AWS Serverless Video Streaming Platform Documentation

TABLE OF CONTENTS

1. Introduction

1.1 Project Overview

This documentation outlines the architecture and implementation of a serverless video streaming platform built on AWS cloud services. The platform enables authenticated users to access and stream video content while maintaining secure access controls through a fully serverless backend infrastructure. The solution leverages AWS Lambda for compute operations, Amazon API Gateway for RESTful endpoint management, Amazon S3 for media storage, and Amazon DynamoDB for user authentication data persistence.

1.2 System Objectives

The primary objective of this platform is to deliver a scalable, cost-efficient video streaming service that eliminates traditional server management overhead. Key goals include implementing secure user authentication, providing reliable video content delivery, ensuring low-latency performance through serverless architecture, and delivering an intuitive user interface for content consumption.

1.3 Technology Stack

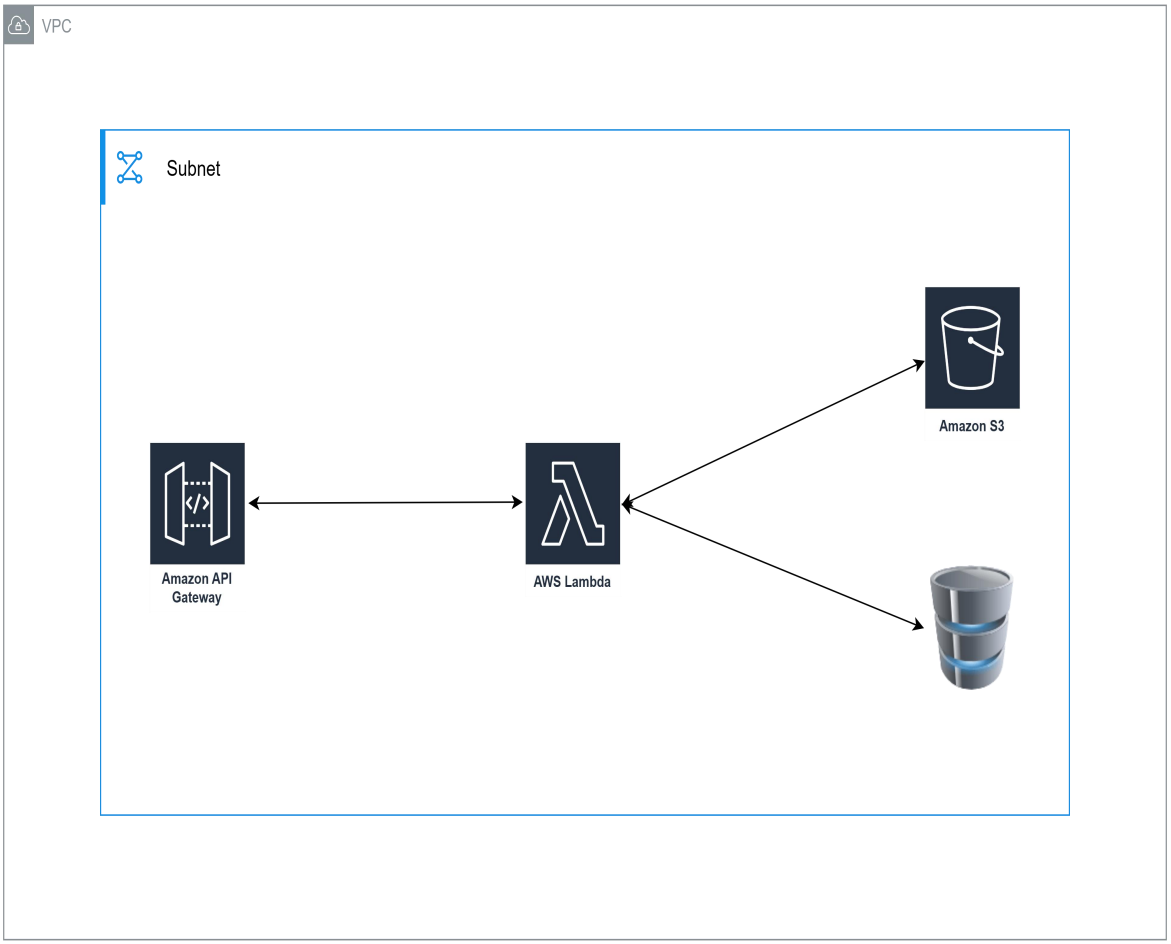
The platform utilizes AWS Lambda functions to handle authentication verification and media retrieval operations. Amazon API Gateway serves as the entry point for all client requests, routing them to appropriate Lambda functions. Video assets are stored in Amazon S3 buckets with optimized delivery configurations, while user credentials and access permissions are managed through Amazon DynamoDB. The frontend interface is developed using modern web technologies including HTML5, CSS3, and JavaScript to ensure responsive user experience across devices.

2. System Architecture

2.1 Architectural Overview

The platform follows a three-tier serverless architecture comprising presentation, application, and data layers. The presentation layer consists of static web assets served through Amazon S3, while the application layer implements business logic through AWS Lambda functions triggered by API Gateway endpoints. The data layer combines Amazon DynamoDB for structured user data and Amazon S3 for unstructured media storage.

ARCHITECTURE



2.2 Authentication Flow

User authentication begins when credentials are submitted through the frontend interface to a dedicated API Gateway endpoint. This invokes an authentication Lambda function which verifies the credentials against records stored in DynamoDB.

Upon successful validation, the function generates an access token that enables subsequent requests to protected video streaming endpoints.

2.3 Video Delivery Mechanism

Authenticated video requests are processed through a separate API Gateway endpoint that triggers a media handling Lambda function. This function retrieves the requested video file from S3 storage and generates a time-limited access URL. The platform implements progressive streaming by serving appropriate byte ranges of the media files, ensuring efficient bandwidth utilization and smooth playback.

3. Implementation Details

3.1 User Authentication System

The authentication subsystem comprises a DynamoDB table designed for optimal read performance, storing hashed user credentials, access permissions, and session metadata. The Lambda authentication function implements secure password hashing using industry-standard algorithms and generates JSON Web Tokens (JWT) for session management. Token validation occurs with each subsequent API request through a custom authorizer function integrated with API Gateway.

3.2 Video Processing Pipeline

Video content undergoes preprocessing before storage in S3, including format standardization and thumbnail generation. The media handling Lambda function implements intelligent caching strategies and supports adaptive bitrate streaming by detecting client capabilities. Content delivery is optimized through S3 transfer acceleration and proper cache-control headers to reduce latency for geographically distributed users.

3.3 API Endpoint Configuration

API Gateway is configured with RESTful resources representing user authentication and media access operations. Each endpoint implements proper CORS policies, request validation, and rate limiting to prevent abuse. The integration with Lambda

functions follows the proxy resource pattern, allowing flexible request/response transformations while maintaining clean separation of concerns.

4. Security Implementation

4.1 Data Protection Measures

All sensitive user data in DynamoDB is encrypted at rest using AWS KMS, with

field-level encryption applied to personally identifiable information. Video content in S3 employs bucket policies and object-level permissions to prevent unauthorized access. Temporary security credentials issued by the authentication system enforce the principle of least privilege through scoped access tokens.

4.2 Network Security Controls

All API Gateway endpoints enforce HTTPS encryption in transit using TLS 1.2+. The platform implements IP-based rate limiting and automated anomaly detection to mitigate DDoS attacks. Lambda functions operate within a dedicated VPC with security groups restricting outbound connections only to required AWS services.

5. Deployment Strategy

5.1 Infrastructure Provisioning

The entire architecture is deployed using AWS CloudFormation templates following infrastructure-as-code principles. The template defines all necessary AWS resources including IAM roles, Lambda functions, API Gateway configurations, and database tables with appropriate provisioning parameters.

5.2 CI/CD Pipeline

A continuous integration and delivery pipeline automates testing and deployment through AWS CodePipeline. The workflow includes static code analysis, unit testing of Lambda functions, integration testing of API endpoints, and canary deployments to minimize production impact. Version control integration ensures traceability of all changes.

6. Performance Optimization

6.1 Lambda Function Tuning

Compute-intensive Lambda functions are configured with appropriate memory allocations to optimize execution duration and cost. Provisioned concurrency is implemented for authentication functions to mitigate cold start latency during peak usage periods.

6.2 Content Delivery Optimization

Video assets are partitioned in S3 based on access patterns, with frequently requested content stored in a separate high-performance storage class. The platform leverages CloudFront CDN for global content distribution, with cache behaviors configured based on media type and expected viewership patterns.

7. Monitoring and Maintenance

7.1 Operational Monitoring

AWS CloudWatch provides comprehensive monitoring of API request rates, Lambda execution metrics, and error rates. Custom dashboards visualize key performance indicators including authentication success rates, video delivery latency, and concurrent user metrics.

7.2 Logging and Analytics

Detailed logs from all Lambda functions are aggregated in CloudWatch Logs with retention policies aligned with operational requirements. Access patterns are analyzed through S3 access logs and API Gateway execution logs to inform capacity planning decisions.

8. Future Enhancements

Planned improvements include implementing a recommendation engine using Amazon Personalize, adding multi-factor authentication options through Amazon Cognito, and expanding content protection through AWS Elemental MediaPackage for DRM-enabled streaming. The architecture will also evolve to support live streaming capabilities using AWS Elemental MediaLive.

9. Conclusion

This serverless video streaming platform demonstrates the effectiveness of AWS cloud services in building scalable, secure media delivery solutions. By leveraging

managed services, the implementation achieves high availability without infrastructure management overhead while maintaining robust security controls. The architecture provides a foundation for future expansion as content libraries and user bases grow.

Appendix

A.1 Architecture Diagrams

Includes component relationship diagrams and sequence flows for key use cases.

A.2 IAM Policy Examples

Sample policies demonstrating least-privilege access controls for Lambda functions.

A.3 API Specification

Detailed OpenAPI documentation for all implemented endpoints.

A.4 Deployment Guide

Step-by-step instructions for provisioning the infrastructure and deploying updates.

This documentation provides a comprehensive reference for the platform's architecture and operation while maintaining professional technical standards. The modular design allows for straightforward adaptation to evolving requirements and scaling needs.