LifeBox Digital Legacy System

Architecture Overview & Implementation Plan

🎯 Project Goals

Primary: Secure, accessible digital legacy storage for family **Secondary**: Portable backup system with guaranteed recovery **Constraint**: Must be usable by non-technical family members

---

🏗️ System Architecture

Dual System Design

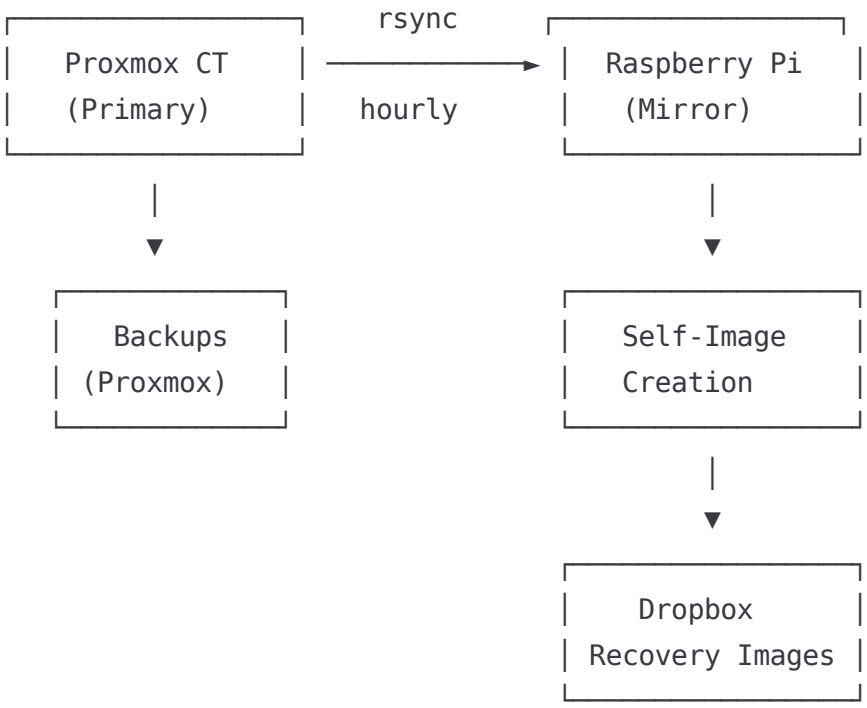
Primary System: Proxmox Container

- Location: Home server infrastructure
- Role: Daily use, full-featured interface
- Storage: Robust with existing backup systems
- Access: Home network web interface

Secondary System: Raspberry Pi

- Location: Portable (pocket-sized)
- Role: Mirror + emergency access
- Storage: SD card + USB backup
- Access: Battery-powered web server

Data Flow



🔄 Synchronization Strategy

Local Network Sync (Proxmox → Pi)

- **Method**: rsync over SSH
- **Frequency**: Hourly
- **Direction**: One-way (Pi never writes back)
- **Security**: SSH keys (no passwords)

Sync Components

bash

```
# Data files
/var/data/ → /home/pi/lifebox/data/

# Application config
/etc/app/ → /home/pi/lifebox/config/

# Database
SQLite dump → /home/pi/lifebox/db/backup.sql

# Media files
/var/media/ → /home/pi/lifebox/media/
```

Exclusions

- Temporary files
- Log files
- Cache directories
- Live database files (use dumps)

💾 Backup & Recovery System

Pi Self-Imaging Process

Daily Automation:

1. Check for system changes (packages, config, data)
2. If changed → create compressed image
3. Upload to shared Dropbox folder
4. Maintain 2 most recent images
5. Log creation details

Image Triggers:

- OS package updates
- Application updates
- Significant data growth (>10%)
- Manual trigger via web interface
- Weekly baseline regardless

Image Storage Structure

```
Dropbox/LifeBox-Recovery/
├── current.img.gz     (latest recovery image)
├── previous.img.gz    (backup image)
├── creation.log       (metadata & timestamps)
└── README.txt         (family instructions)

VPS Backup Location:
├── lifebox-backup.img.gz  (secondary copy)
└── recovery-docs/         (instructions mirror)
```

👨‍👩‍👧‍👦 Family Recovery Process

Scenario 1: Proxmox Failure

- Pi continues serving from last sync
- No family action needed
- Automatic failover

Scenario 2: Both Systems Lost

1. Family accesses shared Dropbox folder
2. Downloads latest `current.img.gz`
3. Flashes to SD card using included instructions
4. Boots Pi with family password
5. System serves all data immediately

Scenario 3: Complete Infrastructure Loss

- Pi image contains encrypted cloud backup credentials
- System can rebuild from multiple cloud services
- Requires family recovery password

🔐 Security Layers

Operational Security

- Simple password-protected web interface
- Local network access only (primary)
- Family password for Pi access

Backup Security

- Recovery images: lightly encrypted (family accessible)
- Cloud backups: strongly encrypted
- Multiple provider redundancy

Access Control

- Family members have Dropbox access
- Clear instructions in recovery folder
- Emergency contact information included

💻 Technical Implementation

Pi Tasks (Cron Jobs)

bash

```
# Sync from Proxmox
0 * * * * /home/pi/scripts/sync-from-proxmox.sh

# Health check & weekly image creation
0 2 * * 0 /home/pi/scripts/weekly-image-check.sh

# Upload to Dropbox
0 3 * * 0 /home/pi/scripts/upload-recovery.sh
```

Sync Script Essentials

bash

```
#!/bin/bash
# sync-from-proxmox.sh

# Dump database on Proxmox first
ssh proxmox 'sqlite3 /var/app/data.db .dump > /tmp/db-backup.sql'

# Sync all components
rsync -avz --delete proxmox:/var/data/ /home/pi/lifebox/data/
rsync -avz --delete proxmox:/etc/app/ /home/pi/lifebox/config/
rsync -avz proxmox:/tmp/db-backup.sql /home/pi/lifebox/db/

# Validate sync
/home/pi/scripts/validate-sync.sh
```

Health Monitoring

- Last sync timestamp
- Storage usage
- Network connectivity
- Service status dashboard

📋 Implementation Phases

Phase 1: Basic Dual System

- Set up Proxmox container
- Configure Pi with web server
- Implement basic rsync

Phase 2: Automation

- Add cron jobs for sync
- Health monitoring dashboard
- Basic image creation

Phase 3: Recovery System

- Self-imaging automation
- Dropbox integration
- Family instruction creation

Phase 4: Polish

- Error handling
- Monitoring alerts
- Documentation completion

🎛️ Control Interface

Pi Web Dashboard

- System status overview
- Last sync status
- Storage usage
- Manual sync trigger
- Image creation trigger
- Health check results

Family Interface

- Simple password entry
- Read-only data access
- Contact information
- Recovery instructions

🔧 Hardware Requirements

Raspberry Pi

- Pi 4 (4GB minimum)
- 64GB+ SD card (Class 10)
- USB 3.0 backup drive
- Battery pack for portability
- Case with cooling

Proxmox Container

- 2GB RAM allocated
- 50GB+ storage
- Standard Debian container

📝 Remaining Questions

1. **Data types**: What specific information needs storage?
2. **Application framework**: Flask, Django, or simple static files?
3. **Data organization**: Folder structure and categorization?

🛠️ Decisions Made

- **Hardware**: Pi 4 for testing (using spares)
- **Interface**: Simple web browser (Mac/Firefox/Chrome compatible)
- **Backup frequency**: Weekly manual backups (low change rate expected)
- **Cloud storage**: Dropbox primary, VPS secondary
- **Family access**: Browser-based, no complex training needed

---

🚀 Next Steps

1. Review architecture decisions
2. Define data structure and types
3. Choose web application framework
4. Set up development environment
5. Build minimal viable system
6. Test recovery procedures
7. Create family documentation

This document serves as the master reference for the LifeBox digital legacy system. All implementation decisions should align with the goals and constraints outlined above.