Redis Database Monitoring 0.5

Contents

Summary

As described by the Redis project, "Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperlogs and geospatial indexes with radius queries. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster."

This Redis database PowerPack supports monitoring of Redis deployments operating as a database, cache or message broker and deployed as part of a wide range of applications.

Supported Versions

This initial PowerPack has been developed and tested using version 8.4 of the ScienceLogic platform. There are no known limitations to its use on any 8.x platform version.

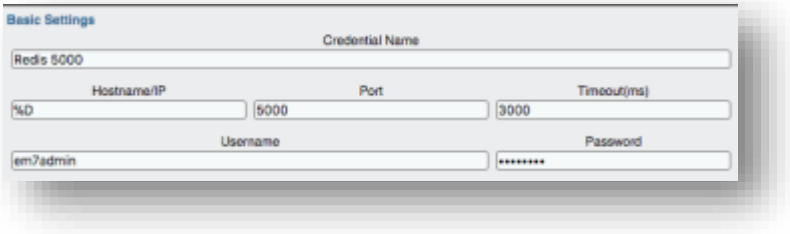
We expect future versions of this PowerPack to require version 8.6.1 of the ScienceLogic platform in order to support automated discovery of Redis clusters.

Deployment

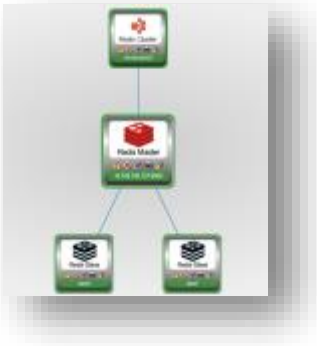
Basic/Snippet Credential

The monitoring "root" device for a Redis database has a single dynamic application aligned with it called *Redis: Master Nodes*

This should be aligned using a basic/snippet credential to match the Redis database being discovered.



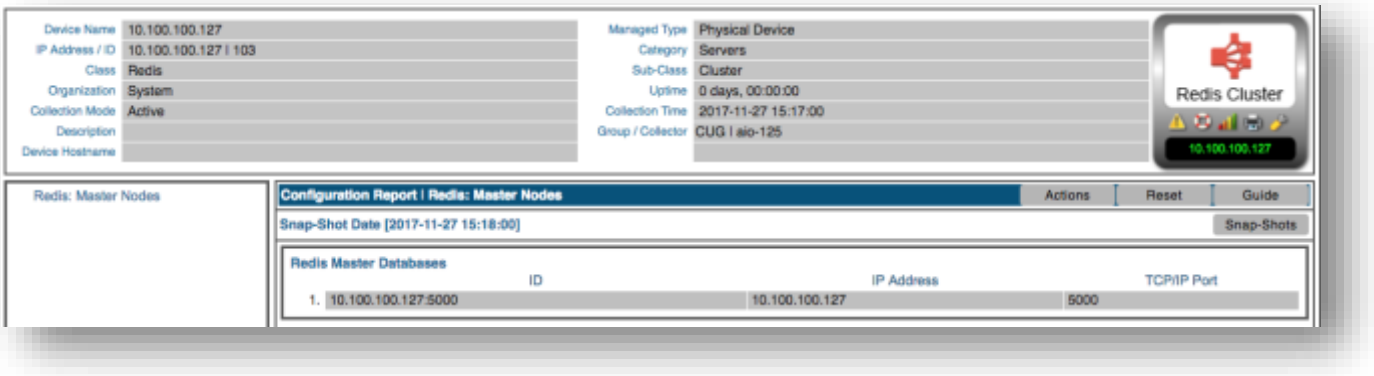
Once this dynamic application is correctly aligned, all other collection will be automatic. Component devices for master and slave nodes will be created as necessary.



Dynamic Applications

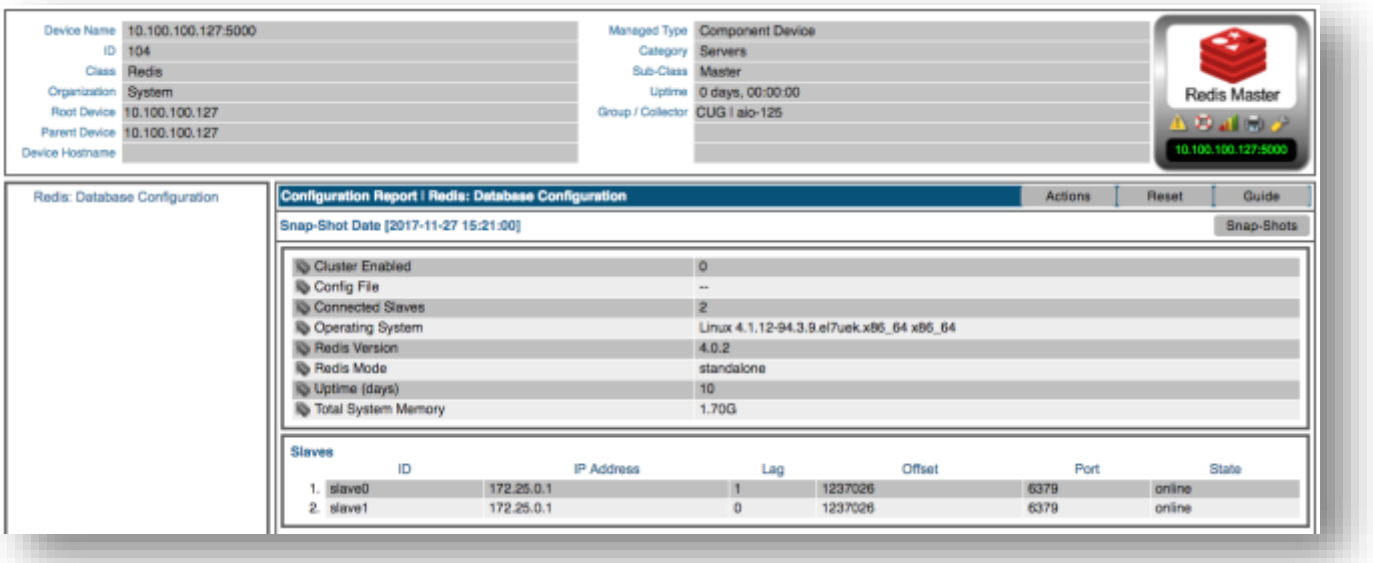
Redis: Master Nodes

This dynamic application provides an inventory of master Redis databases known by this system. This is the only dynamic application aligned with the root node and the only one requiring a specific credential:



Redis: Database Configuration

You can quickly check the overall status of the master database using this table of configuration information on the configuration tab of each master database:

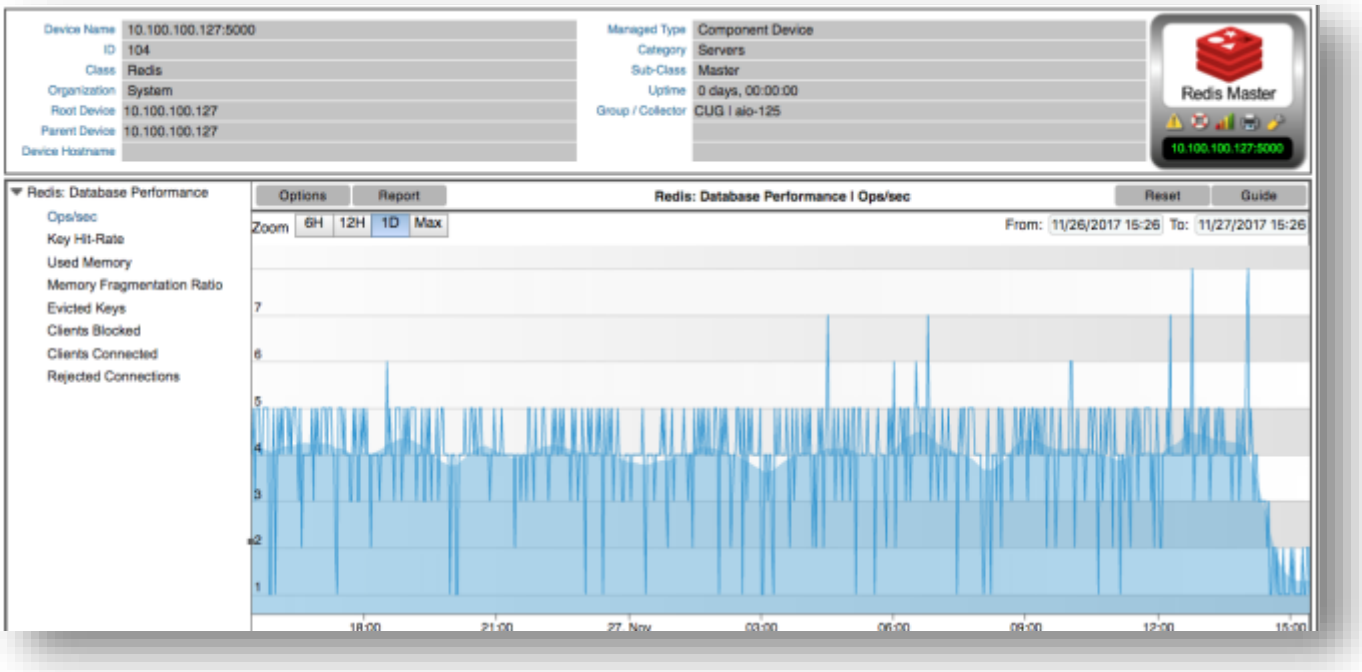


Redis: Database Performance

The metrics gathered by the Redis: Database Performance dynamic application allow you to gather insight into how your Redis is performing over time. Metrics include:

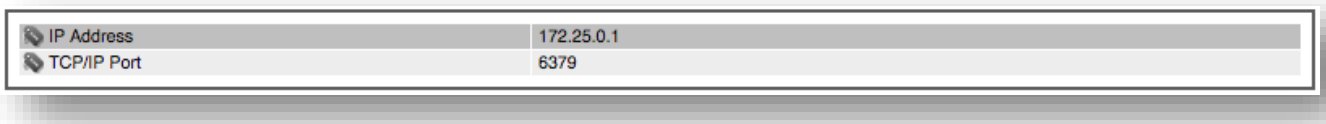
- Ops/Sec – The key usage metric showing utilization of your Redis store / message broker.

- Key Hit-Ratio – When operating Redis as a cache, this is an important measure of the effectiveness of the cache. If the key hit-ratio is to low then your overall application performance may be impacted by having to retrieve data from a slower source.
- Used Memory – A key metric when used in conjunction with a measure of overall system memory, or the maxmemory that has been configured for your particular Redis instance. Excessive used memory can result in a lower cache key-hit ratio or swapping, resulting in a major performance impact.
- Memory Fragmentation Ratio – Gives an indication of how fragmented memory is on the Redis host. For a moderately busy system, this value should be close to one.
- Evicted Keys – Redis can be configured to apply different key eviction policies according to your application needs. In some applications, a steady stream of evictions is normal, whilst in others you expect no evicted keys.
- Clients Blocked – Clients can be blocked waiting for data when performing list operations. Current Redis blocking commands are: BLPOP, BRPOP, and BRPOPLPUSH. In some applications / time periods a number of blocked clients may be normal, but anomalies may reflect application specific failure conditions.
- Rejected Connection – An indication that maximum client connections have been reached. The PowerPack includes a preconfigured event for this metric since rejected connections are always considered a failure condition.



Redis: Slave Configuration

Redis Slave Configuration captures basic slave identifier information.



Redis: Slave Performance

The Redis Slave Performance dynamic application reports on metrics tied to the slave integration with the Redis master database:

- Status – Availability of the slave as seen by the master
- Lag time – How far behind the slave is from the master in terms of replication seconds
- Delta offset – The number of bytes which are different between the master and the slave