**Redis Persistence**

**Snapshot persistence aka RDB**

The first, and default mode, is the snapshot mode (also called RDB). It provides a compromise between performance and persistence. Data are saved as .rdb files to disk periodically, according to a defined number of write operations, asynchronously. It means that in the scope of an incident, recent data are lost. Here are the saving rules chosen by Scalingo:

• Every 900 seconds if at least 1 write operation has occurred

• Every 300 seconds if at least 10 write operations have occurred

• Every 60 seconds if at least 10000 write operations have occurred

**Strong persistence aka AOF**

The second mode provides Redis with the highest level of persistence (also called AOF). Any command impacting the dataset will be saved synchronously in a file. In case of incident, Redis replays the logged operations to reconstruct the database dataset.

The typical use case is when using Redis as a key-value store. We can imagine the comments on a blog post page being stored in a Redis list. Losing comments because of an unexpected Redis restart is not acceptable and activating AOF mode sounds like a good idea.

**OK, so which persistence mode should I use?**

If your target is the same kind of data safety provided by your typical SQL database, you should use both AOF and RDB.

If you care a lot about your data, but still can live with a few minutes of data loss in case of disasters, you can simply use RDB alone, you’ll get better performance.

Whatever is the configuration you choose, backups triggered by our system are generating a RDB snapshot of the database remotely, the snapshot is directly streamed to our backup controllers which encrypt and archive them.

**Cache mode**

The last available configuration is related to how long data are stored in Redis. When cache mode is activated, if the memory is full, Redis will automatically drop the least used data to free up memory for most recent data. Be cautious as any data can be deleted. Don’t use your Redis instance to store important information in this mode.

The typical use case is Scalingo’s homepage and blog you’re browsing right now. It uses Redis as a cache for HTML fragments. In such case, we can afford using the oldest data stored in Redis, and re-compute them if needed.

This feature uses two settings of Redis:

• maxmemory: Amount of data redis accepts before dropping some of them. It is set at the value of the memory available of the database plan you’ve chosen.

• maxmemory-policy: Set at allkeys-lru, which means ‘Least Recently Used’ keys will be evicted first if the maxmemory amount is reached.