🛠️ Sovereign Homelab Architecture Cheat Sheet

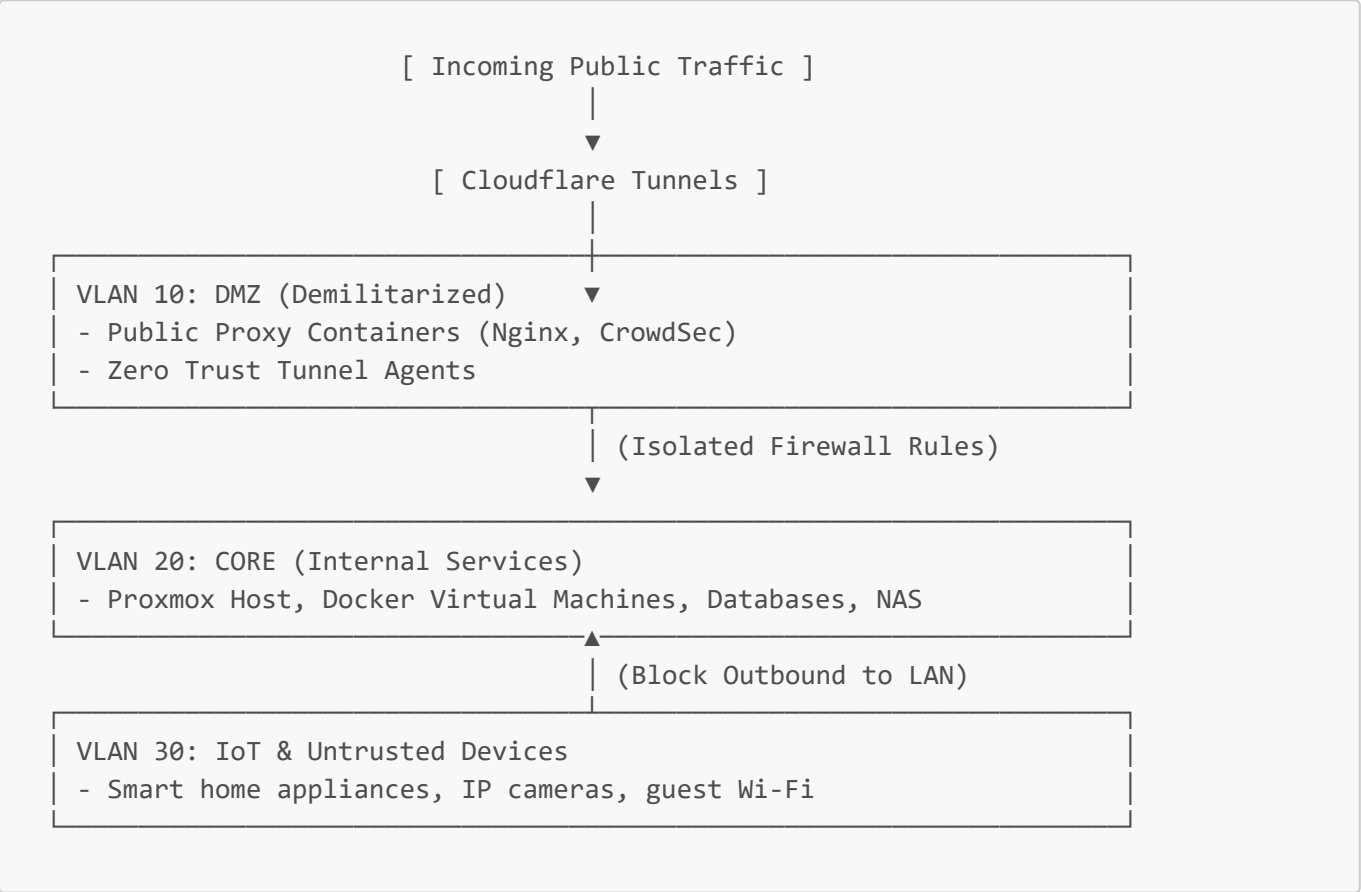
The Definitive Blueprint for Independent Digital Infrastructure

By Ghannams Academy | Protocol Version 1.0

🌐 1. Secure Network Topology: The 3-VLAN Rule

Digital sovereignty begins at the router. Exposing self-hosted applications on a single flat local network leaves all trusted systems vulnerable if one container is compromised.

Recommended Architecture:



💎 Architectural Gems:

- **Default Block Rules**: Configure your firewall to block VLAN 30 (IoT) and VLAN 10 (DMZ) from initiating any connection to VLAN 20 (Core).
- **DNS Protection**: Deploy AdGuard Home or Pi-hole in VLAN 20, and force all VLANs to route DNS requests through it via NAT redirection rules on port 53.

🖥️ 2. Virtualization Protocol: Proxmox VE Optimizations

Proxmox is the ultimate hypervisor for homelabs. However, choosing between virtual machines (VMs) and Linux Containers (LXCs) is crucial for performance.

Attribute	Virtual Machines (VM)	Linux Containers (LXC)
Kernel	Isolated (Independent OS Kernel)	Shared with Host Kernel
Performance	High Overhead (Best for Docker Hosts)	Near-Native Speed (Best for database/Nginx)
Storage	Virtual Disk (.raw / .qcow2)	Mount Points (Direct ZFS dataset mapping)
Use Case	Kubernetes, Docker, Windows Server	PostgreSQL, Vaultwarden, AdGuard Home

💎 Configuration Gems:

- **Docker on VM, Not LXC**: Never run Docker directly inside a standard unprivileged LXC container. Nested container loops cause storage driver overhead and kernel bugs. Deploy Docker inside an Ubuntu Server VM.
- **Direct ZFS Bind Mounts**: For high-performance database workloads on LXC, bypass virtual disks and map a host ZFS dataset directly into the container using:

```
pct set <container_id> -mp0 /pool/dataset,mp=/var/lib/postgresql/data
```

💾 3. Memory Protocol: Swap Space & Swappiness Optimization

Swap memory is a dedicated space on your storage drive (either a file or a partition) used as an overflow buffer when your server's physical RAM becomes fully saturated.

Why is Swap Memory Critical?

If your server runs out of physical RAM and has no swap allocated, the Linux kernel encounters an **Out-Of-Memory (OOM)** error. To prevent a full system freeze, the kernel activates the **OOM-Killer**, which immediately terminates the largest memory-consuming processes—often killing your database (PostgreSQL/MariaDB) or web server (Nginx/Express) without warning.

Swap space acts as a safety valve. It buys you time to access the terminal, debug leaks, and prevents critical services from crashing during traffic spikes.

How to Configure Swap & Optimize Swappiness:

Step 1: Create a 4GB Swap File

We use `fallocate` to reserve space and harden permissions so only root can access it:

```
sudo fallocate -l 4G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

Step 2: Make it Persistent

Ensure the swap partition is mounted automatically on reboot by appending the entry to `/etc/fstab`:

```
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

Step 3: Optimize the Swappiness Gem

By default, Linux has a swappiness value of `60`. This means it will aggressively push inactive memory pages to disk even if you have plenty of RAM left, causing performance degradation on SSDs.

For home servers, configure swappiness to `10`. This instructs the kernel to **only swap as a last resort** (when RAM reaches 90% utilization), preserving SSD lifetime and maintaining maximum response speeds.

```
# Apply immediately
sudo sysctl vm.swappiness=10

# Make persistent across reboots
echo 'vm.swappiness=10' | sudo tee -a /etc/sysctl.conf
```

☁️ 4. Ingress Security: Zero Open Ports Strategy

Traditional self-hosting requires "port forwarding" (ports 80/443 or 22) on your router, exposing your home's public IP address to malicious scanners and DDoS attacks. A sovereign setup avoids this entirely.

```
[ Public Client ] ──► [ Cloudflare Edge Network ]
                                 │
                                 ▼ (Outbound-only TLS Tunnel Connection)
[ Home Router ]   ──► [ cloudflared Container ] ──► [ Local Host Web App ]
(Zero Open Ports)
```

Cloudflare Tunnels: Secure Public Routing

Cloudflare Tunnels connect your home server directly to Cloudflare's global edge network via an outbound-only daemon (`cloudflared`).

💎 **Ingress Gems:**

- **No Static IP Required**: Because the connection is outbound-only, your home server calls out to Cloudflare. If your Internet Service Provider changes your public IP address (which dynamic IPs do constantly), the tunnel automatically reconnections. You do **not** need a static IP, dynamic DNS (DDNS) scripts, or router configuration.
- **Hidden Network Footprint**: Even if you have a static IP, exposing ports invites automated vulnerability exploits. With a Cloudflare Tunnel, your public IP is completely hidden. All client traffic passes through Cloudflare's Web Application Firewall (WAF) first, shielding you from Layer 3/4 DDoS attacks.

🐚 5. Tailscale: Secure Remote Shell & File Transfer

While Cloudflare Tunnels are perfect for public websites, you must never expose admin tools (Proxmox GUI, SSH ports, database managers) through them. Instead, build a private overlay network using **Tailscale**.

Tailscale uses the **WireGuard** protocol to create a secure, encrypted peer-to-peer (P2P) mesh VPN. It assigns every connected device (laptop, phone, home server) a stable internal IP address (e.g., `100.x.y.z`) that works globally.

💎 Management Gems:

- **Bypass Carrier-Grade NAT (CGNAT)**: If your home connection is behind CGNAT (common on 5G or fiber routers where you share a public IP with neighbors), traditional VPNs will fail. Tailscale automatically orchestrates direct P2P connections through NAT traversal, allowing remote access with zero configuration.
- **Tailscale SSH (Terminal over the Web)**: Turn on Tailscale SSH to securely terminal into your server from any device over the internet. You do **not** need to generate, copy, or maintain public/private SSH keys across your laptops and phones. Tailscale handles machine identity and key management out-of-the-box:

  ```
  # Securely connect from your laptop on public Wi-Fi directly to your server:
  ssh username@my-homelab-server
  ```

- **Taildrop (Secure File Transfers)**: Need to send a configuration backup, a movie, or a script file from your phone or laptop directly to your server? Taildrop lets you send files securely between your mesh nodes over the encrypted network with one-click:

  ```
  # Send a file from server to your laptop:
  tailscale file cp backup.tar.gz my-laptop:
  ```

🐳 6. Production-Grade Docker Compose Standards

Docker Compose is the standard for service orchestration. Avoid ad-hoc container configuration by enforcing these strict files guidelines.

```
version: '3.8'

services:
  app-service:
    image: postgres:16-alpine # 💎 GEM 1: Never use 'latest'. Pin major/minor
versions.
    container_name: production-db
    restart: unless-stopped
    environment:
      - POSTGRES_DB=app_db
```

```
    volumes:
      - db_data:/var/lib/postgresql/data # 💎 GEM 2: Use named volumes for
persistent data.
    deploy:
      resources:
        limits:
          cpus: '1.50' # 💎 GEM 3: Set resource limits to prevent memory leaks
from crashing the host.
          memory: 1024M
    networks:
      - database-net

networks:
  database-net:
    driver: bridge # 💎 GEM 4: Keep containers on isolated backend bridges.

volumes:
  db_data:
```

💎 Docker Gems:

- **Log Rotation**: Docker containers can fill up your server's storage with log files. Prevent this globally by configuring `/etc/docker/daemon.json` with:

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "3"
  }
}
```

💰 7. The 3-2-1 Backup Protocol: Restic + Backblaze

A homelab without verified backups is a ticking time bomb. Follow the **3-2-1 backup rule**: 3 copies of your data, across 2 different media types, with 1 copy stored offsite.

Implementation Blueprint:

- **Tool**: **Restic** (Fast, secure, deduplicating, and encrypted backup CLI tool).
- **Media 1 (Primary)**: Active running database/file instance.
- **Media 2 (Local)**: Scheduled Restic backup to an encrypted folder on a local NAS/shared drive.
- **Media 3 (Offsite)**: Scheduled Restic backup to an encrypted Cloud Object (e.g., Backblaze B2 or AWS S3 Glacier).

💎 Backup Gems:

- **Backup Database Dumps**: Never backup an active database file directly. It can cause corrupted backups. Always execute a dump query first (`pg_dump`), then backup the resulting SQL dump file:

```
pg_dump -U owner_user app_db > /backups/db_dump.sql
restic -r s3:s3.us-west-004.backblaze.com/my-bucket backup
/backups/db_dump.sql
```

- **Prune Automations**: Clean up old snapshots automatically on a retention policy:

```
restic forget --keep-daily 7 --keep-weekly 4 --keep-monthly 12 --prune
```