AWS Infrastructure and Observability Setup Report

Just Vault: Building a Secure, Scalable Document Vault

Date: January 31, 2025
Account ID: 491085415425
Region: us-east-1
Project: Just Vault

Executive Summary

This report documents the comprehensive AWS infrastructure setup for Just Vault, a local-first encrypted document vault application for iOS. The infrastructure was designed with security, scalability, and cost-effectiveness as primary considerations. This document covers the creation of core AWS services, the implementation of observability tools, and an overview of the application architecture.

Part One: Core AWS Services

Infrastructure

Authentication and Access Management

The foundation of our AWS infrastructure begins with secure authentication. We implemented AWS Single Sign-On (SSO) to eliminate the security risks associated with long-lived access keys. Traditional AWS credentials pose significant security challenges: they never expire unless manually rotated, can be accidentally committed to version control systems, and provide permanent access if compromised.

AWS SSO addresses these concerns by providing temporary credentials that automatically expire every eight hours. This approach ensures that even if credentials are compromised, the potential damage is limited to a short time window. Additionally, every action performed through SSO is tied to a specific user identity, creating a comprehensive audit trail for security compliance and troubleshooting.

The SSO implementation requires users to authenticate through a browser-based interface connected to their organization's identity provider. Once authenticated, temporary credentials are automatically managed and refreshed, eliminating the need for manual credential rotation and reducing the operational overhead of security management.

Storage Infrastructure: Amazon S3

Amazon Simple Storage Service (S3) serves as the encrypted cloud mirror for user files. The S3 bucket was configured with multiple layers of security and optimization. Public access is completely blocked, ensuring that files are only accessible through authenticated AWS API calls. Server-side encryption using AES-256 is enabled by default, providing an additional layer of protection for data at rest.

The bucket structure implements per-user data isolation through prefix-based organization. Each user's files are stored under a unique prefix following the pattern `users/{identityId}/files/`, where `identityId` is the Cognito Identity Pool identifier for that user. This structure ensures that users can only access their own data through IAM policy enforcement, preventing cross-user data access even if authentication mechanisms were somehow bypassed.

Intelligent-Tiering was considered for cost optimization, which automatically moves infrequently accessed objects to cheaper storage classes. However, for the initial version, we deferred this configuration to keep the setup simple and focused on core functionality. This can be easily added later through the AWS Console when storage patterns become clearer.

Metadata Storage: Amazon DynamoDB

DynamoDB provides the metadata storage layer for Just Vault, using a single-table design optimized for efficient querying and cost management. The table structure uses a composite key design with a partition key (PK) and sort key (SK). The partition key follows the pattern `USER#{identityId}`, ensuring that all data for a specific user is stored together, enabling efficient queries and natural data isolation.

The sort key differentiates between different types of records: `PROFILE` for user profile information, `SPACE#{spaceId}` for organizational spaces, and `FILE#{fileId}` for individual file metadata. This design allows a single query to retrieve all data for a user, or more specific queries to retrieve only spaces or files as needed.

Billing mode is set to on-demand, which eliminates the need for capacity planning during the early stages of the application. On-demand pricing automatically scales with usage, charging only for the read and write units consumed. This approach provides cost predictability while avoiding the complexity of provisioning and managing read and write capacity units.

Authentication Services: Amazon Cognito

Cognito provides the authentication bridge between Apple Sign-In and AWS services. The implementation consists of two primary components: the User Pool and the Identity Pool. The User Pool is configured to accept Apple Sign-In tokens as a federated identity provider. When a user signs in with Apple, the iOS application receives an Apple ID token, which is then exchanged with Cognito for Cognito-specific tokens.

The User Pool Client is configured as a public client, meaning it does not require a client secret. This is the standard approach for mobile applications, as storing secrets in client-side applications is inherently insecure. The client supports the SRP (Secure Remote Password) authentication flow and refresh token authentication, enabling secure token management without requiring user re-authentication for extended periods.

The Identity Pool serves a different purpose: it exchanges Cognito User Pool tokens for temporary AWS credentials. These credentials are scoped to a specific IAM role that has been configured with policies restricting access to only the user's own data. This two-step process ensures that users can authenticate using their preferred method (Apple Sign-In) while still gaining secure, scoped access to AWS resources.

Security Model: IAM Roles and Policies

The IAM security model is the critical layer that ensures users can only access their own data. A dedicated IAM role, `JustVaultAuthenticatedUserRole`, is configured with two primary policies: one for S3 access and one for DynamoDB access.

The S3 access policy implements prefix-based isolation. Users can only list objects within their own prefix (`users/{identityId}/*`) and can only perform read, write, and delete operations on objects within that prefix. Attempts to access objects outside of this prefix are automatically denied by AWS, regardless of how the request is made.

The DynamoDB access policy uses a condition that restricts queries to partition keys matching the user's identity. The condition `ForAllValues:StringLike` with `dynamodb:LeadingKeys` ensures that any query must include the user's identity ID in the partition key. This prevents users from querying other users' data, even if they somehow obtained another user's identity ID.

The trust policy for the IAM role is configured to only allow assumption by the specific Cognito Identity Pool we created. This prevents the role from being assumed by other services or identity pools, further tightening the security boundary.

Part Two: Observability and

Monitoring Infrastructure

Philosophy: AWS-Native Observability

Our observability strategy is built entirely on AWS-native services, avoiding third-party monitoring tools for the initial version. This approach reduces complexity, minimizes costs, and ensures that all monitoring data remains within the AWS ecosystem, simplifying compliance and data governance.

The observability stack consists of three primary components: CloudWatch Logs for application logging, CloudWatch Metrics for custom metrics tracking, and CloudWatch Alarms for automated alerting. CloudWatch Dashboards provide visualization capabilities, enabling real-time monitoring of application health and performance.

CloudWatch Logs: Centralized Application Logging

CloudWatch Logs provides centralized storage and analysis of application logs. We created three dedicated log groups, each serving a specific purpose in the application's observability strategy.

The primary application log group, `/aws/just-vault/app`, captures general application events including user actions, feature usage, and system state changes. This log group has a retention period of seven days, which balances the need for recent log analysis with cost considerations. Most application issues can be diagnosed within this timeframe, and longer retention can be configured if needed for specific compliance requirements.

The sync operation log group, `/aws/just-vault/sync`, specifically tracks file synchronization events between the local device and cloud storage. This includes upload successes and failures, download operations, conflict resolution, and sync performance metrics. Like the application log group, it retains logs for seven days, providing sufficient history to diagnose sync-related issues.

The errors log group, `/aws/just-vault/errors`, has an extended retention period of thirty days. This longer retention is critical for tracking error patterns over time, identifying recurring issues, and supporting post-incident analysis. Error logs include stack traces, error context, user identifiers (when appropriate for debugging), and timestamps, enabling comprehensive error analysis.

Logging Implementation Strategy

The iOS application implements a buffered logging approach to optimize network usage and ensure reliable log delivery. Logs are collected locally in a buffer, and when the buffer reaches a threshold (typically 50 entries), the logs are uploaded to CloudWatch in a single batch operation. This approach reduces the number of API calls, improves battery life on mobile devices, and ensures that logs are delivered even if the device temporarily loses network connectivity.

The logging service is designed to be non-blocking, ensuring that logging operations do not impact application performance. Logs are structured in JSON format, making them easy to parse and analyze using CloudWatch Logs Insights or other log analysis tools. Each log entry includes a timestamp, log level, event type, optional user identifier, and contextual metadata specific to the event being logged.

CloudWatch Metrics: Custom Application Metrics

CloudWatch Metrics enables tracking of custom application metrics that are critical for understanding user behavior, system performance, and business outcomes. We track metrics across several categories: authentication events, file operations, synchronization performance, and subscription management.

Authentication metrics include successful sign-ins, failed authentication attempts, token refresh operations, and credential issuance events. These metrics help identify authentication issues, detect potential security threats such as brute-force attacks, and monitor the health of the authentication flow.

File operation metrics track file imports, exports, deletions, and previews. We also track the size of imported files and the duration of import operations, enabling performance optimization and capacity planning. These metrics help identify which file types are most commonly used, average file sizes, and import performance trends.

Synchronization metrics monitor the success and failure rates of sync operations, as well as the duration of sync processes. This data is essential for identifying sync performance issues, understanding network impact on user experience, and optimizing the sync algorithm for better reliability and speed.

Subscription metrics track conversion rates, subscription activations, cancellations, and renewals. These metrics are critical for business intelligence, enabling analysis of monetization effectiveness and identification of opportunities to improve conversion rates.

Metrics Implementation and Cost Optimization

Custom metrics are implemented using a buffering strategy similar to logging. Metrics are collected locally and uploaded in batches to minimize API calls and reduce costs. CloudWatch provides a free tier that includes ten custom metrics per month, which is sufficient for the initial version of the application. As the application scales, additional metrics can be added with minimal cost impact, as CloudWatch charges only $0.30 per metric per month beyond the free tier.

The metrics namespace is set to "JustVault" to organize all application metrics together. This enables easy filtering and querying of metrics specific to our application, and supports the use of CloudWatch Dashboards to visualize metrics in a cohesive manner.

CloudWatch Alarms: Automated Alerting

CloudWatch Alarms provide automated alerting when specific conditions are met. We configure alarms for critical system health indicators, cost thresholds, and security events. Alarms can trigger notifications through Amazon Simple Notification Service (SNS), enabling email, SMS, or integration with other notification systems.

Critical alarms include high error rates, authentication failure spikes, and sync operation failures. These alarms help identify issues before they significantly impact users, enabling proactive problem resolution. Cost-related alarms monitor estimated daily costs, alerting when spending exceeds expected thresholds, which is essential for budget management and cost optimization.

Informational alarms track positive events such as new user registrations, providing visibility into growth trends. While these alarms may not require immediate action, they provide valuable business intelligence and help identify successful marketing campaigns or feature launches.

IAM Permissions for Observability

To enable the iOS application to send logs and metrics to CloudWatch, we added a CloudWatch access policy to the authenticated user IAM role. This policy grants permissions to create log groups and streams, put log events, and publish custom metrics. The policy is scoped to only allow access to our application's log groups and metrics namespace, following the principle of least privilege.

The CloudWatch permissions are integrated into the same IAM role used for S3 and DynamoDB access, ensuring that users have all necessary permissions for application operation without requiring multiple role assumptions or complex permission management.

Part Three: Application Overview

Just Vault: Purpose and Vision

Just Vault is a local-first, encrypted personal document vault designed for iOS devices. The application addresses a critical need in the digital age: secure, private storage of sensitive documents with the convenience of cloud backup and multi-device synchronization. Unlike traditional cloud storage services, Just Vault implements a zero-knowledge architecture, meaning that the service provider (and by extension, AWS) never has access to unencrypted user data.

The application's core promise is simple: your documents are encrypted on your device before they ever leave your phone, stored locally for instant access, and backed up to the cloud for device loss recovery. This architecture ensures that even if cloud storage were compromised, attackers would only have access to encrypted data that they cannot decrypt without the user's master key, which is stored securely in the device's Secure Enclave.

Three-Layer Architecture

Just Vault implements a three-layer storage architecture that balances performance, security, and reliability. The first layer is the local encrypted vault stored on the user's iOS device. This is the primary storage location, providing instant access to files without requiring network connectivity. Files are encrypted using AES-256-GCM encryption via Apple's CryptoKit framework, ensuring military-grade security for stored data.

The master encryption key is stored in the device's Secure Enclave, a hardware-isolated security processor that provides additional protection against key extraction. The Secure Enclave ensures that even if the device is compromised at the operating system level, the master key remains protected by hardware-level security mechanisms.

The second layer is the encrypted cloud mirror stored in Amazon S3. Files are encrypted on the device before upload, ensuring that AWS never sees plaintext data. This layer serves as insurance against device loss, theft, or damage. Users can recover their encrypted files to a new device by authenticating and downloading their data from S3.

The third layer is metadata synchronization via DynamoDB. This layer stores file metadata such as names, sizes, creation dates, and organizational information (spaces), but never stores file content. This enables features like multi-device synchronization, where a user can see their file list across devices without needing to download all files to each device.

User Experience and Interface Design

The application's user interface is designed around simplicity and visual clarity. The main screen, called the Vault Home, presents a unique "flower" or "petal" layout where organizational spaces are displayed as circular bubbles arranged around a central vault core. This design makes it easy for users to visually organize their documents into categories such as Health, Work, Business, or Personal.

Each space can be customized with a name, icon, and color, enabling users to create a personalized organizational system. The flower layout provides an intuitive way to see all spaces at a glance while maintaining visual appeal and making the application feel more like a personal vault than a traditional file manager.

Below the spaces, a storage meter displays cloud storage usage, showing both the amount used and the total quota. For free tier users, this meter serves as a visual reminder of storage limits and provides a clear upgrade path when approaching the 250MB free tier limit. The meter changes color as usage approaches critical thresholds, providing visual feedback about storage status.

Security and Privacy Features

Security is built into every layer of the application architecture. The zero-knowledge encryption model ensures that user data is encrypted before it leaves the device, providing protection against both external attackers and the service provider itself. The use of industry-standard encryption algorithms (AES-256-GCM) and secure key derivation methods ensures that the encryption implementation meets enterprise security standards.

The BIP39 recovery phrase system provides users with a way to recover their data if they lose access to their device. The recovery phrase consists of 12 or 24 words that can be used to reconstruct the master encryption key. This system is widely used in cryptocurrency applications and provides a proven, secure method for key recovery.

IAM policies ensure that users can only access their own data, even at the infrastructure level. The combination of prefix-based S3 isolation and DynamoDB query restrictions means that even if there were a bug in the application code, users would still be unable to access other users' data due to AWS-level policy enforcement.

Monetization Strategy

Just Vault operates on a freemium subscription model, providing a free tier with limited storage and features, and a Pro tier with expanded capabilities. The free tier includes 250MB of cloud storage and allows users to create up to two organizational spaces. This tier is designed to provide value while creating a natural upgrade path when users need more storage or organizational capabilities.

The Pro tier, priced at $6.99 per month or $59.99 per year, provides 10GB of cloud storage and unlimited spaces. This tier targets power users who need more storage or want to organize documents across many categories. The pricing is competitive with similar services while maintaining healthy profit margins due to the low AWS infrastructure costs.

The conversion strategy focuses on natural usage triggers: when users approach their storage limit or want to create additional spaces, they are presented with upgrade prompts. The storage meter provides visual feedback about approaching limits, and the application makes it easy to upgrade without disrupting the user's workflow.

Technical Implementation

The application is built using SwiftUI, Apple's modern declarative UI framework, ensuring a native iOS experience that feels integrated with the operating system. The AWS SDK for Swift is used for all cloud interactions, providing type-safe APIs and automatic credential management through Cognito integration.

The encryption implementation uses Apple's CryptoKit framework, which provides hardware-accelerated cryptographic operations and ensures that encryption and decryption are performed efficiently on the device. The Secure Enclave integration uses the Security framework to store and retrieve the master key, leveraging hardware-level security features available on modern iOS devices.

Background synchronization is implemented using iOS background processing capabilities, allowing the application to sync files even when not actively in use. This ensures that users' files are backed up regularly without requiring manual intervention or keeping the application open.

Conclusion

The AWS infrastructure for Just Vault represents a comprehensive, security-focused architecture designed for scalability and cost-effectiveness. The use of AWS SSO for administrative access, combined with Cognito for user authentication, provides multiple layers of security while maintaining operational simplicity.

The observability implementation ensures that we can monitor application health, diagnose issues quickly, and understand user behavior patterns. The AWS-native approach keeps costs low while providing all necessary monitoring capabilities.

The application itself represents a modern approach to secure document storage, prioritizing user privacy through zero-knowledge encryption while providing the convenience of cloud backup and multi-device synchronization. The three-layer architecture balances performance, security, and reliability, ensuring that users have instant access to their data while maintaining robust backup capabilities.

As the application grows, the infrastructure is designed to scale automatically, with on-demand pricing models that adjust to usage patterns. The modular architecture allows for easy addition of new features and services as requirements evolve, ensuring that the infrastructure can support the application's growth from initial launch through scale.

Report prepared for: Just Vault Development Team
Infrastructure Status: Production Ready
Next Steps: Implement Cognito token exchange, complete iOS application development, conduct security audit