Introduction to Memorystore in GCP

CTO Cloud Meetup @ Gen

2024-05-02

Petr Zemek

Outline

- What is Redis?
- What is Memorystore?
- Deploying a Memorystore instance
- Connecting to a Memorystore instance
- Monitoring a Memorystore instance
- Migrating from Redis to Memorystore
- Additional topics
- Available documentation
- Summary and discussion

All the code and commands I will use in the live demos are available [redacted]

What is Redis?

redis

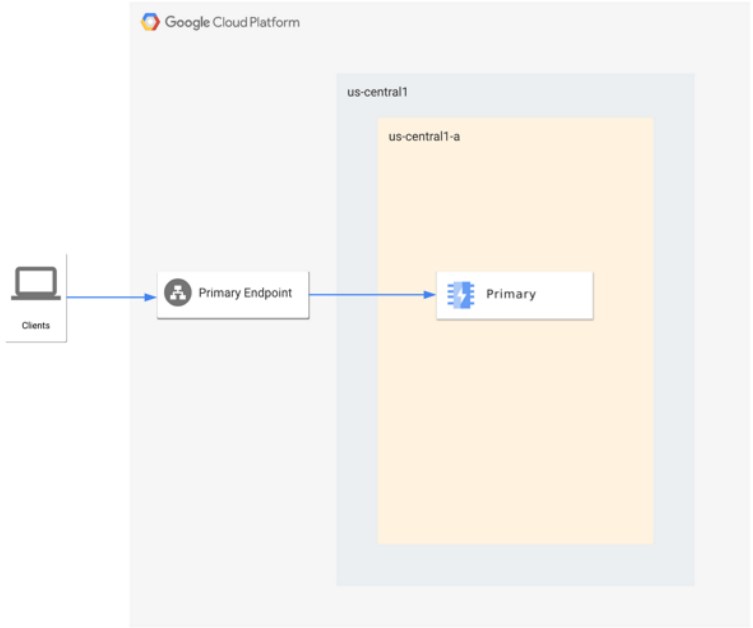
- In-memory data store / key-value database
- Supports **various use cases**
  - General: cache, database, message broker
  - Specific: session storage, rate limiter, counter, distributed lock, rank/leaderboard, ...
- Provides many native **data types** (+ is extensible with others)
  - Basic: integers, strings, lists, sets, sorted sets, hashes
  - Advanced: streams, bitmaps, bitfields, geospatial indexes, probabilistic (HyperLogLog, Bloom filter, ...)
- **High-availability** (HA) and **read-scaling** support via async master-slave replication (Redis Sentinel)
- **Read/write scaling** via sharding (Redis Cluster)
- **Other features**: transactions, key expiration/eviction, (optional) on-disk persistence, Lua scripting
- There are **libraries/clients** for a lot of programming languages
- **Deployment modes**: (1) App-instance-specific cache, (2) Central app cache/database/...
- Originally open-source, **source-available since 2024-03**; open-source fork Valkey and others

What is Memorystore?

- **Fully managed Redis** (or Memcached) service in GCP
- 100% **compatible** with Redis (currently up to 7.2)
- There are some **differences** though (covered later)
- **Connectivity** via Private Service Access (PSA) - similarly to Cloud SQL
- Can be deployed in an **HA mode** (but natively only within a single region)
- Provisioning, replication, failover, and patching are all **automated**
- Provides **monitoring support**
- **Billed** by the hour by the **memory capacity** that you provision (no direct CPU-based costs; threaded I/O)
- Supports **scaling** up to 300 GB of memory, 16 Gbps of network throughput, 5 read replicas
- Optional **authentication via AUTH** (ACL or IAM authentication/authorization is not supported)
- Optional **in-transit TLS encryption**, server CA certificate is valid for 10 years (5-year migration window)
- **Three available tiers** (covered next)
- Documentation: official, [redacted]

Available tier I – Basic tier

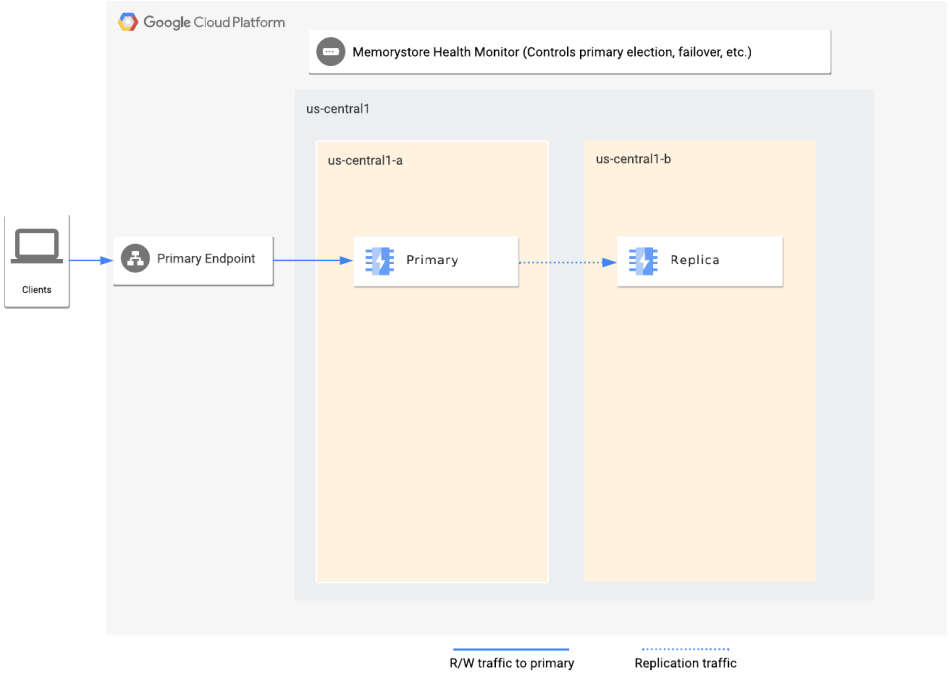
- One Redis server
- Use cases:
  - Dev/test (maybe stage)
  - Ephemeral cache without HA



Source: Google documentation

Available tier II – Standard tier

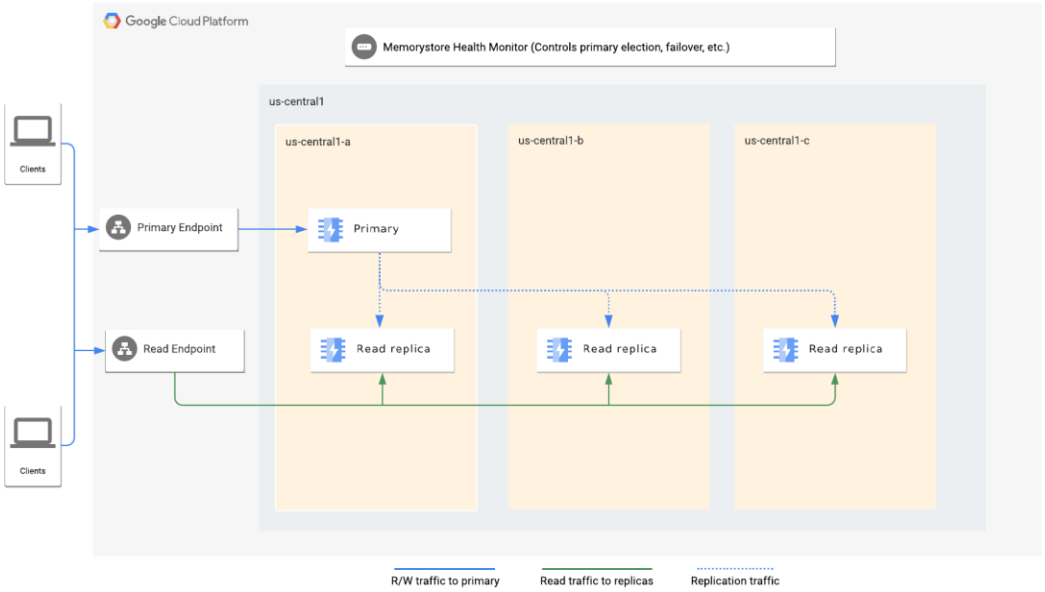
- Provides HA via replication
- Use cases:
  - Production
  - When a single Redis server is enough to handle the read traffic



Source: Google documentation

Available tier III – Standard tier with read replicas

- Provides HA with replication and distributed reads
- Use cases:
  - Production
  - When a single Redis server is not enough to handle the read traffic
  - When you want a higher replication factor



Source: Google documentation

Deploying a Memorystore instance

- (Live demo)
- [redacted]

Connecting to a Memorystore instance

- Connecting from applications
  - (Live demo – Python app running in GKE)
  - [redacted]
- Connecting from your workstation
  - (Live demo – redis-cli)
  - [redacted]

Monitoring a Memorystore instance

- (Live demo)
- Metrics (available metrics)
  - GCP
  - Grafana
- Logs
  - GCP (`resource.type="redis_instance"`)
- [redacted]

Migrating from Redis to Memorystore

- (Live demo)
- [redacted]

Additional topics

- Backups/persistence: **only RDB snapshots** for auto-data-recovery use (1-24 hours), AOF is not supported
- Supports **upgrades** (memory capacity, version) and **downgrades** (memory capacity)
- Some **Redis parameters** can be customized (e.g. `maxmemory-gb` and `maxmemory-policy`)
- Some **Redis commands** are blocked by Google
- Redis Sentinel is **not supported** as it is not needed
- Redis Cluster (**sharding**) is supported since 2023-08
- Supports **manual failover** for **HA testing**
- Follow **best practices / tips**
  - Select a proper `maxmemory-policy` based on your app (`noeviction`, `volatile-*`, `allkeys-*`)
  - Set `maxmemory-gb` to 80-90% of available capacity to reduce potential performance issues
  - Set up monitoring and alerting (at least CPU and memory usage)
  - Be aware that some Redis commands can be CPU-intensive (e.g. `KEYS *`)

Available documentation

- Official Redis documentation
- Official Memorystore documentation
- [redacted]

All the code and commands I have used in the live demos are available [redacted]

Summary and discussion

- **Redis** is an in-memory data store / key-value database
- **Memorystore** is fully managed Redis in GCP
- Can be used for **various purposes**: cache, database, message broker, …
- **Supports** HA, monitoring, backups, import/export of data, automatic provisioning/failover, patching
- **Three available tiers** (single Redis server, HA, HA + read replicas)
- **Deployment** via Terraform
- Optional **authentication** via AUTH, **in-transit encryption**, customization of **eviction policies**
- Does **not support Redis Sentinel** (Google uses their own failover mechanism)
- If you need to scale both reads and writes, you can use **Redis Cluster** (sharded Redis)