***Test Plan***

By: Matthew Lake (2716928)

1. **Software to Test**

I will be testing the OpenSearch client Python library called opensearch-py. OpenSearch is a document-based NoSQL database. OpenSearch databases are mainly used for search and analytics purposes. The opensearch-py library is used to set up and manage an OpenSearch cluster and carry out document queries. In addition, the library also provides a DSL (Domain Specific Language) interface for writing search queries.

1. **Functions to Test**

I am going to test a wide variety of the opensearch-py library’s key functionalities. I am going to test various functions related to setting up and managing an OpenSearch cluster, document CRUD (Create, Read, Update, Delete) operations, searching, and aggregating documents.

* 1. **Initializing an OpenSearch Cluster Client**

For this function I will be testing the opensearch-py library’s ability to connect to a local OpenSearch cluster with different client configurations. The cluster client is used to initiate and instruct an OpenSearch cluster to perform tasks like indexing documents or searching for documents.

* 1. **Creating a Cluster Index**

For this function I will be testing the opensearch-py library’s ability to create a new cluster index. An index is a collection of shards that hold documents. To create an index, I need to have a client cluster initialized. A cluster can be created with different configurations containing a JSON formatted list of settings and a JSON formatted index mapping.

* 1. **Modifying a Cluster Index**

For this function I will be testing the opensearch-py library’s ability to modify an existing cluster index. Before I can test modifying an existing index, I need to set up an index outside of the tests and initialize a client’s cluster. I will test the library’s ability to edit the index’s mapping and settings.

* 1. **Deleting a Cluster Index**

For this function I will be testing the opensearch-py library’s ability to delete an existing cluster index. Before I can test deleting an existing index, I need to set up an index outside of the tests and initialize a cluster client.

* 1. **Reindexing a Cluster Index**

For this function I will be testing the opensearch-py library’s ability to reindex an existing cluster index. Reindexing entails creating a new cluster index and copying the existing documents over to the new index. I will test filtering the index’s documents by using the OpenSearch Query DSL. Before I can test reindexing an existing index, I need to set up an index outside of the tests, seed some documents into the test index, and initialize a client’s cluster.

* 1. **Creating an Index Alias**

For this function I will be testing the opensearch-py library’s ability to create a new cluster index alias. An index alias is used to hide or simplify an index’s name. Before I can test creating an index alias, I need to set up an index outside of the tests and initialize a client’s cluster.

* 1. **Updating an Index Alias**

For this function I will be testing the opensearch-py library’s ability to update a cluster index alias. Before I can test updating an index alias, I need to set up an index outside of the tests, create an index alias, and initialize a client’s cluster.

* 1. **Deleting an Index Alias**

For this function I will be testing the opensearch-py library’s ability to deleting a cluster index alias. Before I can test deleting an index alias, I need to set up an index outside of the tests, create an index alias, and initialize a client’s cluster.

* 1. **Creating and Indexing a Document**

For this function I will be testing the opensearch-py library’s ability to create a document using the DSL interface and index it into an existing cluster index. Before I can test this functionality, I need to set up a cluster index with different mappings and initialize a client’s cluster.

* 1. **Bulk Document Indexing**

For this function I will be testing the opensearch-py library’s ability to index a collection of documents at once using the bulk indexing method. Before I can test this functionality, I need to set up a cluster index, create a collection of documents, and initialize a cluster client.

* 1. **Retrieving a Document by its Identifier**

For this function I will be testing the opensearch-py library’s ability to retrieve an existing document by its identifier. Before I can test this functionality, I need to set up an index outside of the tests, seed some documents into the test index, and initialize a client’s cluster.

* 1. **Update a Document**

For this function I will be testing the opensearch-py library’s ability to update an existing document. Before I can test this functionality, I need to set up an index outside of the tests, seed some documents into the test index, and initialize a client’s cluster.

* 1. **Bulk Document Updating**

For this function I will be testing the opensearch-py library’s ability to update a collection of documents at once using the bulk updating method. This involves creating a script to send to the cluster for it to open and process the bulk update request. Before I can test this functionality, I need to set up a cluster index, create a collection of documents, and initialize a cluster client.

* 1. **Deleting a Document**

For this function I will be testing the opensearch-py library’s ability to delete an existing document. Before I can test this functionality, I need to set up a cluster index, index a document, and initialize a cluster client.

* 1. **Searching for Documents**

For this function I will be testing the opensearch-py library’s ability to create different search queries using the DSL interface, pass the queries to the cluster, and deserialize the response from the cluster. For this functionality I will be using a set of different types of queries. The different queries I will be using during the tests are Full Text, Term Level, Boolean, and Geo. Before I can test this functionality, I need to set up a cluster index, create a collection of documents, and initialize a cluster client.

* 1. **Aggregation**

For this function I will be testing the opensearch-py library’s ability to create different aggregations using the DSL interface, pass the aggregation queries to the cluster, and deserialize the response from the cluster. For testing this functionality, I will be using Bucket and Metric aggregations in my tests. Before I can test this functionality, I need to set up a cluster index, create a collection of documents, and initialize a cluster client.

1. **Tests to be Performed**

For each function there will be a set of black box integration tests. The tests are listed below.

* 1. **Initializing an OpenSearch Cluster Client**
* Test that a client with the correct OpenSearch cluster address is able to ping the cluster.
* Test that a client with an incorrect OpenSearch cluster address is not able to ping the cluster and throws an error message.
  1. **Creating a Cluster Index**
* Test that the library client can correctly create a cluster index using correctly JSON formatted settings and mappings.
* Test that the library client is unable to create a cluster index using incorrectly formatted JSON for the settings and mappings and throws an error message
* Test that the library client is unable to create a cluster index using an incorrect index setting and throws an error message.
* Test that the library client is unable to create a cluster index using an incorrect index mapping type and throws an error message.
  1. **Modifying a Cluster Index**
* Test that the library client can successfully modify an existing index’s settings and add a valid setting.
* Test that the library client can successfully modify an existing index’s mapping and add an attribute with a valid mapping type.
* Test that the library client cannot modify an existing index’s settings by trying to add an invalid setting then throws an error message.
* Test that the library client cannot modify an existing index’s mapping by trying to add an invalid attribute with an incorrect mapping type then throws an error message.
* Test that the client cannot modify a non-existing index and throws an error message.
  1. **Deleting a Cluster Index**
* Test that the library client can successfully delete an existing index.
* Test that the library client cannot delete a non-existing index and throws an error message.
  1. **Reindexing a Cluster Index**
* Test that the library client successfully reindexes an existing index by moving all its documents over to a new index.
* Test that the library client successfully reindexes an existing index by moving all its documents that fall within a valid DSL query filter.
* Test that the library client cannot reindex a non-existing index and throws an error message.
* Test that the library client cannot reindex an existing index with an invalid DSL query filter and throws an error message.
  1. **Creating an Index Alias**
* Test that the library client can successfully create an index alias for an existing index.
* Test that the library client cannot create an index alias for a non-existing index and throws an error message.
  1. **Updating an Index Alias**
* Test that the library client can successfully update an index alias for an existing index.
* Test that the library client cannot update an index alias for a non-existing index and throws an error message.
  1. **Deleting an Index Alias**
* Test that the library client can successfully delete an existing index alias.
* Test that the library client cannot delete a non-existing index alias and throws an error message.
  1. **Creating and Indexing a Document**
* Test that the library client can successfully create a document with the DSL interface and index a document.
* Test that the library client can successfully create a document with the DSL interface.
* Test that the library client can successfully index a correctly made document.
* Test that the library client cannot create a document with an incorrect mapping type and throws an error message.
* Test that the library client cannot index a document with an incorrect attribute name and throws an error message.
* Test that the library client cannot index a document for a non-existing index and throws an error message.
  1. **Bulk Document Indexing**
* Test that the library client can index a collection of correctly formatted documents.
* Test that the library client cannot index a collection of incorrectly formatted documents and throws an error message.
* Test that the library client cannot index a collection of documents for a non-existing index and throws an error message.
  1. **Retrieving a Document by its Identifier**
* Test that the library client can successfully retrieve an existing index document by its identifier.
* Test that the library client gets no documents back for a non-existing document identifier and throws an error message.
* Test that the library client cannot retrieve a document for a non-existing index and throws an error message.
  1. **Update a Document**
* Test that the library client can successfully update an existing document’s attribute.
* Test that the library client cannot update an existing document with a non-existing attribute and throws an error message.
* Test that the library client cannot update a non-existing document and throws an error message.
* Test that the library client cannot update a document for a non-existing index and throws an error message.
  1. **Bulk Document Updating**
* Test that the library client can update a collection of documents that contain a certain attribute value.
* Test that the library client cannot update a collection of documents with an invalid script and throws an error message.
* Test that the library client cannot update a collection of documents for a non-existing index and throws an error message.
  1. **Deleting a Document**
* Test that the library client can successfully delete an existing index document.
* Test that the library client cannot delete a non-existing document and throws an error message.
  1. **Searching for Documents**
* Test that the library client can perform a set of full text queries and correctly deserializes the cluster response. Validate that the documents in the response match the expected list of documents.
* Test that the library client can perform a set of term level queries and correctly deserializes the cluster response. Validate that the documents in the response match the expected list of documents.
* Test that the library client can perform a set of Boolean queries and correctly deserializes the cluster response. Validate that the documents in the response match the expected list of documents.
* Test that the library client can perform a set of Geo queries and correctly deserializes the cluster response. Validate that the documents in the response match the expected list of documents.
* Test that the library client throws an error message for an invalid Full Text query.
* Test that the library client throws an error message for an invalid Term Level query.
* Test that the library client throws an error message for an invalid Boolean query.
* Test that the library client throws an error message for an invalid Geo query.
  1. **Aggregation**
* Test that the library client can perform a bucket aggregation and correctly deserializes the cluster response. Validate that the number of documents that fall under a query matches the expected number of documents.
* Test that the library client can perform a metric aggregation and correctly deserializes the cluster response. Validate that the number of documents that fall under a query matches the expected number of documents.

1. **Testing Strategy**

For testing the functions listed previously I am going to do automated testing. For automated testing I am going to use Docker to create two containers. The first container will be a Python container with the needed libraries installed that holds my test suite. The second container will be an OpenSearch container containing a test cluster. For each integration test within the test suite, I will set up the OpenSearch cluster with the prerequisite things needed and conduct the test. After the test has completed the cluster will be restored back to its initial state.

1. **Testing Schedule**

Each of the tests for each function will require some setup in terms of creating an index, creating documents, ect.. All the tests will be created performed by myself. I expect the creation of the testing environment will take a day or so. I expect the creation of each test suite for each function will take a day or so each. I am expecting to learn more about OpenSearch and spend some time ironing out issues that arise.