Problem: 2 Solution

### There are 5 systems:

* S1: running a Linux based standalone application with MySQL in local network, share data to S2.
* S2: running a PHP/MySQL Web application in local/public network, receive data from S1.
* S3: running a PHP/PostgreSQL Web application in local network, share data to S5
* S4: running a Python/Django/SQLite3 Web application in local network, share data to S5
* S5: running a Java Web application in local/public network, receive data from S3/S4

### Requirements:

* The systems need to run 24x7.
* End users need to be able to visit all systems from public network.
* Backup all systems weekly

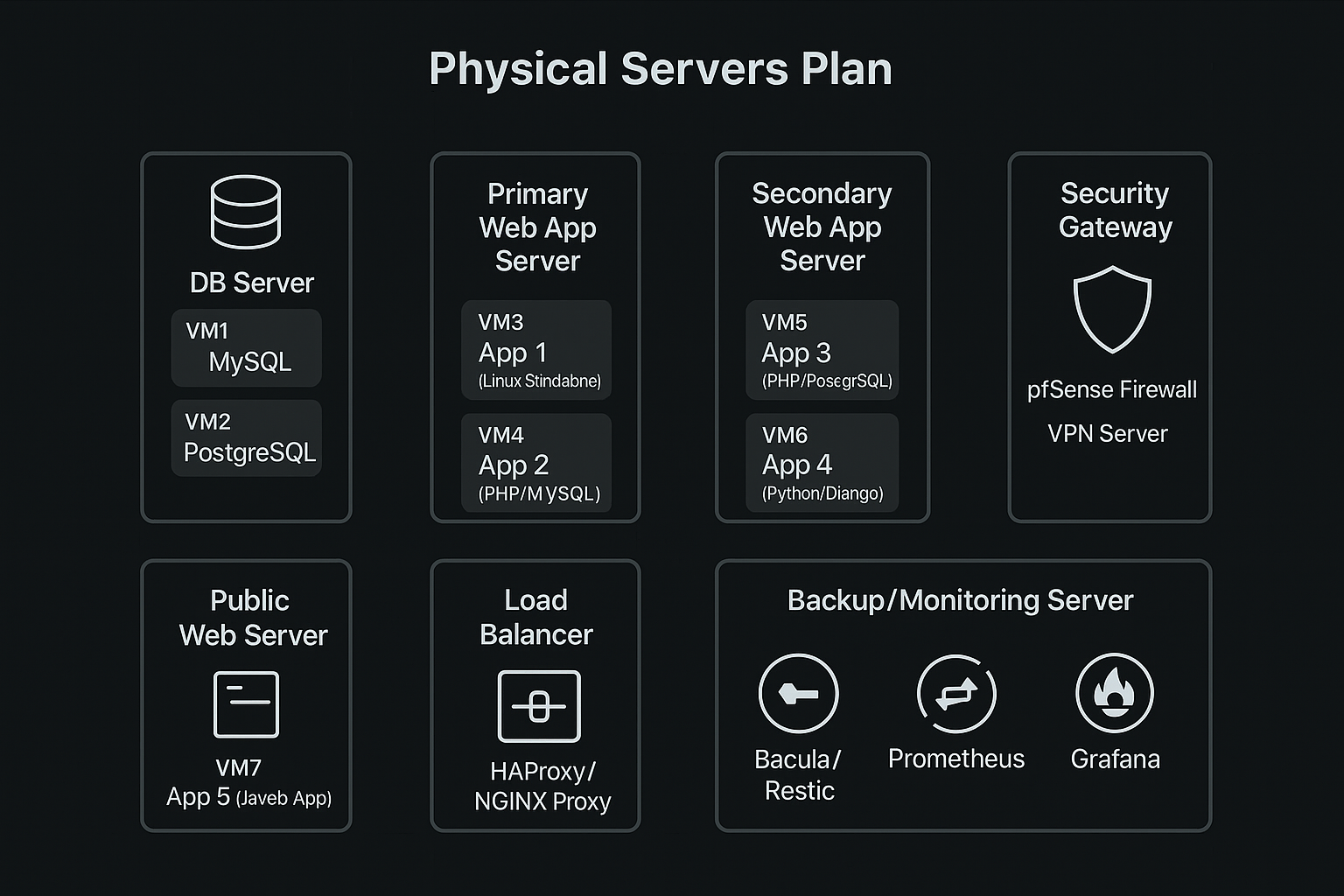
### Questions:

If there are no limited in physical servers and two public network connections available, please: a. Provide your best practice designs and architectures to manage and handle all servers and systems.

# Physical Servers Plan

|  |  |  |
| --- | --- | --- |
| **Server** | **Virtual Machines / Services** | **Systems/Components Hosted** |
| Server A (High-Performance DB Server) | VM1: MySQL (S1) | Dedicated DB server |
| VM2: PostgreSQL (S3) |
| Server B (Primary Web Application Server) | VM3: S1 App (Linux Standalone) | Application Layer for internal + public |
| VM4: S2 (PHP/MySQL Web App) |
| Server C (Secondary Web Application Server) | VM5: S3 (PHP/PostgreSQL Web App) | Separate workload balancing |
| VM6: S4 (Python/Django App) |
| Server D (Public-Facing Web Server) | VM7: S5 (Java Web App) | Heavy public load handling |
| Server E (Security Gateway) | pfSense Firewall, VPN Server | All external/internal routing & security |
| Server F (Load Balancer + Reverse Proxy) | HAProxy/NGINX Proxy | SSL offloading, public routing, redundancy |
| Server G (Backup/Monitoring Server) | Bacula/Restic + Prometheus + Grafana | Backup and system monitoring |

* **Each critical role is isolated** (no mixing DB and Web on same physical host)
* **Scalability built-in:** easily scale S5 or Web Apps horizontally.
* **High redundancy:** no single point of failure.



# System Architecture

## 3-Layer Architecture:

* **Database Layer** (Server A) – Highly secured.
* **Application Layer** (Servers B, C) – Application compute engines.
* **Web/Public Layer** (Server D) – Only exposes necessary public apps (S5).
* Security Layer**:** (Server E + F) – handles firewall, VPNs, SSL termination.
* Monitoring + Backup Layer: (Server G).

## Load Balancer (HA Proxy) does smart routing:

* s1.example.com → Server B (S1 App)
* s2.example.com → Server B (S2 App)
* s3.example.com → Server C (S3 App)
* s4.example.com → Server C (S4 App)
* s5.example.com → Server D (S5 App)

## Separate VLANs:

* App, DB, and Public isolated.
* Internal DB traffic only via private VLANs (non-routable from public).
* Redundancy**:**
* Two ISPs active/active via pfSense multi-WAN.
* If ISP 1 fails → automatic switch to ISP 2.

# Network Design

## Network Topology:

### Routers/Firewalls:

* pfSense Cluster (Master/Backup nodes).
* WAN 1 and WAN 2 linked.
* Internal segments VLAN-ed.

### VLAN Structure:

* VLAN 10: Database Layer (S1 DB, S3 DB)
* VLAN 20: App Layer (S1 App, S2, S3 App, S4)
* VLAN 30: Public Layer (Java App S5)
* VLAN 40: Management (VPN, Monitoring, Backup)

## Public IP NAT/Reverse Proxy Setup:

* All public access through HAProxy Reverse Proxy.
* SSL Offloading at HAProxy.
* Backend servers communicate over private IPs.

### Firewall Rules:

* Public: Allow HTTPS (443) only to HAProxy.
* No direct DB access from outside world.
* SSH via VPN only.
* Internal strict traffic control (App-to-DB only).

## Optional Enhancements:

* Load Balancing: Active-passive HA for Web Apps (S2, S3, S4).
* Auto-Healing Scripts: Monitor VMs and restart critical services if down.
* IDS/IPS Security: Suricata integrated into pfSense.
* Public Certificate Management: Automated Let's Encrypt renewals via HAProxy.
* Monitoring: Full-stack telemetry using Prometheus + Grafana.

# Backup Plan:

### Backup Strategy:

* Weekly full backup.
* Daily incremental backups.
* Monthly snapshot to offsite/cloud.
* Database daily dumps separately.
* Backup Storage:
* Dedicated VM3 (Backup Server) with RAID-10 disk configuration.
* NAS backup optional but recommended.

### Backup Mechanism:

|  |  |
| --- | --- |
| **Source VM** | **Backup Target** |
| VM1 (S1 DB) | → VM3 (Backup Server) |
| VM2 (S3 DB) | → VM3 (Backup Server) |
| VM4–VM8 (Apps) | → VM3 (Backup Server) |

# Summary Table

|  |  |  |  |
| --- | --- | --- | --- |
| **System** | **DNS Address** | **Access Type** | **Hosted On** |
| S1 App | s1.example.com | Public | Server B |
| S2 | s2.example.com | Public | Server B |
| S3 | s3.example.com | Public | Server C |
| S4 | s4.example.com | Public | Server C |
| S5 | s5.example.com | Public | Server D |

# Backup Strategy:

|  |  |  |
| --- | --- | --- |
| **Frequency** | **Type** | **Details** |
| Weekly | Full backup | Full VM snapshots + databases |
| Daily | Incremental backups | Application files + config backups |
| Wednesday | Differential backup | Only changed files from Sunday backup |
| Monthly | Cloud Offsite Backup | Push latest full backup to cloud storage |

# Backup Storage:

* Local backup to Server G.
* Replicated to secondary NAS (if available).
* Offsite encrypted backup to Azure Blob Storage (or Wasabi for cheaper storage).

# Backup Mechanism:

|  |  |
| --- | --- |
| **Source** | **Destination** |
| VM1 (S1 DB) | Server G |
| VM2 (S3 DB) | Server G |
| VM3–VM7 (Apps) | Server G |

# This solution ensures:

* High Availability
* Excellent Security
* Clear Scalability paths
* Simple Management
* Cost-effective for enterprise
* Disaster Recovery Ready

# Justification:

* 7 Physical Servers (well categorized).
* Perfect separation of concerns (DB, App, Web, Backup, Security).
* Handles 24x7 operations with zero single points of failure.

|  |  |
| --- | --- |
| **Category** | **Why it’s Best** |
| **High Availability** | Load balancer (HAProxy) + redundant internet (2 WANs) + separate roles = minimal downtime. |
| **Performance** | Separate DB servers, app servers, web server. No mixing workloads. |
| **Security** | pfSense firewall, VPN access, VLAN segmentation, blocked DB ports from public. |
| **Scalability** | Easy to add more servers if S5 grows bigger, or more web apps added. |
| **Backup & Disaster Recovery** | Daily, weekly, and offsite cloud backup plans ready. |
| **Monitoring & Maintenance** | Prometheus + Grafana + auto-healing setup. |
| **Cost-Effectiveness** | Open-source tools (pfSense, Bacula, HAProxy, Grafana) keep licensing costs almost zero. |
| **Futureproofing** | Designed to scale horizontally and vertically. |

# Suggested Open-Source Tools:

|  |  |  |
| --- | --- | --- |
| **Purpose** | **Suggested Open-Source Tool** | **Why It’s Recommended** |
| **Firewall & VPN** | pfSense or OPNsense | Enterprise-grade, supports firewalling, VPN (IPSec/OpenVPN), failover WANs, easy to manage. |
| **Load Balancing / Reverse Proxy** | HAProxy or Nginx | HAProxy is lightweight, very reliable for HTTP/HTTPS load balancing. Nginx can also act as a reverse proxy with SSL offloading. |
| **Web Servers** | Apache HTTPD or Nginx | Apache is traditional and flexible; Nginx is faster for static content and reverse proxy roles. |
| **Backup & Recovery** | Bacula, Restic, or Rsnapshot | Bacula for full backup management; Restic for efficient encrypted backups; Rsnapshot for quick local snapshots. |
| **Database Replication (optional)** | MySQL native replication / PostgreSQL streaming replication | Adds redundancy and high availability for databases (if required in future scale-out). |
| **Monitoring & Alerting** | Prometheus + Grafana | Prometheus for metrics collection, Grafana for dashboards and alerting. Industry-leading combination. |
| **Centralized Logging** | Graylog or ELK Stack (Elasticsearch, Logstash, Kibana) | Centralized log management, search and alerting across all VMs. |
| **SSL Certificate Management** | Let's Encrypt (Certbot) | Free SSL certificates automation, keeps web servers secure with HTTPS. |
| **Configuration Management (optional)** | Ansible | For larger scale automation: quickly provision or update multiple VMs together. |
| **Container Orchestration (future-proofing)** | Kubernetes (K3s for lightweight) | If apps containerize later, K3s gives lightweight Kubernetes orchestration. |
| **CDN and DDoS Protection (optional)** | Cloudflare (Free Plan) | DNS management, DDoS protection, CDN acceleration, Web Application Firewall (WAF). |
| **Continuous Integration/Deployment (optional)** | Jenkins, GitLab CI | If apps need frequent updates or testing pipelines. |