Monitoring Guide

Proactive monitoring and alerting is essential to managing a healthy Couchbase environment. While the Couchbase Web Console provides detailed statistics and basic alerting functionality, it is not intended to be a realtime dashboard and shouldn't be used as the primary operational monitoring utility.

Integration with external monitoring systems is required for two primary purposes: proactive alerting and high resolution trending. The external monitoring system should be capable of setting alert thresholds on a per-metric basis. As the value of most metrics are workload and environment-specific, they will require establishing a baseline for what is "normal" for your use cases. Trending the Couchbase metrics will help establish the baseline values and alerts can be configured when point-in-time values exceed the "normal" range. Trended metrics also allows Couchbase administrators to observe resource consumption over time, informing when scaling events will become necessary.

This document describes how to poll the Couchbase REST API to obtain metrics for an external monitoring system, describes which metrics are most important to monitor, and provides guidance on how to interpret those metrics.

Obtaining Couchbase Metrics

Couchbase exposes monitoring metrics via REST APIs with responses returned in JSON format. There are two types of statistical APIs available, Cluster Manager (port 8091/18091) stats and Service specific administrative stats.

Cluster Manager stats provide statistical sampling for a given service and/or entities at a particular interval. Each response from /stats endpoint will contain a timestamp property for when the sample was taken that will directly correlate to each of the available stats.

Every Cluster Manager endpoint supports two optional query string parameters:

zoom

The zoom parameter determines the interval of samples to return in the response. The zoom parameter provides the following granularity:

- zoom=minute (default) - Every second for the last minute (60 samples)
- zoom=hour - Every four (4) seconds for the last hour (900 samples)
- zoom=day - Every minute for the last day (1440 samples)
- zoom=week - Every ten (10) minutes for the last week, actually, eight (8) days (1152 samples)
- zoom=year - Every six (6) hours for the last year (1464 samples)

Due to sample frequency, the number of samples returned are plus or minus one (+-1).

haveTStamp

Requests statistics from this timestamp until the current time. The haveTStamp parameter is specified as UNIX epoch time in milliseconds.
To limit the results when using the zoom parameter, post-process the results. For example, if you need samples from the last five (5) minutes, set the zoom parameter to one hour and retrieve the last 75 entries from the JSON list.

Polling the APIs

The REST APIs should be polled minutely via a local agent or remotely using the node(s) IP or hostname. Couchbase REST APIs must be accessed using administrative account credentials; a Read-Only Administrator is recommended for this purpose.

As most of the metrics provided by the REST API are per-node, it is necessary to query every node in the cluster.

Limit the number of requests per API when querying metrics, i.e. return all bucket metrics in one request rather than issuing separate requests per metric. Heavy use of the Couchbase REST APIs can have CPU utilization impacts on the cluster.

Couchbase Service Discovery

Some monitoring systems are capable of discovering new monitoring targets and automatically defining the monitoring profile to be applied. Couchbase supports this by exposing cluster membership, MDS service assignment, and service ports via the Data Service Node API.

Metrics and Services to Monitor

Each section in the list describe the available monitoring metrics exposed by the Couchbase service, a description of each metric, and possible operational responses. Alerts should be configured to be sent from the external monitoring system when metric values fall outside the expected range. Guidance on interpreting the metrics and possible operational responses is provided.

Each guide will contain examples of how to call an endpoint and parse the results. For these examples a tool called `jq` is used, it is a lightweight cli parser for JSON, this is not required and is provided for example purposes only. It can be downloaded at https://stedolan.github.io/jq/download

- Monitoring: Operating System
- Monitoring: Nodes
- Monitoring: Data Service
- Monitoring: XDCR
- Monitoring: Query Service
- Monitoring: Index Service
- Monitoring: FTS Service
- Monitoring: Eventing Service
• Monitoring: Logs

Reference Implementations

Couchbase provides a reference monitoring implementation to demonstrate interacting with the available REST APIs.

• A sample Nagios plugin is available [here](#).
• A complete dockerized monitoring environment is available [here](#).

Third Party Integrations

The following monitoring systems have plugins available for Couchbase. Note that these are third party integrations and may not be complete nor follow the best practices set forth in this document.

• [Couchbase Node Exporter](#) for Prometheus, see the [Prometheus Integration Guide](#) for details
• AppDynamics
• DataDog
• Dynatrace
• New Relic
• SignalFx
• Sensu
• ManageEngine
Monitoring: Data Service

Buckets Overview

Buckets overview provides all available buckets, high-level system information and resource utilization for each bucket in the cluster.

Documentation: https://docs.couchbase.com/server/6.0/rest-api/rest-buckets-summary.html

- Insecure: http://localhost:8091/pools/default/buckets
- Secure: https://localhost:18091/pools/default/buckets

Example

The following example illustrates retrieving all of the buckets in a cluster and displaying basic stats about each bucket.

```bash
curl \
  --user Administrator:password \n  --silent \n  --request GET \n  --data skipMap=true \n  http://localhost:8091/pools/default/buckets | \n  jq -r '.[] | \
  " Bucket: " + .name + "\n" + \
  " Quota Used:" + (.basicStats.quotaPercentUsed | tostring) + "%\n" + \
  " Ops / Sec:" + (.basicStats.opsPerSec | tostring) + "\n" + \
  " Disk Fetches:" + (.basicStats.diskFetches | tostring) + "\n" + \
  " Item Count:" + (.basicStats.itemCount | tostring) + "\n" + \
  " Disk Used:" + (.basicStats.diskUsed / 1024 / 1024 | tostring) + "MB\n" + \
  " Data Used:" + (.basicStats.dataUsed / 1024 / 1024 | tostring) + "MB\n" + \
  " Memory Used:" + (.basicStats.memUsed / 1024 / 1024 | tostring) + "MB"
```

Note: The `skipMap` query string parameter is a boolean value that can be used to include or exclude the current vBucket distribution map for the buckets.

Individual Bucket-Level Stats
Bucket metrics provide detailed information about resource consumption, application workload, and internal operations at the bucket level. The following Bucket stats are available via the Cluster-Wide or Per-Node Endpoints listed below.

Documentation: https://docs.couchbase.com/server/6.0/rest-api/rest-bucket-stats.html

- Insecure: http://localhost:8091/pools/default/buckets/{BUCKET}/stats
- Secure: https://localhost:18091/pools/default/buckets/{BUCKET}/stats

### Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg_active_timestamp_drift</td>
<td>Average drift (in seconds) per mutation on active vBuckets</td>
</tr>
<tr>
<td>avg_bg_wait_time</td>
<td>Average background fetch time in microseconds</td>
</tr>
<tr>
<td>avg_disk_commit_time</td>
<td>Average disk commit time in seconds as from disk_update histogram of timings</td>
</tr>
<tr>
<td>avg_disk_update_time</td>
<td>Average disk update time in microseconds as from disk_update histogram of timings</td>
</tr>
<tr>
<td>avg_replica_timestamp_drift</td>
<td>Average drift (in seconds) per mutation on replica vBuckets</td>
</tr>
<tr>
<td>bg_wait_count</td>
<td>Number of background fetch operations</td>
</tr>
<tr>
<td>bg_wait_total</td>
<td>Background fetch time in microseconds</td>
</tr>
<tr>
<td>bytes_read</td>
<td>Number of bytes per second sent into this bucket</td>
</tr>
<tr>
<td>bytes_written</td>
<td>Number of bytes per second sent from this bucket</td>
</tr>
<tr>
<td>cas_badval</td>
<td>Number of CAS operations per second using an incorrect CAS ID for data that this bucket contains</td>
</tr>
<tr>
<td>cas_hits</td>
<td>Number of CAS operations per second for data that this bucket contains</td>
</tr>
<tr>
<td>cas_misses</td>
<td>Number of CAS operations per second for data that this bucket does not contain</td>
</tr>
<tr>
<td>cmd_get</td>
<td>Number of get operations serviced by this bucket</td>
</tr>
<tr>
<td>cmd_lookup</td>
<td>Number of lookup sub-document operations serviced by this bucket</td>
</tr>
<tr>
<td>cmd_set</td>
<td>Number of set operations serviced by this bucket</td>
</tr>
<tr>
<td>couch_docs_actual_disk_size</td>
<td>The size of all data files for this bucket, including the data itself, metadata and temporary files</td>
</tr>
<tr>
<td>couch_docs_data_size</td>
<td>The size of active data in this bucket</td>
</tr>
<tr>
<td>couch_docs_disk_size</td>
<td>The size of active data in this bucket on disk</td>
</tr>
<tr>
<td>couch_docs_fragmentation</td>
<td>How much fragmented data there is to be compacted compared to real data for the data files in this bucket</td>
</tr>
<tr>
<td>couch_spatial_data_size</td>
<td>The size of all active items in all the spatial indexes for this bucket on disk</td>
</tr>
<tr>
<td>couch_spatial_disk_size</td>
<td>The size of all active items in all the spatial indexes for</td>
</tr>
</tbody>
</table>

Monitoring: Data Service

Couchbase Professional Services
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>couch_spatial_disk_size</td>
<td>The size of all active items in all the spatial indexes for this bucket on disk</td>
</tr>
<tr>
<td>couch_spatial_ops</td>
<td>All the spatial index reads</td>
</tr>
<tr>
<td>couch_total_disk_size</td>
<td>The total size on disk of all data and view files for this bucket.</td>
</tr>
<tr>
<td>couch_views_actual_disk_size</td>
<td>The size of all active items in all the indexes for this bucket on disk.</td>
</tr>
<tr>
<td>couch_views_data_size</td>
<td>The size of active data on for all the view indexes in this bucket.</td>
</tr>
<tr>
<td>couch_views_disk_size</td>
<td>The size of active data on for all the view indexes in this bucket on disk.</td>
</tr>
<tr>
<td>couch_views_fragmentation</td>
<td>How much fragmented data there is to be compacted compared to real data for the view index files in this bucket</td>
</tr>
<tr>
<td>couch_views_ops</td>
<td>All the view reads for all design documents including scatter gather.</td>
</tr>
<tr>
<td>curr_connections</td>
<td>Number of connections to this server including connections from external client SDKs, proxies, DCP requests and internal statistic gathering</td>
</tr>
<tr>
<td>curr_items</td>
<td>Number of unique items in this bucket - only active items, not replica.</td>
</tr>
<tr>
<td>curr_items_tot</td>
<td>Total number of items in this bucket (including replicas).</td>
</tr>
<tr>
<td>decr_hits</td>
<td>Number of decrement operations per second for data that this bucket contains</td>
</tr>
<tr>
<td>decr_misses</td>
<td>Number of decr operations per second for data that this bucket does not contain</td>
</tr>
<tr>
<td>delete_hits</td>
<td>Number of delete operations per second for this bucket.</td>
</tr>
<tr>
<td>delete_misses</td>
<td>Number of delete operations per second for data that this bucket does.</td>
</tr>
<tr>
<td>disk_commit_count</td>
<td>The number of disk comments</td>
</tr>
<tr>
<td>disk_commit_total</td>
<td>The total time spent committing to disk.</td>
</tr>
<tr>
<td>disk_update_count</td>
<td>The total number of disk updates.</td>
</tr>
<tr>
<td>disk_update_total</td>
<td>The total time spent updating disk.</td>
</tr>
<tr>
<td>disk_write_queue</td>
<td>Number of items waiting to be written to disk in this bucket.</td>
</tr>
<tr>
<td>ep_active_ahead_exceptions</td>
<td>Total number of ahead exceptions for all active vBuckets.</td>
</tr>
<tr>
<td>ep_active_hlc_drift</td>
<td>The sum of total_abs_drift for the nodes active vBuckets.</td>
</tr>
<tr>
<td>ep_active_hlc_drift_count</td>
<td>The sum of total_abs_drift_count for the nodes active vBuckets.</td>
</tr>
<tr>
<td>ep_bg_fetched</td>
<td>Number of reads per second from disk for this bucket.</td>
</tr>
<tr>
<td>ep_cache_miss_rate</td>
<td>Percentage of reads per second to this bucket from disk as opposed to RAM.</td>
</tr>
<tr>
<td>ep_cache_miss_rate</td>
<td>Percentage of reads per second to this bucket from disk as opposed to RAM.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>ep_clock_cas_drift_threshold_exceeded</code></td>
<td>Number of disk read failures</td>
</tr>
<tr>
<td><code>ep_data_read_failed</code></td>
<td>Number of disk write failures</td>
</tr>
<tr>
<td><code>ep_dcp_2i_backoff</code></td>
<td>Number of backoffs for index DCP connections</td>
</tr>
<tr>
<td><code>ep_dcp_2i_count</code></td>
<td>Number of internal second index DCP connections in this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_2i_items_remaining</code></td>
<td>Number of secondary index items remaining to be sent to consumer in this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_2i_items_sent</code></td>
<td>Number of secondary index items per second being sent for a producer for this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_2i_producer_count</code></td>
<td>Number of secondary index senders for this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_2i_total_backlog_size</code></td>
<td>Total size in bytes of the DCP backlog for secondary indexes</td>
</tr>
<tr>
<td><code>ep_dcp_2i_total_bytes</code></td>
<td>Number of bytes per second being sent for secondary indexes DCP connections</td>
</tr>
<tr>
<td><code>ep_dcp_cbas_backoff</code></td>
<td>Number of backoffs for Analytics DCP connections</td>
</tr>
<tr>
<td><code>ep_dcp_cbas_count</code></td>
<td>Number of internal Analytics DCP connections in this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_cbas_items_remaining</code></td>
<td>Number of Analytics items remaining to be sent to consumer in this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_cbas_items_sent</code></td>
<td>Number of Analytics items per second being sent for a producer for this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_cbas_producer_count</code></td>
<td>Number of Analytics senders for this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_cbas_total_backlog_size</code></td>
<td>Total size in bytes of the DCP backlog for Analytics</td>
</tr>
<tr>
<td><code>ep_dcp_cbas_total_bytes</code></td>
<td>Number of bytes per second being sent for Analytics DCP connections</td>
</tr>
<tr>
<td><code>ep_dcp_eventing_backoff</code></td>
<td>Number of backoffs for Eventing DCP connections</td>
</tr>
<tr>
<td><code>ep_dcp_eventing_count</code></td>
<td>Number of internal Eventing DCP connections in this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_eventing_items_remaining</code></td>
<td>Number of Eventing items remaining to be sent to consumer in this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_eventing_items_sent</code></td>
<td>Number of Eventing items per second being sent for a producer for this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_eventing_producer_count</code></td>
<td>Number of Eventing senders for this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_eventing_total_backlog_size</code></td>
<td>Total size in bytes of the DCP backlog for Eventing</td>
</tr>
<tr>
<td><code>ep_dcp_eventing_total_bytes</code></td>
<td>Number of bytes per second being sent for Eventing DCP connections</td>
</tr>
<tr>
<td><code>ep_dcp_fts_backoff</code></td>
<td>Number of backoffs for FTS DCP connections</td>
</tr>
<tr>
<td><code>ep_dcp_fts_count</code></td>
<td>Number of internal FTS DCP connections in this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_fts_items_remaining</code></td>
<td>Number of FTS items remaining to be sent to consumer in this bucket</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ep_dcp_fts_items_sent</td>
<td>Number of FTS items per second being sent for a producer for this bucket</td>
</tr>
<tr>
<td>ep_dcp_fts_producer_count</td>
<td>Number of FTS senders for this bucket</td>
</tr>
<tr>
<td>ep_dcp_fts_total_backlog_size</td>
<td>Total size in bytes of the DCP backlog for FTS</td>
</tr>
<tr>
<td>ep_dcp_fts_total_bytes</td>
<td>Number of bytes per second being sent for FTS DCP connections</td>
</tr>
<tr>
<td>ep_dcp_other_backoff</td>
<td>Number of backoffs for other DCP connections</td>
</tr>
<tr>
<td>ep_dcp_other_count</td>
<td>Number of other DCP connections in this bucket</td>
</tr>
<tr>
<td>ep_dcp_other_items_remaining</td>
<td>Number of items remaining to be sent to consumer in this bucket</td>
</tr>
<tr>
<td>ep_dcp_other_items_sent</td>
<td>Number of items per second being sent for a producer for this bucket</td>
</tr>
<tr>
<td>ep_dcp_other_producer_count</td>
<td>Number of other senders for this bucket</td>
</tr>
<tr>
<td>ep_dcp_other_total_backlog_size</td>
<td>Total size in bytes of the DCP backlog for analytics other</td>
</tr>
<tr>
<td>ep_dcp_other_total_bytes</td>
<td>Number of bytes per second being sent for other DCP connections for this bucket</td>
</tr>
<tr>
<td>ep_dcp_replica_backoff</td>
<td>Number of backoffs for replication DCP connections</td>
</tr>
<tr>
<td>ep_dcp_replica_count</td>
<td>Number of internal replication DCP connections in this bucket</td>
</tr>
<tr>
<td>ep_dcp_replica_items_remaining</td>
<td>Number of replication items remaining to be sent to consumer in this bucket</td>
</tr>
<tr>
<td>ep_dcp_replica_items_sent</td>
<td>Number of replication items per second being sent for a producer for this bucket</td>
</tr>
<tr>
<td>ep_dcp_replica_producer_count</td>
<td>Number of replication senders for this bucket</td>
</tr>
<tr>
<td>ep_dcp_replica_total_backlog_size</td>
<td>Total size in bytes of the DCP backlog for replication</td>
</tr>
<tr>
<td>ep_dcp_replica_total_bytes</td>
<td>Number of bytes per second being sent for replication DCP connections</td>
</tr>
<tr>
<td>ep_dcp_views+indexes_backoff</td>
<td>Number of backoffs for view/index DCP connections</td>
</tr>
<tr>
<td>ep_dcp_views+indexes_count</td>
<td>Number of internal view/index DCP connections in this bucket</td>
</tr>
<tr>
<td>ep_dcp_views+indexes_items_remaining</td>
<td>Number of view/index items remaining to be sent to consumer in this bucket</td>
</tr>
<tr>
<td>ep_dcp_views+indexes_items_sent</td>
<td>Number of view/index items per second being sent for a producer for this bucket</td>
</tr>
<tr>
<td>ep_dcp_views+indexes_producer_count</td>
<td>Number of views/index senders for this bucket</td>
</tr>
<tr>
<td>ep_dcp_views+indexes_total_backlog_size</td>
<td>Total size in bytes of the DCP backlog for views/indexes</td>
</tr>
<tr>
<td>ep_dcp_views+indexes_total_bytes</td>
<td>Number of bytes per second being sent for views/indexes DCP connections</td>
</tr>
<tr>
<td>ep_dcp_views_backoff</td>
<td>Number of backoffs for view DCP connections</td>
</tr>
<tr>
<td>ep_dcp_views_count</td>
<td>Number of internal view DCP connections in this bucket</td>
</tr>
<tr>
<td></td>
<td>Number of view items remaining to be sent to consumer</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>ep_dcp_views_items_remaining</code></td>
<td>Number of view items remaining to be sent to consumer in this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_views_items_sent</code></td>
<td>Number of view items per second being sent for a producer for this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_views_producer_count</code></td>
<td>Number of view senders for this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_views_total_backlog_size</code></td>
<td>Total size in bytes of the DCP backlog for views</td>
</tr>
<tr>
<td><code>ep_dcp_views_total_bytes</code></td>
<td>Number of bytes per second being sent for DCP connections</td>
</tr>
<tr>
<td><code>ep_dcp_xdcr_backoff</code></td>
<td>Number of backoffs for XDCR DCP connections</td>
</tr>
<tr>
<td><code>ep_dcp_xdcr_count</code></td>
<td>Number of internal XDCR DCP connections in this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_xdcr_items_remaining</code></td>
<td>Number of XDCR items remaining to be sent to consumer in this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_xdcr_items_sent</code></td>
<td>Number of XDCR items per second being sent for a producer for this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_xdcr_producer_count</code></td>
<td>Number of XDCR senders for this bucket</td>
</tr>
<tr>
<td><code>ep_dcp_xdcr_total_backlog_size</code></td>
<td>Total size in bytes of the DCP backlog for XDCR</td>
</tr>
<tr>
<td><code>ep_dcp_xdcr_total_bytes</code></td>
<td>Number of bytes per second being sent for XDCR DCP connections</td>
</tr>
<tr>
<td><code>ep_diskqueue_drain</code></td>
<td>Total number of items per second being written to disk in this bucket</td>
</tr>
<tr>
<td><code>ep_diskqueue_fill</code></td>
<td>Total number of items per second being put on the disk queue in this bucket</td>
</tr>
<tr>
<td><code>ep_diskqueue_items</code></td>
<td>Total number of items waiting to be written to disk in this bucket</td>
</tr>
<tr>
<td><code>ep_flusher_todo</code></td>
<td>Number of items currently being written.</td>
</tr>
<tr>
<td><code>ep_item_commit_failed</code></td>
<td>Number of times a transaction failed to commit due to storage errors.</td>
</tr>
<tr>
<td><code>ep_kv_size</code></td>
<td>Total amount of user data cached in RAM in this bucket</td>
</tr>
<tr>
<td><code>ep_max_size</code></td>
<td>The maximum amount of memory this bucket can use.</td>
</tr>
<tr>
<td><code>ep_mem_high_wat</code></td>
<td>High water mark for auto-evictions</td>
</tr>
<tr>
<td><code>ep_mem_low_wat</code></td>
<td>Low water mark for auto-evictions</td>
</tr>
<tr>
<td><code>ep_meta_data_memory</code></td>
<td>Total amount of item metadata consuming RAM in this bucket</td>
</tr>
<tr>
<td><code>ep_num_non_resident</code></td>
<td>The number of non-resident items.</td>
</tr>
<tr>
<td><code>ep_num_ops_del_meta</code></td>
<td>Number of delete operations per second for this bucket as the target for XDCR</td>
</tr>
<tr>
<td><code>ep_num_ops_del_ret_meta</code></td>
<td>Number of delRetMeta operations.</td>
</tr>
<tr>
<td><code>ep_num_ops_get_meta</code></td>
<td>Number of metadata read operations per second for this bucket as the target for XDCR</td>
</tr>
<tr>
<td><code>ep_num_ops_set_meta</code></td>
<td>Number of set operations per second for this bucket as the target for XDCR</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ep_num_ops_set_ret_meta</td>
<td>Total number of items per second being ejected to disk in this bucket</td>
</tr>
<tr>
<td>ep_num_value_ejects</td>
<td>Number of times unrecoverable OOMs happened while processing operations.</td>
</tr>
<tr>
<td>ep_oom_errors</td>
<td>Total number of new items being inserted into this bucket</td>
</tr>
<tr>
<td>ep_ops_create</td>
<td>Number of items updated on disk per second for this bucket</td>
</tr>
<tr>
<td>ep_overhead</td>
<td>Extra memory used by transient data like persistence queues, replication queues, checkpoints, etc.</td>
</tr>
<tr>
<td>ep_queue_size</td>
<td>Number of items queued for storage.</td>
</tr>
<tr>
<td>ep_replica_ahead_exceptions</td>
<td>Total number of ahead exceptions for all replica vBuckets</td>
</tr>
<tr>
<td>ep_replica_hlc_drift</td>
<td>The sum of total_abs_drift for the node's active vBuckets</td>
</tr>
<tr>
<td>ep_replica_hlc_drift_count</td>
<td>The sum of total_abs_drift_count for the node's active vBuckets</td>
</tr>
<tr>
<td>ep_resident_items_rate</td>
<td>Percentage of all items cached in RAM in this bucket</td>
</tr>
<tr>
<td>ep_tmp_oom_errors</td>
<td>Number of back-offs sent per second to client SDKs due to “out of memory” situations from this bucket</td>
</tr>
<tr>
<td>ep_vb_total</td>
<td>Total number of vBuckets for this bucket</td>
</tr>
<tr>
<td>evictions</td>
<td>Number of items per second evicted from this bucket</td>
</tr>
<tr>
<td>get_hits</td>
<td>Number of get operations per second for data that this bucket contains</td>
</tr>
<tr>
<td>get_misses</td>
<td>Number of get operations per second for data that this bucket does not contain</td>
</tr>
<tr>
<td>hibernated_requests</td>
<td>Number of hibernated requests</td>
</tr>
<tr>
<td>hibernated_waked</td>
<td>Number of times hibernated waked</td>
</tr>
<tr>
<td>hit_ratio</td>
<td>Percentage of get requests served with data from this bucket</td>
</tr>
<tr>
<td>incr_hits</td>
<td>Number of increment operations per second for data that this bucket contains</td>
</tr>
<tr>
<td>incr_misses</td>
<td>Number of increment operations per second for data that this bucket does not contain</td>
</tr>
<tr>
<td>mem_used</td>
<td>Amount of Memory used</td>
</tr>
<tr>
<td>misses</td>
<td>Total amount of operations per second for that the bucket does not contain</td>
</tr>
<tr>
<td>ops</td>
<td>Total amount of operations per second (including XDCR) to this bucket</td>
</tr>
<tr>
<td>rest_requests</td>
<td></td>
</tr>
<tr>
<td>swap_total</td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>swap_used</td>
<td></td>
</tr>
<tr>
<td>vb_active_eject</td>
<td>Number of items per second being ejected to disk from &quot;active&quot;</td>
</tr>
<tr>
<td>vb_active_itm_memory</td>
<td>Amount of active user data cached in RAM in this bucket</td>
</tr>
<tr>
<td>vb_active_meta_data_memory</td>
<td>Amount of active item metadata consuming RAM in this bucket</td>
</tr>
<tr>
<td>vb_active_num</td>
<td>Number of vBuckets in the &quot;active&quot; state for this bucket</td>
</tr>
<tr>
<td>vb_active_num_non_resident</td>
<td>Number of non-resident items.</td>
</tr>
<tr>
<td>vb_active_ops_create</td>
<td>New items per second being inserted into &quot;active&quot; vBuckets in this bucket</td>
</tr>
<tr>
<td>vb_active_ops_update</td>
<td>Number of items updated on disk per second for this bucket</td>
</tr>
<tr>
<td>vb_active_queue_age</td>
<td>Sum of disk queue item age in milliseconds for &quot;active&quot; vBuckets</td>
</tr>
<tr>
<td>vb_active_queue_drain</td>
<td>Number of active items per second being written to disk in this bucket</td>
</tr>
<tr>
<td>vb_active_queue_fill</td>
<td>Number of active items per second being put on the active item disk queue in this bucket</td>
</tr>
<tr>
<td>vb_active_queue_size</td>
<td>Number of active items waiting to be written to disk in this bucket</td>
</tr>
<tr>
<td>vb_active_resident_items_ratio</td>
<td>Percentage of active items cached in RAM in this bucket</td>
</tr>
<tr>
<td>vb_active_sync_write_aborted_count</td>
<td>Number of vbucket writes aborted</td>
</tr>
<tr>
<td>vb_active_sync_write_accepted_count</td>
<td>Number of vbucket writes accepted</td>
</tr>
<tr>
<td>vb_active_sync_write_committed_count</td>
<td>Number of vbucket writes committed</td>
</tr>
<tr>
<td>vb_avg_active_queue_age</td>
<td>Average age in seconds of active items in the active item queue for this bucket</td>
</tr>
<tr>
<td>vb_avg_pending_queue_age</td>
<td>Average age in seconds of pending items in the pending item queue for this bucket and should be transient during rebalancing</td>
</tr>
<tr>
<td>vb_avg_replica_queue_age</td>
<td>Average age in seconds of replica items in the replica item queue for this bucket</td>
</tr>
<tr>
<td>vb_avg_total_queue_age</td>
<td>Average age in seconds of all items in the disk write queue for this bucket</td>
</tr>
<tr>
<td>vb_pending_curr_items</td>
<td>Number of items in &quot;pending&quot; vBuckets in this bucket and should be transient during rebalancing</td>
</tr>
<tr>
<td>vb_pending_eject</td>
<td>Number of items per second being ejected to disk from &quot;pending&quot; vBuckets in this bucket and should be transient during rebalancing</td>
</tr>
<tr>
<td>vb_pending_itm_memory</td>
<td>Amount of pending user data cached in RAM in this bucket and should be transient during rebalancing</td>
</tr>
<tr>
<td>vb_pending_meta_data_memory</td>
<td>Amount of pending item metadata consuming RAM in this bucket and should be transient during rebalancing</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>vb_pending_num</td>
<td>Number of vBuckets in the &quot;pending&quot; state for this bucket and should be transient during rebalancing</td>
</tr>
<tr>
<td>vb_pending_num_non_resident</td>
<td>Number of non-resident items.</td>
</tr>
<tr>
<td>vb_pending_ops_create</td>
<td>New items per second being instead into &quot;pending&quot; vBuckets in this bucket and should be transient during rebalancing</td>
</tr>
<tr>
<td>vb_pending_ops_update</td>
<td>Number of items updated on disk per second for this bucket</td>
</tr>
<tr>
<td>vb_pending_queue_age</td>
<td>Sum of disk queue item age in milliseconds.</td>
</tr>
<tr>
<td>vb_pending_queue_drain</td>
<td>Number of pending items per second being written to disk in this bucket and should be transient during rebalancing</td>
</tr>
<tr>
<td>vb_pending_queue_fill</td>
<td>Number of pending items per second being put on the pending item disk queue in this bucket and should be transient during rebalancing</td>
</tr>
<tr>
<td>vb_pending_queue_size</td>
<td>Number of pending items waiting to be written to disk in this bucket and should be transient during rebalancing</td>
</tr>
<tr>
<td>vb_pending_resident_items_ratio</td>
<td>Percentage of items in pending state vbuckets cached in RAM in this bucket</td>
</tr>
<tr>
<td>vb_replica_curr_items</td>
<td>Number of items in &quot;replica&quot; vBuckets in this bucket</td>
</tr>
<tr>
<td>vb_replica_eject</td>
<td>Number of items per second being ejected to disk from &quot;replica&quot; vBuckets in this bucket</td>
</tr>
<tr>
<td>vb_replica_itm_memory</td>
<td>Amount of replica user data cached in RAM in this bucket</td>
</tr>
<tr>
<td>vb_replica_meta_data_memory</td>
<td>Amount of replica item metadata consuming in RAM in this bucket</td>
</tr>
<tr>
<td>vb_replica_num</td>
<td>Number of vBuckets in the &quot;replica&quot; state for this bucket</td>
</tr>
<tr>
<td>vb_replica_num_non_resident</td>
<td>Number of non-resident items.</td>
</tr>
<tr>
<td>vb_replica_ops_create</td>
<td>New items per second being inserted into &quot;replica&quot; vBuckets in this bucket</td>
</tr>
<tr>
<td>vb_replica_ops_update</td>
<td>Number of items updated on disk per second for this bucket</td>
</tr>
<tr>
<td>vb_replica_queue_age</td>
<td>Sum of disk queue item age in milliseconds for &quot;replica&quot; vBuckets</td>
</tr>
<tr>
<td>vb_replica_queue_drain</td>
<td>Number of replica items per second being written to disk in this bucket</td>
</tr>
<tr>
<td>vb_replica_queue_fill</td>
<td>Number of replica items per second being put on the replica item disk queue in this bucket</td>
</tr>
<tr>
<td>vb_replica_queue_size</td>
<td>Number of replica items waiting to be written to disk in this bucket</td>
</tr>
<tr>
<td>vb_replica_resident_items_ratio</td>
<td>Percentage of replica items cached in RAM in this bucket</td>
</tr>
<tr>
<td>vb_total_queue_age</td>
<td>Sum of disk queue item age in milliseconds.</td>
</tr>
</tbody>
</table>
**GET** Cluster-Wide Individual Bucket Stats

These endpoints are informational and should not be used for monitoring as they are an aggregate for the entire cluster and the best practice is to monitor each node individually.

- Insecure: http://localhost:8091/pools/default/buckets/{BUCKET}/stats
- Secure: https://localhost:18091/pools/default/buckets/{BUCKET}/stats

**Example: With an average for all samples**

```
BUCKET="travel-sample"

curl \
    --user Administrator:password \ 
    --silent \ 
    --request GET \ 
    --data zoom=minute \ 
    http://localhost:8091/pools/default/buckets/$BUCKET/stats | \
    jq -r '.op.samples | to_entries[] | select(.key != "timestamp") | \
    .key + ": " + (.value | add / length | tostring)'
```

**GET** Node-Level Individual Bucket Stats

Each node in the cluster running the data service should be monitoring individually using the endpoint listed below.

- Insecure: http://localhost:8091/pools/default/buckets/{BUCKET}/nodes/{NODE}/stats
- Secure: https://localhost:18091/pools/default/buckets/{BUCKET}/nodes/{NODE}/stats

**Example: Stats for Individual Node**

The following example demonstrates how to retrieve the bucket stats for a specific node.

```
BUCKET="travel-sample"
NODE="172.17.0.2:8091"

curl \
    --user Administrator:password \ 
    --silent \ 
    --request GET \ 
    --data zoom=minute \ 
    http://localhost:8091/pools/default/buckets/$BUCKET/nodes/$NODE/stats | \
    jq -r -c '.op.samples |'```

Monitoring: Data Service
```bash
jq -r -c '.op.samples |
" cmd_get: " + (.cmd_get | add / length | tostring) +
"n cmd_set: " + (.cmd_set | add / length | tostring) +
"n curr_connections: " + (.curr_connections | add / length | tostring) +
"n curr_items: " + (.curr_items | add / length | tostring) +
"n curr_items_tot: " + (.curr_items_tot | add / length | tostring) +
"n decr_hits: " + (.decr_hits | add / length | tostring) +
"n decr_misses: " + (.decr_misses | add / length | tostring) +
"n delete_hits: " + (.delete_hits | add / length | tostring) +
"n delete_misses: " + (.delete_misses | add / length | tostring) +
"n ep_bg_fetched: " + (.ep_bg_fetched | add / length | tostring) +
"n evictions: " + (.evictions | add / length | tostring) +
"n get_hits: " + (.get_hits | add / length | tostring) +
"n get_misses: " + (.get_misses | add / length | tostring) +
"n hit_ratio: " + (.hit_ratio | add / length | tostring) +
"n incr_hits: " + (.incr_hits | add / length | tostring) +
"n incr_misses: " + (.incr_misses | add / length | tostring) +
"n misses: " + (.misses | add / length | tostring) +
"n ops: " + (.ops | add / length | tostring) +
"n xdc_ops: " + (.xdc_ops | add / length | tostring)
`
```

### Key Metrics to Monitor

<table>
<thead>
<tr>
<th>Couchbase Metric</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>mem_used / ep_kv_size</td>
<td>These four metrics together give insight into how memory is used by the data service.</td>
<td>The amount of fragmentation (mem_used / ep_kv_size) you should expect will depend on the workload, but in general, alert if this value exceeds 115%. If mem_used / memoryTotal are consistently near 90%, that is a trigger to add additional memory or nodes to the cluster. If this value approaches 100%, then you could face an Out of Memory error and the Couchbase process could be killed or crash. Once ep_kv_size = ep_mem_high_wat, Couchbase will start ejecting data to disk. This may be expected depending on your use case, but caching use cases will always want ep_kv_size to be lower than ep_mem_high_wat.</td>
</tr>
<tr>
<td>ep_kv_size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ep_mem_high_wat</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- mem_used
- ep_kv_size
- ep_mem_high_wat
### Monitoring: Data Service

#### ep_mem_high_wat
- **Description:**
  - `ep_mem_high_wat` is the maximum RAM the bucket is expected to use.

#### ep_meta_data_memory
- **Description:**
  - The amount of memory used specifically for document metadata. In Value Ejection mode, it's possible for document metadata to displace document values in cache, reducing cache hit rates and increasing latencies.

  - **Action:**
    - Create a baseline for `ep_meta_data_memory / ep_mem_high_wat`. If this value exceeds 30% and `vb_active_resident_items_ratio` is not 100%, consider configuring Full Ejection on the bucket.

#### ep_queue_size
- **Description:**
  - The amount of data waiting to be written to disk. A large value typically indicates the server is disk IO bound. If this value exceeds 1,000,000 items, the server will start sending `tmp_oom` (backoff) messages to the application.

  - **Action:**
    - Create a baseline for this value as "normal" will be dependent upon your workload and available disk IO. Alert at 2x of baseline. You may need to add nodes or increase the per-node disk IO.

#### ep_flusher_todo
- **Description:**
  - The number of items currently being written to disk. Combined with `ep_queue_size`, this represents the total disk write queue on the server.

  - **Action:**
    - Create a baseline for this value as "normal" will be dependent upon your workload and available disk IO. Alert at 2x of baseline.

#### vb_avg_total_queue_age
- **Description:**
  - The average time in seconds that a write is in queue before persisting to disk. This represents the local node's exposure to potential data loss.

  - **Action:**
    - Create a baseline for this value as "normal" will be dependent upon your workload and available disk IO. Alert at 2x of baseline.

#### ep_dcp_replica_items_remaining
- **Description:**
  - The number of items in the inter-node replication queue. This represents the cluster's exposure to potential data loss.

  - **Action:**
    - Create a baseline for this value as "normal" will be dependent upon your workload and available network IO. Alert at 2x of baseline.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ops</td>
<td>The total number of KV operations occurring against the node.</td>
<td></td>
<td>Create a baseline for this value as &quot;normal&quot; will be dependent on your workload. Alert at 2x of baseline. Abnormally high operations could mean an unexpected change to the application or unusual application traffic patterns.</td>
</tr>
<tr>
<td>cmd_get</td>
<td>The number of KV GET operations occurring against the node.</td>
<td></td>
<td>Create a baseline for this value as &quot;normal&quot; will be dependent on your workload. Alert at 3x of baseline. Abnormally high operations could mean an unexpected change to the application or unusual application traffic patterns.</td>
</tr>
<tr>
<td>cmd_set</td>
<td>The number of KV SET operations occurring against the node.</td>
<td></td>
<td>Create a baseline for this value as &quot;normal&quot; will be dependent on your workload. Alert at 2x of baseline. Abnormally high operations could mean an unexpected change to the application or unusual application traffic patterns.</td>
</tr>
<tr>
<td>delete_hits</td>
<td>The number of KV DELETE operations occurring against the node.</td>
<td></td>
<td>Create a baseline for this value as &quot;normal&quot; will be dependent on your workload. Alert at 2x of baseline. Abnormally high operations could mean an unexpected change to the application or unusual application traffic patterns.</td>
</tr>
<tr>
<td>ep_bg_fetched</td>
<td>The number of items fetched from disk (cache misses).</td>
<td></td>
<td>This value should be close to 0. Establish a baseline for this metric and alert at 2x of baseline.</td>
</tr>
<tr>
<td>curr_connections</td>
<td>The number of client (SDK) connections to Couchbase. More connections will result in increased CPU utilization.</td>
<td></td>
<td>Create a baseline for your environment. Alert at 2x of baseline. Couchbase will begin rejecting connections above 30,000.</td>
</tr>
<tr>
<td>curr_items</td>
<td>The number of items currently active on this node. During warmup, this will be 0 until complete.</td>
<td></td>
<td>Once a baseline number of objects has been established, substantial changes to the baseline could indicate unexpected failures within Couchbase or an application bug.</td>
</tr>
<tr>
<td>vb_active_resident_items_ratio</td>
<td>The percentage of active data in that is memory resident.</td>
<td></td>
<td>For caching use cases, this value should be close to 100%. If this value falls below 100% and ep_bg_fetched is greater than 0, this indicates the bucket needs more RAM. The value should never be less than 15%.</td>
</tr>
<tr>
<td></td>
<td>The percentage of replica data in that</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cluster's exposure to potential data loss.
Alert at 2x of baseline.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Formula</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>vb_replica_resident_items_ratio</td>
<td>Resident A higher percentage for this value will ensure lower latency data access following a failover.</td>
<td>on business requirements for object latency during a failure scenario. The value should never be less than 15%</td>
<td></td>
</tr>
<tr>
<td>ep_tmp_oom_errors</td>
<td>Number of times temporary OOMs were sent to a client. Represents high transient memory pressure within the system.</td>
<td>This error indicates temporary memory pressure after the server has reached ep_mem_high_wat and is ejecting not recently accessed values. Frequent errors indicate the need to scale the cluster.</td>
<td></td>
</tr>
<tr>
<td>ep_oom_errors</td>
<td>Number of times permanent OOMs were sent to a client. Represents very high consistent memory pressure within the system.</td>
<td>This error indicates the bucket has exceeded its total memory allocation and immediately requires additional memory or nodes be added.</td>
<td></td>
</tr>
</tbody>
</table>
| • ep_dcp_views_items_remaining  
• ep_dcp_2i_items_remaining | The number of documents awaiting indexing for views and GSI. | Create a baseline for this value as "normal" will be dependent upon your workload and available disk IO. Alert at 2x baseline. |
| ep_dcp_replica_backoff | Indicates the number of times an internal replication was instructed to slow down. | Alert if this value greater than zero. This indicates a resource constraint within the cluster that should be investigated. |
| ep_dcp_xdcr_backoff | Indicates the number of times an XDCR replication was instructed to slow down. | Should be monitored as a rate. Create a baseline for your environment as "normal" will be dependent on workload patterns and XDCR bandwidth limits. Alert at 2x of baseline. |
| couch_docs_fragmentation | The percentage of data file fragmentation. | By default, compaction should start when this value hits 30%. If this value consistently exceeds 30%, then this typically indicates disk IO contention or a problem with compaction starting that should be investigated. |
| couch_views_fragmentation | The percentage of View index fragmentation. | By default, compaction should start when this value hits 30%. If this value significantly exceeds 30%, then this typically indicates disk IO contention or a problem with compaction starting that should be investigated. |
| vb_replica_num | The number of replica vBuckets. | If this value falls below (1024 * the number of configured replicas) / the number of servers, it indicates that a rebalance is required. |
| vb_active_num | The number of active vBuckets. | This value should always equal 1024 / the number of servers. If it does not, it indicates a node failure and that a failover + rebalance is required. |
**Example**

The following example illustrates getting the verbose stats for an individual bucket.

```bash
BUCKET='travel-sample'

# output the stats for the bucket
curl
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/$BUCKET/stats | \
  jq -r -c '.op.samples | to_entries | sort_by(.key) | .[] | " " + (.key) + ": " + (.value | add / length | tostring)'
```

**Example**

The following example illustrates getting an individual stat for a single bucket.

```bash
BUCKET='travel-sample'
STAT='cmd_get'

# output the stats for the bucket
curl
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/$BUCKET/stats/$STAT | \
  jq -r -c '.nodeStats | to_entries | sort_by(.key) | .[] | " " + (.key) + ": " + (.value | add / length | tostring)'
```

**Example**

This example shows how to retrieve all stats for all buckets.

```bash
# loop over each of the buckets
for bucket in $(curl \
  --user Administrator:password \
  --silent \
  --request GET \
```
```bash
jq -r '.[] | .name')
do
  echo ""
echo "Bucket: $bucket"
echo "================================================================"
  # output the stats for the bucket
curl \
    --user Administrator:password \
    --silent \
    --request GET \
    --data zoom=minute \
    http://localhost:8091/pools/default/buckets/$bucket/stats | \
    jq -r c '.op.samples | to_entries | sort_by(.key) | [] | " " + (.key) + ": " + (.value | add / length | tostring)'
done
```
Monitoring: Eventing Service

Eventing Service-Level Stats

The Eventing stats are an aggregate for all of the Eventing Functions deployed, either for the entire cluster or a specific node.

Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventing/bucket_op_exception_count</td>
<td>Total number of bucket operations inside of an Eventing function which have resulted in an exception</td>
</tr>
<tr>
<td>eventing/checkpoint_failure_count</td>
<td>Total number of failures when checkpointing last processed sequence numbers by v8 worker. Failures are retried using exponential backoff until timeout.</td>
</tr>
<tr>
<td>eventing/dcp_backlog</td>
<td>Remaining mutations to process</td>
</tr>
<tr>
<td>eventing/failed_count</td>
<td>Total number of failed Eventing function operations</td>
</tr>
<tr>
<td>eventing/n1ql_op_exception_count</td>
<td>Total number of N1QL operations inside of an Eventing function which have resulted in an exception</td>
</tr>
<tr>
<td>eventing/on_delete_failure</td>
<td>The total number OnDelete handler executions that have failed for all functions</td>
</tr>
<tr>
<td>eventing/on_delete_success</td>
<td>Total OnDelete handler executions that have succeeded for all functions</td>
</tr>
<tr>
<td>eventing/on_update_failure</td>
<td>Total OnUpdate handler executions that have failed for all functions</td>
</tr>
<tr>
<td>eventing/on_update_success</td>
<td>Total OnUpdate handler executions that have failed for all functions</td>
</tr>
<tr>
<td>eventing/processed_count</td>
<td>Total number of mutations that have been processed</td>
</tr>
<tr>
<td>eventing/timeout_count</td>
<td>Total number of handler executions were terminated because the handler ran longer than the configured script timeout</td>
</tr>
</tbody>
</table>

GET Cluster Eventing Service Stats

These endpoints are informational and should not be used for monitoring as they are an aggregate for the entire cluster and the best practice is to monitor each node individually.

- Insecure: http://localhost:8091/pools/default/buckets/@eventing/stats
- Secure: https://localhost:18091/pools/default/buckets/@eventing/stats

Example
The following example demonstrates how to retrieve the eventing service stats for the cluster.

```
curl \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/@eventing/stats | \
  jq -r '.op.samples | to_entries | sort_by(.key) | .[] | 
    select(.key | split("/" | length == 2) | 
      " " + (.key) + ": " + 
      (.value | add / length | tostring)'```

**GET Node-Level Eventing Service Stats**

Each node in the cluster running the eventing service should be monitoring individually using the endpoint listed below.

- **Insecure**: http://localhost:8091/pools/default/buckets/@eventing/nodes/{NODE}/stats
- **Secure**: https://localhost:18091/pools/default/buckets/@eventing/nodes/{NODE}/stats

**Example: Stats for Individual Node**

The following example demonstrates how to retrieve the eventing service stats for a specific node in the cluster.

```
NODE="172.17.0.2:8091"
curl \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/@eventing/nodes/$NODE/stats | \
  jq -r -c '.op.samples | 
    " eventing/bucket_op_exception_count: " + 
    (.["eventing/bucket_op_exception_count"] | add / length | tostring) + 
    "\n eventing/checkpoint_failure_count: " + 
    (.["eventing/checkpoint_failure_count"] | add / length | tostring) + 
    "\n eventing/dcp_backlog: " + 
    (.["eventing/dcp_backlog"] | add / length | tostring) + 
    "\n eventing/failed_count: " + 
    (.["eventing/failed_count"] | add / length | tostring) + 
    "\n eventing/n1ql_op_exception_count: " +
```
Example: Stats for Each Node Separately

```bash
# loop over each of the buckets
for node in $(curl \
    --user Administrator:password \n    --silent \n    --request GET \n    http://localhost:8091/pools/nodes | \n    jq -r '.nodes[] | select(.services | contains(["eventing"]) == true) | .hostname'
)
do
    echo "$node Function Stats"
    echo "-------------------------------------------------------"
    # get the eventing stats for the specific node
    curl \
        --user Administrator:password \n        --silent \n        --request GET \n        --data zoom=minute \n        http://localhost:8091/pools/default/buckets/@eventing/nodes/$node/stats | \n        jq -r '.op.samples | to_entries | sort_by(.key) | [] | select(.key | split("/")) | length == 2) | " " + (.key | split("/"))[1]) + ": " + (.value | add / length | tostring)'
done
```

Key Metrics to Monitor
### Couchbase Metric

<table>
<thead>
<tr>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any exceptions/failures should be monitored</td>
<td>For this value “normal” is 0, any value other than 0 would indicate exceptions are being thrown and should be investigated</td>
</tr>
<tr>
<td>The number of items to be processed.</td>
<td>Create a baseline for this value as “normal” will be dependent upon your workload and number of functions. Alert at 2x of baseline.</td>
</tr>
</tbody>
</table>

### Eventing Function-Level Stats

The Eventing stats for a specific functions are available only once the function has been deployed. The same stats that are available for the service as a whole are also available on a per-function basis and can be retrieved for the entire cluster or a specific node in the cluster.

### Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventing/{function_name}/bucket_op_exception_count</td>
<td>Total number of operations inside of an Eventing function which have resulted in an exception for the function</td>
</tr>
<tr>
<td>eventing/{function_name}/checkpoint_failure_count</td>
<td>Total number of checkpoint failures for the function</td>
</tr>
<tr>
<td>eventing/{function_name}/dcp_backlog</td>
<td>Remaining mutations to process</td>
</tr>
<tr>
<td>eventing/{function_name}/failed_count</td>
<td>Total number of failed Eventing function operations for the function</td>
</tr>
<tr>
<td>eventing/{function_name}/n1ql_op_exception_count</td>
<td>Total number of N1QL operations inside of an Eventing function which have resulted in an exception for the function</td>
</tr>
<tr>
<td>eventing/{function_name}/on_delete_failure</td>
<td>The total number OnDelete handler executions that have failed for the function</td>
</tr>
<tr>
<td>eventing/{function_name}/on_delete_success</td>
<td>Total OnDelete handler executions that have succeeded for the function</td>
</tr>
<tr>
<td>eventing/{function_name}/on_update_failure</td>
<td>Total OnUpdate handler executions that have failed for the function</td>
</tr>
<tr>
<td>eventing/{function_name}/on_update_success</td>
<td>Total OnUpdate handler executions that have failed for the function</td>
</tr>
<tr>
<td>eventing/{function_name}/processed_count</td>
<td>Total number of mutations that have been processed for the function</td>
</tr>
<tr>
<td>eventing/{function_name}/timeout_count</td>
<td>Total number of handler executions that have resulted in a timeout for the function</td>
</tr>
</tbody>
</table>
GET Cluster Eventing Function Stats

These endpoints are informational and should not be used for monitoring as they are an aggregate for the entire and cluster and the best practice is to monitor each node individually.

- Insecure: http://localhost:8091/pools/default/buckets/@eventing/stats
- Secure: https://localhost:18091/pools/default/buckets/@eventing/stats

Example

The following example demonstrates how to retrieve the eventing service stats for the cluster.

```
curl "http://localhost:8091/pools/default/buckets/@eventing/stats"
```

GET Eventing Function Stats per Node

Each node in the cluster running the eventing service should be monitoring individually, although as functions can be dynamic, from a manageability standpoint, it will be easier to monitor the aggregate stats of the service. However, each individual function can be monitored if you so choose.

- Insecure: http://localhost:8091/pools/default/buckets/@eventing/nodes/{NODE}/stats
- Secure: https://localhost:18091/pools/default/buckets/@eventing/nodes/{NODE}/stats

Example

The following example demonstrates how to retrieve the specific eventing function stats for the node.

```
NODE="172.17.0.2:8091"
curl "http://localhost:8091/pools/default/buckets/@eventing/nodes/$NODE/stats"
```
jq -r '.op.samples as $stats' | $stats | [keys | .[] | select(. | split("/")) | length == 3) | split("/")[1] | sort | unique as $funcs | $funcs | .[] | "Function: " + . + "\n----------------------------------------------------------------\n" + "bucket_op_exception_count: " + ($stats["eventing/" + . + "/bucket_op_exception_count"] | add | tostring) + "checkpoint_failure_count: " + ($stats["eventing/" + . + "/checkpoint_failure_count"] | add | tostring) + "dcp_backlog: " + ($stats["eventing/" + . + "/dcp_backlog"] | add | tostring) + "failed_count: " + ($stats["eventing/" + . + "/failed_count"] | add | tostring) + "n1ql_op_exception_count: " + ($stats["eventing/" + . + "/n1ql_op_exception_count"] | add | tostring) + "on_delete_failure: " + ($stats["eventing/" + . + "/on_delete_failure"] | add / length | tostring) + "on_delete_success: " + ($stats["eventing/" + . + "/on_delete_success"] | add / length | tostring) + "on_update_failure: " + ($stats["eventing/" + . + "/on_update_failure"] | add / length | tostring) + "on_update_success: " + ($stats["eventing/" + . + "/on_update_success"] | add / length | tostring) + "processed_count: " + ($stats["eventing/" + . + "/processed_count"] | add / length | tostring) + "timeout_count: " + ($stats["eventing/" + . + "/timeout_count"] | add | tostring) |

Key Metrics to Monitor

<table>
<thead>
<tr>
<th>Couchbase Metric</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventing/{func_name}/bucket_op_exception_count</td>
<td>Any exceptions/failures should be monitored</td>
<td>For this value &quot;normal&quot; is 0, any value other than 0 would indicate exceptions are being thrown and should be monitored</td>
</tr>
<tr>
<td>eventing/{func_name}/checkpoint_failure_count</td>
<td>Any exceptions/failures should be monitored</td>
<td></td>
</tr>
<tr>
<td>eventing/{func_name}/failed_count</td>
<td>Any exceptions/failures should be monitored</td>
<td></td>
</tr>
<tr>
<td>eventing/{func_name}/n1ql_op_exception_count</td>
<td>Any exceptions/failures should be monitored</td>
<td></td>
</tr>
<tr>
<td>eventing/{func_name}/on_delete_failure</td>
<td>Any exceptions/failures should be monitored</td>
<td></td>
</tr>
<tr>
<td>eventing/{func_name}/on_delete_success</td>
<td>Any exceptions/failures should be monitored</td>
<td></td>
</tr>
<tr>
<td>eventing/{func_name}/on_update_failure</td>
<td>Any exceptions/failures should be monitored</td>
<td></td>
</tr>
<tr>
<td>eventing/{func_name}/on_update_success</td>
<td>Any exceptions/failures should be monitored</td>
<td></td>
</tr>
<tr>
<td>eventing/{func_name}/processed_count</td>
<td>Any exceptions/failures should be monitored</td>
<td></td>
</tr>
<tr>
<td>eventing/{func_name}/timeout_count</td>
<td>Any exceptions/failures should be monitored</td>
<td></td>
</tr>
</tbody>
</table>
### Eventing Service Monitoring

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>eventing/{func_name}/on_update_failure</code></td>
<td><strong>Unmonitored</strong></td>
<td><strong>Thrown and should be investigated</strong></td>
</tr>
<tr>
<td><code>eventing/{func_name}/timeout_count</code></td>
<td><strong>Thrown</strong></td>
<td><strong>Not investigated</strong></td>
</tr>
<tr>
<td><code>eventing/{func_name}/dcp_backlog</code></td>
<td><strong>The number of items to be processed.</strong></td>
<td><strong>Create a baseline for this value as &quot;normal&quot; will be dependent upon your workload and number of functions. Alert at 2x of baseline.</strong></td>
</tr>
</tbody>
</table>
Monitoring: Full-Text Search Service

GET Full-Text Search Indexes

Documentation: https://docs.couchbase.com/server/6.0/rest-api/rest-fts-indexing.html#index-definition

http://localhost:8094/api/index

Retrieve all index definitions and configurations

Example

The following example illustrates how to retrieve each FTS index name

```
curl \
  --user Administrator:password \n  --silent \n  --request GET \n  http://localhost:8094/api/index | 
  jq -r '.indexDefs.indexDefs | keys | .[]'
```

FTS Service-Level Stats

Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fts_curr_batches_blocked_by_herder</td>
<td>The number of batches blocked by the herder</td>
</tr>
<tr>
<td>fts_num_bytes_used_ram</td>
<td>The number of bytes used in memory for the FTS service.</td>
</tr>
<tr>
<td>fts_total_queries_rejected_by_herder</td>
<td>The number of queries rejected by the herder</td>
</tr>
</tbody>
</table>

GET Cluster FTS Service Stats

These endpoints are informational and should not be used for monitoring as they are an aggregate for the entire and cluster and the best practice is to monitor each node individually.

- Insecure: http://localhost:8091/pools/default/buckets/@fts/stats
- Secure: https://localhost:18091/pools/default/buckets/@fts/stats

```
curl \
  --user Administrator:password \
```

Couchbase Professional Services
GET Node-Level FTS Service Stats

Each node in the cluster running the FTS service should be monitoring individually using the endpoint listed below.

- Insecure: http://localhost:8091/pools/default/buckets/@fts/nodes/{NODE}/stats
- Secure: https://localhost:18091/pools/default/buckets/@fts/nodes/{NODE}/stats

Example: Stats for Individual Node

The following example demonstrates how to retrieve the FTS service stats for the cluster.

```
NODE="172.17.0.2:8091"
curl \
   --user Administrator:password \
   --silent \
   --request GET \
   --data zoom=minute \
   http://localhost:8091/pools/default/buckets/@fts/nodes/$NODE/stats | \
   jq -r '.op.samples | "fts_num_bytes_used_ram: " + (.fts_num_bytes_used_ram | add / length | tostring)'
```

Example: Stats for Each Node Separately

```
# loop over each of the buckets
for node in $(curl \
   --user Administrator:password \
   --silent \
   --request GET \
   http://localhost:8091/pools/nodes | \
   jq -r '.nodes[] | select(.services | contains(["fts"]) == true) | .hostname')
```

Monitoring: Full-Text Search Service
Individual FTS-Level Stats

The FTS stats for a specific indexes are available only under the bucket that the index is created on. The same stats that are available for the service as a whole are also available on a per-index basis and can be retrieved for the entire cluster or a specific node in the cluster.

Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fts/{indexName}/avg_queries_latency</td>
<td>The average query latency in milliseconds</td>
</tr>
<tr>
<td>fts/{indexName}/doc_count</td>
<td>The number of documents in the index</td>
</tr>
<tr>
<td>fts/{indexName}/num_bytes_used_disk</td>
<td>Total disk file size used by the index</td>
</tr>
<tr>
<td>fts/{indexName}/num_files_on_disk</td>
<td>Number of files for the index on disk</td>
</tr>
<tr>
<td>fts/{indexName}/num_mutations_to_index</td>
<td>The number of documents pending indexing</td>
</tr>
<tr>
<td>fts/{indexName}/num_pindexes_actual</td>
<td>Number of index partitions (including replica partitions)</td>
</tr>
<tr>
<td>fts/{indexName}/num_pindexes_target</td>
<td>Number of index partitions expected (including replica partitions)</td>
</tr>
<tr>
<td>fts/{indexName}/num_recs_toPersist</td>
<td>Number of index records not yet persisted to disk</td>
</tr>
<tr>
<td>fts/{indexName}/num_root_filesegments</td>
<td>The number of root file segments</td>
</tr>
<tr>
<td>fts/{indexName}/num_root_memorysegments</td>
<td>The number of root memory segments</td>
</tr>
<tr>
<td>fts/{indexName}/total_bytes_indexed</td>
<td>Number of fts bytes indexed per second</td>
</tr>
<tr>
<td>fts/{indexName}/total_bytes_query_results</td>
<td>Number of bytes returned in results per second</td>
</tr>
<tr>
<td>fts/{indexName}/total_compaction_written_bytes</td>
<td>Number of compaction bytes written per second</td>
</tr>
<tr>
<td>fts/{indexName}/total_queries</td>
<td>The number of queries per second</td>
</tr>
<tr>
<td>Endpoint</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>fts/[indexName]/total_queries_error</td>
<td>The number of query errors per second</td>
</tr>
<tr>
<td>fts/[indexName]/total_queries_slow</td>
<td>The number of slow queries per second (&gt;5s)</td>
</tr>
<tr>
<td>fts/[indexName]/total_queries_timeout</td>
<td>The number of queries per second that resulted in a timeout</td>
</tr>
<tr>
<td>fts/[indexName]/total_request_time</td>
<td>Total time spent servicing requests</td>
</tr>
<tr>
<td>fts/[indexName]/total_term_searchers</td>
<td>Number of term searchers started per second</td>
</tr>
</tbody>
</table>

**GET Cluster Individual FTS Stats**

These endpoints are informational and should not be used for monitoring as they are an aggregate for the entire and cluster and the best practice is to monitor each node individually.

- **Insecure:** http://localhost:8091/pools/default/buckets/@fts-{BUCKET}/stats
- **Secure:** https://localhost:18091/pools/default/buckets/@fts-{BUCKET}/stats

**Example**

The following example demonstrates how to retrieve the eventing service stats for the cluster.

```
BUCKET="travel-sample"

curl http://localhost:8091/pools/default/buckets/@fts-$BUCKET/stats | 

data zoom=minute | 

dl | 

```

**GET Individual FTS Stats per Node**

Each node in the cluster running the FTS service should be monitoring individually.

- **Insecure:** http://localhost:8091/pools/default/buckets/@fts-{BUCKET}/nodes/{NODE}/stats
- **Secure:** https://localhost:18091/pools/default/buckets/@fts-{BUCKET}/nodes/{NODE}/stats

**Example: Stats for Individual Node**

The following example demonstrates how to retrieve all of the FTS stats for a specific index in a bucket for a specific node.
NODE="172.17.0.2:8091"
BUCKET="travel-sample"
INDEX="demo"

# get the FTS stats for the bucket
curl \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/@fts-$BUCKET/nodes/$NODE/stats \
  jq -r --arg index "$INDEX" '.op.samples | 
  "  avg_queries_latency: " + 
  (.["fts" + $index + "/avg_queries_latency"] | add / length | tostring) + 
  "\n  doc_count: " + 
  (.["fts" + $index + "/doc_count"] | add / length | tostring) + 
  "\n  num_bytes_used_disk: " + 
  (.["fts" + $index + "/num_bytes_used_disk"] | add / length | tostring) + 
  "\n  num_mutations_to_index: " + 
  (.["fts" + $index + "/num_mutations_to_index"] | add | tostring) + 
  "\n  num_pindexes_actual: " + 
  (.["fts" + $index + "/num_pindexes_actual"] | add | tostring) + 
  "\n  num_pindexes_target: " + 
  (.["fts" + $index + "/num_pindexes_target"] | add | tostring) + 
  "\n  num_recs_toPersist: " + 
  (.["fts" + $index + "/num_recs_to_persist"] | add | tostring) + 
  "\n  total_bytes_indexed: " + 
  (.["fts" + $index + "/total_bytes_indexed"] | add / length | tostring) + 
  "\n  total_bytes_query_results: " + 
  (.["fts" + $index + "/total_bytes_query_results"] | add / length | tostring) + 
  "\n  total_compaction_written_bytes: " + 
  (.["fts" + $index + "/total_compaction_written_bytes"] | add / length | tostring) + 
  "\n  total_queries: " + 
  (.["fts" + $index + "/total_queries"] | add | tostring) + 
  "\n  total_queries_error: " + 
  (.["fts" + $index + "/total_queries_error"] | add | tostring) + 
  "\n  total_queries_slow: " + 
  (.["fts" + $index + "/total_queries_slow"] | add | tostring) + 
  "\n  total_queries_timeout: " + 

Example: Stats for Individual Node

The following example demonstrates how to retrieve all of the FTS stats, for every bucket in the cluster for a single node.

```
NODE="172.17.0.2:8091"

# loop over each of the buckets that has indexes
for bucket in $(curl \\
    --user Administrator:password \\
    --silent \\
    --request GET \\
    http://localhost:8094/api/index \\
    jq '.indexDefs.indexDefs | [ to_entries[] | .value.sourceName ] | sort | unique | .[]')
do
    echo ""
    echo "Bucket: $bucket"
    echo "================================================================"
    # get the FTS stats for the bucket
    curl \\
        --user Administrator:password \\
        --silent \\
        --request GET \\
        --data zoom=minute \\
        http://localhost:8091/pools/default/buckets/@fts-$bucket/nodes/$NODE/stats \\
    # 1. reduce the samples object, by looping over each property, only work with properties
    # who are index specific stat properties and either sum or average samples
    # 2. get all of the unique index keys
    # 3. loop over each index and output the stats
    jq -r 'reduce (.op.samples | to_entries[]) as ${key, $value} ({}
        | if ($key | split("/") | length == 3
```
and ($key | contains("replica ") | not)
  ) then
    if ([
      "num_mutations_to_index","num_pindexes_actual",
      "num_pindexes_target","num_recs_to_persist","total_queries",
      "total_queries_error","total_queries_slow","total_queries_timeout",
      "total_request_time+queued","total_term_searchers"
    ] | .[] | contains($key | split("/") | .[2]) == true) then
      .[$key] += ($value | add)
    else
      .[$key] += ($value | add / length | roundit/100.0)
    end
  else
    .
  end
) | . as $stats |
$stats | keys | map(split("/")[1]) | sort | unique as $indexes |
$indexes | .[] |
"Index: " + . + 
"\n---------------------------------------------------------------" + 
"\n avg_queries_latency: 
 + ($stats["fts/" + . + "/avg_queries_latency"] | tostring ) + 
"\n doc_count: 
 + ($stats["fts/" + . + "/doc_count"] | tostring ) + 
"\n num_bytes_used_disk: 
 + ($stats["fts/" + . + "/num_bytes_used_disk"] | tostring ) + 
"\n num_mutations_to_index: 
 + ($stats["fts/" + . + "/num_mutations_to_index"] | tostring ) + 
"\n num_pindexes_actual: 
 + ($stats["fts/" + . + "/num_pindexes_actual"] | tostring ) + 
"\n num_pindexes_target: 
 + ($stats["fts/" + . + "/num_pindexes_target"] | tostring ) + 
"\n num_recs_toPersist: 
 + ($stats["fts/" + . + "/num_recs_toPersist"] | tostring ) + 
"\n total_bytes_indexed: 
 + ($stats["fts/" + . + "/total_bytes_indexed"] | tostring ) + 
"\n total_bytes_query_results: 
 + ($stats["fts/" + . + "/total_bytes_query_results"] | tostring ) + 
"\n total_compaction_written_bytes: 
 + ($stats["fts/" + . + "/total_compaction_written_bytes"] | tostring ) + 
"\n total_queries: 
 + ($stats["fts/" + . + "/total_queries"] | tostring ) + 
"\n total_queries_error: 
 + ($stats["fts/" + . + "/total_queries_error"] | tostring ) +
Key Metrics to Monitor

<table>
<thead>
<tr>
<th>Couchbase Metric</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg_queries_latency</td>
<td>The average query latency</td>
<td>Create a baseline for this value, as &quot;normal&quot; will depend on the size. Alert at 2x of the baseline. This would indicate a slowdown for index scans to the index.</td>
</tr>
<tr>
<td>total_queries</td>
<td>The number of query requests to the index</td>
<td>Create a baseline for this value, as &quot;normal&quot; will depend on the amount. Alert at 2x of the baseline. This would indicate a dramatic increase in requests.</td>
</tr>
<tr>
<td>total_queries_error</td>
<td>The number of query errors to the index</td>
<td>Alert at any value greater than 0 as this indicates failed requests.</td>
</tr>
<tr>
<td>total_queries_timeout</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FTS Aggregate Stats

The FTS aggregate stats for a specific bucket are available only under the bucket that the indexes exist on and are a total of all of the indexes for that bucket in the cluster or node.

Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fts/doc_count</td>
<td>The number of documents in all fts indexes</td>
</tr>
<tr>
<td>fts/num_bytes_used_disk</td>
<td>Total disk file size used by the indexes</td>
</tr>
<tr>
<td>fts/num_files_on_disk</td>
<td>The number of index files on disk</td>
</tr>
<tr>
<td>fts/num_mutations_to_index</td>
<td>The number of documents pending indexing</td>
</tr>
<tr>
<td>fts/num_pindexes_actual</td>
<td>Number of index partitions (including replica partitions)</td>
</tr>
<tr>
<td>fts/num_pindexes_target</td>
<td>Number of index partitions expected (including replica partitions)</td>
</tr>
<tr>
<td>fts/num_recs_to_persist</td>
<td>Number of index records not yet persisted to disk</td>
</tr>
<tr>
<td>fts/num_root_filesegments</td>
<td>Number of root file segments</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>fts/num_root_filesegments</td>
<td>Number of root file segments</td>
</tr>
<tr>
<td>fts/num_root_memorysegments</td>
<td>Number of root memory segments</td>
</tr>
<tr>
<td>fts/total_bytes_indexed</td>
<td>Number of bytes indexed per second</td>
</tr>
<tr>
<td>fts/total_bytes_query_results</td>
<td>Number of bytes returned in results per second</td>
</tr>
<tr>
<td>fts/total_compaction_written_bytes</td>
<td>Number of compaction bytes written per second</td>
</tr>
<tr>
<td>fts/total_queries</td>
<td>The number of queries per second</td>
</tr>
<tr>
<td>fts/total_queries_error</td>
<td>The number of query errors per second</td>
</tr>
<tr>
<td>fts/total_queries_slow</td>
<td>The number of slow queries per second (&gt;5s)</td>
</tr>
<tr>
<td>fts/total_queries_timeout</td>
<td>The number of queries per second that resulted in a timeout</td>
</tr>
<tr>
<td>fts/total_request_time</td>
<td></td>
</tr>
<tr>
<td>fts/total_term_searchers</td>
<td>Number of term searchers started per second</td>
</tr>
</tbody>
</table>

**GET  Cluster FTS Aggregate Stats**

- Insecure: `http://localhost:8091/pools/default/buckets/@fts-{BUCKET}/stats`
- Secure: `https://localhost:18091/pools/default/buckets/@fts-{BUCKET}/stats`

**Example: Stats for Cluster**

The following example demonstrates how to retrieve all of the fts aggregate stats for a specific bucket in the entire cluster.

```bash
BUCKET="travel-sample"

# get the FTS stats for the bucket

curl \   
    --user Administrator:password \   
    --silent \   
    --request GET \   
    --data zoom=minute \   
    http://localhost:8091/pools/default/buckets/@fts-$BUCKET/stats | \   
    jq -r ".op.samples | " | add | length | tostring) +
        "\n    doc_count: " + (.["fts/doc_count"] | add / length | tostring) +
        "\n    num_bytes_used_disk: " + (.["fts/num_bytes_used_disk"] | add / length | tostring) +
        "\n    num_mutations_to_index: " + (.["fts/num_mutations_to_index"] | add / length | tostring) +
        "\n    num_pindexes_actual: " + (.["fts/num_pindexes_actual"] | add | tostring) +
        "\n    num_pindexes_target: " + (.["fts/num_pindexes_target"] | add / length | tostring) +
        "\n    total_bytes_indexed: " + (.["fts/total_bytes_indexed"] | add / length | tostring) +
```

Monitoring: Full-Text Search Service
GET FTS Aggregate Stats per Node

- Insecure: http://localhost:8091/pools/default/buckets/@fts-{BUCKET}/nodes/{NODE}/stats
- Secure: https://localhost:18091/pools/default/buckets/@fts-{BUCKET}/nodes/{NODE}/stats

Example: Aggregate Stats for Individual Node

The following example demonstrates how to retrieve all of the index aggregate stats for a specific in a bucket for a specific node.

```
BUCKET="travel-sample"
NODE="172.17.0.2:8091"

# get the FTS stats for the bucket

curl "
   --user Administrator:password 
   --silent 
   --request GET 
   --data zoom=minute 
   http://localhost:8091/pools/default/buckets/@fts-$BUCKET/nodes/$NODE/stats
"

   | jq -r ".op.samples | 
   " doc_count: " + (."fts/doc_count" | add / length | tostring) + 
   " num_bytes_used_disk: " + (."fts/num_bytes_used_disk" | add / length | tostring) + 
   " num_mutations_to_index: " + (."fts/num_mutations_to_index" | add / length | tostring) + 
   " num_pindexes_actual: " + (."fts/num_pindexes_actual" | add | tostring) +
```

Monitoring: Full-Text Search Service
Monitoring: Full-Text Search Service

```plaintext
ng) +
  "\n  num_pindexes_target: " + (.["fts/num_pindexes_target"] | add / length | tostring) +
  "\n  total_bytes_indexed: " + (.["fts/total_bytes_indexed"] | add / length | tostring) +
  "\n  total_bytes_query_results: " + (.["fts/total_bytes_query_results"] | add / length | tostring) +
  "\n  total_compaction_written_bytes: " + (.["fts/total_compaction_written_bytes"] | add / length | tostring) +
  "\n  total_queries: " + (.["fts/total_queries"] | add / length | tostring) +
  "\n  total_queries_error: " + (.["fts/total_queries_error"] | add / length | tostring) +
  "\n  total_queries_slow: " + (.["fts/total_queries_slow"] | add / length | tostring) +
  "\n  total_queries_timeout: " + (.["fts/total_queries_timeout"] | add / length | tostring) +
  "\n  total_request_time: " + (.["fts/total_request_time"] | add / tostring) +
  "\n  total_term_searchers: " + (.["fts/total_term_searchers"] | add / tostring)'
```
Monitoring: Index Service

Index Status

The index status API displays all index definitions, node placement and status within the cluster.

- Insecure: http://localhost:8091/indexStatus
- Secure: https://localhost:18091/indexStatus

Response:

```json
{
  "indexes": [
    {
      "storageMode": "plasma",
      "partitioned": false,
      "instId": 4607548507687231469,
      "hosts": ["127.0.0.1:8091"],
      "progress": 100,
      "definition": "CREATE INDEX `def_airportname` ON `travel-sample`(`airportname`) WITH { "defer_build":true },
      "status": "Ready",
      "bucket": "travel-sample",
      "index": "def_airportname",
      "id": 15764219156300962421
    },
    {
      "storageMode": "plasma",
      "partitioned": false,
      "instId": 11862384293590784556,
      "hosts": ["127.0.0.1:8091"],
      "progress": 100,
      "definition": "CREATE INDEX `def_city` ON `travel-sample`(`city`) WITH { "defer_build":true },
      "status": "Ready",
      "bucket": "travel-sample",
      "index": "def_city",
      "id": 2037567312091921182
    }
  ],
  "version": 45110879,
  "warnings": []
}
```

Key Metrics to Monitor
<table>
<thead>
<tr>
<th>Couchbase Metric</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Indicates whether a index is in a “Ready” or “Building” state.</td>
<td>Alert if the value is not “Ready” or “Building”.</td>
</tr>
</tbody>
</table>

**Example**

The following example illustrates outputting each Index Name and Status.

```
curl \
    --user Administrator:password \
    --silent \
    --request GET \
    http://localhost:8091/indexStatus | \
    jq -r '.indexes | sort_by(.bucket) | .[] | .bucket + ": " + .index + " (" + .status + ")"'
```

This example shows outputting all indexes whose status is not “Ready” or “Building”

```
curl \
    --user Administrator:password \
    --silent \
    --request GET \
    http://localhost:8091/indexStatus | \
    jq -r '.indexes | map(select(\n          (.status != "Ready" and .status != "Building")\n      )) | .[] | .bucket + ": " + .index + " (" + .status + ")"
```

**Index Service-Level Stats**

The following Index service stats are available via the Cluster-Wide or Per-Node Endpoints listed below.

**Available Stats**

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index_memory_quota</td>
<td>The cluster wide memory quota.</td>
</tr>
<tr>
<td>index_memory_used</td>
<td>The amount of memory currently used by the indexing service.</td>
</tr>
<tr>
<td>index_ram_percent</td>
<td>The percentage of index entries in ram.</td>
</tr>
<tr>
<td>index_remaining_ram</td>
<td>The amount of memory remaining.</td>
</tr>
</tbody>
</table>
`GET` **Cluster Index Service Stats**

These endpoints are informational and should not be used for monitoring as they are an aggregate for the entire and cluster and the best practice is to monitor each node individually.

- **Insecure**: http://localhost:8091/pools/default/buckets/@index/stats
- **Secure**: https://localhost:18091/pools/default/buckets/@index/stats

**Example**

```bash
curl \
  --user Administrator:password \ 
  --silent \ 
  --request GET \ 
  --data zoom=minute \ 
  http://localhost:8091/pools/default/buckets/@index/stats | \ 
  jq -r ".op.samples | to_entries[] | select(.key != "timestamp") | .key + ": " + (.value | add / length | tostring)"
```

`GET` **Node-Level Index Service Stats**

Each node in the cluster running the index service should be monitoring individually using the endpoint listed below.

- **Insecure**: http://localhost:8091/pools/default/buckets/@index/nodes/{NODE}/stats
- **Secure**: https://localhost:18091/pools/default/buckets/@index/nodes/{NODE}/stats

**Example: Stats for Individual Node**

The following example demonstrates how to retrieve the index service stats for a specific node.

```bash
NODE="172.17.0.2:8091"

curl \
  --user Administrator:password \ 
  --silent \ 
  --request GET \ 
  --data zoom=minute \ 
  http://localhost:8091/pools/default/buckets/@index/nodes/$NODE/stats | \ 
  jq -r -c ".op.samples | 
  " index_memory_quota: " + (.index_memory_quota | add / length | tostring) + 
  
  index_memory_used: " + (.index_memory_used | add / length | tostring) + 
  
  index_ram_percent: " + (.index_ram_percent | add / length | tostring)"
```
Example: Stats for Each Node Separately

```bash
# loop over each of the buckets
for node in $(curl \
    --user Administrator:password \
    --silent \
    --request GET \
    http://localhost:8091/pools/nodes | \
    jq -r '.nodes[] | select(.services | contains(["index"])) == true) | .hostname' )
do
echo "$node Index Stats"
echo "-------------------------------------------------------"
# get the index stats for the specific node
curl \
    --user Administrator:password \
    --silent \
    --request GET \
    --data zoom=minute \n    http://localhost:8091/pools/default/buckets/@index/nodes/$node/stats |
    jq -r '.op.samples | to_entries[] | select(.key != "timestamp") | .key + ": " + (.value | add / length | tostring)' done
```

Key Metrics to Monitor

<table>
<thead>
<tr>
<th>Couchbase Metric</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>index_remaining_ram</td>
<td>The amount of memory remaining.</td>
<td>Alert if this value is 20% or less, as it is an indicative of index growth and new index nodes will need to be expanded.</td>
</tr>
</tbody>
</table>

Individual Index-Level Stats

The Index stats for a specific indexes are available only under the bucket that the index is created on. The same stats that are available for the service as a whole are also available on a per-index basis and can be retrieved for the entire cluster or a specific node in the cluster.
## Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index/{indexName}/avg_item_size</td>
<td>The average index entry size</td>
</tr>
<tr>
<td>index/{indexName}/avg_scan_latency</td>
<td>The average latency when scanning the index</td>
</tr>
<tr>
<td>index/{indexName}/cache_hits</td>
<td>The number of in-memory hits to the index</td>
</tr>
<tr>
<td>index/{indexName}/cache_miss_ratio</td>
<td>The ratio of misses to hits</td>
</tr>
<tr>
<td>index/{indexName}/cache_misses</td>
<td>The number of in-memory misses to the index</td>
</tr>
<tr>
<td>index/{indexName}/data_size</td>
<td>The total data size of the index</td>
</tr>
<tr>
<td>index/{indexName}/data_size_on_disk</td>
<td>The total size of the index data on disk</td>
</tr>
<tr>
<td>index/{indexName}/disk_overhead_estimate</td>
<td>The size of stale data on disk due to fragmentation</td>
</tr>
<tr>
<td>index/{indexName}/disk_size</td>
<td>The size of the index on disk</td>
</tr>
<tr>
<td>index/{indexName}/frag_percent</td>
<td>The index fragmentation percentage</td>
</tr>
<tr>
<td>index/{indexName}/index_frag_percent</td>
<td>The index fragmentation percentage</td>
</tr>
<tr>
<td>index/{indexName}/index_resident_percent</td>
<td>The percentage of the index that is memory resident</td>
</tr>
<tr>
<td>index/{indexName}/items_count</td>
<td>The number of items in the index</td>
</tr>
<tr>
<td>index/{indexName}/log_space_on_disk</td>
<td>The size of the log files on disk</td>
</tr>
<tr>
<td>index/{indexName}/memory_used</td>
<td>The amount of memory used by the index</td>
</tr>
<tr>
<td>index/{indexName}/num_docs_indexed</td>
<td>The number of items indexed since the last restart</td>
</tr>
<tr>
<td>index/{indexName}/num_docs_pending</td>
<td>The number of items pending indexing</td>
</tr>
<tr>
<td>index/{indexName}/num_docs_pending+queued</td>
<td>The number of documents that are pending or queued for indexing</td>
</tr>
<tr>
<td>index/{indexName}/num_docs_queued</td>
<td>The number of documents that are queued for indexing</td>
</tr>
<tr>
<td>index/{indexName}/num_requests</td>
<td>The number of requests to the index</td>
</tr>
<tr>
<td>index/{indexName}/num_rows_returned</td>
<td>The average number of rows returned by a scan</td>
</tr>
<tr>
<td>index/{indexName}/raw_data_size</td>
<td>The raw uncompressed data size</td>
</tr>
<tr>
<td>index/{indexName}/recs_in_mem</td>
<td>The number of records in the index that are in memory</td>
</tr>
<tr>
<td>index/{indexName}/recs_on_disk</td>
<td>The number of records not in memory</td>
</tr>
<tr>
<td>index/{indexName}/scan_bytes_read</td>
<td>The average number of bytes read per scan</td>
</tr>
<tr>
<td>index/{indexName}/total_scan_duration</td>
<td>The total time spent scanning</td>
</tr>
</tbody>
</table>

## GET Cluster Individual Index Stats
These endpoints are informational and should not be used for monitoring as they are an aggregate for the entire and cluster and the best practice is to monitor each node individually.

- Insecure: http://localhost:8091/pools/default/buckets/@index-{BUCKET}/stats
- Secure: https://localhost:18091/pools/default/buckets/@index-{BUCKET}/stats

Example

The following example demonstrates how to retrieve the eventing service stats for the cluster.

```bash
BUCKET="travel-sample"
curl \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/@index-$BUCKET/stats | \
  jq -r '.op.samples | to_entries | sort_by(.key) | .[] | 
    select(.key | split("/") | length == 3) | 
    " " + (.key) + ": " + 
    (.value | add / length | tostring)'
```

GET Individual Index Stats per Node

Each node in the cluster running the index service should be monitoring individually.

- Insecure: http://localhost:8091/pools/default/buckets/@index-{BUCKET}/nodes/{NODE}/stats
- Secure: https://localhost:18091/pools/default/buckets/@index-{BUCKET}/nodes/{NODE}/stats

Example: Stats for Individual Node

The following example demonstrates how to retrieve all of the index stats for a specific index in a bucket for a specific node.

```bash
NODE="172.17.0.2:8091" 
BUCKET="travel-sample" 
INDEX="def_faa"

# get the index stats for the bucket
curl \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/@index-$BUCKET/nodes/$NODE/stats
```
ts | 
jq -r --arg index "INDEX" '.op.samples |
" avg_item_size: " + (."index" + $index + "/avg_item_size"] | add / length | tostring) +


\n avg_scan_latency: " + (."index" + $index + "/avg_scan_latency"] | add / length | tostring) +
\n cache_hits: " + (."index" + $index + "/cache_hits"] | add | tostring) +
\n cache_misses: " + (."index" + $index + "/cache_misses"] | add | tostring) +
\n data_size: " + (."index" + $index + "/data_size"] | add / length | tostring) +
\n disk_overhead_estimate: " + (."index" + $index + "/disk_overhead_estimate"] | add / length | tostring) +
\n disk_size: " + (."index" + $index + "/disk_size"] | add / length | tostring) +
\n frag_percent: " + (."index" + $index + "/frag_percent"] | add / length | tostring) +
\n index_frag_percent: " + (."index" + $index + "/index_frag_percent"] | add / length | tostring) +
\n index_resident_percent: " + (."index" + $index + "/index_resident_percent"] | add / length | tostring) +
\n items_count: " + (."index" + $index + "/items_count"] | add / length | tostring) +
\n memory_used: " + (."index" + $index + "/memory_used"] | add / length | tostring) +
\n num_docs_indexed: " + (."index" + $index + "/num_docs_indexed"] | add | tostring) +
\n num_docs_pending+queued: " + (."index" + $index + "/num_docs_pending+queued"] | add | tostring) +
\n num_docs_queued: " + (."index" + $index + "/num_docs_queued"] | add | tostring) +
\n num_requests: " + (."index" + $index + "/num_requests"] | add | tostring) +
\n num_rows_returned: " + (."index" + $index + "/num_rows_returned"] | add | tostring) +
\n recs_in_mem: " + (."index" + $index + "/recs_in_mem"] | add / length | tostring) +
\n recs_on_disk: " + (."index" + $index + "/recs_on_disk"] | add / length | tostring) +
\n scan_bytes_read: " + (."index" + $index + "/scan_bytes_read"] | add | tostring) +
\n total_scan_duration: " + (."index" + $index + "/total_scan_duration "]) | add | tostring)
Example: Stats for Individual Node

The following example demonstrates how to retrieve all of the index stats, for every bucket in the cluster for a single node.

```
NODE="172.17.0.2:8091"

# loop over each of the buckets that has indexes
for bucket in $(curl
  --user Administrator:password
  --silent
  --request GET
  http://localhost:8091/indexStatus |
  jq -r '[ .indexes[] | .bucket ] | sort | unique | .[]')
do
  echo ""
  echo "Bucket: $bucket"
  echo "================================================================"
  # get the index stats for the bucket
  curl
    --user Administrator:password
    --silent
    --request GET
    --data zoom=minute
    http://localhost:8091/pools/default/buckets/@index-$bucket/nodes/$NODE/stats |
  # 1. reduce the samples object, by looping over each property, only work with properties
  # who are index specific stat properties and either sum or average samples
  # 2. get all of the unique index keys
  # 3. loop over each index and output the stats
  jq -r 'reduce (.op.samples | to_entries[]) as {$key, $value} ({}
    | if ($key | split("/") | length == 3
      and ($key | contains("replica ")) | not
    )
    then
      if ([$
        "cache_hits","cache_misses","num_docs_indexed","num_docs_pending",
        "num_docs_pending+queued","num_docs_queued","num_requests",
      ]
  ```
Monitoring: Index Service

"num_rows Returned","scan_bytes_read","total_scan_duration"
] | .[] | contains($key | split("/" | .[2]) == true) then
  .$key] += ($value | add)
else
  .$key] += ($value | add / length | roundit/100.0)
end
else

end
) | . as $stats |
$stats | keys | map(split("/"[1]) | sort | unique as $indexes |
$indexes | .[] |
"Index: " + . +
"\n---------------------------------------------------------------" +
"\n avg_item_size: " + ($stats["index/" + . + "/avg_item_size"] | tostring ) +
"\n avg_scan_latency: " + ($stats["index/" + . + "/avg_scan_latency "] | tostring) +
"\n cache_hits: " + ($stats["index/" + . + "/cache_hits"] | tostring) +
"\n cache_miss_ratio: " + ($stats["index/" + . + "/cache_miss_ratio "] | tostring) +
"\n cache_misses: " + ($stats["index/" + . + "/cache_misses"] | tostring) +
"\n data_size: " + ($stats["index/" + . + "/data_size"] | tostring) +
"\n disk_overhead_estimate: " + ($stats["index/" + . + "/disk_overhead_estimate"] | tostring) +
"\n disk_size: " + ($stats["index/" + . + "/disk_size"] | tostring) +
"\n frag_percent: " + ($stats["index/" + . + "/frag_percent"] | tostring) +
"\n index_frag_percent: " + ($stats["index/" + . + "/index_frag_percent"] | tostring) +
"\n index_resident_percent: " + ($stats["index/" + . + "/index_resident_percent"] | tostring) +
"\n items_count: " + ($stats["index/" + . + "/items_count"] | tostring) +
"\n memory_used: " + ($stats["index/" + . + "/memory_used"] | tostring) +
"\n num_docs_indexed: " + ($stats["index/" + . + "/num_docs_indexed "] | tostring) +
"\n num_docs_pending: " + ($stats["index/" + . + "/num_docs_pending "] | tostring) +
"\n num_docs_pending+queued: " + ($stats["index/" + . + "/num_docs_ pending+queued"] | tostring) +
Key Metrics to Monitor

<table>
<thead>
<tr>
<th>Couchbase Metric</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg_item_size</td>
<td>The average index entry size</td>
<td>Create a baseline for this value, as “normal” will depend on the size. Alert at 2x of the baseline. This would indicate a dramatic model change.</td>
</tr>
<tr>
<td>avg_scan_latency</td>
<td>The average scan latency</td>
<td>Create a baseline for this value, as “normal” will depend on the size. Alert at 2x of the baseline. This would indicate a slowdown for index scans to the index.</td>
</tr>
<tr>
<td>index_resident_percent</td>
<td>The percentage of the index that is memory resident</td>
<td>Create a baseline for this value as “normal” will depend on SLAs and hard configuration. Alert at 5-10% deviation of the baseline.</td>
</tr>
<tr>
<td>num_requests</td>
<td>The number of index scan requests to the index</td>
<td>Create a baseline for this value, as “normal” will depend on the amount. Alert at 2x of the baseline. This would indicate a dramatic increase in requests.</td>
</tr>
</tbody>
</table>

Index Aggregate Stats

The Index aggregate stats for a specific bucket are available only under the bucket that the indexes exist on and are a total of all of the indexes for that bucket in the cluster or node.

Available Stats
<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index/cache_hits</td>
<td>The number of in-memory hits to the index</td>
</tr>
<tr>
<td>index/cache_misses</td>
<td>The number of in-memory misses to the index</td>
</tr>
<tr>
<td>index/data_size</td>
<td>The total data size of the index</td>
</tr>
<tr>
<td>index/data_size_on_disk</td>
<td>The total data size on disk</td>
</tr>
<tr>
<td>index/disk_overhead_estimate</td>
<td>The size of stale data on disk due to fragmentation</td>
</tr>
<tr>
<td>index/disk_size</td>
<td>The size of the index on disk</td>
</tr>
<tr>
<td>index/frag_percent</td>
<td>The index fragmentation percentage</td>
</tr>
<tr>
<td>index/fragmentation</td>
<td>The index fragmentation percentage</td>
</tr>
<tr>
<td>index/items_count</td>
<td>The number of items in the index</td>
</tr>
<tr>
<td>index/memory_used</td>
<td>The amount of memory used by the index</td>
</tr>
<tr>
<td>index/num_docs_indexed</td>
<td>The number of items indexed since the last restart</td>
</tr>
<tr>
<td>index/num_docs_pending</td>
<td>The number of documents that are pending or queued for indexing</td>
</tr>
<tr>
<td>index/num_docs_queued</td>
<td>The number of documents that are queued for indexing</td>
</tr>
<tr>
<td>index/num_requests</td>
<td>The number of requests to the index</td>
</tr>
<tr>
<td>index/num_rows_returned</td>
<td>The average number of rows returned by a scan</td>
</tr>
<tr>
<td>index/raw_data_size</td>
<td>The raw uncompressed data size</td>
</tr>
<tr>
<td>index/recs_in_mem</td>
<td>The number of records in the index that are in memory</td>
</tr>
<tr>
<td>index/recs_on_disk</td>
<td>The number of records not in memory</td>
</tr>
<tr>
<td>index/scan_bytes_read</td>
<td>The average number of bytes read per scan</td>
</tr>
<tr>
<td>index/total_scan_duration</td>
<td>The total time spent scanning</td>
</tr>
</tbody>
</table>

**GET Cluster Index Aggregate Stats**

- Insecure: [http://localhost:8091/pools/default/buckets/@index-{BUCKET}/stats](http://localhost:8091/pools/default/buckets/@index-{BUCKET}/stats)
- Secure: [https://localhost:18091/pools/default/buckets/@index-{BUCKET}/stats](https://localhost:18091/pools/default/buckets/@index-{BUCKET}/stats)

**Example: Stats for Cluster**

The following example demonstrates how to retrieve all of the index aggregate stats for a specific bucket in the entire cluster.

```bash
BUCKET="travel-sample"

# get the index stats for the bucket
curl \
  --user Administrator:password \
  --silent \
  --request GET \
```

Couchbase Professional Services
--data zoom=minute \
http://localhost:8091/pools/default/buckets/@index-$BUCKET/stats | \
jq -r '.op.samples | to_entries | sort_by(.key) | .[] | select(.key | split("/") | length == 2) | " " + (.key | split("/"))[1]) + " : " + (.value | add / length | tostring)'

GET Index Aggregate Stats per Node

- Insecure: http://localhost:8091/pools/default/buckets/@index-{BUCKET}/nodes/{NODE}/stats
- Secure: https://localhost:18091/pools/default/buckets/@index-{BUCKET}/nodes/{NODE}/stats

Example: Aggregate Stats for Individual Node

The following example demonstrates how to retrieve all of the index aggregate stats for a specific in a bucket for a specific node.

```
BUCKET="travel-sample" 
NODE="172.17.0.2:8091"

# get the index stats for the bucket

curl \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/@index-$BUCKET/nodes/$NODE/stats | \
  jq -r '.op.samples |
    " cache_hits: " + (."index/cache_hits"] | add | tostring) +
    "\n    cache_misses: " + (."index/cache_misses"] | add | tostring) +
    "\n    data_size: " + (."index/data_size"] | add | tostring) +
    "\n    disk_overhead_estimate: " + (."index/disk_overhead_estimate"] | add /
    length | tostring) +
    "\n    disk_size: " + (."index/disk_size"] | add | tostring) +
    "\n    frag_percent: " + (."index/frag_percent"] | add / length | tostring) +
    "\n    fragmentation: " + (."index/fragmentation"] | add / length | tostring) +
    "\n    items_count: " + (."index/items_count"] | add / length | tostring) +
    "\n    memory_used: " + (."index/memory_used"] | add / length | tostring) +
    "\n    num_docs_indexed: " + (."index/num_docs_indexed"] | add | tostring) +
    "\n    num_docs_pending: " + (."index/num_docs_pending"] | add | tostring)+
```
Monitoring: Index Service

```

\n num_docs_queued: " + (."index/num_docs_queued" | add | tostring) +
\n num_requests: " + (."index/num_requests" | add | tostring) +
\n num_rows_returned: " + (."index/num_rows Returned" | add | tostring ) +
\n recs_in_mem: " + (."index/recs in mem" | add | tostring) +
\n recs_on_disk: " + (."index/recs on disk" | add | tostring) +
\n scan_bytes_read: " + (."index/scan_bytes_read" | add | tostring) +
\n total_scan_duration: " + (."index/total_scan_duration" | add | tostring)
`

Couchbase Professional Services

52
Built-in Email Alerts and Logs

Couchbase provides several built-in alerts for when Couchbase is approaching a critical failure or when a critical failure has occurred. It is recommended to enable the built-in email alerts and configure them to be sent to multiple recipients or a distribution list. These alerts should be treated as a fail-safe to proactive alerting from an external monitoring service.

Some environments do not permit Couchbase nodes to send email. This table provides the log-based equivalent of the built-in Couchbase email alerts.

Logs can be monitored via REST using the https://<server>:8091/logs endpoint or via the /opt/couchbase/var/lib/couchbase/logs/info.log file. Alerts can be generated by applying a regular expression to match either the module/code combination or string noted below.

### Available Alerts

<table>
<thead>
<tr>
<th>Alert</th>
<th>Description</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node was auto-failed-over</td>
<td>The sending node has been failed over automatically.</td>
<td>auto_failover_node</td>
</tr>
<tr>
<td>Maximum number of auto-failed-over nodes was reached</td>
<td>The auto-failover system stops auto-failover when the maximum number of spare nodes available has been reached.</td>
<td>auto_failover_maximum_reached</td>
</tr>
<tr>
<td>Node wasn't auto-failed-over as other nodes are down at the same time</td>
<td>Auto-failover does not take place if there is already a node down.</td>
<td>auto_failover_other_nodes_down</td>
</tr>
<tr>
<td>Node was not auto-failed-over as there are not enough nodes in the cluster running the same service</td>
<td>You cannot support auto-failover with less than three nodes.</td>
<td>auto_failover_cluster_too_small</td>
</tr>
<tr>
<td>Node was not auto-failed-over as auto-failover for one or more services running on the node is disabled</td>
<td>Auto-failover does not take place on a node as one or more services running on the node is disabled.</td>
<td>auto_failover_disabled</td>
</tr>
<tr>
<td>Node’s IP address has changed unexpectedly</td>
<td>The IP address of the node has changed, which may indicate a network interface, operating system, or other network or system failure.</td>
<td>ip</td>
</tr>
<tr>
<td>Disk space used for persistent</td>
<td>The disk device configured for</td>
<td></td>
</tr>
</tbody>
</table>
### Logs API

The same log file messages that are available in the Admin UI [http://localhost:8091/ui/index.html#!/logs](http://localhost:8091/ui/index.html#!/logs) are available via a REST API as well.

- **Insecure**: [http://localhost:8091/logs](http://localhost:8091/logs)
- **Secure**: [https://localhost:18091/logs](https://localhost:18091/logs)

### API Parameters

The Logs API supports the following query string parameters

<table>
<thead>
<tr>
<th>Param</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>An integer greater than 0 that limits the overall number of messages returned</td>
</tr>
<tr>
<td>sinceTime</td>
<td>Epoch timestamp in milliseconds to start returning messages from</td>
</tr>
</tbody>
</table>

---

### Storage

- **Storage has reach at least 90% of capacity**
  - Storage of persistent data is nearing full capacity.
  - **disk**

- **Metadata overhead is more than 50%**
  - The amount of data required to store the metadata information for your dataset is now greater than 50% of the available RAM.
  - **overhead**

- **Bucket memory on a node is entirely used for metadata**
  - All the available RAM on a node is being used to store the metadata for the objects stored. This means that there is no memory available for caching values. With no memory left for storing metadata, further requests to store data will also fail.
  - Only applicable to buckets configured for value-only ejection.
  - **ep_oom_errors**

- **Writing data to disk for a specific bucket has failed**
  - The disk or device used for persisting data has failed to store persistent data for a bucket.
  - **ep_item_commit_failed**

- **Writing event to audit log has failed**
  - The audit log event writing has failed.
  - **audit_dropped_events**

- **Approaching full Indexer RAM warning**
  - The indexer RAM limit threshold is approaching warning.
  - **indexer_ram_max_usage**

- **Remote mutation timestamp exceeded drift threshold**
  - The remote mutation timestamp exceeded drift threshold warning.
  - **ep_clock_cas_drift_threshold_exceeded**

- **Communication issues among some nodes in the cluster**
  - There are some communication issues in some nodes within the cluster.
  - **communication_issue**

---

**Logs API**

The same log file messages that are available in the Admin UI [http://localhost:8091/ui/index.html#!/logs](http://localhost:8091/ui/index.html#!/logs) are available via a REST API as well.

- **Insecure**: [http://localhost:8091/logs](http://localhost:8091/logs)
- **Secure**: [https://localhost:18091/logs](https://localhost:18091/logs)

**API Parameters**

The Logs API supports the following query string parameters

<table>
<thead>
<tr>
<th>Param</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>An integer greater than 0 that limits the overall number of messages returned</td>
</tr>
<tr>
<td>sinceTime</td>
<td>Epoch timestamp in milliseconds to start returning messages from</td>
</tr>
</tbody>
</table>
## Log Response Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>A code specified by the module or 0</td>
</tr>
<tr>
<td>module</td>
<td>The module that generated the log message</td>
</tr>
<tr>
<td>node</td>
<td>The node that the message came from</td>
</tr>
<tr>
<td>serverTime</td>
<td>An ISO-8601 timestamp of when the message was logged</td>
</tr>
<tr>
<td>shortText</td>
<td>A short string describing the log entry, most commonly &quot;message&quot;, &quot;node up&quot;, or &quot;node down&quot;</td>
</tr>
<tr>
<td>text</td>
<td>The detailed log message</td>
</tr>
<tr>
<td>tstamp</td>
<td>An Epoch timestamp of when the message was logged</td>
</tr>
<tr>
<td>type</td>
<td>The type of log message, values can be: info, warning, critical</td>
</tr>
</tbody>
</table>

### Example: All Log Messages

```bash
curl \
  --user Administrator:password \ 
  --silent \ 
  --request GET \ 
  --data limit=100 \ 
  http://localhost:8091/logs | \ 
  jq -r ".list[] | "[" + .type + "] " + .serverTime + 
  " Module: " + .module + 
  " Code: " + (.code | tostring) + 
  " Message: " + .text 
```

### Example: Critical Messages Only

```bash
curl \
  --user Administrator:password \ 
  --silent \ 
  --request GET \ 
  --data limit=100 \ 
  http://localhost:8091/logs | \ 
  jq -r ".list[] | select(.type == "critical") | "[" + .type + "] " + .serverTime + 
  " Module: " + .module + 
  " Code: " + (.code | tostring) + 
  " Message: " + .text 
```
Example: Warning Messages Only

curl \ 
  --user Administrator:password \ 
  --silent \ 
  --request GET \ 
  --data limit=100 \ 
  http://localhost:8091/logs | \ 
  jq -r '.list[] | select(.type == "warning") | 
  
  "[" + .type + "] " + .serverTime + 
  " Module: " + .module + 
  " Code: " + (.code | tostring) + 
  " Message: " + .text

Example: Critical or Warning Messages Only

curl \ 
  --user Administrator:password \ 
  --silent \ 
  --request GET \ 
  --data limit=100 \ 
  http://localhost:8091/logs | \ 
  jq -r '.list[] | select(.type == "critical" or .type == "warning") | 
  
  "[" + .type + "] " + .serverTime + 
  " Module: " + .module + 
  " Code: " + (.code | tostring) + 
  " Message: " + .text

Alerts API

Critical alerts that trigger email alerts, are also displayed to users in the Admin UI upon logging in. These alerts can optionally be monitored, should email not be an option.

- Insecure: http://localhost:8091/pools/default
- Secure: https://localhost:18091/pools/default

Alerts are located at the root of the response payload in a property "alerts", which is an array.

Alert Properties
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>The alert message and details</td>
</tr>
<tr>
<td>serverTime</td>
<td>The time the alert was issued</td>
</tr>
</tbody>
</table>

**Example: Retrieve All Alerts**

```bash
curl \
   --user Administrator:password \
   --silent \
   --request GET \
   http://localhost:8091/pools/default | \
   jq -r '.alerts[] | .serverTime + " - " + .msg'
```
Monitoring: Nodes

GET Nodes Overview

http://localhost:8091/pools/nodes

Documentation: https://docs.couchbase.com/server/6.0/rest-api/rest-node-get-info.html

Response

```
{
  "nodes": [
    {
      "hostname": "10.112.170.101:8091",
      "thisNode": true,
      "ports": {
        "sslProxy": 11214,
        "httpsMgmt": 18091,
        "httpsCAPI": 18092,
        "proxy": 11211,
        "direct": 11210
      },
      "services": ["fts", "index", "kv", "n1ql", "cbas", "eventing"]
    }
  ],
  "services": ["fts", "index", "kv", "n1ql", "cbas", "eventing"]
}
```

Each node in the cluster is listed in the "nodes" array. The thisNode attribute indicates the node you have executed the query against. Using this output, a monitoring agent can discover new nodes within the cluster and which services are assigned to those nodes in order to automatically apply the correct monitoring profile.

Key Metrics to Monitor

<table>
<thead>
<tr>
<th>Couchbase Metric</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>This is a meta metric that indicates overall node health.</td>
<td>Alert if the value is &quot;unhealthy&quot;.</td>
</tr>
<tr>
<td>clusterMembership</td>
<td>Indicates whether the node is an active participant in cluster operations. Possible values are &quot;active&quot;, &quot;inactiveAdded&quot;, and &quot;inactiveFailed&quot;.</td>
<td>Alert on &quot;inactiveFailed&quot; and investigate the cause of the node failure.</td>
</tr>
</tbody>
</table>

Example
This example illustrates retrieving the status of each node in the cluster.

```bash
curl \
  --user Administrator:password \n  --silent \n  --request GET \n  http://localhost:8091/pools/nodes | \n  jq -r ".nodes[] | .hostname + " (" +.status + ")"
```

Example

The following example displays the cluster membership of each node

```bash
curl \
  --user Administrator:password \n  --silent \n  --request GET \n  http://localhost:8091/pools/nodes | \n  jq -r ".nodes[] | .hostname + " (" +.clusterMembership + ")"
```

Example

Show the services and system stats for each node cluster.

```bash
curl \
  --user Administrator:password \n  --silent \n  --request GET \n  http://localhost:8091/pools/nodes | \n  jq -r ".nodes[] | .hostname + " (" + (.services | join(" , ")) + ")\\n" + " cpu_utilization_rate: " + ( .systemStats.cpu_utilization_rate | tostring) + "%\\n" + " swap_total: " + ( .systemStats.swap_total / 1024 / 1024 | tostring) + "MB\\n" + " swap_used: " + ( .systemStats.swap_used / 1024 / 1024 | tostring) + "MB (" + ( (.systemStats.swap_used / .systemStats.swap_total) * 100 | tostring) + ")\\n" + " mem_total: " + ( .systemStats.mem_total / 1024 / 1024 | tostring) + "MB\\n" + " mem_free: " + ( .systemStats.mem_free / 1024 / 1024 | tostring) + "MB (" + ( (.systemStats.mem_free / .systemStats.mem_total) * 100 | tostring) + "
```

Couchbase Professional Services
### Monitoring: Query Service

#### Query Service-Level Stats

The following Query stats are available via the Cluster-Wide or Per-Node Endpoints listed below.

#### Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query_avg_req_time</td>
<td>The average total request time.</td>
</tr>
<tr>
<td>query_avg_svc_time</td>
<td>The average time of the query service for requests.</td>
</tr>
<tr>
<td>query_avg_response_size</td>
<td>The average size in bytes of the response.</td>
</tr>
<tr>
<td>query_avg_result_count</td>
<td>The average number of results being returned.</td>
</tr>
<tr>
<td>query_active_requests</td>
<td>The number of active requests.</td>
</tr>
<tr>
<td>query_errors</td>
<td>The number of queries resulting in an error.</td>
</tr>
<tr>
<td>query_invalid_requests</td>
<td>The number of invalid / incorrectly formatted queries.</td>
</tr>
<tr>
<td>query_queued_requests</td>
<td>The number of query requests that have been queued.</td>
</tr>
<tr>
<td>query_request_time</td>
<td>The current request duration.</td>
</tr>
<tr>
<td>query_requests</td>
<td>The current number of requests per second.</td>
</tr>
<tr>
<td>query_requests_1000ms</td>
<td>The number of queries greater than 1000ms.</td>
</tr>
<tr>
<td>query_requests_250ms</td>
<td>The number of queries greater than 250ms.</td>
</tr>
<tr>
<td>query_requests_5000ms</td>
<td>The number of queries greater than 5000ms.</td>
</tr>
<tr>
<td>query_requests_500ms</td>
<td>The number of queries greater than 500ms.</td>
</tr>
<tr>
<td>query_result_count</td>
<td>The number of results returned.</td>
</tr>
<tr>
<td>query_result_size</td>
<td>The result query result size.</td>
</tr>
<tr>
<td>query_selects</td>
<td>The number of selects being executed.</td>
</tr>
<tr>
<td>query_service_time</td>
<td>The time spent by the query service to service the request.</td>
</tr>
<tr>
<td>query_warnings</td>
<td>The number of query warnings generated.</td>
</tr>
</tbody>
</table>

**GET  Cluster Query Service Stats**

These endpoints are informational and should not be used for monitoring as they are an aggregate for the entire and cluster and the best practice is to monitor each node individually.

- **Insecure**: http://localhost:8091/pools/default/buckets/@query/stats
- **Secure**: https://localhost:18091/pools/default/buckets/@query/stats
curl \\n  --user Administrator:password \\n  --silent \\n  --request GET \\n  --data zoom=minute \\n  http://localhost:8091/pools/default/buckets/@query/stats | \\
  jq -r '.op.samples | to_entries[] | select(.key != "timestamp") | \\
  .key + ": " + (.value | add / length | tostring)'

### GET Node-Level Query Service Stats

Each node in the cluster running the query service should be monitoring individually using the endpoint listed below.

- **Insecure**: http://localhost:8091/pools/default/buckets/@query/nodes/{NODE}/stats
- **Secure**: https://localhost:18091/pools/default/buckets/@query/nodes/{NODE}/stats

#### Example: Stats for Individual Node

The following example demonstrates how to retrieve the query service stats for the cluster.

**NODE=172.17.0.2:8091**

curl \\n  --user Administrator:password \\n  --silent \\n  --request GET \\n  --data zoom=minute \\n  http://localhost:8091/pools/default/buckets/@query/nodes/$NODE/stats | \\
  jq -r -c '.op.samples | \\
  " query_avg_req_time: " + (.query_avg_req_time | add / length | tostring) + \\
  "\n query_avg_svc_time: " + (.query_avg_svc_time | add / length | tostring) + \\
  "\n query_avg_response_size: " + (.query_avg_response_size | add / length | tostring) + \\
  "\n query_avg_result_count: " + (.query_avg_result_count | add / length | tostring) + \\
  "\n query_active_requests: " + (.query_active_requests | add | tostring) + \\
  "\n query_errors: " + (.query_errors | add | tostring) + \\
  "\n query_invalid_requests: " + (.query_invalid_requests | add | tostring) + \\
  "\n query_queued_requests: " + (.query_queued_requests | add | tostring)
Example: Stats for Each Node Separately

```bash
# loop over each of the buckets
for node in $(curl \
   --user Administrator:password \
   --silent \
   --request GET \n   http://localhost:8091/pools/nodes | \
   jq -r '.nodes[] | \
     select(.services | contains(["n1ql"]) == true) | \
     .hostname'
)
do
echo "$node Query Stats"
echo "-------------------------------------------------------"
# get the query stats for the specific node
curl \
   --user Administrator:password \
   --silent \
   --request GET \
   --data zoom=minute \
   http://localhost:8091/pools/default/buckets/@query/nodes/$node/stats | \
   jq -r '.op.samples | to_entries[] | select(.key != "timestamp") | \
   .key + ": " + (.value | add / length | tostring)'
done
```

Key Metrics to Monitor

<table>
<thead>
<tr>
<th>Couchbase Metric</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
</table>

Couchbase Professional Services
<table>
<thead>
<tr>
<th><strong>query_avg_svc_time</strong></th>
<th>The average time of the query service for requests.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create a baseline for this value, as &quot;normal&quot; will depend on workload. Alert at 2x of the baseline. This would indicate that more query nodes may be needed or indexes are performing slowly and require investigation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>query_requests</strong></th>
<th>The number of query requests per second.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create a baseline for this value, as &quot;normal&quot; will depend on workload. Alert at 2x of the baseline. This would indicate an increase in query traffic.</td>
</tr>
</tbody>
</table>
Monitoring: Operating System

Operating System Metrics

Just as monitoring Couchbase and the individual services, buckets, indexes, etc. is extremely important to have a solid understanding of overall cluster health, it is also important to monitor the operating system and various stats for each node in the cluster. Each operating system has varying means of retrieving these metrics and many monitoring solutions collect them out of the box.

<table>
<thead>
<tr>
<th>OS Metric</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free RAM</td>
<td>Free + cache memory should always be at least 20% of total system memory. If free + cache memory falls below 20%, scale the cluster.</td>
</tr>
<tr>
<td>Swap usage</td>
<td>Swap usage should always be zero. If swap is used, it means the OS is under very high memory pressure and unable to purge dirty pages fast enough and the cluster should be scaled.</td>
</tr>
<tr>
<td>Memcached process RAM usage</td>
<td>Create a baseline for this value as &quot;normal&quot; will be dependent upon your working set. Alert if this value exceeds 150% of baseline. This may indicate an unusual increase in write traffic, reading of typically cold data, or possible malloc fragmentation. Confirm the Couchbase resident ratios are still correct. Add memory or scale the cluster if necessary.</td>
</tr>
<tr>
<td>Beam.smp process RAM usage</td>
<td>Create a baseline for this value as &quot;normal&quot; will be dependent upon your cluster size and API activity levels. Alert if this value exceeds 120% of baseline. This may indicate a memory leak in the beam process. Contact Couchbase Support if larger than a few gigabytes.</td>
</tr>
<tr>
<td>IO utilization (iostat)</td>
<td>Create a baseline for this value as &quot;normal&quot; will be dependent upon your workload and available disk IO. Overall sustained IO utilization should not exceed 90% of total IO capacity.</td>
</tr>
<tr>
<td>Total CPU utilization</td>
<td>Create a baseline for this value as &quot;normal&quot; will be dependent upon your workload. Sustained CPU utilization &gt;90% indicates a need to scale the cluster.</td>
</tr>
<tr>
<td>Couchbase service CPU utilization</td>
<td>Create a baseline for these values as &quot;normal&quot; will be dependent upon your workload. Alert if this value exceeds 2x of baseline.</td>
</tr>
<tr>
<td>Beam.smp CPU utilization</td>
<td>Create a baseline for this value as &quot;normal&quot; will be dependent upon your workload. Alert if this value exceeds 2x of baseline.</td>
</tr>
<tr>
<td>%steal CPU</td>
<td>This value should always be zero. Anything above zero indicates the VM hypervisor is oversubscribed. Additional physical hosts should be added or collocated VMs should be migrated to other hosts.</td>
</tr>
<tr>
<td>Network utilization</td>
<td>Create a baseline for this value as &quot;normal&quot; will be dependent upon your workload. Alert if this value exceeds 120% of baseline. If the sustained utilization is above 80% of the total available bandwidth, it indicates the need to scale the cluster.</td>
</tr>
<tr>
<td>Presence of beam.smp process</td>
<td>Alert if beam.smp is not present. This indicates Couchbase is offline and needs to be restarted.</td>
</tr>
<tr>
<td></td>
<td>Alert if data/index/query/fts/eventing/analytics processes are not present. This indicates Couchbase is either offline, starting up, or services may have crashed and need to be</td>
</tr>
</tbody>
</table>
Presence of service processes

The processes by service:
- Data Service: memcached
- Data Service: projector
- Data Service: goxdcr
- Index Service: indexer
- Query Service: cbq-engine
- Full Text Search Service: cbft
- Eventing Service: eventing-producer
- Eventing Service: eventing-consumer
- Analytics Service: cbas

NTP clock skew

Couchbase requires all cluster nodes (and any replicated clusters) to have their system clocks synchronized to a common clock source. Monitor clock skew on each server and alert if it is more than 1 minute out of sync.

## Couchbase System Stats

The following Operating System stats are available via the Cluster-Wide or Per-Node Endpoints listed below.

### Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocstall</td>
<td>Number of allocations stalled when reclaiming</td>
</tr>
<tr>
<td>cpu_cores_available</td>
<td>Number of CPU cores available in the cluster or the node</td>
</tr>
<tr>
<td>cpu_irq_rate</td>
<td>The CPU interrupt request rate</td>
</tr>
<tr>
<td>cpu_stolen_rate</td>
<td>CPU steal rate</td>
</tr>
<tr>
<td>cpu_idle_ms</td>
<td>The amount of time the CPU has been idle</td>
</tr>
<tr>
<td>cpu_local_ms</td>
<td></td>
</tr>
<tr>
<td>cpu_utilization_rate</td>
<td>Max CPU utilization %</td>
</tr>
<tr>
<td>hibernated_requests</td>
<td>Idle streaming requests</td>
</tr>
<tr>
<td>hibernated_waked</td>
<td>Streaming wakeups/sec</td>
</tr>
<tr>
<td>mem_actual_free</td>
<td>Amount of RAM available on this server</td>
</tr>
<tr>
<td>mem_actual_used</td>
<td>Amount of RAM used on this server</td>
</tr>
<tr>
<td>mem_free</td>
<td>Amount of RAM available on this server</td>
</tr>
<tr>
<td>mem_limit</td>
<td>The limit for RAM</td>
</tr>
<tr>
<td>mem_total</td>
<td>Amount of RAM used on this server</td>
</tr>
<tr>
<td>mem_used_sys</td>
<td>Amount of RAM available to the OS</td>
</tr>
<tr>
<td>odp_report_failed</td>
<td></td>
</tr>
<tr>
<td>rest_requests</td>
<td>Management port reqs/sec</td>
</tr>
<tr>
<td>swap_total</td>
<td>Amount of swap space available on this server</td>
</tr>
<tr>
<td>swap_used</td>
<td>Amount of swap space in use on this server</td>
</tr>
</tbody>
</table>
**GET Cluster System Stats**

These endpoints are informational and should not be used for monitoring as they are an aggregate for the entire and cluster and the best practice is to monitor each node individually.

- **Insecure:** http://localhost:8091/pools/default/buckets/@system/stats
- **Secure:** https://localhost:18091/pools/default/buckets/@system/stats

```bash
curl \
    --user Administrator:password \
    --silent \
    --request GET \n    --data zoom=minute \n    http://localhost:8091/pools/default/buckets/@query/stats | \ 
    jq -r '.op.samples | to_entries[] | select(.key != "timestamp") | .key + "": " + (.value | add / length | tostring)'
```

**GET Node-Level OS Stats**

Each node in the cluster should be monitoring individually using the endpoint listed below.

- **Insecure:** http://localhost:8091/pools/default/buckets/@system/nodes/{NODE}/stats
- **Secure:** https://localhost:18091/pools/default/buckets/@system/nodes/{NODE}/stats

**Example: Stats for Individual Node**

The following example demonstrates how to retrieve the system stats for the cluster.

```bash
NODE="172.17.0.2:8091"

curl \
    --user Administrator:password \
    --silent \
    --request GET \
    --data "zoom=minute" \n    http://localhost:8091/pools/default/buckets/@system/nodes/$NODE/stats | \ 
    jq -r -c '.op.samples | " cpu_idle_ms: " + (.cpu_idle_ms | add / length | tostring) + \
    " cpu_local_ms: " + (.cpu_local_ms | add / length | tostring) + \
    " cpu_utilization_rate: " + (.cpu_utilization_rate | add / length | tostring) + \
    " hibernated_requests: " + (.hibernated_requests | add / length | tostring) + \
```

Monitoring: Operating System
Example: Stats for Each Node Separately

```bash
# loop over each of the buckets
for node in $(curl \
  --user Administrator:password \ 
  --silent \ 
  --request GET \ 
  http://localhost:8091/pools/nodes | \ 
  jq -r '.nodes[] | .hostname' ) 
  do 
    echo "$node OS Stats"
    echo "-------------------------------------------------------"
    # get the system stats for the specific node
    curl \
      --user Administrator:password \ 
      --silent \ 
      --request GET \ 
      --data zoom=minute \ 
      http://localhost:8091/pools/default/buckets/@system/nodes/$node/stats | \ 
      jq -r '.op.samples | to_entries[] | select(.key != "timestamp") | .key + "": " + (.value | add / length | tostring)' 
    done
```
Monitoring: XDCR

Replication Status

The tasks endpoint will provide cluster wide information on operations such as rebalance, XDCR replications, etc. The response is an array that will need to be filtered for items containing 

Insecure: http://localhost:8091/pools/default/tasks
Secure: http://localhost:18091/pools/default/tasks

Response:

```json
[
    
    "cancelURI": "/controller/cancelXDCR/20763b82bb6b517bd0d15d9f6b78c13c%2Ftravel-sample%2Fdemo",
    "settingsURI": "/settings/replications/20763b82bb6b517bd0d15d9f6b78c13c%2Ftravel-sample%2Fdemo",
    "status": "running",
    "replicationType": "xmem",
    "continuous": true,
    "filterExpression": "",
    "id": "20763b82bb6b517bd0d15d9f6b78c13c/travel-sample/demo",
    "pauseRequested": false,
    "source": "travel-sample",
    "target": "/remoteClusters/20763b82bb6b517bd0d15d9f6b78c13c/buckets/demo",
    "type": "xdcr",
    "recommendedRefreshPeriod": 10,
    "changesLeft": 0,
    "docsChecked": 0,
    "docsWritten": 31591,
    "maxVBReps": null,
    "errors": []
]
```

Key Metrics to Monitor

<table>
<thead>
<tr>
<th>Couchbase Metric</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Indicates whether a replication is in a "running", "paused", or "notRunning" state.

Alert if the value is "paused" or "notRunning".

Note: The replicationId is composed of 3 parts, delimited by a / :

Sample ReplicationId: 6f76c2a07245aef856db44a8e361032/travel-sample/default
Example

The following example illustrates outputting the replication ID and Status.

curl \\
    --user Administrator:password \\
    --silent \\
    --request GET \\
    http://localhost:8091/pools/default/tasks | \\
    jq -r 'map(select(.type | contains("xdcr"))) | \\
    .[] | .id + " (" + .status + ")"'

This example shows outputting all replications whose status is "paused" or "notRunning"

curl \\
    --user Administrator:password \\
    --silent \\
    --request GET \\
    http://localhost:8091/pools/default/tasks | \\
    jq -c 'map(select( \\
        (.type | contains("xdcr")) \\
        and \\
        (.status | contains("paused") or contains("notRunning"))) | .[] | .id + " (" + .status + ")"'

Per Replication Stats

The XDCR stats are an aggregate for all of the configured replications, either for the entire cluster or a specific node.

html

Documentation: https://docs.couchbase.com/server/6.0/rest-api/rest-xdcr-statistics.html

Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replication_changes_left</td>
<td>The total number of changes left across all replications for the bucket</td>
</tr>
<tr>
<td>replication_documents_left</td>
<td>The total number of documents in replications</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>replication_docs_rep_queue</code></td>
<td>The total number of documents in replication queue for all replications for the bucket.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/bandwidth_usage</code></td>
<td>Bandwidth used during replication, measured in bytes per second.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/changes_left</code></td>
<td>Number of mutations to be replicated to the remote cluster.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/data_replicated</code></td>
<td>Size of data replicated in bytes.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/datapool_failed_gets</code></td>
<td>Number of failed gets from the pool.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/dcp_datach_length</code></td>
<td></td>
</tr>
<tr>
<td><code>replications/{replicationId}/dcp_dispatch_time</code></td>
<td></td>
</tr>
<tr>
<td><code>replications/{replicationId}/deletion_docs_written</code></td>
<td>The number of docs deleted that have been written to the target cluster.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/deletion_failed_cr_source</code></td>
<td>The number of deletes that have failed conflict resolution on the source due to optimistic replication.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/deletion_filtered</code></td>
<td>The number of deletes that have been filtered.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/deletion_received_from_dcp</code></td>
<td>The number of deletes that have been received from DCP.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/docs_checked</code></td>
<td>Number of documents checked for changes.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/docs_failed_cr_source</code></td>
<td>The number of docs that have failed conflict resolution on the source due to optimistic replication.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/docs_filtered</code></td>
<td>Number of documents that have been filtered out and not replicated to target cluster.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/docs_opt_rep</code></td>
<td>Number of documents sent optimistically.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/docs_processed</code></td>
<td>The number of documents processed.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/docs_received_from_dcp</code></td>
<td>Number of documents received from DCP.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/docs_rep_queue</code></td>
<td>Number of documents in replication queue.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/docs_unable_to_filter</code></td>
<td>The number of documents where filtering could not be processed.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/docs_written</code></td>
<td>Number of documents written to the target cluster.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/expiry_docs_written</code></td>
<td>The number of expiry documents written to the target cluster.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/expiry_failed_cr_source</code></td>
<td>The number of expiry documents that have failed conflict resolution on the source due to optimistic replication.</td>
</tr>
<tr>
<td><code>expiry_filtered</code></td>
<td>The number of expiry documents that have been filtered out and not replicated to the target cluster.</td>
</tr>
<tr>
<td><code>replications/{replicationId}/expiry_received_from_dcp</code></td>
<td>The number of expiry documents that have been received.</td>
</tr>
<tr>
<td>Endpoint</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>replications/{replicationId}/expiry_stripped</td>
<td>The number of expiry documents removed from replicating</td>
</tr>
<tr>
<td>replications/{replicationId}/num_checkpoints</td>
<td>Number of checkpoints issued in replication queue</td>
</tr>
<tr>
<td>replications/{replicationId}/num_failedckpts</td>
<td>Number of checkpoints failed during replication</td>
</tr>
<tr>
<td>replications/{replicationId}/percent_completeness</td>
<td>Percentage of checked items out of all checked and to-be-replicated items</td>
</tr>
<tr>
<td>replications/{replicationId}/rate_doc_checks</td>
<td>Rate of documents being replicated, measured in documents per second</td>
</tr>
<tr>
<td>replications/{replicationId}/rate_doc_opt_repd</td>
<td>Rate of documents received from DCP per second</td>
</tr>
<tr>
<td>replications/{replicationId}/rate_received_from_dcp</td>
<td>Rate of documents being replicated, measured in documents per second</td>
</tr>
<tr>
<td>replications/{replicationId}/resp_wait_time</td>
<td>The number of sets that have failed conflict resolution on the source due to optimistic replication</td>
</tr>
<tr>
<td>replications/{replicationId}/set_docs_written</td>
<td>The number of sets that have failed conflict resolution on the source due to optimistic replication</td>
</tr>
<tr>
<td>replications/{replicationId}/set_failed_cr_source</td>
<td>The number of sets that have failed conflict resolution on the source due to optimistic replication</td>
</tr>
<tr>
<td>replications/{replicationId}/set_filtered</td>
<td>Number of sets that have been filtered out and not replicated to target cluster</td>
</tr>
<tr>
<td>replications/{replicationId}/set_received_from_dcp</td>
<td>The number of sets that have been received from DCP</td>
</tr>
<tr>
<td>replications/{replicationId}/size_rep_queue</td>
<td>Size of replication queue in bytes</td>
</tr>
<tr>
<td>replications/{replicationId}/throttle_latency</td>
<td>Throttle latency</td>
</tr>
<tr>
<td>replications/{replicationId}/throughput_throttle_latency</td>
<td>Throughput throttle latency</td>
</tr>
<tr>
<td>replications/{replicationId}/time_committing</td>
<td>Seconds elapsed during replication</td>
</tr>
<tr>
<td>replications/{replicationId}/wtavg_docs_latency</td>
<td>Weighted average latency for sending replicated changes to target cluster</td>
</tr>
<tr>
<td>replications/{replicationId}/wtavg_meta_latency</td>
<td>Weighted average time for requesting document metadata. XDCR uses this for conflict resolution prior to sending the document into the replication queue</td>
</tr>
</tbody>
</table>

**GET**  
Cluster-Wide Bucket XDCR Stats

These endpoints are informational and should not be used for monitoring as they are an aggregate for the entire and cluster and the best practice is to monitor each node individually.

- **Insecure**: http://localhost:8091/pools/default/buckets/@xdcr-{BUCKET}/stats
- **Secure**: http://localhost:8091/pools/default/buckets/@xdcr-{BUCKET}/stats
Example: Single Bucket

This example will output the XDCR stats for a specific bucket

curl \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/@xdcr-travel-sample/stats | \
  jq -r '.op.samples | to_entries | sort_by(.key) | .[1] | 
    select(.key | split("/") | length > 1) | 
    " " + (.key) + ": " + 
    (.value | add / length | tostring)'

Example: All Replications

This example will output all XDCR stats for every bucket that has one or more replications configured.

```bash
# loop over each of the buckets
for bucket in $(curl \
  --user Administrator:password \
  --silent \
  --request GET \
  http://localhost:8091/pools/default/tasks | \
  jq -r '[[ .[1] | select(.type == "xdcr") | .source ] | sort | unique | .[]])
  do
    echo ""
    echo "Bucket: $bucket"
    echo "=================================================================================
    # get the xdcr stats for the bucket
    curl \
      --user Administrator:password \
      --silent \
      --request GET \
      --data zoom=minute \
      http://localhost:8091/pools/default/buckets/@xdcr-$bucket/stats | \
      jq -r '.op.samples | to_entries | sort_by(.key) | .[1] | 
      select(.key | split("/") | length > 1) | 
      " " + (.key) + ": " + 
      (.value | add / length | tostring)'
  done
```

GET Node-Level Bucket XDCR Stats
Each data node in the cluster should be monitoring individually using the endpoint listed below.

- Insecure: http://localhost:8091/pools/default/buckets/@xdcr-{BUCKET}/nodes/{NODE}/stats
- Secure: http://localhost:8091/pools/default/buckets/@xdcr-{BUCKET}/nodes/{NODE}/stats

**Example: Single Bucket**

This example will output the XDCR stats for a specific node and bucket.

```bash
curl \
   --user Administrator:password \ 
   --silent \ 
   --request GET \ 
   --data zoom=minute \ 
   http://localhost:8091/pools/default/buckets/@xdcr-travel-sample/nodes/172.17.0.2:8091/stats \ 
   jq -r ".op.samples | to_entries | sort_by(.key) | .[] | select(.key | split("/")) | length > 1) | " + (.key) + ": " + (.value | add / length | tostring)
```

**Example: All Replications**

This example will output all XDCR stats for a single node for every bucket that has one or more replications configured.

```bash
# loop over each of the buckets
for bucket in $(curl \
   --user Administrator:password \ 
   --silent \ 
   --request GET \ 
   http://localhost:8091/pools/default/tasks | \
   jq -r '[] | select(.type == "xdcr") | .source ] | sort | unique | .[]')
do
   echo ""
   echo "Bucket: $bucket"
   echo "=================================

# get the xdcr stats for the bucket

curl \
   --user Administrator:password \ 
   --silent \ 
   --request GET \ 
   --data zoom=minute \ 
   http://localhost:8091/pools/default/buckets/@xdcr-$bucket/nodes/172.17.0.2:8091/stats | \
   jq -r ".op.samples | to_entries | sort_by(.key) | .[] | select(.key | split("/")) | length > 1) |"
Example: All Replications for Each Node

This example will output all XDCR stats for a single node for every bucket that has one or more replications configured.

```bash
# get all of the buckets in the cluster that have 1 or more xdcr replications configured
buckets=$(curl \
  --user Administrator:password \
  --silent \
  --request GET \
  http://localhost:8091/pools/default/tasks | \n  jq -r '[ .[]. | select(.type == "xdcr") | .source ] | sort | unique | .[]')

# get all of the nodes in the cluster running the data service
nodes=$(curl \
  --user Administrator:password \
  --silent \
  --request GET \
  http://localhost:8091/pools/nodes | \n  jq -r '.nodes[] | select(.services | contains(["kv"]) == true) | .hostname'
)

# loop over each of the buckets
for bucket in ${buckets[@]}
do
  echo ""
  echo "Bucket: $bucket"
  echo "================================================================"
  # loop over each of the nodes in the cluster
  for node in ${nodes[@]}
do
    echo "Node: $node"
    echo "================================================================"
    # get the xdcr stats for the bucket on the node
    curl \
      --user Administrator:password \
      --silent \
      --request GET \
      --data zoom=minute \
      http://localhost:8091/pools/default/buckets/@xdcr-$bucket/nodes/$node/stats | \
```

Monitoring: XDCR
**Key Metrics to Monitor**

<table>
<thead>
<tr>
<th>Couchbase Metric</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>changes_left</td>
<td>The number of items pending XDCR replication. This can be used to approximate the degree of eventual consistency between clusters.</td>
<td>Create a baseline for this value as “normal” will depend on workload, XDCR configuration, and available bandwidth. Alert at 2x of baseline. This may indicate a resource bottleneck.</td>
</tr>
<tr>
<td>bandwidth_usage</td>
<td>The amount of bandwidth in bytes used for XDCR replication.</td>
<td>An alert value for this metric should be based on the network interconnect capacity between the clusters and the percentage of the interconnect XDCR is expected or allowed to consume.</td>
</tr>
</tbody>
</table>

**GET**  Per Node Individual Stat for a Replication

Each XDCR replication stat can be retrieved individually. The entire key must be URL-encoded, where /’s are replaced with %2F.

Documentation: [https://docs.couchbase.com/server/6.0/rest-api/rest-xdcr-statistics.html](https://docs.couchbase.com/server/6.0/rest-api/rest-xdcr-statistics.html)

Example

This example shows requesting an individual stat for a single replication and displays the results for each data node in the cluster.

```bash
# set the replication info
REMOTE_CLUSTER='20763b82bb6b517bd0d15d9f6b78c13c'
SOURCE_BUCKET='travel-sample'
target_BUCKET='demo'
STAT_NAME='percent_completeness'

# build the url
STAT_URL="http://localhost:8091/pools/default/buckets/$SOURCE_BUCKET/stats"
STAT_URL="$STAT_URL/replications%2F$REMOTE_CLUSTER%2F$SOURCE_BUCKET"
STAT_URL="$STAT_URL%2F$target_BUCKET%2F$STAT_NAME"

curl \
```
GET Remote Cluster Information

The replicationId is a uniquely generated ID and does not convey the remote cluster details. All configured remote clusters and their associated IDs can be retrieved from the REST API.

Documentation: https://docs.couchbase.com/server/6.0/rest-api/rest-xdcr-get-ref.html

- Insecure: http://localhost:8091/pools/default/remoteClusters
- Secure: https://localhost:18091/pools/default/remoteClusters

Example

This example shows requesting an individual stat for a single replication and displays the results for each data node in the cluster.

```bash
curl \
    --user Administrator:password \ 
    --silent \ 
    --request GET \ 
    http://localhost:8091/pools/default/remoteClusters | \ 
    jq -r '.
```

Bucket XDCR Operations

GET Bucket Incoming XDCR operations

To retrieve the incoming write operations that occur on a target cluster due to replication, make the request on your target cluster and bucket.

Documentation: https://docs.couchbase.com/server/6.0/rest-api/rest-xdcr-statistics.html#rest-xdcr-stats-operations

- Insecure: http://localhost:8091/pools/default/buckets/{BUCKET}/stats
- Secure: http://localhost:8091/pools/default/buckets/{BUCKET}/stats

Available Stats

<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ep_num_ops_get_meta</td>
<td>The number of metadata read operations per second for the bucket as the</td>
</tr>
<tr>
<td></td>
<td>target.</td>
</tr>
</tbody>
</table>
### GET XDCR Timestamp-based Conflict Resolution Stats

When using buckets configured with Timestamp-based Conflict Resolution it is important to monitor the drift related statistics. When a cluster is the destination for XDCR traffic, active vBuckets will calculate drift from their remote cluster peers.

It is normal for a cluster with closely synchronized clocks to show some drift; in general it will be showing how long it took a mutation to be replicated and should remain steady. It is also normal for the active vBucket drift to be zero if no XDCR relationship exists (or if no XDCR traffic is flowing).

**Documentation:** [https://docs.couchbase.com/server/6.0/learn/clusters-and-availability/xdcr-monitor-timestamp-conflict-resolution.html](https://docs.couchbase.com/server/6.0/learn/clusters-and-availability/xdcr-monitor-timestamp-conflict-resolution.html)

- **Insecure:** [http://localhost:8091/pools/default/buckets/{BUCKET}/stats](http://localhost:8091/pools/default/buckets/{BUCKET}/stats)
- **Secure:** [http://localhost:8091/pools/default/buckets/{BUCKET}/stats](http://localhost:8091/pools/default/buckets/{BUCKET}/stats)

### Available Stats

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ep_num_ops_get_meta</td>
<td>The number of get operations per second for the bucket as the target for XDCR</td>
</tr>
<tr>
<td>ep_num_ops_set_meta</td>
<td>The number of set operations per second for the bucket as the target for XDCR</td>
</tr>
<tr>
<td>ep_num_ops_del_meta</td>
<td>The number of delete operations per second for the bucket as the target for XDCR</td>
</tr>
<tr>
<td>xdc_ops</td>
<td>Total XDCR operations per second for this bucket (measured from the sum of the statistics: ep_num_ops_del_meta, ep_num_ops_get_meta, and ep_num_ops_set_meta)</td>
</tr>
</tbody>
</table>

### Example

```
curl \
  --user Administrator:password \
  -s \
  -X GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/travel-sample/stats | \n  jq -r '.op.samples | 
    "ep_num_ops_get_meta: " + (.ep_num_ops_get_meta | add / length | toString) + 
    "ep_num_ops_set_meta: " + (.ep_num_ops_set_meta |add / length | toString) + 
    "ep_num_ops_del_meta: " + (.ep_num_ops_del_meta |add / length | toString) + 
    "xdc_ops: " + (.xdc_ops |add / length | toString)'
```

---

**Monitoring:** XDCR
<table>
<thead>
<tr>
<th>Stat name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg_active_timestamp_drift</td>
<td>The sum of total_abs_drift for the node's active vBuckets</td>
</tr>
<tr>
<td>avg_replica_timestamp_drift</td>
<td>The sum of total_abs_drift for the node's active vBuckets</td>
</tr>
<tr>
<td>ep_active_hlc_drift</td>
<td>The sum of total_abs_drift for the node's active vBuckets</td>
</tr>
<tr>
<td>ep_active_hlc_drift_count</td>
<td>The sum of total_abs_drift_count for the node's active vBuckets</td>
</tr>
<tr>
<td>ep_replica_hlc_drift</td>
<td>The sum of total_abs_drift for the node's active vBuckets</td>
</tr>
<tr>
<td>ep_replica_hlc_drift_count</td>
<td>The sum of total_abs_drift_count for the node's active vBuckets</td>
</tr>
<tr>
<td>ep_active_ahead_exceptions</td>
<td>The sum of drift_ahead_exceeded for the node's active vBuckets</td>
</tr>
<tr>
<td>ep_replica_ahead_exceptions</td>
<td>The sum of drift_ahead_exceeded for the node's replica vBuckets</td>
</tr>
<tr>
<td>ep_clock_cas_drift_threshold_exceeded</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

curl \
  --user Administrator:password \  
  --silent \  
  --request GET \  
  --data zoom=minute \  
  http://localhost:8091/pools/default/buckets/travel-sample/stats | \  
jq -r '.op.samples | 
  "avg_active_timestamp_drift: " +  
  (.avg_active_timestamp_drift | add / length | tostring) +  
  "\navg_replica_timestamp_drift: " +  
  (.avg_replica_timestamp_drift | add / length | tostring) +  
  "\nep_active_hlc_drift: " +  
  (.ep_active_hlc_drift | add / length | tostring) +  
  "\nep_active_hlc_drift_count: " +  
  (.ep_active_hlc_drift_count | add / length | tostring) +  
  "\nep_replica_hlc_drift: " +  
  (.ep_replica_hlc_drift | add / length | tostring) +  
  "\nep_replica_hlc_drift_count: " +  
  (.ep_replica_hlc_drift_count | add / length | tostring) +  
  "\nep_active_ahead_exceptions: " +  
  (.ep_active_ahead_exceptions | add / length | tostring) +  
  "\nep_clock_cas_drift_threshold_exceeded: " +  
  (.ep_clock_cas_drift_threshold_exceeded | add / length | tostring)'