ChatGPT

DNS Failures in Libguestfs virt-customize – Root Cause and Solutions

Background and Recent Changes

What broke: A recent upgrade (Proxmox VE 9/Debian "Trixie") introduced changes in how the libguestfs appliance obtains networking. Historically, the libguestfs appliance (used by `virt-customize`) would use a DHCP client inside the appliance VM to configure networking and populate `/etc/resolv.conf` for DNS. In older libguestfs versions, the appliance included the **ISC DHCP client** (`dhclient`) on Debian-based hosts [1]. However, **Debian replaced** `dhclient` **with** `dhcpcd-base` as the default DHCP client, meaning a **DHCP client was no longer installed by default on the host or pulled into the appliance** [1]. As a result, when `virt-customize` enabled networking (`LIBGUESTFS_NETWORK=1`), the appliance couldn't run *any* DHCP client (e.g. `/init: line 140: dhcpcd: command not found` appeared in debug logs [2]). This left the appliance with **no IP address, no default route, and no DNS resolver configuration**. In short, **nothing changed in your scripts or cloud images – it was the libguestfs environment that lost DNS due to a missing DHCP client**.

Effect on `/etc/resolv.conf`**:** Because the appliance never acquired DNS info, `/etc/resolv.conf` **inside the libguestfs appliance was never created** (the DHCP step that normally writes it never ran) [3]. Libguestfs does try to copy the host's `resolv.conf` into the chroot when running commands, but that mechanism fails if the appliance itself has no resolver file to copy (or if the host's resolv.conf wasn't accessible) [4] [3]. In your case, the libguestfs logs showed the attempt to copy resolv.conf failing, causing package managers (dnf/apt/zypper running in the chroot) to see no DNS configuration and error out with "Could not resolve host" messages. This is consistent with known bug reports where `virt-customize` package installations suddenly fail due to name resolution issues when the libguestfs backend lacks a DHCP client [2] [5]. Upstream developers have noted that handling of `/etc/resolv.conf` in the libguestfs chroot has been a "**continuing source of problems**" over the years [6], and indeed a recent upstream patch was introduced to address the DHCP client situation on Debian hosts [1].

Version-specific changes: The package versions you listed (libguestfs 1.52.3 vs 1.54.1) align with this transition. **Debian 12/13 removed** `isc-dhcp-client` **from default installs, expecting** `dhcpcd` **instead**, but libguestfs 1.54 initially didn't include `dhcpcd` in its appliance. This mismatch is precisely what caused the breakage [1]. In fact, the Debian libguestfs maintainer had to update the packaging: in May 2025, libguestfs 1.54.1-2 switched to depend on `dhcpcd-base` instead of `isc-dhcp-client` [7], and a later update (1.56 in testing) adds `isc-dhcp-client` back into the appliance packagelist as a fallback [8]. These changes are aimed at ensuring **at least one DHCP client is present** in the appliance. Until that fix propagates, a Proxmox/Debian 13 host upgraded to libguestfs 1.54 will have this DNS issue out-of-the-box.

Understanding the Odd `/etc` vs `/usr/etc` Behavior

Your observation that writes to `/etc/resolv.conf` "disappeared" while a file showed up under `/usr/etc/resolv.conf` is explained by how modern distributions handle resolver configuration:

- **Symlink in the Guest OS:** Several distros (notably systemd-based ones like recent **openSUSE** or others) have moved the default location of resolver config to `/usr/etc/resolv.conf`, with `/etc/resolv.conf` being a symbolic link to that location [9]. This is part of a broader trend (e.g. openSUSE's "UsrEtc" concept or systemd's stateless systems) where vendor-provided default configs live in `/usr/etc` and `/etc` is for overrides. In your case, the cloud images for AlmaLinux 9 or openSUSE likely came with such a symlink. We see confirmation of this in the cloud image: `/etc/resolv.conf` **was a dangling symlink** (pointing to `/usr/etc/resolv.conf` or perhaps to a non-existent stub file managed at runtime). That's why `ls /etc/resolv.conf` returned "No such file or directory" – the symlink's target wasn't present until you created it.

- **Why your writes went to `/usr/etc/`:** When you attempted to copy in a resolv.conf via `virt-customize --upload` or `--write`, the libguestfs API likely **followed the symlink**. Instead of replacing the link, it ended up creating the file at the symlink target. Thus, the content landed in `/usr/etc/resolv.conf` (the target) while the link in `/etc/` either remained or was restored later. As a result, **the resolver file existed in `/usr/etc` but was still not seen by software** – because the symlink in `/etc` might have been broken or removed during the process. Note that glibc and typical package managers **only look at** `/etc/resolv.conf` (following symlinks if present). They won't automatically check `/usr/etc`. In your scenario, it appears the `/etc/resolv.conf` link was removed (perhaps by libguestfs' own backup/restore logic), so although you populated `/usr/etc/resolv.conf`, there was no link for the resolver libraries to follow, and DNS still failed. Essentially, **the libguestfs customization steps clobbered the symlink**: it backed up the original link, tried to put a file in place, then restored the link (overwriting the file) after your commands ran. This explains why your manual writes "vanished." The moment `virt-customize` finished an action (or on appliance shutdown), it likely restored the guest's original `/etc/resolv.conf` state (the symlink, still pointing to `/usr/etc/resolv.conf`) [10], leaving your newly written file accessible only at `/usr/etc/resolv.conf` (which the symlink pointed to all along). If that symlink was broken to begin with (e.g. pointed to a stub in `/run` that doesn't exist offline), the file in `/usr/etc` won't be used by anything at runtime.

- **Systemd-resolved/NetworkManager angle:** Many modern OSes use *systemd-resolved* or *NetworkManager* to manage DNS, which can complicate offline edits. For example, Ubuntu or others might have `/etc/resolv.conf` as a link to `/run/systemd/resolve/stub-resolv.conf`. In an offline libguestfs environment, `/run/...` doesn't exist, so that symlink appears broken. Similar issues arise on SUSE with `/usr/etc`. No systemd services are running in the chroot to manage these links, so any automation that doesn't account for them will have trouble. In short, there isn't an active systemd or NM "interfering" during `virt-customize` (since the guest OS isn't actually booted), but the **static presence of those symlinks** can make it *seem* like file operations on `/etc/resolv.conf` are being magically undone. It's really an artifact of the symlink and libguestfs's handling of that file, not a live process fighting you.

Root Cause Summary

In summary, the breakage is due to *two interacting factors:*

1. **Libguestfs no longer automatically provides DNS** in the appliance because the required DHCP client wasn't present. The appliance's `/etc/resolv.conf` never got populated at runtime, leading to failures during package installation (name resolution errors) [5]. This is a regression introduced by changes in libguestfs packaging/behavior in recent versions (post-Proxmox 8.x). Upstream has acknowledged this and provided patches (e.g., adding `dhcpcd-base` support on Debian) [1]. As one forum user noted, "the dhcpcd is used to automatically configure the network inside the virt-customize image" and without it, the VM has no network [11].

2. **Resolver config on the guest images is symlinked or managed**, which confused manual fixes. The cloud images have `/etc/resolv.conf` as a symlink, so injecting a file needed special handling. Libguestfs's own mechanism of backing up and restoring `/etc/resolv.conf` around each chrooted command compounded the issue – it effectively **undid your manual creation of the file** in many cases. This made it look like writes "silently failed," when in reality they were being written to an unexpected location or overwritten when the symlink was restored [10].

Solution Approaches

Fortunately, there are several approaches to resolve this, **ranked from most maintainable to more involved**:

1. Restore the Libguestfs Appliance DNS Functionality (Recommended)

Install a DHCP client on the Proxmox host so that the libguestfs appliance can configure networking as intended. The simplest and safest choice is to install `dhcpcd-base` (on Debian/Proxmox) rather than the full `dhcpcd` service [12]. The `dhcpcd-base` package provides the client binary without automatically running a persistent service on host interfaces. By having this present, the libguestfs appliance will detect that no `dhclient` is available but `dhcpcd` is, and will successfully run `dhcpcd eth0` during appliance init. This will give the appliance an IP (usually on the default libguestfs SLIRP network) and populate `/etc/resolv.conf` inside the appliance VM with a working nameserver (typically a pseudo DNS like 10.0.2.3 or 169.254.2.3 that forwards to the host's resolver) [13].

After doing `apt install dhcpcd-base` on the host, users in the Proxmox forum confirmed *"virt-customize working again"* for package installs [5]. This fix is **distro-agnostic for the guest images** – it addresses the issue at the libguestfs level, so all cloud images (Alma, Ubuntu, Debian, openSUSE, etc.) will once again magically "just work" as they did before. In other words, this makes the need for manual resolv.conf handling inside `virt-customize` unnecessary. The DNS inside the chroot will be properly configured by the appliance just as it was in the past.

Why this is maintainable: It requires a one-time installation on the host. It aligns with upstream's direction – future libguestfs versions are likely to ensure either `dhclient` or `dhcpcd` is present [1] [8], so you won't be fighting against the grain. By using the `-base` variant, we avoid running unwanted services on the host (you should still ensure the `dhcpcd` service is disabled, just in case). This solution was endorsed

by multiple Proxmox users and does not rely on hacks or manual DNS injection [14] [15]. Instead, it restores the built-in automation.

Evidence: After installing a DHCP client, the libguestfs init logs will show it being invoked and succeeding (e.g. `dhcpcd-10.1.0 starting` in debug output [16]), and `virt-customize --install ...` will proceed without DNS errors. This aligns with upstream's recommended fix and the open upstream bug report confirms this step [15].

2. Use Libguestfs Environment Variables (Alternate Approach)

Libguestfs provides an environment variable `LIBGUESTFS_RESOLV_CONF` that can point to a resolv.conf file on the host. In theory, setting this to your host's `/etc/resolv.conf` (which you verified is a normal file with `1.1.1.1` as nameserver) should inject that into the appliance's `/etc/resolv.conf` for all operations, bypassing the need for DHCP. In practice, this method has been hit-or-miss and may depend on libguestfs version. You mentioned attempts to use it did not succeed. It's possible that in the version you have, the environment variable isn't honored due to the same root cause (the code path that would copy it wasn't reached because the network wasn't up). Historically, libguestfs would copy the host's resolv.conf automatically during chroot commands [17] [4], but that assumes the appliance had networking enabled to begin with.

If you want to try this route (perhaps after updating libguestfs), you could do:

```
export LIBGUESTFS_RESOLV_CONF=/etc/resolv.conf
virt-customize --install ...
```

This *might* ensure that even without DHCP, the specified file's contents appear in the guest's `/etc/resolv.conf`. However, be aware that if the guest image's `/etc/resolv.conf` is a symlink, the same symlink issue could apply. You'd essentially be forcing the appliance to use the host's DNS server (1.1.1.1 in your case) directly. This is somewhat less robust than letting the appliance get its own DNS (for example, if the host's resolv.conf has multiple search domains or special servers, it's unclear if all are copied). Still, it's an option to know about.

Maintainability: Medium. This is a libguestfs feature, so it's not a hack per se, but it's not commonly needed if the appliance networking works. It adds an external dependency (that env var must be set in your script environment). If someone else runs the script without that env, it falls back to failing. Given that upstream is actively fixing the root issue, relying on `LIBGUESTFS_RESOLV_CONF` might be more of a short-term workaround than a long-term solution.

3. Manual DNS Injection in the Guest Image (Not Ideal)

You already tried many variants of this, and as you discovered, it's fraught with pitfalls. For completeness, the idea here would be to **ensure** `/etc/resolv.conf` **inside the guest image has a valid nameserver**

before the package manager runs. If we ignore libguestfs's internal handling, one could do something like:

- Use `virt-customize --edit` or `--copy-in` to place a resolv.conf file in the image.
- Remove any problematic symlink and replace it with a regular file.

For example, one could add steps to the script: `--run-command "rm -f /etc/resolv.conf && echo 'nameserver 1.1.1.1' > /etc/resolv.conf"` as a first step, then a second `--install ...` step. But because **virt-customize runs all commands in one single appliance session**, libguestfs will likely do its backup/restore around *each* `--run-command` and around the `--install`. This means your first step might be undone by the time the second runs (as you saw). There are complex ways around this (like combining multiple shell actions in one `--run-command` so that the file isn't restored until after install, or using guestfish offline), but these are essentially **distro-specific hacks**. You would need to handle symlinks conditionally: e.g. detect if `/etc/resolv.conf` is a symlink in the guest (using `test -L`) and then decide whether to write to `/usr/etc/` or replace the symlink. This is exactly the kind of brittle solution you want to avoid.

Given that the goal is a maintainable, generic solution, I would **not recommend** this path. It was only necessary because the appliance's own DNS setup broke. If you fix that, you don't need to manually tamper with each distro's resolver config. In essence, approach #1 (fixing the appliance's DHCP/DNS) obviates this entirely.

4. Use an Alternate Method: Chroot or Secondary VM

If, for some reason, you cannot modify the host environment (e.g., organizational restrictions on installing packages on the Proxmox host), another approach is to **avoid libguestfs networking altogether**. Instead of using `virt-customize` to install packages, you can:

- Attach the cloud image to the host (using `qemu-nbd` or similar) and mount it, then perform a traditional `chroot` into the mounted image to run package installation.
- Or, spin up a temporary lightweight VM (or container) that has network access and mount the image there to run the customization.

For example, Proxmox forum users have suggested using the **NBD + chroot method**: load the QCOW2 image via network block device, mount its filesystems, bind-mount `/proc`, `/dev`, `/sys`, and copy in the host's `/etc/resolv.conf` to the chroot's `/etc` (or bind-mount it). Then run the package manager inside the chroot. This way, you're effectively using the host's network stack directly, so DNS will work as on the host. After installing packages, unmount and disconnect NBD. This approach bypasses libguestfs entirely for package installation.

Similarly, one user opted to **create a small VM running Debian 13 with libguestfs** (with proper DHCP client installed) and run all `virt-customize` operations inside that VM, keeping the Proxmox host untouched [18]. That is a bit heavy, but it isolates the environment.

The downsides of these approaches are complexity and performance. `virt-customize` is convenient because it automates a lot of the above (and is usually faster by not fully booting a system). Using chroot or

another VM might be slower or require more scripting. However, it *will* be reliable and not subject to libguestfs quirks, since you're essentially mimicking how you'd configure the VM if it were running.

Maintainability: Moderate. This is a proven sysadmin approach (nothing magical), but it is more work and requires careful scripting (especially mounting images with potentially multiple partitions, LVM, etc., which `virt-customize`/inspect handles automatically). If you only hit this DNS issue and no others, fixing libguestfs is the cleaner solution. But if libguestfs on Proxmox continues to be an outlier (since it's not officially supported by Proxmox), the chroot method is a stable fallback. It avoids dependency issues (other than requiring qemu-nbd and some privileges on the host).

5. Monitor Upstream and Update

Keep an eye on upstream fixes for this issue. As referenced, **upstream Issue #211** is tracking the DNS/DHCP problem and patches have been merged to address it [15]. When a libguestfs update is released for your distribution (Debian/Proxmox), updating to that version should also resolve the immediate problem. The patches will ensure the appliance includes whichever DHCP client is needed (or possibly use an "internal" DHCP client if provided). For instance, a future libguestfs might incorporate an internal simplified DHCP or automatically use the host's `/etc/resolv.conf` as a fallback when network configuration fails – any such improvements would make virt-customize more robust.

That said, relying on an update alone doesn't help your templates today, so the above solutions bridge the gap.

6. Long-term Considerations

- **Automation Testing:** Now that you've encountered this, consider adding a test step in your template build pipeline: e.g., run `libguestfs-test-tool` or a simple `virt-customize --run-command "ping -c1 google.com"` on a known image to quickly catch whether networking is working. The `libguestfs-test-tool` output was a clue (showing no `/etc/resolv.conf`) [19]; incorporating such checks can alert you early if a future update regresses something similar.

- **Resolv.conf symlinks:** They will become more common as distros adopt systemd-resolved by default (for example, Fedora and Ubuntu are leaning that way, and SUSE had the `/usr/etc` approach for Tumbleweed as noted [9]). Libguestfs's current method (backup and copy) doesn't gracefully handle symlinks. In the 2015 discussion, developers noted edge cases with "dangling symlink" resolv.conf and even suggested overlayfs as a possible solution [20]. While you shouldn't need to implement anything manually if the appliance has proper DNS, just be aware that if you ever *do* need to modify `/etc/resolv.conf` in an offline image, you may need to account for the possibility of it being a symlink. The safest approach is to **write to the symlink's target** (e.g., if you detect `/etc/resolv.conf -> /usr/etc/resolv.conf`, then write the file in `/usr/etc/`). But again, this is mostly for niche cases or if doing manual chroot work. Your cloud-init images typically regenerate resolv.conf on first boot anyway via DHCP or network config, so leaving the image's resolv.conf as is is usually fine.

- **Host Firewall/Networking:** You already checked that ip_forward, nftables, etc., on the host were not interfering. That's good. The libguestfs appliance uses user-mode networking (SLIRP), which shouldn't be affected by host firewall in most cases, but if you ever tighten host egress rules,

remember that the appliance's outgoing connections originate from the host (usually on a high ephemeral port to the destination's port 80/443, appearing like a user-space process connection). Ensure DNS (UDP/53) and HTTP/HTTPS are allowed out from the host if such policies exist.

- **Libguestfs on Proxmox:** Note that Proxmox doesn't ship `virt-customize` by default for the very reason that it's not part of their tested stack. As one forum moderator noted, when you install `libguestfs-tools` on PVE, it may conflict with some things (e.g., earlier it required removing `fuse`, though you indicated that's resolved in your case) [21]. Keep in mind that future PVE upgrades might not prioritize libguestfs compatibility. The **separate VM method** (#4) is a way some have sidestepped that, by not depending on the host's environment. As long as you're comfortable maintaining the package on the host (and it sounds like you are, since you've used it for years), it's fine – just remain vigilant on major upgrades.

Supporting Evidence and References

- Proxmox forum report of **virt-customize losing internet access after upgrade** – confirms only loopback in appliance, eth0 down with link-local, and no DNS. Installing a DHCP client on the host restored functionality [2] [5].
- Upstream **libguestfs patch** discussion – Debian's shift from `dhclient` to `dhcpcd` broke libguestfs networking (no DHCP by default). Patch adds support for `dhcpcd-base` in the appliance [1]. Debian changelog further shows adjustments to ensure a DHCP client is included [7] [8].
- OpenSUSE forum note on `/etc/resolv.conf` **moving to** `/usr/etc/resolv.conf` – explains the symlink you observed and why a file might end up under `/usr/etc` [9]. This is one example of modern distro practices that affected your workflow.
- Libguestfs mailing list (2015) acknowledging **persistent problems with resolv.conf handling** and describing how DHCP (dhclient/dhcpcd) is supposed to populate it in the appliance [3]. This provides context that even back then, missing resolv.conf was a known issue if the DHCP client didn't run.
- Proxmox forum solution suggesting `dhcpcd-base` as a minimal fix (does not auto-start on host) and referencing an upstream issue #211 for tracking [14] [22]. This aligns with our recommendation to install a DHCP client on the host for the appliance to use.

By implementing the recommended solution (#1), you should achieve the *"it just works"* behavior again across all your cloud-init images, with minimal maintenance overhead. This aligns with upstream fixes and avoids brittle per-distro workarounds. Always test after any libguestfs update, but with a working DHCP client in place, your template builds should remain robust going forward.

Sources:

- Libguestfs developer discussion – DHCP client requirement for DNS [1] [3]
- Proxmox user reports & fixes – installing `dhcpcd` to restore virt-customize networking [2] [5] [14]
- openSUSE community note – resolv.conf moved to `/usr/etc/` (symlink in `/etc`) [9]
- Libguestfs Issue Tracker – upstream bug #211 on virt-customize DNS, confirming the problem and patch availability [22]

---

1 [Libguestfs] [PATCH 1/2] appliance: add dhcpcd support on Debian - Libguestfs - Libguestfs List Archives
https://lists.libguestfs.org/archives/list/guestfs@lists.libguestfs.org/message/BBBHBGZY3UYVAUR77ECGAVV7QMWUZIO7/

2 5 11 12 14 15 16 18 21 22 Proxmox 9 upgrade: Virt-customize no longer has internet access | Proxmox Support Forum
https://forum.proxmox.com/threads/proxmox-9-upgrade-virt-customize-no-longer-has-internet-access.169355/

3 4 6 17 19 20 Name resolution not working inside virt-customize - Libguestfs - Libguestfs List Archives
https://lists.libguestfs.org/archives/list/guestfs@lists.libguestfs.org/thread/SO4IBEDH3VGORLWNZ6FTGP7QNXRPCLBZ/

7 8 tracker.debian.org
https://tracker.debian.org/media/packages/libg/libguestfs/changelog-11.56.2-4

9 Broken DNS - #28 by larryr - Network/Internet - openSUSE Forums
https://forums.opensuse.org/t/broken-dns/143311/28

10 13 fail to install packages into image via virt-customize? / Newbie Corner / Arch Linux Forums
https://bbs.archlinux.org/viewtopic.php?id=256966