6 ways to build with Amazon OpenSearch Service

Securely unlock real-time search, monitoring, and analysis of operational data with Amazon OpenSearch Service

Table of contents

What is Amazon OpenSearch Service

Amazon OpenSearch Service is a managed service that makes it easy for you to perform website search, interactive log analytics, real-time application monitoring, and more. Based on the open source platform, OpenSearch, Amazon OpenSearch Service, allows you to search, visualize, and analyze up to petabytes of text and unstructured data. Amazon OpenSearch Service provides integrations with other AWS services and a choice of open source engines, including OpenSearch and ALv2 Elasticsearch. Instead of handling your own cluster management, sizing, scaling, optimizing, patching, and hardware management, Amazon OpenSearch Service delivers a managed Elasticsearch or OpenSearch option.

Whether you need to enable search or analyze large volumes of streaming data, Amazon OpenSearch Service can help you get started fast. At a high level, use cases for Amazon OpenSearch Service fall into two groups: search and streaming data.

Enable seamless, personalized search: Help users quickly find relevant data with a fast, personalized search experience within your applications, websites, and data lake catalogs. The use cases included in this group include:

- Website search
- Application search
- Document repository search
- Semantic search
- Retrieval augmented generation

Enable large-scale logging and observability: Monitor your infrastructure, application performance, and business KPIs with state-of-the-art observability tools. The use cases included in this group include:

- Centralized log analytics
- Observability
- Trace analytics with OpenTelemetry
- Log analytics with open source
- Security analytics
- Anomaly detection

This ebook shows how you can take on various Amazon OpenSearch Service use cases in these two groups using only slightly different architectures.

Search use cases with Amazon OpenSearch Service

Adding search functionality to your site, application, or document repository empowers users to find what they need simply by typing into a search bar.

The challenge is making sure your search can keep up with the volume of queries during peak traffic times and still return results fast. While open source search engines deliver full-text search at high speeds, the infrastructure management can be time consuming. Scale servers up and down to accommodate processing needs, as well as cluster management, optimizing, patching, and hardware management, can pull engineers away from focusing on their applications.

With Amazon OpenSearch Service, that heavy lifting is taken care of—and usually for less cost than on-premises infrastructure. The service then uses machine learning (ML) algorithms to rank results and serve up the most relevant resources for users.

Website search

Build out your website search with Amazon OpenSearch Service to take in large volumes of data, index it, ingest it, and return results at high speed.

Application search

Help your customers find what they need by integrating fast, scalable full-text search capabilities. Amazon OpenSearch Service delivers search functionality for databased-backed applications to power interactive customer experiences. In this case, the search engine mirrors the content of the database and provides ML-powered ranking for relevant results.

Document repository search

Manage growing analytics costs for hot, UltraWarm, and cold tiers. Through Amazon OpenSearch Service, you can search across a large collection of text contained in a data lake or internal wiki to find the right information fast. This allows you to optimize time and resources for strategic work.

Semantic search

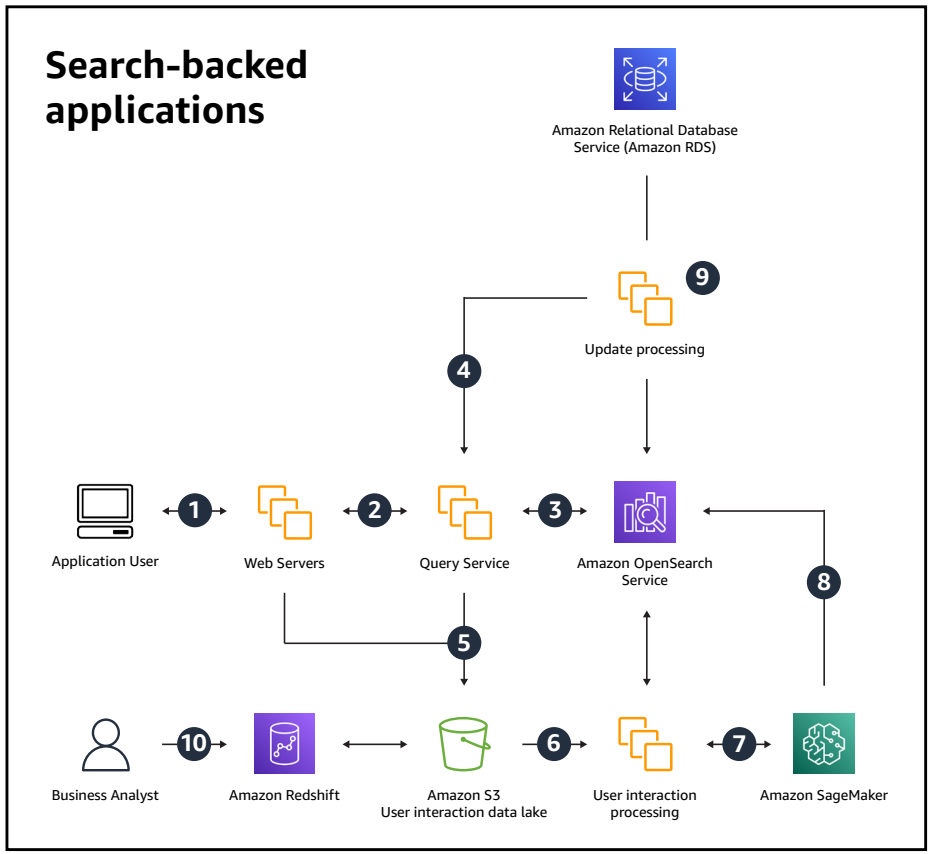
Connect search intent and context to match queries with the most relevant content. Using Amazon OpenSearch Service, you can embed documents and queries into a semantic high-dimension vector space. This allows you to create a semantic search that returns similar items even if they don't share any words with the query.

Retrieval augmented generation

Retrieval augmented generation combines relevant context from Amazon OpenSearch Service with powerful text generation models in Amazon SageMaker or Bedrock. RAG enables you to retrieve data from outside a foundation model and augment your prompts by adding the relevant retrieved data in context to avoid Large Language Model hallucination. You can leverage this approach to enhance the quality and relevance of generated text, improving the user experience in generative AI applications.

Architecture: Search-backed applications

This reference architecture outlines the process to add or improve search for an existing application.

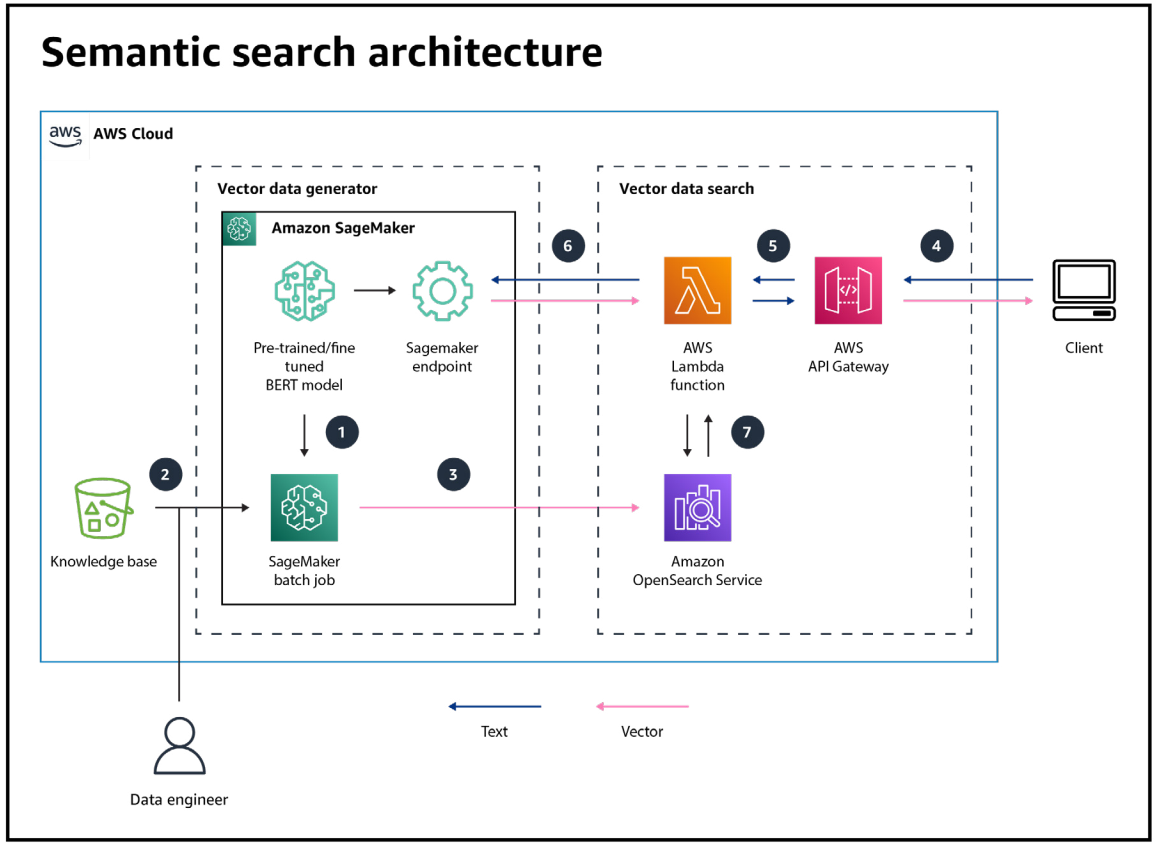


AWS Reference Architecture

1. A user sends a query.
2. The web servers deliver to the query service. At this point, the query service can employ ML models via Amazon SageMaker (arrow not shown) for user segmentation, concept and entity extraction, query-to-click, and other data to enrich the query.
3. The query service enriches, or rewrites the query, based on user segmentation from Amazon SageMaker (line not shown), user preferences from Amazon Relational Database Service (Amazon RDS), and past query performance. It sends the augmented query to Amazon OpenSearch Service.
4. The user sends only searchable data to Amazon OpenSearch Service, employing a relational or NoSQL system as the system of record. The query service retrieves only keys in the search results. It retrieves the full record information from the system of record.
5. The web servers and query service send user interaction data back to an Amazon Simple Storage Service (Amazon S3) data lake or Amazon Redshift.
6. An offline process pulls user interaction from the data lake.
7. It takes any data, e.g. clicks, it needs to augment the records in the catalog, and updates ML relevance models in Amazon SageMaker.
8. Updating records in Amazon OpenSearch Service as needed.
9. Either the web servers are sending catalog updates to Amazon OpenSearch Service, or the user is running change data capture to bring those updates to Amazon OpenSearch Service. (There could also be a separate inventory or other system that holds data for result enrichment.)
10. Business analysts generate reporting, KPIs, etc. from the processed user interactions.

Architecture: Semantic search

This reference architecture outlines the process to create semantic search.

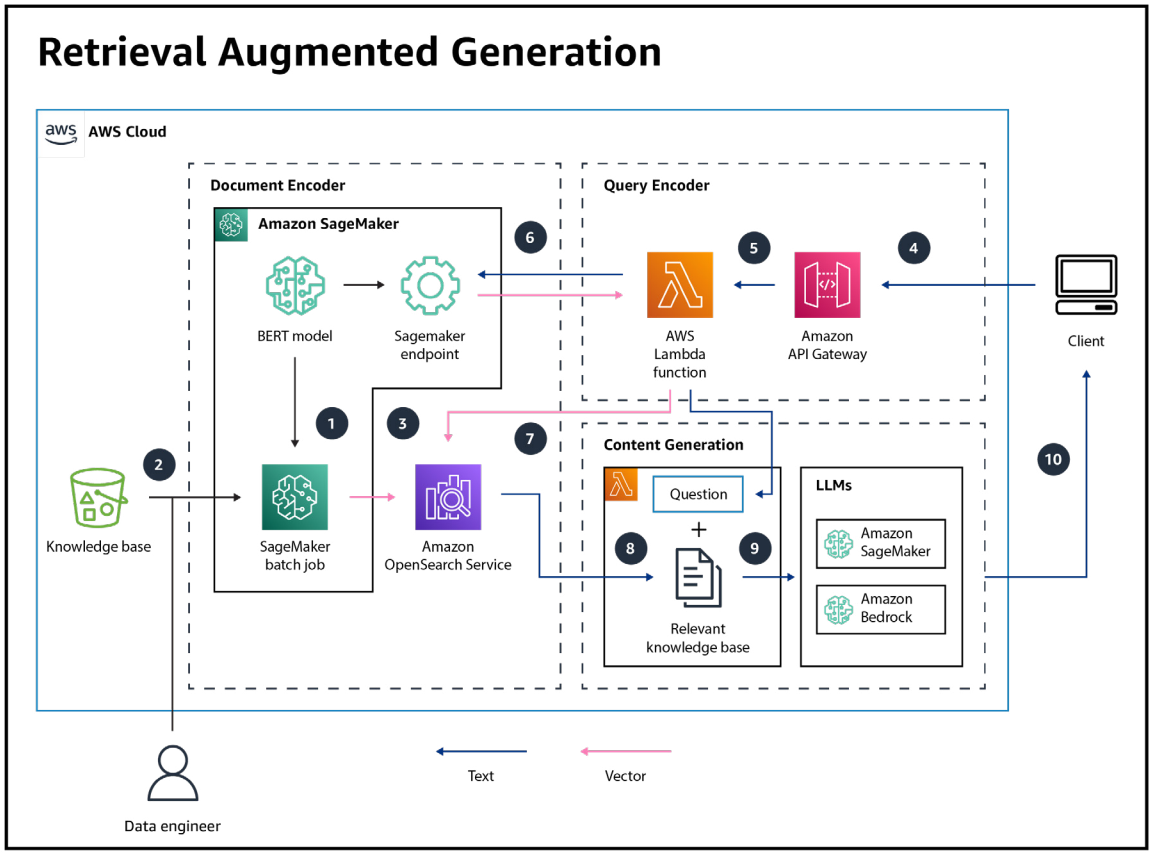


AWS Reference Architecture

1. Load pre-trained or fined tuned BERT models into Amazon SageMaker.
2. Run an Amazon SageMaker batch job to generate a vector for business documents with BERT.
3. Store vector data into Amazon OpenSearch Service.
4. The client submits a search request to Amazon API Gateway.
5. There is an API call to a backend service in AWS Lambda.
6. After a backend service call to an Amazon SageMaker endpoint, the search query is converted
7. Use the k-nearest neighbor (k-NN) algorithm for Amazon OpenSearch Service to get semantic-similar documents that are returned to the client.

Architecture: Retrieval augmented generation

This reference architecture outlines the process for developing a retrieval augmented generation application.



AWS Reference Architecture

1. Load BERT model into SageMaker batch job.
2. Load your knowledge base.
3. Use an Amazon SageMaker language model to generate embeddings for your knowledge base.
4. The client submits one question.
5. An API call is made to a backend service in AWS Lambda.
6. After a backend service call to an Amazon SageMaker endpoint, the search query is converted into a vector.
7. Use Amazon OpenSearch Service vector search to get relevant documents.
8. Return a relevant knowledge base to the backend service.catalog, and updates ML relevance models in Amazon SageMaker.
9. The backend uses the relevant knowledge base as context, combining the user's original question as a prompt to large language models.
10. Foundation models generate factual answers based on relevant knowledge for the original question.

Streaming data use cases with Amazon OpenSearch Service

Streaming data, which is continuously generated from thousands of sources, can include log files generated by customers using your mobile or web applications, ecommerce purchases, in-game player activity, information from social networks, financial trading floors, or geospatial services, and telemetry from connected devices or instrumentation in data centers.

Amazon OpenSearch Service is equipped to handle streaming data from a variety of sources. In heterogenous and changing environments, developers do not have time to build and troubleshoot custom integrations to unite various data sources—it takes implementation and troubleshooting time. That's why Amazon OpenSearch Service has native integrations with Amazon S3, Amazon Kinesis Data Firehose, Amazon CloudWatch, Amazon DynamoDB, Amazon SageMaker, and AWS Key Management Service (KMS).

Centralized log analytics

All your applications, devices, and machines generate a ton of logs at high velocity, which can make it hard to keep pace and parse through everything to identify the meaningful information. The challenge is staying on top of all your disparate logs and being able to predict when a system may go down based on error messages. Whether your log data is generated by AWS cloud solutions or applications running on those services, a cost-effective, secure, scalable, and flexible service is a must. Amazon OpenSearch Service centralizes log events into a unified view and delivers a turnkey environment to begin analyzing your environments for log patterns. This is especially helpful with Internet-of-Things (IoT) devices that can generate huge volumes of logs. Amazon OpenSearch Service can enable predictive insights and operational analytics on log data.

Observability

Applications are becoming increasingly distributed and complex, with many interconnected parts that are constantly updated. Any given change can create a previously unknown type of failure. Monitoring resource usage and the status of underlying networking systems is simply not enough. It becomes imperative to not only understand what is happening, but also rectify any potential issues.

Observability and application performance monitoring (APM) tools provide a systematic way to understand the behavior of these complex systems. Developers and operators can characterize behavior by observing external outputs of a system, accelerating innovation and improving the reliability of complex systems. By analyzing the three foundational observability signals—metrics, logs (semi-structured data), and traces (flows of requests from beginning to end across all dependencies)—DevOps and site reliability engineers can isolate critical events and issues in containerized applications and microservices running anywhere. Learn more.

Trace analytics with OpenTelemetry

Trace analytics alongside log data help you both isolate the source of performance problems and diagnose their root cause. However, correlating trace data with log events often requires navigating multiple interfaces. Developers must also know exactly which visualizations to create to build monitoring views on their log data.

Through Amazon OpenSearch Service, developers and DevOps engineers can easily analyze traces and logs from a single interface, streamlining the way they find and fix performance problems in distributed applications. This helps developers and DevOps engineers figure out how their application is running and handle the tuning and debugging themselves. Learn more.

Log analytics with open source

Analyzing logs from applications and infrastructure can be a daunting task that involves consolidating data from various sources and prepping it for analysis. Still, log analytics help companies avoid risks by ensuring compliance with security and industry regulations. Analyzing logs can also help improve the user experience by highlighting performance issues. Instead of choosing a proprietary software, many companies opt for an open source solution for log analytics because they consider it less inexpensive, more secure, and more stable. Learn more.

Security analytics

Today's security teams are tasked with sorting through backlogs of security alerts to validate the integrity of their IT systems. As a security information event management (SIEM) solution, Amazon OpenSearch Service makes it easier for SecOps teams to manage security and event information. With the ability to centralize and analyze logs from disparate applications and systems across their networks, companies can implement real-time threat detection and incident management.

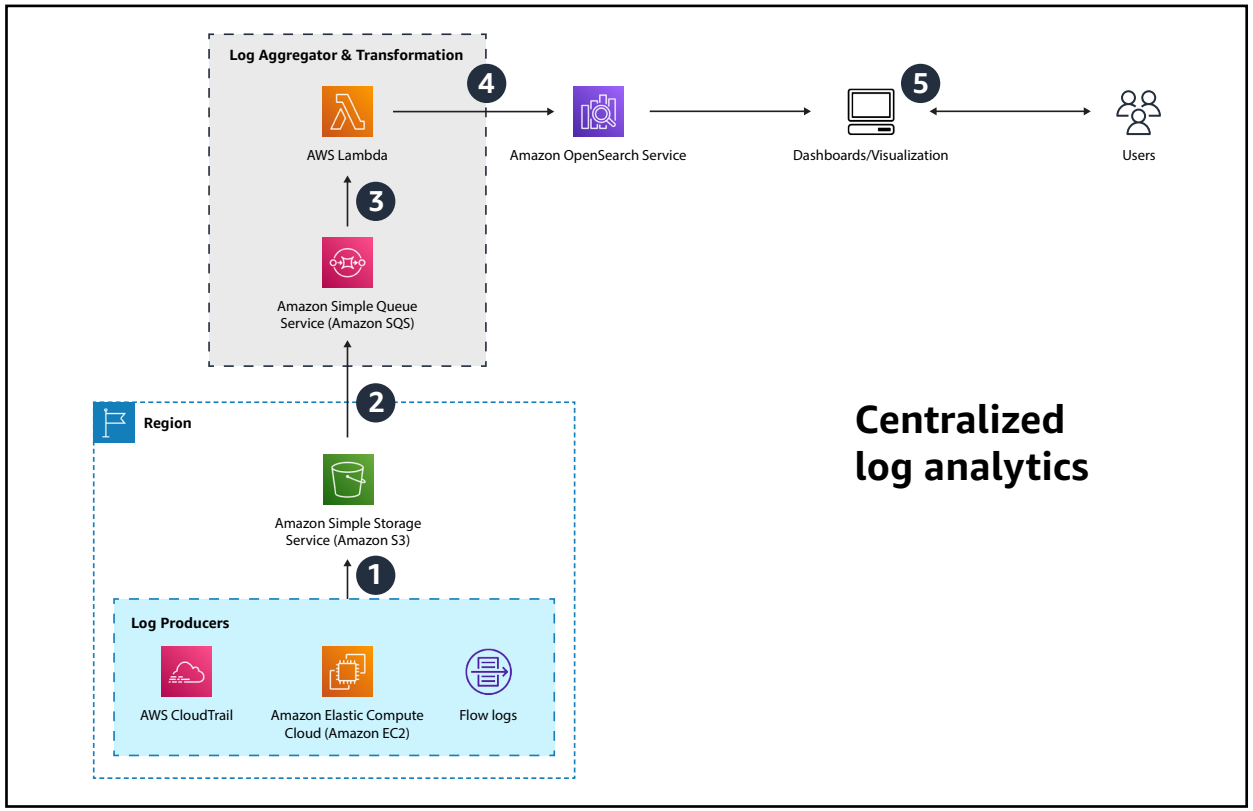
Use Amazon OpenSearch Service as a SIEM solution and integrate AWS Security Hub to store findings for longer periods of time than Security Hub, aggregate findings across multiple administrator accounts, and further correlate Security Hub findings with each other and other log sources. Learn more.

Anomaly detection

As log data grow continues to grow, DevOps teams are increasingly challenged with observability of their applications. This makes it more difficult to conduct timely root cause analyses and uncover anomalies. Anomaly detection in Amazon OpenSearch Service automatically detects anomalies in your OpenSearch data in near-real time by using the Random Cut Forest (RCF) algorithm. RCF is an unsupervised ML algorithm that models a sketch of your incoming data stream. Learn more.

Architecture: Centralized log analytics

Log analytics involves searching, analyzing, and visualizing machine data generated by IT systems and technology infrastructure to gain operational insights. The challenge is knowing which tool to choose for the job. This high-level architecture outlines the different stages of log flow and which tools could be used when.



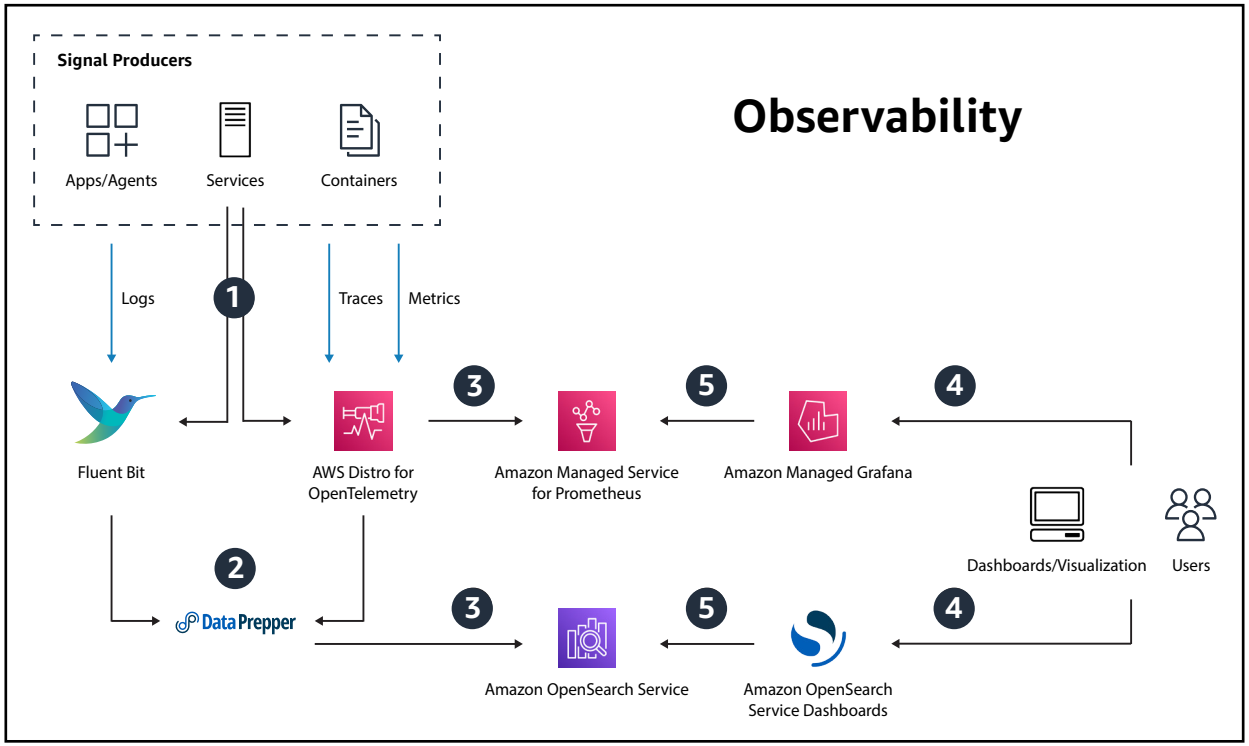
AWS Reference Architecture

1. Collectors like FluentBit, Amazon Kinesis Agent, and Amazon CloudWatch Agent or services like AWS CloudTrail collect log lines and store them in Amazon Simple Storage Service (Amazon S3).
2. Amazon S3 sends an object create event to Amazon Simple Queue Service (SQS).
3. Amazon SQS invokes an AWS Lambda function that transforms the log lines from strings to structured JSON (if necessary).
4. The Lambda function uses the OpenSearch _bulk API to deliver the JSON-formatted log lines to Amazon OpenSearch Service.
5. The user logs in to OpenSearch Dashboards to perform interactive log analytics, build visualizations, or notebooks, and monitor their dashboards.

aws

Architecture: Observability

Using Amazon OpenSearch Service, DevOps engineers gain insights to diagnose performance issues faster and reduce application downtime. To help visualize results and share data stories, the service includes OpenSearch Dashboards and Kibana. Getting a handle on observability is a key component of APM as well as infrastructure monitoring.



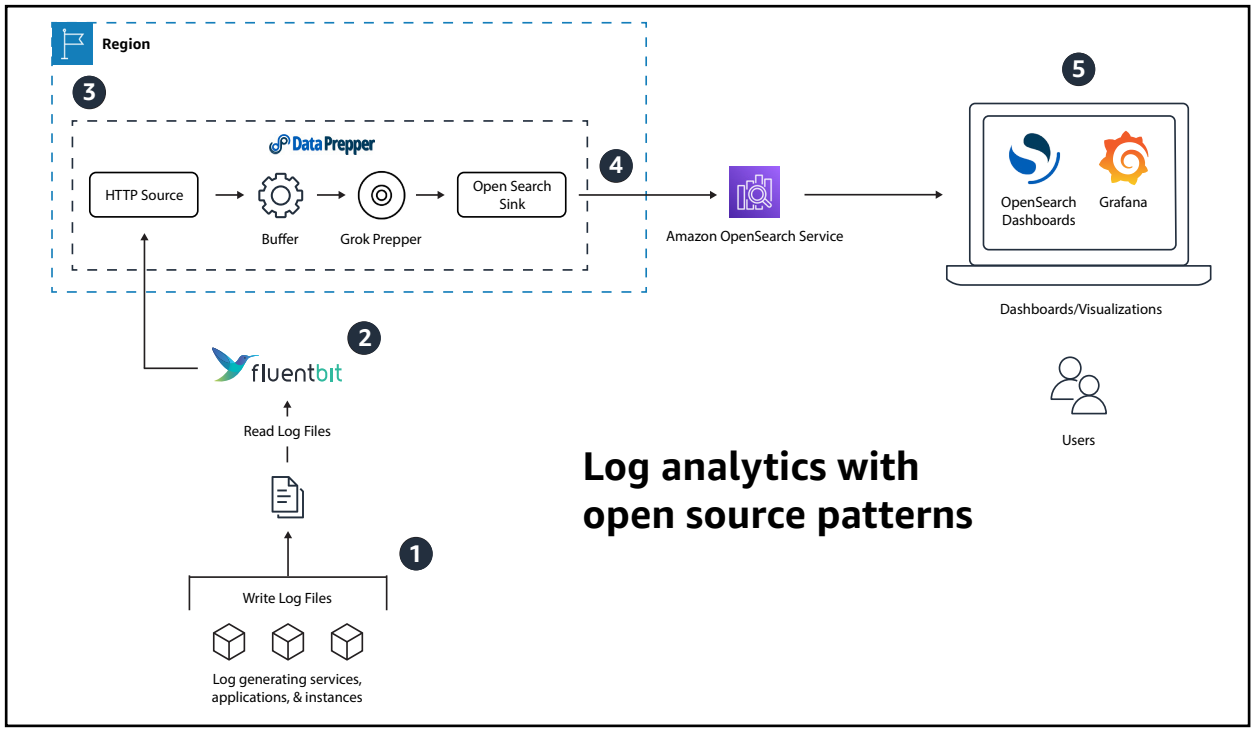
AWS Reference Architecture

1. Applications, services, and containers produce three types of signals: logs, metrics, and traces.
2. Collectors like FluentBit or DataPrepper transform and enrich these signals.
3. The collectors forward the data to different data stores. For example, Amazon OpenSearch Service stores traces and logs received from OpenSearch Data Prepper and Amazon Managed Service for Prometheus stores metrics received from OpenTelemetry metric scrapers.
4. Users create interactive dashboards and visualizations with this signal data using tools like Amazon OpenSearch Service Dashboards and Amazon Managed Service for Grafana.
5. These visualization tools use data stored in Amazon OpenSearch Service and Amazon Managed Service for Prometheus to present information requested by users.

aws

Architecture: Log analytics with open source patterns

As an open source log analytics suite, OpenSearch enables you to load log data into your Amazon OpenSearch Service domain using open source collectors and aggregators. Log sources can include application/infrastructure logs, security logs, AWS service logs, application trace logs, and application/infrastructure metrics. Log data can be collected and aggregated using open source systems such as Beats, FluentBit, Fluentd, and Data Prepper. The refined data can then be loaded into Amazon OpenSearch Service and view/analyzed using OpenSearch Dashboards. This diagram shows an architecture using FluentBit and Data Prepper to collect, aggregate, and transform logs into OpenSearch.



AWS Reference Architecture

1. The application, container system, and associated services generate logs. These can include Docker containers, Kubernetes pods, Amazon Elastic Compute Cloud (EC2) instances, Amazon Elastic Load Balancer (ELB) logs, AWS Lambda, relational database systems, etc.
2. FluentBit, a popular Apache-licensed log forwarder, reads the log files and forwards them to DataPrepper over HTTP.
3. Data Prepper is a server-side data collector capable of filtering, enriching, transforming, normalizing, and aggregating data for downstream analytics and visualization. Data Prepper receives the logs, buffers them, then optionally structures the data via a grok prepper.
4. DataPrepper creates the service map and assembles the traces into trace groups. It then sends the log lines, formatted for easy searching and analysis, to Amazon OpenSearch Service.
5. The user logs into OpenSearch Dashboards (or another open source visualization tool like Grafana) to do interactive log analytics. Grafana is an open source analytics and interactive visualization web application that provides charts, graphs, and alerts.

Get started with Amazon OpenSearch Service

Find out more about Amazon OpenSearch Service or connect with migration experts for advice, assistance with tooling, and information about financial incentives.