Inspecting a Node

Contents

When inspection of a Riak node to gather metrics on performance or potential issues is desired, a number of tools are available to help, and are either included with Riak itself or made available through the Riak community.

This guide provides starting points and details on some of the available tools for inspecting a Riak node.

riak-admin status

`riak-admin status` is a subcommand of the `riak-admin` command that is included with every installation of Riak. The `status` subcommand provides data related to the current operating status for a node. The output of `riak-admin status` is categorized and detailed below.

Please note, for some counters such as `node_get_fsm_objsize` a minimum of 5 transactions is required for statistics to be generated.

Performance

Repeated runs of the `riak-admin status` command does not have a negative performance impact as the statstics are cached internally in Riak.

Active Stats

Active Stats represent current activity on the node.

Stat	Description
`pbc_active`	Number of active Protocol Buffers connections
`node_get_fsm_active`	Number of active GET FSMs
`node_put_fsm_active`	Number of active PUT FSMs
`index_fsm_active`	Number of active Secondary Index FSMs
`list_fsm_active`	Number of active Keylisting FSMs
`node_get_fsm_rejected`	Number of GET FSMs actively being rejected by Sidejob's overload protection
`node_put_fsm_rejected`	Number of PUT FSMs actively being rejected by Sidejob's overload protection

Average Stats

Average Stats represent an average calculated as (total occurrences / number of samples) since this node was started. In the below stats the sample time is 1s, giving us a per-second average. Currently, the only Average Stats are reported by Sidejob - an Erlang library that implements a parallel, capacity-limited request pool.

Stat	Description
`node_get_fsm_in_rate`	Average number of GET FSMs enqueued by Sidejob

`node_get_fsm_out_rate`	Average number of GET FSMs dequeued by Sidejob
`node_put_fsm_in_rate`	Average number of PUT FSMs enqueued by Sidejob
`node_put_fsm_out_rate`	Average number of PUT FSMs dequeued by Sidejob

One-Minute Stats

One-Minute Stats represent the number of times a particular activity has occurred within the last minute on this node.

General One-Minute Stats

Stat	Description
`node_gets`	Number of GETs coordinated by this node, including GETs to non-local vnodes in the last minute
`node_puts`	Number of PUTs coordinated by this node, including PUTs to non-local vnodes in the last minute
`vnode_gets`	Number of GET operations coordinated by local vnodes on this node in the last minute
`vnode_puts`	Number of PUT operations

	coordinated by local vnodes on this node in the last minute
`vnode_index_refreshes`	Number of secondary indexes refreshed on this node during secondary index anti-entropy in the last minute
`vnode_index_reads`	Number of local replicas participating in secondary index reads in the last minute
`vnode_index_writes`	Number of local replcias participating in secondary index writes in the last minute
`vnode_index_writes_postings`	Number of individual secondary index values written in the last minute
`vnode_index_deletes`	Number of local replicas participating in secondary index deletes in the last minute
`vnode_index_deletes_postings`	Number of individual secondary index values deleted in the last minute

`pbc_connects`	Number of Protocol Buffers connections made in the last minute
`node_get_fsm_active_60s`	Number of GET FSMs active in the last minute
`node_put_fsm_active_60s`	Number of PUT FSMs active in the last minute
`node_get_fsm_rejected_60s`	Number of GET FSMs rejected by Sidejob's overload protection in the last minute
`node_put_fsm_rejected_60s`	Number of PUT FSMs rejected by Sidejob's overload protection in the last minute
`index_fsm_create`	Number of Secondary Index query FSMs created in the last minute
`index_fsm_create_error`	Number of Secondary Index query FSM creation errors in the last minute
`list_fsm_create`	Number of Keylisting FSMs created in the last minute
`list_fsm_create_error`	Number of Keylisting

	FSM creation errors in the last minute
`read_repairs`	Number of read repair operations this node has coordinated in the last minute
`read_repairs_primary_outofdate_one`	Number of read repair operations performed on primary vnodes in the last minute due to stale replicas
`read_repairs_primary_notfound_one`	Number of read repair operations performed on primary vnodes in the last minute due to missing replicas
`read_repairs_fallback_outofdate_one`	Number of read repair operations performed on fallback vnodes in the last minute due to stale replicas
`read_repairs_fallback_notfound_one`	Number of read repair operations performed on fallback vnodes in the last minute due to missing replicas

FSM Time

FSM Time Stats represent the amount of time in microseconds required to traverse the GET or PUT Finite State Machine code, offering a picture of general node health. From your application's perspective, FSM Time effectively represents experienced latency. Mean, Median, and 95th-, 99th-, and 100th-percentile (Max) counters are

displayed. These are one-minute stats.

Stat	Description
`node_get_fsm_time_mean`	Mean time between reception of client GET request and subsequent response to client
`node_get_fsm_time_median`	Median time between reception of client GET request and subsequent response to client
`node_get_fsm_time_95`	95th percentile time between reception of client GET request and subsequent response to client
`node_get_fsm_time_99`	99th percentile time between reception of client GET request and subsequent response to client
`node_get_fsm_time_100`	100th percentile time between reception of client GET request and subsequent response to client
`node_put_fsm_time_mean`	Mean time between reception of client PUT request and subsequent response to client
`node_put_fsm_time_median`	Median time between reception of client PUT request and subsequent response to client
`node_put_fsm_time_95`	95th percentile time between reception of client PUT request and subsequent response to client
`node_put_fsm_time_99`	99th percentile time between reception of client PUT request and subsequent response to client

`node_put_fsm_time_100`	100th percentile time between reception of client PUT request and subsequent response to client

GET FSM Siblings

GET FSM Sibling Stats offer a count of the number of siblings encountered by this node on the occasion of a GET request. These are one-minute stats.

Stat	Description
`node_get_fsm_siblings_mean`	Mean number of siblings encountered during all GET operations by this node within the last minute
`node_get_fsm_siblings_median`	Median number of siblings encountered during all GET operations by this node within the last minute
`node_get_fsm_siblings_95`	95th percentile of siblings encountered during all GET operations by this node within the last minute
`node_get_fsm_siblings_99`	99th percentile of siblings encountered during all GET operations by this node within the last minute
`node_get_fsm_siblings_100`	100th percentile of siblings encountered during all GET operations by this node within the last minute

GET FSM Objsize

GET FSM Objsize Stats represent a view of the sizes of objects flowing through this node's GET FSMs. The size of an object is obtained by summing the length of the bucket name, key, serialized vector clock, value, and serialized metadata of each sibling. GET FSM Objsize and GET FSM Siblings are inextricably linked. These are one-minute stats.

Stat	Description
`node_get_fsm_objsize_mean`	Mean object size encountered by this node within the last minute
`node_get_fsm_objsize_median`	Median object size encountered by this node within the last minute
`node_get_fsm_objsize_95`	95th percentile object size encountered by this node within the last minute
`node_get_fsm_objsize_99`	99th percentile object size encountered by this node within the last minute
`node_get_fsm_objsize_100`	100th percentile object size encountered by this node within the last minute

Total Stats

Total Stats represent the total number of times a particular activity has occurred since this node was started.

Stat	Description
`node_gets_total`	Total number of GETs coordinated by this node,

	including GETs to non-local vnodes
`node_puts_total`	Total number of PUTs coordinated by this node, including PUTs to non-local vnodes
`vnode_gets_total`	Total number of GETs coordinated by local vnodes
`vnode_puts_total`	Total number of PUTS coordinated by local vnodes
`read_repairs_total`	Total number of Read Repairs this node has coordinated
`coord_redirs_total`	Total number of requests this node has redirected to other nodes for coordination
`vnode_index_refreshes_total`	Total number of indexes refreshed during secondary index anti-entropy
`vnode_index_reads_total`	Total number of local replicas participating in secondary index reads

`vnode_index_writes_total`	Total number of local replicas participating in secondary index writes
`vnode_index_writes_postings_total`	Total number of individual secondary index values written
`vnode_index_deletes_total`	Total number of local replicas participating in secondary index deletes
`vnode_index_deletes_postings_total`	Total number of individual secondary index values deleted
`pbc_connects_total`	Total number of Protocol Buffers connections made
`precommit_fail`	Total number of pre-commit hook failures
`postcommit_fail`	Total number of post-commit hook failures
`node_get_fsm_rejected_total`	Total number of GET FSMs rejected by Sidejob's overload protection

`node_put_fsm_rejected_total`	Total number of PUT FSMs rejected by Sidejob's overload protection
`read_repairs_primary_outofdate_count`	Total number of read repair operations performed on primary vnodes due to stale replicas
`read_repairs_primary_notfound_count`	Total number of read repair operations performed on primary vnodes due to missing replicas
`read_repairs_fallback_outofdate_count`	Total number of read repair operations performed on fallback vnodes due to stale replicas
`read_repairs_fallback_notfound_count`	Total number of read repair operations performed on fallback vnodes due to missing replicas

Timestamps

Some of the Erlang applications that Riak is comprised of contribute statistics to `riak-admin status`. The below timestamps record, in Epoch time, the last

time statistics for that application were generated.

Stat	Description
`riak_kv_stat_ts`	The last time Riak KV stats were generated.
`riak_pipe_stat_ts`	The last time Riak Pipe stats were generated.

Ring

General ring information is reported in `riak-admin status`.

Stat	Description
`ring_members`	List of nodes that are members of the ring
`ring_num_partitions`	The number of partitions in the ring
`ring_ownership`	List of all nodes in the ring and their associated partition ownership
`ring_creation_size`	Ring size this cluster was created with

CPU and Memory

CPU statistics are taken directly from Erlang's cpu_sup module. Documentation for which can be found at ErlDocs: cpu_sup.

Stat	Description
`cpu_nprocs`	Number of operating system processes
`cpu_avg1`	The average number of active processes for the last 1 minute (equivalent to top(1) command's

	load average when divided by 256())
`cpu_avg5`	The average number of active processes for the last 5 minutes (equivalent to top(1) command's load average when divided by 256())
`cpu_avg15`	The average number of active processes for the last 15 minutes (equivalent to top(1) command's load average when divided by 256())

Memory statistics are taken directly from the Erlang virtual machine. Documentation for which can be found at ErlDocs: Memory.

Stat	Description
`memory_total`	Total allocated memory (sum of processes and system)
`memory_processes`	Total amount of memory allocated for Erlang processes
`memory_processes_used`	Total amount of memory used by Erlang processes
`memory_system`	Total allocated memory that is not directly related to an Erlang process
`memory_atom`	Total amount of memory currently allocated for atom storage
`memory_atom_used`	Total amount of memory currently used for atom storage
`memory_binary`	Total amount of memory used for binaries
`memory_code`	Total amount of memory allocated for Erlang code

`memory_ets`	Total memory allocated for Erlang Term Storage
`mem_total`	Total available system memory
`mem_allocated`	Total memory allocated for this node

Erlang VM

The below statistics describe properties of the Erlang VM.

Stat	Description
`nodename`	The name this node uses to identify itself
`connected_nodes`	A list of the nodes that this node is aware of at this time
`sys_driver_version`	String representing the Erlang driver version in use by the runtime system
`sys_global_heaps_size`	Current size of the shared global heap
`sys_heap_type`	String representing the heap type in use (one of private, shared, hybrid)
`sys_logical_processors`	Number of logical processors available on the system
`sys_otp_release`	Erlang OTP release version in use on the node
`sys_process_count`	Number of processes currently running in the Erlang VM

`sys_smp_support`	Boolean value representing whether symmetric multi-processing (SMP) is available
`sys_system_version`	Detailed Erlang version information
`sys_system_architecture`	The node operating system and hardware architecture
`sys_threads_enabled`	Boolean value representing whether threads are enabled
`sys_thread_pool_size`	Number of threads in the asynchronous thread pool
`sys_wordsize`	Size of Erlang term words in bytes as an integer, for examples, on 32-bit architectures 4 is returned and on 64-bit architectures 8 is returned

Miscellaneous Information

Miscellaneous Information provide additional details particular to this node.

Stat	Description
`leveldb_read_block_error`	The number of LevelDB read block errors. Will read as undefined if LevelDB is not being used.
`disk`	Information about the disk, taken from Erlang's disksup module. Reported as [{“ID”,KBytes_Used,Percent_Util}].
`storage_backend`	The storage backend currently in use.

Pipeline Metrics

The following metrics from from riak_pipe are generated during MapReduce operations.

Stat	Description
`pipeline_active`	The number of pipelines active in the last 60 seconds
`pipeline_create_count`	The total number of pipelines created since the node was started
`pipeline_create_error_count`	The total number of pipeline creation errors since the node was started
`pipeline_create_error_one`	The number of pipelines created in the last 60 seconds
`pipeline_create_one`	The number of pipeline creation errors in the last 60 seconds

Application and Subsystem Versions

The specific version of each Erlang application and subsystem which makes up a Riak node is present in the `riak-admin status` output. Each application is linked below next to it's version identifier.

Stat	Description

`erlydtl_version`	ErlyDTL
`riak_control_version`	Riak Control
`cluster_info_version`	Cluster Information
`riak_search_version`	Riak Search
`merge_index_version`	Merge Index
`riak_kv_version`	Riak KV
`sidejob_version`	Sidejob
`riak_api_version`	Riak API
`riak_pipe_version`	Riak Pipe
`riak_core_version`	Riak Core
`bitcask_version`	Bitcask
`basho_stats_version`	Basho Stats
`webmachine_version`	Webmachine
`mochiweb_version`	MochiWeb
`inets_version`	inets
`erlang_js_version`	Erlang JS
`runtime_tools_version`	Erlang Runtime Tools
`os_mon_version`	Erlang Operating System Monitor
`riak_sysmon_version`	Riak System Monitor
`ssl_version`	Erlang Secure Sockets Layer (SSL)
`public_key_version`	Erlang Public Key

`crypto_version`	Erlang crypto
`sasl_version`	SASL
`lager_version`	Lager
`goldrush_version`	Goldrush
`compiler_version`	Erlang Compiler
`syntax_tools_version`	Erlang Syntax Tools
`stdlib_version`	Standard Library
`kernel_version`	Kernel

Riak Search Statistics

The following statistics related to Riak Search message queues are available.

Stat	Description
`riak_search_vnodeq_max`	Maximum number of unprocessed messages all virtual node (vnode) message queues in the Riak Search subsystem have received on this node in the last minute
`riak_search_vnodeq_mean`	Mean number of unprocessed messages all vnode message queues in the Riak Search subsystem have received on this node in the last minute
`riak_search_vnodeq_median`	Median number of unprocessed messages all vnode message

	queues in the Riak Search subsystem have received on this node in the last minute
`riak_search_vnodeq_min`	Minimum number of unprocessed messages all vnode message queues in the Riak Search subsystem have received on this node in the last minute
`riak_search_vnodeq_total`	Total number of unprocessed messages all vnode message queues in the Riak Search subsystem have received on this node since it was started
`riak_search_vnodes_running`	Total number of vnodes currently running in the Riak Search subsystem

Note that under ideal operation and with the exception of `riak_search_vnodes_running` these statistics should contain low values (e.g., 0-10). Presence of higher values could be indicative of an issue.

Riaknostic

Riaknostic is a small suite of diagnostic checks that can be run against a Riak node to discover common problems, and recommend how to resolve them. These checks are derived from the experience of the Basho Client Services Team as well as numerous public discussions on the mailing list, `#riak` IRC channel, and other online media.

As of Riak version 1.3, Riaknostic is installed by default.

Riaknostic is included with Riak and exposed through the `riak-admin`

`diag` command. It is an open source project developed by Basho Technologies and Riak community members. The code is available in the Riaknostic Github repository.

Related Resources

- Configuration and Management: Command Line Tools: riak-admin
- Riaknostic
- HTTP API Status

These May Also Interest You

- Downloads
- Log Messages FAQs
- Operating Riak FAQs
- Configuration Files
- Load Balancing and Proxy Configuration
- Configuring Secondary Indexes