**AWS Opensearch for Analytics and Distributed Search**

(earlier referred as Elasticsearch)

AWS Opensearch is a community driven fork of Elasticsearch and Kibana. It does analytics and distributed search and quickly became the search engine being widely used for use cases like operational intelligence, log and business analytics, real-time application monitoring, full-text, website search etc. The data is sent as json files using Opensearch API and ingestion tools like [Logstash](https://aws.amazon.com/opensearch-service/the-elk-stack/logstash/),  [Kinesis Firehose.](https://aws.amazon.com/kinesis/data-firehose/)  The original document is automatically stored and a searchable reference of it is added the cluster’s index. The document can then be searched and retrieved using the Opensearch API. Kiban[a](https://aws.amazon.com/opensearch-service/the-elk-stack/kibana/) can also be used as a visualization tool for visualizing the data and building some interactive dashboards.

AWS Opensearch is s a fully managed service, which means one does not have to think about cluster management, software patching, hardware provisioning, failure recovery, monitoring and backups.

* **Basic Features**

### **a) Security**

* Access control is provided by making use of AWS Identity and Access Management (IAM).
* Node-to-node encryption of data is done by default.
* Field -level, index-level, and document-level security is assured.
* HTTP basic authentication present for Kibana by default.

### **b) Flexibility**

* Provision of custom packages for improving the search results.
* SQL support is provided for integration with BI tools.

### **c) Scalability**

* Provision of upto 3PB of storage attached.
* Supports **Ultrawarm storage** for storing the read-only data, it is a cost-effective way.
* Configurable options for CPU, storage capacity and memory.

### **d) Stability**

* **Automated snapshot** feature to take back up of opensearch domains, the restoration process is also automatic.
* Multiple geographical locations (called Availability Zones and Region)
* Allocation of nodes across multiple Availability zones in the same Region.
* Dedicated master nodes offered.

### **e) Third-party and AWS integrations**

* Easy integrations with services like Kibana in case of data visualization.
* Integration with AWS CloudWatch for monitoring the domain metrics and setting the alarms.
* Integrations with different Amazon services like DynamoDB, S3, and Kinesis etc for streaming data applications.
* Integration with AWS CloudTrail for purpose of auditing.
* Integrations with Amazon SNS in case of threshold breach.
* **Advanced features**
* **Advance level security**

It gives facility to do auditing, authentication, authorization and encryption features. Integrations along with AD, LDAP, Kerberos, SAML, JWT is there. There is fine-grained control, role-based control of documents, indices and fields.

* **Query Syntax of SQL**

It gives syntax familiar to SQL allowing usage of aggregations, where and group by to analyze data . Data can be read in CSV and JSON giving full flexibility.

* **Reporting feature**

Scheduling, exporting and sharing of reports from, alerts, dashboards, saved searches and visualizations.

* **Anomaly Detection**

Using ML for doing anomaly detection on the basis of [Random Cut Forest algorithm](https://github.com/aws/random-cut-forest-by-aws) which will detect automatically the anomalies in ingested data. This feature is combined with alert feature for real time monitoring of data.

* **Management of Index**

It allows defining the custom policies for automation of routine tasks of index management like rollover, delete, transformations.

* **RCA Framework and Performance Analyzer**

A number of query performance metrics and aggregations can be done using command line interface which quickly displays and analyzes them. The root cause analysis (RCA) framework is also there for investigation of reliability and performance and issues.

* **Asynchronous Search**

Asynchronous search gets run in the background so the user doesnot have to worry about the timing of query. Query progress is tracked and partial results are made available as and when they are available.

* **Trace Analytics**

It means ingestion and visualization of telemetry data in distributed applications. Visualization of events flow between the applications identifies the performance issues.

* **Alerting feature**

Automatic monitoring of data is done, on the basis of which alert notifications are sent to the stakeholders. The interface is quite intuitive and the API is powerful which lets users to easily setup and manage alerts. Very specific alerting conditions can be set.

* **KNN search**

By using ML, the nearest neighbor search algorithm is very similar to running query in OpenSearch when the document set is large. In use cases like product recommendations, image and video search, fraud detection, related document search aggregations and filter clauses are used for further refinements.

* **Piped Processing Language**

This feature gives the facility of familiar query syntax by providing a detailed set of commands to query data, these commands are delimited by pipes (|).

* **Dashboard Notebooks**

Combination of dashboards, text and visualizations provides in-depth explanations during data analysis.

* **Advantages**
* **Ease of utility**

As in Amazon Opensearch the services are fully managed, which makes it easy to use it efficiently. This also saves time for software patching, failure recovery, backup and monitoring. The users have the ability to deploy the production-ready within few minutes without worrying about maintenance and installation points.

* **Highly secure**

VPC settings, AWS IAM and Cognito policies help in managing the authentication, network isolation and access control.

* **Very cost-effective**

Pay as you use feature is one of the biggest advantages, only pay for resources consumed. On-demand pricing option is there without any upfront costs. Being a fully managed service, the cost of operations is reduced as there is less need of experts for managing and monitoring.

* **High scalability and availability**

It is a tool with high scalability as it enables users to store upto 3 PB of data in one cluster. The cluster can also be easily setup and down using single API call or by console. Multi-AZ deployments are done which allows data replication in 3 availability zones leading to high availability.

### **Tight integration with AWS Services**

### Built-in integrations with AWS services like IOT, Kinesis Firehose for seamless data ingestion and cloudwatch logs.

### **Supporting open-source APIs**

### It provides directly accessible open-source APIs and supports third-party softwares like Logstash and Kibana.

## Limitations

* It permits the users to launch the domain in a VPC and deploy a public endpoint but both the actions cannot be performed together at a time.
* The free tier is given for 12 months after which users are supposed to signup and pay for its usage.
* **Best practices**
* One should apply fine-grained access control along with the [principle of least privilege](https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege) while giving any kind of access to Opensearch and configuration API.
* Configuring at least single replica for every index is the default setting.
* Using 3 [dedicated master nodes](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/managedomains-dedicatedmasternodes.html).
* Deployment of the domain should be done across [3 Availability Zones](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/managedomains-multiaz.html)(AZ) so as to distribute the replica of shards to different AZs rather than corresponding the primary set shards.
* Use latest versions of Opensearch software.
* Appropriate sizing should be done for the domain, depending on the workload.
* Never have greater than 1,000 shards for any data node.
* Never use T2 and T3.small instances on production as they become unstable under heavy load conditions.
* C[reate the domain inside a VPC](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/vpc.html).
* Always select [data encryption at rest](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/encryption-at-rest.html) option and [node-to-node encryption](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/ntn.html).
* [Enabling slow search logs](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/createdomain-configure-slow-logs.html#createdomain-configure-slow-logs-console) and  [setting logging thresholds](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/createdomain-configure-slow-logs.html#createdomain-configure-slow-logs-indices) for every index to find root cause of queries that are running slower.
* Set a timeout parameter in query payloads for prevention of excess work depending on expected time for query completion. By default, the queries have no timeout.