schritt 6
Restore-GPO -Name 'RDS-C-Basis' -Path 'C:\Abnahme\Contoso\<Datum>'
```

WARNUNG · Rückbau ist kein Zeitreise-Knopf

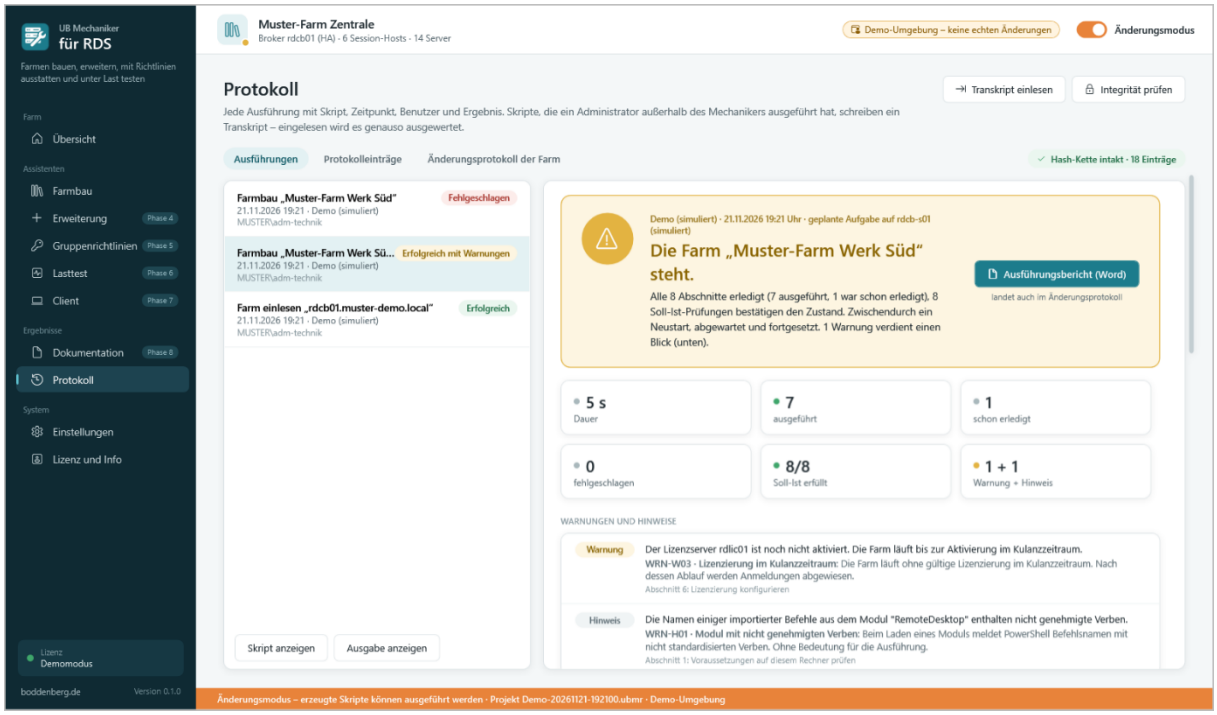
Remove-RDServer entfernt den Server aus der Bereitstellung, deinstalliert aber nicht automatisch jede Rolle und setzt keine Anwendung zurück. Bei größeren Änderungen ist der sicherste Rückweg deshalb ein Snapshot oder eine Sicherung der betroffenen Server vor dem Lauf, abgestimmt mit dem Kunden. Bei virtualisierten Brokern in Größe L gilt besondere Vorsicht: Snapshots von Brokern und Datenbank müssen zueinander passen.

SO HILFT DER UB MECHANIKER FÜR RDS · Wiederverwendbare Projekte, Skripterzeugung, Protokoll und Farmdokumentation

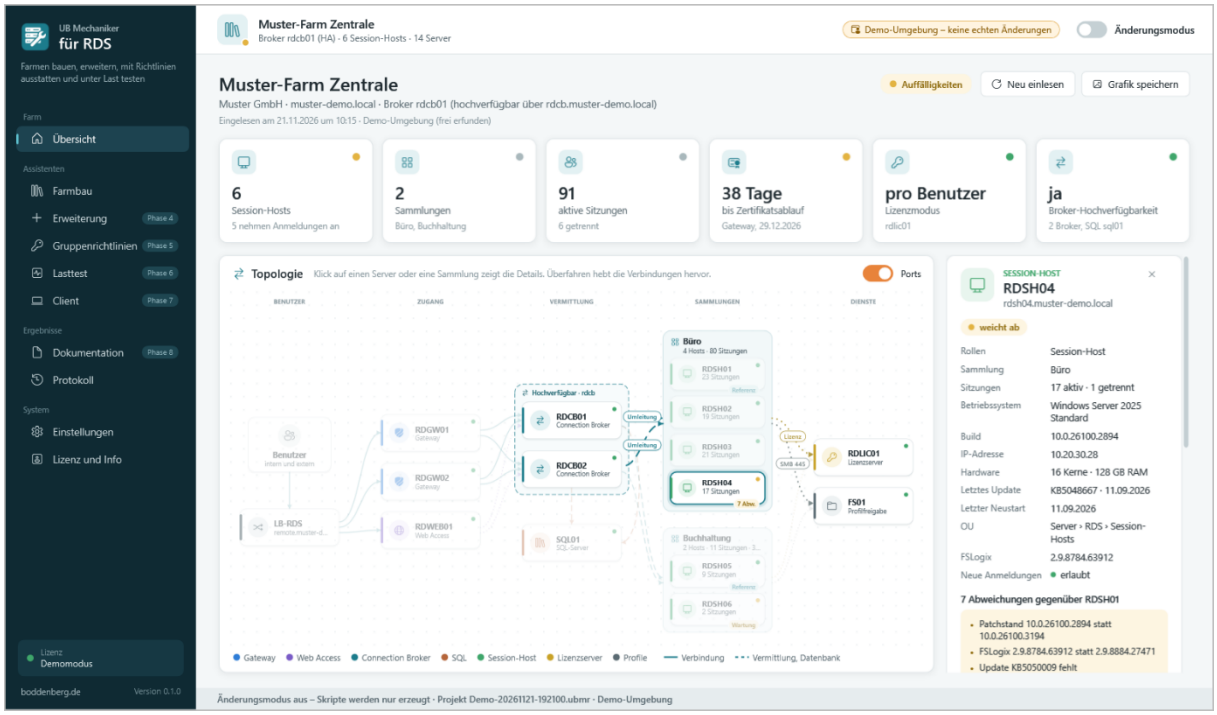
Der UB Mechaniker für RDS bildet den Ablauf aus diesem Beitrag in einer Oberfläche ab. Eine Farm planst du als Projekt und verwendest die Planung für die nächste Kundenfarm wieder. Aus geprüften, versionierten PowerShell-Bausteinen erzeugt der Mechaniker das Skript, zeigt es als Skriptvorschau und speichert es zunächst als Trockenlauf. Das ist die Stelle, an der die zweite Person prüft. Eine Vorabprüfung mit Ampel kontrolliert vorher DNS, Netz, Domäne, Zertifikate, Rollen und Ausführungsort.

Bei der Ausführung siehst du den Fortschritt je Abschnitt, Neustarts werden erkannt. Danach liest der Mechaniker die Farm erneut ein und prüft Soll gegen Ist, ein Fehlerkatalog nennt Ursache und Behebung, und zu jedem Lauf entsteht ein Rückbau-Skript. Jede Ausführung landet in einem Protokoll mit Hash-Kette, sodass sich nachträgliche Änderungen an den Einträgen erkennen lassen. Für Erweiterungen hilft der Referenzhost-Vergleich, für die Übergabe die Farmdokumentation in Word. Bestandsfarmen liest du im Nur-lesend-Modus ein. Ein Assistent für Gruppenrichtlinien ist in Fertigstellung.

Mehr dazu auf der Produktseite UB Mechaniker für RDS. Anfrage über die Kontaktseite.



So siehst du im Protokoll jede Ausführung mit Ergebnis, Kennzahlen und Warnungen und öffnest den Ausführungsbericht als Word-Datei, Demo-Daten.



So zeigt dir die Topologie, welcher Session Host vom Referenzhost abweicht und worin, Demo-Daten.

Wer zusätzlich laufend prüfen will, ob Kundenfarmen vom abgenommenen Stand abweichen, findet Baselines und Konfigurationsdrift im RDS Diagnostiker.

Vorbeugung: den Standard pflegen statt einfrieren

Ein Standard, der nie aktualisiert wird, ist nach zwei Jahren ein Museum. Ein Standard, der ständig geändert wird, ist keiner. Dazwischen liegt ein einfacher Rhythmus.

- **Versionsnummer:** Der Standard hat eine Version, jede Kundenfarm in ihrer Doku die Version, nach der sie gebaut oder zuletzt abgenommen wurde.
- **Quartalsrunde:** Einmal im Quartal prüft das Team neue Updates, FSLogix-Versionen und Erfahrungen aus Tickets und beschließt, was in die nächste Version kommt.
- **Abweichungsregister:** Jede Abweichung vom Standard wird mit Grund, Verantwortlichem und Ablaufdatum dokumentiert. Abweichungen ohne Ablaufdatum werden zum neuen Standard oder verschwinden.
- **Regelmäßiger Abgleich:** Die lesenden Befehle aus dem Vorgehen laufen per Aufgabenplanung regelmäßig und werden mit dem abgenommenen Stand verglichen, siehe RDS Health Check: Farm regelmäßig prüfen statt nur reagieren.
- **Fristenkalender:** Zertifikatsabläufe, Supportenden und Lizenzlaufzeiten aller Kunden in einem gemeinsamen Kalender. Zertifikate aus einer eigenen PKI lassen sich besser planen, siehe PKI mit ADCS.
- **Betriebssystem-Basis:** Neue Farmen baust du auf einer aktuellen Windows-Server-Version. Farmen auf Windows Server 2016 brauchen einen Plan, denn der erweiterte Support endet am 12.01.2027, siehe RDS-Migration von Windows Server 2016 auf 2025: die richtige Reihenfolge.
- **Ausbildung:** Jeder Techniker, der Farmen baut, hat den Standard einmal selbst in einer Übungsumgebung aufgebaut, abgenommen und zurückgebaut.

Für den letzten Punkt bietet die Schulung Remote Desktop Services einen Workshop Farmbau an: Dort baust du eine Standardfarm per Skript, prüfst sie mit einem Abnahmeprotokoll und übst den Rückbau. Wenn du den Standard für dein Systemhaus erst entwerfen oder an einer schwierigen Kundenfarm erproben willst, unterstützt dich die Beratung zu Remote Desktop Services.

TIPP · Der Praxistest für deinen Standard

Gib einem Techniker, der eine Kundenfarm noch nie gesehen hat, die Doku und eine Aufgabe: „Füge einen Session Host hinzu.“ Schafft er das ohne Rückfrage und besteht der neue Host den Referenzvergleich, funktioniert dein Standard. Braucht er drei Anrufe, hast du eine Vorlage, aber noch keinen Standard.

FAQ: RDS-Farm standardisieren

Wie standardisiere ich mehrere RDS-Farmen im Systemhaus?

Mit wenigen festen Bausteinen: Namenskonvention, drei Architekturgrößen, Baseline-GPOs, Installationsskripte aus versionierten Bausteinen, Abnahmeprotokoll, Dokumentationsvorlage und einem Verfahren für Erweiterung und Rückbau. Bestandsfarmen liest du zuerst lesend aus und gleichst sie gegen den Standard ab.

Welche Namenskonvention eignet sich für Terminalserver?

Eine, die Rolle und laufende Nummer zeigt: RDCB01 für den Connection Broker, RDGW01 für das Gateway, RDLIC01 für den Lizenzserver, RDSH01 bis RDSH06 für die Session Hosts. Nummern werden nicht neu vergeben, Standorte hältst du in OU und Beschreibung fest.

Welche Größen braucht eine Standard-RDS-Architektur?

Drei reichen meist: S mit zusammengelegten Rollen und zwei Session Hosts, M mit getrennten Rollen und mehreren RDSH, L mit hochverfügbarem Connection Broker, gemeinsamer SQL-Datenbank und Gateways hinter einem Load Balancer.

Wie füge ich einen Session Host hinzu, ohne dass die Farm auseinanderläuft?

Host per Skript aus denselben Bausteinen aufbauen, mit Add-RDSessionHost in die Sammlung aufnehmen, mit Set-RDSessionHost -NewConnectionAllowed No sperren, gegen den Referenzhost vergleichen und erst bei Gleichheit freigeben.

Welche Gruppenrichtlinien gehören in eine RDS-Baseline?

Loopback, Sitzungszeitlimits, Lizenzmodus und Lizenzserver, FSLogix, Härtung und eine Benutzer-GPO über Loopback. Kundenspezifisches kommt in eigene, klar benannte GPOs. Die Baseline sicherst du mit Backup-GPO und verteilst sie mit Import-GPO.

Was gehört in ein Abnahmeprotokoll für eine Remotedesktopdienste-Farm?

Rollen und Server, Sammlungen, Zertifikate mit Ablauf, Lizenzmodus und Lizenzserver, bei Größe L der Zustand der Broker-Hochverfügbarkeit, die Gleichheit der Session Hosts, die verknüpften GPOs, Erreichbarkeit, Ereignisse und ein Anmeldetest je Sammlung.

Wie setze ich ein Vier-Augen-Prinzip für RDS-Änderungen um?

Person A erzeugt das Skript und startet einen Trockenlauf, Person B prüft Skriptvorschau und Ausgabe und gibt den SHA256-Hashwert frei. Vor dem Ausführen wird der Hashwert verglichen; weicht er ab, gilt die Freigabe nicht mehr.

Haben die RemoteDesktop-Cmdlets einen Trockenlauf mit -WhatIf?

Nicht durchgängig. Verlässlicher ist ein eigener Schalter im Skript, der ohne ausdrückliche Freigabe nur ausgibt, was passieren würde.

Fazit: langweilige Farmen sind gute Farmen

Eine RDS-Farm zu standardisieren heißt nicht, alle Kunden in dieselbe Größe zu pressen, sondern jede Farm nach denselben Regeln zu bauen: gleiche Namen, drei feste Architekturen, Baseline-

GPOs, Skripte aus Bausteinen, ein Abnahmeprotokoll und eine Dokumentationsvorlage. Erweiterungen laufen über Referenzhost und Soll-Ist-Vergleich, Änderungen über Trockenlauf, Skriptvorschau und eine zweite Person, und jeder Schritt hat einen Rückweg.

Der Lohn ist unspektakulär: kürzere Tickets, planbare Angebote, Techniker, die jede Farm verstehen, und Kunden, die nicht merken, dass ihre Farm gestern erweitert wurde. Die einzige Spannung, die übrig bleibt, ist die Frage, wer beim nächsten Teamessen bezahlt. Das lässt sich leider nicht standardisieren.

WEITERLESEN · Weiter in dieser Serie

› RDS-Farm per PowerShell installieren: Schritt für Schritt
› Session Host zur bestehenden RDS-Farm hinzufügen, ohne Überraschungen
› RDS Connection Broker hochverfügbar machen
› Gruppenrichtlinien für Terminalserver: Best Practice mit Loopback
› Session Hosts verhalten sich unterschiedlich: Konfigurationsdrift erkennen
› RDS-Farm dokumentieren: was in eine gute Farmdokumentation gehört
› Wie viele Benutzer pro Terminalserver? Lasttest statt Daumenregel