***Test Case Specification***

By: Matthew Lake (2716928)

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

|  |  |
| --- | --- |
| Test Case Identifier | TC 001 |
| Test Items/feature | Opensearch-py client instance is able to ping an OpenSearch cluster. |
| Preconditions | An OpenSearch cluster is running. |
| Test Steps | 1. Initialize client with the correct cluster address and port number. 2. Ping the cluster using the .ping() method. 3. Verify that the response of .ping() is true. |
| Output Specifications | The method client.ping() returns true. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 002 |
| Test Items/feature | Opensearch-py client instance is not able to ping an OpenSearch Cluster. |
| Preconditions | An OpenSearch cluster is running. |
| Test Steps | 1. Initialize client with an incorrect cluster address and port number. 2. Ping the cluster using the .ping() method. 3. Verify that the response of .ping() is false. |
| Output Specifications | The method client.ping() returns false. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Creating a Cluster Index**

|  |  |
| --- | --- |
| Test Case Identifier | TC 003 |
| Test Items/feature | Opensearch-py client can create an index with correct index settings and a mapping. |
| Preconditions | An OpenSearch cluster is running, and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string and a dictionary body that contains settings and mapping. 2. Use the client method .indices.create(index=’index\_name’, body={body}) to create the index. 3. Check the response of calling that method for the acknowledgement and the index name. 4. Check the responses of the methods .get\_settings(index=’index\_name’) and .get\_mapping(index=’index\_name’) to verify the outputs match the dictonary body’s settings and mapping. |
| Output Specifications | .indices.create() returns an acknowledgement and the index name in a dictionary. .get\_settings() and .get\_mapping() responses should match the dictionary body’s settings and mapping. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 004 |
| Test Items/feature | Opensearch-py client cannot create an index with an incorrectly formatted index mapping. |
| Preconditions | An OpenSearch cluster is running, and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string and a dictionary body containing an incorrectly formatted index mapping. 2. Use the client method .indices.create(index=’index\_name’, body={body}) to create the index. 3. Verify that calling that method results in an incorrect index mapping format error. |
| Output Specifications | .indices.create() returns an incorrect mapping format error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 005 |
| Test Items/feature | Opensearch-py client cannot create an index with an incorrect index mapping type. |
| Preconditions | An OpenSearch cluster is running, and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string and a dictionary body containing an incorrectly formatted index mapping. 2. Use the client method .indices.create(index=’index\_name’, body={body}) to create the index. 3. Verify that calling that method results in an incorrect index mapping type error. |
| Output Specifications | .indices.create() returns an incorrect mapping type error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 006 |
| Test Items/feature | Opensearch-py client cannot create an index with an incorrect index setting. |
| Preconditions | An OpenSearch cluster is running, and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string and a dictionary body containing an incorrectly spelled index setting. 2. Use the client method .indices.create(index=’index\_name’, body={body}) to create the index. 3. Verify that calling that method results in an incorrect index setting error. |
| Output Specifications | .indices.create() returns an incorrect index setting error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Modifying a Cluster Index**

|  |  |
| --- | --- |
| Test Case Identifier | TC 007 |
| Test Items/feature | Opensearch-py client can modify an existing index settings by adding a valid index setting. |
| Preconditions | An OpenSearch cluster is running with an index defined and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize the defined index name string and a dictionary body containing a valid index setting like refresh\_interval. 2. Use the client method .indices.put\_settings(index=’index\_name’, body={body}) to update the index with the new setting. 3. Verify the response of the client method .get\_settings(index=’defined\_index\_name’) has the refresh\_interval setting in it. |
| Output Specifications | .indices.get\_settings() should contain the index setting refresh\_interval that was just added. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 008 |
| Test Items/feature | Opensearch-py client can modify an existing index mapping by adding a valid typed attribute to the mapping. |
| Preconditions | An OpenSearch cluster is running with an index defined and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize the defined index name string and a dictionary body containing a valid index mapping addition with the type text. 2. Use the client method .indices.put\_mapping(index=’index\_name’, body={body}) to update the index with the new attribute. 3. Verify that in the response of the client method .indices.get\_mapping(index=’defined\_index\_name’) should contain the new attribute. |
| Output Specifications | .indices.get\_mapping(index=’defined\_index\_name’) should contain the index mapping addition. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 009 |
| Test Items/feature | Opensearch-py client cannot modify an existing index settings with an invalid index setting. |
| Preconditions | An OpenSearch cluster is running with an index defined and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize the defined index name string and a dictionary body containing a not valid index setting like refresh\_intervals. 2. Use the client method .indices.put\_settings(index=’index\_name’, body={body}) to update the index with the new incorrect index setting. 3. Verify that calling the client .indices.put\_settings() method results in an incorrect index setting error. |
| Output Specifications | .indices.put\_settings() should raise an incorrect index setting error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 010 |
| Test Items/feature | Opensearch-py client cannot modify an existing index mapping with an invalid index attribute type. |
| Preconditions | An OpenSearch cluster is running with an index defined and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize the defined index name string and the dictionary body containing a not valid index mapping type like ‘txet’. 2. Use the client method .indices.put\_mapping(index=’index\_name’, body={body}) to update the index with the new incorrect index mapping addition. 3. Verify that calling the client .indices.put\_settings() method results in an incorrect index mapping type error. |
| Output Specifications | .indices.put\_mapping() should raise an incorrect index mapping attribute type error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 011 |
| Test Items/feature | Opensearch-py client cannot modify an non existent cluster index. |
| Preconditions | An OpenSearch cluster is running and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize a random index name string and dictionary body containing valid index settings additions. 2. Use the client method .indices.put\_setting(index=’index\_name’, body={body}) to update the index settings. 3. Verify that calling the client .indices.put\_settings() method results in an missing index error. |
| Output Specifications | .indices.put\_settings() should raise a missing index error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Deleting a Cluster Index**

|  |  |
| --- | --- |
| Test Case Identifier | TC 012 |
| Test Items/feature | Opensearch-py client can delete an existing index. |
| Preconditions | An OpenSearch cluster is running with an index defined and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Use the client method .indices.delete(index=’predefined index name’) to delete the index. 3. Verify that the response of that method is an acknowledgement dictionary. |
| Output Specifications | The response of .indices.delete(index=’predefined index name’) is {'acknowledged': True}. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 013 |
| Test Items/feature | Opensearch-py client cannot delete a non-existing index. |
| Preconditions | An OpenSearch cluster is running, and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string with a random value. 2. Use the client method .indices.delete(index=’name’) to delete the non-existing index. 3. Verify that calling the client method .indices.delete() results in an 404 error. |
| Output Specifications | The response of .indices.delete(index=’name’) is a 404 error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Reindexing a Cluster Index**

|  |  |
| --- | --- |
| Test Case Identifier | TC 014 |
| Test Items/feature | Opensearch-py client can reindex an index. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is made with 5 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name and initialize the new index’s name as a string for the reindexed documents. 2. Put the created strings into a reindex body dictionary. 3. Use the client method .reindex (body= reindex\_body\_dict) to reindex the index. 4. Verify that the failures key’s value in the response dictionary is empty and the created and total keys values are 5. |
| Output Specifications | The response of the .reindex() should be a dictionary with a ‘failures’ key and the value should be an empty list. Also, the value for ‘created’ and ‘total’ should be 5 for the number of documents. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 015 |
| Test Items/feature | Opensearch-py client can reindex an index and only move over documents that meet a certain criteria. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is made with 5 documents. 3 documents should have an attribute named ‘condition’ and the value should be true. The other 2 documents should have the value false. |
| Test Steps | 1. Initialize an index name string with the running index’s name and initialize the new index’s name as a string for the reindexed documents. Initialize a query dictionary and use a Boolean query type looking for ‘condition’ to be true. 2. Put the created strings and the query dictionary into a reindex body dictionary. 3. Use the client method .reindex (body= reindex\_body\_dict) to reindex the index 4. Verify that the failures key’s value in the response dictionary is empty and the created and total keys values are 5. |
| Output Specifications | The response of the .reindex() should be a dictionary with a ‘failures’ key and the value should be an empty list. Also, the value for ‘created’ and ‘total’ should be 3 for the number of documents. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 016 |
| Test Items/feature | Opensearch-py client cannot reindex a non-existing index. |
| Preconditions | An OpenSearch cluster is running, and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string with a random value and initialize the new index’s name as a string for the reindexed documents. 2. Put the created strings into a reindex body dictionary. 3. Use the client method .reindex (body= reindex\_body\_dict) to reindex the index. 4. Verify that the .reindex() method raised a 404 error. |
| Output Specifications | The response of .indices.reindex() is a 404 error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 017 |
| Test Items/feature | Opensearch-py client cannot reindex an index with an invalid DSL query. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is made with 5 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name and initialize the new index’s name as a string for the reindexed documents. Initialize a query dictionary with an ill formed query. 2. Put the created strings and query dictionary into a reindex body dictionary. 3. Use the client method .reindex (body= reindex\_body\_dict) to reindex the index. 4. Verify that the .reindex() method raised an incorrect query error. |
| Output Specifications | The response of the .reindex() should be a dictionary with a ‘failures’ key and the value should not be an empty list. There should be a query error in the list. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Creating an Index Alias**

|  |  |
| --- | --- |
| Test Case Identifier | TC 018 |
| Test Items/feature | Opensearch-py client can create an index alias for an existing index. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created. |
| Test Steps | 1. Initialize an index name string with the running index’s name and initialize the new index’s alias as a string. 2. Use the client method .indices.put\_alias(index=’index\_name’, name=’alias\_name’) to create an alias for the index. 3. Verify that the response of . indices.put\_alias() contains an acknowledgement dictionary. 4. Verify that the new index alias is in the response of the client method .get\_alias(). |
| Output Specifications | The response of the .indices.put\_alias() method should be an acknowledgement dictionary. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 019 |
| Test Items/feature | Opensearch-py client cannot create an index alias for a non-existing index. |
| Preconditions | An OpenSearch cluster is running, and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string with a random string and initialize an index alias string. 2. Use the client method .indices.put\_alias(index=’index\_name’, name=’alias\_name’) to create an alias for the index. 3. Verify that the .put\_alias() method raised a 404 error. |
| Output Specifications | The response of .indices.put\_alias() is a 404 error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Updating an Index Alias**

|  |  |
| --- | --- |
| Test Case Identifier | TC 020 |
| Test Items/feature | Opensearch-py client can update an index alias for an existing index. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with an alias. |
| Test Steps | 1. Initialize an index name string with the running index’s name and initialize the new index’s alias as a string. 2. Put those created strings into a dictionary body. 3. Use the client method .indices.update\_aliases(body={body}) to update the alias. 4. Verify that the response of . indices.update\_alias() contains an acknowledgement dictionary. 5. Verify that the newly updated index alias is in the response of the client method .get\_alias(). |
| Output Specifications | The response of the .indices.update\_aliases() method should be an acknowledgement dictionary. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 021 |
| Test Items/feature | Opensearch-py client cannot update an index alias for a non-existing index. |
| Preconditions | An OpenSearch cluster is running, and a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string with a random string and initialize an index alias string. 2. Put those created strings into a dictionary body. 3. Use the client method .indices.update\_aliases(body={body}) to update the alias. 4. Verify that the .update\_aliases() method raised a 404 error. |
| Output Specifications | The response of .indices.update\_aliases() is a 404 error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Deleting an Index Alias**

|  |  |
| --- | --- |
| Test Case Identifier | TC 022 |
| Test Items/feature | Opensearch-py client can delete an existing index alias. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with an alias. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Use the client method .indices.delete\_alias(index=’index\_name’) to delete the index alias. 3. Verify that the response of . indices.delete\_alias() contains an acknowledgement dictionary. 4. Verify that the old index alias is not in the response of the method .get\_alias(). |
| Output Specifications | The response of the .indices.delete\_alias() method should be an acknowledgement dictionary. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 023 |
| Test Items/feature | Opensearch-py client throws a missing index alias error trying to delete a missing index alias. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created without an alias. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Use the client method .indices.delete\_alias(index=’index\_name’) to delete the index alias. 3. Verify that the .delete\_alias() method raised a 404 error. |
| Output Specifications | The response of the .indices.delete\_alias() method should throw a missing index alias error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

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

|  |  |
| --- | --- |
| Test Case Identifier | TC 024 |
| Test Items/feature | Opensearch-py client can index a document created with its DSL interface. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Use client methods to make a Document object. Methods such as Integer(), Text(), ect.. 3. Verify that the Document is correct 4. Use the client method .index(index=’index\_name’, body={Document as a dictionary}). 5. Verify that the document got indexed and the document is correct with the client method .get(id=document\_id). |
| Output Specifications | The client method .get(id=document\_id) should not be empty and contain the created document. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 025 |
| Test Items/feature | Opensearch-py client cannot index a document created incorrectly with the DSL interface. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Use client methods to make a Document object. Methods such as Integer(), Text(), ect.. But do not use the correct method for one of the attributes. 3. Verify that the Document is incorrect. 4. Use the client method .index(index=’index\_name’, body={Document as a dictionary}). 5. Verify that a mapping error gets raised when calling the .index() method. |
| Output Specifications | The client method .index() should throw a mapping error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 026 |
| Test Items/feature | Opensearch-py client cannot index a document created incorrectly with an incorrectly spelled attribute name. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Use client methods to make a Document object. Methods such as Integer(), Text(), ect.. But misspell a attribute name. 3. Verify that the Document is incorrect. 4. Use the client method .index(index=’index\_name’, body={Document as a dictionary}). 5. Verify that a mapping error gets raised when calling the .index() method. |
| Output Specifications | The client method .index() should throw a mapping error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 027 |
| Test Items/feature | Opensearch-py client cannot index a document in a missing index. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string with a random value. 2. Use client methods to make a Document object. Methods such as Integer(), Text(), ect.. 3. Use the client method .index(index=’index\_name’, body={Document as a dictionary}). 4. Verify that a missing index error gets raised when calling the .index() method. |
| Output Specifications | The client method .index() should throw a missing index error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Bulk Document Indexing**

|  |  |
| --- | --- |
| Test Case Identifier | TC 028 |
| Test Items/feature | Opensearch-py client can index multiple valid documents at once using the bulk index interface. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Use client methods to make 2 Document object. Methods such as Integer(), Text(), ect.. 3. Verify that the Documents are correct. 4. Use the client method .bulk([document dictionaries]). 5. Verify that the documents got indexed and the documents are correct with the client method .get(). |
| Output Specifications | The client method .get() should be a list of length 2 with the 2 documents that were just indexed. The Documents should be correct too. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 029 |
| Test Items/feature | Opensearch-py client throws a BulkIndexError error when there is an invalid document trying to be indexed with the bulk method. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Use client methods to make 2 Document object. Methods such as Integer(), Text(), ect.. Make sure one of the documents are created incorrectly. 3. Verify that the 1 of the documents are valid and the other in invalid. 4. Use the client method .bulk([document dictionaries]) 5. Verify that a BulkIndexError error gets raised. 6. Verify that the 1 valid got indexed with the .get() method. |
| Output Specifications | The client method raises a BulkIndexError error. The index should have the 1 valid document in there. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 030 |
| Test Items/feature | Opensearch-py client cannot index multiple document in a missing index. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string with a random value. 2. Use client methods to make a Document object. Methods such as Integer(), Text(), ect.. 3. Use the client method .bulk([document dictionary]). 4. Verify that a missing index error gets raised when calling the .bulk() method. |
| Output Specifications | The client method .bulk() should throw a missing index error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

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

|  |  |
| --- | --- |
| Test Case Identifier | TC 031 |
| Test Items/feature | Opensearch-py client can fetch an indexed document by its ID. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 2 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name. Initialize an integer with a document ID. 2. Use the client method .get(index=’index\_name’, id=document\_id). 3. Verify that the response of .get() is a list with a length of 1. Verify that the document is the intended one. |
| Output Specifications | The client method .get() should be a list of length 1 with the intended document. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 032 |
| Test Items/feature | Opensearch-py client throws a missing index error when trying to fetch a document from a missing index. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string with a random value. 2. Use the client method .get(index=’index\_name’, id=document\_id). 3. Verify that a missing index error gets raised when calling the .get() method. |
| Output Specifications | The client method .get() should throw a missing index error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Update a Document**

|  |  |
| --- | --- |
| Test Case Identifier | TC 033 |
| Test Items/feature | Opensearch-py client can update one of the document’s attributes. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with a document. |
| Test Steps | 1. Initialize an index name string with the running index’s name. Initialize an integer with a document ID. 2. Initialize a dictionary with the document attribute update. 3. Use the client method .update(index=’index\_name’, id=document\_id, body={document update dictionary}). 4. Verify that the response of the .update() is a dictionary and the ‘\_version’ key’s value is 2. Also, the ‘\_shards’ key’s key `successful` is equal to 1. 5. Verify that the response of .get(index=’index\_name’, id=document\_id) is a list with a length of 1. Verify that the document is updated with the new attribute value. |
| Output Specifications | The client method .update() should be a dictionary and the ‘\_version’ key’s value is 2. Also, the ‘\_shards’ key’s key `successful` is equal to 1. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 034 |
| Test Items/feature | Opensearch-py client cannot update a document with a unknown ID. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with a document. |
| Test Steps | 1. Initialize an index name string with the running index’s name. Initialize an integer with a random document ID. 2. Initialize a dictionary with the document attribute update. 3. Use the client method .update(index=’index\_name’, id=document\_id, body={document update dictionary}). 4. Verify that the response of the .update() is a dictionary and, the ‘\_shards’ key’s key `failed` is equal to 1. |
| Output Specifications | The client method .update() should be a dictionary and, the ‘\_shards’ key’s key `failed` is equal to 1. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 035 |
| Test Items/feature | Opensearch-py client throws a missing index error when trying to update a document from a missing index. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string with a random index’s name. Initialize an integer with a random document ID. 2. Initialize a dictionary with the document attribute update. 3. Use the client method .update(index=’index\_name’, id=document\_id, body={document update dictionary}). 4. Verify that a missing index error gets raised when calling the .update() method. |
| Output Specifications | The client method .update() should throw a missing index error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Bulk Document Updating**

|  |  |
| --- | --- |
| Test Case Identifier | TC 036 |
| Test Items/feature | Opensearch-py client can update multiple documents at once using the bulk update method. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 3 documents. |
| Test Steps | 1. Initialize a list of dictionaries containing the update operation type, running index name, document id, and a script for OpenSearch to update the documents. 2. Use the method helpers.bulk(client=client\_instance, body={initialized\_dictionary}. 3. Verify that the documents are updated by using the .get() client method. |
| Output Specifications | The bulk update method helpers.bulk() should complete successfully and the documents should be updated. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 037 |
| Test Items/feature | Opensearch-py client cannot update multiple documents at once using the bulk update method with an invalid script. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 3 documents. |
| Test Steps | 1. Initialize a list of dictionaries containing the update operation type, running index name, document id, and a script for OpenSearch to update the documents. 2. Use the method helpers.bulk(client=client\_instance, body={initialized\_dictionary}. 3. Verify that the helpers.bulk() method throws a invalid script error. |
| Output Specifications | The bulk update method helpers.bulk() should throw a invalid script error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 038 |
| Test Items/feature | Opensearch-py client throws a missing index error when trying to bulk update documents in a missing index. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize a list of dictionaries containing the update operation type, running index name, document id, and a script for OpenSearch to update the documents. 2. Use the method helpers.bulk(client=client\_instance, body={initialized\_dictionary}. 3. Verify that the helpers.bulk() method throws a missing index error. |
| Output Specifications | The method helpers.bulk() should throw a missing index error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Deleting a Document**

|  |  |
| --- | --- |
| Test Case Identifier | TC 039 |
| Test Items/feature | Opensearch-py client can delete an indexed document by its ID. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 2 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name. Initialize an integer with a document ID. 2. Use the client method .delete(index=’index\_name’, id=document\_id) to delete 1 of the 2 documents. 3. Verify that the response of the .delete() method is an acknowledgement dictionary. 4. Use the client method .get() to get the other document and verify it is not deleted. |
| Output Specifications | The client method .delete() returns a acknowledgement dictionary and only 1 of the documents are deleted. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 040 |
| Test Items/feature | Opensearch-py client throws an missing index error when trying to delete a document from a missing index. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize an index name string with a random value. Initialize an integer with a random document ID. 2. Use the client method .delete(index=’index\_name’, id=document\_id). 3. Verify that a missing index error gets raised when calling the .delete() method. |
| Output Specifications | The client method .delete() should throw a missing index error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Searching for Documents**

|  |  |
| --- | --- |
| Test Case Identifier | TC 041 |
| Test Items/feature | Opensearch-py client can create a Full Text query using the DSL interface and execute the query |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 2 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Initialize a Full Text query dictionary that would result in 1 of the 2 documents that would get picked. 3. Verify that the query is valid. 4. Use the client method .search(index=’index\_name’, body={query dictionary}. 5. Verify that the response of the .search() is a Python List object with 1 Python Dictionary representing the document. Verify it is the intended document. |
| Output Specifications | The client method .search() returns only the 1 document as a Python List Dictionary. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 042 |
| Test Items/feature | Opensearch-py client can create a Term Level query using the DSL interface and execute the query. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 2 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Initialize a Term Level query dictionary that would result in 1 of the 2 documents that would get picked. 3. Verify that the query is valid. 4. Use the client method .search(index=’index\_name’, body={query dictionary}). 5. Verify that the response of the .search() is a Python List object with 1 Python Dictionary representing the document. Verify it is the intended document. |
| Output Specifications | The client method .search() returns only the 1 document as a Python List Dictionary. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 043 |
| Test Items/feature | Opensearch-py client can create a Boolean query using the DSL interface and execute the query. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 2 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Initialize a Boolean query dictionary that would result in 1 of the 2 documents that would get picked. 3. Verify that the query is valid. 4. Use the client method .search(index=’index\_name’, body={query dictionary}). 5. Verify that the response of the .search() is a Python List object with 1 Python Dictionary representing the document. Verify it is the intended document. |
| Output Specifications | The client method .search() returns only the 1 document as a Python List Dictionary. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 044 |
| Test Items/feature | Opensearch-py client can create a Geo query using the DSL interface and execute the query. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 2 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Initialize a Geo query dictionary that would result in 1 of the 2 documents that would get picked. 3. Verify that the query is valid. 4. Use the client method .search(index=’index\_name’, body={query dictionary}). 5. Verify that the response of the .search() is a Python List object with 1 Python Dictionary representing the document. Verify it is the intended document. |
| Output Specifications | The client method .search() returns only the 1 document as a Python List Dictionary. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 045 |
| Test Items/feature | Opensearch-py client can execute a general search query. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 2 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Use the client method .search(index=’index\_name’). 3. Verify that the response of the .search() is a Python List object with 2 Python Dictionaries representing the documents. |
| Output Specifications | The client method .search() returns both documents as a Python List Dictionary. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 046 |
| Test Items/feature | Opensearch-py client can get no documents back when searching an empty index. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with no documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Use the client method .search(index=’index\_name’). 3. Verify that the response of the .search() is an empty Python List object. |
| Output Specifications | The client method .search() returns an empty Python List. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 047 |
| Test Items/feature | Opensearch-py client throws a missing index error when trying to search. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized. |
| Test Steps | 1. Initialize a random index name string. 2. Use the client method .search(index=’index\_name’). 3. Verify that .search() throws a missing index error. |
| Output Specifications | The client method .search() throws a missing index error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 048 |
| Test Items/feature | Opensearch-py client throws an invalid query error when trying to search. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created. |
| Test Steps | 1. Initialize a index name string using the running index’s name. 2. Use the client method .search(index=’index\_name’, query={“query”:{“matc”: {“condition”: true}}). 3. Verify that .search() throws a invalid query error. |
| Output Specifications | The client method .search() throws a invalid query error. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

* 1. **Aggregation**

|  |  |
| --- | --- |
| Test Case Identifier | TC 049 |
| Test Items/feature | Opensearch-py client can perform a Bucket Aggregation with a valid Full Text query. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 2 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Initialize an Aggregation Full Text query dictionary that would result in 1 of the 2 documents that would get picked. 3. Verify that the query is valid. 4. Use the client method .search(index=’index\_name’, body={query dictionary}. 5. Verify that the response of the .search() is a Python List object with 1 Python Dictionary representing the document. Verify is the intended document. |
| Output Specifications | The client method .search() returns only the 1 document as a Python List Dictionary. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |

|  |  |
| --- | --- |
| Test Case Identifier | TC 050 |
| Test Items/feature | Opensearch-py client can perform a Metric Aggregation with a valid Full Text query. |
| Preconditions | An OpenSearch cluster is running, a client instance is made that can connect to the cluster is initialized, and an index is created with 2 documents. |
| Test Steps | 1. Initialize an index name string with the running index’s name. 2. Initialize an Metric Aggregation Full Text query dictionary that would result in 1 of the 2 documents that would get picked. 3. Verify that the query is valid 4. Use the client method .search(index=’index\_name’, body={query dictionary}. 5. Verify that the response of the .search() is a Python List object with 1 Python Dictionary representing the document. Verify it is the intended document. |
| Output Specifications | The client method .search() returns only the 1 document as a Python List Dictionary. |
| Inter-case Dependencies | None |
| **Test Result** | **TBD** |