Redis Connect

Version 0.10.1

Table of Contents

Introduction 1
Key Terms 2
Source distributions 3
Production Deployment 4
Cluster and Job Configuration 11
Job execution configuration 21

Introduction

Redis Connect is a distributed platform that enables real-time event streaming, transformation, and propagation of changed-data events from heterogeneous data platforms to Redis Stack, Redis Cloud, and Redis Enterprise.

Redis Connect effectively captures change data events from source databases and writes that data to Redis. This allows you to keep a Redis database in sync with a variety of source databases. You can then use Redis to serve this data to downstream applications at low latencies.

This document outlines key terms, installation instructions, production readiness guidelines, and definitions for the various configuration directives for Redis Connect clusters and jobs.

- Key Terms
- Source Distributions
- Production Deployment
- Configuration

Key Terms

To understand how Redis Connect works, it's important review some key terms:

Source

A database, such as MySQL, whose data will be replicated to Redis. Redis Connect replicates data from a source database to Redis.

Target

A database to write data to. With Redis Connect, the target is usually Redis.

Job

A stream of change-data events replicating from source to target. For example, you can replicate all changes from a given set of MySQL tables to Redis and maintain consistency between these tables and their Redis representations in real time.

Job Types

Redis Connect supports two types of jobs: **initial load jobs** and **stream jobs**.

- **Initial load jobs** create a point-in-time snapshot of the tables to be replicated and then transfer their data to Redis.
- **Stream jobs** (also known as CDC or "change data capture" jobs) replicate changes from the source tables to Redis as those changes occur.

Instance

A single JVM process running Redis Connect. Because Redis Connect is a distributed platform, it may run as one or more coordinated instances.

Cluster

One or more **instances** of Redis Connect running in a coordinated fashion.

Partition

A way of dividing jobs to scale them horizontally. Each job may be divided into one or more partitions. Partitions are divided automatically among Redis Connect instances. If a Redis Connect instances becomes unavailable, all jobs partitions will be migrated to another available instance.

Source distributions

Redis Connect releases are distributed on Github.

See the Redis Connect Release History to download the latest distributions of Redis Connect.

The source distribution contains three relevant folders:

lib

JARs for Redis Connect and its dependencies.

extlib

JARs for Redis Connect custom stages and database drivers not included the Redis Connect distribution (e.g., Oracle and DB2). These database driver must be provided by the Redis Connect user.

config

Working configuration files, sample payloads for configuring jobs, and Grafana dashboard configurations.

bin

Scripts for running Redis Connect on Linux VMs, in container environments, and on Windows.

Production Deployment

Redis Connect is deployed as one or more JVM instances coordinated as a cluster. Below are recommendations for running Redis Connect in production.

Environment

Redis Connect can be deployed on physical servers, virtual machines, or using Docker or any Kubernetes-based environment.

The minimum resource requirements per Redis Connect instance are as follows:

- 4 CPU cores
- 2 GB memory
- 20 GB of free disk space
- 1 Gbps networking

We recommend allocating one thread per job partition. You can deploy more than one Redis Connect instance on a single VM. The number of Redis Connect instances that can effectively be deployed on a single machine or VM will depend on that VM's physical memory and number of CPU cores. However, for high availability, you must deploy your Redis Connect instances across more than one physical server or VM. See High Availability below for more details.

Operating System and JVM

Redis Connect can run on any operating system hosting a Java runtime environment. However, for production deployments, we recommend Linux.

Redis Connect is supported on Java versions 11 and greater.

For information on deploying to Kubernetes environments, see the Redis Connect Kubernetes documentation.

Environment Variables

Redis Connect recognizes and depends upon several environment variables. You can see example of these in the startup scripts included in the Redis Connect distribution.

- `REDISCONNECT_MIN_JAVA_VERSION="11"`
- `REDISCONNECT_HOME="${REDIS_CONNECT_HOME_DIR}"`
- `REDISCONNECT_JOB_MANAGER_CONFIG_PATH="$REDISCONNECT_HOME/config/jobmanager.properties"`
- `REDISCONNECT_LOGBACK_CONFIG="$REDISCONNECT_HOME/config/logback.xml"`
- `REDISCONNECT_LOGBACK_CLI_CONFIG="$REDISCONNECT_HOME/config/logback-cli.xml"`
- `REDISCONNECT_JAVA_OPTIONS="-XX:+HeapDumpOnOutOfMemoryError -Xms1g -Xmx2g"`
- `REDISCONNECT_EXTLIB_DIR="$REDISCONNECT_HOME/extlib"`

- `REDISCONNECT_LIB_DIR="$REDISCONNECT_HOME/lib/:$REDISCONNECT_EXTLIB_DIR/"`

JVM Flags

We recommend the following JVM flags for each Redis Connect JVM instance:

- Enable heap dump on OOM: `-XX:+HeapDumpOnOutOfMemoryError`
- Min heap size of 1 GB: `-Xms1g`
- Max heap size of 2 GB: `-Xmx2g`

Redis Requirements

Redis Connect requires a working Redis Enterprise Software or Redis Cloud installation.

We recommend provisioning two Redis databases. These database can reside in the same Redis Enterprise cluster.

The first database will serve as your Redis Connect cluster's configuration store. This database should be configured as follows:

- Data persistence enabled (RDF + AOF every second)
- High availability enabled
- ACLs enabled (See Security below)

The second database will serve as the target for replication from the source database. Since this is an operational database receiving change-data events, the sizing for this database depends on the sizes of the tables being replicated and on the volume of change-data events. In all cases, we still recommend:

- High availability enabled
- ACLs enabled (See Security below)

Redis Connect Management

Once Redis Connect is installed and running, you manage Redis Connect using its REST API or command line interface (CLI).

REST API

Each Redis Connect instance can be configured to expose a REST API with a Swagger interface for ease of use.

The table below shows a few of the commonly-used REST endpoints. For the complete REST API documentation, see the Redis Connect Swagger API Docs.

Table 1. Common REST API Endpoints

Endpoint name	Description	Documentation
Create job	Saves job configuration for the provided `jobName`. Many of the optional job configuration attributes have default values which can be reviewed once the job configuration is saved or found in the documentation. Example Redis key: {connect}:job:config:jobName	Create job Swagger endpoint
Start job	Starts a job, including all job partitions. This includes both initial load and stream jobs. For a job start, all job partitions must be stopped or never before started. There is no guarantee on which cluster instance will claim a job partition, and there is no advantage to initiating this operation from a specific cluster member. Before a start is initiated, a job configuration must be created and a validated to confirm enough remaining capacity exists across the cluster for all job partitions (this does not apply to initial load).	Start job Swagger endpoint

Endpoint name	Description	Documentation
Stop job	Stops a job, including all job partitions. You cannot stop initial load jobs since they are removed automatically upon completion. For a job to be stopped, all job partitions must be active. Job claims, metrics, and checkpoints will all be preserved upon a job stop. This ensures that the job can later resume from where it was stopped.	Stop job Swagger endpoint
Job claim status	Returns job claims from across the cluster that match the requested jobStatus. Valid job statuses are `staged`, `stopped`, and `all`. Use this endpoint to see which jobs the cluster is managing, and their status.	Job claim status Swagger endpoint

REST API Security

In a production environment, the Swagger API may require an open port in a firewall. By default, the API is available on port 8282, but this is configurable.

CLI

Redis Connect includes a command line interface that exposes the management functions provided by the REST API. You can start a CLI instance from the command line as follows:

```
$ ./bin/redisconnect.sh cli
```

Once the CLI has started, type `help` to see the available commands.

Logging

Redis Connect uses Logback for logging. See your Redis Connect distribution's `config/logback.xml` for a sample Logback configuration file.

Redis Connect has been designed to provide descriptive logs to make troubleshooting easier.

If you need to change your application log level at runtime, you can do this using the REST API. See the loglevel REST endpoint documentation for details.

Note that log level changes are not global; they apply only to the instance whose REST API you are connected to. To change the log level for a given instance, connect directly to that instance's REST API.

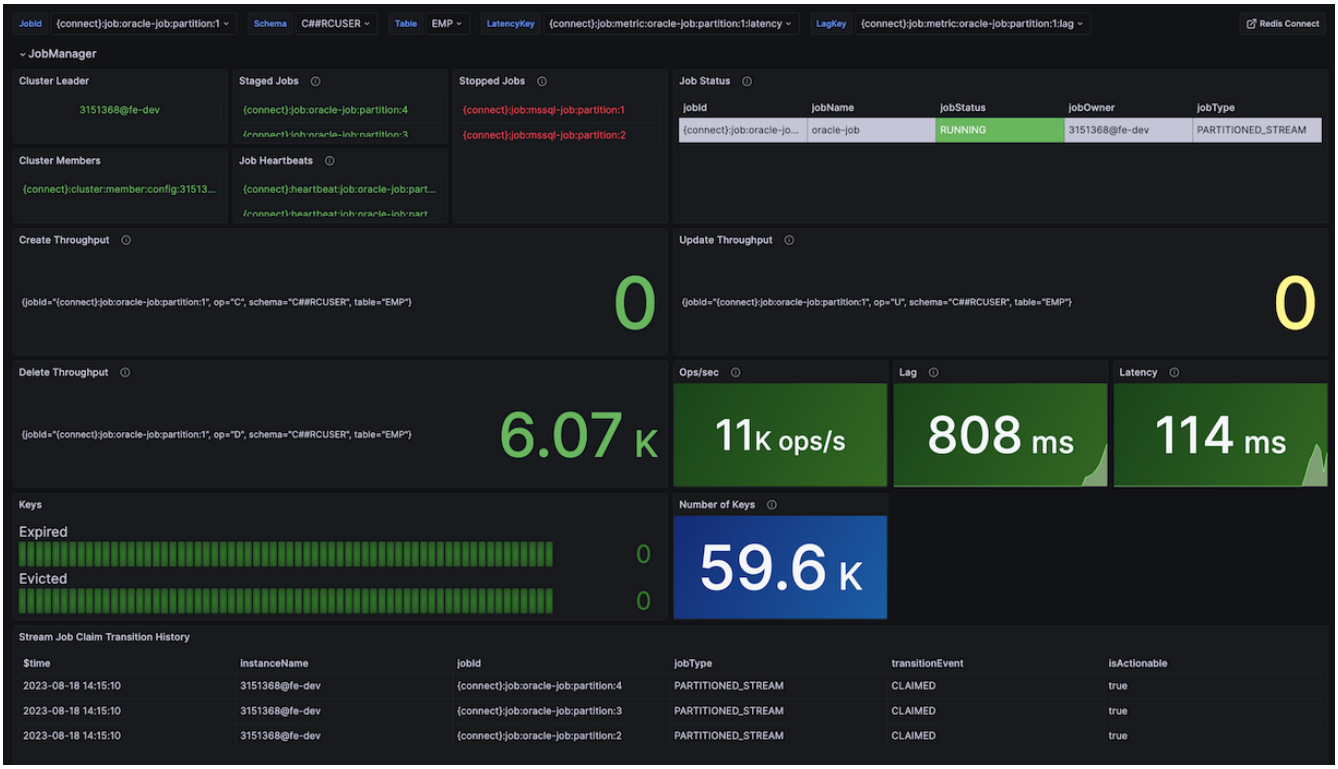
Monitoring

Redis Connect publishes performance metrics to job manager's configuration database. This takes advantage of Redis' time series capabilities.

You can view these metrics in Grafana using the Redis Datasource for Grafana.

Your Redis Connect distribution includes a pre-configured dashboard for viewing key operational metrics. You can find this dashboard and its Grafana configuration at `config/samples/dashboard`, and this configuration can be modified to fit your monitoring requirements.

The sample dashboard looks like this:



Important metrics include the following:

Staged jobs

The list of all configured jobs in the Redis Connect cluster.

Job heartbeats

Heartbeats for the currently active job partitions. Each active partition records a heartbeat every second, indicating that the job is actively being run by a Redis Connect instance.

Ops per second

The number of write operations against the target database completed per second.

Lag

The average elapsed time between the moment a change event is published to the source database and the moment that event is written to the target database (i.e., Redis). High lag values may indicate that your Redis Connect cluster is failing to keep up with the volume of CDC changes. Note that this metric only applies to stream jobs.

Latency

The average amount of time, in milliseconds, that it takes to publish a change event from the source database to Redis.

Security

Redis Connect should be run in a secure network environment. Because Redis Connect connects to source and target databases, those connections may be secured with TLS/SSL and must be secured using authentication credentials.

To support additional security requirements, Redis Connect supports file-based credential rotation, optionally powered by secrets management frameworks like Hashicorp Vault.

TLS/SSL Support

Most database systems support one-way and mutual TLS, and TLS authentication is often required in production environments.

Redis Connect manages certificate-based authentication using Java's KeyStore and TrustStore support. To configure the Java KeyStore and TrustStore, see the KeyStore and TrustStore configuration reference.

To configure Redis Enterprise with TLS, see the documentation:

- TLS with Redis Cloud
- TLS with Redis Enterprise Software

Database ACLs

The Redis job manager and the job target databases should have ACLs enabled. To keep your Redis databases secure and prevent accidental misconfigurations, we recommend the following policis:

1. Create separate Redis database users for your Redis Connect instances, Redis Connect administrators, and developers.
2. Developers should be provided with read-only access to these Redis databases.
3. Ensure that any application user connecting Redis Connect's Redis databases has `dangerous` commands disabled.
4. User should not have permission to delete keys starting with "{connect}". This prevents

accidental deletion of important configuration.

Authentication Credentials

pair source and sink source starts with source name 3 credentails file at minimim 1 for the instance 1 for source 1 for sink we have examples in sample the location is in jobmanager.properties

property files

linux permissions for these files? these contain actual passwords

passwords are in plain text

we listen to these files so that you can rotate these credentails can enable for jobmanager, source, or target

High Availability

TODO

Scaling out

TODO

Supported Source Databases

Redis Connect can capture change data from several RDBMS and NoSQL databases.

The following database are supported. Here, each supported database links to a sa

MySql, Oracle, Postgres, SQLServer, MongoDB, and Gemfire. DB2, Splunk, Vertica, and text files are supported only for initial load jobs.

Define stream vs initial load.

Link to the example in dist. Link to debezium connectors: https://debezium.io/documentation/reference/2.3/connectors/index.html

Note that source database needs to be configured according to Debezium docs - For Demo and POC purposes, you can refer to our container example

Table 2. Supported source databases

Database name	Stream	Initial Load	Debezium Documentation

Cluster and Job Configuration

Configuration

TODO: Link to the Postgres example and explain how it works.

The "jobmanager.properties" file in the "/config" directory is Redis Connect's main configuration file.

Redis Connect runs one or more change-data capture jobs. Each job represents a stream of change-data events that are replicated from a source database to Redis.

To keep track of each of its jobs, Redis Connect stores its state in a Redis database.

The most basic Redis Connect configuration includes the url of the Redis database used to store this state.

This URL is stored at the parameter `redis.connection.url`. The location of the credentials files for the job sources & sinks, 'credentials.file.path'.

This last value should point to the directory where any and all credential files for different jobs will be located.

Cluster properties

Table 3. Cluster properties

Property name	Type	Description	Default
cluster.name	String	Metadata purposes only. Non-functional	default
cluster.leader.heartbeat. lease.renewal.ttl	Integer	TTL (Time-to-Live) which is renewed upon each `cluster.election.attempt.interval` iteration by the cluster leader. Measured in milliseconds with a minimum of 1 second (1000 ms).	5000

Property name	Type	Description	Default
cluster.election.attempt.interval	Integer	Fixed rate scheduled thread which either renews or elects a new cluster leader. Runs on each Redis Connect Instance (JVM) when `job.manager.services.enabled=true`. Measured in milliseconds with a minimum of 1 second (1000 ms)	5000
cluster.timeseries.metrics.enabled	Boolean	Enables creation of a scheduled thread for job metrics reporting to RedisTimeSeries within the Job Management database. RedisTimeSeries is a dependency for this capability. See (Section X.X) for installation instructions.	false

Job manager services properties

Table 4. Job manager services properties

Property name	Type	Description	Default
job.manager.services.enabled	Boolean	Enables creation of scheduled thread(s) to participate in cluster leader elections, facilitate REST API / CLI (Job Manager service), and identify staged jobs without a heartbeat lease (Job Reaper service). When this property is disabled, the Redis Connect instance may still participate in job execution and job claim attempts (Job Claimer service).	true
job.manager.services.threadpool.size	Integer	For non-production deployments, one thread is adequate. In production, we recommend two threads.	2
job.reap.attempt.interval	Integer	The interval between attempts to identify staged jobs without a heartbeat lease. Implemented as a scheduled thread that runs on each Redis Connect Instance (JVM) when `job.manager.services.enabled=true`. Measured in milliseconds with a minimum of 1 second (1000 ms)	7000

Property name	Type	Description	Default
job.claim.service.enabled		Enables creation of scheduled thread(s) to attempt to claim ownership for UNASSIGNED staged jobs (Job Claimer Service), job execution, and job-level metrics reporting (Metrics Reporter service). When this property is disabled, the Redis Connect instance may still participate in cluster leader election, facilitate REST API / CLI, and perform Job Reaper services.	true
job.claim.attempt.interval	Integer	Interval at which this scheduled thread attempts to claim ownership for UNASSIGNED staged jobs. Runs on each Redis Connect Instance (JVM) when `job.claim.service.enabled=true`. Measured in milliseconds with a minimum of 1 second (1000 ms)	5000
job.claim.batch.size.per.attempt	Long	Specifies how many jobs can be claimed per attempt interval. If a sparse topology across many Redis Connect instances is desired, then lowering this interval is recommended.	4

Property name	Type	Description	Default
job.claim.max.capacity	Integer	Specifies the maximum number of jobs that a single Redis Connect instance can claim at any given time.	4
job.claim.heartbeat.lease.renewal.ttl	Integer	TTL (Time-to-Live) which is renewed upon each iteration of a fixed rate scheduled thread that shares its value. Measured in milliseconds with a minimum of 1 second (1000 ms)	10000

REST API Properties

Table 5. REST API properties

Property name	Type	Description	Default
rest.api.enabled	Boolean	Instantiates an embedded Spring Boot Application to host the REST API and/or CLI. To initiate the interactive CLI, start a Redis Connect instance (Java process) from the command line, and pass in the argument "CLI".	true
rest.api.port	Integer	Specifies the port used for the REST API (and SWAGGER) powered by an embedded Spring Boot Application. If you are running multiple Redis Connect instances on the same server, each instance will require a different port for its REST API.	8282

Job Management Database Properties

Table 6. Job Management Database Properties

Property name	Type	Description	Default
redis.connection.url	String	A Redis URI indicating which Redis server to use for job management. For the Redis URI spec, the Lettuce documentation.	n/a
redis.connection.insecure	Boolean	Passed to Lettuce's `RedisURI.verifyPeer`. If true then verifyMode=FULL. Otherwise, if false, then verifyMode=NONE. When peer verification is disabled, Lettuce uses Netty's InsecureTrustManagerFactory.INSTANCE as the trust manager factory. Its javadoc notes that it should never be used in production and that it is purely for testing purposes.	false
redis.connection.timeout.duration	Integer	The timeout is canceled upon command completion/cancellation. Measured in seconds.	1

Property name	Type	Description	Default
redis.connection.auto.reconnect	Boolean	Determine whether the driver will attempt to automatically reconnect to Redis. When enabled, then on disconnect, the client will try to reconnect, activate the connection and re-issue any queued commands.	true
redis.connection.suspend.reconnect.on.protocol.failure	Boolean	When set to `true`, reconnect will be suspended on protocol errors. The reconnect itself has two phases: Socket connection and protocol/connection activation. In case a connection timeout occurs, a connection reset, or host lookup fails, this does not affect the cancellation of commands. In contrast, where the protocol/connection activation fails due to SSL errors or PING before activating connection failure, queued commands are canceled.	true
redis.connection.sslEnabled	Boolean	Enables SSL for one-way or mutual authentication. If this flag is set to `false`, TrustStore and KeyStore will not be passed to the client.	false

Property name	Type	Description	Default
truststore.file.path	String	File path of the Java TrustStore (containing certificates trusted by the client)	n/a
keystore.file.path	String	File path of the Java KeyStore, which stores private key entries, certificates with public keys, or any other secret keys used for various cryptographic purposes.	n/a
credentials.dir.path	String	The name of the directory containing the Redis Connect credentials file. This directory path must include a properties file named `redisconnect_credentials_jobmanager.properties`. Redis Connect never caches or persists credentials. Therefore, on each connection with the source, target, or job manager database, the credentials are read from a file. This enhances security and allows for seamless credential rotations and integration with secret management frameworks such as HashiCorp Vault.	../config/ samples/ credentials

Property name	Type	Description	Default
credentials.rotation.event.listener.enabled	Boolean	When set to `true`, a listener will be created on the `redisconnect_credentials_jobmanager.properties` file within the `credentials.dir.path` to rotate credentials when they change. This lets you rotate credentials without restarting your Redis Connect instance.	false
credentials.rotation.event.listener.interval	Integer	When `credentials.rotation.event.listener.enabled` is set to `true`, this flag sets the frequency at which is scanned for changes. Measured in milliseconds with a minimum of 60 seconds (60000 ms)	60000

Email Alerting Properties

Table 7. Email Alerting Properties

Property name	Type	Description	Default
mail.alert.enabled	Boolean	Enables email alerts when any error forces a job to stop.	false
mail.smtp.host	String	Hostname of the outgoing mail server.	smtp.gmail.com
mail.smtp.port	Integer	Set the non-SSL port number of the outgoing mail server.	587

Property name	Type	Description	Default
mail.smtp.start.tls.enable	Boolean	Set or disable STARTTLS encryption. StartTLS is an extension of the SMTP protocol that tells the email server that the email client wants to use a secure connection using TLS or SSL.	true
mail.smtp.start.tls.required	Boolean	Set or disable the required STARTTLS encryption.	false
mail.to	String	The email address to send alerts to. This email address will also be used as the personal name. Multiple recipients can be added by delimiting them with a comma.	n/a
mail.debug	Boolean	Set session debugging on or off.	false

Job execution configuration

Job properties

Table 8. Job properties

Property name	Type	Description	Constraints	Default
jobName	String	Unique name which is used to derive all other Redis metadata keys related to the job execution workflow. jobName should not be confused with jobId. jobIds are created as part of a job claim. They add-on a namespace to the jobName to identify the jobType and partitionId (if jobType=PARTITIONED_STREAM). When jobName is used in logging or administrative processes (i.e., stopJob), the jobName represents ALL job partitions.	min=4, max=50 regex pattern="[\\s<>(){}\\[\\]'\"/\\\\;`!@#$%&*\|]*$"	n/a

Property name	Type	Description	Constraints	Default
partitions	Integer	Indicates how many partitions to create during startJob process. This attribute is ONLY used to partition a job with jobType=PARTITIONED_STREAM. Not jobType=LOAD. CAUTION: Once a job has started, and job claims are created, a job cannot be repartitioned without deleting all job claims and existing checkpoints. Please reach out to Support to assist with the migration of checkpoints to avoid undesired outcomes.	min=1	1

Property name	Type	Description	Constraints	Default
maxPartitionsPerClusterMember	Integer	The number of job partitions that can be claimed, and executed, on the same Redis Connect instance (JVM). If the limit forces partitions to span more instances than are currently deployed, then the job will not be able to start nor migrate. For example, if maxPartitionsPerClusterMember=1 and partitions=3, then the Redis Connect cluster will require at least 3 instances (JVMs) each with at least 1 available capacity to claim a job partition. This is not a global limit; it is only specific at the job level. 0 represents no limit.	min=0	0
pipeline	Object	See Section 4.3	Not Null	n/a
source	Object	See Section 4.2	Not Null	n/a

Job Source Properties

Table 9. Job Source Properties

Property name	Type	Description	Constraints	Default
pollSource Interval	Long	Fixed rate interval representing how long to pause the producer's polling event loop if no new change events were found in the batch. Measured in milliseconds.	min=5	50
batchSize	Integer	Maximum # of events to dequeue from the source-event-queue AND maximum # of events to query from the source transaction log/table/queue upon each interval of the producer's polling event loop.	min=1	500
source Transaction TimeSequence Enabled	Boolean	When enabled, the source commit/transaction timestamp (and sequence# if the timestamp is the same) will be used to calculate latency metrics and passed along as metadata for Redis Streams sink(s).	n/a	false

Property name	Type	Description	Constraints	Default
slowConsumer MaxRetry Attempts	Integer	-1 = UNLIMITED 0 = DISABLED 1+ = MAX_ATTEMPTS Used as part of back-pressure support for the data pipeline in the event of a slow consumer. If the maximum attempts limit is reached, the job will be stopped for purposes of manual intervention.	min=-1	50
intermittent EventSleep Duration	Integer	Used as part of back-pressure support for the data pipeline in the event of a slow consumer or the circuit breaker is open. Forces the event loop to pause for the configured duration of time. Measured in milliseconds.	min=0	3
source Connection MaxRetry Attempts	Integer	0 = DISABLED 1+ = MAX_ATTEMPTS Maximum retry attempts to reconnect with the source in the event that a connection is lost.	min=0	3

Property name	Type	Description	Constraints	Default
source Connection MaxRetry Duration	Integer	In addition to sourceConnection MaxRetryAttempts, you can also add a max duration, after which retries will stop if the max attempts haven't already been reached. Measured in minutes.	min=1	5
source Connection RetryDelay Interval	Long	Fixed delay in between sourceConnection MaxRetryAttempts. Measured in seconds.	min=0 sourceConnection RetryDelayInterval must be < than sourceConnection RetryDelayInterval sourceConnection RetryDelayInterval must be < than sourceConnection MaxRetryDuration	60
source Connection RetryMaxDelay Interval	Long	Provides an upper bound to calculate the delay interval when sourceConnection RetryDelayFactor is enabled. Measured in seconds.	min=0	240

Property name	Type	Description	Constraints	Default
source Connection RetryDelay Factor	Integer	0 = DISABLED 1+ = DELAY_FACTOR Factor by which delays are exponentially increased after each source connection retry attempt.	min=0	2
database	Object	See Section 4.4 Configuration for all source databases.	Not Null	n/a
tables	Map<String, Table>	See section 4.2.2 Configuration for all source tables/collections/regions/logs properties. Each table within the map requires a unique name which will be used as part of target key composition.	Not Null	n/a

Job Source Database Properties

See Section 4.4

4.2.2 Job Source Table Properties

Table 10. Job Source Table Properties

Property name	Type	Description	Constraints	Default
autoConfig Columns Enabled	Boolean	When enabled, source metadata is queried during the (re)start process to determine sourceColumn names so users do not need to enumerate each within the column's configuration. The columns configuration can be used to override source metadata (i.e., targetName, type, etc.). However, targetKey designation cannot be overridden since only the source table's primary key will be used. This is a common configuration in POCs and development environments since the design of Redis key names are less important than in production. It also allows for less knowledge about the source table schema. This is only supported for RDB sources.	n/a	false

Property name	Type	Description	Constraints	Default
dynamicSchemaEnabled	Boolean	When enabled, columns that are not provided in the columns configuration will be passed through, as-is, to the target. This is currently only supported for MongoDB, Redis Streams Broker, and Files.	n/a	false
prefixTableNameToTargetKeyEnabled	Boolean	When enabled, adds the tableName (defined in the tables configuration) as a prefix to the target Redis key before all other targetKey enabled columns are computed and applied.	n/a	false

Property name	Type	Description	Constraints	Default
deleteOnPrimaryKeyUpdate Enabled	Boolean	When enabled, if the primary key is changed at the source, then an additional operation to DELETE the existing target key will accompany the UPDATE event. This is only supported for RDB sources since primary key changes require a delete and insert of a new row. The DELETE event shares an offset with the UPDATE event both at the source and checkpoint. Redis Connect will handle them within a single pipeline iteration.	n/a	true

Property name	Type	Description	Constraints	Default
changedColumnsOnlyEnabled	Boolean	When enabled, only allows changed (delta) column values to be replicated to the target. This does not include targetKey column(s) which cannot be bypassed. When disabled, all column values will be replicated to the target unless they are individually bypassed at the column-level using changedColumnOnlyEnabled. (See Section 4.2.2) When enabled, the column-level changedColumnOnlyEnabled flag will be overridden for all columns other than those designated as targetKey(s). This is currently only supported for RDB sources.	n/a	false
columns	Job Source Table Column[]	See Job Source Table Column Properties (Section 4.2.2.1)	n/a	Null
initialLoad	Initial Load	See Initial Load Properties (See Section 4.2.2.2)	n/a	Null

Job Source Table Column Properties

Table 11. Job Source Table Column Properties

Property name	Type	Description	Constraints	Default
targetKey	Boolean	Designates this column's value as part of the target's key composition process. When more than one column is designated, the order in which they are listed will impact the order in which they are appended to the key.	n/a	false
sourceColumn	String	Exact match identifier for source column name.	non-empty String	n/a
targetColumn	String	Preferred field name to be used in the target.	Not Empty String	n/a

Property name	Type	Description	Constraints	Default
type	String	Identifies the source column's data type which is used to transform the column value to a properly formatted String within the target. Supported types include: [STRING, VARCHAR, TEXT, INT, DATE, DATE_TIME, BYTE, DEC, NUMERIC, DECIMAL, DOUBLE, FLOAT, LONG, SHORT, RAW, BLOB, CLOB, HASHMAP, CUSTOM] CUSTOM data type is unique in that it bypasses column value transformation to a String which allows it to be converted manually within a Custom Stage. An example would be converting to a proprietary Oracle Timestamp format. Failure to convert this data type manually will cause errors in Redis-based sinks.	regexp = "^[\\s<>(){}\\[\\]'\"/\\\\;`!@#$%&*\|]*$"	STROING

Property name	Type	Description	Constraints	Default
changedColumnOnlyEnabled	Boolean	When enabled, only allows changed (delta) column values to be replicated to the target unless targetKey is enabled. When changedColumnsOnlyEnabled=true at the table-level, this flag will be overridden. This is currently only supported for RDB sources.	n/a	false
passThroughEnabled	Boolean	When disabled, the source column value will not be published to the pipeline therefore it cannot be accessed within a custom stage nor any sink. The purpose of this flag is to allow source column values to be used for targetKey composition without adding the column's name/value pair as a field within the target. As an example, this is common for sources like MongoDB which generate a "_id" key which can be used as a targetKey but has no value as a field.	n/a	true

Property name	Type	Description	Constraints	Default
index	Integer	This is currently for metadata purposes only and has no functional value.	n/a	n/a
dateFormat	String	Used by DATE and DATE_TIME type to override their default. Default formats are as follows: DATE = YYYY-MM-dd DATE_TIME = YYYY-MM-dd HH:mm:ss.S		n/a
nullFormat	String	Users can define how a column value=NULL will be represented in the target.	n/a	Default is an EMPTY String.

Job Source Table Initial Load Properties

Table 12. Job Source Table Column Properties

Property name	Type	Description	Constraints	Default
partitions	Integer	Indicates how many partitions to create during startJob process. This attribute is ONLY used to partition an initial load with jobType=LOAD. Each table should be partitioned based on its own size and release window SLAs. It's common practice to leverage more partitions for an initial load than on streaming. Please see the Production Readiness section for more detail. Disclaimer: If the source table has fewer than 500 rows, which is common in a POC/dev environment, all but partition:1 will be stopped so all the rows are loaded from a single partition.	min=1	1

Property name	Type	Description	Constraints	Default
maxPartitionsPerClusterMember	Integer	Limits how many task partitions can be claimed, and executed, multi-tenant on the same Redis Connect instance (JVM). If the limit forces partitions to span more nodes than are currently deployed, then the initial load will queue the instantiation of tasks until capacity is reallocated (e.g. earlier tasks complete their load partition). This is not a job-level limit; it is only specific at the table level. 0 represents no limit.	min=0	0

Property name	Type	Description	Constraints	Default
customWhere Clause	String	Users can specify a WHERE clause to filter the rows required for initial load. Only the following sources are supported: - RDB sources support JDBC compliant WHERE statements - MongoDB supports a BSON filter - Gemfire supports an Apache Geode WHERE Clause		
rowIndexUsed AsTargetKey Enabled	Boolean	RDB sources can have tables without primary keys. For those cases, rowIndex can be used as a unique identifier for partitioning purposes. This is only supported for RDB sources and only for initial load only / ETL jobs.		false

Job Pipeline Properties

Table 13. Job Pipeline Properties

Property name	Type	Description	Constraints	Default
pipelineBuffer Size	Integer	Redis Connect's pipeline is powered by the LMAX Disruptor library (High Performance Inter-Thread Messaging). The buffer size sets the number of slots allocated within the Disruptor's internal ring buffer "queue". Increasing the buffer size will impact the JVM heap space required to store all transient changed data events within the queue. For most cases, this can be left as default.	min=1024 Must be a power of 2	4096
preprocessor Name	String	Functional interface (Consumer) that can be run before changed-data events are transformed and published to the pipeline. This is currently not extendable by end-users.		n/a

Property name	Type	Description	Constraints	Default
postprocessor Name	String	Functional interface (Consumer) that can be run after changed-data events are transformed and published to the pipeline. This is currently not extendable by end-users.		n/a
stages	Job Pipeline Stage[]	See Job Pipeline Stage Properties (Section 4.3.1)		

Job Pipeline Stage Properties

Table 14. Job Pipeline Stage Properties

Property name	Type	Description	Constraints	Default
stageName	String	Unique name which is used as an exact match reference to a custom-built target sink or a user-defined custom stage.		n/a
index	Integer	Specifies the sequence in which the stages of the pipeline should be orchestrated.	min=1 Begins with 1 and each subsequent index should increment by 1	n/a

Property name	Type	Description	Constraints	Default
metricsEnabled	Boolean	When enabled, the target sink stage will report throughput and latency related metrics for persistence in RedisTimeSeries. This can subsequently be visualized in Grafana.		false
metricsRetentionInHours	Long	Maximum duration for metrics samples as compared to the highest reported timestamp before they expire. Measured in hours.	min=1	4
checkpointStageIndicator	Boolean	Indicates which sink will be responsible for committing the checkpoint to the target database. This is typically performed by the last stage of the pipeline and, often times, it is the only stage in the pipeline.	Job pipeline can only have a single stage with checkpointStageIndicator=true	false

Property name	Type	Description	Constraints	Default
Checkpoint Transactions Enabled	Boolean	Although the producer's polling event loop enqueues changed-data events in batches, each event is processed individually through the pipeline. This is because Redis Connect updates the checkpoint at the changed-data event level and not the batch. When enabled, the checkpoint will be committed as part of an atomic Redis transaction. This eliminates consistency issues and improves performance. Rollback capability is built in to handle any failure scenarios during the transaction so that no data will be lost. When disabled, the checkpoint will be committed after the the changed-data events are written. This adds another network round trip for each changed-data event.	Distributed checkpoints require RediSearch. We use RediSearch to index checkpoint keys so that recovery from the latest checkpoint is immediate.	false

Property name	Type	Description	Constraints	Default
keyPrefix	String	Adds a prefix to the target Redis key before the tableName and composition of targetKey enabled columns.		
userDefinedType		To create a custom stage, a factory interface must be extended so that Redis Connect can have visibility to it from a class loading perspective. See section X.X.X. The interface will force the user to create a `getType()` method which returns a unique String to represent the custom factory. This property must exactly match that custom unique String so that Redis Connect can properly discover and handle it as a custom stage.		
database		See Database Properties (Section 4.4) Configuration for all target database configuration.		

Property name	Type	Description	Constraints	Default
checkpoint Database		See Database Properties (Section 4.4) Checkpoint database configuration. This is only required if Redis is not the target destination, which is only supported for Splunk.		

Database Properties

Table 15. Database Properties

Property name	Type	Description	Default
connectionType	String	Distinguishes between Job Manager, Job Source, Job Target, and Job Checkpoint databases. This field is auto-generated.	
databaseType		The following database types are supported: [DB2, FILES, GEMFIRE, MONGODB, MYSQL, ORACLE, POSTGRES, REDIS, REDIS_STREAMS_MESSAGE_BROKER, SPLUNK, SQL_SERVER, VERTICA] NONE is used for custom stages. Also see userDefinedType. This is a required field.	
databaseURL			
credentials DirectoryPath			

Property name	Type	Description	Default
credentials RotationEvent ListenerEnabled			
credentials RotationEvent ListenerInterval			
custom Configuration			