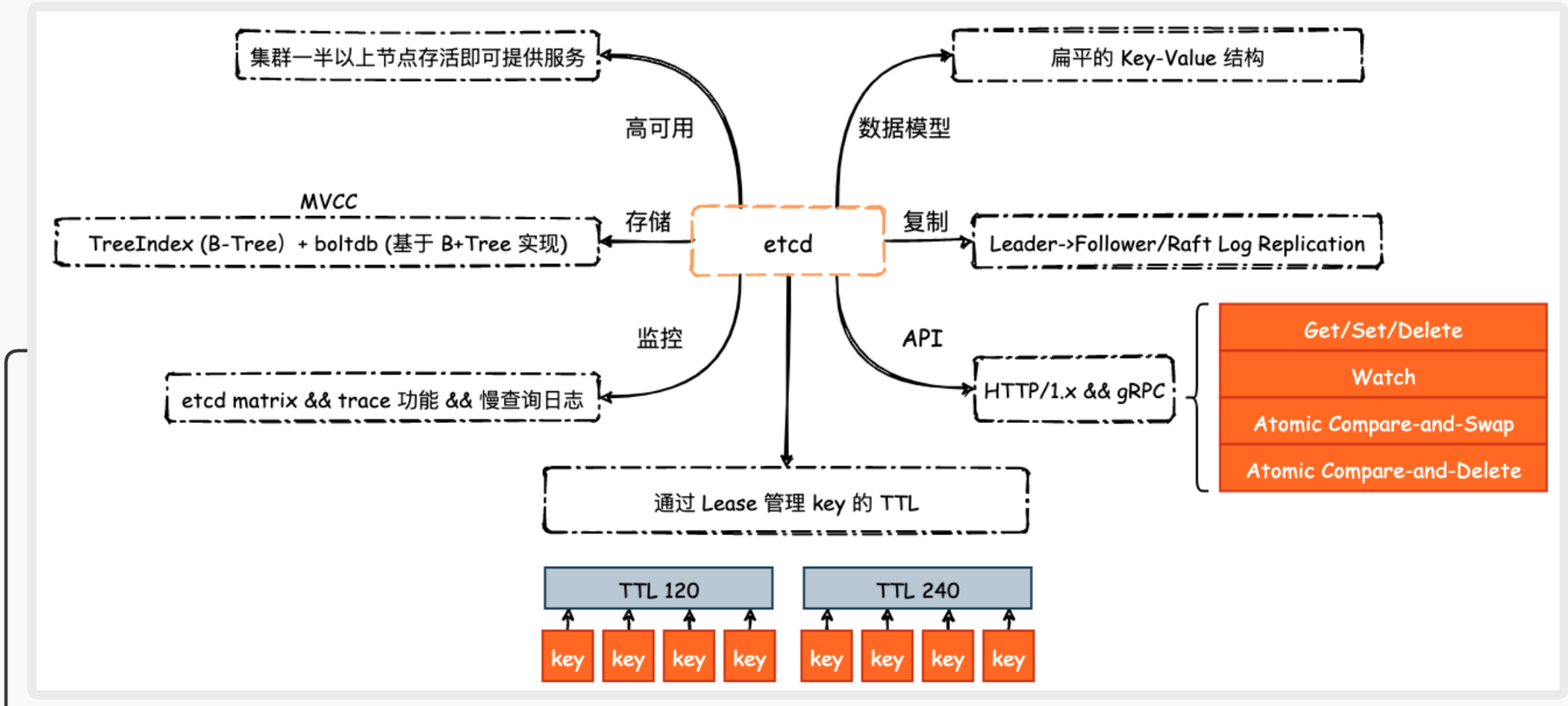
etcd 概述

Feature



- 高可用
  - etcd 基于 Raft 共识算法实现，因此只要集群中有超过半数以上节点存活，Leader 选举、日志复制等功能就可正常运行
- 强一致性
  - etcd 不仅仅能保证集群的高可用，并且能够保证数据的读写强一致性，这同样是基于 Raft 一致性协议实现的。正因如此，etcd 的读写性能本身不会特别高
- 存储与查询
  - etcd v3 采用了 IndexTree（B-Tree） 和 boltdb（B+Tree存储）实现了一个 MVCC 数据库，数据模型从原来的基于目录的层次结构转变为扁平的 key-value 结构，同时支持多个 key 的事务
- TTL
  - etcd v3 使用 Lease 优化了原有的 TTL 机制，每一个 Lease 具有一个 TTL，相同 TTL 的 key 关联至同一个 Lease。当 Lease 到期后，删除掉与该 Lease 所有相关联的 key，无需为每一个 key 单独设置 TTL
- Watch API
  - **etcd 能作为 k8s 最核心的部件之一的原因就在于 etcd 支持 Watch API。k8s 的声明式 API 可以说就是依靠 etcd 的 Watch API 实现的: k8s controller 通过 Watch API 监视 Deployment、Service 等资源的变化，对比实际状态与期望状态是否保持一致，并根据条件进行"调谐"**
  - Watch API 简单的来说就是当某一条数据发生变化时，etcd 将会通过 HTTP 请求将变化告知外界
- 低容量存储
  - etcd 在设计之初只是需要一个协调服务来存储服务配置信息、提供分布式锁以及 Watch API 等功能，也就是说，它存储的是"元数据"，而非应用数据。因此，etcd 并不适合用于存储海量数据，社区建议 etcd 存储数据大小不超过 8G

VS Redis

- 我们可以通过数据模型、数据复制、性能以及提供的 API 这 4 个维度和 Redis 进行比较
- 数据模型
  - Redis
    - Redis 提供了诸如 List、Hash、Set、ZSet 以及 Bitmap 等多种数据结构
  - etcd
    - etcd 则只提供了 key-value 存储
- 数据复制
  - Redis
    - Redis 通常是主备异步复制，可能会丢失数据
  - etcd
    - etcd 则需要将日志发送给超过半数的 Follower，并且接收到了它们的确认才能够真正的复制一条数据，保证了强一致性
- 性能
  - 在性能方面，Redis 甩 etcd 十条街，因为 Redis 完全基于内存实现，而 etcd 需要将数据持久化至底层 DB 中
- API
  - Redis
    - Redis 对外仅提供了 socket API，我们要么使用 redis-client 客户端工具，要么使用语言所实现的第三方库来连接和使用其 API，并没有提供诸如 HTTP 或者是 gRPC 接口
  - etcd
    - 提供了 HTTP/1.x 以及 gRPC 接口，方便开发人员调试以及降低开发人员的使用成本。k8s 内部就是用 gRPC 的方式和 etcd 进行通信，利用 gRPC 多路复用、Header 压缩等特性降低通信成本