

**Amazon Elasticsearch Service**

**Immersion Day**

**Lab: Cluster creation and text search**

# Introduction

## Overview

This lab demonstrates the basic steps required to get started with Amazon Elasticsearch Service: creating clusters, cluster node configurations, storage configurations and Identity Access Policies (IAM)

## Topics covered

By the end of this lab you will be able to:

* Deploy an Amazon Elasticsearch Service domain
* Create an index for a product catalog
* Publish catalog data to the Elasticsearch cluster
* Create a dashboard to summarize catalog data
* Monitor Amazon Elasticsearch Cluster Metrics

## Prerequisites

* Some familiarity with IAM Roles and EC2 Instances is recommended. Previous kowledge of Kibana and Elasticsearch is desirable.
* Have correctly installed and configured the AWS command line interface. See [the documentation](https://aws.amazon.com/cli/) for instructions.

# Amazon Elasticsearch Service

## Amazon Elasticsearch Service introduction

Amazon Elasticsearch Service is a managed service that makes it easy to deploy, operate, and scale Elasticsearch in the AWS cloud. Elasticsearch is a popular open-source search and analytics engine for use cases, such as log analytics, real-time application monitoring, click stream analytics, and text search.

With Amazon Elasticsearch Service, you get direct access to the Elasticsearch open-source API so that existing code and applications will work seamlessly. You can set up and configure your Amazon Elasticsearch cluster in minutes from the AWS Management Console.

Amazon Elasticsearch Service provisions all the resources for your cluster and launches it. Amazon Elasticsearch Service automatically detects and replaces failed Amazon Elasticsearch nodes, reducing the overhead associated with self-managed infrastructures. You can deploy an Amazon Elasticsearch cluster in minutes using the AWS Management Console. There are no upfront costs to set up Amazon Elasticsearch clusters, and you pay only for the service resources that you use.

Amazon Elasticsearch Service offers the following benefits of a managed service:

* Simple cluster scaling via API
* Self-healing clusters
* High availability on-demand
* Automatic cluster snapshots for data durability
* Security
* Cluster monitoring

## Prerequisites components of Amazon Elasticsearch Service

Amazon Elasticsearch Service contains the following components:

**Domain:** An Amazon Elasticsearch domain comprises an Elasticsearch cluster – hardware and software – along with additional hardware and software providing load-balancing, security, and monitoring. The domain is exposed by service endpoints for Amazon Elasticsearch Service, with a name that must meet the following criteria:

* Uniquely identifies a domain within an AWS account
* Starts with a letter or number
* Contains at least three characters, but not more than 28 characters
* Contains only lowercase characters a-z, the numbers 0-9, and the hyphen (-)

**Cluster:** A cluster is a collection of one or more data nodes, optional dedicated master nodes, and storage required to run Elasticsearch .

**Node:**  A node is single instance within an Elasticsearch cluster that has the ability to recognize and process or forward messages to other nodes.

**Storage:** Amazon Elasticsearch Service supports two distinct storage types, the Instance (default) storage or Elastic Block Store (EBS) – general purpose (SSD), provisioned IOPS (SSD), and magnetic.

## Related Services

Amazon Elasticsearch Service is commonly used with the following services:

## [AWS CloudTrail](http://aws.amazon.com/documentation/cloudtrail/)

Use AWS CloudTrail to get a history of the Amazon Elasticsearch Service API calls and related events for your account. CloudTrail is a web service that records API calls from your accounts and delivers the resulting log files to your Amazon S3 bucket. You can also use CloudTrail to track changes that were made to your AWS resources.

## [Amazon CloudWatch](http://aws.amazon.com/documentation/cloudwatch/)

An Elasticsearch domain automatically sends metrics to Amazon CloudWatch so that you can gather and analyze performance statistics. You can monitor these metrics by using the AWS CLI or the AWS SDKs. You can also easily subscribe a CloudWatch Logs log group to an Amazon Elasticsearch Service domain to load the data in that log group into Amazon ES.

## [Amazon Kinesis](http://aws.amazon.com/documentation/kinesis/)

Amazon Kinesis is a managed service that scales elastically for real-time processing of streaming data at a massive scale. Amazon Elasticsearch Service provides Lambda sample code for integration with Kinesis.

## [Amazon S3](http://aws.amazon.com/documentation/s3/)

Amazon Simple Storage Service (Amazon S3) is storage for the Internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. Amazon Elasticsearch Service provides Lambda sample code for integration with S3.

## [AWS Identity and Access Management (IAM](http://aws.amazon.com/iam/))

AWS Identity and Access Management (IAM) is a web service that you can use to manage users and user permissions in AWS. Use IAM to create user-based or IP-based access policies for your Amazon Elasticsearch Service domains.

## Amazon Elasticsearch Service Integration with Other Services

Amazon Elasticsearch Service integrates with the following services to provide data ingestion:

## [AWS Lambda](http://aws.amazon.com/documentation/lambda/)

AWS Lambda is a zero-administration compute platform for back-end web developers that runs your code in the AWS [cloud](http://aws.amazon.com/what-is-cloud-computing/) and provides you with a fine-grained pricing structure. Amazon Elasticsearch Service provides sample code to run on Lambda that integrates with Amazon Kinesis and Amazon S3.

## [Amazon DynamoDB](http://aws.amazon.com/documentation/dynamodb/)

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. Amazon Elasticsearch Service provides a Logstash plugin to support DynamoDB streams and sign AWS service requests.

# Create and Test an Amazon Elasticsearch Domain

## Deploy an Amazon Elasticsearch Service Domain

1. In the AWS Management Console, click **Elasticsearch Service**.
2. Click **Get Started**.
3. In **Step 1:** **Elasticsearch domain name**,type **mytestdomain** into the box, and click **Next**.
4. In **Step 2:** **Configure cluster** page, under the **Node configuration section**, select the following information from the drop-down lists:
   1. In the **Instance type** section, select **t2.small.elasticsearch (Free tier eligible)**.

When you launch an instance, the instance type that you specify determines the hardware of the host computer used for your instance. Each instance type offers different compute, memory, and storage capabilities.

* 1. In the **Instance count** drop-down list, select **4 (default)**.

This is the number of instances that you wish to deploy with your cluster.

* 1. Do not select **Enable dedicated master**.

A dedicated master node is a cluster node that performs cluster management tasks, but does not hold data or respond to data upload requests. This offloading of cluster management tasks increases the stability of your Amazon Elasticsearch clusters. We recommend that you avoid allocating dedicated master nodes for all small and short-lived Amazon Elasticsearch domains.

* 1. Select **Enable zone awareness**.

If you enable zone awareness, you should use Amazon Elasticsearch API to set up replicas for your cluster. Amazon Elasticsearch Service will distribute replicas across the nodes in Availability Zones. This will increase the availability of your cluster.

1. In the **Storage Configuration** section, select the following information from the drop-down lists:
   1. **Storage Type EBS**

Elasticsearch provides two different storage option types: Instance store and Elastic Block Store (EBS).

* 1. **EBS volume type General Purpose (SSD)**

General Purpose (SSD) storage is suitable for a wide variety of database workloads that have moderate I/O requirements. The baseline of 3 IOPS per GB and the ability to burst up to 3,000 IOPS will provide you with predictable performance well-suited to many applications.

* 1. **EBS volume size 30**

Amazon EBS provides durable, block-level storage volumes that you can attach to a running instance. You can use Amazon EBS as a primary storage device for data that requires frequent and granular updates.

1. In the **Snapshot configuration** section, select the following from the drop-down lists:
   1. **Automated snapshots start hour 00:00 UTC (default)**
2. In the **Advanced options** section, leave the values at the default settings.
3. Click **Next**.
4. In the **Network configuration** section select **Public Access**. For production systems select VPC access. Public access will be used for ease of accessibility for this lab.
5. In the **Step 3: Set up access policy** page,select the following from the drop-down lists:
   1. **Set the domain access** **policy to** : **Allow or deny access to one or more AWS accounts or IAM users**
   2. **In the resulting dialog**
      1. **Set the “Account ID or ARN” to: Your AWS account ID**
      2. Click **OK.**
6. Click **Next.**
7. Review the system configuration, and click **Confirm and create**.

Note: The service can take ten minutes to deploy. While waiting for the service to deploy, you can complete the steps in the next section.

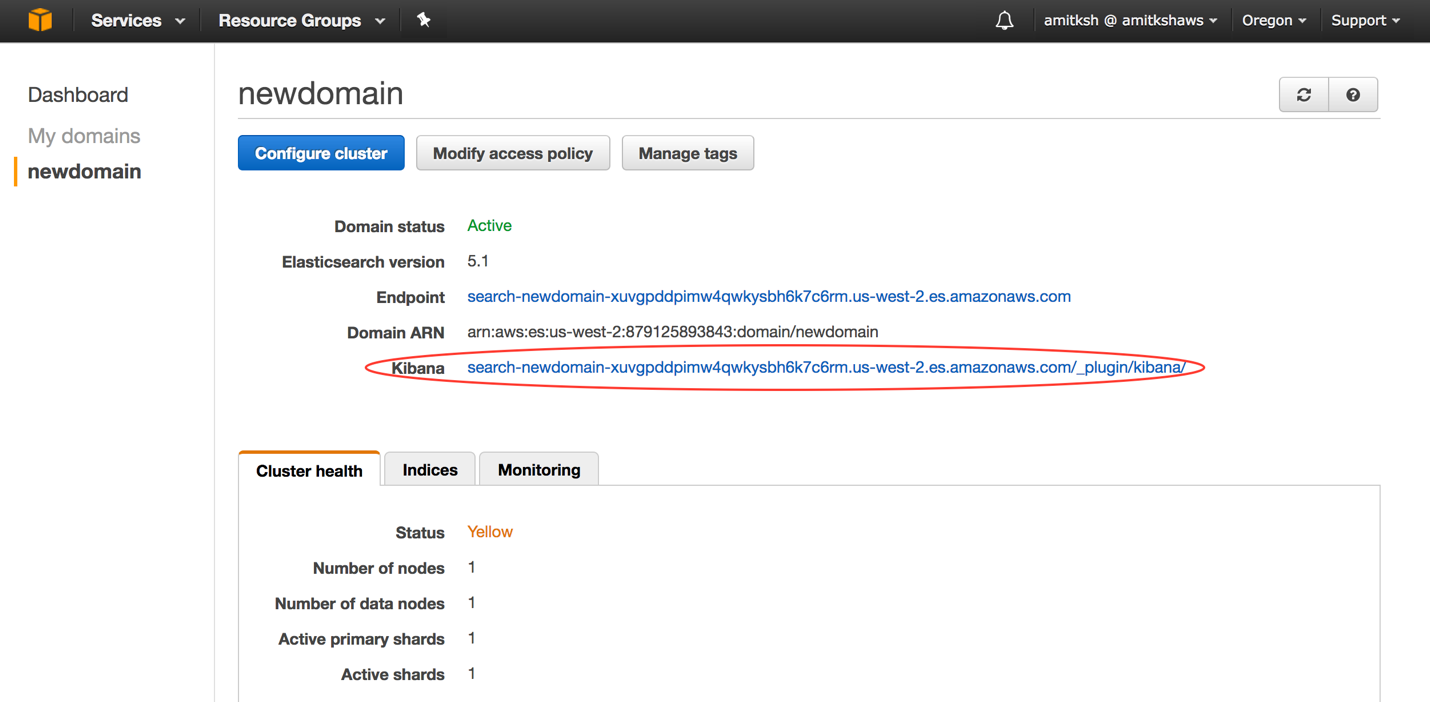
## Create a product catalog

In this exercise, you will ingest the sample marketplace data into your AES Cluster. Once the data has been ingested, you will start with basic queries and gradually fine tune the queries to make it more relevant for a particular business use case. Each row of sample data is about an item in marketplace and following fields are included -

* The date and time when item was listed
* Category of item
* Title of item
* Popularity of item
* Price of item
* Rating of item – between 0 and 4.99

**Note**: It is recommended to do this from an EC2 instance in order to reduce data transfer times from S3. A T2.micro is sufficient for this lab.

1. Download an index of the sample data from S3 using this URL: <https://s3.amazonaws.com/amitkshamazon/aes-day/lab4/lab4_aes_day.json>
2. Download the upload shell script from S3: <https://s3.amazonaws.com/amitkshamazon/aes-day/lab4/lab4_dumpdata.sh>
3. Once the cluster is running, connect to it and create the index and mapping for the product data. You can use Kibana Dev Tools via a web browser or use cURL commands. This lab will use the web interface of Kibana. From the Amazon Elasticsearch dashboard click on the Kibana link found in details of the domain you created for this lab as shown below.



1. On the Kibana dashboard click **Dev Tools** on the left-hand side panel
2. To confirm that you are connected to the cluster execute the following command in the console of Kibana Dev Tools:

GET /

1. The output should be similar to the following:

{

"name": "-Rpi1Xz",

"cluster\_name": "776347453069:sample-domain",

"cluster\_uuid": "JOXEsjpfT2qv0wjHii-V8Q",

"version": {

"number": "6.0.1",

"build\_hash": "d951bbf",

"build\_date": "2017-12-07T01:43:54.348Z",

"build\_snapshot": false,

"lucene\_version": "7.0.1",

"minimum\_wire\_compatibility\_version": "5.6.0",

"minimum\_index\_compatibility\_version": "5.0.0"

},

"tagline": "You Know, for Search"

}

1. Create an *Products* index with the mapping for the various fields expected in the catalog. Create it using the following command in the Kibana Dev Tools console:

PUT /products/

{

"mappings": {

"product": {

"properties": {

"popularity": {

"type": "integer"

},

"rating": {

"type": "scaled\_float",

"scaling\_factor": 100

},

"price": {

"type": "scaled\_float",

"scaling\_factor": 100

},

"category": {

"type": "keyword"

},

"title": {

"type": "text"

},

"launchDate": {

"type": "date",

"format": "yyyy-MM-dd HH:mm:ss"

}

}

}

}

}

This index has a custom date format specified for the “launchDate” field. Also we have set type to “keyword” for the “category” field because we want this field to be indexed but to only match whole words, exactly meeting the category names. Both price and rating fields have also been configured to use “scaled\_float” instead of float. This allows the trade off of accuracy for storage space. Internally both of these fields will be stored as integers with rouding off done to the scaling\_factor. Since integers are much better compressed as compared to float, hence this makes disk utilization efficient without impacting results. Note that, from user or API perspectives, there is no change and queries can still be done as float numbers.

1. Confirm that the index has been created by using the following command:

GET \_cat/indices

1. Confirm the mapping creation using this command:

GET /products/\_mapping/product

## Upload the product catalog

1. Edit the lab4\_dumpdata.sh script downloaded earlier and replace “YOURCLUSTER” with the cluster details from the AWS console. Make sure that the lab4\_aes\_day.json index is in the same folder as the script. Execute the shell script using the following:

$ bash lab4\_dumpdata.sh

This should load all the data into the cluster, execution time should be about 10 minutes. Once the script ends there should be just over 10,000 product documents inserted into the cluster.

1. Confirm the product catalog has been imported using the following command in the Kibana Dev Tools console:

GET /products/product/\_search

{

"query": {

"match\_all": {}

}

}

The “hits” section should indicate there are 10,034 records.

## Query the data and check relevancy

1. Query all the products launched in the last 365 days. Then search for all products launched in the last 345 days.

GET \_search

{

"query": {

"range": {

"launchDate": {

"gte": "now-**365**d/d",

"lt": "now/d"

}

}

}

}

1. The results returned by the previous command are not sorted by date. To sort the output by date use:

GET /products/product/\_search

{

"sort" : [

{ "launchDate" : {"order" : "desc"}},

"\_score"

],

"query": {

"range" : {

"launchDate" : {

"gte" : "now-345d/d",

"lt" : "now/d"

}

}

}

}

1. To enhance this further, and assuming the customer wants to query only the phones which were launched in this time period use a bool filter to combine the fields for “launchDate” and “category”:

POST /products/product/\_search

{

"sort": [

{

"launchDate": {

"order": "desc"

}

},

"\_score"

],

"query": {

"bool": {

"filter": {

"term": {

"category": "Phone"

}

},

"must": {

"range": {

"launchDate": {

"gte": "now-345d/d",

"lt": "now/d"

}

}

}

}

}

}

1. Try changing the query and replace “launchDate” with “rating” to sort the result by “rating”
2. Try boosting the search to search for “red” items with a preference for items matching the word “dress”:

GET /products/product/\_search

{

"query": {

"query\_string": {

"default\_field": "title",

"query": "red OR dress^2"

}

}

}

## Use Kibana to visualize your product catalog

1. Return to the Elasticsearch Service browser tab you initially opened.
2. Return to the **Elasticsearch Service** dashboard.
3. In the left navigation pane, click **mytestdomain**.
4. Copy your domain’s **Endpoint**.
5. Run **aws-es-kibana** on your laptop, using the **endpoint** from the last step.
6. In the **Index name or pattern** text box, delete all the current text “logstash-\*, and enter the **products** symbol.
7. In the **Time-field name** drop-down list, select **launchDate**, then click **Create**.
8. Click the **Discover** tab to view the product entries
9. Click **Visualize** on the left hand side to create graphs that will be used to create a **Dashboard**
10. Click the **+** button to start a new visualization
11. Select **Vertical Bar** and the **products** index to get started
12. For the X-Axis select **Date Histogram** for the aggregation using **launchDate** as the field and **Weekly** for the interval
13. Click the **play-arrow** button to test the graph and click **Save** to name the visualization **“Launch count per week”**
14. From the Visualization summary page click the **+** button and select **‘Metric’** for the graph type
15. Select the **products** index
16. For the Split Groups bucket select **Terms** as the aggregation
17. Select **category** for the Field drop down and test the visualization using the **play-arrow** button
18. Click save and name the visualization **Top product count by category**
19. On the left click Dashboard and the **Create Dashboard** button
20. Click **‘Add’** at the top of the screen and select the two visualizations previously created, positioning them to create your dashboard
21. When complete click **‘Save’** and name the dashboard **‘Product Summary’**

## Monitoring Amazon Elasticsearch Cluster Metrics and Statistics

1. Return to the **Elasticsearch Service** dashboard.
2. Click **mytestdomain**.
3. Click the **Indices tab**.

This shows you the document that has been uploaded to Kibana. Click the drop-down arrows to display further data: **Count**, **Size in bytes**, **Query total**, and **Mappings**.

1. Click the **Monitoring** tab.

This displays the various statistics of the cluster: CPU Utilization, Read Latency, Write Latency, and other core system resources will be displayed.

1. Scroll down, and select the **CPUUtilization** metric.
2. From the **Statistic** drop-down list, select **Maximum**.
3. Click **Update graph**.

# Conclusion

Congratulations! You now know how to:

* Deploy an Amazon Elasticsearch Service domain
* Create an index and field mapping
* Populate the cluster with data
* Create a dashboard to monitor your data
* Monitor Amazon Elasticsearch Cluster Metrics

# Additional Resources

* For more information about Elasticsearch, see <https://aws.amazon.com/elasticsearch-service/>
* For more information about Cloudtrail, see <https://aws.amazon.com/cloudtrail/>
* For more information about Cloudwatch, see <https://aws.amazon.com/cloudwatch/>