# Cassandra.yaml in cassandra server

# Cassandra storage config YAML

# NOTE:

# See http://wiki.apache.org/cassandra/StorageConfiguration for

# full explanations of configuration directives

# /NOTE

# The name of the cluster. This is mainly used to prevent machines in

# one logical cluster from joining another.

**cluster\_name: 'Canvas Cluster'**

# This defines the number of tokens randomly assigned to this node on the ring

# The more tokens, relative to other nodes, the larger the proportion of data

# that this node will store. You probably want all nodes to have the same number

# of tokens assuming they have equal hardware capability.

#

# If you leave this unspecified, Cassandra will use the default of 1 token for legacy compatibility,

# and will use the initial\_token as described below.

#

# Specifying initial\_token will override this setting on the node's initial start,

# on subsequent starts, this setting will apply even if initial token is set.

#

# If you already have a cluster with 1 token per node, and wish to migrate to

# multiple tokens per node, see http://wiki.apache.org/cassandra/Operations

num\_tokens: 256

# Triggers automatic allocation of num\_tokens tokens for this node. The allocation

# algorithm attempts to choose tokens in a way that optimizes replicated load over

# the nodes in the datacenter for the replication strategy used by the specified

# keyspace.

#

# The load assigned to each node will be close to proportional to its number of

# vnodes.

#

# Only supported with the Murmur3Partitioner.

# allocate\_tokens\_for\_keyspace: KEYSPACE

# initial\_token allows you to specify tokens manually. While you can use it with

# vnodes (num\_tokens > 1, above) -- in which case you should provide a

# comma-separated list -- it's primarily used when adding nodes to legacy clusters

# that do not have vnodes enabled.

# initial\_token:

# See http://wiki.apache.org/cassandra/HintedHandoff

# May either be "true" or "false" to enable globally

hinted\_handoff\_enabled: true

# When hinted\_handoff\_enabled is true, a black list of data centers that will not

# perform hinted handoff

# hinted\_handoff\_disabled\_datacenters:

# - DC1

# - DC2

# this defines the maximum amount of time a dead host will have hints

# generated. After it has been dead this long, new hints for it will not be

# created until it has been seen alive and gone down again.

max\_hint\_window\_in\_ms: 10800000 # 3 hours

# Maximum throttle in KBs per second, per delivery thread. This will be

# reduced proportionally to the number of nodes in the cluster. (If there

# are two nodes in the cluster, each delivery thread will use the maximum

# rate; if there are three, each will throttle to half of the maximum,

# since we expect two nodes to be delivering hints simultaneously.)

hinted\_handoff\_throttle\_in\_kb: 1024

# Number of threads with which to deliver hints;

# Consider increasing this number when you have multi-dc deployments, since

# cross-dc handoff tends to be slower

max\_hints\_delivery\_threads: 2

# Directory where Cassandra should store hints.

# If not set, the default directory is $CASSANDRA\_HOME/data/hints.

# hints\_directory: /var/lib/cassandra/hints

# How often hints should be flushed from the internal buffers to disk.

# Will \*not\* trigger fsync.

hints\_flush\_period\_in\_ms: 10000

# Maximum size for a single hints file, in megabytes.

max\_hints\_file\_size\_in\_mb: 128

# Compression to apply to the hint files. If omitted, hints files

# will be written uncompressed. LZ4, Snappy, and Deflate compressors

# are supported.

#hints\_compression:

# - class\_name: LZ4Compressor

# parameters:

# -

# Maximum throttle in KBs per second, total. This will be

# reduced proportionally to the number of nodes in the cluster.

batchlog\_replay\_throttle\_in\_kb: 1024

# Authentication backend, implementing IAuthenticator; used to identify users

# Out of the box, Cassandra provides org.apache.cassandra.auth.{AllowAllAuthenticator,

# PasswordAuthenticator}.

#

# - AllowAllAuthenticator performs no checks - set it to disable authentication.

# - PasswordAuthenticator relies on username/password pairs to authenticate

# users. It keeps usernames and hashed passwords in system\_auth.roles table.

# Please increase system\_auth keyspace replication factor if you use this authenticator.

# If using PasswordAuthenticator, CassandraRoleManager must also be used (see below)

authenticator: AllowAllAuthenticator

# Authorization backend, implementing IAuthorizer; used to limit access/provide permissions

# Out of the box, Cassandra provides org.apache.cassandra.auth.{AllowAllAuthorizer,

# CassandraAuthorizer}.

#

# - AllowAllAuthorizer allows any action to any user - set it to disable authorization.

# - CassandraAuthorizer stores permissions in system\_auth.role\_permissions table. Please

# increase system\_auth keyspace replication factor if you use this authorizer.

authorizer: AllowAllAuthorizer

# Part of the Authentication & Authorization backend, implementing IRoleManager; used

# to maintain grants and memberships between roles.

# Out of the box, Cassandra provides org.apache.cassandra.auth.CassandraRoleManager,

# which stores role information in the system\_auth keyspace. Most functions of the

# IRoleManager require an authenticated login, so unless the configured IAuthenticator

# actually implements authentication, most of this functionality will be unavailable.

#

# - CassandraRoleManager stores role data in the system\_auth keyspace. Please

# increase system\_auth keyspace replication factor if you use this role manager.

role\_manager: CassandraRoleManager

# Validity period for roles cache (fetching granted roles can be an expensive

# operation depending on the role manager, CassandraRoleManager is one example)

# Granted roles are cached for authenticated sessions in AuthenticatedUser and

# after the period specified here, become eligible for (async) reload.

# Defaults to 2000, set to 0 to disable caching entirely.

# Will be disabled automatically for AllowAllAuthenticator.

roles\_validity\_in\_ms: 2000

# Refresh interval for roles cache (if enabled).

# After this interval, cache entries become eligible for refresh. Upon next

# access, an async reload is scheduled and the old value returned until it

# completes. If roles\_validity\_in\_ms is non-zero, then this must be

# also.

# Defaults to the same value as roles\_validity\_in\_ms.

# roles\_update\_interval\_in\_ms: 2000

# Validity period for permissions cache (fetching permissions can be an

# expensive operation depending on the authorizer, CassandraAuthorizer is

# one example). Defaults to 2000, set to 0 to disable.

# Will be disabled automatically for AllowAllAuthorizer.

permissions\_validity\_in\_ms: 2000

# Refresh interval for permissions cache (if enabled).

# After this interval, cache entries become eligible for refresh. Upon next

# access, an async reload is scheduled and the old value returned until it

# completes. If permissions\_validity\_in\_ms is non-zero, then this must be

# also.

# Defaults to the same value as permissions\_validity\_in\_ms.

# permissions\_update\_interval\_in\_ms: 2000

# Validity period for credentials cache. This cache is tightly coupled to

# the provided PasswordAuthenticator implementation of IAuthenticator. If

# another IAuthenticator implementation is configured, this cache will not

# be automatically used and so the following settings will have no effect.

# Please note, credentials are cached in their encrypted form, so while

# activating this cache may reduce the number of queries made to the

# underlying table, it may not bring a significant reduction in the

# latency of individual authentication attempts.

# Defaults to 2000, set to 0 to disable credentials caching.

credentials\_validity\_in\_ms: 2000

# Refresh interval for credentials cache (if enabled).

# After this interval, cache entries become eligible for refresh. Upon next

# access, an async reload is scheduled and the old value returned until it

# completes. If credentials\_validity\_in\_ms is non-zero, then this must be

# also.

# Defaults to the same value as credentials\_validity\_in\_ms.

# credentials\_update\_interval\_in\_ms: 2000

# The partitioner is responsible for distributing groups of rows (by

# partition key) across nodes in the cluster. You should leave this

# alone for new clusters. The partitioner can NOT be changed without

# reloading all data, so when upgrading you should set this to the

# same partitioner you were already using.

#

# Besides Murmur3Partitioner, partitioners included for backwards

# compatibility include RandomPartitioner, ByteOrderedPartitioner, and

# OrderPreservingPartitioner.

#

partitioner: org.apache.cassandra.dht.Murmur3Partitioner

# Directories where Cassandra should store data on disk. Cassandra

# will spread data evenly across them, subject to the granularity of

# the configured compaction strategy.

# If not set, the default directory is $CASSANDRA\_HOME/data/data.

data\_file\_directories:

- /var/lib/cassandra/data

# commit log. when running on magnetic HDD, this should be a

# separate spindle than the data directories.

# If not set, the default directory is $CASSANDRA\_HOME/data/commitlog.

commitlog\_directory: /var/lib/cassandra/commitlog

# Enable / disable CDC functionality on a per-node basis. This modifies the logic used

# for write path allocation rejection (standard: never reject. cdc: reject Mutation

# containing a CDC-enabled table if at space limit in cdc\_raw\_directory).

cdc\_enabled: false

# CommitLogSegments are moved to this directory on flush if cdc\_enabled: true and the

# segment contains mutations for a CDC-enabled table. This should be placed on a

# separate spindle than the data directories. If not set, the default directory is

# $CASSANDRA\_HOME/data/cdc\_raw.

# cdc\_raw\_directory: /var/lib/cassandra/cdc\_raw

# Policy for data disk failures:

#

# die

# shut down gossip and client transports and kill the JVM for any fs errors or

# single-sstable errors, so the node can be replaced.

#

# stop\_paranoid

# shut down gossip and client transports even for single-sstable errors,

# kill the JVM for errors during startup.

#

# stop

# shut down gossip and client transports, leaving the node effectively dead, but

# can still be inspected via JMX, kill the JVM for errors during startup.

#

# best\_effort

# stop using the failed disk and respond to requests based on

# remaining available sstables. This means you WILL see obsolete

# data at CL.ONE!

#

# ignore

# ignore fatal errors and let requests fail, as in pre-1.2 Cassandra

disk\_failure\_policy: stop

# Policy for commit disk failures:

#

# die

# shut down gossip and Thrift and kill the JVM, so the node can be replaced.

#

# stop

# shut down gossip and Thrift, leaving the node effectively dead, but

# can still be inspected via JMX.

#

# stop\_commit

# shutdown the commit log, letting writes collect but

# continuing to service reads, as in pre-2.0.5 Cassandra

#

# ignore

# ignore fatal errors and let the batches fail

commit\_failure\_policy: stop

# Maximum size of the native protocol prepared statement cache

#

# Valid values are either "auto" (omitting the value) or a value greater 0.

#

# Note that specifying a too large value will result in long running GCs and possbily

# out-of-memory errors. Keep the value at a small fraction of the heap.

#

# If you constantly see "prepared statements discarded in the last minute because

# cache limit reached" messages, the first step is to investigate the root cause

# of these messages and check whether prepared statements are used correctly -

# i.e. use bind markers for variable parts.

#

# Do only change the default value, if you really have more prepared statements than

# fit in the cache. In most cases it is not neccessary to change this value.

# Constantly re-preparing statements is a performance penalty.

#

# Default value ("auto") is 1/256th of the heap or 10MB, whichever is greater

prepared\_statements\_cache\_size\_mb:

# Maximum size of the Thrift prepared statement cache

#

# If you do not use Thrift at all, it is safe to leave this value at "auto".

#

# See description of 'prepared\_statements\_cache\_size\_mb' above for more information.

#

# Default value ("auto") is 1/256th of the heap or 10MB, whichever is greater

thrift\_prepared\_statements\_cache\_size\_mb:

# Maximum size of the key cache in memory.

#

# Each key cache hit saves 1 seek and each row cache hit saves 2 seeks at the

# minimum, sometimes more. The key cache is fairly tiny for the amount of

# time it saves, so it's worthwhile to use it at large numbers.

# The row cache saves even more time, but must contain the entire row,

# so it is extremely space-intensive. It's best to only use the

# row cache if you have hot rows or static rows.

#

# NOTE: if you reduce the size, you may not get you hottest keys loaded on startup.

#

# Default value is empty to make it "auto" (min(5% of Heap (in MB), 100MB)). Set to 0 to disable key cache.

key\_cache\_size\_in\_mb:

# Duration in seconds after which Cassandra should

# save the key cache. Caches are saved to saved\_caches\_directory as

# specified in this configuration file.

#

# Saved caches greatly improve cold-start speeds, and is relatively cheap in

# terms of I/O for the key cache. Row cache saving is much more expensive and

# has limited use.

#

# Default is 14400 or 4 hours.

key\_cache\_save\_period: 14400

# Number of keys from the key cache to save

# Disabled by default, meaning all keys are going to be saved

# key\_cache\_keys\_to\_save: 100

# Row cache implementation class name. Available implementations:

#

# org.apache.cassandra.cache.OHCProvider

# Fully off-heap row cache implementation (default).

#

# org.apache.cassandra.cache.SerializingCacheProvider

# This is the row cache implementation available

# in previous releases of Cassandra.

# row\_cache\_class\_name: org.apache.cassandra.cache.OHCProvider

# Maximum size of the row cache in memory.

# Please note that OHC cache implementation requires some additional off-heap memory to manage

# the map structures and some in-flight memory during operations before/after cache entries can be

# accounted against the cache capacity. This overhead is usually small compared to the whole capacity.

# Do not specify more memory that the system can afford in the worst usual situation and leave some

# headroom for OS block level cache. Do never allow your system to swap.

#

# Default value is 0, to disable row caching.

row\_cache\_size\_in\_mb: 0

# Duration in seconds after which Cassandra should save the row cache.

# Caches are saved to saved\_caches\_directory as specified in this configuration file.

#

# Saved caches greatly improve cold-start speeds, and is relatively cheap in

# terms of I/O for the key cache. Row cache saving is much more expensive and

# has limited use.

#

# Default is 0 to disable saving the row cache.

row\_cache\_save\_period: 0

# Number of keys from the row cache to save.

# Specify 0 (which is the default), meaning all keys are going to be saved

# row\_cache\_keys\_to\_save: 100

# Maximum size of the counter cache in memory.

#

# Counter cache helps to reduce counter locks' contention for hot counter cells.

# In case of RF = 1 a counter cache hit will cause Cassandra to skip the read before

# write entirely. With RF > 1 a counter cache hit will still help to reduce the duration

# of the lock hold, helping with hot counter cell updates, but will not allow skipping

# the read entirely. Only the local (clock, count) tuple of a counter cell is kept

# in memory, not the whole counter, so it's relatively cheap.

#

# NOTE: if you reduce the size, you may not get you hottest keys loaded on startup.

#

# Default value is empty to make it "auto" (min(2.5% of Heap (in MB), 50MB)). Set to 0 to disable counter cache.

# NOTE: if you perform counter deletes and rely on low gcgs, you should disable the counter cache.

counter\_cache\_size\_in\_mb:

# Duration in seconds after which Cassandra should

# save the counter cache (keys only). Caches are saved to saved\_caches\_directory as

# specified in this configuration file.

#

# Default is 7200 or 2 hours.

counter\_cache\_save\_period: 7200

# Number of keys from the counter cache to save

# Disabled by default, meaning all keys are going to be saved

# counter\_cache\_keys\_to\_save: 100

# saved caches

# If not set, the default directory is $CASSANDRA\_HOME/data/saved\_caches.

saved\_caches\_directory: /var/lib/cassandra/saved\_caches

# commitlog\_sync may be either "periodic" or "batch."

#

# When in batch mode, Cassandra won't ack writes until the commit log

# has been fsynced to disk. It will wait

# commitlog\_sync\_batch\_window\_in\_ms milliseconds between fsyncs.

# This window should be kept short because the writer threads will

# be unable to do extra work while waiting. (You may need to increase

# concurrent\_writes for the same reason.)

#

# commitlog\_sync: batch

# commitlog\_sync\_batch\_window\_in\_ms: 2

#

# the other option is "periodic" where writes may be acked immediately

# and the CommitLog is simply synced every commitlog\_sync\_period\_in\_ms

# milliseconds.

commitlog\_sync: periodic

commitlog\_sync\_period\_in\_ms: 10000

# The size of the individual commitlog file segments. A commitlog

# segment may be archived, deleted, or recycled once all the data

# in it (potentially from each columnfamily in the system) has been

# flushed to sstables.

#

# The default size is 32, which is almost always fine, but if you are

# archiving commitlog segments (see commitlog\_archiving.properties),

# then you probably want a finer granularity of archiving; 8 or 16 MB

# is reasonable.

# Max mutation size is also configurable via max\_mutation\_size\_in\_kb setting in

# cassandra.yaml. The default is half the size commitlog\_segment\_size\_in\_mb \* 1024.

# This should be positive and less than 2048.

#

# NOTE: If max\_mutation\_size\_in\_kb is set explicitly then commitlog\_segment\_size\_in\_mb must

# be set to at least twice the size of max\_mutation\_size\_in\_kb / 1024

#

commitlog\_segment\_size\_in\_mb: 32

# Compression to apply to the commit log. If omitted, the commit log

# will be written uncompressed. LZ4, Snappy, and Deflate compressors

# are supported.

# commitlog\_compression:

# - class\_name: LZ4Compressor

# parameters:

# -

# any class that implements the SeedProvider interface and has a

# constructor that takes a Map<String, String> of parameters will do.

seed\_provider:

# Addresses of hosts that are deemed contact points.

# Cassandra nodes use this list of hosts to find each other and learn

# the topology of the ring. You must change this if you are running

# multiple nodes!

- class\_name: org.apache.cassandra.locator.SimpleSeedProvider

parameters:

# seeds is actually a comma-delimited list of addresses.

# Ex: "<ip1>,<ip2>,<ip3>"

**- seeds: "192.168.1.111,192.168.1.112"**

# For workloads with more data than can fit in memory, Cassandra's

# bottleneck will be reads that need to fetch data from

# disk. "concurrent\_reads" should be set to (16 \* number\_of\_drives) in

# order to allow the operations to enqueue low enough in the stack

# that the OS and drives can reorder them. Same applies to

# "concurrent\_counter\_writes", since counter writes read the current

# values before incrementing and writing them back.

#

# On the other hand, since writes are almost never IO bound, the ideal

# number of "concurrent\_writes" is dependent on the number of cores in

# your system; (8 \* number\_of\_cores) is a good rule of thumb.

**concurrent\_reads: 10000**

**concurrent\_writes: 10000**

**concurrent\_counter\_writes: 10000**

# For materialized view writes, as there is a read involved, so this should

# be limited by the less of concurrent reads or concurrent writes.

**concurrent\_materialized\_view\_writes: 10000**

# Maximum memory to use for sstable chunk cache and buffer pooling.

# 32MB of this are reserved for pooling buffers, the rest is used as an

# cache that holds uncompressed sstable chunks.

# Defaults to the smaller of 1/4 of heap or 512MB. This pool is allocated off-heap,

# so is in addition to the memory allocated for heap. The cache also has on-heap

# overhead which is roughly 128 bytes per chunk (i.e. 0.2% of the reserved size

# if the default 64k chunk size is used).

# Memory is only allocated when needed.

# file\_cache\_size\_in\_mb: 512

# Flag indicating whether to allocate on or off heap when the sstable buffer

# pool is exhausted, that is when it has exceeded the maximum memory

# file\_cache\_size\_in\_mb, beyond which it will not cache buffers but allocate on request.

# buffer\_pool\_use\_heap\_if\_exhausted: true

# The strategy for optimizing disk read

# Possible values are:

# ssd (for solid state disks, the default)

# spinning (for spinning disks)

# disk\_optimization\_strategy: ssd

# Total permitted memory to use for memtables. Cassandra will stop

# accepting writes when the limit is exceeded until a flush completes,

# and will trigger a flush based on memtable\_cleanup\_threshold

# If omitted, Cassandra will set both to 1/4 the size of the heap.

# memtable\_heap\_space\_in\_mb: 2048

# memtable\_offheap\_space\_in\_mb: 2048

# memtable\_cleanup\_threshold is deprecated. The default calculation

# is the only reasonable choice. See the comments on memtable\_flush\_writers

# for more information.

#

# Ratio of occupied non-flushing memtable size to total permitted size

# that will trigger a flush of the largest memtable. Larger mct will

# mean larger flushes and hence less compaction, but also less concurrent

# flush activity which can make it difficult to keep your disks fed

# under heavy write load.

#

# memtable\_cleanup\_threshold defaults to 1 / (memtable\_flush\_writers + 1)

# memtable\_cleanup\_threshold: 0.11

# Specify the way Cassandra allocates and manages memtable memory.

# Options are:

#

# heap\_buffers

# on heap nio buffers

#

# offheap\_buffers

# off heap (direct) nio buffers

#

# offheap\_objects

# off heap objects

memtable\_allocation\_type: heap\_buffers

# Limits the maximum Merkle tree depth to avoid consuming too much

# memory during repairs.

#

# The default setting of 18 generates trees of maximum size around

# 50 MiB / tree. If you are running out of memory during repairs consider

# lowering this to 15 (~6 MiB / tree) or lower, but try not to lower it

# too much past that or you will lose too much resolution and stream

# too much redundant data during repair. Cannot be set lower than 10.

#

# For more details see https://issues.apache.org/jira/browse/CASSANDRA-14096.

#

# repair\_session\_max\_tree\_depth: 18

# Total space to use for commit logs on disk.

#

# If space gets above this value, Cassandra will flush every dirty CF

# in the oldest segment and remove it. So a small total commitlog space

# will tend to cause more flush activity on less-active columnfamilies.

#

# The default value is the smaller of 8192, and 1/4 of the total space

# of the commitlog volume.

#

# commitlog\_total\_space\_in\_mb: 8192

# This sets the number of memtable flush writer threads per disk

# as well as the total number of memtables that can be flushed concurrently.

# These are generally a combination of compute and IO bound.

#

# Memtable flushing is more CPU efficient than memtable ingest and a single thread

# can keep up with the ingest rate of a whole server on a single fast disk

# until it temporarily becomes IO bound under contention typically with compaction.

# At that point you need multiple flush threads. At some point in the future

# it may become CPU bound all the time.

#

# You can tell if flushing is falling behind using the MemtablePool.BlockedOnAllocation

# metric which should be 0, but will be non-zero if threads are blocked waiting on flushing

# to free memory.

#

# memtable\_flush\_writers defaults to two for a single data directory.

# This means that two memtables can be flushed concurrently to the single data directory.

# If you have multiple data directories the default is one memtable flushing at a time

# but the flush will use a thread per data directory so you will get two or more writers.

#

# Two is generally enough to flush on a fast disk [array] mounted as a single data directory.

# Adding more flush writers will result in smaller more frequent flushes that introduce more

# compaction overhead.

#

# There is a direct tradeoff between number of memtables that can be flushed concurrently

# and flush size and frequency. More is not better you just need enough flush writers

# to never stall waiting for flushing to free memory.

#

#memtable\_flush\_writers: 2

# Total space to use for change-data-capture logs on disk.

#

# If space gets above this value, Cassandra will throw WriteTimeoutException

# on Mutations including tables with CDC enabled. A CDCCompactor is responsible

# for parsing the raw CDC logs and deleting them when parsing is completed.

#

# The default value is the min of 4096 mb and 1/8th of the total space

# of the drive where cdc\_raw\_directory resides.

# cdc\_total\_space\_in\_mb: 4096

# When we hit our cdc\_raw limit and the CDCCompactor is either running behind

# or experiencing backpressure, we check at the following interval to see if any

# new space for cdc-tracked tables has been made available. Default to 250ms

# cdc\_free\_space\_check\_interval\_ms: 250

# A fixed memory pool size in MB for for SSTable index summaries. If left

# empty, this will default to 5% of the heap size. If the memory usage of

# all index summaries exceeds this limit, SSTables with low read rates will

# shrink their index summaries in order to meet this limit. However, this

# is a best-effort process. In extreme conditions Cassandra may need to use

# more than this amount of memory.

index\_summary\_capacity\_in\_mb:

# How frequently index summaries should be resampled. This is done

# periodically to redistribute memory from the fixed-size pool to sstables

# proportional their recent read rates. Setting to -1 will disable this

# process, leaving existing index summaries at their current sampling level.

index\_summary\_resize\_interval\_in\_minutes: 60

# Whether to, when doing sequential writing, fsync() at intervals in

# order to force the operating system to flush the dirty

# buffers. Enable this to avoid sudden dirty buffer flushing from

# impacting read latencies. Almost always a good idea on SSDs; not

# necessarily on platters.

trickle\_fsync: false

trickle\_fsync\_interval\_in\_kb: 10240

# TCP port, for commands and data

# For security reasons, you should not expose this port to the internet. Firewall it if needed.

storage\_port: 7000

# SSL port, for encrypted communication. Unused unless enabled in

# encryption\_options

# For security reasons, you should not expose this port to the internet. Firewall it if needed.

ssl\_storage\_port: 7001

# Address or interface to bind to and tell other Cassandra nodes to connect to.

# You \_must\_ change this if you want multiple nodes to be able to communicate!

#

# Set listen\_address OR listen\_interface, not both.

#

# Leaving it blank leaves it up to InetAddress.getLocalHost(). This

# will always do the Right Thing \_if\_ the node is properly configured

# (hostname, name resolution, etc), and the Right Thing is to use the

# address associated with the hostname (it might not be).

#

# Setting listen\_address to 0.0.0.0 is always wrong.

#

**listen\_address: 192.168.1.111**

# Set listen\_address OR listen\_interface, not both. Interfaces must correspond

# to a single address, IP aliasing is not supported.

# listen\_interface: eth0

# If you choose to specify the interface by name and the interface has an ipv4 and an ipv6 address

# you can specify which should be chosen using listen\_interface\_prefer\_ipv6. If false the first ipv4

# address will be used. If true the first ipv6 address will be used. Defaults to false preferring

# ipv4. If there is only one address it will be selected regardless of ipv4/ipv6.

# listen\_interface\_prefer\_ipv6: false

# Address to broadcast to other Cassandra nodes

# Leaving this blank will set it to the same value as listen\_address

# broadcast\_address: 1.2.3.4

# When using multiple physical network interfaces, set this

# to true to listen on broadcast\_address in addition to

# the listen\_address, allowing nodes to communicate in both

# interfaces.

# Ignore this property if the network configuration automatically

# routes between the public and private networks such as EC2.

# listen\_on\_broadcast\_address: false

# Internode authentication backend, implementing IInternodeAuthenticator;

# used to allow/disallow connections from peer nodes.

# internode\_authenticator: org.apache.cassandra.auth.AllowAllInternodeAuthenticator

# Whether to start the native transport server.

# Please note that the address on which the native transport is bound is the

# same as the rpc\_address. The port however is different and specified below.

start\_native\_transport: true

# port for the CQL native transport to listen for clients on

# For security reasons, you should not expose this port to the internet. Firewall it if needed.

native\_transport\_port: 9042

# Enabling native transport encryption in client\_encryption\_options allows you to either use

# encryption for the standard port or to use a dedicated, additional port along with the unencrypted

# standard native\_transport\_port.

# Enabling client encryption and keeping native\_transport\_port\_ssl disabled will use encryption

# for native\_transport\_port. Setting native\_transport\_port\_ssl to a different value

# from native\_transport\_port will use encryption for native\_transport\_port\_ssl while

# keeping native\_transport\_port unencrypted.

# native\_transport\_port\_ssl: 9142

# The maximum threads for handling requests when the native transport is used.

# This is similar to rpc\_max\_threads though the default differs slightly (and

# there is no native\_transport\_min\_threads, idle threads will always be stopped

# after 30 seconds).

# native\_transport\_max\_threads: 128

#

# The maximum size of allowed frame. Frame (requests) larger than this will

# be rejected as invalid. The default is 256MB. If you're changing this parameter,

# you may want to adjust max\_value\_size\_in\_mb accordingly. This should be positive and less than 2048.

# native\_transport\_max\_frame\_size\_in\_mb: 256

# The maximum number of concurrent client connections.

# The default is -1, which means unlimited.

# native\_transport\_max\_concurrent\_connections: -1

# The maximum number of concurrent client connections per source ip.

# The default is -1, which means unlimited.

# native\_transport\_max\_concurrent\_connections\_per\_ip: -1

# Whether to start the thrift rpc server.

**start\_rpc: true**

# The address or interface to bind the Thrift RPC service and native transport

# server to.

#

# Set rpc\_address OR rpc\_interface, not both.

#

# Leaving rpc\_address blank has the same effect as on listen\_address

# (i.e. it will be based on the configured hostname of the node).

#

# Note that unlike listen\_address, you can specify 0.0.0.0, but you must also

# set broadcast\_rpc\_address to a value other than 0.0.0.0.

#

# For security reasons, you should not expose this port to the internet. Firewall it if needed.

**rpc\_address: 0.0.0.0**

# Set rpc\_address OR rpc\_interface, not both. Interfaces must correspond

# to a single address, IP aliasing is not supported.

# rpc\_interface: eth1

# If you choose to specify the interface by name and the interface has an ipv4 and an ipv6 address

# you can specify which should be chosen using rpc\_interface\_prefer\_ipv6. If false the first ipv4

# address will be used. If true the first ipv6 address will be used. Defaults to false preferring

# ipv4. If there is only one address it will be selected regardless of ipv4/ipv6.

# rpc\_interface\_prefer\_ipv6: false

# port for Thrift to listen for clients on

rpc\_port: 9160

# RPC address to broadcast to drivers and other Cassandra nodes. This cannot

# be set to 0.0.0.0. If left blank, this will be set to the value of

# rpc\_address. If rpc\_address is set to 0.0.0.0, broadcast\_rpc\_address must

# be set.

# broadcast\_rpc\_address: 1.2.3.4

**broadcast\_rpc\_address: 192.168.1.111**

# enable or disable keepalive on rpc/native connections

rpc\_keepalive: true

# Cassandra provides two out-of-the-box options for the RPC Server:

#

# sync

# One thread per thrift connection. For a very large number of clients, memory

# will be your limiting factor. On a 64 bit JVM, 180KB is the minimum stack size

# per thread, and that will correspond to your use of virtual memory (but physical memory

# may be limited depending on use of stack space).

#

# hsha

# Stands for "half synchronous, half asynchronous." All thrift clients are handled

# asynchronously using a small number of threads that does not vary with the amount

# of thrift clients (and thus scales well to many clients). The rpc requests are still

# synchronous (one thread per active request). If hsha is selected then it is essential

# that rpc\_max\_threads is changed from the default value of unlimited.

#

# The default is sync because on Windows hsha is about 30% slower. On Linux,

# sync/hsha performance is about the same, with hsha of course using less memory.

#

# Alternatively, can provide your own RPC server by providing the fully-qualified class name

# of an o.a.c.t.TServerFactory that can create an instance of it.

rpc\_server\_type: sync

# Uncomment rpc\_min|max\_thread to set request pool size limits.

#

# Regardless of your choice of RPC server (see above), the number of maximum requests in the

# RPC thread pool dictates how many concurrent requests are possible (but if you are using the sync

# RPC server, it also dictates the number of clients that can be connected at all).

#

# The default is unlimited and thus provides no protection against clients overwhelming the server. You are

# encouraged to set a maximum that makes sense for you in production, but do keep in mind that

# rpc\_max\_threads represents the maximum number of client requests this server may execute concurrently.

#

# rpc\_min\_threads: 16

# rpc\_max\_threads: 2048

# uncomment to set socket buffer sizes on rpc connections

# rpc\_send\_buff\_size\_in\_bytes:

# rpc\_recv\_buff\_size\_in\_bytes:

# Uncomment to set socket buffer size for internode communication

# Note that when setting this, the buffer size is limited by net.core.wmem\_max

# and when not setting it it is defined by net.ipv4.tcp\_wmem

# See also:

# /proc/sys/net/core/wmem\_max

# /proc/sys/net/core/rmem\_max

# /proc/sys/net/ipv4/tcp\_wmem

# /proc/sys/net/ipv4/tcp\_wmem

# and 'man tcp'

# internode\_send\_buff\_size\_in\_bytes:

# Uncomment to set socket buffer size for internode communication

# Note that when setting this, the buffer size is limited by net.core.wmem\_max

# and when not setting it it is defined by net.ipv4.tcp\_wmem

# internode\_recv\_buff\_size\_in\_bytes:

# Frame size for thrift (maximum message length).

thrift\_framed\_transport\_size\_in\_mb: 15

# Set to true to have Cassandra create a hard link to each sstable

# flushed or streamed locally in a backups/ subdirectory of the

# keyspace data. Removing these links is the operator's

# responsibility.

incremental\_backups: false

# Whether or not to take a snapshot before each compaction. Be

# careful using this option, since Cassandra won't clean up the

# snapshots for you. Mostly useful if you're paranoid when there

# is a data format change.

snapshot\_before\_compaction: false

# Whether or not a snapshot is taken of the data before keyspace truncation

# or dropping of column families. The STRONGLY advised default of true

# should be used to provide data safety. If you set this flag to false, you will

# lose data on truncation or drop.

auto\_snapshot: true

# Granularity of the collation index of rows within a partition.

# Increase if your rows are large, or if you have a very large

# number of rows per partition. The competing goals are these:

#

# - a smaller granularity means more index entries are generated

# and looking up rows withing the partition by collation column

# is faster

# - but, Cassandra will keep the collation index in memory for hot

# rows (as part of the key cache), so a larger granularity means

# you can cache more hot rows

column\_index\_size\_in\_kb: 64

# Per sstable indexed key cache entries (the collation index in memory

# mentioned above) exceeding this size will not be held on heap.

# This means that only partition information is held on heap and the

# index entries are read from disk.

#

# Note that this size refers to the size of the

# serialized index information and not the size of the partition.

column\_index\_cache\_size\_in\_kb: 2

# Number of simultaneous compactions to allow, NOT including

# validation "compactions" for anti-entropy repair. Simultaneous

# compactions can help preserve read performance in a mixed read/write

# workload, by mitigating the tendency of small sstables to accumulate

# during a single long running compactions. The default is usually

# fine and if you experience problems with compaction running too

# slowly or too fast, you should look at

# compaction\_throughput\_mb\_per\_sec first.

#

# concurrent\_compactors defaults to the smaller of (number of disks,

# number of cores), with a minimum of 2 and a maximum of 8.

#

# If your data directories are backed by SSD, you should increase this

# to the number of cores.

#concurrent\_compactors: 1

# Throttles compaction to the given total throughput across the entire

# system. The faster you insert data, the faster you need to compact in

# order to keep the sstable count down, but in general, setting this to

# 16 to 32 times the rate you are inserting data is more than sufficient.

# Setting this to 0 disables throttling. Note that this account for all types

# of compaction, including validation compaction.

compaction\_throughput\_mb\_per\_sec: 16

# When compacting, the replacement sstable(s) can be opened before they

# are completely written, and used in place of the prior sstables for

# any range that has been written. This helps to smoothly transfer reads

# between the sstables, reducing page cache churn and keeping hot rows hot

sstable\_preemptive\_open\_interval\_in\_mb: 50

# Throttles all outbound streaming file transfers on this node to the

# given total throughput in Mbps. This is necessary because Cassandra does

# mostly sequential IO when streaming data during bootstrap or repair, which

# can lead to saturating the network connection and degrading rpc performance.

# When unset, the default is 200 Mbps or 25 MB/s.

# stream\_throughput\_outbound\_megabits\_per\_sec: 200

# Throttles all streaming file transfer between the datacenters,

# this setting allows users to throttle inter dc stream throughput in addition

# to throttling all network stream traffic as configured with

# stream\_throughput\_outbound\_megabits\_per\_sec

# When unset, the default is 200 Mbps or 25 MB/s

# inter\_dc\_stream\_throughput\_outbound\_megabits\_per\_sec: 200

# How long the coordinator should wait for read operations to complete

read\_request\_timeout\_in\_ms: 5000

# How long the coordinator should wait for seq or index scans to complete

range\_request\_timeout\_in\_ms: 10000

# How long the coordinator should wait for writes to complete

write\_request\_timeout\_in\_ms: 2000

# How long the coordinator should wait for counter writes to complete

counter\_write\_request\_timeout\_in\_ms: 5000

# How long a coordinator should continue to retry a CAS operation

# that contends with other proposals for the same row

cas\_contention\_timeout\_in\_ms: 1000

# How long the coordinator should wait for truncates to complete

# (This can be much longer, because unless auto\_snapshot is disabled

# we need to flush first so we can snapshot before removing the data.)

truncate\_request\_timeout\_in\_ms: 60000

# The default timeout for other, miscellaneous operations

request\_timeout\_in\_ms: 10000

# How long before a node logs slow queries. Select queries that take longer than

# this timeout to execute, will generate an aggregated log message, so that slow queries

# can be identified. Set this value to zero to disable slow query logging.

slow\_query\_log\_timeout\_in\_ms: 500

# Enable operation timeout information exchange between nodes to accurately

# measure request timeouts. If disabled, replicas will assume that requests

# were forwarded to them instantly by the coordinator, which means that

# under overload conditions we will waste that much extra time processing

# already-timed-out requests.

#

# Warning: before enabling this property make sure to ntp is installed

# and the times are synchronized between the nodes.

cross\_node\_timeout: false

# Set keep-alive period for streaming

# This node will send a keep-alive message periodically with this period.

# If the node does not receive a keep-alive message from the peer for

# 2 keep-alive cycles the stream session times out and fail

# Default value is 300s (5 minutes), which means stalled stream

# times out in 10 minutes by default

# streaming\_keep\_alive\_period\_in\_secs: 300

# phi value that must be reached for a host to be marked down.

# most users should never need to adjust this.

# phi\_convict\_threshold: 8

# endpoint\_snitch -- Set this to a class that implements

# IEndpointSnitch. The snitch has two functions:

#

# - it teaches Cassandra enough about your network topology to route

# requests efficiently

# - it allows Cassandra to spread replicas around your cluster to avoid

# correlated failures. It does this by grouping machines into

# "datacenters" and "racks." Cassandra will do its best not to have

# more than one replica on the same "rack" (which may not actually

# be a physical location)

#

# CASSANDRA WILL NOT ALLOW YOU TO SWITCH TO AN INCOMPATIBLE SNITCH

# ONCE DATA IS INSERTED INTO THE CLUSTER. This would cause data loss.

# This means that if you start with the default SimpleSnitch, which

# locates every node on "rack1" in "datacenter1", your only options

# if you need to add another datacenter are GossipingPropertyFileSnitch

# (and the older PFS). From there, if you want to migrate to an

# incompatible snitch like Ec2Snitch you can do it by adding new nodes

# under Ec2Snitch (which will locate them in a new "datacenter") and

# decommissioning the old ones.

#

# Out of the box, Cassandra provides:

#

# SimpleSnitch:

# Treats Strategy order as proximity. This can improve cache

# locality when disabling read repair. Only appropriate for

# single-datacenter deployments.

#

# GossipingPropertyFileSnitch

# This should be your go-to snitch for production use. The rack

# and datacenter for the local node are defined in

# cassandra-rackdc.properties and propagated to other nodes via

# gossip. If cassandra-topology.properties exists, it is used as a

# fallback, allowing migration from the PropertyFileSnitch.

#

# PropertyFileSnitch:

# Proximity is determined by rack and data center, which are

# explicitly configured in cassandra-topology.properties.

#

# Ec2Snitch:

# Appropriate for EC2 deployments in a single Region. Loads Region

# and Availability Zone information from the EC2 API. The Region is

# treated as the datacenter, and the Availability Zone as the rack.

# Only private IPs are used, so this will not work across multiple

# Regions.

#

# Ec2MultiRegionSnitch:

# Uses public IPs as broadcast\_address to allow cross-region

# connectivity. (Thus, you should set seed addresses to the public

# IP as well.) You will need to open the storage\_port or

# ssl\_storage\_port on the public IP firewall. (For intra-Region

# traffic, Cassandra will switch to the private IP after

# establishing a connection.)

#

# RackInferringSnitch:

# Proximity is determined by rack and data center, which are

# assumed to correspond to the 3rd and 2nd octet of each node's IP

# address, respectively. Unless this happens to match your

# deployment conventions, this is best used as an example of

# writing a custom Snitch class and is provided in that spirit.

#

# You can use a custom Snitch by setting this to the full class name

# of the snitch, which will be assumed to be on your classpath.

**endpoint\_snitch: GossipingPropertyFileSnitch**

# controls how often to perform the more expensive part of host score

# calculation

dynamic\_snitch\_update\_interval\_in\_ms: 100

# controls how often to reset all host scores, allowing a bad host to

# possibly recover

dynamic\_snitch\_reset\_interval\_in\_ms: 600000

# if set greater than zero and read\_repair\_chance is < 1.0, this will allow

# 'pinning' of replicas to hosts in order to increase cache capacity.

# The badness threshold will control how much worse the pinned host has to be

# before the dynamic snitch will prefer other replicas over it. This is

# expressed as a double which represents a percentage. Thus, a value of

# 0.2 means Cassandra would continue to prefer the static snitch values

# until the pinned host was 20% worse than the fastest.

dynamic\_snitch\_badness\_threshold: 0.1

# request\_scheduler -- Set this to a class that implements

# RequestScheduler, which will schedule incoming client requests

# according to the specific policy. This is useful for multi-tenancy

# with a single Cassandra cluster.

# NOTE: This is specifically for requests from the client and does

# not affect inter node communication.

# org.apache.cassandra.scheduler.NoScheduler - No scheduling takes place

# org.apache.cassandra.scheduler.RoundRobinScheduler - Round robin of

# client requests to a node with a separate queue for each

# request\_scheduler\_id. The scheduler is further customized by

# request\_scheduler\_options as described below.

request\_scheduler: org.apache.cassandra.scheduler.NoScheduler

# Scheduler Options vary based on the type of scheduler

#

# NoScheduler

# Has no options

#

# RoundRobin

# throttle\_limit

# The throttle\_limit is the number of in-flight

# requests per client. Requests beyond

# that limit are queued up until

# running requests can complete.

# The value of 80 here is twice the number of

# concurrent\_reads + concurrent\_writes.

# default\_weight

# default\_weight is optional and allows for

# overriding the default which is 1.

# weights

# Weights are optional and will default to 1 or the

# overridden default\_weight. The weight translates into how

# many requests are handled during each turn of the

# RoundRobin, based on the scheduler id.

#

# request\_scheduler\_options:

# throttle\_limit: 80

# default\_weight: 5

# weights:

# Keyspace1: 1

# Keyspace2: 5

# request\_scheduler\_id -- An identifier based on which to perform

# the request scheduling. Currently the only valid option is keyspace.

# request\_scheduler\_id: keyspace

# Enable or disable inter-node encryption

# JVM defaults for supported SSL socket protocols and cipher suites can

# be replaced using custom encryption options. This is not recommended

# unless you have policies in place that dictate certain settings, or

# need to disable vulnerable ciphers or protocols in case the JVM cannot

# be updated.

# FIPS compliant settings can be configured at JVM level and should not

# involve changing encryption settings here:

# https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/FIPS.html

# \*NOTE\* No custom encryption options are enabled at the moment

# The available internode options are : all, none, dc, rack

#

# If set to dc cassandra will encrypt the traffic between the DCs

# If set to rack cassandra will encrypt the traffic between the racks

#

# The passwords used in these options must match the passwords used when generating

# the keystore and truststore. For instructions on generating these files, see:

# http://download.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html#CreateKeystore

#

server\_encryption\_options:

internode\_encryption: none

keystore: conf/.keystore

keystore\_password: cassandra

truststore: conf/.truststore

truststore\_password: cassandra

# More advanced defaults below:

# protocol: TLS

# algorithm: SunX509

# store\_type: JKS

# cipher\_suites: [TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA,TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA,TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA,TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA,TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA,TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA]

# require\_client\_auth: false

# require\_endpoint\_verification: false

# enable or disable client/server encryption.

client\_encryption\_options:

enabled: false

# If enabled and optional is set to true encrypted and unencrypted connections are handled.

optional: false

keystore: conf/.keystore

keystore\_password: cassandra

# require\_client\_auth: false

# Set trustore and truststore\_password if require\_client\_auth is true

# truststore: conf/.truststore

# truststore\_password: cassandra

# More advanced defaults below:

# protocol: TLS

# algorithm: SunX509

# store\_type: JKS

# cipher\_suites: [TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA,TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA,TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA,TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA,TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA,TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA]

# internode\_compression controls whether traffic between nodes is

# compressed.

# Can be:

#

# all

# all traffic is compressed

#

# dc

# traffic between different datacenters is compressed

#

# none

# nothing is compressed.

internode\_compression: dc

# Enable or disable tcp\_nodelay for inter-dc communication.

# Disabling it will result in larger (but fewer) network packets being sent,

# reducing overhead from the TCP protocol itself, at the cost of increasing

# latency if you block for cross-datacenter responses.

inter\_dc\_tcp\_nodelay: false

# TTL for different trace types used during logging of the repair process.

tracetype\_query\_ttl: 86400

tracetype\_repair\_ttl: 604800

# By default, Cassandra logs GC Pauses greater than 200 ms at INFO level

# This threshold can be adjusted to minimize logging if necessary

# gc\_log\_threshold\_in\_ms: 200

# If unset, all GC Pauses greater than gc\_log\_threshold\_in\_ms will log at

# INFO level

# UDFs (user defined functions) are disabled by default.

# As of Cassandra 3.0 there is a sandbox in place that should prevent execution of evil code.

enable\_user\_defined\_functions: false

# Enables scripted UDFs (JavaScript UDFs).

# Java UDFs are always enabled, if enable\_user\_defined\_functions is true.

# Enable this option to be able to use UDFs with "language javascript" or any custom JSR-223 provider.

# This option has no effect, if enable\_user\_defined\_functions is false.

enable\_scripted\_user\_defined\_functions: false

# The default Windows kernel timer and scheduling resolution is 15.6ms for power conservation.

# Lowering this value on Windows can provide much tighter latency and better throughput, however

# some virtualized environments may see a negative performance impact from changing this setting

# below their system default. The sysinternals 'clockres' tool can confirm your system's default

# setting.

windows\_timer\_interval: 1

# Enables encrypting data at-rest (on disk). Different key providers can be plugged in, but the default reads from

# a JCE-style keystore. A single keystore can hold multiple keys, but the one referenced by

# the "key\_alias" is the only key that will be used for encrypt opertaions; previously used keys

# can still (and should!) be in the keystore and will be used on decrypt operations

# (to handle the case of key rotation).

#

# It is strongly recommended to download and install Java Cryptography Extension (JCE)

# Unlimited Strength Jurisdiction Policy Files for your version of the JDK.

# (current link: http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html)

#

# Currently, only the following file types are supported for transparent data encryption, although

# more are coming in future cassandra releases: commitlog, hints

transparent\_data\_encryption\_options:

enabled: false

chunk\_length\_kb: 64

cipher: AES/CBC/PKCS5Padding

key\_alias: testing:1

# CBC IV length for AES needs to be 16 bytes (which is also the default size)

# iv\_length: 16

key\_provider:

- class\_name: org.apache.cassandra.security.JKSKeyProvider

parameters:

- keystore: conf/.keystore

keystore\_password: cassandra

store\_type: JCEKS

key\_password: cassandra

#####################

# SAFETY THRESHOLDS #

#####################

# When executing a scan, within or across a partition, we need to keep the

# tombstones seen in memory so we can return them to the coordinator, which

# will use them to make sure other replicas also know about the deleted rows.

# With workloads that generate a lot of tombstones, this can cause performance

# problems and even exaust the server heap.

# (http://www.datastax.com/dev/blog/cassandra-anti-patterns-queues-and-queue-like-datasets)

# Adjust the thresholds here if you understand the dangers and want to

# scan more tombstones anyway. These thresholds may also be adjusted at runtime

# using the StorageService mbean.

tombstone\_warn\_threshold: 1000

tombstone\_failure\_threshold: 100000

# Filtering and secondary index queries at read consistency levels above ONE/LOCAL\_ONE use a

# mechanism called replica filtering protection to ensure that results from stale replicas do

# not violate consistency. (See CASSANDRA-8272 and CASSANDRA-15907 for more details.) This

# mechanism materializes replica results by partition on-heap at the coordinator. The more possibly

# stale results returned by the replicas, the more rows materialized during the query.

replica\_filtering\_protection:

# These thresholds exist to limit the damage severely out-of-date replicas can cause during these

# queries. They limit the number of rows from all replicas individual index and filtering queries

# can materialize on-heap to return correct results at the desired read consistency level.

#

# "cached\_replica\_rows\_warn\_threshold" is the per-query threshold at which a warning will be logged.

# "cached\_replica\_rows\_fail\_threshold" is the per-query threshold at which the query will fail.

#

# These thresholds may also be adjusted at runtime using the StorageService mbean.

#

# If the failure threshold is breached, it is likely that either the current page/fetch size

# is too large or one or more replicas is severely out-of-sync and in need of repair.

cached\_rows\_warn\_threshold: 2000

cached\_rows\_fail\_threshold: 32000

# Log WARN on any multiple-partition batch size exceeding this value. 5kb per batch by default.

# Caution should be taken on increasing the size of this threshold as it can lead to node instability.

batch\_size\_warn\_threshold\_in\_kb: 5

# Fail any multiple-partition batch exceeding this value. 50kb (10x warn threshold) by default.

batch\_size\_fail\_threshold\_in\_kb: 50

# Log WARN on any batches not of type LOGGED than span across more partitions than this limit

unlogged\_batch\_across\_partitions\_warn\_threshold: 10

# Log a warning when compacting partitions larger than this value

compaction\_large\_partition\_warning\_threshold\_mb: 100

# GC Pauses greater than gc\_warn\_threshold\_in\_ms will be logged at WARN level

# Adjust the threshold based on your application throughput requirement

# By default, Cassandra logs GC Pauses greater than 200 ms at INFO level

gc\_warn\_threshold\_in\_ms: 1000

# Maximum size of any value in SSTables. Safety measure to detect SSTable corruption

# early. Any value size larger than this threshold will result into marking an SSTable

# as corrupted. This should be positive and less than 2048.

# max\_value\_size\_in\_mb: 256

# Back-pressure settings #

# If enabled, the coordinator will apply the back-pressure strategy specified below to each mutation

# sent to replicas, with the aim of reducing pressure on overloaded replicas.

back\_pressure\_enabled: false

# The back-pressure strategy applied.

# The default implementation, RateBasedBackPressure, takes three arguments:

# high ratio, factor, and flow type, and uses the ratio between incoming mutation responses and outgoing mutation requests.

# If below high ratio, outgoing mutations are rate limited according to the incoming rate decreased by the given factor;

# if above high ratio, the rate limiting is increased by the given factor;

# such factor is usually best configured between 1 and 10, use larger values for a faster recovery

# at the expense of potentially more dropped mutations;

# the rate limiting is applied according to the flow type: if FAST, it's rate limited at the speed of the fastest replica,

# if SLOW at the speed of the slowest one.

# New strategies can be added. Implementors need to implement org.apache.cassandra.net.BackpressureStrategy and

# provide a public constructor accepting a Map<String, Object>.

back\_pressure\_strategy:

- class\_name: org.apache.cassandra.net.RateBasedBackPressure

parameters:

- high\_ratio: 0.90

factor: 5

flow: FAST

# Coalescing Strategies #

# Coalescing multiples messages turns out to significantly boost message processing throughput (think doubling or more).

# On bare metal, the floor for packet processing throughput is high enough that many applications won't notice, but in

# virtualized environments, the point at which an application can be bound by network packet processing can be

# surprisingly low compared to the throughput of task processing that is possible inside a VM. It's not that bare metal

# doesn't benefit from coalescing messages, it's that the number of packets a bare metal network interface can process

# is sufficient for many applications such that no load starvation is experienced even without coalescing.

# There are other benefits to coalescing network messages that are harder to isolate with a simple metric like messages

# per second. By coalescing multiple tasks together, a network thread can process multiple messages for the cost of one

# trip to read from a socket, and all the task submission work can be done at the same time reducing context switching

# and increasing cache friendliness of network message processing.

# See CASSANDRA-8692 for details.

# Strategy to use for coalescing messages in OutboundTcpConnection.

# Can be fixed, movingaverage, timehorizon, disabled (default).

# You can also specify a subclass of CoalescingStrategies.CoalescingStrategy by name.

# otc\_coalescing\_strategy: DISABLED

# How many microseconds to wait for coalescing. For fixed strategy this is the amount of time after the first

# message is received before it will be sent with any accompanying messages. For moving average this is the

# maximum amount of time that will be waited as well as the interval at which messages must arrive on average

# for coalescing to be enabled.

# otc\_coalescing\_window\_us: 200

# Do not try to coalesce messages if we already got that many messages. This should be more than 2 and less than 128.

# otc\_coalescing\_enough\_coalesced\_messages: 8

# How many milliseconds to wait between two expiration runs on the backlog (queue) of the OutboundTcpConnection.

# Expiration is done if messages are piling up in the backlog. Droppable messages are expired to free the memory

# taken by expired messages. The interval should be between 0 and 1000, and in most installations the default value

# will be appropriate. A smaller value could potentially expire messages slightly sooner at the expense of more CPU

# time and queue contention while iterating the backlog of messages.

# An interval of 0 disables any wait time, which is the behavior of former Cassandra versions.

#

# otc\_backlog\_expiration\_interval\_ms: 200

#########################

# EXPERIMENTAL FEATURES #

#########################

# Enables materialized view creation on this node.

# Materialized views are considered experimental and are not recommended for production use.

enable\_materialized\_views: true

# Enables SASI index creation on this node.

# SASI indexes are considered experimental and are not recommended for production use.

enable\_sasi\_indexes: true

**auto\_bootstrap: false**