Amazon CloudSearch

Developer Guide

API Version 2013-01-01

Amazon CloudSearch: Developer Guide

Table of Contents

What Is Amazon CloudSearch?

Amazon CloudSearch is a fully managed service in the cloud that makes it easy to set up, manage, and scale a search solution for your website or application.

With Amazon CloudSearch you can search large collections of data such as web pages, document files, forum posts, or product information. You can quickly add search capabilities without having to become a search expert or worry about hardware provisioning, setup, and maintenance. As your volume of data and traffic fluctuates, Amazon CloudSearch scales to meet your needs.

> **Note**
> This document describes the Amazon CloudSearch 2013-01-01 API. If you have 2011-02-01 search domains and need to reference the old documentation, you can download a PDF of the 2011-02-01 Developer Guide.

You can use Amazon CloudSearch to index and search both structured data and plain text. Amazon CloudSearch features:

- Full text search with language-specific text processing
- Boolean search
- Prefix searches
- Range searches
- Term boosting
- Faceting
- Highlighting
- Autocomplete Suggestions

You can get search results in JSON or XML, sort and filter results based on field values, and sort results alphabetically, numerically, or according to custom expressions.

To build a search solution with Amazon CloudSearch, you take the following steps:

- **Create and configure a search domain.** A search domain includes your searchable data and the search instances that handle your search requests. If you have multiple collections of data that you want to make searchable, you can create multiple search domains.
- **Upload the data you want to search to your domain.** Amazon CloudSearch indexes your data and deploys the search index to one or more search instances.
- **Search your domain.** You send a search request to your domain's search endpoint as an HTTP/HTTPS GET request.

Topics

Are You New to Amazon CloudSearch?

For a high-level overview of Amazon CloudSearch, service highlights, and pricing information, see the Amazon CloudSearch detail page. If you are ready to start using Amazon CloudSearch, you should begin with Getting Started with Amazon CloudSearch (p. 8).

You can interact with Amazon CloudSearch through the AWS Management Console, AWS SDKs, or AWS CLI. While you can also submit API requests directly to Amazon CloudSearch, the SDKs and AWS CLI automatically sign your requests as needed and provide centralized tools for interacting with Amazon CloudSearch domains in conjunction with other AWS services. For information about the AWS SDKs, see Tools for Amazon Web Services. For information about installing and using the AWS CLI, see the AWS Command Line Interface User Guide.

For more information about configuring and managing your search domains, getting your data into Amazon CloudSearch, submitting search requests, and processing the responses, see:

- Preparing Your Data (p. 58)—how to format your data so you can upload it to an Amazon CloudSearch domain for indexing.
- Configuring Index Fields (p. 64)—how to configure indexing options for an Amazon CloudSearch domain.
- Searching Your Data with Amazon CloudSearch (p. 94)—how to use the Amazon CloudSearch query language.
- Controlling Search Results (p. 128)—how to sort, filter, and paginate search results.

How Search Works

The collection of data that you want to search (sometimes referred to as your *corpus*) can consist of unstructured full-text documents, semi-structured documents such as those formatted in mark-up languages like XML, or structured data that conforms to a strict data model. Each item that you want to be able to search (such as a forum post or web page) is represented as a document. Every document has a unique ID and one or more fields that contain the data that you want to search and include in results.

To make your data searchable, you represent it as a batch of documents in either JSON or XML and upload the batch to your search domain. Amazon CloudSearch then generates a search index from your document data according to your domain's configuration options. You submit queries against this index to find the documents that meet specific search criteria.

As your data changes, you submit updates to add, change, or delete documents from your index. Updates are applied continuously in the order they are received.

For information about how to format your data, see Preparing Your Data (p. 58).

Indexing in Amazon CloudSearch

To build a search index from your data, Amazon CloudSearch needs the following information:

- Which document fields do you want to search?
- Which document field values do you want to retrieve with the search results?

- Which document fields represent categories that you want to use to refine and filter search results?
- How should the text within a particular field be processed?

You define this metadata in your domain configuration by configuring indexing options. You use indexing options to specify the fields included in the search index and control how you can use those fields.

You must configure a corresponding index field for each document field that occurs in your data—there's a one-to-one mapping between document fields and the fields in your Amazon CloudSearch index. In addition to the index field name, you specify the following:

- The index field type
- Whether the field is searchable (`text` and `text-array` fields are always searchable)
- Whether the field can be used as a category (facet)
- Whether the field value can be returned with the search results
- Whether the field can be used to sort the results
- Whether highlights can be returned for the field
- A default value to use if no value is specified in the document data.

For information about how to configure index fields for Amazon CloudSearch, see Configuring Index Fields (p. 64).

Facets in Amazon CloudSearch

A facet is an index field that represents a category that you want to use to refine and filter search results. When you submit search requests to Amazon CloudSearch, you can request facet information to find out how many hits share the same value in a facet. You can display this information along with the search results and use it to enable users to interactively refine their searches. (This is often referred to as faceted navigation or faceted search.)

A facet can be any date, literal, or numeric field that has faceting enabled in your domain configuration. For each facet, Amazon CloudSearch calculates the number of hits that share the same value. You can define buckets to calculate facet counts for particular subsets of the facet values. Only buckets that have matches are included in the facet results.

For information about configuring facets, see Configuring Index Fields (p. 64). For information about using facet information to support faceted navigation, see Getting and Using Facet Information in Amazon CloudSearch (p. 116).

Text Processing in Amazon CloudSearch

During indexing, Amazon CloudSearch processes the contents of `text` and `text-array` fields according to the language-specific analysis scheme configured for the field. An analysis scheme controls how the text is normalized, tokenized, and stemmed, and specifies any stopwords or synonyms to take into account during indexing. Amazon CloudSearch provides default analysis schemes for each supported language. For information about configuring custom analysis schemes, see Configuring Analysis Schemes (p. 70). For information about how Amazon CloudSearch normalizes and tokenizes text and applies configured text options when indexing text fields and processing search requests, see Text Processing in Amazon CloudSearch (p. 79).

Sorting Results in Amazon CloudSearch

You can customize how search results are ranked by defining expressions that calculate custom values for every document that matches your search criteria. For example, you might define an

expression that takes into account the value in a document's `popularity` field as well as the default relevance score calculated by Amazon CloudSearch Expressions are simply numeric expressions that use standard numeric operators and functions. Expressions can reference `int` and `double` fields, other expressions, a document's relevance score (_score), as well as the epoch time (_time). When you submit search requests, you specify the expression(s) you want to use to sort the search results. You can also reference expressions within your search criteria.

A document's relevance `_score` indicates how relevant a particular search hit is to the search request. To calculate the relevance score, Amazon CloudSearch takes into account how many times the search terms appear in a document relative to the other documents in the index.

For information about how to configure expressions for your domain, see Configuring Expressions (p. 129).

Search Requests in Amazon CloudSearch

You submit search requests to your domain's search endpoint as HTTP/HTTPS GET requests. You can specify a variety of options to constrain your search, request facet information, control ranking, and specify what you want to be returned in the results. You can get search results in either JSON or XML. By default, Amazon CloudSearch returns results in JSON.

When you submit a search request, Amazon CloudSearch performs text processing on the search string. The search string is processed to:

- Convert all characters to lowercase
- Split the string into separate terms on whitespace and punctuation boundaries
- Remove terms that are on the stopword list for the field being searched.
- Map stems and synonyms according to the stemming and synonym options configure for the field being searched.

After this preprocessing is complete, Amazon CloudSearch looks up the search terms in the index and identifies all of the documents that match the request. To generate a response, Amazon CloudSearch processes this list of search hits to filter and sort the matching documents and compute facets. Amazon CloudSearch then returns the response in JSON or XML.

By default, Amazon CloudSearch returns search results ranked according to the hits' relevance _scores. Alternatively, your request can specify the index field or expression that you want to use to sort the hits. For example, you might want to sort hits by an index field that contains the price or an expression that calculates popularity.

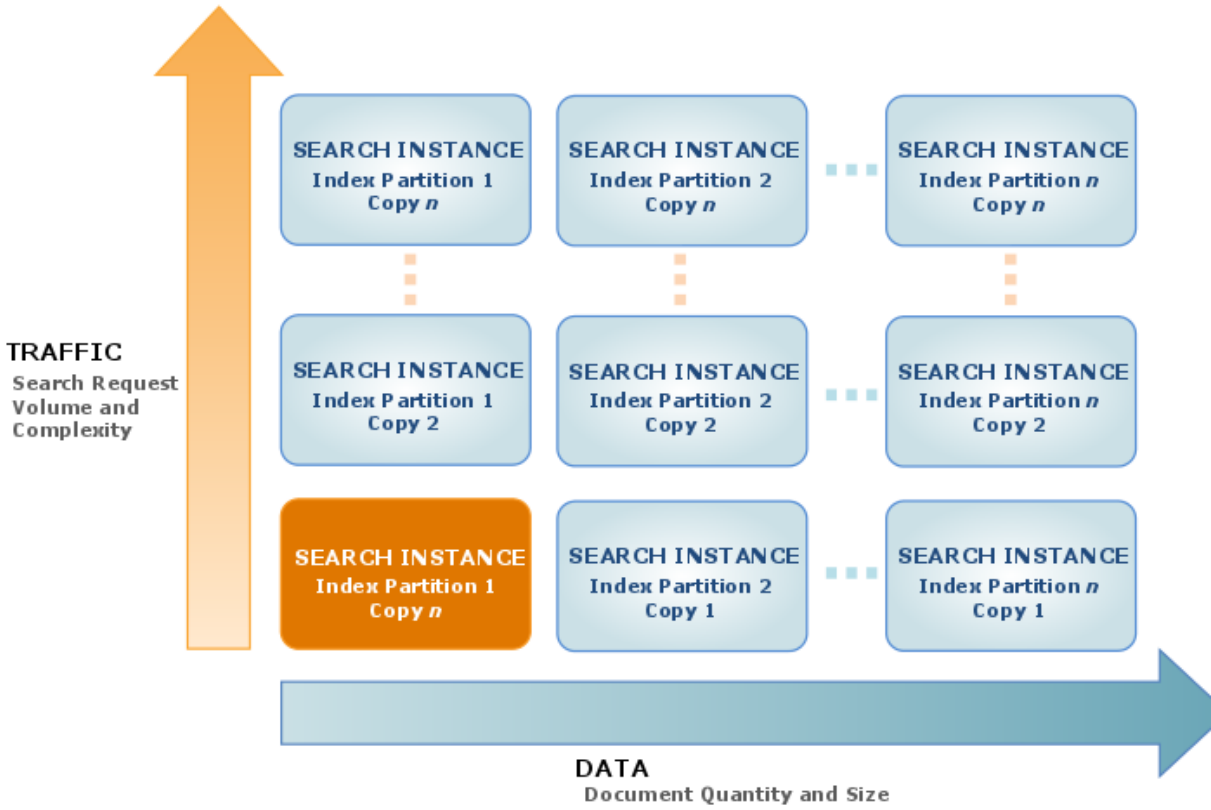
For more information about searching, ranking, and paginating results, see Searching Your Data with Amazon CloudSearch (p. 94).

Automatic Scaling in Amazon CloudSearch

A search domain has one or more search instances, each with a finite amount of RAM and CPU resources for indexing data and processing requests. How many search instances a domain needs depends on the documents in your collection and the volume and complexity of your search requests.

Amazon CloudSearch can determine the size and number of search instances required to deliver low latency, high throughput search performance. When you upload your data and configure your index, Amazon CloudSearch builds an index and picks the appropriate initial search instance type. As you use your search domain, Amazon CloudSearch can scale to accommodate the amount of data uploaded to the domain and the volume and complexity of search requests.

When you create a search domain, a single instance is deployed for the domain. As the following illustration shows, you always have at least one instance for your domain. Amazon CloudSearch automatically scales the domain by adding instances as the volume of data or traffic increases.



Scaling for Data

When the amount of data you add to your domain exceeds the capacity of the initial search instance type, Amazon CloudSearch scales your search domain to a larger search instance type. After a domain exceeds the capacity of the largest search instance type, Amazon CloudSearch partitions the search index across multiple search instances. (The number of search instances required to hold the index partitions is sometimes referred to as the domain's *width*.)

When the volume of data in your domain shrinks, Amazon CloudSearch scales down your domain to fewer search instances or a smaller search instance type to minimize costs.

Note
If your domain has scaled up to accommodate your index size and you delete a large number of documents, the domain scales down the next time the full index is rebuilt. Although the

index is automatically rebuilt periodically, to scale down as quickly as possible you can explicitly run indexing (p. 91) when you are done deleting documents.

Scaling for Traffic

As your search request volume or complexity increases, it takes more processing power to handle the load. A high volume of document uploads also increases the load on a domain's search instances. When a search instance nears its maximum load, Amazon CloudSearch deploys a duplicate search instance to provide additional processing power. (The number of duplicate search instances is sometimes referred to as the domain's *depth*.)

When traffic drops, Amazon CloudSearch removes search instances to minimize costs. For example, a new domain might scale up to handle the initial influx of documents, and scale back down after you have finished uploading your data and are only submitting updates.

If your domain experiences a sudden surge in traffic, Amazon CloudSearch deploys additional search instances. It takes a few minutes to set up the new instances, however, so you might see an increase in 5xx errors until the new instances can start processing requests. For more information about handling 5xx errors, see Handling Errors (p. 137).

Keep in mind that the type and complexity of your search requests affect overall search performance and in some cases increase the number of search instances required to operate your domain. Submitting a high volume of small or single-document batches can affect your search domain's performance. For more information, see Tuning Search Request Performance in Amazon CloudSearch (p. 111).

Accessing Amazon CloudSearch

You can access Amazon CloudSearch through the Amazon CloudSearch console, the AWS SDKs, or the AWS CLI.

- The Amazon CloudSearch console enables you to easily create, configure, and monitor your search domains, upload documents, and run test searches. Using the console is the easiest way to get started with Amazon CloudSearch and provides a central command center for ongoing management of your search domains.
- The AWS SDKs support all of the Amazon CloudSearch API operations, making it easy to manage and interact with your search domains using your preferred technology. The SDKs automatically sign requests as needed using your AWS credentials.
- The AWS CLI wraps all of the Amazon CloudSearch API operations to provide a simple way to create and configure search domains, upload the data you want to search, and submit search requests. The AWS CLI automatically signs requests as needed using your AWS credentials.

The standalone Amazon CloudSearch command line tools provide higher level tools for generating document batches and automatically configuring indexing options based on the contents of a batch. For more information about installing the Amazon CloudSearch tools, see the Command Line Tool Reference (p. 139).

Regions and Endpoints for Amazon CloudSearch

Amazon CloudSearch provides regional endpoints for accessing the configuration service and domain-specific endpoints for accessing the search and document services.

You use the configuration service to create and manage your search domains. The region-specific configuration service endpoints are of the form: `cloudsearch.`*`region`*`.amazonaws.com`. For

example, `cloudsearch.us-east-1.amazonaws.com`. For a current list of supported regions, see Regions and Endpoints in the AWS General Reference.

To access the Amazon CloudSearch search and document services, you use separate domain-specific endpoints:

- `http://doc-`*`domainname`*`-`*`domainid`*`.us-east-1.cloudsearch.amazonaws.com`—a domain's document service endpoint is used to upload documents.
- `http://search-`*`domainname`*`-`*`domainid`*`.us-east-1.cloudsearch.amazonaws.com`—a domain's search endpoint is used to submit search requests.

Signing Amazon CloudSearch Requests

If you're using a language for which AWS provides an SDK, we recommend that you use the SDK to submit Amazon CloudSearch requests. All of the AWS SDKs greatly simplify the process of signing requests and save you a significant amount of time when compared with using the Amazon CloudSearch APIs directly. The SDKs integrate easily with your development environment and provide easy access to related commands. You can also use the Amazon CloudSearch console and AWS CLI to submit signed requests with no additional effort.

If you choose to call the Amazon CloudSearch APIs directly, you must sign your own requests. Configuration service requests must always be signed. Upload, search, and suggest requests must be signed unless you configure anonymous access for those services. To sign a request, you calculate a digital signature using a cryptographic hash function, which returns a hash value based on the input. The input includes the text of your request and your secret access key. The hash function returns a hash value that you include in the request as your signature. The signature is part of the Authorization header of your request. After receiving your request, Amazon CloudSearch recalculates the signature using the same hash function and input that you used to sign the request. If the resulting signature matches the signature in the request, Amazon CloudSearch processes the request. Otherwise, the request is rejected.

Amazon CloudSearch supports authentication using AWS Signature Version 4. For more information, see Signature Version 4 Signing Process.

Getting Started with Amazon CloudSearch

To start searching your data with Amazon CloudSearch, you simply take the following steps:

- Create and configure a search domain
- Upload and index the data you want to search
- Send search requests to your domain

This tutorial shows you how to get up and running using the AWS Management Console for Amazon CloudSearch. To make it even easier to get started, we've generated a sample data set of 5,000 popular movie titles that you can download and examine, upload to your own search domain, and submit search queries against to see how Amazon CloudSearch works.

Using the AWS Management Console and the sample movie data, you'll have your own search domain up and running in about half an hour.

To begin, Get Signed Up (p. 8).

Topics

Before You Begin with Amazon CloudSearch

To use Amazon CloudSearch, you need an Amazon Web Services (AWS) account. Your AWS account enables you to access Amazon CloudSearch and other AWS services, such as Amazon Simple Storage Service (Amazon S3) and Amazon Elastic Compute Cloud (Amazon EC2). As with other AWS services, you pay only for the Amazon CloudSearch resources you use. There are no sign up fees and charges are not incurred until you create a search domain.

If you already have an AWS account, you are automatically signed up for Amazon CloudSearch.

For console access, use your IAM user name and password to sign in to the AWS Management Console using the IAM sign-in page. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about creating access keys, see How Do I Get Security Credentials? in the *AWS General Reference*.

To create an AWS account

1. Go to https://aws.amazon.com and click **Sign Up Now**.
2. Follow the instructions to sign up. You will need to enter payment information before you can begin using Amazon CloudSearch.

Step 1: Create an Amazon CloudSearch Domain

An Amazon CloudSearch domain encapsulates a collection of data you want to search, the search instances that process your search requests, and a configuration that controls how your data is indexed and searched. You create a separate search domain for each collection of data you want to make searchable. For each domain, you configure indexing options that describe the fields you want to include in your index and how you want to use them, analysis schemes that specify language-specific text processing options for individual fields, expressions that you can use to customize how search results are ranked, and access policies that control access to the domain's document and search endpoints.

You interact with a search domain to:

- Configure index and search options
- Submit data for indexing
- Perform searches

Each domain has a unique endpoint through which you submit search requests to the domain. For example, the endpoint for a domain called *movies* created in the US East (N. Virginia) region might be:

```
search-movies-mtshfsu2rje7ywr66uit3dei4m.us-east-1.cloudsearch.amazonaws.com
```

When creating a search domain, you specify a unique name for the domain. Domain names must start with a letter or number and be at least 3 and no more than 28 characters long. The allowed characters are: a-z, 0-9, and hyphen (-). By default, new domains are created in the US East (N. Virginia) region. To create a domain in another region, you must explicitly specify the region when creating the domain.

To configure the new domain, you must specify:

- Indexing options for the data you want to search.
- Access policies for the domain's document service and search service endpoints.

This tutorial shows you how to create and interact with a domain using the Amazon CloudSearch console. For information about how to use the command line tools and APIs, see Creating a Search Domain (p. 26).

> **Important**
> The domain you're about to create will be live and you will incur the standard Amazon CloudSearch usage fees for the domain until you delete it. For more information about Amazon CloudSearch usage rates, go to the Amazon CloudSearch detail page.

To create your movies domain

1. Go to the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. On the Welcome to Amazon CloudSearch page, click **Create Your First Search Domain**.
3. In the **NAME YOUR DOMAIN** step, enter a name for your new domain and click **Continue**. Domain names must start with a letter or number and be at least 3 and no more than 28 characters. Domain names can contain the following characters: a-z (lower case), 0-9, and - (hyphen). Upper case letters and underscores are not allowed.
4. In the **CONFIGURE INDEX** step, click **Use a predefined configuration**, select **IMDB movies (demo)**, and click **Continue**. You can also automatically configure a search domain by analyzing a sample of your data.
5. In the **REVIEW INDEX CONFIGURATION** step, review the index fields being configured. Eleven fields are configured automatically for the imdb-movie data: actors, directors, genres, image_url, plot, rank, rating, release_date, running_time_secs, title, and year.

   Note
   By default, all options are enabled for each field. While this is convenient for development and testing, fine-tuning the options configured for each field according to how you use those fields can reduce the size of your index. If your domain uses more than a single small search instance, tuning can help minimize the cost of running your domain.

   When you are finished reviewing the indexing options, click **Continue**.
6. In the **SET UP ACCESS POLICIES** step, click **Recommended rules** and click **Continue**. The recommended rules allow access to the search endpoint from all IP addresses, and restrict access to the document service to the IP address you specify.

   Important
   If you do not configure access rules for your search domain, you will only be able to interact with the domain through the Amazon CloudSearch console. By default, the document service and search service endpoints are configured to block all IP addresses.

   Keep in mind that if you do not have a static IP address, you must re-authorize your computer whenever your IP address changes. If your IP address is assigned dynamically, it is also likely that you're sharing that address with other computers on your network. This means that when you authorize the IP address, all computers that share it will be able to access your search domain's document service endpoint.
7. In the **CONFIRM** step, review the domain configuration and click **Confirm** to create your domain.
8. Once the domain has been created, click **OK** to exit the Create New Search Domain wizard and go to the domain's dashboard.

When you create a new domain, Amazon CloudSearch initializes resources for the domain, which can take about ten minutes. During this initialization process, the status of the domain will be LOADING. Once the status changes to ACTIVE, you can upload your data and start searching.

Step 2: Upload Data to Amazon CloudSearch for Indexing

You upload the data you want to search to your domain so that Amazon CloudSearch can build and deploy a searchable index. To be indexed by Amazon CloudSearch, the data must be formatted in either JSON or XML. The Amazon CloudSearch console can automatically convert the following file types to the required JSON or XML format:

- Comma Separated Value (.csv)

- Adobe Portable Document Format (.pdf)
- HTML (.htm, .html)
- Microsoft Excel (.xls, .xlsx)
- Microsoft PowerPoint (.ppt, .pptx)
- Microsoft Word (.doc, .docx)
- Text Documents (.txt)

When you upload a CSV file, Amazon CloudSearch parses each row separately. The first row defines the document fields, and each subsequent row becomes a separate document. For all other file types Amazon CloudSearch creates a single document and the contents of the file are mapped to a single text field. If metadata is available for the file, the metadata is mapped to corresponding document fields—the fields generated from the document metadata vary depending on the file type.

The sample IMDB movies data is already formatted in JSON.

This tutorial shows how to submit data through the Amazon CloudSearch console, but you can also convert (p. 62) and upload documents (p. 87) with the command line tools, and upload documents using the `documents/batch (p. 231)` resource. (To upload more than 5 MB of data, you must use the command line tools or API.)

To upload the sample data to your movies domain

1. Go to the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** panel, click the name of your movies domain to view the domain dashboard.
3. At the top of the domain dashboard, click the **Upload Documents** button.

   Note
   The **Upload Documents** button is available once the domain status is ACTIVE.

4. On the **DOCUMENT SOURCE** step, select **Predefined data**, choose **IMDB movies (demo)**, and click **Continue**.
5. On the **REVIEW DOCUMENTS** step, review the upload summary and click **Upload Documents** to send the data to your domain for indexing.

   Note
   If you'd like to see how the data is formatted, click **Download the generated document batch**. For more information about preparing your own data, see Preparing Your Data (p. 58).

6. On the **DOCUMENT SUMMARY** step, click **Finish** to return to the domain dashboard.

That's it! You now have a fully functional Amazon CloudSearch domain that you can start searching. Updates are applied continuously in the order they are received, so you can start searching your domain right away.

Step 3: Search Your Amazon CloudSearch Domain

You can use the search tester in the Amazon CloudSearch console to submit sample search requests and view the results. You can also submit sample search requests through a Web browser or using cURL. In your application, you can use any HTTP library to send search traffic to your Amazon CloudSearch domain.

Searching with the Search Tester

The search tester in the Amazon CloudSearch console enables you to submit sample search requests using any of the supported query parsers: simple, structured, lucene, or dismax. By default, requests are processed with the simple query parser. You can specify options for the selected parser, filter and sort the results, and browse the configured facets. The search hits are automatically highlighted in the search results. For information about how this is done, see Highlighting Search Hits in Amazon CloudSearch (p. 123). You can also select a suggester to get suggestions as you enter terms in the **Search** field. (You must configure a suggester before you can get suggestions. For more information see Getting Autocomplete Suggestions in Amazon CloudSearch (p. 123).)

By default, results are sorted according to an automatically-generated relevance score, *_score*. For information about customizing how results are ranked, see Sorting Results in Amazon CloudSearch (p. 128).

To search your domain

You can specify additional options for the selected query parser to configure the default operator and control which operators can be used in a query. For more information, see Search Request Parameters (p. 241).

1. Go to the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** panel, click the name of your movies domain and then click the **Run a Test Search** link.
3. To perform a simple text search, enter the text you want to search for and click **Go**. By default, all `text` and `text-array` fields are searched.
4. To search particular fields, click the **More Parameters** link and enter a comma-separated list of the fields you want to search in the **Search Fields** field. You can append a weight to each field with a caret (^) to control the relative importance of each field in the search results. For example, specifying `title^5, description` weights hits in the `title` field five times more than hits in the `description` field when calculating relevance scores for each matching document.
5. To use the structured query syntax, select **Structured** from the **Query Parser** menu. Once you've selected the structured query parser, enter your structured query in the **Search** field and click **Go**. For example, to find all of the movies with *star* in the title that were released in the year 2000 or earlier, you could enter: `(and title:'star' year:{,2000])`. For more information, see Constructing Compound Queries (p. 97). To submit Lucene or DisMax queries, select the appropriate query parser.

To view the HTTP search request that was sent to your domain's search endpoint and the response returned by Amazon CloudSearch, click the **view raw** link for the response format you want to see.

You can copy and paste the request URL to submit the request and view the response from a Web browser. Requests can be sent via HTTP or HTTPS.

Submitting Search Requests from a Web Browser

You can submit search requests directly to your search endpoint from any Web browser. You can use any of the query parsers (simple, structured, lucene, or dismax) and specify a variety of options to constrain your search, request facet information, customize ranking, and control what information is returned in the results.

For example, to search your movies domain and get the titles of all of the available *Star Wars* movies, append the following search string to your search endpoint. (2013-01-01 is the API version and must be specified.)

```
/2013-01-01/search?q=star+wars&return=title
```

Note
Your domain's search endpoint is shown on the domain dashboard. You can also perform a search from the AWS Management Console, view the raw request and response, and copy the request URL from the Search Request field. A domain's search and document service endpoints remain the same for the life of the domain.

By default, Amazon CloudSearch returns the response in JSON. You can also get the search results formatted in XML by specifying the `format` parameter, `format=xml`. (Note that errors can be returned in either JSON or XML, depending on where the error originated.)

Searching Numeric Fields

You can use the structured query syntax, `q.parser=structured`, to find documents that have particular numeric attributes. You can search for an exact value or a range of values within any numeric field (`double`, `double-array`, `int`, `int-array`). To search for a range, you specify the upper and lower bounds, separated by a comma, and enclose the range in brackets or braces. Use square brackets ([,]) when you want to include the bounds, and curly braces ({,}) to exclude the bounds. For example:

- `year:2000` matches documents whose year field contains the value 2000.
- `year:[2000,}` matches documents whose year field contains a value greater than or equal to 2000
- `year:{,2000]` matches documents whose year field contains a value less than or equal to 2000
- `year:[2000,2011]` matches documents whose year field contains a value between 2000 and 2011, inclusive.
- `year:{2000,2011}` matches documents whose year field contains a value between 2000 and 2011, exclusive.

You can also search date fields for a specific date or date range, but you must enclose each date string in single quotes: `release_date:['2000-01-01T00:00:00Z','2011-01-01T00:00:00Z']`.

For example, the following structured query searches for "star" in the title field, finds all of the matching movies that were released before 2000, and returns the title, year, and relevance score for each one:

```
q=(and title:'star' year:
{,2000])&q.parser=structured&return=title,year,_score
```

The response shows the status of the request, the number of matching documents, and the requested fields for each hit.

```
{
    "status": {
        "rid": "hLPckLsoEQoELQo=",
        "time-ms": 2
    },
    "hits": {
        "found": 15,
        "start": 0,
        "hit": [
            {
                "id": "tt0076759",
                "fields": {
                    "title": "Star Wars",
                    "year": "1977",
                    "_score": "5.7601414"
                }
```

```
            },
            .
            .
            .
            {
                "id": "tt0088170",
                "fields": {
                    "title": "Star Trek III: The Search for Spock",
                    "year": "1984",
                    "_score": "4.2371693"
                }
            }
        ]
    }
}
```

For more information about constructing search queries, see Searching Your Data with Amazon CloudSearch (p. 94).

Sorting the Search Results

By default, Amazon CloudSearch sorts the search results according to an automatically generated relevance `_score`. You can change how results are ranked by using the *sort* parameter in your search request to specify the field or expression you want to use for ranking. (An expression is a custom numeric expression that can be evaluated for each document in the set of matching documents. For information about defining your own expressions, see Configuring Expressions (p. 129).)

If you specify a text field with the `sort` parameter, the results are sorted alphabetically according to that field. For example, to sort results from your movies domain alphabetically by title, add `&sort=title asc` to your query string:

```
2013-01-01/search?q=(and genres:'Sci-Fi' year:
{,2000])&q.parser=structured&return=title,year&sort=title asc
```

Note that you must explicitly specify the sort direction, `asc` (ascending) or `desc` (descending). When you sort alphabetically, Amazon CloudSearch sorts by Unicode codepoint. This means numbers come before letters and uppercase letters come before lowercase letters. Numbers are sorted as strings; for example, 10 will come before 2.

Similarly, you can specify an integer field with the `sort` parameter to sort the results numerically.

If you specify a comma separated list of fields or expressions, the first field or expression is used as the primary sort criteria, the second is used as the secondary sort criteria, and so on.

For more information about ranking results, see Sorting Results in Amazon CloudSearch (p. 128)

Getting Facet Information

A facet is an index field that represents a category that you want to use to refine and filter search results. When you submit search requests to Amazon CloudSearch, you can request facet information to find out how many hits share the same value in a facet. You can display this information along with the search results and use it to enable users to interactively refine their searches. (This is often referred to as faceted navigation or faceted search.)

A facet can be any date, literal, or numeric field that has faceting enabled in your domain configuration. For each facet, Amazon CloudSearch calculates the number of hits that share the same value. You

can define buckets to calculate facet counts for particular subsets of the facet values. Only buckets that have matches are included in the facet results.

To get facet counts with your search results

- Use the `facet.FIELD` option to specify a field for which you want to compute facets. For the sample IMDB movies data faceting is enabled for the following fields: `genres`, `rank`, `rating`, `release_date`, `running_time_secs`, and `year`. Facet options are specified as a JSON object. If the JSON object is empty, `facet.FIELD={}`, facet counts are computed for all field values, the facets are sorted by facet count, and the top 10 facets are returned in the results:

```
q=star&return=title&facet.genres={}
```

The facets appear below the hits in the results.

```
facets": {

    "genres": {
        "buckets": [
            {"value": "Comedy","count": 41},
            .
            .
            .
            {"value": "Sport", "count": 7}
        ]
    }
}
```

You can specify options to calculate facets for selected field values, specify the maximum number of facet values to include in the results, and control how the facets are sorted.

To define buckets to compute facet counts for selected field values, you specify the `buckets` option. For example, the following request sorts the facet counts for the year field by decade:

```
q=star&facet.year={buckets:["[1970,1979]","[1980,1989]","[1990,1999]"]}
```

This constrains the facet counts to the three specified ranges:

```
"facets": {
        "year": {
            "buckets": [
                {"value": "[1970,1979]", "count": 3},
                {"value": "[1980,1989]","count": 7},
                {"value": "[1990,1999]","count": 12}
            ]
        }
}
```

For more information about specifying facet options, see Getting and Using Facet Information in Amazon CloudSearch (p. 116).

Getting Search Highlights

A search highlight is an excerpt of a text or text-array field that shows where the search term occurs within the field.

To get highlight information with your search results

- Use the `highlight.FIELD` option to specify the text or text-array field you want to get highlights for. The field must be highlight enabled in your domain's indexing options. For the sample IMDB movies data highlighting is enabled for the following fields: `actors`, `directors`, `plot`, and `title`. Highlight options are specified as a JSON object. If the JSON object is empty, `highlight.FIELD={}`, Amazon CloudSearch highlights all occurrences of the search term(s) by enclosing them in HTML emphasis tags, `<em>term</em>`, and the excerpts are returned as HTML.

```
q=title:'star'&q.parser=structured&return=_no_fields&highlight.title={}
```

The highlight information is included with each search hit.

```
hits": {
    "found": 29,
    "start": 0,
    "hit": [
        {
            "id": "tt0796366",
            "highlights": {
                "title": "<em>Star</em> Trek"
            }
        },
        .
        .
        .
        {
            "id": "tt2488496",
            "highlights": {
                "title": "<em>Star</em> Wars: Episode VII"
            }
        }
    ]
}
```

For more information about specifying highlight options, see Highlighting Search Hits in Amazon CloudSearch (p. 123).

Step 4: Delete Your Amazon CloudSearch Movies Domain

When you are finished experimenting with your movies domain, **you must delete it to avoid incurring additional usage fees.**

> **Important**
> Deleting a domain deletes the index associated with the domain and takes the domain's document and search endpoints offline permanently.

To delete your imdb-movies domain

1. Go to the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** panel, click the name of your movies domain to view to the domain dashboard.

3. At the top of the domain dashboard, click the **Delete this Domain** button.
4. In the **Delete Domain** dialog box, select the **Delete the domain** option and click **OK** to permanently remove the domain and all of its data.

   Note
   It can take around 15 minutes to delete the domain and its resources. Until then, the domain status will be *BEING DELETED*.

Wondering where to go next? Are You New to Amazon CloudSearch? (p. 2) has a guide to the rest of the Amazon CloudSearch developer documentation. For more information about the Amazon CloudSearch query language, see Searching Your Data with Amazon CloudSearch (p. 94). If you're ready to set up a domain with your own data, see Preparing Your Data (p. 58) and Uploading Data to an Amazon CloudSearch Domain (p. 87).

Migrating to the Amazon CloudSearch 2013-01-01 API

The Amazon CloudSearch 2013-01-01 API offers several new features, including support for multiple languages, highlighting search terms in the results, and getting suggestions. To use these features, you create and configure a new 2013-01-01 search domain, modify your data pipeline to populate the new domain using the 2013-01-01 data format, and update your query pipeline to submit requests in the 2013-01-01 request format. This migration guide summarizes the API changes and highlights the ones that are most likely to affect your application.

Creating 2013-01-01 Amazon CloudSearch Domains

If you created Amazon CloudSearch domains prior to the launch of the 2013-01-01 API, you can choose which API version to use when you create a new domain. To create a 2013-01-01 domain through the console, select the 2013-01-01 version in the Create Domain Wizard. To create a 2013-01-01 domain from the command line, download and install the AWS CLI and run the `aws cloudsearch create-domain` command.

> **Note**
> To create and interact with 2013-01-01 domains, you must use the AWS CLI tools. To create and interact with 2011-02-01 domains, you must use the v1 tools.

For more information about installing and using the command line tools, see Command Line Tool Reference (p. 139).

Configuring 2013-01-01 Amazon CloudSearch Domains

You can configure 2013-01-01 domains through the console, command line tools, or AWS SDKs. 2013-01-01 domains support several new configuration options:

- **Analysis Schemes**—you configure analysis schemes to specify language-specific text processing options for `text` and `text-array` fields. Amazon CloudSearch now supports 33 languages, as well as an option for multi-language fields. For more information, see Configuring Analysis Schemes (p. 70). For the complete list of supported languages, see Supported Languages (p. 80).
- **Availability Options**—you can enable the Multi-AZ option to expand a domain into a second availability zone to ensure availability in the event of a service disruption. For more information, see Configuring Availability Options (p. 42).
- **Scaling Options**—you can set the desired instance type and desired replication count to increase upload or search capacity, speed up search requests, and improve fault tolerance. For more information, see Configuring Scaling Options (p. 39).
- **Suggesters**—you can configure suggesters to implement autocomplete functionality. For more information, see Configuring Suggesters for Amazon CloudSearch (p. 124).

Access to the Amazon CloudSearch configuration service is managed through IAM and now enables you to control access to specific configuration actions. Note that the Amazon CloudSearch ARN has also changed. Access to your domain's document and search endpoints is managed through the Amazon CloudSearch configuration service. For more information, see Configuring Access for Amazon CloudSearch (p. 29).

2013-01-01 domains also support an expanded set of indexing options:

- **Analysis Scheme**—you configure language-specific text-processing on a per field basis by specifying an analysis scheme for each `text` and `text-array` field. For more information, see Configuring Analysis Schemes (p. 70).
- **Field Types**—Amazon CloudSearch now supports 11 field types:
  - date—contains a timestamp. Dates and times are specified in UTC (Coordinated Universal Time) according to IETF RFC3339: yyyy-mm-ddT00:00:00Z. In UTC, for example, 5:00 PM August 23, 1970 is: 1970-08-23T17:00:00Z.
  - date-array—a date field that can contain multiple values.
  - double—contains a double-precision 64-bit floating point value.
  - double-array—a double field that can contain multiple values.
  - int—contains a 64-bit signed integer value.
  - int-array—an integer field that can contain multiple values.
  - latlon—contains a location stored as a latitude and longitude value pair.
  - literal—contains an identifier or other data that you want to be able to match exactly.
  - literal-array—a literal field that can contain multiple values.
  - text—contains arbitrary alphanumeric data.
  - text-array—a text field that can contain multiple values.
- **Highlight**—when you enable the highlight option for a field, you can retrieve excerpts that show where the search terms occur within that field. For more information, see Highlighting Search Hits in Amazon CloudSearch (p. 123).
- **Source**—you can specify a source for a field to copy data from one field to another, enabling you to use the same source data in different ways by configuring different options for the fields.

When configuring your 2013-01-01 domain, there are several things to keep in mind:

- By default, when you add a field, all options valid for that field type are enabled. While this is useful for development and testing, disabling options you don't need can reduce the size of your index and improve performance.
- You must use the separate array type fields for multi-valued fields.
- Only single-value fields can be sort enabled.

- Only `text` and `text-array` fields can be highlight enabled.
- All fields *except* `text` and `text-array` fields can be facet enabled.
- Literal fields are now case-sensitive.
- You no longer have to store floating point values as integers—use a `double` field.
- You can store locations using the new `latlon` field type. For more information, see Searching and Ranking Results by Geographic Location in Amazon CloudSearch (p. 105).
- An `int` field is a 64-bit signed integer.
- Instead of configuring a default search field, you can specify which fields to search with the `q.options` parameter in your search requests. The `q.options` parameter also enables you to specify weights for each of the fields.
- When sorting and configuring expressions, you reference the default relevance score with the name `_score`. Due to changes in the relevance algorithm, the calculated scores will be different than they were under the 2011-02-01 API. For more information, see Configuring Expressions (p. 129).
- Expressions now support the `logn`, `atan2`, and `haversin` functions as well as the `_score` (text relevance score) and `_time` (epoch time) variables. If you store locations in `latlon` fields, you can reference the latitude and longitude values as `FIELD.latitude` and `FIELD.longitude`. You can also reference both `int` and `double` fields in expressions. The following functions are no longer supported: `cs.text_relevance`, `erf`, `lgamma`, `rand`, and `time`. For more information, see Configuring Expressions (p. 129).

For more information about configuring indexing options for a 2013-01-01 domain, see Configuring Index Fields (p. 64). For more information about configuring availability options, scaling options, text processing options, suggesters, and expressions see Creating and Managing Search Domains (p. 26).

New Amazon CloudSearch Configuration Service Actions and Options

The following actions have been added to the 2013-01-01 Configuration Service API:

- DefineAnalysisScheme
- DefineExpression
- DefineSuggester
- DeleteAnalysisScheme
- DeleteExpression
- DeleteSuggester
- DexcribeAnalysisSchemes
- DescribeAvailabilityOptions
- DescribeExpressions
- DescribeScalingParameters
- DescribeSuggesters
- ListDomainNames
- UpdateAvailabilityOptions
- UpdateScalingParameters

The `deployed` option has been added to the describe actions for index fields, access policies, and suggesters. Set the `deployed` option to true to show the active configuration and exclude pending changes.

Obsolete Amazon CloudSearch Configuration Service Actions and Options

The following actions are not supported in the 2013-01-01 Configuration Service API:

- DefineRankExpression
- DescribeRankExpression
- DeleteRankExpression
- DescribeDefaultSearchField
- DescribeStemmingOptions
- DescribeStopwordOptions
- DescribeSynonymOptions
- UpdateDefaultSearchField
- UpdateStemmingOptions
- UpdateStopwordOptions
- UpdateSynonymOptions

Uploading Data to 2013-01-01 Amazon CloudSearch Domains

With the 2013-01-01 API, you no longer have to specify document versions—updates are applied in the order they are received. You also no longer specify the `lang` attribute for each document—you control language-specific text processing by configuring an analysis scheme for each `text` and `text-array` field. You can use the `cs-import-documents` command to convert 2011-02-01 batches to the 2013-01-01 format.

To upload your data to a 2013-01-01 domain, you need to:

- Omit the `version` and `lang` attributes from your document batches. You can use `cs-import-documents` to convert 2011-02-01 SDF batches to the 2013-01-01 format.
- Make sure all of the document fields correspond to index fields configured for your domain. Unrecognized fields are no longer ignored, they will generate an error.
- Post the document batches to your 2013-01-01 domain's doc endpoint. Note that you must specify the 2013-01-01 API version. For example, the following request posts the batch contained in `data1.json` to the `doc-movies-123456789012.us-east-1.cloudsearch.amazonaws.com` endpoint.

```
curl -X POST --upload-file data1.json doc-movies-123456789012.us-east-1.
cloudsearch.amazonaws.com/2013-01-01/documents/batch --header "Content-Type:
application/json"
```

The 2013-01-01 API supports prescaling your domain to increase upload capacity. If you have a large amount of data to upload, configure your domain's scaling options and select a larger desired instance type. Moving to a larger instance type enables you to upload batches in parallel and reduces the time it takes for the data to be indexed. For more information, see Configuring Scaling Options (p. 39).

For more information about formatting your data, see Preparing Your Data (p. 58).

Searching 2013-01-01 Amazon CloudSearch Domains

Much of the effort required to migrate an existing Amazon CloudSearch search domain to the 2013-01-01 API is updating your query pipeline to submit 2013-01-01 compatible search requests.

- Use the 2013-01-01 API version in all requests.
- Use the `q` parameter to specify search criteria for all requests. The `bq` parameter is no longer supported. To use the structured (Boolean) search syntax, specify `q.parser=structured` in the request.
- Parameters cannot be repeated in a search request.
- The wildcard character (*) is only supported when using the simple query parser. Use the `prefix` operator to perform prefix matching with the structured query parser. For example, `q=(prefix 'oce')&q.parser=structured`.
- Use the field name `_id` to reference the document ID field in a search request. The `docid` field name is no longer supported.
- Use the `range` operator search a field for a value within the specified range. The `filter` operator is no longer supported.
- Use the new range syntax to search for ranges of values, including dates and locations stored in `latlon` fields. The double dot (..) notation is no longer supported. Separate the upper and lower bounds with a comma (,), and enclose the range in brackets or braces. A square bracket ([,]) indicates that the bound is included, a curly brace ({,}) excludes the bound. For example, `year:2008..2011` is now expressed as `year:[2008,2011]`. An open ended range such as `year:..2011` is now expressed as `year:{,2011]`.
- Use the `term` operator to search a field for a particular value. The `field` operator is no longer supported.
- Use the `q.options` parameter to specify field weights. The `cs.text_relevance` function is no longer supported. For example, `q.options={fields:['title^2','plot^0.5']}`.
- Use the `fq` parameter to filter results without affecting how the matching documents are scored and sorted.
- Use a dot (.) as a separator rather than a hyphen (-) in the prefix parameters: `expr.NAME`, `facet.FIELD`, `highlight.FIELD`.
- Use the `facet.FIELD` parameter to specify all facet options. The `facet-FIELD-top-N`, `facet-FIELD-sort`, and `facet-FIELD-constraints` parameters are no longer supported.
- Use the `sort` parameter to specify the fields or expressions you want to use for sorting. You must explicitly specify the sort direction in the sort parameter. For example, `sort=rank asc, date desc`. The `rank` parameter is no longer supported.
- Use `expr.NAME` to define an expression in a search request. The `rank-RANKNAME` parameter is no longer supported.
- Use `format=xml` to get results as XML. The `result-type` parameter is no longer supported.

The 2013-01-01 search API also supports several new features:

- Term boosting—use the `boost` option in a structured query to increase the importance of one part of the query relative to the other parts. For more information, see Constructing Compound Queries (p. 97).
- Sloppy phrase search—use the `near` operator in a structured query to search a `text` or `text-array` field for multiple terms and find documents that contain the terms within the specified distance of one another. You can also perform a sloppy phrase search with the simple query parser by appending the ~ operator and a value to the phrase. For more information, see Searching for Phrases (p. 101).

- Fuzzy search—use the ~ operator to perform fuzzy searches with the simple query parser. Append the ~ operator and a value to a term to indicate how much terms can differ and still be considered a match. For more information, see Searching for Individual Terms (p. 99).
- Highlighting—use the `highlight.FIELD` parameter to highlight matches in a particular field. For more information, see Highlighting Search Hits in Amazon CloudSearch (p. 123).
- Autocomplete—configure a suggester and submit requests to the `suggester` resource to get a list of query completions and the documents in which they were found. For more information, see Getting Autocomplete Suggestions in Amazon CloudSearch (p. 123).
- Partial search results—use the `partial=true` parameter to retrieve partial results when one or more index partitions are unavailable. By default Amazon CloudSearch only returns results if every partition can be queried.
- Deep paging—use the `cursor` parameter to paginate results when you have a large result set. For more information, see Paginating Results (p. 135).
- Match all documents—use the `matchall` structured query operator to retrieve all of the documents in the index.
- New query parsers—use the `q.parser` parameter to select the Lucene or DisMax parsers instead of the simple or structured parser, `q.parser=lucene` or `q.parser=dismax`.

You'll also notice some changes in behavior when searching:

- Strings are no longer tokenized on case boundaries and periods that aren't followed by a space are considered part of the term. For more information, see Text Processing in Amazon CloudSearch (p. 79).
- Literal fields are now case-sensitive.
- Search responses no longer include the rank, match expression, or CPU time. The only status information returned is the resource ID (rid) and processing time (time-ms).
- When you get facet information for an `int` field, `min` and `max` values are no longer returned.

For more information about searching your data, see Searching Your Data with Amazon CloudSearch (p. 94) and the Search API Reference (p. 239).

New Parameters and Options in the Amazon CloudSearch 2013-01-01 Search API

The following parameters have been added to the 2013-01-01 Search API:

- `cursor.FIELD`
- `expr.NAME`
- `facet.FIELD`
- `format`
- `fq`
- `highlight.FIELD`
- `partial`
- `pretty`
- `q.options`
- `q.parser`
- `return`
- `sort`

The ~ operator has been added to the simple query language to support fuzzy searches and sloppy phrase searches.

The following operators have been added to the structured query language:

- `boost`
- `matchall`
- `near`
- `phrase`
- `prefix`
- `range`
- `term`

Obsolete Amazon CloudSearch Search Parameters and Options

The following parameters are no longer supported in the 2013-01-01 search API:

- bq
- facet-FIELD-top-N
- facet-FIELD-sort
- facet-FIELD-constraints
- rank
- rank-RANKNAME
- return-fields
- result-type
- t-FIELD

The following operators and shortcuts are no longer supported in structured queries:

- field
- filter
- -
- |
- +
- *

Updated Limits in Amazon CloudSearch 2013-01-01

This table summarizes the changes and additions to the Amazon CloudSearch limits. For the complete list of Amazon CloudSearch limits, see Limits (p. 265).

Change	Summary
Reserved names	*score* is the only reserved name.

Change	Summary
No limit on return data	Data returned from a text field is no longer truncated at 2 KB. However, keep in mind that the maximum document size is 1 MB.
No limit on stemming, stopword, or synonym dictionaries.	Stemming, stopword, and synonym dictionaries are configured in an analysis scheme and there is no limit on the size of an analysis scheme definition.
Maximum number of field values	An array type field can contain up to 1000 values.
Field size	The maximum size of `literal` fields is 4096 Unicode code points.
Int field range	An `int` field can contain values in the range -9,223,372,036,854,775,808 - 9,223,372,036,854,775,807 (inclusive).
Maximum number of highlights	The maximum number of occurrences of the search term(s) that can be highlighted is 5.
Maximum number of suggesters	The maximum number of suggesters you can configure for a domain is 10.
Maximum number of hits you can retrieve at once	The maximum number of hits you can retrieve at once is 10,000. The `size` parameter can contain values in the range 0 - 10000.

Creating and Managing Amazon CloudSearch Domains

A search domain encapsulates the data you want to search, indexing options that control how you can search the data and what information you can retrieve from your search domain, and the search instances that index your data and process search requests. You can create (p. 26), monitor (p. 44), and delete (p. 53) domains using the Amazon CloudSearch console, Amazon CloudSearch command line tools, or AWS SDKs. All domain management actions are implemented by the Amazon CloudSearch configuration service. For more information, see the Configuration API Reference for Amazon CloudSearch (p. 148).

Topics

Creating an Amazon CloudSearch Domain

To search your data with Amazon CloudSearch, the first thing you need to do is create a search domain. If you have multiple collections of data that you want to make searchable, you can create multiple search domains. Before you can send search requests (p. 94) to a new domain, you must also configure access policies (p. 29), configure index fields (p. 64), and upload the data you want to search (p. 87).

When you create a search domain, you must give it a unique name. Domain names must start with a letter or number and be at least 3 and no more than 28 characters long. The allowed characters are: a-z, 0-9, and hyphen (-). Upper case letters, underscores (_), and other special characters are not allowed in domain names.

By default, all new domains are created using the 2013-01-01 API version. If you have previously created search domains with the 2011-02-01 API version, you can opt to use the old API for your new domain. However, we recommend using the 2013-01-01 API for all new use cases. All domains will need to migrate to the 2013-01-01 API when the 2011-02-01 API is retired.

You can choose the AWS region where you want to create your search domain. Typically, you should choose the region closest to your operations. For example, if you reside in Europe, create your search domain in the EU (Ireland) region (eu-west-1). For a current list of supported regions and endpoints, see Regions and Endpoints. For more information about choosing a region, see Regions and Endpoints for Amazon CloudSearch (p. 6).

> **Note**
> Amazon CloudSearch domains in different regions are entirely independent. For example, if you create a search domain called *my-domain* in us-east-1, and another domain called *my-domain* in eu-west-1, they are completely independent and do not share any data.

Each search domain has unique endpoints through which you upload data for indexing and submit search requests. A domain's document and search endpoints remain the same for the life of the domain. For example, the endpoints for a domain called imdb-movies might be:

```
doc-imdb-movies-nypdffbzrfkoudsurkxvgwbpi4.us-
east-1.cloudsearch.amazonaws.com
search-imdb-movies-nypdffbzrfkoudsurkxvgwbpi4.us-
east-1.cloudsearch.amazonaws.com
```

> **Important**
> By default, access to a new domain's document and search endpoints is blocked for all IP addresses. You must configure access policies for the domain to be able to submit search requests to the domain's search endpoint and upload data from the command line or through the domain's document endpoint. You can upload documents and search the domain through the Amazon CloudSearch console without configuring access policies.

You can create a search domain from the Amazon CloudSearch console (p. 27), using the `aws cloudsearch create-domain` command, or using one of the AWS SDKs.

Topics

Creating a Domain Using the Amazon CloudSearch Console

The Amazon CloudSearch console enables you to easily create new search domains and provides a variety of options for configuring indexing options.

To create a domain

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. At the top of the **Navigation** pane, click **Create a New Domain**. (If you are creating a domain for the first time, click **Create Your First Search Domain** on the Welcome page.)
3. In the **NAME YOUR DOMAIN** step, enter a name for your new domain and click **Continue**. Domain names must start with a letter or number and be at least 3 and no more than 28

characters long. Domain names can contain the following characters: a-z (lower case), 0-9, and - (hyphen). Upper case letters, underscores (_), and other special characters are not allowed in domain names.

Optionally, you can set the **Desired Instance Type** and **Desired Replication Count** to prescale your domain. For more information, see Configuring Scaling Options (p. 39).

4. In the **CONFIGURE INDEX** step, select **Manual Configuration** and click **Continue**. You can configure index fields and access policies when you first create the domain, or simply create a domain and configure it later. For more information about using the Amazon CloudSearch console to configure the domain, see Configuring Index Fields (p. 64) and Configuring Access for Amazon CloudSearch (p. 29).
5. In the **REVIEW INDEX CONFIGURATION** step, click **Continue** to configure the index fields later. For more information about configuring index fields, see Configuring Index Fields (p. 64).
6. In the **SET UP ACCESS POLICIES** step, click **Continue** to set up access policies later. For more information about configuring access policies, see Configuring Access for Amazon CloudSearch (p. 29).

   Note
   Until you configure access policies, you will only be able to upload documents and submit search queries through the console. By default, the Document and Search endpoints are configured to block all IP addresses.
7. In the **CONFIRM** step, review the domain configuration and click **Confirm** to create your domain.
8. Once the domain has been created, click **OK** to exit the Create New Search Domain wizard and go to the domain's dashboard. The domain's document and search service endpoints are displayed on the domain dashboard when the domain reaches the ACTIVE state. At that point, you can upload documents for indexing and start searching your data.

Creating a Domain Using the AWS CLI

You use the `aws cloudsearch create-domain` command to create search domains. For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

Note
If you are using the 2.0.0.1 version of the Amazon CloudSearch command line tools, you can use the `cs-create-domain` command to create a search domain. However, we recommend that you migrate to the AWS CLI, which provides a cross-service CLI with a simplified installation, unified configuration, and consistent command line syntax.

To create a domain

- Run the `aws cloudsearch create-domain` command and specify the name of the domain you want to create with the `--domain-name` option. For example, to create a domain called *movies*:

```
aws cloudsearch create-domain --domain-name movies
{
  "DomainStatus": {
      "DomainId": "965407640801/movies",
      "Created": true,
      "Deleted": false,
      "SearchInstanceCount": 0,
      "DomainName": "movies",
      "SearchService": {},
      "RequiresIndexDocuments": false,
      "Processing": false,
      "DocService": {},
```

```
      "ARN": "arn:aws:cloudsearch:us-east-1:965407640801:domain/movies",
      "SearchPartitionCount": 0
  }
}
```

The `aws cloudsearch create-domain` command returns immediately. It takes about ten minutes to create endpoints for a new domain. You can use the `aws cloudsearch describe-domains` command to view a summary of the domain's status and configuration. For more information, see Getting Information About an Amazon CloudSearch Domain (p. 44).

Important
Once a domain's endpoints are active, they remain the same for the life of the domain. You should cache the endpoints—there's no need to query for the endpoint before submitting a document or search service request and doing so is likely to result in your requests being throttled.

Creating an Amazon CloudSearch Domain Using the AWS SDKs

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `CreateDomain (p. 155)`. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Configuring Access for Amazon CloudSearch

You use AWS Identity and Access Management (IAM) access policies to control access to the Amazon CloudSearch configuration service and each search domain's document, search, and suggest services. An IAM access policy is a JSON document that explicitly lists permissions that define what actions people or processes are allowed to perform. For an introduction to IAM access policies, see Overview of AWS IAM Policies.

You control access to the Amazon CloudSearch configuration service APIs and the domain services APIs independently. For example, you might choose to restrict who can modify the configuration of your production domain, but allow team members to create and manage their own domains for development and testing. Similarly, you might configure your development and test domains to accept anonymous requests to the upload, search, and suggest services, but lock down your production domain so that it accepts only authenticated requests from your application.

When AWS receives a request, it authenticates that the request is from a known AWS user, and then checks relevant policies to determine whether the user is authorized to perform the requested actions using the requested resources. If a user has not been explicitly granted permission to perform an action, the request is denied. During policy evaluation, if AWS encounters an explicit deny, the deny effect takes precedence over any explicit allow effects that are in force.

Important
To enable authentication, Amazon CloudSearch requests must be signed with an access key. The only exception is if you allow anonymous access to a domain's upload, search, or suggest services. For more information, see Signing Requests (p. 7).

Topics

- Configuring Access for Amazon CloudSearch with the AWS CLI (p. 38)
- Configuring Access to a Domain's Endpoints Using the AWS SDKs (p. 39)

Writing Access Policies for Amazon CloudSearch

Amazon CloudSearch supports both *user-based policies* and *resource-based policies*:

- **User-based policies** are attached to a particular IAM user, group, or role. A user-based policy specifies which of your account's search domains a person or process can access and what actions they can perform. To attach a user-based policy to a user, group, or role, you use the IAM console, AWS CLI, or AWS SDKs. **You must define user-based policies to control access to the Amazon CloudSearch configuration service actions.** (The *user* in this context isn't necessarily a person, it's just an identity with associated permissions. For example, you might create an IAM user to represent an application that needs to have credentials to submit search requests to your domain.)
- **Resource-based policies** for Amazon CloudSearch are attached to a particular search domain. A resource-based policy specifies who has access to the search domain and which domain services they can use. Resource-based policies control access only to a particular domain's document, search, and suggest services; they cannot be used to configure access to the Amazon CloudSearch configuration service actions. To attach a resource-based policy to a domain, you use the Amazon CloudSearch console, AWS CLI or AWS SDKs.

In general, we recommend managing access to Amazon CloudSearch APIs by configuring user-based policies. This enables you to manage all of your permissions in one place and any changes you need to make take effect almost immediately. However, to allow public access to a domain's search service or restrict access based on IP addresses, you must configure a resource-based policy for the domain. (We recommend replacing your old IP based access policies with user-based policies at your earliest convenience.) You can also use resource-based policies to easily allow other accounts to access a domain. Keep in mind that processing changes to a domain's resource-based policies takes significantly longer than applying changes to user-based policies.

You can use the IAM Policy Generator to write both user-based and resource-based policies for Amazon CloudSearch. For more information, see Managing IAM Policies.

> **Tip**
> As a best practice, we recommend that you configure permissions for a group and assign IAM users to that group instead of defining permissions for individual users. Similarly, you can assign permissions to roles for applications that run on Amazon EC2 instances rather than passing user credentials to each instance. For more IAM recommendations for managing access to your AWS resources, see IAM Best Practices.

Contents of an Access Policy for Amazon CloudSearch

You specify the following information in your access policies for Amazon CloudSearch:

- `Version` specifies the policy language version that the statement is compatible with. The version is always set to `2012-10-17`.
- `Resource` is the ARN (Amazon Resource Name) for the domain to which a user-based policy applies. `Resource` is not specified in resource-based policies configured through the Amazon CloudSearch configuration service, because the policy is attached directly to the resource. For more information about Amazon CloudSearch ARNs, see Amazon CloudSearch ARNs (p. 31).
- `Effect` specifies whether the statement authorizes or blocks access to the specified action(s). It must be `Allow` or `Deny`
- `Sid` is an optional string that you can use to provide a descriptive name for the policy statement.
- `Action` specifies which Amazon CloudSearch actions the statement applies to. For the supported actions, see Amazon CloudSearch Actions (p. 32). You can use a wildcard (*) as the action to

configure access for all actions when you need to grant administrative access to select users. (In this case, you might also want to enable multi-factor authorization for additional security. For more information, see Configuring MFA-Protected API Access.) Wildcards are also supported within action names. For example, `"Action":["cloudsearch:Describe*]` matches all of the configuration service `Describe` actions, such as `DescribeDomains` and `DescribeServiceAccessPolicies`.

- `Condition` specifies conditions for when the policy is in effect. When configuring anonymous, IP-based access, you would specify the IP addresses that the access rule applies to, for example `"IpAddress": {"aws:SourceIp": ["192.0.2.0/32"]}`.
- `Principal` specifies who is allowed access to the domain in a resource-based policy. `Principal` is not specified in user-based policies configured through IAM. The `Principal` value for a resource-based policy can specify other AWS accounts or IAM users in your own account. For example, to grant access to the account 555555555555, you would specify `"Principal":{"AWS": ["arn:aws:iam::555555555555:root"]}`. Specifying a wildcard (*) enables anonymous access to the domain. Anonymous access is not recommended. If you enable anonymous access, you should at least specify a condition to restrict which IP addresses can submit requests to the domain. For more information, see Granting Access to a Domain from Selected IP Addresses (p. 35).

For examples of access policies for Amazon CloudSearch, see Amazon CloudSearch Policy Examples (p. 32).

Amazon CloudSearch ARNs

A policy's Amazon Resource Name (ARN) uniquely specifies the domain that the policy applies to. The ARN is a standard format that AWS uses to identify resources. The 12-digit number in the ARN is your AWS account ID. Amazon CloudSearch ARNs are of the form `arn:aws:cloudsearch:`**`REGION`**`:`**`ACCOUNT-ID`**`:domain/`**`DOMAIN-NAME`**.

The following list describes the variable elements in the ARN:

- `REGION` is the AWS region where the Amazon CloudSearch domain for which you are configuring permissions resides. You can use a wildcard (*) for the `REGION` for all regions.
- `ACCOUNT-ID` is your AWS account ID with no hyphens; for example, 111122223333.
- `DOMAIN-NAME` identifies a specific search domain. You can use a wildcard (*) for the `DOMAIN-NAME` for all of your account's domains in the specified region. If you have multiple domains whose names start with the same prefix, you can use a wildcard to match all of those domains. For example, `dev-*` matches `dev-test`, `dev-movies`, `dev-sandbox`, and so on. Note that if you name new domains with the same prefix, the policy will also apply to those new domains.

For example, the following ARN identifies the `imdb-movies` domain in the `us-east-1` region owned by account 111122223333:

```
arn:aws:cloudsearch:us-east-1:111122223333:domain/imdb-movies
```

The following example shows how the ARN is used to specify the resource in a user-based policy.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["cloudsearch:search"],
      "Resource": "arn:aws:cloudsearch:us-east-1:111122223333:domain/movies"
    }
  ]
```

```
}
```

A domain's ARN is displayed on the domain dashboard in the Amazon CloudSearch console and is also available by calling `DescribeDomains`.

Important
When specifying an ARN for a domain created with the 2011-02-01 API, you must use the former Amazon CloudSearch service name, `cs`. For example, `arn:aws:cs:us-east-1:111122223333:domain/imdb-movies`. If you need to define policies that configure access for both 2011 and 2013 domains, make sure to specify the correct ARN format for each domain. For more information, see Configuration Service Access Policies Not Working (p. 262).

Amazon CloudSearch Actions

The actions you specify control which Amazon CloudSearch APIs the statement applies to. All Amazon CloudSearch actions are prefixed with `cloudsearch:`, such as `cloudsearch:search`. The following list shows the supported actions:

- `cloudsearch:document` allows access to the document service API. Permission to use the `document` action is required to upload documents to a search domain for indexing.
- `cloudsearch:search` allows access to the search API. Permission to use the `search` action is required to submit search requests to a domain.
- `cloudsearch:suggest` allows access to the suggest API. Permission to use the `suggest` action is required to get suggestions from a domain.
- `cloudsearch:`**`CONFIGURATION-ACTION`** allows access to the specified configuration service action. Permission to use the `DescribeDomains` and `ListDomainNames` configuration actions is required to access the Amazon CloudSearch console. Configuration actions can be specified only in user-based policies. For the complete list of actions, see Actions (p. 151).

Amazon CloudSearch Policy Examples

This section presents a few examples of Amazon CloudSearch access policies.

Topics

Granting Read-only Access to the Amazon CloudSearch Configuration Service

You can grant read-only access to the configuration service by allowing only the following actions. This might be useful if you want to allow users to view the configuration of a production domain without being able to make changes.

- `cloudsearch:DescribeAnalysisSchemes`
- `cloudsearch:DescribeAvailabilityOptions`

- `cloudsearch:DescribeDomains`
- `cloudsearch:DescibeExpressions`
- `cloudsearch:DescribeIndexFields`
- `cloudsearch:DescribeScalingParameters`
- `cloudsearch:DescribeServiceAccessPolicies`
- `cloudsearch:DescribeSuggesters`
- `cloudsearch:ListDomainNames`

The following user-based policy grants read-only access to the configuration service for a `movies` domain owned by the account 555555555555. The policy uses wildcards for the actions, since it grants access to all actions that begin with *Describe* or *List*. Note that this will also grant access to any describe or list actions that might be added to the API in the future.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["cloudsearch:Describe*",
                 "cloudsearch:List*"],
      "Resource": "arn:aws:cloudsearch:us-east-1:555555555555:domain/movies"
    }
  ]
}
```

Granting Access to All Amazon CloudSearch Configuration Service Actions

You can grant access to all Amazon CloudSearch configuration service actions by including an `Allow` statement that grants access to all configuration service actions, but not the domain services actions. This enables you to grant administrative access without authorizing a user to upload or retrieve data from a domain. One way to do this is to use a wildcard to grant access to all Amazon CloudSearch actions, and then include a deny statement that blocks access to the domain services actions. The following user-based policy grants access to the configuration service for all domains owned by the 111122223333 account in the `us-west-2` region.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["cloudsearch:*"],
      "Resource": "arn:aws:cloudsearch:us-west-2:111122223333:domain/*"
    },
    {
      "Effect": "Deny",
      "Action": ["cloudsearch:document",
                 "cloudsearch:search",
                 "cloudsearch:suggest"],
      "Resource": "arn:aws:cloudsearch:us-west-2:111122223333:domain/*"
    }
  ]
}
```

Granting Unrestricted Access to All Amazon CloudSearch Services

You can grant unrestricted access to all Amazon CloudSearch services, including all configuration service actions and all domain services with a user-based policy. To do this, you specify wildcards for the actions, region, and domain name. The following policy enables the user to access all Amazon CloudSearch actions for any domain in any region that's owned by the 111122223333 account.

> **Note**
> We recommend that when you give highly privileged access to IAM users, as illustrated in this policy, that you also enable multi-factor authorization (MFA) for those users. For more information, see IAM Best Practices in the IAM User guide.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["cloudsearch:*"],
      "Resource": "arn:aws:cloudsearch:*:111122223333:domain/*"
    }
  ]
}
```

Granting Permission to Upload Documents to an Amazon CloudSearch Domain

You can grant an IAM user permission to upload documents to a search domain by specifying the `cloudsearch:document` action. For example, the following user-based policy enables the user to upload documents to the `movies` domain in `us-east-1` owned by the 111122223333 account.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["cloudsearch:document"],
      "Resource": "arn:aws:cloudsearch:us-east-1:111122223333:domain/movies"
    }
  ]
}
```

Granting Amazon CloudSearch Access to Another AWS Account

You have two options to configure cross-account access for a CloudSearch domain:

Option	Description
Configure an IAM role for cross-account access.	Increased security, but requires complex request signing. For more information, see Cross-Account API Access Using IAM Roles in the IAM documentation.

Option	Description
Attach a resource-based policy to the CloudSearch domain and attach a user-based managed policy to an IAM role.	Easier to implement. For more information, see Creating a Role to Delegate Permissions to an IAM User and Walkthrough: Delegating Access Across AWS Accounts For Accounts You Own Using IAM Roles in the IAM documentation.

This topic provides an example of the second option, adding a resource-based policy to the CloudSearch domain. Assume that account #1 is owned by account id `111111111111` and account #2 is owned by account id `999999999999`. Account #1 wants to grant access to account #2 to use the search service for the `movies` domain, which requires two steps:

1. Account #1 attaches a resource-based policy to the domain using the Amazon CloudSearch console that grants access to account #2.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"search_only",
      "Effect":"Allow",
      "Action":["cloudsearch:search"],
      "Principal":{"AWS":["arn:aws:iam::999999999999:root"]}
    }
  ]
}
```

2. Account #2 attaches a user-based managed policy to an IAM role owned by that account using the IAM console.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["cloudsearch:search"],
      "Resource": "arn:aws:cloudsearch:us-east-1:111111111111:domain/
movies"
    }
  ]
}
```

Important
To configure resource-based policies for Amazon CloudSearch, you must have permission to use the `cloudsearch:UpdateServiceAccessPolicies` action.

Granting Access to an Amazon CloudSearch Domain from Selected IP Addresses

Resource-based access policies set through the Amazon CloudSearch configuration service support anonymous access, which enables you to submit unsigned requests to a search domain's services. To allow anonymous access from selected IP addresses, use a wildcard for the `Principal` value and specify the allowed IP addresses as a `Condition` element in the policy.

Important
Allowing anonymous access from selected IP addresses is inherently less secure than requiring user credentials to access your search domains. We recommend against allowing anonymous access even if it is permitted only from select IP addresses. If you currently allow anonymous access, you should upgrade your applications to submit signed requests and control access by configuring user-based or resource-based policies.

If you are creating a resource-based policy that grants access to requests coming from an Amazon EC2 instance, you need to specify the instance's public IP address.

IP addresses are specified in the standard Classless Inter-Domain Routing (CIDR) format. For example 10.24.34.0/24 specifies the range 10.24.34.0 - 10.24.34.255, while 10.24.34.0/32 specifies the single IP address 10.24.34.0. For more information about CIDR notation, see RFC 4632.

For example, the following policy grants access to the search action for the `movies` domain owned by AWS account 111122223333 from the IP address 192.0.2.0/32.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"search_only",
      "Effect":"Allow",
      "Principal":"*",
      "Action":["cloudsearch:search"],
      "Condition":{"IpAddress":{"aws:SourceIp":"192.0.2.0/32"}}
    }
  ]
}
```

Granting Public Access to an Amazon CloudSearch Domain's Search Service

If you need to allow public access to your domain's search endpoint, you can configure a resource-based policy with no conditions. This enables unsigned requests to be sent from any IP address.

Important
Allowing public access to a search domain means you have no control over the volume of requests submitted to the domain. Malicious users could flood the domain with requests, impacting legitimate users as well as your operating costs.

For example, the following policy grants public access to the search action for the `movies` domain owned by AWS account 111122223333.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"public_search",
      "Effect":"Allow",
      "Principal":"*",
      "Action":["cloudsearch:search"]
    }
  ]
}
```

Configuring Access for Amazon CloudSearch Using the AWS Management Console

To configure user-based policies

1. Sign in to the Identity and Access Management (IAM) console at https://console.aws.amazon.com/iam/.
2. Configure Amazon CloudSearch permissions by attaching a policy to a user, group, or role. For more information, see Managing Policies (AWS Management Console). For more information about user-based policies for Amazon CloudSearch see Writing Access Policies for Amazon CloudSearch (p. 30).

To configure resource-based policies

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain you want to configure, and then click the domain's **Access Policies** link.
3. In the domain's **Access Policies** pane, choose one of the shortcuts or enter the IP addresses you want to authorize or block. To add additional IP addresses or address ranges to the rule, click the add (+) icon in the **IP Ranges** column. To remove an address or range from the rule, click its delete (-) icon in the **IP Ranges** column. To add a new rule to the policy, click the **Add a New Rule** button. To remove a rule from the policy, click the remove (x) button in the **Remove** column.
4. When you are done making changes to your access rules, click **Submit**. To exit without saving your changes, click **Revert**.

The Amazon CloudSearch console enables you to easily add access rules to authorize or block particular IP addresses or address ranges. However, resource-based policies are not restricted to IP-based policies. You can use the AWS CLI or AWS SDKs to configure resource-based policies that grant access to particular IAM users or AWS accounts.

The console provides five shortcuts for specifying access rules:
Search and Suggester service: Allow all. Document Service: Account owner only. Allow everyone access to all services (not recommended because anyone can upload documents) Deny everyone access to all services (except through the console or by account owner) Copy access policy from another domain

- **Search and Suggester service: Allow all. Document Service: Account owner only**—enables anyone to search your data and get suggestions, but only you will be able to add and delete documents. Your domain's search endpoint will allow anonymous access from any IP address, but only you will have access to the document endpoint.
- **Allow everyone access to all services**—enables *anyone* to search your data and add and delete documents. Your domain's endpoints will allow anonymous access from any IP address.
- **Deny everyone access to all services**—search and document requests must either be submitted through the console or authenticated with your account credentials. The document and search endpoints do not allow anonymous access or accept requests from other AWS users.
- **Copy access policy from another domain**—copy the access policies configured for another of your search domains. (Only visible if you have more than one domain.)

You can start with one of the shortcuts, and add additional rules to fine-tune access to your domain's endpoints. Deny rules take precedence over Allow rules.

Updating resource-based access policies takes some time to complete. The state of a domain's policies is displayed on the **Access Policies** pane. Once the policy has been applied, the state changes from `PROCESSING` to `ACTIVE`.

Configuring Access for Amazon CloudSearch with the AWS CLI

You can configure both user-based policies and resource-based policies for Amazon CloudSearch with the AWS CLI. For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

To configure user-based policies

- Configure Amazon CloudSearch permissions by attaching a policy to a user, group, or role with the `aws put-user-policy`, `aws put-group-policy`, or `aws put-role-policy` command. For more information, see Managing Policies (AWS Management Console). For more information about user-based policies for Amazon CloudSearch see Writing Access Policies for Amazon CloudSearch (p. 30).

To configure resource-based policies

- Run the `aws cloudsearch update-service-access-policies` command and specify an access policy with the `--access-policies` option. The access policy must be enclosed in quotes and all quotes within the access policy must be escaped with a backslash. For more information about resource-based policies for Amazon CloudSearch see Writing Access Policies for Amazon CloudSearch (p. 30).

  Note
  If you are using the 2.0.0.1 version of the Amazon CloudSearch command line tools, you can use the `cs-configure-access-policies` command to configure access to a domain's endpoints. However, we recommend that you migrate to the AWS CLI, which provides a cross-service CLI with a simplified installation, unified configuration, and consistent command line syntax.

  The following example configures the `movies` domain to accept search requests from the IP address `192.0.2.0`.

```
aws cloudsearch update-service-access-policies --domain-name movies
--access-policies "{\"Version\":\"2012-10-17\",\"Statement\":[{
  \"Sid\":\"search_only\",
  \"Effect\":\"Allow\",
  \"Principal\": \"*\",
  \"Action\":\"cloudsearch:search\",
  \"Condition\":{\"IpAddress\":{\"aws:SourceIp\":\"192.0.2.0/32\"}}}
]}"
{
  "AccessPolicies": {
    "Status": {
      "PendingDeletion": false,
      "State": "Processing",
      "CreationDate": "2014-04-30T22:07:30Z",
      "UpdateVersion": 9,
      "UpdateDate": "2014-04-30T22:07:30Z"
    },
    "Options":
      "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":\"\",
        \"Effect\":\"Allow\",\"Principal\":\"*\",
```

```
        \"Action\":\"cloudsearch:search\",
        \"Condition\":{\"IpAddress\":{\"aws:SourceIp\":
        \"192.0.2.0/32\"}}}]}"
    }
}
```

Updating resource-based access policies takes some time to complete. You can check the state of the policy with the `aws cloudsearch describe-service-access-policies` command. Once the policy has been applied, the state of the policy changes to `Active`.

You can retrieve your domain's policies using the `aws cloudsearch describe-service-access-policies` command.

Configuring Access to a Domain's Endpoints Using the AWS SDKs

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `UpdateServiceAccessPolicies (p. 197)`. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Configuring Scaling Options in Amazon CloudSearch

A search domain has one or more search instances, each with a finite amount of RAM and CPU resources for indexing data and processing requests. You can configure scaling options to control the instance type that is used, the number of instances your search index is distributed across (partition count), and the number of replicas of each index partition (replication count). All instances for a domain are always of the same type.

You can configure the desired instance type, partition count, or replication count for an Amazon CloudSearch domain to:

- **Increase upload capacity** By default, all search domains start out on a `search.m1.small` instance. You can increase your domain's document upload capacity by changing the desired instance type. If you have a large amount of data to upload—for example, when you are initially populating your search domain—you can choose a larger instance type to increase the number of updates that can be submitted in parallel and reduce how long it takes to index your data. If you are already using the largest instance type, you can increase the desired partition count to further increase upload capacity. For more information, see Bulk Uploads (p. 89). Note that increasing the desired replication count does *not* generally increase a domain's upload capacity.
- **Speed up search requests.** Choosing a larger desired instance type can also speed up search requests. If you've tuned your requests and still aren't meeting your performance targets, try choosing a larger instance type. If you are already using the largest instance type, you can increase the desired partition count to further boost query performance. For more information, see Tuning Search Request Performance in Amazon CloudSearch (p. 111).
- **Increase search capacity.** By default, Amazon CloudSearch uses one instance per index partition. When Amazon CloudSearch scales the domain automatically, it adds replicas based on the resources needed to process the query traffic. To increase a domain's search capacity, you set the desired replication count. However, deploying additional instances takes some time. If you know in advance that you will need additional capacity—for example, before a big launch or announcement—add replicas ahead of time to ensure that your search domain is ready to handle the load.

- **Improve fault tolerance.** Increasing the desired replication count also improves the domain's fault-tolerance—if there's a problem with one of the replicas, the others will continue to handle requests while it is being recovered. However, note that the replicas reside in the same Availability Zone. If you need to ensure availability of your domain in the event of an Availability Zone service disruption, you should enable the MultiAZ option. For more information, see Configuring Availability Options (p. 42).

When you set the desired instance type, desired number of replicas, or desired partition count, Amazon CloudSearch scales your domain as necessary, but will never scale the domain to an instance type that's smaller than the desired type, use fewer replicas than the desired number of replicas, or reduce the partition count below the desired partition count.

You can change your scaling options at any time. If your need for additional capacity is temporary, you can prescale your domain by setting the scaling options and then revert the changes after your volume of uploads or queries returns to your domain's steady state. When you make changes, you need to re-index your domain, which can take some time for the changes to take effect. How long it takes to re-index depends on the amount of data in your index. You can monitor the domain status to determine when indexing is complete—the status changes from PROCESSING to ACTIVE.

Topics

Choosing Scaling Options in Amazon CloudSearch

When you set scaling options for a domain, you make a trade-off between cost and performance—changing the desired instance type, replication count, and partition count can significantly impact the cost of running your domain.

To determine which instance type to select to handle your upload traffic, monitor your upload performance as you increase the upload rate. If you start seeing a large number of 504 or 507 errors before you reach your desired upload rate, select a larger instance type. If you are already on the largest instance type, you can increase the number of partitions to further increase upload capacity.

For datasets of less than 1 GB of data or fewer than one million 1 KB documents, a small search instance should be sufficient. To upload data sets between 1 GB and 8 GB, we recommend setting the desired instance type to `search.m3.large` before you begin uploading. For datasets between 8 GB and 16 GB, start with a `search.m3.xlarge`. For datasets between 16 GB and 32 GB, start with a `search.m3.2xlarge`. If you have more than 32 GB to upload, select the `search.m3.2xlarge` instance type and increase the desired partition count to accommodate your data set. Each partition can contain up to 32 GB of data. Submit a Service Increase Limit Request if you need more upload capacity or have more than 500 GB to index.

To determine how many replicas you need to handle a given query volume, do some testing using a sample of your expected queries at the rate you need to support. Keep in mind that query performance depends heavily on the type of queries being processed. In general, searches that return a large volume of hits and complex structured queries are more resource intensive than simple text queries that match a small percentage of the documents in your search domain. If you expect a high volume of complex queries, choose a larger desired instance type and increase the desired replication count.

Configuring Scaling Options through the Amazon CloudSearch Console

To configure a search domain's scaling options

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console. In the **Navigation** pane, click the name of the domain you want to configure, and then click the **Scaling Options** link.
2. Select an instance type from the **Desired Instance Type** menu. If you select the `search.m3.2xlarge` instance type, you also have the option of setting the **Desired Partition Count**. You should increase the desired partition count if you have more data to upload than will fit on a single `search.m3.2xlarge` partition. For more information, see Bulk Uploads (p. 89).
3. Select the number of replicas you want to use from the **Desired Replication Count** menu.
4. Select the number of index partitions you want to use from the **Desired Partition Count** menu.
5. Click **Submit** to save your changes and then click **OK** to confirm that you want to modify your domain's scaling options. Note that changing the desired instance type and replication count can significantly increase the cost of running your domain. To exit without saving your changes, click **Cancel** and then click **Revert**.
6. After you finish making changes to your domain configuration, click `Run Indexing` to update and deploy your index to the new instances.

Configuring Scaling Options through the AWS CLI

You use the `aws cloudsearch update-scaling-parameters` command to configure scaling options for a search domain. For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

To configure a search domain's scaling options

- Run the `aws cloudsearch update-scaling-parameters` command. You can specify the desired instance type and desired replication count. If you choose the largest instance type (`search.m3.2xlarge`), you can also set the desired partition count. For example, the following command sets the desired instance type to `search.m3.xlarge` and the desired replication count to two. You must specify both the `--domain-name` and `--scaling-parameters` options.

```
aws cloudsearch update-scaling-parameters --domain-name movies --scaling-
parameters DesiredInstanceType=search.m3.xlarge,DesiredReplicationCount=2
{
    "ScalingParameters": {
        "Status": {
            "PendingDeletion": false,
            "State": "RequiresIndexDocuments",
            "CreationDate": "2014-06-25T21:41:21Z",
            "UpdateVersion": 10,
            "UpdateDate": "2014-06-25T21:41:21Z"
        },
        "Options": {
            "DesiredInstanceType": "search.m3.xlarge",
            "DesiredReplicationCount": 2
        }
    }
}
```

For the changes to take effect, you must initiate an index build. You can rebuild the index by calling `aws cloudsearch index-documents`.

Configuring Scaling Options through the AWS SDK

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `UpdateScalingParameters` (p. 195). For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Configuring Availability Options in Amazon CloudSearch

You can expand an Amazon CloudSearch domain to an additional Availability Zone in the same region to increase fault tolerance in the event of a service disruption. Availability Zones are physically separate locations with independent infrastructure engineered to be insulated from failures in other Availability Zones. For more information, see Regions and Availability Zones in the Amazon EC2 User Guide for Linux Instances.

When you turn on the Multi-AZ option, Amazon CloudSearch provisions and maintains extra instances for your search domain in a second Availability Zone to ensure high availability. The maximum number of Availability Zones a domain can be deployed in is two.

Turning on Multi-AZ does not affect a search domain's service endpoints or increase the volume of data or traffic your search domain can handle. Updates are automatically applied to the instances in both Availability Zones. Search traffic is distributed across all of the instances and the instances in either zone are capable of handling the full load in the event of a failure.

If there's an Availability Zone service disruption or the instances in one zone become degraded, Amazon CloudSearch routes all traffic to the other Availability Zone. Redundant instances are restored in a separate Availability Zone without any administrative intervention or disruption in service.

You expand an existing search domain to a second Availability Zone by turning on the Multi-AZ option. Similarly, you can turn off the Multi-AZ option to downgrade the domain to a single Availability Zone. Turning the Multi-AZ option on or off takes about half an hour.

You can configure a domain's availability options through the Amazon CloudSearch console, using the `aws cloudsearch update-availability-options` command, or the AWS SDKs.

> **Important**
> If your domain is running on a single search instance, enabling the Multi-AZ option adds a second search instance in a different availability zone, which doubles the cost of running your domain. Similarly, if your index is split across multiple partitions, a new instance is deployed in the second Availability Zone for each partition. Additional replicas are added to ensure that either Availability Zone has enough capacity to handle all of your traffic—when Multi-AZ is enabled, your domain will have at least one replica of each index partition. If you set the desired number of replicas and enable the Multi-AZ option, Amazon CloudSearch ensures that you have at least that many replicas available in total across the two availability zones. You can monitor the number of instances being used for your domain from the domain dashboard.

Topics

Configuring Availability Options through the Amazon CloudSearch Console

To configure a search domain's availability options

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console. In the **Navigation** pane, click the name of the domain you want to configure, and then click the **Availability Options** link.
2. To turn on the Multi-AZ option, click **Turn Multi-AZ on**. To turn off the Multi-AZ option, click **Turn Multi-AZ off**.
3. When prompted, click **OK** to confirm that you want to modify your domain's availability options. If your domain currently uses a single search instance, turning on the Multi-AZ option adds a second search instance, which can significantly increase the cost of running your domain. To exit without saving your changes, click **Cancel**.

Configuring Amazon CloudSearch Availability Options Using the AWS CLI

You use the `aws cloudsearch update-availability-options` command to configure availability options for a search domain. For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

> **Note**
> If you are using the 2.0.0.1 version of the Amazon CloudSearch command line tools, you can use the `cs-configure-availability-options` command to update availability options. However, we recommend that you migrate to the AWS CLI, which provides a cross-service CLI with a simplified installation, unified configuration, and consistent command line syntax.

To configure a search domain's availability options

- Run the `aws cloudsearch update-availability-options` command and specify the `--multi-az` option to turn on MultiAZ for the domain, or `--no-multi-az` to turn MultiAZ off. For example, the following request enables MultiAZ for the `movies` domain:

```
aws cloudsearch update-availability-options --domain-name movies --multi-
az

{
    "AvailabilityOptions": {
        "Status": {
            "PendingDeletion": false,
            "State": "Processing",
            "CreationDate": "2014-04-30T20:42:57Z",
            "UpdateVersion": 13,
            "UpdateDate": "2014-05-01T00:17:45Z"
        },
        "Options": true
    }
}
```

Configuring Availability Options through the AWS SDK

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `UpdateAvailabilityOptions (p. 189)`. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Monitoring Amazon CloudSearch Domains

The AWS Management Console enables you to easily monitor the status and configuration of your search domains and view your Amazon CloudSearch usage. You can also get configuration information about particular domains with the AWS CLI and AWS SDKs.

Topics

Getting Information About an Amazon CloudSearch Domain

You can retrieve the following information about each of your search domains:

- **Domain Name**—The name of the domain.
- **ARN**—The domain's Amazon Resource Name (ARN).
- **Document Endpoint**—The endpoint through which you can submit document updates.
- **Search Endpoint**—The endpoint through which you can submit search requests.
- **Searchable Documents**—The number of documents that have been indexed.
- **Access Policies**—The access policies configured for the domain's document and search endpoints.
- **Analysis Schemes**—The text analysis schemes that can be applied to the domain's index fields.
- **Index Fields**—The name and type of each configured index field.
- **Expressions**—The expressions that can be used for sorting search results.
- **Suggesters**—The suggesters that can be used to retrieve suggestions for incomplete queries.

When a domain is first created, the domain status will indicate that the domain is currently being activated and no other information is available. Once your domain's document and search endpoints are available, the domain status shows the endpoint addresses that you can use to add data and submit search requests. If you haven't submitted any data for indexing, the number of searchable documents is zero.

You can view all of the information about your domain through the Amazon CloudSearch console (p. 45). When you use the `aws cloudsearch describe-domains` command or the AWS SDKs, the domain's ARN is shown within the domain's access policies.

To get the number of searchable documents, use the console or submit a `matchall` request to your domain's search endpoint.

```
q=matchall&q.parser=structured&size=0
```

Topics

Getting Domain Information Using the Amazon CloudSearch Console

You can use the Amazon CloudSearch console to view information about all of your domains. The dashboard of the console shows a summary of each domain that you have created, including the domain name, status, and number of searchable documents. To update the table with the latest information, click the **Refresh** button at the top of the page.

A domain can be in one of five states:

- **LOADING**—The domain has just been created and is still being initialized. You must wait until the domain status changes to PROCESSING, NEEDS INDEXING, or ACTIVE before you can start uploading documents.
- **ACTIVE**—The domain is running and all configured fields have been indexed.
- **NEEDS INDEXING**—You have made changes to the domain configuration that require rebuilding the index. If you search the domain, these changes won't be reflected in the results. When you are done making changes, click **Run Indexing** to rebuild your index.
- **PROCESSING**—Configuration changes are being applied to your domain. If you search the domain, the most recent configuration changes might not be reflected in the results.
- **BEING DELETED**—You chose to delete the domain and its contents, and the domain and all of its resources are in the process of being removed. When deletion is complete, the domain will be removed from the list of domains.

From the Amazon CloudSearch dashboard, you can do the following:

- View the status of your search domains
- Access the dashboard for a particular domain
- Access the Amazon CloudSearch documentation and other resources

To view detailed information about a particular domain

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. Click the name of the domain in the **Navigation** pane.

The domain dashboard shows the status summary for the selected domain. From the domain dashboard, you can do the following:

- View the status of the domain
- Upload documents to the domain
- Search the domain
- Access the domain configuration pages
- Delete the domain

To view the access policies configured for the domain

- Click the domain's **Access Policies** link in the **Navigation** pane. For more information about access policies, see Configuring Access for Amazon CloudSearch (p. 29).

To view the availability options configured for the domain

- Click the domain's **Availability Options** link in the **Navigation** pane. For more information about access policies, see Configuring Availability Options (p. 42).

To view the index fields configured for the domain

- Click the domain's **Indexing Options** link in the **Navigation** pane. For more information about index fields, see Configuring Index Fields (p. 64).

To view the scaling options configured for the domain

- Click the domain's **Scaling Options** link in the **Navigation** pane. For more information about index fields, see Configuring Scaling Options (p. 39).

To view the suggesters configured for the domain

- Click the domain's **Suggesters** link in the **Navigation** pane. For more information about index fields, see Configuring Suggesters for Amazon CloudSearch (p. 124).

To view the expressions configured for the domain

- Click the domain's **Expressions** link in the **Navigation** pane. For more information about expressions, see Configuring Expressions (p. 129).

To view the text-processing options configured for the domain

- Click the domain's **Analysis Schemes** link in the **Navigation** pane. For information about text options, see Configuring Analysis Schemes (p. 70).

Getting Amazon CloudSearch Domain Information Using the AWS CLI

You use the `aws cloudsearch describe-domains` command to get the status of your search domains. To get specific information such as the access policies, availability options, and scaling options configured for a domain, you use the separate describe command for each option. For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

> **Note**
> If you are using the 2.0.0.1 version of the Amazon CloudSearch command line tools, you can use the `cs-describe-domain` command to get information about your search domains. However, we recommend that you migrate to the AWS CLI, which provides a cross-service CLI with a simplified installation, unified configuration, and consistent command line syntax.

To get domain status information

- Run the `aws cloudsearch describe-domains` command to get information about all of your domains. To get information about specific domains, use the `--domain-names` option to specify

the domains that you are interested in. For example, the following request gets the status of the `movies` domain:

```
aws cloudsearch describe-domains --domain-names movies

{
    "DomainStatusList": [
        {
            "SearchInstanceType": "search.m1.small",
            "DomainId": "965407640801/movies",
            "Created": true,
            "Deleted": false,
            "SearchInstanceCount": 1,
            "DomainName": "movies",
            "SearchService": {
                "Endpoint": "search-movies-m4fcjhuxgj6i76smhyiz7pfxsu.us-
east-1.cloudsearch.amazonaws.com"
            },
            "RequiresIndexDocuments": false,
            "Processing": true,
            "DocService": {
                "Endpoint": "doc-movies-m4fcjhuxgj6i76smhyiz7pfxsu.us-
east-1.cloudsearch.amazonaws.com"
            },
            "ARN": "arn:aws:cloudsearch:us-east-1:965407640801:domain/
movies",
            "SearchPartitionCount": 1
        }
    ]
}
```

The `describe-domains` command does not return the number of searchable documents in the domain. To get the number of searchable documents, use the console or submit a `matchall` request to your domain's search endpoint:

```
q=matchall&q.parser=structured&size=0
```

To get the analysis schemes configured for a domain

- Run the `aws cloudsearch describe-analysis-schemes` command. For example, the following request gets the analysis schemes configured for the `movies` domain:

```
aws cloudsearch describe-analysis-schemes --domain-name movies

{
    "AnalysisSchemes": [
        {
            "Status": {
                "PendingDeletion": false,
                "State": "Active",
                "CreationDate": "2014-03-28T19:27:30Z",
                "UpdateVersion": 31,
                "UpdateDate": "2014-03-28T19:27:30Z"
            },
            "Options": {
                "AnalysisSchemeLanguage": "en",
```

```
                "AnalysisSchemeName": "samplescheme",
                "AnalysisOptions": {
                    "AlgorithmicStemming": "none",
                    "Synonyms": "{\"aliases\":{\"youth\":[\"young adult
\"]},\"groups\":[[\"tool box\",\"toolbox\"],[\"band saw\",\"bandsaw\"],
[\"drill press\",\"drillpress\"]]}",
                    "StemmingDictionary": "{}",
                    "Stopwords": "[]"
                }
            }
        }
    ]
}
```

To get the availability options configured for a domain

- Run the `aws cloudsearch describe-availability-options` command. For example, the following request gets the availability options configured for the `movies` domain. If Multi-AZ is enabled for the domain, the `Options` value is set to `true`:

```
aws cloudsearch describe-availability-options --domain-name movies

{
    "AvailabilityOptions": {
        "Status": {
            "PendingDeletion": false,
            "State": "Processing",
            "CreationDate": "2014-04-30T20:42:57Z",
            "UpdateVersion": 13,
            "UpdateDate": "2014-05-01T00:17:45Z"
        },
        "Options": true
    }
}
```

To get the expressions configured for a domain

- Run the `aws cloudsearch describe-expressions` command. For example, the following request gets the expressions configured for the `movies` domain:

```
aws cloudsearch describe-expressions --domain-name movies

{
    "Expression": {
        "Status": {
            "PendingDeletion": false,
            "State": "Processing",
            "CreationDate": "2014-05-01T01:15:18Z",
            "UpdateVersion": 52,
            "UpdateDate": "2014-05-01T01:15:18Z"
        },
        "Options": {
            "ExpressionName": "popularhits",
            "ExpressionValue": "((0.3*popularity)/10.0)+(0.7* _score)"
        }
```

```
    }
}
```

Getting Domain Information Using the AWS SDKs

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `DescribeDomains (p. 177)`. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

The `DescribeDomains` action does not return the number of searchable documents in the domain. To get the number of searchable documents, use the console or submit a `matchall` request to your domain's search endpoint:

```
q=matchall&q.parser=structured&size=0
```

Monitoring an Amazon CloudSearch Domain with Amazon CloudWatch

Amazon CloudSearch automatically sends metrics to Amazon CloudWatch so that you can gather and analyze performance statistics. You can monitor these metrics by using the Amazon CloudSearch console, or by using the CloudWatch console, AWS CLI, or AWS SDKs. Each of your domain's search instances sends metrics to CloudWatch at one-minute intervals. The metrics are archived for two weeks; after that period, the data is discarded.

There is no charge for the Amazon CloudSearch metrics that are reported through CloudWatch. If you set alarms on the metrics, you will be billed at standard CloudWatch rates. You can use the metrics in all regions supported by Amazon CloudSearch.

Topics

Not all statistics, such as *Average* or *Sum*, are applicable for every metric. However, all of these values are available through the Amazon CloudSearch console, or by using the CloudWatch console, AWS CLI, or AWS SDKs for all metrics. In the following table, each metric has a list of Valid Statistics that is applicable to that metric.

Amazon CloudSearch Metrics

The `AWS/CloudSearch` namespace includes the following metrics.

Metric	Description
`SuccessfulRequests`	The number of search requests successfully processed by a search instance. Units: Count Valid statistics: Maximum, Sum
`SearchableDocuments`	The number of searchable documents in the domain's search index. Units: Count

Metric	Description
	Valid statistics: Maximum
`IndexUtilization`	The percentage of the search instance's index capacity that has been used. The Maximum value indicates the percentage of the domain's index capacity that has been used. Units: Percent Valid statistics: Average, Maximum
`Partitions`	The number of partitions the index is distributed across. Units: Count Valid statistics: Minimum, Maximum

Dimensions for Amazon CloudSearch Metrics

Amazon CloudSearch sends the ClientId and DomainName dimensions to CloudWatch.

Dimension	Description
`ClientId`	The AWS account ID.
`DomainName`	The name of the search domain.

Viewing CloudWatch Metrics for an Amazon CloudSearch Domain

The Amazon CloudSearch console graphs the metrics reported to CloudWatch. You can also access the metrics through the CloudWatch console, AWS CLI, and AWS SDKs. For more information, see Viewing, Graphing, and Publishing Metrics in the *Amazon CloudWatch Developer Guide*.

To view metrics for a search domain using the Amazon CloudSearch console

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch.
2. In the navigation pane, click the name of the domain, and then click the domain's **Monitoring** link.

Logging Amazon CloudSearch Configuration Service Calls Using AWS CloudTrail

Amazon CloudSearch is integrated with CloudTrail, a service that logs all AWS API calls made by, or on behalf of, your AWS account. The log files are delivered to the Amazon S3 bucket that you specify. CloudTrail captures all Amazon CloudSearch configuration service API calls, including those submitted by the Amazon CloudSearch console.

You can use the information collected by CloudTrail to monitor activity for your search domains. You can determine what request was made to Amazon CloudSearch, the source IP address from which the request was made, who made the request, when it was made, and so on. To learn more about CloudTrail, including how to configure and enable it, see the *AWS CloudTrail User Guide*.

Amazon CloudSearch Information in CloudTrail

When CloudTrail logging is enabled in your AWS account, API calls made to Amazon CloudSearch actions are tracked in log files. Amazon CloudSearch records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

All Amazon CloudSearch configuration service actions are logged. For example, calls to `CreateDomain`, `DescribeDomains`, and `UpdateServiceAccessPolicies` generate entries in the CloudTrail log files. For the complete list of actions, see Actions (p. 151).

Every log entry contains information about who generated the request. The user identity information in the log helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information, see the `userIdentity` field in the CloudTrail Event Reference.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

You can choose to have CloudTrail publish Amazon SNS notifications when new log files are delivered if you want to take quick action upon log file delivery. For more information, see Configuring Amazon SNS Notifications.

You can also aggregate Amazon CloudSearch log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket.

Understanding Amazon CloudSearch Log File Entries

CloudTrail log files contain one or more log entries where each entry is made up of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any parameters, the date and time of the action, and so on. The log entries are not guaranteed to be in any particular order—they are not an ordered stack trace of the public API calls.

CloudTrail log files include events for all AWS API calls for your AWS account, not just calls to the Amazon CloudSearch configuration service API. However, you can read the log files and scan for the `eventSource` `cloudsearch.amazonaws.com`. The `eventName` element contains the name of the configuration service action that was called.

The following example shows a CloudTrail log for a user who created a search domain and then configured an index field for the domain. The corresponding API calls (`CreateDomain` and `DefineIndexField`) are shown in the `eventName` element for each record. Information about the user (Alice) is shown in the `userIdentity` element:

```
{
  "Records": [
    {
      "eventVersion": "1.01",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAI2JXM4FBZZEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-12T19:47:32Z",
      "eventSource": "cloudsearch.amazonaws.com",
      "eventName": "CreateDomain",
```

```
      "awsRegion": "us-east-1",
      "sourceIPAddress": "198.51.100.0",
      "requestParameters": {
        "domainName": "imdb-movies"
      },
      "responseElements": {
        "domainStatus": {
          "created": true,
          "searchService": {

          },
          "processing": false,
          "docService": {

          },
          "domainName": "imdb-movies",
          "domainId": "123456789012\/imdb-movies",
          "requiresIndexDocuments": false,
          "searchPartitionCount": 0,
          "deleted": false,
          "arn": "arn:aws:cloudsearch:us-east-1:123456789012:domain\/imdb-
movies",
          "searchInstanceCount": 0
        }
      },
    {
      "eventVersion": "1.01",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAI2JXM4FBZZEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-12T19:47:34Z",
      "eventSource": "cloudsearch.amazonaws.com",
      "eventName": "DefineIndexField",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "198.51.100.0",
      "requestParameters": {
        "domainName": "imdb-movies",
        "indexField": {
          "indexFieldType": "text",
          "indexFieldName": "plot",
          "textOptions": {
            "highlightEnabled": true,
            "returnEnabled": true,
            "analysisScheme": "_en_default_",
            "sortEnabled": true
          }
        }
      },
      "responseElements": {
        "indexField": {
          "options": {
            "indexFieldType": "text",
            "indexFieldName": "plot",
            "textOptions": {
              "highlightEnabled": true,
```

```
              "returnEnabled": true,
              "analysisScheme": "_en_default_",
              "sortEnabled": true
            }
          },
          "status": {
            "pendingDeletion": false,
            "state": "RequiresIndexDocuments",
            "updateDate": "Sep 12, 2014 12:47:33 PM",
            "creationDate": "Sep 12, 2014 12:47:33 PM",
            "updateVersion": 5
          }
        }
      },
      "requestID": "98c6c9f4-7e0f-4982-ae43-67a183e74968",
      "eventID": "3a7fe907-b482-46de-9f25-0ac035e84d1d"
    }
  ]
}
```


Tracking your Amazon CloudSearch Usage and Charges

The AWS account activity page enables you to track your Amazon CloudSearch usage and charges.

To get your Amazon CloudSearch usage information

1. Go to aws.amazon.com and select **Account Activity** from the **My Account/Console** menu. (If you are not already logged in to the AWS portal, you will be prompted to enter your user name and password.)
2. Scroll down to the CloudSearch entry in the **Details** table and click **Download Usage Report**.
3. Specify the information you want to include in the report and click the download button for the data format that you want to download. Reports can be downloaded as either XML or CSV.

Deleting an Amazon CloudSearch Domain

If you are no longer using a search domain, you must delete it to avoid incurring additional usage fees. You will still be charged for a domain even if it does not contain any documents—deleting all documents does not delete the domain. Deleting a domain deletes the index associated with the domain and takes the domain's document and search endpoints offline permanently. It can take some time to completely remove a domain and decommission all of its resources. Small domains are typically deleted in a short amount of time, while especially large domains may require an extended amount of time to be deleted. During this process, the domain status is set to `BEING DELETED` and your account is not charged.

You can delete a domain from the Amazon CloudSearch console, using the `aws cloudsearch delete-domain` command, or using the AWS SDKs.

Topics

Deleting a Domain Using the Amazon CloudSearch Console

You can easily delete a domain from the domain dashboard in the Amazon CloudSearch console.

To delete a domain

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain you want to delete.
3. On the domain dashboard, click the **Delete this Domain** button.
4. In the **Delete Domain** dialog box, enable the checkbox and click **OK** to confirm that you want to delete the domain.

Deleting a Domain Using the AWS CLI

You use the `aws cloudsearch delete-domain` command to delete a search domain and all of its resources. For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

> **Note**
> If you are using the 2.0.0.1 version of the Amazon CloudSearch command line tools, you can use the `cs-delete-domain` command to delete a search domain. However, we recommend that you migrate to the AWS CLI, which provides a cross-service CLI with a simplified installation, unified configuration, and consistent command line syntax.

To delete a domain

- Run the `aws cloudsearch delete-domain` command and specify the name of the domain you want to delete. For example, to delete the *movies* domain, you specify `--domain-name movies`.

```
aws cloudsearch delete-domain --domain-name movies
```

Deleting Amazon CloudSearch Domains Using the AWS SDKs

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `DeleteDomain (p. 166)`. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Tagging Amazon CloudSearch Domains

Use Amazon CloudSearch tags to attach metadata to your search domains. AWS does not apply any semantic meaning to your tags; tags are interpreted strictly as character strings. All tags contain the following elements.

Tag Element	Description
Tag key	The tag key is the required name of the tag. Tag keys must be unique for the domain to which they are attached. For a list of basic restrictions on tag keys and values, see Tag Restrictions.
Tag value	The tag value is an optional string value of the tag. Tag values can be null and do not have to be unique in a tag set. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity. For a list of basic restrictions on tag keys and values, see Tag Restrictions.

Each Amazon CloudSearch domain has a tag set, which contains all the tags that are assigned to that domain. AWS does not automatically set any tags on Amazon CloudSearch domains. A tag set can contain as many as ten tags, or it can be empty. If you add a tag to an Amazon CloudSearch domain that has the same key as an existing tag for a resource, the new value overwrites the old value.

You can use a tag key to define a category, and the tag value can be a item in that category. For example, you could define a tag key of `project` and a tag value of `Salix` indicating that the domain is assigned to the Salix project. You could also use tags to designate domains for test or production environments by using keys such as `environment=test` and `environment=production`. We recommend that you use a consistent set of tag keys to make it easier to track metadata associated with your search domains.

You also can use tags to organize your AWS bill to reflect your own cost structure and to track costs by grouping expenses for similarly tagged resources. To do this, sign up to get your AWS account bill with tag key values included. Then, organize your billing information according to resources with the same tag key values to see the cost of combined resources. For example, you can tag several Amazon CloudSearch domains with key-value pairs, and then organize your billing information to see the total cost for each domain across several services. For more information, see Cost Allocation and Tagging in the *AWS Billing and Cost Management* documentation.

> **Note**
> Tags are cached for authorization purposes. Because of this, additions and updates to tags on Amazon CloudSearch domains might take several minutes before they are available.

Working with Tags (Console)

Use the following procedure to create a resource tag with the Amazon CloudSearch console.

To create a tag

1. Go to https://aws.amazon.com and choose Sign In to the Console.
2. Under Application Services, choose **CloudSearch**.
3. On the navigation pane, choose your domain.
4. On the navigation pane, choose **Manage tags**.
5. In the **Key** column, enter a tag key.
6. (Optional) In the **Value** column, enter a tag value.
7. Choose **Submit**.

To delete a tag

1. Go to https://aws.amazon.com and choose Sign In to the Console.
2. Under Application Services, choose **CloudSearch**.
3. On the navigation pane, choose your domain.

4. On the navigation pane, choose **Manage tags**.
5. Next to the tag that you want to delete, choose **Remove tag**.
6. Choose **Submit**.

For more information about using the console to work with tags, see Working with the Tag Editor in the *AWS Management Console Getting Started Guide*.

Working with Tags (AWS CLI)

You can create resource tags for your Amazon CloudSearch domains using the AWS CLI with the **add-tags** command.

Parameter	Description
`--arn`	Amazon resource name for the domain to which the tag is attached.
`--tag-list`	Set of space-separated key-value pairs in the following format: `Key=<key>,Value=<value>`

Example

The following example creates two tags for the `logs` domain:

```
aws cloudsearch add-tags --arn arn:aws:cs:us-east-1:1:379931976431:domain/
logs --tag-list Key=service,Value=CloudSearch Key=instances,Value=m3.2xlarge
```

You can remove tags from a domain using the **remove-tags** command.

Syntax

```
RemoveTags --arn=<domain_arn> --tag-keys Key=<key>,Value=<value>
```

Parameter	Description
`--arn`	Amazon Resource Name (ARN) for the domain to which the tag is attached.
`--tag-keys`	Set of space-separated tag keys that you want to remove from the domain.

Example

The following example removes two tags from the `logs` domain that were created in the preceding example:

```
aws cloudsearch remove-tags --arn arn:aws:cs:us-east-1:379931976431:domain/
logs --tag-keys service instances
```

You can view the existing tags for a domain with the **list-tags** command:

Syntax

```
list-tags --arn=<domain_arn>
```

Parameter	Description
`--arn`	Amazon Resource Name (ARN) for the domain to which the tags are attached.

Example

The following example lists all resource tags for the `logs` domain:

```
aws cloudsearch list-tags --arn arn:aws:cs:us-east-1:379931976431:domain/logs
```

Working with Tags (AWS SDKs)

The AWS SDKs (except the Android and iOS SDKs) support all of the actions defined in the Amazon CloudSearch Configuration API Reference (p. 148), including the `AddTags`, `ListTags` and `RemoveTags` commands. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Controlling How Data is Indexed in Amazon CloudSearch

You control how your data is indexed by configuring indexing options and analysis schemes for your domain. Indexing options control how your data is mapped to index fields and what information you can search and retrieve from the index. The data you upload must contain the same fields configured in your domain's indexing options, and the field values must be compatible with the configured field types. Analysis schemes control how `text` and `text-array` fields are processed during indexing by defining language-specific stemming, stopword, and synonym options.

Topics

Preparing Your Data for Amazon CloudSearch

You need to format your data in JSON or XML before you can upload it to your search domain for indexing. Each item that you want to be able to receive as a search result is represented as a document. Every document has a unique document ID and one or more fields that contain the data that you want to search and return in results. These document fields are used to populate the index fields you configure for your domain. For more information, see Configuring Index Fields (p. 64).

Creating Document Batches (p. 59) describes how to format your data. For a detailed description of the Amazon CloudSearch JSON and XML schemas, see the Document Service API Reference (p. 230).

Topics

Mapping Document Data to Index Fields in Amazon CloudSearch

To populate the fields in your index, Amazon CloudSearch reads the data from the corresponding document fields. Every field specified in your document data must be configured in your indexing options. Documents can contain a subset of the fields configured for the domain—every document does not have to contain all fields. In addition, you can populate additional fields in your index by copying the data from one field to another. This enables you to use the same source data in different ways by configuring different options for the fields.

An array field such as `text-array` can contain up to 1000 values. At search time, the document is returned as a hit if any of those values match the search query.

Creating Document Batches in Amazon CloudSearch

You create document batches to describe the data that you want to make searchable. When you send document batches to a domain, the data is indexed automatically according to the domain's indexing options. The command line tools and Amazon CloudSearch console can automatically generate document batches from a variety of source documents.

A document batch is a collection of add and delete operations that represent the documents you want to add, update, or delete from your domain. Batches can be described in either JSON or XML. The maximum batch size is 5 MB. The maximum size of an individual document is 1 MB.

To get the best possible upload performance, group add and delete operations in batches that are close to the maximum batch size. Submitting a large volume of single-document batches to the document service can increase the time it takes for your changes to become visible in search results. If you have a large amount of data to upload, you can send batches in parallel. The number of simultaneous uploaders you can use depends on the search instance type. You can prescale for bulk uploads by setting the desired instance type option for your domain. For more information, see Configuring Scaling Options (p. 39).

For each document in a batch, you must specify:

- The operation you want to perform: *add* or *delete*.
- A unique ID for the document. A document ID can contain any letter or number and the following characters: _ - = # ; : / ? @ &. Document IDs must be at least 1 and no more than 128 characters long.
- A name-value pair for each document field. To specify the value for a `latlon` field, you specify the latitude and longitude as a comma-separated list; for example, `"location_field": "35.628611,-120.694152"`. When specifying documents in JSON, the value for a field cannot be `null`. (You can, however, omit the field entirely.)

For example, the following JSON batch adds one document and deletes one document:

```
[
 {"type": "add",
  "id":   "tt0484562",
  "fields": {
    "title": "The Seeker: The Dark Is Rising",
    "directors": "Cunningham, David L.",
    "genres": ["Adventure","Drama","Fantasy","Thriller"],
    "actors": ["McShane, Ian","Eccleston, Christopher","Conroy, Frances",
               "Crewson, Wendy","Ludwig, Alexander","Cosmo, James",
```

```
              "Warner, Amelia","Hickey, John Benjamin","Piddock, Jim",
              "Lockhart, Emma"]
  }
 },
 {"type": "delete",
  "id":   "tt0484575"
 }
]
```

The same batch formatted in XML looks like this:

```
<batch>
 <add  id="tt0484562">
  <field name="title">The Seeker: The Dark Is Rising</field>
  <field name="directors">Cunningham, David L.</field>
  <field name="genres">Adventure</field>
  <field name="genres">Drama</field>
  <field name="genres">Fantasy</field>
  <field name="genres">Thriller</field>
  <field name="actors">McShane, Ian</field>
  <field name="actors">Eccleston, Christopher</field>
  <field name="actors">Conroy, Frances</field>
  <field name="actors">Ludwig, Alexander</field>
  <field name="actors">Crewson, Wendy</field>
  <field name="actors">Warner, Amelia</field>
  <field name="actors">Cosmo, James</field>
  <field name="actors">Hickey, John Benjamin</field>
  <field name="actors">Piddock, Jim</field>
  <field name="actors">Lockhart, Emma</field>
 </add>
 <delete id="tt0484575" />
</batch>
```

Uploading document batches that contain invalid JSON or XML will produce unpredictable results. Processing stops when an error is encountered, but the preceding add and delete operations are applied to the domain. You can verify the validity of your JSON or XML data using tools such as `xmllint` and `jsonlint`.

Both JSON and XML batches can only contain UTF-8 characters that are valid in XML. Valid characters are the control characters tab (0009), carriage return (000D), and line feed (000A), and the legal characters of Unicode and ISO/IEC 10646. FFFE, FFFF, and the surrogate blocks D800–DBFF and DC00–DFFF are invalid and will cause errors. (For more information, see Extensible Markup Language (XML) 1.0 (Fifth Edition).) You can use the following regular expression to match invalid characters so you can remove them: `/[^\u0009\u000a\u000d\u0020-\uD7FF\uE000-\uFFFD]/`.

When formatting your data in JSON, quotes (") and backslashes (\) within field values must be escaped with a backslash. For example:

```
"title":"Where the Wild Things Are"
"isbn":"0-06-025492-0"
"image":"images\\covers\\Where_The_Wild_Things_Are_(book)_cover.jpg"
"comment":"Sendak's \"Where the Wild Things Are\" is a children's classic."
```

When formatting your data in XML, ampersands (&) and less-than symbols (<) within field values need to be represented with the corresponding entity references (`&` and `<`).

For example:

```
<field name="title">Little Cow &amp;amp; the Turtle</field>
<field name="isbn">0-84466-4774</field>
<field name="image">images\covers\Little_Cow_&amp;amp;_the_Turtle.jpg</field>
<field name="comment">&lt;insert comment></field>
```

If you have large blocks of user-generated content, you might want to wrap the entire field in a CDATA section, rather than replacing every occurrence with the entity reference. For example:

```
<field name="comment"><!CDATA[Monsters & mayhem--what's not to like! ]]>
```

Adding and Updating Documents in Amazon CloudSearch

An add operation specifies either a new document that you want to add to the index or an existing document that you want to update.

When you add or update a document, you specify the document's ID and all of the fields the document contains. You don't have to specify every configured field for every document—documents can contain a subset of the configured fields. However, every field in the document must correspond to a field configured for the domain.

To add a document to a search domain

1. Specify an add operation that contains the ID of the document you want to add and each of the fields that you want to be able to search or return in results. If the document already exists, the add operation will replace it. (You cannot update selected fields, the document is overwritten with the new version.) For example, the following operation adds document tt0484562:

   ```
   { "type": "add",
     "id":   "tt0484562",
     "fields": {
       "title": "The Seeker: The Dark Is Rising",
       "directors": ["Cunningham, David L."],
       "genres": ["Adventure","Drama","Fantasy","Thriller"],
       "actors": ["McShane, Ian","Eccleston, Christopher","Conroy, Frances",
                  "Crewson, Wendy","Ludwig, Alexander","Cosmo, James",
                  "Warner, Amelia","Hickey, John Benjamin","Piddock, Jim",
                  "Lockhart, Emma"]
     }
   }
   ```

2. Include the add operation in a document batch and upload the batch to your domain. You should avoid uploading individual documents and batch operations in up to 5 MB batches. (Uploading a large number of single-document batches slows the update process.) You can upload data through the Amazon CloudSearch console, using the `cs-import-documents` command, or by posting a request directly to the domain's document service endpoint. For more information, see Uploading Data to an Amazon CloudSearch Domain (p. 87).

Deleting Documents in Amazon CloudSearch

A delete operation specifies a document that you want to remove from a domain's index. Once a document is deleted, it will no longer be searchable or returned in results.

When posting updates to delete documents, you have to specify each document that you want to delete.

If your domain has scaled up to accommodate your index size and you delete a large number of documents, the domain scales down the next time the full index is rebuilt. Although the index is automatically rebuilt periodically, to scale down as quickly as possible you can explicitly run indexing (p. 91) when you are done deleting documents.

Note
To delete documents, you upload document batches that contain delete operations. You are billed for the total number of document batches uploaded to your search domain, including batches that contain delete operations. For more information about Amazon CloudSearch pricing, see aws.amazon.com/cloudsearch/pricing/.

To delete a document from a search domain

1. Specify a delete operation that contains the ID of the document you want to remove. For example, the following operation would remove document tt0484575:

```
{ "type": "delete",
  "id":   "tt0484575"
}
```

2. Include the delete operation in a document batch and upload the batch to your domain. You should avoid deleting individual documents and batch operations in up to 5 MB batches. (Uploading a large number of single-document batches slows the update process.) You can upload batches through the Amazon CloudSearch console, using the `cs-import-documents` command, or by posting a request directly to the domain's document service endpoint. For more information, see Uploading Data to an Amazon CloudSearch Domain (p. 87).

Processing Your Source Data for Amazon CloudSearch

To upload data for indexing, you need to format your data in either JSON or XML. The command line tools and Amazon CloudSearch console provide a way to automatically generate properly formatted JSON or XML from several common file types: PDF, Microsoft Excel, Microsoft PowerPoint, Microsoft Word, CSV, text, and HTML. You can also process batches formatted for the Amazon CloudSearch 2011-02-01 API to convert them to the 2013-01-01 format.

For most file types, each source file is represented as a separate document in the generated JSON or XML. If metadata is available for the file, the metadata is mapped to corresponding document fields—the fields generated from the document metadata vary depending on the file type. The contents of the source file are parsed into a single text field. If the file contains more than 1 MB of data, the data mapped to the text field is truncated so that the document does not exceed 1 MB.

CSV files are handled differently. When processing a CSV file, Amazon CloudSearch uses the contents of the first row to define the document fields, and creates a separate document for each following row. If there is a column header called *docid*, the values in that column are used as the document IDs. If necessary, the docid values are normalized to conform to the allowed character set. A document ID can contain any letter or number and the following characters: _ - = # ; : / ? @ &. If there is no docid column, a unique ID is generated for each document based on the filename and row number.

If you upload multiple types of files, CSV files are parsed row-by-row, and non-CSV files are treated as individual documents.

Note
Currently, only CSV files are parsed to automatically extract custom field data and generate multiple documents.

You can also process data stored in DynamoDB. Amazon CloudSearch represents each item read from the table as a separate document.

Processing Source Data Using the Amazon CloudSearch Console

When you upload source documents or DynamoDB items through the Amazon CloudSearch console, they are automatically converted to the Amazon CloudSearch JSON format. You can use the console to upload up to 5 MB of data at a time. If you choose, you can download the generated JSON file. For more information about uploading data through the console, see Uploading Data to an Amazon CloudSearch Domain (p. 87) and Uploading DynamoDB Data (p. 109).

Processing Source Data Using the Amazon CloudSearch Command Line Tools

You use the `cs-import-documents` command to process local files, data stored in Amazon S3, or data from a DynamoDB table and upload it to a search domain for indexing. You can also store the generated JSON or XML files locally or in Amazon S3

To process your source data

- Run the `cs-import-documents` command and specify the source data you want to process with the `--source` option. You can process data from multiple locations by specifying multiple sources. For example: `--source c:\DataSet1 c:\DataSet2`. The `cs-import-documents` command also supports wildcards for filenames, directories, and S3 prefixes: ? (matches any single character), * (matches zero or more characters), ** (matches zero or more directories or prefixes).

  To upload the processed data directly to a search domain, specify the `--domain` option. To save the processed data to a your local filesystem or Amazon S3 instead of uploading it, specify `--output` option. By default, `cs-import-documents` outputs your data in JSON. To generate XML, specify the `-format xml` option.

  If you are reading data from your local file system or Amazon S3, you can use the `--modified-after` option to restrict processing to files or Amazon S3 objects modified after a particular time. If you are reading data from an DynamoDB table, you can specify a start key to read from a particular point in the table and the number of rows to read. For more information about the `start-hash-key`, `start-range-key`, and `--num-rows` options see cs-import-documents (p. 144).

  For example, the following command processes the contents of the `myAmazingDataSet` directory and saves the resulting XML document batches to `c:\myAmazingDataSet\XML`.

```
cs-import-documents --source c:\myAmazingDataSet\*
--modified-after 2014-02-28T00:00:00PDT -format xml
--output c:\myAmazingDataSet\XML
```

To process CSV data

- Run the `cs-import-documents` command and specify the CSV file(s) you want to process with the `--source` option. By default:
  - Each row is parsed as a separate document. You can disable this by specifying the `--single-doc-per-csv` option.
  - The character used to delimit fields in a CSV file is assumed to be a comma (,). You can set the delimiter to a different character, such as a semicolon (;) or TAB (\t), with the `--delimiter` option.
  - Each field is assumed to contain a single value. To extract multiple values from one or more fields, use the `--multivalued` option to specify the fields that contain multiple values. (If you don't specify any fields, all fields other than `docid` are processed as multi-valued fields.)

- The character used to encapsulate individual values of a multi-valued field in a CSV file is assumed to be a double quote ("). You can set the encapsulator to a different character, such as a single quote ('), with the `--encapsulator` option.
- The character used to identify comments in a CSV file is assumed to be a hash character (#). You can set the comment character to a different character, such as an asterisk (*), with the `--comment-character` option.

For example, the following command processes the tab-delimited CSV files in the `myAmazingDataSet` directory and treats the fields as multi-valued fields where individual values are enclosed in single quotes.

```
cs-import-documents -d mydomain --source c:\myAmazingDataSet\*.csv
--delimiter \t --multivalued --encapsulator '
```

Configuring Index Fields for an Amazon CloudSearch Domain

Each document that you add to your search domain has a collection of fields that contain the data that can be searched or returned. Every document must have a unique document ID and at least one field.

In your domain configuration, you define an index field for each of the fields that occur in your documents. You cannot upload documents that contain unrecognized fields. However, every document does not have to contain all fields—documents can contain a subset of the fields configured for the domain.

Topics

Amazon CloudSearch supports the following index field types:

- `date`—contains a timestamp. Dates and times are specified in UTC (Coordinated Universal Time) according to IETF RFC3339: `yyyy-mm-ddTHH:mm:ss.SSSZ`. In UTC, for example, 5:00 PM August 23, 1970 is: `1970-08-23T17:00:00Z`. Note that you can also specify fractional seconds when specifying times in UTC. For example, `1967-01-31T23:20:50.650Z`.
- `date-array`—a date field that can contain multiple values.
- `double`—contains a double-precision 64-bit floating point value.
- `double-array`—a double field that can contain multiple values.
- `int`—contains a 64-bit signed integer value.
- `int-array`—an integer field that can contain multiple values.
- `latlon`—contains a location stored as a latitude and longitude value pair (`lat, lon`).
- `literal`—contains an identifier or other data that you want to be able to match exactly. Literal fields are case-sensitive.
- `literal-array`—a literal field that can contain multiple values.
- `text`—contains arbitrary alphanumeric data.

- `text-array`—a text field that can contain multiple values.

Regular index field names must begin with a letter and be at least 3 and no more than 64 characters long. The allowed characters are: a-z (lower-case letters), 0-9, and _ (underscore). The name *score* is reserved and cannot be specified as a field name. All field and expression names must be unique.

Dynamic field names must either begin or end with a wildcard (*). The string before or after the wildcard can contain the same set of characters as a regular index field. For more information about dynamic fields, see the section called “Using Dynamic Fields” (p. 68).

The options you can configure for a field vary according to the field type:

- `HighlightEnabled`—You can get highlighting information for the search hits in any `HighlightEnabled` text field. Valid for: `text`, `text-array`.
- `FacetEnabled`—You can get facet information for any `FacetEnabled` field. Text fields cannot be used for faceting. Valid for: `int`, `int-array`, `date`, `date-array`, `double`, `double-array`, `latlon`, `literal`, `literal-array`.
- `ReturnEnabled`—You can retrieve the value of any `ReturnEnabled` field with your search results. Note that this increases the size of your index, which can increase the cost of running your domain. When possible, it's best to retrieve large amounts of data from an external source, rather than embedding it in your index. Since it can take some time to apply document updates across the domain, critical data such as pricing information should be retrieved from an external source using the returned document IDs. Valid for: `int`, `int-array`, `date`, `date-array`, `double`, `double-array`, `latlon`, `literal`, `literal-array`, `text`, `text-array`.
- `SearchEnabled`—You can search the contents of any `SearchEnabled` field. Text fields are always searchable. Valid for: `int`, `int-array`, `date`, `date-array`, `double`, `double-array`, `latlon`, `literal`, `literal-array`, `text`, `text-array`.
- `SortEnabled`—You can sort the search results alphabetically or numerically using any `SortEnabled` field. Array-type fields cannot be `SortEnabled`. Only sort enabled numeric fields can be used in expressions. Valid for: `int`, `date`, `latlon`, `double`, `literal`, `text`.

You can also specify a default value and a source for any field. Specifying a default value can be important if you are using a numeric field in an expression and that field is not present in every document. Specifying a source copies data from one field to another, enabling you to use the same source data in different ways by configuring different options for the fields. You can use a wildcard (*) when specifying the source name to copy data from all fields that match the specified pattern.

When you add fields or modify existing fields, you must explicitly issue a request to re-index your data when you are done making configuration changes. For more information, see Indexing Document Data (p. 91).

Important
If you change the type of a field and have documents in your index that contain data that is incompatible with the new field type, all fields being processed are put in the `FailedToValidate` state when you run indexing and the indexing operation fails. Rolling back the incompatible configuration change will enable you to successfully rebuild your index. If the change is necessary, you must update or remove the incompatible documents from your index to use the new configuration.

Configuring Individual Index Fields with the AWS CLI

You use the `aws cloudsearch define-index-field` command to configure individual index fields for a search domain. For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

Note
If you are using the 2.0.0.1 version of the Amazon CloudSearch command line tools, you can use the `cs-configure-fields` command to define individual fields. However, we recommend that you migrate to the AWS CLI, which provides a cross-service CLI with a simplified installation, unified configuration, and consistent command line syntax.

To add an index field to your domain

- Run the `aws cloudsearch define-index-field` command and specify the name of the new field with the `--name` option, and the field type with the `--type` option. The following example adds an `int` field called `year` to the movies domain.

```
aws cloudsearch define-index-field --domain-name movies --name year --type
 int
{
    "IndexField": {
        "Status": {
            "PendingDeletion": false,
            "State": "RequiresIndexDocuments",
            "CreationDate": "2014-06-25T23:03:06Z",
            "UpdateVersion": 15,
            "UpdateDate": "2014-06-25T23:03:06Z"
        },
        "Options": {
            "IndexFieldType": "int",
            "IndexFieldName": "year"
        }
    }
}
```

Note
When you add fields or modify existing fields, you must explicitly issue a request to re-index your data when you are done making configuration changes. For more information, see Indexing Document Data (p. 91).

Automatically Configuring Index Fields Based on Document Batches in Amazon CloudSearch

You can use the `cs-configure-from-batches` command in the standalone Amazon CloudSearch command line tools to analyze the contents of a document batch and automatically configure corresponding index fields for a domain.

To automatically configure your domain's index fields

1. Run the `cs-configure-from-batches` command to analyze one or more document batches and configure fields for your domain. (For information about how to create document batches, see Preparing Your Data (p. 58).) For example, to configure fields for the `imdb-movies` domain based on the document batch defined in `moviedata.json`:

```
cs-configure-from-batches --domain-name movies --source moviedata.json
```

2. When prompted, enter `y` to confirm that you want to configure your domain with the specified fields. (You can easily modify the configuration later through the console or using the `aws cloudsearch define-index-field` command.)

```
Configure [imdb-movies] with analyzed fields y/N: y
```

Note
When you add fields or modify existing fields, you must explicitly issue a request to re-index your data when you are done making configuration changes. For more information, see Indexing Document Data (p. 91).

Configuring Index Fields Using the Amazon CloudSearch Console

You can easily configure individual index fields (p. 67) for your domain through the **Indexing Options** panel in the Amazon CloudSearch console.

Configuring Individual Fields Using the Amazon CloudSearch Console

To configure a new index field

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain that you want to configure, and then click the domain's **Indexing Options** link.
3. To create a new index field, click **Add Index Field** to add a field specification to the list. (If you haven't created any fields yet, a blank field specification is shown on the Indexing Options page by default.)
4. Specify a unique name for the field and select the field type: `date`, `date-array`, `double`, `double-array`, `int`, `int-array`, `literal`, `literal-array`, `text`, `text-array`. Field names must begin with a letter and be at least 3 and no more than 64 characters long. The allowed characters are: a-z (lower-case letters), 0-9, and _ (underscore). The name *score* is reserved and cannot be used as a field name.
5. Select the options you want to enable for the field. For more information about specifying indexing options, see Configuring Index Fields (p. 64)
6. Specify a default value for the field (optional). This value is used when no value is specified for the field in the document data.
7. Select the analysis scheme you want to use for each text field. The analysis scheme specifies the language-specific text processing options that are used during indexing. By default, text fields use the `_en_default_` analysis scheme. For more information, see Configuring Analysis Schemes (p. 70).
8. To configure additional fields, click **Add Index Field** and repeat these configuration steps.
9. When you are done configuring fields, click **Submit** to save your changes. To restore the previous field configurations, click **Revert**.

Note
When you add fields or modify existing fields, you must explicitly issue a request to re-index your data when you are done making configuration changes. For more information, see Indexing Document Data (p. 91).

Configuring Amazon CloudSearch Index Fields Using the AWS SDKs

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `DefineIndexField (p. 160)`. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Using Dynamic Fields in Amazon CloudSearch

Dynamic fields provide a way to index documents without knowing in advance exactly what fields they contain. For example, consider the case where you want to search a set of products. You might not know the names of all of the possible product attributes across all product categories, but you can structure your data so that all text-based attributes are stored in fields that end in `_t`, and all integer values are stored in fields that end in `_i`. With dynamic fields, you can map the attribute fields to the appropriate field type without having to configure a field for every possible attribute. This reduces the amount of configuration that you need to do up front, and eliminates the need to modify your domain configuration every time a product with a new attribute is added. You can also use dynamic fields to essentially ignore new fields by mapping them to a field that is not searchable or returnable.

Topics

Configuring Dynamic Fields in Amazon CloudSearch

You designate a field as a dynamic field by specifying a wildcard (*) as the first, last, or only character in the field name. Dynamic field names must either begin or end with a wildcard (*). Multiple wildcards and wildcards embedded within a string are not supported.

A dynamic field's name defines a pattern. The wildcard matches zero or more arbitrary characters. Any unrecognized fields that match that pattern are configured with the dynamic field's indexing options. Regular index fields take precedence over dynamic fields. If a document field name matches both a regular index field and a dynamic field pattern, it is mapped to the regular index field. The options you can configure for dynamic fields are the same as for static fields.

For example, if you establish the naming convention that `_i` is appended to the name of any new `int` field, you can define a dynamic field with the pattern `*_i` that sets the field type to `int` and configures a set of predefined indexing options for new `int` fields. When you add a field such as `review_rating_i`, it's configured according to the `*_i` options and indexed automatically.

If a document field matches more than one dynamic field pattern, the longest matching pattern is used. If the patterns are the same length, the dynamic field that occurs first when the field names are sorted alphabetically is used.

You can define * as a dynamic field to match any fields that don't map to an explicitly defined field or a longer dynamic field pattern. This is useful if you want to simply ignore unrecognized fields. For more information, see Ignoring Unrecognized Document Fields (p. 69).

Dynamic fields count toward the total number of fields defined for a domain. A domain can have a maximum of 1,000 fields, which includes dynamic fields. However, the pattern defined by a single

dynamic field typically matches multiple document fields, so the total number of fields in your index can exceed 1,000. When using dynamic fields, keep in mind that significantly increasing the number of fields in your index can impact query performance.

Adding new fields to your domain configuration can affect how fields that were generated dynamically are validated during indexing. If the validation fails, indexing will fail. For example, if you define a dynamic field called `*_new` and upload documents that contain a field called `rating_new`, the `rating_new` field will be added to your index. If you then explicitly configure a field called `rating_new`, that new field configuration will be used to validate the contents of your document's `rating_new` field when you run indexing. If `*_new` is configured as a `text` field and you configure `rating_new` as an `int` field, validation will fail if the existing `rating_new` fields contain non-integer data.

For more information about configuring index fields, see Configuring Index Fields (p. 64).

Using a Dynamic Field to Ignore Unrecognized Fields in Amazon CloudSearch

Amazon CloudSearch requires that you configure an index field for every field that occurs in the documents you are indexing. In some cases, however, you want to index a particular set of fields and simply ignore everything else. You can use dynamic fields to ignore all unrecognized fields by defining a literal field called * and disabling all indexing options for the field. Any unrecognized fields will inherit those options and will be added to your domain; however, the field contents won't be searchable or returnable, so they'll have minimal impact on the size of your index. (They do, however, count toward the total number of fields configured for the domain.) Similarly, you can selectively ignore fields that match a particular pattern, such as *_n.

To ignore unrecognized fields

1. Configure the fields that you want to index, search, or return in the results.
2. Add a dynamic field that matches any other fields that are found in the documents and disables all indexing options for them:
   - Specify * as the name of the field, with no prefix or suffix string. (You can also specify a more specific pattern to selectively disable fields.)
   - Set the field type to `literal` and disable the `search`, `facet`, and `return` options. Note that the maximum size of a literal field is 4096 Unicode code points.

Because longer dynamic field patterns are matched first, you can still use dynamic fields to configure options for fields that you want to use. Any fields that don't map to a regular index field or a longer dynamic field will match the * pattern.

> **Note**
> When you create a dynamic field with the name *, it means that your index can potentially contain any valid field name. This also means that you can reference any valid field name in your search requests, whether or not it actually exists in your index.

Searching Dynamic Fields in Amazon CloudSearch

You can reference dynamically generated fields by name in your search requests and expressions, just like any other field. For example, to search the dynamically generated field `color_t` for the color `red`, you use the structured query parser:

```
q=color_t:'red'&q.parser=structured
```

If you've defined a catch-all dynamic field (*) to map any fields that don't match regular fields or more specific dynamic field patterns, you can specify *any* valid field name in your search requests, whether or not the field actually exists in your index.

Wildcards are not supported within field names, so you cannot reference the dynamic field itself. For example, specifying `q=*_t:'red'` would return an error.

The options a dynamically generated field inherits from the dynamic field configuration control how you can use the field in your search requests, for example, whether you can search it, get facets or highlights, use it for sorting, or return in it results. Note that dynamically generated fields must be searched explicitly—dynamic fields are NOT included in the fields that are searched by default when you use the simple query parser or do not specify a field when searching with the structured query parser.

You can specify dynamic fields as sources for other fields. A field's source attribute supports wildcards, which enables you to specify a pattern that matches a group of dynamic fields. For example, to search all fields generated from the `*_t` dynamic field, you could create a field called `all_t_fields` and set its source attribute to `*_t`. This copies the contents of all fields whose names end in `_t` into `all_t_fields`. Note, however, that searching this field will search *all* fields that match the pattern, not only dynamically generated fields.

For more information about constructing and submitting search requests, see Searching Your Data with Amazon CloudSearch (p. 94).

Configuring Text Analysis Schemes for Amazon CloudSearch

Amazon CloudSearch enables you to configure a language-specific analysis scheme for each `text` and `text-array` field. An analysis scheme controls how the contents of the field are processed during indexing. Although the defaults for each language work well in many cases, fine-tuning the analysis options enables you to optimize the search results based on your knowledge of the data you are searching. For a list of supported languages, see Supported Languages (p. 80).

An analysis scheme specifies the language of the text to be processed and the following analysis options:

- **Algorithmic stemming**—specifies the level of algorithmic stemming to perform. The available stemming levels vary depending on the language.
- **Japanese Tokenization Dictionary**—specifies overrides of the algorithmic tokenization when processing Japanese. The dictionary specifies how particular sets of characters should be grouped into words.
- **Stemming dictionary**—specifies overrides for the results of the algorithmic stemming. The dictionary maps specific related words to a common root word or stem.
- **Stopwords**—specifies words that should be ignored during indexing and searching.
- **Synonyms**—specifies words that have the same meaning as words that occur in your data and should produce the same search results.

During text processing, field values and search terms are converted to lowercase (case-folded), so stopwords, stems, and synonyms are not case-sensitive. For more information about how Amazon CloudSearch processes text during indexing and when handling search requests, see Text Processing in Amazon CloudSearch (p. 79).

You must specify a language for each analysis scheme and configure an analysis scheme for each `text` and `text-array` field. When you configure fields through the Amazon CloudSearch console or

command line tools, the analysis scheme defaults to the `_en_default_` analysis scheme. If you do not specify analysis options for an analysis scheme, Amazon CloudSearch uses the default options for the specified language. For information about the defaults for each language, see Language Specific Settings (p. 80).

The easiest way to define analysis schemes is through the **Analysis Schemes** page in the Amazon CloudSearch console. You must apply an analysis scheme to a field for it to take effect. You can apply an analysis scheme to a field from the **Indexing Options** page. You can also define analysis schemes and configure an analysis scheme for each field through the command line tools and AWS SDKs.

When you apply a new analysis scheme to an index field or modify an analysis scheme that's in use, you must explicitly rebuild the index (p. 91) for the changes to be reflected in search results.

Topics

Stemming in Amazon CloudSearch

Stemming is the process of mapping related words to a common stem. A stem is typically the root or base word from which variants are derived. For example, *run* is the stem of *running* and *ran*. Stemming is performed during indexing as well as at query time. Stemming reduces the number of terms that are included in the index, and facilitates matches when the search term is a variant of a term that occurs in the content being searched. For example, if you map the term *running* to the stem *run* and then search for *running*, the request matches documents that contain *run* as well as *running*.

Amazon CloudSearch supports both algorithmic stemming and explicit stemming dictionaries. You configure algorithmic stemming by specifying the level of stemming that you want to use. The available levels of algorithmic stemming vary depending on the language:

- none—disable algorithmic stemming
- minimal—perform basic stemming by removing plural suffixes
- light—target the most common noun/adjective inflections and derived suffixes
- full—aggressively stem inflections and suffixes

In addition to controlling the degree of algorithmic stemming that's performed, you can specify a stemming dictionary that maps specific related words to a common stem. You specify the dictionary as a JSON object that contains a collection of string:value pairs that map a term to its stem, for example, `{"term1": "stem1", "term2": "stem2", "term3": "stem3"}`. The stemming dictionary is applied in addition to any algorithmic stemming. This enables you to override the results of the algorithmic stemming to correct specific cases of overstemming or understemming. The maximum size of a stemming dictionary is 500 KB. Stemming dictionary entries must be lowercase.

You use the `StemmingDictionary` key to define a custom stemming dictionary in an analysis scheme. Because you pass the dictionary to Amazon CloudSearch as a string, you must escape all double quotes within the string. For example, the following analysis scheme defines stems for *running* and *jumping*:

```
{
    "AnalysisSchemeName": "myscheme",
    "AnalysisSchemeLanguage": "en",
    "AnalysisOptions": {
        "AlgorithmicStemming": "light",
        "StemmingDictionary": "{\"running\": \"run\",\"jumping\": \"jump\"}"
    }
}
```

If you do not specify the level of algorithmic stemming or a stemming dictionary in your analysis scheme, Amazon CloudSearch uses the default algorithmic stemming level for the specified language. While stemming can help users find relevant documents that might otherwise be excluded from the search results, overstemming can result in too many matches with questionable relevance. The default level of algorithmic stemming configured for each language works well for most use cases. In general, it's best to start with the default and then make adjustments to optimize the relevance of the search results for your use case. For information about the defaults for each language, see Language Specific Settings (p. 80).

Stopwords in Amazon CloudSearch

Stopwords are words that should typically be ignored both during indexing and at search time because they are either insignificant or so common that including them would result in a massive number of matches.

During indexing, Amazon CloudSearch uses the stopword dictionary when it processes `text` and `text-array` fields. In most cases, stopwords are not included in the index. The stopword dictionary is also used to filter search requests.

A stopwords dictionary is a JSON array of terms, for example, `["a", "an", "the", "of"]`. The stopwords dictionary must explicitly list each word that you want to ignore. Wildcards and regular expressions are not supported.

You use the `Stopwords` key to define a custom stopwords dictionary in an analysis scheme. Because you pass the dictionary to Amazon CloudSearch as a string, you must escape all double quotes within the string. For example, the following analysis scheme configures the stopwords *a*, *an*, and *the*:

```
{
    "AnalysisSchemeName": "myscheme",
    "AnalysisSchemeLanguage": "en",
    "AnalysisOptions": {
        "Stopwords": "[\"a\",\"an\",\"the\"]"
    }
}
```

If you do not specify a stopwords dictionary in your analysis scheme, Amazon CloudSearch uses the default stopword dictionary for the specified language. The default stopwords configured for each language work well for most use cases. In general, it's best to start with the default and then make adjustments to optimize the relevance of the search results for your use case. For information about the defaults for each language, see Language Specific Settings (p. 80).

Synonyms in Amazon CloudSearch

You can configure synonyms for terms that appear in the data that you are searching. That way, if a user searches for the synonym rather than the indexed term, the results will include documents that contain the indexed term. For example, you might define custom synonyms to do the following:

- Map common misspellings to the correct spelling
- Define equivalent terms, such as `film` and `movie`
- Map a general term to a more specific one, such as `fish` and `barracuda`
- Map multiple words to a single word or vice versa, such as `tool box` and `toolbox`

When you define a synonym, the synonym is added to the index everywhere the base token occurs. For example, if you define `fish` as a synonym of `barracuda`, the term `fish` is added to every document that contains the term `barracuda`. Adding a large number of synonyms can increase the size of the index as well as query latency—synonyms increase the number of matches and the more matches, the longer it takes to process the results.

The synonym dictionary is used during indexing to configure mappings for terms that occur in text fields. No synonym processing is done on search requests. By default, Amazon CloudSearch does not define any synonyms.

You can specify synonyms in two ways:

- As a *conflation group* where each term in the group is considered a synonym of every other term in the group.
- As an *alias* for a specific term. An alias is considered a synonym of the specified term, but the term is not considered a synonym of the alias.

A synonym dictionary is specified as a JSON object that defines the synonym groups and aliases. The `groups` value is an array of arrays, where each sub-array is a conflation group. The `aliases` value is an object that contains a collection of string:value pairs where the string specifies a term and the array of values specifies each of the synonyms for that term. The following example includes both conflation groups and aliases:

```
{
    "groups": [["1st", "first", "one"], ["2nd", "second", "two"]],
    "aliases": { "youth": ["child", "kid", "boy", "girl"],
                 "adult": ["men", "women"] }
}
```

Both groups and aliases support multiword synonyms. In the following example, multiword synonyms are used in a conflation group as well as an alias:

```
{
    "groups": [["tool box", "toolbox"], ["band saw", "bandsaw"]],
    "aliases": { "workbench": ["work bench"]}
}
```

You use the `Synonyms` key to define a custom synonym dictionary in an analysis scheme. Because you pass the dictionary to Amazon CloudSearch as a string, you must escape all double quotes within the string. For example, the following analysis scheme configures aliases for the term *youth*:

```
{
    "AnalysisSchemeName": "myscheme",
    "AnalysisSchemeLanguage": "en",
    "AnalysisOptions": {
        Synonyms": "{\"aliases\": {\"youth\": [\"child\",\"kid\"]}}"
    }
}
```

Configuring Analysis Schemes Using the Amazon CloudSearch Console

You can define analysis schemes from the **Analysis Schemes** pane in the Amazon CloudSearch console.

To define an analysis scheme

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain, and then click the domain's **Analysis Schemes** link.
3. In the **Analysis Schemes** pane, click **Add Analysis Scheme**.
4. Specify a name for the analysis scheme, select a language, and configure the scheme's text stopword, stemming, and synonym options. You can configure individual stopwords, stems, and synonyms, or edit the displayed dictionaries directly. The dictionaries are formatted in JSON. Stopwords are specified as an array of strings. Stems are specified as an object that contains one or more key:value pairs. Synonym aliases are also specified as a JSON object with one or move key:value pairs, where the alias values are specified as an array of strings. A synonym group is specified as a JSON array. (The synonym dictionary is an array of arrays.)

   If you select Japanese as the language, you also have the option of specifying a custom tokenization dictionary that overrides the default tokenization of specific phrases. For more information, see Customizing Japanese Tokenization (p. 76).
5. Click **Create** to save your changes.

Important
To use an analysis scheme, you must apply it to one or more `text` or `text-array` fields and rebuild the index. You can configure a field's analysis scheme from the **Indexing Options** page. To rebuild your index, click the **Run Indexing** button.

Configuring Analysis Schemes Using the AWS CLI

You use the `aws cloudsearch define-analysis-scheme` command to define language-specific text processing options, including stemming options, stopwords, and synonyms. For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

You specify an analysis scheme as part of the configuration of each `text` or `text-array` field. For more information, see Configuring Index Fields (p. 64).

Note
If you are using the 2.0.0.1 version of the Amazon CloudSearch command line tools, you can use the `cs-configure-analysis-scheme` command to define analysis schemes. However, we recommend that you migrate to the AWS CLI, which provides a cross-service CLI with a simplified installation, unified configuration, and consistent command line syntax.

To define an analysis scheme

- Run the `aws cloudsearch define-analysis-scheme` command and specify the `--analysis-scheme` option and a JSON object that contains your analysis options. The analysis scheme must be valid JSON. The analysis option key and value pairs must be enclosed in quotes, and all quotes within the option values must be escaped with a backslash. For the format of the analysis options, see define-analysis-scheme in the AWS Command Line Interface Reference. See Configuring Analysis Schemes (p. 70) for more information about specifying stemming, stopword, and synonym options.

If you specify Japanese (`ja`) as the language, you also have the option of specifying a custom tokenization dictionary that overrides the default tokenization of specific phrases. For more information, see Customizing Japanese Tokenization (p. 76).

Tip
The easiest way to configure an analysis scheme with the AWS CLI is to store the analysis scheme in a text file and specify that file as the `--analysis-scheme` value. This enables you to format the scheme so that it's easier to read. For example, the following scheme defines an English analysis scheme called `myscheme` that uses light algorithmic stemming and configures two stopwords:

```
{
    "AnalysisSchemeName": "myscheme",
    "AnalysisSchemeLanguage": "en",
    "AnalysisOptions": {
        "AlgorithmicStemming": "light",
        "Stopwords": "[\"a\", \"the\"]"
    }
}
```

If you save this scheme in a text file called `myscheme.txt`, you can pass the file in as the value of the `--analysis-scheme` parameter:

```
aws cloudsearch define-analysis-scheme --region us-east-1 --domain-
name movies --analysis-scheme file://myscheme.txt
```

Important
To use an analysis scheme, you must apply it to one or more `text` or `text-array` fields and rebuild the index. You can configure a field's analysis scheme with the `aws cloudsearch define-index-field` command. To rebuild the index, call `aws cloudsearch index-documents`.

Configuring Analysis Schemes Using the AWS SDKs

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `DefineAnalysisScheme (p. 156)`. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Important
To use an analysis scheme, you must apply it to one or more `text` or `text-array` fields and rebuild the index. You can configure a field's analysis scheme with the define index field method. To rebuild your index, you use the index documents method.

Indexing Bigrams for Chinese, Japanese, and Korean in Amazon CloudSearch

Chinese, Japanese, and Korean do not have explicit word boundaries. Simply indexing individual characters (unigrams) can result in matches that aren't very relevant to a search query. One solution is to index *bigrams*. A bigram is every sequence of two adjacent characters in a string. For example, the following example shows bigrams for the string 我的氣墊船裝滿了鱔魚:

While indexing bigrams can improve search result quality, keep in mind that it can significantly increase the size of your index.

To index bigrams for Chinese, Japanese, and Korean

1. Create a text analysis scheme and set the language to multiple languages (`mul`).
2. Configure the index field that contains the CJK data to use your multi-language analysis scheme.

When you assign an analysis scheme that sets a field's language to `mul`, Amazon CloudSearch automatically generates bigrams for all Chinese, Japanese, and Korean text within the field.

For more information about creating and using analysis schemes, see Configuring Analysis Schemes (p. 70).

If you are indexing Japanese content, you might also be interested in using a custom tokenization dictionary with the standard Japanese language processor. For more information, see Customizing Japanese Tokenization (p. 76).

Customizing Japanese Tokenization in Amazon CloudSearch

If you need more control over how Amazon CloudSearch tokenizes Japanese, you can add a custom Japanese tokenization dictionary to your analysis scheme. Configuring a custom tokenization dictionary enables you to override how specific entries are tokenized by the standard Japanese language processor. This can improve search result accuracy in some cases, particularly when you need to index and retrieve domain-specific phrases.

A tokenization dictionary is a collection of entries where each entry specifies a set of characters, how the characters should be tokenized, how each token should be pronounced (readings), and a part-of-speech tag. You specify the dictionary as an array, and each dictionary entry is an array of strings. The entries are of the following form:

```
["<text>","<token 1> ... <token n>","<reading 1> ... <reading n>","<part-of-speech tag>"]
```

You must specify a reading for each token and the part-of-speech tag for the entry. See Japanese Part-of-Speech Tags (p. 77) for the part of speech tags that are treated as stopwords.

You use the `JapaneseTokenizationDictionary` key to define a custom tokenization dictionary in an analysis scheme. Because you pass the tokenization dictionary to Amazon CloudSearch as a string, you must escape all double quotes within the string. For example, the dictionary in the following analysis scheme specifies segmentation overrides for Kanji and Katakana compounds, and a custom reading for a proper name:

```
    "AnalysisSchemeName": "jascheme",
    "AnalysisSchemeLanguage": "ja",
    "AnalysisOptions": {
        "Stopwords": "[\"a\", \"the\"]",
        "AlgorithmicStemming": "full",
        "JapaneseTokenizationDictionary": "[
            [\"日本経済新聞\",\"日本 経済 新聞\",\"ニホン ケイザイ
            [\"トートバッグ\",\"トート バッグ\",\"トート バッグ\",\
            [\"朝青龍\",\"朝青龍\",\"アサショウリュウ\",\"カスタム人
        ]"
    }
}
```

When configuring an analysis scheme with the AWS CLI, you can store the analysis scheme in a text file and specify that file as the `--analysis-scheme` value. This enables you to format the scheme so that it's easier to read. For example, if you store the `jascheme` analysis scheme in a file called `jascheme.txt`, you can pass that file in when you call `aws cloudsearch define-analysis-scheme`:

```
aws cloudsearch define-analysis-scheme --region us-east-1 --domain-name
mydomain --analysis-scheme file://jascheme.txt
```

For more information about creating and using analysis schemes, see Configuring Analysis Schemes (p. 70).

Japanese Part-of-Speech Tags in Amazon CloudSearch

When you use a custom tokenization dictionary for Japanese, you specify a part-of-speech tag for each entry. If the part-of-speech tag matches one of the tags configured as a stop tag, the entry is treated as a stopword.

The following table shows the part of speech tags configured as stop tags in Amazon CloudSearch.

Stop Tags

Tag	Part-of-Speech	Description
助動詞	Auxiliary-verb	A verb that adds functional or grammatical meaning to the clause in which it appears.
接続詞	Conjunction	Conjunctions that can occur independently.
フィラー	Filler	Aizuchi that occurs during a conversation or sounds inserted as filler.

Tag	Part-of-Speech	Description
非言語音	Non-verbal	Non-verbal sound.
その他-間投	Other-interjection	Words that are hard to classify as noun-suffixes or sentence-final particles.
助詞-副詞化	Particle-adnominalizer	The "ni" and "to" that appear following nouns and adverbs.
助詞-連体化	Particle-adnominalizer	The "no" that attaches to nouns and modifies non-inflectional words.
助詞-副助詞	Particle-adverbial	An adverb used to show position, direction of movement, and so on.
助詞-副助詞／並立助詞／終助詞	Particle-adverbial/conjunctive/final	The particle "ka" when unknown whether it is adverbial, conjunctive, or sentence final.
助詞-格助詞-連語	Particle-case-compound	Compounds of particles and verbs that mainly behave like case particles.
助詞-格助詞-一般	Particle-case-misc	Case particles.
助詞-格助詞-引用	Particle-case-quote	The "to" that appears after nouns, a person's speech, quotation marks, expressions of decisions from a meeting, reasons, judgements, conjectures, and so on.
助詞-格助詞	Particle-case	Case particles where the subclassification is undefined.
助詞-接続助詞	Particle-conjunctive	Conjunctive particles.
助詞-並立助詞	Particle-coordinate	Coordinate particles.
助詞-係助詞	Particle-dependency	Dependency particles.
助詞-終助詞	Particle-final	Final particles.
助詞-間投助詞	Particle-interjective	Particles with interjective grammatical roles.

Tag	Part-of-Speech	Description
助詞-特殊	Particle-special	A particle that does not fit into any of the other classifications. This includes particles that are used in Tanka, Haiku, and other poetry.
助詞	Particle	Unclassified particles.
記号-括弧閉	Symbol-close_bracket	Close bracket:].
記号-読点	Symbol-comma	Comma: ,.
記号-一般	Symbol-misc	A general symbol not in one of the other categories.
記号-括弧開	Symbol-open_bracket	Open bracket: [.
記号-句点	Symbol-period	Periods and full stops.
記号-空白	Symbol-space	Full-width whitespace.
記号	Symbol	Unclassified symbols.

Text Processing in Amazon CloudSearch

During indexing, Amazon CloudSearch processes `text` and `text-array` fields according to the analysis scheme configured for the field to determine what terms to add to the index. Before the analysis options are applied, the text is *tokenized* and *normalized*.

During tokenization, the stream of text in a field is split into separate tokens on detectable boundaries using the word break rules defined in the Unicode Text Segmentation algorithm. For more information, see Unicode Text Segmentation.

According to the word break rules, strings separated by whitespace such as spaces and tabs are treated as separate tokens. In many cases, punctuation is dropped and treated as whitespace. For example, strings are split at hyphens (-) and the at symbol (@). However, periods that are not followed by whitespace are considered part of the token.

Note that strings are not split on case boundaries—*CamelCase* strings are not tokenized.

During normalization, upper case characters are converted to lower case. Accents are typically handled according to the stemming options configured in the field's analysis scheme. (The default analysis scheme for English removes accents.)

Once tokenization and normalization are complete, the stemming options, stopwords, and synonyms specified in the analysis scheme are applied.

When you submit a search request, the text you're searching for undergoes the same text processing so that it can be matched against the terms that appear in the index. However, no text analysis is

performed on the search term when you perform a prefix search. This means that a search for a prefix that ends in `s` typically won't match the singular version of the term when stemming is enabled. This can happen for any term that ends in `s`, not just plurals. For example, if you search the `actor` field in the sample movie data for `Anders`, there are three matching movies. If you search for `Ander*`, you get those movies as well as several others. However, if you search for `Anders*` there are no matches. This is because the term is stored in the index as `ander`, `anders` does not appear in the index.

If stemming is preventing your wildcard searches from returning all of the relevant matches, you can suppress stemming for the text field by setting the `AlgorithmicStemming` option to none, or you can map the data to a `literal` field instead of a `text` field.

Topics

Supported Languages in Amazon CloudSearch

Arabic (ar)	Armenian (hy)	Basque (eu)
Bulgarian (bg)	Catalan (ca)	Chinese - Simplified (zh-Hans)
Chinese - Traditional (zh-Hant)	Czech (cs)	Danish (da)
Dutch (nl)	English (en)	Finnish (fi)
French (fr)	Galician (gl)	German (de)
Greek (el)	Hindi (hi)	Hebrew (he)
Hungarian (hu)	Indonesian (id)	Irish (ga)
Italian (it)	Japanese (ja)	Korean (ko)
Latvian (lv)	Multiple (mul)	Norwegian (no)
Persian (fa)	Portuguese (pt)	Romanian (ro)
Russian (ru)	Spanish (es)	Swedish (sv)
Thai (th)	Turkish (tr)	

Language Specific Text Processing Settings in Amazon CloudSearch

Arabic (ar)

Algorithmic stemming options: `light`

Default analysis scheme: `_ar_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

Armenian (hy)

Algorithmic stemming options: `full`

Default analysis scheme: `_hy_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary

Basque (eu)

Algorithmic stemming options: `full`

Default analysis scheme: `_eu_default_`

- Algorithmic stemming options: `full`
- Default stopword dictionary

Bulgarian (bg)

Algorithmic stemming options: `light`

Default analysis scheme: `_bg_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

Catalan (ca)

Algorithmic stemming options: `full`

Elision filter enabled

Default analysis scheme: `_ca_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary

Chinese - Simplified (zh-Hans)

Algorithmic stemming not supported

Stemming dictionary not supported

Default analysis scheme: `_zh-Hans_default_`

Chinese - Traditional (zh-Hant)

Algorithmic stemming not supported

Stemming dictionary not supported

Default analysis scheme: `_zh-Hant_default_`

Czech (cs)

Algorithmic stemming options: `light`

Default analysis scheme: `_cs_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

Danish (da)

Algorithmic stemming options: `full`

Default analysis scheme: `_da_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary

Dutch (nl)

Algorithmic stemming options: `full`

Default analysis scheme: `_nl_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary
- Default stemming dictionary

English (en)

Algorithmic stemming options: `minimal|light|full`

Default analysis scheme: `_en_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary

Finnish (fi)

Algorithmic stemming options: `light|full`

Default analysis scheme: `_fi_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

French (fr)

Algorithmic stemming options: `minimal|light|full`

Elision filter enabled

Default analysis scheme: `_fr_default_`

- Algorithmic stemming: `minimal`
- Default stopword dictionary

Galician (gl)

Algorithmic stemming options: `minimal|full`

Default analysis scheme: `_gl_default_`

- Algorithmic stemming: `minimal`
- Default stopword dictionary

German (de)

Algorithmic stemming options: `minimal|light|full`

Default analysis scheme: `_de_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

Greek (el)

Algorithmic stemming options: `full`

Default analysis scheme: `_el_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary

Hebrew (h3)

Algorithmic stemming options: `full`

Default analysis scheme: `_he_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary

Hindi (hi)

Algorithmic stemming options: `full`

Default analysis scheme: `_hi_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary

Hungarian (hu)

Algorithmic stemming options: `light|full`

Default analysis scheme: `_hu_default_`

- Algorithmic stemming: `light`

- Default stopword dictionary

Indonesian (id)

Algorithmic stemming options: `light|full`

Default analysis scheme: `id_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary

Irish (ga)

Algorithmic stemming options: `full`

Elision filter enabled

Default analysis scheme: `_ga_default_`

- Algorithmic stemming options: `full`
- Default stopword dictionary

Italian (it)

Algorithmic stemming options: `light|full`

Elision filter enabled

Default analysis scheme: `_it_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

Japanese (ja)

Algorithmic stemming options: `full`

Algorithmic decompounding enabled

Optional tokenization dictionary

Default analysis scheme: `_ja_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary

Korean (ko)

Algorithmic stemming not supported

Algorithmic decompounding enabled

Default analysis scheme: `_ko_default_`

- Default stopword dictionary

Latvian (lv)

Algorithmic stemming: `light`

Default analysis scheme: `_lv_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

Multiple (mul)

Algorithmic stemming: not supported

Default analysis scheme: `_mul_default_`

- Default stopword dictionary

Norwegian (no)

Algorithmic stemming options: `minimal|light|full`

Default analysis scheme: `_no_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

Persian (fa)

Algorithmic stemming not supported

Default analysis scheme: `_fa_default_`

- Default stopword dictionary

Portuguese (pt)

Algorithmic stemming options: `minimal|light|full`

Default analysis scheme: `_pt_default_`

- Algorithmic stemming: `minimal`
- Default stopword dictionary

Romanian (ro)

Algorithmic stemming options: `full`

Default analysis scheme: `_ro_default_`

- Algorithmic stemming: `full`

- Default stopword dictionary

Russian (ru)

Algorithmic stemming options: `light|full`

Default analysis scheme: `_ru_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

Spanish (es)

Algorithmic stemming options: `light|full`

Default analysis scheme: `_es_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

Swedish (sv)

Algorithmic stemming options: `light|full`

Default analysis scheme: `_sv_default_`

- Algorithmic stemming: `light`
- Default stopword dictionary

Thai (th)

Algorithmic stemming not supported

Stemming dictionary not supported

Default analysis scheme: `_th_default_`

- Default stopword dictionary

Turkish (tr)

Algorithmic stemming: `full`

Default analysis scheme: `_tr_default_`

- Algorithmic stemming: `full`
- Default stopword dictionary

Uploading and Indexing Data in Amazon CloudSearch

To make your data searchable, you need to format it in JSON or XML as described in Preparing Your Data (p. 58) and upload it to your search domain for indexing. In most cases, Amazon CloudSearch automatically indexes your data and the changes are visible in search results in just a few minutes. However, certain changes to your domain configuration put the domain in the `NEEDS INDEXING` state. For those changes to take effect, you must explicitly run indexing to rebuild your index. Currently, you also need to periodically run indexing so your suggesters reflect the most recent data in your index. The following sections describe how to upload data to your domain and run indexing when it's needed.

Topics

Uploading Data to an Amazon CloudSearch Domain

You create document batches to describe the data that you want to upload to an Amazon CloudSearch domain. A document batch is a collection of add and delete operations that represent the documents you want to add, update, or delete from your domain. Batches can be described in either JSON or XML. When you upload document batches to a domain, the data is indexed automatically according to the domain's indexing options.

As your data changes, you upload batches to add, change, or delete documents from your index. Amazon CloudSearch applies updates continuously. You only have to explicitly reindex your data when you make configuration changes that put your domain in the `NEEDS INDEXING` state or need to update suggesters.

To upload data to your domain, it must be formatted as a valid JSON or XML batch. The fields specified in each document must correspond to index fields configured for the domain. However, a document does not have to contain every configured index field. For information about creating document batches, see Preparing Your Data (p. 58). For information about configuring index fields for a domain, see Configuring Index Fields (p. 64).

You are billed for the total number of document batches uploaded to your search domain, including batches that contain delete operations. For more information about Amazon CloudSearch pricing, see aws.amazon.com/cloudsearch/pricing/.

> **Important**
> A domain's document and search endpoints remain the same for the life of the domain. You should cache the endpoints rather than retrieving them before every upload or search request. Querying the Amazon CloudSearch configuration service by calling `aws cloudsearch describe-domains` or `DescribeDomains` before every request is likely to result in your requests being throttled.

You can submit a document batch to a domain using the Amazon CloudSearch console (p. 89), AWS CLI, or by posting it directly (p. 91) to the domain's document service endpoint.

For more information about the document service API, see the Document Service API Reference (p. 230).

Topics

Submitting Document Upload Requests to an Amazon CloudSearch Domain

We recommend using one of the AWS SDKs or the AWS CLI to submit document upload requests. The SDKs and AWS CLI handle request signing for you and provide an easy way to perform all Amazon CloudSearch actions. To process source documents and upload the generated JSON or XML batches to your domain in one step, you can use the `cs-import-documents` command in the standalone Amazon CloudSearch command line tools. For more information, see Processing Your Source Data (p. 62). You can also use the Amazon CloudSearch console to upload individual batches and import data from DynamoDB or S3.

> **Important**
> A domain's document and search endpoints remain the same for the life of the domain. You should cache the endpoints rather than retrieving them before every upload or search request. Querying the Amazon CloudSearch configuration service by calling `aws cloudsearch describe-domains` or `DescribeDomains` before every request is likely to result in your requests being throttled.

For example, the following request uploads a batch using the AWS CLI.

```
aws cloudsearchdomain --endpoint-url http://doc-movies-
y6gelr4lv3jeu4rvoelunxsl2e.us-east-1.cloudsearch.amazonaws.com upload-
documents --content-type
 application/json --documents movie-data-2013.json
```

For development and testing purposes, you can allow anonymous access to your domain's document service and submit unsigned HTTP POST requests directly to your domain's document service. In a production environment, restrict access to your domain to specific IAM users, groups, or roles and submit signed requests. For information about controlling access for Amazon CloudSearch, see Configuring Access for Amazon CloudSearch (p. 29). For more information about request signing, see Signing AWS API Requests.

For example, the following POST request uploads a batch of documents formatted in JSON to the domain endpoint doc-movies-123456789012.us-east-1.cloudsearch.amazonaws.com.

```
curl -X POST --upload-file data1.json doc-movies-123456789012.us-
east-1.cloudsearch.amazonaws.com/2013-01-01/documents/batch --header
 "Content-Type: application/json"
```

Bulk Uploads in Amazon CloudSearch

Document batches are limited to 5 MB per batch. However, you can upload batches in parallel to reduce the amount of time it takes to upload all of your data.

To perform a bulk upload:

- Make sure your batches are as close to the 5 MB limit as possible. Uploading a larger amount of smaller batches slows down the upload and indexing process.
- Set your desired instance type to a larger instance type than the default `search.m1.small`. The number of upload threads you can use depends on the type of search instance your domain is using and the nature of your data and indexing options. Larger instance types have a higher upload capacity. Attempting to upload batches in parallel to a `search.m1.small` instance usually results in a high rate of 504 or 507 errors. For more information about setting the desired instance type, see Configuring Scaling Options (p. 39).
- Start uploading data once your configuration changes are active. If you encounter a high rate of 5xx errors, you either need to reduce your upload rate or switch to a larger instance type. If you are already using the largest instance type, you can increase the desired partition count to further increase upload capacity.

  Important
  If you submit a large volume of updates while your domain is in the PROCESSING state, it can increase the amount of time it takes for the updates to be applied to your search index. To avoid this update lag, wait until your domain is in the ACTIVE state before starting your bulk upload.
- When you are finished with your bulk upload, you can change the desired instance type back to a smaller instance type. If your index fits on a smaller type, Amazon CloudSearch will automatically scale your domain back down. Amazon CloudSearch will not scale to an instance type that's smaller than the desired instance type configured for your domain.

For datasets of less than 1 GB of data or fewer than one million 1 KB documents, a small search instance should be sufficient. To upload data sets between 1 GB and 8 GB, we recommend setting the desired instance type to `search.m3.large` before you begin uploading. For datasets between 8 GB and 16 GB, start with a `search.m3.xlarge`. For datasets between 16 GB and 32 GB, start with a `search.m3.2xlarge`. If you have more than 32 GB to upload, select the `search.m3.2xlarge` instance type and increase the desired partition count to accommodate your data set. Each partition can contain up to 32 GB of data. Submit a Service Increase Limit Request if you need more upload capacity or have more than 500 GB to index.

Uploading Data Using the Amazon CloudSearch Console

In the Amazon CloudSearch console, you can upload data from your local file system or Amazon S3 to your domain from the domain dashboard. The console can automatically convert the following types of files to document batches during the upload process:

- Comma Separated Value (.csv)

- Adobe Portable Document Format (.pdf)
- HTML (.htm, .html)
- Microsoft Excel (.xls, .xlsx)
- Microsoft PowerPoint (.ppt, .pptx)
- Microsoft Word (.doc, .docx)
- Text Documents (.txt)

You can also convert and upload items from an DynamoDB table. For more information, see Uploading DynamoDB Data (p. 109).

> **Note**
> To upload data from Amazon S3 or DynamoDB, you must have permission to access both the service and the resources you want to upload. For more information, see Using Bucket Policies and User Policies and Using IAM to Control Access to DynamoDB Resources.

CSV files are parsed row-by-row and a separate document is generated for each row. All other types of files are treated as a single document. For more information about automatically generating document batches, see Preparing Your Data (p. 58).

> **Note**
> Uploading data to Amazon CloudSearch from an Amazon S3 bucket or DynamoDB table requires access to those services and resources.

To send data to a domain for indexing

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain.
3. At the top of the domain dashboard, click **Upload Documents.**
4. Select the location of the data you want to upload to your domain:

   - File(s) on my local disk
   - Object(s) from Amazon S3
   - Item(s) from DynamoDB
   - Predefined data

   If you upload data that isn't formatted as document batches, it will automatically be converted during the upload process.

   > **Note**
   > If a batch is invalid, Amazon CloudSearch converts the content to a valid batch that contains a single content field and generic metadata fields. Since these are not normally the fields configured for the domain, you will get errors stating that the fields don't exist.
5. If you are uploading local files, click **Browse** to choose the file(s) to upload:
6. If you are uploading objects from Amazon S3, select the bucket you want to upload from. To upload the entire contents of the bucket, leave the **Prefix** field empty and click **Add**. To upload selected objects, enter a filter in the **Prefix** field and click **Add**. (You can add multiple prefixes.)
7. If you are uploading items from DynamoDB, select the table you want to upload from. To start reading from a particular item, specify a start key. To limit the read capacity units that can be consumed while reading from the table, enter the maximum percentage of read capacity units.
8. If are uploading predefined sample data, choose the data set that you want to use:
9. Once you've selected the data you want to upload, click **Continue**.
10. In the **Review Documents** step, review the documents to be uploaded and click **Upload Documents** to continue.

11. In the **Document Summary** step, if a document batch has been automatically generated from your data, you can click **Download the generated document batch** to get it. Click **Finish** to return to the domain dashboard.

Uploading Data Using the AWS CLI

You use the `aws cloudsearch upload-documents` command to send document batches to your search domain. For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

Alternatively, you can use the standalone Amazon CloudSearch command line tools to generate document batches and upload them to your domain in a single step with the `cs-import-documents` command. The `cs-import-documents` command enables you to process and upload local data as well as data stored in Amazon S3 and DynamoDB. For more information, see Processing Source Data Using the CLTs (p. 63).

To send document batches to a domain for indexing

- Run the `aws cloudsearch upload-documents` command to upload your batches to your domain. You must specify at least one `--source` option to specify the location of the batch you want to upload.

```
aws cloudsearchdomain --endpoint-url http://doc-movies-
y6gelr4lv3jeu4rvoelunxsl2e.us-east-1.cloudsearch.amazonaws.com upload-
documents --content-type application/json --documents movie-data-2013.json
{
    "status": "success",
    "adds": 5000,
    "deletes": 0
}
```

Posting Documents to an Amazon CloudSearch Domain's Document Service Endpoint via HTTP

You use the `documents/batch (p. 231)` resource to post document batches to your domain to add, update, or remove documents. For example:

```
curl -X POST --upload-file movie-data-2013.json doc-movies-123456789012.us-
east-1.cloudsearch.amazonaws.com/2013-01-01/documents/batch --header
 "Content-Type:application/json"
```

Indexing Document Data with Amazon CloudSearch

When you send document updates to your domain, Amazon CloudSearch automatically updates the domain's search index with the new data. You don't have to do anything for the updates to be indexed. However, if you change the configuration of your domain's index fields or text options, you must explicitly rebuild your search index for those changes to be visible in search results. Because rebuilding the index can take a significant amount of time if you have a lot of data, you should finish making all of your configuration changes before re-indexing your documents.

Important
If you change the type of a field and have documents in your index that contain data that is incompatible with the new field type, all fields being processed are put in the `FailedToValidate` state when you run indexing and the indexing operation fails. Rolling back the incompatible configuration change will enable you to successfully rebuild your index. If the change is necessary, you must update or remove the incompatible documents from your index to use the new configuration.

When you make changes that require re-indexing, the domain status changes to NEEDS INDEXING. While the index is being rebuilt, the domain's status is PROCESSING. You can continue to submit search requests while indexing is in process, but the configuration changes won't be visible in search results until indexing completes and the domain's status changes to ACTIVE. You can also continue to upload document batches to your domain. However, if you submit a large volume of updates while your domain is in the PROCESSING state, it can increase the amount of time it takes for the updates to be applied to your search index. If this becomes an issue, slow your update rate until the domain returns to the ACTIVE state.

Note
Depending on the volume of data, building a full index can take a considerable amount of compute power. Amazon CloudSearch automatically manages the resources needed to build the index in a timely fashion. Most data updates and simple domain configuration changes are built and deployed in minutes. Indexing large volumes of data and applying configuration changes that require rebuilding the full index will take longer to complete.

You can initiate indexing from the Amazon CloudSearch console (p. 92), using the `aws cloudsearch index-documents` command, or through the AWS SDKs.

Topics

Indexing Documents Using the Amazon CloudSearch Console

When you make changes that require your domain's index to be rebuilt, the status shown on the domain dashboard changes to NEEDS INDEXING. The console also displays a message at the top of the configuration pages prompting you to run indexing when you are done making changes.

To run indexing

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain that needs indexing.
3. On the domain dashboard, click the **Run Indexing** button.
4. Click **OK** in the **Starting Indexing** dialog box to return to the domain dashboard.

Indexing Documents Using the Amazon CloudSearch AWS CLI

You use the `aws cloudsearch index-documents` command to rebuild your domain's search index. For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

Note
If you are using the 2.0.0.1 version of the Amazon CloudSearch command line tools, you can use the `cs-index-documents` command to rebuild your index. However, we recommend that you migrate to the AWS CLI, which provides a cross-service CLI with a simplified installation, unified configuration, and consistent command line syntax.

To explicitly index your domain

- Run the `aws cloudsearch index-documents` command. The following example rebuilds the index for a domain called *movies*.

```
aws cloudsearch index-documents --domain-name movies
```

Indexing Documents with the AWS SDK

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `IndexDocuments (p. 187)`. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Searching Your Data with Amazon CloudSearch

You specify the terms or values you want to search for with the q parameter. How you specify the search criteria depends on which query parser you use. Amazon CloudSearch supports four query parsers:

- `simple`—search all `text` and `text-array` fields for the specified string. The simple query parser enables you to search for phrases, individual terms, and prefixes. You can designate terms as required or optional, or exclude matches that contain particular terms. To search particular fields, you can specify the fields you want to search with the `q.options` parameter. The `simple` query parser is used by default if the `q.parser` parameter is not specified.
- `structured`—search specific fields, construct compound queries using Boolean operators, and use advanced features such as term boosting and proximity searching.
- `lucene`—specify search criteria using the Apache Lucene query parser syntax. If you currently use the Lucene syntax, using the `lucene` query parser enables you to migrate your search services to an Amazon CloudSearch domain without having to completely rewrite your search queries in the Amazon CloudSearch structured search syntax.
- `dismax`—specify search criteria using the simplified subset of the Apache Lucene query parser syntax defined by the DisMax query parser. If you are currently using the DisMax syntax, using the `dismax` query parser enables you to migrate your search services to an Amazon CloudSearch domain without having to completely rewrite your search queries in the Amazon CloudSearch structured search syntax.

You can use additional search parameters to control how search results are returned (p. 128) and include additional information (p. 114) such as facets, highlights, and suggestions with your search results.

For information about all of the Amazon CloudSearch search parameters, see the Search API Reference (p. 239).

Topics

Submitting Search Requests to an Amazon CloudSearch Domain

We recommend using one of the AWS SDKs or the AWS CLI to submit search requests. The SDKs and AWS CLI handle request signing for you and provide an easy way to perform all Amazon CloudSearch actions. You can also use the Search Tester in the Amazon CloudSearch console to search your data, browse the results, and view the generated request URLs and JSON and XML responses. For more information, see Searching with the Search Tester (p. 12).

> **Important**
> A domain's document and search endpoints remain the same for the life of the domain. You should cache the endpoints rather than retrieving them before every upload or search request. Querying the Amazon CloudSearch configuration service by calling `aws cloudsearch describe-domains` or `DescribeDomains` before every request is likely to result in your requests being throttled. Note that your domain's IP address **CAN** change over time, so it's important to cache the endpoint as shown in the console and returned by the `aws cloudsearch describe-domains` command rather than the IP address. For more information, see Setting the JVM TTL for DNS Name Lookups.

For example, the following request submits a simple text search for `wolverine` using the AWS CLI and returns just the IDs of the matching documents.

```
aws cloudsearchdomain --endpoint-url http://search-movies-
y6gelr4lv3jeu4rvoelunxsl2e.us-east-1.cloudsearch.amazonaws.com search --
search-query wolverine  --return _no_fields
{
    "status": {
        "rid": "/rnE+e4oCAqfEEs=",
        "time-ms": 6
    },
    "hits": {
        "found": 3,
        "hit": [
            {
                "id": "tt1430132"
            },
            {
                "id": "tt0458525"
            },
            {
                "id": "tt1877832"
            }
        ],
        "start": 0
    }
}
```

By default, Amazon CloudSearch returns the response in JSON. You can get the results formatted in XML by specifying the `format` parameter. Setting the response format only affects responses

to successful requests. The format of an error response depends on the origin of the error. Errors returned by the search service are always returned in JSON. 5xx errors due to server timeouts and other request routing problems are returned in XML.

For development and testing purposes, you can allow anonymous access to your domain's search service and submit unsigned HTTP GET or POST requests directly to your domain's search endpoint. In a production environment, restrict access to your domain to specific IAM users, groups, or roles and submit signed requests using the AWS SDKs or AWS CLI. For information about controlling access for Amazon CloudSearch, see Configuring Access for Amazon CloudSearch (p. 29). For more information about request signing, see Signing AWS API Requests.

You can use any method you want to send HTTP requests directly to your domain's search endpoint—you can enter the request URL directly in a Web browser, use cURL to submit the request, or generate an HTTP call using your favorite HTTP library. To specify your search criteria, you specify a query string that specifies the constraints for your search and what you want to get back in the response. The query string must be URL-encoded. The maximum size of a search request submitted via GET is 8190 bytes, including the HTTP method, URI, and protocol version. You can submit larger requests using HTTP POST; however, keep in mind that large, complex requests take longer to process and are more likely to time out. For more information, see Tuning Search Request Performance in Amazon CloudSearch (p. 111).

For example, the following request submits a structured query to the `search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.amazonaws.com` domain and gets the contents of the `title` field.

```
http://search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.
amazonaws.com/2013-01-01/search?q=(and+(term+field%3Dtitle+'star')
(term+field%3Dyear+1977))&q.parser=structured&return=title
```

Important
Special characters in the query string must be URL-encoded. For example, you must encode the = operator in a structured query as `%3D`: `(term+field%3Dtitle+'star')`. If you don't encode the special characters when you submit the search request, you'll get an `InvalidQueryString` error.

Searching with the Search Tester

The search tester in the Amazon CloudSearch console enables you to submit sample search requests using any of the supported query parsers: simple, structured, lucene, or dismax. By default, requests are processed with the simple query parser. You can specify options for the selected parser, filter and sort the results, and browse the configured facets. The search hits are automatically highlighted in the search results. For information about how this is done, see Highlighting Search Hits in Amazon CloudSearch (p. 123). You can also select a suggester to get suggestions as you enter terms in the **Search** field. (You must configure a suggester before you can get suggestions. For more information see Getting Autocomplete Suggestions in Amazon CloudSearch (p. 123).)

By default, results are sorted according to an automatically-generated relevance score, *_score*. For information about customizing how results are ranked, see Sorting Results in Amazon CloudSearch (p. 128).

To search your domain

You can specify additional options for the selected query parser to configure the default operator and control which operators can be used in a query. For more information, see Search Request Parameters (p. 241).

1. Go to the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.

2. In the **Navigation** panel, click the name of your movies domain and then click the **Run a Test Search** link.
3. To perform a simple text search, enter the text you want to search for and click **Go**. By default, all `text` and `text-array` fields are searched.
4. To search particular fields, click the **More Parameters** link and enter a comma-separated list of the fields you want to search in the **Search Fields** field. You can append a weight to each field with a caret (^) to control the relative importance of each field in the search results. For example, specifying `title^5, description` weights hits in the `title` field five times more than hits in the `description` field when calculating relevance scores for each matching document.
5. To use the structured query syntax, select **Structured** from the **Query Parser** menu. Once you've selected the structured query parser, enter your structured query in the **Search** field and click **Go**. For example, to find all of the movies with *star* in the title that were released in the year 2000 or earlier, you could enter: `(and title:'star' year:{,2000])`. For more information, see Constructing Compound Queries (p. 97). To submit Lucene or DisMax queries, select the appropriate query parser.

To view the HTTP search request that was sent to your domain's search endpoint and the response returned by Amazon CloudSearch, click the **view raw** link for the response format you want to see.

You can copy and paste the request URL to submit the request and view the response from a Web browser. Requests can be sent via HTTP or HTTPS.

Constructing Compound Queries in Amazon CloudSearch

You can use the structured query parser to combine match expressions using Boolean `and`, `or`, and `not` operators. To select the structured query parser, you include `q.parser=structured` in your query. The structured query operators are specified as *prefix* operators. The syntax is:

- `(and boost=N EXPRESSION1 EXPRESSION2 ... EXPRESSIONn)`
- `(or boost=N EXPRESSION1 EXPRESSION2 ... EXPRESSIONn)`
- `(not boost=N EXPRESSION)`

For example, the following query matches all movies in the sample data set that contain *star* in the title, and either Harrison Ford or William Shatner appear in the `actors` field, but Zachary Quinto does not.

```
(and title:'star' (or actors:'Harrison Ford' actors:'William Shatner')(not
 actors:'Zachary Quinto'))
```

When using the structured query operators, you specify the name of the operator, options for the operator, and then the match expression being operated on, `(OPERATOR OPTIONS EXPRESSION)`. The match expression can be a simple text string, or a subclause of your compound query. Any options must be specified before the terms. For example, `(and (not field=genres 'Sci-Fi')(or (term field=title boost=2 'star')(term field=plot 'star')))`.

Parentheses control the order of evaluation of the expressions. When an expression is enclosed in parentheses, that expression is evaluated first, and then the resulting value is used in the evaluation of the remainder of the compound query.

Important
You must URL-encode special characters in the query string. For example, you must encode the = operator in a structured query as `%3D`: `(term+field%3Dtitle+'star')`. Amazon

CloudSearch returns an `InvalidQueryString` error if special characters are not URL-encoded. For a complete reference of URL-encodings, see the W3C HTML URL Encoding Reference.

For example, the following query searches the `title` field for the phrase `star wars` and excludes matches that have a value less than 2000 in the `year` field.

```
(and (phrase field='title' 'star wars') (not (range field=year {,2000})))
```

To submit this search request, you need to encode the query string and specify the `structured` query parser with the `q.parser` parameter.

```
http://search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.
amazonaws.com/2013-01-01/search?q=(and+(phrase+field='title'+'star
 wars')+(not+(range+field%3Dyear+{,2000})))&q.parser=structured
```

The structured query syntax enables you to combine searches against multiple fields. If you don't specify a field to search, all `text` and `text-array` fields are searched. For example, the following query searches all `text` and `text-array` fields for the term *star*, and excludes documents that contain *Zachary Quinto* in the `actors` field.

```
(and 'star' (not actors:'Zachary Quinto'))
```

You can specify a `boost` value to increase the importance of one expression in a compound query in relation to the others. The boost value increases the scores of the matching documents. For example, the following query boosts matches for the term *star* if they occur in the `title` field rather than the `description` field.

```
(and (range field=year [2013,}) (or (term field=title boost=2 'star') (term
 field=plot 'star'))
```

Boost values must be greater than zero.

In addition to `and`, `or`, and `not`, the Amazon CloudSearch structured search syntax supports several specialized operators:

- `matchall`—Matches every document in the domain. Syntax: `matchall`.
- `near`—Supports sloppy phrase queries. The `distance` value specifies the maximum number of words that can separate the words in the phrase; for example, `(near field='plot' distance=4 'naval mutiny demonstration')`. Use the `near` operator to enable matching when the specified terms are in close proximity, but not adjacent. For more information about sloppy phrase searches, see Searching for Phrases (p. 101). Syntax: `(near field=FIELD distance=N boost=N 'STRING')`.
- `phrase`—Searches for a phrase in `text` or `text-array` fields; for example, `(phrase field="title" 'teenage mutant ninja')`. Supports boosting documents that match the expression. For more information about phrase searches, see Searching for Phrases (p. 101). Syntax: `(phrase field=FIELD boost=N 'STRING')`.
- `prefix`—Searches a text, text-array, literal, or literal-array field for the specified prefix followed by zero or more characters; for example, `(prefix field='title' 'wait')`. Supports boosting documents that match the expression. For more information about prefix searches, see Searching for Prefixes (p. 102).Syntax: `(prefix field=FIELD boost=N 'STRING')`.
- `range`—Searches for a range of values in a numeric field; for example: `(range field=year [2000,2013])`. For more information about range searches, see Searching for a Range of Values (p. 104). Syntax: `(range field=FIELD boost=N RANGE)`.

- `term`—Searches for an individual term or value in any field; for example: `(and (term field=title 'star')(term field=year 1977))`. Syntax: `(term field=FIELD boost=N 'STRING'|VALUE)`.

For more information about searching particular types of data, see the following sections. For more information about the structured search syntax, see Structured Search Syntax (p. 247).

Searching for Text in Amazon CloudSearch

You can search both text and literal fields for a text string:

- `Text` and `text-array` fields are always searchable. You can search for individual terms as well as phrases. Searches within `text` and `text-array` fields are not case-sensitive.
- `Literal` and `literal-array` fields can only be searched if they are search enabled in the domain's indexing options. You can search for an exact match of your search string. Searches in literal fields are case-sensitive.

If you use the simple query parser or do not specify a field when searching with the structured query parser, by default all `text` and `text-array` fields are searched. Literal fields are *not* searched by default. You can specify which fields you want to search with the `q.options` parameter.

You can search the unique document ID field like any text field. To reference the document ID field in a search request, you use the field name `_id`. Document IDs are always returned in the search results.

Topics

Searching for Individual Terms in Amazon CloudSearch

When you search `text` and `text-array` fields for individual terms, Amazon CloudSearch finds all documents that contain the search terms anywhere within the specified field, in any order. For example, in the sample movie data, the `title` field is configured as a `text` field. If you search the `title` field for *star*, you will find all of the movies that contain *star* anywhere in the `title` field, such as *star*, *star wars*, and *a star is born*. This differs from searching `literal` fields, where the field value must be identical to the search string to be considered a match.

The `simple` query parser provides an easy way to search `text` and `text-array` fields for one or more terms. The `simple` query parser is used by default unless you use the `q.parser` parameter to specify a different query parser.

For example, to search for *katniss*, specify `katniss` in the query string. By default, Amazon CloudSearch includes all return enabled fields in the search results. You can specify the `return` parameter to specify which fields you want to return.

```
https://search-domainname-domainid.us-east-1.cloudsearch.amazonaws.com/
2013-01-01/search?q=katniss&return=title
```

By default, the response is returned in JSON:

```
{
    "status": {
        "rid": "rd+5+r0oMAo6swY=",
        "time-ms": 9
    },
    "hits": {
        "found": 3,
        "start": 0,
        "hit": [
            {
                "id": "tt1951265",
                "fields": {
                    "title": "The Hunger Games: Mockingjay - Part 1"
                }
            },
            {
                "id": "tt1951264",
                "fields": {
                    "title": "The Hunger Games: Catching Fire"
                }
            },
            {
                "id": "tt1392170",
                "fields": {
                    "title": "The Hunger Games"
                }
            }
        ]
    }
}
```

To specify multiple terms, separate the terms with a space. For example: `star wars`. When you specify multiple search terms, by default documents must contain all of the terms to be considered a match. The terms can occur anywhere within the text field, in any order.

By default, all `text` and `text-array` fields are searched when you use the simple query parser. You can specify which fields you want to search by specifying the `q.options` parameter. For example, this query constrains the search to the `title` and `description` fields and boosts the importance of matches in the `title` field over matches in the `description` field.

```
q=star wars&q.options={fields: ['title^5','description']}
```

When you use the simple query parser, you can use the following prefixes to designate individual terms as required, optional, or to be excluded from the search results:

- **+**—matching documents must contain the term. This is the default—separating terms with a space is equivalent to preceding them with the + prefix.
- **-**—exclude documents that contain the term from the search results. The `-` operator only applies to individual terms. For example, to exclude documents that contain the term *star* in the default search field, specify: `-star`. Searching for `search?q=-star wars` retrieves all documents that do not contain the term *star*, but do contain the term *wars*.
- **|**—include documents that contain the term in the search results, even if they don't contain the other terms. The `|` operator only applies to individual terms. For example, to include documents that contain either of two terms, specify: `term1 |term2`. Searching for `search?q=star wars |trek` includes documents that contain both *star* and *wars*, or the term *trek*.

These prefixes only apply to individual terms in a simple query. To construct compound queries, you need to use the structured query parser, rather than the simple query parser. For example, to search for the terms *star* and *wars* using the structured query parser you would specify:

```
(and 'star' 'wars')
```

Note that this query matches documents that contain each of the terms in any of the fields being searched. The terms do not have to be in the same field to be considered a match. If, however, you specify `(and 'star wars' 'luke')`, *star* and *wars* must occur within the same field, and *luke* can occur in any of the fields.

If you don't specify any fields when you use the `structured` query parser, all `text` and `text-array` fields are searched by default, just like with the `simple` parser. Similarly, you can use the `q.options` parameter to control which fields are searched and to boost the importance of selected fields. For more information, see Constructing Compound Queries (p. 97).

You can also perform *fuzzy* searches with the simple query parser. To perform a fuzzy search, append the ~ operator and a value that indicates how much terms can differ from the user query string and still be considered a match. For example, the specifying `planit~1` searches for the term *planit* and allows matches to differ by up to one character, which means the results will include hits for *planet*.

Searching for Phrases in Amazon CloudSearch

When you search for a phrase, Amazon CloudSearch finds all documents that contain the complete phrase in the order specified. You can also perform *sloppy* phrase searches where the terms appear within the specified distance of one another.

To match a complete phrase rather than the individual terms in the phrase when you search with the simple query parser, enclose the phrase in double quotes. For example, the following query searches for the phrase *with love*.

```
q="with love"
```

To perform a sloppy phrase search with the simple query parser, append the ~ operator and a distance value. The distance value specifies the maximum number of words that can separate the words in the phrase. For example, the following query searches for the terms *with love* within three words of one another.

```
q="with love"~3
```

In a compound query, you use the `phrase` operator to specify the phrase you want to match; for example:

```
(phrase field=title 'star wars')
```

To perform a sloppy phrase search in a compound query, you use the `near` operator. The `near` operator enables you to specify the phrase you are looking for and how far apart the terms can be within a field and still be considered a match. For example, the following query matches documents that have the terms *star* and *wars* no more than three words apart in the `title` field.

```
(near field=title distance=3 'star wars')
```

For more information, see Constructing Compound Queries (p. 97).

Searching for Literal Strings in Amazon CloudSearch

When you search a literal field for a string, Amazon CloudSearch returns only those documents that contain an exact match for the complete search string in the specified field, including case. For example, if the `title` field is configured as a literal field and you search for *Star*, the value of the `title` field must be *Star* to be considered a match—*star*, *star wars* and *a star is born* will not be included in the search results. This differs from text fields, where searches are not case-sensitive and the specified search terms can appear anywhere within the field in any order.

To search a literal field, prefix the search string with the name of the literal field you want to search, followed by a colon. The search string must be enclosed in single quotes. For example, the following query searches for the literal string *Sci-Fi*.

```
genres:'Sci-Fi'
```

This example searches the genre field of each document and matches all documents whose genre field contains the value *Sci-Fi*. To be a match, the field value must be an exact match for the search string, including case. For example, documents that contain the value *Sci-Fi* in the genre field will not be included in the search results if you search for *sci-fi* or *young adult sci-fi*.

In a compound query, you use the `term` operator syntax to search literal fields. For example, `(term field=genres 'Sci-Fi')`. For more information, see Constructing Compound Queries (p. 97).

You can use literal fields in conjunction with faceting to enable users to drill down into the results according to the faceted attributes. For more information about faceting, see Getting and Using Facet Information in Amazon CloudSearch (p. 116).

Searching for Prefixes in Amazon CloudSearch

You can search `text`, `text-array`, `literal`, and `literal-array` fields for a *prefix* rather than for a complete term. This matches results that contain the prefix followed by zero or more characters. You must specify at least one character as the prefix. (To match all documents, use the `matchall` operator in a structured query.) In general, you should use a prefix that contains at least two characters to avoid matching an excessive number of documents.

When you search a `text` or `text-array` field, terms that match the prefix can occur anywhere within the contents of the field. When you search literal fields, the entire search string, up to and including the prefix characters, must match exactly.

- Simple query parser—use the `*` (asterisk) wildcard operator to search for a prefix, for example `pre*`.
- Structured query parser—use the `prefix` operator to search for a prefix, for example `prefix 'pre'`

For example, the following query searches for the prefix *oce* in the title field and returns the title of each hit:

```
q=oce*&q.options={fields:['title']}&return=title
```

If you perform this search against the sample movie data, it returns as *Ocean's Eleven* and *Ocean's Twelve*:

```
{
```

```
    "status": {
        "rid": "hIbIxb8oRAo6swY=",
        "time-ms": 2
    },
    "hits": {
        "found": 2,
        "start": 0,
        "hit": [
            {
                "id": "tt0240772",
                "fields": {
                    "title": "Ocean's Eleven"
                }
            },
            {
                "id": "tt0349903",
                "fields": {
                    "title": "Ocean's Twelve"
                }
            }
        ]
    }

}
```

In a compound query, you use the `prefix` operator to search for prefixes. For example, to search the `title` field for the prefix *oce*, you specify:

```
(prefix field=title 'oce')
```

For more information, see Constructing Compound Queries (p. 97).

Note
When performing wildcard searches on text fields, keep in mind that Amazon CloudSearch tokenizes the text fields during indexing and performs stemming according to the analysis scheme configured for the field. Normally, Amazon CloudSearch performs the same text processing on the search query. However, when you search for a prefix with the wildcard operator (*) or `prefix` operator, no stemming is performed on the prefix. This means that a search for a prefix that ends in `s` won't match the singular version of the term. This can happen for any term that ends in `s`, not just plurals. For example, if you search the `actor` field in the sample movie data for `Anders`, there are three matching movies. If you search for `Ander*`, you get those movies as well as several others. However, if you search for `Anders*` there are no matches. This is because the term is stored in the index as `ander`, `anders` does not appear in the index. For more information about how Amazon CloudSearch processes text and how it can affect searches, see Text Processing in Amazon CloudSearch (p. 79).

Searching for Numbers in Amazon CloudSearch

You can use structured queries to search any search enabled numeric field for a particular value or range of values (p. 104). Amazon CloudSearch supports four numeric field types: `double`, `double-array`, `int`, and `int-array`. For more information, see Configuring Index Fields (p. 64).

The basic syntax for searching a field for a single value is **`FIELD`**:**`VALUE`**. For example, `year:2010` searches the sample movie data for movies released in 2010.

You must use the structured query parser to use the field syntax. Note that numeric values are *not* enclosed in quotes—quotes designate a value as a string. To search for a range of values, use a

comma (,) to separate the upper and lower bounds, and enclose the range using brackets or braces. For more information, see Searching for a Range of Values (p. 104).

In a compound query, you use the `term` operator syntax to search for a single value: `(term field=year 2010)`.

Searching for Dates and Times in Amazon CloudSearch

You can use structured queries to search any search enabled date field for a particular date and time or a date-time range (p. 104). Amazon CloudSearch supports two date field types, `date` and `date-array`. For more information, see Configuring Index Fields (p. 64).

Dates and times are specified in UTC (Coordinated Universal Time) according to IETF RFC3339: `yyyy-mm-ddTHH:mm:ss.SSSZ`. In UTC, for example, 5:00 PM August 23, 1970 is: `1970-08-23T17:00:00Z`. Note that you can also specify fractional seconds when specifying times in UTC. For example, `1967-01-31T23:20:50.650Z`.

To search for a date (or time) in a `date` field, you must enclose the date string in single quotes. For example, both of the following queries search the movie data for all movies released on December 25, 2001:

```
release_date: '2001-12-25T00:00:00Z'
(term field=release_date '2001-12-25T00:00:00Z')
```

Searching for a Range of Values in Amazon CloudSearch

You can use structured queries to search a field for a range of values. To specify a range of values, use a comma (,) to separate the upper and lower bounds and enclose the range using brackets or braces. A square brace, [or], indicates that the bound is included in the range, a curly brace, { or }, excludes the bound.

For example, to search the sample data set for movies released from 2008 to 2010 (inclusive), specify the range as `[2008,2010]`.

To specify an open-ended range, omit the bound. For example, `year:[2002,}` matches all movies released from 2002 onward, and `year:{,1970]` matches all movies released through 1970. When you omit a bound, you must use a curly brace.

In a compound query, you use the `range` operator syntax to search for a range of values; for example: `(range field=year [1967,})`.

Searching for a Date Range

To search for a range of dates (or times) in a `date` field, you use the same bracketed range syntax that you use for numeric values, but you must enclose the date string in single quotes. For example, the following request searches the movie data for all movies with a release date of January 1, 2013 or later:

```
release_date:['2013-01-01T00:00:00Z',}
```

Searching for a Location Range

You can perform a bounding box search by searching for a range of locations. To search for a range of locations in a `latlon` field, you use the same bracketed range syntax that you use for numeric values, but you must enclose the latitude/longitude pair in single quotes.

For example, if you include a `location` field in each document, you could specify your bounding box filter as `location:['nn.n,nn.n','nn.n,nn.n']`. In the following example, the matches for *restaurant* are filtered so that only matches within the downtown area of Paso Robles, CA are included in the results.

```
q='restaurant'&fq=location:
['35.628611,-120.694152','35.621966,-120.686706']&q.parser=structured
```

For more information, see Searching and Ranking Results by Geographic Location in Amazon CloudSearch (p. 105).

Searching for a Text Range

You can also search a text or literal field for a range of values using the bracketed range syntax. Like dates, the text strings must be enclosed in single quotes. For example, the following request searches the movie data for a range of document IDs. To reference a document's ID, you use the special field name `_id`.

```
_id:['tt1000000','tt1005000']
```

Searching and Ranking Results by Geographic Location in Amazon CloudSearch

If you store locations in your document data using a `latlon` field, you can use the `haversin` function in an Amazon CloudSearch expression to compute the distance between two locations. Storing locations with your document data also enables you to easily search within particular areas.

Topics

Searching Within an Area in Amazon CloudSearch

To associate a location with a search document, you can store the location's latitude and longitude in a `latlon` field using decimal degree notation. The values are specified as a comma-separated list, `lat,lon`—for example `35.628611,-120.694152`. Associating a location with a document enables you to easily constrain search hits to a particular area with the `fq` parameter.

To use a bounding box to constrain results to a particular area

1. Determine the latitude and longitude of the upper-left and lower-right corners of the area you are interested in.
2. Use the `fq` parameter to filter the matching documents using those bounding box coordinates. For example, if you include a `location` field in each document, you could specify your bounding box

filter as `fq=location:['nn.n,nn.n','nn.n,nn.n']` . In the following example, the matches for *restaurant* are filtered so that only matches within the downtown area of Paso Robles, CA are included in the results.

```
q='restaurant'&fq=location:
['35.628611,-120.694152','35.621966,-120.686706']&q.parser=structured
```

Sorting Results by Distance in Amazon CloudSearch

You can define an expression as part of your search request to sort results by distance. Amazon CloudSearch expressions support the `haversin` function, which computes the great-circle distance between two points on a sphere using the latitude and longitude of each point. (For more information, see Haversine formula.) The resulting distance is returned in kilometers.

To calculate the distance between each matching document and the user, you pass the user's location into the `haversin` function and reference the document locations stored in a `latlon` field. You specify the user latitude and longitude in decimal degree notation and access the latitude and longitude stored in a `latlon` as `FIELD.latitude` and `FIELD.longitude`. For example, `expr.distance=haversin(`**`userlat,userlon`**`, location.latitude,location.longitude)`.

To use the expression to sort the search results, you specify the `sort` parameter.

For example, the following query searches for restaurants and sorts the results by distance from the user.

```
q=restaurant&expr.distance=haversin(35.621966,-120.686706,location.latitude,location.longit
 asc
```

Note that you must explicitly specify the sort direction, `asc` or `desc`.

You can include the distance calculated for each document in the search results by specifying the name of the expression with the `return` parameter. For example, `return=distance`.

You can also use the distance value in more complex expressions to take other characteristics into account, such as a document's relevance `_score`. In the following example, a second rank expression uses both the document's calculated `distance` and its relevance `_score`.

```
expr.distance=haversin(38.958687,-77.343149,latitude,longitude)&expr.myrank=_score/
log(distance)&sort=-myrank
```

For more information about using expressions to sort search results, see Controlling Search Results (p. 128).

Searching DynamoDB Data with Amazon CloudSearch

You can specify a DynamoDB table as a source when configuring indexing options or uploading data to a search domain through the console or command line tools. This enables you to quickly set up a search domain to experiment with searching data stored in DynamoDB database tables.

To keep your search domain in sync with changes to the table, you can send updates to both your table and your search domain, or you can periodically load the entire table into a new search domain.

Topics

Configuring an Amazon CloudSearch Domain to Search DynamoDB Data

The easiest way to configure a search domain to search DynamoDB data is to use the Amazon CloudSearch console. The console's configuration wizard analyzes your table data and suggests indexing options based on the attributes in the table. You can modify the suggested configuration to control which table attributes are indexed.

You can also use the command line tools to generate document batches from your table and automatically configure your domain, or you can configure indexing options manually. For general information about configuring indexing options, see Configuring Index Fields (p. 64).

> **Note**
> To upload data from DynamoDB, you must have permission to access both the service and the resources you want to upload. For more information, see Using IAM to Control Access to DynamoDB Resources.

When you automatically configure a search domain from a DynamoDB table, a maximum of 200 unique attributes can be mapped to index fields. (You cannot configure more than 200 fields for a search domain, so you can only upload data from DynamoDB tables with 200 or fewer attributes.) When Amazon CloudSearch detects an attribute that has a small number of distinct values, the field is facet enabled in the suggested configuration.

> **Important**
> When you use a DynamoDB table to configure a domain, the data is not automatically uploaded to the domain for indexing. You must upload the data for indexing as a separate step after you configure the domain.

Configuring a Domain to Search DynamoDB using the Amazon CloudSearch Console

You can use the Amazon CloudSearch console to analyze data from a DynamoDB table to configure a search domain. A maximum of 5 MB is read from the table regardless of the table size. By default, Amazon CloudSearch reads from the beginning of the table. You can specify a start key to begin reading from a particular item.

To configure a search domain using a DynamoDB table

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain, and then click the domain's **Indexing Options** link.
3. At the top of the **Indexing Options** pane, click the **configuration wizard** link.
4. In the **Choose Source** step, select **Analyze sample item(s) from DynamoDB**.
5. From the **DynamoDB Table** list, select the DynamoDB table that you want to analyze.

- To limit the read capacity units that can be consumed while reading from the table, enter the maximum percentage of read capacity units you want to use.
- To start reading from a particular item, specify a **Start Hash Key**. If the table uses a hash and range type primary key, specify both the hash attribute and the range attribute for the item.

6. When you finish specifying the table options, click **Continue**.
7. In the **Review Configuration** step, review the suggested configuration. You can edit these fields and add additional fields.
8. When you finish, click **Apply Configuration**.
9. In the **Apply Configuration** step, you can choose to run indexing when you exit the configuration wizard. If you haven't uploaded data to your domain yet, clear the **Run Indexing Now** checkbox to exit without indexing. If you are done making configuration changes and are ready to index your data with the new configuration, make sure **Run Indexing Now** is selected. When you are ready to apply the changes, click **Finish**.

You can also use a DynamoDB table to configure indexing options when you first create a domain. In the **Configure Index** step, select **Analyze sample item(s) from DynamoDB** and select the table to analyze.

Configuring a Domain to Search a DynamoDB Table Using the Amazon CloudSearch Command Line Tools

You can use the cs-import-documents (p. 144) and cs-configure-from-batches (p. 142) commands to configure a domain based on the data in a DynamoDB table.

To configure a search domain using a DynamoDB table

1. Run the `cs-import-documents` command and specify the `--source` and `--output` options. The source is the name of a DynamoDB table. The output is the local directory or Amazon S3 bucket where you want to save the generated document batches.

```
cs-import-documents --source ddb://myDDBTable --output c:\myddbdata
```

Note
You can make changes to the generated document data before you use it to configure your domain. For more information about mapping your data to index fields, see Preparing Your Data (p. 58). For information about customizing your domain configuration, see Configuring Index Fields (p. 64).

2. Run the `cs-configure-from-batches` command and specify the `--domain` and `--source` options. The domain is the name of the search domain you are configuring. The source specifies the document batch (or batches) to use to configure the domain.

```
cs-configure-from-batches --domain ddb-cs-search --source c:\myddbdata\*
```

Uploading Data to Amazon CloudSearch from DynamoDB

You can upload DynamoDB data to a search domain through the Amazon CloudSearch console or with the Amazon CloudSearch command line tools. When you upload data from a DynamoDB table, Amazon CloudSearch converts it to document batches so it can be indexed. You select define index

fields for each of the attributes in your domain configuration. For more information, see Configuring an Amazon CloudSearch Domain to Search DynamoDB Data (p. 107).

You can upload data from more than one DynamoDB table to the same Amazon CloudSearch domain. However, keep in mind that you can upload a maximum of 200 attributes from all tables combined. If an item with the same key appears in more than one uploaded table, the last-applied item overwrites all previous versions.

When converting table data to document batches, Amazon CloudSearch generates a document for each item it reads from the table, and represents each item attribute as a document field. The unique ID for each document is either read from the `docid` item attribute (if it exists) or assigned an alphanumeric value based on the primary key.

When Amazon CloudSearch generates documents for table items:

- Sets of strings and sets of numbers are represented as multi-value fields. If a DynamoDB set contains more than 100 values, only the first 100 values are added to the multi-value field.
- DynamoDB binary attributes are ignored.
- Attribute names are modified to conform to the Amazon CloudSearch naming conventions for field names:
  - All uppercase letters are converted to lowercase.
  - If the DynamoDB attribute name does not begin with a letter, the field name is prefixed with `f_`.
  - Any characters other than a-z, 0-9, and _ (underscore) are replaced by an underscore. If this transformation results in a duplicate field name, a number is appended to make the field name unique. For example, the attribute names `håt`, `h-t`, `hát` would be mapped to `h_t`, `h_t1`, and `h_t2` respectively.
  - If the DynamoDB attribute name exceeds 64 characters, the first 56 characters of the attribute name are concatenated with the 8-character MD5 hash of the full attribute name to form the field name.
  - If the attribute name is `body`, it is mapped to the field name `f_body`.
  - If the attribute name is `_score` it is mapped to the field name `f__score`.
- Number attributes are mapped to Amazon CloudSearch int fields and the values are transformed to 32-bit unsigned integers:
  - If a number attribute contains a decimal value, only the integral part of the value is stored. Everything to the right of the decimal point is dropped.
  - If the value is larger than can be stored as an unsigned integer, the value is truncated.
  - Negative integers are treated as unsigned positive integers.

Uploading DynamoDB Data to a Domain through the Amazon CloudSearch Console

You can use the Amazon CloudSearch console to upload up to 5 MB of data from a DynamoDB table to a search domain. To upload a larger amount of data from a DynamoDB table, use the command line tools (p. 110).

To upload DynamoDB data using the console

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain.
3. At the top of the domain dashboard, click **Upload Documents**.
4. In the **Document Source** step, select **Item(s) from DynamoDB**.
5. In the **DynamoDB Table** list, select the DynamoDB table that contains your data.

- To limit the read capacity units that can be consumed while reading from the table, enter the maximum percentage of read capacity units.
- To start reading from a particular item, specify a **Start Hash Key**. If the table uses a hash and range type primary key, specify both the hash attribute and the range attribute for the item.

6. When you finish specifying the table options, click **Continue**.
7. In the **Review Documents** step, review the items that will be uploaded. (You can also save the generated document batch by clicking **Download the generated document batch**.) When you finish, click **Upload Documents**.
8. In the **Document Summary** step, click **Finish** to exit the upload documents wizard.

Uploading DynamoDB Data to a Domain Using the Amazon CloudSearch Command Line Tools

You can use the cs-import-documents (p. 144) command to process and upload items in a DynamoDB table.

To upload DynamoDB data using the command line tools

- Run the `cs-import-documents` command and specify the `--source` and `--domain` options. The source is the name of the DynamoDB table that contains your data. The domain is the name of the search domain you want to use to search the data.

```
cs-import-documents --domain ddb-cs-search --source ddb://myDDBTable
```

Note
You can save the generated document batches to your local file system or an Amazon S3 bucket by specifying the `--output` option instead of the `--domain` option. This enables you to review and modify the document data before uploading it with the cs-import-documents (p. 144) command.

Synchronizing a Search Domain with a DynamoDB Table

To keep your search domain in sync with updates to your DynamoDB table, you can either programmatically track and apply updates to your domain, or periodically create a new domain and upload the entire table again. If you have a large amount of data, it's best to track and apply updates programmatically.

Programmatically Synchronizing Updates

To synchronize changes and additions to your DynamoDB table, you can create a separate update table to track the changes to the table you are searching and periodically upload the contents of the update table to the corresponding search domain.

To remove documents from the search domain, you must generate and upload document batches that contain a delete operation for each deleted document. One option is to use a separate DynamoDB table to track deleted items, periodically process the table to generate a batch of delete operations, and upload the batch to your search domain.

To make sure that you don't lose any changes that are made during the initial data upload, you must begin collecting tracking changes before the initial data upload. While you might update some Amazon

CloudSearch documents with identical data, you ensure that no changes are lost and your search domain contains an up-to-date version of every document.

How often you synchronize updates depends on the volume of changes and your update latency tolerance. One approach is to accumulate changes over a fixed time period and at the end of the time period upload the changes and delete the period's tracking tables.

For example, to synchronize changes and additions once a day, at the beginning of each day you could create a table called updates_YYYY_MM_DD to collect the daily updates. At the end of the day, you upload the updates_YYYY_MM_DD table to your search domain. (If the update table is larger than 5 MB, you must use the command line tools.) After the upload is complete, you can delete the update table and create a new one for the next day.

Switching to a New Search Domain

If you don't want to track and apply individual updates to your table, you can periodically load the entire table into a new search domain and then switch your query traffic over to the new domain.

To switch to a new search domain

1. Create a new search domain and copy the configuration from your existing domain.
2. Upload the entire DynamoDB table to the new domain. (If the table is larger than 5 MB, you must use the command line tools to upload it.) For more information, see Uploading Data to Amazon CloudSearch from DynamoDB (p. 108).
3. After the new domain is active, update the DNS entry that directs query traffic to the old search domain to point to the new domain. For example, if you use Amazon Route 53, you can simply update the recordset with your new search service endpoint.
4. Delete the old domain.

Filtering Matching Documents in Amazon CloudSearch

You use the `fq` parameter to filter the documents that match the search criteria specified with the `q` parameter without affecting the relevance scores of the documents included in the search results. Specifying a filter just controls which matching documents are included in the results, it has no effect on how they are scored and sorted.

The `fq` parameter supports the structured query syntax described in Search API Reference (p. 239).

For example, you could add an `available` field to your documents to indicate whether or not an item is in stock, and filter on that field to limit the results to in-stock items:

```
search?q=star+wars&fq=available:'true'&return=title
```

Tuning Search Request Performance in Amazon CloudSearch

Search requests can become very resource intensive to process, which can have an impact on the performance and cost of running your search domain. In general, searches that return a large volume of hits and complex structured queries are more resource intensive than simple text queries that match a small percentage of the documents in your search domain.

If you experience slow response times, frequently encounter internal server errors (typically 507 or 509 errors), or see the number of instance hours your search domain consumes increase without a substantial increase in the volume of data you're searching, you can fine-tune your search requests to help reduce the processing overhead. This section reviews what to look for and steps you can take to tune your search requests.

Analyzing Query Latency

Before you can tune your requests, you must analyze your current search performance. Log your search requests and response times so that you can see which requests take the longest to process. Slow searches can disproportionally affect overall performance by tying up your search domain's resources. Optimizing the slowest search requests speeds up *all* of your searches.

Topics

Reducing the Number of Hits

Query latency is directly proportional to the number of matching documents. Searches that match the most documents are generally the slowest.

Eliminating two types of searches that commonly result in a huge number of matching documents can significantly improve overall performance:

- Queries that match every document in your corpus (`matchall`). While this can be a convenient way to list all the documents in your domain, it's a resource intensive query. If you have a lot of documents, not only can it cause other requests to time out, it's likely to time out itself.
- Prefix (wildcard) searches with only one or two characters specified. If you're using this type of search to provide instant results as the user types, wait until the user has entered at least two characters before you start submitting requests and displaying the possible matches.

To reduce the number of documents that match your requests, you can also do the following:

- Eliminate irrelevant words from your corpus so they aren't using for matching. The easiest way to do this is to add them to the stopwords list dictionary for the analysis scheme(s) you're using. Alternatively, you can preprocess your data to strip out irrelevant words. Eliminating irrelevant words also has the benefit of reducing the size of your index, which can help reduce costs.
- Explicitly filter the results based on the value of a particular field using the `fq` parameter.

If you still have requests that match a lot of documents, you can reduce latency by minimizing the amount of processing to be done on the result set:

- Minimize the facet information that you request. Generating the facet counts adds to the time it takes to process the request and increases the likelihood that other requests will time out. If you do request facet information, keep in mind that the more facets you specify, the longer it takes to process the request.
- Avoid using your own expressions for sorting. The additional processing required to sort the results increases the likelihood that requests will time out. If you must customize how the results are sorted, it is generally faster to use a field than to use an expression.

Keep in mind that returning a large amount of data in the search results can increase the transport time and affect query latency. Minimize the number of return fields you use to improve performance and reduce the size of your index.

Simplifying Structured Queries

The more clauses there are in the query criteria, the longer it takes to process the query.

If you have complex structured queries that don't perform well, you need to find a way to reduce the number of clauses. In some cases, you might simply be able to set a limit or reformulate the query. In others, you might need to modify your domain configuration to accommodate simpler queries.

Querying Your Search Domain for More Information in Amazon CloudSearch

When you submit a search request, Amazon CloudSearch returns a collection of the documents that match your search criteria. You can also retrieve:

- The contents of selected fields
- Facet information that enables you to categorize the results
- Statistics for the values contained in numeric fields
- Highlights that show the search hits in the field data
- Autocomplete suggestions

Topics

Retrieving Data from Index Fields in Amazon CloudSearch

By default, the search results include all return enabled fields. To return a subset of the return enabled fields or return expression values for the matching documents, you can specify the `return` parameter. To return only the document IDs for the matching documents, specify `return=_no_fields`. To retrieve the relevance score calculated for each document, specify `return=_score`. You specify multiple return fields as a comma separated list. For example, `return=title,_score` returns just the title and relevance score of each matching document.

Only fields configured to be return enabled can be included in the search results. Making fields return enabled increases the size of your index, which can increase the cost of running your domain. You should only store document data in the search index by making fields return enabled when it's difficult

or costly to retrieve the data using other means. Because it can take some time to apply document updates across the domain, you should retrieve critical data such as pricing information by using the returned document IDs instead of returned from the index.

For example, to include the *title* and relevance *_score* in the search results, specify the following:

```
search?q=star -wars&return=title,_score&size=3
```

The specified fields are included with each hit in the search results:

```
{
  "status" : {
    "rid" : "y9Dzhs8oEwqMHnk=",
    "time-ms" : 2
  },
  "hits" : {
    "found" : 76,
    "start" : 0,
    "hit" : [ {
      "id" : "tt1411664",
      "fields" : {
        "title" : "Bucky Larson: Born to Be a Star",
        "_score" : "9.231539"
      }
    }, {
      "id" : "tt1911658",
      "fields" : {
        "title" : "The Penguins of Madagascar",
        "_score" : "7.1051397"
      }
    }, {
      "id" : "tt0120601",
      "fields" : {
        "title" : "Being John Malkovich",
        "_score" : "6.206055"
      }
    } ]
  }
}
```

Getting Statistics for Numeric Fields in Amazon CloudSearch

Amazon CloudSearch can return the following statistics for facet-enabled numeric fields:

- `count`–The number of documents that contain a value in the specified field.
- `max`–The maximum value found in the specified field.
- `mean`–The average of the values found in the specified field.
- `min`–The minimum value found in the specified field.
- `missing`–The number of documents that do not contain a value in the specified field.
- `stddev`–A measure to quantify the amount of deviation, or variation, in the field values. A low standard deviation indicates that the values across all documents are close to the mean. A high standard deviation indicates that the values are spread out over a large range. The standard

deviation is calculated by taking the square root of the variance, which is the average of the squared differences from the mean.

- `sum`–The sum of the field values across all documents.
- `sumOfSquares`–The sum of all field values squared.

To get statistics for a field you use the `stats.FIELD` parameter. `FIELD` is the name of a facet-enabled numeric field. You specify an empty JSON object, `stats.FIELD={}`, to get all of the available statistics for the specified field. (The `stats.FIELD` parameter does not support any options; you must pass an empty JSON object.) You can request statistics for multiple fields in the same request.

You can get statistics only for facet-enabled numeric fields: `date`, `date-array`, `double`, `double-array`, `int`, or `int-array`. Note that only the `count`, `max`, `min`, and `missing` statistics are returned for `date` and `date-array` fields. For more information about enabling a field to return facets, see Configuring Index Fields (p. 64).

For example, to search for *star* and get statistics for the *year* field, specify the following:

```
search?q=star&stats.year={}
```

Getting and Using Facet Information in Amazon CloudSearch

Topics

A *facet* is an index field that represents a category that you want to use to refine and filter search results. When you submit search requests to Amazon CloudSearch, you can request facet information to find out how many documents share the same value in a particular field. You can display this information along with the search results, and use it to enable users to interactively refine their searches. (This is often referred to as faceted navigation or faceted search.)

You can get facet information for any facet-enabled field by specifying the `facet.FIELD` parameter in your search request. By default, Amazon CloudSearch returns facet counts for the top 10 values. For more information about enabling a field to return facets, see Configuring Index Fields (p. 64). For a description of the `facet.FIELD` parameter, see Search Request Parameters (p. 241) in the Search API reference.

You can specify facet options to control the sorting of the facet values for each field, limit the number of facet values returned, or choose what facet values to count and return.

Getting Facet Information in Amazon CloudSearch

To get facet information for a field, you use the `facet.FIELD` parameter. `FIELD` is the name of a facet-enabled field. You specify facet options as a JSON object. If the JSON object is empty (`facet.FIELD={}`), facet counts are computed for all field values, the facets are sorted by facet count, and the top 10 facets are returned in the results. You can request facet information for multiple fields in the same request.

You can retrieve facet information in two ways:

- `sort`—Returns facet information sorted either by facet counts or facet values.
- `buckets`—Returns facet information for particular facet values or ranges.

Sorting Facet Information

You specify the `sort` option to control how the facet information is sorted. There are two sort options: `count` and `bucket`:

- Use `count` to sort the facets by facet counts. For example, `facet.year={sort:'count'}` counts the number of matches that have the same year value and sorts the facet information by that number.
- Use `bucket` to sort the facets by the facet values. For example, `facet.year={sort:'bucket'}` .

When you use the `sort` option, you can specify the `size` option to control the maximum number of facet values returned in the results. The `size` option is valid only when you use the `sort` option.

In the following example, facet information is calculated for the `genres` field, the genres are sorted by facet value, and the first 5 genres are returned in the results:

```
facet.genres={sort:'bucket', size:5}
```

Bucketing Facet Information

You can explicitly specify the facet values or ranges that you want to count by using the `buckets` option. Buckets are specified as an array of values or ranges, for example, `facet.color={buckets:["red","green","blue"]}`.

To specify a range of values, use a comma (,) to separate the upper and lower bounds and enclose the range using brackets or braces. A square bracket, [or], indicates that the bound is included in the range, a curly brace, { or }, excludes the bound. You can omit the upper or lower bound to specify an open-ended range. When omitting a bound, you must use a curly brace. For example, `facet.year={buckets:["[1970,1979]","[1980,1989]","[1990,1999]","[2000,2009]","[2010,}"]}`.

The `sort` and `size` options are not valid if you specify buckets.

Amazon CloudSearch supports two methods for calculating bucket counts, `filter` and `interval`. By default, the `filter` method is used, which simply submits an additional filter query for each bucket to get the bucket counts. While this works well in many cases, if you have a high update rate or are retrieving a large number of facets, performance can suffer because those queries can't take advantage of the internal caching mechanism.

If you're experiencing slow query performance for bucketed facets, try setting the buckets method to `interval`, which post-processes the result set rather than submitting multiple queries:

```
facet.year={buckets:
["[1970,1979]","[1980,1989]","[1990,1999]"],method:"interval"}
```

We recommend doing your own performance testing to determine which method is best for your application. In general, the `filter` method is faster if you have a fairly low update rate and aren't retrieving a large number of buckets. However, if you have a high update rate or a lot of buckets, using the `interval` method to post-process the result set can result in significantly faster query performance.

Using Facet Information in Amazon CloudSearch

You can display facet information to enable users to more easily browse search results and identify the information they are interested in. For example, if a user is trying to find one of the Star Trek movies, but can't remember the full title, he might start by searching for *star*. If you want to display top facets for

genre, you would include `facet.FIELD` in the query, along with the number of facet values that you want to retrieve for each facet:

```
search?q=star&facet.genres={sort:'count',size:5}&format=xml&return=_no_fields
```

The preceding example gives you the following information in the search response:

```
<results>
    <status rid="v7r9hs8oFQqMHnk=" time-ms="3"/>
    <hits found="85" start="0">
        <hit id="tt1411664"/>
        <hit id="tt1911658"/>
        <hit id="tt0086190"/>
        <hit id="tt0120601"/>
        <hit id="tt2141761"/>
        <hit id="tt1674771"/>
        <hit id="tt0056687"/>
        <hit id="tt0397892"/>
        <hit id="tt0258153"/>
        <hit id="tt0796366"/>
    </hits>
    <facets>
        <facet name="genres">
            <bucket value="Comedy" count="41"/><bucket value="Drama"
 count="35"/>
            <bucket value="Adventure" count="29"/>
            <bucket value="Sci-Fi" count="24"/>
            <bucket value="Action" count="20"/>
        </facet>
    </facets>
</results>
```

Multi-Select Facets in Amazon CloudSearch

If you want to display the available facets and enable users to select multiple values to refine the results, you can submit one request to get the documents that match the facet constraints and additional requests to get the facet counts.

For example, in the sample movie data, the `genres`, `rating`, and `year` fields are facet enabled. If the user searches for the term *poet*, you can submit the following request to get the matching movies and the facet counts for the `genres`, `rating`, and `year` fields:

```
q=poet&facet.genres={}&facet.rating={}&facet.year={}&return=_no_fields
```

Because no `facet.FIELD` options are specified, Amazon CloudSearch counts all of the facet values and returns the top 10 values for each facet:

```
{
  "status" : {
    "rid" : "it3T8tIoDgrUSvA=",
    "time-ms" : 5
  },
  "hits" : {
    "found" : 14,
    "start" : 0,
    "hit" : [
```

```
       {"id" : "tt0097165"},
       {"id" : "tt0059113"},
       { "id" : "tt0108174"},
       {"id" : "tt1067765"},
       { "id" : "tt1311071"},
       {"id" : "tt0810784"},
       {"id" : "tt0819714"},
       {"id" : "tt0203009"},
       {"id" : "tt0114702"},
       {"id" : "tt0107840"} ]
  },
  "facets" : {
    "genres" : {
      "buckets" : [
         {"value" : "Drama","count" : 12},
         {"value" : "Romance","count" : 9},
         {"value" : "Biography", "count" : 4},
         {"value" : "Comedy","count" : 2},
         {"value" : "Thriller","count" : 2},
         {"value" : "War","count" : 2},
         {"value" : "Crime","count" : 1},
         {"value" : "History","count" : 1},
         {"value" : "Musical","count" : 1} ]
    },
    "rating" : {
      "buckets" : [
         {"value" : "6.3","count" : 3},
         {"value" : "6.2","count" : 2},
         {"value" : "7.1","count" : 2},
         {"value" : "7.9","count" : 2},
         {"value" : "5.3","count" : 1},
         {"value" : "6.1""count" : 1},
         {"value" : "6.4","count" : 1},
         {"value" : "6.9","count" : 1},
         {"value" : "7.6","count" : 1} ]
    },
    "year" : {
      "buckets" : [
         {"value" : "2013","count" : 3},
         {"value" : "1993","count" : 2},
         {"value" : "1965","count" : 1},
         {"value" : "1989","count" : 1},
         {"value" : "1995","count" : 1},
         {"value" : "2001","count" : 1},
         {"value" : "2004","count" : 1},
         {"value" : "2006","count" : 1},
         {"value" : "2008","count" : 1},
         {"value" : "2009","count" : 1} ]
    }
  }
}
```

When the user refines the search by selecting facet values, you use those facet selections to filter the results. For example, if the user selects *2013*, *2012*, and *1993*, the following request gets the matching movies released during those years:

```
q=poet&fq=(or year:2013 year:2012 year:1993)&facet.genres={}&facet.rating={}
&facet.year={}&return=_no_fields
```

This gets the documents that match the user's selection and the facet counts with the filter applied:

```
{
  "status" : {
    "rid" : "zMP38tIoDwrUSvA=",
    "time-ms" : 6
  },
  "hits" : {
    "found" : 6,
    "start" : 0,
    "hit" : [
       {"id" : "tt0108174"},
       {"id" : "tt1067765"},
       {"id" : "tt1311071"},
       {"id" : "tt0107840"},
       {"id" : "tt1462411"},
       {"id" : "tt0455323"} ]
  },
  "facets" : {
    "genres" : {
      "buckets" : [
         {"value" : "Drama","count" : 4},
         {"value" : "Romance","count" : 3},
         {"value" : "Comedy","count" : 2},
         {"value" : "Thriller","count" : 2},
         {"value" : "Biography","count" : 1},
         {"value" : "Crime","count" : 1} ]
    },
    "rating" : {
      "buckets" : [
         {"value" : "6.3","count" : 2},
         {"value" : "5.3","count" : 1},
         {"value" : "6.2","count" : 1},
         {"value" : "6.4","count" : 1},
         {"value" : "7.1","count" : 1} ]
    },
    "year" : {
      "buckets" : [
         {"value" : "2013","count" : 3},
         {"value" : "1993","count" : 2},
         {"value" : "2012","count" : 1} ]
    }
  }
}
```

This is what you want to show for the genres and ratings. However, to enable the user to change the year filter, you need to get the facet counts for the years that *aren't* selected. To do this, you submit a second request to retrieve the facet counts for the year field without the filter:

```
q=poet&facet.year={}&size=0
```

There's no need to retrieve the matching documents, so the `size` parameter is set to zero to minimize the request latency. The request returns just the facet information for the `year` field:

```
{
  "status" : {
    "rid" : "x/7r0NIoRwqlHfo=",
```

```
    "time-ms" : 4
  },
  "hits" : {
    "found" : 14,
    "start" : 0,
    "hit" : [ ]
  },
  "facets" : {
    "year" : {
      "buckets" : [
         {"value" : "2013","count" : 3},
         {"value" : "1993","count" : 2},
         {"value" : "1965","count" : 1},
         {"value" : "1989","count" : 1},
         {"value" : "1995","count" : 1},
         {"value" : "2001","count" : 1},
         {"value" : "2004","count" : 1},
         {"value" : "2006","count" : 1},
         {"value" : "2008","count" : 1},
         {"value" : "2009","count" : 1} ]
    }
  }
}
```

To minimize the response time, you can send this request in parallel with the request to get the filtered results. However, keep in mind that these additional requests can impact your overall query performance, and it might be necessary to scale your domain up to handle the additional traffic. (For more information about scaling, see Configuring Scaling Options (p. 39).)

If the user further refines the search by selecting a genre or rating, you add that to the filter criteria to get the matching documents. For example, the following request gets the movies released in 2013, 2012, or 1993 that have a rating of 6.3:

```
q=poet&fq=(and rating:6.3 (or year:2013 year:2012
 year:1993))&facet.genres={}&return=_no_fields
```

Getting the facet information for genres in this request returns the facet counts with the rating and year filters applied:

```
{
  "status" : {
    "rid" : "l66b89IoEArUSvA=",
    "time-ms" : 6
  },
  "hits" : {
    "found" : 2,
    "start" : 0,
    "hit" : [
       {"id" : "tt1462411"},
       {"id" : "tt0455323"} ]
  },
  "facets" : {
    "genres" : {
      "buckets" : [
         {"value" : "Drama","count" : 2} ]
    }
  }
```

```
}
```

To enable the user to select a different rating, you submit an additional request to get the rating facet counts with only the year filter applied:

```
q=poet&fq=(or year:2013 year:2012 year:1993)&facet.rating={}&size=0
```

This request gets the following response:

```
{
  "status" : {
    "rid" : "jqWj89IoEQrUSvA=",
    "time-ms" : 5
  },
  "hits" : {
    "found" : 6,
    "start" : 0,
    "hit" : [ ]
  },
  "facets" : {
    "rating" : {
      "buckets" : [
         {"value" : "6.3","count" : 2},
         {"value" : "5.3","count" : 1},
         {"value" : "6.2","count" : 1},
         {"value" : "6.4","count" : 1},
         {"value" : "7.1","count" : 1} ]
    }
  }
}
```

Similarly, you need another request to get the year facet counts with only the rating filter applied:

```
q=poet&fq=rating:6.3&facet.year={}&size=0
```

This request gets the following response:

```
{
  "status" : {
    "rid" : "4L6F8NIoDQrUSvA=",
    "time-ms" : 4
  },
  "hits" : {
    "found" : 3,
    "start" : 0,
    "hit" : [ ]
  },
  "facets" : {
    "year" : {
      "buckets" : [
         {"value" : "1995","count" : 1},
         {"value" : "2012","count" : 1},
         {"value" : "2013","count" : 1} ]
    }
  }
}
```

Highlighting Search Hits in Amazon CloudSearch

Amazon CloudSearch can return excerpts with the search results to show where the search terms occur within a particular field of a matching document. For example, in the following excerpt the search terms *luke skywalker* are highlighted within the `plot` field:

```
highlights": {
    "plot": "After the rebels have been brutally overpowered by the Empire on
    their newly established base, *Luke* *Skywalker* takes advanced Jedi
    training with Master Yoda, while his friends are pursued by Darth Vader
    as part of his plan to capture *Luke*."
}
```

If you search for a phrase, the matching documents must contain that phrase. However, when you retrieve highlights, the terms in the phrase are highlighted individually. If you search for the phrase `"Luke Skywalker"` and retrieve highlights for the `plot` field as shown in the previous example, the term `Luke` is highlighted even when it isn't followed by `Skywalker`. Highlights are returned for the first 10 KB of data in a field. If the field contains more than 10 KB of data and the search terms appear past the 10 KB limit, they are not highlighted.

You can get highlights for any highlight enabled field by specifying the `highlight.FIELD` parameter in your search request. For example, to get highlights for the `plot` field shown, you could specify the following:

```
search?q=star wars&highlight.plot={}
```

For more information about enabling a field to return highlights, see Configuring Index Fields (p. 64).

You can control how many occurrences of the search term(s) within an excerpt are highlighted, how they should be highlighted, and whether the excerpt is returned as plain text or HTML. When Amazon CloudSearch returns excerpts as HTML, non-alphanumeric characters are escaped with HTML entity encoding. This is done to minimize the risks associated with embedding untrusted HTML content, since the field might have originally been populated with user-generated content.

You specify highlight options as a JSON object. If the JSON object is empty, `highlight.FIELD={}`, Amazon CloudSearch highlights all occurrences of the search term(s) by enclosing them in HTML emphasis tags, *term*, and the excerpts are returned as HTML.

- To specify whether the excerpt should be returned as `text` or `html`, use the `format` option; for example, `highlight.plot={format:'text'}`.
- To specify the maximum number of occurrences of the search term(s) you want to highlight, use the `max_phrases` option; for example, `highlight.plot={max_phrases:3}`. The default is 1, the maximum is 5.
- To specify the string to prepend to each highlighted term, use the `pre_tag` option; for example, `highlight.plot={pre_tag:'<strong>', post_tag:'</strong>'}`.
- To specify the string to append to each highlighted term, use the `post_tag` option; for example, `highlight.plot={pre_tag:'<strong>', post_tag:'</strong>'}`.

Getting Autocomplete Suggestions in Amazon CloudSearch

This section describes how to configure suggesters so you can retrieve suggestions. Suggestions are possible matches for an incomplete search query—they enable you to display likely matches before

users finish typing their queries. In Amazon CloudSearch, suggestions are based on the contents of a particular text field. When you request suggestions, Amazon CloudSearch finds all of the documents whose values in the suggester field start with the specified query string—the beginning of the field must match the query string to be considered a match. The return data includes the field value and document ID for each match. You can configure suggesters to find matches for the exact query string, or to perform approximate string matching (fuzzy matching) to correct for typographical errors and misspellings.

For more information about the suggest API, see Suggest (p. 256) in the Search API Reference (p. 239).

Topics

Configuring Suggesters for Amazon CloudSearch

When you configure a suggester, you must specify the name of the text field you want to search for possible matches and a unique name for the suggester. Fields used for suggestions must be return enabled. Only the first 512 bytes of data in the field are used to generate suggestions.

Suggester names must begin with a letter and be at least three and no more than 64 characters long. The allowed characters are: a-z (lower-case letters), 0-9, and _ (underscore). The suggester name is specified in the query string when you retrieve suggestions, so it's best to use short names. The name *score* is reserved and cannot be used as a suggester name.

Suggesters also support two options:

- `FuzzyMatching`—You can set the level of fuzziness allowed when suggesting matches for a string to none, low, or high. With none, the specified string is treated as an exact prefix. With low, suggestions must differ from the specified string by no more than one character. With high, suggestions can differ by up to two characters. The default is none.
- `SortExpression`—You can configure this expression to compute a score for each suggestion to control how they are sorted. The scores are rounded to the nearest integer, with a floor of 0 and a ceiling of 2^31-1. A document's relevance score is not computed for suggestions, so sort expressions cannot reference the `_score` value. To sort suggestions using a numeric field or existing expression, simply specify the name of the field or expression. If no expression is configured for the suggester, the suggestions are sorted with the closest matches listed first. Note that an expression defined within a suggester cannot be referenced in search requests or other expressions. If want to use an expression for other purposes, add it to your domain configuration and reference it by name from the suggester. For more information about expressions, see Configuring Expressions (p. 129).

If you want to get suggestions from multiple text fields, you define a suggester for each field and submit separate suggestion requests to get matches from each suggester. You can configure up to ten suggesters.

The easiest way to define suggesters is through the **Suggesters** page in the Amazon CloudSearch console. You can also define suggesters using the AWS SDKs or AWS CLI.

> **Important**
> After you add a suggester to your search domain, you must run indexing before you can use it to retrieve suggestions. As you add and delete documents, you must periodically rebuild your index to update the suggestions. Suggestions won't reflect added or deleted documents until you call `IndexDocuments`.

Configuring Suggesters through the Amazon CloudSearch Console

You can easily add, update, and delete suggesters through the Amazon CloudSearch console.

To add a suggester

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the navigation pane, click the name of the domain, and then click the domain's **Suggesters** link.
3. In the **Suggesters** pane, click the **Add a New Suggester** button. The button is below the list of suggesters configured for the domain.
4. Enter a name for the new suggester in the **Name** field.
5. Specify the text field you want to use for suggestions in the **Source** field.
6. If you want to include suggestions that correct for minor misspellings or typos, set the **Fuzzy Matching** option to **low** or **high**. When set to low, the suggestions include terms that differ from the user query string by a single character. When set to high, the suggestions include terms that differ by up to two characters.
7. If you want to control how the suggestions are sorted, enter a numeric expression in the **Sort Expression** field. The expression can simply be the name of the numeric field you want to use to sort the suggestions, the name of an existing expression, or any valid expression. For more information about expressions, see Configuring Expressions (p. 129).
8. Click **Submit** to save your changes.
9. When you are done configuring suggesters for your search domain, you must re-index your domain before you can use the suggesters. To run indexing, go to the domain dashboard and click the **Run Indexing** button on the domain dashboard.

To update a suggester

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain, and then click the domain's **Suggesters** link.
3. In the **Suggesters** pane, modify the suggester settings.
4. Click **Submit** to save your changes.

To delete a suggester

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain, and then click the domain's **Suggesters** link.
3. In the **Suggesters** pane, click the **Delete this Suggester** link for the suggester(s) you want to remove.
4. Click **Submit** to save your changes.

Configuring Suggesters with the AWS CLI

You can add or update suggesters with the `aws cloudsearch define-suggester` command. To remove a suggester, you use `aws cloudsearch delete-suggester`.

To add or update a suggester

- Run the `aws cloudsearch define-suggester` command. You specify the configuration of the suggester in JSON with the `--suggester` option. The suggester configuration must be enclosed in quotes and all quotes within the configuration must be escaped with a backslash. For the format of the suggester configuration, see define-suggester in the AWS Command Line Interface Reference. For example, the following command configures a suggester called `mysuggester` to return suggestions based on the `title` field.

```
aws cloudsearch define-suggester --domain-name movies --suggester
 "{\"SuggesterName\": \"mysuggester\", \"DocumentSuggesterOptions\":
 {\"SourceField\":\"title\"}}"
{
    "Suggester": {
        "Status": {
            "PendingDeletion": false,
            "State": "RequiresIndexDocuments",
            "CreationDate": "2014-06-26T17:26:43Z",
            "UpdateVersion": 27,
            "UpdateDate": "2014-06-26T17:26:43Z"
        },
        "Options": {
            "DocumentSuggesterOptions": {
                "SourceField": "title"
            },
            "SuggesterName": "mysuggester"
        }
    }
}
```

  You can use the `--fuzzy-matching` option to include suggestions that correct for minor misspellings or typos. Valid values for fuzzy matching are `none`, `low`, and `high`. (The default is `none`.) When set to `low`, the suggestions will include terms that differ from the user query string by a single character. When set to `high`, the suggestions will include terms that differ by up to two characters. For example, the following command configures `mysuggester` to include suggestions that differ from the user query string by just one character:

```
aws cloudsearch  --name mysuggester --source title
  --fuzzy-matching low
```

  You can use the `--sort-expression` option to control how the returned suggestions are sorted. You can use any valid expression for sorting. (Often, this will just be the name of a numeric field or a predefined expression.) For example, to sort the suggestions returned by `mysuggester` according to the value in the `year` field, specify:

```
aws cloudsearch define-suggester --name mysuggester --source title
  --fuzzy-matching low --sort-expression year
```

To delete a suggester

- Run the `aws cloudsearch delete-suggester` command and specify the `--name` option. For example, to delete `mysuggester`:

```
aws cloudsearch delete-suggester --name mysuggester --delete
```

Configuring Suggesters Using the AWS SDKs

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `DefineSuggester (p. 162)`. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Retrieving Suggestions in Amazon CloudSearch

You retrieve suggestions by sending requests to the `suggest` resource on a domain's search endpoint via HTTP GET. For example:

```
http://search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.
amazonaws.com/2013-01-01/suggest?q=kat&suggester=mysuggester
```

You must specify the API version in the request and the query string must be URL-encoded. The maximum size of a suggestion request is 8190 bytes, including the HTTP method, URI, and protocol version.

The `suggest` resource supports four parameters:

- `q`—The string that you want to get suggestions for.
- `suggester`—The name of the suggester you want to use.
- `size`—The number of suggestions to retrieve. By default, the top ten suggestions are returned. (The suggestions are sorted according the sort expression defined in the suggester. If no sort expression is defined in the suggester, the suggestions are sorted with the closest matches listed first.)
- `format`—The content type of the response, `json` or `xml`. By default, suggestions are returned in JSON.

The `q` and `suggester` parameters must be specified. No suggestions are returned if you request suggestions for an empty string. The `size` and `format` parameters are optional.

The following example gets suggestions for the string `oce` based on the contents of the `title` field.

```
http://search-imdb-hd6ebyouhw2lczkueyuqksnuzu.us-
west-2.cloudsearch.amazonaws.com/2013-01-01/suggest -
d"q=oce&suggester=suggest_title"

{"status":{"rid":"646f5s0oDAr8pVk=","time-ms":2},
 "suggest":{
   "query":"oce",
   "found":3,
   "suggestions":[
     {"suggestion":"Ocean's Eleven","score":0,"id":"tt0054135"},
     {"suggestion":"Ocean's Thirteen","score":0,"id":"tt0496806"},
     {"suggestion":"Ocean's Twelve","score":0,"id":"tt0349903"}
   ]
 }
}
```

Controlling How Search Results are Returned in Amazon CloudSearch

You can specify parameters in your search request to control how the search results are sorted, return results in XML rather than JSON, and paginate through the result set. You can define expressions that calculate a custom value that can be used to specify search constraints or sort results.

Topics

Sorting Results in Amazon CloudSearch

By default, search results are sorted according to their relevance to the search request. A document's relevance score (`_score`) is based on how often the search terms appear in the document compared to how common the term is across all documents in the domain. Relevance scores are positive values that can vary widely depending on your data and queries. The scores for each clause in your query are additive, so queries with more clauses will naturally have higher scores than queries with just one or two. If you know what your typical queries will look like, you can do some test queries to get an idea of the range of scores you're likely to see.

To change how search results are sorted, you can:

- Use a `text` or `literal` field to sort results alphabetically. Note that Amazon CloudSearch sorts by Unicode codepoint, so numbers come before letters and uppercase letters come before lowercase letters. Numbers are sorted as strings, not by value; for example, 10 will come before 2.
- Use an `int` or `double` field to sort results numerically.
- Use a `date` field to sort results by date.
- Use a custom expression to sort results.

To use a field to sort the search results, you must configure the field to be `SortEnabled`. Only single-value fields can be `SortEnabled`—you cannot use the array-type fields for sorting. For more information about configuring fields, see Configuring Index Fields (p. 64).

To use an expression for sorting, you construct a numeric expression using `int` fields, other expressions, a document's relevance score, and numeric operators and functions. You can define expressions in your domain configuration, or within a search request. For more information about configuring expressions, see Configuring Expressions (p. 129).

You use the `sort` parameter to specify the field or expression you want to use to sort the results. You must explicitly specify the sort direction along with the name of the field or expression. For example, `sort=year asc` or `sort=year desc`.

When you use a field for sorting, documents without a value in that field are listed last. If you specify a comma separated list of fields or expressions, the first field or expression is used as the primary sort criteria, the second is used as the secondary sort criteria, and so on.

If you do not specify the `sort` parameter, the search results are ranked using the documents' default relevance scores with the highest-scoring documents listed first. This is equivalent to specifying `sort=_score desc`.

You can use the `q.options` parameter to specify field weights to apply when calculating a document's relevance `_score`. For more information, see Using Relative Field Weighting to Customize Text Relevance (p. 129).

Using Relative Field Weighting to Customize Relevance Ranking in Amazon CloudSearch

You can assign weights to selected fields so you can boost the relevance `_score` of documents with matches in key fields such as a `title` field, and minimize the impact of matches in less important fields. By default all fields have a weight of 1.

Field weights are set with the `q.options` `fields` option. You specify fields as an array of strings. To set the weight for a field, you append a caret (^) and a positive numeric value to the field name. You cannot set a field weight to zero or use mathematical functions or expressions to define a field weight.

For example, if you want matches within the `title` field to score higher than matches within the `plot` field, you could set the weight of the `title` field to 2 and the weight of the `plot` field to 0.5:

```
q.options={fields:['title^2','plot^0.5']}
```

In addition to controlling field weights, the `fields` option defines the set of fields that are searched by default if you use the simple query parser or don't specify a field in part of a compound expression when using the structured query parser. For more information, see Search Request Parameters (p. 241) in the Search API Reference.

To reference the weighted relevance score in the definition of an expression, you use `_score`. You can use the weighted `_score` value in conjunction with numeric fields, other expressions, and the standard numeric operators and functions. For more information, see Configuring Expressions (p. 129).

Configuring Expressions in Amazon CloudSearch

You can define numeric expressions and use them to sort search results. Expressions can also be returned in search results. You can add expressions to the domain configuration or define expressions within search requests.

Topics

Writing Expressions for Amazon CloudSearch

Amazon CloudSearch expressions can contain:

- Single value, sort enabled numeric fields (`int`, `double`, `date`). (You must specify a specific field, wildcards are not supported.)
- Other expressions
- The `_score` variable, which references a document's relevance score
- The `_time` variable, which references the current epoch time
- The `_rand` variable, which returns a randomly generated value
- Integer, floating point, hex, and octal literals
- Arithmetic operators: `+ - * / %`
- Bitwise operators: `| & ^ ~ << >> >>>`
- Boolean operators (including the ternary operator): `&& || ! ?:`
- Comparison operators: `< <= == >= >`
- Mathematical functions: `abs ceil exp floor ln log10 logn max min pow sqrt pow`
- Trigonometric functions: `acos acosh asin asinh atan atan2 atanh cos cosh sin sinh tanh tan`
- The `haversin` distance function

JavaScript order of precedence rules apply for operators. You can override operator precedence by using parentheses.

Shortcut evaluation is used when evaluating logical expressions—if the value of the expression can be determined after evaluating the first argument, the second argument is not evaluated. For example, in the expression `a || b`, b is only evaluated if a is not true.

Expressions always return an integer value from 0 to the maximum 64-bit signed integer value (2^63 - 1). Intermediate results are calculated as double-precision floating point values and the return value is rounded to the nearest integer. If the expression is invalid or evaluates to a negative value, it returns 0. If the expression evaluates to a value greater than the maximum, it returns the maximum value.

Expression names must begin with a letter and be at least 3 and no more than 64 characters long. The following characters are allowed: a-z (lower-case letters), 0-9, and _ (underscore). The name *score* is reserved and cannot be used as an expression name.

For example, if you define an `int` field named *popularity* for your domain, you could use that field in conjunction with the default relevance `_score` to construct a custom expression.

```
(0.3*popularity)+(0.7*_score)
```

Note that this simple example assumes that the popularity ranking and the relevance _score values are in about the same range. To tune your expressions for ranking results, you need to do some testing to determine how to weight the components of your expressions to get the results you want.

Using Date Fields in Amazon CloudSearch Expressions

The value from a `date` field is stored as an epoch time with millisecond resolution. This means you can use the mathematical and comparison operators to construct expressions using dates stored in your documents and the current epoch time (`_time`). For example, using the following expression to sort search results from the movies domain pushes movies with recent release dates toward the top of the list.

```
_score/(_time - release_date)
```

Defining Amazon CloudSearch Expressions in Search Requests

You can define and use expressions directly within a search request so that you can iterate quickly while you fine-tune expressions that you use to sort results. By defining an expression within a search request, you can also incorporate contextual information into the expression, such as the user's geographic location. You can override an expression defined in the domain configuration by defining an expression with the same name within a search request.

When you define an expression within a search request, it is not stored as part of your domain configuration. If you want to use the expression in other requests, you must define the expression in each request or add the expression to your domain configuration. Defining an expression in every request rather than adding it to the domain configuration increases the request overhead, which can result in slower response times and potentially increase the cost of running your domain. For information about adding expressions to the domain configuration, see Configuring Expressions (p. 129).

You can define and use multiple expressions in a search request. The definition of an expression can reference other expressions defined within the request, as well as expressions configured as part of the domain configuration.

There are no restrictions on how you can use expressions that you define in a search request. You can use the expression to sort the search results, define other expressions, or return computed information in the search results.

To define an expression in a search request

1. Use the `expr.`*`NAME`* parameter, where NAME is the name of the expression you are defining. For example:

   ```
   expr.rank1=log10(clicks)*_score
   ```

2. To use the expression to sort the results, specify the name of the expression with the `sort` parameter:

   ```
   search?q=terminator&expr.rank1=log10(clicks)*_score&sort=rank1 desc
   ```

3. To include the computed value in the search results, add the expression to the list of `return` fields:

   ```
   search?q=terminator&expr.rank1=log10(clicks)*_score&sort=rank1
    desc&return=rank1
   ```

For example, the following request creates two expressions that are used to sort the results and returns one of them in the search results:

```
search?
q=terminator&expr.rank1=sin( _score)&expression.rank2=cos( _score)&sort=rank1
 desc,rank2 desc&return=title,_score,rank2
```

Configuring Reusable Expressions for a Search Domain in Amazon CloudSearch

Topics

When you define an expression in a domain's configuration, you can reference the expression in any search request. Adding an expression to the domain configuration reduces the overhead of specifying it in every request, and helps maximize response times and minimize costs.

When you add an expression to your domain configuration, it takes some time for the change to be processed and the new expression to become active. To quickly test changes to an expression, you can define and use the expression directly in a search request, as described in Defining Expressions in Search Requests (p. 131). After you have finished testing and tuning an expression, you should add it to your domain configuration.

Configuring Expressions Using the Amazon CloudSearch Console

To configure an expression

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, click the name of the domain, and then click the domain's **Expressions** link.
3. In the **Expressions** pane, click the **Add a New Expression** button. The button is below the list of expressions configured for the domain.
4. Enter a name for the new expression in the **Name** field.
5. Enter the numerical expression you want to evaluate at search time in the **Expression** field. You can use the **insert...** menu to insert special values and mathematical and trigonometric functions.
6. Click **Add a New Expression** to configure additional expressions.
7. Click **Submit** to save your changes.

Configuring Amazon CloudSearch Expressions Using the AWS CLI

You use the `aws cloudsearch configure-expressions` command to define computed expressions for a domain.

To configure an expression

- Run the `aws cloudsearch define-expression` command to define a new expression. You specify a name for the expression with the `--name` option, and the numeric expression that you want to evaluate with the `--expression` option. For example, the following request creates an expression called `popularhits` that takes into account a document's `popularity` and relevance `_score`.

```
aws cloudsearch define-expression --domain-name movies --name popularhits
 --expression '((0.3*popularity)/10.0)+(0.7* _score)'

{
    "Expression": {
        "Status": {
            "PendingDeletion": false,
            "State": "Processing",
            "CreationDate": "2014-05-01T01:15:18Z",
            "UpdateVersion": 52,
            "UpdateDate": "2014-05-01T01:15:18Z"
        },
        "Options": {
            "ExpressionName": "popularhits",
            "ExpressionValue": "((0.3*popularity)/10.0)+(0.7* _score)"
        }
    }
}
```

Configuring Expressions Using the Amazon CloudSearch Configuration API

The AWS SDKs (except the Android and iOS SDKs) support all of the Amazon CloudSearch actions defined in the Amazon CloudSearch Configuration API, including `DefineExpression (p. 158)`. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

Comparing Expressions in Amazon CloudSearch

You can use the Amazon CloudSearch console to compare expressions to see how changes to the expression and field weights affect how search results are sorted.

To compare expressions

1. Sign in to the AWS Management Console and open the Amazon CloudSearch console at https://console.aws.amazon.com/cloudsearch/home.
2. In the **Navigation** pane, select a domain, and then click the domain's **Compare Expressions** link.
3. In the **Compare Expressions** pane, specify the rank expressions you want to compare. In each Expression editor, you can add a new expression or select an existing expression from the **Expressions** menu. New expressions are validated when you submit a search request. If errors are detected, the expression is highlighted in red and a description of the problem is displayed.
4. Specify the field weights to use for each expression by adjusting the sliders in the **Field Weights** menu. You can also edit the field weights directly in the expression. Field weights must be in the range 0.0 to 10.0, inclusive. By default, the weight for all fields is set to 1.0. You can set individual field weights to control how much matches in particular text or literal fields affect a document's relevance _score. You can also change the default weight.

Note
Adjusting field weights only affects result ranking if the expression references the `_score` value. You can modify the expression to change how the weight relevance `_score` contributes to a document's overall ranking. For more information, see Using Relative Field Weighting to Customize Text Relevance (p. 129).

5. Enter the terms you want to search for in the **Search** field and click **GO**. The results for the search are ranked using the specified expressions and weights. The results are refreshed whenever you make changes to the expressions or weights.

   The search results for the two expressions are shown side-by-side. (If the expression is empty, the results are sorted according to the default relevance `_score`.) Four icons highlight the differences:

    Green Up Arrow
   The document is ranked higher in the search results using the second expression.

    Red Down Arrow
   The document is ranked lower in the search results using the second expression.

    Yellow Plus
   The document is included in the search results using the second expression, but was omitted from the search results using the first expression.

    Red Minus
   The document was omitted from the search results using the second expression, but was included in the search results using the first expression.

Note
You can save expressions to your domain configuration directly from the **Compare Expressions** pane. To save either expression, select **Save Expression** from the **Expressions** menu, enter a name for the expression, and click **OK**.

Getting Results as XML in Amazon CloudSearch

By default, Amazon CloudSearch search responses are formatted in JSON. To get results as XML, specify the query parameter `format=xml` in your search request:

```
search?q=star wars&return=_no_fields&format=xml
```

Search responses formatted in XML contain exactly the same information as a JSON response:

```
<results>
    <status rid="3abhhs8oEAqMHnk=" time-ms="2"/>
    <hits found="9" start="0">
        <hit id="tt0076759"/>
        <hit id="tt0086190"/>
        <hit id="tt0121766"/>
        <hit id="tt2488496"/>
        <hit id="tt1408101"/>
        <hit id="tt0489049"/>
        <hit id="tt0120915"/>
        <hit id="tt0080684"/>
        <hit id="tt0121765"/>
    </hits>
```

```
</results>
```

For detailed information about the JSON and XML response formats for search requests, see Search Response (p. 252).

Paginating Results in Amazon CloudSearch

By default, Amazon CloudSearch returns the top ten hits according to the specified sort order. To control the number of hits returned in a result set, you use the `size` parameter.

To get the next set of hits beginning from a particular offset, you can use the `start` parameter. Note that the result set is zero-based—the first result is at index 0. You can get the first 10,000 hits using the `size` and `start` parameters. To page through more than 10,000 hits, use the `cursor` parameter. For more information, see Deep Paging Beyond 10,000 Hits (p. 135).

For example, `search?q=wolverine` returns the first 10 hits that contain *wolverine*, starting at index 0. The following example sets the `start` parameter to 10 to get the next set of ten hits.

```
search?q=wolverine&start=10
```

If you want to retrieve 25 hits at a time, set the `size` parameter to 25. To get the first set of hits, you don't have to set the `start` parameter.

```
search?q=wolverine&size=25
```

For subsequent requests, use the `start` parameter to retrieve the set of hits you want. For example, to get the third batch of 25 hits, specify the following:

```
search?q=wolverine&size=25&start=50
```

Deep Paging Beyond 10,000 Hits in Amazon CloudSearch

Using `size` and `start` to page through results works well if you only need to access the first few pages of results. However, if you need to page through thousands of hits, using a cursor is more efficient. To page through more than 10,000 hits, you must use a `cursor`. (You can only access the first 10,000 hits using the `start` and `size` parameters.)

To page through results using a cursor, you specify `cursor=initial` in your initial search request and include the `size` parameter to specify how many hits you want to get. Amazon CloudSearch returns a cursor value in the response that you use to get the next set of hits. Cursors return sequential sets of hits; however, you can use them to simulate random access of a deep page if you need to. Keep in mind that cursors are intended to be used to page through a result set within a reasonable amount of time of the initial request. Using a stale cursor can return stale results if updates have been posted to the index in the interim.

> **Important**
> When you use a cursor to page through a result set that is sorted by document score (`_score`), you can get inconsistent results if the index is updated between requests. This can also occur if your domain's replication count is greater than one, because updates are applied in an eventually consistent manner across the instances in the domain. If this is an issue, avoid sorting the results by score. You can either use the `sort` option to sort by a

particular field, or use `fq` instead of `q` to specify your search criteria. (Document scores are not calculated for filter queries.)

For example, the following request sets the `cursor` value to `initial` and the `size` parameter to `100` to get the first set of hits.

```
search?q=-star&cursor=initial&size=100
```

The cursor for the next set of hits is included in the response.

```
{
    "status": {
        "rid": "z67+3L0oHgo6swY=",
        "time-ms": 7
    },
    "hits": {
        "found": 1649,
        "start": 0,
        "cursor": "Vb-HSS4YQW9JSVFKeFpvQ2wwZERBek16SXpOems9Aw",
        "hit": [
            {
                "id": "tt0397892"
            },
            .
            .
            .
            {
                "id": "tt0332379"
            }
        ]
    }
}
```

In the next request, the `cursor` parameter specifies the returned cursor value.

```
search?q=-star&cursor=HSS4YQW9JSVFKeFpvQ2wwZERBek16SXpOems9Aw&size=100
```

Handling Errors in Amazon CloudSearch

This section provides information about how to handle errors when interacting with Amazon CloudSearch programmatically. For information about specific error codes returned by the Amazon CloudSearch services, see:

- Search Service Errors (p. 258)
- documents/batch Status Codes (p. 237)
- Configuration Service Common Errors (p. 229). For the specific errors that can be returned from a particular action, see the documentation for that action (p. 151).

Topics

Error Types in Amazon CloudSearch

The HTTP status codes returned by the Amazon CloudSearch APIs indicate whether the request completed successfully, or if a client or server error occurred while processing the request:

- 2xx status codes indicate that the client request was processed successfully.
- 4xx status codes indicate that there was a problem with the client request. Common client request errors include providing invalid credentials and omitting required parameters. When you get a 4xx error, you need to correct the problem and resubmit a properly formed client request.
- 5xx status codes indicate that a server error occurred while processing the client request. Server errors are typically transient and are often the result of server timeouts, throttling, or capacity limitations. We recommend catching and retrying all 5xx errors.

An HTTP status code is returned for every request. In addition, the body of the response provides additional warning and error information.

Messages in a `search` response indicate the severity level, the warning or error code, and a description of the problem with the search request. For a list of the warnings and errors that can

be returned by the search service, see Search Response Properties (JSON) (p. 254) or Search Response Elements (XML) (p. 255).

Errors and warnings in a `documents/batch` response provide information about parsing and validation issues encountered while processing the document data. For more information, see documents/batch Response (JSON) (p. 233) or documents/batch Response (XML) (p. 236).

Errors returned in a configuration service response provide information about what caused the request to return a 4xx or 5xx status code. For information about the common errors that all actions use, see Common Errors (p. 229). Action-specific errors are listed in the action topics in the Configuration API Reference for Amazon CloudSearch (p. 148).

Retrying Requests in Amazon CloudSearch

For your application to run smoothly, you need to build in logic to catch and respond to errors. One typical approach is to implement your request within a try block or if-then statement.

We recommend catching and retrying all server errors (5xx). Because errors can be generated from anywhere within the request pipeline, you should implement a fallback for unexpected 5xx errors in addition to any special handling for specific status codes.

507 and 509 errors typically indicate that your search service is overloaded. This can be due to the volume or complexity of search requests that you are submitting. Amazon CloudSearch normally scales automatically to handle the load. Because it takes some time to deploy additional search instances, we recommend using an exponential backoff retry policy to temporarily reduce the request rate and minimize request failures. For more information, see Error Retries and Exponential Backoff.

Certain usage patterns, such as submitting complex search queries to a single small search instance, can sometimes result in timeouts without triggering automatic scaling. If you repeatedly experience a high error rate, you can explicitly request additional capacity through the Amazon CloudSearch Service Limit Request form.

Client errors (4xx) typically indicate that you need to revise the request to correct the problem—simply retrying the same request will most likely result in the same error. 409 errors returned by the configuration service can indicate that the request was rejected because a resource limit has been reached. For more information, see Limits (p. 265).

Command Line Tool Reference for Amazon CloudSearch

Topics

You can use the AWS CLI to perform all Amazon CloudSearch API operations for search domains created with the 2013-01-01 API, including creating, configuring, and managing search domains; uploading documents; and submitting search requests. The AWS CLI is a cross-service command line interface with a simplified installation, unified configuration, and consistent command line syntax between AWS services. The AWS CLI is supported on Linux/UNIX, Windows, and Mac. For information about installing and using the AWS CLI, see the AWS Command Line Interface User Guide.

The standalone Amazon CloudSearch command line tools provide higher level tools for generating document batches and automatically configuring indexing options based on the contents of a batch.

Note
Version 2.0.0.1 of the Amazon CloudSearch command line tools also supports configuration service operations. However, no bug fixes or enhancements will be released for these operations, and they will be removed in a future version. Although you can continue to use the 2.0.0.1 command line tools to manage your search domains, you are encouraged to migrate to the AWS CLI at your earliest convenience.

The Amazon CloudSearch command line tools and sample IMDB movie data set are available from the Amazon CloudSearch developer tools page.

This section describes the commands available in version 2.0.0.2 of the Amazon CloudSearch command line tools. You can also access the reference information for each tool from the command line by specifying the `--help` option. For example, `cs-import-documents --help`.

Using the Command Line Tools for Amazon CloudSearch

Topics

This section describes how to install and run the standalone Amazon CloudSearch command line tools. For information about installing and using the AWS CLI, see the AWS Command Line Interface User Guide.

Prerequisites for Installing the Amazon CloudSearch Command Line Tools

To use the Amazon CloudSearch command line tools, you need:

- A basic familiarity with working in a Linux/UNIX or Windows environment.
- A Java 7-compatible Java Runtime Environment (JRE). You can download the latest JRE from java.com.
- A `JAVA_HOME` environment variable that points to your Java runtime. This environment variable should be set to the full path of the directory that contains the bin directory that contains the `java` (Linux/UNIX) or `java.exe` (Windows) executable.
- An AWS access key ID and secret access key. Use IAM user access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about getting credentials, see How Do I Get Security Credentials? in the *AWS General Reference*.

Installing the Command Line Tools for Amazon CloudSearch

To install the Amazon CloudSearch command line tools

1. To download the command line tools for Windows, go to https://aws.amazon.com/developertools/4320728073503020 and click the **Download** button.
2. To download the command line tools for Mac OS/Linux, go to https://aws.amazon.com/developertools/9054800585729911 and click the **Download** button.
3. Unpack the .zip or .tar.gz file. On Windows, we recommend unzipping the tools in the `C:\CloudSearch` directory.
4. Set the `CS_HOME` environment variable to point to the directory where you unpacked the tools.

   On Linux and UNIX, enter following command:

   ```
   export CS_HOME=install_directory_path
   ```

   On Windows, enter the following command:

   ```
   set CS_HOME=install_directory_path
   ```

   Note
   These examples temporarily set the `CS_HOME` and `PATH` variables for the duration of your terminal session. You can also set them permanently. On Linux and MacOSX, add the export commands to your shell startup file (.profile, .bashrc, .tcshrc, or .zshrc) in your home directory. On Windows, you can do this through the Control Panel: Control Panel > System and Security > System > Advanced > Environment Variables.

5. Add the CS_HOME environment variable to your PATH.

   On Linux and UNIX, enter following command:

   ```
   export PATH=$PATH:$CS_HOME/bin
   ```

   On Windows, enter the following command:

   ```
   set PATH=%PATH%;%CS_HOME%\bin
   ```

6. Make sure you have the Java 7 (or later) JRE installed and that the JAVA_HOME environment variable is set to the full path of the directory that contains the bin directory in which the Java executable resides. For information about checking your Java installation, go to java.com.

   Note
   On Mac OS X, JAVA_HOME should be set using the /usr/libexec/java_home command. For example: `export JAVA_HOME=$(/usr/libexec/java_home)`. For more information, see QA1170 on developer.apple.com.

7. Configure the command line tools to use your AWS identifiers. The Amazon CloudSearch command line tools look for your AWS identifiers in a text file on your local system in the location specified by the AWS_CREDENTIAL_FILE environment variable. If you have not already configured an AWS credential file, follow these steps:

   a. Use a text editor to create a two-line text file that specifies your AWS identifiers. The first line sets the accessKey property and the second line sets the secretKey property, as shown in this example:

      ```
      accessKey=AKIAIOSFODNN7EXAMPLE
      secretKey=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
      ```

   b. Save the file using any name you want (for example, account-key).

   c. Limit the file permissions to only the file owner. (For example, use `chmod 600` on the file if you are using Linux/UNIX).

   d. Set the AWS_CREDENTIAL_FILE environment variable.

      On Linux and UNIX, enter following command:

      ```
      export AWS_CREDENTIAL_FILE=credential_file_path
      ```

      On Windows, enter the following command:

      ```
      set AWS_CREDENTIAL_FILE=credential_file_path
      ```

8. Set the CS_ENDPOINT environment variable to specify the Amazon CloudSearch configuration service endpoint for the AWS region where you want to create and configure search domains. See Amazon CloudSearch Regions and Endpoints for a list of supported regions.

   On Linux and UNIX, enter following command:

   ```
   export CS_ENDPOINT=cloudsearch.region.amazonaws.com
   ```

   On Windows, enter the following command:

   ```
   set CS_ENDPOINT=cloudsearch.region.amazonaws.com
   ```

Note
If `CS_ENDPOINT` is not set, the Amazon CloudSearch command line tools default to the configuration service endpoint in the US East (N. Virginia) Region, `cloudsearch.us-east-1.amazonaws.com`. You can also explicitly set the endpoint by specifying the `--endpoint` option when you run Amazon CloudSearch commands.

9. To verify that the Amazon CloudSearch tools are configured correctly, run the `cs-import-documents` command with no arguments.

```
cs-import-documents
```

The command should return with the message, "Expected at least one input source". If it returns a different error, check the following:

- If the system cannot find the specified path, your need to set your `JAVA_HOME` environment variable to the location where the JRE is installed, such as `C:\Program Files\Java\jre6`.
- If `cs-import-documents` is not recognized as a command, make sure your PATH contains the bin directory for the command line tools, such as `/Users/username/CloudSearch/tools/bin`.

Note
If you get an `InvalidClientTokenId` error when you attempt to import documents into a domain, your AWS credentials are not configured correctly. Make sure that you've configured the `AWS_CREDENTIAL_FILE` environment variable and that your credential file contains a valid AWS access key ID and secret access key.

Running the Amazon CloudSearch Commands

All of the Amazon CloudSearch commands require an AWS access key ID and secret access key. The easiest way to specify your keys is to set up an AWS credential file and set the `AWS_CREDENTIAL_FILE` environment variable as described in the installation instructions.

You can also explicitly specify your keys with each request, either by using the `--aws-credential-file` option to specify the location of your credential file, or by specifying both the `--access-key` and `--secret-key` options.

You must also specify the name of your search domain with the `-d` or `--domain-name` option.

cs-configure-from-batches

```
NAME
       cs-configure-from-batches

DESCRIPTION
       Scans document data formatted in JSON or XML and configures index
       fields for all of the document fields. Prompts for confirmation before
       making any changes unless you specify the --force option. By default,
       fields that have already been configured are left as-is. Use the
       --replace option to overwrite the existing configuration. For more
       information, see the developer guide topic "Configuring Index Fields
       for an Amazon CloudSearch Domain".

SYNOPSIS
```

```
       cs-configure-from-batches --source FILE|S3_URI
           [--replace] [--force]
           COMMON_OPTIONS

COMMON OPTIONS
       -a, --access-key STRING
           Your AWS access key ID. Used in conjunction with --secret-key.
           Required if you do not use an AWS credential file.

       -c, --aws-credential-file FILE
           The path to the file that contains your AWS access key ID and
           secret access key. Required if you have not set the
           AWS_CREDENTIAL_FILE environment variable or explicitly set your
           credentials with --access-key and --secret-key.

       -d, --domain-name STRING
           The name of the domain that you are querying or configuring.
           Required.

       -e,  --endpoint URL
            The endpoint for the Amazon CloudSearch Configuration Service.
            Defaults to the CS_ENDPOINT environment variable or
            cloudsearch.us-east-1.amazonaws.com if the environment variable
            is not set. Optional.

       -h, --help
           Display this help message. Optional.

       -k, --secret-key STRING
           Your AWS secret access key. Used in conjunction with --access-key.
           Required if you do not use an AWS credential file.

       -ve, --verbose
           Display verbose log messages. Optional.

       -v, --version
           Display the version number of the command line tools. Optional.

CONFIGURE OPTIONS
       -f,  --force
           Apply changes to the domain's configuration without confirmation.
           Optional.

       -re, --replace
           Upload configuration information for all identified fields and
           overwrite the configuration of any fields that were already
           defined. (Prompts for confirmation unless you also specify
           --force.) Optional.

       -s, --source FILE|S3 URI
           The path to a file or an S3 URI that contains the data you want
           to scan. You can specify multiple files or S3 URIs. For example:
           --source batch1.json batch2.json. Required.

EXAMPLE
       cs-configure-from-batches -d mydomain --source s3://mybucket/
myAmazingDataSet
           COMMON_OPTIONS
```

cs-import-documents

```
NAME
       cs-import-documents

DESCRIPTION
       Uploads documents to a search domain for indexing. If necessary,
       the source data is analyzed and converted to JSON or XML so it
       can be indexed by Amazon CloudSearch. The data source can
       be a local directory or file, an S3 bucket, or a DynamoDB table.

       Specify the --domain option to upload the documents to your search
       domain. To save the generated JSON or XML files to your local file
       system or an S3 bucket, specify the --output option.

       The cs-import-documents command can process the following content
       types:

       text/csv
       text/html
       text/plain
       application/msword
       application/pdf
       application/vnd.ms-excel
       application/vnd.ms-powerpoint
       application/vnd.openxmlformats-officedocument.
           presentationml.presentation
       application/vnd.openxmlformats-officedocument.
           spreadsheetml.sheet
       application/vnd.openxmlformats-officedocument.
           wordprocessingml.document

       For more information, see the developer guide topic "Generating JSON
       or XML from Your Source Data for Amazon CloudSearch".

SYNOPSIS
       cs-import-documents --source PATH|S3_URI|DDB_TABLE
           [--output PATH|S3_URI]
           [--modified-after yyyy-MM-ddTHH:mm:ssZ]
           [--exclude-metadata] [--exclude-content]
           [--single-doc-per-csv] [--multivalued FIELDS]
           [--sdf-format json|xml] [--docid-prefix STRING]
           [--batch-size MB] [--batch-docs NUM]
           [--num-rows NUM] [--dynamodb-rcu-percent NUM]
           [--start-hash-key STRING] [--start-range-key STRING]
           [--delimiter CHAR] [--encapsulator CHAR]
           [--comment-character CHAR]
           COMMON_OPTIONS

COMMON OPTIONS
       -a, --access-key STRING
           Your AWS access key ID. Used in conjunction with --secret-key.
           Required if you do not use an AWS credential file.

       -c, --aws-credential-file FILE
           The path to the file that contains your AWS access key ID and
           secret access key. Required if you have not set the
```

```
            AWS_CREDENTIAL_FILE environment variable or explicitly set your
            credentials with --access-key and --secret-key.

       -d, --domain-name STRING
            The name of the domain that you are querying or configuring.
            Required.

       -e,  --endpoint URL
             The endpoint for the Amazon CloudSearch Configuration Service.
             Defaults to the CS_ENDPOINT environment variable or
             cloudsearch.us-east-1.amazonaws.com if the environment variable
             is not set. Optional.

       -h, --help
            Display this help message. Optional.

       -k, --secret-key STRING
            Your AWS secret access key. Used in conjunction with --access-key.
            Required if you do not use an AWS credential file.

       -ve, --verbose
            Display verbose log messages. Optional.

       -v, --version
            Display the version number of the command line tools. Optional.

BASIC OPTIONS
       -o, --output PATH|S3_URI
            The local directory or S3 bucket where you want to save the
            generated JSON or XML files. Required if you do not specify the
            --domain option to upload the documents to a search domain.

       -s, --source PATH|S3_URI|DDB_TABLE
            The local directory or file, S3 bucket, or DynamoDB table that
            contains your source data. You can process data from multiple
            locations by specifying multiple sources. For example:
            --source c:\DataSet1 c:\DataSet2. Supports wildcards for
            filenames, directories, and S3 prefixes: ? matches any single
            character, * matches zero or more characters, ** matches zero
            or more directories or prefixes. Required.

ADVANCED OPTIONS
       -bd, --batch-docs NUM
            The maximum number of documents in a batch. Optional.

       -bs, --batch-size MB
            The maximum batch size in MB. Defaults to 5MB. Optional.

       -char, --comment-character CHAR
            The character used to identify comments in CSV source files. If
 not
            specified, the default comment character is a hash character (#).
            Optional.

       -del, --delimiter CHAR
            The character used to delimit fields in CSV source files. If not
            specified, the default delimiter is a comma (,). Optional.
```

```
       -dp, --docid-prefix STRING
           The prefix to prepend to the document ID while processing CSV
 data.
           If not specified, the filename is used as the --docid-prefix. The
           docid column is used as the document ID if it is included in the
           CSV data; otherwise, the row number is used as the document ID.
           Optional.

       -enc, --encapsulator CHAR
           The character used to encapsulate individual values of a
           multi-valued field in CSV source files. If not specified, the
           default encapsulator is a double quote (\"). Optional.

       -ec, --exclude-content
           Do not include the content of the source files in the generated
           JSON or XML, only process the metadata. Optional.

       -em, --exclude-metadata
           Do not include the metadata of the source files in the generated
           JSON or XML, only  process the content. Optional.

       -m, --modified-after TIMESTAMP
           Only process files or S3 objects modified after the specified date
           and time. Specified in RFC 822 time zone format
           (yyyy-MM-dd'T'HH:mm:ssZ). For example, 2012-12-12T01:00:00GMT.
           Optional.

       -mv, --multivalued FIELDS
           Treat the specified fields as multi-valued fields when processing
           CSV files. Specify multiple fields as a comma-separated list.

           If no fields are specified, all fields other than docid are
           processed as multi-valued fields. This option is not valid if
           the -sdpc option is specified and it has no effect on non-CSV
           files. Optional.

       -format, --sdf-format json|xml
           The format of the generated documents: json or xml. Defaults to
           json. Optional.

        -sdpc, --single-doc-per-csv
           Treat each CSV file as a single document. If this option is
           specified, the contents of a CSV file are treated as a single text
           field. This option is not valid if the -mv  option is specified
 and
           it has no effect on non-CSV files. Optional.

DynamoDB SOURCE OPTIONS

       -drp, --dynamodb-rcu-percent
           The maximum percentage of configured read capacity units to use
           while reading from the DynamoDB table. By default, the maximum
           number of read capacity units is set to 20% the table's configured
           read capacity units. Optional.

       -n, --num-rows
           The maximum number of rows to read from the DynamoDB table.
 Optional.
           By default, the entire table is read.
```

```
       -shk, --start-hash-key
           The hash attribute of the item in the DynamoDB table where you
 want
           to begin reading. If the table has a hash and range type primary
           key, the --start-range-key option must also be specified. By
           default, the table is read starting with the first item.
 Optional.

       -srk, --start-range-key
           The range attribute of the item in the DynamoDB table where you
           want to begin reading. Required if --start-hash-key is specified
           and the DynamoDB table has a hash and range type primary key. Not
           used if the table has a hash type primary key. Optional.

EXAMPLES
       Process all of the source documents in a directory and upload the data
       for indexing:

       cs-import-documents -d mydomain --source c:\myAmazingDataSet\*
           COMMON_OPTIONS

       Process a DynamoDB table and save the generated XML files to a local
       directory:

       cs-import-documents --source ddb://myDDBTable
           --output c:\myAmazingDataSet\SDF\batch -format xml
           COMMON_OPTIONS
```

Amazon CloudSearch API Reference

You use three APIs to interact with Amazon CloudSearch:

- Configuration API (p. 148)—Set up and manage your search domain.
- Document Service API (p. 230)—Submit the data you want to search.
- Search API (p. 239)—Search your domain.

Configuration API Reference for Amazon CloudSearch

You use the Amazon CloudSearch Configuration API to create, configure, and manage search domains. For more information configuring search domains, see Creating and Managing Search Domains (p. 26).

The other APIs you use to interact with Amazon CloudSearch are:

- Document Service API Reference (p. 230)—Submit the data you want to search.
- Search API Reference (p. 239)—Search your domain.

Topics

Submitting Configuration Requests in Amazon CloudSearch

Important
The easiest way to submit configuration requests is to use the Amazon CloudSearch console, Amazon CloudSearch command line tools, or the AWS SDK for Java, JavaScript, .NET, PHP, Ruby, or Python (Boto). The command line tools and SDKs handle the signing process for you and ensure that Amazon CloudSearch configuration requests are properly formed. For more information about using the command line tools, see the Command Line Tool Reference (p. 139). For more information about the AWS SDKs, see AWS Software Development Kits.

You submit Amazon CloudSearch configuration requests to the Amazon CloudSearch endpoint for your region using the AWS Query protocol. For the current list of supported regions and endpoints, see Regions and Endpoints.

AWS Query requests are HTTP or HTTPS requests submitted via HTTP GET or POST with a Query parameter named Action. You must specify the API version in all configuration requests and that version must match the API version specified when the domain was created.

Requests submitted to the Configuration API are authenticated using your AWS access key ID and secret access key. Use IAM user access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about getting credentials, see How Do I Get Security Credentials? in the *AWS General Reference*.

You must include authorization parameters and a digital signature in every request. Amazon CloudSearch supports AWS Signature Version 4. For detailed signing instructions, see Signature V4 Signing Process in the AWS General Reference.

Structure of a Configuration Request

This reference shows Amazon CloudSearch configuration requests as URLs, which can be used directly in a browser. (Although the GET requests are shown as URLs, the parameter values are shown unencoded to make them easier to read. Keep in mind that you must URL encode parameter values when submitting requests.) The URL contains three parts:

- Endpoint—the Web service entry point to act on, `cloudsearch.us-east-1.amazonaws.com`.
- Action—the Amazon CloudSearch configuration action you want to perform. For a complete list of actions, see Actions (p. 151).
- Parameters—any request parameters required for the specified action. Each query request must also include some common parameters to handle authentication. For more information, see Request Authentication (p. 150).

You must specify the `Version` parameter in every Amazon CloudSearch configuration request. The current Amazon CloudSearch API version is 2013-01-01.

For example, the following GET request creates a new search domain called *movies*:

```
https://cloudsearch.us-east-1.amazonaws.com
?Action=CreateDomain
&DomainName=movies
&Version=2013-01-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120712/us-east-1/cloudsearch/aws4
_request
&X-Amz-Date=2012-07-12T21:41:29.094Z
```

```
&X-Amz-SignedHeaders=host
&X-Amz-
Signature=c7600a00fea082dac002b247f9d6812f25195fbaf7f0a6fc4ce08a39666c6a10
3c8dcb
```

Request Authentication

Requests submitted to the Configuration API are authenticated using your AWS access keys. You must include authorization parameters and a digital signature in every request. Amazon CloudSearch supports AWS Signature Version 4. For detailed signing instructions, see Signature V4 Signing Process in the AWS General Reference.

To create a signature for a request, you create a canonicalized version of the query string and compute an RFC 2104-compliant HMAC signature using a signing key derived from your AWS Secret Access key.

> **Note**
> If you are just getting started signing your own AWS requests, take a look at how the SDKs implement signing. The source for most of the AWS SDKs is available at https://github.com/aws.

For example, to construct a `CreateDomain` request, you need the following information:

```
Region name: us-east-1
Service name: cloudsearch
API version: 2013-01-01
Date: 2014-03-12T21:41:29.094Z
Access key: AKIAIOSFODNN7EXAMPLE
Secret key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Action: CreateDomain
Action Parameters: DomainName=movies
```

The canonical query string for a `CreateDomain` request looks like this:

```
Action=CreateDomain
&DomainName=movies
&Version=2013-01-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120712/us-east-1/cloudsearch/aws4
_request
&X-Amz-Date=2012-07-12T21:41:29.094Z
&X-Amz-SignedHeaders=host
```

The final signed request looks like this:

```
https://cloudsearch.us-east-1.amazonaws.com
?Action=CreateDomain
&DomainName=movies
&Version=2013-01-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120712/us-east-1/cloudsearch/aws4
_request
&X-Amz-Date=2014-03-12T21:41:29.094Z
&X-Amz-SignedHeaders=host
&X-Amz-
Signature=c7600a00fea082dac002b247f9d6812f25195fbaf7f0a6fc4ce08a39666c6a10
```

Actions

The following actions are supported:

AddTags

Description

Attaches resource tags to an Amazon CloudSearch domain. For more information, see Tagging Amazon CloudSearch Domains (p. 54) in the *Amazon CloudSearch Developer Guide*. Use the `GET` HTTP method with this action.

Request Parameters

`ARN`

Amazon Resource Name (ARN) for the Amazon CloudSearch domain to which you want to attach resource tags. For more information, see IAM ARNs in the *AWS Identity and Access Management* documentation.

Type: String

Required: Yes

`TagList`

List of resource tags for the specified Amazon CloudSearch domain.

Type: TagList, a list of strings specifying the resource tags for the domain.

Required: Yes

Response Elements

Not applicable. The `AddTags` action does not return a data structure.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

LimitExceededException
The request contains more than the allowed number and type of Amazon CloudSearch domain resources. Returns HTTP status code 409.

ValidationException
The request contains invalid input or is missing required input. Returns HTTP status code 400.

InternalException
The processing of the request failed because of an internal service error. Returns HTTP status code 500.

Example

The following example attaches a single resource tag with a tag key of `project` to the `logs` Amazon CloudSearch domain in the us-west-2 region:

Request

```
GET https://cloudsearch.us-west-2.amazonaws.com?
Action=AddTags&ARN=arn:aws:cloudsearch:us-west-2:408853051459:domain/
logs&TagList.Tag.1.Key='environment'
```

```
&TagList.Tag.1.Value='production'&Version=2013-01-01
```

Response

```
<AddTagsResponse xmlns="http://cloudsearch.amazonaws.com/doc/2013-01-01/">
  <ResponseMetadata>
    <RequestId>5646a576-d1ee-11e5-bc4d-27ea242580ce</RequestId>
  </ResponseMetadata>
</AddTagsResponse>
```

BuildSuggesters

Description

Indexes the search suggestions. For more information, see Configuring Suggesters in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `BuildSuggestersResult`.

FieldNames
A list of field names.

Type: String list

Length constraints: Minimum length of 1. Maximum length of 64.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

ResourceNotFound
The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

CreateDomain

Description

Creates a new search domain. For more information, see Creating a Search Domain in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A name for the domain you are creating. Allowed characters are a-z (lower-case letters), 0-9, and hyphen (-). Domain names must start with a letter or number and be at least 3 and no more than 28 characters long.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `CreateDomainResult`.

DomainStatus
The current status of the search domain.

Type: DomainStatus (p. 210)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

LimitExceeded
The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

DefineAnalysisScheme

Description

Configures an analysis scheme that can be applied to a `text` or `text-array` field to define language-specific text processing options. For more information, see Configuring Analysis Schemes in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

AnalysisScheme
Configuration information for an analysis scheme. Each analysis scheme has a unique name and specifies the language of the text to be processed. The following options can be configured for an analysis scheme: `Synonyms`, `Stopwords`, `StemmingDictionary`, `JapaneseTokenizationDictionary` and `AlgorithmicStemming`.

Type: AnalysisScheme (p. 201)

Required: Yes

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DefineAnalysisSchemeResult`.

AnalysisScheme
The status and configuration of an `AnalysisScheme`.

Type: AnalysisSchemeStatus (p. 201)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DefineExpression

Description

Configures an `Expression` for the search domain. Used to create new expressions and modify existing ones. If the expression exists, the new configuration replaces the old one. For more information, see Configuring Expressions in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Expression
A named expression that can be evaluated at search time. Can be used to sort the search results, define other expressions, or return computed information in the search results.

Type: Expression (p. 213)

Required: Yes

Response Elements

The following element is returned in a structure named `DefineExpressionResult`.

Expression
The value of an `Expression` and its current status.

Type: ExpressionStatus (p. 213)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded
The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound
The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DefineIndexField

Description

Configures an `IndexField` for the search domain. Used to create new fields and modify existing ones. You must specify the name of the domain you are configuring and an index field configuration. The index field configuration specifies a unique name, the index field type, and the options you want to configure for the field. The options you can specify depend on the `IndexFieldType`. If the field exists, the new configuration replaces the old one. For more information, see Configuring Index Fields in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

IndexField
The index field and field options you want to configure.

Type: IndexField (p. 214)

Required: Yes

Response Elements

The following element is returned in a structure named `DefineIndexFieldResult`.

IndexField
The value of an `IndexField` and its current status.

Type: IndexFieldStatus (p. 216)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DefineSuggester

Description

Configures a suggester for a domain. A suggester enables you to display possible matches before users finish typing their queries. When you configure a suggester, you must specify the name of the text field you want to search for possible matches and a unique name for the suggester. For more information, see Getting Search Suggestions in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Suggester
Configuration information for a search suggester. Each suggester has a unique name and specifies the text field you want to use for suggestions. The following options can be configured for a suggester: `FuzzyMatching`, `SortExpression`.

Type: Suggester (p. 224)

Required: Yes

Response Elements

The following element is returned in a structure named `DefineSuggesterResult`.

Suggester
The value of a `Suggester` and its current status.

Type: SuggesterStatus (p. 224)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DeleteAnalysisScheme

Description

Deletes an analysis scheme. For more information, see Configuring Analysis Schemes in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

AnalysisSchemeName
The name of the analysis scheme you want to delete.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DeleteAnalysisSchemeResult`.

AnalysisScheme
The status of the analysis scheme being deleted.

Type: AnalysisSchemeStatus (p. 201)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DeleteDomain

Description

Permanently deletes a search domain and all of its data. Once a domain has been deleted, it cannot be recovered. For more information, see Deleting a Search Domain in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
The name of the domain you want to permanently delete.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DeleteDomainResult`.

DomainStatus
The current status of the search domain.

Type: DomainStatus (p. 210)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

DeleteExpression

Description

Removes an `Expression` from the search domain. For more information, see Configuring Expressions in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

ExpressionName
The name of the `Expression` to delete.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

Response Elements

The following element is returned in a structure named `DeleteExpressionResult`.

Expression
The status of the expression being deleted.

Type: ExpressionStatus (p. 213)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DeleteIndexField

Description

Removes an `IndexField` from the search domain. For more information, see Configuring Index Fields in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

IndexFieldName
The name of the index field your want to remove from the domain's indexing options.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

Response Elements

The following element is returned in a structure named `DeleteIndexFieldResult`.

IndexField
The status of the index field being deleted.

Type: IndexFieldStatus (p. 216)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DeleteSuggester

Description

Deletes a suggester. For more information, see Getting Search Suggestions in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

SuggesterName
Specifies the name of the suggester you want to delete.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

Response Elements

The following element is returned in a structure named `DeleteSuggesterResult`.

Suggester
The status of the suggester being deleted.

Type: SuggesterStatus (p. 224)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeAnalysisSchemes

Description

Gets the analysis schemes configured for a domain. An analysis scheme defines language-specific text processing options for a `text` field. Can be limited to specific analysis schemes by name. By default, shows all analysis schemes and includes any pending changes to the configuration. Set the `Deployed` option to `true` to show the active configuration and exclude pending changes. For more information, see Configuring Analysis Schemes in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

AnalysisSchemeNames.member.N
The analysis schemes you want to describe.

Type: String list

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

Deployed
Whether to display the deployed configuration (`true`) or include any pending changes (`false`). Defaults to `false`.

Type: Boolean

Required: No

DomainName
The name of the domain you want to describe.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DescribeAnalysisSchemesResult`.

AnalysisSchemes
The analysis scheme descriptions.

Type: AnalysisSchemeStatus (p. 201) list

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeAvailabilityOptions

Description

Gets the availability options configured for a domain. By default, shows the configuration with any pending changes. Set the `Deployed` option to `true` to show the active configuration and exclude pending changes. For more information, see Configuring Availability Options in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

Deployed
Whether to display the deployed configuration (`true`) or include any pending changes (`false`). Defaults to `false`.

Type: Boolean

Required: No

DomainName
The name of the domain you want to describe.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DescribeAvailabilityOptionsResult`.

AvailabilityOptions
The availability options configured for the domain. Indicates whether Multi-AZ is enabled for the domain.

Type: AvailabilityOptionsStatus (p. 202)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

DisabledOperation
The request was rejected because it attempted an operation which is not enabled.

HTTP Status Code: 409

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded
The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound
The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeDomains

Description

Gets information about the search domains owned by this account. Can be limited to specific domains. Shows all domains by default. To get the number of searchable documents in a domain, use the console or submit a `matchall` request to your domain's search endpoint: `q=matchall&q.parser=structured&size=0`. For more information, see Getting Information about a Search Domain in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainNames.member.N
The names of the domains you want to include in the response.

Type: String list

Length constraints: Minimum length of 3. Maximum length of 28.

Required: No

Response Elements

The following element is returned in a structure named `DescribeDomainsResult`.

DomainStatusList
A list that contains the status of each requested domain.

Type: DomainStatus (p. 210) list

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

DescribeExpressions

Description

Gets the expressions configured for the search domain. Can be limited to specific expressions by name. By default, shows all expressions and includes any pending changes to the configuration. Set the `Deployed` option to `true` to show the active configuration and exclude pending changes. For more information, see Configuring Expressions in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

Deployed
Whether to display the deployed configuration (`true`) or include any pending changes (`false`). Defaults to `false`.

Type: Boolean

Required: No

DomainName
The name of the domain you want to describe.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

ExpressionNames.member.N
Limits the `DescribeExpressions` response to the specified expressions. If not specified, all expressions are shown.

Type: String list

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

Response Elements

The following element is returned in a structure named `DescribeExpressionsResult`.

Expressions
The expressions configured for the domain.

Type: ExpressionStatus (p. 213) list

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeIndexFields

Description

Gets information about the index fields configured for the search domain. Can be limited to specific fields by name. By default, shows all fields and includes any pending changes to the configuration. Set the `Deployed` option to `true` to show the active configuration and exclude pending changes. For more information, see Getting Domain Information in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

Deployed
Whether to display the deployed configuration (`true`) or include any pending changes (`false`). Defaults to `false`.

Type: Boolean

Required: No

DomainName
The name of the domain you want to describe.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

FieldNames.member.N
A list of the index fields you want to describe. If not specified, information is returned for all configured index fields.

Type: String list

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

Response Elements

The following element is returned in a structure named `DescribeIndexFieldsResult`.

IndexFields
The index fields configured for the domain.

Type: IndexFieldStatus (p. 216) list

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeScalingParameters

Description

Gets the scaling parameters configured for a domain. A domain's scaling parameters specify the desired search instance type and replication count. For more information, see Configuring Scaling Options in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DescribeScalingParametersResult`.

ScalingParameters
The status and configuration of a search domain's scaling parameters.

Type: ScalingParametersStatus (p. 223)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

ResourceNotFound
The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeServiceAccessPolicies

Description

Gets information about the access policies that control access to the domain's document and search endpoints. By default, shows the configuration with any pending changes. Set the `Deployed` option to `true` to show the active configuration and exclude pending changes. For more information, see Configuring Access for a Search Domain in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

Deployed
Whether to display the deployed configuration (`true`) or include any pending changes (`false`). Defaults to `false`.

Type: Boolean

Required: No

DomainName
The name of the domain you want to describe.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DescribeServiceAccessPoliciesResult`.

AccessPolicies
The access rules configured for the domain specified in the request.

Type: AccessPoliciesStatus (p. 199)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

ResourceNotFound
The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeSuggesters

Description

Gets the suggesters configured for a domain. A suggester enables you to display possible matches before users finish typing their queries. Can be limited to specific suggesters by name. By default, shows all suggesters and includes any pending changes to the configuration. Set the `Deployed` option to `true` to show the active configuration and exclude pending changes. For more information, see Getting Search Suggestions in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

Deployed
Whether to display the deployed configuration (`true`) or include any pending changes (`false`). Defaults to `false`.

Type: Boolean

Required: No

DomainName
The name of the domain you want to describe.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

SuggesterNames.member.N
The suggesters you want to describe.

Type: String list

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

Response Elements

The following element is returned in a structure named `DescribeSuggestersResult`.

Suggesters
The suggesters configured for the domain specified in the request.

Type: SuggesterStatus (p. 224) list

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

IndexDocuments

Description

Tells the search domain to start indexing its documents using the latest indexing options. This operation must be invoked to activate options whose OptionStatus (p. 222) is `RequiresIndexDocuments`.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `IndexDocumentsResult`.

FieldNames
The names of the fields that are currently being indexed.

Type: String list

Length constraints: Minimum length of 1. Maximum length of 64.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

ResourceNotFound
The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

ListDomainNames

Description

Lists all search domains owned by an account.

Response Elements

The following element is returned in a structure named `ListDomainNamesResult`.

DomainNames
The names of the search domains owned by an account.

Type: String to String map

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

UpdateAvailabilityOptions

Description

Configures the availability options for a domain. Enabling the Multi-AZ option expands an Amazon CloudSearch domain to an additional Availability Zone in the same Region to increase fault tolerance in the event of a service disruption. Changes to the Multi-AZ option can take about half an hour to become active. For more information, see Configuring Availability Options in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

MultiAZ
You expand an existing search domain to a second Availability Zone by setting the Multi-AZ option to true. Similarly, you can turn off the Multi-AZ option to downgrade the domain to a single Availability Zone by setting the Multi-AZ option to `false`.

Type: Boolean

Required: Yes

Response Elements

The following element is returned in a structure named `UpdateAvailabilityOptionsResult`.

AvailabilityOptions
The newly-configured availability options. Indicates whether Multi-AZ is enabled for the domain.

Type: AvailabilityOptionsStatus (p. 202)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

DisabledOperation
The request was rejected because it attempted an operation which is not enabled.

HTTP Status Code: 409

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded
The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound
The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

ListTags

Description

Displays all of the resource tags for an Amazon CloudSearch domain. Use the `GET` HTTP method with this action. For more information, see Tagging Amazon CloudSearch Domains (p. 54) in the *Amazon CloudSearch Developer Guide*.

Request Parameters

`ARN`

Amazon Resource Name (ARN) for the Amazon CloudSearch domain to which you want to attach resource tags. For more information, see IAM ARNs in the *AWS Identity and Access Management* documentation.

Type: String

Required: Yes

Response Elements

`TagList`

List of resource tags for the specified Amazon CloudSearch domain.

Type: TagList, a list of strings specifying the resource tags for the domain.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
: An error occurred while processing the request.

LimitExceededException
: The request contains more than the allowed number and type of Amazon CloudSearch domain resources. Returns HTTP status code 409.

ValidationException
: The request contains invalid input or is missing required input. Returns HTTP status code 400.

InternalException
: The processing of the request failed because of an internal service error. Returns HTTP status code 500.

Example

The following example lists the tags attached to the `logs` Amazon CloudSearch domain in the us-west-2 region:

```
GET https://cloudsearch.us-west-2.amazonaws.com?
Action=ListTags&ARN=arn:aws:cloudsearch:us-west-2:408853051459:domain/
logs&Version=2013-01-01
```

The operation returns the following response:

```
<ListTagsResponse xmlns="http://cloudsearch.amazonaws.com/doc/2013-01-01/">
  <ListTagsResult>
```

```
    <TagList>
      <member>
        <Value>environment</Value>
        <Key>production</Key>
      </member>
    </TagList>
  </ListTagsResult>
  <ResponseMetadata>
    <RequestId>29948ea4-d1dc-11e5-8914-51ab8964f46d</RequestId>
  </ResponseMetadata>
</ListTagsResponse>
```

RemoveTags

Description

Removes the specified resource tags from an Amazon ES domain. For more information, see Tagging Amazon CloudSearch Domains (p. 54) in the *Amazon CloudSearch Developer Guide*. Use the `GET` HTTP method with this action.

Request Parameters

`ARN`

Amazon Resource Name (ARN) for the Amazon CloudSearch domain to which you want to attach resource tags. For more information, see IAM ARNs in the *AWS Identity and Access Management* documentation.

Type: String

Required: Yes

`TagKeys`

List of `TagKey` elements for the resource tags that you want to remove from an Amazon CloudSearch domain. A `TagKey` element is a string with a maximum of 128 characters that contains the name of the resource tag.

Response Elements

This operation does not return a response element.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

ValidationException
The request contains invalid input or is missing required input. Returns HTTP status code 400.

InternalException
The processing of the request failed because of an internal service error. Returns HTTP status code 500.

Example

The following example deletes a resource tag with a tag key of `project` from the `logs` Amazon CloudSearch domain in the us-west-2 region:

```
GET https://cloudsearch.us-west-2.amazonaws.com?
Action=RemoveTags&ARN=arn:aws:cloudsearch:us-west-2:408853051459:domain/
logs&TagKeys.member.1='environment'&Version=2013-01-01
```

Response

```
<RemoveTagsResponse xmlns="http://cloudsearch.amazonaws.com/doc/2013-01-01/">
  <ResponseMetadata>
    <RequestId>2bf75153-d1f1-11e5-8f64-f17d14275591</RequestId>
```

```
  </ResponseMetadata>
</RemoveTagsResponse>
```

UpdateScalingParameters

Description

Configures scaling parameters for a domain. A domain's scaling parameters specify the desired search instance type and replication count. Amazon CloudSearch will still automatically scale your domain based on the volume of data and traffic, but not below the desired instance type and replication count. If the Multi-AZ option is enabled, these values control the resources used per Availability Zone. For more information, see Configuring Scaling Options in the *Amazon CloudSearch Developer Guide*.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

ScalingParameters
The desired instance type and desired number of replicas of each index partition.

Type: ScalingParameters (p. 222)

Required: Yes

Response Elements

The following element is returned in a structure named `UpdateScalingParametersResult`.

ScalingParameters
The status and configuration of a search domain's scaling parameters.

Type: ScalingParametersStatus (p. 223)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

UpdateServiceAccessPolicies

Description

Configures the access rules that control access to the domain's document and search endpoints. For more information, see Configuring Access for an Amazon CloudSearch Domain.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters (p. 227).

AccessPolicies
The access rules you want to configure. These rules replace any existing rules.

Type: String

Required: Yes

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `UpdateServiceAccessPoliciesResult`.

AccessPolicies
The access rules configured for the domain.

Type: AccessPoliciesStatus (p. 199)

Errors

For information about the errors that are common to all actions, see Common Errors (p. 229).

Base
An error occurred while processing the request.

HTTP Status Code: 400

Internal
An internal error occurred while processing the request. If this problem persists, report an issue from the Service Health Dashboard.

HTTP Status Code: 500

InvalidType
The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded
The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound
The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

Data Types

The Amazon CloudSearch Configuration Service API contains several data types that various actions use. This section describes each data type in detail.

Note
The order of each element in the response is not guaranteed. Applications should not assume a particular order.

The following data types are supported:

AccessPoliciesStatus

Description

The configured access rules for the domain's document and search endpoints, and the current status of those rules.

Contents

Options

Access rules for a domain's document or search service endpoints. For more information, see Configuring Access for a Search Domain in the *Amazon CloudSearch Developer Guide*. The maximum size of a policy document is 100 KB.

Type: String

Required: Yes

Status

The status of domain configuration option.

Type: OptionStatus (p. 222)

Required: Yes

AnalysisOptions

Description

Synonyms, stopwords, and stemming options for an analysis scheme. Includes tokenization dictionary for Japanese.

Contents

AlgorithmicStemming
The level of algorithmic stemming to perform: `none`, `minimal`, `light`, or `full`. The available levels vary depending on the language. For more information, see Language Specific Text Processing Settings in the *Amazon CloudSearch Developer Guide*

Type: String

Valid Values: `none | minimal | light | full`

Required: No

JapaneseTokenizationDictionary
A JSON array that contains a collection of terms, tokens, readings and part of speech for Japanese Tokenizaiton. The Japanese tokenization dictionary enables you to override the default tokenization for selected terms. This is only valid for Japanese language fields.

Type: String

Required: No

StemmingDictionary
A JSON object that contains a collection of string:value pairs that each map a term to its stem. For example, `{"term1": "stem1", "term2": "stem2", "term3": "stem3"}`. The stemming dictionary is applied in addition to any algorithmic stemming. This enables you to override the results of the algorithmic stemming to correct specific cases of overstemming or understemming. The maximum size of a stemming dictionary is 500 KB.

Type: String

Required: No

Stopwords
A JSON array of terms to ignore during indexing and searching. For example, `["a", "an", "the", "of"]`. The stopwords dictionary must explicitly list each word you want to ignore. Wildcards and regular expressions are not supported.

Type: String

Required: No

Synonyms
A JSON object that defines synonym groups and aliases. A synonym group is an array of arrays, where each sub-array is a group of terms where each term in the group is considered a synonym of every other term in the group. The aliases value is an object that contains a collection of string:value pairs where the string specifies a term and the array of values specifies each of the aliases for that term. An alias is considered a synonym of the specified term, but the term is not considered a synonym of the alias. For more information about specifying synonyms, see Synonyms in the *Amazon CloudSearch Developer Guide*.

Type: String

Required: No

AnalysisScheme

Description

Configuration information for an analysis scheme. Each analysis scheme has a unique name and specifies the language of the text to be processed. The following options can be configured for an analysis scheme: `Synonyms`, `Stopwords`, `StemmingDictionary`, `JapaneseTokenizationDictionary` and `AlgorithmicStemming`.

Contents

AnalysisOptions
Synonyms, stopwords, and stemming options for an analysis scheme. Includes tokenization dictionary for Japanese.

Type: AnalysisOptions (p. 200)

Required: No

AnalysisSchemeLanguage
An IETF RFC 4646 language code or `mul` for multiple languages.

Type: String

Valid Values: `ar | bg | ca | cs | da | de | el | en | es | eu | fa | fi | fr | ga | gl | he | hi | hu | hy | id | it | ja | ko | lv | mul | nl | no | pt | ro | ru | sv | th | tr | zh-Hans | zh-Hant`

Required: Yes

AnalysisSchemeName
Names must begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore).

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

AnalysisSchemeStatus

Description

The status and configuration of an `AnalysisScheme`.

Contents

Options
Configuration information for an analysis scheme. Each analysis scheme has a unique name and specifies the language of the text to be processed. The following options can be configured for an analysis scheme: `Synonyms`, `Stopwords`, `StemmingDictionary`, `JapaneseTokenizationDictionary` and `AlgorithmicStemming`.

Type: AnalysisScheme (p. 201)

Required: Yes

Status
The status of domain configuration option.

Type: OptionStatus (p. 222)

Required: Yes

AvailabilityOptionsStatus

Description

The status and configuration of the domain's availability options.

Contents

Options
The availability options configured for the domain.

Type: Boolean

Required: Yes

Status
The status of domain configuration option.

Type: OptionStatus (p. 222)

Required: Yes

BuildSuggestersResult

Description

The result of a `BuildSuggester` request. Contains a list of the fields used for suggestions.

Contents

FieldNames
A list of field names.

Type: String list

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

CreateDomainResult

Description

The result of a `CreateDomainRequest`. Contains the status of a newly created domain.

Contents

DomainStatus
The current status of the search domain.

Type: DomainStatus (p. 210)

Required: No

DateArrayOptions

Description

Options for a field that contains an array of dates. Present if `IndexFieldType` specifies the field is of type `date-array`. All options are enabled by default.

Contents

DefaultValue
A value to use for the field if the field isn't specified for a document.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

FacetEnabled
Whether facet information can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SearchEnabled
Whether the contents of the field are searchable.

Type: Boolean

Required: No

SourceFields
A list of source fields to map to the field.

Type: String

Required: No

DateOptions

Description

Options for a date field. Dates and times are specified in UTC (Coordinated Universal Time) according to IETF RFC3339: yyyy-mm-ddT00:00:00Z. Present if `IndexFieldType` specifies the field is of type `date`. All options are enabled by default.

Contents

DefaultValue
A value to use for the field if the field isn't specified for a document.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

FacetEnabled
Whether facet information can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SearchEnabled
Whether the contents of the field are searchable.

Type: Boolean

Required: No

SortEnabled
Whether the field can be used to sort the search results.

Type: Boolean

Required: No

SourceField
A string that represents the name of an index field. CloudSearch supports regular index fields as well as dynamic fields. A dynamic field's name defines a pattern that begins or ends with a wildcard. Any document fields that don't map to a regular index field but do match a dynamic field's pattern are configured with the dynamic field's indexing options.

Regular field names begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore). Dynamic field names must begin or end with a wildcard (*). The wildcard can also be the only character in a dynamic field name. Multiple wildcards, and wildcards embedded within a string are not supported.

The name `score` is reserved and cannot be used as a field name. To reference a document's ID, you can use the name `_id`.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

DefineAnalysisSchemeResult

Description

The result of a `DefineAnalysisScheme` request. Contains the status of the newly-configured analysis scheme.

Contents

AnalysisScheme
The status and configuration of an `AnalysisScheme`.

Type: AnalysisSchemeStatus (p. 201)

Required: Yes

DefineExpressionResult

Description

The result of a `DefineExpression` request. Contains the status of the newly-configured expression.

Contents

Expression
The value of an `Expression` and its current status.

Type: ExpressionStatus (p. 213)

Required: Yes

DefineIndexFieldResult

Description

The result of a `DefineIndexField` request. Contains the status of the newly-configured index field.

Contents

IndexField
The value of an `IndexField` and its current status.

Type: IndexFieldStatus (p. 216)

Required: Yes

DefineSuggesterResult

Description

The result of a `DefineSuggester` request. Contains the status of the newly-configured suggester.

Contents

Suggester
The value of a `Suggester` and its current status.

Type: SuggesterStatus (p. 224)

Required: Yes

DeleteAnalysisSchemeResult

Description

The result of a `DeleteAnalysisScheme` request. Contains the status of the deleted analysis scheme.

Contents

AnalysisScheme
The status of the analysis scheme being deleted.

Type: AnalysisSchemeStatus (p. 201)

Required: Yes

DeleteDomainResult

Description

The result of a `DeleteDomain` request. Contains the status of a newly deleted domain, or no status if the domain has already been completely deleted.

Contents

DomainStatus
The current status of the search domain.

Type: DomainStatus (p. 210)

Required: No

DeleteExpressionResult

Description

The result of a `DeleteExpression` request. Specifies the expression being deleted.

Contents

Expression
The status of the expression being deleted.

Type: ExpressionStatus (p. 213)

Required: Yes

DeleteIndexFieldResult

Description

The result of a `DeleteIndexField` request.

Contents

IndexField
The status of the index field being deleted.

Type: IndexFieldStatus (p. 216)

Required: Yes

DeleteSuggesterResult

Description

The result of a `DeleteSuggester` request. Contains the status of the deleted suggester.

Contents

Suggester
The status of the suggester being deleted.

Type: SuggesterStatus (p. 224)

Required: Yes

DescribeAnalysisSchemesResult

Description

The result of a `DescribeAnalysisSchemes` request. Contains the analysis schemes configured for the domain specified in the request.

Contents

AnalysisSchemes
The analysis scheme descriptions.

Type: AnalysisSchemeStatus (p. 201) list

Required: Yes

DescribeAvailabilityOptionsResult

Description

The result of a `DescribeAvailabilityOptions` request. Indicates whether or not the Multi-AZ option is enabled for the domain specified in the request.

Contents

AvailabilityOptions
The availability options configured for the domain. Indicates whether Multi-AZ is enabled for the domain.

Type: AvailabilityOptionsStatus (p. 202)

Required: No

DescribeDomainsResult

Description

The result of a `DescribeDomains` request. Contains the status of the domains specified in the request or all domains owned by the account.

Contents

DomainStatusList
A list that contains the status of each requested domain.

Type: DomainStatus (p. 210) list

Required: Yes

DescribeExpressionsResult

Description

The result of a `DescribeExpressions` request. Contains the expressions configured for the domain specified in the request.

Contents

Expressions
The expressions configured for the domain.

Type: ExpressionStatus (p. 213) list

Required: Yes

DescribeIndexFieldsResult

Description

The result of a `DescribeIndexFields` request. Contains the index fields configured for the domain specified in the request.

Contents

IndexFields
The index fields configured for the domain.

Type: IndexFieldStatus (p. 216) list

Required: Yes

DescribeScalingParametersResult

Description

The result of a `DescribeScalingParameters` request. Contains the scaling parameters configured for the domain specified in the request.

Contents

ScalingParameters
The status and configuration of a search domain's scaling parameters.

Type: ScalingParametersStatus (p. 223)

Required: Yes

DescribeServiceAccessPoliciesResult

Description

The result of a `DescribeServiceAccessPolicies` request.

Contents

AccessPolicies
The access rules configured for the domain specified in the request.

Type: AccessPoliciesStatus (p. 199)

Required: Yes

DescribeSuggestersResult

Description

The result of a `DescribeSuggesters` request.

Contents

Suggesters
The suggesters configured for the domain specified in the request.

Type: SuggesterStatus (p. 224) list

Required: Yes

DocumentSuggesterOptions

Description

Options for a search suggester.

Contents

FuzzyMatching
The level of fuzziness allowed when suggesting matches for a string: `none`, `low`, or `high`. With none, the specified string is treated as an exact prefix. With low, suggestions must differ from the specified string by no more than one character. With high, suggestions can differ by up to two characters. The default is none.

Type: String

Valid Values: `none | low | high`

Required: No

SortExpression
An expression that computes a score for each suggestion to control how they are sorted. The scores are rounded to the nearest integer, with a floor of 0 and a ceiling of 2^31-1. A document's relevance score is not computed for suggestions, so sort expressions cannot reference the `_score` value. To sort suggestions using a numeric field or existing expression, simply specify the name of the field or expression. If no expression is configured for the suggester, the suggestions are sorted with the closest matches listed first.

Type: String

Required: No

SourceField
The name of the index field you want to use for suggestions.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

DomainStatus

Description

The current status of the search domain.

Contents

ARN
The Amazon Resource Name (ARN) of the search domain. See Identifiers for IAM Entities in *Using AWS Identity and Access Management* for more information.

Type: String

Required: No

Created
True if the search domain is created. It can take several minutes to initialize a domain when CreateDomain (p. 155) is called. Newly created search domains are returned from DescribeDomains (p. 177) with a false value for Created until domain creation is complete.

Type: Boolean

Required: No

Deleted
True if the search domain has been deleted. The system must clean up resources dedicated to the search domain when DeleteDomain (p. 166) is called. Newly deleted search domains are returned from DescribeDomains (p. 177) with a true value for IsDeleted for several minutes until resource cleanup is complete.

Type: Boolean

Required: No

DocService
The service endpoint for updating documents in a search domain.

Type: ServiceEndpoint (p. 223)

Required: No

DomainId
An internally generated unique identifier for a domain.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

DomainName
A string that represents the name of a domain. Domain names are unique across the domains owned by an account within an AWS region. Domain names start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen).

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Limits
Type: Limits (p. 219)

Required: No

Processing
True if processing is being done to activate the current domain configuration.

Type: Boolean

Required: No

RequiresIndexDocuments
True if IndexDocuments (p. 187) needs to be called to activate the current domain configuration.

Type: Boolean

Required: Yes

SearchInstanceCount
The number of search instances that are available to process search requests.

Type: Integer

Required: No

SearchInstanceType
The instance type that is being used to process search requests.

Type: String

Required: No

SearchPartitionCount
The number of partitions across which the search index is spread.

Type: Integer

Required: No

SearchService
The service endpoint for requesting search results from a search domain.

Type: ServiceEndpoint (p. 223)

Required: No

DoubleArrayOptions

Description

Options for a field that contains an array of double-precision 64-bit floating point values. Present if `IndexFieldType` specifies the field is of type `double-array`. All options are enabled by default.

Contents

DefaultValue
A value to use for the field if the field isn't specified for a document.

Type: Double

Required: No

FacetEnabled
Whether facet information can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SearchEnabled
Whether the contents of the field are searchable.

Type: Boolean

Required: No

SourceFields
A list of source fields to map to the field.

Type: String

Required: No

DoubleOptions

Description

Options for a double-precision 64-bit floating point field. Present if `IndexFieldType` specifies the field is of type `double`. All options are enabled by default.

Contents

DefaultValue
A value to use for the field if the field isn't specified for a document. This can be important if you are using the field in an expression and that field is not present in every document.

Type: Double

Required: No

FacetEnabled
Whether facet information can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SearchEnabled
Whether the contents of the field are searchable.

Type: Boolean

Required: No

SortEnabled
Whether the field can be used to sort the search results.

Type: Boolean

Required: No

SourceField
The name of the source field to map to the field.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

Expression

Description

A named expression that can be evaluated at search time. Can be used to sort the search results, define other expressions, or return computed information in the search results.

Contents

ExpressionName
Names must begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore).

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

ExpressionValue
The expression to evaluate for sorting while processing a search request. The `Expression` syntax is based on JavaScript expressions. For more information, see Configuring Expressions in the *Amazon CloudSearch Developer Guide*.

Type: String

Length constraints: Minimum length of 1. Maximum length of 10240.

Required: Yes

ExpressionStatus

Description

The value of an `Expression` and its current status.

Contents

Options
The expression that is evaluated for sorting while processing a search request.

Type: Expression (p. 213)

Required: Yes

Status
The status of domain configuration option.

Type: OptionStatus (p. 222)

Required: Yes

IndexDocumentsResult

Description

The result of an `IndexDocuments` request. Contains the status of the indexing operation, including the fields being indexed.

Contents

FieldNames
The names of the fields that are currently being indexed.

Type: String list

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

IndexField

Description

Configuration information for a field in the index, including its name, type, and options. The supported options depend on the `IndexFieldType` .

Contents

DateArrayOptions
Options for a field that contains an array of dates. Present if `IndexFieldType` specifies the field is of type `date-array`. All options are enabled by default.

Type: DateArrayOptions (p. 203)

Required: No

DateOptions
Options for a date field. Dates and times are specified in UTC (Coordinated Universal Time) according to IETF RFC3339: yyyy-mm-ddT00:00:00Z. Present if `IndexFieldType` specifies the field is of type `date`. All options are enabled by default.

Type: DateOptions (p. 203)

Required: No

DoubleArrayOptions
Options for a field that contains an array of double-precision 64-bit floating point values. Present if `IndexFieldType` specifies the field is of type `double-array`. All options are enabled by default.

Type: DoubleArrayOptions (p. 211)

Required: No

DoubleOptions
Options for a double-precision 64-bit floating point field. Present if `IndexFieldType` specifies the field is of type `double`. All options are enabled by default.

Type: DoubleOptions (p. 212)

Required: No

IndexFieldName
A string that represents the name of an index field. CloudSearch supports regular index fields as well as dynamic fields. A dynamic field's name defines a pattern that begins or ends with a wildcard. Any document fields that don't map to a regular index field but do match a dynamic field's pattern are configured with the dynamic field's indexing options.

Regular field names begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore). Dynamic field names must begin or end with a wildcard (*). The wildcard can also be the only character in a dynamic field name. Multiple wildcards, and wildcards embedded within a string are not supported.

The name `score` is reserved and cannot be used as a field name. To reference a document's ID, you can use the name `_id`.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

IndexFieldType
The type of field. The valid options for a field depend on the field type. For more information about the supported field types, see Configuring Index Fields in the *Amazon CloudSearch Developer Guide*.

Type: String

Valid Values: `int | double | literal | text | date | latlon | int-array | double-array | literal-array | text-array | date-array`

Required: Yes

IntArrayOptions
Options for a field that contains an array of 64-bit signed integers. Present if `IndexFieldType` specifies the field is of type `int-array`. All options are enabled by default.

Type: IntArrayOptions (p. 217)

Required: No

IntOptions
Options for a 64-bit signed integer field. Present if `IndexFieldType` specifies the field is of type `int`. All options are enabled by default.

Type: IntOptions (p. 217)

Required: No

LatLonOptions
Options for a latlon field. A latlon field contains a location stored as a latitude and longitude value pair. Present if `IndexFieldType` specifies the field is of type `latlon`. All options are enabled by default.

Type: LatLonOptions (p. 218)

Required: No

LiteralArrayOptions
Options for a field that contains an array of literal strings. Present if `IndexFieldType` specifies the field is of type `literal-array`. All options are enabled by default.

Type: LiteralArrayOptions (p. 220)

Required: No

LiteralOptions
Options for literal field. Present if `IndexFieldType` specifies the field is of type `literal`. All options are enabled by default.

Type: LiteralOptions (p. 221)

Required: No

TextArrayOptions
Options for a field that contains an array of text strings. Present if `IndexFieldType` specifies the field is of type `text-array`. A `text-array` field is always searchable. All options are enabled by default.

Type: TextArrayOptions (p. 225)

Required: No

TextOptions
Options for text field. Present if `IndexFieldType` specifies the field is of type `text`. A `text` field is always searchable. All options are enabled by default.

Type: TextOptions (p. 226)

Required: No

IndexFieldStatus

Description

The value of an `IndexField` and its current status.

Contents

Options
Configuration information for a field in the index, including its name, type, and options. The supported options depend on the `IndexFieldType` .

Type: IndexField (p. 214)

Required: Yes

Status
The status of domain configuration option.

Type: OptionStatus (p. 222)

Required: Yes

IntArrayOptions

Description

Options for a field that contains an array of 64-bit signed integers. Present if `IndexFieldType` specifies the field is of type `int-array`. All options are enabled by default.

Contents

DefaultValue
A value to use for the field if the field isn't specified for a document.

Type: Long

Required: No

FacetEnabled
Whether facet information can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SearchEnabled
Whether the contents of the field are searchable.

Type: Boolean

Required: No

SourceFields
A list of source fields to map to the field.

Type: String

Required: No

IntOptions

Description

Options for a 64-bit signed integer field. Present if `IndexFieldType` specifies the field is of type `int`. All options are enabled by default.

Contents

DefaultValue
A value to use for the field if the field isn't specified for a document. This can be important if you are using the field in an expression and that field is not present in every document.

Type: Long

Required: No

FacetEnabled
Whether facet information can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SearchEnabled
Whether the contents of the field are searchable.

Type: Boolean

Required: No

SortEnabled
Whether the field can be used to sort the search results.

Type: Boolean

Required: No

SourceField
The name of the source field to map to the field.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

LatLonOptions

Description

Options for a latlon field. A latlon field contains a location stored as a latitude and longitude value pair. Present if `IndexFieldType` specifies the field is of type `latlon`. All options are enabled by default.

Contents

DefaultValue
A value to use for the field if the field isn't specified for a document.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

FacetEnabled
Whether facet information can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SearchEnabled
Whether the contents of the field are searchable.

Type: Boolean

Required: No

SortEnabled
Whether the field can be used to sort the search results.

Type: Boolean

Required: No

SourceField
A string that represents the name of an index field. CloudSearch supports regular index fields as well as dynamic fields. A dynamic field's name defines a pattern that begins or ends with a wildcard. Any document fields that don't map to a regular index field but do match a dynamic field's pattern are configured with the dynamic field's indexing options.

Regular field names begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore). Dynamic field names must begin or end with a wildcard (*). The wildcard can also be the only character in a dynamic field name. Multiple wildcards, and wildcards embedded within a string are not supported.

The name `score` is reserved and cannot be used as a field name. To reference a document's ID, you can use the name `_id`.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

Limits

Description
No action documentation available.

Contents

MaximumPartitionCount
Type: Integer

Required: Yes

MaximumReplicationCount
Type: Integer

Required: Yes

ListDomainNamesResult

Description

The result of a `ListDomainNames` request. Contains a list of the domains owned by an account.

Contents

DomainNames
The names of the search domains owned by an account.

Type: String to String map

Required: No

LiteralArrayOptions

Description

Options for a field that contains an array of literal strings. Present if `IndexFieldType` specifies the field is of type `literal-array`. All options are enabled by default.

Contents

DefaultValue
A value to use for the field if the field isn't specified for a document.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

FacetEnabled
Whether facet information can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SearchEnabled
Whether the contents of the field are searchable.

Type: Boolean

Required: No

SourceFields
A list of source fields to map to the field.

Type: String

Required: No

LiteralOptions

Description

Options for literal field. Present if `IndexFieldType` specifies the field is of type `literal`. All options are enabled by default.

Contents

DefaultValue
A value to use for the field if the field isn't specified for a document.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

FacetEnabled
Whether facet information can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SearchEnabled
Whether the contents of the field are searchable.

Type: Boolean

Required: No

SortEnabled
Whether the field can be used to sort the search results.

Type: Boolean

Required: No

SourceField
A string that represents the name of an index field. CloudSearch supports regular index fields as well as dynamic fields. A dynamic field's name defines a pattern that begins or ends with a wildcard. Any document fields that don't map to a regular index field but do match a dynamic field's pattern are configured with the dynamic field's indexing options.

Regular field names begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore). Dynamic field names must begin or end with a wildcard (*). The wildcard can also be the only character in a dynamic field name. Multiple wildcards, and wildcards embedded within a string are not supported.

The name `score` is reserved and cannot be used as a field name. To reference a document's ID, you can use the name `_id`.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

OptionStatus

Description

The status of domain configuration option.

Contents

CreationDate
A timestamp for when this option was created.

Type: DateTime

Required: Yes

PendingDeletion
Indicates that the option will be deleted once processing is complete.

Type: Boolean

Required: No

State
The state of processing a change to an option. Possible values:

- `RequiresIndexDocuments`: the option's latest value will not be deployed until IndexDocuments (p. 187) has been called and indexing is complete.
- `Processing`: the option's latest value is in the process of being activated.
- `Active`: the option's latest value is completely deployed.
- `FailedToValidate`: the option value is not compatible with the domain's data and cannot be used to index the data. You must either modify the option value or update or remove the incompatible documents.

Type: String

Valid Values: `RequiresIndexDocuments | Processing | Active | FailedToValidate`

Required: Yes

UpdateDate
A timestamp for when this option was last updated.

Type: DateTime

Required: Yes

UpdateVersion
A unique integer that indicates when this option was last updated.

Type: Integer

Required: No

ScalingParameters

Description

The desired instance type and desired number of replicas of each index partition.

Contents

DesiredInstanceType
The instance type that you want to preconfigure for your domain. For example, `search.m1.small`.

Type: String

Valid Values: `search.m1.small | search.m3.medium | search.m3.large | search.m3.xlarge | search.m3.2xlarge`

> **Note**
> For domains created prior to February, 2015, valid values may also include `search.m1.large`, `search.m2.xlarge`, and `search.m2.2xlarge`.

Required: No

DesiredPartitionCount
The number of partitions you want to preconfigure for your domain. Only valid when you select `m3.2xlarge` as the instance type.

Type: Integer

Required: No

DesiredReplicationCount
The number of replicas you want to preconfigure for each index partition.

Type: Integer

Required: No

ScalingParametersStatus

Description

The status and configuration of a search domain's scaling parameters.

Contents

Options
The desired instance type and desired number of replicas of each index partition.

Type: ScalingParameters (p. 222)

Required: Yes

Status
The status of domain configuration option.

Type: OptionStatus (p. 222)

Required: Yes

ServiceEndpoint

Description

The endpoint to which service requests can be submitted.

Contents

Endpoint
The endpoint to which service requests can be submitted. For example, `search-imdb-movies-oopcnjfn6ugofer3zx5iadxxca.eu-west-1.cloudsearch.amazonaws.com` or `doc-imdb-movies-oopcnjfn6ugofer3zx5iadxxca.eu-west-1.cloudsearch.amazonaws.com`.

Type: String

Required: No

Suggester

Description

Configuration information for a search suggester. Each suggester has a unique name and specifies the text field you want to use for suggestions. The following options can be configured for a suggester: `FuzzyMatching`, `SortExpression`.

Contents

DocumentSuggesterOptions
Options for a search suggester.

Type: DocumentSuggesterOptions (p. 209)

Required: Yes

SuggesterName
Names must begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore).

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

SuggesterStatus

Description

The value of a `Suggester` and its current status.

Contents

Options
Configuration information for a search suggester. Each suggester has a unique name and specifies the text field you want to use for suggestions. The following options can be configured for a suggester: `FuzzyMatching`, `SortExpression`.

Type: Suggester (p. 224)

Required: Yes

Status
The status of domain configuration option.

Type: OptionStatus (p. 222)

Required: Yes

TagList

Description

A list of resource tags for the specified Amazon CloudSearch domain. The list is a result of a `ListTags` request.

Contents

`TagList`

List of resource tags for the specified Amazon CloudSearch domain.

Type: a list of strings that specify the resource tags for the domain.

TextArrayOptions

Description

Options for a field that contains an array of text strings. Present if `IndexFieldType` specifies the field is of type `text-array`. A `text-array` field is always searchable. All options are enabled by default.

Contents

AnalysisScheme
The name of an analysis scheme for a `text-array` field.

Type: String

Required: No

DefaultValue
A value to use for the field if the field isn't specified for a document.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

HighlightEnabled
Whether highlights can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SourceFields
A list of source fields to map to the field.

Type: String

Required: No

TextOptions

Description

Options for text field. Present if `IndexFieldType` specifies the field is of type `text`. A `text` field is always searchable. All options are enabled by default.

Contents

AnalysisScheme
The name of an analysis scheme for a `text` field.

Type: String

Required: No

DefaultValue
A value to use for the field if the field isn't specified for a document.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

HighlightEnabled
Whether highlights can be returned for the field.

Type: Boolean

Required: No

ReturnEnabled
Whether the contents of the field can be returned in the search results.

Type: Boolean

Required: No

SortEnabled
Whether the field can be used to sort the search results.

Type: Boolean

Required: No

SourceField
A string that represents the name of an index field. CloudSearch supports regular index fields as well as dynamic fields. A dynamic field's name defines a pattern that begins or ends with a wildcard. Any document fields that don't map to a regular index field but do match a dynamic field's pattern are configured with the dynamic field's indexing options.

Regular field names begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore). Dynamic field names must begin or end with a wildcard (*). The wildcard can also be the only character in a dynamic field name. Multiple wildcards, and wildcards embedded within a string are not supported.

The name `score` is reserved and cannot be used as a field name. To reference a document's ID, you can use the name `_id`.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

UpdateAvailabilityOptionsResult

Description

The result of a `UpdateAvailabilityOptions` request. Contains the status of the domain's availability options.

Contents

AvailabilityOptions
The newly-configured availability options. Indicates whether Multi-AZ is enabled for the domain.

Type: AvailabilityOptionsStatus (p. 202)

Required: No

UpdateScalingParametersResult

Description

The result of a `UpdateScalingParameters` request. Contains the status of the newly-configured scaling parameters.

Contents

ScalingParameters
The status and configuration of a search domain's scaling parameters.

Type: ScalingParametersStatus (p. 223)

Required: Yes

UpdateServiceAccessPoliciesResult

Description

The result of an `UpdateServiceAccessPolicies` request. Contains the new access policies.

Contents

AccessPolicies
The access rules configured for the domain.

Type: AccessPoliciesStatus (p. 199)

Required: Yes

Common Parameters

This section lists the request parameters that all actions use. Any action-specific parameters are listed in the topic for the action.

Action
The action to be performed.

Default: None

Type: string

Required: Yes

AuthParams

The parameters that are required to authenticate a Conditional request. Contains:

- AWSAccessKeyID
- SignatureVersion
- Timestamp
- Signature

Default: None

Required: Conditional

AWSAccessKeyId

The access key ID that corresponds to the secret access key that you used to sign the request.

Default: None

Type: string

Required: Yes

Expires

The date and time when the request signature expires, expressed in the format YYYY-MM-DDThh:mm:ssZ, as specified in the ISO 8601 standard.

Condition: Requests must include either *Timestamp* or *Expires*, but not both.

Default: None

Type: string

Required: Conditional

SecurityToken

The temporary security token that was obtained through a call to AWS Security Token Service. For a list of services that support AWS Security Token Service, go to Using Temporary Security Credentials to Access AWS in **Using Temporary Security Credentials**.

Default: None

Type: string

Required: No

Signature

The digital signature that you created for the request. For information about generating a signature, go to the service's developer documentation.

Default: None

Type: string

Required: Yes

SignatureMethod

The hash algorithm that you used to create the request signature.

Default: None

Type: string

Valid Values: `HmacSHA256 | HmacSHA1`

Required: Yes

SignatureVersion
The signature version you use to sign the request. Set this to the value that is recommended for your service.

Default: None

Type: string

Required: Yes

Timestamp
The date and time when the request was signed, expressed in the format YYYY-MM-DDThh:mm:ssZ, as specified in the ISO 8601 standard.

Condition: Requests must include either *Timestamp* or *Expires*, but not both.

Default: None

Type: string

Required: Conditional

Version
The API version that the request is written for, expressed in the format YYYY-MM-DD.

Default: None

Type: string

Required: Yes

Common Errors

This section lists the common errors that all actions return. Any action-specific errors are listed in the topic for the action.

IncompleteSignature
The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure
The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

InvalidAction
The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientTokenId
The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

InvalidParameterCombination
Parameters that must not be used together were used together.

HTTP Status Code: 400

InvalidParameterValue
An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

InvalidQueryParameter
The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

MalformedQueryString
The query string contains a syntax error.

HTTP Status Code: 404

MissingAction
The request is missing an action or a required parameter.

HTTP Status Code: 400

MissingAuthenticationToken
The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

MissingParameter
A required parameter for the specified action is not supplied.

HTTP Status Code: 400

OptInRequired
The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired
The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable
The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

Throttling
The request was denied due to request throttling.

HTTP Status Code: 400

ValidationError
The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

Document Service API Reference for Amazon CloudSearch

You use the document service API to add, replace, or delete documents in your Amazon CloudSearch domain. For more information managing the documents in your search domain, see Uploading Data to an Amazon CloudSearch Domain (p. 87).

The other APIs you use to interact with Amazon CloudSearch are:

- Configuration API Reference for Amazon CloudSearch (p. 148)—Set up and manage your search domain.
- Search API Reference (p. 239)—Search your domain.

Submitting Document Service Requests in Amazon CloudSearch

We recommend using one of the AWS SDKs or the AWS CLI to submit document upload requests. The SDKs and AWS CLI handle request signing for you and provide an easy way to perform all Amazon CloudSearch actions. To process source documents and upload the generated JSON or XML batches to your domain in one step, you can use the `cs-import-documents` command in the standalone Amazon CloudSearch command line tools. For more information, see Processing Your Source Data (p. 62). You can also use the Amazon CloudSearch console to upload individual batches and import data from DynamoDB or S3.

> **Important**
> A domain's document and search endpoints remain the same for the life of the domain. You should cache the endpoints rather than retrieving them before every upload or search request. Querying the Amazon CloudSearch configuration service by calling `aws cloudsearch describe-domains` or `DescribeDomains` before every request is likely to result in your requests being throttled.

For example, the following request uploads a batch using the AWS CLI.

```
aws cloudsearchdomain --endpoint-url http://doc-movies-
y6gelr4lv3jeu4rvoelunxsl2e.us-east-1.cloudsearch.amazonaws.com upload-
documents --content-type
 application/json --documents movie-data-2013.json
```

For development and testing purposes, you can allow anonymous access to your domain's document service and submit unsigned HTTP POST requests directly to your domain's document service. In a production environment, restrict access to your domain to specific IAM users, groups, or roles and submit signed requests. For information about controlling access for Amazon CloudSearch, see Configuring Access for Amazon CloudSearch (p. 29). For more information about request signing, see Signing AWS API Requests.

For example, the following POST request uploads a batch of documents formatted in JSON to the domain endpoint doc-movies-123456789012.us-east-1.cloudsearch.amazonaws.com.

```
curl -X POST --upload-file data1.json doc-movies-123456789012.us-
east-1.cloudsearch.amazonaws.com/2013-01-01/documents/batch --header
 "Content-Type: application/json"
```

documents/batch

This section describes the HTTP request and response messages for the `documents/batch` resource.

You create document batches to describe the data that you want to upload to an Amazon CloudSearch domain. A document batch is a collection of add and delete operations that represent the documents you want to add, update, or delete from your domain. Batches can be described in either JSON or XML. A batch provides all of the information Amazon CloudSearch needs for indexing. Each item that

you want to be able to return as a search result (such as a product) is represented as a document—a batch is simply a collection of add and delete requests for individual documents. Every document has a unique ID and one or more fields that contain the data that you want to search and return in results.

To update a document, you specify an add request with the document ID of the document you want to update. For more information, see Adding and Updating Documents in Amazon CloudSearch (p. 61). Similarly, to delete a document, you submit a delete request with the document ID of the document you want to delete. For information about deleting documents, see Deleting Documents in Amazon CloudSearch (p. 61).

For more information about submitting data for indexing, see Uploading Data to an Amazon CloudSearch Domain (p. 87).

documents/batch JSON API

JSON documents/batch Requests

The body of a `documents/batch` request uses JSON or XML to specify the document operations you want to perform. A JSON representation of a batch is a collection of objects that define individual add and delete operations. The `type` property identifies whether an object represents an add or delete operation. For example, the following JSON batch adds one document and deletes one document:

```
[
{ "type": "add",
  "id":   "tt0484562",
  "fields": {
    "title": "The Seeker: The Dark Is Rising",
    "directors": ["Cunningham, David L."],
    "genres": ["Adventure","Drama","Fantasy","Thriller"],
    "actors": ["McShane, Ian","Eccleston, Christopher","Conroy, Frances",
               "Crewson, Wendy","Ludwig, Alexander","Cosmo, James",
               "Warner, Amelia","Hickey, John Benjamin","Piddock, Jim",
               "Lockhart, Emma"]
  }
},
{ "type": "delete",
  "id":   "tt0484575"
}]
```

Note
When specifying document batches in JSON, the value for a field cannot be `null`.

The JSON schema representation of a batch is shown below:

```
{
    "type": "array",
    "minItems": 1,
    "items": {
        "type": "object",
        "properties": {
            "type": {
                "type": "string",
                "enum": ["add", "delete"],
                "required": true
            },
            "id": {
                "type": "string",
                "pattern": "[a-z0-9][a-z0-9_]{0,127}",
```

```
                "minLength": 1,
                "maxLength": 128,
                "required": true
            },
            "fields": {
                "type": "object",
                "patternProperties": {
                    "[a-zA-Z0-9][a-zA-Z0-9_]{0,63}": {
                        "type": "string",
                    }
                }
            }
        }
    }
}
```

documents/batch Request Properties (JSON)

Property	Description	Required
type	The operation type, `add` or `delete`.	Yes
id	An alphanumeric string. Allowed characters are: A-Z (upper-case letters), -a-z (lower-case letters), 0-9, _ (underscore), - (hyphen), / (forward slash), # (hash sign), : (colon). The max length is 128 characters.	Yes
fields	A collection of one or more *field_name* properties that define the fields the document contains. Condition: Required for add operations. Must contain at least one *field_name* property.	Conditional
field_name	Specifies a field within the document being added. Field names must begin with a letter and can contain the following characters: a-z (lower case), 0-9, and _ (underscore). Field names must be at least 3 and no more than 64 characters. The name *score* is reserved and cannot be used as a field name. To specify multiple values for a field, you specify an array of values instead of a single value. For example: `"genre": ["Adventure","Drama","Fantasy","Thriller"]` Condition: At least one field must be specified in the fields object.	Conditional

documents/batch Response (JSON)

The response body lists the number of adds and deletes that were performed and any errors or warnings that were generated.

The JSON schema representation of a document service API response is shown below:

```
{
    "type": "object",
    "properties": {
        "status": {
            "type": "text",
            "enum": ["success", "error"],
            "required": true
        },
        "adds": {
            "type": "integer",
            "minimum": 0,
            "required": true
        },
        "deletes": {
            "type": "integer",
            "minimum": 0,
            "required": true
        },
        "errors": {
            "type": "array",
            "required": false,
            "items": {
                "type": "object",
                "properties": {
                    "message": {
                        "type": "string",
                        "required": true
                    }
                }
            }
        },
        "warnings": {
            "type": "array",
            "required": false,
            "items": {
                "type": "object",
                "properties": {
                    "message": {
                        "type": "string",
                        "required": true
                    }
                }
            }
        }
    }
}
```

documents/batch Response Properties (JSON)

Property	Description
status	The result status, which is either `success` or `error`.
adds	The number of add document operations that were performed. Always zero when the status is `error`.
deletes	The number of delete document operations that were performed. Always zero when the status is `error`.

Property	Description
errors	Provides information about a parsing or validation error. Specified only if the status is `error`.
warning	Provides information about a warning generated during parsing or validation.

documents/batch XML API

XML documents/batch Requests

The body of a `documents/batch` request specifies the document operations you want to perform in XML. For example:

```
<batch>
 <add  id="tt0484562">
  <field name="title">The Seeker: The Dark Is Rising</field>
  <field name="director">Cunningham, David L.</field>
  <field name="genre">Adventure</field>
  <field name="genre">Drama</field>
  <field name="genre">Fantasy</field>
  <field name="genre">Thriller</field>
  <field name="actor">McShane, Ian</field>
  <field name="actor">Eccleston, Christopher</field>
  <field name="actor">Conroy, Frances</field>
  <field name="actor">Ludwig, Alexander</field>
  <field name="actor">Crewson, Wendy</field>
  <field name="actor">Warner, Amelia</field>
  <field name="actor">Cosmo, James</field>
  <field name="actor">Hickey, John Benjamin</field>
  <field name="actor">Piddock, Jim</field>
  <field name="actor">Lockhart, Emma</field>
 </add>
 <delete id="tt0301199" />
</batch>
```

documents/batch Request Elements (XML)

Element	Description	Required
batch	The collection of add or delete operations that you want to submit to your search domain. A batch must contain at least one add or delete element.	Yes
add	Specifies a document that you want to add to your search domain. The id attributes is required and an add element must contain at least one field. Attributes: • `id`—An alphanumeric string. Any characters other than A-Z (upper or lower case) and 0-9 are illegal. The max length is 128 characters.	No

Element	Description	Required

field	Specifies a field in the document being added. The name attribute and a field value are required. Field names must begin with a letter and can contain the following characters: a-z (lower case), 0-9, and _ (underscore). The name *score* is reserved and cannot be used as a field name. The field value can be text or CDATA.
	
	To specify multiple values for a field, you include multiple field elements with the same name. For example:
	
	`<field name="genre">Adventure</field>`
	`<field name="genre">Drama</field>`
	`<field name="genre">Fantasy</field>`
	`<field name="genre">Thriller</field>`
	
	Constraints:
	
	• `name`—An alphanumeric string that begins with a letter. Can contain a-z (lower case), 0-9, _ (underscore), - (hyphen), and . (period).
	
	Condition: At least one field must be specified in an add element.
		Conditional

delete	Specifies a document that you want to remove from your search domain. The id attribute is required. A delete element must be empty. Constraints: • `id`—An alphanumeric string. Any characters other than A-Z (upper or lower case) and 0-9 are illegal.	No

documents/batch Response (XML)

The response body lists the number of adds and deletes that were performed and any errors or warnings that were generated.

The RelaxNG schema of a document service API response is:

```
 start = response

response = element response {
    attribute status { "success" | "error" },
    attribute adds { xsd:integer },
    attribute deletes { xsd:integer },
    element errors {
        element error {
            text
        }+
    }? &
```

```
    element warnings {
        element warning {
            text
        }+
    }?
}
```

documents/batch Response Elements (XML)

Element	Description
result	Contains elements that list the errors and warnings generated when parsing and validating the request. Attributes: • `status`—The result status, which is either `success` or `error`. • `adds`—The number of added documents. If the status is `error`, this is always zero. • `deletes`—The number of deleted documents. If the status is `error`, this is always zero. Constraints: If the status is `error`, the results element contains a list of errors. If the status is `success`, the results element can contain a list of warnings, but no errors.
errors	Contains a collection of error elements that identify the errors that occurred when parsing and validating the request.
error	Provides information about a parsing or validation error. The value provides a description of the error.
warnings	Contains a collection of warning elements that identify the warnings that were generated when parsing and validating the request.
warning	Provides information about a parsing or validation warning. The value provides a description of the error.

documents/batch Status Codes

A document service request can return three types of status codes:

- 5xx status codes indicate that there was an internal server error. We recommend catching and retrying all 5xx error codes as they typically represent transient error conditions.
- 4xx status codes indicate that the request was malformed.
- 2xx status codes indicate that the request was processed successfully.

Error	Description	HTTP Status Code
No Content-Type	The Content-Type header is missing.	400
No Content-Length	The Content-Length header is missing.	411
Incorrect Path	URL path does not match "/YYYY-MM-DD/documents/batch".	404

Error	Description	HTTP Status Code
Invalid HTTP Method	The HTTP method is not POST. Requests must be posted to documents/batch.	405
Invalid Accept Type	Accept header specifies a content type other than "application/xml" or "application/json". Responses can be sent only as XML or JSON.	406
Request Too Large	The length of the request body is larger than the maximum allowed value.	413
Invalid Content Type	The content type is something other than "application/json" or "application/xml".	415
Invalid Character Set	The character set is something other than "ASCII", "ISO-8859-1", or "UTF-8".	415

Common Request Headers

Name	Description	Required
Content-Type	A standard MIME type describing the format of the object data. For more information, see W3C RFC 2616 Section 14. Default: application/json Constraints: application/json or application/xml only	Required
Content-Length	The length in bytes of the body of the request.	Yes
Accept	A standard MIME type describing the format of the response data. For more information, see W3C RFC 2616 Section 14. Default: the content-type of the request Constraints: application/json or application/xml only	No

Common Response Headers

Name	Description
Content-Type	A standard MIME type describing the format of the object data. For more information, see W3C RFC 2616 Section 14. Default: the value of the Accept header in the request, or the Content-Type of the request if the Accept header is missing or doesn't specify either application/xml or application/json. Constraints: application/xml or application/json only
Content-Length	The length in bytes of the body in the response.

Search API Reference for Amazon CloudSearch

Topics

You use the Search API to submit search or suggestion requests to your Amazon CloudSearch domain. For more information about searching, see Searching Your Data with Amazon CloudSearch (p. 94). For more information about suggestions, see Getting Autocomplete Suggestions in Amazon CloudSearch (p. 123).

The other APIs you use to interact with Amazon CloudSearch are:

- Configuration API (p. 148)—Set up and manage your search domain.
- Document Service API (p. 230)—Submit the data you want to search.

Submitting Search Requests in Amazon CloudSearch

We recommend using one of the AWS SDKs or the AWS CLI to submit search requests. The SDKs and AWS CLI handle request signing for you and provide an easy way to perform all Amazon CloudSearch actions. You can also use the Search Tester in the Amazon CloudSearch console to search your data, browse the results, and view the generated request URLs and JSON and XML responses. For more information, see Searching with the Search Tester (p. 12).

> **Important**
> A domain's document and search endpoints remain the same for the life of the domain. You should cache the endpoints rather than retrieving them before every upload or search request. Querying the Amazon CloudSearch configuration service by calling `aws cloudsearch describe-domains` or `DescribeDomains` before every request is likely to result in your requests being throttled. Note that your domain's IP address **CAN** change over time, so it's important to cache the endpoint as shown in the console and returned by the `aws cloudsearch describe-domains` command rather than the IP address. For more information, see Setting the JVM TTL for DNS Name Lookups.

For example, the following request submits a simple text search for `wolverine` using the AWS CLI and returns just the IDs of the matching documents.

```
aws cloudsearchdomain --endpoint-url http://search-movies-
y6gelr4lv3jeu4rvoelunxsl2e.us-east-1.cloudsearch.amazonaws.com search --
search-query wolverine  --return _no_fields
{
    "status": {
        "rid": "/rnE+e4oCAqfEEs=",
        "time-ms": 6
    },
    "hits": {
        "found": 3,
        "hit": [
            {
```

```
                "id": "tt1430132"
            },
            {
                "id": "tt0458525"
            },
            {
                "id": "tt1877832"
            }
        ],
        "start": 0
    }
}
```

By default, Amazon CloudSearch returns the response in JSON. You can get the results formatted in XML by specifying the `format` parameter. Setting the response format only affects responses to successful requests. The format of an error response depends on the origin of the error. Errors returned by the search service are always returned in JSON. 5xx errors due to server timeouts and other request routing problems are returned in XML.

For development and testing purposes, you can allow anonymous access to your domain's search service and submit unsigned HTTP GET or POST requests directly to your domain's search endpoint. In a production environment, restrict access to your domain to specific IAM users, groups, or roles and submit signed requests using the AWS SDKs or AWS CLI. For information about controlling access for Amazon CloudSearch, see Configuring Access for Amazon CloudSearch (p. 29). For more information about request signing, see Signing AWS API Requests.

You can use any method you want to send HTTP requests directly to your domain's search endpoint—you can enter the request URL directly in a Web browser, use cURL to submit the request, or generate an HTTP call using your favorite HTTP library. To specify your search criteria, you specify a query string that specifies the constraints for your search and what you want to get back in the response. The query string must be URL-encoded. The maximum size of a search request submitted via GET is 8190 bytes, including the HTTP method, URI, and protocol version. You can submit larger requests using HTTP POST; however, keep in mind that large, complex requests take longer to process and are more likely to time out. For more information, see Tuning Search Request Performance in Amazon CloudSearch (p. 111).

For example, the following request submits a structured query to the `search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.amazonaws.com` domain and gets the contents of the `title` field.

```
http://search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.
amazonaws.com/2013-01-01/search?q=(and+(term+field%3Dtitle+'star')
(term+field%3Dyear+1977))&q.parser=structured&return=title
```

Important
Special characters in the query string must be URL-encoded. For example, you must encode the = operator in a structured query as `%3D`: `(term+field%3Dtitle+'star')`. If you don't encode the special characters when you submit the search request, you'll get an `InvalidQueryString` error.

Search

This section describes the HTTP request and response messages for the search resource.

Search Syntax

```
GET /2013-01-01/search
```

Search Request Headers

HOST
: The search request endpoint for the domain you're querying. You can use DescribeDomains (p. 177) to retrieve your domain's search request endpoint.

  Required: Yes

Search Request Parameters

cursor
: Retrieves a cursor value you can use to page through large result sets. Use the `size` parameter to control the number of hits you want to include in each response. You can specify either the `cursor` or `start` parameter in a request, they are mutually exclusive. For more information, see Paginating Results (p. 135).

  To get the first cursor, specify `cursor=initial` in your initial request. In subsequent requests, specify the cursor value returned in the hits section of the response.

  For example, the following request sets the cursor value to `initial` and the `size` parameter to 100 to get the first set of hits. The cursor for the next set of hits is included in the response.

```
search?q=john&cursor=initial&size=100&return=_no_fields
{
   "status": {
      "rid": "+/Xu5s0oHwojC6o=",
      "time-ms": 15
   },
   "hits": {
      "found": 503,
      "start": 0,
      cursor": "VegKzpYYQW9JSVFFRU1UeWwwZERBd09EUTNPRGM9ZA",
      "hit": [
         {"id": "tt0120601"},
         {"id": "tt1801552"},
         ...
      ]
   }
}
```

  To get the next set of hits, you specify the cursor value and the number of hits to retrieve.

```
search?q=john&cursor=VegKzpYYQW9JSVFFRU1UeWwwZERBd09EUTNPRGM9ZA&size=100
```

  Type: String

  Required: No

expr.NAME
: Defines an expression that can be used to sort results. You can also specify an expression as a return field. For more information about defining and using expressions, see Configuring Expressions (p. 129).

  You can define and use multiple expressions in a search request. For example, the following request creates two expressions that are used to sort the results and includes them in the search results:

```
search?q=(and (term field=genres 'Sci-Fi')(term field=genres
 'Comedy'))&q.parser=structured
&expr.expression1=_score*rating
&expr.expression2=(1/rank)*year
&sort=expression1 desc,expression2 desc
&return=title,rating,rank,year,_score,expression1,expression2
```

Type: String

Required: No

facet.FIELD

Specifies a field that you want to get facet information for—`FIELD` is the name of the field. The specified field must be facet enabled in the domain configuration. Facet options are specified as a JSON object. If the JSON object is empty, `facet.FIELD={}`, facet counts are computed for all field values, the facets are sorted by facet count, and the top 10 facets are returned in the results.

You can specify three options in the JSON object:

- `sort` specifies how you want to sort the facets in the results: `bucket` or `count`. Specify `bucket` to sort alphabetically or numerically by facet value (in ascending order). Specify `count` to sort by the facet counts computed for each facet value (in descending order). To retrieve facet counts for particular values or ranges of values, use the `buckets` option instead of `sort`.
- `buckets` specifies an array of the facet values or ranges you want to count. Buckets are returned in the order they are specified in the request. To specify a range of values, use a comma (,) to separate the upper and lower bounds and enclose the range using brackets or braces. A square bracket, [or], indicates that the bound is included in the range, a curly brace, { or }, excludes the bound. You can omit the upper or lower bound to specify an open-ended range. When omitting a bound, you must use a curly brace. The `sort` and `size` options are not valid if you specify `buckets`.
- `size` specifies the maximum number of facets to include in the results. By default, Amazon CloudSearch returns counts for the top 10. The `size` parameter is only valid when you specify the `sort` option; it cannot be used in conjunction with `buckets`.

For example, the following request gets facet counts for the `year` field, sorts the facet counts by value and returns counts for the top three:

```
facet.year={sort:"bucket", size:3}
```

To specify which values or range of values you want to calculate facet counts for, use the `buckets` option. For example, the following request calculates and returns the facet counts by decade:

```
facet.year={buckets:["[1970,1979]","[1980,1989]",
             "[1990,1999]","[2000,2009]",
             "[2010,}"]}
```

You can also specify individual values as buckets:

```
facet.genres={buckets:["Action","Adventure","Sci-Fi"]}
```

Note that the facet values are case-sensitive—with the sample IMDB movie data, if you specify `["action","adventure","sci-fi"]` instead of `["Action","Adventure","Sci-Fi"]`, all facet counts are zero.

Type: String

Required: No

format
: Specifies the content type of the response.

  Type: String

  Valid Values: json|xml

  Default: json

  Required: No

fq
: Specifies a structured query that filters the results of a search without affecting how the results are scored and sorted. You use `fq` in conjunction with the `q` parameter to filter the documents that match the constraints specified in the `q` parameter. Specifying a filter just controls which matching documents are included in the results, it has no effect on how they are scored and sorted. The `fq` parameter supports the full structured query syntax. For more information about using filters, see Filtering Matching Documents (p. 111). For more information about structured queries, see Structured Search Syntax (p. 247).

  Type: String

  Required: No

highlight.FIELD
: Retrieves highlights for matches in the specified `text` or `text-array` field. Highlight options are specified as a JSON object. If the JSON object is empty, the returned field text is treated as HTML and the first match is highlighted with emphasis tags: `<em>search-term</em>`.

  You can specify four options in the JSON object:

  - `format`—specifies the format of the data in the text field: `text` or `html`. When data is returned as HTML, all non-alphanumeric characters are encoded. The default is `html`.
  - `max_phrases`—specifies the maximum number of occurrences of the search term(s) you want to highlight. By default, the first occurrence is highlighted.
  - `pre_tag`—specifies the string to prepend to an occurrence of a search term. The default for HTML highlights is `<em>`. The default for text highlights is `*`.
  - `post_tag`—specifies the string to append to an occurrence of a search term. The default for HTML highlights is `</em>`. The default for text highlights is `*`.

  Examples: `highlight.plot={}`,
  `highlight.plot={format:'text',max_phrases:2,pre_tag:'<b>',post_tag:'</b>'}`

  Type: String

  Required: No

partial
: Controls whether partial results are returned if one or more index partitions are unavailable. When your search index is partitioned across multiple search instances, by default Amazon CloudSearch only returns results if every partition can be queried. This means that the failure of a single search instance can result in 5xx (internal server) errors. When you specify `partial=true`. Amazon CloudSearch returns whatever results are available and includes the percentage of documents searched in the search results (`percent-searched`). This enables you to more gracefully degrade your users' search experience. For example, rather than displaying no results, you could display the partial results and a message indicating that the results might be incomplete due to a temporary system outage.

  Type: Boolean

  Default: False

Required: No

pretty
: Formats JSON output so it's easier to read.

  Type: Boolean

  Default: False

  Required: No

q
: The search criteria for the request. How you specify the search criteria depends on the query parser used for the request and the parser options specified in the `q.options` parameter. By default, the `simple` query parser is used to process requests. To use the `structured`, `lucene`, or `dismax` query parser, you must also specify the `q.parser` parameter. For more information about specifying search criteria, see Searching Your Data with Amazon CloudSearch (p. 94).

  Type: String

  Required: Yes

q.options
: Configure options for the query parser specified in the `q.parser` parameter. The options are specified as a JSON object, for example: `q.options={defaultOperator: 'or', fields: ['title^5','description']}`.

  The options you can configure vary according to which parser you use:

  - `defaultOperator`—The default operator used to combine individual terms in the search string. For example: `defaultOperator: 'or'`. For the `dismax` parser, you specify a percentage that represents the percentage of terms in the search string (rounded down) that must match, rather than a default operator. A value of `0%` is the equivalent to OR, and a value of `100%` is equivalent to AND. The percentage must be specified as a value in the range 0-100 followed by the percent (%) symbol. For example, `defaultOperator: 50%`. Valid values: `and`, `or`, a percentage in the range 0%-100% (`dismax`). Default: `and` (`simple`, `structured`, `lucene`) or `100` (`dismax`). Valid for: `simple`, `structured`, `lucene`, and `dismax`.
  - `fields`—An array of the fields to search when no fields are specified in a search. If no fields are specified in a search and this option is not specified, all statically configured `text` and `text-array` fields are searched. You can specify a weight for each field to control the relative importance of each field when Amazon CloudSearch calculates relevance scores. To specify a field weight, append a caret (`^`) symbol and the weight to the field name. For example, to boost the importance of the `title` field over the `description` field you could specify: `fields: ['title^5','description']`. Valid values: The name of any configured field and an optional numeric value greater than zero. Default: All statically configured `text` and `text-array` fields. Dynamic fields and `literal` fields are not searched by default. Valid for: `simple`, `structured`, `lucene`, and `dismax`.
  - `operators`—An array of the operators or special characters you want to disable for the simple query parser. If you disable the `and`, `or`, or `not` operators, the corresponding operators (`+`, `|`, `-`) have no special meaning and are dropped from the search string. Similarly, disabling `prefix` disables the wildcard operator (`*`) and disabling `phrase` disables the ability to search for phrases by enclosing phrases in double quotes. Disabling precedence disables the ability to control order of precedence using parentheses. Disabling `near` disables the ability to use the ~ operator to perform a sloppy phrase search. Disabling the `fuzzy` operator disables the ability to use the ~ operator to perform a fuzzy search. `escape` disables the ability to use a backslash (`\`) to escape special characters within the search string. Disabling whitespace is an advanced option that prevents the parser from tokenizing on whitespace, which can be useful for Vietnamese. (It prevents Vietnamese words from being split incorrectly.) For example, you could disable all operators other than the phrase operator to support just simple term and phrase queries: `operators:['and', 'not', 'or', 'prefix']`. Valid values: `and`, `escape`, `fuzzy`, `near`, `not`, `or`, `phrase`, `precedence`, `prefix`, `whitespace`. Default: All operators and special characters are enabled. Valid for: `simple`.

- `phraseFields`—An array of the `text` or `text-array` fields you want to use for phrase searches. When the terms in the search string appear in close proximity within a field, the field scores higher. You can specify a weight for each field to boost that score. The `phraseSlop` option controls how much the matches can deviate from the search string and still be boosted. To specify a field weight, append a caret (^) symbol and the weight to the field name. For example, to boost phrase matches in the `title` field over the `abstract` field, you could specify: `phraseFields:['title^3', 'abstract']` Valid values: The name of any `text` or `text-array` field and an optional numeric value greater than zero. Default: No fields. If you don't specify any fields with `phraseFields`, proximity scoring is disabled even if `phraseSlop` is specified. Valid for: `dismax`.
- `phraseSlop`—An integer value that specifies how much matches can deviate from the search phrase and still be boosted according to the weights specified in the `phraseFields` option. For example, `phraseSlop: 2`. You must also specify `phraseFields` to enable proximity scoring. Valid values: positive integers. Default: 0. Valid for: `dismax`.
- `explicitPhraseSlop`—An integer value that specifies how much a match can deviate from the search phrase when the phrase is enclosed in double quotes in the search string. (Phrases that exceed this proximity distance are not considered a match.) `explicitPhraseSlop: 5`. Valid values: positive integers. Default: 0. Valid for: `dismax`.
- `tieBreaker`—When a term in the search string is found in a document's field, a score is calculated for that field based on how common the word is in that field compared to other documents. If the term occurs in multiple fields within a document, by default only the highest scoring field contributes to the document's overall score. You can specify a `tieBreaker` value to enable the matches in lower-scoring fields to contribute to the document's score. That way, if two documents have the same max field score for a particular term, the score for the document that has matches in more fields will be higher. The formula for calculating the score with a tieBreaker is:

  ```
  (max field score) + (tieBreaker) * (sum of the scores for the rest of
   the matching fields)
  ```

  For example, the following query searches for the term *dog* in the `title`, `description`, and `review` fields and sets `tieBreaker` to 0.1:

  ```
  q=dog&q.parser=dismax&q.options={fields:['title', 'description',
   'review'], tieBreaker: 0.1}
  ```

  If *dog* occurs in all three fields of a document and the scores for each field are title=1, description=3, and review=1, the overall score for the term dog is:

  ```
  3 +  0.1 * (1+1) = 3.2
  ```

  Set `tieBreaker` to 0 to disregard all but the highest scoring field (pure max). Set to 1 to sum the scores from all fields (pure sum). Valid values: 0.0 to 1.0. Default: 0.0. Valid for: `dismax`.

Type: JSON object

Default: See individual option descriptions.

Required: No

q.parser

Specifies which query parser to use to process the request: `simple`, `structured`, `lucene`, and `dismax`. If `q.parser` is not specified, Amazon CloudSearch uses the `simple` query parser.

- `simple`—perform simple searches of `text` and `text-array` fields. By default, the `simple` query parser searches all statically configured `text` and `text-array` fields. You can specify which fields to search by with the `q.options` parameter. If you prefix a search term with a

plus sign (+) documents must contain the term to be considered a match. (This is the default, unless you configure the default operator with the `q.options` parameter.) You can use the `-` (NOT), `|` (OR), and `*` (wildcard) operators to exclude particular terms, find results that match any of the specified terms, or search for a prefix. To search for a phrase rather than individual terms, enclose the phrase in double quotes. For more information, see Searching Your Data with Amazon CloudSearch (p. 94).

- `structured`—perform advanced searches by combining multiple expressions to define the search criteria. You can also search within particular fields, search for values and ranges of values, and use advanced options such as term boosting, `matchall`, and `near`. For more information, see Constructing Compound Queries (p. 97).
- `lucene`—search using the Apache Lucene query parser syntax. For more information, see Apache Lucene Query Parser Syntax.
- `dismax`—search using the simplified subset of the Apache Lucene query parser syntax defined by the DisMax query parser. For more information, see DisMax Query Parser Syntax.

Type: String

Default: `simple`

Required: No

return
: The field and expression values to include in the response, specified as a comma-separated list. By default, a search response includes all return enabled fields (`return=_all_fields`). To return only the document IDs for the matching documents, specify `return=_no_fields`. To retrieve the relevance score calculated for each document, specify `return=_score`. You specify multiple return fields as a comma separated list. For example, `return=title,_score` returns just the title and relevance score of each matching document.

  Type: String

  Required: No

size
: The maximum number of search hits to return.

  Type: Positive integer

  Default: 10

  Required: No

sort
: A comma-separated list of fields or custom expressions to use to sort the search results. You must specify the sort direction (`asc` or `desc`) for each field. For example, `sort=year desc,title asc`. You can specify a maximum of 10 fields and expressions. To use a field to sort results, it must be sort enabled in the domain configuration. Array type fields cannot be used for sorting. If no `sort` parameter is specified, results are sorted by their default relevance scores in descending order: `sort=_score desc`. You can also sort by document ID (`sort=_id`) and version (`sort=_version`).

  Type: String

  Required: No

start
: The offset of the first search hit you want to return. You can specify either the `start` or `cursor` parameter in a request, they are mutually exclusive. For more information, see Paginating Results (p. 135).

  Type: Positive integer

  Default: 0 (the first hit)

Required: No

Structured Search Syntax

You use the Amazon CloudSearch structured search syntax to define search criteria when using the `structured` query parser, and to specify filter criteria with the `fq` parameter.

When using the structured query operators, you specify the name of the operator, options for the operator, and then the terms being operated on, `(OPERATOR OPTIONS STRING|EXPRESSION)`. Any options must be specified before the string or expression. For example, `(and (not field=genres 'Sci-Fi')(or (term field=title boost=2 'star')(term field=plot 'star')))`.

Important
You must URL-encode special characters in the query string. For example, you must encode the = operator in a structured query as `%3D`: `(term+field%3Dtitle+'star')`. Amazon CloudSearch returns an `InvalidQueryString` error if special characters are not URL-encoded. For a complete reference of URL-encodings, see the W3C HTML URL Encoding Reference.

If you do not specify the field you want to search when using the structured query parser, all statically configured `text` and `text-array` fields are searched. Dynamic fields and `literal` fields are *not* searched by default. You can specify which fields you want to search by default with the `q.options` parameter.

Parentheses control the order of evaluation of the expressions in a compound query. When an expression is enclosed in parentheses, that expression is evaluated first, and then the resulting value is used in the evaluation of the remainder of the query. The expressions can contain any of the structured query operators.

You can also use the structured query parser to search for a simple text string—just enclose the string you want to search for in single quotes: `q='black swan'&q.parser="structured"`.

For more information about constructing compound queries with the structured query operators, see Constructing Compound Queries (p. 97).

FIELD
Syntax: `FIELD: 'STRING'|value`

Searches the specified field for a string, numeric value, date, or range of numeric values or dates.

Strings must be enclosed in single quotes. Any single quotation marks or backslashes in the string must be escaped with a backslash. To specify a range of values, use a comma (,) to separate the upper and lower bounds and enclose the range using brackets or braces. A square bracket, [or], indicates that the bound is included in the range, a curly brace, { or }, excludes the bound. You can omit the upper or lower bound to specify an open-ended range. When omitting a bound, you must use a curly brace.

Dates and times are specified in UTC (Coordinated Universal Time) according to IETF RFC3339: `yyyy-mm-ddTHH:mm:ss.SSSZ`. In UTC, for example, 5:00 PM August 23, 1970 is: `1970-08-23T17:00:00Z`. Note that you can also specify fractional seconds when specifying times in UTC. For example, `1967-01-31T23:20:50.650Z`.

Examples:

```
title:'star'
year:2000
year:[1998,2000]
year:{,2011]
release_date:['2013-01-01T00:00:00Z',}
```

and

Syntax: `(and boost=N EXPRESSION EXPRESSION ... EXPRESSIONn)`

Includes a document only if it matches all of the specified expressions.(Boolean `AND` operator.) The expressions can contain any of the structured query operators, or a simple search string. Search strings must be enclosed in single quotes. Note that to match documents that contain the specified terms in any of the fields being searched, you specify each term as a separate expression: `(and 'star' 'wars')`. If you specify `(and 'star wars')`, *star* and *wars* must occur within the same field to be considered a match.

The boost value is a positive numeric value that increases the importance of this part of the search query relative to the other parts.

Example:

```
(and title:'star' actors:'Harrison Ford' year:{,2000])
```

matchall

Syntax: `matchall`

Matches every document in the domain. By default, returns the first 10. Use the `size` and `start` parameters to page through the results.

near

Syntax: `(near field=FIELD distance=N boost=N 'STRING')`

Searches a `text` or `text-array` field for the specified multi-term string and matches documents that contain the terms within the specified distance of one another. (This is sometimes called a *sloppy* phrase search.) If you omit the `field` option, Amazon CloudSearch searches all statically configured `text` and `text-array` fields by default. Dynamic fields and `literal` fields are not searched by default. You can specify which fields you want to search by default by specifying the `q.options` `fields` option.

The distance value must be a positive integer. For example, to find all documents where *teenage* occurs within 10 words of *vampire* in the `plot` field, you specify a distance value of 10: `(near field=plot distance=10 'teenage vampire')`.

The boost value is a positive numeric value that increases the importance of this part of the search query relative to the other parts.

Example:

```
(near field=plot distance=10 'teenage vampire')
```

not

Syntax: `(not boost=N EXPRESSION)`

Excludes a document if it matches the specified expression. (Boolean `NOT` operator.) The expression can contain any of the structured query operators, or a simple search string. Search strings must be enclosed in single quotes.

The boost value is a positive numeric value that increases the importance of this part of the search query relative to the other parts.

Example:

```
(not (or actors:'Harrison Ford' year:{,2010]))
```

or

Syntax: `(or boost=N EXPRESSION1 EXPRESSION2 ... EXPRESSIONn)`

Includes a document if it matches any of the specified expressions. (Boolean `OR` operator.) The expressions can contain any of the structured query operators, or a simple search string. Search strings must be enclosed in single quotes.

The boost value is a positive numeric value that increases the importance of this part of the search query relative to the other parts.

Example:

```
(or actors:'Alec Guinness' actors:'Harrison Ford' actors:'James Earl
 Jones')
```

phrase

Syntax: `(phrase field=FIELD boost=N 'STRING')`

Searches a `text` or `text-array` field for the specified phrase. If you omit the `field` option, Amazon CloudSearch searches all statically configured `text` and `text-array` fields by default. Dynamic fields and `literal` fields are not searched by default. You can specify which fields you want to search by default by specifying the `q.options` `fields` option.

Use the `phrase` operator to combine a phrase search with other search criteria in a structured query. For example `q=(and (term field=title 'star') (range field=year {,2000]))` matches all documents that contain *star* in the title field and have a year value less than or equal to 2000.

The boost value is a positive numeric value that increases the importance of this part of the search query relative to the other parts.

Example:

```
(phrase field=plot 'teenage girl')
```

prefix

Syntax: `(prefix field=FIELD boost=N 'STRING')`

Searches a `text`, `text-array`, `literal`, or `literal-array` field for the specified prefix followed by zero or more characters. If you omit the `field` option, Amazon CloudSearch searches all statically configured `text` and `text-array` fields by default. Dynamic fields and `literal` fields are not searched by default. You can specify which fields you want to search by default by specifying the `q.options` `fields` option.

Use the `prefix` operator to combine a prefix search with other search criteria in a structured query. For example, `q=(and (prefix field=title 'sta') (range field=year {,2000]))` matches all documents that contain the prefix *sta* in the title field and have a year value of less than or equal to 2000.

The boost value is a positive numeric value that increases the importance of this part of the search query relative to the other parts.

Note
To implement search suggestions, you should configure and query a suggester, rather than performing prefix searches. For more information see Suggestion Requests (p. 256).

Example:

```
(prefix field=title 'star')
```

range

Syntax: `(range field=FIELD boost=N RANGE)`

Searches a numeric field (double, double-array, int, int-array) or date field (date, date-array) for values in the specified range. Matches documents that have at least one value in the field within the specified range. The `field` option must be specified.

Use the `range` operator to combine a range search with other search criteria in a structured query. For example `q=(and (term field=title 'star') (range field=year {,2000]))` matches all documents that contain *star* in the title field and have a year value of less than or equal to 2000.

To specify a range of values, use a comma (,) to separate the upper and lower bounds and enclose the range using brackets or braces. A square bracket, [or], indicates that the bound is included in the range, a curly brace, { or }, excludes the bound. You can omit the upper or lower bound to specify an open-ended range. When omitting a bound, you must use a curly brace.

Dates and times are specified in UTC (Coordinated Universal Time) according to IETF RFC3339: `yyyy-mm-ddTHH:mm:ss.SSSZ`. In UTC, for example, 5:00 PM August 23, 1970 is: `1970-08-23T17:00:00Z`. Note that you can also specify fractional seconds when specifying times in UTC. For example, `1967-01-31T23:20:50.650Z`.

The boost value is a positive numeric value that increases the importance of this part of the search query relative to the other parts.

Examples:

```
(range field=year [1990,2000])
(range field=year {,2000])
(range field=year [1990,})
```

term

Syntax: `(term field=FIELD boost=N 'STRING'|VALUE)`

Searches the specified field for a string, numeric value, or date. The `field` option must be specified when searching for a value. If you omit the `field` option, Amazon CloudSearch searches all statically configured `text` and `text-array` fields by default. Dynamic fields and `literal` fields are not searched by default. You can specify which fields you want to search by default by specifying the `q.options fields` option.

Use the `term` operator to combine a term search with other search criteria in a structured query. For example, `q=(and (term field=title 'star') (range field=year {,2000]))` matches all documents that contain *star* in the title field and have a year value of less than or equal to 2000.

Strings and dates must be enclosed in single quotes. Any single quotation marks or backslashes in a string must be escaped with a backslash.

Dates and times are specified in UTC (Coordinated Universal Time) according to IETF RFC3339: `yyyy-mm-ddTHH:mm:ss.SSSZ`. In UTC, for example, 5:00 PM August 23, 1970 is: `1970-08-23T17:00:00Z`. Note that you can also specify fractional seconds when specifying times in UTC. For example, `1967-01-31T23:20:50.650Z`.

The boost value is a positive numeric value that increases the importance of this part of the search query relative to the other parts.

Examples:

```
(term field=title 'star')
(term field=year 2000)
```

Simple Search Syntax

You use the Amazon CloudSearch simple search syntax to define search criteria when using the `simple` query parser. The simple query parser is used by default if you do not specify the `q.parser` parameter.

You use the simple query parser to search for individual terms or phrases. By default, all statically configured `text` and `text-array` fields are searched. Dynamic fields and `literal` fields are *not* searched by default. You can use the `q.options` parameter to specify which fields you want to search, change the default operator used to combine individual terms in the search string, or disable any of the simple parser operators (`and`, `escape`, `fuzzy`, `near`, `not`, `or`, `phrase`, `precedence`, `prefix`, `whitespace`).

For more information about using the simple query parser, see Searching for Text in Amazon CloudSearch (p. 99).

+ (and)
  Syntax: `+TERM`

  Requires the specified term. To match, documents must contain the specified term.

  Example: +star

\ (escape)
  Syntax: `\CHAR`

  Escapes special characters that you want to search for. You must escape the following characters if you want them to be part of the query: + - & | ! ( ) { } [ ] ^ " ~ * ? : \ /.

  Example: `M\*A\*S\*H`

~ (fuzzy)
  Syntax: `TERM~N`

  Performs a fuzzy search. Append the ~ operator and a value to a term to indicate how much terms can differ and still be considered a match.

  Example: `stor~1`

~ (near)
  Syntax: `"PHRASE"~N`

  Performs a sloppy phrase search. Append the ~ operator and a value to a phrase to indicate how far apart the terms can be and still be considered a match for the phrase.

  Example: `"star wars"~4`

- (not)
  Syntax: `-TERM`

  Prohibits the specified term. To match, documents must not contain the term.

  Example: star -wars

| (or)
  Syntax: `|TERM`

  Makes the specified term optional.

  Example: star |wars

"..." (phrase)
  Syntax: `"PHRASE"`

  Performs a search for the entire phrase. Can be combined with the ~ operator to perform a sloppy phrase search.

  Example: "star wars"

(...) (precedence)
Syntax: `( . . . )`

Controls the order in which the query constraints are evaluated. The contents of the inner-most parentheses are evaluated first.

Example: `+(war|trek)+star`

* (prefix)
Syntax: `CHARS*`

Matches documents that contain terms that have the specified prefix.

Example: `sta*`

Search Response

When a request completes successfully, the response body contains the search results. By default, search results are returned in JSON. If the `format` parameter is set to `xml`, search results are returned in XML.

Unless you explicitly specify the `return` parameter, the document ID and all returnable fields are included for each matching document (hit). The response also shows the total number of hits found (`found`) and the index of the first document listed (`start`). By default, the response contains the first 10 hits. You specify the `size` parameter in your request to control how many hits are included in each response. To page through the hits, you can use the `start` or `cursor` parameter. For more information, see Paginating Results (p. 135).

The following example shows a typical JSON response.

```
{
    "status": {
        "rid": "rtKz7rkoeAojlvk=",
        "time-ms": 10
    },
    "hits": {
        "found": 3,
        "start": 0,
        "hit": [
            {
                "id": "tt1142977",
                "fields": {
                    "rating": "6.9",
                    "genres": [
                        "Animation",
                        "Comedy",
                        "Family",
                        "Horror",
                        "Sci-Fi"
                    ],
                    "plot": "Young Victor conducts a science experiment to
                             bring his beloved dog Sparky back to life, only
                              to face unintended, sometimes monstrous,
                              consequences.",
                    "release_date": "2012-09-20T00:00:00Z",
                    "title": "Frankenweenie",
                    "rank": "1462",
                    "running_time_secs": "5220",
                    "directors": [
```

```
                        "Tim Burton"
                    ],
                    "image_url": "http://ia.media-imdb.com/images/M/MV5BMjIx
                                  ODY3MjEwNV5BMl5BanBnXkFtZTcwOTMzNjc4Nw@@._
                                  V1_SX400_.jpg",
                    "year": "2012",
                    "actors": [
                        "Winona Ryder",
                        "Catherine O'Hara",
                        "Martin Short"
                    ]
                }
            },
   .
   .
   .
        ]
    }
}
```

The following example shows the equivalent XML response.

```
<results>
    <status rid="itzL7rkoeQojlvk=" time-ms="34"/>
    <hits found="3" start="0">
        <hit id="tt1142977">
            <field name="rating">6.9</field>
            <field name="genres">Animation</field>
            <field name="genres">Comedy</field>
            <field name="genres">Family</field>
            <field name="genres">Horror</field>
            <field name="genres">Sci-Fi</field>
            <field name="plot">Young Victor conducts a science experiment to
                               bring his beloved dog Sparky back to life,
 only
                               to face unintended, sometimes monstrous,
                               consequences.
            </field>
            <field name="release_date">2012-09-20T00:00:00Z</field>
            <field name="title">Frankenweenie</field>
            <field name="rank">1462</field>
            <field name="running_time_secs">5220</field>
            <field name="directors">Tim Burton</field>
            <field name="image_url">http://ia.media-imdb.com/images/M/MV5BMjI

 xODY3MjEwNV5BMl5BanBnXkFtZTcwOTMzNjc4Nw@@.
                                    _V1_SX400_.jpg
            </field>
            <field name="year">2012</field>
            <field name="actors">Winona Ryder</field>
            <field name="actors">Catherine O'Hara</field>
            <field name="actors">Martin Short</field>
        </hit>
        .
        .
        .
    </hits>
</results>
```

Setting the response format only affects responses to successful requests. The format of an error response depends on the origin of the error. Errors returned by the search service are always returned in JSON. 5xx errors due to server timeouts and other request routing problems are returned in XML. When a request returns an error code, the body of the response contains information about the error that occurred. If an error occurs while the request body is parsed and validated, the error code is set to 400 and the response body includes a list of the errors and where they occurred.

Search Response Headers

Content-Type
: A standard MIME type describing the format of the object data. For more information, see W3C RFC 2616 Section 14.

  Valid values: application/json or application/xml

  Default: application/json

Content-Length
: The length in bytes of the body in the response.

Search Response Properties (JSON)

status
: Contains the resource id (rid) and the time it took to process the request (time-ms).

  rid
  : The encrypted Resource ID.

  time-ms
  : How long it took to process the search request in milliseconds.

hits
: Contains the number of matching documents (`found`), the index of the first document included in the response (`start`), and an array (`hit`) that lists the document IDs and data for each hit.

  found
  : The total number of hits that match the search request after Amazon CloudSearch finished processing the request.

  start
  : The index of the first hit returned in this response.

  hit
  : An array that lists the document IDs and data for each hit.

    id
    : The unique identifier for a document.

    fields
    : A list of returned fields.

    facets
    : Contains facet information and facet counts.

    `FACETFIELD`
    : A field for which facets were calculated.

    buckets
    : An array of the calculated facet values and counts.

    value
    : The facet value being counted.

    count
    : The number of hits that contain the facet value in `FACETFIELD`.

Search Response Elements (XML)

results
: Contains the search results. Any errors that occurred while processing the request are returned as messages in the info element.

  status
  : Contains the resource id (`rid`) and the time it took to process the request (`time-ms`).

  hits
  : Contains hit statistics and a collection of hit elements. The found attribute is the total number of hits that match the search request after Amazon CloudSearch finished processing the results. The contained hit elements are ordered according to their relevance scores or the `sort` option specified in the search request.

    hit
    : A document that matched the search request. The id attribute is the document's unique id. Contains a d (data) element for each returned field.

      field
      : A field returned from a hit. Hit elements contain a d (data) element for each returned field.

      facets
      : Contains a facet element for each facet requested in the search request.

        facet
        : Contains a bucket element for each value of a field for which a facet count was calculated. The `facet.FIELD` size option can be used to specify how many constraints to return. By default, facet counts are returned for the top 10 constraints. The `facet.FIELD` buckets option can be used to explicitly specify which values to count.

          bucket
          : A facet field value and the number of occurrences (count) of that value within the search hits.

Submitting Suggest Requests in Amazon CloudSearch

You submit suggest requests via HTTP GET to your domain's search endpoint at `2013-01-01/suggest`. For information about controlling access to the suggest service, see Configuring Access for Amazon CloudSearch (p. 29).

You must specify the API version in all suggest requests and that version must match the API version specified when the domain was created.

For example, the following request gets suggestions from the `search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.amazonaws.com` domain for the query string `oce` using the suggester called `title`.

```
http://search-imdb-hd6ebyouhw2lczkueyuqksnuzu.us-
west-2.cloudsearch.amazonaws.com/2013-01-01/suggest -
d"q=oce&suggester=suggest_title"
```

You can use any method you want to send GET requests to your domain's search endpoint—you can enter the request URL directly in a Web browser, use cURL to submit the request, or generate an HTTP call using your favorite HTTP library. You can also use the Search Tester in the Amazon CloudSearch console to get suggestions. For more information, see Searching with the Search Tester (p. 12).

Important
A domain's document and search endpoints remain the same for the life of the domain. You should cache the endpoints rather than retrieving them before every upload or search request. Querying the Amazon CloudSearch configuration service by calling `aws cloudsearch describe-domains` or `DescribeDomains` before every request is likely to result in your requests being throttled.

By default, Amazon CloudSearch returns the response in JSON. You can get the results formatted in XML by specifying the `format` parameter, `format=xml`. Setting the response format only affects responses to successful requests. The format of an error response depends on the origin of the error. Errors returned by the search service are always returned in JSON. 5xx errors due to server timeouts and other request routing problems are returned in XML.

Suggest

Suggestion Requests

Suggest Syntax in Amazon CloudSearch

```
GET /2013-01-01/suggest
```

Suggest Request Headers in Amazon CloudSearch

HOST
: The search request endpoint for the domain you're querying. You can use DescribeDomains (p. 177) to retrieve your domain's search request endpoint.

  Required: Yes

Suggest Request Parameters in Amazon CloudSearch

q
: The string to get suggestions for.

  Type: String

  Required: Yes

suggester
: The name of the suggester to use to find suggested matches.

  Type: String

  Required: Yes

size
: The maximum number of suggestions to return.

  Type: Positive integer

  Default: 10

  Required: No

format
: Specifies the content type of the response.

  Type: String

Valid Values: json|xml

Default: json

Required: No

Suggest Response

When a request completes successfully, the response body contains the suggestions. By default, suggestions are returned in JSON. Set the `format` parameter to `xml` to get the results in XML.

Setting the response format only affects responses to successful requests. The format of an error response depends on the origin of the error. Errors returned by the search service are always returned in JSON. 5xx errors due to server timeouts and other request routing problems are returned in XML. When a request returns an error code, the body of the response contains information about the error that occurred. If an error occurs while the request body is parsed and validated, the error code is set to 400 and the response body includes a list of the errors and where they occurred.

The following example shows a JSON response to a request for suggestions:

```
{
   "status": {
      "rid": "qOSM5s0oCwr8pVk=",
      "time-ms": 2
   },
   "suggest": {
      "query": "oce",
      "found": 3,
      "suggestions": [
         {
          "suggestion": "Ocean's Eleven",
           "score": 0,
           "id": "tt0054135"
         },
         {
          "suggestion": "Ocean's Thirteen",
          "score": 0,
          "id": "tt0496806"
         },
         {
          "suggestion": "Ocean's Twelve",
          "score": 0,
          "id": "tt0349903"
         }
      ]
   }
}
```

The following example shows the equivalent XML response:

```
<results>
   <status rid="/pSz580oDQr8pVk=" time-ms="2"/>
   <suggest query="oce" found="3">
      <suggestions>
         <item suggestion="Ocean's Eleven" score="0" id="tt0054135"/>
         <item suggestion="Ocean's Thirteen" score="0" id="tt0496806"/>
         <item suggestion="Ocean's Twelve" score="0" id="tt0349903"/>
```

```
      </suggestions>
   </suggest>
</results>
```

Search Service Errors

A search or suggestion request can return three types of status codes:

- 5xx status codes indicate that there was an internal server error. You should catch and retry all 5xx error codes as they typically represent transient error conditions. For more information, see Handling Errors (p. 137).
- 4xx status codes indicate that the request was malformed. Correct the error(s) before resubmitting your request.
- 2xx status codes indicate that the request was processed successfully.

The format of an error response depends on the origin of the error. Errors returned by the search service are always returned in JSON. 5xx errors due to server timeouts and other request routing problems are returned in XML.

Errors returned by the search service contain the following information:

error
: Contains an error message returned by the search service. The `code` and `msg` properties are included for each error.

code
: The error code.

msg
: A description of the error that was returned by the search service.

Troubleshooting Amazon CloudSearch

The following topics describe solutions to problems you might encounter when using Amazon CloudSearch.

Topics

Uploading Documents

If your document data is not formatted correctly or contains invalid values, you will get errors when you attempt to upload it or use it to configure fields for your domain. Here are some common problems and their solutions:

- **Invalid JSON**—if you are using JSON, the first thing to do is make sure there are no JSON syntax errors in your document batch. To do that, run it through a validation tool such as the JSON Validator. This will identify any fundamental issues with the data.

- **Invalid XML**—document batches must be well-formed XML. You are especially likely to encounter issues if your fields contain XML data—the data must be XML-encoded or enclosed in CDATA sections. To identify any problems, run your document batch through a validation tool such as the W3C Markup Validation Service.
- **Not Recognized as a Document Batch**—if Amazon CloudSearch doesn't recognize your data as a valid document batch when you are uploading data with `cs-import-documents` or the console, Amazon CloudSearch generates a valid batch that contains a single content field and generic metadata fields such as `content_encoding`, `content_type`, and `resourcename`. Since these are not normally the fields configured for the domain, you get errors stating that the fields don't exist. Similarly, if you attempt to configure a domain from an invalid batch, Amazon CloudSearch responds with the content and meta-data fields instead of the fields in the batch.

  First, make sure that the batch is valid XML or JSON. If it is, check for invalid document IDs and make sure you have specified the operation type for each document. For add operations, make sure that the type, ID, and at least one field are specified for each document. Delete operations only need to specify the type and ID. For more information about formatting your data, see Creating Document Batches (p. 59).
- **Document IDs with bad values**—A document ID can contain any letter or number and the following characters: _ - = # ; : / ? @ &. Document IDs must be at least 1 and no more than 128 characters long.
- **Multi-valued fields without a value**—when specifying document data in JSON, you cannot specify an empty array as the value of a field. Multi-valued fields must contain at least one value.
- **Bad characters**—one problem that can be difficult to detect if you do not filter your data while generating your document batch is that can contain characters that are invalid in XML. Both JSON and XML batches can contain only UTF-8 characters that are valid in XML. You can use a validation tool such as the JSON Validator or W3C Markup Validation Service to identify invalid characters.

Deleting All Documents in an Amazon CloudSearch Domain

Amazon CloudSearch currently does not provide a mechanism for deleting all of the documents in a domain.

Amazon CloudSearch Domain Not Scaling Down After Deleting Documents

If your domain has scaled up to accommodate your index size and you delete a large number of documents, the domain scales down the next time the full index is rebuilt. Although the index is automatically rebuilt periodically, to scale down as quickly as possible you can explicitly run indexing (p. 91) when you are done deleting documents.

Document Update Latency

Sending a large volume of single-document batches can increase the amount of time it takes each document to become searchable. If you have a large amount of update traffic, you need to batch your updates. We recommend using a batch size close to the 5 MB limit. For more information, see Creating Document Batches (p. 59).

You can load up to 10,000 document batches per day (every 24 hours), with each batch size up to 5 MB. Loading more data per day significantly increases the latency of document updates. To

mitigate this risk, you can increase your update capacity by selecting a larger instance type. For more information, see Configuring Scaling Options (p. 39).

Large Number of 5xx Errors When Uploading Documents to an Amazon CloudSearch Domain

If you parallelize uploads and your domain is on a search.m1.small instance, you might experience an unacceptably high rate of 504 or 507 errors. Setting the desired instance type to a larger instance type will increase your update capacity and reduce the error rate. For more information about handling 5xx errors, see Handling Errors (p. 137). For information about prescaling your domain to increase upload capacity, see Configuring Scaling Options (p. 39).

Search Latency and Timeouts in Amazon CloudSearch

If you are experiencing slow response times, frequently encountering internal server errors (typically 507 or 509 errors), or seeing the number of instance hours your search domain is consuming increase without a substantial increase in the volume of data you're searching, fine-tuning your search requests to reduce the processing overhead can help. For more information, see Tuning Search Request Performance in Amazon CloudSearch (p. 111). Increasing the desired replication count can also speed up search request processing. For more information, see Configuring Scaling Options (p. 39).

507 and 509 errors typically indicate that your search service is overloaded. This can be due to the volume or complexity of search requests that you are submitting. Amazon CloudSearch normally scales automatically to handle the load. Because it takes some time to deploy additional search instances, we recommend using an exponential backoff retry policy to temporarily reduce the request rate and minimize request failures. For more information, see Error Retries and Exponential Backoff.

Certain usage patterns, such as submitting complex search queries to a single small search instance, can sometimes result in timeouts without triggering automatic scaling. If you repeatedly experience a high error rate, you can explicitly request additional capacity through the Amazon CloudSearch Service Limit Request form.

Search Latency for Faceted Queries in Amazon CloudSearch

If you are bucketing facet information with the `buckets` option and experiencing slow query performance, try setting the buckets method to `interval`. For more information, see Bucketing Facet Information (p. 117).

Sudden Increase in 5xx Errors When Searching an Amazon CloudSearch Domain

If your search domain experiences a sudden spike in traffic, Amazon CloudSearch responds by adding search instances to your domain to handle the increased load. However, it takes a few minutes to set

up the new instances. You are likely to see a temporary increase in 5xx errors until the new instances are ready to start processing requests. For more information about handling 5xx errors, see Handling Errors (p. 137). For information about pre-scaling your domain to handle an expected spike in search requests, see Configuring Scaling Options (p. 39).

Indexing Failures after Updating Indexing Options in Amazon CloudSearch

If you make changes to a domain's index configuration, in certain cases you might encounter Failed to Validate errors when you run indexing. This means that the index field options you set are not compatible with the documents that are already in your index. Specifically, you changed the type of an index field, and there are documents in your index that contain data that is incompatible with that type. For example, this might happen if you change a `literal` field to an `int` field, and some of your documents contain alphanumeric data in that field. When this happens, Amazon CloudSearch sets the status of ALL fields that were being processed to the `FailedToValidate` state. Rolling back the incompatible configuration change will enable you to successfully rebuild your index. If the change is necessary, you must update or remove the incompatible documents from your index to use the new configuration. If you can't identify the change that caused the error or need assistance identifying the incompatible documents, contact support.

Domain Not Found When Submitting Amazon CloudSearch Requests

You cannot access a 2013-01-01 domain with the 2011-02-01 command line tools or SDKs. Similarly, you cannot access a 2011-02-01 domain with the 2013-01-01 command line tools or SDKs. Make sure you are specifying the correct API version in your request and using the appropriate command line tools or SDK.

Number of Searchable Documents Not Returned with Domain Information

The `aws cloudsearch describe-domains` and `DescribeDomains` do not return the number of searchable documents in the domain. To get the number of searchable documents, use the console or submit a `matchall` request to your domain's search endpoint.

```
q=matchall&q.parser=structured&size=0
```

Configuration Service Access Policies Not Working in Amazon CloudSearch

If you have both 2011 and 2013 domains, have configured IAM policies for accessing the configuration service, and are getting *not authorized* errors, note that the Amazon CloudSearch ARN is different for

the 2011-02-01 API and the 2013-01-01 API. To allow users to access both 2011 and 2013 domains, you must allow access to both ARNs in the IAM policy. For example:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudsearch:*",
       ],
      "Resource": "arn:aws:cloudsearch:*",
      "Resource": "arn:aws:cs:*"
    }
  ]
}
```

If your 2011 policy granted access to particular domains or actions, you must include those restrictions in your policy. Note that the only supported action for 2011 domains is `cloudsearch:*` and you might encounter other errors when attempting to configure resource-level permissions for domains created with the 2011-01-01 API.

Search and Document Service Access Policies Not Working in Amazon CloudSearch

If you have configured access policies for you domain's search and document service endpoints, but are getting the error *403 Request forbidden by administrative rules*, it is likely due to one of the following issues.

- Make sure the API version and resource name are specified in your requests. For example, to upload documents with the 2013-01-01 API, you must append `/2013-01-01/documents/batch` to your domain's document service endpoint:

  ```
  doc-movies-123456789012.us-east-1.cloudsearch.amazonaws.com/2013-01-01/documents/batch
  ```

  To submit search requests using the 2013-01-01 API, you must append `/2013-01-01/search` to your domain's search endpoint:

  ```
  search-movies-123456789012.us-east-1.cloudsearch.amazonaws.com/2013-01-01/search?q=star+wars&return=title
  ```

  To get suggestions using the 2013-01-01 API, you must append `/2013-01-01/suggest` to your domain's search endpoint:

  ```
  search-movies-123456789012.us-east-1.cloudsearch.amazonaws.com/2013-01-01/suggest?q=kat&suggester=mysuggester
  ```

- If you are connecting from an EC2 instance, make sure the access policy specifies your EC2 instance's public IP address.
- If the machine you are connecting from is behind a router, make sure the access policy specifies your public facing IP address.

For more information, see Configuring Access for Amazon CloudSearch (p. 29).

Amazon CloudSearch Console Permissions Errors

To access to the console, you must have permissions for the `DescribeDomains` action. Access to particular domains and actions might be restricted by to the configured IAM access policies. In addition, uploading data from an Amazon S3 bucket or DynamoDB table requires access to those services and resources. For more information about Amazon CloudSearch access policies, see Configuring Access for Amazon CloudSearch (p. 29).

Using Wildcards to Search Text Fields Doesn't Produce Expected Results

When you submit a search request, the text you're searching for undergoes the same text processing so that it can be matched against the terms that appear in the index. However, no text analysis is performed on the search term when you perform a prefix search. This means that a search for a prefix that ends in `s` typically won't match the singular version of the term when stemming is enabled. This can happen for any term that ends in `s`, not just plurals. For example, if you search the `actor` field in the sample movie data for `Anders`, there are three matching movies. If you search for `Ander*`, you get those movies as well as several others. However, if you search for `Anders*` there are no matches. This is because the term is stored in the index as `ander`, `anders` does not appear in the index.

If stemming is preventing your wildcard searches from returning all of the relevant matches, you can suppress stemming for the text field by setting the `AlgorithmicStemming` option to none, or you can map the data to a `literal` field instead of a `text` field.

For more information about how Amazon CloudSearch processes text, see Text Processing in Amazon CloudSearch (p. 79).

Inconsistent Results When Using Cursors for Deep Paging

When you use a cursor to page through a result set that is sorted by document score (`_score`), you can get inconsistent results if the index is updated between requests. This can also occur if your domain's replication count is greater than one, because updates are applied in an eventually consistent manner across the instances in the domain. If this is an issue, avoid sorting the results by score. You can either use the `sort` option to sort by a particular field, or use `fq` instead of `q` to specify your search criteria. (Document scores are not calculated for filter queries.)

Understanding Amazon CloudSearch Limits

This table shows naming and size restrictions within Amazon CloudSearch. You can submit a request if you need to increase the maximum number of search instances or partitions for a search domain. For information about increasing other limits such as the maximum number of search domains, contact Amazon CloudSearch.

The current Amazon CloudSearch limits are summarized in the following table.

Item	Limit
Batch size	The maximum batch size is 5 MB.
Data loading volume	You can load up to 10,000 document batches per day (every 24 hours), with each batch size up to 5 MB. Loading more data per day significantly increases the latency of document updates. To mitigate this risk, you can increase your update capacity by selecting a larger instance type. For more information, see Creating Document Batches (p. 59).
Document size	The maximum document size is 1 MB.
Expressions	• Up to 50 expressions can be configured for a domain. • The maximum size of an expression is 10240 bytes. • The maximum value that can be returned by an expression is max(int64_t).
Highlighting	• The maximum number of occurrences of the search term(s) that can be highlighted is 5. • Highlights are only returned for the first 10 KB of data in a text field.
Index fields	• Up to 200 index fields can be configured for a domain. • Up to 1000 values can be specified in a field. • Up to 20 sources can be specified for an array-type field. • The maximum size of a literal field is 4096 UTF-8 code points. • The maximum size of a default value for a field is 1 KB.

Item	Limit
	• An int field can contain values from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (inclusive). • Individual terms within a text or text-array field are treated as stopwords if they exceed 256 characters.
Naming conventions	• Domain Names: Allowed characters are a-z (lower-case letters), 0-9, and hyphen (-). Domain names must start with a letter or number and be at least 3 and no more than 28 characters long. • Field Names: Allowed characters are a-z (lower-case letters), 0-9, and _ (underscore). Field names must begin with a letter and be at least 1 and no more than 64 characters long. The name *score* is reserved and cannot be used as a field name. • Expression Names: Allowed characters are a-z (lower-case letters), 0-9, and _ (underscore). Expression names must begin with a letter and be at least 3 and no more than 64 characters long. The name *score* is reserved and cannot be used as an expression name. • Document IDs: A document ID (_id) can contain any letter or number and the following characters: _ - = # ; : / ? @ &. Document IDs must be at least 1 and no more than 128 characters long.
Policy document size	The maximum size of an Amazon CloudSearch policy document is 100 KB.
Region restriction	The ap-northeast-2 region supports only m4 instance types.
_score	A document's text relevance score is a positive floating point value.
Search domains	Each AWS account can create up to 100 search domains.
Search instances	• The maximum number of search instances that can be deployed for a domain is 50. • The available instance types are: search.m1.small, search.m3.medium, search.m3.large, search.m3.xlarge, search.m3.2xlarge.
Search partitions	A search index can be split across a maximum of 10 partitions

Item	Limit
Search requests	• compound queries: Can contain a maximum of 1024 clauses. • GET requests: The maximum size of a search request submitted as an HTTP GET request is 8190 bytes. • facet parameter: The maximum number of facet values you can return is 10,000. • size parameter: Can contain values in the range 0 - 10000. The sum of the size and start parameters cannot exceed 10,000. If you need to page through more than 10,000 hits, use a cursor. • sort parameter: Can contain up to 10 int fields and expressions. • start parameter: Can contain values in the range 0 - 10000. The sum of the size and start parameters cannot exceed 10,000. If you need to page through more than 10,000 hits, use a cursor.
Suggesters	• You can define a maximum of 10 suggesters for a domain. • Only the first 512 bytes of a text field are used to generate suggestions. • The scores computed from a suggester's `SortExpression` are rounded to the nearest integer, with a floor of 0 and a ceiling of 2^31-1.

Amazon CloudSearch Resources

The following table lists resources that you might find useful as you work with Amazon CloudSearch.

Resource	Description
Amazon CloudSearch Command Line Tools	Download the Amazon CloudSearch command line tools for Mac OS/Linux or Windows to create and configure a search domain and upload the data you want to search.
AWS SDKs	Most of the AWS SDKs support Amazon CloudSearch, including the Java, .NET, Node.js, PHP, Python, and Ruby SDKs.
Amazon CloudSearch Sample Data	Download the IMDb Sample Data to get a search domain up and running quickly with the command line tools or Configuration Service API and see how to format your own data for Amazon CloudSearch.
Amazon CloudSearch Discussion Forum	The forum where Amazon CloudSearch users can post questions and discuss various Amazon CloudSearch topics.
Amazon CloudSearch Pricing	Pricing information for Amazon CloudSearch.
Request to Increase Limits	The form to request an increase in the maximum number of search instances or partitions for a search domain.
Latest Amazon CloudSearch Documentation	The most recent update of the Amazon CloudSearch Developer Guide for the 2013-01-01 API is available from the AWS website: View HTML | Download PDF
Amazon CloudSearch 2011-02-01 Developer Guide	The 2011-02-01 Amazon CloudSearch Developer Guide is available in PDF only: Download PDF.

Document History for Amazon CloudSearch

This Document History describes the important changes to the documentation in this release of *Amazon CloudSearch*.

Relevant Dates to this History:

- **Current product version—**2013-01-01
- **Latest product release—**10 February 2016
- **Latest documentation update—**10 December 2016

Change	Description	Release Date
Support for resource tagging	Amazon CloudSearch added support for resource tagging. For more information, see Tagging Amazon CloudSearch Domains (p. 54) in this service guide.	10 February 2016
AP (Seoul) support	Amazon CloudSearch added support for the AP (Seoul) `ap-northeast-2` region. For a list of regions supported by Amazon CloudSearch, see AWS Regions and Endpoints in the AWS General Reference.	28 January 2016
Integration with Amazon CloudWatch and support for index field statistics	You can now use Amazon CloudWatch to monitor your Amazon CloudSearch domains. CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. Amazon CloudSearch automatically sends metrics to CloudWatch so that you can gather and analyze performance statistics. You can monitor these metrics by using the Amazon CloudSearch console, or by using the CloudWatch console, AWS CLI, or AWS SDKs. There is no charge for the Amazon CloudSearch metrics that are reported through CloudWatch. For more information about using CloudWatch to monitor search domains, see Monitoring a Domain with Amazon CloudWatch (p. 49). You also can now retrieve statistics against facet-enabled numeric fields. Amazon CloudSearch can return the following statistics against indexed numeric fields in the	5 March 2015

Change	Description	Release Date
	documents: count, min, max, mean, missing, stddev, sum, and sumOfSquares. To learn more about index field statistics, see Querying For More Information (p. 114).	
Support for M3 instance types	You can now use M3 instances to power your Amazon CloudSearch domains. Amazon CloudSearch now supports the following instance types for newly created domains: `m1.small`, `m3.medium`, `m3.large`, `m3.xlarge`, and `m3.2xlarge`. For more information about newly available instance types and how to modify existing domains, see Configuring Scaling Options (p. 39).	10 February 2015
Support for Dynamic Fields	Dynamic fields provide a way to index documents without knowing in advance exactly what fields they contain. A dynamic field's name defines a pattern that contains a wildcard (*) as the first, last, or only character. Any unrecognized document field that matches the pattern is configured with the dynamic field's indexing options. For more information, see Using Dynamic Fields (p. 68).	11 December 2014
Enhanced Japanese Language Processing and CloudTrail Support	You can now control how Amazon CloudSearch tokenizes Japanese by adding a custom Japanese tokenization dictionary to the analysis scheme that you use for fields that contain Japanese. Configuring a custom tokenization dictionary can improve search result accuracy by facilitating indexing and retrieval of domain-specific phrases. To learn more about using custom dictionaries, see Customizing Japanese Tokenization (p. 76). You can also index bigrams for Chinese, Japanese, and Korean. For more information, see Indexing Bigrams for Chinese, Japanese, and Korean (p. 75). You can also now use AWS CloudTrail to get a history of Amazon CloudSearch configuration API calls and related events for your account. CloudTrail is a web service that records your account's API calls 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. For example, you can use the API call history to perform a security analysis or troubleshoot operational issues. CloudTrail also makes it easier for you to demonstrate compliance with internal policies or regulatory standards. For more information, see the Security at Scale: Logging in AWS whitepaper. For more information about using CloudTrail to log Amazon CloudSearch calls, see Logging Configuration Service Calls Using CloudTrail (p. 50).	15 October 2014
Documentation Update	This update clarifies that you must URL-encode search query strings and provides additional information about getting facet information for selected buckets. For more information about bucketing facets, see Getting Facet Information (p. 116).	19 September 2014

Change	Description	Release Date
Enhanced IAM Integration	You can now use IAM to control access to each domain's document, search, and suggest services and use AWS Signature Version 4 to sign all Amazon CloudSearch requests. Requests are signed automatically when you use the latest AWS SDKs and the AWS CLI. For more information, see Configuring Access for Amazon CloudSearch (p. 29). In conjunction with this release, there is an update of the Amazon CloudSearch command line tools. The updated CLTs now automatically sign document upload requests submitted through the `cs-import-documents` command. You can download the new CLT bundle from the Amazon CloudSearch developer tools page . **Important** This CLT update contains just two commands: cs-import-documents and cs-configure-from-batches. All configuration actions should be performed using the AWS CLI. The AWS CLI also supports uploading documents and submitting search and suggest requests. For more information, see the AWS Command Line Interface User Guide.	14 August 2014
Enhanced Amazon CloudSearch Support in the AWS SDKs and AWS CLI	The AWS SDKs and AWS CLI now provide full support for all Amazon CloudSearch 2013-01-01 API operations, including creating, configuring, and managing search domains, uploading documents, and submitting search requests. For information about installing and using the AWS CLI, see the AWS Command Line Interface User Guide. **Note** To generate document batches and automatically configure indexing options based on the contents of a batch, you still need to use the standalone Amazon CloudSearch command line tools.	26 June 2014
Hebrew Language Support and Desired Partition Count Scaling Option	Amazon CloudSearch now supports Hebrew in addition to the 33 other Supported Languages (p. 80). This update also adds a new scaling option, desired partition count. You can use this option to preconfigure the number of index partitions for a domain that uses the m2.2xlarge search instance type. If you have a large amount of search data, preconfiguring a domain to use more partitions can enable you to load data faster. You can also configure a domain with additional partitions to drop the per-partition document count and speed up complex queries. Amazon CloudSearch will still scale the domain up or down based on the volume of data or traffic, but the number of partitions will never drop below your desired partition count. For more information, see Configuring Scaling Options (p. 39).	24 March 2014

Change	Description	Release Date
Amazon CloudSearch 2013-01-01 API	Amazon CloudSearch has a new API version with many improvements and new features. The new API is not backward-compatible with the 2011-02-01 API. To use the new features, you must create a new search domain with the 2013-01-01 API. In conjunction with this release, there is also a new set of command line tools. Note that the new tools require a Java 7 compatible JRE, so you might need to update Java to use the tools. For more information, see What's New in Amazon CloudSearch and Migrating to the Amazon CloudSearch 2013-01-01 API (p. 18).	24 March 2014

AWS Glossary

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

Numbers and Symbols

100-continue	A method that enables a client to see if a server can accept a request before actually sending it. For large PUT requests, this method can save both time and bandwidth charges.

A

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

AAD	See additional authenticated data.
access control list (ACL)	A document that defines who can access a particular bucket (p. 285) or object. Each bucket (p. 285) and object in Amazon S3 (p. 278) has an ACL. The document defines what each type of user can do, such as write and read permissions.
access identifiers	See credentials.
access key	The combination of an access key ID (p. 273) (like `AKIAIOSFODNN7EXAMPLE`) and a secret access key (p. 313) (like `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). You use access keys to sign API requests that you make to AWS.
access key ID	A unique identifier that's associated with a secret access key (p. 313); the access key ID and secret access key are used together to sign programmatic AWS requests cryptographically.
access key rotation	A method to increase security by changing the AWS access key ID. This method enables you to retire an old key at your discretion.

access policy language	A language for writing documents (that is, *policies* (p. 306)) that specify who can access a particular AWS resource (p. 310) and under what conditions.
account	A formal relationship with AWS that is associated with (1) the owner email address and password, (2) the control of resource (p. 310)s created under its umbrella, and (3) payment for the AWS activity related to those resources. The AWS account has permission to do anything and everything with all the AWS account resources. This is in contrast to a user (p. 319), which is an entity contained within the account.
account activity	A web page showing your month-to-date AWS usage and costs. The account activity page is located at http://aws.amazon.com/account-activity/.
ACL	See access control list (ACL).
ACM	See AWS Certificate Manager (ACM).
action	An API function. Also called *operation* or *call*. The activity the principal (p. 307) has permission to perform. The action is B in the statement "A has permission to do B to C where D applies." For example, Jane sends a request to Amazon SQS (p. 278) with Action=ReceiveMessage. Amazon CloudWatch (p. 275): The response initiated by the change in an alarm's state: for example, from OK to ALARM. The state change may be triggered by a metric reaching the alarm threshold, or by a SetAlarmState request. Each alarm can have one or more actions assigned to each state. Actions are performed once each time the alarm changes to a state that has an action assigned, such as an Amazon Simple Notification Service (p. 278) notification, an Auto Scaling (p. 280) policy (p. 306) execution or an Amazon EC2 (p. 276) instance (p. 298) stop/terminate action.
active trusted signers	A list showing each of the trusted signers you've specified and the IDs of the corresponding active key pairs that Amazon CloudFront (p. 275) is aware of. To be able to create working signed URLs, a trusted signer must appear in this list with at least one key pair ID.
additional authenticated data	Information that is checked for integrity but not encrypted, such as headers or other contextual metadata.
administrative suspension	Auto Scaling (p. 280) might suspend processes for Auto Scaling group (p. 280) that repeatedly fail to launch instances. Auto Scaling groups that most commonly experience administrative suspension have zero running instances, have been trying to launch instances for more than 24 hours, and have not succeeded in that time.
alarm	An item that watches a single metric over a specified time period, and triggers an Amazon SNS (p. 278) topic (p. 318) or an Auto Scaling (p. 280) policy (p. 306) if the value of the metric crosses a threshold value over a predetermined number of time periods.
allow	One of two possible outcomes (the other is deny (p. 291)) when an IAM (p. 282) access policy (p. 306) is evaluated. When a user makes a request to AWS, AWS evaluates the request based on all permissions that apply to the user and then returns either allow or deny.
Amazon API Gateway	A fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. See Also http://aws.amazon.com/api-gateway.
Amazon AppStream	A web service for streaming existing Windows applications from the cloud to any device.

	See Also http://aws.amazon.com/appstream/.
Amazon Aurora	A fully managed MySQL-compatible relational database engine that combines the speed and availability of commercial databases with the simplicity and cost-effectiveness of open source databases. See Also http://aws.amazon.com/rds/aurora/.
Amazon CloudFront	An AWS content delivery service that helps you improve the performance, reliability, and availability of your websites and applications. See Also http://aws.amazon.com/cloudfront.
Amazon CloudSearch	A fully managed service in the AWS cloud that makes it easy to set up, manage, and scale a search solution for your website or application.
Amazon CloudWatch	A web service that enables you to monitor and manage various metrics, and configure alarm actions based on data from those metrics. See Also http://aws.amazon.com/cloudwatch.
Amazon CloudWatch Events	A web service that enables you to deliver a timely stream of system events that describe changes in AWS resource (p. 310)s to AWS Lambda (p. 282) functions, streams in Amazon Kinesis Streams (p. 277), Amazon Simple Notification Service (p. 278) topics, or built-in targets. See Also http://aws.amazon.com/cloudwatch.
Amazon CloudWatch Logs	A web service for monitoring and troubleshooting your systems and applications from your existing system, application, and custom log files. You can send your existing log files to CloudWatch Logs and monitor these logs in near real-time. See Also http://aws.amazon.com/cloudwatch.
Amazon Cognito	A web service that makes it easy to save mobile user data, such as app preferences or game state, in the AWS cloud without writing any back-end code or managing any infrastructure. Amazon Cognito offers mobile identity management and data synchronization across devices. See Also http://aws.amazon.com/cognito/.
Amazon DevPay	An easy-to-use online billing and account management service that makes it easy for you to sell an Amazon EC2 (p. 276) AMI (p. 277) or an application built on Amazon S3 (p. 278). See Also http://aws.amazon.com/devpay.
Amazon DynamoDB	A fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. See Also http://aws.amazon.com/dynamodb/.
Amazon DynamoDB Storage Backend for Titan	A storage backend for the Titan graph database implemented on top of Amazon DynamoDB. Titan is a scalable graph database optimized for storing and querying graphs. See Also http://aws.amazon.com/dynamodb/.
Amazon DynamoDB Streams	An AWS service that captures a time-ordered sequence of item-level modifications in any Amazon DynamoDB table, and stores this information in a log for up to 24 hours. Applications can access this log and view the data items as they appeared before and after they were modified, in near real time. See Also http://aws.amazon.com/dynamodb/.
Amazon Elastic Block Store (Amazon EBS)	A service that provides block level storage volume (p. 320)s for use with EC2 instance (p. 292)s. See Also http://aws.amazon.com/ebs.

Amazon EBS-backed AMI	A type of Amazon Machine Image (AMI) (p. 277) whose instance (p. 298)s use an Amazon EBS (p. 275) volume (p. 320) as their root device. Compare this with instances launched from instance store-backed AMI (p. 298)s, which use the instance store (p. 298) as the root device.
Amazon EC2 Container Registry (Amazon ECR)	A fully managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon EC2 Container Service (Amazon ECS) (p. 276) and AWS Identity and Access Management (IAM) (p. 282). See Also http://aws.amazon.com/ecr.
Amazon EC2 Container Service (Amazon ECS)	A highly scalable, fast, container (p. 288) management service that makes it easy to run, stop, and manage Docker containers on a cluster (p. 287) of EC2 instance (p. 292)s. See Also http://aws.amazon.com/ecs.
Amazon ECS service	A service for running and maintaining a specified number of task (p. 318)s (instantiations of a task definition (p. 318)) simultaneously.
Amazon EC2 VM Import Connector	See http://aws.amazon.com/ec2/vm-import.
Amazon Elastic Compute Cloud (Amazon EC2)	A web service that enables you to launch and manage Linux/UNIX and Windows server instance (p. 298)s in Amazon's data centers. See Also http://aws.amazon.com/ec2.
Amazon Elastic File System (Amazon EFS)	A file storage service for EC2 (p. 276) instance (p. 298)s. Amazon EFS is easy to use and provides a simple interface with which you can create and configure file systems. Amazon EFS storage capacity grows and shrinks automatically as you add and remove files. See Also http://aws.amazon.com/efs/.
Amazon EMR (Amazon EMR)	A web service that makes it easy to process large amounts of data efficiently. Amazon EMR uses Hadoop (p. 296) processing combined with several AWS products to do such tasks as web indexing, data mining, log file analysis, machine learning, scientific simulation, and data warehousing. See Also http://aws.amazon.com/elasticmapreduce.
Amazon Elastic Transcoder	A cloud-based media transcoding service. Elastic Transcoder is a highly scalable tool for converting (or *transcoding*) media files from their source format into versions that will play on devices like smartphones, tablets, and PCs. See Also http://aws.amazon.com/elastictranscoder/.
Amazon ElastiCache	A web service that simplifies deploying, operating, and scaling an in-memory cache in the cloud. The service improves the performance of web applications by providing information retrieval from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases. See Also http://aws.amazon.com/elasticache/.
Amazon Elasticsearch Service (Amazon ES)	An AWS-managed service for deploying, operating, and scaling Elasticsearch, an open-source search and analytics engine, in the AWS Cloud. Amazon Elasticsearch Service (Amazon ES) also offers security options, high availability, data durability, and direct access to the Elasticsearch APIs. See Also http://aws.amazon.com/elasticsearch-service.
Amazon GameLift	A managed service for deploying, operating, and scaling session-based multiplayer games. See Also http://aws.amazon.com/gamelift/.
Amazon Glacier	A secure, durable, and low-cost storage service for data archiving and long-term backup. You can reliably store large or small amounts of data for

	significantly less than on-premises solutions. Amazon Glacier is optimized for infrequently accessed data, where a retrieval time of several hours is suitable. See Also http://aws.amazon.com/glacier/.
Amazon Inspector	An automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector automatically assesses applications for vulnerabilities or deviations from best practices. After performing an assessment, Amazon Inspector produces a detailed report with prioritized steps for remediation. See Also http://aws.amazon.com/inspector.
Amazon Kinesis	A platform for streaming data on AWS. Amazon Kinesis offers services that simplify the loading and analysis of streaming data. See Also http://aws.amazon.com/kinesis/.
Amazon Kinesis Firehose	A fully managed service for loading streaming data into AWS. Firehose can capture and automatically load streaming data into Amazon S3 (p. 278) and Amazon Redshift (p. 277), enabling near real-time analytics with existing business intelligence tools and dashboards. Firehose automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, and encrypt the data before loading it. See Also http://aws.amazon.com/kinesis/firehose/.
Amazon Kinesis Streams	A web service for building custom applications that process or analyze streaming data for specialized needs. Amazon Kinesis Streams can continuously capture and store terabytes of data per hour from hundreds of thousands of sources. See Also http://aws.amazon.com/kinesis/streams/.
Amazon Lumberyard	A cross-platform, 3D game engine for creating high-quality games. You can connect games to the compute and storage of the AWS cloud and engage fans on Twitch. See Also http://aws.amazon.com/lumberyard/.
Amazon Machine Image (AMI)	An encrypted machine image stored in Amazon Elastic Block Store (Amazon EBS) (p. 275) or Amazon Simple Storage Service (p. 278). AMIs are like a template of a computer's root drive. They contain the operating system and can also include software and layers of your application, such as database servers, middleware, web servers, and so on.
Amazon Machine Learning	A cloud-based service that creates machine learning (ML) models by finding patterns in your data, and uses these models to process new data and generate predictions. See Also http://aws.amazon.com/machine-learning/.
Amazon ML	See Amazon Machine Learning.
Amazon Mobile Analytics	A service for collecting, visualizing, understanding, and extracting mobile app usage data at scale. See Also http://aws.amazon.com/mobileanalytics.
Amazon Redshift	A fully managed, petabyte-scale data warehouse service in the cloud. With Amazon Redshift you can analyze your data using your existing business intelligence tools. See Also http://aws.amazon.com/redshift/.
Amazon Relational Database Service (Amazon RDS)	A web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

	See Also http://aws.amazon.com/rds.
Amazon Resource Name (ARN)	A standardized way to refer to an AWS resource (p. 310). For example: arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Bob.
Amazon Route 53	A web service you can use to create a new DNS service or to migrate your existing DNS service to the cloud. See Also http://aws.amazon.com/route53.
Amazon S3	See Amazon Simple Storage Service (Amazon S3).
Amazon S3-Backed AMI	See instance store-backed AMI.
Amazon Silk	A next-generation web browser available only on Fire OS tablets and phones. Built on a split architecture that divides processing between the client and the AWS cloud, Amazon Silk is designed to create a faster, more responsive mobile browsing experience.
Amazon Simple Email Service (Amazon SES)	An easy-to-use, cost-effective email solution for applications. See Also http://aws.amazon.com/ses.
Amazon Simple Notification Service (Amazon SNS)	A web service that enables applications, end-users, and devices to instantly send and receive notifications from the cloud. See Also http://aws.amazon.com/sns.
Amazon Simple Queue Service (Amazon SQS)	Reliable and scalable hosted queues for storing messages as they travel between computers. See Also http://aws.amazon.com/sqs.
Amazon Simple Storage Service (Amazon S3)	Storage for the Internet. You can use it to store and retrieve any amount of data at any time, from anywhere on the web. See Also http://aws.amazon.com/s3.
Amazon Simple Workflow Service (Amazon SWF)	A fully managed service that helps developers build, run, and scale background jobs that have parallel or sequential steps. Amazon SWF is like a state tracker and task coordinator in the cloud. See Also http://aws.amazon.com/swf/.
Amazon Virtual Private Cloud (Amazon VPC)	A web service for provisioning a logically isolated section of the AWS cloud where you can launch AWS resource (p. 310)s in a virtual network that you define. You control your virtual networking environment, including selection of your own IP address range, creation of subnet (p. 316)s, and configuration of route table (p. 311)s and network gateways. See Also http://aws.amazon.com/vpc.
Amazon VPC	See Amazon Virtual Private Cloud (Amazon VPC).
Amazon Web Services (AWS)	An infrastructure web services platform in the cloud for companies of all sizes. See Also http://aws.amazon.com/what-is-cloud-computing/.
Amazon WorkDocs	A managed, secure enterprise document storage and sharing service with administrative controls and feedback capabilities. See Also http://aws.amazon.com/workdocs/.
Amazon WorkMail	A managed, secure business email and calendar service with support for existing desktop and mobile email clients. See Also http://aws.amazon.com/workmail/.
Amazon WorkSpaces	A managed, secure desktop computing service for provisioning cloud-based desktops and providing users access to documents, applications, and resource (p. 310)s from supported devices. See Also http://aws.amazon.com/workspaces/.

Amazon WorkSpaces Application Manager (Amazon WAM)	A web service for deploying and managing applications for Amazon WorkSpaces. Amazon WAM accelerates software deployment, upgrades, patching, and retirement by packaging Windows desktop applications into virtualized application containers. See Also http://aws.amazon.com/workspaces/applicationmanager.
AMI	See Amazon Machine Image (AMI).
analysis scheme	Amazon CloudSearch (p. 275): Language-specific text analysis options that are applied to a text field to control stemming and configure stopwords and synonyms.
application	AWS Elastic Beanstalk (p. 281): A logical collection of components, including environments, versions, and environment configurations. An application is conceptually similar to a folder. AWS CodeDeploy (p. 281): A name that uniquely identifies the application to be deployed. AWS CodeDeploy uses this name to ensure the correct combination of revision, deployment configuration, and deployment group are referenced during a deployment.
Application Billing	The location where your customers manage the Amazon DevPay products they've purchased. The web address is http://www.amazon.com/dp-applications.
application revision	AWS CodeDeploy (p. 281): An archive file containing source content—such as source code, web pages, executable files, and deployment scripts—along with an application specification file (p. 279). Revisions are stored in Amazon S3 (p. 278) bucket (p. 285)s or GitHub repositories. For Amazon S3, a revision is uniquely identified by its Amazon S3 object key and its ETag, version, or both. For GitHub, a revision is uniquely identified by its commit ID.
application specification file	AWS CodeDeploy (p. 281): A YAML-formatted file used to map the source files in an application revision to destinations on the instance; specify custom permissions for deployed files; and specify scripts to be run on each instance at various stages of the deployment process.
application version	AWS Elastic Beanstalk (p. 281): A specific, labeled iteration of an application that represents a functionally consistent set of deployable application code. A version points to an Amazon S3 (p. 278) object (a JAVA WAR file) that contains the application code.
AppSpec file	See application specification file.
AUC	Area Under a Curve. An industry-standard metric to evaluate the quality of a binary classification machine learning model. AUC measures the ability of the model to predict a higher score for positive examples, those that are “correct,” than for negative examples, those that are “incorrect.” The AUC metric returns a decimal value from 0 to 1. AUC values near 1 indicate an ML model that is highly accurate.
ARN	See Amazon Resource Name (ARN).
artifact	AWS CodePipeline (p. 281): A copy of the files or changes that will be worked upon by the pipeline.
asymmetric encryption	Encryption (p. 293) that uses both a public key and a private key.
asynchronous bounce	A type of bounce (p. 285) that occurs when a receiver (p. 309) initially accepts an email message for delivery and then subsequently fails to deliver it.

atomic counter	DynamoDB: A method of incrementing or decrementing the value of an existing attribute without interfering with other write requests.
attribute	A fundamental data element, something that does not need to be broken down any further. In DynamoDB, attributes are similar in many ways to fields or columns in other database systems. Amazon Machine Learning: A unique, named property within an observation in a data set. In tabular data, such as spreadsheets or comma-separated values (.csv) files, the column headings represent the attributes, and the rows contain values for each attribute.
Aurora	See Amazon Aurora.
authenticated encryption	Encryption (p. 293) that provides confidentiality, data integrity, and authenticity assurances of the encrypted data.
authentication	The process of proving your identity to a system.
Auto Scaling	A web service designed to launch or terminate instance (p. 298)s automatically based on user-defined policies (p. 306), schedules, and health check (p. 296)s. See Also http://aws.amazon.com//autoscaling.
Auto Scaling group	A representation of multiple EC2 instance (p. 292)s that share similar characteristics, and that are treated as a logical grouping for the purposes of instance scaling and management.
Availability Zone	A distinct location within a region (p. 309) that is insulated from failures in other Availability Zones, and provides inexpensive, low-latency network connectivity to other Availability Zones in the same region.
AWS	See Amazon Web Services (AWS).
AWS Application Discovery Service	A web service that helps you plan to migrate to AWS by identifying IT assets in a data center—including servers, virtual machines, applications, application dependencies, and network infrastructure. See Also http://aws.amazon.com/about-aws/whats-new/2016/04/aws-application-discovery-service/.
AWS Billing and Cost Management	The AWS cloud computing model in which you pay for services on demand and use as much or as little at any given time as you need. While resource (p. 310)s are active under your account, you pay for the cost of allocating those resources and for any incidental usage associated with those resources, such as data transfer or allocated storage. See Also http://aws.amazon.com/billing/new-user-faqs/.
AWS Certificate Manager (ACM)	A web service for provisioning, managing, and deploying Secure Sockets Layer/Transport Layer Security (p. 319) (SSL/TLS) certificates for use with AWS services. See Also http://aws.amazon.com/certificate-manager/.
AWS CloudFormation	A service for writing or changing templates that create and delete related AWS resource (p. 310)s together as a unit. See Also http://aws.amazon.com/cloudformation.
AWS CloudHSM	A web service that helps you meet corporate, contractual, and regulatory compliance requirements for data security by using dedicated hardware security module (HSM) appliances within the AWS cloud. See Also http://aws.amazon.com/cloudhsm/.

AWS CloudTrail	A web service that records AWS API calls for your account and delivers log files to you. The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service. See Also http://aws.amazon.com/cloudtrail/.
AWS CodeCommit	A fully managed source control service that makes it easy for companies to host secure and highly scalable private Git repositories. See Also http://aws.amazon.com/codecommit.
AWS CodeDeploy	A service that automates code deployments to any instance, including EC2 instance (p. 292)s and instance (p. 298)s running on-premises. See Also http://aws.amazon.com/codedeploy.
AWS CodeDeploy agent	A software package that, when installed and configured on an instance, enables that instance to be used in AWS CodeDeploy deployments.
AWS CodePipeline	A continuous delivery service for fast and reliable application updates. See Also http://aws.amazon.com/codepipeline.
AWS Command Line Interface (AWS CLI)	A unified downloadable and configurable tool for managing AWS services. Control multiple AWS services from the command line and automate them through scripts. See Also http://aws.amazon.com/cli/.
AWS Config	A fully managed service that provides an AWS resource (p. 310) inventory, configuration history, and configuration change notifications for better security and governance. You can create rules that automatically check the configuration of AWS resources that AWS Config records. See Also http://aws.amazon.com/config/.
AWS Database Migration Service	A web service that can help you migrate data to and from many widely used commercial and open-source databases. See Also http://aws.amazon.com/dms.
AWS Data Pipeline	A web service for processing and moving data between different AWS compute and storage services, as well as on-premises data sources, at specified intervals. See Also http://aws.amazon.com/datapipeline.
AWS Device Farm	An app testing service that allows developers to test Android, iOS, and Fire OS devices on real, physical phones and tablets that are hosted by AWS. See Also http://aws.amazon.com/device-farm.
AWS Direct Connect	A web service that simplifies establishing a dedicated network connection from your premises to AWS. Using AWS Direct Connect, you can establish private connectivity between AWS and your data center, office, or colocation environment. See Also http://aws.amazon.com/directconnect.
AWS Directory Service	A managed service for connecting your AWS resource (p. 310)s to an existing on-premises Microsoft Active Directory or to set up and operate a new, standalone directory in the AWS cloud. See Also http://aws.amazon.com/directoryservice.
AWS Elastic Beanstalk	A web service for deploying and managing applications in the AWS cloud without worrying about the infrastructure that runs those applications. See Also http://aws.amazon.com/elasticbeanstalk.
AWS GovCloud (US)	An isolated AWS Region designed to host sensitive workloads in the cloud, ensuring that this work meets the US government's regulatory and

	compliance requirements. The AWS GovCloud (US) Region adheres to United States International Traffic in Arms Regulations (ITAR), Federal Risk and Authorization Management Program (FedRAMP) requirements, Department of Defense (DOD) Cloud Security Requirements Guide (SRG) Levels 2 and 4, and Criminal Justice Information Services (CJIS) Security Policy requirements. See Also http://aws.amazon.com/govcloud-us/.
AWS Identity and Access Management (IAM)	A web service that enables Amazon Web Services (AWS) (p. 278) customers to manage users and user permissions within AWS. See Also http://aws.amazon.com/iam.
AWS Import/Export	A service for transferring large amounts of data between AWS and portable storage devices. See Also http://aws.amazon.com/importexport.
AWS IoT	A managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices. See Also http://aws.amazon.com/iot.
AWS Key Management Service (AWS KMS)	A managed service that simplifies the creation and control of encryption (p. 293) keys that are used to encrypt data. See Also http://aws.amazon.com/kms.
AWS Lambda	A web service that lets you run code without provisioning or managing servers. You can run code for virtually any type of application or back-end service with zero administration. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app. See Also http://aws.amazon.com/lambda/.
AWS managed key	One of two types of customer master key (CMK) (p. 289)s in AWS Key Management Service (AWS KMS) (p. 282).
AWS managed policy	An IAM (p. 282) managed policy (p. 301) that is created and managed by AWS.
AWS Management Console	A graphical interface to manage compute, storage, and other cloud resource (p. 310)s. See Also http://aws.amazon.com/console.
AWS Management Portal for vCenter	A web service for managing your AWS resource (p. 310)s using VMware vCenter. You install the portal as a vCenter plug-in within your existing vCenter environment. Once installed, you can migrate VMware VMs to Amazon EC2 (p. 276) and manage AWS resources from within vCenter. See Also http://aws.amazon.com/ec2/vcenter-portal/.
AWS Marketplace	A web portal where qualified partners to market and sell their software to AWS customers. AWS Marketplace is an online software store that helps customers find, buy, and immediately start using the software and services that run on AWS. See Also http://aws.amazon.com/partners/aws-marketplace/.
AWS Mobile Hub	An integrated console that for building, testing, and monitoring mobile apps. See Also http://aws.amazon.com/mobile.
AWS Mobile SDK	A software development kit whose libraries, code samples, and documentation help you build high quality mobile apps for the iOS, Android, Fire OS, Unity, and Xamarin platforms. See Also http://aws.amazon.com/mobile/sdk.
AWS OpsWorks	A configuration management service that helps you use Chef to configure and operate groups of instances and applications. You can define the application's

	architecture and the specification of each component including package installation, software configuration, and resource (p. 310)s such as storage. You can automate tasks based on time, load, lifecycle events, and more. See Also http://aws.amazon.com/opsworks/.
AWS SDK for Go	A software development kit for integrating your Go application with the full suite of AWS services. See Also http://aws.amazon.com/sdk-for-go/.
AWS SDK for Java	A software development kit that provides Java APIs for many AWS services including Amazon S3 (p. 278), Amazon EC2 (p. 276), Amazon DynamoDB (p. 275), and more. The single, downloadable package includes the AWS Java library, code samples, and documentation. See Also http://aws.amazon.com/sdkforjava/.
AWS SDK for JavaScript in the Browser	A software development kit for accessing AWS services from JavaScript code running in the browser. Authenticate users through Facebook, Google, or Login with Amazon using web identity federation. Store application data in Amazon DynamoDB (p. 275), and save user files to Amazon S3 (p. 278). See Also http://aws.amazon.com/sdk-for-browser/.
AWS SDK for JavaScript in Node.js	A software development kit for accessing AWS services from JavaScript in Node.js. The SDK provides JavaScript objects for AWS services, including Amazon S3 (p. 278), Amazon EC2 (p. 276), Amazon DynamoDB (p. 275), and Amazon Simple Workflow Service (Amazon SWF) (p. 278) . The single, downloadable package includes the AWS JavaScript library and documentation. See Also http://aws.amazon.com/sdk-for-node-js/.
AWS SDK for .NET	A software development kit that provides .NET API actions for AWS services including Amazon S3 (p. 278), Amazon EC2 (p. 276), IAM (p. 282), and more. You can download the SDK as multiple service-specific packages on NuGet. See Also http://aws.amazon.com/sdkfornet/.
AWS SDK for PHP	A software development kit and open-source PHP library for integrating your PHP application with AWS services like Amazon S3 (p. 278), Amazon Glacier (p. 276), and Amazon DynamoDB (p. 275). See Also http://aws.amazon.com/sdkforphp/.
AWS SDK for Python (Boto)	A software development kit for using Python to access AWS services like Amazon EC2 (p. 276), Amazon EMR (p. 276), Auto Scaling (p. 280), Amazon Kinesis (p. 277), AWS Lambda (p. 282), and more. See Also http://boto.readthedocs.org/en/latest/.
AWS SDK for Ruby	A software development kit for accessing AWS services from Ruby. The SDK provides Ruby classes for many AWS services including Amazon S3 (p. 278), Amazon EC2 (p. 276), Amazon DynamoDB (p. 275). and more. The single, downloadable package includes the AWS Ruby Library and documentation. See Also http://aws.amazon.com/sdkforruby/.
AWS Security Token Service (AWS STS)	A web service for requesting temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) (p. 282) users or for users that you authenticate (federated users (p. 295)). See Also http://aws.amazon.com/iam/.
AWS Service Catalog	A web service that helps organizations create and manage catalogs of IT services that are approved for use on AWS. These IT services can include everything from virtual machine images, servers, software, and databases to complete multitier application architectures.

	See Also http://aws.amazon.com/servicecatalog/.
AWS Storage Gateway	A web service that connects an on-premises software appliance with cloud-based storage to provide seamless and secure integration between an organization's on-premises IT environment and AWS's storage infrastructure. See Also http://aws.amazon.com/storagegateway/.
AWS Toolkit for Eclipse	An open-source plug-in for the Eclipse Java IDE that makes it easier for developers to develop, debug, and deploy Java applications using Amazon Web Services. See Also http://aws.amazon.com/eclipse/.
AWS Toolkit for Visual Studio	An extension for Microsoft Visual Studio that helps developers develop, debug, and deploy .NET applications using Amazon Web Services. See Also http://aws.amazon.com/visualstudio/.
AWS Tools for Windows PowerShell	A set of PowerShell cmdlets to help developers and administrators manage their AWS services from the Windows PowerShell scripting environment. See Also http://aws.amazon.com/powershell/.
AWS Trusted Advisor	A web service that inspects your AWS environment and makes recommendations for saving money, improving system availability and performance, and helping to close security gaps. See Also http://aws.amazon.com/premiumsupport/trustedadvisor/.
AWS VPN CloudHub	Enables secure communication between branch offices using a simple hub-and-spoke model, with or without a VPC (p. 320).
AWS WAF	A web application firewall service that controls access to content by allowing or blocking web requests based on criteria that you specify, such as header values or the IP addresses that the requests originate from. AWS WAF helps protect web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources. See Also http://aws.amazon.com/waf/.

B

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

basic monitoring	Monitoring of AWS-provided metrics derived at a 5-minute frequency.
batch	See document batch.
BGP ASN	Border Gateway Protocol Autonomous System Number. A unique identifier for a network, for use in BGP routing. Amazon EC2 (p. 276) supports all 2-byte ASN numbers in the range of 1 – 65335, with the exception of 7224, which is reserved.
batch prediction	Amazon Machine Learning: An operation that processes multiple input data observations at one time (asynchronously). Unlike real-time predictions, batch predictions are not available until all predictions have been processed. See Also real-time prediction.
billing	See AWS Billing and Cost Management.
binary attribute	Amazon Machine Learning: An attribute for which one of two possible values is possible. Valid positive values are 1, y, yes, t, and true answers. Valid negative

	values are 0, n, no, f, and false. Amazon Machine Learning outputs 1 for positive values and 0 for negative values. See Also attribute.
binary classification model	Amazon Machine Learning: A machine learning model that predicts the answer to questions where the answer can be expressed as a binary variable. For example, questions with answers of “1” or “0”, “yes” or “no”, “will click” or “will not click” are questions that have binary answers. The result for a binary classification model is always either a “1” (for a “true” or affirmative answers) or a “0” (for a “false” or negative answers).
blacklist	A list of IP addresses, email addresses, or domains that an Internet service provider (p. 298) suspects to be the source of spam (p. 315). The ISP blocks incoming email from these addresses or domains.
block	A data set. Amazon EMR (p. 276) breaks large amounts of data into subsets. Each subset is called a data block. Amazon EMR assigns an ID to each block and uses a hash table to keep track of block processing.
block device	A storage device that supports reading and (optionally) writing data in fixed-size blocks, sectors, or clusters.
block device mapping	A mapping structure for every AMI (p. 277) and instance (p. 298) that specifies the block devices attached to the instance.
bootstrap action	A user-specified default or custom action that runs a script or an application on all nodes of a job flow before Hadoop (p. 296) starts.
Border Gateway Protocol Autonomous System Number	See BGP ASN.
bounce	A failed email delivery attempt.
breach	Auto Scaling (p. 280): The condition in which a user-set threshold (upper or lower boundary) is passed. If the duration of the breach is significant, as set by a breach duration parameter, it can possibly start a scaling activity (p. 312).
bucket	Amazon Simple Storage Service (Amazon S3) (p. 278): A container for stored objects. Every object is contained in a bucket. For example, if the object named `photos/puppy.jpg` is stored in the `johnsmith` bucket, then authorized users can access the object with the URL `http://johnsmith.s3.amazonaws.com/photos/puppy.jpg`.
bucket owner	The person or organization that owns a bucket (p. 285) in Amazon S3 (p. 278). Just as Amazon is the only owner of the domain name Amazon.com, only one person or organization can own a bucket.
bundling	A commonly used term for creating an Amazon Machine Image (AMI) (p. 277). It specifically refers to creating instance store-backed AMI (p. 298)s.

C

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

cache cluster	A logical cache distributed over multiple cache node (p. 286)s. A cache cluster can be set up with a specific number of cache nodes.

cache cluster identifier	Customer-supplied identifier for the cache cluster that must be unique for that customer in an AWS region (p. 309).
cache engine version	The version of the Memcached service that is running on the cache node.
cache node	A fixed-size chunk of secure, network-attached RAM. Each cache node runs an instance of the Memcached service, and has its own DNS name and port. Multiple types of cache nodes are supported, each with varying amounts of associated memory.
cache node type	An EC2 instance (p. 292) type used to run the cache node.
cache parameter group	A container for cache engine parameter values that can be applied to one or more cache clusters.
cache security group	A group maintained by ElastiCache that combines ingress authorizations to cache nodes for hosts belonging to Amazon EC2 (p. 276) security group (p. 313)s specified through the console or the API or command line tools.
canned access policy	A standard access control policy that you can apply to a bucket (p. 285) or object. Options include: private, public-read, public-read-write, and authenticated-read.
canonicalization	The process of converting data into a standard format that a service such as Amazon S3 (p. 278) can recognize.
capacity	The amount of available compute size at a given time. Each Auto Scaling group (p. 280) is defined with a minimum and maximum compute size. A scaling activity (p. 312) increases or decreases the capacity within the defined minimum and maximum values.
cartesian product processor	A processor that calculates a cartesian product. Also known as a *cartesian data processor*.
cartesian product	A mathematical operation that returns a product from multiple sets.
certificate	A credential that some AWS products use to authenticate AWS account (p. 274)s and users. Also known as an X.509 certificate (p. 321) . The certificate is paired with a private key.
chargeable resources	Features or services whose use incurs fees. Although some AWS products are free, others include charges. For example, in an AWS CloudFormation (p. 280) stack (p. 315), AWS resource (p. 310)s that have been created incur charges. The amount charged depends on the usage load. Use the Amazon Web Services Simple Monthly Calculator at http://calculator.s3.amazonaws.com/calc5.html to estimate your cost prior to creating instances, stacks, or other resources.
CIDR block	Classless Inter-Domain Routing. An Internet protocol address allocation and route aggregation methodology. See Also Classless Inter-Domain Routing in Wikipedia.
ciphertext	Information that has been encrypted (p. 293), as opposed to plaintext (p. 306), which is information that has not.
ClassicLink	A feature for linking an EC2-Classic instance (p. 298) to a VPC (p. 320), allowing your EC2-Classic instance to communicate with VPC instances using private IP addresses. See Also link to VPC, unlink from VPC.

classification	In machine learning, a type of problem that seeks to place (classify) a data sample into a single category or “class.” Often, classification problems are modeled to choose one category (class) out of two. These are binary classification problems. Problems where more than two categories (classes) are available are called "multiclass classification" problems. See Also binary classification model, multiclass classification model.
cloud service provider	A company that provides subscribers with access to Internet-hosted computing, storage, and software services.
CloudHub	See AWS VPN CloudHub.
CLI	See AWS Command Line Interface (AWS CLI).
cluster	A logical grouping of container instance (p. 288)s that you can place task (p. 318)s on. Amazon Elasticsearch Service (Amazon ES) (p. 276): A logical grouping of one or more data nodes, optional dedicated master nodes, and storage required to run Amazon Elasticsearch Service (Amazon ES) and operate your Amazon ES domain. See Also data node, dedicated master node, node.
cluster compute instance	A type of instance (p. 298) that provides a great amount of CPU power coupled with increased networking performance, making it well suited for High Performance Compute (HPC) applications and other demanding network-bound applications.
cluster placement group	A logical cluster compute instance (p. 287) grouping to provide lower latency and high-bandwidth connectivity between the instance (p. 298)s.
cluster status	Amazon Elasticsearch Service (Amazon ES) (p. 276): An indicator of the health of a cluster. A status can be green, yellow, or red. At the shard level, green means that all shards are allocated to nodes in a cluster, yellow means that the primary shard is allocated but the replica shards are not, and red means that the primary and replica shards of at least one index are not allocated. The shard status determines the index status, and the index status determines the cluster status.
CMK	See customer master key (CMK).
CNAME	Canonical Name Record. A type of resource record (p. 310) in the Domain Name System (DNS) that specifies that the domain name is an alias of another, canonical domain name. More simply, it is an entry in a DNS table that lets you alias one fully qualified domain name to another.
complaint	The event in which a recipient (p. 309) who does not want to receive an email message clicks "Mark as Spam" within the email client, and the Internet service provider (p. 298) sends a notification to Amazon SES (p. 278).
compound query	Amazon CloudSearch (p. 275): A search request that specifies multiple search criteria using the Amazon CloudSearch structured search syntax.
condition	IAM (p. 282): Any restriction or detail about a permission. The condition is *D* in the statement "A has permission to do B to C where D applies." AWS WAF (p. 284): A set of attributes that AWS WAF searches for in web requests to AWS resource (p. 310)s such as Amazon CloudFront (p. 275) distributions. Conditions can include values such as the IP addresses that web requests originate from or values in request headers. Based on the specified

	conditions, you can configure AWS WAF to allow or block web requests to AWS resources.
conditional parameter	See mapping.
configuration API	Amazon CloudSearch (p. 275): The API call that you use to create, configure, and manage search domains.
configuration template	A series of key–value pairs that define parameters for various AWS products so that AWS Elastic Beanstalk (p. 281) can provision them for an environment.
consistency model	The method a service uses to achieve high availability. For example, it could involve replicating data across multiple servers in a data center. See Also eventual consistency.
console	See AWS Management Console.
consolidated billing	A feature of the AWS Billing and Cost Management (p. 280) service for consolidating payment for multiple AWS accounts within your company by designating a single paying account. You can see a combined view of AWS costs incurred by all accounts, as well as obtain a detailed cost report for each of the individual AWS accounts associated with your paying account. Consolidated billing is offered at no additional charge.
container	A Linux container that was created from a Docker image as part of a task (p. 318).
container definition	Specifies which Docker image (p. 291) to use for a container (p. 288), how much CPU and memory the container is allocated, and more options. The container definition is included as part of a task definition (p. 318).
container instance	An EC2 instance (p. 292) that is running the Amazon EC2 Container Service (Amazon ECS) (p. 276) agent and has been registered into a cluster (p. 287). Amazon ECS task (p. 318)s are placed on active container instances.
container registry	Stores, manages, and deploys Docker image (p. 291)s.
continuous delivery	A software development practice in which code changes are automatically built, tested, and prepared for a release to production. See Also http://aws.amazon.com/devops/continuous-delivery/.
continuous integration	A software development practice in which developers regularly merge code changes into a central repository, after which automated builds and tests are run. See Also http://aws.amazon.com/devops/continuous-integration/.
cooldown period	Amount of time during which Auto Scaling (p. 280) does not allow the desired size of the Auto Scaling group (p. 280) to be changed by any other notification from an Amazon CloudWatch (p. 275) alarm (p. 274).
core node	An EC2 instance (p. 292) that runs Hadoop (p. 296) map and reduce tasks and stores data using the Hadoop Distributed File System (HDFS). Core nodes are managed by the master node (p. 302), which assigns Hadoop tasks to nodes and monitors their status. The EC2 instances you assign as core nodes are capacity that must be allotted for the entire job flow run. Because core nodes store data, you can't remove them from a job flow. However, you can add more core nodes to a running job flow. Core nodes run both the DataNodes and TaskTracker Hadoop daemons.
corpus	Amazon CloudSearch (p. 275): A collection of data that you want to search.

credential helper — AWS CodeCommit (p. 281): A program that stores credentials for repositories and supplies them to Git when making connections to those repositories. The AWS CLI (p. 281) includes a credential helper that you can use with Git when connecting to AWS CodeCommit repositories.

credentials — Also called *access credentials* or *security credentials*. In authentication and authorization, a system uses credentials to identify who is making a call and whether to allow the requested access. In AWS, these credentials are typically the access key ID (p. 273) and the secret access key (p. 313).

cross-account access — The process of permitting limited, controlled use of resource (p. 310)s in one AWS account (p. 274) by a user in another AWS account. For example, in AWS CodeCommit (p. 281) and AWS CodeDeploy (p. 281) you can configure cross-account access so that a user in AWS account A can access an AWS CodeCommit repository created by account B. Or a pipeline in AWS CodePipeline (p. 281) created by account A can use AWS CodeDeploy resources created by account B. In IAM (p. 282) you use a role (p. 311) to delegate (p. 290) temporary access to a user (p. 319) in one account to resources in another.

cross-region replication — A client-side solution for maintaining identical copies of Amazon DynamoDB (p. 275) tables across different AWS region (p. 309)s, in near real time.

customer gateway — A router or software application on your side of a VPN tunnel that is managed by Amazon VPC (p. 278). The internal interfaces of the customer gateway are attached to one or more devices in your home network. The external interface is attached to the VPG (p. 320) across the VPN tunnel.

customer managed policy — An IAM (p. 282) managed policy (p. 301) that you create and manage in your AWS account (p. 274).

customer master key (CMK) — The fundamental resource (p. 310) that AWS Key Management Service (AWS KMS) (p. 282) manages. CMKs can be either customer-managed keys or AWS-managed keys. Use CMKs inside AWS KMS to encrypt (p. 293) or decrypt up to 4 kilobytes of data directly or to encrypt generated data keys, which are then used to encrypt or decrypt larger amounts of data outside of the service.

D

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

dashboard — See service health dashboard.

data consistency — A concept that describes when data is written or updated successfully and all copies of the data are updated in all AWS region (p. 309)s. However, it takes time for the data to propagate to all storage locations. To support varied application requirements, Amazon DynamoDB (p. 275) supports both eventually consistent and strongly consistent reads.
See Also eventual consistency, eventually consistent read, strongly consistent read.

data node — Amazon Elasticsearch Service (Amazon ES) (p. 276): An Elasticsearch instance that holds data and responds to data upload requests.

	See Also dedicated master node, node.
data schema	See schema.
data source	The database, file, or repository that provides information required by an application or database. For example, in AWS OpsWorks (p. 282), valid data sources include an instance (p. 298) for a stack's MySQL layer or a stack's Amazon RDS (p. 277) service layer. In Amazon Redshift (p. 277), valid data sources include text files in an Amazon S3 (p. 278) bucket (p. 285), in an Amazon EMR (p. 276) cluster, or on a remote host that a cluster can access through an SSH connection. See Also datasource.
database engine	The database software and version running on the DB instance (p. 290).
database name	The name of a database hosted in a DB instance (p. 290). A DB instance can host multiple databases, but databases hosted by the same DB instance must each have a unique name within that instance.
datasource	Amazon Machine Learning (p. 277): An object that contains metadata about the input data. Amazon ML reads the input data, computes descriptive statistics on its attributes, and stores the statistics—along with a schema and other information—as part of the datasource object. Amazon ML uses datasources to train and evaluate a machine learning model and generate batch predictions. See Also data source.
DB compute class	Size of the database compute platform used to run the instance.
DB instance	An isolated database environment running in the cloud. A DB instance can contain multiple user-created databases.
DB instance identifier	User-supplied identifier for the DB instance. The identifier must be unique for that user in an AWS region (p. 309).
DB parameter group	A container for database engine parameter values that apply to one or more DB instance (p. 290)s.
DB security group	A method that controls access to the DB instance (p. 290). By default, network access is turned off to DB instances. After ingress is configured for a security group (p. 313), the same rules apply to all DB instances associated with that group.
DB snapshot	A user-initiated point backup of a DB instance (p. 290).
Dedicated Host	A physical server with EC2 instance (p. 292) capacity fully dedicated to a user.
Dedicated Instance	An instance (p. 298) that is physically isolated at the host hardware level and launched within a VPC (p. 320).
dedicated master node	Amazon Elasticsearch Service (Amazon ES) (p. 276): An Elasticsearch instance that performs cluster management tasks, but does not hold data or respond to data upload requests. Amazon Elasticsearch Service (Amazon ES) uses dedicated master nodes to increase cluster stability. See Also data node, node.
Dedicated Reserved Instance	An option that you purchase to guarantee that sufficient capacity will be available to launch Dedicated Instance (p. 290)s into a VPC (p. 320).
delegation	Within a single AWS account (p. 274): Giving AWS user (p. 319)s access to resource (p. 310)s in your AWS account.

	Between two AWS accounts: Setting up a trust between the account that owns the resource (the trusting account), and the account that contains the users that need to access the resource (the trusted account). See Also trust policy.
delete marker	An object with a key and version ID, but without content. Amazon S3 (p. 278) inserts delete markers automatically into versioned bucket (p. 285)s when an object is deleted.
deliverability	The likelihood that an email message will arrive at its intended destination.
deliveries	The number of email messages, sent through Amazon SES (p. 278), that were accepted by an Internet service provider (p. 298) for delivery to recipient (p. 309)s over a period of time.
deny	The result of a policy (p. 306) statement that includes deny as the effect, so that a specific action or actions are expressly forbidden for a user, group, or role. Explicit deny take precedence over explicit allow (p. 274).
deployment configuration	AWS CodeDeploy (p. 281): A set of deployment rules and success and failure conditions used by the service during a deployment.
deployment group	AWS CodeDeploy (p. 281): A set of individually tagged instance (p. 298)s, EC2 instance (p. 292)s in Auto Scaling group (p. 280)s, or both.
detailed monitoring	Monitoring of AWS-provided metrics derived at a 1-minute frequency.
Description property	A property added to parameters, resource (p. 310)s, resource properties, mappings, and outputs to help you to document AWS CloudFormation (p. 280) template elements.
dimension	A name–value pair (for example, InstanceType=m1.small, or EngineName=mysql), that contains additional information to identify a metric.
discussion forums	A place where AWS users can post technical questions and feedback to help accelerate their development efforts and to engage with the AWS community. The discussion forums are located at http://aws.amazon.com/forums/.
distribution	A link between an origin server (such as an Amazon S3 (p. 278) bucket (p. 285)) and a domain name, which CloudFront (p. 275) automatically assigns. Through this link, CloudFront identifies the object you have stored in your origin server (p. 305).
DKIM	DomainKeys Identified Mail. A standard that email senders use to sign their messages. ISPs use those signatures to verify that messages are legitimate. For more information, see http://www.dkim.org.
DNS	See Domain Name System.
Docker image	A layered file system template that is the basis of a Docker container (p. 288). Docker images can comprise specific operating systems or applications.
document	Amazon CloudSearch (p. 275): An item that can be returned as a search result. Each document has a collection of fields that contain the data that can be searched or returned. The value of a field can be either a string or a number. Each document must have a unique ID and at least one field.
document batch	Amazon CloudSearch (p. 275): A collection of add and delete document operations. You use the document service API to submit batches to update the data in your search domain.

document service API — Amazon CloudSearch (p. 275): The API call that you use to submit document batches to update the data in a search domain.

document service endpoint — Amazon CloudSearch (p. 275): The URL that you connect to when sending document updates to an Amazon CloudSearch domain. Each search domain has a unique document service endpoint that remains the same for the life of the domain.

domain — Amazon Elasticsearch Service (Amazon ES) (p. 276): The hardware, software, and data exposed by Amazon Elasticsearch Service (Amazon ES) endpoints. An Amazon ES domain is a service wrapper around an Elasticsearch cluster. An Amazon ES domain encapsulates the engine instances that process Amazon ES requests, the indexed data that you want to search, snapshots of the domain, access policies, and metadata.
See Also cluster, Elasticsearch.

Domain Name System — A service that routes Internet traffic to websites by translating friendly domain names like www.example.com into the numeric IP addresses like 192.0.2.1 that computers use to connect to each other.

Donation button — An HTML-coded button to provide an easy and secure way for US-based, IRS-certified 501(c)3 nonprofit organizations to solicit donations.

DynamoDB stream — An ordered flow of information about changes to items in anAmazon DynamoDB (p. 275) table. When you enable a stream on a table, DynamoDB captures information about every modification to data items in the table.
See Also Amazon DynamoDB Streams.

E

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

EBS — See Amazon Elastic Block Store (Amazon EBS).

EC2 — See Amazon Elastic Compute Cloud (Amazon EC2).

EC2 compute unit — An AWS standard for compute CPU and memory. You can use this measure to evaluate the CPU capacity of different EC2 instance (p. 292) types.

EC2 instance — A compute instance (p. 298) in the Amazon EC2 (p. 276) service. Other AWS services use the term *EC2 instance* to distinguish these instances from other types of instances they support.

ECR — See Amazon EC2 Container Registry (Amazon ECR).

ECS — See Amazon EC2 Container Service (Amazon ECS).

edge location — A site that CloudFront (p. 275) uses to cache copies of your content for faster delivery to users at any location.

EFS — See Amazon Elastic File System (Amazon EFS).

Elastic — A company that provides open-source solutions—including Elasticsearch, Logstash, Kibana, and Beats—that are designed to take data from any source and search, analyze, and visualize it in real time.

Amazon Elasticsearch Service (Amazon ES) is an AWS-managed service for deploying, operating, and scaling Elasticsearch in the AWS Cloud.

	See Also Amazon Elasticsearch Service (Amazon ES), Elasticsearch.
Elastic Block Store	See Amazon Elastic Block Store (Amazon EBS).
Elastic IP address	A fixed (static) IP address that you have allocated in Amazon EC2 (p. 276) or Amazon VPC (p. 278) and then attached to an instance (p. 298). Elastic IP addresses are associated with your account, not a specific instance. They are *elastic* because you can easily allocate, attach, detach, and free them as your needs change. Unlike traditional static IP addresses, Elastic IP addresses allow you to mask instance or Availability Zone (p. 280) failures by rapidly remapping your public IP addresses to another instance.
Elastic Load Balancing	A web service that improves an application's availability by distributing incoming traffic between two or more EC2 instance (p. 292)s. See Also http://aws.amazon.com/elasticloadbalancing.
elastic network interface	An additional network interface that can be attached to an instance (p. 298). ENIs include a primary private IP address, one or more secondary private IP addresses, an elastic IP address (optional), a MAC address, membership in specified security group (p. 313)s, a description, and a source/destination check flag. You can create an ENI, attach it to an instance, detach it from an instance, and attach it to another instance.
Elasticsearch	An open source, real-time distributed search and analytics engine used for full-text search, structured search, and analytics. Elasticsearch was developed by the Elastic company. Amazon Elasticsearch Service (Amazon ES) is an AWS-managed service for deploying, operating, and scaling Elasticsearch in the AWS Cloud. See Also Amazon Elasticsearch Service (Amazon ES), Elastic.
EMR	See Amazon EMR (Amazon EMR).
encrypt	To use a mathematical algorithm to make data unintelligible to unauthorized user (p. 319)s while allowing authorized users a method (such as a key or password) to convert the altered data back to its original state.
encryption context	A set of key–value pairs that contains additional information associated with AWS Key Management Service (AWS KMS) (p. 282)–encrypted information.
endpoint	A URL that identifies a host and port as the entry point for a web service. Every web service request contains an endpoint. Most AWS products provide regional endpoints to enable faster connectivity. Amazon ElastiCache (p. 276): The DNS name of a cache node (p. 286). Amazon RDS (p. 277): The DNS name of a DB instance (p. 290). AWS CloudFormation (p. 280): The DNS name or IP address of the server that receives an HTTP request.
endpoint port	Amazon ElastiCache (p. 276): The port number used by a cache node (p. 286). Amazon RDS (p. 277): The port number used by a DB instance (p. 290).
envelope encryption	The use of a master key and a data key to algorithmically protect data. The master key is used to encrypt and decrypt the data key and the data key is used to encrypt and decrypt the data itself.
environment	AWS Elastic Beanstalk (p. 281): A specific running instance of an application (p. 279). The application has a CNAME and includes an

	application version and a customizable configuration (which is inherited from the default container type).
environment configuration	A collection of parameters and settings that define how an environment and its associated resources behave.
ephemeral store	See instance store.
epoch	The date from which time is measured. For most Unix environments, the epoch is January 1, 1970.
evaluation	Amazon Machine Learning: The process of measuring the predictive performance of a machine learning (ML) model. Also a machine learning object that stores the details and result of an ML model evaluation.
evaluation datasource	The data that Amazon Machine Learning uses to evaluate the predictive accuracy of a machine learning model.
eventual consistency	The method through which AWS products achieve high availability, which involves replicating data across multiple servers in Amazon's data centers. When data is written or updated and `Success` is returned, all copies of the data are updated. However, it takes time for the data to propagate to all storage locations. The data will eventually be consistent, but an immediate read might not show the change. Consistency is usually reached within seconds. See Also data consistency, eventually consistent read, strongly consistent read.
eventually consistent read	A read process that returns data from only one region and might not show the most recent write information. However, if you repeat your read request after a short time, the response should eventually return the latest data. See Also data consistency, eventual consistency, strongly consistent read.
eviction	The deletion by CloudFront (p. 275) of an object from an edge location (p. 292) before its expiration time. If an object in an edge location isn't frequently requested, CloudFront might evict the object (remove the object before its expiration date) to make room for objects that are more popular.
exbibyte	A contraction of exa binary byte, an exbibyte is 2^60 or 1,152,921,504,606,846,976 bytes. An exabyte (EB) is 10^18 or 1,000,000,000,000,000,000 bytes. 1,024 EiB is a zebibyte (p. 321).
expiration	For CloudFront (p. 275) caching, the time when CloudFront stops responding to user requests with an object. If you don't use headers or CloudFront distribution (p. 291) settings to specify how long you want objects to stay in an edge location (p. 292), the objects expire after 24 hours. The next time a user requests an object that has expired, CloudFront forwards the request to the origin (p. 305).
explicit launch permission	An Amazon Machine Image (AMI) (p. 277) launch permission granted to a specific AWS account (p. 274).
exponential backoff	A strategy that incrementally increases the wait between retry attempts in order to reduce the load on the system and increase the likelihood that repeated requests will succeed. For example, client applications might wait up to 400 milliseconds before attempting the first retry, up to 1600 milliseconds before the second, up to 6400 milliseconds (6.4 seconds) before the third, and so on.
expression	Amazon CloudSearch (p. 275): A numeric expression that you can use to control how search hits are sorted. You can construct Amazon CloudSearch

expressions using numeric fields, other rank expressions, a document's default relevance score, and standard numeric operators and functions. When you use the `sort` option to specify an expression in a search request, the expression is evaluated for each search hit and the hits are listed according to their expression values.

F

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

facet
: Amazon CloudSearch (p. 275): An index field that represents a category that you want to use to refine and filter search results.

facet enabled
: Amazon CloudSearch (p. 275): An index field option that enables facet information to be calculated for the field.

FBL
: See feedback loop.

feature transformation
: Amazon Machine Learning: The machine learning process of constructing more predictive input representations or “features” from the raw input variables to optimize a machine learning model’s ability to learn and generalize. Also known as *data transformation* or *feature engineering*.

federated identity management
: Allows individuals to sign in to different networks or services, using the same group or personal credentials to access data across all networks. With identity federation in AWS, external identities (federated users) are granted secure access to resource (p. 310)s in an AWS account (p. 274) without having to create IAM user (p. 319)s. These external identities can come from a corporate identity store (such as LDAP or Windows Active Directory) or from a third party (such as Login with Amazon, Facebook, or Google). AWS federation also supports SAML 2.0.

federated user
: See federated identity management.

federation
: See federated identity management.

feedback loop
: The mechanism by which a mailbox provider (for example, an Internet service provider (p. 298)) forwards a recipient (p. 309)'s complaint (p. 287) back to the sender (p. 313).

field weight
: The relative importance of a text field in a search index. Field weights control how much matches in particular text fields affect a document's relevance score.

filter
: A criterion that you specify to limit the results when you list or describe your Amazon EC2 (p. 276) resource (p. 310)s.

filter query
: A way to filter search results without affecting how the results are scored and sorted. Specified with the Amazon CloudSearch (p. 275) `fq` parameter.

FIM
: See federated identity management.

Firehose
: See Amazon Kinesis Firehose.

format version
: See template format version.

forums
: See discussion forums.

function — See intrinsic function.

fuzzy search — A simple search query that uses approximate string matching (fuzzy matching) to correct for typographical errors and misspellings.

G

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

geospatial search — A search query that uses locations specified as a latitude and longitude to determine matches and sort the results.

gibibyte — A contraction of giga binary byte, a gibibyte is 2^30 or 1,073,741,824 bytes. A gigabyte (GB) is 10^9 or 1,000,000,000 bytes. 1,024 GiB is a tebibyte (p. 318).

global secondary index — An index with a partition key and a sort key that can be different from those on the table. A global secondary index is considered global because queries on the index can span all of the data in a table, across all partitions.
See Also local secondary index.

grant — AWS Key Management Service (AWS KMS) (p. 282): A mechanism for giving AWS principal (p. 307)s long-term permissions to use customer master key (CMK) (p. 289)s.

grant token — A type of identifier that allows the permissions in a grant (p. 296) to take effect immediately.

ground truth — The observations used in the machine learning (ML) model training process that include the correct value for the target attribute. To train an ML model to predict house sales prices, the input observations would typically include prices of previous house sales in the area. The sale prices of these houses constitute the ground truth.

group — A collection of IAM (p. 282) user (p. 319)s. You can use IAM groups to simplify specifying and managing permissions for multiple users.

H

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

Hadoop — Software that enables distributed processing for big data by using clusters and simple programming models. For more information, see http://hadoop.apache.org.

hard bounce — A persistent email delivery failure such as "mailbox does not exist."

hardware VPN — A hardware-based IPsec VPN connection over the Internet.

health check — A system call to check on the health status of each instance in an Auto Scaling (p. 280) group.

high-quality email	Email that recipients find valuable and want to receive. Value means different things to different recipients and can come in the form of offers, order confirmations, receipts, newsletters, etc.
highlights	Amazon CloudSearch (p. 275): Excerpts returned with search results that show where the search terms appear within the text of the matching documents.
highlight enabled	Amazon CloudSearch (p. 275): An index field option that enables matches within the field to be highlighted.
hit	A document that matches the criteria specified in a search request. Also referred to as a *search result*.
HMAC	Hash-based Message Authentication Code. A specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret key. You can use it to verify both the data integrity and the authenticity of a message at the same time. AWS calculates the HMAC using a standard, cryptographic hash algorithm, such as SHA-256.
hosted zone	A collection of resource record (p. 310) sets that Amazon Route 53 (p. 278) hosts. Like a traditional DNS zone file, a hosted zone represents a collection of records that are managed together under a single domain name.
HVM virtualization	Hardware Virtual Machine virtualization. Allows the guest VM to run as though it is on a native hardware platform, except that it still uses paravirtual (PV) network and storage drivers for improved performance. See Also PV virtualization.

I

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

IAM	See AWS Identity and Access Management (IAM).
IAM group	See group.
IAM policy simulator	See policy simulator.
IAM role	See role.
IAM user	See user.
Identity and Access Management	See AWS Identity and Access Management (IAM).
identity provider (IdP)	An IAM (p. 282) entity that holds metadata about external identity providers.
IdP	See identity provider (IdP) .
image	See Amazon Machine Image (AMI).
import/export station	A machine that uploads or downloads your data to or from Amazon S3 (p. 278).
import log	A report that contains details about how AWS Import/Export (p. 282) processed your data.

index	See search index.
index field	A name–value pair that is included in an Amazon CloudSearch (p. 275) domain's index. An index field can contain text or numeric data, dates, or a location.
indexing options	Configuration settings that define an Amazon CloudSearch (p. 275) domain's index fields, how document data is mapped to those index fields, and how the index fields can be used.
inline policy	An IAM (p. 282) policy (p. 306) that is embedded in a single IAM user (p. 319), group (p. 296), or role (p. 311).
input data	Amazon Machine Learning: The observations that you provide to Amazon Machine Learning to train and evaluate a machine learning model and generate predictions.
instance	A copy of an Amazon Machine Image (AMI) (p. 277) running as a virtual server in the AWS cloud.
instance family	A general instance type (p. 298) grouping using either storage or CPU capacity.
instance group	A Hadoop (p. 296) cluster contains one master instance group that contains one master node (p. 302), a core instance group containing one or more core node (p. 288) and an optional task node (p. 318) instance group, which can contain any number of task nodes.
instance profile	A container that passes IAM (p. 282) role (p. 311) information to an EC2 instance (p. 292) at launch.
instance store	Disk storage that is physically attached to the host computer for an EC2 instance (p. 292), and therefore has the same lifespan as the instance. When the instance is terminated, you lose any data in the instance store.
instance store-backed AMI	A type of Amazon Machine Image (AMI) (p. 277) whose instance (p. 298)s use an instance store (p. 298) volume (p. 320) as the root device. Compare this with instances launched from Amazon EBS (p. 275)-backed AMIs, which use an Amazon EBS volume as the root device.
instance type	A specification that defines the memory, CPU, storage capacity, and hourly cost for an instance (p. 298). Some instance types are designed for standard applications, whereas others are designed for CPU-intensive, memory-intensive applications, and so on.
Internet gateway	Connects a network to the Internet. You can route traffic for IP addresses outside your VPC (p. 320) to the Internet gateway.
Internet service provider	A company that provides subscribers with access to the Internet. Many ISPs are also mailbox provider (p. 301)s. Mailbox providers are sometimes referred to as ISPs, even if they only provide mailbox services.
intrinsic function	A special action in a AWS CloudFormation (p. 280) template that assigns values to properties not available until runtime. These functions follow the format *Fn::Attribute*, such as `Fn::GetAtt`. Arguments for intrinsic functions can be parameters, pseudo parameters, or the output of other intrinsic functions.
IP address	A numerical address (for example, 192.0.2.44) that networked devices use to communicate with one another using the Internet Protocol (IP). All EC2 instance (p. 292)s are assigned two IP addresses at launch, which

	are directly mapped to each other through network address translation (NAT (p. 303)): a private IP address (following RFC 1918) and a public IP address. Instances launched in a VPC (p. 278) are assigned only a private IP address. Instances launched in your default VPC are assigned both a private IP address and a public IP address.
IP match condition	AWS WAF (p. 284): An attribute that specifies the IP addresses or IP address ranges that web requests originate from. Based on the specified IP addresses, you can configure AWS WAF to allow or block web requests to AWS resource (p. 310)s such as Amazon CloudFront (p. 275) distributions.
ISP	See Internet service provider.
issuer	The person who writes a policy (p. 306) to grant permissions to a resource (p. 310). The issuer (by definition) is always the resource owner. AWS does not permit Amazon SQS (p. 278) users to create policies for resources they don't own. If John is the resource owner, AWS authenticates John's identity when he submits the policy he's written to grant permissions for that resource.
item	A group of attributes that is uniquely identifiable among all of the other items. Items in Amazon DynamoDB (p. 275) are similar in many ways to rows, records, or tuples in other database systems.

J

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

job flow	Amazon EMR (p. 276): One or more step (p. 316)s that specify all of the functions to be performed on the data.
job ID	A five-character, alphanumeric string that uniquely identifies an AWS Import/Export (p. 282) storage device in your shipment. AWS issues the job ID in response to a `CREATE JOB` email command.
job prefix	An optional string that you can add to the beginning of an AWS Import/Export (p. 282) log file name to prevent collisions with objects of the same name. See Also key prefix.
JSON	JavaScript Object Notation. A lightweight data interchange format. For information about JSON, see http://www.json.org/.
junk folder	The location where email messages that various filters determine to be of lesser value are collected so that they do not arrive in the recipient (p. 309)'s inbox but are still accessible to the recipient. This is also referred to as a spam (p. 315) or bulk folder.

K

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

key
: A credential that identifies an AWS account (p. 274) or user (p. 319) to AWS (such as the AWS secret access key (p. 313)).
Amazon Simple Storage Service (Amazon S3) (p. 278), Amazon EMR (Amazon EMR) (p. 276): The unique identifier for an object in a bucket (p. 285). Every object in a bucket has exactly one key. Because a bucket and key together uniquely identify each object, you can think of Amazon S3 as a basic data map between the *bucket* + *key*, and the object itself. You can uniquely address every object in Amazon S3 through the combination of the web service endpoint, bucket name, and key, as in this example: `http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl`, where `doc` is the name of the bucket, and `2006-03-01/AmazonS3.wsdl` is the key.
AWS Import/Export (p. 282): The name of an object in Amazon S3. It is a sequence of Unicode characters whose UTF-8 encoding cannot exceed 1024 bytes. If a key, for example, logPrefix + import-log-JOBID, is longer than 1024 bytes, AWS Elastic Beanstalk (p. 281) returns an `InvalidManifestField` error.
IAM (p. 282): In a policy (p. 306), a specific characteristic that is the basis for restricting access (such as the current time, or the IP address of the requester).
Tagging resources: A general tag (p. 317) label that acts like a category for more specific tag values. For example, you might have EC2 instance (p. 292) with the tag key of *Owner* and the tag value of *Jan*. You can tag an AWS resource (p. 310) with up to 10 key–value pairs. Not all AWS resources can be tagged.

key pair
: A set of security credentials that you use to prove your identity electronically. A key pair consists of a private key and a public key.

key prefix
: A logical grouping of the objects in a bucket (p. 285). The prefix value is similar to a directory name that enables you to store similar data under the same directory in a bucket.

kibibyte
: A contraction of kilo binary byte, a kibibyte is 2^10 or 1,024 bytes. A kilobyte (KB) is 10^3 or 1,000 bytes. 1,024 KiB is a mebibyte (p. 302).

KMS
: See AWS Key Management Service (AWS KMS).

L

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

labeled data
: In machine learning, data for which you already know the target or “correct” answer.

launch configuration
: A set of descriptive parameters used to create new EC2 instance (p. 292)s in an Auto Scaling (p. 280) activity.

A template that an Auto Scaling group (p. 280) uses to launch new EC2 instances. The launch configuration contains information such as the Amazon Machine Image (AMI) (p. 277) ID, the instance type, key pairs, security group (p. 313)s, and block device mappings, among other configuration settings.

launch permission
: An Amazon Machine Image (AMI) (p. 277) attribute that allows users to launch an AMI.

lifecycle	The lifecycle state of the EC2 instance (p. 292) contained in an Auto Scaling group (p. 280). EC2 instances progress through several states over their lifespan; these include *Pending*, *InService*, *Terminating* and *Terminated*.
lifecycle action	An action that can be paused by Auto Scaling, such as launching or terminating an EC2 instance.
lifecycle hook	Enables you to pause Auto Scaling after it launches or terminates an EC2 instance so that you can perform a custom action while the instance is not in service.
link to VPC	The process of linking (or attaching) an EC2-Classic instance (p. 298) to a ClassicLink-enabled VPC (p. 320). See Also ClassicLink, unlink from VPC.
load balancer	A DNS name combined with a set of ports, which together provide a destination for all requests intended for your application. A load balancer can distribute traffic to multiple application instances across every Availability Zone (p. 280) within a region (p. 309). Load balancers can span multiple Availability Zones within an Amazon EC2 (p. 276) region, but they cannot span multiple regions.
local secondary index	An index that has the same partition key as the table, but a different sort key. A local secondary index is local in the sense that every partition of a local secondary index is scoped to a table partition that has the same partition key value. See Also local secondary index.
logical name	A case-sensitive unique string within an AWS CloudFormation (p. 280) template that identifies a resource (p. 310), mapping (p. 302), parameter, or output. In an AWS CloudFormation template, each parameter, resource (p. 310), property, mapping, and output must be declared with a unique logical name. You use the logical name when dereferencing these items using the `Ref` function.

M

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

Mail Transfer Agent (MTA)	Software that transports email messages from one computer to another by using a client-server architecture.
mailbox provider	An organization that provides email mailbox hosting services. Mailbox providers are sometimes referred to as Internet service provider (p. 298)s, even if they only provide mailbox services.
mailbox simulator	A set of email addresses that you can use to test an Amazon SES (p. 278)-based email sending application without sending messages to actual recipients. Each email address represents a specific scenario (such as a bounce or complaint) and generates a typical response that is specific to the scenario.
main route table	The default route table (p. 311) that any new VPC (p. 320) subnet (p. 316) uses for routing. You can associate a subnet with a different route table of your choice. You can also change which route table is the main route table.
managed policy	A standalone IAM (p. 282) policy (p. 306) that you can attach to multiple user (p. 319)s, group (p. 296)s, and role (p. 311)s in your IAM

	account (p. 274). Managed policies can either be AWS managed policies (which are created and managed by AWS) or customer managed policies (which you create and manage in your AWS account).
manifest	When sending a *create job* request for an import or export operation, you describe your job in a text file called a manifest. The manifest file is a YAML-formatted file that specifies how to transfer data between your storage device and the AWS cloud.
manifest file	Amazon Machine Learning: The file used for describing batch predictions. The manifest file relates each input data file with its associated batch prediction results. It is stored in the Amazon S3 output location.
mapping	A way to add conditional parameter values to an AWS CloudFormation (p. 280) template. You specify mappings in the template's optional Mappings section and retrieve the desired value using the `FN::FindInMap` function.
marker	See pagination token.
master node	A process running on an Amazon Machine Image (AMI) (p. 277) that keeps track of the work its core and task nodes complete.
maximum price	The maximum price you will pay to launch one or more Spot Instance (p. 315)s. If your maximum price exceeds the current Spot price (p. 315) and your restrictions are met, Amazon EC2 (p. 276) launches instances on your behalf.
maximum send rate	The maximum number of email messages that you can send per second using Amazon SES (p. 278).
mebibyte	A contraction of mega binary byte, a mebibyte is 2^20 or 1,048,576 bytes. A megabyte (MB) is 10^6 or 1,000,000 bytes. 1,024 MiB is a gibibyte (p. 296).
member resources	See resource.
message ID	Amazon Simple Email Service (Amazon SES) (p. 278): A unique identifier that is assigned to every email message that is sent. Amazon Simple Queue Service (Amazon SQS) (p. 278): The identifier returned when you send a message to a queue.
metadata	Information about other data or objects. In Amazon Simple Storage Service (Amazon S3) (p. 278) and Amazon EMR (Amazon EMR) (p. 276) metadata takes the form of name–value pairs that describe the object. These include default metadata such as the date last modified and standard HTTP metadata such as Content-Type. Users can also specify custom metadata at the time they store an object. In Amazon Elastic Compute Cloud (Amazon EC2) (p. 276) metadata includes data about an EC2 instance (p. 292) that the instance can retrieve to determine things about itself, such as the instance type, the IP address, and so on.
metric	An element of time-series data defined by a unique combination of exactly one namespace (p. 303), exactly one metric name, and between zero and ten dimensions. Metrics and the statistics derived from them are the basis of Amazon CloudWatch (p. 275).
metric name	The primary identifier of a metric, used in combination with a namespace (p. 303) and optional dimensions.
MFA	See multi-factor authentication (MFA).

micro instance — A type of EC2 instance (p. 292) that is more economical to use if you have occasional bursts of high CPU activity.

MIME — See Multipurpose Internet Mail Extensions (MIME).

ML model — In machine learning (ML), a mathematical model that generates predictions by finding patterns in data. Amazon Machine Learning supports three types of ML models: binary classification, multiclass classification, and regression. Also known as a *predictive model*.
See Also binary classification model, multiclass classification model, regression model.

MTA — See Mail Transfer Agent (MTA).

Multi-AZ deployment — A primary DB instance (p. 290) that has a synchronous standby replica in a different Availability Zone (p. 280). The primary DB instance is synchronously replicated across Availability Zones to the standby replica.

multiclass classification model — A machine learning model that predicts values that belong to a limited, pre-defined set of permissible values. For example, "Is this product a book, movie, or clothing?"

multi-factor authentication (MFA) — An optional AWS account (p. 274) security feature. Once you enable AWS MFA, you must provide a six-digit, single-use code in addition to your sign-in credentials whenever you access secure AWS webpages or the AWS Management Console (p. 282). You get this single-use code from an authentication device that you keep in your physical possession.
See Also http://aws.amazon.com/mfa/.

multi-valued attribute — An attribute with more than one value.

multipart upload — A feature that allows you to upload a single object as a set of parts.

Multipurpose Internet Mail Extensions (MIME) — An Internet standard that extends the email protocol to include non-ASCII text and nontext elements like attachments.

Multitool — A cascading application that provides a simple command-line interface for managing large datasets.

N

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

namespace — An abstract container that provides context for the items (names, or technical terms, or words) it holds, and allows disambiguation of homonym items residing in different namespaces.

NAT — Network address translation. A strategy of mapping one or more IP addresses to another while data packets are in transit across a traffic routing device. This is commonly used to restrict Internet communication to private instances while allowing outgoing traffic.
See Also Network Address Translation and Protocol Translation, NAT gateway, NAT instance.

NAT gateway — A NAT (p. 303) device, managed by AWS, that performs network address translation in a private subnet (p. 316), to secure inbound Internet traffic. A NAT gateway uses both NAT and port address translation.

	See Also NAT instance.
NAT instance	A NAT (p. 303) device, configured by a user, that performs network address translation in a VPC (p. 320) public subnet (p. 316) to secure inbound Internet traffic. See Also NAT gateway.
network ACL	An optional layer of security that acts as a firewall for controlling traffic in and out of a subnet (p. 316). You can associate multiple subnets with a single network ACL (p. 273), but a subnet can be associated with only one network ACL at a time.
Network Address Translation and Protocol Translation	(NAT (p. 303)-PT) An Internet protocol standard defined in RFC 2766. See Also NAT instance, NAT gateway.
n-gram processor	A processor that performs n-gram transformations. See Also n-gram transformation.
n-gram transformation	Amazon Machine Learning: A transformation that aids in text string analysis. An n-gram transformation takes a text variable as input and outputs strings by sliding a window of size *n* words, where *n* is specified by the user, over the text, and outputting every string of words of size *n* and all smaller sizes. For example, specifying the n-gram transformation with window size =2 returns all the two-word combinations and all of the single words.
node	Amazon Elasticsearch Service (Amazon ES) (p. 276): An Elasticsearch instance. A node can be either a data instance or a dedicated master instance. See Also dedicated master node.
NoEcho	A property of AWS CloudFormation (p. 280) parameters that prevent the otherwise default reporting of names and values of a template parameter. Declaring the *NoEcho* property causes the parameter value to be masked with asterisks in the report by the `cfn-describe-stacks` command.
NoSQL	Nonrelational database systems that are highly available, scalable, and optimized for high performance. Instead of the relational model, NoSQL databases (like Amazon DynamoDB (p. 275)) use alternate models for data management, such as key–value pairs or document storage.
null object	A null object is one whose version ID is null. Amazon S3 (p. 278) adds a null object to a bucket (p. 285) when versioning (p. 320) for that bucket is suspended. It is possible to have only one null object for each key in a bucket.
number of passes	The number of times that you allow Amazon Machine Learning to use the same data records to train a machine learning model.

O

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

object	Amazon Simple Storage Service (Amazon S3) (p. 278): The fundamental entity type stored in Amazon S3. Objects consist of object data and metadata. The data portion is opaque to Amazon S3. Amazon CloudFront (p. 275): Any entity that can be served either over HTTP or a version of RTMP.

observation — Amazon Machine Learning: A single instance of data that Amazon Machine Learning (Amazon ML) uses to either train a machine learning model how to predict or to generate a prediction. Each row in an Amazon ML input data file is an observation.

On-Demand Instance — An Amazon EC2 (p. 276) pricing option that charges you for compute capacity by the hour with no long-term commitment.

operation — An API function. Also called an *action*.

optimistic locking — A strategy to ensure that an item that you want to update has not been modified by others before you perform the update. For Amazon DynamoDB (p. 275), optimistic locking support is provided by the AWS SDKs.

origin access identity — Also called OAI. When using Amazon CloudFront (p. 275) to serve content with an Amazon S3 (p. 278) bucket (p. 285) as the origin, a virtual identity that you use to require users to access your content through CloudFront URLs instead of Amazon S3 URLs. Usually used with CloudFront private content.

origin server — The Amazon S3 (p. 278) bucket (p. 285) or custom origin containing the definitive original version of the content you deliver through CloudFront (p. 275).

OSB transformation — Orthogonal sparse bigram transformation. In machine learning, a transformation that aids in text string analysis and that is an alternative to the n-gram transformation. OSB transformations are generated by sliding the window of size *n* words over the text, and outputting every pair of words that includes the first word in the window.
See Also n-gram transformation.

output location — Amazon Machine Learning: An Amazon S3 location where the results of a batch prediction are stored.

P

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

pagination — The process of responding to an API request by returning a large list of records in small separate parts. Pagination can occur in the following situations:

- The client sets the maximum number of returned records to a value below the total number of records.
- The service has a default maximum number of returned records that is lower than the total number of records.

When an API response is paginated, the service sends a subset of the large list of records and a pagination token that indicates that more records are available. The client includes this pagination token in a subsequent API request, and the service responds with the next subset of records. This continues until the service responds with a subset of records and no pagination token, indicating that all records have been sent.

pagination token — A marker that indicates that an API response contains a subset of a larger list of records. The client can return this marker in a subsequent API request to

	retrieve the next subset of records until the service responds with a subset of records and no pagination token, indicating that all records have been sent. See Also pagination.
paid AMI	An Amazon Machine Image (AMI) (p. 277) that you sell to other Amazon EC2 (p. 276) users on AWS Marketplace (p. 282).
paravirtual virtualization	See PV virtualization.
part	A contiguous portion of the object's data in a multipart upload request.
partition key	A simple primary key, composed of one attribute (also known as a *hash attribute*). See Also partition key, sort key.
PAT	Port address translation.
pebibyte	A contraction of peta binary byte, a pebibyte is 2^50 or 1,125,899,906,842,624 bytes. A petabyte (PB) is 10^15 or 1,000,000,000,000,000 bytes. 1,024 PiB is an exbibyte (p. 294).
period	See sampling period.
permission	A statement within a policy (p. 306) that allows or denies access to a particular resource (p. 310). You can state any permission like this: "A has permission to do B to C." For example, Jane (A) has permission to read messages (B) from John's Amazon SQS (p. 278) queue (C). Whenever Jane sends a request to Amazon SQS to use John's queue, the service checks to see if she has permission and if the request satisfies the conditions John set forth in the permission.
persistent storage	A data storage solution where the data remains intact until it is deleted. Options within AWS (p. 278) include: Amazon S3 (p. 278), Amazon RDS (p. 277), Amazon DynamoDB (p. 275), and other services.
physical name	A unique label that AWS CloudFormation (p. 280) assigns to each resource (p. 310) when creating a stack (p. 315). Some AWS CloudFormation commands accept the physical name as a value with the `--physical-name` parameter.
pipeline	AWS CodePipeline (p. 281): A workflow construct that defines the way software changes go through a release process.
plaintext	Information that has not been encrypted (p. 293), as opposed to ciphertext (p. 286).
policy	IAM (p. 282): A document defining permissions that apply to a user, group, or role; the permissions in turn determine what users can do in AWS. A policy typically allow (p. 274)s access to specific actions, and can optionally grant that the actions are allowed for specific resource (p. 310)s, like EC2 instance (p. 292)s, Amazon S3 (p. 278) bucket (p. 285)s, and so on. Policies can also explicitly deny (p. 291) access. Auto Scaling (p. 280): An object that stores the information needed to launch or terminate instances for an Auto Scaling group. Executing the policy causes instances to be launched or terminated. You can configure an alarm (p. 274) to invoke an Auto Scaling policy.
policy generator	A tool in the IAM (p. 282) AWS Management Console (p. 282) that helps you build a policy (p. 306) by selecting elements from lists of available options.

policy simulator	A tool in the IAM (p. 282) AWS Management Console (p. 282) that helps you test and troubleshoot policies (p. 306) so you can see their effects in real-world scenarios.
policy validator	A tool in the IAM (p. 282) AWS Management Console (p. 282) that examines your existing IAM access control policies (p. 306) to ensure that they comply with the IAM policy grammar.
presigned URL	A web address that uses query string authentication (p. 308).
prefix	See job prefix.
Premium Support	A one-on-one, fast-response support channel that AWS customers can subscribe to for support for AWS infrastructure services. See Also http://aws.amazon.com/premiumsupport/.
primary key	One or two attributes that uniquely identify each item in a Amazon DynamoDB (p. 275) table, so that no two items can have the same key. See Also partition key, sort key.
primary shard	See Also shard.
principal	The user (p. 319), service, or account (p. 274) that receives permissions that are defined in a policy (p. 306). The principal is A in the statement "A has permission to do B to C."
private content	When using Amazon CloudFront (p. 275) to serve content with an Amazon S3 (p. 278) bucket (p. 285) as the origin, a method of controlling access to your content by requiring users to use signed URLs. Signed URLs can restrict user access based on the current date and time and/or the IP addresses that the requests originate from.
private IP address	A private numerical address (for example, 192.0.2.44) that networked devices use to communicate with one another using the Internet Protocol (IP). All EC2 instance (p. 292)ss are assigned two IP addresses at launch, which are directly mapped to each other through Network Address Translation (NAT (p. 303)): a private address (following RFC 1918) and a public address. *Exception:* Instances launched in Amazon VPC (p. 278) are assigned only a private IP address.
private subnet	A VPC (p. 320) subnet (p. 316) whose instances cannot be reached from the Internet.
product code	An identifier provided by AWS when you submit a product to AWS Marketplace (p. 282).
properties	See resource property.
property rule	A JSON (p. 299)-compliant markup standard for declaring properties, mappings, and output values in an AWS CloudFormation (p. 280) template.
Provisioned IOPS	A storage option designed to deliver fast, predictable, and consistent I/O performance. When you specify an IOPS rate while creating a DB instance, Amazon RDS (p. 277) provisions that IOPS rate for the lifetime of the DB instance.
pseudo parameter	A predefined setting, such as `AWS:StackName` that can be used in AWS CloudFormation (p. 280) templates without having to declare them. You can use pseudo parameters anywhere you can use a regular parameter.
public AMI	An Amazon Machine Image (AMI) (p. 277) that all AWS account (p. 274)s have permission to launch.

public data set
: A large collection of public information that can be seamlessly integrated into AWS cloud-based applications. Amazon stores public data sets at no charge to the community and, like all AWS services, users pay only for the compute and storage they use for their own applications. These data sets currently include data from the Human Genome Project, the U.S. Census, Wikipedia, and other sources.
See Also http://aws.amazon.com/publicdatasets.

public IP address
: A pubic numerical address (for example, 192.0.2.44) that networked devices use to communicate with one another using the Internet Protocol (IP). EC2 instance (p. 292)s are assigned two IP addresses at launch, which are directly mapped to each other through Network Address Translation (NAT (p. 303)): a private address (following RFC 1918) and a public address. *Exception:* Instances launched in Amazon VPC (p. 278) are assigned only a private IP address.

public subnet
: A subnet (p. 316) whose instances can be reached from the Internet.

PV virtualization
: Paravirtual virtualization. Allows guest VMs to run on host systems that do not have special support extensions for full hardware and CPU virtualization. Because PV guests run a modified operating system that does not use hardware emulation, they cannot provide hardware-related features such as enhanced networking or GPU support.
See Also HVM virtualization.

Q

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

quartile binning transformation
: Amazon Machine Learning: A process that takes two inputs, a numerical variable and a parameter called a bin number, and outputs a categorical variable. Quartile binning transformations discover non-linearity in a variable's distribution by enabling the machine learning model to learn separate importance values for parts of the numeric variable's distribution.

Query
: A type of HTTP-based request interface that generally uses only the GET or POST HTTP method and a query string with parameters.
See Also REST, REST-Query.

query string authentication
: An AWS feature that lets you place the authentication information in the HTTP request query string instead of in the `Authorization` header, which enables URL-based access to objects in a bucket (p. 285).

queue
: A sequence of messages or jobs that are held in temporary storage awaiting transmission or processing.

queue URL
: A web address that uniquely identifies a queue.

quota
: Amazon RDS (p. 277): The maximum number of DB instance (p. 290)s and available storage you can use.

  Amazon ElastiCache (p. 276): The maximum number of the following items:

  - The number of cache clusters for each AWS account (p. 274)
  - The number of cache nodes per cache cluster

- The total number of cache nodes per AWS account across all cache clusters created by that AWS account

R

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

range GET
: A request that specifies a byte range of data to get for a download. If an object is large, you can break up a download into smaller units by sending multiple range GET requests that each specify a different byte range to GET.

raw email
: A type of *sendmail* request with which you can specify the email headers and MIME types.

RDS
: See Amazon Relational Database Service (Amazon RDS).

read replica
: Amazon RDS (p. 277): An active copy of another DB instance. Any updates to the data on the source DB instance are replicated to the read replica DB instance using the built-in replication feature of MySQL 5.1.

real-time predictions
: Amazon Machine Learning: Synchronously generated predictions for individual data observations.
See Also batch prediction.

receipt handle
: Amazon SQS (p. 278): An identifier that you get when you receive a message from the queue. This identifier is required to delete a message from the queue or when changing a message's visibility timeout.

receiver
: The entity that consists of the network systems, software, and policies that manage email delivery for a recipient (p. 309).

recipient
: Amazon Simple Email Service (Amazon SES) (p. 278): The person or entity receiving an email message. For example, a person named in the "To" field of a message.

reference
: A means of inserting a property from one AWS resource (p. 310) into another. For example, you could insert an Amazon EC2 (p. 276) security group (p. 313) property into an Amazon RDS (p. 277) resource.

region
: A named set of AWS resource (p. 310)s in the same geographical area. A region comprises at least two Availability Zone (p. 280)s.

regression model
: Amazon Machine Learning: Preformatted instructions for common data transformations that fine-tune machine learning model performance.

regression model
: A type of machine learning model that predicts a numeric value, such as the exact purchase price of a house.

regularization
: A machine learning (ML) parameter that you can tune to obtain higher-quality ML models. Regularization helps prevent ML models from memorizing training data examples instead of learning how to generalize the patterns it sees (called overfitting). When training data is overfitted, the ML model performs well on the training data but does not perform well on the evaluation data or on new data.

replica shard
: See Also shard.

reply path	The email address to which an email reply is sent. This is different from the return path (p. 311).
reputation	1. An Amazon SES (p. 278) metric, based on factors that might include bounce (p. 285)s, complaint (p. 287)s, and other metrics, regarding whether or not a customer is sending high-quality email. 2. A measure of confidence, as judged by an Internet service provider (p. 298) or other entity that an IP address that they are receiving email from is not the source of spam (p. 315).
requester	The person (or application) that sends a request to AWS to perform a specific action. When AWS receives a request, it first evaluates the requester's permissions to determine whether the requester is allowed to perform the request action (if applicable, for the requested resource (p. 310)).
Requester Pays	An Amazon S3 (p. 278) feature that allows a bucket owner (p. 285) to specify that anyone who requests access to objects in a particular bucket (p. 285) must pay the data transfer and request costs.
reservation	A collection of EC2 instance (p. 292)s started as part of the same launch request. Not to be confused with a Reserved Instance (p. 310).
Reserved Instance	A pricing option for EC2 instance (p. 292)s that discounts the on-demand (p. 305) usage charge for instances that meet the specified parameters. Customers pay for the entire term of the instance, regardless of how they use it.
Reserved Instance Marketplace	An online exchange that matches sellers who have reserved capacity that they no longer need with buyers who are looking to purchase additional capacity. Reserved Instance (p. 310)s that you purchase from third-party sellers have less than a full standard term remaining and can be sold at different upfront prices. The usage or reoccurring fees remain the same as the fees set when the Reserved Instances were originally purchased. Full standard terms for Reserved Instances available from AWS run for one year or three years.
resource	An entity that users can work with in AWS, such as an EC2 instance (p. 292), a Amazon DynamoDB (p. 275) table, an Amazon S3 (p. 278) bucket (p. 285), an IAM (p. 282) user, an AWS OpsWorks (p. 282) stack (p. 315), and so on.
resource property	A value required when including an AWS resource (p. 310) in an AWS CloudFormation (p. 280) stack (p. 315). Each resource may have one or more properties associated with it. For example, an `AWS::EC2::Instance` resource may have a `UserData` property. In an AWS CloudFormation template, resources must declare a properties section, even if the resource has no properties.
resource record	Also called *resource record set*. The fundamental information elements in the Domain Name System (DNS). See Also Domain Name System in Wikipedia.
REST	A type of HTTP-based request interface that generally uses only the GET or POST HTTP method and a query string with parameters. Sometimes known as Query (p. 308). In some implementations of a REST interface, other HTTP verbs besides GET and POST are used.
REST-Query	Also known as Query (p. 308) or HTTP Query. This is a type of HTTP request that generally uses only the GET or POST HTTP method and a query string with parameters. Compare this with REST (p. 310), which is a

	type of HTTP request that uses any HTTP method (GET, DELETE, POST, etc.), a resource (p. 310), HTTP headers, and possibly a query string with parameters.
return enabled	Amazon CloudSearch (p. 275): An index field option that enables the field's values to be returned in the search results.
return path	The email address to which bounced email is returned. The return path is specified in the header of the original email. This is different from the reply path (p. 310).
revision	AWS CodePipeline (p. 281): A change made to a source that is configured in a source action, such as a pushed commit to a GitHub repository or an update to a file in a versioned Amazon S3 (p. 278) bucket (p. 285).
role	A tool for giving temporary access to AWS resource (p. 310)s in your AWS account (p. 274).
rollback	A return to a previous state that follows the failure to create an object, such as AWS CloudFormation (p. 280) stack (p. 315). All resource (p. 310)s associated with the failure are deleted during the rollback. For AWS CloudFormation, you can override this behavior using the `--disable-rollback` option on the command line.
root credentials	Authentication information associated with the AWS account (p. 274) owner.
root device volume	A volume (p. 320) that contains the image used to boot the instance (p. 298). If you launched the instance from an AMI (p. 277) backed by instance store (p. 298), this is an instance store volume (p. 320) created from a template stored in Amazon S3 (p. 278). If you launched the instance from an AMI backed by Amazon EBS (p. 275), this is an Amazon EBS volume created from an Amazon EBS snapshot.
route table	A set of routing rules that controls the traffic leaving any subnet (p. 316) that is associated with the route table. You can associate multiple subnets with a single route table, but a subnet can be associated with only one route table at a time.
row identifier	row ID.Amazon Machine Learning: An attribute in the input data that you can include in the evaluation or prediction output to make it easier to associate a prediction with an observation.
rule	AWS WAF (p. 284): A set of conditions that AWS WAF searches for in web requests to AWS resource (p. 310)s such as Amazon CloudFront (p. 275) distributions. You add rules to a web ACL (p. 321), and then specify whether you want to allow or block web requests based on each rule.

S

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

S3	See Amazon Simple Storage Service (Amazon S3).
sampling period	A defined duration of time, such as one minute, over which Amazon CloudWatch (p. 275) computes a statistic (p. 315).

sandbox	A testing location where you can test the functionality of your application without affecting production, incurring charges, or purchasing products. Amazon SES (p. 278): An environment that is designed for developers to test and evaluate the service. In the sandbox, you have full access to the Amazon SES API, but you can only send messages to verified email addresses and the mailbox simulator. To get out of the sandbox, you need to apply for production access. Accounts in the sandbox also have lower sending limits (p. 313) than production accounts.
scale in	To remove EC2 instances from an Auto Scaling group (p. 280).
scale out	To add EC2 instances to an Auto Scaling group (p. 280).
scaling policy	A description of how Auto Scaling should automatically scale an Auto Scaling group (p. 280) in response to changing demand.
scaling activity	A process that changes the size, configuration, or makeup of an Auto Scaling group (p. 280) by launching or terminating instances.
scheduler	The method used for placing task (p. 318)s on container instance (p. 288)s.
schema	Amazon Machine Learning: The information needed to interpret the input data for a machine learning model, including attribute names and their assigned data types, and the names of special attributes.
score cut-off value	Amazon Machine Learning: A binary classification models output a score that ranges from 0 to 1. To decide whether an observation should be classified as 1 or 0, you pick a classification threshold, or cut-off, and Amazon ML compares the score against it. Observations with scores higher than the cut-off are predicted as target equals 1, and scores lower than the cut-off are predicted as target equals 0.
search API	Amazon CloudSearch (p. 275): The API that you use to submit search requests to a search domain (p. 312).
search domain	Amazon CloudSearch (p. 275): Encapsulates your searchable data and the search instances that handle your search requests. You typically set up a separate Amazon CloudSearch domain for each different collection of data that you want to search.
search domain configuration	Amazon CloudSearch (p. 275): An domain's indexing options, analysis scheme (p. 279)s, expression (p. 294)s, suggester (p. 317)s, access policies, and scaling and availability options.
search enabled	Amazon CloudSearch (p. 275): An index field option that enables the field data to be searched.
search endpoint	Amazon CloudSearch (p. 275): The URL that you connect to when sending search requests to a search domain. Each Amazon CloudSearch domain has a unique search endpoint that remains the same for the life of the domain.
search index	Amazon CloudSearch (p. 275): A representation of your searchable data that facilitates fast and accurate data retrieval.
search instance	Amazon CloudSearch (p. 275): A compute resource (p. 310) that indexes your data and processes search requests. An Amazon CloudSearch domain has one or more search instances, each with a finite amount of RAM and CPU resources. As your data volume grows, more search instances or larger search instances are deployed to contain your indexed data. When necessary, your

	index is automatically partitioned across multiple search instances. As your request volume or complexity increases, each search partition is automatically replicated to provide additional processing capacity.
search request	Amazon CloudSearch (p. 275): A request that is sent to an Amazon CloudSearch domain's search endpoint to retrieve documents from the index that match particular search criteria.
search result	Amazon CloudSearch (p. 275): A document that matches a search request. Also referred to as a *search hit*.
secret access key	A key that is used in conjunction with the access key ID (p. 273) to cryptographically sign programmatic AWS requests. Signing a request identifies the sender and prevents the request from being altered. You can generate secret access keys for your AWS account (p. 274), individual IAM user (p. 319)s, and temporary sessions.
security group	A named set of allowed inbound network connections for an instance. (Security groups in Amazon VPC (p. 278) also include support for outbound connections.) Each security group consists of a list of protocols, ports, and IP address ranges. A security group can apply to multiple instances, and multiple groups can regulate a single instance.
sender	The person or entity sending an email message.
Sender ID	A Microsoft-controlled version of SPF (p. 315). An email authentication and anti-spoofing system. For more information about Sender ID, see Sender ID in Wikipedia.
sending limits	The sending quota (p. 313) and maximum send rate (p. 302) that are associated with every Amazon SES (p. 278) account.
sending quota	The maximum number of email messages that you can send using Amazon SES (p. 278) in a 24-hour period.
server-side encryption (SSE)	The encrypting (p. 293) of data at the server level. Amazon S3 (p. 278) supports three modes of server-side encryption: SSE-S3, in which Amazon S3 manages the keys; SSE-C, in which the customer manages the keys; and SSE-KMS, in which AWS Key Management Service (AWS KMS) (p. 282) manages keys.
service	See Amazon ECS service.
service endpoint	See endpoint.
service health dashboard	A web page showing up-to-the-minute information about AWS service availability. The dashboard is located at http://status.aws.amazon.com/.
service role	An IAM (p. 282) role (p. 311) that grants permissions to an AWS service so it can access AWS resource (p. 310)s. The policies that you attach to the service role determine which AWS resources the service can access and what it can do with those resources.
SES	See Amazon Simple Email Service (Amazon SES).
session	The period during which the temporary security credentials provided by AWS Security Token Service (AWS STS) (p. 283) allow access to your AWS account.
SHA	Secure Hash Algorithm. SHA1 is an earlier version of the algorithm, which AWS has deprecated in favor of SHA256.

shard	Amazon Elasticsearch Service (Amazon ES) (p. 276): A partition of data in an index. You can split an index into multiple shards, which can include primary shards (original shards) and replica shards (copies of the primary shards). Replica shards provide failover, which means that a replica shard is promoted to a primary shard if a cluster node that contains a primary shard fails. Replica shards also can handle requests.
shared AMI	An Amazon Machine Image (AMI) (p. 277) that a developer builds and makes available for others to use.
shutdown action	Amazon EMR (p. 276): A predefined bootstrap action that launches a script that executes a series of commands in parallel before terminating the job flow.
signature	Refers to a *digital signature*, which is a mathematical way to confirm the authenticity of a digital message. AWS uses signatures to authenticate the requests you send to our web services. For more information, to http://aws.amazon.com/security.
SIGNATURE file	AWS Import/Export (p. 282): A file you copy to the root directory of your storage device. The file contains a job ID, manifest file, and a signature.
Signature Version 4	Protocol for authenticating inbound API requests to AWS services in all AWS regions.
Simple Mail Transfer Protocol	See SMTP.
Simple Storage Service	See Amazon Simple Storage Service (Amazon S3).
Single-AZ DB instance	A standard (non-Multi-AZ) DB instance (p. 290) that is deployed in one Availability Zone (p. 280), without a standby replica in another Availability Zone. See Also Multi-AZ deployment.
sloppy phrase search	A search for a phrase that specifies how close the terms must be to one another to be considered a match.
SMTP	Simple Mail Transfer Protocol. The standard that is used to exchange email messages between Internet hosts for the purpose of routing and delivery.
snapshot	Amazon Elastic Block Store (Amazon EBS) (p. 275): A backup of your volume (p. 320)s that is stored in Amazon S3 (p. 278). You can use these snapshots as the starting point for new Amazon EBS volumes or to protect your data for long-term durability. See Also DB snapshot.
SNS	See Amazon Simple Notification Service (Amazon SNS).
Snowball	An AWS Import/Export (p. 282) feature that uses Amazon-owned Snowball appliances for transferring your data. See Also http://aws.amazon.com/importexport.
soft bounce	A temporary email delivery failure such as one resulting from a full mailbox.
software VPN	A software appliance-based VPN connection over the Internet.
sort enabled	Amazon CloudSearch (p. 275): An index field option that enables a field to be used to sort the search results.
sort key	An attribute used to sort the order of partition keys in a composite primary key (also known as a *range attribute*).

	See Also partition key, primary key.
source/destination checking	A security measure to verify that an EC2 instance (p. 292) is the origin of all traffic that it sends and the ultimate destination of all traffic that it receives; that is, that the instance is not relaying traffic. Source/destination checking is enabled by default. For instances that function as gateways, such as VPC (p. 320) NAT (p. 303) instances, source/destination checking must be disabled.
spam	Unsolicited bulk email.
spamtrap	An email address that is set up by an anti-spam (p. 315) entity, not for correspondence, but to monitor unsolicited email. This is also called a *honeypot*.
SPF	Sender Policy Framework. A standard for authenticating email. See Also http://www.openspf.org.
Spot Instance	A type of EC2 instance (p. 292) that you can bid on to take advantage of unused Amazon EC2 (p. 276) capacity.
Spot price	The price for a Spot Instance (p. 315) at any given time. If your maximum price exceeds the current price and your restrictions are met, Amazon EC2 (p. 276) launches instances on your behalf.
SQL injection match condition	AWS WAF (p. 284): An attribute that specifies the part of web requests, such as a header or a query string, that AWS WAF inspects for malicious SQL code. Based on the specified conditions, you can configure AWS WAF to allow or block web requests to AWS resource (p. 310)s such as Amazon CloudFront (p. 275) distributions.
SQS	See Amazon Simple Queue Service (Amazon SQS).
SSE	See server-side encryption (SSE).
SSL	Secure Sockets Layer See Also Transport Layer Security.
stack	AWS CloudFormation (p. 280): A collection of AWS resource (p. 310)s that you create and delete as a single unit. AWS OpsWorks (p. 282): A set of instances that you manage collectively, typically because they have a common purpose such as serving PHP applications. A stack serves as a container and handles tasks that apply to the group of instances as a whole, such as managing applications and cookbooks.
station	AWS CodePipeline (p. 281): A portion of a pipeline workflow where one or more actions are performed.
station	A place at an AWS facility where your AWS Import/Export data is transferred on to, or off of, your storage device.
statistic	One of five functions of the values submitted for a given sampling period (p. 311). These functions are `Maximum`, `Minimum`, `Sum`, `Average`, and `SampleCount`.
stem	The common root or substring shared by a set of related words.
stemming	The process of mapping related words to a common stem. This enables matching on variants of a word. For example, a search for "horse" could return matches for horses, horseback, and horsing, as well as horse. Amazon

	CloudSearch (p. 275) supports both dictionary based and algorithmic stemming.
step	Amazon EMR (p. 276): A single function applied to the data in a job flow (p. 299). The sum of all steps comprises a job flow.
step type	Amazon EMR (p. 276): The type of work done in a step. There are a limited number of step types, such as moving data from Amazon S3 (p. 278) to Amazon EC2 (p. 276) or from Amazon EC2 to Amazon S3.
sticky session	A feature of the Elastic Load Balancing (p. 293) load balancer that binds a user's session to a specific application instance so that all requests coming from the user during the session are sent to the same application instance. By contrast, a load balancer defaults to route each request independently to the application instance with the smallest load.
stopping	The process of filtering stop words from an index or search request.
stopword	A word that is not indexed and is automatically filtered out of search requests because it is either insignificant or so common that including it would result in too many matches to be useful. Stop words are language-specific.
streaming	Amazon EMR (Amazon EMR) (p. 276): A utility that comes with Hadoop (p. 296) that enables you to develop MapReduce executables in languages other than Java. Amazon CloudFront (p. 275): The ability to use a media file in real time—as it is transmitted in a steady stream from a server.
streaming distribution	A special kind of distribution (p. 291) that serves streamed media files using a Real Time Messaging Protocol (RTMP) connection.
Streams	See Amazon Kinesis Streams.
string-to-sign	Before you calculate an HMAC (p. 297) signature, you first assemble the required components in a canonical order. The preencrypted string is the string-to-sign.
string match condition	AWS WAF (p. 284): An attribute that specifies the strings that AWS WAF searches for in a web request, such as a value in a header or a query string. Based on the specified strings, you can configure AWS WAF to allow or block web requests to AWS resource (p. 310)s such as CloudFront (p. 275) distributions.
strongly consistent read	A read process that returns a response with the most up-to-date data, reflecting the updates from all prior write operations that were successful—regardless of the region. See Also data consistency, eventual consistency, eventually consistent read.
structured query	Search criteria specified using the Amazon CloudSearch (p. 275) structured query language. You use the structured query language to construct compound queries that use advanced search options and combine multiple search criteria using Boolean operators.
STS	See AWS Security Token Service (AWS STS).
subnet	A segment of the IP address range of a VPC (p. 320) that EC2 instance (p. 292)s can be attached to. You can create subnets to group instances according to security and operational needs.
Subscription button	An HTML-coded button that enables an easy way to charge customers a recurring fee.

suggester — Amazon CloudSearch (p. 275): Specifies an index field you want to use to get autocomplete suggestions and options that can enable fuzzy matches and control how suggestions are sorted.

suggestions — Documents that contain a match for the partial search string in the field designated by the suggester (p. 317). Amazon CloudSearch (p. 275) suggestions include the document IDs and field values for each matching document. To be a match, the string must match the contents of the field starting from the beginning of the field.

supported AMI — An Amazon Machine Image (AMI) (p. 277) similar to a paid AMI (p. 306), except that the owner charges for additional software or a service that customers use with their own AMIs.

SWF — See Amazon Simple Workflow Service (Amazon SWF).

symmetric encryption — Encryption (p. 293) that uses a private key only.
See Also asymmetric encryption.

synchronous bounce — A type of bounce (p. 285) that occurs while the email servers of the sender (p. 313) and receiver (p. 309) are actively communicating.

synonym — A word that is the same or nearly the same as an indexed word and that should produce the same results when specified in a search request. For example, a search for "Rocky Four" or "Rocky 4" should return the fourth *Rocky* movie. This can be done by designating that `four` and `4` are synonyms for `IV`. Synonyms are language-specific.

T

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

table — A collection of data. Similar to other database systems, DynamoDB stores data in tables.

tag — Metadata that you can define and assign to AWS resource (p. 310)s, such as an EC2 instance (p. 292). Not all AWS resources can be tagged.

tagging — Tagging resources: Applying a tag (p. 317) to an AWS resource (p. 310).

Amazon SES (p. 278): Also called *labeling*. A way to format return path (p. 311) email addresses so that you can specify a different return path for each recipient of a message. Tagging enables you to support VERP (p. 320). For example, if Andrew manages a mailing list, he can use the return paths andrew+recipient1@example.net and andrew+recipient2@example.net so that he can determine which email bounced.

target attribute — Amazon Machine Learning (Amazon ML): The attribute in the input data that contains the “correct” answers. Amazon ML uses the target attribute to learn how to make predictions on new data. For example, if you were building a model for predicting the sale price of a house, the target attribute would be “target sale price in USD.”

target revision — AWS CodeDeploy (p. 281): The most recent version of the application revision that has been uploaded to the repository and will be deployed to the instances in a deployment group. In other words, the application revision

	currently targeted for deployment. This is also the revision that will be pulled for automatic deployments.
task	An instantiation of a task definition (p. 318) that is running on a container instance (p. 288).
task definition	The blueprint for your task. Specifies the name of the task (p. 318), revisions, container definition (p. 288)s, and volume (p. 320) information.
task node	An EC2 instance (p. 292) that runs Hadoop (p. 296) map and reduce tasks, but does not store data. Task nodes are managed by the master node (p. 302), which assigns Hadoop tasks to nodes and monitors their status. While a job flow is running you can increase and decrease the number of task nodes. Because they don't store data and can be added and removed from a job flow, you can use task nodes to manage the EC2 instance capacity your job flow uses, increasing capacity to handle peak loads and decreasing it later. Task nodes only run a TaskTracker Hadoop daemon.
tebibyte	A contraction of tera binary byte, a tebibyte is 2^40 or 1,099,511,627,776 bytes. A terabyte (TB) is 10^12 or 1,000,000,000,000 bytes. 1,024 TiB is a pebibyte (p. 306).
template format version	The version of an AWS CloudFormation (p. 280) template design that determines the available features. If you omit the `AWSTemplateFormatVersion` section from your template, AWS CloudFormation assumes the most recent format version.
template validation	The process of confirming the use of JSON (p. 299) code in an AWS CloudFormation (p. 280) template. You can validate any AWS CloudFormation template using the `cfn-validate-template` command.
temporary security credentials	Authentication information that is provided by AWS STS (p. 283) when you call an STS API action. Includes an access key ID (p. 273), a secret access key (p. 313), a session (p. 313) token, and an expiration time.
throttling	The automatic restricting or slowing down of a process based on one or more limits. Examples: Amazon Kinesis Streams (p. 277) throttles operations if an application (or group of applications operating on the same stream) attempts to get data from a shard at a rate faster than the shard limit. Amazon API Gateway (p. 274) uses throttling to limit the steady-state request rates for a single account. Amazon SES (p. 278) uses throttling to reject attempts to send email that exceeds the sending limits (p. 313).
time series data	Data provided as part of a metric. The time value is assumed to be when the value occurred. A metric is the fundamental concept for Amazon CloudWatch (p. 275) and represents a time-ordered set of data points. You publish metric data points into CloudWatch and later retrieve statistics about those data points as a time-series ordered data set.
time stamp	A date/time string in ISO 8601 format.
TLS	See Transport Layer Security.
tokenization	The process of splitting a stream of text into separate tokens on detectable boundaries such as whitespace and hyphens.
topic	A communication channel to send messages and subscribe to notifications. It provides an access point for publishers and subscribers to communicate with each other.

training datasource	A datasource that contains the data that Amazon Machine Learning uses to train the machine learning model to make predictions.
transition	AWS CodePipeline (p. 281): The act of a revision in a pipeline continuing from one stage to the next in a workflow.
Transport Layer Security	A cryptographic protocol that provides security for communication over the Internet. Its predecessor is Secure Sockets Layer (SSL).
trust policy	An IAM (p. 282) policy (p. 306) that is an inherent part of an IAM role (p. 311). The trust policy specifies which principal (p. 307)s are allowed to use the role.
trusted signers	AWS account (p. 274)s that the CloudFront (p. 275) distribution owner has given permission to create signed URLs for a distribution's content.
tuning	Selecting the number and type of AMIs (p. 277) to run a Hadoop (p. 296) job flow most efficiently.
tunnel	A route for transmission of private network traffic that uses the Internet to connect nodes in the private network. The tunnel uses encryption and secure protocols such as PPTP to prevent the traffic from being intercepted as it passes through public routing nodes.

U

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

unbounded	The number of potential occurrences is not limited by a set number. This value is often used when defining a data type that is a list (for example, `maxOccurs="unbounded"`), in Web Services Description Language (p. 321).
unit	Standard measurement for the values submitted to Amazon CloudWatch (p. 275) as metric data. Units include seconds, percent, bytes, bits, count, bytes/second, bits/second, count/second, and none.
unlink from VPC	The process of unlinking (or detaching) an EC2-Classic instance (p. 298) from a ClassicLink-enabled VPC (p. 320). See Also ClassicLink, link to VPC.
usage report	An AWS record that details your usage of a particular AWS service. You can generate and download usage reports from http://aws.amazon.com/usage-reports/.
user	A person or application under an account (p. 274) that needs to make API calls to AWS products. Each user has a unique name within the AWS account, and a set of security credentials not shared with other users. These credentials are separate from the AWS account's security credentials. Each user is associated with one and only one AWS account.

V

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) |

N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

validation	See template validation.
value	Instances of attributes (p. 280) for an item, such as cells in a spreadsheet. An attribute might have multiple values. Tagging resources: A specific tag (p. 317) label that acts as a descriptor within a tag category (key). For example, you might have EC2 instance (p. 292) with the tag key of *Owner* and the tag value of *Jan*. You can tag an AWS resource (p. 310) with up to 10 key–value pairs. Not all AWS resources can be tagged.
Variable Envelope Return Path	See VERP.
verification	The process of confirming that you own an email address or a domain so that you can send email from or to it.
VERP	Variable Envelope Return Path. A way in which email sending applications can match bounce (p. 285)d email with the undeliverable address that caused the bounce by using a different return path (p. 311) for each recipient. VERP is typically used for mailing lists. With VERP, the recipient's email address is embedded in the address of the return path, which is where bounced email is returned. This makes it possible to automate the processing of bounced email without having to open the bounce messages, which may vary in content.
versioning	Every object in Amazon S3 (p. 278) has a key and a version ID. Objects with the same key, but different version IDs can be stored in the same bucket (p. 285). Versioning is enabled at the bucket layer using PUT Bucket versioning.
virtualization	Allows multiple guest virtual machines (VM) to run on a host operating system. Guest VMs can run on one or more levels above the host hardware, depending on the type of virtualization. See Also PV virtualization, HVM virtualization.
virtual private cloud	See VPC.
virtual private gateway	See VPG.
visibility timeout	The period of time that a message is invisible to the rest of your application after an application component gets it from the queue. During the visibility timeout, the component that received the message usually processes it, and then deletes it from the queue. This prevents multiple components from processing the same message.
volume	A fixed amount of storage on an instance (p. 298). You can share volume data between container (p. 288)s and persist the data on the container instance (p. 288) when the containers are no longer running.
VPC	Virtual private cloud. An elastic network populated by infrastructure, platform, and application services that share common security and interconnection.
VPC endpoint	A feature that enables you to create a private connection between your VPC (p. 320) and an another AWS service without requiring access over the Internet, through a NAT (p. 303) instance, a VPN connection (p. 321), or AWS Direct Connect (p. 281).
VPG	Virtual private gateway. The Amazon side of a VPN connection (p. 321) that maintains connectivity. The internal interfaces of the virtual private gateway

	connect to your VPC (p. 320) via the VPN attachment and the external interfaces connect to the VPN connection, which leads to the customer gateway (p. 289).
VPN CloudHub	See AWS VPN CloudHub.
VPN connection	Amazon Web Services (AWS) (p. 278): The IPsec connection between a VPC (p. 320) and some other network, such as a corporate data center, home network, or co-location facility.

W

Numbers and Symbols (p. 273) | A (p. 273) | B (p. 284) | C (p. 285) | D (p. 289) | E (p. 292) | F (p. 295) | G (p. 296) | H (p. 296) | I (p. 297) | J (p. 299) | K (p. 299) | L (p. 300) | M (p. 301) | N (p. 303) | O (p. 304) | P (p. 305) | Q (p. 308) | R (p. 309) | S (p. 311) | T (p. 317) | U (p. 319) | V (p. 319) | W (p. 321) | X, Y, Z (p. 321)

WAM	See Amazon WorkSpaces Application Manager (Amazon WAM).
web access control list	AWS WAF (p. 284): A set of rules that defines the conditions that AWS WAF searches for in web requests to AWS resource (p. 310)s such as Amazon CloudFront (p. 275) distributions. A web access control list (web ACL) specifies whether to allow, block, or count the requests.
Web Services Description Language	A language used to describe the actions that a web service can perform, along with the syntax of action requests and responses. Your SOAP or other toolkit interprets a WSDL file to provide your application access to the actions provided by the web service. For most toolkits, your application calls a service action using routines and classes provided or generated by the toolkit.

X, Y, Z

X.509 certificate	An digital document that uses the X.509 public key infrastructure (PKI) standard to verify that a public key belongs to the entity described in the certificate (p. 286).
yobibyte	A contraction of yotta binary byte, a yobibyte is 2^80 or 1,208,925,819,614,629,174,706,176 bytes. A yottabyte (YB) is 10^24 or 1,000,000,000,000,000,000,000,000 bytes.
zebibyte	A contraction of zetta binary byte, a zebibyte is 2^70 or 1,180,591,620,717,411,303,424 bytes. A zettabyte (ZB) is 10^21 or 1,000,000,000,000,000,000,000 bytes. 1,024 ZiB is a yobibyte (p. 321).
zone awareness	Amazon Elasticsearch Service (Amazon ES) (p. 276): A configuration that distributes nodes in a cluster across two Availability Zone (p. 280)s in the same region. Zone awareness helps to prevent data loss and minimizes downtime in the event of node and data center failure. If you enable zone awareness, you must have an even number of data instances in the instance count, and you also must use the Amazon Elasticsearch Service Configuration API to replicate your data for your Elasticsearch cluster.