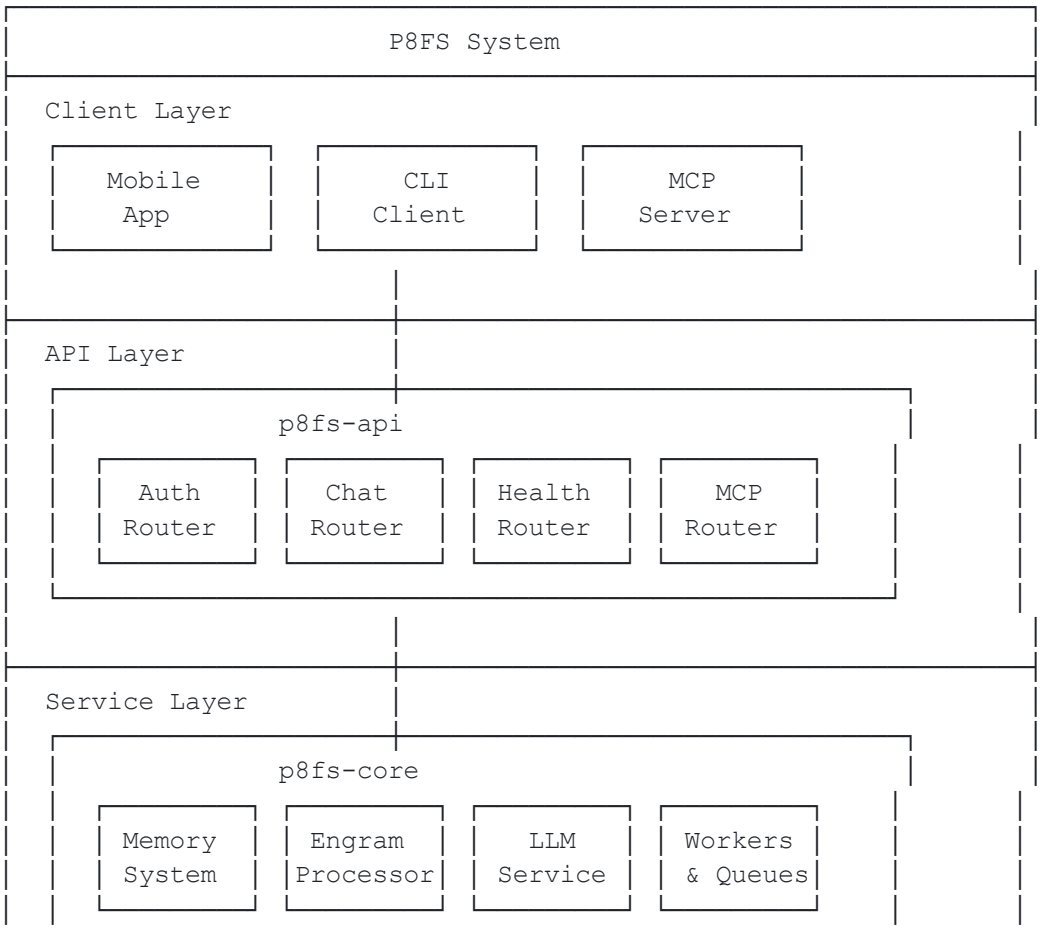
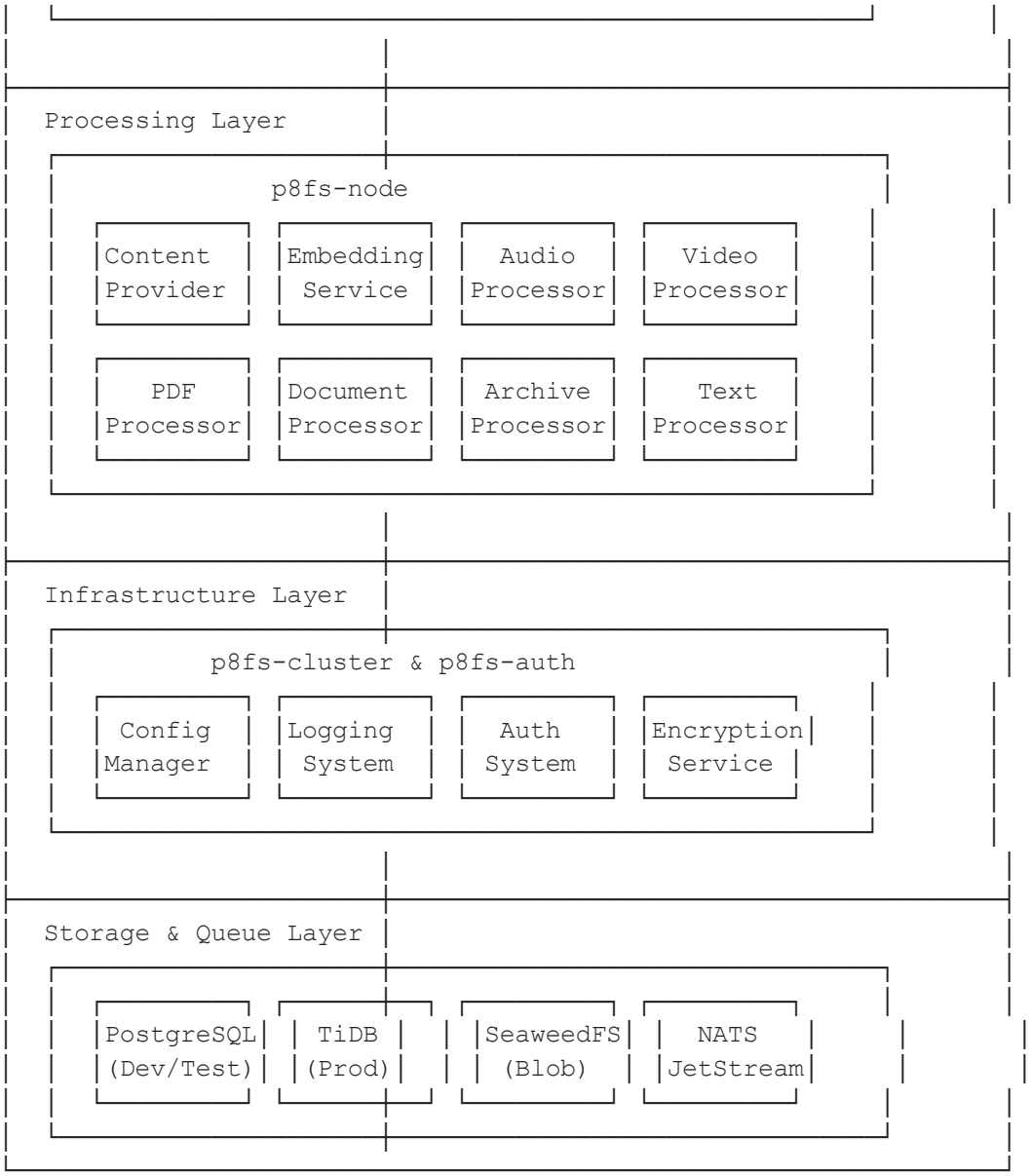
P8FS - Next Generation Smart Content Management System

P8FS is a distributed content management system designed for secure, scalable storage with advanced indexing capabilities. The system leverages S3-compatible blob storage (SeaweedFS) and TiDB/TiKV for managing a secure "memory vault" where users can upload and manage content with end-to-end encryption.

System Architecture





Module Architecture

📁 p8fs-api

FastAPI, MCP and CLI interface to the entire system. Provides RESTful endpoints, streaming chat interfaces, and Model Context Protocol server for IDE integration.

Key Components:

- Auth router with OAuth 2.1 implementation
- Chat router with streaming LLM endpoints
- MCP server for development tooling
- Health monitoring and metrics

📁 p8fs-core

The percolate memory system core that handles RAG/IR features, database repositories, and content indexing. Supports both PostgreSQL (dev/test) and TiDB (production) backends.

Key Components:

- Memory management with vector/graph indexing
- Engram processor for content chunking
- LLM service abstractions
- Background workers and job queues
- Repository layer with multi-database support

📁 p8fs-node

Content processing engine with dual Python/Rust implementation. Handles file format conversion, embedding generation, and content transformation.

Key Components:

- Content provider registry (PDF, audio, video, documents)
- Embedding service
- Multi-format processors
- Rust-based high-performance components

📁 p8fs-auth

Authentication and encryption module providing mobile-first keypair generation and OAuth 2.1 token issuance with end-to-end encryption capabilities.

Key Components:

- Mobile keypair generation
- OAuth 2.1 token service
- End-to-end encryption utilities
- Public key infrastructure

📁 p8fs-cluster

Centralized configuration and runtime management for cluster deployments. Provides shared logging, environment management, and system coordination.

Key Components:

- Centralized configuration system
- Logging infrastructure
- Environment variable management
- Cluster coordination utilities

Quick Start

Prerequisites

1. Install uv (Python package manager)

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

2. `# or via pip: pip install uv`
3. Start Development Services

```
cd p8fs-core
```

4. `docker-compose up postgres -d  # Start PostgreSQL for development`

uv Workspace Setup

This project uses uv workspaces for seamless monorepo development with automatic editable installs.

1. Install All Dependencies

```
cd p8fs-modules
```

2. `uv sync  # Installs all workspace members with editable installs`
3. Run Development Servers

```
# API server with hot reload
cd p8fs-api
uv run uvicorn p8fs_api.main:app --reload

# CLI tools
uv run -p p8fs-node p8fs-node process --help
```

4. `uv run -p p8fs-auth p8fs-auth generate-keypair`
5. Run Tests

```
# Run all workspace tests
uv run pytest

# Run specific module tests
```

6. `uv run -p p8fs-core pytest tests/`

Development Benefits

With uv workspaces configured:

- ✅ Automatic editable installs: Changes in any module immediately available to dependents
- ✅ No manual reinstalls: Modify p8fs-auth models → instantly reflected in p8fs-api
- ✅ Hot reload support: uvicorn `--reload` detects changes across all modules
- ✅ Consistent dependencies: Single lockfile ensures version compatibility
- ✅ Fast iteration: Cross-module development without friction

Alternative Setup (Legacy)

For environments without uv workspace support:

```
# Install direnv for automatic Python path setup
brew install direnv
cd p8fs-modules

direnv allow
```

Design Principles

- Separation of Concerns: Each module handles a single responsibility
- Security First: End-to-end encryption with client-held keys

- Minimal Code: Lean implementations avoiding complexity
- Testability: Unit tests with mocks and integration tests with real services
- Scalability: Horizontal scaling through KEDA and distributed storage
- Clean Architecture: Well-defined interfaces between components

Getting Started with AI Agents

🚀 New to P8FS? Start here! Check out our comprehensive getting-started guide:

```
cd p8fs-modules
```

```
uv run jupyter notebook getting_started.ipynb
```

The notebook demonstrates:

- Creating AI agents with function calling (weather agent example)
- Using MemoryProxy in normal, streaming, and batch modes
- Detailed CallingContext configurations
- Advanced function registration patterns
-
- Complete working examples with mock implementations