Migrating Plex on a home Proxmox Server to a Cloud Based Virtual Private Server Written by Anthony Pierre

Introduction

In my previous submission, I explored the deployment and management of Plex Media Server in a home-based Proxmox environment, highlighting its potential as a reliable and scalable media platform. In this continuation, I pivot toward a more cloud-resilient approach by migrating Plex from a local Proxmox node to a Proxmox-powered Virtual Private Server (VPS). This decision reflects both a desire for 24/7 uptime and an evolution toward hybrid infrastructure that bridges on-prem and cloud-based resources.

Rather than focusing solely on Proxmox's features in isolation, this paper serves to demonstrate how Proxmox can power media workloads in a remote environment, providing flexibility, redundancy, and persistent availability without sacrificing manageability or control.

What is a VPS?

A Virtual Private Server (VPS) is a virtualized server environment hosted on a larger physical machine, provided by a cloud or hosting company. It behaves like a standalone server complete with its own operating system, storage, RAM, and network interface but it's actually one of several isolated environments running on a single, more powerful host system.

In essence, a VPS gives users many of the benefits of a dedicated server (like root access, custom configurations, and full control over services), but at a significantly lower cost because you're only renting a portion of the host machine's resources.

Think of a VPS like a flat in a shared apartment building:

- Each resident (VPS) has their own locked apartment (isolated OS environment). • They can decorate, cook, and manage their own space however they like (full root/admin access).
- While they share the same building infrastructure (underlying hardware), what happens in one apartment doesn't directly affect the others.
- If you outgrow your flat (need more CPU/RAM), you can move to a bigger one without buying the entire building.

Here is a real-world example. Imagine you run a media server, a personal wiki, and a small automation dashboard at home using a Proxmox server. You go on holiday or have an outage at home, and suddenly everything is offline. By setting up a VPS in the cloud, you can:

- Migrate critical workloads to keep them available 24/7.
- Use it as a backup node, testing ground, or public-facing edge device. • Create a Proxmox node on that VPS, extending your local lab into a hybrid cloud model.

Why Plex on Proxmox VPS Makes Strategic Sense

As mentioned in the previous section some were brought out of sense of pain! The

learnings around the creation of a homelab often result in the deployed software solutions…well breaking. But the joy of finding and correcting errors, elevates a user's fundamental learnings.
Most VPS give a choice of operating systems to use, with the Ubuntu flavours of Linux being some of the most popular. But depending on the company Proxmox can also be an option. My VPS provider Ihostart provides this option. Here a list of reasons to maintain the environment in the 'Cloud':

1. Environmental Consistency Across On-Prem and Cloud

Deploying Proxmox on both the home server and the VPS ensures a consistent administrative and technical environment. This continuity reduces friction when managing services like Plex backups, container creation, permissions, and network settings behave identically on both systems. For services as nuanced as Plex (with metadata paths, transcoding libraries, and media mounts), this uniformity means fewer platform-specific issues and smoother transitions between environments. It also reduces the need for re-documentation or training on alternate systems.

2. Enhanced Uptime and Remote Availability

One of the primary motivations for migrating Plex to a VPS is to improve accessibility and uptime. While home servers are vulnerable to power cuts, local ISP issues, and maintenance downtimes, a VPS in a reputable data centre offers greater network redundancy, power resilience, and guaranteed uptime SLAs. This shift makes Plex reliably available for both local and remote users, a crucial factor when media services are shared with friends, family, or remote clients. It also ensures uninterrupted access when away from home, without reliance on dynamic DNS or residential upload speeds.

3. Offloading Home Bandwidth and Resource Constraints

By hosting Plex in the cloud, the demands on your home internet especially upstream bandwidth are significantly reduced. This is particularly important when streaming high-bitrate media or serving multiple clients. Offloading Plex to the VPS also lessens the processing load on the home Proxmox node, making it available for other workloads or improving energy efficiency. The VPS can be provisioned with specific resource allocations for Plex (e.g., 4 vCPU, 8 GB RAM) to ensure consistent media delivery without interrupting home services.

4. Centralised and Remote-Friendly Management

The Proxmox web UI remains a powerful asset in this model. Even in a cloud context, Proxmox offers an intuitive, GUI-based management layer to monitor CPU usage, storage consumption, and container health all without needing to SSH into the VPS or juggle third-party orchestration tools. For Plex, this allows direct visibility into its resource footprint, and the ability to perform actions like container restarts, snapshotting, or restoring from backup, with just a few clicks even from a mobile device.

5. Structured, Reliable Backup and Recovery

With Proxmox-native tooling like vzdump, container backups can be scheduled, stored, and transferred securely between nodes. This enables regular Plex backups that include all configurations, metadata, and system state. Whether for disaster recovery, rolling

back failed updates, or migrating to new hardware, this process is highly repeatable and requires no reliance on third-party container platforms. Additionally, backups can be archived locally, pushed to object storage (e.g., S3), or synchronized back to the home server for extra redundancy.

6. Flexible Scaling and Future-Proofing

Unlike local hardware, a VPS can be scaled vertically or horizontally with minimal effort. Need more CPU for transcoding? Upgrade the VPS tier. Want to split frontend access from backend storage? Deploy a separate Proxmox container or LXC mount. Plex evolves quickly with changes in codec support, client apps, and hardware acceleration and the cloud environment offers a platform that can adapt without physical upgrades or downtime. Proxmox gives you the flexibility to move services, clone containers, and trial changes without major disruption.

Migration Objective

The purpose of this migration is to transition Plex Media Server, previously hosted on an on premises Proxmox LXC container, to a Proxmox-based Virtual Private Server (VPS) environment. This shift is not merely about relocating an application; it represents a strategic evolution toward a hybrid infrastructure model that delivers higher availability, bandwidth optimization, and operational continuity all without abandoning the familiar and powerful Proxmox ecosystem.

Migration Goals

- Ensure functional parity between local and VPS Plex instances, including configuration, metadata, and media access.
- Leverage Proxmox-native tooling for repeatable, reliable migration workflows. • Improve uptime and remote access to Plex without dependence on home network conditions or ISP limitations.
- Maintain a secure, manageable architecture that supports future scalability and recovery planning.

Technical Scope

- Source Environment: Proxmox VE running at home, hosting Plex in an LXC container with media volumes mounted via local disk or NFS.
- Destination Environment: VPS provisioned with Proxmox VE (KVM-supported), capable of restoring LXC containers with nested virtualization enabled.
- Plex Deployment: Migrated as-is via container backup and restore. Any hardware dependent elements (e.g. GPU transcoding) re-evaluated post-migration. • Storage Configuration: Plex config and metadata stored within the container; media libraries optionally mounted via external storage or cloud integration (e.g., rclone, CIFS, S3-fuse).

Migration Methodology

The following section outlines a detailed, CLI-focused process to migrate Plex Media Server from a home Proxmox environment to a remote VPS running Proxmox VE. This guide assumes containerised deployment, root access to both environments, and the creation of SSH keys for secure transfer.

What is SSH?

SSH (Secure Shell) is a cryptographic network protocol used to securely access and control remote machines over an unsecured network. It provides a command-line interface that allows users typically administrators or developers to log into another computer, execute commands, transfer files, or manage system processes with encryption and authentication in place.

Key Features of SSH

- Encryption: All data transmitted over SSH is encrypted, protecting credentials, commands, and file transfers from eavesdropping or tampering.
- Authentication: SSH supports multiple forms of authentication, the most common being:
- Username/password (basic, less secure)
- SSH key pairs (more secure, used for automation and access control) • Port Forwarding & Tunneling: SSH can create encrypted tunnels between machines useful for securely accessing internal services or forwarding traffic.
- File Transfer: Tools like scp and rsync use SSH under the hood to securely copy files between machines.

How SSH Key Authentication Works

1. Key Pair Generation

The user generates a private key and a public key. The private key remains on the client device (e.g., your VPS), and the public key is copied to the target server (e.g., your home Proxmox node).

2. Authentication

When the client connects, the remote server checks if the incoming user has a matching public key. If it does, and the private key is present and unlocked on the client, access is granted without a password.

3. Security Advantage

Unlike passwords, SSH keys are extremely hard to brute-force. They also allow fine-grained control (e.g., restricting keys to specific commands or IPs), making them ideal for secure automation and scripting.

So, imagine SSH as a secure, remote-access tunnel between your laptop/desktop and another computer. It’s like having a personal, encrypted hallway to a building where only your custom-shaped key fits the door. Anyone watching from the outside sees only encrypted traffic, not what you're doing inside.

What Is Cloudflare and Why It Matters

Cloudflare is a global network and security provider offering services like CDN, DDoS protection, DNS hosting, and secure tunnels. One key feature Cloudflare Tunnel (formerly Argo Tunnel) allows you to expose internal services without public IPs or open firewall ports

Core Uses of Cloudflare Tunnel

- Secure Access without Public Ports: Generates outbound-only connections from your server to Cloudflare's edge, letting you access services like Proxmox or Plex via hostname, without exposing ports.
- Simplified Networking: Cloudflare name resolution means you can SSH or transfer files using stable hostnames, even if your home IP changes or is private.
- No VPN Needed: Cloudflare handles secure connectivity, eliminating the need for VPN tunnels or static IP management.

Cost Overview

The purchase of a cheap domain allows for tunnel usage is free, but is also even on the Cloudflare Free Plan

Benefits Summary

- Security: No open inbound ports; all traffic comes through Cloudflare's edge. • Reliability: Access via stable hostnames, not IPs.
- Performance: Utilises Cloudflare's global network and optimised routing.
- Scalability: Host multiple services behind the same tunnel.
- Zero Trust Ready: You can layer Access controls for user-level

authentication. **The Use of Cloudflare Tunnels on Proxmox (Home)**

If the source Proxmox server already uses a Cloudflare Tunnel, this can significantly simplify connectivity during migration. For instance, instead of relying on a dynamic IP address or setting up port forwarding, the Cloudflare Tunnel allows you to access the home server using a stable, human-readable hostname, such as:

With the planning, architecture, and rationale now established, the following sections will shift focus to the practical execution of the migration process. This phase includes direct interaction with both the source and destination Proxmox environments and introduces command-line interface (CLI) operations used to complete the transition securely and reproducibly.

Where applicable, example commands are included with placeholder values (such as container IDs and IP addresses) to illustrate real-world syntax. Each step is designed to reflect a repeatable, minimal-intervention workflow using native Proxmox tooling suitable for both production and lab environments.

VPS Environment Setup

Before initiating the Plex container migration, the destination VPS must be prepared to receive and run LXC containers within a Proxmox VE environment. This section outlines the high-level steps required to prepare a clean VPS host to the same operational standard as the home-based Proxmox node.

Since this paper is focused on the migration process, it assumes Proxmox VE is already installed on the home environment. The steps below mirror that setup on the new VPS.

• **Base VPS Setup Instructions**

Note: The following assumes the VPS provider allows ISO installation or nested virtualisation. If not, nested KVM or LXC usage may be limited.

1. Install Proxmox VE

• Download and mount the Proxmox VE ISO.
• Use the provider's virtual console to install Proxmox on the VPS.
• Assign the public IP (203.0.113.50) as the management interface during install.
• Set the hostname as pve-vps and configure DNS accordingly.

2. Verify and Enable Nested Virtualisation

Check if the VPS supports nested virtualisation:

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

If not already enabled (common with KVM VPSes), enable nested virtualization on the VPS host kernel:

```
echo "options kvm-intel nested=Y" >
/etc/modprobe.d/kvm-intel.conf modprobe -r kvm_intel && modprobe
kvm_intel
```

Verify with:

```
cat /sys/module/kvm_intel/parameters/nested
```

Expected output: Y

3. Update and Configure Proxmox VE

```
apt update && apt full-upgrade -y
pveversion
```

(Optional) Add the non-subscription repository to avoid errors during updates:

```
sed -i 's|enterprise|no-subscription|'
/etc/apt/sources.list.d/pve enterprise.list
```

4. Configure Network Bridge (vmbr0)

Ensure vmbr0 is configured to pass traffic to the public interface and, if

required, route to a private WireGuard subnet:

```
auto lo
iface lo inet loopback
iface enp1s0 inet manual
auto vmbr0
iface vmbr0 inet static
 address 203.0.113.50
 netmask 255.255.255.0
 gateway 203.0.113.1
 bridge_ports enp1s0
 bridge_stp off
 bridge_fd
```

5. Create Storage Directory or Attach LVM

Verify /var/lib/vz is writable and has sufficient capacity for containers:

```
df -h /var/lib/vz
```

If using local-lvm (Proxmox default):

```
pvesm status
pvesm set local-lvm --content rootdir,images
```

Final VPS Prep Checklist

- Proxmox VE installed and accessible via web UI
- Public IP (203.0.113.50) correctly assigned
- Nested virtualization enabled (vmx/svm present)
- vmbr0 configured and online
- Storage ready (local-lvm or directory-based)

Making the Move

With the prerequisites in place, we can begin to move the LXC from Local to the cloud. **Use SSH Key-Based Authentication for Transfers**

Authenticate from the VPS to your home Proxmox using SSH keys:

On the VPS, generate a new key pair:

```
ssh-keygen -t ed25519 -f ~/.ssh/proxmox-home-key -C
"pve-vps→pve home"
```

Copy the public key to the home server:

```
ssh-copy-id -i ~/.ssh/proxmox-home-key.pub root@192.168.0.10
```

Test:

```
ssh -i ~/.ssh/proxmox-home-key root@192.168.0.10 hostname
```

If the hostname returns without prompting for a password, you're good to proceed. This ensures secure, passwordless file transfers for vzdump backups.

Create a Consistent Container Backup

On the home server (pve-home), either locally or via SSH:

```
vzdump 100 \
 --compress zstd \
 --dumpdir /var/lib/vz/dump \
 --mode stop
```

Confirm the backup file:

```
ls -lh /var/lib/vz/dump/vzdump-lxc-100-*.tar.zst
```

Pull the Backup File from the VPS

```
scp -i ~/.ssh/proxmox-home-key \
 root@192.168.0.10:/var/lib/vz/dump/vzdump-lxc-100-*.tar.zst
\  /var/lib/vz/dump/
```

This ensures a secure, authenticated file pull without needing to manually log into the home server.

Restore Container on VPS

On pve-vps, restore the container:

```
pct restore 100
/var/lib/vz/dump/vzdump-lxc-106-*.tar.zst \  --storage
local-lvm \
 --hostname plex-vps
```

Review or modify network settings in:

```
nano /etc/pve/lxc/100.conf
```

Update the container's IP for the VPS environment:

```
lxc.net.0.type: veth
lxc.net.0.link: vmbr0
lxc.net.0.ip: 10.99.99.10/24
lxc.net.0.gateway: 10.99.99.1
```

Start the container:

```
pct start 100
```

Verify IP:

```
pct exec 100 ip a
```

Conclusion - Evolving Your Homelab with Cloud Resilience

The migration of Plex from a local Proxmox LXC to a cloud-hosted Proxmox VPS represents a pivotal step in transforming a hobbyist homelab into a resilient, hybrid infrastructure. By leveraging consistent tooling across environments, integrating SSH and Cloudflare for secure access, and embracing cloud-native reliability, the result is more than just uptime it's future proofing

Whether you're exploring remote workloads, improving uptime for shared media, or experimenting with disaster recovery, this strategy offers a replicable model for evolving infrastructure without losing the hands-on control that makes Proxmox so appealing.

Next steps could include:

- Automating backups between environments using rsync or rclone.

- Using Terraform or Ansible for container orchestration.
- Introducing monitoring tools like Grafana + Prometheus to keep tabs on resource use. - Exploring Kubernetes or Docker Swarm for scaling multi-service apps.

Ultimately, the cloud isn't here to replace your homelab it's here to extend it, and the learnings on network infrastructure could set you in good stead for client environments.