
SDE

Setting up SDE VDI desktop pools

Learn how to set up on-demand desktop pools for Windows and Linux using SDE VDI with Proxmox, Active Directory, and Guacamole. This guide is for engineers deploying SDE VDI and covers everything from preparation to day-to-day operations.

AUTHOR

Paul Flanders

PUBLISHED

26 Oct 2026

DOCUMENT

InfoHub — Knowledge, news, and updates from EssingtonITS

Setting up SDE VDI desktop pools

Paul Flanders · 26 Oct 2026

Add on-demand desktop pools ("Windows 11 Desktop", "Linux Desktop") to an SDE Remote deployment: Proxmox, Active Directory, Guacamole, the sde-vdi unit, the gateway and the app, end to end.

Applies to: SDE Remote gateway 1.0.12+, SDE Remote app 1.1.2+, sde-vdi 0.5.9+, Proxmox VE 8/9 (single node or cluster) · Audience: engineers deploying SDE VDI for a site · Time: about half a day, plus building the gold template

Contents

1. [What you're setting up](#)
2. [Before you start](#)
3. [Proxmox: account and permissions](#)
4. [Active Directory](#)
5. [Guacamole](#)
6. [Gold template](#)
7. [Deploy the sde-vdi unit](#)
8. [Admin page on the portal \(nginx\)](#)
9. [Switch pools on in the gateway](#)
10. [Create the first pool](#)
11. [Give users access](#)
12. [The SDE Remote app](#)
13. [Acceptance tests](#)
14. [Day-to-day operations](#)
15. [Troubleshooting](#)

16. [Removing SDE VDI](#)

17. [Related pages](#)

1. What you're setting up

SDE VDI keeps a pool of desktops ready on Proxmox, cloned from a gold template. A user clicks the pool's tile in SDE Remote and gets a free desktop. When they sign out of Windows, that desktop is deleted and a fresh one is cloned. It's an optional add-on to an existing SDE Remote deployment, in its own GL unit.

Part	Role	Changes needed
sde-vdi unit (new)	Clones, prepares and recycles desktops. Hands them out. Admin page.	Deploy it (section 7)
Proxmox	Runs the desktops	An API token scoped to its own pools (section 3)
Active Directory	Desktops join it; users sign in with it	A desktops OU and a join account (section 4)
Guacamole	Shows each pool as a connection; decides who may use it	None on a stock install, apart from the LDAP check (section 5)
sde-gateway (broker + rdpgw)	Asks sde-vdi for a desktop and tunnels RDP to it	Version 1.0.12 or later and two environment lines (section 9)
sde-nginx	Serves the admin page at <code>/sde-vdi/</code>	One location block (section 8)
SDE Remote app	Pool tiles, the "Preparing..." wait, the Admin → Desktop pools menu entry	Version 1.1.2 or later (section 12)

How a desktop is prepared. sde-vdi makes a linked clone and boots it. It waits for Windows setup to finish, then renames the desktop and restarts it. Next it sets DNS (if configured) and the classic Netlogon secure channel, joins the domain under the final name and restarts again. It then checks the domain trust, grants Remote Desktop to the pool's groups, makes sure the desktop's name resolves to its own address, and finally waits for RDP. Then the desktop is *spare*. Expect 5-8 minutes per desktop on the LAN and longer on

an SDE Network, which is why the pool keeps spares ready.

Four scripts do most of the setup (in the repo's `scripts/vdi` and `deploy/glitch-units` folders):

Script	Run it	It does
<code>pve-bootstrap-sde-vdi.sh</code>	Once, as root on any Proxmox node	Pools, role, user, token and permissions (section 3)
<code>ad-bootstrap-sde-vdi.ps1</code>	Once, on a domain controller	OU, delegated join account, group, Group Policy, time check (section 4)
<code>template-prep.ps1 / .sh</code> + <code>pve-template.sh</code>	Per gold template	Prepares the master and releases template versions (section 6)
<code>deploy-sde-vdi.sh --wire</code>	Once, on the gateway host	Installs the unit, writes both units' settings, adds the nginx route, tests (sections 7-9)

Each script prints the settings the next one needs, and each has a `--check` (`-Check`) mode that changes nothing. Afterwards, the admin page's **Readiness** tab checks everything live.

How a user connects. The app asks the gateway. The gateway checks in Guacamole that the user may use the pool, asks sde-vdi for a desktop, and issues a short-lived gateway token for it. FreeRDP connects through the gateway with the user's own credentials. While FreeRDP runs, the app checks in every minute.

Network paths to allow

- Gateway host → Proxmox API: `8006` .

- Gateway host (sde-vdi and rdpgw) → desktops: 3389 .
- Gateway host (sde-vdi) → domain controllers: LDAPS 636 and DNS.
- Desktops → domain controllers: DNS, Kerberos, LDAP, SMB, time.
- Admins' browsers → portal: 443 .

sde-vdi's own port (8096) only needs to be reachable from the gateway and nginx; never publish it outside. Set SDE_VDI_V1_NETWORKS so the gateway API answers only the gateway ([section 14.1](#)).

2. Before you start

You need	Notes
A working SDE Remote deployment	Guacamole + <code>sde-gateway</code> + <code>sde-nginx</code> , as in the main install guide. Gateway image 1.0.12 or later (<code>sde-remote-gateway:latest</code>).
Proxmox VE 8 or 9	Root access to one node's Shell for the one-off setup. Note which storage will hold the template: linked clones need LVM-thin, ZFS, Ceph or qcow2.
Windows licensing	For live, Windows 11 Enterprise with KMS or AD-Based Activation, and Microsoft 365 Apps with Shared Computer Activation if you install Office. For dev, an evaluation ISO is fine.
Active Directory admin	To create an OU and a join account, and optionally GPOs.
Guacamole admin	To grant the pool entries to users or groups.
Values to collect	Proxmox node URL(s), the DC name and IP, the AD domain (FQDN and NetBIOS), the desktops OU path, and <code>gua_srv</code> 's password.

3. Proxmox: account and permissions

sde-vdi uses an API token that can only touch its own desktops and clone your templates. It's never root. Copy the repo's `scripts/vdi` folder to any node and run, once, as root (on a cluster it applies to every node):

```
bash pve-bootstrap-sde-vdi.sh          # add --template <vmid> for existing
templates                             # later: shows OK / FIX for each
bash pve-bootstrap-sde-vdi.sh --check  # item, changes nothing
```

It detects Proxmox 8 or 9 for the guest-agent privileges, grants every storage that can hold disks (or only `--storage x`), and ends by printing the `SDE_VDI_PVE_*` lines for [section 7](#), including the token secret, which is shown only once. `--rotate-token` replaces the token and prints a new secret.

3.1 What it does, as commands (to do it by hand)

```
pveum pool add sde-vdi --comment "SDE VDI desktops (managed by sde-vdi)"
pveum pool add sde-gold --comment "SDE VDI gold templates"

# Proxmox 9:
pveum role add SDEVDI --privs "VM.Allocate VM.Clone VM.Audit VM.PowerMgmt
VM.Config.Options VM.Config.Network VM.Config.Disk VM.Config.CDRom VM.Config.Cloudinit
VM.GuestAgent.Audit VM.GuestAgent.Unrestricted"
# Proxmox 8: the same, but with VM.Monitor instead of the two VM.GuestAgent.*
privileges.

pveum user add sde-vdi@pve --comment "SDE VDI service account"
pveum user token add sde-vdi@pve vdi --privsep 0      # copy the secret it prints
(shown once)

pveum acl modify / --users sde-vdi@pve --roles PVEAuditor
pveum acl modify /pool/sde-vdi --users sde-vdi@pve --roles SDEVDI
pveum acl modify /pool/sde-gold --users sde-vdi@pve --roles PVETemplateUser
pveum acl modify /sdn/zones/localnetwork --users sde-vdi@pve --roles PVESDNUser
# With SDE Network, sde-network grants the same role on /sdn/zones/sde (or use the
script's --zone sde)
pveum acl modify /storage/<template-storage> --users sde-vdi@pve --roles
PVEDatastoreUser # e.g. /storage/datastore01
```

Put each gold template in the `sde-gold` pool (`pveum pool modify sde-gold --vms <template-vmid>`), or allow the token to clone it directly:

```
pveum acl modify /vms/<template-vmid> --users sde-vdi@pve --roles PVETemplateUser
```

Check it The admin page's *Readiness* tab tests the token's privileges on the desktops pool, each pool's template and storage, and the network zone, and gives the exact command for anything missing.

Placement on a cluster

- **Shared storage (Ceph, NFS):** clones can run on any node.
- **Local storage:** clones run on the template's node. Tick only that node in the pool's settings, or keep an identical template, with the same name, on each node.

4. Active Directory

Skip this section only for workgroup desktops, where users sign in with local accounts on the desktop. That's suitable for dev, not live.

On a domain controller (or a machine with the AD and Group Policy tools), in an elevated PowerShell, run once:

```
powershell -ExecutionPolicy Bypass -File ad-bootstrap-sde-vdi.ps1          # -WhatIf  
to preview  
powershell -ExecutionPolicy Bypass -File ad-bootstrap-sde-vdi.ps1 -Check  # later:  
OK / FIX only
```

It creates `OU=sde` with `OU=desktops` under it, the join account `sde-vdi-join` (random password, never expires, **not** a Domain Admin) with only the delegated rights below, the group `sde-vdi-users`, and a GPO with the session time limits. It also checks the PDC's time source. It prints the `SDE_VDI_AD_*` lines for [section 7](#); the password is shown only once (`-ResetPassword` sets a new one). Options: `-ParentOU`, `-OUName`, `-JoinAccount`, `-RdpGroup`, `-DisconnectedMinutes`, `-IdleMinutes`, `-NoGpo`.

The subsections explain what it sets up, for doing it by hand or checking it.

4.1 OU for the desktops

Create an OU, for example `OU=desktops,OU=sde,DC=corp,DC=local`. sde-vdi joins every desktop there under its pool name (`w11-0FF-001` ...). Names are reused, so the computer objects are reused too.

4.2 Join account

Create a user, for example `sde-vdi-join`. Set **Password never expires**, and don't make it a Domain Admin. Delegate it the rights to create, delete and join computers in the desktops OU only. In an elevated prompt on a DC, swapping in your OU and account:

```
$ou = "OU=desktops,OU=sde,DC=corp,DC=local"
$who = "CORP\sde-vdi-join"
dsacls $ou /I:T /G "${who}:CCDC;computer"
dsacls $ou /I:S /G "${who}:CA;Reset Password;computer" "${who}:RPWP;Account
Restrictions;computer" `
"${who}:WS;Validated write to DNS host name;computer"
"${who}:WS;Validated write to service principal name;computer"
```

The same account reuses the computer objects it created, so Windows' rule against reusing someone else's computer account doesn't apply.

Deleting computer objects is part of the delegation because AD clean-up ([section 14](#)) uses this account too.

4.3 Group Policy on the desktops OU (recommended for live)

- **Session time limits.** Under *Computer Configuration* → *Administrative Templates* → *Windows Components* → *Remote Desktop Services* → *Remote Desktop Session Host* → *Session Time Limits*: disconnected sessions **30 min**; "End session when time limits are reached" **Enabled**. sde-vdi recycles a desktop once its user is signed out, so this is what clears abandoned sessions.
- **Remote Desktop rights.** Use Restricted Groups / Group Policy Preferences to add your users' group to *Remote Desktop Users*. Or let sde-vdi do it per pool ("Allow Remote Desktop for", [section 10](#)).

- **Profiles.** Desktops are wiped at sign-out. Use FSLogix or folder redirection plus home and project shares for anything users need to keep.

4.4 DNS and time

- **DNS:** desktops must resolve the domain. If DHCP hands out a DNS server that can't (for example a router), set the pool's "DNS servers for the domain" to the DCs. The gateway host must also resolve `<desktop>.<domain>`, because the pool connects by name.
- **Time:** the PDC emulator must have correct time, from external NTP. A DC clock that's hours out makes newly joined desktops lose their domain trust. On Proxmox, register a Windows DC VM with a Windows OS type, so its hardware clock is local time:

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

- **Netlogon secure channel:** sde-vdi sets `UseKerberosForSecureChannels=0` on each desktop when it joins, so Windows uses the classic Netlogon channel. Without it, Windows 11 desktops on an SDE Network can't take domain sign-ins from a Windows Server 2025 DC. Nothing to set up; `SDE_VDI_NETLOGON_KERBEROS=true` turns it off.
- **Service accounts:** the gateway's LDAP bind account and Guacamole's search-bind account must **never expire**. An expired one breaks sign-in in confusing ways.

5. Guacamole

Nothing is installed into Guacamole. sde-vdi uses its standard API, with the same service account the gateway uses (`gua_srv`, a Guacamole administrator). Check two things:

- **The LDAP user base covers your SDE users.** In `guacamole.properties` (and the unit's environment, if the image builds its config from it), `ldap-user-base-dn` must contain every SDE user, for example `OU=sde,DC=corp,DC=local`. Users moved outside it can no longer sign in. Set `ldap-group-base-dn` to the whole domain so groups are

found wherever they live.

- **Guacamole-only accounts**, such as `guacadmin`, sign in to the app because the gateway skips its AD pre-check for them (`SDE_LOCAL_ACCOUNTS`, default `guacadmin`). Add any other break-glass accounts there. These accounts can't sign in to a domain desktop, so they're refused pool desktops unless the pool has **Users sign in on the desktop** ticked.

When the first pool is saved, sde-vdi creates the connection group "**Desktop Pools (open in SDE Remote)**" and a connection per pool in it.

6. Gold template

Build it with the template guides:

- [Building a Windows 11 gold template](#)
- [Building a Linux desktop gold template](#)

The scripts in the repo's `scripts/vdi` folder automate most of that: `template-prep.ps1` / `template-prep.sh` run inside the master, and `pve-template.sh` (push, release, check) runs on the Proxmox node. The release step also grants the token clone rights.

Otherwise put it in the `sde-gold` pool or grant the token clone rights on it ([section 3.1](#)), and check it on the admin page's **Templates** tab. Guest agent, network and no-ISO should all pass, plus cloud-init for Linux.

7. Deploy the sde-vdi unit

On the gateway host, from a checkout of the `sde-remote` repo, one interactive run does sections 7, 8 and 9:

```
sudo deploy/glitch-units/deploy-sde-vdi.sh --wire --dry-run # shows what it would change
sudo deploy/glitch-units/deploy-sde-vdi.sh --wire
```

1. Installs the unit if it isn't there.
2. Asks you to paste the lines printed by the two bootstrap scripts (an empty line ends the paste), then asks for anything missing. The Guacamole URL and `gua_srv` password default to the gateway's. It can fetch and install Proxmox's CA, so the certificate is checked.
3. Generates the shared secret.
4. Writes the sde-vdi settings and the gateway's two lines, showing the change with secrets masked and keeping a timestamped backup of each file.
5. Adds the `/sde-vdi/` route to sde-nginx and tests the config (it restores the backup if the test fails).
6. Asks before restarting sde-vdi, the gateway and nginx.
7. Tests: sde-vdi answers, the gateway reaches it with the right secret, the admin page loads through the portal, and `/sde-vdi/v1/` is **refused** through the portal.

Run `deploy-sde-vdi.sh --check` any time to repeat only the tests. The rest of this section, and sections 8 and 9, describe the same changes for doing them by hand.

By hand:

```
sudo deploy/glitch-units/deploy-sde-vdi.sh      # pulls sde-vdi:latest and installs the unit
```

The first run installs `/var/glitch-units/sde-vdi/glitch/config/eviroment` and stops, asking for the secrets. Edit it:

Setting	Value
SDE_VDI_PVE_URL	Proxmox API, e.g. <code>https://pve01.corp.local:8006</code> . On a cluster, list several, comma-separated.
SDE_VDI_PVE_TOKEN_ID / _SECRET	<code>sde-vdi@pve!vdi</code> and the secret from section 3
SDE_VDI_PVE_INSECURE or _CA_FILE	Proxmox's own certificate: trust its CA file (<code>/etc/pve/pve-root-ca.pem</code>), or <code>INSECURE=true</code> on dev
SDE_VDI_GUAC_URL	Same as the gateway's <code>SDE_GUAC_URL</code>
SDE_VDI_GUAC_ADMIN_USER / _PASS	<code>gua_srv</code> and its password
SDE_VDI_SHARED_SECRET	<code>openssl rand -hex 32</code> . The gateway gets the same value (section 9).
SDE_VDI_AD_DOMAIN / _USER / _PASS	The AD domain (FQDN), the join account and its password. Leave unset for workgroup desktops.
SDE_VDI_AD_OU	The desktops OU. The pool editor suggests it for new pools.
SDE_VDI_AD_LDAP_*	Optional. sde-vdi reads AD over LDAPS as the join account (Readiness, pool defaults, clean-up). It finds a DC through DNS and trusts the CA the DC presents; set <code>_HOST</code> , <code>_SERVERNAME</code> or <code>_CA_FILE</code> only if that fails.

SDE_VDI_V1_NETWORKS	Recommended. Addresses allowed to call the gateway API, e.g. <code>172.16.0.0/12</code> (docker networks). See 14.1 .
SDE_VDI_ADMIN_NETWORKS	Optional. Who may open the admin page (default: private ranges).
SDE_VDI_PAUSED	<code>true</code> for the first start (read-only). Remove it afterwards.
SDE_VDI_MEM_CHECK / _NODE_MEM_RESERVE_MIB	Free-memory guard per node (default on, 2 GiB headroom). <code>false</code> only on tight dev kit.

Every setting, with its default, is in [the settings reference](#).

```
sudo chmod 600 /var/glitch-units/sde-vdi/glitch/config/enviroment
sudo deploy/glitch-units/deploy-sde-vdi.sh      # starts it
docker logs sde-vdi 2>&1 | head                 # expect: "Proxmox OK", "PAUSED",
"listening on :8096"
```

Pool definitions and leases live in `/var/glitch-units/sde-vdi/data/state`. Keep it, and back it up with the host. sde-vdi also keeps a daily copy of its state in `data/state/backups` (the last 14 days). When the paused start looks right, comment out `SDE_VDI_PAUSED` and restart: `gl -d sde-vdi && gl -s sde-vdi` (or `php /var/glitch-cli/gl.php ...`).

8. Admin page on the portal (nginx)

Add this to the portal's `server { listen 443 ... }` block, next to `/sde-broker/`. A ready-made copy is in `deploy/nginx/sde-vdi.conf`. Add an `upstream sde_vdi { server <host>:8096; }` line if your config uses upstreams.

```
location = /sde-vdi { return 301 /sde-vdi/; }
location ^~ /sde-vdi/v1/ { return 404; }      # the gateway<->sde-vdi API is never
exposed
location /sde-vdi/ {
```

```

proxy_pass      http://<host>:8096/;          # trailing slash strips the prefix
proxy_set_header Host $host;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_read_timeout 60s;
}

```

```

docker exec sde-nginx nginx -t && docker exec sde-nginx nginx -s reload    # reload
doesn't drop connections

```

Open <https://<portal>/sde-vdi/> and sign in as a Guacamole administrator. It only opens from inside the network.

9. Switch pools on in the gateway

Add two lines to `/var/glitch-units/sde-gateway/glitch/config/enviroment` , then restart the gateway. The restart takes about 10 seconds and **drops every native RDP session** going through the gateway, pool desktops included. Users reconnect from SDE Remote; a pool user gets the same desktop and session back. Do it at a quiet time.

```

SDE_VDI_URL=http://<host>:8096
SDE_VDI_SECRET=<the same value as SDE_VDI_SHARED_SECRET>

```

```

gl -d sde-gateway && gl -s sde-gateway
docker logs sde-gateway 2>&1 | grep "desktop pools"    # expect: "desktop pools: sde-
vdi 0.5.x ... (api v1)"

```

Check it Open the admin page's **Readiness** tab. Every line should be **OK**, or **Check** with a reason you accept. Each problem comes with its fix.

10. Create the first pool

On the admin page, go to **Pools → New pool**. Start with spare 1, maximum 2, and raise them once it's proven.

Field	Windows + AD	Linux
Display name	What users see, e.g. <i>Windows 11 Desktop</i>	<i>Linux Desktop</i>
Desktop OS	Windows	Linux. This sets RDP to TLS and ticks "Users sign in on the desktop"
Gold template	Pick from the list; the checks show underneath	Same
Desktop names	<code>w11-OFF-{nnn}</code> (15 characters max)	<code>lnx-dsk-{nnn}</code>
Spare desktops / Maximum desktops	Spares are ready and waiting. The maximum caps spare + in use + being prepared.	Same
Network	Existing LAN (the template's network card) , or a private network from SDE Network	Same
Grant to Guacamole groups	Who gets the pool, e.g. <code>sde-vdi-users</code> (section 11)	Same
Spare desktops by time of day	Optional rules, e.g. 6 spare Mon-Fri 07:50-09:30 for the morning rush. The highest matching rule wins; otherwise the spare count.	Same

Advanced → Nodes	The template's node, on local storage	Same
Advanced → Join domain: OU	The desktops OU (section 4.1)	Leave empty
Advanced → DNS servers for the domain	The DCs' IPs, if DHCP's DNS can't resolve the domain	Empty
Advanced → DNS suffix	The AD domain, e.g. corp.local (connect by name)	Empty (connect by IP)
Advanced → Allow Remote Desktop for	e.g. CORP\Domain Users or CORP\sde-vdi-users , unless a GPO does it	Empty
Advanced → RDP security / domain	NLA, domain = NetBIOS name (e.g. CORP)	TLS
Advanced → Remote Desktop → Users sign in on the desktop (dev / local accounts)	Off (SSO with their AD account)	On (local accounts)
Advanced → Disconnected session kept for / Maximum session length	30 min / 12 h	Same

For a new Windows pool, the editor fills the OU, DNS servers, DNS suffix and RDP domain from Active Directory, and suggests "Allow Remote Desktop for" from the Guacamole groups (for example CORP\sde-vdi-users). Check them under Advanced before saving.

Save it. Watch the **Desktops** tab: the first desktop goes from Cloning through Windows setup, Renaming, Joining the domain, Checking domain trust, Granting Remote Desktop access and Waiting for Remote Desktop, to **Spare**. If it stops on a step, see [Troubleshooting](#).

11. Give users access

A user may use a pool when they can see its connection in Guacamole. The easy way is the pool's **Grant to Guacamole groups** field: put the AD group's name there (for example `sde-vdi-users`). sde-vdi creates the Guacamole user group of the same name if it's missing, and grants it the pool. Guacamole matches LDAP group membership by name, so adding someone to the AD group gives them the pool. Taking a group off the field removes only that grant.

Or grant "**<pool name>**", in "Desktop Pools (open in SDE Remote)", by hand in Guacamole:

- **To a group:** create a Guacamole user group with the *same name* as an AD group and grant the pool to it.
- **Per user:** Settings → Users → the user (add them with the same username if they aren't listed) → tick the pool.

Users also need Remote Desktop rights on the desktops ([4.3](#) or the pool setting).

12. The SDE Remote app

Use the current app (**1.1.2**, Windows or Linux):

- Each pool shows as a **DESKTOP POOL** tile with "N ready" and a Connect button. With no spare, Connect shows "Preparing your desktop..." and keeps trying for up to 15 minutes.
- Closing FreeRDP keeps the desktop for the pool's **Disconnected session kept for** (30 minutes by default). Reconnecting returns the user to the same session.

- Guacamole administrators get **Admin → Desktop pools**. It opens the admin page, already signed in.
- When a desktop refuses the sign-in, the app says "Windows refused the sign-in" and names the FreeRDP log.

13. Acceptance tests

- Readiness: no **Fix** lines. **Settings → Test connections**: Proxmox, Guacamole and Guacamole entries all OK.
- The first desktop reaches Spare without errors. Note the time it takes.
- The desktop is in the OU in AD, and `<name>.<domain>` resolves from the gateway host.
- A user with the grant sees the tile with "1 ready". Connect gives a desktop signed in as them, with no password prompt.
- The next spare starts building straight away.
- Closing FreeRDP and reconnecting within 30 minutes gives the same desktop.
- A Guacamole-only account (for example `guacadmin`) is refused the pool with a message, and no desktop is taken.
- Signing out of Windows deletes the desktop within a couple of minutes, and the pool refills.
- Two users at once get different desktops. With the pool full, a third gets "No desktops free".
- A non-admin can't open the admin page. Nothing at `/sde-vdi/v1/` answers through the portal (`deploy-sde-vdi.sh --check` tests it).
- The usage report (`/report`) shows the session under "Desktop pools" with the pool and desktop name.

14. Day-to-day operations

Task	How
Update the image	Build a new template version (template guide, section 11), then choose it as the pool's Gold template . Spares are replaced at once; desktops in use are replaced as users sign out.
More or fewer spares	Pool → Spare desktops / Maximum desktops . Changes apply within seconds; extra spares are removed after 10 minutes.
A stuck or broken desktop	Desktops → Delete now . The pool replaces it.
Who's on what, what happened	The Desktops and Activity tabs. Full history: <code>docker logs sde-vdi (AUDIT vdi- ... lines)</code> and the gateway's <code>AUDIT authorize ... vdi_...</code> .
Upgrade sde-vdi	<code>gl -d sde-vdi && gl -s sde-vdi pulls :latest</code> . Leases and pools survive (state volume).
Pause all changes	<code>SDE_VDI_PAUSED=true</code> and restart: nothing is cloned or deleted, and handing out spares still works.

Busy times	Pool → Spare desktops by time of day. Start a rule about 10 minutes before the rush, because a desktop takes that long to prepare.
Clean up AD	Readiness → Active Directory clean-up → Scan. It lists computer objects whose names won't come back (the pool shrank, its name pattern or OU changed, or it was deleted). Tick and delete; their DNS records go too. Recycled desktops' objects are kept on purpose, because the next desktop of that name reuses them.
Restore the state	<code>gl -d sde-vdi</code> , copy a file from <code>data/state/backups</code> over <code>data/state/state.json</code> , <code>gl -s sde-vdi</code> . Desktops created since that copy show as quarantined; delete them.
Rotate the Proxmox token	<code>pve-bootstrap-sde-vdi.sh --rotate-token</code> on a node, put the new secret in the unit, restart it.

14.1 Hardening for live

- **Gateway API only from the gateway:** `SDE_VDI_V1_NETWORKS=172.16.0.0/12` (the docker networks; Readiness shows the address the gateway calls from). Through nginx, `/v1` is always refused.
- **Join account delegated, never a Domain Admin** (`ad-bootstrap-sde-vdi.ps1` sets it up that way).

- **Proxmox certificate verified:** `SDE_VDI_PVE_CA_FILE` instead of `SDE_VDI_PVE_INSECURE` .
- **Secrets file:** `chmod 600` on both units' `enviroment` files. Rotate the Proxmox token if its secret was ever shared.
- **Admin page:** `SDE_VDI_ADMIN_NETWORKS` limited to the admins' subnets.
- **Backups:** include `/var/glitch-units/sde-vdi/data/state` (with its daily copies) in the host backup.

15. Troubleshooting

Symptom	Cause and fix
Desktop stuck at "Windows setup"	The template's unattend file doesn't skip OOBE. Add <code>SkipMachineOOBE / SkipUserOOBE</code> and release a new version (Windows guide § 8).
"domain join ... could not be contacted"	The desktop can't resolve the domain. Set the pool's DNS servers to the DCs.
Desktop stays at <code>waiting-dns</code> , or its name resolves to an old address	An older computer object owns the DNS record, so the DC refuses the desktop's update (event 8018). sde-vdi deletes the record and re-registers it every 2 minutes; if the join account may not delete it, delete the A record in DNS Manager.
Stuck at "Checking domain trust", or users rejected with 0xC000018D	The DC's time or replication is broken. Fix the PDC's time; sde-vdi repairs the trust automatically after 3 minutes. If only pools on an SDE Network are refused with 0xC000018D, see Troubleshooting (Netlogon secure channel).
"Windows refused the sign-in", or FreeRDP closes straight after Connect	Wrong password, an account Windows doesn't know, a broken trust, or no Remote Desktop rights on the desktop (set the pool's groups or a GPO). Event 4625 on the desktop gives the reason. On Linux, check RDP security is TLS, not NLA.

"... is an SDE-only account, so it can't sign in to the desktops in ..."	A Guacamole-only account (<code>SDE_LOCAL_ACCOUNTS</code>) tried a pool that passes the user's sign-in through. Sign in with a domain account.
An AD user can't sign in to the app, but a Guacamole-only user can	The user is outside Guacamole's <code>ldap-user-base-dn</code> , or an LDAP service account has expired (section 5).
Readiness: "DC clock" fails	The DC is more than 4 minutes out. Kerberos fails at 5. Fix the PDC's NTP source; on Proxmox the DC VM needs a Windows OS type.
Readiness: "Sign in to AD" fails	Wrong join password, or no LDAPS from the gateway host to a DC (port 636). Set <code>SDE_VDI_AD_LDAP_HOST</code> to a DC if the domain name doesn't resolve on the host.
A pool's Guacamole entry shows "grant: ..."	The service account can't create user groups or grant permissions. Give <code>gua_srv</code> "Create new user groups" or "Administer system".
"No desktops free"	The pool is at its maximum. Raise it if the host has memory.
Pool shows "Can't add desktops"	Read the message: template not found, no node with room, or a token permission. The Templates and Settings tabs help.
Clones fail or the host struggles	Too many clones at once: lower <code>SDE_VDI_MAX_PARALLEL_CLONES</code> (default 2) or add memory or I/O.

More problems, with the commands to look inside a desktop: [Troubleshooting SDE VDI desktop pools](#).

16. Removing SDE VDI

1. In the admin page, switch each pool off, wait for its desktops to drain, then delete the pool. Its Guacamole entry is removed too.
2. Remove `SDE_VDI_URL` / `SDE_VDI_SECRET` from the gateway and restart it. Pool tiles disappear from the app.
3. `gl -d sde-vdi` , remove the nginx block, and reload nginx.
4. Optionally remove the Proxmox token, role and pools, and the AD join account and OU.

17. Related pages

- [SDE VDI settings reference](#): every setting, desktop state and preparation step
- [Troubleshooting SDE VDI desktop pools](#): symptoms, causes and fixes
- [Building a Windows 11 gold template](#): building a Windows 11 gold template
- [Building a Linux desktop gold template](#): building a Linux desktop gold template