Apache Cassandra : Audit Logs & Metrics

Audit Logging

Audit Logging

Audit Logging is a new feature in Apache Cassandra 4.0 (CASSANDRA-12151). This new feature is safe for production use, with configurable limits to heap memory and disk space to prevent out-of-memory errors. All database activity is logged per-node as file-based records to a specified local filesystem directory. The audit log files are rolled periodically based on a configurable value.

Some of the features of audit logging are:

- No additional database capacity is needed to store audit logs.
- No query tool is required to store the audit logs.
- Latency of database operations is not affected, so there is no performance impact.
- Heap memory usage is bounded by a weighted queue, with configurable maximum weight sitting in front of logging thread.
- Disk utilization is bounded by a configurable size, deleting old log segments once the limit is reached.
- Can be enabled or disabled at startup time using `cassandra.yaml` or at runtime using the JMX tool, `nodetool`.
- Can configure the settings in either the `cassandra.yaml` file or by using `nodetool`.

Audit logging includes all CQL requests, both successful and failed. It also captures all successful and failed authentication and authorization events, such as login attempts. The difference between Full Query Logging (FQL) and audit logging is that FQL captures only successful CQL requests, which allow replay or comparison of logs. Audit logs are useful for compliance and debugging, while FQL is useful for debugging, performance benchmarking, testing and auditing CQL queries.

Audit information logged

The audit log contains:

- all events in the configured keyspaces to include
- all events in the configured categories to include
- all events executed by the configured users to include

The audit log does not contain:

- configuration changes made in `cassandra.yaml` file
- `nodetool` commands

- Passwords mentioned as part of DCL statements: Passwords will be obfuscated as *******.
    - Statements that fail to parse will have everything after the appearance of the word password obfuscated as *******.
    - Statements with a mistyped word 'password' will be logged without obfuscation. Please make sure to use a different password on retries.

The audit log is a series of log entries. An audit log entry contains:

- keyspace (String) - Keyspace on which request is made
- operation (String) - Database operation such as CQL command
- user (String) - User name
- scope (String) - Scope of request such as Table/Function/Aggregate name
- type (AuditLogEntryType) - Type of request
    - CQL Audit Log Entry Type
    - Common Audit Log Entry Type
- source (InetAddressAndPort) - Source IP Address from which request originated
- timestamp (long) - Timestamp of the request
- batch (UUID) - Batch of request
- options (QueryOptions) - CQL Query options
- state (QueryState) - State related to a given query

Each entry contains all applicable attributes for the given event, concatenated with a pipe (|).

CQL audit log entry types are the following CQL commands. Each command is assigned to a particular specified category to log:

Category	CQL commands
DDL	ALTER_KEYSPACE, CREATE_KEYSPACE, DROP_KEYSPACE, ALTER_TABLE, CREATE_TABLE, DROP_TABLE, CREATE_FUNCTION, DROP_FUNCTION, CREATE_AGGREGATE, DROP_AGGREGATE, CREATE_INDEX, DROP_INDEX, ALTER_TYPE, CREATE_TYPE, DROP_TYPE, CREATE_TRIGGER, DROP_TRIGGER, ALTER_VIEW, CREATE_VIEW, DROP_VIEW, TRUNCATE
DML	BATCH, DELETE, UPDATE
DCL	GRANT, REVOKE, ALTER_ROLE, CREATE_ROLE, DROP_ROLE, LIST_ROLES, LIST_PERMISSIONS, LIST_USERS
OTHER	USE_KEYSPACE
QUERY	SELECT
PREPARE	PREPARE_STATEMENT

Common audit log entry types are one of the following:

Category	CQL commands
AUTH	LOGIN_SUCCESS, LOGIN_ERROR, UNAUTHORIZED_ATTEMPT
ERROR	REQUEST_FAILURE

Availability and durability

NOTE	Unlike data, audit log entries are not replicated

For a given query, the corresponding audit entry is only stored on the coordinator node. For example, an `INSERT` in a keyspace with replication factor of 3 will produce an audit entry on one node, the coordinator who handled the request, and not on the two other nodes. For this reason, and depending on compliance requirements you must meet, make sure that audit logs are stored on a non-ephemeral storage.

You can achieve custom needs with the archive_command option.

Configuring audit logging in cassandra.yaml

The `cassandra.yaml` file can be used to configure and enable audit logging. Configuration and enablement may be the same or different on each node, depending on the `cassandra.yaml` file settings.

Audit logging can also be configured using `nodetool` when enabling the feature, and will override any values set in the `cassandra.yaml` file, as discussed in Enabling Audit Logging with nodetool.

Audit logs are generated on each enabled node, so logs on each node will have that node's queries. All options for audit logging can be set in the `cassandra.yaml` file under the `audit_logging_options:`.

The file includes the following options that can be uncommented for use:

```
# Audit logging - Logs every incoming CQL command request, authentication to a node. See
the docs
# on audit_logging for full details about the various configuration options.
audit_logging_options:
    enabled: false
    logger:
      - class_name: BinAuditLogger
    # audit_logs_dir:
    # included_keyspaces:
    # excluded_keyspaces: system, system_schema, system_virtual_schema
    # included_categories:
```

```
    # excluded_categories:
    # included_users:
    # excluded_users:
    # roll_cycle: HOURLY
    # block: true
    # max_queue_weight: 268435456 # 256 MiB
    # max_log_size: 17179869184 # 16 GiB
    ## archive command is "/path/to/script.sh %path" where %path is replaced with the
file being rolled:
    # archive_command:
    # max_archive_retries: 10
```

enabled

Control whether audit logging is enabled or disabled (default).

To enable audit logging set `enabled: true`.

If this option is enabled, audit logging will start when Cassandra is started. It can be disabled afterwards at runtime with nodetool.

TIP You can monitor whether audit logging is enabled with `AuditLogEnabled` attribute of the JMX MBean `org.apache.cassandra.db:type=StorageService`.

logger

The type of audit logger is set with the `logger` option. Supported values are:

- `BinAuditLogger` (default)
- `FileAuditLogger`
- `NoOpAuditLogger`

`BinAuditLogger` logs events to a file in binary format. `FileAuditLogger` uses the standard logging mechanism, `slf4j` to log events to the `audit/audit.log` file. It is a synchronous, file-based audit logger. The roll_cycle will be set in the `logback.xml` file. `NoOpAuditLogger` is a no-op implementation of the audit logger that shoudl be specified when audit logging is disabled.

For example:

```
logger:
  - class_name: FileAuditLogger
```

TIP `BinAuditLogger` make use of open source Chronicle Queue under the hood. If you consider

> using audit logging for regulatory compliance purpose, it might be wise to be somewhat familiar with this library. See archive_command and roll_cycle for an example of the implications.

audit_logs_dir

To write audit logs, an existing directory must be set in `audit_logs_dir`.

The directory must have appropriate permissions set to allow reading, writing, and executing. Logging will recursively delete the directory contents as needed. Do not place links in this directory to other sections of the filesystem. For example, `audit_logs_dir: /non_ephemeral_storage/audit/logs/hourly`.

The audit log directory can also be configured using the system property `cassandra.logdir.audit`, which by default is set to `cassandra.logdir + /audit/`.

included_keyspaces and excluded_keyspaces

Set the keyspaces to include with the `included_keyspaces` option and the keyspaces to exclude with the `excluded_keyspaces` option. By default, `system`, `system_schema` and `system_virtual_schema` are excluded, and all other keyspaces are included.

For example:

```
included_keyspaces: test, demo
excluded_keyspaces: system, system_schema, system_virtual_schema
```

included_categories and excluded_categories

The categories of database operations to include are specified with the `included_categories` option as a comma-separated list. The categories of database operations to exclude are specified with `excluded_categories` option as a comma-separated list. The supported categories for audit log are: `AUTH`, `DCL`, `DDL`, `DML`, `ERROR`, `OTHER`, `PREPARE`, and `QUERY`. By default, all supported categories are included, and no category is excluded.

```
included_categories: AUTH, ERROR, DCL
excluded_categories: DDL, DML, QUERY, PREPARE
```

included_users and excluded_users

Users to audit log are set with the `included_users` and `excluded_users` options. The `included_users` option specifies a comma-separated list of users to include explicitly. The `excluded_users` option specifies a comma-separated list of users to exclude explicitly. By default, all users are included, and no

users are excluded.

```
included_users:
excluded_users: john, mary
```

roll_cycle

The `roll_cycle` defines the frequency with which the audit log segments are rolled. Supported values are:

- `MINUTELY`
- `FIVE_MINUTELY`
- `TEN_MINUTELY`
- `TWENTY_MINUTELY`
- `HALF_HOURLY`
- `HOURLY` (default)
- `TWO_HOURLY`
- `FOUR_HOURLY`
- `SIX_HOURLY`
- `DAILY`

For example: `roll_cycle: DAILY`

WARNING	Read the following paragraph when changing `roll_cycle` on a production node.

With the `BinLogger` implementation, any attempt to modify the roll cycle on a node where audit logging was previously enabled will fail silentely due to Chronicle Queue roll cycle inference mechanism (even if you delete the `metadata.cq4t` file).

Here is an example of such an override visible in Cassandra logs:

```
INFO  [main] <DATE TIME> BinLog.java:420 - Attempting to configure bin log: Path:
/path/to/audit Roll cycle: TWO_HOURLY [...]
WARN  [main] <DATE TIME> SingleChronicleQueueBuilder.java:477 - Overriding roll cycle
from TWO_HOURLY to FIVE_MINUTE
```

In order to change `roll_cycle` on a node, you have to:

1. Stop Cassandra
2. Move or offload all audit logs somewhere else (in a safe and durable location)

3. Restart Cassandra.
4. Check Cassandra logs
5. Make sure that audit log filenames under `audit_logs_dir` correspond to the new roll cycle.

block

The `block` option specifies whether audit logging should block writing or drop log records if the audit logging falls behind. Supported boolean values are `true` (default) or `false`.

For example: `block: false` to drop records (e.g. if audit is used for troobleshooting)

For regulatory compliance purposes, it's a good practice to explicitly set `block: true` to prevent any regression in case of future default value change.

max_queue_weight

The `max_queue_weight` option sets the maximum weight of in-memory queue for records waiting to be written to the file before blocking or dropping. The option must be set to a positive value. The default value is 268435456, or 256 MiB.

For example, to change the default: `max_queue_weight: 134217728 # 128 MiB`

max_log_size

The `max_log_size` option sets the maximum size of the rolled files to retain on disk before deleting the oldest file. The option must be set to a positive value. The default is 17179869184, or 16 GiB. For example, to change the default: `max_log_size: 34359738368 # 32 GiB`

WARNING	`max_log_size` is ignored if `archive_command` option is set.

archive_command

NOTE	If `archive_command` option is empty or unset (default), Cassandra uses a built-in DeletingArchiver that deletes the oldest files if `max_log_size` is reached.

The `archive_command` option sets the user-defined archive script to execute on rolled log files. For example: `archive_command: "/usr/local/bin/archiveit.sh %path"`

`%path` is replaced with the absolute file path of the file being rolled.

When using a user-defined script, Cassandra does **not** use the DeletingArchiver, so it's the responsibility of the script to make any required cleanup.

Cassandra will call the user-defined script as soon as the log file is rolled. It means that Chronicle Queue's QueueFileShrinkManager will not be able to shrink the sparse log file because it's done asynchronously. In other words, all log files will have at least the size of the default block size (80 MiB), even if there are only a few KB of real data. Consequently, some warnings will appear in Cassandra system.log:

```
WARN  [main/queue~file~shrink~daemon] <DATE TIME> QueueFileShrinkManager.java:63 - Failed
to shrink file as it exists no longer, file=/path/to/xxx.cq4
```

TIP

Because Cassandra does not make use of Pretoucher, you can configure Chronicle Queue to shrink files synchronously — i.e. as soon as the file is rolled — with `chronicle.queue.synchronousFileShrinking` JVM properties. For instance, you can add the following line at the end of `cassandra-env.sh`: `JVM_OPTS="$JVM_OPTS -Dchronicle.queue.synchronousFileShrinking=true"`

max_archive_retries

The `max_archive_retries` option sets the max number of retries of failed archive commands. The default is 10.

For example: `max_archive_retries: 10`

Interval between each retry is hard coded to 5 minutes.

Enabling Audit Logging with `nodetool`

Audit logging is enabled on a per-node basis using the `nodetool enableauditlog` command. The logging directory must be defined with `audit_logs_dir` in the `cassandra.yaml` file or uses the default value `cassandra.logdir.audit`.

The syntax of the `nodetool enableauditlog` command has all the same options that can be set in the `cassandra.yaml` file except `audit_logs_dir`. In addition, `nodetool` has options to set which host and port to run the command on, and username and password if the command requires authentication.

```
    nodetool [(-h <host> | --host <host>)] [(-p <port> | --port <port>)]
             [(-pp | --print-port)] [(-pw <password> | --password <password>)]
             [(-pwf <passwordFilePath> | --password-file <passwordFilePath>)]
             [(-u <username> | --username <username>)] enableauditlog
             [--excluded-categories <excluded_categories>]
             [--excluded-keyspaces <excluded_keyspaces>]
             [--excluded-users <excluded_users>]
             [--included-categories <included_categories>]
             [--included-keyspaces <included_keyspaces>]
```

```
                [--included-users <included_users>] [--logger <logger>]

OPTIONS
        --excluded-categories <excluded_categories>
            Comma separated list of Audit Log Categories to be excluded for
            audit log. If not set the value from cassandra.yaml will be used

        --excluded-keyspaces <excluded_keyspaces>
            Comma separated list of keyspaces to be excluded for audit log. If
            not set the value from cassandra.yaml will be used

        --excluded-users <excluded_users>
            Comma separated list of users to be excluded for audit log. If not
            set the value from cassandra.yaml will be used

        -h <host>, --host <host>
            Node hostname or ip address

        --included-categories <included_categories>
            Comma separated list of Audit Log Categories to be included for
            audit log. If not set the value from cassandra.yaml will be used

        --included-keyspaces <included_keyspaces>
            Comma separated list of keyspaces to be included for audit log. If
            not set the value from cassandra.yaml will be used

        --included-users <included_users>
            Comma separated list of users to be included for audit log. If not
            set the value from cassandra.yaml will be used

        --logger <logger>
            Logger name to be used for AuditLogging. Default BinAuditLogger. If
            not set the value from cassandra.yaml will be used

        -p <port>, --port <port>
            Remote jmx agent port number

        -pp, --print-port
            Operate in 4.0 mode with hosts disambiguated by port number

        -pw <password>, --password <password>
            Remote jmx agent password

        -pwf <passwordFilePath>, --password-file <passwordFilePath>
            Path to the JMX password file

        -u <username>, --username <username>
```

```
            Remote jmx agent username
```

To enable audit logging, run following command on each node in the cluster on which you want to enable logging:

```
$ nodetool enableauditlog
```

Disabling audit logging

Use the `nodetool disableauditlog` command to disable audit logging.

Viewing audit logs

The `auditlogviewer` tool is used to view (dump) audit logs if the logger was `BinAuditLogger`.. `auditlogviewer` converts the binary log files into human-readable format; only the audit log directory must be supplied as a command-line option. If the logger `FileAuditLogger` was set, the log file are already in human-readable format and `auditlogviewer` is not needed to read files.

The syntax of `auditlogviewer` is:

```
auditlogviewer

Audit log files directory path is a required argument.
usage: auditlogviewer <path1> [<path2>...<pathN>] [options]
--
View the audit log contents in human readable format
--
Options are:
-f,--follow       Upon reaching the end of the log continue indefinitely
                  waiting for more records
-h,--help         display this help message
-r,--roll_cycle   How often to roll the log file was rolled. May be
                  necessary for Chronicle to correctly parse file names. (MINUTELY,
HOURLY,
                  DAILY). Default HOURLY.
```

Example

1. To demonstrate audit logging, first configure the `cassandra.yaml` file with the following settings:

```
audit_logging_options:
```

```
   enabled: true
   logger: BinAuditLogger
   audit_logs_dir: "/cassandra/audit/logs/hourly"
   # included_keyspaces:
   # excluded_keyspaces: system, system_schema, system_virtual_schema
   # included_categories:
   # excluded_categories:
   # included_users:
   # excluded_users:
   roll_cycle: HOURLY
   # block: true
   # max_queue_weight: 268435456 # 256 MiB
   # max_log_size: 17179869184 # 16 GiB
   ## archive command is "/path/to/script.sh %path" where %path is replaced with the file
being rolled:
   # archive_command:
   # max_archive_retries: 10
```

2. Create the audit log directory `/cassandra/audit/logs/hourly` and set the directory permissions to read, write, and execute for all.

3. Now create a demo keyspace and table and insert some data using `cqlsh`:

```
cqlsh> CREATE KEYSPACE auditlogkeyspace
  ... WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 1};
cqlsh> USE auditlogkeyspace;
cqlsh:auditlogkeyspace> CREATE TABLE t (
...id int,
...k int,
...v text,
...PRIMARY KEY (id)
... );
cqlsh:auditlogkeyspace> INSERT INTO t (id, k, v) VALUES (0, 0, 'val0');
cqlsh:auditlogkeyspace> INSERT INTO t (id, k, v) VALUES (0, 1, 'val1');
```

All the supported CQL commands will be logged to the audit log directory.

4. Change directory to the audit logs directory.

```
$ cd /cassandra/audit/logs/hourly
```

5. List the audit log files and directories.

```
$ ls -l
```

You should see results similar to:

```
total 28
-rw-rw-r--. 1 ec2-user ec2-user    65536 Aug  2 03:01 directory-listing.cq4t
-rw-rw-r--. 1 ec2-user ec2-user 83886080 Aug  2 03:01 20190802-02.cq4
-rw-rw-r--. 1 ec2-user ec2-user 83886080 Aug  2 03:01 20190802-03.cq4
```

The audit log files will all be listed with a `.cq4` file type. The audit directory is of `.cq4t` type.

6. Run `auditlogviewer` tool to view the audit logs.

```
$ auditlogviewer /cassandra/audit/logs/hourly
```

This command will return a readable version of the log. Here is a partial sample of the log for the commands in this demo:

```
WARN  03:12:11,124 Using Pauser.sleepy() as not enough processors, have 2, needs 8+
Type: AuditLog
LogMessage:
user:anonymous|host:10.0.2.238:7000|source:/127.0.0.1|port:46264|timestamp:1564711427328|
type :USE_KEYSPACE|category:OTHER|ks:auditlogkeyspace|operation:USE AuditLogKeyspace;
Type: AuditLog
LogMessage:
user:anonymous|host:10.0.2.238:7000|source:/127.0.0.1|port:46264|timestamp:1564711427329|
type :USE_KEYSPACE|category:OTHER|ks:auditlogkeyspace|operation:USE "auditlogkeyspace"
Type: AuditLog
LogMessage:
user:anonymous|host:10.0.2.238:7000|source:/127.0.0.1|port:46264|timestamp:1564711446279|
type :SELECT|category:QUERY|ks:auditlogkeyspace|scope:t|operation:SELECT * FROM t;
Type: AuditLog
LogMessage:
user:anonymous|host:10.0.2.238:7000|source:/127.0.0.1|port:46264|timestamp:1564713878834|
type :DROP_TABLE|category:DDL|ks:auditlogkeyspace|scope:t|operation:DROP TABLE IF EXISTS
AuditLogKeyspace.t;
Type: AuditLog
LogMessage:
user:anonymous|host:10.0.2.238:7000|source:/3.91.56.164|port:42382|timestamp:156471461836
0|ty
pe:REQUEST_FAILURE|category:ERROR|operation:CREATE KEYSPACE AuditLogKeyspace
WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 1};; Cannot add
existing keyspace "auditlogkeyspace"
```

```
Type: AuditLog
LogMessage:
user:anonymous|host:10.0.2.238:7000|source:/127.0.0.1|port:46264|timestamp:1564714690968|
type :DROP_KEYSPACE|category:DDL|ks:auditlogkeyspace|operation:DROP KEYSPACE
AuditLogKeyspace;
Type: AuditLog
LogMessage:
user:anonymous|host:10.0.2.238:7000|source:/3.91.56.164|port:42406|timestamp:156471470832
9|ty pe:CREATE_KEYSPACE|category:DDL|ks:auditlogkeyspace|operation:CREATE KEYSPACE
AuditLogKeyspace
WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 1};
Type: AuditLog
LogMessage:
user:anonymous|host:10.0.2.238:7000|source:/127.0.0.1|port:46264|timestamp:1564714870678|
type :USE_KEYSPACE|category:OTHER|ks:auditlogkeyspace|operation:USE auditlogkeyspace;

Password obfuscation examples:
LogMessage:
user:cassandra|host:localhost/127.0.0.1:7000|source:/127.0.0.1|port:65282|timestamp:16226
30496708|type:CREATE_ROLE|category:DCL|operation:CREATE ROLE role1 WITH PASSWORD =
'*******';
Type: audit
LogMessage:
user:cassandra|host:localhost/127.0.0.1:7000|source:/127.0.0.1|port:65282|timestamp:16226
30634552|type:ALTER_ROLE|category:DCL|operation:ATLER ROLE role1 WITH PASSWORD =
'*******';
Type: audit
LogMessage:
user:cassandra|host:localhost/127.0.0.1:7000|source:/127.0.0.1|port:65282|timestamp:16226
30698686|type:CREATE_ROLE|category:DCL|operation:CREATE USER user1 WITH PASSWORD
'*******';
Type: audit
LogMessage:
user:cassandra|host:localhost/127.0.0.1:7000|source:/127.0.0.1|port:65282|timestamp:16226
30747344|type:ALTER_ROLE|category:DCL|operation:ALTER USER user1 WITH PASSWORD '*******';
```

Diagnostic events for user audit logging

Any native transport-enabled client can subscribe to audit log events for diagnosing cluster issues. These events can be consumed by external tools to implement a Cassandra user auditing solution.

Audit Logging

Audit Logging

Audit logging in Cassandra logs every incoming CQL command request, as well as authentication (successful/unsuccessful login) to a Cassandra node. Currently, there are two implementations provided. The custom logger can be implemented and injected with the class name as a parameter in the `cassandra.yaml` file.

- `BinAuditLogger`: an efficient way to log events to file in a binary format (community-recommended logger for performance)
- `FileAuditLogger`: logs events to `audit/audit.log` file using slf4j logger

What does audit logging captures

Audit logging captures following events:

- Successful as well as unsuccessful login attempts
- All database commands executed via native CQL protocol attempted or successfully executed

Limitations

Executing prepared statements will log the query as provided by the client in the prepare call, along with the execution timestamp and all other attributes (see below). Actual values bound for prepared statement execution will not show up in the audit log.

What does audit logging logs

Each audit log implementation has access to the following attributes, and for the default text based logger these fields are concatenated with pipes to yield the final message.

- `user`: User name(if available)
- `host`: Host IP, where the command is being executed
- `source ip address`: Source IP address from where the request initiated
- `source port`: Source port number from where the request initiated
- `timestamp`: unix time stamp
- `type`: Type of the request (SELECT, INSERT, etc.,)
- `category` - Category of the request (DDL, DML, etc.,)
- `keyspace` - Keyspace(If applicable) on which request is targeted to be executed

- `scope` - Table/Aggregate name/ function name/ trigger name etc., as applicable
- `operation` - CQL command being executed

How to configure

Auditlog can be configured using the `cassandra.yaml` file. To use audit logging on one node, either edit that file or enable and configure using `nodetool`.

cassandra.yaml configurations for AuditLog

The following options are supported:

- `enabled`: This option enables/ disables audit log
- `logger`: Class name of the logger/ custom logger.
- `audit_logs_dir`: Auditlogs directory location, if not set, default to cassandra.logdir.audit or cassandra.logdir
  /audit/
- `included_keyspaces`: Comma separated list of keyspaces to be included in audit log, default - includes all keyspaces
- `excluded_keyspaces`: Comma separated list of keyspaces to be excluded from audit log, default - excludes no keyspace except system, system_schema and system_virtual_schema
- `included_categories`: Comma separated list of Audit Log Categories to be included in audit log, default - includes all categories
- `excluded_categories`: Comma separated list of Audit Log Categories to be excluded from audit log, default - excludes no category
- `included_users`: Comma separated list of users to be included in audit log, default - includes all users
- `excluded_users`: Comma separated list of users to be excluded from audit log, default - excludes no user

List of available categories are: QUERY, DML, DDL, DCL, OTHER, AUTH, ERROR, PREPARE

NodeTool command to enable AuditLog

The `nodetool enableauditlog` command enables AuditLog with the `cassandra.yaml` file defaults. Those defaults can be overridden using options with this nodetool command.

```
nodetool enableauditlog
```

Options

`--excluded-categories`

Comma separated list of Audit Log Categories to be excluded for audit log. If not set the value from cassandra.yaml will be used

`--excluded-keyspaces`

Comma separated list of keyspaces to be excluded for audit log. If not set the value from cassandra.yaml will be used. Please remeber that system, system_schema and system_virtual_schema are excluded by default, if you are overwriting this option via nodetool, remember to add these keyspaces back if you dont want them in audit logs

`--excluded-users`

Comma separated list of users to be excluded for audit log. If not set the value from cassandra.yaml will be used

`--included-categories`

Comma separated list of Audit Log Categories to be included for audit log. If not set the value from cassandra.yaml will be used

`--included-keyspaces`

Comma separated list of keyspaces to be included for audit log. If not set the value from cassandra.yaml will be used

`--included-users`

Comma separated list of users to be included for audit log. If not set the value from cassandra.yaml will be used

`--logger`

Logger name to be used for AuditLogging. Default BinAuditLogger. If not set the value from cassandra.yaml will be used

NodeTool command to disable AuditLog

The `nodetool disableauditlog` command disables AuditLog.

```
nodetool disableuditlog
```

NodeTool command to reload AuditLog filters

The `nodetool enableauditlog` command can be used to reload auditlog filters with either defaults or previous `loggername` and updated filters:

```
nodetool enableauditlog --loggername <Default/ existing loggerName> --included-keyspaces
<New Filter values>
```

View the contents of AuditLog Files

The `auditlogviewer` is used to view the contents of the audit binlog file in human readable text format.

```
auditlogviewer <path1> [<path2>...<pathN>] [options]
```

Options

`-f,--follow`

Upon reacahing the end of the log continue indefinitely

waiting for more records

`-r,--roll_cycle`

How often to roll the log file was rolled. May be

necessary for Chronicle to correctly parse file names. (MINUTELY, HOURLY, DAILY). Default HOURLY.

`-h,--help`

display this help message

For example, to dump the contents of audit log files to the console:

```
auditlogviewer /logs/cassandra/audit
```

results in

```
LogMessage:
user:anonymous|host:localhost/X.X.X.X|source:/X.X.X.X|port:60878|timestamp:1521158923615|
type:USE_KS|category:DDL|ks:dev1|operation:USE "dev1"
```

Configuring BinAuditLogger

To use `BinAuditLogger` as a logger in AuditLogging, set the logger to `BinAuditLogger` in the `cassandra.yaml` file under the `audit_logging_options` section. `BinAuditLogger` can be futher configued using its advanced options in `cassandra.yaml`.

Advanced Options for BinAuditLogger

`block`

Indicates if the AuditLog should block if the it falls behind or should drop audit log records. Default is set to `true` so that AuditLog records wont be lost

`max_queue_weight`

Maximum weight of in memory queue for records waiting to be written to the audit log file before blocking or dropping the log records. Default is set to `256 * 1024 * 1024`

`max_log_size`

Maximum size of the rolled files to retain on disk before deleting the oldest file. Default is set to `16L * 1024L * 1024L * 1024L`

`roll_cycle`

How often to roll Audit log segments so they can potentially be reclaimed. Available options are: MINUTELY, HOURLY, DAILY, LARGE_DAILY, XLARGE_DAILY, HUGE_DAILY.For more options, refer: net.openhft.chronicle.queue.RollCycles. Default is set to `"HOURLY"`

Configuring FileAuditLogger

To use `FileAuditLogger` as a logger in AuditLogging, set the class name in the `cassandra.yaml` file and configure the audit log events to flow through separate log file instead of system.log.

```
<!-- Audit Logging (FileAuditLogger) rolling file appender to audit.log -->
<appender name="AUDIT" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${cassandra.logdir}/audit/audit.log</file>
  <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
    <!-- rollover daily -->
    <fileNamePattern>${cassandra.logdir}/audit/audit.log.%d{yyyy-MM-
dd}.%i.zip</fileNamePattern>
    <!-- each file should be at most 50MB, keep 30 days worth of history, but at most 5GB
-->
    <maxFileSize>50MB</maxFileSize>
    <maxHistory>30</maxHistory>
    <totalSizeCap>5GB</totalSizeCap>
  </rollingPolicy>
  <encoder>
    <pattern>%-5level [%thread] %date{ISO8601} %F:%L - %msg%n</pattern>
  </encoder>
</appender>

<!-- Audit Logging additivity to redirect audt logging events to audit/audit.log -->
<logger name="org.apache.cassandra.audit" additivity="false" level="INFO">
```

```
    <appender-ref ref="AUDIT"/>
</logger>
```

Full Query Logging

Full Query Logging

Apache Cassandra 4.0 adds a new highly performant feature that supports live query logging (CASSANDRA-13983). FQL is safe for production use, with configurable limits to heap memory and disk space to prevent out-of-memory errors. This feature is useful for live traffic capture, as well as traffic replay. The tool provided can be used for both debugging query traffic and migration. New `nodetool` options are also added to enable, disable or reset FQL, as well as a new tool to read and replay the binary logs. The full query logging (FQL) capability uses Chronicle-Queue to rotate a log of queries. Full query logs will be referred to as **logs** for the remainder of the page.

Some of the features of FQL are:

- The impact on query latency is reduced by asynchronous single-thread log entry writes to disk.
- Heap memory usage is bounded by a weighted queue, with configurable maximum weight sitting in front of logging thread.
- If the weighted queue is full, producers can be blocked or samples can be dropped.
- Disk utilization is bounded by a configurable size, deleting old log segments once the limit is reached.
- A flexible schema binary format, Chronicle-Wire, for on-disk serialization that can skip unrecognized fields, add new ones, and omit old ones.
- Can be enabled, disabled, or reset (to delete on-disk data) using the JMX tool, `nodetool`.
- Can configure the settings in either the `cassandra.yaml` file or by using `nodetool`.
- Introduces new `fqltool` that currently can `Dump` the binary logs to a readable format. Other options are `Replay` and `Compare`.

FQL logs all successful Cassandra Query Language (CQL) requests, both events that modify the data and those that query. While audit logs also include CQL requests, FQL logs only the CQL request. This difference means that FQL can be used to replay or compare logs, which audit logging cannot. FQL is useful for debugging, performance benchmarking, testing and auditing CQL queries, while audit logs are useful for compliance.

In performance testing, FQL appears to have little or no overhead in `WRITE` only workloads, and a minor overhead in `MIXED` workload.

Query information logged

The query log contains:

- all queries invoked
- approximate time they were invoked
- any parameters necessary to bind wildcard values
- all query options

The logger writes single or batched CQL queries after they finish, so only successfully completed queries are logged. Failed or timed-out queries are not logged. Different data is logged, depending on the type of query.

A single CQL query log entry contains:

- query - CQL query text
- queryOptions - Options associated with the query invocation
- queryState - Timestamp state associated with the query invocation
- queryTimeMillis - Approximate time in milliseconds since the epoch since the query was invoked

A batch CQL query log entry contains:

- queries - CQL text of the queries
- queryOptions - Options associated with the query invocation
- queryState - Timestamp state associated with the query invocation
- batchTimeMillis - Approximate time in milliseconds since the epoch since the batch was invoked
- type - The type of the batch
- values - Values to bind to as parameters for the queries

Because FQL is backed by `Binlog`, the performance and footprint are predictable, with minimal impact on log record producers. Performance safety prevents the producers from overloading the log, using a weighted queue to drop records if the logging falls behind. Single-thread asynchronous writing produces the logs. Chronicle-Queue provides an easy method of rolling the logs.

Logging information logged

FQL also tracks information about the stored log files:

- Stored log files that are added and their storage impact. Deletes them if over storage limit.
- The log files in Chronicle-Queue that have already rolled
- The number of bytes in the log files that have already rolled

Logging sequence

The logger follows a well-defined sequence of events:

1. The consumer thread that writes log records is started. This action can occur only once.
2. The consumer thread offers a record to the log. If the in-memory queue is full, the record will be dropped and offer returns a `false` value.
3. If accepted, the record is entered into the log. If the in-memory queue is full, the putting thread will be blocked until there is space or it is interrupted.
4. The buffers are cleaned up at thread exit. Finalization will check again, to ensure there are no stragglers in the queue.
5. The consumer thread is stopped. It can be called multiple times.

Using FQL

To use FQL, two actions must be completed. FQL must be configured using either the `cassandra.yaml` file or `nodetool`, and logging must be enabled using `nodetool enablefullquerylog`. With either method, at a minimum, the path to the log directory must be specified. Both actions are completed on a per-node basis. Full query logs are generated on each enabled node, so logs on each node will have that node's queries.

Configuring FQL in cassandra.yaml

The `cassandra.yaml` file can be used to configure FQL before enabling the feature with `nodetool`.

The file includes the following options that can be uncommented for use:

```
# default options for full query logging - these can be overridden from command line
# when executing nodetool enablefullquerylog
#full_query_logging_options:
   # log_dir:
   # roll_cycle: HOURLY
   # block: true
   # max_queue_weight: 268435456 # 256 MiB
   # max_log_size: 17179869184 # 16 GiB
   # archive command is "/path/to/script.sh %path" where %path is replaced with the file
being rolled:
   # archive_command:
   # max_archive_retries: 10
```

log_dir

To write logs, an existing directory must be set in `log_dir`.

The directory must have appropriate permissions set to allow reading, writing, and executing. Logging will recursively delete the directory contents as needed. Do not place links in this directory to other sections of the filesystem. For example, `log_dir: /tmp/cassandrafullquerylog`.

roll_cycle

The `roll_cycle` defines the frequency with which the log segments are rolled. Supported values are `HOURLY` (default), `MINUTELY`, and `DAILY`. For example: `roll_cycle: DAILY`

block

The `block` option specifies whether FQL should block writing or drop log records if FQL falls behind. Supported boolean values are `true` (default) or `false`. For example: `block: false` to drop records

max_queue_weight

The `max_queue_weight` option sets the maximum weight of in-memory queue for records waiting to be written to the file before blocking or dropping. The option must be set to a positive value. The default value is 268435456, or 256 MiB. For example, to change the default: `max_queue_weight: 134217728 # 128 MiB`

max_log_size

The `max_log_size` option sets the maximum size of the rolled files to retain on disk before deleting the oldest file. The option must be set to a positive value. The default is 17179869184, or 16 GiB. For example, to change the default: `max_log_size: 34359738368 # 32 GiB`

archive_command

The `archive_command` option sets the user-defined archive script to execute on rolled log files. When not defined, files are deleted, with the default `""` which then maps to `org.apache.cassandra.utils.binlog.DeletingArchiver`. For example: `archive_command: /usr/local/bin/archiveit.sh %path # %path is the file being rolled`

max_archive_retries

The `max_archive_retries` option sets the max number of retries of failed archive commands. The default is 10. For example: `max_archive_retries: 10`

FQL can also be configured using `nodetool` when enabling the feature, and will override any values set in the `cassandra.yaml` file, as discussed in the next section.

Enabling FQL

FQL is enabled on a per-node basis using the `nodetool enablefullquerylog` command. At a minimum, the path to the logging directory must be defined, if `log_dir` is not set in the `cassandra.yaml` file.

The syntax of the `nodetool enablefullquerylog` command has all the same options that can be set in the `cassandra.yaml` file. In addition, `nodetool` has options to set which host and port to run the command on, and username and password if the command requires authentication.

```
 nodetool [(-h <host> | --host <host>)] [(-p <port> | --port <port>)]
[(-pp | --print-port)] [(-pw <password> | --password <password>)]
[(-pwf <passwordFilePath> | --password-file <passwordFilePath>)]
[(-u <username> | --username <username>)] enablefullquerylog
[--archive-command <archive_command>] [--blocking]
[--max-archive-retries <archive_retries>]
[--max-log-size <max_log_size>] [--max-queue-weight <max_queue_weight>]
[--path <path>] [--roll-cycle <roll_cycle>]

OPTIONS
  --archive-command <archive_command>
 Command that will handle archiving rolled full query log files.
 Format is "/path/to/script.sh %path" where %path will be replaced
 with the file to archive

  --blocking
 If the queue is full whether to block producers or drop samples.

  -h <host>, --host <host>
 Node hostname or ip address

  --max-archive-retries <archive_retries>
 Max number of archive retries.

  --max-log-size <max_log_size>
 How many bytes of log data to store before dropping segments. Might
 not be respected if a log file hasn't rolled so it can be deleted.

  --max-queue-weight <max_queue_weight>
 Maximum number of bytes of query data to queue to disk before
 blocking or dropping samples.

  -p <port>, --port <port>
 Remote jmx agent port number
```

```
 --path <path>
Path to store the full query log at. Will have it's contents
recursively deleted.

 -pp, --print-port
Operate in 4.0 mode with hosts disambiguated by port number

 -pw <password>, --password <password>
Remote jmx agent password

 -pwf <passwordFilePath>, --password-file <passwordFilePath>
Path to the JMX password file

 --roll-cycle <roll_cycle>
How often to roll the log file (MINUTELY, HOURLY, DAILY).

 -u <username>, --username <username>
Remote jmx agent username
```

To enable FQL, run the following command on each node in the cluster on which you want to enable logging:

```
$ nodetool enablefullquerylog --path /tmp/cassandrafullquerylog
```

Disabling or resetting FQL

Use the `nodetool disablefullquerylog` to disable logging. Use `nodetool resetfullquerylog` to stop FQL and clear the log files in the configured directory. **IMPORTANT:** Using `nodetool resetfullquerylog` will delete the log files! Do not use this command unless you need to delete all log files.

fqltool

The `fqltool` command is used to view (dump), replay, or compare logs. `fqltool dump` converts the binary log files into human-readable format; only the log directory must be supplied as a command-line option.

`fqltool replay` (CASSANDRA-14618) enables replay of logs. The command can run from a different machine or cluster for testing, debugging, or performance benchmarking. The command can also be used to recreate a dropped database object. Use `fqltool replay` to record and compare different runs of production traffic against different versions/configurations of Cassandra or different clusters. Another use is to gather logs from several machines and replay them in “order” by the timestamps recorded.

The syntax of `fqltool replay` is:

```
 fqltool replay [--keyspace <keyspace>] [--results <results>]
[--store-queries <store_queries>] --target <target>... [--] <path1>
[<path2>...<pathN>]

OPTIONS
  --keyspace <keyspace>
 Only replay queries against this keyspace and queries without
 keyspace set.

  --results <results>
 Where to store the results of the queries, this should be a
 directory. Leave this option out to avoid storing results.

  --store-queries <store_queries>
 Path to store the queries executed. Stores queries in the same order
 as the result sets are in the result files. Requires --results

  --target <target>
 Hosts to replay the logs to, can be repeated to replay to more
 hosts.

  --
 This option can be used to separate command-line options from the
 list of argument, (useful when arguments might be mistaken for
 command-line options

  <path1> [<path2>...<pathN>]
 Paths containing the FQ logs to replay.
```

`fqltool compare` (CASSANDRA-14619) compares result files generated by `fqltool replay`. The command uses recorded runs from `fqltool replay` and compareslog, outputting any differences (potentially all queries). It also stores each row as a separate chronicle document to avoid reading the entire result from in-memory when comparing.

The syntax of `fqltool compare` is:

```
  fqltool compare --queries <queries> [--] <path1> [<path2>...<pathN>]

OPTIONS
  --queries <queries>
 Directory to read the queries from. It is produced by the fqltool
 replay --store-queries option.

  --
```

```
  This option can be used to separate command-line options from the
  list of argument, (useful when arguments might be mistaken for
  command-line options

   <path1> [<path2>...<pathN>]
  Directories containing result files to compare.
```

The comparison sets the following marks:

- Mark the beginning of a query set:

```
version: int16
type: column_definitions
column_count: int32;
column_definition: text, text
column_definition: text, text
....
```

- Mark a failed query set:

```
version: int16
type: query_failed
message: text
```

- Mark a row set:

```
version: int16
type: row
row_column_count: int32
column: bytes
```

- Mark the end of a result set:

```
version: int16
type: end_resultset
```

Example

1. To demonstrate FQL, first configure and enable FQL on a node in your cluster:

```
$ nodetool enablefullquerylog --path /tmp/cassandrafullquerylog
```

2. Now create a demo keyspace and table and insert some data using `cqlsh`:

```
cqlsh> CREATE KEYSPACE querylogkeyspace
  ... WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 1};
cqlsh> USE querylogkeyspace;
cqlsh:querylogkeyspace> CREATE TABLE t (
...id int,
...k int,
...v text,
...PRIMARY KEY (id)
... );
cqlsh:querylogkeyspace> INSERT INTO t (id, k, v) VALUES (0, 0, 'val0');
cqlsh:querylogkeyspace> INSERT INTO t (id, k, v) VALUES (0, 1, 'val1');
```

3. Then check that the data is inserted:

```
cqlsh:querylogkeyspace> SELECT * FROM t;

id | k | v
----+---+------
 0 | 1 | val1

(1 rows)
```

4. Use the `fqltool dump` command to view the logs.

```
$ fqltool dump /tmp/cassandrafullquerylog
```

This command will return a readable version of the log. Here is a partial sample of the log for the commands in this demo:

```
WARN  [main] 2019-08-02 03:07:53,635 Slf4jExceptionHandler.java:42 - Using
Pauser.sleepy() as not enough processors, have 2, needs 8+
      Type: single-query
      Query start time: 1564708322030
      Protocol version: 4
      Generated timestamp:-9223372036854775808
      Generated nowInSeconds:1564708322
      Query: SELECT * FROM system.peers
      Values:
```

```
Type: single-query
Query start time: 1564708322054
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:1564708322
Query: SELECT * FROM system.local WHERE key='local'
Values:

Type: single-query
Query start time: 1564708322109
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:1564708322
Query: SELECT * FROM system_schema.keyspaces
Values:

Type: single-query
Query start time: 1564708322116
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:1564708322
Query: SELECT * FROM system_schema.tables
Values:

Type: single-query
Query start time: 1564708322139
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:1564708322
Query: SELECT * FROM system_schema.columns
Values:

Type: single-query
Query start time: 1564708322142
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:1564708322
Query: SELECT * FROM system_schema.functions
Values:

Type: single-query
Query start time: 1564708322141
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:1564708322
Query: SELECT * FROM system_schema.aggregates
Values:
```

```
Type: single-query
Query start time: 1564708322143
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:1564708322
Query: SELECT * FROM system_schema.types
Values:

Type: single-query
Query start time: 1564708322144
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:1564708322
Query: SELECT * FROM system_schema.indexes
Values:

Type: single-query
Query start time: 1564708322145
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:1564708322
Query: SELECT * FROM system_schema.views
Values:

Type: single-query
Query start time: 1564708345408
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:-2147483648
Query: CREATE KEYSPACE querylogkeyspace
WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 1};
Values:

Type: single-query
Query start time: 1564708360873
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:-2147483648
Query: USE querylogkeyspace;
Values:

Type: single-query
Query start time: 1564708360874
Protocol version: 4
Generated timestamp:-9223372036854775808
Generated nowInSeconds:-2147483648
Query: USE "querylogkeyspace"
```

```
      Values:

      Type: single-query
      Query start time: 1564708378837
      Protocol version: 4
      Generated timestamp:-9223372036854775808
      Generated nowInSeconds:-2147483648
      Query: CREATE TABLE t (
          id int,
          k int,
          v text,
          PRIMARY KEY (id)
      );
      Values:

      Type: single-query
      Query start time: 1564708379247
      Protocol version: 4
      Generated timestamp:-9223372036854775808
      Generated nowInSeconds:1564708379
      Query: SELECT * FROM system_schema.tables WHERE keyspace_name = 'querylogkeyspace'
AND table_name = 't'
      Values:

      Type: single-query
      Query start time: 1564708397144
      Protocol version: 4
      Generated timestamp:-9223372036854775808
      Generated nowInSeconds:1564708397
      Query: INSERT INTO t (id, k, v) VALUES (0, 0, 'val0');
      Values:

      Type: single-query
      Query start time: 1564708434782
      Protocol version: 4
      Generated timestamp:-9223372036854775808
      Generated nowInSeconds:1564708434
      Query: SELECT * FROM t;
      Values:
```

5. To demonstrate `fqltool replay`, first drop the keyspace.

```
cqlsh:querylogkeyspace> DROP KEYSPACE querylogkeyspace;
```

6. Now run `fqltool replay` specifying the directories in which to store the results of the queries and the list of queries run, respectively, in `--results` and `--store-queries`:

```
$ fqltool replay \
--keyspace querylogkeyspace --results /cassandra/fql/logs/results/replay \
--store-queries /cassandra/fql/logs/queries/replay \
-- target 3.91.56.164 \
/tmp/cassandrafullquerylog
```

The `--results` and `--store-queries` directories are optional, but if `--store-queries` is set, then `--results` must also be set. The `--target` specifies the node on which to replay to logs.

7. Check that the keyspace was replayed and exists again using the `DESCRIBE KEYSPACES` command:

```
cqlsh:querylogkeyspace> DESC KEYSPACES;

system_schema  system  system_distributed  system_virtual_schema
system_auth    querylogkeyspace  system_traces  system_views
```

Monitoring

Monitoring

Metrics in Cassandra are managed using the Dropwizard Metrics library. Metrics can be queried via JMX, **Virtual Tables**, or pushed to external monitoring systems using a variety of built-in reporters or third-party reporter plug-ins.

Metrics are collected for a single node. It's up to the operator to use an external monitoring system to aggregate them.

Metric Types

All metrics reported by cassandra fit into one of the following types.

`Gauge`

An instantaneous measurement of a value.

`Counter`

A gauge for an `AtomicLong` instance. Typically this is consumed by monitoring the change since the last call to see if there is a large increase compared to the norm.

`Histogram`

Measures the statistical distribution of values in a stream of data. + In addition to minimum, maximum, mean, etc., it also measures median, 75th, 90th, 95th, 98th, 99th, and 99.9th percentiles.

`Timer`

Measures both the rate that a particular piece of code is called and the histogram of its duration.

`Latency`

Special type that tracks latency (in microseconds) with a `Timer` plus a `Counter` that tracks the total latency accrued since starting. The former is useful if you track the change in total latency since the last check. Each metric name of this type will have 'Latency' and 'TotalLatency' appended to it.

`Meter`

A meter metric which measures mean throughput and one-, five-, and fifteen-minute exponentially-weighted moving average throughputs.

Table Metrics

Each table in Cassandra has metrics responsible for tracking its state and performance.

The metric names are all appended with the specific `Keyspace` and `Table` name.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.Table.<MetricName>.<Keyspace>.<Table>`

JMX MBean

`org.apache.cassandra.metrics:type=Table keyspace=<Keyspace> scope=<Table> name=<MetricName>`

NOTE	There is a special table called 'all' without a keyspace. This represents the aggregation of metrics across **all** tables and keyspaces on the node.

Name	Type	Description
MemtableOnHeapSize	Gauge<Long>	Total amount of data stored in the memtable that resides **on** -heap, including column related overhead and partitions overwritten.
MemtableOffHeapSize	Gauge<Long>	Total amount of data stored in the memtable that resides **off** -heap, including column related overhead and partitions overwritten.
MemtableLiveDataSize	Gauge<Long>	Total amount of live data stored in the memtable, excluding any data structure overhead.
AllMemtablesOnHeapSize	Gauge<Long>	Total amount of data stored in the memtables (2i and pending flush memtables included) that resides **on**-heap.
AllMemtablesOffHeapSize	Gauge<Long>	Total amount of data stored in the memtables (2i and pending flush memtables included) that resides **off**-heap.
AllMemtablesLiveDataSize	Gauge<Long>	Total amount of live data stored in the memtables (2i and pending flush memtables included) that resides off-heap, excluding any data structure overhead.
MemtableColumnsCount	Gauge<Long>	Total number of columns present in the memtable.

Name	Type	Description
MemtableSwitchCount	Counter	Number of times flush has resulted in the memtable being switched out.
CompressionRatio	Gauge<Double>	Current compression ratio for all SSTables.
EstimatedPartitionSizeHistogram	Gauge<long[]>	Histogram of estimated partition size (in bytes).
EstimatedPartitionCount	Gauge<Long>	Approximate number of keys in table.
EstimatedColumnCountHistogram	Gauge<long[]>	Histogram of estimated number of columns.
SSTablesPerReadHistogram	Histogram	Histogram of the number of sstable data files accessed per single partition read. SSTables skipped due to Bloom Filters, min-max key or partition index lookup are not taken into acoount.
ReadLatency	Latency	Local read latency for this table.
RangeLatency	Latency	Local range scan latency for this table.
WriteLatency	Latency	Local write latency for this table.
CoordinatorReadLatency	Timer	Coordinator read latency for this table.
CoordinatorWriteLatency	Timer	Coordinator write latency for this table.
CoordinatorScanLatency	Timer	Coordinator range scan latency for this table.
PendingFlushes	Counter	Estimated number of flush tasks pending for this table.
BytesFlushed	Counter	Total number of bytes flushed since server [re]start.
CompactionBytesWritten	Counter	Total number of bytes written by compaction since server [re]start.
PendingCompactions	Gauge<Integer>	Estimate of number of pending compactions for this table.

Name	Type	Description
LiveSSTableCount	Gauge<Integer>	Number of SSTables on disk for this table.
LiveDiskSpaceUsed	Counter	Disk space used by SSTables belonging to this table (in bytes).
TotalDiskSpaceUsed	Counter	Total disk space used by SSTables belonging to this table, including obsolete ones waiting to be GC'd.
MaxSSTableSize	Gauge<Long>	Maximum size of SSTable of this table - the physical size on disk of all components for such SSTable in bytes. Equals to zero if there is not any SSTable on disk.
MaxSSTableDuration	Gauge<Long>	Maximum duration in milliseconds of an SSTable for this table, computed as `maxTimestamp - minTimestamp`. Equals to zero if min or max timestamp is `Long.MAX_VALUE`.
MinPartitionSize	Gauge<Long>	Size of the smallest compacted partition (in bytes).
MaxPartitionSize	Gauge<Long>	Size of the largest compacted partition (in bytes).
MeanPartitionSize	Gauge<Long>	Size of the average compacted partition (in bytes).
BloomFilterFalsePositives	Gauge<Long>	Number of false positives on table's bloom filter.
BloomFilterFalseRatio	Gauge<Double>	False positive ratio of table's bloom filter.
BloomFilterDiskSpaceUsed	Gauge<Long>	Disk space used by bloom filter (in bytes).
BloomFilterOffHeapMemoryUsed	Gauge<Long>	Off-heap memory used by bloom filter.
IndexSummaryOffHeapMemoryUsed	Gauge<Long>	Off-heap memory used by index summary.
CompressionMetadataOffHeapMemoryUsed	Gauge<Long>	Off-heap memory used by compression meta data.
KeyCacheHitRate	Gauge<Double>	Key cache hit rate for this table.

Name	Type	Description
TombstoneScannedHistogram	Histogram	Histogram of tombstones scanned in queries on this table.
LiveScannedHistogram	Histogram	Histogram of live cells scanned in queries on this table.
ColUpdateTimeDeltaHistogram	Histogram	Histogram of column update time delta on this table.
ViewLockAcquireTime	Timer	Time taken acquiring a partition lock for materialized view updates on this table.
ViewReadTime	Timer	Time taken during the local read of a materialized view update.
TrueSnapshotsSize	Gauge<Long>	Disk space used by snapshots of this table including all SSTable components.
RowCacheHitOutOfRange	Counter	Number of table row cache hits that do not satisfy the query filter, thus went to disk.
RowCacheHit	Counter	Number of table row cache hits.
RowCacheMiss	Counter	Number of table row cache misses.
CasPrepare	Latency	Latency of paxos prepare round.
CasPropose	Latency	Latency of paxos propose round.
CasCommit	Latency	Latency of paxos commit round.
PercentRepaired	Gauge<Double>	Percent of table data that is repaired on disk.
BytesRepaired	Gauge<Long>	Size of table data repaired on disk
BytesUnrepaired	Gauge<Long>	Size of table data unrepaired on disk
BytesPendingRepair	Gauge<Long>	Size of table data isolated for an ongoing incremental repair
SpeculativeRetries	Counter	Number of times speculative retries were sent for this table.
SpeculativeFailedRetries	Counter	Number of speculative retries that failed to prevent a timeout

Name	Type	Description
SpeculativeInsufficientReplicas	Counter	Number of speculative retries that couldn't be attempted due to lack of replicas
SpeculativeSampleLatencyNanos	Gauge<Long>	Number of nanoseconds to wait before speculation is attempted. Value may be statically configured or updated periodically based on coordinator latency.
AnticompactionTime	Timer	Time spent anticompacting before a consistent repair.
ValidationTime	Timer	Time spent doing validation compaction during repair.
SyncTime	Timer	Time spent doing streaming during repair.
BytesValidated	Histogram	Histogram over the amount of bytes read during validation.
PartitionsValidated	Histogram	Histogram over the number of partitions read during validation.
BytesAnticompacted	Counter	How many bytes we anticompacted.
BytesMutatedAnticompaction	Counter	How many bytes we avoided anticompacting because the sstable was fully contained in the repaired range.
MutatedAnticompactionGauge	Gauge<Double>	Ratio of bytes mutated vs total bytes repaired.

Keyspace Metrics

Each keyspace in Cassandra has metrics responsible for tracking its state and performance.

Most of these metrics are the same as the `Table Metrics` above, only they are aggregated at the Keyspace level. The keyspace specific metrics are specified in the table below.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.keyspace.<MetricName>.<Keyspace>`

JMX MBean

`org.apache.cassandra.metrics:type=Keyspace scope=<Keyspace> name=<MetricName>`

Name	Type	Description
WriteFailedIdeaCL	Counter	Number of writes that failed to achieve the configured ideal consistency level or 0 if none is configured
IdealCLWriteLatency	Latency	Coordinator latency of writes at the configured ideal consistency level. No values are recorded if ideal consistency level is not configured
RepairTime	Timer	Total time spent as repair coordinator.
RepairPrepareTime	Timer	Total time spent preparing for repair.

ThreadPool Metrics

Cassandra splits work of a particular type into its own thread pool. This provides back-pressure and asynchrony for requests on a node. It's important to monitor the state of these thread pools since they can tell you how saturated a node is.

The metric names are all appended with the specific `ThreadPool` name. The thread pools are also categorized under a specific type.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.ThreadPools.<MetricName>.<Path>.<ThreadPoolName>`

JMX MBean

`org.apache.cassandra.metrics:type=ThreadPools path=<Path> scope=<ThreadPoolName> name=<MetricName>`

Name	Type	Description
ActiveTasks	Gauge<Integer>	Number of tasks being actively worked on by this pool.

Name	Type	Description
PendingTasks	Gauge<Integer>	Number of queued tasks queued up on this pool.
CompletedTasks	Counter	Number of tasks completed.
TotalBlockedTasks	Counter	Number of tasks that were blocked due to queue saturation.
CurrentlyBlockedTask	Counter	Number of tasks that are currently blocked due to queue saturation but on retry will become unblocked.
MaxPoolSize	Gauge<Integer>	The maximum number of threads in this pool.
MaxTasksQueued	Gauge<Integer>	The maximum number of tasks queued before a task get blocked.

The following thread pools can be monitored.

Name	Type	Description
Native-Transport-Requests	transport	Handles client CQL requests
CounterMutationStage	request	Responsible for counter writes
ViewMutationStage	request	Responsible for materialized view writes
MutationStage	request	Responsible for all other writes
ReadRepairStage	request	ReadRepair happens on this thread pool
ReadStage	request	Local reads run on this thread pool
RequestResponseStage	request	Coordinator requests to the cluster run on this thread pool
AntiEntropyStage	internal	Builds merkle tree for repairs
CacheCleanupExecutor	internal	Cache maintenance performed on this thread pool
CompactionExecutor	internal	Compactions are run on these threads
GossipStage	internal	Handles gossip requests
HintsDispatcher	internal	Performs hinted handoff

Name	Type	Description
InternalResponseStage	internal	Responsible for intra-cluster callbacks
MemtableFlushWriter	internal	Writes memtables to disk
MemtablePostFlush	internal	Cleans up commit log after memtable is written to disk
MemtableReclaimMemory	internal	Memtable recycling
MigrationStage	internal	Runs schema migrations
MiscStage	internal	Misceleneous tasks run here
PendingRangeCalculator	internal	Calculates token range
PerDiskMemtableFlushWriter_0	internal	Responsible for writing a spec (there is one of these per disk 0-N)
Sampler	internal	Responsible for re-sampling the index summaries of SStables
SecondaryIndexManagement	internal	Performs updates to secondary indexes
ValidationExecutor	internal	Performs validation compaction or scrubbing
ViewBuildExecutor	internal	Performs materialized views initial build

Client Request Metrics

Client requests have their own set of metrics that encapsulate the work happening at coordinator level.

Different types of client requests are broken down by `RequestType`.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.ClientRequest.<MetricName>.<RequestType>`

JMX MBean

`org.apache.cassandra.metrics:type=ClientRequest scope=<RequestType> name=<MetricName>`

RequestType

CASRead

Description

Metrics related to transactional read requests.

Metrics

Name	Type	Description
Timeouts	Counter	Number of timeouts encountered.
Failures	Counter	Number of transaction failures encountered.
	Latency	Transaction read latency.
Unavailables	Counter	Number of unavailable exceptions encountered.
UnfinishedCommit	Counter	Number of transactions that were committed on read.
ConditionNotMet	Counter	Number of transaction preconditions did not match current values.
ContentionHistogram	Histogram	How many contended reads were encountered

RequestType

CASWrite

Description

Metrics related to transactional write requests.

Metrics

Name	Type	Description
Timeouts	Counter	Number of timeouts encountered.
Failures	Counter	Number of transaction failures encountered.
	Latency	Transaction write latency.
Unavailables	Counter	Number of unavailable exceptions encountered.
UnfinishedCommit	Counter	Number of transactions that were committed on write.

Name	Type	Description
ConditionNotMet	Counter	Number of transaction preconditions did not match current values.
ContentionHistogram	Histogram	How many contended writes were encountered
MutationSizeHistogram	Histogram	Total size in bytes of the requests mutations.

RequestType

Read

Description

Metrics related to standard read requests.

Metrics

Name	Type	Description
Timeouts	Counter	Number of timeouts encountered.
Failures	Counter	Number of read failures encountered.
	Latency	Read latency.
Unavailables	Counter	Number of unavailable exceptions encountered.

RequestType

RangeSlice

Description

Metrics related to token range read requests.

Metrics

Name	Type	Description
Timeouts	Counter	Number of timeouts encountered.
Failures	Counter	Number of range query failures encountered.
	Latency	Range query latency.

Name	Type	Description
Unavailables	Counter	Number of unavailable exceptions encountered.

RequestType

Write

Description

Metrics related to regular write requests.

Metrics

Name	Type	Description
Timeouts	Counter	Number of timeouts encountered.
Failures	Counter	Number of write failures encountered.
	Latency	Write latency.
Unavailables	Counter	Number of unavailable exceptions encountered.
MutationSizeHistogram	Histogram	Total size in bytes of the requests mutations.

RequestType

ViewWrite

Description

Metrics related to materialized view write wrtes.

Metrics

Timeouts	Counter	Number of timeouts encountered.
Failures	Counter	Number of transaction failures encountered.
Unavailables	Counter	Number of unavailable exceptions encountered.
ViewReplicasAttempted	Counter	Total number of attempted view replica writes.
ViewReplicasSuccess	Counter	Total number of succeded view replica writes.

Timeouts	Counter	Number of timeouts encountered.
ViewPendingMutations	Gauge<Long>	ViewReplicasAttempted - ViewReplicasSuccess.
ViewWriteLatency	Timer	Time between when mutation is applied to base table and when CL.ONE is achieved on view.

Cache Metrics

Cassandra caches have metrics to track the effectivness of the caches. Though the `Table Metrics` might be more useful.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.Cache.<MetricName>.<CacheName>`

JMX MBean

`org.apache.cassandra.metrics:type=Cache scope=<CacheName> name=<MetricName>`

Name	Type	Description
Capacity	Gauge<Long>	Cache capacity in bytes.
Entries	Gauge<Integer>	Total number of cache entries.
FifteenMinuteCacheHitRate	Gauge<Double>	15m cache hit rate.
FiveMinuteCacheHitRate	Gauge<Double>	5m cache hit rate.
OneMinuteCacheHitRate	Gauge<Double>	1m cache hit rate.
HitRate	Gauge<Double>	All time cache hit rate.
Hits	Meter	Total number of cache hits.
Misses	Meter	Total number of cache misses.
MissLatency	Timer	Latency of misses.
Requests	Gauge<Long>	Total number of cache requests.
Size	Gauge<Long>	Total size of occupied cache, in bytes.

The following caches are covered:

Name	Description
CounterCache	Keeps hot counters in memory for performance.
ChunkCache	In process uncompressed page cache.
KeyCache	Cache for partition to sstable offsets.
RowCache	Cache for rows kept in memory.

NOTE Misses and MissLatency are only defined for the ChunkCache

CQL Metrics

Metrics specific to CQL prepared statement caching.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.CQL.<MetricName>`

JMX MBean

`org.apache.cassandra.metrics:type=CQL name=<MetricName>`

Name	Type	Description
PreparedStatementsCount	Gauge<Integer>	Number of cached prepared statements.
PreparedStatementsEvicted	Counter	Number of prepared statements evicted from the prepared statement cache
PreparedStatementsExecuted	Counter	Number of prepared statements executed.
RegularStatementsExecuted	Counter	Number of **non** prepared statements executed.
PreparedStatementsRatio	Gauge<Double>	Percentage of statements that are prepared vs unprepared.

DroppedMessage Metrics

Metrics specific to tracking dropped messages for different types of requests. Dropped writes are stored and retried by `Hinted Handoff`

Reported name format:

Metric Name

`org.apache.cassandra.metrics.DroppedMessage.<MetricName>.<Type>`

JMX MBean

`org.apache.cassandra.metrics:type=DroppedMessage scope=<Type> name=<MetricName>`

Name	Type	Description
CrossNodeDroppedLatency	Timer	The dropped latency across nodes.
InternalDroppedLatency	Timer	The dropped latency within node.
Dropped	Meter	Number of dropped messages.

The different types of messages tracked are:

Name	Description
BATCH_STORE	Batchlog write
BATCH_REMOVE	Batchlog cleanup (after succesfully applied)
COUNTER_MUTATION	Counter writes
HINT	Hint replay
MUTATION	Regular writes
READ	Regular reads
READ_REPAIR	Read repair
PAGED_SLICE	Paged read
RANGE_SLICE	Token range read
REQUEST_RESPONSE	RPC Callbacks
_TRACE	Tracing writes

Streaming Metrics

Metrics reported during `Streaming` operations, such as repair, bootstrap, rebuild.

These metrics are specific to a peer endpoint, with the source node being the node you are pulling the metrics from.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.Streaming.<MetricName>.<PeerIP>`

JMX MBean

`org.apache.cassandra.metrics:type=Streaming scope=<PeerIP> name=<MetricName>`

Name	Type	Description
IncomingBytes	Counter	Number of bytes streamed to this node from the peer.
OutgoingBytes	Counter	Number of bytes streamed to the peer endpoint from this node.

Compaction Metrics

Metrics specific to `Compaction` work.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.Compaction.<MetricName>`

JMX MBean

`org.apache.cassandra.metrics:type=Compaction name=<MetricName>`

Name	Type	Description
BytesCompacted	Counter	Total number of bytes compacted since server [re]start.
PendingTasks	Gauge<Integer>	Estimated number of compactions remaining to perform.
CompletedTasks	Gauge<Long>	Number of completed compactions since server [re]start.
TotalCompactionsCompleted	Meter	Throughput of completed compactions since server [re]start.
PendingTasksByTableName	Gauge<Map<String, Map<String, Integer>>>	Estimated number of compactions remaining to perform, grouped by keyspace and then table name. This info is also kept in `Table Metrics`.

CommitLog Metrics

Metrics specific to the `CommitLog`

Reported name format:

Metric Name

`org.apache.cassandra.metrics.CommitLog.<MetricName>`

JMX MBean

`org.apache.cassandra.metrics:type=CommitLog name=<MetricName>`

Name	Type	Description
CompletedTasks	Gauge<Long>	Total number of commit log messages written since [re]start.
PendingTasks	Gauge<Long>	Number of commit log messages written but yet to be fsync'd.
TotalCommitLogSize	Gauge<Long>	Current size, in bytes, used by all the commit log segments.
WaitingOnSegmentAllocation	Timer	Time spent waiting for a CommitLogSegment to be allocated - under normal conditions this should be zero.
WaitingOnCommit	Timer	The time spent waiting on CL fsync; for Periodic this is only occurs when the sync is lagging its sync interval.

Storage Metrics

Metrics specific to the storage engine.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.Storage.<MetricName>`

JMX MBean

`org.apache.cassandra.metrics:type=Storage name=<MetricName>`

Name	Type	Description
Exceptions	Counter	Number of internal exceptions caught. Under normal exceptions this should be zero.
Load	Counter	Size, in bytes, of the on disk data size this node manages.
TotalHints	Counter	Number of hint messages written to this node since [re]start. Includes one entry for each host to be hinted per hint.
TotalHintsInProgress	Counter	Number of hints attemping to be sent currently.

HintedHandoff Metrics

Metrics specific to Hinted Handoff. There are also some metrics related to hints tracked in `Storage Metrics`

These metrics include the peer endpoint **in the metric name**

Reported name format:

Metric Name

`org.apache.cassandra.metrics.HintedHandOffManager.<MetricName>`

JMX MBean

`org.apache.cassandra.metrics:type=HintedHandOffManager name=<MetricName>`

Name	Type	Description
Hints_created-<PeerIP>	Counter	Number of hints on disk for this peer.
Hints_not_stored-<PeerIP>	Counter	Number of hints not stored for this peer, due to being down past the configured hint window.

HintsService Metrics

Metrics specific to the Hints delivery service. There are also some metrics related to hints tracked in `Storage Metrics`

These metrics include the peer endpoint **in the metric name**

Reported name format:

Metric Name

`org.apache.cassandra.metrics.HintsService.<MetricName>`

JMX MBean

`org.apache.cassandra.metrics:type=HintsService name=<MetricName>`

Name	Type	Description
HintsSucceeded	Meter	A meter of the hints successfully delivered
HintsFailed	Meter	A meter of the hints that failed deliver
HintsTimedOut	Meter	A meter of the hints that timed out
Hint_delays	Histogram	Histogram of hint delivery delays (in milliseconds)
Hint_delays-<PeerIP>	Histogram	Histogram of hint delivery delays (in milliseconds) per peer

SSTable Index Metrics

Metrics specific to the SSTable index metadata.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.Index.<MetricName>.RowIndexEntry`

JMX MBean

`org.apache.cassandra.metrics:type=Index scope=RowIndexEntry name=<MetricName>`

Name	Type	Description
IndexedEntrySize	Histogram	Histogram of the on-heap size, in bytes, of the index across all SSTables.
IndexInfoCount	Histogram	Histogram of the number of on-heap index entries managed across all SSTables.
IndexInfoGets	Histogram	Histogram of the number index seeks performed per SSTable.

BufferPool Metrics

Metrics specific to the internal recycled buffer pool Cassandra manages. This pool is meant to keep allocations and GC lower by recycling on and off heap buffers.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.BufferPool.<MetricName>`

JMX MBean

`org.apache.cassandra.metrics:type=BufferPool name=<MetricName>`

Name	Type	Description
Size	Gauge<Long>	Size, in bytes, of the managed buffer pool
Misses	Meter	The rate of misses in the pool. The higher this is the more allocations incurred.

Client Metrics

Metrics specifc to client managment.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.Client.<MetricName>`

JMX MBean

`org.apache.cassandra.metrics:type=Client name=<MetricName>`

Name	Type	Description
connectedNativeClients	Gauge<Integer>	Number of clients connected to this nodes native protocol server
connections	Gauge<List<Map<String, String>>	List of all connections and their state information
connectedNativeClientsByUser	Gauge<Map<String, Int>	Number of connnective native clients by username

Batch Metrics

Metrics specifc to batch statements.

Reported name format:

Metric Name

`org.apache.cassandra.metrics.Batch.<MetricName>`

JMX MBean

`org.apache.cassandra.metrics:type=Batch name=<MetricName>`

Name	Type	Description
PartitionsPerCounterBatch	Histogram	Distribution of the number of partitions processed per counter batch
PartitionsPerLoggedBatch	Histogram	Distribution of the number of partitions processed per logged batch
PartitionsPerUnloggedBatch	Histogram	Distribution of the number of partitions processed per unlogged batch

JVM Metrics

JVM metrics such as memory and garbage collection statistics can either be accessed by connecting to the JVM using JMX or can be exported using Metric Reporters.

BufferPool

Metric Name

`jvm.buffers.<direct|mapped>.<MetricName>`

JMX MBean

`java.nio:type=BufferPool name=<direct|mapped>`

Name	Type	Description
Capacity	Gauge<Long>	Estimated total capacity of the buffers in this pool
Count	Gauge<Long>	Estimated number of buffers in the pool
Used	Gauge<Long>	Estimated memory that the Java virtual machine is using for this buffer pool

FileDescriptorRatio

Metric Name

`jvm.fd.<MetricName>`

JMX MBean

`java.lang:type=OperatingSystem name=<OpenFileDescriptorCount|MaxFileDescriptorCount>`

Name	Type	Description
Usage	Ratio	Ratio of used to total file descriptors

GarbageCollector

Metric Name

`jvm.gc.<gc_type>.<MetricName>`

JMX MBean

`java.lang:type=GarbageCollector name=<gc_type>`

Name	Type	Description
Count	Gauge<Long>	Total number of collections that have occurred

Name	Type	Description
Time	Gauge<Long>	Approximate accumulated collection elapsed time in milliseconds

Memory

Metric Name

`jvm.memory.<heap/non-heap/total>.<MetricName>`

JMX MBean

`java.lang:type=Memory`

Committed	Gauge<Long>	Amount of memory in bytes that is committed for the JVM to use
Init	Gauge<Long>	Amount of memory in bytes that the JVM initially requests from the OS
Max	Gauge<Long>	Maximum amount of memory in bytes that can be used for memory management
Usage	Ratio	Ratio of used to maximum memory
Used	Gauge<Long>	Amount of used memory in bytes

MemoryPool

Metric Name

`jvm.memory.pools.<memory_pool>.<MetricName>`

JMX MBean

`java.lang:type=MemoryPool name=<memory_pool>`

Committed	Gauge<Long>	Amount of memory in bytes that is committed for the JVM to use
Init	Gauge<Long>	Amount of memory in bytes that the JVM initially requests from the OS

Max	Gauge<Long>	Maximum amount of memory in bytes that can be used for memory management
Usage	Ratio	Ratio of used to maximum memory
Used	Gauge<Long>	Amount of used memory in bytes

JMX

Any JMX based client can access metrics from cassandra.

If you wish to access JMX metrics over http it's possible to download Mx4jTool and place `mx4j-tools.jar` into the classpath. On startup you will see in the log:

```
HttpAdaptor version 3.0.2 started on port 8081
```

To choose a different port (8081 is the default) or a different listen address (0.0.0.0 is not the default) edit `conf/cassandra-env.sh` and uncomment:

```
#MX4J_ADDRESS="-Dmx4jaddress=0.0.0.0"

#MX4J_PORT="-Dmx4jport=8081"
```

Metric Reporters

As mentioned at the top of this section the Cassandra metrics can be exported to a number of external monitoring system using built-in reporters or third-party reporter plugins.