magnetodb

Release

August 08, 2016

Contents

CHAPTER 1

Peoject status

Project is not maintained anymore.

CHAPTER 2

Introduction

MagnetoDB is a key-value store service for OpenStack. It provides horizontally scalable, queryable storage, accessible via REST API. MagnetoDB supports Amazon DynamoDB API as well.

MagnetoDB has been designed and developed in order to provide:

- **Easy integration** REST-like API. Support Amazon DynamoDB API
- **OpenStack interoperability** Integrated with Keystone. Follows OpenStack design tenets, packaging and distribution
- **Database pluggability** Supports Cassandra,any other databases like MongoDB, HBase could be easily added
- **Horizontal scalability** MagnetoDB is scalable linearly by amount of data and requests
- **High availability** Just one working API node is enough to continue handling requests

CHAPTER 3

Developer documentation

3.1 Developer guide

3.1.1 Developer quick-start

Setting Up a Development Environment

This page describes how to setup a working Python development environment that can be used in developing MagnetoDB on Ubuntu. These instructions assume you're already familiar with git. Following these instructions will allow you to run the MagnetoDB unit tests. If you want to be able to run MagnetoDB, you will also need to install Cassandra and Devstack.

Virtual environments

The easiest way to build a fully functional development environment is with DevStack. Create a machine (such as a VM or Vagrant box) running a distribution supported by DevStack and install DevStack there. For example, there is a Vagrant script for DevStack here. You can also use this documentation.

NOTE: If you prefer not to use devstack, you can still check out source code on your local machine and develop from there.

Linux Systems

NOTE: This section is tested for MagnetoDB on Ubuntu (12.04-64) distribution. Feel free to add notes and change according to your experiences or operating system.

Install the prerequisite packages:

```
$ sudo apt-get install python-dev python-pip git-core
```

Getting the code

Grab the code from GitHub:

```
$ git clone https://git.openstack.org/stackforge/magnetodb.git
$ cd magnetodb
```

Running unit tests

The unit tests will run by default inside a virtualenv in the *.venv* directory. Run the unit tests by doing:

```
$ ./run_tests.sh
```

The first time you run them, you will be asked if you want to create a virtual environment (hit "y"):

```
No virtual environment found...create one? (Y/n)
```

See Unit Tests for more details.

Manually installing and using the virtualenv

You can manually install the virtual environment instead of having *run_tests.sh* do it for you:

```
$ python tools/install_venv.py
```

This will install all of the Python packages listed in the requirements.txt file and also those listed in the *test-requirements.txt* file into your virtualenv. There will also be some additional packages (pip, setuptools, greenlet) that are installed by the *tools/install_venv.py* file into the virutalenv.

If all goes well, you should get a message something like this:

```
MagnetoDB development environment setup is complete.
```

To activate the MagnetoDB virtualenv for the extent of your current shell session you can run:

```
$ source .venv/bin/activate
```

Or, if you prefer, you can run commands in the virtualenv on a case by case basis by running:

```
$ tools/with_venv.sh <your command>
```

Remote development

Some modern IDE such as PyCharm (commercial/open source) support remote developing. Some useful links:

Configuring Remote Interpreters via SSH
How PyCharm helps you with remote development
Configuring to work on a VM

Also, watch this video setting up dev environment for cases when MagnetoDB installed on the separate machines with Devstack:

MagnetoDB dev env configuration

Contributing Your Work

Once your work is complete you may wish to contribute it to the project. MagnetoDB uses the Gerrit code review system.

Unit Tests

MagnetoDB contains a suite of unit tests, in the */magnetodb/tests/unittests* directory.

Any proposed code change will be automatically rejected by the OpenStack Jenkins server if the change causes unit test failures.

Preferred way to run the tests

The preferred way to run the unit tests is using tox. See the unit testing section of the Testing wiki page for more information.

To run the Python 2.7 tests:

```
$ tox -e py27
```

To run the style tests:

```
$ tox -e pep8
```

You can request multiple tests, separated by commas:

```
$ tox -e py27, pep8
```

Older way to run the tests

Using tox is preferred. It is also possible to run the unit tests using the *run_tests.sh* script found at the top level of the project. The remainder of this document is focused on *run_tests.sh*.

Run the unit tests by doing:

```
$ ./run_tests.sh
```

This script is a wrapper around the testr test runner and the flake8 checker.

Flags

The *run_tests.sh* script supports several flags. You can view a list of flags by doing:

```
$ ./run_tests.sh -h
```

This will show the following help information:

```
Usage: ./run_tests.sh [OPTION]...
Run MagnetoDB's test suite(s)

 -V, --virtual-env        Use virtualenv.  Install automatically if not present.
                          (Default is to run tests in local environment)
 -F, --force              Force a clean re-build of the virtual environment. Useful when dependencies
 -f, --func               Functional tests have been removed.
 -u, --unit               Run unit tests (default when nothing specified)
 -p, --pep8               Run pep8 tests
 --all                    Run pep8 and unit tests
 -c, --coverage           Generate coverage report
 -d, --debug              Run tests with testtools instead of testr. This allows you to use the debug
 -h, --help               Print this usage message
```

Because *run_tests.sh* is a wrapper around testrepository, it also accepts the same flags as testr. See the testr user manual for details about these additional flags.

Running a subset of tests

Instead of running all tests, you can specify an individual directory, file, class, or method that contains test code.

To run the tests in the */magnetodb/tests/unittests/api/openstack/v1* directory:

```
$ ./run_tests.sh v1
```

To run the tests in the */magnetodb/tests/unittests/api/openstack/v1/test_get_item.py* file:

```
$ ./run_tests.sh test_get_item
```

To run the tests in the GetItemTestCase class in */magnetodb/tests/unittests/api/openstack/v1/test_get_item.py*:

```
$ ./run_tests.sh test_get_item.GetItemTestCase
```

To run the GetItemTestCase.test_get_item test method in */magnetodb/tests/unittests/api/openstack/v1/test_get_item.py*:

```
$ ./run_tests.sh test_get_item.GetItemTestCase.test_get_item
```

Also note, that as all these tests (using *tox* or *run_tests.sh*) are run by *testr* test runner, it is not possible to use *pdb* breakpoints in tests or the code being tested. To be able to use debugger breakpoints you should directly use *testtools* as in the following:

```
$ python -m testtools.run magnetodb.tests.unittests.test_get_item.GetItemTestCase.test_get_item
```

Virtualenv

By default, the tests use the Python packages installed inside a virtualenv. (This is equivalent to using the *-V, –virtualenv* flag).

If you wish to recreate the virtualenv, call *run_tests.sh* with the flag:

```
-f, --force
```

Recreating the virtualenv is useful if the package dependencies have changed since the virtualenv was last created. If the *requirements.txt* or *tools/install_venv.py* files have changed, it's a good idea to recreate the virtualenv.

Integration and functional tests

MagnetoDB contains a suite of integration tests (in the */magnetodb/tests/storage* directory) and functional tests (in the */contrib/tempest* directory).

Any proposed code change will be automatically rejected by the OpenStack Jenkins server if the change causes unit test failures.

Refer to Tests on environment with devstack for information, how to install and set environment and how to run such kind of tests.

3.1.2 Source documentation

magnetodb.common

magnetodb.common.cassandra

magnetodb.common.middleware

magnetodb.storage

magnetodb.storage.driver

magnetodb.storage.manager

magnetodb.storage.table_info_repo

3.2 API Reference

3.2.1 RESTful Web API (v1)

MagnetoDB API is a RESTful API what uses JSON media type for interaction between client and server

Authentication

Headers

Each request is expected to have following headers:

- User-Agent
- Content-Type: application/json
- Accept: application/json
- X-Auth-Token keystone auth token

Common Errors

This section lists the common errors that all actions return. Any action-specific errors will be listed in the topic for the action.

Operation details

- table_name parameter will be provided via URL
- API will use different HTTP methods for different operations (POST for create, PUT for update, etc)

Note: operations with items in the table(GetItem, PutItem, Scan, etc) will use POST method.

MagnetoDB actions

CreateTable

POST v1/data/{project_id}/tables

Request Syntax

```
{
    "attribute_definitions": [
        {
            "attribute_name": "string",
            "attribute_type": "string"
        }
    ],
    "key_schema": [
        {
            "attribute_name": "string",
            "key_type": "string"
        }
    ],
    "local_secondary_indexes": [
        {
            "index_name": "string",
            "key_schema": [
                {
                    "attribute_name": "string",
                    "key_type": "string"
                }
            ],
            "projection": {
                "non_key_attributes": [
                    "string"
                ],
                "projection_type": "string"
            }
        }
    ],
    "table_name": "string"
}
```

Request Parameters

table_name

The name of the table. Unique per project.
Type: string
Required: Yes

attribute_definitions

An array of attributes that describe the key schema for the table and indexes.
Type: array of AttributeDefinition objects
Required: Yes

key_schema

Specifies the attributes that make up the primary key for a table or an index.

Type: array of key_schemaElement objects
Required: Yes

local_secondary_indexes

One or more local secondary indexes to be created on the table.
Type: array of objects
Required: No

Response Syntax

```
{
    "table_description": {
        "attribute_definitions": [
            {
                "attribute_name": "string",
                "attribute_type": "string"
            }
        ],
        "creation_datetime": "number",
        "item_count": "number",
        "key_schema": [
            {
                "attribute_name": "string",
                "key_type": "string"
            }
        ],
        "local_secondary_indexes": [
            {
                "index_name": "string",
                "index_size_bytes": "number",
                "item_count": "number",
                "key_schema": [
                    {
                        "attribute_name": "string",
                        "key_type": "string"
                    }
                ],
                "projection": {
                    "non_key_attributes": [
                        "string"
                    ],
                    "projection_type": "string"
                }
            }
        ],
        "links": [
            {
                "href": "url",
                "rel": "self"
            },
            {
                "href": "url",
                "rel": "bookmark"
            }
        ],
        "table_size_bytes": "number",
        "table_status": "string"
```

```
    }
}
```

Response Elements

table_description

Represents the properties of a table.
Type: table_description object

Errors

BackendInteractionException
ClusterIsNotConnectedException
TableAlreadyExistsException
ValidationError

Sample Request

```
{
    "attribute_definitions": [
        {
            "attribute_name": "ForumName",
            "attribute_type": "S"
        },
        {
            "attribute_name": "Subject",
            "attribute_type": "S"
        },
        {
            "attribute_name": "LastPostDateTime",
            "attribute_type": "S"
        }
    ],
    "table_name": "Thread",
    "key_schema": [
        {
            "attribute_name": "ForumName",
            "key_type": "HASH"
        },
        {
            "attribute_name": "Subject",
            "key_type": "RANGE"
        }
    ],
    "local_secondary_indexes": [
        {
            "index_name": "LastPostIndex",
            "key_schema": [
                {
                    "attribute_name": "ForumName",
                    "key_type": "HASH"
                },
                {
                    "attribute_name": "LastPostDateTime",
                    "key_type": "RANGE"
```

```
                }
            ],
            "projection": {
                "projection_type": "KEYS_ONLY"
            }
        }
    ]
}
```

Sample Response

```
{
    "table_description": {
        "attribute_definitions": [
            {
                "attribute_name": "Subject",
                "attribute_type": "S"
            },
            {
                "attribute_name": "LastPostDateTime",
                "attribute_type": "S"
            },
            {
                "attribute_name": "ForumName",
                "attribute_type": "S"
            }
        ],
        "creation_date_time": 0,
        "item_count": 0,
        "key_schema": [
            {
                "attribute_name": "ForumName",
                "key_type": "HASH"
            },
            {
                "attribute_name": "Subject",
                "key_type": "RANGE"
            }
        ],
        "local_secondary_indexes": [
            {
                "index_name": "LastPostIndex",
                "index_size_bytes": 0,
                "item_count": 0,
                "key_schema": [
                    {
                        "attribute_name": "ForumName",
                        "key_type": "HASH"
                    },
                    {
                        "attribute_name": "LastPostDateTime",
                        "key_type": "RANGE"
                    }
                ],
                "projection": {
                    "projection_type": "ALL"
                }
            }
```

```
        ],
        "table_name": "Thread",
        "table_size_bytes": 0,
        "table_status": "ACTIVE",
        "links": [
            {
                "href": "http://localhost:8480/v1/fake_project_id/data/tables/Thread",
                "rel": "self"
            },
            {
                "href": "http://localhost:8480/v1/fake_project_id/data/tables/Thread",
                "rel": "bookmark"
            }
        ]
    }
}
```

UpdateTable (*)

This action is defined and reserved for future implementation.

PUT v1/data/{project_id}/tables/{table_name}

DescribeTable

GET v1/data/{project_id}/tables/{table_name}

Request Syntax

This operation does not require a request body

Response Syntax

```
{
    "table": {
        "attribute_definitions": [
            {
                "attribute_name": "string",
                "attribute_type": "string"
            }
        ],
        "creation_datetime": "number",
        "item_count": "number",
        "key_schema": [
            {
                "attribute_name": "string",
                "key_type": "string"
            }
        ],
        "local_secondary_indexes": [
            {
                "index_name": "string",
                "index_size_bytes": "number",
                "item_count": "number",
                "key_schema": [
```

```
                    {
                        "attribute_name": "string",
                        "key_type": "string"
                    }
                ],
                "projection": {
                    "non_key_attributes": [
                        "string"
                    ],
                    "projection_type": "string"
                }
            }
        ],
        "links": [
            {
                "href": "url",
                "rel": "self"
            }
        ],
        "table_name": "string",
        "table_size_bytes": "number",
        "table_status": "string"
    }
}
```

Response Elements

table

Represents the properties of a table.
Type: table_description object

Table Statuses

- ACTIVE
- CREATING
- CREATE_FAILURE
- DELETING
- DELETE_FAILURE

Errors

BackendInteractionException
ClusterIsNotConnectedException
TableNotExistsException
ValidationError

Sample Response

```
{
    "table": {
        "attribute_definitions": [
            {
```

```
                "attribute_name": "Subject",
                "attribute_type": "S"
            },
            {
                "attribute_name": "LastPostDateTime",
                "attribute_type": "S"
            },
            {
                "attribute_name": "ForumName",
                "attribute_type": "S"
            }
        ],
        "creation_date_time": 0,
        "item_count": 0,
        "key_schema": [
            {
                "attribute_name": "ForumName",
                "key_type": "HASH"
            },
            {
                "attribute_name": "Subject",
                "key_type": "RANGE"
            }
        ],
        "local_secondary_indexes": [
            {
                "index_name": "LastPostIndex",
                "index_size_bytes": 0,
                "item_count": 0,
                "key_schema": [
                    {
                        "attribute_name": "ForumName",
                        "key_type": "HASH"
                    },
                    {
                        "attribute_name": "LastPostDateTime",
                        "key_type": "RANGE"
                    }
                ],
                "projection": {
                    "projection_type": "ALL"
                }
            }
        ],
        "table_name": "Thread",
        "table_size_bytes": 0,
        "table_status": "ACTIVE",
        "links": [
            {
                "href": "http://localhost:8480/v1/fake_project_id/data/tables/Thread",
                "rel": "self"
            },
            {
                "href": "http://localhost:8480/v1/fake_project_id/data/tables/Thread",
                "rel": "bookmark"
            }
        ]
    }
```

```
}
```

DeleteTable

DELETE v1/data/{project_id}/tables/{table_name}

Request Syntax

This operation does not require a request body

Response Syntax

```
{
    "table_description": {
        "attribute_definitions": [
            {
                "attribute_name": "string",
                "attribute_type": "string"
            }
        ],
        "creation_datetime": "number",
        "item_count": "number",
        "key_schema": [
            {
                "attribute_name": "string",
                "key_type": "string"
            }
        ],
        "local_secondary_indexes": [
            {
                "index_name": "string",
                "index_size_bytes": "number",
                "item_count": "number",
                "key_schema": [
                    {
                        "attribute_name": "string",
                        "key_type": "string"
                    }
                ],
                "projection": {
                    "non_key_attributes": [
                        "string"
                    ],
                    "projection_type": "string"
                }
            }
        ],
        "links": [
            {
                "href": "url",
                "rel": "self"
            }
        ],
        "table_size_bytes": "number",
        "table_status": "string"
```

```
    }
}
```

Response Elements

table_description

Represents the properties of a table.
Type: table_description object

Errors

BackendInteractionException
ClusterIsNotConnectedException
ResourceInUseException
TableNotExistsException
ValidationError

Sample Response

```
{
    "table_description": {
        "attribute_definitions": [
            {
                "attribute_name": "Subject",
                "attribute_type": "S"
            },
            {
                "attribute_name": "LastPostDateTime",
                "attribute_type": "S"
            },
            {
                "attribute_name": "ForumName",
                "attribute_type": "S"
            }
        ],
        "creation_date_time": 0,
        "item_count": 0,
        "key_schema": [
            {
                "attribute_name": "ForumName",
                "key_type": "HASH"
            },
            {
                "attribute_name": "Subject",
                "key_type": "RANGE"
            }
        ],
        "local_secondary_indexes": [
            {
                "index_name": "LastPostIndex",
                "index_size_bytes": 0,
                "item_count": 0,
                "key_schema": [
                    {
                        "attribute_name": "ForumName",
```

```
                        "key_type": "HASH"
                    },
                    {
                        "attribute_name": "LastPostDateTime",
                        "key_type": "RANGE"
                    }
                ],
                "projection": {
                    "projection_type": "ALL"
                }
            }
        ],
        "table_name": "Thread",
        "table_size_bytes": 0,
        "table_status": "DELETING",
        "links": [
            {
                "href": "http://localhost:8480/v1/fake_project_id/data/tables/Thread",
                "rel": "self"
            },
            {
                "href": "http://localhost:8480/v1/fake_project_id/data/tables/Thread",
                "rel": "bookmark"
            }
        ]
    }
}
```


ListTables

GET v1/data/{project_id}/tables

Request Syntax

This operation does not require a request body

Request Parameters

Parameters should be provided via GET query string.

exclusive_start_table_name

The first table name that this operation will evaluate. Use the value that was returned for last_evaluated_table_name in the previous operation.

Type: xsd:string

Required: No

limit

A maximum number of the items to return.

Type: xsd:int

Required: No

Response Syntax

```
{
    "last_evaluated_table_name": "string",
    "tables": [
        {
            "rel": "string",
            "href": "url"
        }
    ]
}
```

Response Elements

last_evaluated_table_name

The name of the last table in the current page of results.
Type: String

tables

Array of the table info items
Type: array of structs

Errors

BackendInteractionException
ClusterIsNotConnectedException

Sample Response

```
{
    "tables": [
        {
            "href": "http://localhost:8480/v1/data/fake_project_id/tables/Thread",
            "rel": "self"
        },
        {
            "href": "http://localhost:8480/v1/data/fake_project_id/tables/test_hash",
            "rel": "self"
        }
    ]
}
```

PutItem

POST v1/data/{project_id}/tables/{table_name}/put_item

Description Adds or rewrites existing item in table. If item already exists and it is just replaced if no other option is specified. It also supports conditions what are applied before inserting/updating item. If condition passes the item is updated or inserted.

Conditions support You can define list of conditions what should be checked before applying insert or update. You can evaluate if specific item attribute value is equal to given in request or check its existing. Please look at sample below where ForumName is a hash key and Subject is a range key.

```
{
    "item": {
        "ForumName": {
            "S": "MagnetoDB"
        },
        "Subject": {
            "S": "How do I update multiple items?"
        },
        "Message": {
            "S": "Message text"
        },
        "Read": {
            "S": "true"
        }
    },
    "expected": {
        "Message": {
            "value": {"S": "Message text"}
        },
        "Read": {
            "exists": true
        }
    }
}
```

Also you can check if item with given primary key exists in table at all. To do it, you have to build condition if hash key exists. Technically the items is queried by hash and range key (if range key is defined) and after that condition is applied, so adding range key to condition is useless and ValidationError will be rose.

Request Syntax

```
{
    "expected": {
        "string": {
            "exists": "boolean",
            "value": {
                "B": "blob",
                "BS": [
                    "blob"
                ],
                "N": "string",
                "NS": [
                    "string"
                ],
                "S": "string",
                "SS": [
                    "string"
                ]
            }
        }
    },
    "item": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
```

```
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
    "time_to_live": "number",
    "return_values": "string"
}
```

Request Parameters

item

A map of attribute name/value pairs, one for each attribute. Only the primary key attributes are required.

Type: String to Attributevalue object map

Required: Yes

expected

The conditional block for the PutItem operation.

Type: String to expected Attributevalue object map

Required: No

time_to_live

Defines time to live for item

Type: number

Valid values: 0 - MAX_NUMBER

Required: No

return_values

Use return_values if you want to get the item attributes as they appeared before they were updated.

Type: String

Valid values: NONE | ALL_OLD

Required: No

Response Syntax

```
{
    "attributes": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
```

```
                "string"
            ]
        }
    }
}
```

Response Elements

attributes

The attribute values as they appeared before the PutiItem operation.
Type: String to attribute struct

Errors

BackendInteractionException
ClusterIsNotConnectedException
ConditionalCheckFailedException
ValidationError

Samples Sample Request

```
{
    "item": {
        "LastPostDateTime": {
            "S": "201303190422"
        },
        "Tags": {
            "SS": ["Update", "Multiple items", "HelpMe"]
        },
        "ForumName": {
            "S": "MagnetoDB"
        },
        "Message": {
            "S": "I want to update multiple items."
        },
        "Subject": {
            "S": "How do I update multiple items?"
        },
        "LastPostedBy": {
            "S": "fred@example.com"
        }
    },
    "expected": {
        "ForumName": {
            "exists": false
        },
        "Subject": {
            "exists": false
        }
    },
    "return_values": "ALL_OLD"
}
```

Sample Response

```
{
    "attributes": {
        "ForumName": {
            "S": "MagnetoDB"
        },
        "LastPostDateTime": {
            "S": "201303190422"
        },
        "LastPostedBy": {
            "S": "fred@example.com"
        },
        "Message": {
            "S": "I want to update multiple items."
        },
        "Subject": {
            "S": "How do I update multiple items?"
        },
        "Tags": {
            "SS": ["HelpMe", "Multiple items", "Update"]
        }
    }
}
```

GetItem

POST v1/data/{project_id}/tables/{table_name}/get_item

Request Syntax

```
{
    "attributes_to_get": [
        "string"
    ],
    "consistent_read": "boolean",
    "key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    }
}
```

Request Parameters

key

The primary key of the item to retrieve.
Type: String to object map
Required: Yes

attributes_to_get

The names of one or more attributes to retrieve.
Type: array of Strings
Required: No

consistent_read

Use or not use strongly consistent read.
Type: Boolean
Required: No

Response Syntax

```
{
    "item": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    }
}
```

Response Elements

item

An itemi with attributes.
Type: String to object map

Errors

BackendInteractionException
ClusterIsNotConnectedException
ValidationError

Sample Request

```
{
    "key": {
        "ForumName": {
            "S": "MagnetoDB"
        },
```

```
        "Subject": {
            "S": "How do I update multiple items?"
        }
    },
    "attributes_to_get": ["LastPostDateTime", "Message", "Tags"],
    "consistent_read": true
}
```

Sample Response

```
{
    "Item": {
        "Tags": {
            "SS": ["Update", "Multiple Items", "HelpMe"]
        },
        "LastPostDateTime": {
            "S": "201303190436"
        },
        "Message": {
            "S": "I want to update multiple items in a single API call. What's the best way to do tha
        }
    }
}
```

UpdateItem

POST v1/data/{project_id}/tables/{table_name}/update_item

Request Syntax

```
{
    "attribute_updates": {
        "string": {
            "action": "string",
            "value": {
                "B": "blob",
                "BS": [
                    "blob"
                ],
                "N": "string",
                "NS": [
                    "string"
                ],
                "S": "string",
                "SS": [
                    "string"
                ]
            }
        }
    },
    "time_to_live": "number",
    "expected": {
        "string": {
            "exists": "boolean",
            "value": {
                "B": "blob",
```

```
                "BS": [
                    "blob"
                ],
                "N": "string",
                "NS": [
                    "string"
                ],
                "S": "string",
                "SS": [
                    "string"
                ]
            }
        }
    },
    "key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
    "return_values": "string"
}
```

Request Parameters

key

The primary key of the item to retrieve.

Type: String to object map

Required: Yes

attribute_updates

The names of attributes to be modified, the action to perform on each, and the new value for each. If you are updating an attribute that is an index key attribute for any indexes on that table, the attribute type must match the index key type defined in the attribute_definition of the table description. You can use UpdateItem to update any non-key attributes.

Attribute values cannot be null. String and binary type attributes must have lengths greater than zero. Set type attributes must not be empty. Requests with empty values will be rejected with a ValidationError exception.

Each attribute_updates element consists of an attribute name to modify, along with the following:

- value - the new value, if applicable, for this attribute;
- action - specifies how to perform the update. Valid values for action are PUT, DELETE, and ADD. The behavior depends on whether the specified primary key already exists in the table.

  If an item with the specified key is found in the table:

  - PUT - Adds the specified attribute to the item. If the attribute already exists, it is replaced by the new value.
  - DELETE - If no value is specified, the attribute and its value are removed from the item. The data type of the specified value must match the existing value's data type. If a set of values is specified, then those values are subtracted from the old set. For example, if the attribute value was the set [a,b,c] and the DELETE action specified [a,c], then the final attribute value would be [b]. Specifying an empty set is an error.
  - ADD - If the attribute does not already exist, then the attribute and its values are added to the item. If the attribute does exist, then the behavior of ADD depends on the data type of the attribute:
    - if the existing attribute is a number, and if value is also a number, then the value is mathematically added to the existing attribute. If value is a negative number, then it is subtracted from the existing attribute;
    - if the existing data type is a set, and if the value is also a set, then the value is added to the existing set. (This is a set operation, not mathematical addition.) For example, if the attribute value was the set [1,2], and the ADD action specified [3], then the final attribute value would be [1,2,3]. An error occurs if an Add action is specified for a set attribute and the attribute type specified does not match the existing set type.

    Both sets must have the same primitive data type. For example, if the existing data type is a set of strings, the value must also be a set of strings. The same holds true for number sets and binary sets.

  This action is only valid for an existing attribute whose data type is number or is a set. Do not use ADD for any other data types.

  If no item with the specified key is found:

  - PUT - MagnetoDB creates a new item with the specified primary key, and then adds the attribute.
  - DELETE - Nothing happens; there is no attribute to delete.
  - ADD - MagnetoDB creates an item with the supplied primary key and number (or set of numbers) for the attribute value. The only data types allowed are number and number set; no other data types can be specified.

If you specify any attributes that are part of an index key, then the data types for those attributes must match those of the schema in the table's attribute definition.

Type: String to object map

Required: No

time_to_live

Defines time to live for item
Type: number
Valid values: 0 - MAX_NUMBER
Required: No

expected

The conditional block for the Updateitem operation. All the conditions must be met for the operation to succeed.
Type: String to object map
Required: No

return_values

Type: String
Valid values: NONE | ALL_OLD | UPDATED_OLD | ALL_NEW | UPDATED_NEW
Required: No

Response Syntax

```
{
    "attributes": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    }
}
```

Response Elements

attributes

Item attributes
Type: String to object map

Errors

BackendInteractionException
ClusterIsNotConnectedException
ConditionalCheckFailedException

TableNotExistsException
ValidationError

Sample Request

```
{
    "key": {
        "ForumName": {
            "S": "MagnetoDB"
        },
        "Subject": {
            "S": "How do I delete an item?"
        }
    },
    "attribute_updates": {
        "LastPostedBy": {
            "value": {
                "S": "me@test.com"
            },
            "action": "PUT"
        }
    },
    "expected": {
        "LastPostedBy": {
            "value": {
                "S": "fred@example.com"
            }
        },
         "Replies": {
            "exists": false
        }
   },
    "return_values": "ALL_NEW"
}
```

Sample Response

```
{
    "attributes": {
        "LastPostedBy": {
            "S": "me@test.com"
        }
    }
}
```

Atomic Counters with Examples

Atomic counters garentee the updateitem operation to increment or decrement the value of an existing attribute will not interfer with other write requests.

If action is “ADD” then MagnetoDB guarantees that operation will be performed atomically. Sets and numbers are only valid types for ADD action.

Only if attribute type is number then this attribute can be used as atomic counter.

1. Create table for forum threads. We have 4 attributes: name, subject, tags and views count (we want to use it like atomic counter)

```
{
    "table_name": "Thread",
    "attribute_definitions": [
        {
            "attribute_name": "ForumName",
            "attribute_type": "S"
        },
        {
            "attribute_name": "Subject",
            "attribute_type": "S"
        },
        {
            "attribute_name": "Tags",
            "attribute_type": "SS"
        },
        {
            "attribute_name": "ViewsCount",
            "attribute_type": "N"
        }
    ],
    "key_schema": [
        {
            "attribute_name": "ForumName",
            "key_type": "HASH"
        },
        {
            "attribute_name": "Subject",
            "key_type": "RANGE"
        }
    ]
}
```

2. Put new item into Thread table.

```
{
    "item": {
      "ForumName": {"S": "MagnetoDB"},
      "Subject": {"S": "How do I delete an item?"},
      "LastPostedBy": {"S": "user1@test.com"},
      "Tags": {"SS": ["Update", "Multiple Items"]},
      "ViewsCount": {"N": "0"}
    }
}
```

If we try to query an item now we'll see that ViewsCount = 0 and Tags = ["Update", "Multiple Items"].

3. Use update_item to update Tags

```
{
    "key": {
        "ForumName": {
            "S": "MagnetoDB"
        },
        "Subject": {
            "S": "How do I delete an item?"
        }
    },
    "attribute_updates": {
        "Tags": {
            "action": "ADD",
```

```
            "value": {
              "SS": ["HelpMe"]
            }
        }
    }
}
```

If we try to query an item now we'll see that ViewsCount = 0 and Tags = ["Update", "Multiple Items", "HelpMe"]

With Atomic counters different users looking the thread and we need to update ViewsCount

Any time when different users try to query our item, they shell see same value of ViewsCount. After our request ViewsCount should be 1.

Different users can update this value simultaneously.

```
{
    "key": {
        "ForumName": {
            "S": "MagnetoDB"
        },
        "Subject": {
            "S": "How do I delete an item?"
        }
    },
    "attribute_updates": {
        "ViewsCount": {
            "action": "ADD",
          "value": {
            "N": "1"
          }
        }
    }
}
```

DeleteItem

POST v1/data/{project_id}/tables/{table_name}/delete_item

Request Syntax

```
{
    "key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
```

```
    "expected": {
        "string": {
            "exists": "boolean",
            "value": {
                "B": "blob",
                "BS": [
                    "blob"
                ],
                "N": "string",
                "NS": [
                    "string"
                ],
                "S": "string",
                "SS": [
                    "string"
                ]
            }
        }
    },
    "return_values": "string"
}
```

Request Parameters

key

Primary key of the item to delete.
Type: String to object map
Required: Yes

expected

The conditional block for the DeleteItem operation. All the conditions must be met for the operation to succeed.
Type: String to object map
Required: No

return_values

Type: String
Valid values: NONE | ALL_OLD | UPDATED_OLD | ALL_NEW | UPDATED_NEW
Required: No

Response Syntax

```
{
    "attributes": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
```

```
            "SS": [
                "string"
            ]
        }
    }
}
```

Response Elements

attributes

Item attributes
Type: String to Attributevalue object map

Errors

BackendInteractionException
ClusterIsNotConnectedException
ConditionalCheckFailedException
ValidationError

Sample Request

```
{
    "key": {
        "ForumName": {
            "S": "MagnetoDB"
        },
        "Subject": {
            "S": "How do I delete an item?"
        }
    },
    "expected": {
        "Subject": {
            "value": {
                "S": "How do I delete an item?"
            }
        },
        "Replies": {
            "exists": false
        }
    },
    "returnValues": "ALL_OLD"
}
```

Sample Response

```
{
    "attributes": {
        "LastPostDateTime": {
            "S": "201303190422"
        },
        "ForumName": {
            "S": "MagnetoDB"
        },
        "Message": {
```

```
            "S": "I want to delete item."
        },
        "Subject": {
            "S": "How do I delete an item?"
        },
        "LastPostedBy": {
            "S": "fred@example.com"
        }
    }
}
```

Query

POST v1/data/{project_id}/tables/{table_name}/query

Request Syntax

```
{
    "attributes_to_get": [
        "string"
    ],
    "consistent_read": "boolean",
    "exclusive_start_key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
    "index_name": "string",
    "key_conditions": {
        "string": {
            "attribute_value_list": [
                {
                    "B": "blob",
                    "BS": [
                        "blob"
                    ],
                    "N": "string",
                    "NS": [
                        "string"
                    ],
                    "S": "string",
                    "SS": [
                        "string"
                    ]
                }
```

```
            ],
            "comparison_operator": "string"
        }
    },
    "limit": "number",
    "scan_index_forward": "boolean",
    "select": "string"
}
```

Request Parameters

attributes_to_get

Type: array of Strings
Required: No

consistent_read

Type: Boolean
Required: No

exclusive_start_key

The primary key of the first item that this operation will evaluate.
Type: String to object map
Required: No

index_name

The name of an index to query.
Type: String
Required: No

key_conditions

The selection criteria for the query.
Type: String to Condition object map
Required: Yes

limit

Type: Number
Required: No

scan_index_forward

Type: Boolean
Required: No

select

The attributes to be returned in the result.
Type: String
Valid values: ALL_ATTRIBUTES | ALL_PROJECTED_ATTRIBUTES | SPECIFIC_ATTRIBUTES | COUNT
Required: No

Response Syntax

```
{
    "count": "number",
    "items": [

        {
            "string": {
                "B": "blob",
                "BS": [
                    "blob"
                ],
                "N": "string",
                "NS": [
                    "string"
                ],
                "S": "string",
                "SS": [
                    "string"
                ]
            }
        }
    ],
    "last_evaluated_key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    }
}
```

Response Elements

count

The number of items in the response.
Type: Number

items

An array of items.
Type: array of items

last_evaluated_key

The primary key of the item where the operation stopped.
Type: String to AttributeValue object map

Errors

BackendInteractionException
ClusterIsNotConnectedException
ValidationError

Sample Request

```
{
    "attributes_to_get": [
        "ForumName", "LastPostDateTime", "Posts"
    ],
    "exclusive_start_key": {
        "ForumName": {
            "S": "Testing OS API"
        },
        "LastPostDayTime": {
            "S": "3/1/14"
        }
    },
    "index_name": "LastPostIndex",
    "limit": 2,
    "consistent_read": true,
    "scan_index_forward": true,
    "key_conditions": {
        "ForumName": {
            "attribute_value_list": [
                {
                    "S": "Testing OS API"
                }
            ],
            "comparison_operator": "EQ"
        },
        "LastPostDateTime": {
            "attribute_value_list": [
                {
                    "S": "3/10/14"
                }
            ],
            "comparison_operator": "GT"
        }
    },
    "select": "SPECIFIC_ATTRIBUTES"
}
```

Sample Response

```
{
    "count": 2,
    "items": [
        {
            "ForumName": {
                "S": "Testing OS API"
            },
            "LastPostDateTime": {
                "S": "3/18/14"
            },
            "Posts": {
                "SS": ["Opening post"]
            }
```

```
        },
        {
            "ForumName": {
                "S": "Testing OS API"
            },
            "LastPostDateTime": {
                "S": "3/19/14"
            },
            "Posts": {
                "SS": ["Hello", "Hi"]
            }
        }
    ],
    "last_evaluated_key": {
        "ForumName": {
            "S": "Testing OS API"
        },
        "LastPostDateTime": {
            "S": "3/19/14"
        }
    }
}
```

Scan

POST v1/data/{project_id}/tables/{table_name}/scan

Request Syntax

```
{
    "attributes_to_get": [
        "string"
    ],
    "exclusive_start_key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
    "limit": "number",
    "scan_filter": {
        "string": {
            "attribute_value_list": [
                {
                    "B": "blob",
                    "BS": [
```

```
                        "blob"
                    ],
                    "N": "string",
                    "NS": [
                        "string"
                    ],
                    "S": "string",
                    "SS": [
                        "string"
                    ]
                }
            ],
            "comparison_operator": "string"
        }
    },
    "segment": "number",
    "select": "string",
    "total_segments": "number"
}
```

Request Parameters

attributes_to_get

Type: array of Strings
Required: No

exclusive_start_key

The primary key of the first item that this operation will evaluate.
Type: String to object map
Required: No

limit

Type: Number
Required: No

scan_filter

Scan conditions.
Type: String to Condition object map
Required: No

segment

Segment for parallel scan.
Type: Number
Required: No

select

The attributes to be returned in the result.
Type: String
Valid values: ALL_ATTRIBUTES | ALL_PROJECTED_ATTRIBUTES | SPECIFIC_ATTRIBUTES | COUNT
Required: No

total_segments

Number of segments for parallel scan.
Type: Number
Required: No

Response Syntax

```
{
    "count": "number",
    "items": [
        {
            "string": {
                "B": "blob",
                "BS": [
                    "blob"
                ],
                "N": "string",
                "NS": [
                    "string"
                ],
                "S": "string",
                "SS": [
                    "string"
                ]
            }
        }
    ],
    "last_evaluated_key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
    "scanned_count": "number"
}
```

Response Elements

count

The number of items in the response.
Type: Number

items

An array of items.
Type: array of items

last_evaluated_key

The primary key of the item where the operation stopped.
Type: String to AttributeValue object map

scanned_count

Type: Number

Errors

BackendInteractionException
ClusterIsNotConnectedException
ValidationError

Sample Request

```
{
    "attributes_to_get": [
        "ForumName", "LastPostDateTime", "Posts"
    ],
    "exclusive_start_key": {
        "ForumName": {
            "S": "Another forum"
        }
    },
    "limit": 2,
    "scan_filter": {
        "LastPostDateTime": {
            "attribute_value_list": [
                {
                    "S": "3/10/14"
                }
            ],
            "comparison_operator": "GT"
        }
    },
    "segment": 0,
    "select": "SPECIFIC_ATTRIBUTES",
    "total_segments": 1
}
```

Sample Response

```
{
    "count": 2,
    "items": [
        {
            "ForumName": {
                "S": "Gerrit workflow"
            },
            "LastPostDateTime": {
                "S": "3/19/14"
            },
            "Posts": {
                "SS": ["Hello", "Hi"]
            }
        },
        {
            "ForumName": {
```

```
                "S": "Testing OS API"
            },
            "LastPostDateTime": {
                "S": "3/18/14"
            },
            "Posts": {
                "SS": ["Opening post"]
            }
        }
    ],
    "last_evaluated_key": {
        "ForumName": {
            "S": "Testing OS API"
        },
        "Subject": {
            "S": "Some subject"
        }
    },
    "scanned_count": 10
}
```

BatchGetItem

POST v1/data/{project_id}/batch_get_item

Request Syntax

```
{
    "request_items": {
        "string": {
            "attributes_to_get": [
                "string"
            ],
            "consistent_read": "boolean",
            "keys": [

                {
                    "string": {
                        "B": "blob",
                        "BS": [
                            "blob"
                        ],
                        "N": "string",
                        "NS": [
                            "string"
                        ],
                        "S": "string",
                        "SS": [
                            "string"
                        ]
                    }
                }
            ]
        }
    }
}
```

Request Parameters

request_items

Type: String to object map
Required: Yes

Response Syntax

```
{
    "responses": {
        "string": [

            {
                "string": {
                    "B": "blob",
                    "BS": [
                        "blob"
                    ],
                    "N": "string",
                    "NS": [
                        "string"
                    ],
                    "S": "string",
                    "SS": [
                        "string"
                    ]
                }
            }
        ]
    },
    "unprocessed_keys": {
        "string": {
            "attributes_to_get": [
                "string"
            ],
            "consistent_read": "boolean",
            "keys": [

                {
                    "string": {
                        "B": "blob",
                        "BS": [
                            "blob"
                        ],
                        "N": "string",
                        "NS": [
                            "string"
                        ],
                        "S": "string",
                        "SS": [
                            "string"
                        ]
                    }
                }
            ]
        }
    }
}
```

Response Elements

responses

Type: String to map

unprocessed_keys

Type: String to object map

Errors

BackendInteractionException
ClusterIsNotConnectedException
ValidationError

Sample Request

```
{
    "request_items": {
        "Forum": {
            "keys": [
                {
                    "Name": {
                        "S": "MagnetoDB"
                    },
                    "Category": {
                        "S": "OpenStack KVaaS"
                    }
                },
                {
                    "Name": {
                        "S": "Nova"
                    },
                    "Category": {
                        "S": "OpenStack Core"
                    }
                }
            ]
        },
        "Thread": {
            "keys": [
                {
                    "Name": {
                        "S": "MagnetoDB"
                    },
                    "Category": {
                        "S": "OpenStack KVaaS"
                    }
                },
                {
                    "Name": {
                        "S": "Nova"
                    },
                    "Category": {
                        "S": "OpenStack Core"
                    }
                }
            ]
```

```
        }
    }
}
```

Sample Response

```
{
    "responses": {
        "Forum": {
            "keys": [
                {
                    "Name": {
                        "S": "MagnetoDB"
                    },
                    "Category": {
                        "S": "OpenStack KVaaS"
                    }
                },
                {
                    "Name": {
                        "S": "Nova"
                    },
                    "Category": {
                        "S": "OpenStack Core"
                    }
                }
            ]
        }
    },
    "unprocessed_keys": {
        "Thread": {
            "keys": [
                {
                    "Name": {
                        "S": "MagnetoDB"
                    },
                    "Category": {
                        "S": "OpenStack KVaaS"
                    }
                },
                {
                    "Name": {
                        "S": "Nova"
                    },
                    "Category": {
                        "S": "OpenStack Core"
                    }
                }
            ]
        }
    }
}
```

BatchWriteItem

POST v1/data/{project_id}/batch_write_item

Request Syntax

```
{
    "request_items": {
        "string": [
            {
                "delete_request": {
                    "key": {
                        "string": {
                            "B": "blob",
                            "BS": [
                                "blob"
                            ],
                            "N": "string",
                            "NS": [
                                "string"
                            ],
                            "S": "string",
                            "SS": [
                                "string"
                            ]
                        }
                    }
                },
                "put_request": {
                    "item": {
                        "string": {
                            "B": "blob",
                            "BS": [
                                "blob"
                            ],
                            "N": "string",
                            "NS": [
                                "string"
                            ],
                            "S": "string",
                            "SS": [
                                "string"
                            ]
                        }
                    },
                    "time_to_live": "number"
                }
            }
        ]
    }
}
```

Request Parameters

request_items

Type: String to object map
Required: Yes

Response Syntax

```
{
    "unprocessed_items": {
        "string": [
            {
                "delete_request": {
                    "key": {
                        "string": {
                            "B": "blob",
                            "BS": [
                                "blob"
                            ],
                            "N": "string",
                            "NS": [
                                "string"
                            ],
                            "S": "string",
                            "SS": [
                                "string"
                            ]
                        }
                    }
                },
                "put_request": {
                    "item": {
                        "string": {
                            "B": "blob",
                            "BS": [
                                "blob"
                            ],
                            "N": "string",
                            "NS": [
                                "string"
                            ],
                            "S": "string",
                            "SS": [
                                "string"
                            ]
                        }
                    },
                    "time_to_live": "number"
                }
            }
        ]
    }
}
```

Response Elements

unprocessed_keys

Type: String to object map

Errors

BackendInteractionException
ClusterIsNotConnectedException
NotImplementedError

ValidationError

Sample Request

```
{
    "request_items": {
        "Forum": [
            {
                "put_request": {
                    "item": {
                        "Name": {
                            "S": "MagnetoDB"
                        },
                        "Category": {
                            "S": "OpenStack KVaaS"
                        }
                    }
                }
            },
            {
                "put_request": {
                    "item": {
                        "Name": {
                            "S": "Nova"
                        },
                        "Category": {
                            "S": "OpenStack Core"
                        }
                    }
                }
            },
            {
                "put_request": {
                    "item": {
                        "Name": {
                            "S": "KeyStone"
                        },
                        "Category": {
                            "S": "OpenStack Core"
                        }
                    }
                }
            },
            {
                "delete_request": {
                    "key": {
                        "Name": {
                            "S": "Cinder"
                        },
                        "Category": {
                            "S": "OpenStack Core"
                        }
                    }
                }
            }
        ]
    }
}
```

Sample Response

```
{
    "unprocessed_items": {
        "Forum": [
            {
                "put_request": {
                    "item": {
                        "Name": {
                            "S": "Nova"
                        },
                        "Category": {
                            "S": "OpenStack Core"
                        }
                    }
                }
            }
        ]
    }
}
```

3.2.2 Monitoring API

As a magnetodb user I need to know how much data I have in table. As a magnetodb administrator I need to know now much space is used with user's table. As a accountant department I need to know how big user's table is in order to create a bill.

MagnetoDB monitoring actions

TableUsageDetails

`GET v1/monitoring/{project_id}/tables/{table_name}?metrics=metric1,metric2`

Request Syntax

This operation does not require a request body

Request Parameters:

Parameters should be provided via GET query string.

metrics

- Names of metrics to get
- Type: string
- Required: No

Response Syntax

```
{
    "size": "number",
    "item_count": "number"
}
```

Response Elements

size

Table size in bytes.
Type: Number

item_count

Number of items in table.
Type: Number

Errors

500

Sample Response

```
{
    "size": 1003432,
    "item_count": 3000
}
```

ProjectUsageDetails

GET v1/monitoring/projects/{project_id}?metrics=metric1,metric2

Request Syntax

This operation does not require a request body

Request Parameters:

Parameters should be provided via GET query string.

metrics

- Names of metrics to get
- Type: string
- Required: No

Response Syntax

```
[
    {
        "table_name": "string",
        "usage_detailes": {
            "size": "number",
            "item_count": "number"
        }
    }
]
```

Response Elements

table_name

Table name
Type: String

usage_detailes

Table usage detailes
Type: Object

size

Table size in bytes.
Type: Number

item_count

Number of items in table.
Type: Number

Errors

500

Sample Response

```
[
    {
        "table_name": "my_table",
        "usage_detailes": {
            "size": 1003432,
            "item_count": 3000
        }
    }
]
```

AllProjectsUsageDetails

GET v1/monitoring/projects?metrics=metric1,metric2

Request Syntax

This operation does not require a request body

Request Parameters:

Parameters should be provided via GET query string.

metrics

- Names of metrics to get
- Type: string
- Required: No

last_evaluated_project

- Last evaluated project ID (for pagination)
- Type: string
- Required: No

last_evaluated_table

- Last evaluated table name (for pagination)
- Type: string
- Required: No

limit

- Limit for response items count (for pagination)
- Type: Number

- Required: No

Response Syntax

```
[
    {
        "tenant": "string",
        "name": "string",
        "status": "string",
        "usage_detailes": {
            "size": "number",
            "item_count": "number"
        }
    }
]
```

Response Elements

tenant

Project ID
Type: String

name

Table name
Type: String

status

Table status
Type: String

usage_detailes

Table usage detailes
Type: Object

size

Table size in bytes.
Type: Number

item_count

Number of items in table.
Type: Number

Errors

500

Sample Response

```
[
    {
        "tenant": "12345678",
        "name": "my_table",
        "status": "ACTIVE",
        "usage_detailes": {
            "size": 1003432,
            "item_count": 3000
        }
```

```
    }
]
```

Monitoring API metric list

Name	Description
Size	Represents the total of the space used by table in bytes.
Item Count	Represents count of items in table.

Examples

All metrics for all tables in all projects:

http://{host}:8480/v1/monitoring/projects

All metrics for all tables in specified project:

http://{host}:8480/v1/monitoring/projects/{project_id}

All metrics for specified table:

http://{host}:8480/v1/monitoring/projects/{project_id}/tables/table_name

One metric for specified table:

http://{host}:8480/v1/monitoring/projects/{project_id}/ tables/table_name?metrics=size

Few metrics for specified table:

http://{host}:8480/v1/monitoring/projects/{project_id}/ tables/table_name?metrics=size,item_count

3.2.3 DynamoDB API

TBW

3.2.4 HealthCheck

`GET health_check`

Request Parameters

fullcheck

If fullcheck is 'true' subsystems availability will be checked
Syntax: healthcheck?fullcheck={true,false}
default: false

Request Syntax

This operation does not require a request body

Response Syntax

Response Status

200 or 503

Response Body

Content-Type: application/json
{“API”: “string”, “Identity”: “string”, “Messaging”: “string”, “Storage”: “string”}

Sample Response Body

{“API”: “OK”, “Identity”: “ERROR”, “Messaging”: “OK”, “Storage”: “OK”}

CHAPTER 4

User documentation

4.1 User guide

4.1.1 MagnetoDB Data Model

Data Model Concepts - Tables, Items, and Attributes

The MagnetoDB data model concepts include tables, items and attributes. In MagnetoDB, a database is a collection of tables. A table is a collection of items and each item is a collection of attributes. Except for the required primary key, a MagnetoDB table is schema-less. Individual items in a MagnetoDB table can have any number of attributes, although there is a limit of 64 KB on the item size. An item size is the sum of lengths of its attribute names and values (binary and UTF-8 lengths). Each attribute in an item is a name-value pair. An attribute can be single- valued or multi-valued set. For example, a person can have *Name* and *Phones* attributes. Each person has one name but can have several phone numbers. The multi-valued attribute is a set; duplicate values are not allowed. For example, consider storing a list of users in MagnetoDB.

You can create a table, *Users*, with the *Id* attribute as its primary key.:

```
"Users" table

{
    "Id": 1001,
    "Login": "admin",
    "Name": "John Doe",
    "OfficeNo": 42
}

{
    "Id": 1002,
    "Login": "raj",
    "Name": "Rajesh Koothrappali",
    "Phones": ["555-1212121", "555-1313131"],
    "DeptId": 34
}
```

In the example, the *Users* table contains two people with different sets of attributes. Person #1002 has *Phones* - multi-valued attribute. The *Id* is the only required attribute. Note that attribute values are shown using JSON-like syntax for illustration purposes. MagnetoDB does not allow null or empty string attribute values.

Primary Key

When you create a table, in addition to the table name, you must specify the primary key of the table. MagnetoDB supports the following two types of primary keys:

- **Hash Type Primary Key** — In this case the primary key is made of one attribute, a hash attribute. MagnetoDB builds an unordered hash index on this primary key attribute. In the preceding example, the hash attribute for the Users table is Id.
- **Hash and Range Type Primary Key** — In this case, the primary key is made of two attributes. The first attribute is the hash attribute and the second one is the range attribute. MagnetoDB builds an unordered hash index on the hash primary key attribute and a sorted range index on the range primary key attribute.

For example, to model a discussion forum, you can create a table, *Threads*, with the *Subject* attribute as a hash key and the *PostDateTime* as a range key. We will use *Subject* to identify discussion thread and *PostDateTime* identifies a message in the thread. Hence, pair of *Subject* and *PostDateTime* will uniquely defines a message through the whole discussion forum.:

```
"Threads" items

{
    "Subject": "Help needed",
    "PostDateTime": "2014-06-01 14:21:00",
    "AuthorId": 1002,
    "MessageBody": "I need help with my PC"
}

{
    "Subject": "Help needed",
    "PostDateTime": "2014-06-01 14:32:00",
    "AuthorId": 1001,
    "MessageBody": "Bring it to my office"
    "RepliesTo": "2014-06-01 14:21:00"
}
```

When designing MagnetoDB tables you have to take into account the fact that MagnetoDB does not support cross-table joins. In the example above, the *Threads* table stores only *AuthorId*. If you need the author's name, you can then parse the *AuthorId* attribute and use it to query the Users table.

Secondary Indexes

When you create a table with a hash-and-range key, you can optionally define one or more secondary indexes on that table. A secondary index lets you query the data in the table using an alternate key, in addition to queries against the primary key.

With the *Threads* table, you can query data items by *Subject* (hash) or by *Subject* and *PostDateTime* (hash and range). If you had an attribute in the table — *AuthorId*, with a secondary index on *AuthorId*, you could query the data by *Subject* (hash) and *AuthorId* (range). Such a query would let you retrieve all the replies posted by a particular user in a thread, with maximum efficiency and without having to access any other items.

The kind of secondary index that MagnetoDB supports is a local secondary index — an index that has the same hash key as the table, but a different range key. Technically, you can define as many local secondary indexes per table as you need. But note, that each index decreases performance of PutItem and UpdateItem operations.

MagnetoDB Data Types

MagnetoDB supports the following data types: * **Scalar data types** — Number, String, and Binary. * **Multi-valued types** — String Set, Number Set, and Binary Set.

Note that primary key attributes can be any scalar types, but not multi-valued types. The following are descriptions of each data type, along with examples. Note that the examples use JSON syntax.

String

Strings are Unicode with UTF8 binary encoding. There is no upper limit to the string size when you assign it to an attribute except when the attribute is part of the primary key. The length of the attribute must be greater than zero. String value comparison is used when returning ordered results in the Query and Scan APIs. Comparison is based on ASCII character code values. For example, “a” is greater that “A” , and “aa” is greater than “B”.

Example:

```
{"S": "John Doe"}
```

Number

Numbers are positive or negative exact-value decimals and integers. The representation in MagnetoDB is of variable length. Leading and trailing zeroes are trimmed. Serialized numbers are sent to MagnetoDB as String types, which maximizes compatibility across languages and libraries, however MagnetoDB handles them as the Number type for mathematical operations.

Example:

```
{"N": "42"}
```

Binary

Binary type attributes can store any binary data, for example, compressed data, encrypted data, or images. MagnetoDB treats each byte of the binary data as unsigned when it compares binary values, for example, when evaluating query expressions. The length of the attribute must be greater than zero. The following example is a binary attribute, using Base64-encoded text.

Example:

```
{"B": "MjAxNC0wMy0yMw=="}
```

String, Number, and Binary Sets

MagnetoDB also supports number sets, string sets and binary sets. Multi-valued attributes such as Authors attribute in a book item and Color attribute of a product item are examples of string set type attributes. Because it is a set, the values in the set must be unique. Attribute sets are not ordered; the order of the values returned in a set is not preserved. MagnetoDB does not support empty sets.

Examples:

```
{"SS": ["John Doe","Jane Smith"] }
{"NS": ["42","3.14","2.71828", "-12"] }
{"BS": ["MjAxNC0wMy0yMw==","MjAxNS0wMy0yNA==","MjAxNi0wNi0yNg=="] }
```

4.1.2 Supported Operations in MagnetoDB

To work with tables and items, MagnetoDB offers the following set of operations:

Table Operations

MagnetoDB provides operations to create and delete tables. MagnetoDB also supports an operation to retrieve table information (the DescribeTable operation) including the current status of the table, the primary key, and when the table was created. The ListTables operation enables you to get a list of tables.

Item Operations

Item operations enable you to add, update and delete items from a table. The UpdateItem operation allows you to update existing attribute values, add new attributes, and delete existing attributes from an item. You can also perform conditional updates. For example, if you are updating a price value, you can set a condition so the update happens only if the current price is $10.

MagnetoDB provides an operation to retrieve a single item (GetItem) or multiple items (BatchGetItem). You can use the BatchGetItem operation to retrieve items from multiple tables.

Query and Scan

The Query operation enables you to query a table using the hash attribute and an optional range filter. If the table has a secondary index, you can also Query the index using its key. You can query only tables whose primary key is of hash-and-range type; you can also query any secondary index on such tables. Query is the most efficient way to retrieve items from a table or a secondary index.

MagnetoDB also supports a Scan operation, which you can use on a query or a secondary index. The Scan operation reads every item in the table or secondary index. For large tables and secondary indexes, a Scan can consume a large amount of resources; for this reason, we recommend that you design your applications so that you can use the Query operation mostly, and use Scan only where appropriate. You can use conditional expressions in both the Query and Scan operations to control which items are returned.

4.1.3 Accessing MagnetoDB

MagnetoDB is a web service that uses HTTP and HTTPS as a transport and JavaScript Object Notation (JSON) as a message serialization format. Your application code can make requests directly to the MagnetoDB web service API. Each request must contain a valid JSON payload and correct HTTP headers, including a valid authentication token.

4.2 Magnetodb CLI

4.2.1 MagnetoDB CLI

MagnetoDB CLI client is dedicated to simplify communication with magnetodb server. It uses keystone v2/v3 for authentication. To provide auth info you need to export environment variables OS_TENANT_NAME, OS_USERNAME, OS_PASSWORD and OS_AUTH_URL or provide them on command call, e.g.:

```
$ magnetodb --os-auth-url http://127.0.0.1:5000/v3 --os-password pass --os-tenant-name admin --os-use
```

MagnetoDB CLI client provides following commands:

table-create

Creates a new table in MagnetoDB.

Command syntax:

```
$ magnetodb table-create --request-file <FILE>
```

<FILE> - path to file that contains json request:

```
{
    "attribute_definitions": [
        {
            "attribute_name": "string",
            "attribute_type": "string"
        }
    ],
    "key_schema": [
        {
            "attribute_name": "string",
            "key_type": "string"
        }
    ],
    "local_secondary_indexes": [
        {
            "index_name": "string",
            "key_schema": [
                {
                    "attribute_name": "string",
                    "key_type": "string"
                }
            ],
            "projection": {
                "non_key_attributes": [
                    "string"
                ],
                "projection_type": "string"
            }
        }
    ],
    "table_name": "string"
}
```

Sample:

```
$ magnetodb table-create --request-file ~/table-create-request.json
```

~/table-create-request.json contains:

```
{
    "attribute_definitions": [
        {
            "attribute_name": "ForumName",
            "attribute_type": "S"
        },
        {
            "attribute_name": "Subject",
            "attribute_type": "S"
        },
        {
```

```
            "attribute_name": "LastPostDateTime",
            "attribute_type": "S"
        }
    ],
    "table_name": "Thread",
    "key_schema": [
        {
            "attribute_name": "ForumName",
            "key_type": "HASH"
        },
        {
            "attribute_name": "Subject",
            "key_type": "RANGE"
        }
    ],
    "local_secondary_indexes": [
        {
            "index_name": "LastPostIndex",
            "key_schema": [
                {
                    "attribute_name": "ForumName",
                    "key_type": "HASH"
                },
                {
                    "attribute_name": "LastPostDateTime",
                    "key_type": "RANGE"
                }
            ],
            "projection": {
                "projection_type": "KEYS_ONLY"
            }
        }
    ]
}
```

table-list

Prints a list of tables in tenant.

Command syntax:

```
$ magnetodb table-list [--limit <max-tables-to-show>] [--start-table-name <table-name>]
```

–limit - max tables to show in a list –start-table-name - table name, after which tables will be listed

Sample:

```
$ magnetodb table-list --limit 3 --start-table-name Thread
```

table-delete

Deletes a table.

Command syntax:

```
$ magnetodb table-delete <table-name>
```

Sample:

```
$ magnetodb table-delete Thread
```

table-describe

Prints a description of a table.

Command syntax:

```
$ magnetodb table-describe <table-name>
```

Sample:

```
$ magnetodb table-describe Thread
```

index-list

Prints list of indexes of a given table.

Command syntax:

```
$ magnetodb index-list <table-name>
```

Sample:

```
$ magnetodb index-list Thread
```

index-show

Describes index of a table.

Command syntax:

```
$ magnetodb index-show <table-name> <index-name>
```

Sample:

```
$ magnetodb index-show Thread LastPostIndex
```

item-put

Puts item to a given table.

Command syntax:

```
$ magnetodb item-put <table-name> --request-file <FILE>
```

<FILE> - path to file that contains json request:

```
{
    "expected": {
        "string": {
            "exists": "boolean",
            "value": {
                "B": "blob",
```

```
                "BS": [
                    "blob"
                ],
                "N": "string",
                "NS": [
                    "string"
                ],
                "S": "string",
                "SS": [
                    "string"
                ]
            }
        }
    },
    "item": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
    "time_to_live": "number",
    "return_values": "string"
}
```

Sample:

```
$ magnetodb item-put Thread --request-file ~/item-put-request.json
```

~/item-put-request.json contains:

```
{
    "item": {
        "LastPostDateTime": {
            "S": "201303190422"
        },
        "Tags": {
            "SS": ["Update", "Multiple items", "HelpMe"]
        },
        "ForumName": {
            "S": "MagnetoDB"
        },
        "Message": {
            "S": "I want to update multiple items."
        },
        "Subject": {
            "S": "How do I update multiple items?"
        },
        "LastPostedBy": {
            "S": "fred@example.com"
```

```
        }
    },
    "expected": {
        "ForumName": {
            "exists": false
        },
        "Subject": {
            "exists": false
        }
    },
    "return_values": "ALL_OLD"
}
```

item-update

Updates item.

Command syntax:

```
$ magnetodb item-update <table-name> --request-file <FILE>
```

<FILE> - path to file that contains json request:

```
{
    "attribute_updates": {
        "string": {
            "action": "string",
            "value": {
                "B": "blob",
                "BS": [
                    "blob"
                ],
                "N": "string",
                "NS": [
                    "string"
                ],
                "S": "string",
                "SS": [
                    "string"
                ]
            }
        }
    },
    "time_to_live": "number",
    "expected": {
        "string": {
            "exists": "boolean",
            "value": {
                "B": "blob",
                "BS": [
                    "blob"
                ],
                "N": "string",
                "NS": [
                    "string"
                ],
                "S": "string",
                "SS": [
```

```
                    "string"
                ]
            }
        }
    },
    "key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
    "return_values": "string"
}
```

Sample:

```
$ magnetodb item-update Thread --request-file ~/item-update-request.json
```

~/item-put-request.json contains:

```
{
    "key": {
        "ForumName": {
            "S": "MagnetoDB"
        },
        "Subject": {
            "S": "How do I delete an item?"
        }
    },
    "attribute_updates": {
        "LastPostedBy": {
            "value": {
                "S": "me@test.com"
            },
            "action": "PUT"
        }
    },
    "expected": {
        "LastPostedBy": {
            "value": {
                "S": "fred@example.com"
            }
        },
         "Replies": {
            "exists": false
        }
   },
    "return_values": "ALL_NEW"
}
```

item-delete

Deletes item from a given table.

Command syntax:

```
$ magnetodb item-delete <table-name> --request-file <FILE>
```

<FILE> - path to file that contains json request:

```
{
    "key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
    "expected": {
        "string": {
            "exists": "boolean",
            "value": {
                "B": "blob",
                "BS": [
                    "blob"
                ],
                "N": "string",
                "NS": [
                    "string"
                ],
                "S": "string",
                "SS": [
                    "string"
                ]
            }
        }
    },
    "return_values": "string"
}
```

Sample:

```
$ magnetodb item-delete Thread --request-file ~/item-delete-request.json
```

~/item-delete-request.json contains:

```
{
    "key": {
        "ForumName": {
            "S": "MagnetoDB"
        },
```

```
        "Subject": {
            "S": "How do I delete an item?"
        }
    },
    "attribute_updates": {
        "LastPostedBy": {
            "value": {
                "S": "me@test.com"
            },
            "action": "PUT"
        }
    },
    "expected": {
        "LastPostedBy": {
            "value": {
                "S": "fred@example.com"
            }
        },
         "Replies": {
            "exists": false
        }
   },
    "return_values": "ALL_NEW"
}
```

item-get

Gets item from a given table.

Command syntax:

```
$ magnetodb item-get <table-name> --request-file <FILE>
```

<FILE> - path to file that contains json request:

```
{
    "attributes_to_get": [
        "string"
    ],
    "consistent_read": "boolean",
    "key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    }
}
```

Sample:

```
$ magnetodb item-get Thread --request-file ~/item-get-request.json
```

~/item-get-request.json contains:

```
{
    "key": {
        "ForumName": {
            "S": "MagnetoDB"
        },
        "Subject": {
            "S": "How do I update multiple items?"
        }
    },
    "attributes_to_get": ["LastPostDateTime", "Message", "Tags"],
    "consistent_read": true
}
```

query

Makes query request to a given table.

Command syntax:

```
$ magnetodb query <table-name> --request-file <FILE>
```

<FILE> - path to file that contains json request:

```
{
    "attributes_to_get": [
        "string"
    ],
    "consistent_read": "boolean",
    "exclusive_start_key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
    "index_name": "string",
    "key_conditions": {
        "string": {
            "attribute_value_list": [
                {
                    "B": "blob",
                    "BS": [
                        "blob"
                    ],
```

```
                    "N": "string",
                    "NS": [
                        "string"
                    ],
                    "S": "string",
                    "SS": [
                        "string"
                    ]
                }
            ],
            "comparison_operator": "string"
        }
    },
    "limit": "number",
    "scan_index_forward": "boolean",
    "select": "string"
}
```

Sample:

```
$ magnetodb query Thread --request-file ~/query-request.json
```

~/query-request.json contains:

```
{
    "attributes_to_get": [
        "ForumName", "LastPostDateTime", "Posts"
    ],
    "exclusive_start_key": {
        "ForumName": {
            "S": "Testing OS API"
        },
        "LastPostDayTime": {
            "S": "3/1/14"
        }
    },
    "index_name": "LastPostIndex",
    "limit": 2,
    "consistent_read": true,
    "scan_index_forward": true,
    "key_conditions": {
        "ForumName": {
            "attribute_value_list": [
                {
                    "S": "Testing OS API"
                }
            ],
            "comparison_operator": "EQ"
        },
        "LastPostDateTime": {
            "attribute_value_list": [
                {
                    "S": "3/10/14"
                }
            ],
            "comparison_operator": "GT"
        }
    },
    "select": "SPECIFIC_ATTRIBUTES"
```

```
}
```

scan

Makes scan request to a given table.

Command syntax:

```
$ magnetodb scan <table-name> --request-file <FILE>
```

<FILE> - path to file that contains json request:

```
{
    "attributes_to_get": [
        "string"
    ],
    "exclusive_start_key": {
        "string": {
            "B": "blob",
            "BS": [
                "blob"
            ],
            "N": "string",
            "NS": [
                "string"
            ],
            "S": "string",
            "SS": [
                "string"
            ]
        }
    },
    "limit": "number",
    "scan_filter": {
        "string": {
            "attribute_value_list": [
                {
                    "B": "blob",
                    "BS": [
                        "blob"
                    ],
                    "N": "string",
                    "NS": [
                        "string"
                    ],
                    "S": "string",
                    "SS": [
                        "string"
                    ]
                }
            ],
            "comparison_operator": "string"
        }
    },
    "segment": "number",
    "select": "string",
    "total_segments": "number"
}
```

Sample:

```
$ magnetodb scan Thread --request-file ~/scan-request.json
```

~/scan-request.json contains:

```
{
    "attributes_to_get": [
        "ForumName", "LastPostDateTime", "Posts"
    ],
    "exclusive_start_key": {
        "ForumName": {
            "S": "Another forum"
        }
    },
    "limit": 2,
    "scan_filter": {
        "LastPostDateTime": {
            "attribute_value_list": [
                {
                    "S": "3/10/14"
                }
            ],
            "comparison_operator": "GT"
        }
    },
    "segment": 0,
    "select": "SPECIFIC_ATTRIBUTES",
    "total_segments": 1
}
```

batch-write

Makes batch write item request.

Command syntax:

```
$ magnetodb batch-write --request-file <FILE>
```

<FILE> - path to file that contains json request:

```
{
    "request_items": {
        "string": [
            {
                "delete_request": {
                    "key": {
                        "string": {
                            "B": "blob",
                            "BS": [
                                "blob"
                            ],
                            "N": "string",
                            "NS": [
                                "string"
                            ],
                            "S": "string",
                            "SS": [
                                "string"
```

```
                            ]
                        }
                    }
                },
                "put_request": {
                    "item": {
                        "string": {
                            "B": "blob",
                            "BS": [
                                "blob"
                            ],
                            "N": "string",
                            "NS": [
                                "string"
                            ],
                            "S": "string",
                            "SS": [
                                "string"
                            ]
                        }
                    },
                    "time_to_live": "number"
                }
            }
        ]
    }
}
```

Sample:

```
$ magnetodb batch-write --request-file ~/batch-write-request.json
```

~/batch-write-request.json contains:

```
{
    "request_items": {
        "Forum": [
            {
                "put_request": {
                    "item": {
                        "Name": {
                            "S": "MagnetoDB"
                        },
                        "Category": {
                            "S": "OpenStack KVaaS"
                        }
                    }
                }
            },
            {
                "put_request": {
                    "item": {
                        "Name": {
                            "S": "Nova"
                        },
                        "Category": {
                            "S": "OpenStack Core"
                        }
                    }
```

```
                }
            },
            {
                "put_request": {
                    "item": {
                        "Name": {
                            "S": "KeyStone"
                        },
                        "Category": {
                            "S": "OpenStack Core"
                        }
                    }
                }
            },
            {
                "delete_request": {
                    "key": {
                        "Name": {
                            "S": "Cinder"
                        },
                        "Category": {
                            "S": "OpenStack Core"
                        }
                    }
                }
            }
        ]
    }
}
```

batch-get

Makes batch get item request.

Command syntax:

```
$ magnetodb batch-get --request-file <FILE>
```

<FILE> - path to file that contains json request:

```
{
    "request_items": {
        "string": {
            "attributes_to_get": [
                "string"
            ],
            "consistent_read": "boolean",
            "keys": [

                {
                    "string": {
                        "B": "blob",
                        "BS": [
                            "blob"
                        ],
                        "N": "string",
                        "NS": [
                            "string"
```

```
                        ],
                        "S": "string",
                        "SS": [
                            "string"
                        ]
                    }
                }
            ]
        }
    }
}
```

Sample:

```
$ magnetodb batch-get --request-file ~/batch-get-request.json
```

~/batch-get-request.json contains:

```
{
    "request_items": {
        "Forum": {
            "keys": [
                {
                    "Name": {
                        "S": "MagnetoDB"
                    },
                    "Category": {
                        "S": "OpenStack KVaaS"
                    }
                },
                {
                    "Name": {
                        "S": "Nova"
                    },
                    "Category": {
                        "S": "OpenStack Core"
                    }
                }
            ]
        },
        "Thread": {
            "keys": [
                {
                    "Name": {
                        "S": "MagnetoDB"
                    },
                    "Category": {
                        "S": "OpenStack KVaaS"
                    }
                },
                {
                    "Name": {
                        "S": "Nova"
                    },
                    "Category": {
                        "S": "OpenStack Core"
                    }
                }
            ]
```

```
            }
        }
}
```

CHAPTER 5

Admin guides

5.1 Installation guide

Content

5.1.1 Introduction

This document describes how to install MagnetoDB on Ubuntu 12.04 using Apache Cassandra as database backend.

The following components will be installed also

- Cassandra cluster with 3 nodes
- JDK and JNA
- Python 2.7

All packages will be installed in `/opt/`

5.1.2 Requirements

MagnetoDB uses authorization via Keystone. Keystone should be configured beforehand on this or another host. You should configure user space for MagnetoDB in Keystone. This document does not describe configuring Keystone.

You should have root access or be logged in as any user with rights to execute commands via sudo.

5.1.3 Creating a User

Creating a user ‘’magneto’‘:

```
groupadd magneto
useradd -g magneto -s /bin/bash -d /home/magneto -m magneto
passwd magneto
Enter new UNIX password:*****
Retype new UNIX password:*****
```

Giving magneto user passwordless sudo privileges:

```
grep -q "^#includedir.*/etc/sudoers.d" /etc/sudoers || echo "#includedir /etc/sudoers.d" >> /etc/sudo
( umask 226 && echo "magneto ALL=(ALL) NOPASSWD:ALL" > /etc/sudoers.d/50_magneto )

su magneto
cd ~
```

5.1.4 Installing JDK and JNA

Installing packages:

```
sudo apt-get -y install openjdk-7-jdk libjna-java
sudo update-alternatives --set java /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java
```

5.1.5 Installing Python 2.7

Ubuntu 12.04 already has python2.7

5.1.6 Installing Cassandra

It is recommended to deploy Cassandra cluster on dedicated hardware. However in order to try it, you can use one node installation as described below.

Please don't do it for production.

To install the Cassandra cluster on the same node, we recommend using the CCM (Cassandra Cluster Manager) https://github.com/pcmanus/ccm

ccm works on localhost only for now. So if you want to create more than one node clusters the simplest way is to use multiple loopback aliases. In this example we will build a cluster of three nodes.

Creating loopback aliases:

```
sudo ip addr add 127.0.0.2/8 dev lo
sudo ip addr add 127.0.0.3/8 dev lo

# Checking results using this command:
sudo ip addr show lo
```

Installing required packages:

```
sudo apt-get update
sudo apt-get -y install ant libyaml-0-2 libyaml-dev python-setuptools python-yaml libev4 libev-dev
```

Installing Cassandra Cluster Manager:

```
sudo mkdir -p /opt/ccm
sudo chown -R magneto:magneto /opt/ccm
```

```
git clone https://github.com/pcmanus/ccm.git /opt/ccm
cd /opt/ccm
sudo ./setup.py install
```

Creating a cluster named ''Storage" of three nodes of Cassandra 2.1.3:

```
ccm create Storage -v 2.1.3
ccm populate -n 3
```

Starting Cassandra Cluster:

```
ccm start

# Checking results using this commands:
ccm status
ccm node1 ring
```

Creating keyspaces in cassandra:

```
# Replication factor is 3
echo "CREATE KEYSPACE magnetodb WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor
echo "CREATE KEYSPACE user_default_tenant WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replicati
echo 'CREATE TABLE magnetodb.table_info(tenant text, name text, id uuid, exists int, "schema" text, s
echo 'CREATE TABLE magnetodb.backup_info(tenant text, table_name text, id uuid, name text, status tex
echo 'CREATE TABLE magnetodb.restore_info(tenant text, table_name text, id uuid, status text, backup_
echo 'CREATE TABLE magnetodb.dummy(id int PRIMARY KEY);' >> ~/.ccm/cql.txt
ccm node1 cqlsh -f ~/.ccm/cql.txt
```

5.1.7 Installing MagnetoDB

Installing required packages:

```
sudo apt-get -y install build-essential python-dev
sudo easy_install-2.7 pip
```

Installing MagnetoDB:

```
sudo mkdir -p /opt/magnetodb
sudo chown -R magneto:magneto /opt/magnetodb

git clone https://git.openstack.org/stackforge/magnetodb.git /opt/magnetodb
cd /opt/magnetodb
sudo pip2.7 install -r requirements.txt -r test-requirements.txt
```

Creating directories and log files:

```
sudo mkdir -p /var/log/magnetodb
sudo touch /var/log/magnetodb/magnetodb.log
sudo touch /var/log/magnetodb/magnetodb-streaming.log
sudo touch /var/log/magnetodb/magnetodb-async-task-executor.log
sudo chown -R magneto:magneto /var/log/magnetodb
```

Configuring MagnetoDB

Before starting magnetos must specify your own values for some variables in the configuration files: `/opt/magnetodb/etc/api-paste.ini`, `/opt/magnetodb/etc/streaming-api-paste.ini`, `/opt/magnetodb/etc/magnetodb-api.conf`, `/opt/magnetodb/etc/magnetodb-async-task-executor.conf`
As a minimum, you must specify a value for the following variables as example:

```
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = magnetodb
admin_password = magneto-password

auth_uri = http://127.0.0.1:5000/v3

rabbit_host = localhost
rabbit_userid = userid
rabbit_password = pass
```

Running MagnetoDB:

```
python /opt/magnetodb/bin/magnetodb-api-server --config-file /opt/magnetodb/etc/magnetodb-api-server
python /opt/magnetodb/bin/magnetodb-streaming-api-server --config-file /opt/magnetodb/etc/magnetodb-s
python /opt/magnetodb/bin/magnetodb-async-task-executor --config-file /opt/magnetodb/etc/magnetodb-as
```

5.2 Configuration guide

5.2.1 Configuring MagnetoDB

Once MagnetoDB is installed, it is configured via configuration files (etc/magnetodb/magnetodb-api-server.conf, etc/magnetodb/magnetodb-api.conf), a PasteDeploy configuration file (etc/magnetodb/api-paste.ini).

By default, MagnetoDB starts a service on port 8480 (you can change bind port and host in *magnetodb-api-server.conf*).

Also you can run MagnetoDB with gunicorn (multithread). Use configs *-gunicorn*

Starting and Stopping MagnetoDB

Start MagnetoDB services using the command:

```
$ magnetodb-api-server --config-file /etc/magnetodb/magnetodb-api-server.conf
```

For start MagnetoDB with gunicorn WSGI server use command:

```
$ magnetodb-api-server-gunicorn --config-file /etc/magnetodb/magnetodb-api-server-gunicorn.conf
```

Also you can specify number of workers in magnetodb-api-server-gunicorn.conf (example magnetodb_api_workers = 4).

Invoking these commands starts up wsgi.Server instance (the primary/public API interface). Stop the process using Control-C.

Configuration Files

The MagnetoDB configuration files are an *ini* file format based on Paste, a common system used to configure Python WSGI based applications. The PasteDeploy configuration entries (WSGI pipeline definitions) can be provided in a separate *api-paste.ini* file, while general and driver-specific configuration parameters are in the primary configuration file *magnetodb-api.conf*. The api-paste.ini configuration file is organized into the following sections:

[pipeline:main] is used when you need apply a number of filters. It takes one configuration key pipeline (plus any global configuration overrides you want). pipeline is a list of filters ended by an application.

Two main filters are ec2authtoken and tokenauth:

[filter:ec2authtoken] - check EC2 credentials in request headers (for DynamoDB API)

- auth_uri: complete public Identity API endpoint (string value) for checking EC2 credentials (http://127.0.0.1:5000/v3)

[filter:tokenauth] - checks the validity of the token in Keystone from the service user (usually "magnetodb"). For this action rol

- auth_host: host providing the admin Identity API endpoint (string value)
- auth_port: port of the admin Identity API endpoint (integer value)
- auth_protocol: protocol of the admin Identity API endpoint(http or https)
- admin_tenant_name: Keystone service account tenant name to validate user tokens (string value)
- admin_user: Keystone account username (string value)
- admin_password: Keystone account password (string value)
- auth_version: API version of the admin Identity API endpoint (string value)
- admin_token: single shared secret with the Keystone configuration used for bootstrapping a Keystone installation, or otherwise bypassing the normal authentication process (string value)
- signing_dir: directory used to cache files related to PKI tokens (string value)

Note: signing_dir is configurable, but the default behavior of the authtoken middleware should be sufficient. It will create a temporary directory in the home directory for the user the MagnetoDB process is running as.

.conf file

The magnetodb-api.conf configuration file is organized into the following sections:

DEFAULT (logging configuration)
RPC Configuration Options
Notification System Options
Storage Manager Config

[DEFAULT]

- verbose: show more verbose log output (sets INFO log level output) <boolean value>
- debug: show debugging output in logs (sets DEBUG log level output) <boolean value>
- log_file: path to log file <string value>
- log_config: path to logging config file, if it is specified, options 'verbose', 'debug', 'log_file', 'use_syslog', 'use_stderr', 'publish_errors', 'log_format', 'default_log_levels', 'log_date_format' will be ignored
- use_syslog: use Syslog for logging <boolean value>
- syslog_log_facility: Syslog facility to receive log lines <string value>
- logging_exception_prefix: format exception prefix <string value>

[PROBE]

- enabled: enables additional diagnostic log output
- suppress_args: suppresses args output

[RPC Configuration Options]

- rpc_backend: the messaging module to use (one of rabbit, qpid, zmq; rabbit by default)
- rpc_thread_pool_size: size of rpc thread pool
- rpc_conn_pool_size: size of RPC connection pool
- rpc_response_timeout: seconds to wait for a response from call or multicall
- rpc_cast_timeout: seconds to wait before a cast expires (only supported by impl_zmq)
- allowed_rpc_exception_modules: modules of exceptions that are permitted to be recreated upon receiving exception data from an rpc call (neutron.openstack.common.exception, nova.exception)
- control_exchange: AMQP exchange to connect to if using RabbitMQ or QPID
- fake_rabbit: if passed, use a fake RabbitMQ provider

Configuration options if sending notifications via rabbit rpc (these are the defaults):

- kombu_ssl_version: SSL version to use (valid only if SSL enabled)
- kombu_ssl_keyfile: SSL key file (valid only if SSL enabled)
- kombu_ssl_certfile: SSL cert file (valid only if SSL enabled)
- kombu_ssl_ca_certs: SSL certification authority file (valid only if SSL enabled)
- rabbit_host: IP address of the RabbitMQ installation
- rabbit_password: password of the RabbitMQ server
- rabbit_port: port where RabbitMQ server is running/listening
- rabbit_hosts: RabbitMQ single or HA cluster (host:port pairs i.e: host1:5672, host2:5672) rabbit_hosts is defaulted to '$rabbit_host:$rabbit_port'
- rabbit_userid: user ID used for RabbitMQ connections
- rabbit_virtual_host: location of a virtual RabbitMQ installation.
- rabbit_max_retries: maximum retries with trying to connect to RabbitMQ (the default of 0 implies an infinite retry count)
- rabbit_retry_interval: RabbitMQ connection retry interval
- rabbit_ha_queues: use HA queues in RabbitMQ (x-ha-policy: all). You need to wipe RabbitMQ database when changing this option (boolean value)

QPID (rpc_backend=qpid):

- qpid_hostname: Qpid broker hostname
- qpid_port: Qpid broker port
- qpid_hosts: Qpid single or HA cluster (host:port pairs i.e: host1:5672, host2:5672) qpid_hosts is defaulted to '$qpid_hostname:$qpid_port'
- qpid_username: username for qpid connection
- qpid_password: password for qpid connection

- qpid_sasl_mechanisms: space separated list of SASL mechanisms to use for auth
- qpid_heartbeat: seconds between connection keepalive heartbeats
- qpid_protocol: transport to use, either 'tcp' or 'ssl'
- qpid_tcp_nodelay: disable Nagle algorithm

ZMQ (rpc_backend=zmq):

- rpc_zmq_bind_address: ZeroMQ bind address. Should be a wildcard (*), an ethernet interface, or IP. The "host" option should point or resolve to this address.

[Notification System Options] Notifications can be sent when tables are created, or deleted, or data items are inserted/deleted/updated/retrieved. There are three methods of sending notifications: logging (via the log_file directive), rpc (via a message queue) and noop (no notifications sent, the default):

<magnetodb property>

- notification_service: together with default_publisher_id, this becomes the publisher_id (for example: magnetodb.myhost.com)

<notification engine property>

- notification_driver = no_op (do nothing driver)
- notification_driver = log (logging driver)
- notification_driver = messaging (RPC driver)
- default_publisher_id: default_publisher_id is a part of the notification payload
- notification_topics: defined in messaging driver , can be comma separated values. AMQP topic used for OpenStack notifications.

Note: notification_driver can be defined multiple times.

[Storage Manager Config] Storage manager config it is a simple string from the point of view of oslo.config. But this string should be a well-formed JSON which is a map of object specifications for object instantiation. Each element of this map is object specification and it is JSON of tne next format:

```
{
    "type": "<factory method or class object name>",
    "args": [<position arguments for object initialization >],
    "kwargs": {<keyword arguments map for object initialization>}
}
```

Each of these objects will be created and added to result context (map of object name to object). You can specify name of object in context as argument value to initialize another object in context using "@" prefix. For example if you define context like:

```
{
    "cluster_params": {
        "type": "cassandra.cluster.Cluster",
        "kwargs": {
            "contact_points": ["localhost"],
            "control_connection_timeout": 60,
            "max_schema_agreement_wait": 300
        }
    },
    "cluster_handler": {
        "type": "magnetodb.common.cassandra.cluster_handler.ClusterHandler",
```

```
        "kwargs": {
            "cluster_params": "@cluster_params",
            "query_timeout": 60,
            "concurrent_queries": 100
        }
    }
}
```

Object with name "cluster_params" will be created at the beginning and then this object will be used for initialization of object with name "cluster_handler".

Also you can escape you "@" using "@@" if you need to specify string which starts with @, not a link to another object from context.

cassandra_connection:

- type: <factory method or class object name>
- args: <position arguments for object initialization >
- **kwargs: <keyword arguments map for object initialization>**
  - in_buffer_size
  - out_buffer_size
  - cql_version: if a specific version of CQL should be used, this may be set to that string version. Otherwise, the highest CQL version supported by the server will be automatically used.
  - protocol_version: the version of the native protocol to use (with Cassandra 2.0+ you should use protocol version 2).
  - keyspace
  - compression: controls compression for communications between the driver and Cassandra. If left as the default of True, either lz4 or snappy compression may be used, depending on what is supported by both the driver and Cassandra. If both are fully supported, lz4 will be preferred. You may also set this to 'snappy' or 'lz4' to request that specific compression type. Setting this to False disables compression.
  - compressor
  - decompressor
  - ssl_options: a optional dict which will be used as kwargs for ssl.wrap_socket() when new sockets are created. This should be used when client encryption is enabled in Cassandra. By default, a ca_certs value should be supplied (the value should be a string pointing to the location of the CA certs file), and you probably want to specify ssl_version as ssl.PROTOCOL_TLSv1 to match Cassandra's default protocol.
  - last_error
  - in_flight
  - is_defunct
  - is_closed
  - lock
  - is_control_connection

cluster_params:

- type: <factory method or class object name>

- args: <position arguments for object initialization >
- **kwargs: <keyword arguments map for object initialization>**
  - connection_class - Cassandra connection class.
  - contact_points
  - port: the server-side port to open connections to (defaults to 9042).
  - compression: controls compression for communications between the driver and Cassandra. If left as the default of True, either lz4 or snappy compression may be used, depending on what is supported by both the driver and Cassandra. If both are fully supported, lz4 will be preferred. You may also set this to 'snappy' or 'lz4' to request that specific compression type. Setting this to False disables compression.
  - auth_provider: when protocol_version is 2 or higher, this should be an instance of a subclass of AuthProvider, such as PlainTextAuthProvider. When not using authentication, this should be left as None.
  - load_balancing_policy: an instance of policies.LoadBalancingPolicy or one of its subclasses. Defaults to RoundRobinPolicy.
  - reconnection_policy: an instance of policies.ReconnectionPolicy. Defaults to an instance of ExponentialReconnectionPolicy with a base delay of one second and a max delay of ten minutes.
  - default_retry_policy: a default policies.RetryPolicy instance to use for all Statement objects which do not have a retry_policy explicitly set.
  - conviction_policy_factory: a factory function which creates instances of policies.ConvictionPolicy. Defaults to policies.SimpleConvictionPolicy ;
  - metrics_enabled: whether or not metric collection is enabled. If enabled, cluster_metrics will be an instance of Metrics.
  - **connection_class: this determines what event loop system will be used for managing I/O with Cassandra. These**
    - cassandra.io.asyncorereactor.AsyncoreConnection
    - cassandra.io.libevreactor.LibevConnection
    - cassandra.io.libevreactor.GeventConnection (requires monkey-patching)
    - cassandra.io.libevreactor.TwistedConnection

      By default, AsyncoreConnection will be used, which uses the asyncore module in the Python standard library. The performance is slightly worse than with libev, but it is supported on a wider range of systems. If libev is installed, LibevConnection will be used instead. If gevent monkey-patching of the standard library is detected, GeventConnection will be used automatically.
  - ssl_options: a optional dict which will be used as kwargs for ssl.wrap_socket() when new sockets are created. This should be used when client encryption is enabled in Cassandra. By default, a ca_certs value should be supplied (the value should be a string pointing to the location of the CA certs file), and you probably want to specify ssl_version as ssl.PROTOCOL_TLSv1 to match Cassandra's default protocol.
  - sockopts: an optional list of tuples which will be used as arguments to socket.setsockopt() for all created sockets.
  - cql_version: if a specific version of CQL should be used, this may be set to that string version. Otherwise, the highest CQL version supported by the server will be automatically used.

  - protocol_version: the version of the native protocol to use (with Cassandra 2.0+ you should use protocol version 2).
  - executor_threads
  - max_schema_agreement_wait: the maximum duration (in seconds) that the driver will wait for schema agreement across the cluster. Defaults to ten seconds.
  - control_connection_timeout: a timeout, in seconds, for queries made by the control connection, such as querying the current schema and information about nodes in the cluster. If set to None, there will be no timeout for these queries.

cluster_handler:

- type: <factory method or class object name>
- args: <position arguments for object initialization >
- **kwargs: <keyword arguments map for object initialization>**
  - cluster - Cluster object
  - query_timeout - Seconds count to wait for CQL query completion
  - concurrent_queries - max number of started but not completed CLQ queries

table_info_repo:

- type: <factory method or class object name>
- args: <position arguments for object initialization >
- **kwargs: <keyword arguments map for object initialization>**
  - cluster_handler - ClusterHandler object

storage_driver:

- type: <factory method or class object name>
- args: <position arguments for object initialization >
- **kwargs: <keyword arguments map for object initialization>**
  - cluster_handler - ClusterHandler object
  - default_keyspace_opts - map of Cassandra keyspace properties, which will be used for tenant's keyspace creation if it doesn't exist

storage_manager:

- type: <factory method or class object name>
- args: <position arguments for object initialization >
- **kwargs: <keyword arguments map for object initialization>**
  - storage_driver - StorageDriver object
  - table_info_repo - TableInfoRepo object
  - concurrent_tasks - max number of started but not completed storage_driver methods invocations
  - batch_chunk_size - size of internal chunks to which original batch will be split. It is needed because large batches may impact Cassandra latency for another concurrent queries
  - schema_operation_timeout - timeout in seconds, after which CREATING or DELETING table state will be changed to CREATE_FAILURE or DELETE_FAILURE respectively

Default storage manager config

```
storage_manager_config =
    {
        "cassandra_connection": {
            "type": "eval",
            "args": [
                "importutils.import_class('magnetodb.common.cassandra.io.eventletreactor.EventletConn
            ]
        },
        "cluster_params": {
            "type": "dict",
            "kwargs": {
                "connection_class": "@cassandra_connection",
                "contact_points": ["localhost"],
                "control_connection_timeout": 60,
                "max_schema_agreement_wait": 300
            }
        },
        "cluster_handler": {
            "type": "magnetodb.common.cassandra.cluster_handler.ClusterHandler",
            "kwargs": {
                "cluster_params": "@cluster_params",
                "query_timeout": 60,
                "concurrent_queries": 100
            }
        },
        "table_info_repo": {
            "type": "magnetodb.storage.table_info_repo.cassandra_impl.CassandraTableInfoRepository",
            "kwargs": {
                "cluster_handler": "@cluster_handler"
            }
        },
        "storage_driver": {
            "type": "magnetodb.storage.driver.cassandra.cassandra_impl.CassandraStorageDriver",
            "kwargs": {
                "cluster_handler": "@cluster_handler",
                "table_info_repo": "@table_info_repo",
                "default_keyspace_opts": {
                    "replication": {
                        "replication_factor": 3,
                        "class": "SimpleStrategy"
                    }
                }
            }
        },
        "storage_manager": {
            "type": "magnetodb.storage.manager.queued_impl.QueuedStorageManager",
            "kwargs": {
                "storage_driver": "@storage_driver",
                "table_info_repo": "@table_info_repo",
                "concurrent_tasks": 1000,
                "batch_chunk_size": 25,
                "schema_operation_timeout": 300
            }
        }
    }
```

5.2.2 Configuring MagnetoDB Async Task Executor

Along with MagnetoDB package comes MagnetoDB Async Task Executor which is required for the service to operate properly if MagnetoDB configured to use QueuedStorageManager (default). Usually only one instance of MagnetoDB Async Task Executor is required per MagnetoDB cluster.

It is started using the following command:

```
$ magnetodb-async-task-executor --config-file /etc/magnetodb/magnetodb-async-task-executor.conf
```

It is configured via configuration file *etc/magnetodb/magnetodb-async-task-executor.conf*. It's structure mostly coincides with the structure of *magnetodb-api.conf*.

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*

/health_check

/v1