Self-Hosted Docker Apps ARM64 vs AMD64 Compatibility Matrix

TechRounder PDF Edition

Live article:
https://www.techrounder.com/docker-container/self-hosted-docker-apps-arm64-vs-amd64-compatibility-matrix/

By Vipin PG | Published April 7, 2026 | Updated April 7, 2026 | Format: Comparison | 6 min read

Bottom line

Selecting between ARM64 and AMD64 for self-hosted Docker apps requires verifying specific image tags and hardware dependencies rather than assuming universal compatibility. While most modern services now offer robust multi-arch support, AMD64 remains the safer choice for hardware-intensive tasks like media transcoding, whereas ARM64 is the efficient default for lightweight utility and automation workloads.

Key points

- Verify Tags, Not Marketing: Compatibility depends on the specific Docker manifest list; always check if the image tag explicitly publishes both ‘linux/amd64‘ and ‘linux/arm64‘ .
- ARM64 for Utility: Services like Pi-hole, Vaultwarden, Traefik, and Gitea are excellent candidates for ARM64 due to low power consumption and mature multi-arch support.
- AMD64 for Media and AI: Applications involving hardware acceleration (like Jellyfin via Intel Quick Sync) or specific CPU instructions (like Immich’s machine learning requirements) often still favor x86 hardware.
- Watch for Hidden Dependencies: Deployments often fail on ARM not because of the main app, but because of helper containers or sidecars that lack ARM-compatible binaries.
- The Multi-Arch Shift: As of 2026, multi-arch support is a first-class expectation for official Docker images, making ARM64 a viable primary choice for many homelabs.
- Hardware Follows Software: Choose your hardware only after confirming that your specific software stack and its required hardware paths (GPU, drivers) are fully supported on that architecture.

The easy part is buying the box. The expensive part is finding out three hours later that the image you want only publishes one architecture, your media stack wants a different hardware path, or a helper container quietly pulls an x86-only dependency and dies with ‘exec format error‘. That is usually the moment when "ARM saves power" or "x86 is safer" stops being theory.

In practice, most modern self-hosted apps run perfectly well on both ARM64 and AMD64. The trouble starts when people treat Docker support as architecture support. They are not the same thing. What matters is whether the exact image tag you deploy publishes both ‘linux/amd64‘ and ‘linux/arm64‘, and whether the workload has side conditions like GPU access, transcoding, machine learning, or vendor binaries that change the answer.

Docker itself has supported multi-platform image workflows for a while now, and the whole mechanism is built around manifest lists that let one tag point to separate architecture-specific images. The official guidance from Docker is still the right mental model: check the published platforms first, then assume nothing else. multi-platform image builds make this possible, but they do not guarantee that every upstream project has done the work.

What this matrix is actually checking

I am not treating "someone on Reddit got it running once" as compatibility. This matrix is based on official docs, upstream image pages, and current tag metadata where available. An app counts as supported only when there is current evidence for both AMD64 and ARM64 in official or upstream-maintained images, or in the project’s own documentation.

That matters a lot on NAS hardware and small servers. If you are deciding between a low-power ARM board and an x86 mini PC, your architecture choice affects not only raw performance but also how much room you have for odd dependencies, sidecar containers, and hardware-specific features. That same tradeoff shows up in real NAS deployments like this Synology container setup, where the container itself is easy, but long-term compatibility is what decides whether the box stays useful.

ARM64 vs AMD64 compatibility matrix for popular self-hosted Docker apps

Data last verified: April 2026

App: Traefik | Primary Use: Reverse proxy | AMD64: Yes | ARM64: Yes | Current Reality: Clean multi-arch fit for both platforms | Practical Recommendation: Choose based on host preference, not Traefik itself

App: PostgreSQL | Primary Use: Database | AMD64: Yes | ARM64: Yes | Current Reality: Official image family publishes architecture variants | Practical Recommendation: Safe on both; storage and backup design matter more

App: Gitea | Primary Use: Git hosting | AMD64: Yes | ARM64: Yes | Current Reality: Docker deployment is straightforward on both | Practical Recommendation: ARM64 is fine for small and medium teams

App: Vaultwarden | Primary Use: Password vault | AMD64: Yes | ARM64: Yes | Current Reality: Strong multi-arch support, also lighter than full Bitwarden stack | Practical Recommendation: Very good ARM64 candidate

App: Pi-hole | Primary Use: DNS ad blocking | AMD64: Yes | ARM64: Yes | Current Reality: FTL ships for x86_64 and armv8/aarch64 | Practical Recommendation: Excellent on ARM64 and still fine on AMD64

App: Jellyfin | Primary Use: Media server | AMD64: Yes | ARM64: Yes | Current Reality: Core app is cross-platform; acceleration path needs extra attention | Practical Recommendation: Check transcoding hardware before choosing platform

App: Home Assistant Container | Primary Use: Home automation | AMD64: Yes | ARM64: Yes | Current Reality: Official images cover both platforms | Practical Recommendation: Both work; ecosystem add-ons may influence host choice

App: n8n | Primary Use: Workflow automation | AMD64: Yes | ARM64: Yes | Current Reality: Official Docker path covers both architectures | Practical Recommendation: ARM64 is usually fine unless you add x86-only helpers

App: Immich | Primary Use: Photo management | AMD64: Yes | ARM64: Yes | Current Reality: Official support on both; ML container has CPU caveats on x86 | Practical Recommendation: Good on both, but read the ML notes before upgrades

App: Outline | Primary Use: Wiki / knowledge base | AMD64: Yes | ARM64: Yes | Current Reality: Current tags publish both, though older user reports show ARM edge cases | Practical Recommendation: Verify the exact tag before production rollout

App: Nextcloud | Primary Use: Files and collaboration | AMD64: Yes | ARM64: Yes | Current Reality: Official image ecosystem includes arm64 variants | Practical Recommendation: Platform is rarely the blocker; storage layout is

App: ONLYOFFICE Docs | Primary Use: Document editing suite | AMD64: Yes | ARM64: Yes | Current Reality: ARM64 support exists in current community releases | Practical Recommendation: Now realistic on ARM64, but watch memory and integration overhead

Where ARM64 is already the sensible default

DNS, password management, Git hosting, automation, and lightweight reverse proxying are the easiest wins. Pi-hole, Vaultwarden, Gitea, n8n, and Traefik are exactly the kind of services that benefit from lower power draw without asking much from the CPU. For home labs that stay on 24/7, ARM64 stops being a compromise pretty quickly.

Pi-hole is a good example because the project does not leave architecture support vague. Its FTL binary is prebuilt for x86_64 and armv8/aarch64, which removes the usual guesswork around "will this break on ARM later?" If your broader question is which network filter belongs in that slot, the existing Pi-hole vs AdGuard comparison pairs nicely with this matrix.

Vaultwarden is another one I would not hesitate to run on ARM64. Upstream documents multi-arch container builds for AMD64, ARM64, and ARMv7, which is exactly what you want from a service that is supposed to disappear into the background and stay boring.

Where AMD64 still buys you margin

Media, AI-assisted photo stacks, and anything that leans on hardware acceleration still deserve more caution. The app may support both architectures perfectly well while the host GPU path, transcoding stack, or third-party sidecar does not. That is why people get caught by the wrong layer of the stack. The container launches, but the feature they actually care about does not.

Jellyfin is the classic case. The server itself is broadly cross-platform, but the real question is not whether Jellyfin starts. It is whether your chosen host exposes the video acceleration path you need with sane drivers and stable container permissions. That is also where AMD64 mini PCs still have an easier story for many people, especially if Intel Quick Sync is part of the plan.

Immich is more subtle. The project officially supports both AMD64 and ARM64, but its current requirements note that the machine learning container on AMD64 needs an ‘x86-64-v2‘ capable CPU starting with v2.6. That does not make ARM64 harder. It makes the old assumption that "x86 is the safe option" a lot less reliable than it used to be. Immich platform requirements are worth reading before you commit a box to it.

How to read image support without wasting an evening

The fastest check is simple: inspect the tag, not the marketing page. If the tag publishes both ‘linux/amd64‘ and ‘linux/arm64‘, you are past the first gate. Docker’s manifest tooling exists for exactly this reason, and it is still the most honest answer when docs are vague. Docker manifest tooling tells you what the registry will actually hand back to a host.

After that, look for these four things: official docs that mention both platforms, current tags that show both architectures, sidecar containers used by the stack, and hardware assumptions hidden behind optional features. Community threads are useful here, but mostly as warning signs. They tell you where people still trip, not what is officially guaranteed.

This gets even more important once you start publishing services behind clean hostnames. The app might be architecture-safe, but your layout still needs sane DNS and reverse-proxy design. The simplest explanation of that problem on TechRounder is this piece on clean local hostnames, which lines up well with Traefik, Caddy, and similar container-first setups.

What the current ecosystem says about self-hosting in 2026

The direction is obvious now. Multi-arch support is no longer a niche checkbox for Raspberry Pi users. Docker’s own official image program treats multiple architectures as a first-class expectation, and upstream projects are increasingly building and testing both AMD64 and ARM64 as part of normal release work. Docker Official Images make that trend easy to see.

That is why older advice like "just get x86 so nothing breaks" has started aging badly. It is still reasonable for very specific workloads, especially media acceleration or random vendor binaries. It is not a reliable default for the whole self-hosted category anymore.

You can see the same shift in practical NAS guides. A couple of years ago, many Docker-on-NAS articles quietly assumed x86 hardware. That assumption is getting weaker as more projects publish proper ARM64 images. A recent example is this Proton Bridge on Synology walkthrough, which would have been much more awkward when ARM builds were rarer.

Pick the host after you pick the stack

If your shortlist is mostly Traefik, Pi-hole, Vaultwarden, Gitea, n8n, and a database, ARM64 is already a very safe place to be. If the shortlist includes heavy transcoding, unusual vendor containers, or GPU-dependent features, check every image and every helper container before you buy hardware. The architecture argument is much less ideological than it used to be. The right answer is whichever host cleanly runs the exact tags your stack needs next month, not just today.

References

1. docs.docker.com - build / building - https://docs.docker.com/build/building/multi-platform/
2. docs.immich.app - install / requirements - https://docs.immich.app/install/requirements
3. docs.docker.com - reference / cli - https://docs.docker.com/reference/cli/docker/manifest/
4. github.com - docker-library / official-images - https://github.com/docker-library/official-images