Kafka uses key-value pairs in the property file format for configuration. These values can be supplied either from a file or programmatically.

3.1 Broker Configs

The essential configurations are the following:

- `broker.id`
- `log.dirs`
- `zookeeper.connect`

Topic-level configurations and defaults are discussed in more detail below.

Name	Description	Type	Default	
zookeeper.connect	Zookeeper host string	string		
advertised.host.name	DEPRECATED: only used when \`advertised.listeners\` or \`listeners\` are not set. Use \`advertised.listeners\` instead. Hostname to publish to ZooKeeper for clients to use. In IaaS environments, this may need to be different from the interface to which the broker binds. If this is not set, it will use the value for \`host.name\` if configured. Otherwise it will use the value returned from java.net.InetAddress.getCanonicalHostName().	string	null	
advertised.listeners	Listeners to publish to ZooKeeper for clients to use, if different than the listeners above. In IaaS environments, this may need to be different from the interface to which the broker binds. If this is not set, the value for \`listeners\` will be used.	string	null	
advertised.port	DEPRECATED: only used when \`advertised.listeners\` or \`listeners\` are not set. Use \`advertised.listeners\` instead. The port to publish to ZooKeeper for clients to use. In IaaS environments, this may need to be different from the port to which the broker binds. If this is not set, it will publish the same port that the broker binds to.	int	null	
auto.create.topics.enable	Enable auto creation of topic on the server	boolean	true	
auto.leader.rebalance.enable	Enables auto leader balancing. A background thread checks and triggers leader balance if required at regular intervals	boolean	true	
background.threads	The number of threads to use for various background processing tasks	int	10	[1
broker.id	The broker id for this server. If unset, a unique broker id will be generated.To avoid conflicts between zookeeper generated broker id's and user configured broker id's, generated broker idsstart from reserved.broker.max.id + 1.	int	-1	
compression.type	Specify the final compression type for a given topic. This configuration accepts the standard compression codecs ('gzip', 'snappy', 'lz4'). It additionally accepts 'uncompressed' which is equivalent to no compression; and 'producer' which means retain the original compression codec set by the producer.	string	producer	
delete.topic.enable	Enables delete topic. Delete topic through the admin tool will have no effect if this config is turned off	boolean	false	
host.name	DEPRECATED: only used when \`listeners\` is not set. Use \`listeners\` instead. hostname of broker. If this is set, it will only bind to this address. If this is not set, it will bind to all interfaces	string	""	
leader.imbalance.check.interval.seconds	The frequency with which the partition rebalance check is triggered by the controller	long	300	
leader.imbalance.per.broker.percentage	The ratio of leader imbalance allowed per broker. The controller would trigger a leader balance if it goes above this value per broker. The value is specified in percentage.	int	10	
listeners	Listener List - Comma-separated list of URIs we will listen on and their protocols. Specify hostname as 0.0.0.0 to bind to all interfaces. Leave hostname empty to bind to default interface. Examples of legal listener lists: PLAINTEXT://myhost:9092,TRACE://:9091 PLAINTEXT://0.0.0.0:9092, TRACE://localhost:9093	string	null	

log.dir	The directory in which the log data is kept (supplemental for log.dirs property)	string	/tmp/kafka-logs	
log.dirs	The directories in which the log data is kept. If not set, the value in log.dir is used	string	null	
log.flush.interval.messages	The number of messages accumulated on a log partition before messages are flushed to disk	long	9223372036854775807	[1
log.flush.interval.ms	The maximum time in ms that a message in any topic is kept in memory before flushed to disk. If not set, the value in log.flush.scheduler.interval.ms is used	long	null	
log.flush.offset.checkpoint.interval.ms	The frequency with which we update the persistent record of the last flush which acts as the log recovery point	int	60000	[0
log.flush.scheduler.interval.ms	The frequency in ms that the log flusher checks whether any log needs to be flushed to disk	long	9223372036854775807	
log.retention.bytes	The maximum size of the log before deleting it	long	-1	
log.retention.hours	The number of hours to keep a log file before deleting it (in hours), tertiary to log.retention.ms property	int	168	
log.retention.minutes	The number of minutes to keep a log file before deleting it (in minutes), secondary to log.retention.ms property. If not set, the value in log.retention.hours is used	int	null	
log.retention.ms	The number of milliseconds to keep a log file before deleting it (in milliseconds), If not set, the value in log.retention.minutes is used	long	null	
log.roll.hours	The maximum time before a new log segment is rolled out (in hours), secondary to log.roll.ms property	int	168	[1
log.roll.jitter.hours	The maximum jitter to subtract from logRollTimeMillis (in hours), secondary to log.roll.jitter.ms property	int	0	[0
log.roll.jitter.ms	The maximum jitter to subtract from logRollTimeMillis (in milliseconds). If not set, the value in log.roll.jitter.hours is used	long	null	
log.roll.ms	The maximum time before a new log segment is rolled out (in milliseconds). If not set, the value in log.roll.hours is used	long	null	
log.segment.bytes	The maximum size of a single log file	int	1073741824	[1
log.segment.delete.delay.ms	The amount of time to wait before deleting a file from the filesystem	long	60000	[0
message.max.bytes	The maximum size of message that the server can receive	int	1000012	[0
min.insync.replicas	When a producer sets acks to "all" (or "-1"), min.insync.replicas specifies the minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum cannot be met, then the producer will raise an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, min.insync.replicas and acks allow you to enforce greater durability guarantees. A typical scenario would be to create a topic with a replication factor of 3, set min.insync.replicas to 2, and produce with acks of "all". This will ensure that the producer raises an exception if a majority of replicas do not receive a write.	int	1	[1
num.io.threads	The number of io threads that the server uses for carrying out network requests	int	8	[1
num.network.threads	the number of network threads that the server uses for handling network requests	int	3	[1
num.recovery.threads.per.data.dir	The number of threads per data directory to be used for log recovery at startup and flushing at shutdown	int	1	[1
num.replica.fetchers	Number of fetcher threads used to replicate messages from a source broker. Increasing this value can increase the degree of I/O parallelism in the follower broker.	int	1	
offset.metadata.max.bytes	The maximum size for a metadata entry associated with an offset commit	int	4096	
offsets.commit.required.acks	The required acks before the commit can be accepted. In general, the default (-1) should not be overridden	short	-1	
	Offset commit will be delayed until all replicas			

offsets.commit.timeout.ms	for the offsets topic receive the commit or this timeout is reached. This is similar to the producer request timeout.	int	5000	[1
offsets.load.buffer.size	Batch size for reading from the offsets segments when loading offsets into the cache.	int	5242880	[1
offsets.retention.check.interval.ms	Frequency at which to check for stale offsets	long	600000	[1
offsets.retention.minutes	Log retention window in minutes for offsets topic	int	1440	[1
offsets.topic.compression.codec	Compression codec for the offsets topic - compression may be used to achieve "atomic" commits	int	0	
offsets.topic.num.partitions	The number of partitions for the offset commit topic (should not change after deployment)	int	50	[1
offsets.topic.replication.factor	The replication factor for the offsets topic (set higher to ensure availability). To ensure that the effective replication factor of the offsets topic is the configured value, the number of alive brokers has to be at least the replication factor at the time of the first request for the offsets topic. If not, either the offsets topic creation will fail or it will get a replication factor of min(alive brokers, configured replication factor)	short	3	[1
offsets.topic.segment.bytes	The offsets topic segment bytes should be kept relatively small in order to facilitate faster log compaction and cache loads	int	104857600	[1
port	DEPRECATED: only used when `listeners` is not set. Use `listeners` instead. the port to listen and accept connections on	int	9092	
queued.max.requests	The number of queued requests allowed before blocking the network threads	int	500	[1
quota.consumer.default	Any consumer distinguished by clientId/consumer group will get throttled if it fetches more bytes than this value per-second	long	9223372036854775807	[1
quota.producer.default	Any producer distinguished by clientId will get throttled if it produces more bytes than this value per-second	long	9223372036854775807	[1
replica.fetch.max.bytes	The number of bytes of messages to attempt to fetch	int	1048576	
replica.fetch.min.bytes	Minimum bytes expected for each fetch response. If not enough bytes, wait up to replicaMaxWaitTimeMs	int	1	
replica.fetch.wait.max.ms	max wait time for each fetcher request issued by follower replicas. This value should always be less than the replica.lag.time.max.ms at all times to prevent frequent shrinking of ISR for low throughput topics	int	500	
replica.high.watermark.checkpoint.interval.ms	The frequency with which the high watermark is saved out to disk	long	5000	
replica.lag.time.max.ms	If a follower hasn't sent any fetch requests or hasn't consumed up to the leaders log end offset for at least this time, the leader will remove the follower from isr	long	10000	
replica.socket.receive.buffer.bytes	The socket receive buffer for network requests	int	65536	
replica.socket.timeout.ms	The socket timeout for network requests. Its value should be at least replica.fetch.wait.max.ms	int	30000	
request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.	int	30000	
socket.receive.buffer.bytes	The SO_RCVBUF buffer of the socket sever sockets. If the value is -1, the OS default will be used.	int	102400	
socket.request.max.bytes	The maximum number of bytes in a socket request	int	104857600	[1
socket.send.buffer.bytes	The SO_SNDBUF buffer of the socket sever sockets. If the value is -1, the OS default will be used.	int	102400	
unclean.leader.election.enable	Indicates whether to enable replicas not in the ISR set to be elected as leader as a last resort, even though doing so may result in data loss	boolean	true	
zookeeper.connection.timeout.ms	The max time that the client waits to establish a connection to zookeeper. If not set, the value	int	null	

	in zookeeper.session.timeout.ms is used			
zookeeper.session.timeout.ms	Zookeeper session timeout	int	6000	
zookeeper.set.acl	Set client to use secure ACLs	boolean	false	
broker.id.generation.enable	Enable automatic broker id generation on the server. When enabled the value configured for reserved.broker.max.id should be reviewed.	boolean	true	
broker.rack	Rack of the broker. This will be used in rack aware replication assignment for fault tolerance. Examples: \`RACK1\`, \`us-east-1d\`	string	null	
connections.max.idle.ms	Idle connections timeout: the server socket processor threads close the connections that idle more than this	long	600000	
controlled.shutdown.enable	Enable controlled shutdown of the server	boolean	true	
controlled.shutdown.max.retries	Controlled shutdown can fail for multiple reasons. This determines the number of retries when such failure happens	int	3	
controlled.shutdown.retry.backoff.ms	Before each retry, the system needs time to recover from the state that caused the previous failure (Controller fail over, replica lag etc). This config determines the amount of time to wait before retrying.	long	5000	
controller.socket.timeout.ms	The socket timeout for controller-to-broker channels	int	30000	
default.replication.factor	default replication factors for automatically created topics	int	1	
fetch.purgatory.purge.interval.requests	The purge interval (in number of requests) of the fetch request purgatory	int	1000	
group.max.session.timeout.ms	The maximum allowed session timeout for registered consumers. Longer timeouts give consumers more time to process messages in between heartbeats at the cost of a longer time to detect failures.	int	300000	
group.min.session.timeout.ms	The minimum allowed session timeout for registered consumers. Shorter timeouts leader to quicker failure detection at the cost of more frequent consumer heartbeating, which can overwhelm broker resources.	int	6000	
inter.broker.protocol.version	Specify which version of the inter-broker protocol will be used. This is typically bumped after all brokers were upgraded to a new version. Example of some valid values are: 0.8.0, 0.8.1, 0.8.1.1, 0.8.2, 0.8.2.0, 0.8.2.1, 0.9.0.0, 0.9.0.1 Check ApiVersion for the full list.	string	0.10.0-IV1	
log.cleaner.backoff.ms	The amount of time to sleep when there are no logs to clean	long	15000	[0
log.cleaner.dedupe.buffer.size	The total memory used for log deduplication across all cleaner threads	long	134217728	
log.cleaner.delete.retention.ms	How long are delete records retained?	long	86400000	
log.cleaner.enable	Enable the log cleaner process to run on the server? Should be enabled if using any topics with a cleanup.policy=compact including the internal offsets topic. If disabled those topics will not be compacted and continually grow in size.	boolean	true	
log.cleaner.io.buffer.load.factor	Log cleaner dedupe buffer load factor. The percentage full the dedupe buffer can become. A higher value will allow more log to be cleaned at once but will lead to more hash collisions	double	0.9	
log.cleaner.io.buffer.size	The total memory used for log cleaner I/O buffers across all cleaner threads	int	524288	[0
log.cleaner.io.max.bytes.per.second	The log cleaner will be throttled so that the sum of its read and write i/o will be less than this value on average	double	1.7976931348623157E308	
log.cleaner.min.cleanable.ratio	The minimum ratio of dirty log to total log for a log to eligible for cleaning	double	0.5	
log.cleaner.threads	The number of background threads to use for log cleaning	int	1	[0
log.cleanup.policy	The default cleanup policy for segments beyond the retention window, must be either "delete" or "compact"	string	delete	[c
log.index.interval.bytes	The interval with which we add an entry to the offset index	int	4096	[0
log.index.size.max.bytes	The maximum size in bytes of the offset index	int	10485760	[4

log.message.format.version	Specify the message format version the broker will use to append messages to the logs. The value should be a valid ApiVersion. Some examples are: 0.8.2, 0.9.0.0, 0.10.0, check ApiVersion for more details. By setting a particular message format version, the user is certifying that all the existing messages on disk are smaller or equal than the specified version. Setting this value incorrectly will cause consumers with older versions to break as they will receive messages with a format that they don't understand.	string	0.10.0-IV1	
log.message.timestamp.difference.max.ms	The maximum difference allowed between the timestamp when a broker receives a message and the timestamp specified in the message. If log.message.timestamp.type=CreateTime, a message will be rejected if the difference in timestamp exceeds this threshold. This configuration is ignored if log.message.timestamp.type=LogAppendTime.	long	9223372036854775807	[0
log.message.timestamp.type	Define whether the timestamp in the message is message create time or log append time. The value should be either `CreateTime` or `LogAppendTime`	string	CreateTime	[C L
log.preallocate	Should pre allocate file when create new segment? If you are using Kafka on Windows, you probably need to set it to true.	boolean	false	
log.retention.check.interval.ms	The frequency in milliseconds that the log cleaner checks whether any log is eligible for deletion	long	300000	[1
max.connections.per.ip	The maximum number of connections we allow from each ip address	int	2147483647	[1
max.connections.per.ip.overrides	Per-ip or hostname overrides to the default maximum number of connections	string	""	
num.partitions	The default number of log partitions per topic	int	1	[1
principal.builder.class	The fully qualified name of a class that implements the PrincipalBuilder interface, which is currently used to build the Principal for connections with the SSL SecurityProtocol.	class	class org.apache.kafka.common.security.auth.DefaultPrincipalBuilder	
producer.purgatory.purge.interval.requests	The purge interval (in number of requests) of the producer request purgatory	int	1000	
replica.fetch.backoff.ms	The amount of time to sleep when fetch partition error occurs.	int	1000	[0
reserved.broker.max.id	Max number that can be used for a broker.id	int	1000	[0
sasl.enabled.mechanisms	The list of SASL mechanisms enabled in the Kafka server. The list may contain any mechanism for which a security provider is available. Only GSSAPI is enabled by default.	list	[GSSAPI]	
sasl.kerberos.kinit.cmd	Kerberos kinit command path.	string	/usr/bin/kinit	
sasl.kerberos.min.time.before.relogin	Login thread sleep time between refresh attempts.	long	60000	
sasl.kerberos.principal.to.local.rules	A list of rules for mapping from principal names to short names (typically operating system usernames). The rules are evaluated in order and the first rule that matches a principal name is used to map it to a short name. Any later rules in the list are ignored. By default, principal names of the form {username}/{hostname}@{REALM} are mapped to {username}. For more details on the format please see security authorization and acls.	list	[DEFAULT]	
sasl.kerberos.service.name	The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.	string	null	
sasl.kerberos.ticket.renew.jitter	Percentage of random jitter added to the renewal time.	double	0.05	
sasl.kerberos.ticket.renew.window.factor	Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.	double	0.8	
sasl.mechanism.inter.broker.protocol	SASL mechanism used for inter-broker communication. Default is GSSAPI.	string	GSSAPI	
security.inter.broker.protocol	Security protocol used to communicate between brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.	string	PLAINTEXT	

ssl.cipher.suites	A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol.By default all the available cipher suites are supported.	list	null	
ssl.client.auth	Configures kafka broker to request client authentication. The following settings are common: • `ssl.client.auth=required` If set to required client authentication is required. • `ssl.client.auth=requested` This means client authentication is optional. unlike requested , if this option is set client can choose not to provide authentication information about itself • `ssl.client.auth=none` This means client authentication is not needed.	string	none	[r re
ssl.enabled.protocols	The list of protocols enabled for SSL connections.	list	[TLSv1.2, TLSv1.1, TLSv1]	
ssl.key.password	The password of the private key in the key store file. This is optional for client.	password	null	
ssl.keymanager.algorithm	The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.	string	SunX509	
ssl.keystore.location	The location of the key store file. This is optional for client and can be used for two-way authentication for client.	string	null	
ssl.keystore.password	The store password for the key store file.This is optional for client and only needed if ssl.keystore.location is configured.	password	null	
ssl.keystore.type	The file format of the key store file. This is optional for client.	string	JKS	
ssl.protocol	The SSL protocol used to generate the SSLContext. Default setting is TLS, which is fine for most cases. Allowed values in recent JVMs are TLS, TLSv1.1 and TLSv1.2. SSL, SSLv2 and SSLv3 may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities.	string	TLS	
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	string	null	
ssl.trustmanager.algorithm	The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.	string	PKIX	
ssl.truststore.location	The location of the trust store file.	string	null	
ssl.truststore.password	The password for the trust store file.	password	null	
ssl.truststore.type	The file format of the trust store file.	string	JKS	
authorizer.class.name	The authorizer class that should be used for authorization	string	""	
metric.reporters	A list of classes to use as metrics reporters. Implementing the `MetricReporter` interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	[]	
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1
metrics.sample.window.ms	The window of time a metrics sample is computed over.	long	30000	[1
quota.window.num	The number of samples to retain in memory	int	11	[1
quota.window.size.seconds	The time span of each sample	int	1	[1
ssl.endpoint.identification.algorithm	The endpoint identification algorithm to validate server hostname using server certificate.	string	null	
zookeeper.sync.time.ms	How far a ZK follower can be behind a ZK leader	int	2000	

More details about broker configuration can be found in the scala class `kafka.server.KafkaConfig`.

Topic-level configuration Configurations pertinent to topics have both a server default as well an optional per-topic override. If no per-topic configuration is given the server default is used. The override can be set at topic creation time by giving one or more `--config` options. This example creates a topic named *my-topic* with a custom max message size and flush rate:

```
 > bin/kafka-topics.sh --zookeeper localhost:2181 --create --topic my-topic --partitions 1
        --replication-factor 1 --config max.message.bytes=64000 --config flush.messages=1
```

Overrides can also be changed or set later using the alter topic command. This example updates the max message size for *my-topic*:

```
 > bin/kafka-topics.sh --zookeeper localhost:2181 --alter --topic my-topic
    --config max.message.bytes=128000
```

To remove an override you can do

```
 > bin/kafka-topics.sh --zookeeper localhost:2181 --alter --topic my-topic
    --delete-config max.message.bytes
```

The following are the topic-level configurations. The server's default configuration for this property is given under the Server Default Property heading. A given server default config value only applies to a topic if it does not have an explicit topic config override.

Name	Description	Type	Default	Valid Values	Server Default Property
cleanup.policy	A string that is either "delete" or "compact". This string designates the retention policy to use on old log segments. The default policy ("delete") will discard old segments when their retention time or size limit has been reached. The "compact" setting will enable log compaction on the topic.	string	delete	[compact, delete]	log.cleanup.policy
compression.type	Specify the final compression type for a given topic. This configuration accepts the standard compression codecs ('gzip', 'snappy', lz4). It additionally accepts 'uncompressed' which is equivalent to no compression; and 'producer' which means retain the original compression codec set by the producer.	string	producer	[uncompressed, snappy, lz4, gzip, producer]	compression.type
delete.retention.ms	The amount of time to retain delete tombstone markers for log compacted topics. This setting also gives a bound on the time in which a consumer must complete a read if they begin from offset 0 to ensure that they get a valid snapshot of the final stage (otherwise delete tombstones may be collected before they complete their scan).	long	86400000	[0,...]	log.cleaner.delete.retention.ms
file.delete.delay.ms	The time to wait before deleting a file from the filesystem	long	60000	[0,...]	log.segment.delete.delay.ms
flush.messages	This setting allows specifying an interval at which we will force an fsync of data written to the log. For example if this was set to 1 we would fsync after every message; if it were 5 we would fsync after every five messages. In general we recommend you not set this and use replication for durability and allow the operating system's background flush capabilities as it is more efficient. This setting can be overridden on a per-topic basis (see the per-topic configuration section).	long	9223372036854775807	[0,...]	log.flush.interval.messages
flush.ms	This setting allows specifying a time interval at which we will force an fsync of data written to the log. For example if this was set to 1000 we would fsync after 1000 ms had passed. In general we recommend you not set this and use replication for durability and allow the operating system's background flush capabilities as it is more efficient.	long	9223372036854775807	[0,...]	log.flush.interval.ms
index.interval.bytes	This setting controls how frequently Kafka adds an index entry to it's offset index. The default setting ensures that we index a message roughly every 4096 bytes. More indexing allows reads to jump closer to the exact position in the log but makes the index larger. You probably don't need to change this.	int	4096	[0,...]	log.index.interval.bytes
max.message.bytes	This is largest message size Kafka will allow to be appended. Note that if you increase this size you must also increase your consumer's fetch size so they can fetch messages this large.	int	1000012	[0,...]	message.max.bytes
	Specify the message format version the				

message.format.version	broker will use to append messages to the logs. The value should be a valid ApiVersion. Some examples are: 0.8.2, 0.9.0.0, 0.10.0, check ApiVersion for more details. By setting a particular message format version, the user is certifying that all the existing messages on disk are smaller or equal than the specified version. Setting this value incorrectly will cause consumers with older versions to break as they will receive messages with a format that they don't understand.	string	0.10.0-IV1		log.message.format.version
message.timestamp.difference.max.ms	The maximum difference allowed between the timestamp when a broker receives a message and the timestamp specified in the message. If message.timestamp.type=CreateTime, a message will be rejected if the difference in timestamp exceeds this threshold. This configuration is ignored if message.timestamp.type=LogAppendTime.	long	9223372036854775807	[0,...]	log.message.timestamp.difference.max
message.timestamp.type	Define whether the timestamp in the message is message create time or log append time. The value should be either `CreateTime` or `LogAppendTime`	string	CreateTime		log.message.timestamp.type
min.cleanable.dirty.ratio	This configuration controls how frequently the log compactor will attempt to clean the log (assuming log compaction is enabled). By default we will avoid cleaning a log where more than 50% of the log has been compacted. This ratio bounds the maximum space wasted in the log by duplicates (at 50% at most 50% of the log could be duplicates). A higher ratio will mean fewer, more efficient cleanings but will mean more wasted space in the log.	double	0.5	[0,...,1]	log.cleaner.min.cleanable.ratio
min.insync.replicas	When a producer sets acks to "all" (or "-1"), min.insync.replicas specifies the minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum cannot be met, then the producer will raise an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, min.insync.replicas and acks allow you to enforce greater durability guarantees. A typical scenario would be to create a topic with a replication factor of 3, set min.insync.replicas to 2, and produce with acks of "all". This will ensure that the producer raises an exception if a majority of replicas do not receive a write.	int	1	[1,...]	min.insync.replicas
preallocate	Should pre allocate file when create new segment?	boolean	false		log.preallocate
retention.bytes	This configuration controls the maximum size a log can grow to before we will discard old log segments to free up space if we are using the "delete" retention policy. By default there is no size limit only a time limit.	long	-1		log.retention.bytes
retention.ms	This configuration controls the maximum time we will retain a log before we will discard old log segments to free up space if we are using the "delete" retention policy. This represents an SLA on how soon consumers must read their data.	long	604800000		log.retention.ms
segment.bytes	This configuration controls the segment file size for the log. Retention and cleaning is always done a file at a time so a larger segment size means fewer files but less granular control over retention.	int	1073741824	[14,...]	log.segment.bytes
segment.index.bytes	This configuration controls the size of the index that maps offsets to file positions. We preallocate this index file and shrink it only after log rolls. You generally should not need to change this setting.	int	10485760	[0,...]	log.index.size.max.bytes
segment.jitter.ms	The maximum random jitter subtracted from the scheduled segment roll time to	long	0	[0,...]	log.roll.jitter.ms

	avoid thundering herds of segment rolling				
segment.ms	This configuration controls the period of time after which Kafka will force the log to roll even if the segment file isn't full to ensure that retention can delete or compact old data.	long	604800000	[0,...]	log.roll.ms
unclean.leader.election.enable	Indicates whether to enable replicas not in the ISR set to be elected as leader as a last resort, even though doing so may result in data loss	boolean	true		unclean.leader.election.enable

3.2 Producer Configs

Below is the configuration of the Java producer:

Name	Description	Type	Default	Valid Values	Im
bootstrap.servers	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form `host1:port1,host2:port2,...`. Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).	list			hig
key.serializer	Serializer class for key that implements the `Serializer` interface.	class			hig
value.serializer	Serializer class for value that implements the `Serializer` interface.	class			hig
acks	The number of acknowledgments the producer requires the leader to have received before considering a request complete. This controls the durability of records that are sent. The following settings are common: • `acks=0` If set to zero then the producer will not wait for any acknowledgment from the server at all. The record will be immediately added to the socket buffer and considered sent. No guarantee can be made that the server has received the record in this case, and the `retries` configuration will not take effect (as the client won't generally know of any failures). The offset given back for each record will always be set to -1. • `acks=1` This will mean the leader will write the record to its local log but will respond without awaiting full acknowledgement from all followers. In this case should the leader fail immediately after acknowledging the record but before the followers have replicated it then the record will be lost. • `acks=all` This means the leader will wait for the full set of in-sync replicas to acknowledge the record.	string	1	[all, -1, 0, 1]	hig

	This guarantees that the record will not be lost as long as at least one in-sync replica remains alive. This is the strongest available guarantee.				
buffer.memory	The total bytes of memory the producer can use to buffer records waiting to be sent to the server. If records are sent faster than they can be delivered to the server the producer will block for `max.block.ms` after which it will throw an exception. This setting should correspond roughly to the total memory the producer will use, but is not a hard bound since not all memory the producer uses is used for buffering. Some additional memory will be used for compression (if compression is enabled) as well as for maintaining in-flight requests.	long	33554432	[0,...]	hig
compression.type	The compression type for all data generated by the producer. The default is none (i.e. no compression). Valid values are `none`, `gzip`, `snappy`, or `lz4`. Compression is of full batches of data, so the efficacy of batching will also impact the compression ratio (more batching means better compression).	string	none		hig
retries	Setting a value greater than zero will cause the client to resend any record whose send fails with a potentially transient error. Note that this retry is no different than if the client resent the record upon receiving the error. Allowing retries will potentially change the ordering of records because if two records are sent to a single partition, and the first fails and is retried but the second succeeds, then the second record may appear first.	int	0	[0,...,2147483647]	hig
ssl.key.password	The password of the private key in the key store file. This is optional for client.	password	null		hig
ssl.keystore.location	The location of the key store file. This is optional for client and can be used for two-way authentication for client.	string	null		hig
ssl.keystore.password	The store password for the key store file.This is optional for client and only needed if ssl.keystore.location is configured.	password	null		hig
ssl.truststore.location	The location of the trust store file.	string	null		hig
ssl.truststore.password	The password for the trust store file.	password	null		hig
batch.size	The producer will attempt to batch records together into fewer requests whenever multiple records are being sent to the same partition. This helps performance on both the client and the server. This configuration controls the default batch size in bytes. No attempt will be made to batch records larger than this size. Requests sent to brokers will contain multiple batches, one for each partition with data available to be sent.	int	16384	[0,...]	me

	A small batch size will make batching less common and may reduce throughput (a batch size of zero will disable batching entirely). A very large batch size may use memory a bit more wastefully as we will always allocate a buffer of the specified batch size in anticipation of additional records.				
client.id	An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.	string	""		me
connections.max.idle.ms	Close idle connections after the number of milliseconds specified by this config.	long	540000		me
linger.ms	The producer groups together any records that arrive in between request transmissions into a single batched request. Normally this occurs only under load when records arrive faster than they can be sent out. However in some circumstances the client may want to reduce the number of requests even under moderate load. This setting accomplishes this by adding a small amount of artificial delay—that is, rather than immediately sending out a record the producer will wait for up to the given delay to allow other records to be sent so that the sends can be batched together. This can be thought of as analogous to Nagle's algorithm in TCP. This setting gives the upper bound on the delay for batching: once we get `batch.size` worth of records for a partition it will be sent immediately regardless of this setting, however if we have fewer than this many bytes accumulated for this partition we will 'linger' for the specified time waiting for more records to show up. This setting defaults to 0 (i.e. no delay). Setting `linger.ms=5`, for example, would have the effect of reducing the number of requests sent but would add up to 5ms of latency to records sent in the absense of load.	long	0	[0,...]	me
max.block.ms	The configuration controls how long `KafkaProducer.send()` and `KafkaProducer.partitionsFor()` will block.These methods can be blocked either because the buffer is full or metadata unavailable.Blocking in the user-supplied serializers or partitioner will not be counted against this timeout.	long	60000	[0,...]	me
max.request.size	The maximum size of a request in bytes. This is also effectively a cap on the maximum record size. Note that the server has its own cap on record size which may be different from this. This setting will limit the number of record batches the producer will send in a single request to avoid sending huge requests.	int	1048576	[0,...]	me
partitioner.class	Partitioner class that implements the `Partitioner` interface.	class	class org.apache.kafka.clients.producer.internals.DefaultPartitioner		me
	The size of the TCP receive buffer (SO_RCVBUF) to use when				

receive.buffer.bytes	reading data. If the value is -1, the OS default will be used.	int	32768	[0,...]	me
request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.	int	30000	[0,...]	me
sasl.kerberos.service.name	The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.	string	null		me
sasl.mechanism	SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.	string	GSSAPI		me
security.protocol	Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.	string	PLAINTEXT		me
send.buffer.bytes	The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.	int	131072	[0,...]	me
ssl.enabled.protocols	The list of protocols enabled for SSL connections.	list	[TLSv1.2, TLSv1.1, TLSv1]		me
ssl.keystore.type	The file format of the key store file. This is optional for client.	string	JKS		me
ssl.protocol	The SSL protocol used to generate the SSLContext. Default setting is TLS, which is fine for most cases. Allowed values in recent JVMs are TLS, TLSv1.1 and TLSv1.2. SSL, SSLv2 and SSLv3 may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities.	string	TLS		me
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	string	null		me
ssl.truststore.type	The file format of the trust store file.	string	JKS		me
timeout.ms	The configuration controls the maximum amount of time the server will wait for acknowledgments from followers to meet the acknowledgment requirements the producer has specified with the `acks` configuration. If the requested number of acknowledgments are not met when the timeout elapses an error will be returned. This timeout is measured on the server side and does not include the network latency of the request.	int	30000	[0,...]	me
block.on.buffer.full	When our memory buffer is exhausted we must either stop accepting new records (block) or throw errors. By default this setting is false and the producer will no longer throw a BufferExhaustException but instead will use the `max.block.ms` value to block, after which it will throw a TimeoutException. Setting this property to true will set the `max.block.ms` to Long.MAX_VALUE. *Also if this property is set to true, parameter `metadata.fetch.timeout.ms` is not longer honored.* This parameter is deprecated and	boolean	false		low

	will be removed in a future release. Parameter `max.block.ms` should be used instead.				
interceptor.classes	A list of classes to use as interceptors. Implementing the `ProducerInterceptor` interface allows you to intercept (and possibly mutate) the records received by the producer before they are published to the Kafka cluster. By default, there are no interceptors.	list	null		low
max.in.flight.requests.per.connection	The maximum number of unacknowledged requests the client will send on a single connection before blocking. Note that if this setting is set to be greater than 1 and there are failed sends, there is a risk of message re-ordering due to retries (i.e., if retries are enabled).	int	5	[1,...]	low
metadata.fetch.timeout.ms	The first time data is sent to a topic we must fetch metadata about that topic to know which servers host the topic's partitions. This fetch to succeed before throwing an exception back to the client.	long	60000	[0,...]	low
metadata.max.age.ms	The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.	long	300000	[0,...]	low
metric.reporters	A list of classes to use as metrics reporters. Implementing the `MetricReporter` interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	[]		low
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1,...]	low
metrics.sample.window.ms	The window of time a metrics sample is computed over.	long	30000	[0,...]	low
reconnect.backoff.ms	The amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all requests sent by the consumer to the broker.	long	50	[0,...]	low
retry.backoff.ms	The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.	long	100	[0,...]	low
sasl.kerberos.kinit.cmd	Kerberos kinit command path.	string	/usr/bin/kinit		low
sasl.kerberos.min.time.before.relogin	Login thread sleep time between refresh attempts.	long	60000		low
sasl.kerberos.ticket.renew.jitter	Percentage of random jitter added to the renewal time.	double	0.05		low
sasl.kerberos.ticket.renew.window.factor	Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.	double	0.8		low
ssl.cipher.suites	A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol.By default all the available cipher suites are supported.	list	null		low
ssl.endpoint.identification.algorithm	The endpoint identification algorithm to validate server hostname using server certificate.	string	null		low

ssl.keymanager.algorithm	The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.	string	SunX509	low
ssl.trustmanager.algorithm	The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.	string	PKIX	low

For those interested in the legacy Scala producer configs, information can be found here.

3.3 Consumer Configs

We introduce both the old 0.8 consumer configs and the new consumer configs respectively below.

3.3.1 Old Consumer Configs

The essential old consumer configurations are the following:

- `group.id`
- `zookeeper.connect`

Property	Default	Description
group.id		A string that uniquely identifies the group of consumer processes to which this consumer belongs. By setting the same group id multiple processes indicate that they are all part of the same consumer group.
zookeeper.connect		Specifies the ZooKeeper connection string in the form `hostname:port` where host and port are the host and port of a ZooKeeper server. To allow connecting through other ZooKeeper nodes when that ZooKeeper machine is down you can also specify multiple hosts in the form `hostname1:port1,hostname2:port2,hostname3:port3`. The server may also have a ZooKeeper chroot path as part of it's ZooKeeper connection string which puts its data under some path in the global ZooKeeper namespace. If so the consumer should use the same chroot path in its connection string. For example to give a chroot path of `/chroot/path` you would give the connection string as `hostname1:port1,hostname2:port2,hostname3:port3/chroot/path`.
consumer.id	null	Generated automatically if not set.
socket.timeout.ms	30 * 1000	The socket timeout for network requests. The actual timeout set will be max.fetch.wait + socket.timeout.ms.
socket.receive.buffer.bytes	64 * 1024	The socket receive buffer for network requests
fetch.message.max.bytes	1024 * 1024	The number of bytes of messages to attempt to fetch for each topic-partition in each fetch request. These bytes will be read into memory for each partition, so this helps control the memory used by the consumer. The fetch request size must be at least as large as the maximum message size the server allows or else it is possible for the producer to send messages larger than the consumer can fetch.
num.consumer.fetchers	1	The number fetcher threads used to fetch data.
auto.commit.enable	true	If true, periodically commit to ZooKeeper the offset of messages already fetched by the consumer. This committed offset will be used when the process fails as the position from which the new consumer will begin.
auto.commit.interval.ms	60 * 1000	The frequency in ms that the consumer offsets are committed to zookeeper.
queued.max.message.chunks	2	Max number of message chunks buffered for consumption. Each chunk can be up to fetch.message.max.bytes.
rebalance.max.retries	4	When a new consumer joins a consumer group the set of consumers attempt to "rebalance" the load to assign partitions to each consumer. If the set of consumers changes while this assignment is taking place the rebalance will fail and retry. This setting controls the maximum number of attempts before giving up.
fetch.min.bytes	1	The minimum amount of data the server should return for a fetch request. If insufficient data is available the request will wait for that much data to accumulate before answering the request.
fetch.wait.max.ms	100	The maximum amount of time the server will block before answering the fetch request if there isn't sufficient data to immediately satisfy fetch.min.bytes
rebalance.backoff.ms	2000	Backoff time between retries during rebalance. If not set explicitly, the value in zookeeper.sync.time.ms is used.
refresh.leader.backoff.ms	200	Backoff time to wait before trying to determine the leader of a partition that has just lost its leader.
auto.offset.reset	largest	What to do when there is no initial offset in ZooKeeper or if an offset is out of range: * smallest : automatically reset the offset to the smallest offset * largest : automatically reset the offset to the largest offset * anything else: throw exception to the consumer
consumer.timeout.ms	-1	Throw a timeout exception to the consumer if no message is available for consumption after the specified interval
exclude.internal.topics	true	Whether messages from internal topics (such as offsets) should be exposed to the consumer.
client.id	group id value	The client id is a user-specified string sent in each request to help trace calls. It should logically identify the application making the request.
zookeeper.session.timeout.msÂ	6000	ZooKeeper session timeout. If the consumer fails to heartbeat to ZooKeeper for this period of time it is considered dead and a rebalance will occur.
zookeeper.connection.timeout.ms	6000	The max time that the client waits while establishing a connection to zookeeper.
zookeeper.sync.time.msÂ	2000	How far a ZK follower can be behind a ZK leader
offsets.storage	zookeeper	Select where offsets should be stored (zookeeper or kafka).
offsets.channel.backoff.ms	1000	The backoff period when reconnecting the offsets channel or retrying failed offset fetch/commit requests.
offsets.channel.socket.timeout.ms	10000	Socket timeout when reading responses for offset fetch/commit requests. This timeout is also used for ConsumerMetadata requests that are used to query for the offset manager.

offsets.commit.max.retries	5	Retry the offset commit up to this many times on failure. This retry count only applies to offset commits during shut-down. It does not apply to commits originating from the auto-commit thread. It also does not apply to attempts to query for the offset coordinator before committing offsets. i.e., if a consumer metadata request fails for any reason, it will be retried and that retry does not count toward this limit.
dual.commit.enabled	true	If you are using "kafka" as offsets.storage, you can dual commit offsets to ZooKeeper (in addition to Kafka). This is required during migration from zookeeper-based offset storage to kafka-based offset storage. With respect to any given consumer group, it is safe to turn this off after all instances within that group have been migrated to the new version that commits offsets to the broker (instead of directly to ZooKeeper).
partition.assignment.strategy	range	Select between the "range" or "roundrobin" strategy for assigning partitions to consumer streams. The round-robin partition assignor lays out all the available partitions and all the available consumer threads. It then proceeds to do a round-robin assignment from partition to consumer thread. If the subscriptions of all consumer instances are identical, then the partitions will be uniformly distributed. (i.e., the partition ownership counts will be within a delta of exactly one across all consumer threads.) Round-robin assignment is permitted only if: (a) Every topic has the same number of streams within a consumer instance (b) The set of subscribed topics is identical for every consumer instance within the group. Range partitioning works on a per-topic basis. For each topic, we lay out the available partitions in numeric order and the consumer threads in lexicographic order. We then divide the number of partitions by the total number of consumer streams (threads) to determine the number of partitions to assign to each consumer. If it does not evenly divide, then the first few consumers will have one extra partition.

More details about consumer configuration can be found in the scala class `kafka.consumer.ConsumerConfig`.

3.3.2 New Consumer Configs

Since 0.9.0.0 we have been working on a replacement for our existing simple and high-level consumers. The code is considered beta quality. Below is the configuration for the new consumer:

Name	Description	Type	Default	Valid Values	Importance
bootstrap.servers	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form `host1:port1,host2:port2,...`. Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).	list			high
key.deserializer	Deserializer class for key that implements the `Deserializer` interface.	class			high
value.deserializer	Deserializer class for value that implements the `Deserializer` interface.	class			high
fetch.min.bytes	The minimum amount of data the server should return for a fetch request. If insufficient data is available the request will wait for that much data to accumulate before answering the request. The default setting of 1 byte means that fetch requests are answered as soon as a single byte of data is available or the fetch request times out waiting for data to arrive. Setting this to something greater than 1 will cause the server to wait for larger amounts of data to accumulate which can improve server throughput a bit at the cost of some additional latency.	int	1	[0,...]	high
group.id	A unique string that identifies the consumer group this consumer belongs to. This property is required if the consumer uses either the group management functionality by using `subscribe(topic)` or the Kafka-based offset management strategy.	string	""		high
heartbeat.interval.ms	The expected time between heartbeats to the consumer coordinator when using Kafka's group management facilities. Heartbeats are used to ensure that the consumer's session stays active and to facilitate rebalancing when new consumers join or leave the group. The value must be set lower than `session.timeout.ms`, but typically should be set no higher than 1/3 of that value. It can be adjusted even lower to	int	3000		high

	control the expected time for normal rebalances.				
max.partition.fetch.bytes	The maximum amount of data per-partition the server will return. The maximum total memory used for a request will be `#partitions * max.partition.fetch.bytes`. This size must be at least as large as the maximum message size the server allows or else it is possible for the producer to send messages larger than the consumer can fetch. If that happens, the consumer can get stuck trying to fetch a large message on a certain partition.	int	1048576	[0,...]	high
session.timeout.ms	The timeout used to detect failures when using Kafka's group management facilities. When a consumer's heartbeat is not received within the session timeout, the broker will mark the consumer as failed and rebalance the group. Since heartbeats are sent only when poll() is invoked, a higher session timeout allows more time for message processing in the consumer's poll loop at the cost of a longer time to detect hard failures. See also `max.poll.records` for another option to control the processing time in the poll loop. Note that the value must be in the allowable range as configured in the broker configuration by `group.min.session.timeout.ms` and `group.max.session.timeout.ms`.	int	30000		high
ssl.key.password	The password of the private key in the key store file. This is optional for client.	password	null		high
ssl.keystore.location	The location of the key store file. This is optional for client and can be used for two-way authentication for client.	string	null		high
ssl.keystore.password	The store password for the key store file.This is optional for client and only needed if ssl.keystore.location is configured.	password	null		high
ssl.truststore.location	The location of the trust store file.	string	null		high
ssl.truststore.password	The password for the trust store file.	password	null		high
auto.offset.reset	What to do when there is no initial offset in Kafka or if the current offset does not exist any more on the server (e.g. because that data has been deleted): • earliest: automatically reset the offset to the earliest offset • latest: automatically reset the offset to the latest offset • none: throw exception to the consumer if no previous offset is found for the consumer's group • anything else: throw exception to the consumer.	string	latest	[latest, earliest, none]	medium
connections.max.idle.ms	Close idle connections after the number of milliseconds specified by this config.	long	540000		medium
enable.auto.commit	If true the consumer's offset will be periodically committed in the background.	boolean	true		medium
exclude.internal.topics	Whether records from internal topics (such as offsets) should be exposed to the consumer. If set to `true` the only way to receive records from an internal topic is subscribing to it.	boolean	true		medium
max.poll.records	The maximum number of records returned in a single call to poll().	int	2147483647	[1,...]	medium
partition.assignment.strategy	The class name of the partition assignment strategy that the client will use to distribute partition ownership amongst consumer instances when group management is used	list	[org.apache.kafka.clients.consumer.RangeAssignor]		medium
receive.buffer.bytes	The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.	int	65536	[0,...]	medium
	The configuration controls the maximum amount of time the client will wait for the				

request.timeout.ms	response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.	int	40000	[0,...]	medium
sasl.kerberos.service.name	The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.	string	null		medium
sasl.mechanism	SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.	string	GSSAPI		medium
security.protocol	Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.	string	PLAINTEXT		medium
send.buffer.bytes	The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.	int	131072	[0,...]	medium
ssl.enabled.protocols	The list of protocols enabled for SSL connections.	list	[TLSv1.2, TLSv1.1, TLSv1]		medium
ssl.keystore.type	The file format of the key store file. This is optional for client.	string	JKS		medium
ssl.protocol	The SSL protocol used to generate the SSLContext. Default setting is TLS, which is fine for most cases. Allowed values in recent JVMs are TLS, TLSv1.1 and TLSv1.2. SSL, SSLv2 and SSLv3 may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities.	string	TLS		medium
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	string	null		medium
ssl.truststore.type	The file format of the trust store file.	string	JKS		medium
auto.commit.interval.ms	The frequency in milliseconds that the consumer offsets are auto-committed to Kafka if `enable.auto.commit` is set to `true`.	long	5000	[0,...]	low
check.crcs	Automatically check the CRC32 of the records consumed. This ensures no on-the-wire or on-disk corruption to the messages occurred. This check adds some overhead, so it may be disabled in cases seeking extreme performance.	boolean	true		low
client.id	An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.	string	""		low
fetch.max.wait.ms	The maximum amount of time the server will block before answering the fetch request if there isn't sufficient data to immediately satisfy the requirement given by fetch.min.bytes.	int	500	[0,...]	low
interceptor.classes	A list of classes to use as interceptors. Implementing the `ConsumerInterceptor` interface allows you to intercept (and possibly mutate) records received by the consumer. By default, there are no interceptors.	list	null		low
metadata.max.age.ms	The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.	long	300000	[0,...]	low
metric.reporters	A list of classes to use as metrics reporters. Implementing the `MetricReporter` interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	[]		low
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1,...]	low
metrics.sample.window.ms	The window of time a metrics sample is computed over.	long	30000	[0,...]	low
	The amount of time to wait before				

reconnect.backoff.ms	attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all requests sent by the consumer to the broker.	long	50	[0,...]	low
retry.backoff.ms	The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.	long	100	[0,...]	low
sasl.kerberos.kinit.cmd	Kerberos kinit command path.	string	/usr/bin/kinit		low
sasl.kerberos.min.time.before.relogin	Login thread sleep time between refresh attempts.	long	60000		low
sasl.kerberos.ticket.renew.jitter	Percentage of random jitter added to the renewal time.	double	0.05		low
sasl.kerberos.ticket.renew.window.factor	Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.	double	0.8		low
ssl.cipher.suites	A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol.By default all the available cipher suites are supported.	list	null		low
ssl.endpoint.identification.algorithm	The endpoint identification algorithm to validate server hostname using server certificate.	string	null		low
ssl.keymanager.algorithm	The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.	string	SunX509		low
ssl.trustmanager.algorithm	The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.	string	PKIX		low

3.4 Kafka Connect Configs

Below is the configuration of the Kafka Connect framework.

Name	Description	Type	Default	Valid Values	Importance
config.storage.topic	kafka topic to store configs	string			high
group.id	A unique string that identifies the Connect cluster group this worker belongs to.	string			high
internal.key.converter	Converter class for internal key Connect data that implements the `Converter` interface. Used for converting data like offsets and configs.	class			high
internal.value.converter	Converter class for offset value Connect data that implements the `Converter` interface. Used for converting data like offsets and configs.	class			high
key.converter	Converter class for key Connect data that implements the `Converter` interface.	class			high
offset.storage.topic	kafka topic to store connector offsets in	string			high
status.storage.topic	kafka topic to track connector and task status	string			high
value.converter	Converter class for value Connect data that implements the `Converter` interface.	class			high
bootstrap.servers	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form `host1:port1,host2:port2,...`. Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).	list	[localhost:9092]		high
heartbeat.interval.ms	The expected time between heartbeats to the group coordinator when using Kafka's group management facilities. Heartbeats are used to ensure that the worker's session stays active and to facilitate rebalancing when new members join or leave the group. The value must be set lower than `session.timeout.ms`, but typically should be set no higher than 1/3 of that value. It can be adjusted even lower to control the expected time for normal rebalances.	int	3000		high
session.timeout.ms	The timeout used to detect failures when using Kafka's group management facilities.	int	30000		high
ssl.key.password	The password of the private key in the key store file. This is optional for client.	password	null		high
ssl.keystore.location	The location of the key store file. This is optional for client and can be used for two-way authentication for client.	string	null		high
ssl.keystore.password	The store password for the key store file.This is optional for client and only needed if ssl.keystore.location is configured.	password	null		high
ssl.truststore.location	The location of the trust store file.	string	null		high
ssl.truststore.password	The password for the trust store file.	password	null		high

connections.max.idle.ms	Close idle connections after the number of milliseconds specified by this config.	long	540000		medium
receive.buffer.bytes	The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.	int	32768	[0,...]	medium
request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.	int	40000	[0,...]	medium
sasl.kerberos.service.name	The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.	string	null		medium
sasl.mechanism	SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.	string	GSSAPI		medium
security.protocol	Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.	string	PLAINTEXT		medium
send.buffer.bytes	The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.	int	131072	[0,...]	medium
ssl.enabled.protocols	The list of protocols enabled for SSL connections.	list	[TLSv1.2, TLSv1.1, TLSv1]		medium
ssl.keystore.type	The file format of the key store file. This is optional for client.	string	JKS		medium
ssl.protocol	The SSL protocol used to generate the SSLContext. Default setting is TLS, which is fine for most cases. Allowed values in recent JVMs are TLS, TLSv1.1 and TLSv1.2. SSL, SSLv2 and SSLv3 may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities.	string	TLS		medium
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	string	null		medium
ssl.truststore.type	The file format of the trust store file.	string	JKS		medium
worker.sync.timeout.ms	When the worker is out of sync with other workers and needs to resynchronize configurations, wait up to this amount of time before giving up, leaving the group, and waiting a backoff period before rejoining.	int	3000		medium
worker.unsync.backoff.ms	When the worker is out of sync with other workers and fails to catch up within worker.sync.timeout.ms, leave the Connect cluster for this long before rejoining.	int	300000		medium
access.control.allow.methods	Sets the methods supported for cross origin requests by setting the Access-Control-Allow-Methods header. The default value of the Access-Control-Allow-Methods header allows cross origin requests for GET, POST and HEAD.	string	""		low
access.control.allow.origin	Value to set the Access-Control-Allow-Origin header to for REST API requests.To enable cross origin access, set this to the domain of the application that should be permitted to access the API, or '*' to allow access from any domain. The default value only allows access from the domain of the REST API.	string	""		low
client.id	An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.	string	""		low
metadata.max.age.ms	The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.	long	300000	[0,...]	low
metric.reporters	A list of classes to use as metrics reporters. Implementing the `MetricReporter` interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	[]		low
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1,...]	low
metrics.sample.window.ms	The window of time a metrics sample is computed over.	long	30000	[0,...]	low
offset.flush.interval.ms	Interval at which to try committing offsets for tasks.	long	60000		low
offset.flush.timeout.ms	Maximum number of milliseconds to wait for records to flush and partition offset data to be committed to offset storage before cancelling the process and restoring the offset data to be committed in a future attempt.	long	5000		low
reconnect.backoff.ms	The amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all requests sent by the consumer to the broker.	long	50	[0,...]	low
rest.advertised.host.name	If this is set, this is the hostname that will be given out to other workers to connect to.	string	null		low
rest.advertised.port	If this is set, this is the port that will be given out to other workers to connect to.	int	null		low
rest.host.name	Hostname for the REST API. If this is set, it will only bind to this interface.	string	null		low
rest.port	Port for the REST API to listen on.	int	8083		low
retry.backoff.ms	The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.	long	100	[0,...]	low
sasl.kerberos.kinit.cmd	Kerberos kinit command path.	string	/usr/bin/kinit		low
sasl.kerberos.min.time.before.relogin	Login thread sleep time between refresh attempts.	long	60000		low
sasl.kerberos.ticket.renew.jitter	Percentage of random jitter added to the renewal time.	double	0.05		low
sasl.kerberos.ticket.renew.window.factor	Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.	double	0.8		low
ssl.cipher.suites	A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol.By default	list	null		low

	all the available cipher suites are supported.			
ssl.endpoint.identification.algorithm	The endpoint identification algorithm to validate server hostname using server certificate.	string	null	low
ssl.keymanager.algorithm	The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.	string	SunX509	low
ssl.trustmanager.algorithm	The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.	string	PKIX	low
task.shutdown.graceful.timeout.ms	Amount of time to wait for tasks to shutdown gracefully. This is the total amount of time, not per task. All task have shutdown triggered, then they are waited on sequentially.	long	5000	low

3.5 Kafka Streams Configs

Below is the configuration of the Kafka Streams client library.

Name	Description	Type	Default	Valid Values	Importance
application.id	An identifier for the stream processing application. Must be unique within the Kafka cluster. It is used as 1) the default client-id prefix, 2) the group-id for membership management, 3) the changelog topic prefix.	string			high
bootstrap.servers	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form `host1:port1,host2:port2,...`. Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).	list			high
client.id	An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.	string	""		high
zookeeper.connect	Zookeeper connect string for Kafka topics management.	string	""		high
key.serde	Serializer / deserializer class for key that implements the `Serde` interface.	class	class org.apache.kafka.common.serialization.Serdes$ByteArraySerde		medium
partition.grouper	Partition grouper class that implements the `PartitionGrouper` interface.	class	class org.apache.kafka.streams.processor.DefaultPartitionGrouper		medium
replication.factor	The replication factor for change log topics and repartition topics created by the stream processing application.	int	1		medium
state.dir	Directory location for state store.	string	/tmp/kafka-streams		medium
timestamp.extractor	Timestamp extractor class that implements the `TimestampExtractor` interface.	class	class org.apache.kafka.streams.processor.ConsumerRecordTimestampExtractor		medium
value.serde	Serializer / deserializer class for value that implements the `Serde` interface.	class	class org.apache.kafka.common.serialization.Serdes$ByteArraySerde		medium
buffered.records.per.partition	The maximum number of records to buffer per partition.	int	1000		low
commit.interval.ms	The frequency with which to save the position of the processor.	long	30000		low
metric.reporters	A list of classes to use as metrics reporters. Implementing the `MetricReporter` interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	[]		low
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1,...]	low
	The window of time a metrics sample				

metrics.sample.window.ms	is computed over.	long	30000	[0,...]	low
num.standby.replicas	The number of standby replicas for each task.	int	0		low
num.stream.threads	The number of threads to execute stream processing.	int	1		low
poll.ms	The amount of time in milliseconds to block waiting for input.	long	100		low
state.cleanup.delay.ms	The amount of time in milliseconds to wait before deleting state when a partition has migrated.	long	60000		low