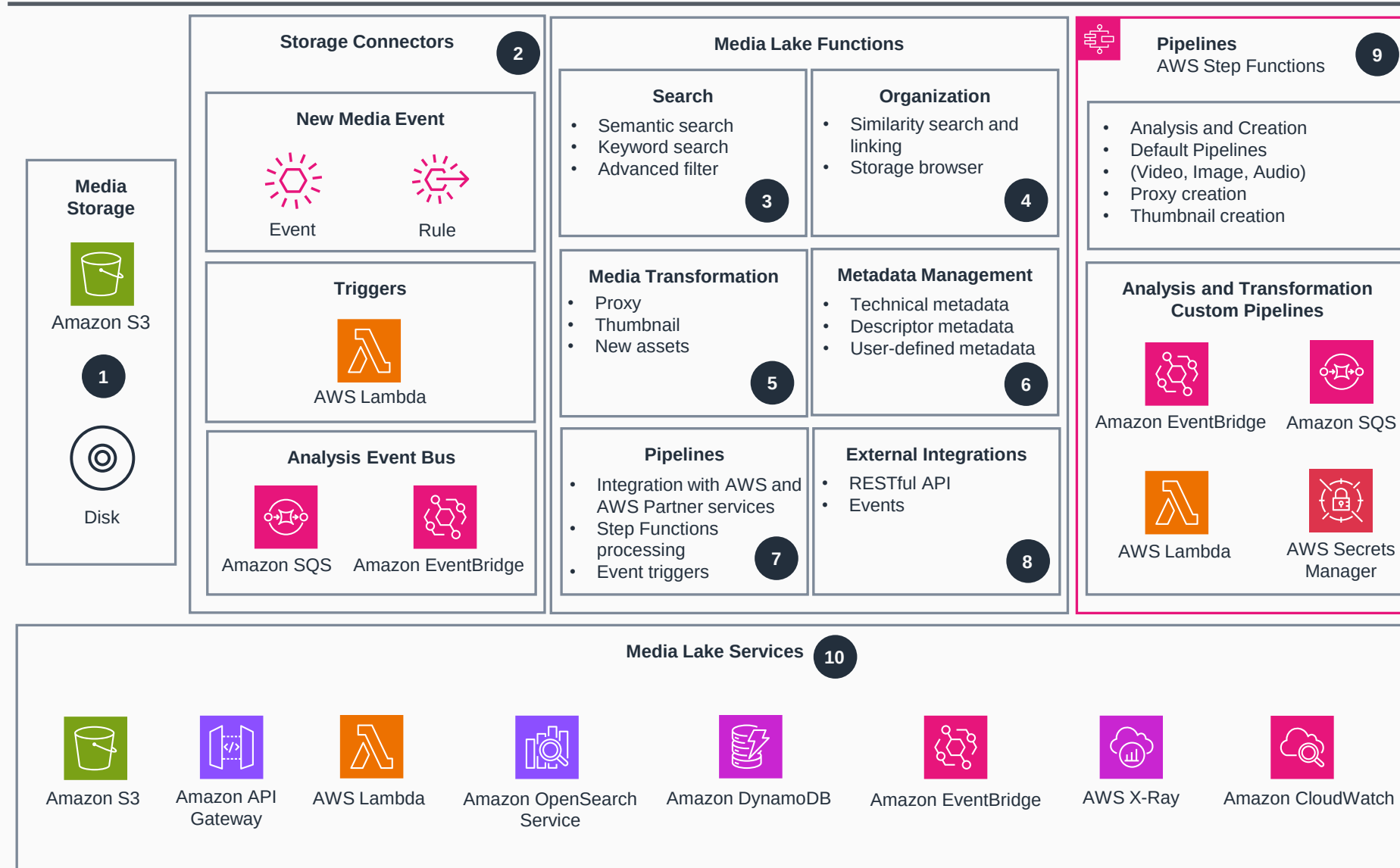


Guidance for a Media Lake on AWS

Overview

This architecture diagram provides a functional overview of the capabilities of a media lake on AWS.



- 1 Upload new media files to **Amazon Simple Storage Service (Amazon S3)**. Upload triggers an event to initiate processing.
- 2 **AWS Lambda**, **Amazon Simple Queue Service (Amazon SQS)**, and **Amazon EventBridge** coordinate the flow of events after ingestion. **Lambda** functions handle initial processing, and **EventBridge** routes events to transformation, enrichment, and pipeline components.
- 3 Search features support semantic and keyword search in addition to filtering of indexed assets.
- 4 Organization logic groups related assets using metadata or similarity scoring. A storage browser is used to explore assets in the connector.
- 5 Media transformation creates proxies, thumbnails, or derivative assets when triggered.
- 6 Metadata management extracts technical- and user-defined metadata to support powerful search and discovery.
- 7 Default or custom pipelines coordinate analysis, enrichment, and transformation using AWS and partner services.
- 8 RESTful APIs and workflow and API events enable integration with external systems, allowing ingestion, search, and asset and metadata retrieval.
- 9 **Lambda** and **EventBridge** coordinate the execution of custom analysis and transformation pipelines, accessing credentials in **AWS Secrets Manager** enabling secure workflows.
- 10 **Amazon S3**, **Amazon API Gateway**, **Lambda**, **Amazon OpenSearch Service**, **Amazon DynamoDB**, **EventBridge**, **Amazon SQS**, **Amazon CloudWatch**, and **AWS X-Ray** power media lake functions.



Reviewed for technical accuracy June 30, 2025

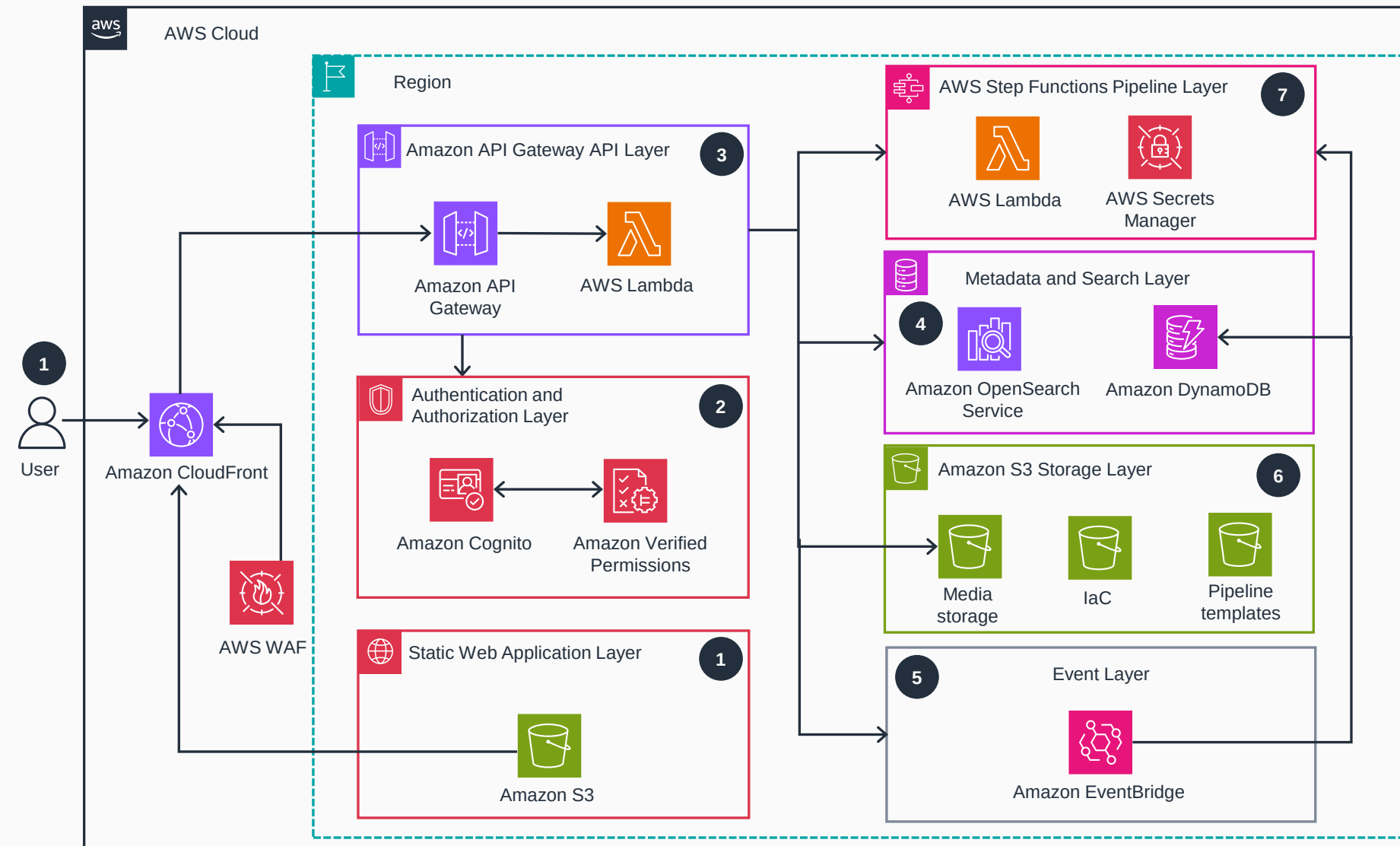
© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Reference Architecture

Guidance for a Media Lake on AWS

High-level application architecture

This architecture diagram shows the high-level API, storage, and back-end architecture of a media lake on AWS.



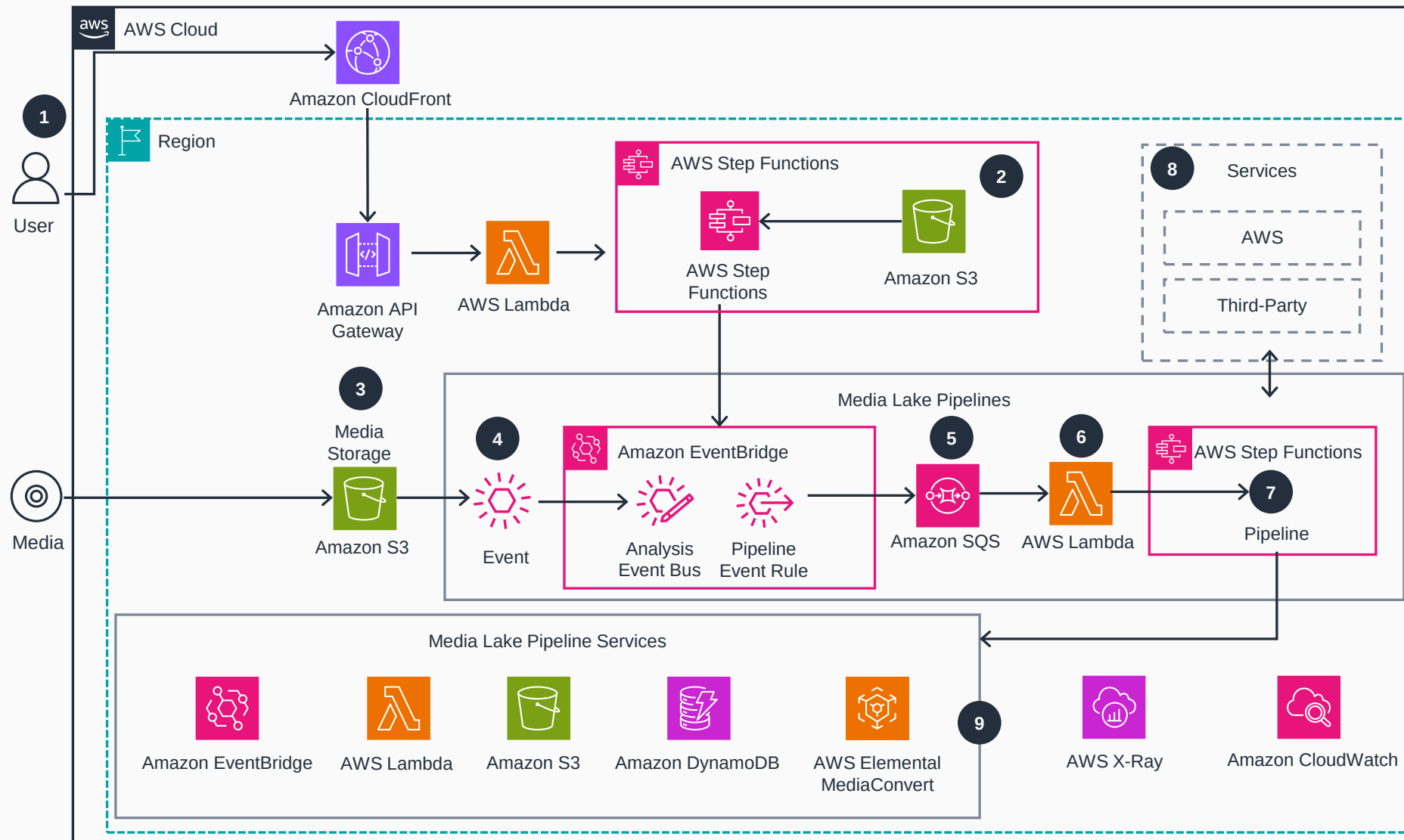
- 1 Operators access the media lake user interface through **Amazon CloudFront** with protection provided by **AWS WAF**. **CloudFront** serves the static web application from **Amazon S3**.
- 2 **Amazon Cognito** performs user authentication with authorization managed through **Amazon Verified Permissions**.
- 3 **API Gateway** routes authenticated requests, which are processed by **Lambda** functions that invoke backend services as needed.
- 4 **Lambda** queries **OpenSearch Service** to return search and retrieval results.
- 5 **EventBridge** receives internal events from the media lake through its API layer and pipeline layer, powering downstream processes such as pipeline execution, audit logging, and compliance tracking.
- 6 **Amazon S3** stores media files and assets in the Storage Layer, while **DynamoDB** stores metadata. This layer also includes infrastructure as code (IaC) and pipeline templates to enable scalable, reusable workflows.
- 7 **EventBridge** triggers pipelines upon receiving events. These pipelines pull media from **Amazon S3**, metadata from **Amazon DynamoDB**, and credentials from **Secrets Manager**. **Lambda** functions carry out operations such as proxy generation, embedding generation, and media enrichment, all orchestrated through **Step Functions**.



Guidance for a Media Lake on AWS

Pipeline execution and deployment

This architecture diagram shows the deployment and execution of pipelines used in a media lake to process media and produce metadata to aid search and render new versions for use with downstream systems.



- 1 Users define media processing workflows, through a no-code drag-and-drop canvas, save them, and deploy them as pipelines.
- 2 **Lambda** sends requests to **Step Functions**, accessing IaC in **Amazon S3**.
- 3 **Amazon S3** generates event notifications when new media is uploaded, which are copied and sent to the media lake analysis event bus.
- 4 The media lake creates **EventBridge** event rules that trigger pipelines based on new asset events or the completion of previous pipelines.
- 5 **Amazon SQS** queues incoming events, allowing them to be buffered and processed asynchronously.
- 6 **Lambda** handles events from the queue and triggers the **Step Functions** that represent deployed pipelines.
- 7 **Step Functions** define each pipeline as an individual state machine, executing the logic configured in the canvas.
- 8 **Step Functions** enable pipelines to integrate with AWS services, AWS internal software vendor (ISV) partners, or third-party systems as needed.
- 9 **Step Functions** coordinates the entire pipeline, reading media from **Amazon S3**, invoking **Lambda** (monitored through **CloudWatch** and **X-Ray**) to extract metadata and write it to **DynamoDB**, and finally, using **AWS Elemental MediaConvert** to generate proxies. It then stores outputs back in **Amazon S3**.

