Monitoring Guide

Couchbase
NoEQUAL

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.

```
curl \
  --user Administrator:password \
  --silent \
  --request GET \
  --data skipMap=true \
  http://localhost:8091/pools/default/buckets | \
  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\n"
  '
```

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

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

couch_spatial_disk_size	this bucket on disk
couch_spatial_ops	All the spatial index reads
couch_total_disk_size	The total size on disk of all data and view files for this bucket.
couch_views_actual_disk_size	The size of all active items in all the indexes for this bucket on disk
couch_views_data_size	The size of active data on for all the view indexes in this bucket
couch_views_disk_size	The size of active data on for all the view indexes in this bucket on disk
couch_views_fragmentation	How much fragmented data there is to be compacted compared to real data for the view index files in this bucket
couch_views_ops	All the view reads for all design documents including scatter gather.
curr_connections	Number of connections to this server including connections from external client SDKs, proxies, DCP requests and internal statistic gathering
curr_items	Number of unique items in this bucket - only active items, not replica
curr_items_tot	Total number of items in this bucket (including replicas)
decr_hits	Number of decrement operations per second for data that this bucket contains
decr_misses	Number of decr operations per second for data that this bucket does not contain
delete_hits	Number of delete operations per second for this bucket
delete_misses	Number of delete operations per second for data that this bucket does
disk_commit_count	The number of disk comments
disk_commit_total	The total time spent committing to disk
disk_update_count	The total number of disk updates
disk_update_total	The total time spent updating disk
disk_write_queue	Number of items waiting to be written to disk in this bucket
ep_active_ahead_exceptions	Total number of ahead exceptions for all active vBuckets
ep_active_hlc_drift	The sum of total_abs_drift for the nodes active vBuckets
ep_active_hlc_drift_count	The sum of total_abs_drift_count for the nodes active vBuckets
ep_bg_fetched	Number of reads per second from disk for this bucket
ep_cache_miss_rate	Percentage of reads per second to this bucket from disk as opposed to RAM

ep_clock_cas_drift_threshold_exceeded	
ep_data_read_failed	Number of disk read failures
ep_data_write_failed	Number of disk write failures
ep_dcp_2i_backoff	Number of backoffs for index DCP connections
ep_dcp_2i_count	Number of internal second index DCP connections in this bucket
ep_dcp_2i_items_remaining	Number of secondary index items remaining to be sent to consumer in this bucket
ep_dcp_2i_items_sent	Number of secondary index items per second being sent for a producer for this bucket
ep_dcp_2i_producer_count	Number of secondary index senders for this bucket
ep_dcp_2i_total_backlog_size	Total size in bytes of the DCP backlog for secondary indexes
ep_dcp_2i_total_bytes	Number of bytes per second being sent for secondary indexes DCP connections
ep_dcp_cbas_backoff	Number of backoffs for Analytics DCP connections
ep_dcp_cbas_count	Number of internal Analytics DCP connections in this bucket
ep_dcp_cbas_items_remaining	Number of Analytics items remaining to be sent to consumer in this bucket
ep_dcp_cbas_items_sent	Number of Analytics items per second being sent for a producer for this bucket
ep_dcp_cbas_producer_count	Number of Analytics senders for this bucket
ep_dcp_cbas_total_backlog_size	Total size in bytes of the DCP backlog for Analytics
ep_dcp_cbas_total_bytes	Number of bytes per second being sent for Analytics DCP connections
ep_dcp_eventing_backoff	Number of backoffs for Eventing DCP connections
ep_dcp_eventing_count	Number of internal Eventing DCP connections in this bucket
ep_dcp_eventing_items_remaining	Number of Eventing items remaining to be sent to consumer in this bucket
ep_dcp_eventing_items_sent	Number of Eventing items per second being sent for a producer for this bucket
ep_dcp_eventing_producer_count	Number of Eventing senders for this bucket
ep_dcp_eventing_total_backlog_size	Total size in bytes of the DCP backlog for Eventing
ep_dcp_eventing_total_bytes	Number of bytes per second being sent for Eventing DCP connections
ep_dcp_fts_backoff	Number of backoffs for FTS DCP connections
ep_dcp_fts_count	Number of internal FTS DCP connections in this bucket
ep_dcp_fts_items_remaining	Number of FTS items remaining to be sent to consumer in this bucket

ep_dcp_fts_items_sent	Number of FTS items per second being sent for a producer for this bucket
ep_dcp_fts_producer_count	Number of FTS senders for this bucket
ep_dcp_fts_total_backlog_size	Total size in bytes of the DCP backlog for FTS
ep_dcp_fts_total_bytes	Number of bytes per second being sent for FTS DCP connections
ep_dcp_other_backoff	Number of backoffs for other DCP connections
ep_dcp_other_count	Number of other DCP connections in this bucket
ep_dcp_other_items_remaining	Number of items remaining to be sent to consumer in this bucket
ep_dcp_other_items_sent	Number of items per second being sent for a producer for this bucket
ep_dcp_other_producer_count	Number of other senders for this bucket
ep_dcp_other_total_backlog_size	Total size in bytes of the DCP backlog for analytics other
ep_dcp_other_total_bytes	Number of bytes per second being sent for other DCP connections for this bucket
ep_dcp_replica_backoff	Number of backoffs for replication DCP connections
ep_dcp_replica_count	Number of internal replication DCP connections in this bucket
ep_dcp_replica_items_remaining	Number of replication items remaining to be sent to consumer in this bucket
ep_dcp_replica_items_sent	Number of replication items per second being sent for a producer for this bucket
ep_dcp_replica_producer_count	Number of replication senders for this bucket
ep_dcp_replica_total_backlog_size	Total size in bytes of the DCP backlog for replication
ep_dcp_replica_total_bytes	Number of bytes per second being sent for replication DCP connections
ep_dcp_views+indexes_backoff	Number of backoffs for view/index DCP connections
ep_dcp_views+indexes_count	Number of internal view/index DCP connections in this bucket
ep_dcp_views+indexes_items_remaining	Number of view/index items remaining to be sent to consumer in this bucket
ep_dcp_views+indexes_items_sent	Number of view/index items per second being sent for a producer for this bucket
ep_dcp_views+indexes_producer_count	Number of views/index senders for this bucket
ep_dcp_views+indexes_total_backlog_size	Total size in bytes of the DCP backlog for views/indexes
ep_dcp_views+indexes_total_bytes	Number of bytes per second being sent for views/indexes DCP connections
ep_dcp_views_backoff	Number of backoffs for view DCP connections
ep_dcp_views_count	Number of internal view DCP connections in this bucket
	Number of view items remaining to be sent to consumer

ep_dcp_views_items_remaining	Number of view items remaining to be sent to consumer in this bucket
ep_dcp_views_items_sent	Number of view items per second being sent for a producer for this bucket
ep_dcp_views_producer_count	Number of view senders for this bucket
ep_dcp_views_total_backlog_size	Total size in bytes of the DCP backlog for views
ep_dcp_views_total_bytes	Number of bytes per second being sent for view DCP connections
ep_dcp_xdcr_backoff	Number of backoffs for XDCR DCP connections
ep_dcp_xdcr_count	Number of internal XDCR DCP connections in this bucket
ep_dcp_xdcr_items_remaining	Number of XDCR items remaining to be sent to consumer in this bucket
ep_dcp_xdcr_items_sent	Number of XDCR items per second being sent for a producer for this bucket
ep_dcp_xdcr_producer_count	Number of XDCR senders for this bucket
ep_dcp_xdcr_total_backlog_size	Total size in bytes of the DCP backlog for XDCR
ep_dcp_xdcr_total_bytes	Number of bytes per second being sent for XDCR DCP connections
ep_diskqueue_drain	Total number of items per second being written to disk in this bucket
ep_diskqueue_fill	Total number of items per second being put on the disk queue in this
ep_diskqueue_items	Total number of items waiting to be written to disk in this bucket
ep_flusher_todo	Number of items currently being written.
ep_item_commit_failed	Number of times a transaction failed to commit due to storage errors.
ep_kv_size	Total amount of user data cached in RAM in this bucket
ep_max_size	The maximum amount of memory this bucket can use.
ep_mem_high_wat	High water mark for auto-evictions
ep_mem_low_wat	Low water mark for auto-evictions
ep_meta_data_memory	Total amount of item metadata consuming RAM in this bucket
ep_num_non_resident	The number of non-resident items.
ep_num_ops_del_meta	Number of delete operations per second for this bucket as the target for XDCR
ep_num_ops_del_ret_meta	Number of delRetMeta operations.
ep_num_ops_get_meta	Number of metadata read operations per second for this bucket as the target for XDCR
ep_num_ops_set_meta	Number of set operations per second for this bucket as the target for XDCR

ep_num_ops_set_ret_meta	
ep_num_value_ejects	Total number of items per second being ejected to disk in this bucket
ep_oom_errors	Number of times unrecoverable OOMs happened while processing operations.
ep_ops_create	Total number of new items being inserted into this bucket
ep_ops_update	Number of items updated on disk per second for this bucket
ep_overhead	Extra memory used by transient data like persistence queues, replication queues, checkpoints, etc.
ep_queue_size	Number of items queued for storage.
ep_replica_ahead_exceptions	Total number of ahead exceptions for all replica vBuckets
ep_replica_hlc_drift	The sum of total_abs_drift for the node's active vBuckets
ep_replica_hlc_drift_count	The sum of total_abs_drift_count for the node's active vBuckets
ep_resident_items_rate	Percentage of all items cached in RAM in this bucket
ep_tmp_oom_errors	Number of back-offs sent per second to client SDKs due to "out of memory" situations from this bucket
ep_vb_total	Total number of vBuckets for this bucket
evictions	Number of items per second evicted from this bucket
get_hits	Number of get operations per second for data that this bucket contains
get_misses	Number of get operations per second for data that this bucket does not contain
hibernated_requests	Number of hibernated requests
hibernated_waked	Number of times hibernated waked
hit_ratio	Percentage of get requests served with data from this bucket
incr_hits	Number of increment operations per second for data that this bucket contains
incr_misses	Number of increment operations per second for data that this bucket does not contain
mem_used	Amount of Memory used
misses	Total amount of operations per second for that that the bucket does not contain
ops	Total amount of operations per second (including XDCR) to this bucket
rest_requests	
swap_total	

swap_used	
vb_active_eject	Number of items per second being ejected to disk from "active"
vb_active_itm_memory	Amount of active user data cached in RAM in this bucket
vb_active_meta_data_memory	Amount of active item metadata consuming RAM in this bucket
vb_active_num	Number of vBuckets in the "active" state for this bucket
vb_active_num_non_resident	Number of non-resident items.
vb_active_ops_create	New items per second being inserted into "active" vBuckets in this bucket
vb_active_ops_update	Number of items updated on disk per second for this bucket
vb_active_queue_age	Sum of disk queue item age in milliseconds for "active" vBuckets
vb_active_queue_drain	Number of active items per second being written to disk in this bucket
vb_active_queue_fill	Number of active items per second being put on the active item disk queue in this bucket
vb_active_queue_size	Number of active items waiting to be written to disk in this bucket
vb_active_resident_items_ratio	Percentage of active items cached in RAM in this bucket
vb_active_sync_write_aborted_count	Number of vbucket writes aborted
vb_active_sync_write_accepted_count	Number of vbucket writes accepted
vb_active_sync_write_committed_count	Number of vbucket writes committed
vb_avg_active_queue_age	Average age in seconds of active items in the active item queue for this bucket
vb_avg_pending_queue_age	Average age in seconds of pending items in the pending item queue for this bucket and should be transient during rebalancing
vb_avg_replica_queue_age	Average age in seconds of replica items in the replica item queue for this bucket
vb_avg_total_queue_age	Average age in seconds of all items in the disk write queue for this bucket
vb_pending_curr_items	Number of items in "pending" vBuckets in this bucket and should be transient during rebalancing
vb_pending_eject	Number of items per second being ejected to disk from "pending" vBuckets in this bucket and should be transient during rebalancing
vb_pending_itm_memory	Amount of pending user data cached in RAM in this bucket and should be transient during rebalancing
vb_pending_meta_data_memory	Amount of pending item metadata consuming RAM in this bucket and should be transient during rebalancing

vb_pending_num	Number of vBuckets in the "pending" state for this bucket and should be transient during rebalancing
vb_pending_num_non_resident	Number of non-resident items.
vb_pending_ops_create	New items per second being instead into "pending" vBuckets in this bucket and should be transient during rebalancing
vb_pending_ops_update	Number of items updated on disk per second for this bucket
vb_pending_queue_age	Sum of disk queue item age in milliseconds.
vb_pending_queue_drain	Number of pending items per second being written to disk in this bucket and should be transient during rebalancing
vb_pending_queue_fill	Number of pending items per second being put on the pending item disk queue in this bucket and should be transient during rebalancing
vb_pending_queue_size	Number of pending items waiting to be written to disk in this bucket and should be transient during rebalancing
vb_pending_resident_items_ratio	Percentage of items in pending state vbuckets cached in RAM in this bucket
vb_replica_curr_items	Number of items in "replica" vBuckets in this bucket
vb_replica_eject	Number of items per second being ejected to disk from "replica" vBuckets in this bucket
vb_replica_itm_memory	Amount of replica user data cached in RAM in this bucket
vb_replica_meta_data_memory	Amount of replica item metadata consuming in RAM in this bucket
vb_replica_num	Number of vBuckets in the "replica" state for this bucket
vb_replica_num_non_resident	Number of non-resident items.
vb_replica_ops_create	New items per second being inserted into "replica" vBuckets in this bucket
vb_replica_ops_update	Number of items updated on disk per second for this bucket
vb_replica_queue_age	Sum of disk queue item age in milliseconds for "replica" vBuckets
vb_replica_queue_drain	Number of replica items per second being written to disk in this bucket
vb_replica_queue_fill	Number of replica items per second being put on the replica item disk queue in this bucket
vb_replica_queue_size	Number of replica items waiting to be written to disk in this bucket
vb_replica_resident_items_ratio	Percentage of replica items cached in RAM in this bucket
vb_total_queue_age	Sum of disk queue item age in milliseconds.

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 and 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 |
```

```
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

Couchbase Metric	Description	Response
• mem_used • ep_kv_size • ep_mem_high_wat	These four metrics together give insight into how memory is used by the data service. mem_used / ep_kv_size represents fragmentation within the KV engine. out mem_used is the actual memory utilization whereas ep_kv_size is the sum of the metadata and values expected to be in RAM.mem_used / memoryTotal should be less than 90%. ep_kv_size / ep_mem_high_wat	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.

	quota utilization. ep_mem_high_wat is the maximum RAM the bucket is expected to use.	
ep_meta_data_memory	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.	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	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.	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	The number of items currently being written to disk. Combined with ep_queue_size, this represents the total disk write queue on the server.	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	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.	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	The number of items in the inter-node replication queue. This represents the cluster's exposure to potential data	Create a baseline for this value as "normal" will be dependent upon your workload and available network IO. Alert at 2x of baseline.

	cluster's exposure to potential data loss.	Alert at 2x of baseline.
ops	The total number of KV operations occurring against the node.	Create a baseline for this value as "normal" 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.
cmd_get	The number of KV GET operations occurring against the node.	Create a baseline for this value as "normal" 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.
cmd_set	The number of KV SET operations occurring against the node.	Create a baseline for this value as "normal" 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.
delete_hits	The number of KV DELETE operations occurring against the node.	Create a baseline for this value as "normal" 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.
ep_bg_fetched	The number of items fetched from disk (cache misses).	This value should be close to 0. Establish a baseline for this metric and alert at 2x of baseline.
curr_connections	The number of client (SDK) connections to Couchbase. More connections will result in increased CPU utilization.	Create a baseline for your environment. Alert at 2x of baseline. Couchbase will begin rejecting connections above 30,000.
curr_items	The number of items currently active on this node. During warmup, this will be 0 until complete.	Once a baseline number of objects has been established, substantial changes to the baseline could indicate unexpected failures within Couchbase or an application bug
vb_active_resident_items_ratio	The percentage of active data in that is memory resident.	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%.
	The percentage of replica data in that	

vb_replica_resident_items_ratio	resident. A higher percentage for this value will ensure lower latency data access following a failover.	on business requirements for object latency during a failure scenario. The value should never be less than 15%
ep_tmp_oom_errors	Number of times temporary OOMs were sent to a client. Represents high transient memory pressure within the system.	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.
ep_oom_errors	Number of times permanent OOMs were sent to a client. Represents very high consistent memory pressure within the system.	This error indicates the bucket has exceeded its total memory allocation and immediately requires additional memory or nodes be added.
• 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.

vb_active_num	active vBuckets.	indicates a node failure and that a failover + rebalance is required.

Example

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

```
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.

```
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.

```
# loop over each of the buckets
for bucket in $(curl \
  --user Administrator:password \
  --silent \
  --request GET \
```

```
  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

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

`GET` Cluster Eventing 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/@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: " +
```

```
    (.["eventing/n1ql_op_exception_count"] | add / length | tostring) +
  "\n  eventing/on_delete_failure: " +
    (.["eventing/on_delete_failure"] | add / length | tostring) +
  "\n  eventing/on_delete_success: " +
    (.["eventing/on_delete_success"] | add / length | tostring) +
  "\n  eventing/on_update_failure: " +
    (.["eventing/on_update_failure"] | add / length | tostring) +
  "\n  eventing/on_update_success: " +
    (.["eventing/on_update_success"] | add / length | tostring) +
  "\n  eventing/processed_count: " +
    (.["eventing/processed_count"] | add / length | tostring) +
  "\n  eventing/timeout_count: " +
    (.["eventing/timeout_count"] | add / length | tostring)'
```

Example: Stats for Each Node Separately

```
# loop over each of the buckets
for node in $(curl \
  --user Administrator:password \
  --silent \
  --request GET \
  http://localhost:8091/pools/nodes | \
  jq -r '.nodes[] |
    select(.services | contains(["eventing"]) == true) |
    .hostname'
  )
do
  echo "$node Function Stats"
  echo "------------------------------------------------------"
  # get the eventing stats for the specific node
  curl \
    --user Administrator:password \
    --silent \
    --request GET \
    --data zoom=minute \
    http://localhost:8091/pools/default/buckets/@eventing/nodes/$node/stats
| \
    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	Description	Response
eventing/bucket_op_exception_count eventing/failed_count eventing/n1ql_op_exception_count eventing/on_delete_failure eventing/on_update_failure eventing/timeout_count	Any exceptions/failures should be monitored	For this value "normal" is 0, any value other than 0 would indicate exceptions are being thrown and should be investigated
eventing/dcp_backlog	The number of items to be processed.	Create a baseline for this value as "normal" will be dependent upon your workload and number of functions. Alert at 2x of baseline.

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

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

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 \
  --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 == 3) |
    "  " + (.key) + ": " +
      (.value | add / length | tostring)'
```

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 \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  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 | tostri
ng) +
    "\n  checkpoint_failure_count: " +
      ($stats["eventing/" + . + "/checkpoint_failure_count"] | add | tostrin
g) +
    "\n  dcp_backlog: " +
      ($stats["eventing/" + . + "/dcp_backlog"] | add | tostring) +
    "\n  failed_count: " +
      ($stats["eventing/" + . + "/failed_count"] | add | tostring) +
    "\n  n1ql_op_exception_count: " +
      ($stats["eventing/" + . + "/n1ql_op_exception_count"] | add | tostring
) +
    "\n  on_delete_failure: " +
      ($stats["eventing/" + . + "/on_delete_failure"] | add / length | tostr
ing) +
    "\n  on_delete_success: " +
      ($stats["eventing/" + . + "/on_delete_success"] | add / length  | tost
ring) +
    "\n  on_update_failure: " +
      ($stats["eventing/" + . + "/on_update_failure"] | add / length | tostr
ing) +
    "\n  on_update_success: " +
      ($stats["eventing/" + . + "/on_update_success"] | add / length | tostr
ing) +
    "\n  processed_count: " +
      ($stats["eventing/" + . + "/processed_count"] | add / length | tostrin
g) +
    "\n  timeout_count: " +
      ($stats["eventing/" + . + "/timeout_count"] | add | tostring)
    '
```

Key Metrics to Monitor

Couchbase Metric	Description	Response
eventing/{func_name}/bucket_op_exception_count eventing/{func_name}/failed_count eventing/{func_name}/n1ql_op_exception_count eventing/{func_name}/on_delete_failure eventing/{func_name}/on_update_failure	Any exceptions/failures should be monitored	For this value "normal" is 0, any value other than 0 would indicate exceptions are being thrown and should be

eventing/{func_name}/on_update_failure eventing/{func_name}/timeout_count	monitored	thrown and should be investigated
eventing/{func_name}/dcp_backlog	The number of items to be processed.	Create a baseline for this value as "normal" will be dependent upon your workload and number of functions. Alert at 2x of baseline.

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

FTS Service-Level Stats

Available Stats

Stat name	Description
fts_curr_batches_blocked_by_herder	The number of batches blocked by the herder
fts_num_bytes_used_ram	The number of bytes used in memory for the FTS service.
fts_total_queries_rejected_by_herder	The number of queries rejected by the herder

---

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 \
```

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

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'
  )
```

```
do
  echo "$node FTS Stats"
  echo "------------------------------------------------------------"
  # get the FTS stats for the specific node
  curl \
    --user Administrator:password \
    --silent \
    --request GET \
    --data zoom=minute \
    http://localhost:8091/pools/default/buckets/@fts/nodes/$node/stats | \
    jq -r '.op.samples |
      "fts_num_bytes_used_ram:  " + (.fts_num_bytes_used_ram | add / length
| tostring)'
done
```

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

Stat name	Description
fts/{indexName}/avg_queries_latency	The average query latency in milliseconds
fts/{indexName}/doc_count	The number of documents in the index
fts/{indexName}/num_bytes_used_disk	Total disk file size used by the index
fts/{indexName}/num_files_on_disk	Number of files for the index on disk
fts/{indexName}/num_mutations_to_index	The number of documents pending indexing
fts/{indexName}/num_pindexes_actual	Number of index partitions (including replica partitions)
fts/{indexName}/num_pindexes_target	Number of index partitions expected (including replica partitions)
fts/{indexName}/num_recs_to_persist	Number of index records not yet persisted to disk
fts/{indexName}/num_root_filesegments	The number of root file segments
fts/{indexName}/num_root_memorysegments	The number of root memory segments
fts/{indexName}/total_bytes_indexed	Number of fts bytes indexed per second
fts/{indexName}/total_bytes_query_results	Number of bytes returned in results per second
fts/{indexName}/total_compaction_written_bytes	Number of compaction bytes written per second
fts/{indexName}/total_queries	The number of queries per second

fts/{indexName}/total_queries_error	The number of query errors per second
fts/{indexName}/total_queries_slow	The number of slow queries per second (>5s)
fts/{indexName}/total_queries_timeout	The number of queries per second that resulted in a timeout
fts/{indexName}/total_request_time	Total time spent servicing requests
fts/{indexName}/total_term_searchers	Number of term searchers started per second

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

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_to_persist: " +
    (.["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 | tost
ring) +
  "\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: " +
```

```
    (.["fts/" + $index + "/total_queries_timeout"] | add | tostring) +
  "\n  total_request_time+queued: " +
    (.["fts/" + $index + "/total_request_time"] | add | tostring) +
  "\n  total_term_searchers: " +
    (.["fts/" + $index + "/total_term_searchers"] | add | tostring)'
```

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 -r '.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/$NODE/sta
ts | \
    # 1. reduce the samples object, by looping over each property, only work
 with properties
    # who are index specific stat properties and either sum or average sampl
es
    # 2. get all of the unique index keys
    # 3. loop over each index and output the stats
    jq -r '
      reduce (.op.samples | to_entries[]) as {$key, $value} (
        {};
        if (
          $key | split("/") | length == 3
```

```
          and ($key | contains("replica ") | not)
        ) then
          if ([
            "num_mutations_to_index","num_pindexes_actual",
            "num_pindexes_target","num_recs_to_persist","total_queries",
            "total_queries_error","total_queries_slow","total_queries_timeou
t",
            "total_request_time+queued","total_term_searchers"
          ] | .[] | contains($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: "
        + ($stats["fts\/" + . + "\/num_pindexes_target"] | tostring) +
      "\n  num_recs_to_persist: "
        + ($stats["fts\/" + . + "\/num_recs_to_persist"] | 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: "
        + ($stats["fts\/" + . + "\/total_compaction_written_bytes"] | tostri
ng) +
      "\n  total_queries: "
        + ($stats["fts\/" + . + "\/total_queries"] | tostring) +
      "\n  total_queries_error: "
        + ($stats["fts\/" + . + "\/total_queries_error"] | tostring) +
```

```
      "\n  total_queries_slow: "
        + ($stats["fts\/" + . + "\/total_queries_slow"] | tostring) +
      "\n  total_queries_timeout: "
        + ($stats["fts\/" + . + "\/total_queries_timeout"] | tostring) +
      "\n  total_request_time: "
        + ($stats["fts\/" + . + "\/total_request_time"] | tostring) +
      "\n  total_term_searchers: "
        + ($stats["fts\/" + . + "\/total_term_searchers"] | tostring) +
      "\n"
    '
done
```

Key Metrics to Monitor

Couchbase Metric	Description	Response
avg_queries_latency	The average query latency	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.
total_queries	The number of query requests to the index	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.
total_queries_error total_queries_timeout	The number of query errors to the index	Alert at any value greater than 0 as this indicates failed requests.

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

Stat name	Description
fts/doc_count	The number of documents in all fts indexes
fts/num_bytes_used_disk	Total disk file size used by the indexes
fts/num_files_on_disk	The number of index files on disk
fts/num_mutations_to_index	The number of documents pending indexing
fts/num_pindexes_actual	Number of index partitions (including replica partitions)
fts/num_pindexes_target	Number of index partitions expected (including replica partitions)
fts/num_recs_to_persist	Number of index records not yet persisted to disk
fts/num_root_filesegments	Number of root file segments

fts/num_root_filesegments	Number of root file segments
fts/num_root_memorysegments	Number of root memory segments
fts/total_bytes_indexed	Number of fts bytes indexed per second
fts/total_bytes_query_results	Number of bytes returned in results per second
fts/total_compaction_written_bytes	Number of compaction bytes written per second
fts/total_queries	The number of queries per second
fts/total_queries_error	The number of query errors per second
fts/total_queries_slow	The number of slow queries per second (>5s)
fts/total_queries_timeout	The number of queries per second that resulted in a timeout
fts/total_request_time	
fts/total_term_searchers	Number of term searchers started per second

`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.

```
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 |
  "  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 | tostri
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 / le
ngth | tostring) +
  "\n  total_request_time: " + (.["fts/total_request_time"] | add | tostring
) +
  "\n  total_term_searchers: " + (.["fts/total_term_searchers"] | add | tost
ring)'
```

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) +
  "\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 | tostri
```

```
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 / le
ngth | tostring) +
  "\n  total_request_time: " + (.["fts/total_request_time"] | add | tostring
) +
  "\n  total_term_searchers: " + (.["fts/total_term_searchers"] | add | tost
ring)'
```

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:


```
{
  "indexes": [{
    "storageMode": "plasma",
    "partitioned": false,
    "instId": 4607548507687231469,
    "hosts": ["127.0.0.1:8091"],
    "progress": 100,
    "definition": "CREATE INDEX `def_airportname` ON `travel-sample`(`airpor
tname`) 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

Couchbase Metric	Description	Response
status	Indicates whether a index is in a "Ready" or "Building" state.	Alert if the value is not "Ready" or "Building".

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(
    (.status != "Ready" and .status != "Building")
  )) | .[] | .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

Stat name	Description
index_memory_quota	The cluster wide memory quota.
index_memory_used	The amount of memory currently used by the indexing service.
index_ram_percent	The percentage of index entries in ram.
index_remaining_ram	The amount of memory remaining.

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

```
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.

```
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)
 +
  "\n  index_memory_used: " + (.index_memory_used | add / length | tostring)
 +
  "\n  index_ram_percent: " + (.index_ram_percent | add / length | tostring)
```

```
 +
  "\n  index_remaining_ram: " + (.index_remaining_ram | add / length | tostr
ing)'
```

Example: Stats for Each Node Separately

```
# loop over each of the buckets
for node in $(curl \
  --user Administrator:password \
  --silent \
  --request GET \
  http://localhost:8091/pools/nodes | \
  jq -r '.nodes[] |
    select(.services | contains(["index"]) == true) |
    .hostname'
  )
do
  echo "$node Index Stats"
  echo "------------------------------------------------------"
  # get the index stats for the specific node
  curl \
    --user Administrator:password \
    --silent \
    --request GET \
    --data zoom=minute \
    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

Couchbase Metric	Description	Response
index_remaining_ram	The amount of memory remaining.	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.

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

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

`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.

```
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.

```
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/sta
```

```
ts | \
  jq -r --arg index "$INDEX" '.op.samples |
  "  avg_item_size: " + (.["index/" + $index + "/avg_item_size"] | add / len
gth | tostring) +
  "\n  avg_scan_latency: " + (.["index/" + $index + "/avg_scan_latency"] | a
dd / length | tostring) +
  "\n  cache_hits: " + (.["index/" + $index + "/cache_hits"] | add | tostrin
g) +
  "\n  cache_miss_ratio: " + (.["index/" + $index + "/cache_miss_ratio"] | a
dd / length | tostring) +
  "\n  cache_misses: " + (.["index/" + $index + "/cache_misses"] | add | tos
tring) +
  "\n  data_size: " + (.["index/" + $index + "/data_size"] | add / length |
tostring) +
  "\n  disk_overhead_estimate: " + (.["index/" + $index + "/disk_overhead_es
timate"] | add / length | tostring) +
  "\n  disk_size: " + (.["index/" + $index + "/disk_size"] | add / length |
tostring) +
  "\n  frag_percent: " + (.["index/" + $index + "/frag_percent"] | add / len
gth | tostring) +
  "\n  index_frag_percent: " + (.["index/" + $index + "/index_frag_percent"]
 | add / length | tostring) +
  "\n  index_resident_percent: " + (.["index/" + $index + "/index_resident_p
ercent"] | add / length | tostring) +
  "\n  items_count: " + (.["index/" + $index + "/items_count"] | add / lengt
h | tostring) +
  "\n  memory_used: " + (.["index/" + $index + "/memory_used"] | add / lengt
h | tostring) +
  "\n  num_docs_indexed: " + (.["index/" + $index + "/num_docs_indexed"] | a
dd | tostring) +
  "\n  num_docs_pending+queued: " + (.["index/" + $index + "/num_docs_pendin
g+queued"] | add | tostring) +
  "\n  num_docs_queued: " + (.["index/" + $index + "/num_docs_queued"] | add
 | tostring) +
  "\n  num_requests: " + (.["index/" + $index + "/num_requests"] | add | tos
tring) +
  "\n  num_rows_returned: " + (.["index/" + $index + "/num_rows_returned"] |
 add | tostring) +
  "\n  recs_in_mem: " + (.["index/" + $index + "/recs_in_mem"] | add / lengt
h | tostring) +
  "\n  recs_on_disk: " + (.["index/" + $index + "/recs_on_disk"] | add / len
gth | 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/$NODE/s
tats | \
    # 1. reduce the samples object, by looping over each property, only work
 with properties
    # who are index specific stat properties and either sum or average sampl
es
    # 2. get all of the unique index keys
    # 3. loop over each index and output the stats
    jq -r 'reduce (.op.samples | to_entries[]) as {$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",
```

```
            "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"] | t
ostring ) +
      "\n  avg_scan_latency: " + ($stats["index\/" + . + "\/avg_scan_latency
"] | tostring) +
      "\n  cache_hits: " + ($stats["index\/" + . + "\/cache_hits"] | tostrin
g) +
      "\n  cache_miss_ratio: " + ($stats["index\/" + . + "\/cache_miss_ratio
"] | tostring) +
      "\n  cache_misses: " + ($stats["index\/" + . + "\/cache_misses"] | tos
tring) +
      "\n  data_size: " + ($stats["index\/" + . + "\/data_size"] | tostring)
 +
      "\n  disk_overhead_estimate: " + ($stats["index\/" + . + "\/disk_overh
ead_estimate"] | tostring) +
      "\n  disk_size: " + ($stats["index\/" + . + "\/disk_size"] | tostring)
 +
      "\n  frag_percent: " + ($stats["index\/" + . + "\/frag_percent"] | tos
tring) +
      "\n  index_frag_percent: " + ($stats["index\/" + . + "\/index_frag_per
cent"] | tostring) +
      "\n  index_resident_percent: " + ($stats["index\/" + . + "\/index_resi
dent_percent"] | tostring) +
      "\n  items_count: " + ($stats["index\/" + . + "\/items_count"] | tostr
ing) +
      "\n  memory_used: " + ($stats["index\/" + . + "\/memory_used"] | tostr
ing) +
      "\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) +
```

```
      "\n  num_docs_queued: " + ($stats["index\/" + . + "\/num_docs_queued"]
 | tostring) +
      "\n  num_requests: " + ($stats["index\/" + . + "\/num_requests"] | tos
tring) +
      "\n  num_rows_returned: " + ($stats["index\/" + . + "\/num_rows_return
ed"] | tostring) +
      "\n  recs_in_mem: " + ($stats["index\/" + . + "\/recs_in_mem"] | tostr
ing) +
      "\n  recs_on_disk: " + ($stats["index\/" + . + "\/recs_on_disk"] | tos
tring) +
      "\n  scan_bytes_read: " + ($stats["index\/" + . + "\/scan_bytes_read"]
 | tostring) +
      "\n  avg_scan_latency: " + ($stats["index\/" + . + "\/avg_scan_latency
"] | tostring) +
      "\n  total_scan_duration: " + ($stats["index\/" + . + "\/total_scan_du
ration"] | tostring) +
      "\n"
    '
done
```

Key Metrics to Monitor

Couchbase Metric	Description	Response
avg_item_size	The average index entry size	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.
avg_scan_latency	The average scan latency	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.
index_resident_percent	The percentage of the index that is memory resident	Create a baseline for this value as "normal" will depend on SLAs and hard configuration. Alert at 5-10% deviation of the baseline.
num_requests	The number of index scan requests to the index	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.

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

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

`GET` Cluster Index Aggregate Stats

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

```
BUCKET="travel-sample"

# 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/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/sta
ts | \
  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 | tostrin
g) +
  "\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)
+
```

```
  "\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 | tost
ring)
  '
```

Monitoring: Logs

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

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

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 are available via a REST API as well.

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

API Parameters

The Logs API supports the following query string parameters

Param	Description
limit	An integer greater than 0 that limits the overall number of messages returned
sinceTime	Epoch timestamp in milliseconds to start returning messages from

Log Response Properties

Property	Description
code	A code specified by the module or 0
module	The module that generated the log message
node	The node that the message came from
serverTime	An ISO-8601 timestamp of when the message was logged
shortText	A short string describing the log entry, most commonly "message", "node up", or "node down"
text	The detailed log message
tstamp	An Epoch timestamp of when the message was logged
type	The type of log message, values can be: info, warning, critical

Example: All Log Messages

```
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

```
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

Property	Description
msg	The alert message and details
serverTime	The time the alert was issued

Example: Retrieve All Alerts

```
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"]
    }]
}
```


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

Couchbase Metric	Description	Response
status	This is a meta metric that indicates overall node health.	Alert if the value is "unhealthy".
clusterMembership	Indicates whether the node is an active participant in cluster operations. Possible values are "active", "inactiveAdded", and "inactiveFailed".	Alert on "inactiveFailed" and investigate the cause of the node failure.

Example

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

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

Example

The following example displays the cluster membership of each node

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

Example

Show the services and system stats for each node cluster.

```
curl \
  --user Administrator:password \
  --silent \
    --request GET \
  http://localhost:8091/pools/nodes | \
  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) + "
%)"
   '
```

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

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

`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 \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  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 \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  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 | tostrin
g) +
  "\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)
```

```
+
  "\n  query_request_time: " + (.query_request_time | add | tostring) +
  "\n  query_requests: " + (.query_requests | add | tostring) +
  "\n  query_requests_1000ms: " + (.query_requests_1000ms | add | tostring)
+
  "\n  query_requests_250ms: " + (.query_requests_250ms | add | tostring) +
  "\n  query_requests_5000ms: " + (.query_requests_5000ms | add | tostring)
+
  "\n  query_requests_500ms: " + (.query_requests_500ms | add | tostring) +
  "\n  query_result_count: " + (.query_result_count | add | tostring) +
  "\n  query_result_size: " + (.query_result_size | add | tostring) +
  "\n  query_selects: " + (.query_selects | add | tostring) +
  "\n  query_service_time: " + (.query_service_time | add | tostring) +
  "\n  query_warnings: " + (.query_warnings | add | tostring)'
```

Example: Stats for Each Node Separately

```
# loop over each of the buckets
for node in $(curl \
  --user Administrator:password \
  --silent \
  --request GET \
  http://localhost:8091/pools/nodes | \
  jq -r '.nodes[] |
    select(.services | contains(["n1ql"]) == true) |
    .hostname'
  )
do
  echo "$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

Couchbase Metric	Description	Response
	The	

query_avg_svc_time	The average time of the query service for requests.	Create a baseline for this value, as "normal" 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.
query_requests	The number of query requests per second.	Create a baseline for this value, as "normal" will depend on workload. Alert at 2x of the baseline. This would indicate an increase in query traffic.

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.

OS Metric	Response
Free RAM	Free + cache memory should always be at least 20% of total system memory. If free + cache memory falls below 20%, scale the cluster.
Swap usage	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.
Memcached process RAM usage	Create a baseline for this value as "normal" 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.
Beam.smp process RAM usage	Create a baseline for this value as "normal" 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.
IO utilization (iostat)	Create a baseline for this value as "normal" will be dependent upon your workload and available disk IO. Overall sustained IO utilization should not exceed 90% of total IO capacity.
Total CPU utilization	Create a baseline for this value as "normal" will be dependent upon your workload. Sustained CPU utilization >90% indicates a need to scale the cluster.
Couchbase service CPU utilization	Create a baseline for these values as "normal" will be dependent upon your workload. Alert if this value exceeds 2x of baseline.
Beam.smp CPU utilization	Create a baseline for this value as "normal" will be dependent upon your workload. Alert if this value exceeds 2x of baseline
%steal CPU	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.
Network utilization	Create a baseline for this value as "normal" 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.
Presence of beam.smp process	Alert if beam.smp is not present. This indicates Couchbase is offline and needs to be restarted.
	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

Presence of service processes	restarted. Below are 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

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

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

```
curl \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  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.

```
NODE="172.17.0.2:8091"

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

```
  "\n  hibernated_waked: " + (.hibernated_waked | add / length | tostring) +
  "\n  mem_actual_free: " + (.mem_actual_free | add / length | tostring) +
  "\n  mem_actual_used: " + (.mem_actual_used | add / length | tostring) +
  "\n  mem_free: " + (.mem_free | add / length | tostring) +
  "\n  mem_total: " + (.mem_total | add / length | tostring) +
  "\n  mem_used_sys: " + (.mem_used_sys | add / length | tostring) +
  "\n  rest_requests: " + (.rest_requests | add / length | tostring) +
  "\n  swap_total: " + (.swap_total | add / length | tostring) +
  "\n  swap_used: " + (.swap_used | add / length | tostring)'
```

Example: Stats for Each Node Separately

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

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

Response:


```
[{
  "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

Couchbase Metric	Description	Response
status	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`

- Remote Cluster ID
- Source Bucket
- Target Bucket

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

Stat name	Description
replication_changes_left	The total number of changes left across all replications for the bucket
	The total number of documents in replication

replication_docs_rep_queue	The total number of documents in replication queue for all replications for the bucket
replications/{replicationId}/bandwidth_usage	Bandwidth used during replication, measured in bytes per second.
replications/{replicationId}/changes_left	Number of mutations to be replicated to the remote cluster
replications/{replicationId}/data_replicated	Size of data replicated in bytes
replications/{replicationId}/datapool_failed_gets	Number of failed gets from the pool
replications/{replicationId}/dcp_datach_length	
replications/{replicationId}/dcp_dispatch_time	
replications/{replicationId}/deletion_docs_written	The number of docs deleted that have been written to the target cluster
replications/{replicationId}/deletion_failed_cr_source	The number of deletes that have failed conflict resolution on the source due to optimistic replication
replications/{replicationId}/deletion_filtered	The number of deletes that have been filtered
replications/{replicationId}/deletion_received_from_dcp	The number of deletes that have been received from DCP
replications/{replicationId}/docs_checked	Number of documents checked for changes
replications/{replicationId}/docs_failed_cr_source	The number of docs that have failed conflict resolution on the source due to optimistic replication
replications/{replicationId}/docs_filtered	Number of documents that have been filtered out and not replicated to target cluster
replications/{replicationId}/docs_opt_repd	Number of documents sent optimistically
replications/{replicationId}/docs_processed	The number of documents processed
replications/{replicationId}/docs_received_from_dcp	Number of documents received from DCP
replications/{replicationId}/docs_rep_queue	Number of documents in replication queue
replications/{replicationId}/docs_unable_to_filter	The number of documents where filtering could not be processed
replications/{replicationId}/docs_written	Number of documents written to the target cluster
replications/{replicationId}/expiry_docs_written	The number of expiry documents written to the target cluster
replications/{replicationId}/expiry_failed_cr_source	The number of expiries that have failed conflict resolution on the source due to optimistic replication
expiry_filtered	The number of expiry documents that have been filtered out and not replicated to the target cluster
replications/{replicationId}/expiry_received_from_dcp	The number of expiry documents that have been received
	The number of expiry documents removed

replications/{replicationId}/expiry_stripped	The number of expiry documents removed from replicating
replications/{replicationId}/num_checkpoints	Number of checkpoints issued in replication queue
replications/{replicationId}/num_failedckpts	Number of checkpoints failed during replication
replications/{replicationId}/percent_completeness	Percentage of checked items out of all checked and to-be-replicated items
replications/{replicationId}/rate_doc_checks	
replications/{replicationId}/rate_doc_opt_repd	
replications/{replicationId}/rate_received_from_dcp	Number of documents received from DCP per second
replications/{replicationId}/rate_replicated	Rate of documents being replicated, measured in documents per second
replications/{replicationId}/resp_wait_time	
replications/{replicationId}/set_docs_written	The number of sets that have failed conflict resolution on the source due to optimistic replication
replications/{replicationId}/set_failed_cr_source	The number of sets that have failed conflict resolution on the source due to optimistic replication
replications/{replicationId}/set_filtered	Number of sets that have been filtered out and not replicated to target cluster
replications/{replicationId}/set_received_from_dcp	The number of sets that have been received from DCP
replications/{replicationId}/size_rep_queue	Size of replication queue in bytes
replications/{replicationId}/throttle_latency	Throttle latency
replications/{replicationId}/throughput_throttle_latency	Throughput throttle latency
replications/{replicationId}/time_committing	Seconds elapsed during replication
replications/{replicationId}/wtavg_docs_latency	Weighted average latency for sending replicated changes to target cluster
replications/{replicationId}/wtavg_meta_latency	Weighted average time for requesting document metadata. XDCR uses this for conflict resolution prior to sending the document into the replication queue

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) | .[] |
    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.

```
# 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/stats | \
    jq -r '.op.samples | to_entries | sort_by(.key) | .[] |
      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.

```
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.

```
# 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) |
```

```
      "  " + (.key) + ": " +
        (.value | add / length | tostring)'
done
```

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.

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

```
      jq -r '.op.samples | to_entries | sort_by(.key) | .[] |
        select(.key | split("/") | length > 1) |
        "  " + (.key) + ": " +
          (.value | add / length | tostring)'
    echo ""
  done
done
```

Key Metrics to Monitor

Couchbase Metric	Description	Response
changes_left	The number of items pending XDCR replication. This can be used to approximate the degree of eventual consistency between clusters.	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.
bandwidth_usage	The amount of bandwidth in bytes used for XDCR replication.	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.

`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

Example

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

```
# 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 \
```

```
  --user Administrator:password \
  --silent \
  $STAT_URL | \
  jq -r '.nodeStats | to_entries | .[] |
    (.key | split(":") | .[0]) + ": " + (.value | add / length | tostring)'
```

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.

```
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

Stat name	Description
ep_num_ops_get_meta	The number of metadata read operations per second for the bucket as the

~~ep_num_ops_get_meta~~	target for XDCR
ep_num_ops_set_meta	The number of set operations per second for the bucket as the target for XDCR
ep_num_ops_del_meta	The number of delete operations per second for the bucket as the target for XDCR
xdc_ops	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)

Example

```
curl \
  --user Administrator:password \
  --silent \
  --request GET \
  --data zoom=minute \
  http://localhost:8091/pools/default/buckets/travel-sample/stats | \
  jq -r '.op.samples |
      "ep_num_ops_get_meta: " + (.ep_num_ops_get_meta | add / length | tostr
ing) +
      "\nep_num_ops_set_meta: " + (.ep_num_ops_set_meta |add / length | tost
ring) +
      "\nep_num_ops_del_meta: " + (.ep_num_ops_del_meta |add / length | tost
ring) +
      "\nxdc_ops: " + (.xdc_ops |add / length | tostring)'
```

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

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

Available Stats

Stat name	Description
avg_active_timestamp_drift	
avg_replica_timestamp_drift	
ep_active_hlc_drift	The sum of total_abs_drift for the node's active vBuckets
ep_active_hlc_drift_count	The sum of total_abs_drift_count for the node's active vBuckets
ep_replica_hlc_drift	The sum of total_abs_drift for the node's active vBuckets
ep_replica_hlc_drift_count	The sum of total_abs_drift_count for the node's active vBuckets
ep_active_ahead_exceptions	The sum of drift_ahead_exceeded for the node's active vBuckets
ep_replica_ahead_exceptions	The sum of drift_ahead_exceeded for the node's replica vBuckets
ep_clock_cas_drift_threshold_exceeded	

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)'
```