Redis Administration

This page contains topics related to the administration of Redis instances. Every topic is self contained in form of a FAQ. New topics will be created in the future.

Redis setup hints

- We suggest deploying Redis using the **Linux operating system**. Redis is also tested heavily on OS X, and tested from time to time on FreeBSD and OpenBSD systems. However Linux is where we do all the major stress testing, and where most production deployments are running.
- Make sure to set the Linux kernel **overcommit memory setting to 1**. Add `vm.overcommit_memory = 1` to `/etc/sysctl.conf` and then reboot or run the command `sysctl vm.overcommit_memory=1` for this to take effect immediately.
- Make sure Redis won't be affected by Linux kernel feature *transparent huge pages*, otherwise it will impact greatly both memory usage and latency in a negative way. This is accomplished with the following command: `echo madvise > /sys/kernel/mm/transparent_hugepage/enabled`.
- Make sure to **setup some swap** in your system (we suggest as much as swap as memory). If Linux does not have swap and your Redis instance accidentally consumes too much memory, either Redis will crash when it is out of memory or the Linux kernel OOM killer will kill the Redis process. When swapping is enabled Redis will work in a bad way, but you'll likely notice the latency spikes and do something before it's too late.
- Set an explicit `maxmemory` option limit in your instance in order to make sure that the instance will report errors instead of failing when the system memory limit is near to be reached. Note that `maxmemory` should be set calculating the overhead that Redis has, other than data, and the fragmentation overhead. So if you think you have 10 GB of free memory, set it to 8 or 9.+ If you are using Redis in a very write-heavy application, while saving an RDB file on disk or rewriting the AOF log **Redis may use up to 2 times the memory normally used**. The additional memory used is proportional to the number of memory pages modified by writes during the saving process, so it is often proportional to the number of keys (or aggregate types items) touched during this time. Make sure to size your memory accordingly.
- Use `daemonize no` when running under daemontools.
- Make sure to setup some non trivial replication backlog, which must be set in proportion to the amount of memory Redis is using. In a 20 GB instance it does not make sense to have just 1 MB of backlog. The backlog will allow replicas to resynchronize with the master instance much easily.

- Even if you have persistence disabled, Redis will need to perform RDB saves if you use replication, unless you use the new diskless replication feature. If you have no disk usage on the master, make sure to enable diskless replication.
- If you are using replication, make sure that either your master has persistence enabled, or that it does not automatically restarts on crashes: replicas will try to be an exact copy of the master, so if a master restarts with an empty data set, replicas will be wiped as well.
- By default Redis does not require **any authentication and listens to all the network interfaces**. This is a big security issue if you leave Redis exposed on the internet or other places where attackers can reach it. See for example this attack to see how dangerous it can be. Please check our security page and the quick start for information about how to secure Redis.
- LATENCY DOCTOR and MEMORY DOCTOR are your friends.

Running Redis on EC2

- Use HVM based instances, not PV based instances.
- Don't use old instances families, for example: use m3.medium with HVM instead of m1.medium with PV.
- The use of Redis persistence with **EC2 EBS volumes** needs to be handled with care since sometimes EBS volumes have high latency characteristics.
- You may want to try the new **diskless replication** if you have issues when replicas are synchronizing with the master.

Upgrading or restarting a Redis instance without downtime

Redis is designed to be a very long running process in your server. For instance many configuration options can be modified without any kind of restart using the CONFIG SET command.

Starting from Redis 2.2 it is even possible to switch from AOF to RDB snapshots persistence or the other way around without restarting Redis. Check the output of the `CONFIG GET *` command for more information.

However from time to time a restart is mandatory, for instance in order to upgrade the Redis process to a newer version, or when you need to modify some configuration parameter that is currently not supported by the CONFIG command.

The following steps provide a very commonly used way in order to avoid any downtime.

- Setup your new Redis instance as a slave for your current Redis instance. In order to do so you need a different server, or a server that has enough RAM to keep two instances of Redis running at the same time.
- If you use a single server, make sure that the slave is started in a different port than the master instance, otherwise the slave will not be able to start at all.
- Wait for the replication initial synchronization to complete (check the slave log file).

- Make sure using INFO that there are the same number of keys in the master and in the slave. Check with redis-cli that the slave is working as you wish and is replying to your commands.
- Allow writes to the slave using **CONFIG SET slave-read-only no**
- Configure all your clients in order to use the new instance (that is, the slave). Note that you may want to use the CLIENT PAUSE command in order to make sure that no client can write to the old master during the switch.
- Once you are sure that the master is no longer receiving any query (you can check this with the MONITOR command), elect the slave to master using the **SLAVEOF NO ONE** command, and shut down your master.

If you are using Redis Sentinel or Redis Cluster, the simplest way in order to upgrade to newer versions, is to upgrade a slave after the other, then perform a manual fail-over in order to promote one of the upgraded replicas as master, and finally promote the last slave.

Note however that Redis Cluster 4.0 is not compatible with Redis Cluster 3.2 at cluster bus protocol level, so a mass restart is needed in this case. However Redis 5 cluster bus is backward compatible with Redis 4.
