Problem 1: 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 two physical servers available and one public network connection, please:

a. Provide a physical servers plan for all systems.

b. Provide a systems architecture.

c. Provide a network design for all systems.

# Suggested Open-Source Tools:

|  |  |
| --- | --- |
| Purpose | Tools |
| Firewall | pfSense or OPNsense |
| Load Balancer/Reverse Proxy | HAProxy + Nginx (with Let's Encrypt SSL) |
| Backup | Bacula or Restic |
| Monitoring | Prometheus + Grafana |
| Log Management (Optional) | Loki, ELK Stack |
| Database Replication (Optional) | Native MySQL/PostgreSQL replication |

# Physical Servers Plan

|  |  |  |
| --- | --- | --- |
| **Server** | **Virtual Machines** | **Systems Hosted** |
| Server A (Database Priority) | VM1: S1 | Linux App + MySQL (S1) |
| VM2: S2 | PHP/MySQL Web App (S2) |
| VM3: Backup Server | Backup system (Bacula/Restic) |
| Server B (Web App Priority) | VM4: S3 | PHP/PostgreSQL Web App (S3) |
| VM5: S4 | Python/Django/SQLite3 Web App (S4) |
| VM6: S5 | Java Web App (S5) |
|  |  |  |

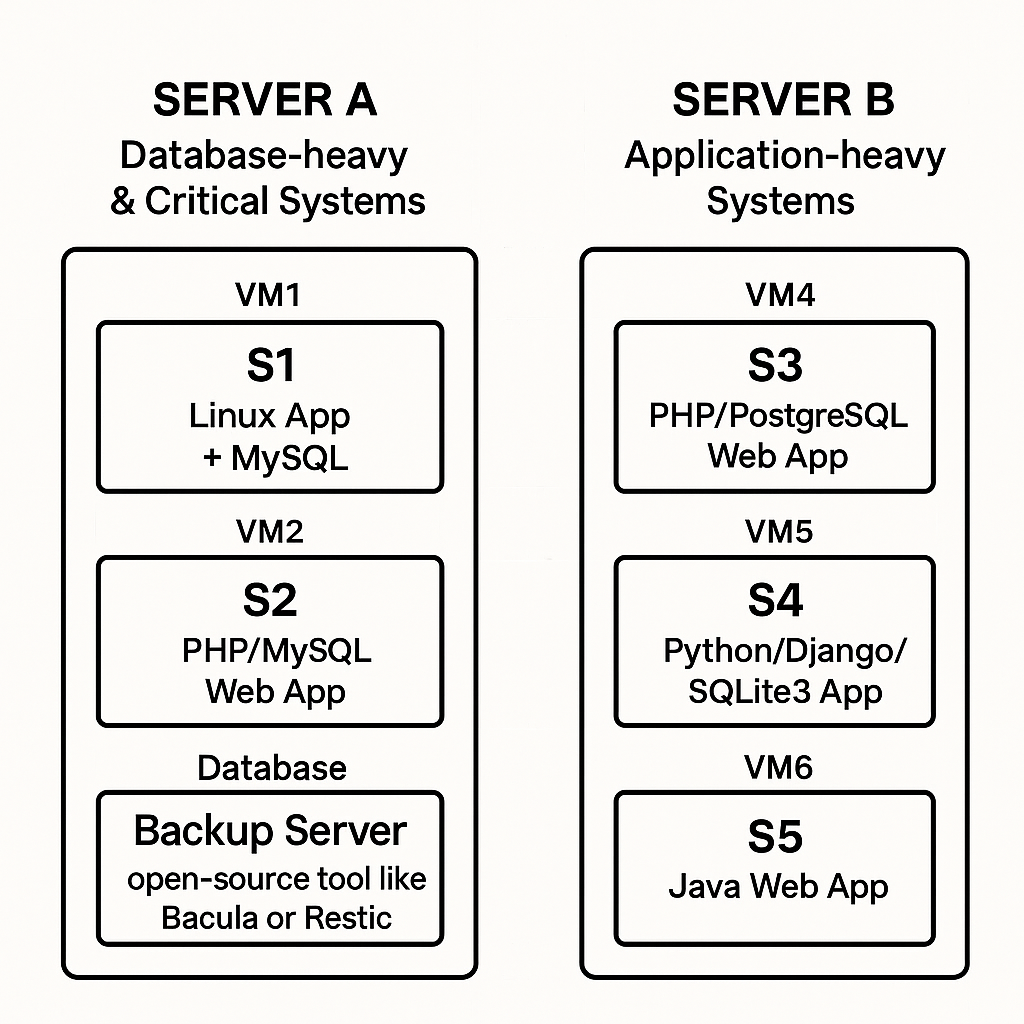


Fig. 1

## Server A (High Priority: Database-Intensive & Critical Systems)

### VM1: S1 – Linux App + MySQL

* Purpose: Core application logic and local database operations.
* Reasoning: Central to data processing and sharing with S2.

### VM2: S2 – PHP/MySQL Web App

* Purpose: Web interface for users, receiving data from S1.
* Reasoning: Public-facing application requiring high availability.

### VM3: Backup Server (e.g., Bacula or Restic)

* Purpose: Handles weekly backups for all systems.
* Reasoning: Centralized backup management to ensure data integrity.

## Server B (Application-Heavy Systems)

### VM4: S3 – PHP/PostgreSQL Web App

* Purpose: Web application interfacing with PostgreSQL database.
* Reasoning: Handles user interactions and data sharing with S5.

### VM5: S4 – Python/Django/SQLite3 App

* Purpose: Web application with Django framework and SQLite3.
* Reasoning: Lightweight application sharing data with S5.

### VM6: S5 – Java Web App

* Purpose: Receives and processes data from S3 and S4.
* Reasoning: Central hub for data aggregation and user access.

### Note:

* Spreading the load: Heavy databases (S1/S2) on Server A, multiple lighter web apps (S3/S4/S5) on Server B.
* Backup server manages all backups independently from critical app VMs.

# System Architecture

A diagram of a network

AI-generated content may be incorrect.

Fig. 2

A diagram of a computer

AI-generated content may be incorrect.

Fig. 3

# Network Design

## Network Topology:

### **Routers/Firewalls**:

* Open-source firewall (pfSense or OPNsense) for security and VPN.
* VLAN segmentation for security zones.

### **Private VLANs**:

* **VLAN 10**: Server A Systems (S1, S2, Backup)
* **VLAN 20**: Server B Systems (S3, S4, S5)

### **Public IP NAT or Reverse Proxy Setup**:

* **Expose all systems** to public network:
  + **s1.example.com** → VM1 (S1)
  + **s2.example.com** → VM2 (S2)
  + **s3.example.com** → VM4 (S3)
  + **s4.example.com** → VM5 (S4)
  + **s5.example.com** → VM6 (S5)

### **Firewall Rules**:

* Allow HTTPS (port 443) access for S1–S5.
* Block database ports (MySQL/PostgreSQL/SQLite3) from public access.
* Allow internal database communication only.
* SSH access restricted to VPN or trusted IPs.

## **Optional Enhancements**:

* Setup VPN access for management.
* Enable HAProxy SSL offloading and health checks.

## Backup Plan:

* Weekly snapshots using **Bacula**, **Restic**, or even **rsnapshot**.
* Store backups on a separate disk or NAS if available.
* **Public Access Plan:**
* Public DNS:
* s2.example.com → NAT → VM2
* s5.example.com → NAT → VM6

# Summary Table

|  |  |  |
| --- | --- | --- |
| **System** | **DNS Address** | **Access Type** |
| S1 | s1.example.com | Public Access |
| S2 | s2.example.com | Public Access |
| S3 | s3.example.com | Public Access |
| S4 | s4.example.com | Public Access |
| S5 | s5.example.com | Public Access |

## Backup Plan (for all systems)

## Backup Strategy:

* **Weekly Full Backups** (every Sunday night).
* **Daily Incremental Backups** (Monday–Saturday).
* **Mid-week Differential Backup** (Wednesday night).

### Backup Storage:

* Dedicated storage disk or NAS.
* Optional cloud backup for disaster recovery.

## Backup Mechanism:

* Install Restic or Bacula agent inside each VM.
* Backup flows:
* VM1 (S1) → Backup Server (VM3)
* VM2 (S2) → Backup Server (VM3)
* VM4 (S3) → Backup Server (VM3)
* VM5 (S4) → Backup Server (VM3)
* VM6 (S5) → Backup Server (VM3)