# 大规模 APISIX 场景下的 etcd 治理实践—Kstone

王超凡

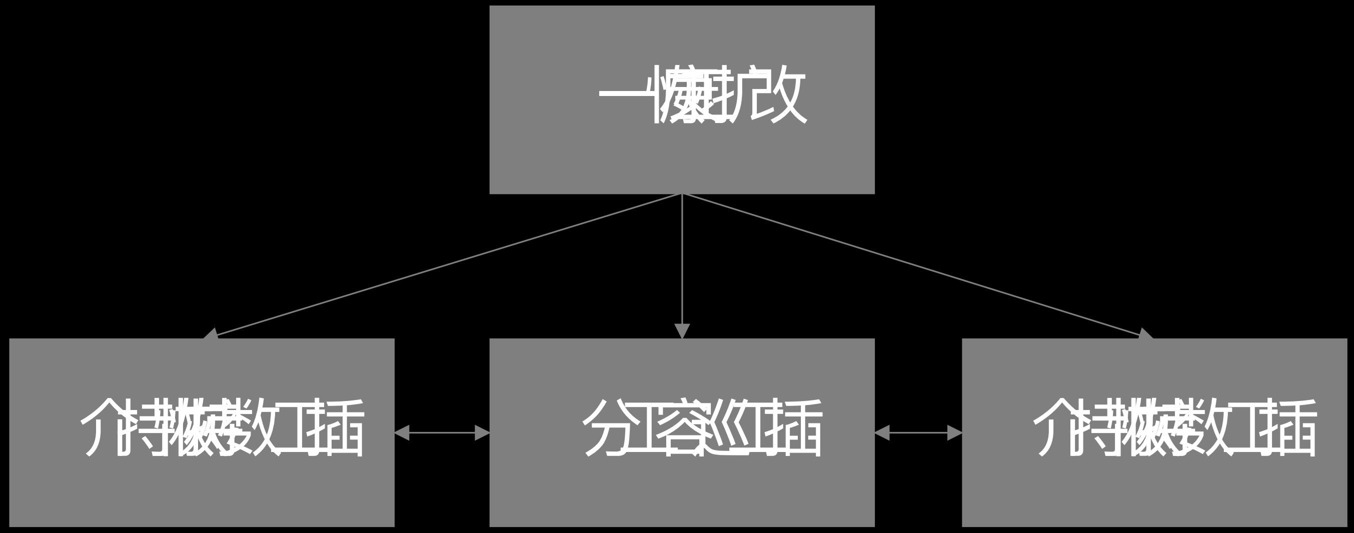
腾讯高级工程师

APACHE APISIX CONNECTS THE WORLD

## CONTENT

## 01 etcd

## 什么是 etcd?



The name “etcd” originated from two ideas, the unix “/etc” folder and “d"istributed systems. The “/etc” folder is a place to store configuration data for a single system whereas etcd stores configuration information for large scale distributed systems. Hence, a “d"istributed “/etc” is “etcd”.

- 分布式、强一致、高可用的 key-value 健值关键元数据存储系统
- 典型应用场景：服务发现、分布式锁、Leader 选举、配置管理： Kubernetes，APISIX 等用来存储关键元数据配置
- 可用性: 一半以上节点存活即可提供服务
- 一致性: raft 算法保证各节点数据一致性
- API:
  - Put a b
  - Range a b
  - Txn
  - Watch
  - …

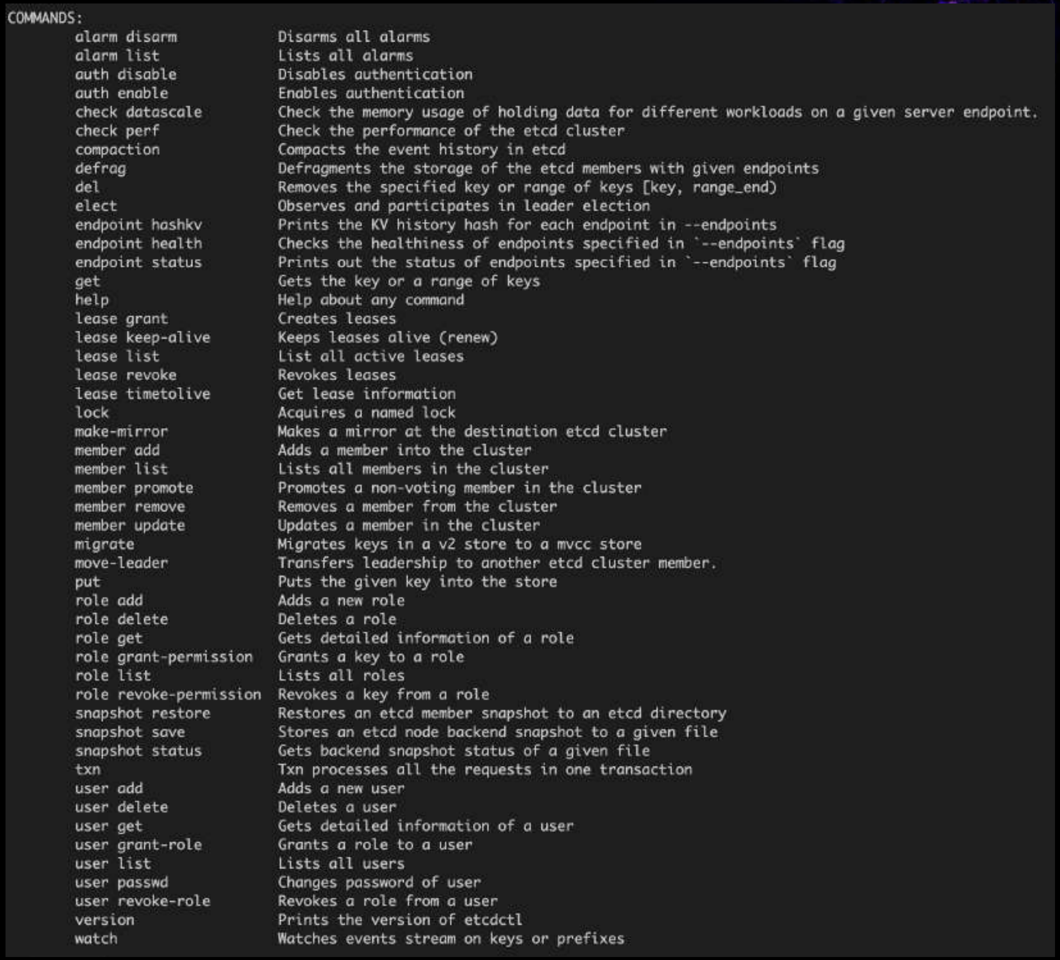
## etcd API 简介

数据处理相关 API

- KV - Range, Put, DeleteRange, Txn, Compact. 对数据进行增删改查，版本压缩等
- Watch – Watch. 监听 Key 的变化
- Lease – LeaseGrant, LeaseRevoke, LeaseTimeToLive, LeaseLeases, LeaseKeepAlive. 租约相关操作

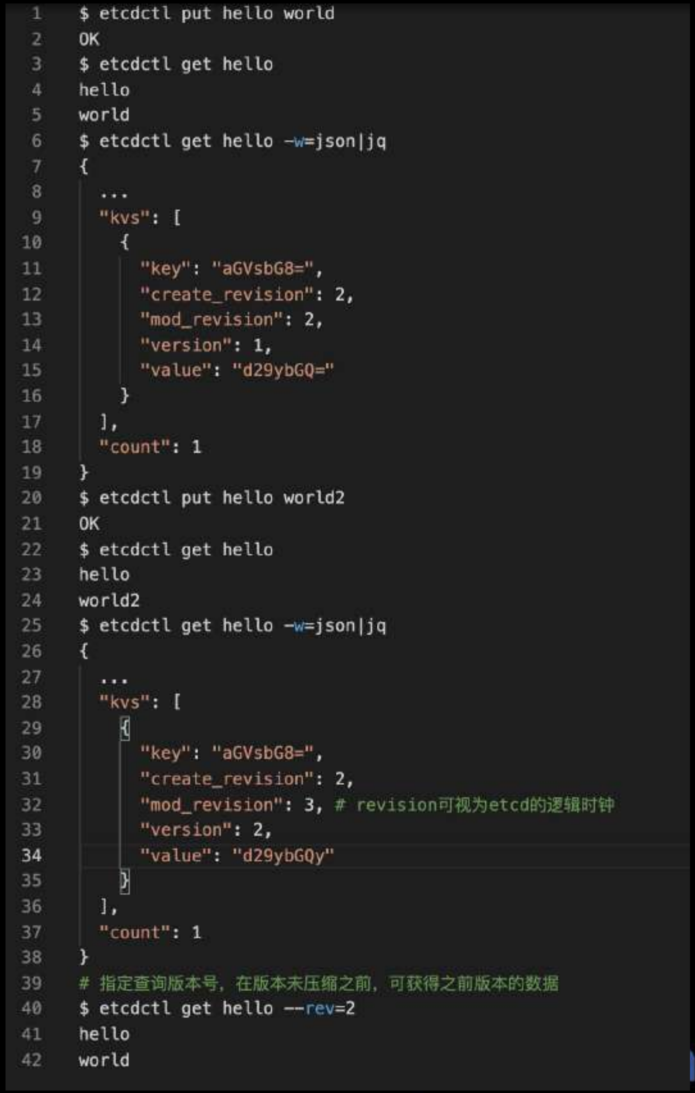
集群管理和运维相关的 API:

- Auth – 基于 RBAC 的鉴权相关的API.
- Cluster – 成员变更管理相关的 API.
- Maintenance – 集群备份恢复，碎片整理相关的 API.

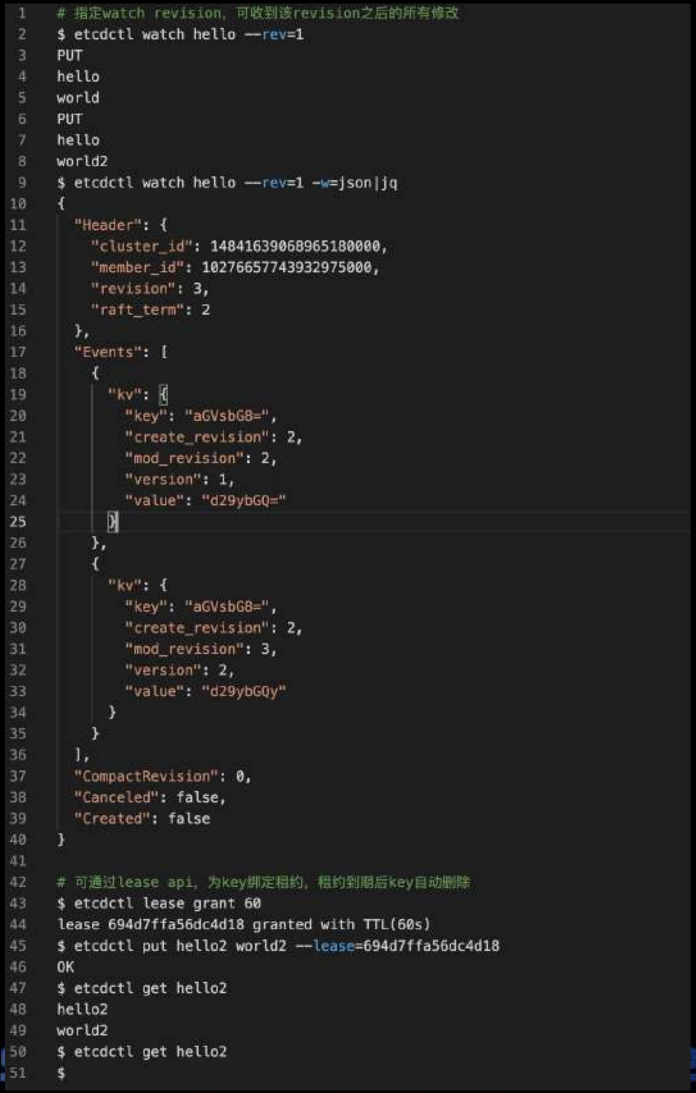


```
COMMANDS:
        alarm disarm            Disarms all alarms
        alarm list              Lists all alarms
        auth disable            Disables authentication
        auth enable             Enables authentication
        check datascale         Check the memory usage of holding data for different workloads on a given server endpoint.
        check perf              Check the performance of the etcd cluster
        compaction              Compacts the event history in etcd
        defrag                  Defragments the storage of the etcd members with given endpoints
        del                     Removes the specified key or range of keys [key, range_end)
        elect                   Observes and participates in leader election
        endpoint hashkv         Prints the KV history hash for each endpoint in --endpoints
        endpoint health         Checks the healthiness of endpoints specified in `--endpoints` flag
        endpoint status         Prints out the status of endpoints specified in `--endpoints` flag
        get                     Gets the key or a range of keys
        help                    Help about any command
        lease grant             Creates leases
        lease keep-alive        Keeps leases alive (renew)
        lease list              List all active leases
        lease revoke            Revokes leases
        lease timetolive        Get lease information
        lock                    Acquires a named lock
        make-mirror             Makes a mirror at the destination etcd cluster
        member add              Adds a member into the cluster
        member list             Lists all members in the cluster
        member promote          Promotes a non-voting member in the cluster
        member remove           Removes a member from the cluster
        member update           Updates a member in the cluster
        migrate                 Migrates keys in a v2 store to a mvcc store
        move-leader             Transfers leadership to another etcd cluster member.
        put                     Puts the given key into the store
        role add                Adds a new role
        role delete             Deletes a role
        role get                Gets detailed information of a role
        role grant-permission   Grants a key to a role
        role list               Lists all roles
        role revoke-permission  Revokes a key from a role
        snapshot restore        Restores an etcd member snapshot to an etcd directory
        snapshot save           Stores an etcd node backend snapshot to a given file
        snapshot status         Gets backend snapshot status of a given file
        txn                     Txn processes all the requests in one transaction
        user add                Adds a new user
        user delete             Deletes a user
        user get                Gets detailed information of a user
        user grant-role         Grants a role to a user
        user list               Lists all users
        user passwd             Changes password of user
        user revoke-role        Revokes a role from a user
        version                 Prints the version of etcdctl
        watch                   Watches events stream on keys or prefixes
```

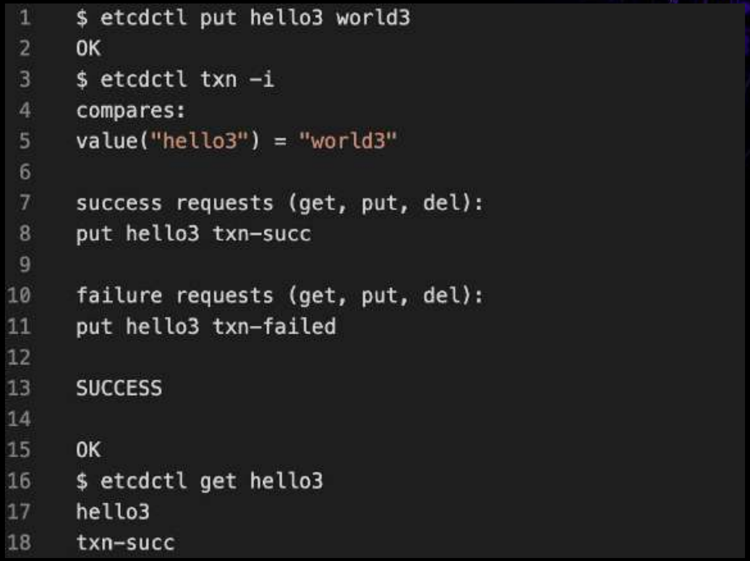
## etcd API 体验



```
$ etcdctl put hello world
OK
$ etcdctl get hello
hello
world
$ etcdctl get hello -w=json|jq
{
  ...
  "kvs": [
    {
      "key": "aGVsbG8=",
      "create_revision": 2,
      "mod_revision": 2,
      "version": 1,
      "value": "d29ybGQ="
    }
  ],
  "count": 1
}
$ etcdctl put hello world2
OK
$ etcdctl get hello
hello
world2
$ etcdctl get hello -w=json|jq
{
  ...
  "kvs": [
    {
      "key": "aGVsbG8=",
      "create_revision": 2,
      "mod_revision": 3, # revision可视为etcd的逻辑时钟
      "version": 2,
      "value": "d29ybGQy"
    }
  ],
  "count": 1
}
# 指定查询版本号，在版本未压缩之前，可获得之前版本的数据
$ etcdctl get hello --rev=2
hello
world
```

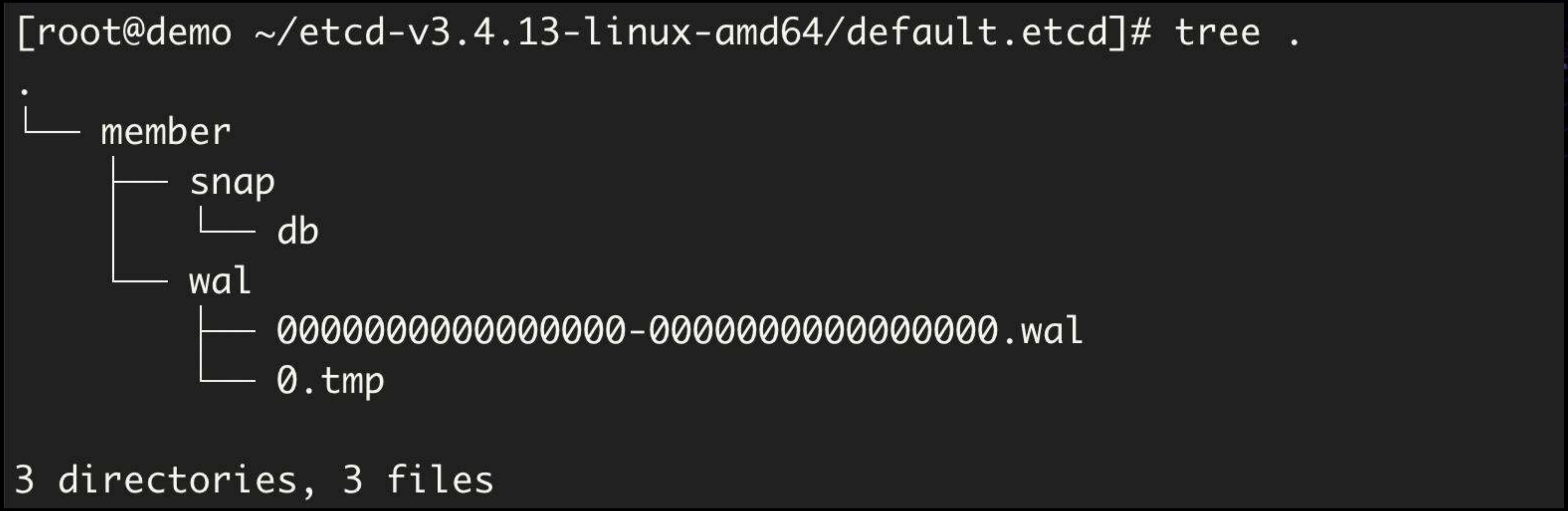


```
# 指定watch revision，可收到该revision之后的所有修改
$ etcdctl watch hello --rev=1
PUT
hello
world
PUT
hello
world2
$ etcdctl watch hello --rev=1 -w=json|jq
{
  "Header": {
    "cluster_id": 14841639068965180000,
    "member_id": 10276657743932975000,
    "revision": 3,
    "raft_term": 2
  },
  "Events": [
    {
      "kv": {
        "key": "aGVsbG8=",
        "create_revision": 2,
        "mod_revision": 2,
        "version": 1,
        "value": "d29ybGQ="
      }
    },
    {
      "kv": {
        "key": "aGVsbG8=",
        "create_revision": 2,
        "mod_revision": 3,
        "version": 2,
        "value": "d29ybGQy"
      }
    }
  ],
  "CompactRevision": 0,
  "Canceled": false,
  "Created": false
}

# 可通过lease api，为key绑定租约，租约到期后key自动删除
$ etcdctl lease grant 60
lease 694d7ffa56dc4d18 granted with TTL(60s)
$ etcdctl put hello2 world2 --lease=694d7ffa56dc4d18
OK
$ etcdctl get hello2
hello2
world2
$ etcdctl get hello2
$
```



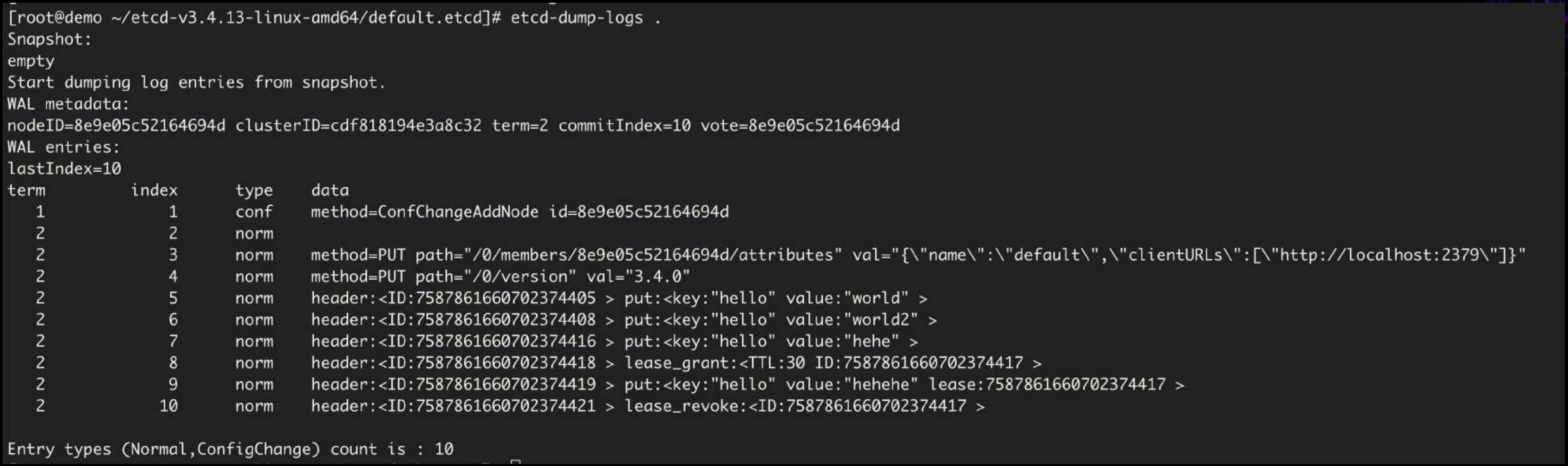
```
$ etcdctl put hello3 world3
OK
$ etcdctl txn -i
compares:
value("hello3") = "world3"

success requests (get, put, del):
put hello3 txn-succ

failure requests (get, put, del):
put hello3 txn-failed

SUCCESS

OK
$ etcdctl get hello3
hello3
txn-succ
```

## etcd 数据存储格式



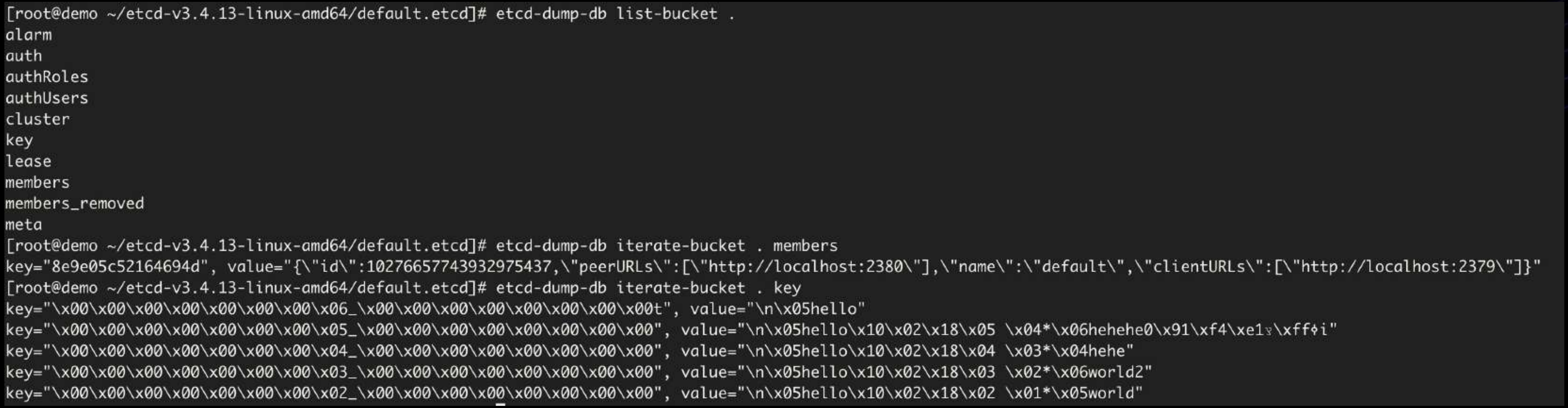
```
[root@demo ~/etcd-v3.4.13-linux-amd64/default.etcd]# tree .
.
└── member
    ├── snap
    │   └── db
    └── wal
        ├── 0000000000000000-0000000000000000.wal
        └── 0.tmp

3 directories, 3 files
```

## WAL 文件存储了什么数据？



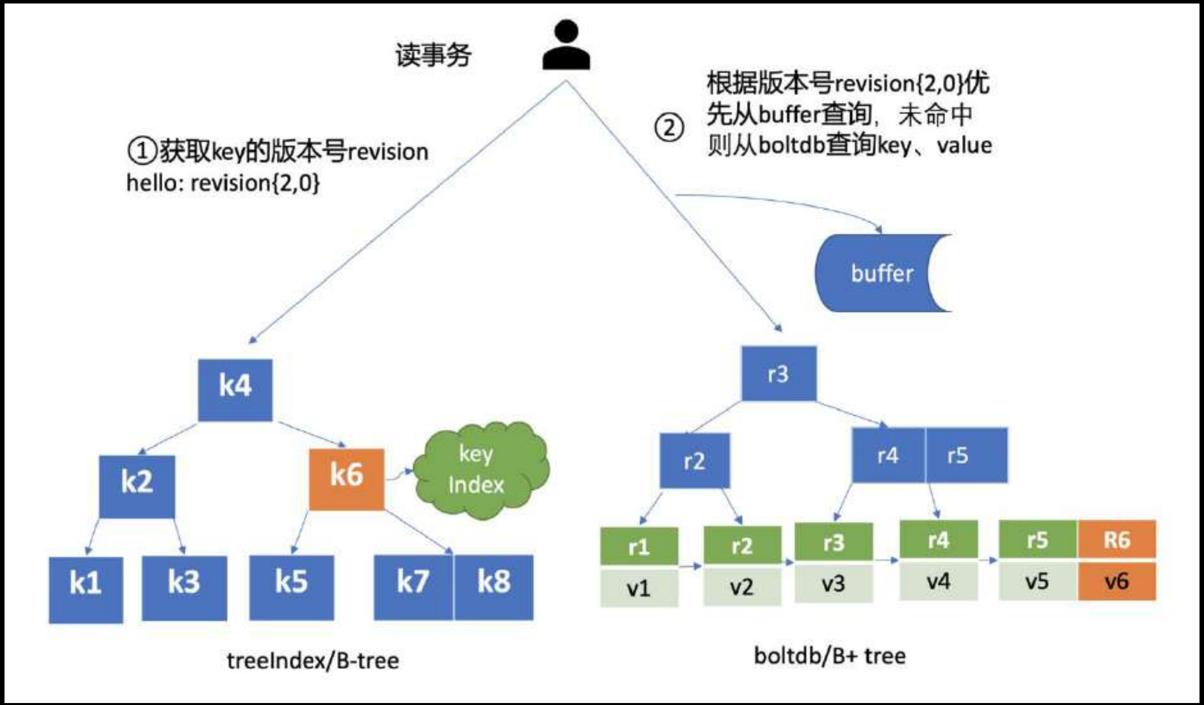
```
[root@demo ~/etcd-v3.4.13-linux-amd64/default.etcd]# etcd-dump-logs .
Snapshot:
empty
Start dumping log entries from snapshot.
WAL metadata:
nodeID=8e9e05c52164694d clusterID=cdf818194e3a8c32 term=2 commitIndex=10 vote=8e9e05c52164694d
WAL entries:
lastIndex=10
term         index      type    data
   1             1      conf    method=ConfChangeAddNode id=8e9e05c52164694d
   2             2      norm
   2             3      norm    method=PUT path="/0/members/8e9e05c52164694d/attributes" val="{\"name\":\"default\",\"clientURLs\":[\"http://localhost:2379\"]}"
   2             4      norm    method=PUT path="/0/version" val="3.4.0"
   2             5      norm    header:<ID:7587861660702374405 > put:<key:"hello" value:"world" >
   2             6      norm    header:<ID:7587861660702374408 > put:<key:"hello" value:"world2" >
   2             7      norm    header:<ID:7587861660702374416 > put:<key:"hello" value:"hehe" >
   2             8      norm    header:<ID:7587861660702374418 > lease_grant:<TTL:30 ID:7587861660702374417 >
   2             9      norm    header:<ID:7587861660702374419 > put:<key:"hello" value:"hehehe" lease:7587861660702374417 >
   2            10      norm    header:<ID:7587861660702374421 > lease_revoke:<ID:7587861660702374417 >

Entry types (Normal,ConfigChange) count is : 10
```

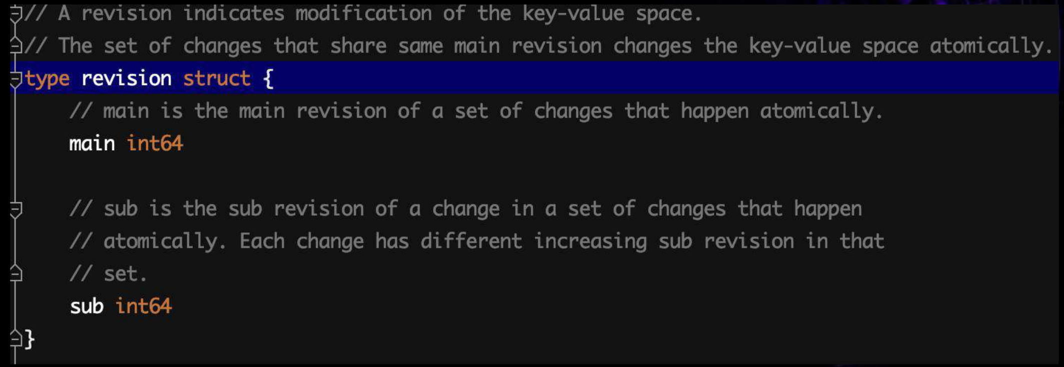
## DB 文件存储了什么数据？



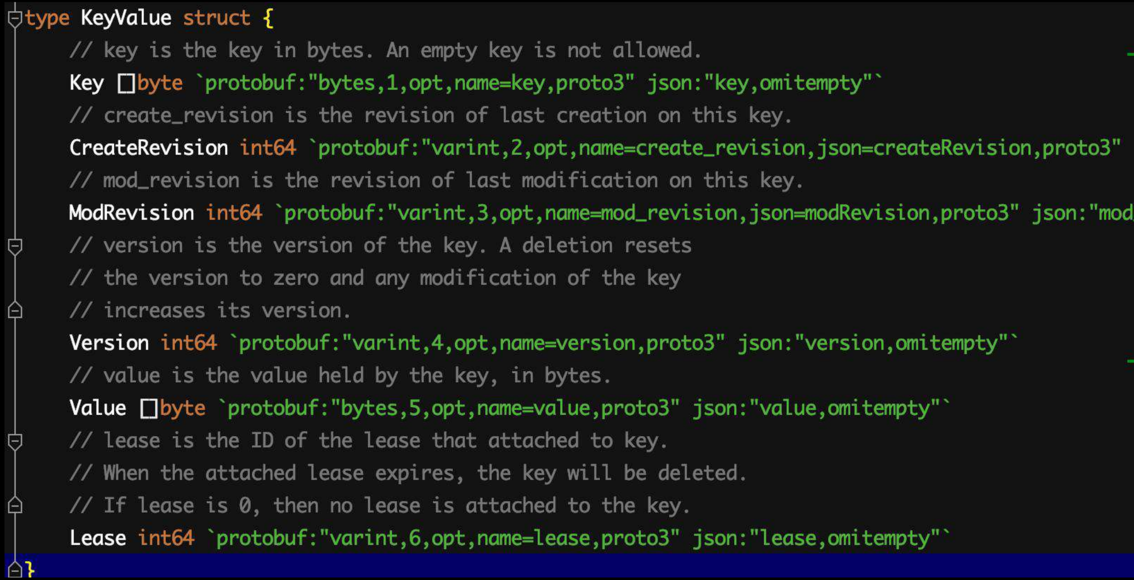
```
[root@demo ~/etcd-v3.4.13-linux-amd64/default.etcd]# etcd-dump-db list-bucket .
alarm
auth
authRoles
authUsers
cluster
key
lease
members
members_removed
meta
[root@demo ~/etcd-v3.4.13-linux-amd64/default.etcd]# etcd-dump-db iterate-bucket . members
key="8e9e05c52164694d", value="{\"id\":10276657743932975437,\"peerURLs\":[\"http://localhost:2380\"],\"name\":\"default\",\"clientURLs\":[\"http://localhost:2379\"]}"
[root@demo ~/etcd-v3.4.13-linux-amd64/default.etcd]# etcd-dump-db iterate-bucket . key
key="\x00\x00\x00\x00\x00\x00\x00\x06_\x00\x00\x00\x00\x00\x00\x00\x00t", value="\n\x05hello"
key="\x00\x00\x00\x00\x00\x00\x00\x05_\x00\x00\x00\x00\x00\x00\x00\x00", value="\n\x05hello\x10\x02\x18\x05 \x04*\x06hehehe0\x91\xf4\xe1ɤ\xff¢i"
key="\x00\x00\x00\x00\x00\x00\x00\x04_\x00\x00\x00\x00\x00\x00\x00\x00", value="\n\x05hello\x10\x02\x18\x04 \x03*\x04hehe"
key="\x00\x00\x00\x00\x00\x00\x00\x03_\x00\x00\x00\x00\x00\x00\x00\x00", value="\n\x05hello\x10\x02\x18\x03 \x02*\x06world2"
key="\x00\x00\x00\x00\x00\x00\x00\x02_\x00\x00\x00\x00\x00\x00\x00\x00", value="\n\x05hello\x10\x02\x18\x02 \x01*\x05world"
```

# etcd MVCC



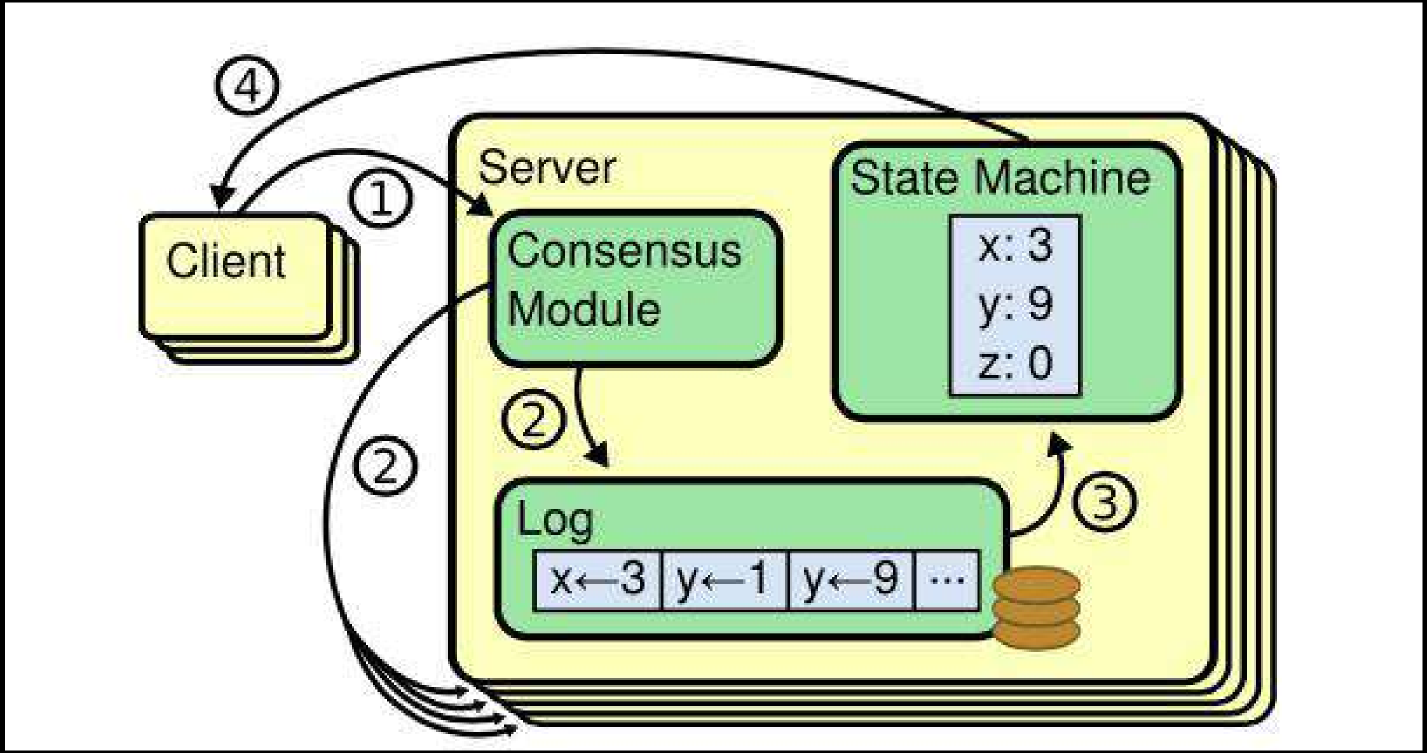


```go
// A revision indicates modification of the key-value space.
// The set of changes that share same main revision changes the key-value space atomically.
type revision struct {
    // main is the main revision of a set of changes that happen atomically.
    main int64

    // sub is the sub revision of a change in a set of changes that happen
    // atomically. Each change has different increasing sub revision in that
    // set.
    sub int64
}
```



```go
type KeyValue struct {
    // key is the key in bytes. An empty key is not allowed.
    Key []byte `protobuf:"bytes,1,opt,name=key,proto3" json:"key,omitempty"`
    // create_revision is the revision of last creation on this key.
    CreateRevision int64 `protobuf:"varint,2,opt,name=create_revision,json=createRevision,proto3"
    // mod_revision is the revision of last modification on this key.
    ModRevision int64 `protobuf:"varint,3,opt,name=mod_revision,json=modRevision,proto3" json:"mod
    // version is the version of the key. A deletion resets
    // the version to zero and any modification of the key
    // increases its version.
    Version int64 `protobuf:"varint,4,opt,name=version,proto3" json:"version,omitempty"`
    // value is the value held by the key, in bytes.
    Value []byte `protobuf:"bytes,5,opt,name=value,proto3" json:"value,omitempty"`
    // lease is the ID of the lease that attached to key.
    // When the attached lease expires, the key will be deleted.
    // If lease is 0, then no lease is attached to the key.
    Lease int64 `protobuf:"varint,6,opt,name=lease,proto3" json:"lease,omitempty"`
}
```

# etcd raft

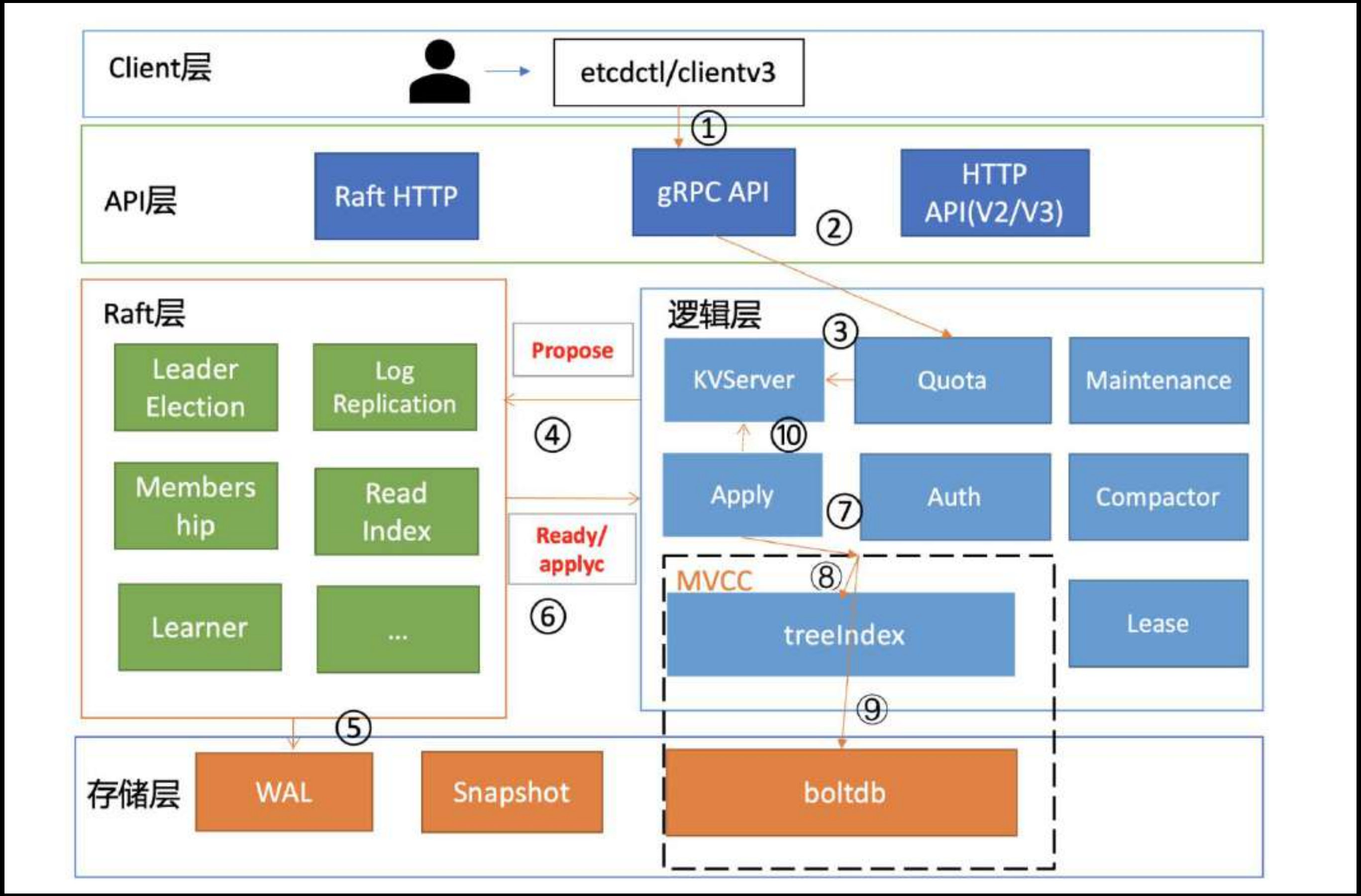


- Leader 选举，Leader 故障后集群能快速选出新 Leader；

- 日志复制， 集群只有 Leader 能写入日志， Leader 负责复制日志到 Follower 节点，并强制 Follower 节点与自己保持相同；

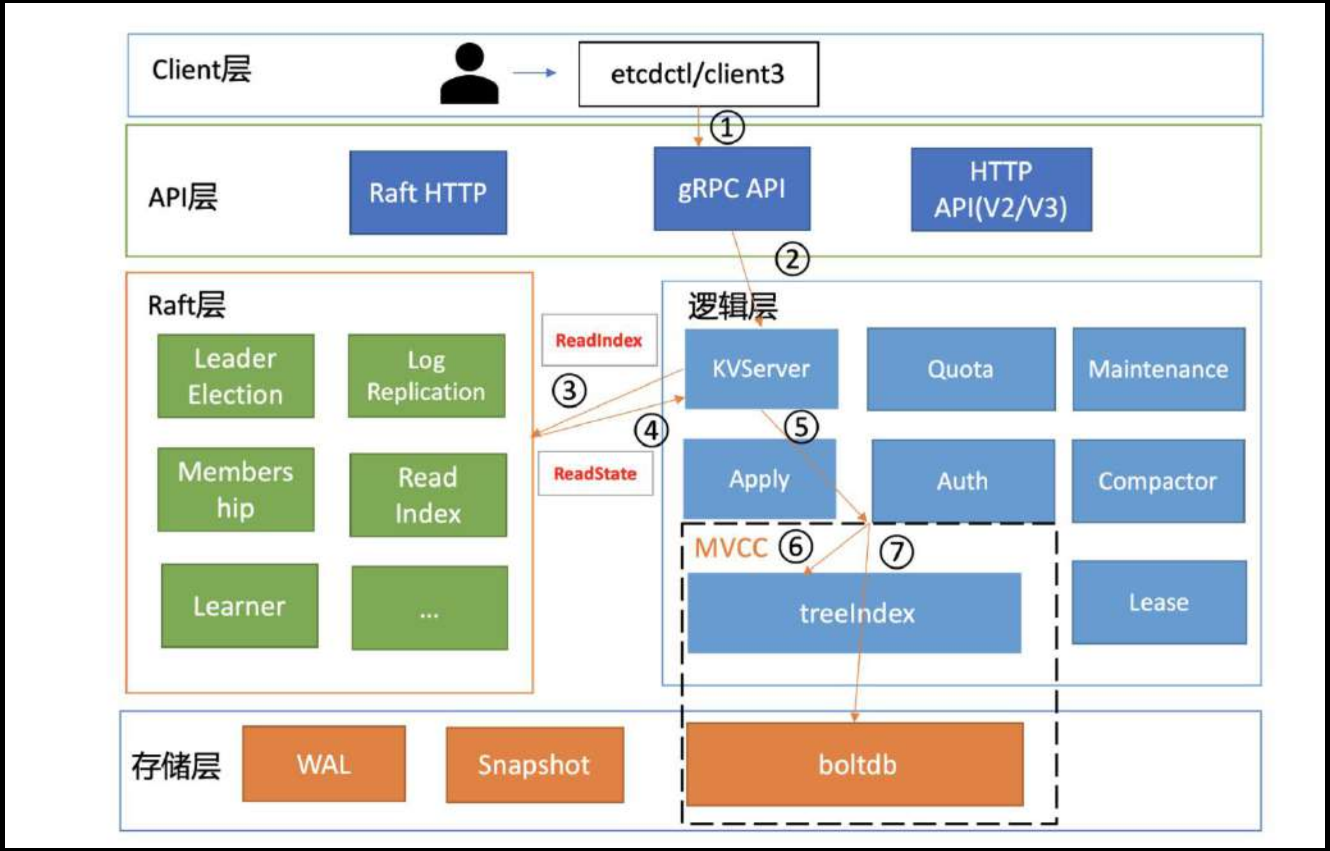
- 安全性，一个任期内集群只能产生一个 Leader、已提交的日志条目在发生 Leader 选举时，一定会存在更高任期的新 Leader 日志中、各个节点的状态机应用的任意位置的日志条目内容应一样等。

# etcd 读写原理-写请求流程



- clientv3库，put->gRPC KVServer Put API，负载均衡算法，round-robin
- etcd 任一节点的 etcd server模块收到Client 写请求（如果是 follower 节点，会先通过 Raft 模块将请求转发至 leader 节点处理）
- etcd server 将请求封装为Raft请求，然后提交给 Raft 模块处理
- Leader 通过 Raft 协议与集群中 follower 节点进行交互，将消息复制到 follower 节点，于此同时，并行的将日志持久化到 WAL
- Follower 节点对该请求进行响应，回复自己是否同意该请求
- 当集群中超过半数节点（(n/2)+1 members ）同意接收这条日志数据时，表示该请求可以被Commit，Raft 模块通知etcd server 该日志数据已经 Commit，可以进行 Apply
- 各个节点的 etcd server 的 applierV3 模块异步进行 Apply 操作，并通过 MVCC 模块写入后端存储 BoltDB
- 当 client 所连接的节点数据 apply 成功后，会返回给客户端 apply 的结果

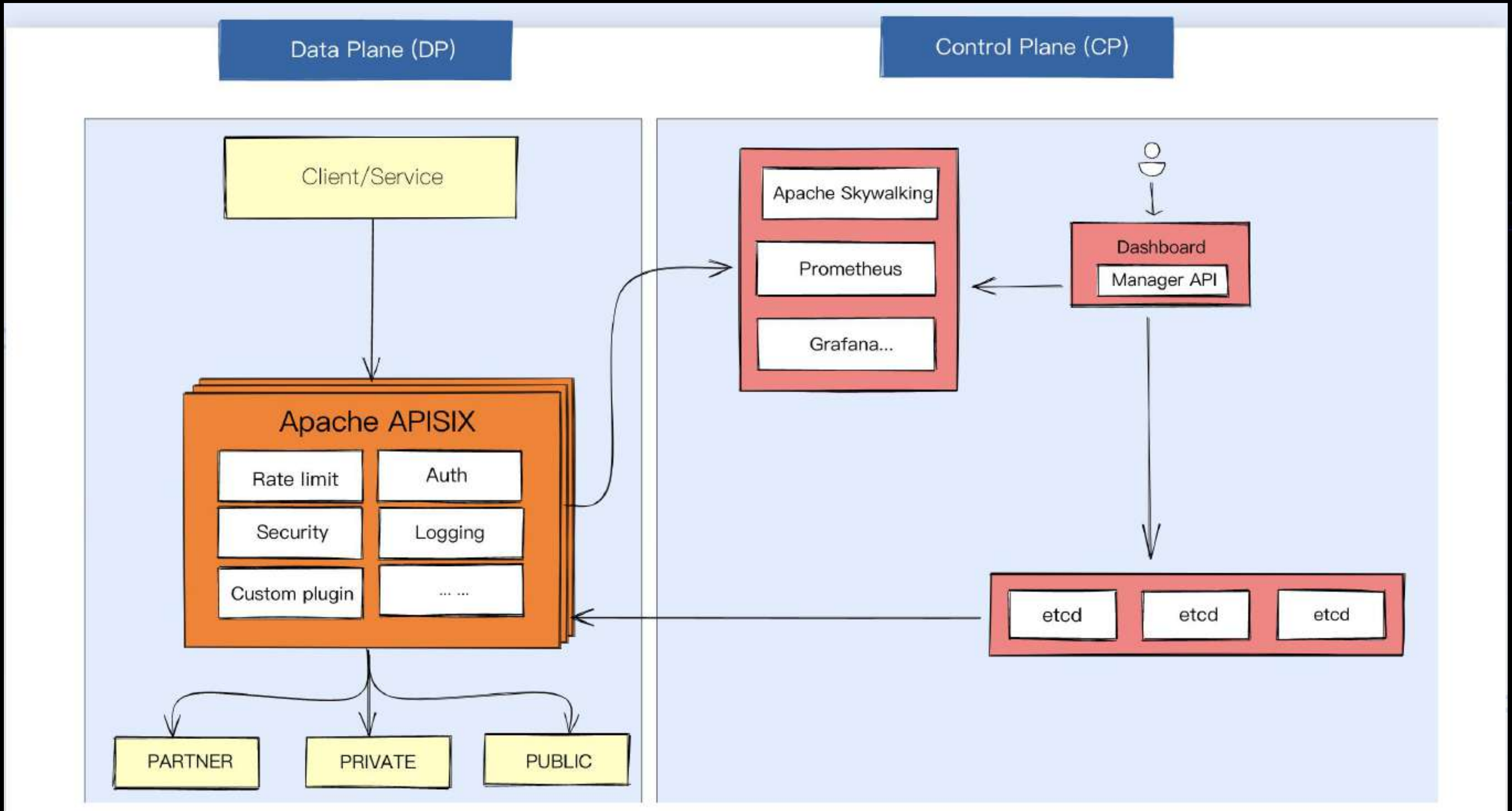
# etcd 读写原理-读请求流程



- clientv3库，Range->gRPC KVServer Range API，负载均衡算法，round-robin
- etcd 任一节点的 etcd server 模块收到客户端读请求（Range 请求）
- 判断读请求类型，如果是串行化读（serializable）则直接进入 Apply 流程
- 如果是线性一致性读（linearizable），则进入 Raft 模块
- Raft 模块向 leader 发出 ReadIndex 请求，获取当前集群已经提交的最新数据 Index
- 等待本地 AppliedIndex 大于或等于 ReadIndex 获取的 CommittedIndex 时，进入 Apply 流程
- Apply 流程，通过 Key 名从 KV Index 模块获取 Key 最新的 Revision，再通过 Revision 从BoltDB 获取对应的 Key 和 Value

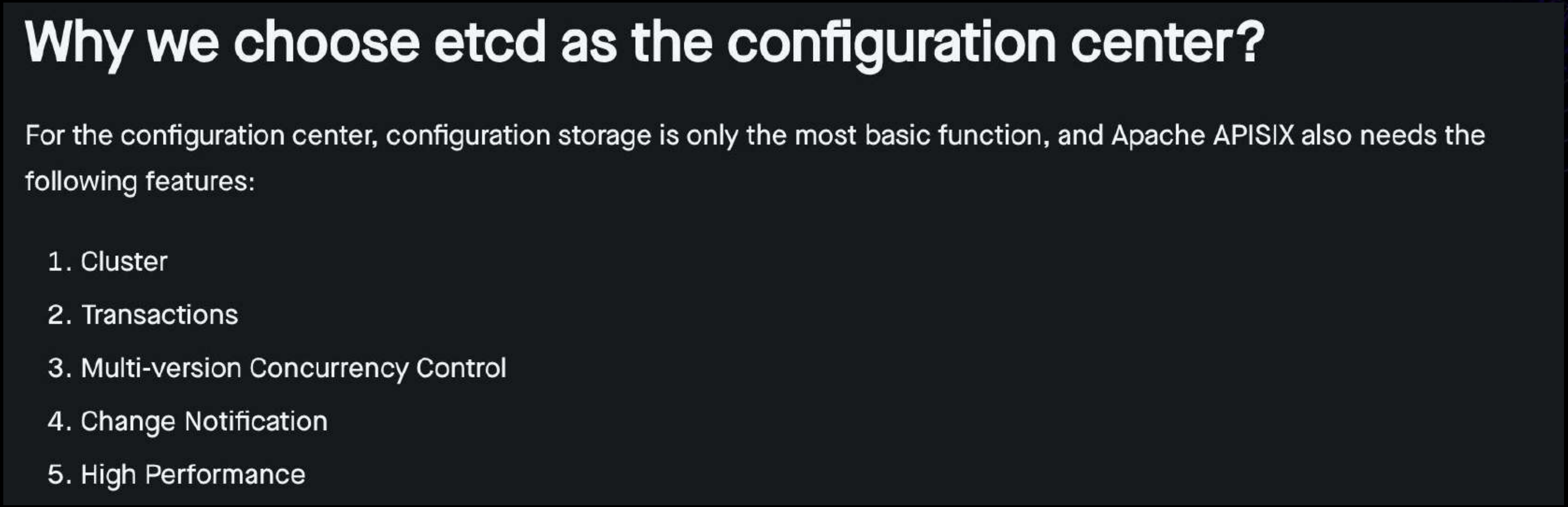
## 02 etcd APISIX

## APISIX 架构



Apache APISIX is based on Nginx and etcd. Compared with traditional API gateways, APISIX has dynamic routing and hot-loading plugins

## APISIX 为何选用 etcd?

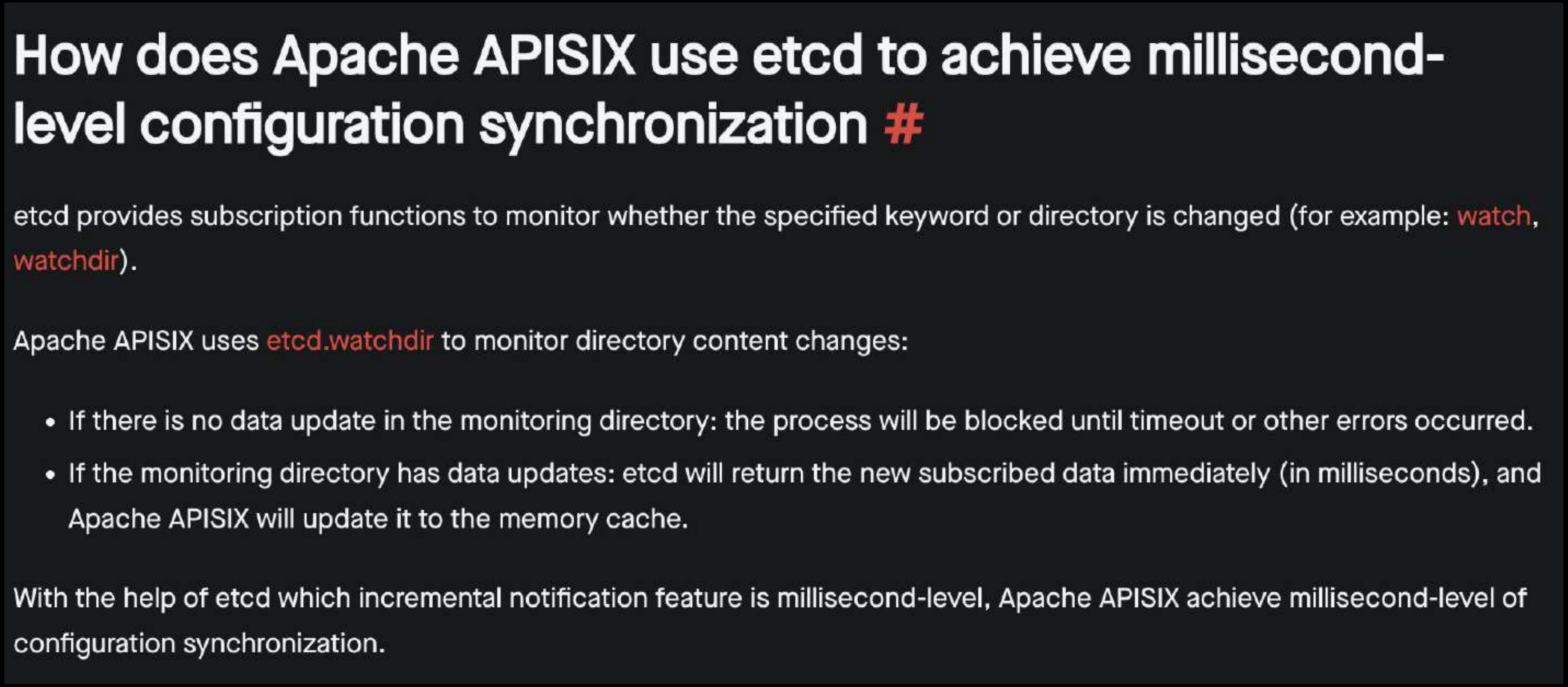


### Why we choose etcd as the configuration center?

For the configuration center, configuration storage is only the most basic function, and Apache APISIX also needs the following features:

1. Cluster
2. Transactions
3. Multi-version Concurrency Control
4. Change Notification
5. High Performance

## APISIX 为何选用 etcd？



### How does Apache APISIX use etcd to achieve millisecond-level configuration synchronization #

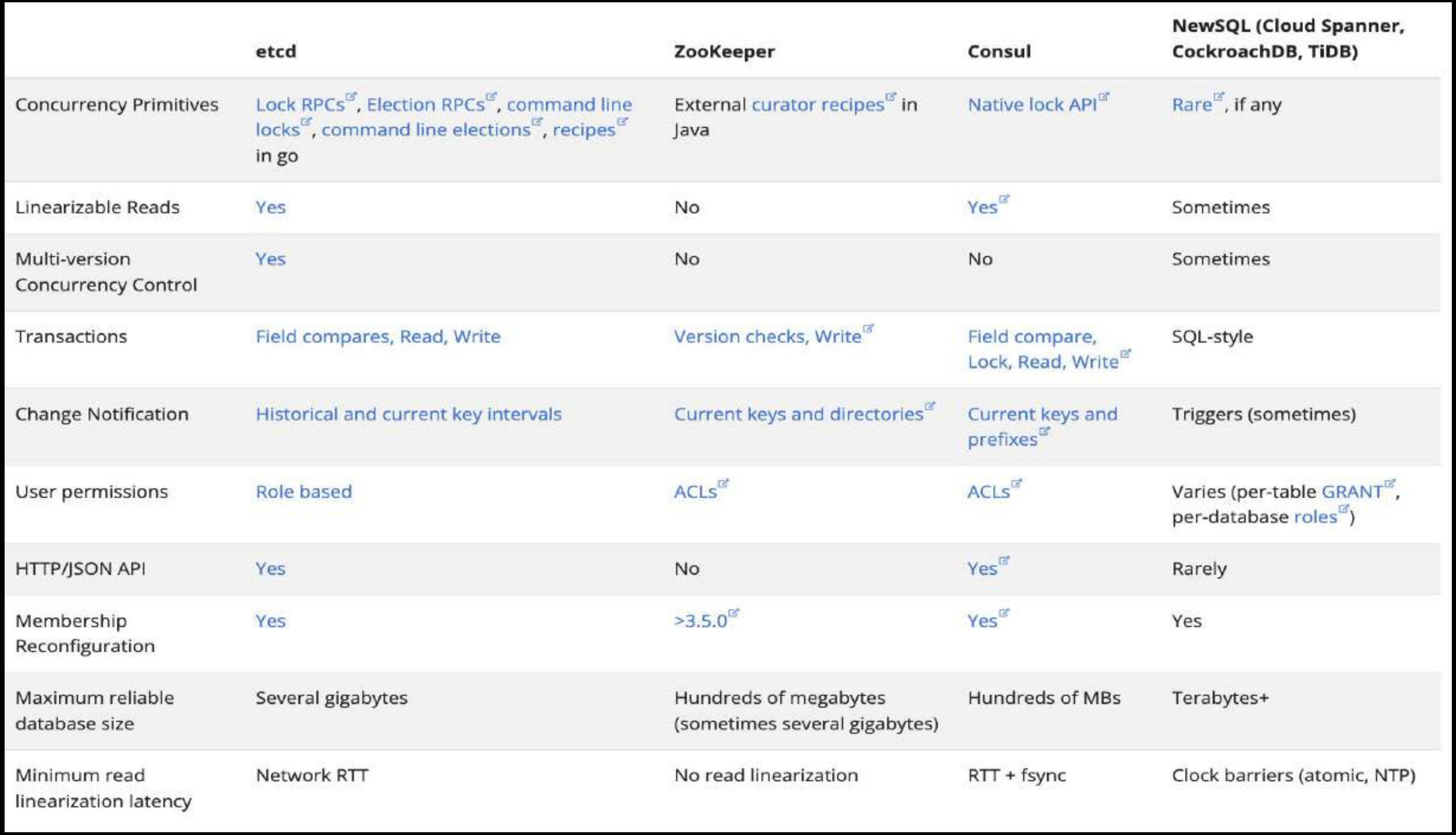
etcd provides subscription functions to monitor whether the specified keyword or directory is changed (for example: watch, watchdir).

Apache APISIX uses etcd.watchdir to monitor directory content changes:

- If there is no data update in the monitoring directory: the process will be blocked until timeout or other errors occurred.
- If the monitoring directory has data updates: etcd will return the new subscribed data immediately (in milliseconds), and Apache APISIX will update it to the memory cache.

With the help of etcd which incremental notification feature is millisecond-level, Apache APISIX achieve millisecond-level of configuration synchronization.

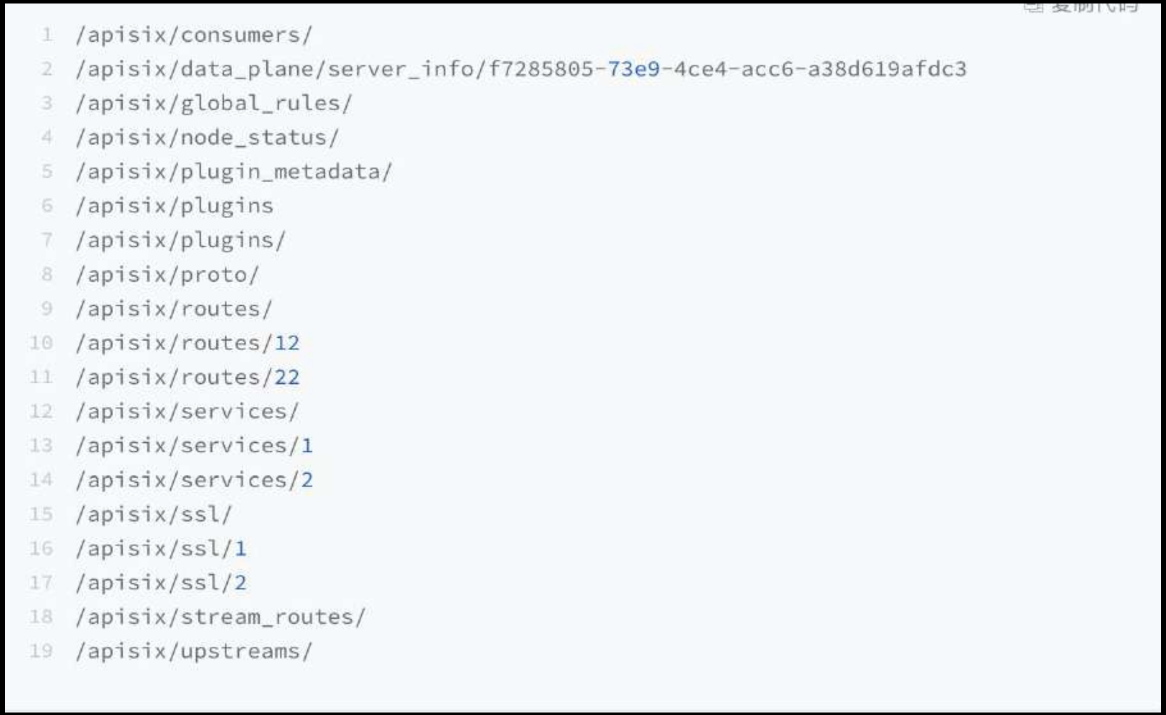
## APISIX 为何选用 etcd？



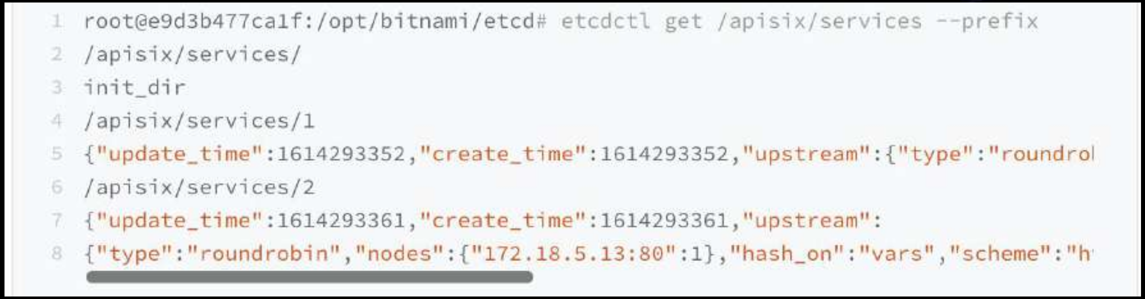
| | etcd | ZooKeeper | Consul | NewSQL (Cloud Spanner, CockroachDB, TiDB) |
|---|---|---|---|---|
| Concurrency Primitives | Lock RPCs, Election RPCs, command line locks, command line elections, recipes in go | External curator recipes in Java | Native lock API | Rare, if any |
| Linearizable Reads | Yes | No | Yes | Sometimes |
| Multi-version Concurrency Control | Yes | No | No | Sometimes |
| Transactions | Field compares, Read, Write | Version checks, Write | Field compare, Lock, Read, Write | SQL-style |
| Change Notification | Historical and current key intervals | Current keys and directories | Current keys and prefixes | Triggers (sometimes) |
| User permissions | Role based | ACLs | ACLs | Varies (per-table GRANT, per-database roles) |
| HTTP/JSON API | Yes | No | Yes | Rarely |
| Membership Reconfiguration | Yes | >3.5.0 | Yes | Yes |
| Maximum reliable database size | Several gigabytes | Hundreds of megabytes (sometimes several gigabytes) | Hundreds of MBs | Terabytes+ |
| Minimum read linearization latency | Network RTT | No read linearization | RTT + fsync | Clock barriers (atomic, NTP) |

[illegible]

## APISIX 如何使用 etcd-存储格式

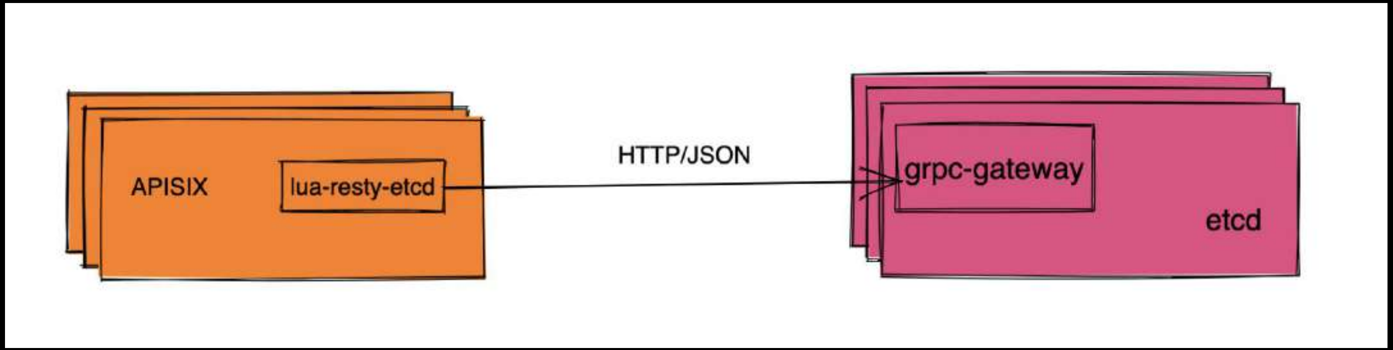


```
/apisix/consumers/
/apisix/data_plane/server_info/f7285805-73e9-4ce4-acc6-a38d619afdc3
/apisix/global_rules/
/apisix/node_status/
/apisix/plugin_metadata/
/apisix/plugins
/apisix/plugins/
/apisix/proto/
/apisix/routes/
/apisix/routes/12
/apisix/routes/22
/apisix/services/
/apisix/services/1
/apisix/services/2
/apisix/ssl/
/apisix/ssl/1
/apisix/ssl/2
/apisix/stream_routes/
/apisix/upstreams/
```



```
root@e9d3b477ca1f:/opt/bitnami/etcd# etcdctl get /apisix/services --prefix
/apisix/services/
init_dir
/apisix/services/1
{"update_time":1614293352,"create_time":1614293352,"upstream":{"type":"roundrol
/apisix/services/2
{"update_time":1614293361,"create_time":1614293361,"upstream":
{"type":"roundrobin","nodes":{"172.18.5.13:80":1},"hash_on":"vars","scheme":"h
```

## APISIX 如何使用 etcd-调用方式

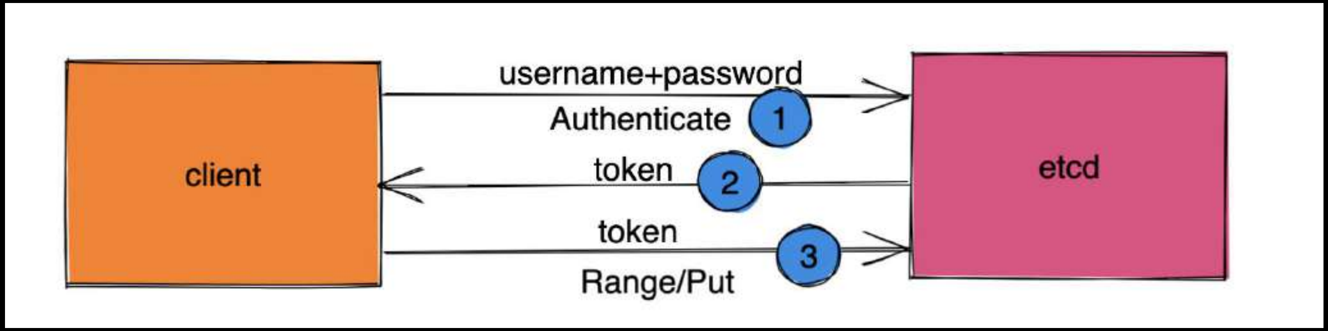


优点：
HTTP/JSON 调用，使用比较简单

缺点：
grpc-gateway 当前不支持使用证书鉴权

## APISIX 使用 etcd 过程中遇到了哪些坑？

—鉴权问题



为什么开启鉴权后我的请求这么慢？

- Authenticate接口会占用etcd大量cpu
- 低版本etcd在进行Authenticate时临界区较大，导致请求阻塞
- 未缓存token频繁发起Authenticate会导致etcd cpu负载飙高，请求阻塞

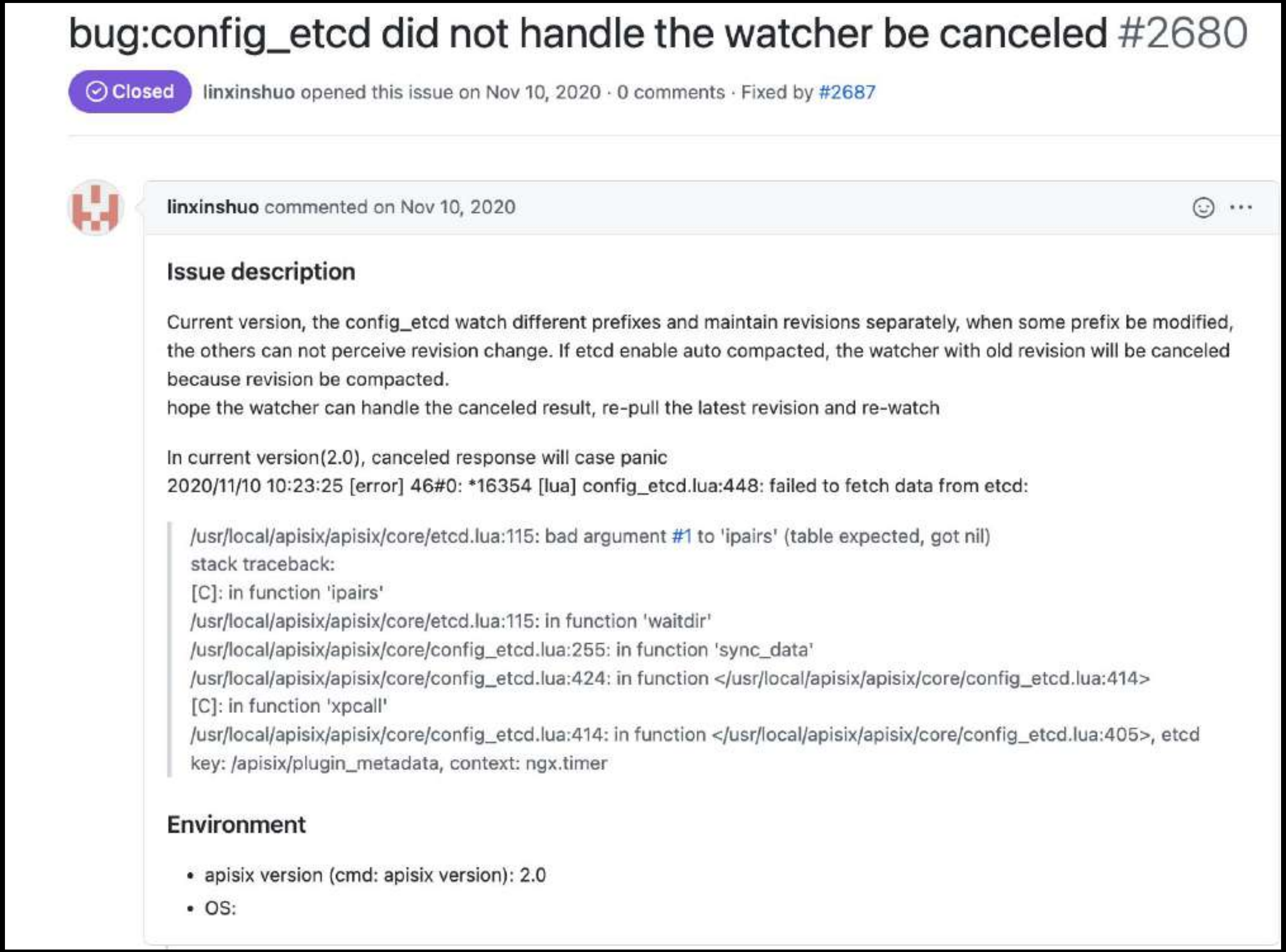
相关issue及PR：
https://github.com/apache/apisix/issues/2899
https://github.com/apache/apisix/pull/2932
https://github.com/api7/lua-resty-etcd/pull/100
https://github.com/etcd-io/etcd/pull/11735

## 如何正确使用鉴权？

- etcd 开启鉴权需要显示的调用 AuthEnable API，开启前需要新建 root 用户及密码
- etcd v2启用鉴权后 guest 默认具备所有权限，需显式吊销 guest 权限
- etcd V2 和 V3 鉴权相互独立，开启鉴权时需全部开启/禁用 v2 api
- 生产环境尽量使用 https，并开启客户端证书认证
- 密码鉴权性能较差，能用证书鉴权最好用证书鉴权，如使用密码鉴权，etcd token 尽量使用 jwt，客户端尽量复用连接和 token
- gprc-proxy 和 grpc-gateway 不支持使用证书的 CN 鉴权

## APISIX 使用 etcd 过程中遇到了哪些坑？

—Watch 问题



bug:config_etcd did not handle the watcher be canceled #2680

Closed linxinshuo opened this issue on Nov 10, 2020 · 0 comments · Fixed by #2687

linxinshuo commented on Nov 10, 2020

**Issue description**

Current version, the config_etcd watch different prefixes and maintain revisions separately, when some prefix be modified, the others can not perceive revision change. If etcd enable auto compacted, the watcher with old revision will be canceled because revision be compacted.
hope the watcher can handle the canceled result, re-pull the latest revision and re-watch

In current version(2.0), canceled response will case panic
2020/11/10 10:23:25 [error] 46#0: *16354 [lua] config_etcd.lua:448: failed to fetch data from etcd:

```
/usr/local/apisix/apisix/core/etcd.lua:115: bad argument #1 to 'ipairs' (table expected, got nil)
stack traceback:
[C]: in function 'ipairs'
/usr/local/apisix/apisix/core/etcd.lua:115: in function 'waitdir'
/usr/local/apisix/apisix/core/config_etcd.lua:255: in function 'sync_data'
/usr/local/apisix/apisix/core/config_etcd.lua:424: in function </usr/local/apisix/apisix/core/config_etcd.lua:414>
[C]: in function 'xpcall'
/usr/local/apisix/apisix/core/config_etcd.lua:414: in function </usr/local/apisix/apisix/core/config_etcd.lua:405>, etcd
key: /apisix/plugin_metadata, context: ngx.timer
```

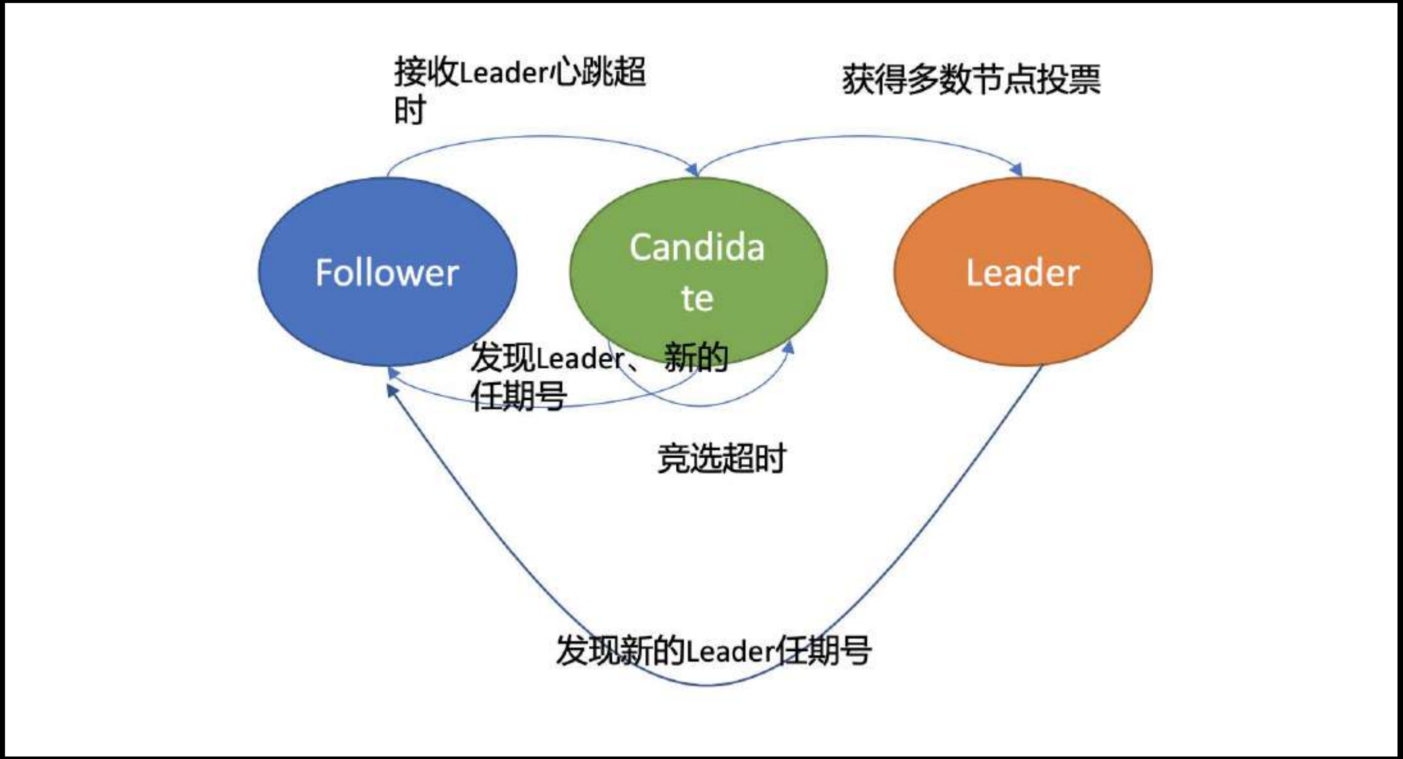
**Environment**

- apisix version (cmd: apisix version): 2.0
- OS:

未正确处理 ErrCompacted 错误导致Watch异常

## 03 etcd 使用常见问题

## 为什么我的集群频繁 leader 选举？



- 磁盘 IO 性能差
- 网络质量差
- Cpu/内存高负载
- 心跳超时参数过短
- Snapshot 参数过小/不一致

## 为什么我的 db 被写满了？

请求 etcd 出现报错：etcdserver: mvcc: database space exceeded

- etcd dbsize 默认 2G，未开启自动压缩且更较频繁的情况下很容易导致 db 被写满，从而触发 etcd 的写保护
- 生产环境最好启用 auto compaction
- DB 被写满后，可根据实际情况进行 Compact，调大 dbsize 参数/对 db 进行 defrag 操作，调整之后需进行 etcdctl alarm disarm 来清除告警
- 注意，Compact 操作会压缩和清理历史版本，但并不能压缩 db 大小，压缩 db 大小需使用 defrag（有损）

## 我能定期对 etcd 进行 defrag 操作吗？

如非必要，最好不要：

Defrag操作相当于将当前的boltdb数据遍历后写到一个新的db文件，然后replace掉老的db文件，期间会对整个db文件加锁，阻塞该节点的读写操作，造成访问该节点的请求受到影响

## Lease

现网真实案例：

某业务使用etcd作为服务发现，客户端每隔心跳间隔会为对应的key绑定一个新的lease，最终造成lease大量堆积，集群不可用

原因分析：

这是一个典型的错误使用场景，etcd3 的 lease 作为一个独立的对象存在，每次更新 key 如果都新生成一个新的 lease，并且客户端没有对老的 lease 进行吊销的话，那么老的 lease 会持续存在，直到过期后被 etcd 吊销。但是 etcd 服务端对 lease 的过期吊销是有频率限制的（当前限制时 1000/s），如果客户端生成 ease 的频率大于 etcd 过期吊销的频率，就会造成 lease 大量堆积（上百万级别），不仅会造成 etcd 高负载，同时会导致正常的 lease 过期后也无法被吊销，影响使用。

这里正确的使用方式时每个客户端使用一个 lease，定期使用 KeepAlive 进行续期

## 使用 etcd 需要注意哪些容量限制？

- dbsize: 默认2G，建议不超过8G，db过大会影响性能，同时影响启动和恢复速度
- lease数量：建议不超过1w，同时需注意lease TTL和etcd吊销限频的影响，如TTL为3s，限频1000/s，则lease超过3000就有可能导致部分lease吊销慢
- Key-value大小: 默认1.5MB，最大建议不超过10MB，过大会影响性能，同时导致内存较高
- 最大Txn ops：默认128，过大会影响性能
- SnapshotCount：3.4默认100000，早期版本默认10000，每个节点需保持一致，可根据kv大小、写入频率、磁盘性能等进行调整
- 机器带宽：防止大流量导致机器带宽被打满

## etcd-gateway，grpc-proxy，grpc-gateway

## ——傻傻分不清楚

- etcd-gateway：四层透明代理，用于屏蔽后端 etcd 地址，避免地址变动造成客户端访问地址需要变更
- grpc-proxy：无状态的 7 层反向代理，可横向扩展，聚合 watch 和 lease 请求，支持序列化读请求，对数据一致性要求不敏感的场景可用于提升性能
- grpc-gateway：etcd 内置，默认启用，可将 http 请求转换为 grpc 请求，使 etcdv3 支持 http 请求

## 灾难恢复：
## Snapshot restore OR force-new-cluster?

Q：5个节点挂掉3个，该使用什么方式恢复呢？

A：建议使用snapshot进行恢复，force-new-cluster在有节点存活的情况下可能会导致集群panic。

常见问题和误区：

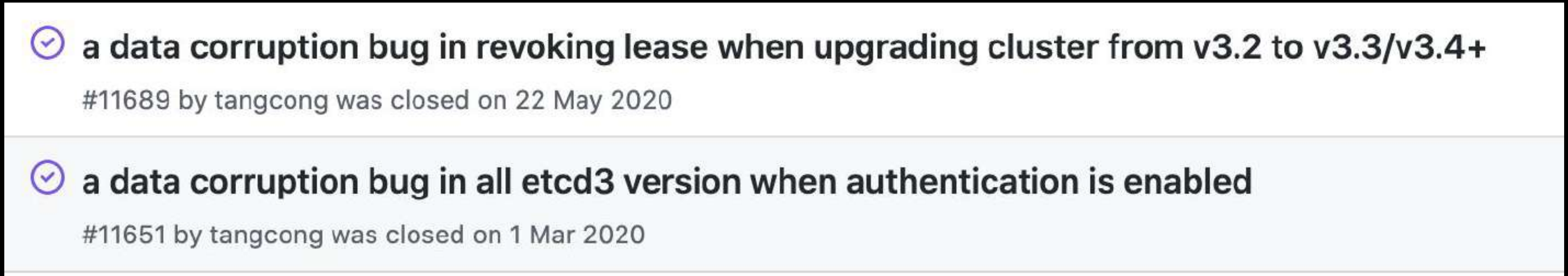
Q：如果我snapshot是在挂掉前一个小时进行的，那近一个小时的数据不是都会丢吗？

A：Snapshot API不走raft协议，只要有节点存活即可进行，即使挂掉一般以上的节点，也可以针对剩下的节点做实时做一个Snapshot，然后通过该Snapshot恢复，可保证数据尽量最全。

Q：我使用snapshot restore进行恢复，为什么集群起不来/恢复的有问题呢？

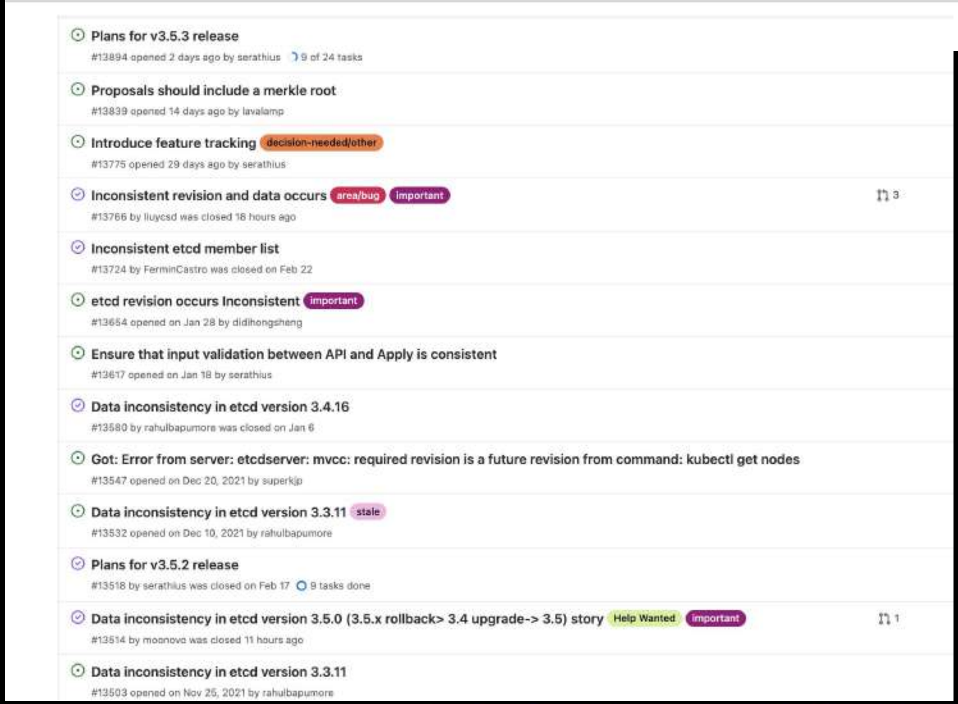
A：执行snapshot restore命令时，不同节点使用的命令行参数不同，需根据节点适配，请严格按照官方文档示例进行操作：https://etcd.io/docs/v3.5/op-guide/recovery/

# 强一致存储为什么会出现数据不一致？



a data corruption bug in revoking lease when upgrading cluster from v3.2 to v3.3/v3.4+
#11689 by tangcong was closed on 22 May 2020

a data corruption bug in all etcd3 version when authentication is enabled
#11651 by tangcong was closed on 1 Mar 2020



Plans for v3.5.3 release
#13894 opened 2 days ago by serathius 9 of 24 tasks

Proposals should include a merkle root
#13839 opened 14 days ago by lavalamp

Introduce feature tracking decision-needed/other
#13775 opened 29 days ago by serathius

Inconsistent revision and data occurs area/bug important 3
#13766 by liuycsd was closed 18 hours ago

Inconsistent etcd member list
#13724 by FerminCastro was closed on Feb 22

etcd revision occurs Inconsistent important
#13654 opened on Jan 28 by didihongsheng

Ensure that input validation between API and Apply is consistent
#13617 opened on Jan 18 by serathius

Data inconsistency in etcd version 3.4.16
#13580 by rahulbapumore was closed on Jan 6

Got: Error from server: etcdserver: mvcc: required revision is a future revision from command: kubectl get nodes
#13547 opened on Dec 20, 2021 by superkjp

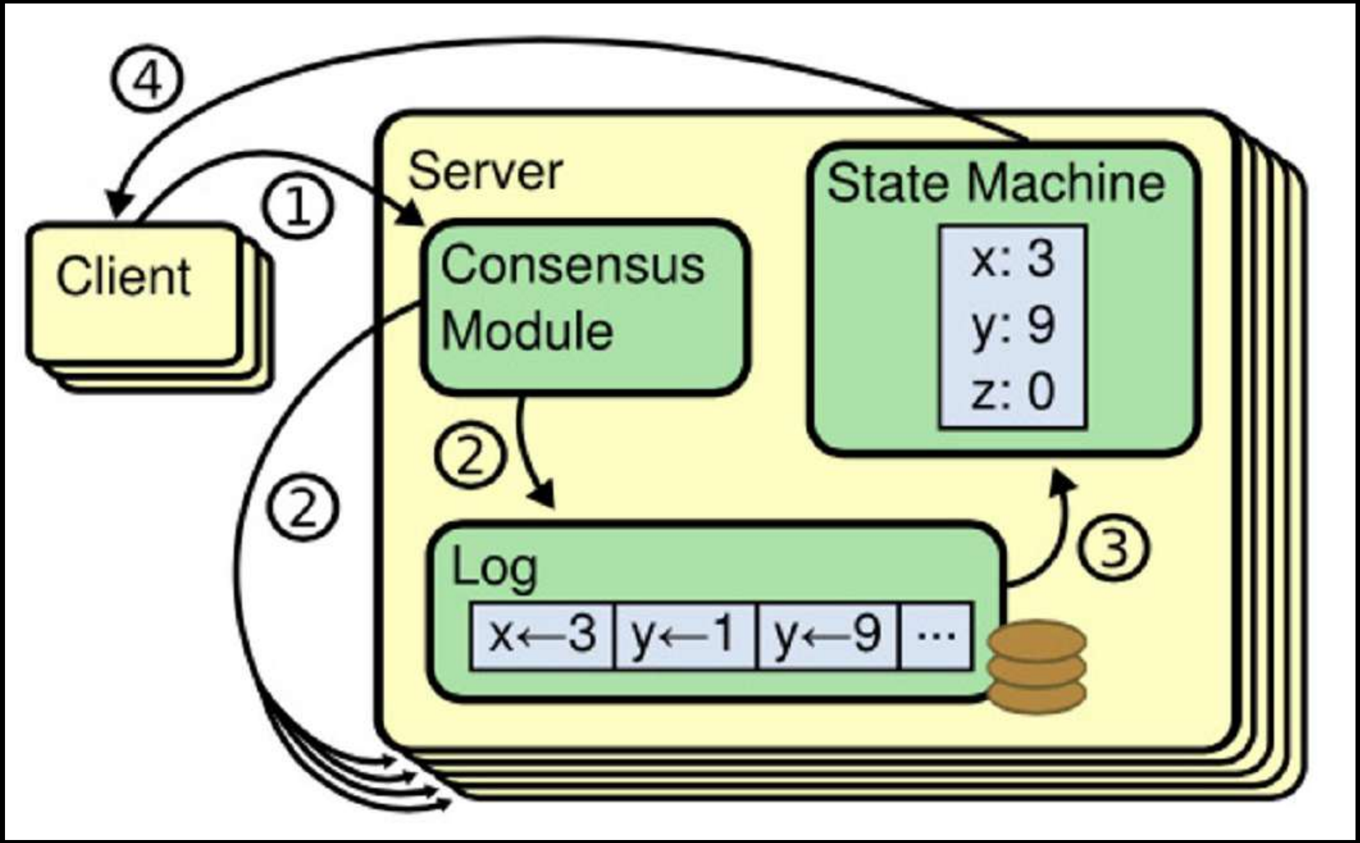
Data inconsistency in etcd version 3.3.11 stale
#13532 opened on Dec 10, 2021 by rahulbapumore

Plans for v3.5.2 release
#13518 by serathius was closed on Feb 17 9 tasks done

Data inconsistency in etcd version 3.5.0 (3.5.x rollback> 3.4 upgrade-> 3.5) story Help Wanted important 1
#13514 by moonovo was closed 11 hours ago

Data inconsistency in etcd version 3.3.11
#13503 opened on Nov 25, 2021 by rahulbapumore

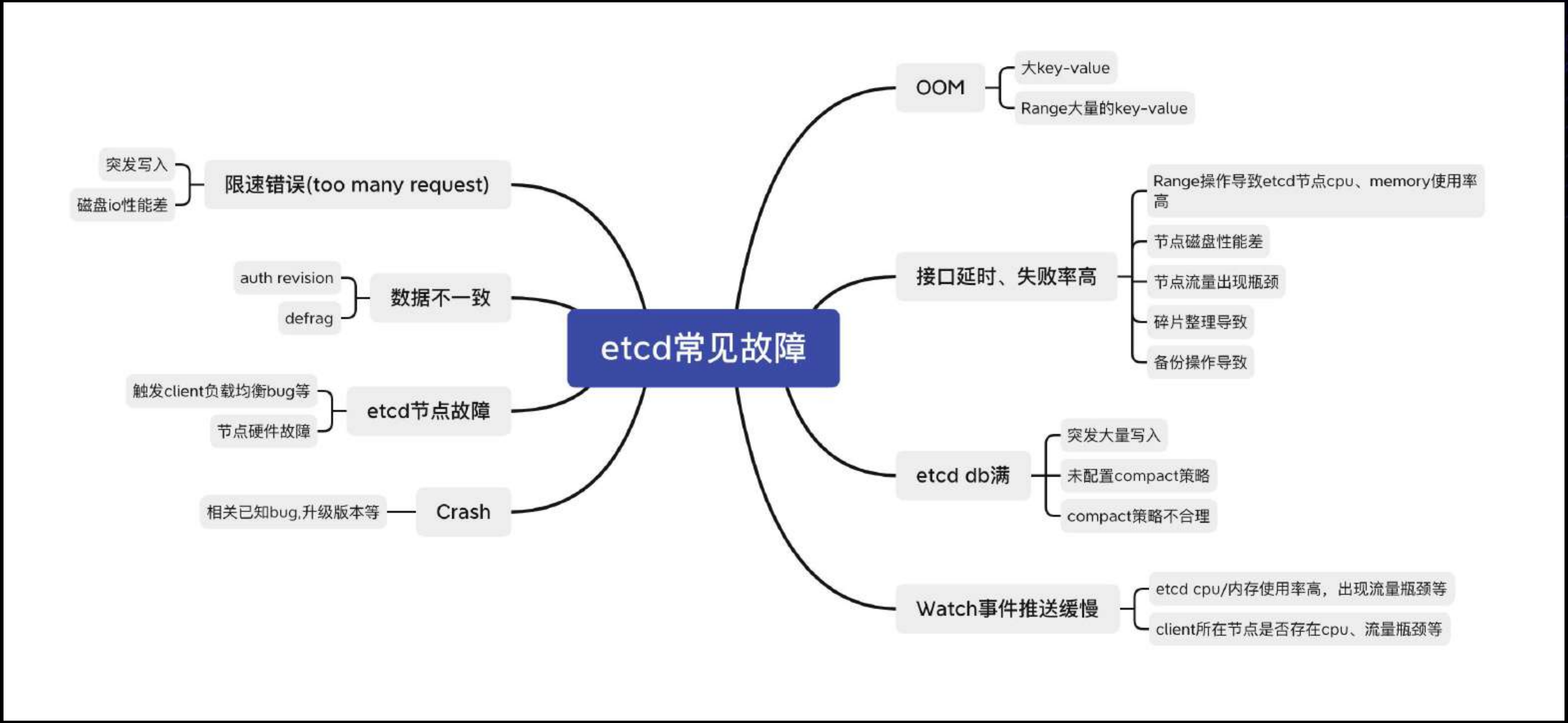
## 数据不一致本质原因



对于流程三来说，server 从日志里面获取已提交的日志条目，将其应用到状态机的过程，跟 Raft 算法本身无关，属于 server 本身的数据存储逻辑。

也就是说有可能存在 server 应用日志条目到状态机失败，进而导致各个节点出现数据不一致。但是这个不一致并非 Raft 模块导致的，它已超过 Raft 模块的功能界限。

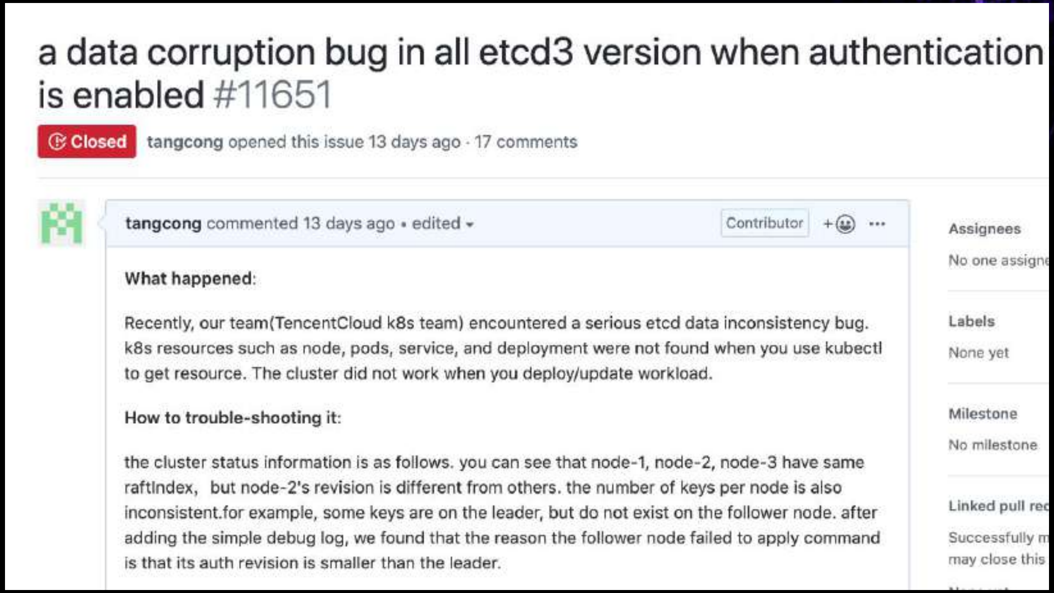
## etcd 常见问题总结


etcd常见故障
限速错误(too many request)
突发写入
磁盘io性能差
数据不一致
auth revision
defrag
etcd节点故障
触发client负载均衡bug等
节点硬件故障
Crash
相关已知bug,升级版本等
OOM
大key-value
Range大量的key-value
接口延时、失败率高
Range操作导致etcd节点cpu、memory使用率高
节点磁盘性能差
节点流量出现瓶颈
碎片整理导致
备份操作导致
etcd db满
突发大量写入
未配置compact策略
compact策略不合理
Watch事件推送缓慢
etcd cpu/内存使用率高，出现流量瓶颈等
client所在节点是否存在cpu、流量瓶颈等

## 04 大规模场景下的 etcd 治理—Kstone

## 大规模 etcd 集群治理挑战-内核

动持原


a data corruption bug in all etcd3 version when authentication is enabled #11651

Closed tangcong opened this issue 13 days ago · 17 comments

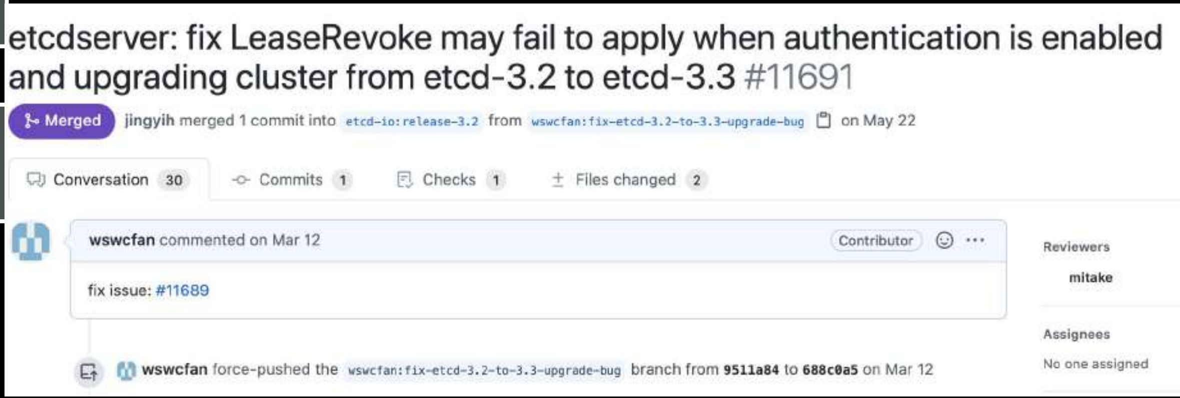
tangcong commented 13 days ago • edited Contributor

**What happened:**

Recently, our team(TencentCloud k8s team) encountered a serious etcd data inconsistency bug. k8s resources such as node, pods, service, and deployment were not found when you use kubectl to get resource. The cluster did not work when you deploy/update workload.

**How to trouble-shooting it:**

the cluster status information is as follows. you can see that node-1, node-2, node-3 have same raftIndex, but node-2's revision is different from others. the number of keys per node is also inconsistent.for example, some keys are on the leader, but do not exist on the follower node. after adding the simple debug log, we found that the reason the follower node failed to apply command is that its auth revision is smaller than the leader.


etcdserver: fix LeaseRevoke may fail to apply when authentication is enabled and upgrading cluster from etcd-3.2 to etcd-3.3 #11691

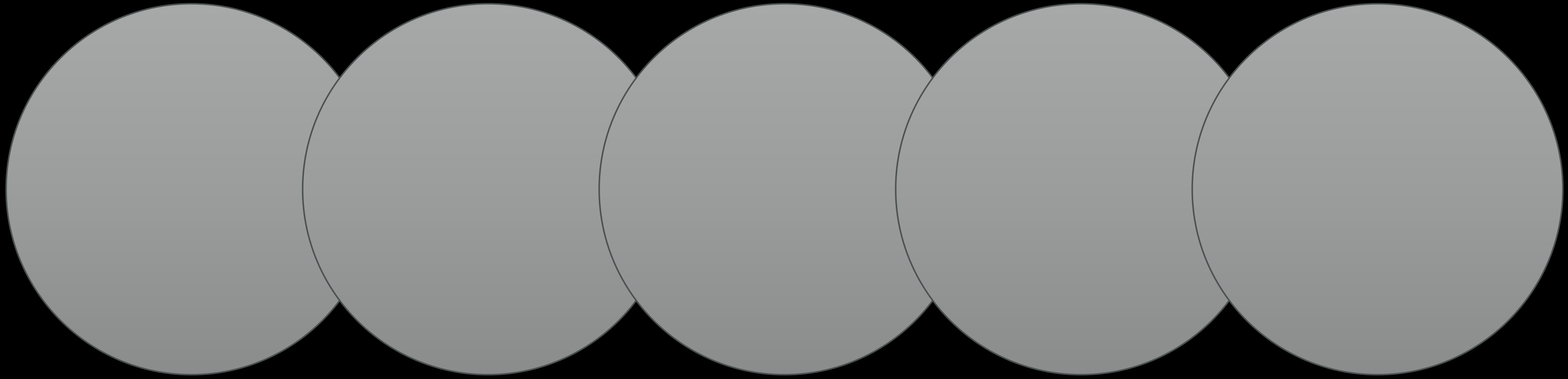
Merged jingyih merged 1 commit into etcd-io:release-3.2 from wswcfan:fix-etcd-3.2-to-3.3-upgrade-bug on May 22

wswcfan commented on Mar 12 Contributor

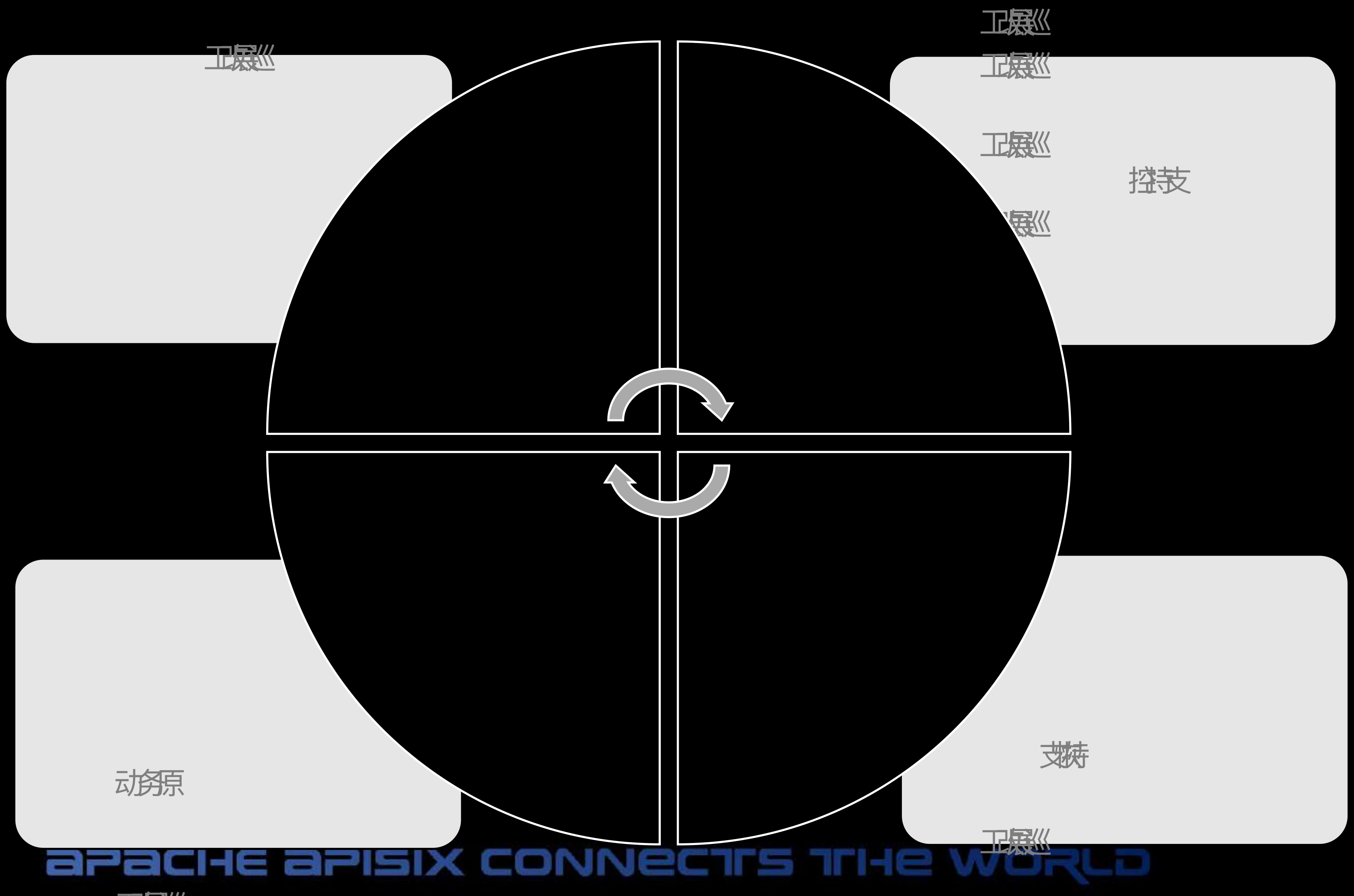
fix issue: #11689

wswcfan force-pushed the wswcfan:fix-etcd-3.2-to-3.3-upgrade-bug branch from 9511a84 to 688c0a5 on Mar 12

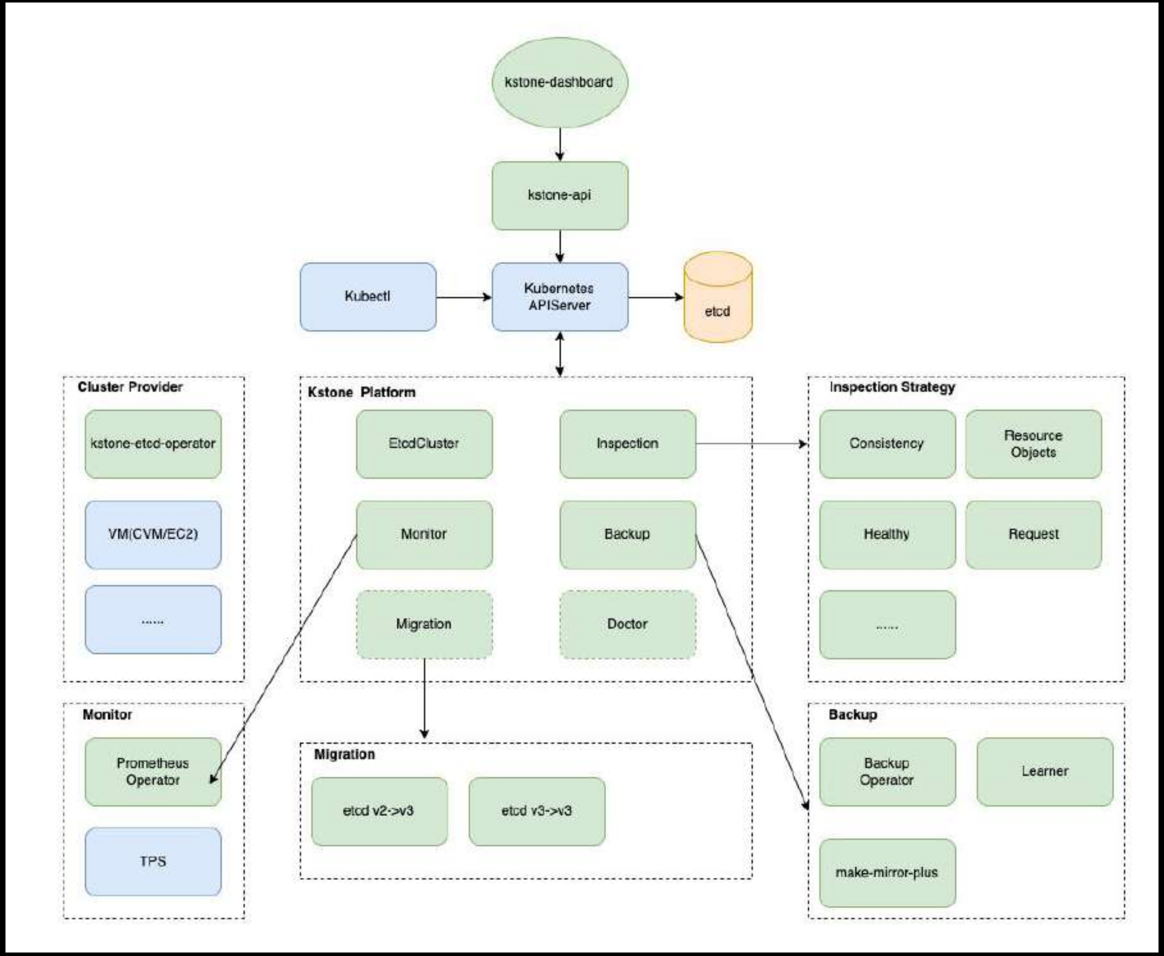
## 大规模 etcd 集群治理挑战-运维



## etcd -



# Kstone

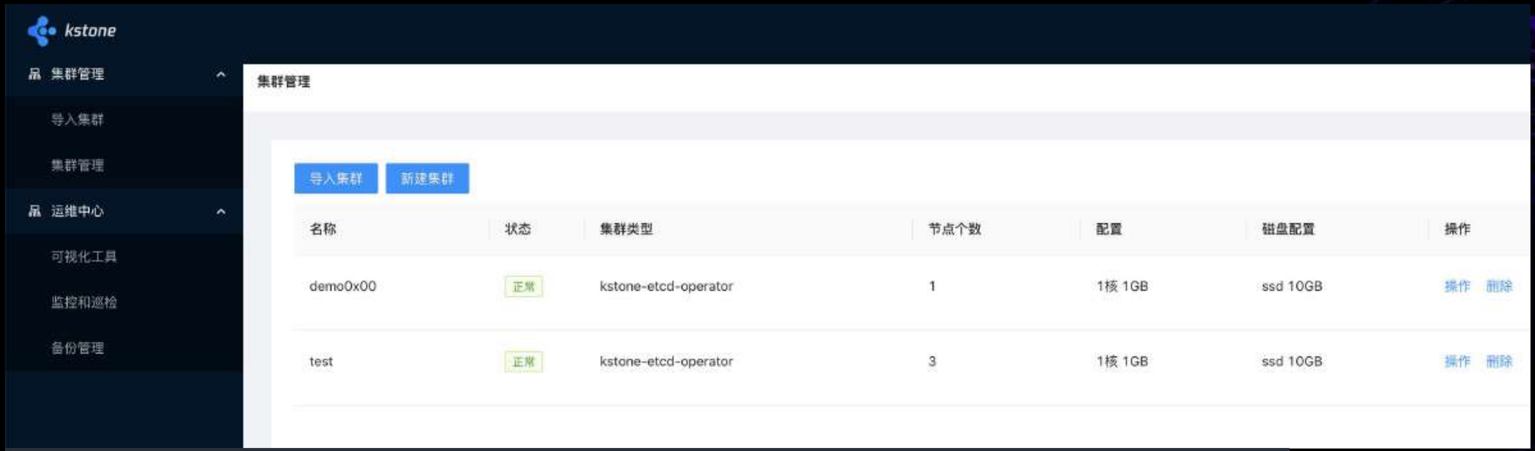


- 支持导入已有集群、创建新 etcd 集群
- 支持 Prometheu s 监控，内置丰富的 etcd Grafana 面板图
- 支持多种数据备份方式(分钟级备份到对象存储、部署 Learner 实时备份)
- 支持多种巡检策略（数据一致性、健康度、写请求、资源对象数等)
- 支持可视化查看 etcd 数据（特别支持 Kubernetes 资源对象数据的可视化查看）
- 极大简化运维成本，在集群导入、创建完成时，即可自动开启监控、备份、巡检等特性

[illegible]

APACHE APISIX CONNECTS THE WORLD

## 可视化集群管理



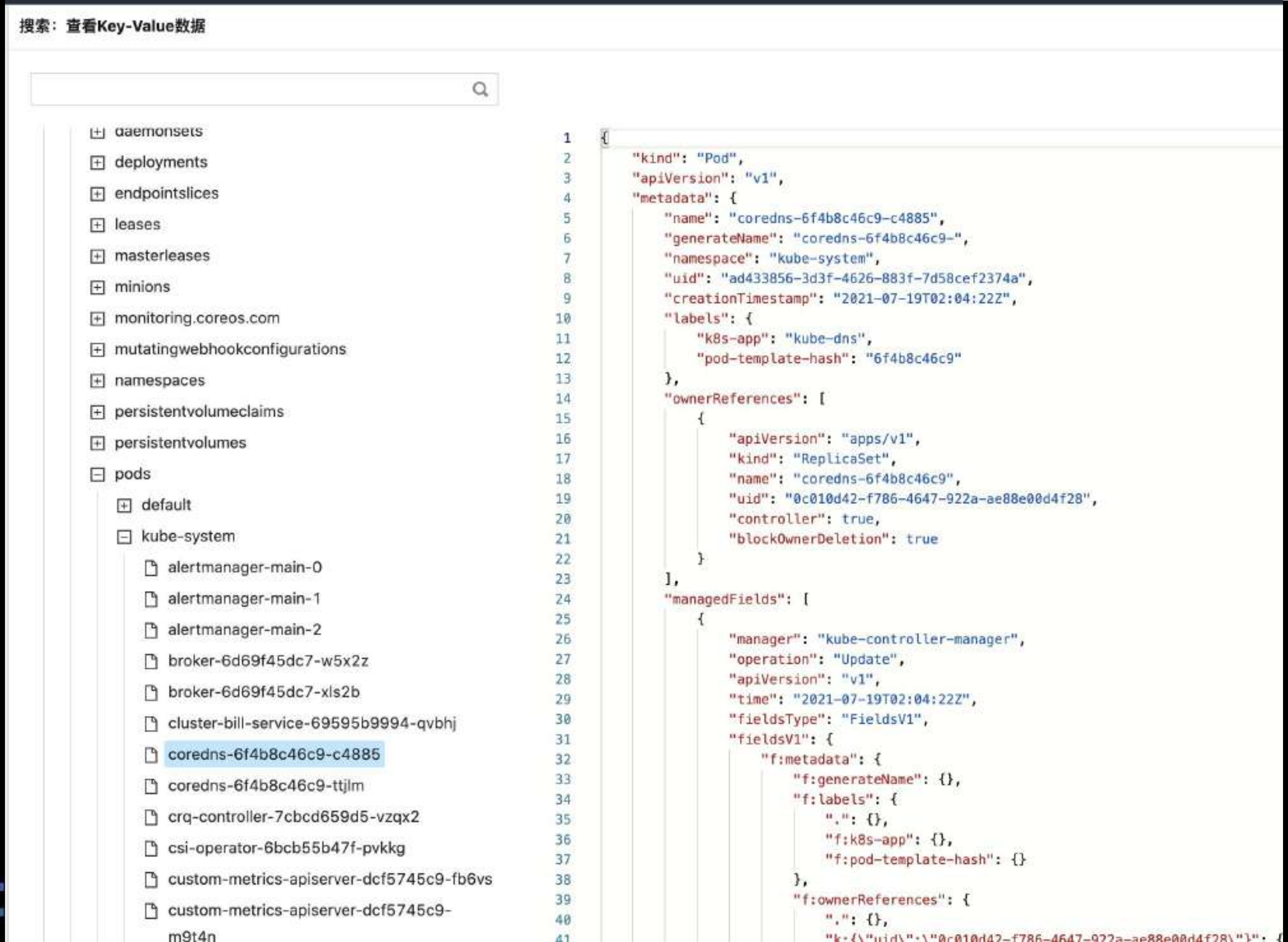
kstone

- 集群管理
  - 导入集群
  - 集群管理
- 运维中心
  - 可视化工具
  - 监控和巡检
  - 备份管理

集群管理

导入集群 新建集群

| 名称 | 状态 | 集群类型 | 节点个数 | 配置 | 磁盘配置 | 操作 |
|---|---|---|---|---|---|---|
| demo0x00 | 正常 | kstone-etcd-operator | 1 | 1核 1GB | ssd 10GB | 操作 删除 |
| test | 正常 | kstone-etcd-operator | 3 | 1核 1GB | ssd 10GB | 操作 删除 |

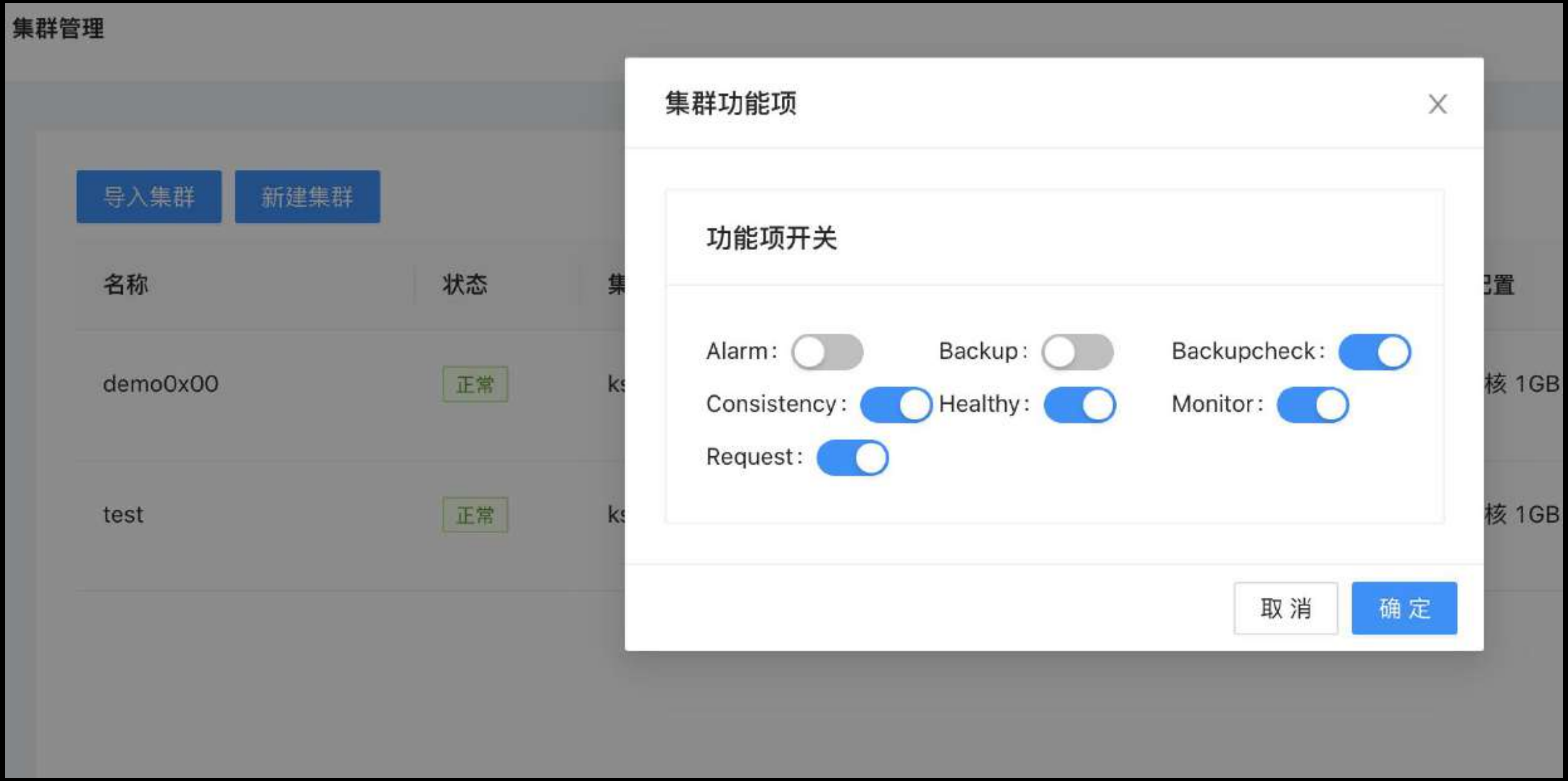


搜索：查看Key-Value数据

```
⊞ daemonsets
⊞ deployments
⊞ endpointslices
⊞ leases
⊞ masterleases
⊞ minions
⊞ monitoring.coreos.com
⊞ mutatingwebhookconfigurations
⊞ namespaces
⊞ persistentvolumeclaims
⊞ persistentvolumes
⊟ pods
    ⊞ default
    ⊟ kube-system
        alertmanager-main-0
        alertmanager-main-1
        alertmanager-main-2
        broker-6d69f45dc7-w5x2z
        broker-6d69f45dc7-xls2b
        cluster-bill-service-69595b9994-qvbhj
        coredns-6f4b8c46c9-c4885
        coredns-6f4b8c46c9-ttjlm
        crq-controller-7cbcd659d5-vzqx2
        csi-operator-6bcb55b47f-pvkkg
        custom-metrics-apiserver-dcf5745c9-fb6vs
        custom-metrics-apiserver-dcf5745c9-
        m9t4n
```

```
{
    "kind": "Pod",
    "apiVersion": "v1",
    "metadata": {
        "name": "coredns-6f4b8c46c9-c4885",
        "generateName": "coredns-6f4b8c46c9-",
        "namespace": "kube-system",
        "uid": "ad433856-3d3f-4626-883f-7d58cef2374a",
        "creationTimestamp": "2021-07-19T02:04:22Z",
        "labels": {
            "k8s-app": "kube-dns",
            "pod-template-hash": "6f4b8c46c9"
        },
        "ownerReferences": [
            {
                "apiVersion": "apps/v1",
                "kind": "ReplicaSet",
                "name": "coredns-6f4b8c46c9",
                "uid": "0c010d42-f786-4647-922a-ae88e00d4f28",
                "controller": true,
                "blockOwnerDeletion": true
            }
        ],
        "managedFields": [
            {
                "manager": "kube-controller-manager",
                "operation": "Update",
                "apiVersion": "v1",
                "time": "2021-07-19T02:04:22Z",
                "fieldsType": "FieldsV1",
                "fieldsV1": {
                    "f:metadata": {
                        "f:generateName": {},
                        "f:labels": {
                            ".": {},
                            "f:k8s-app": {},
                            "f:pod-template-hash": {}
                        },
                        "f:ownerReferences": {
                            ".": {},
                            "k:{\"uid\":\"0c010d42-f786-4647-922a-ae88e00d4f28\"}": {
```

## Kstone


集群管理

导入集群 新建集群

| 名称 | 状态 | 集 | 配置 |
|---|---|---|---|
| demo0x00 | 正常 | ks | 核 1GB |
| test | 正常 | ks | 核 1GB |

集群功能项

功能项开关

Alarm: off
Backup: off
Backupcheck: on
Consistency: on
Healthy: on
Monitor: on
Request: on

•Monitor: 开启 etcd 监控，采集 metrics

•Backup: 开启 etcd 定时备份

•Backupcheck: 检查 etcd 在对象存储系统备份文件数是否正常

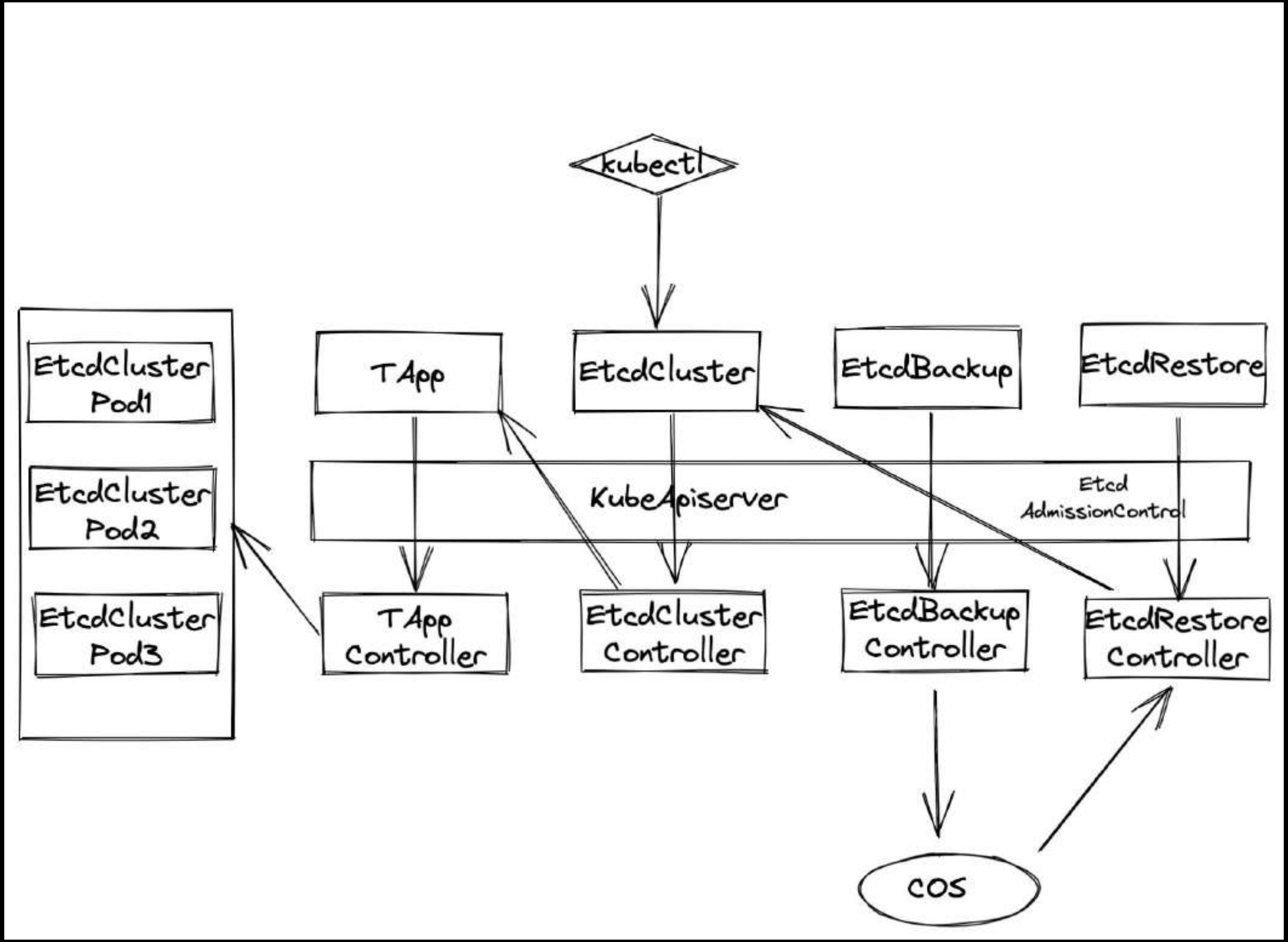
•Consistency：数据一致性巡检

•Healthy: 健康度巡检

•Request: 开启 etcd 热点写请求、key 数量分析统计

• Alarm: 检查 etcd 集群是否存在 db 满、数据毁坏的异常告警

## —kstone-etcd-operator

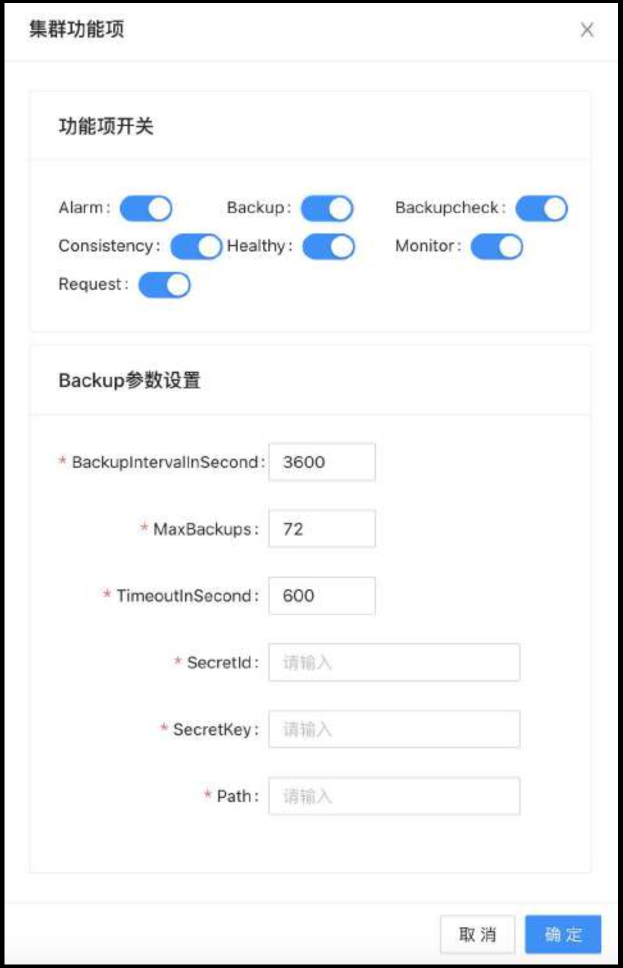


- 声明式 API 运维更加方便
- Tapp 相对 StatefulSet 更加灵活
  - 支持持久化存储等 StatefulSet 原生能力
  - 支持不同 Pod 使用不同模板
  - 支持原地更新
  - 支持停止指定 Pod
  - 腾讯内部大规模使用
- 使用 AdmissionControl 可进行复杂的参数校验及转换
- 自定义 Controller 可进行更安全高效的自愈动作及复杂的运维操作

## Kstone-etcd-operator

- 集群生命周期管理
- 持久化存储支持
- 高可用及容灾，故障自愈
- 集群升级
- 水平扩容&垂直扩容
- 支持HTTPS，自动签发证书，自定义域名等
- 自定义参数
- 备份恢复
- 删除保护

# Kstone



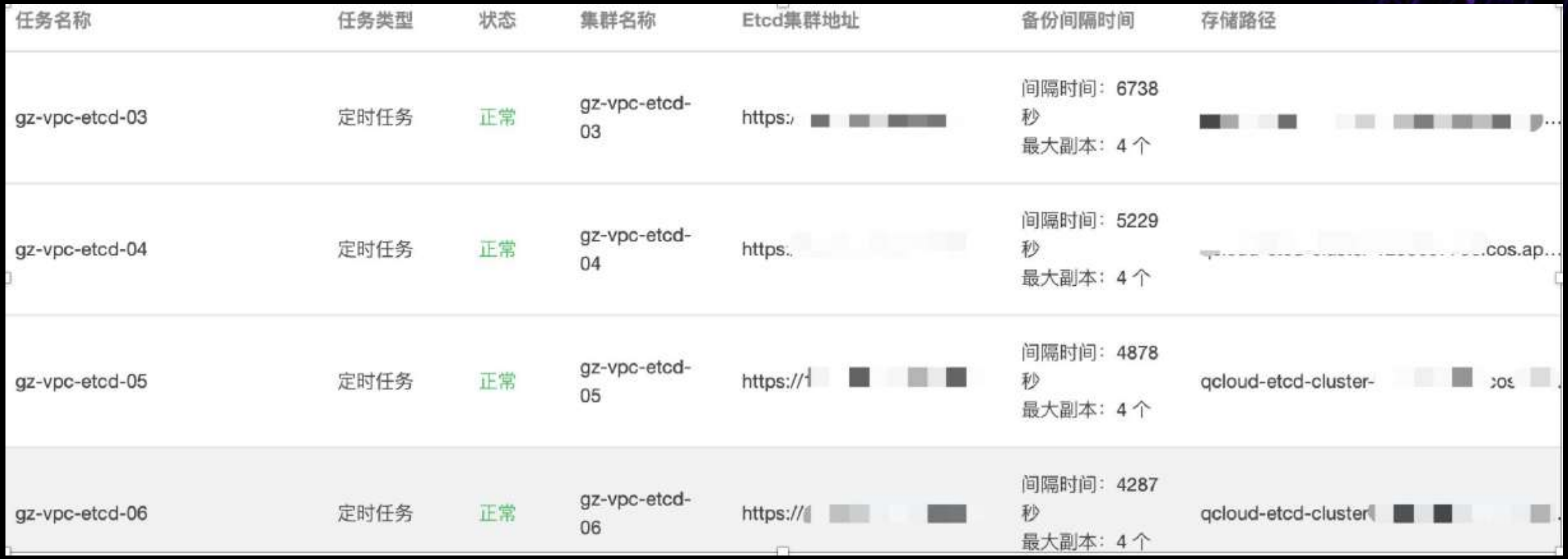
**集群功能项**

**功能项开关**

| | | |
|---|---|---|
| Alarm: ✓ | Backup: ✓ | Backupcheck: ✓ |
| Consistency: ✓ | Healthy: ✓ | Monitor: ✓ |
| Request: ✓ | | |

**Backup参数设置**

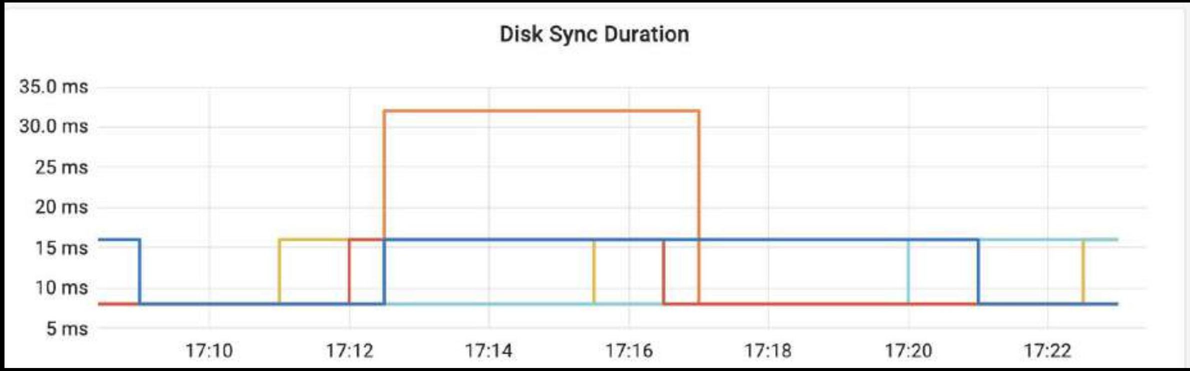
| | |
|---|---|
| * BackupIntervalInSecond: | 3600 |
| * MaxBackups: | 72 |
| * TimeoutInSecond: | 600 |
| * SecretId: | 请输入 |
| * SecretKey: | 请输入 |
| * Path: | 请输入 |

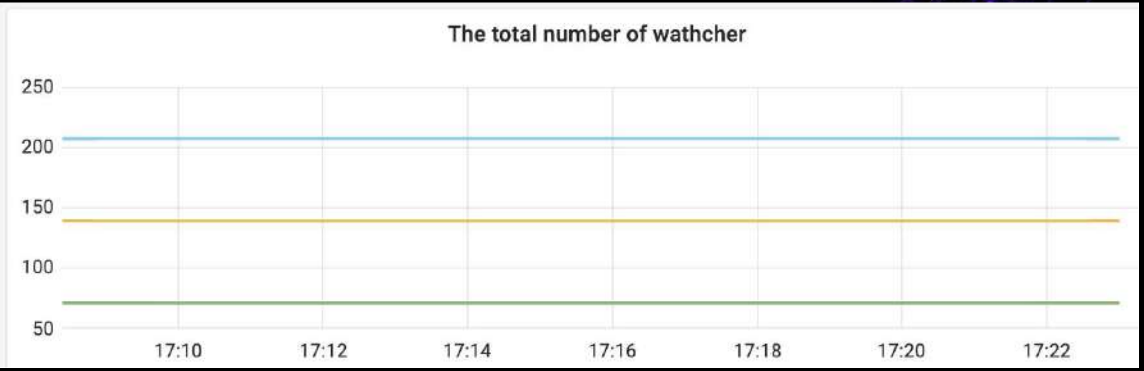


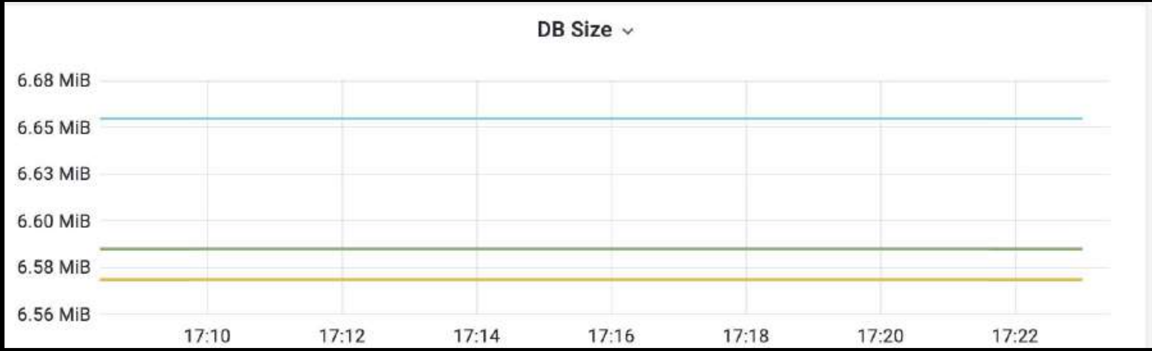
| 任务名称 | 任务类型 | 状态 | 集群名称 | Etcd集群地址 | 备份间隔时间 | 存储路径 |
|---|---|---|---|---|---|---|
| gz-vpc-etcd-03 | 定时任务 | 正常 | gz-vpc-etcd-03 | https:/[redacted] | 间隔时间：6738 秒 最大副本：4 个 | [redacted] |
| gz-vpc-etcd-04 | 定时任务 | 正常 | gz-vpc-etcd-04 | https:[redacted] | 间隔时间：5229 秒 最大副本：4 个 | [redacted].cos.ap... |
| gz-vpc-etcd-05 | 定时任务 | 正常 | gz-vpc-etcd-05 | https://[redacted] | 间隔时间：4878 秒 最大副本：4 个 | qcloud-etcd-cluster-[redacted] |
| gz-vpc-etcd-06 | 定时任务 | 正常 | gz-vpc-etcd-06 | https://[redacted] | 间隔时间：4287 秒 最大副本：4 个 | qcloud-etcd-cluster[redacted] |

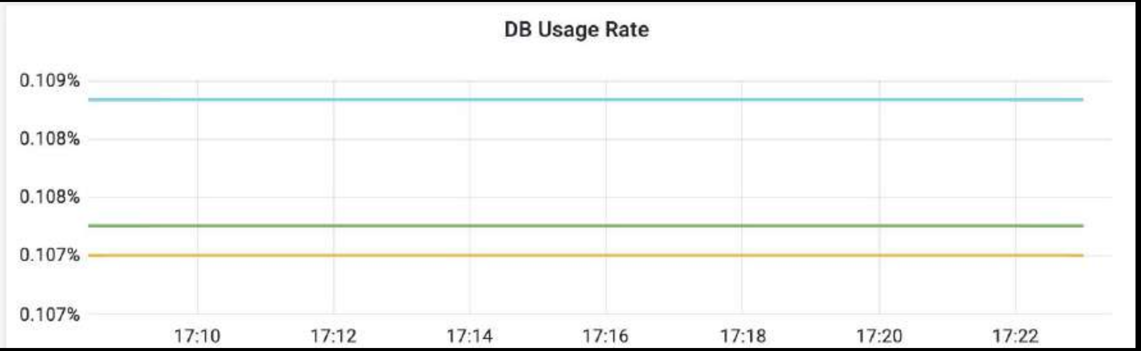
- 默认支持cos对象存储，支持分钟级备份到COS
- S3/oss/等对象存储计划支持中

## Kstone etcd

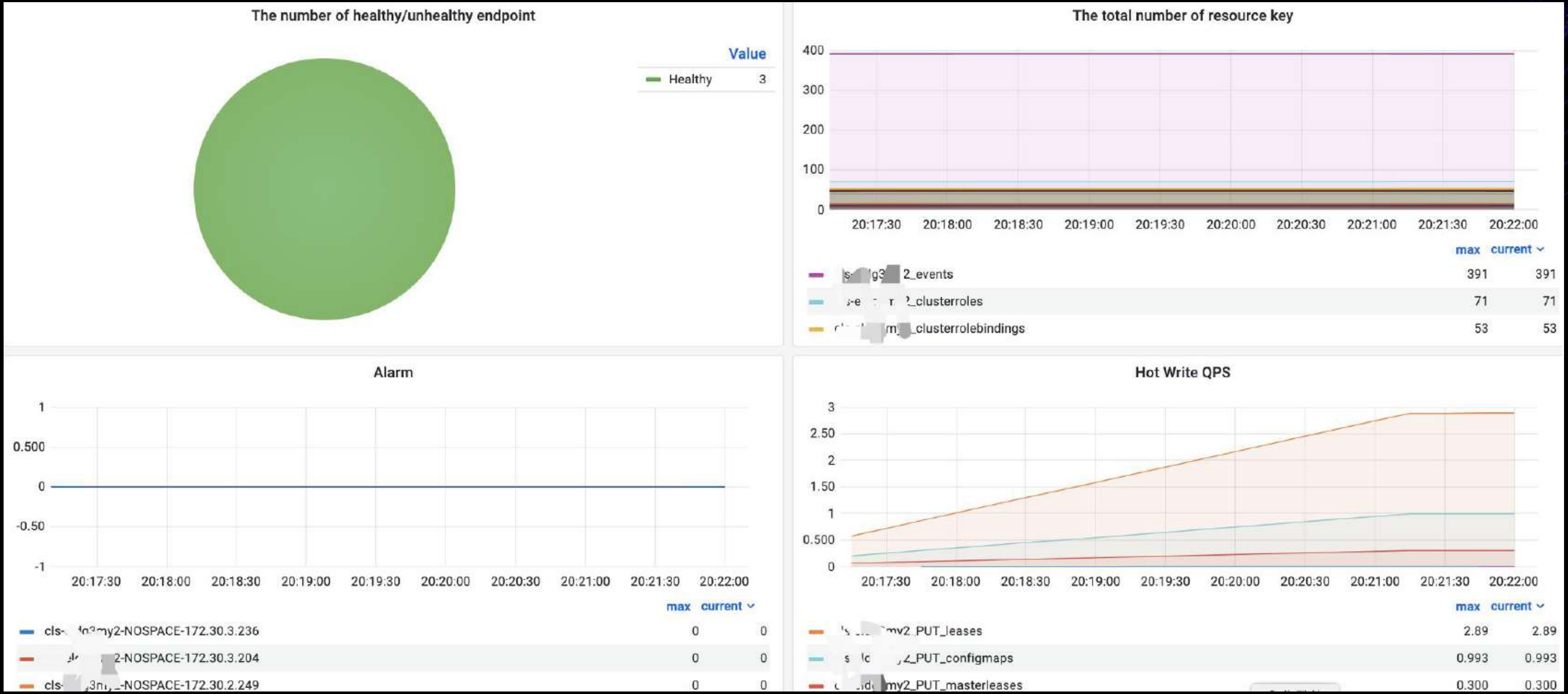

Disk Sync Duration
35.0 ms
30.0 ms
25 ms
20 ms
15 ms
10 ms
5 ms
17:10
17:12
17:14
17:16
17:18
17:20
17:22


The total number of wathcher
250
200
150
100
50
17:10
17:12
17:14
17:16
17:18
17:20
17:22


DB Size
6.68 MiB
6.65 MiB
6.63 MiB
6.60 MiB
6.58 MiB
6.56 MiB
17:10
17:12
17:14
17:16
17:18
17:20
17:22


DB Usage Rate
0.109%
0.108%
0.108%
0.107%
0.107%
17:10
17:12
17:14
17:16
17:18
17:20
17:22

## Kstone etcd



**The number of healthy/unhealthy endpoint**

| | Value |
|---|---|
| 🟩 Healthy | 3 |

**The total number of resource key**

| | max | current |
|---|---|---|
| [redacted] 2_events | 391 | 391 |
| [redacted] 2_clusterroles | 71 | 71 |
| [redacted] _clusterrolebindings | 53 | 53 |

**Alarm**

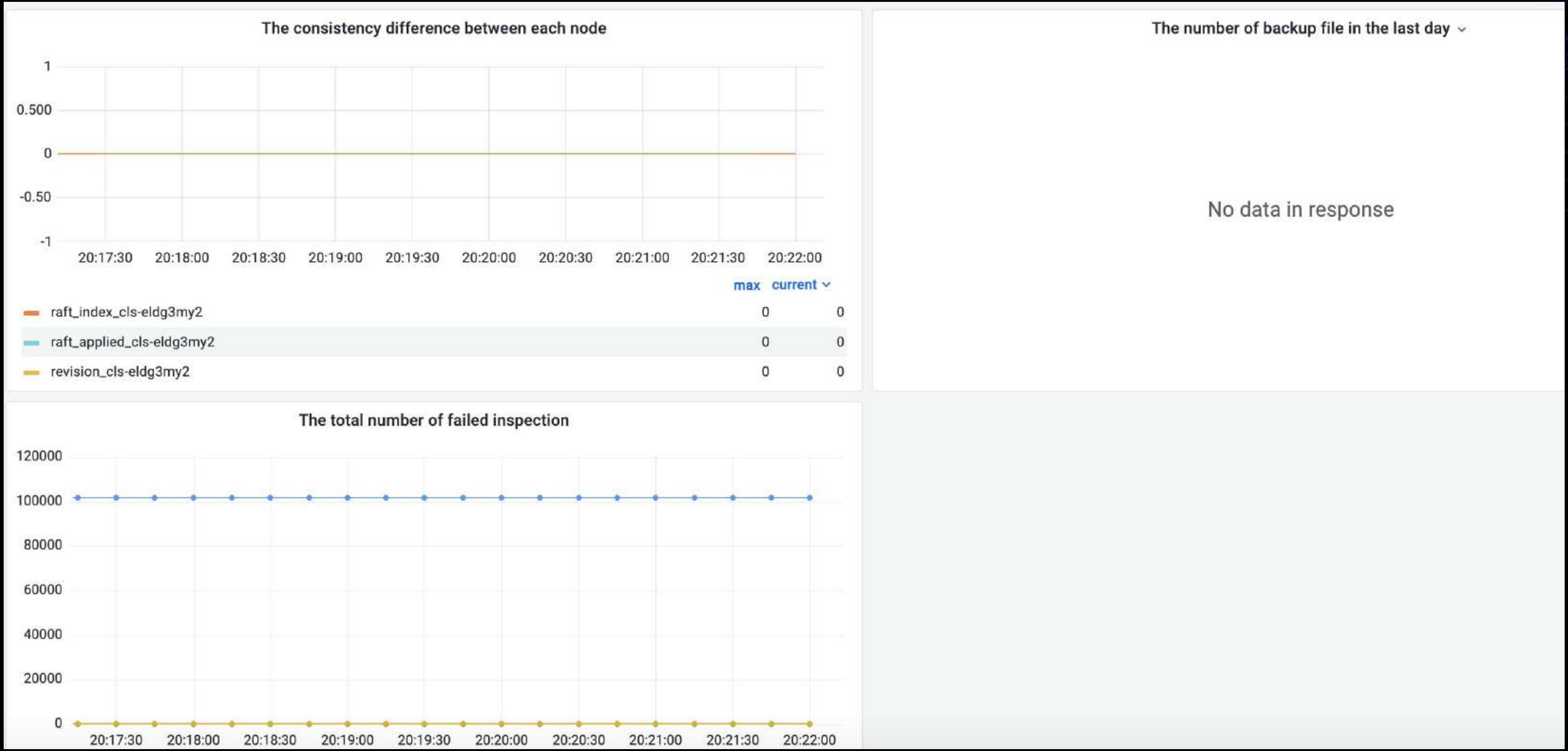
| | max | current |
|---|---|---|
| cls-[redacted]my2-NOSPACE-172.30.3.236 | 0 | 0 |
| [redacted]2-NOSPACE-172.30.3.204 | 0 | 0 |
| cls-[redacted]-NOSPACE-172.30.2.249 | 0 | 0 |

**Hot Write QPS**

| | max | current |
|---|---|---|
| [redacted]my2_PUT_leases | 2.89 | 2.89 |
| [redacted]2_PUT_configmaps | 0.993 | 0.993 |
| [redacted]my2_PUT_masterleases | 0.300 | 0.300 |

apache apisix

## Kstone etcd

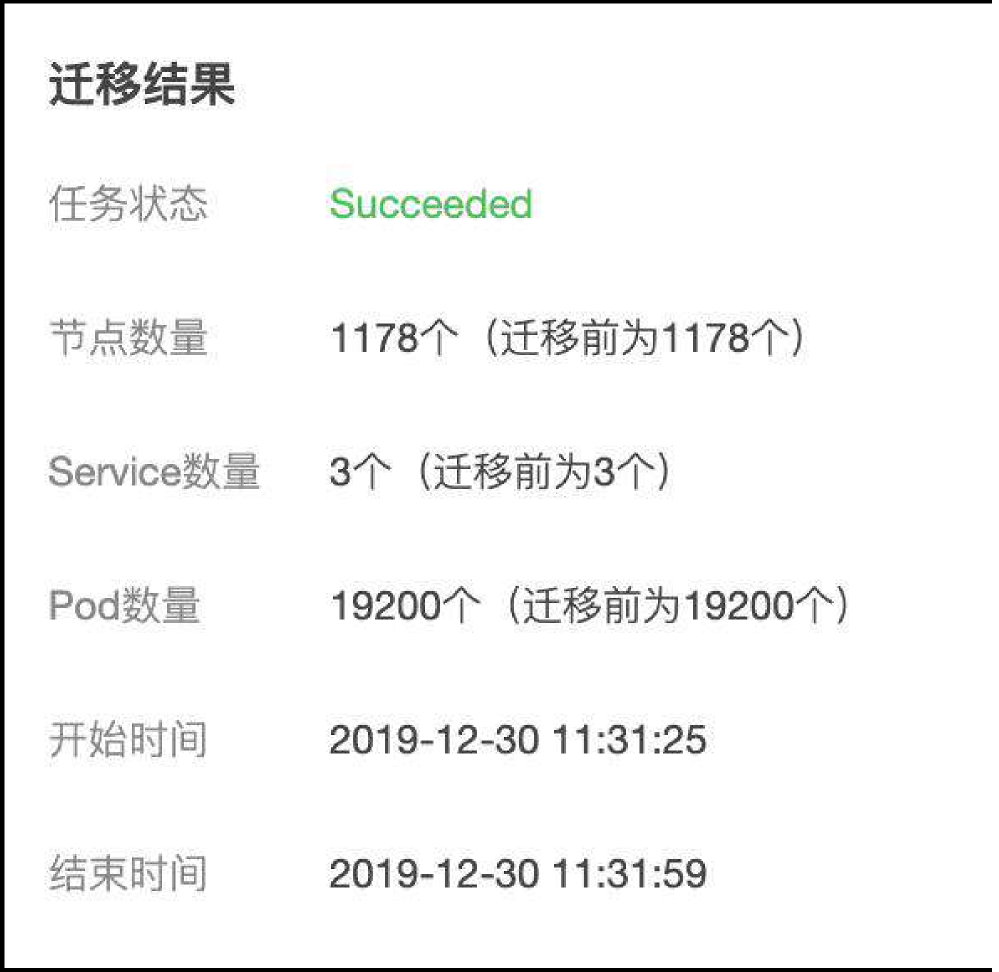


| | max | current |
|---|---|---|
| raft_index_cls-eldg3my2 | 0 | 0 |
| raft_applied_cls-eldg3my2 | 0 | 0 |
| revision_cls-eldg3my2 | 0 | 0 |

## Kstone

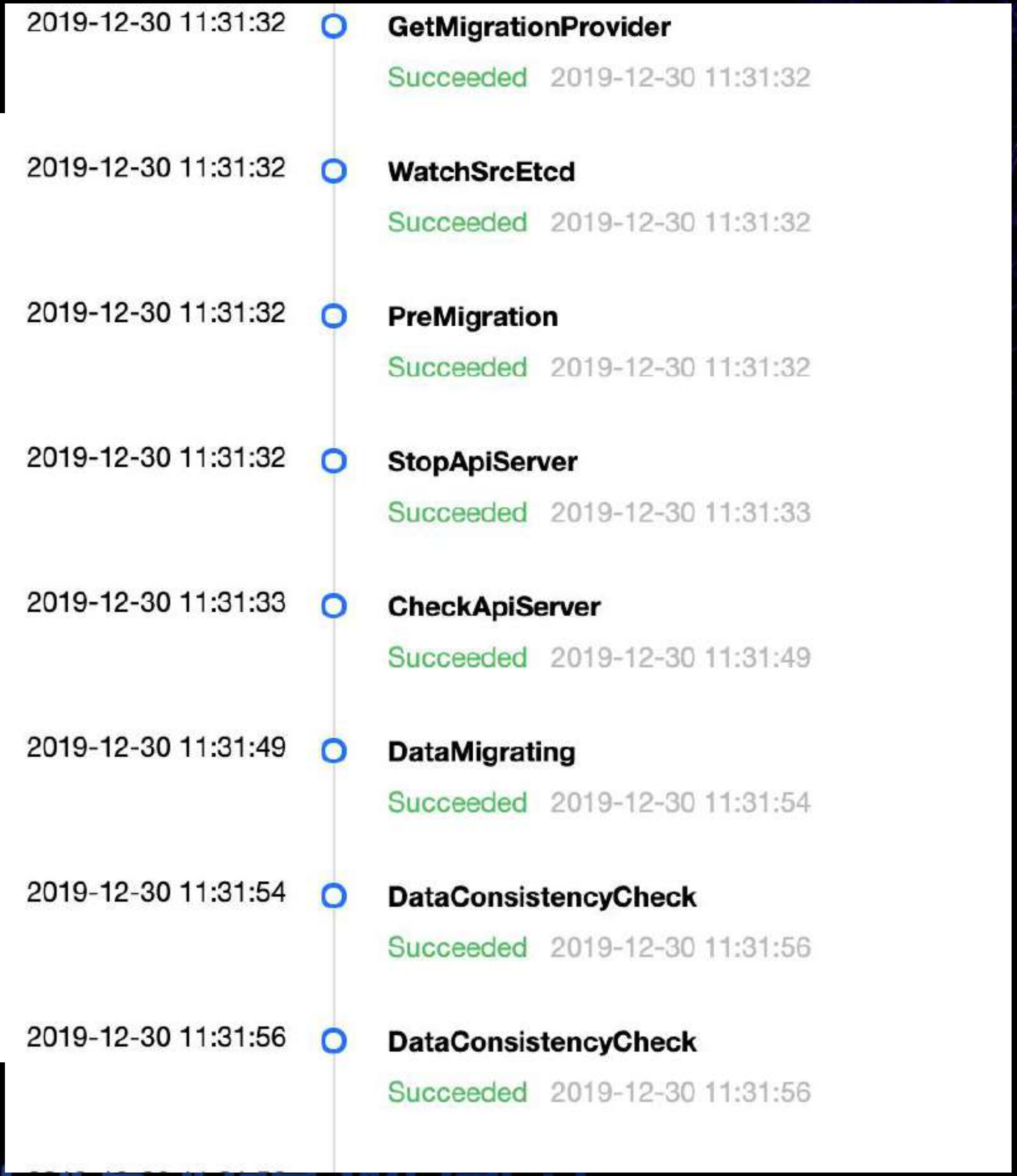
- 新增 EtcdMigration 迁移任务 CRD，描述迁移源 etcd、目的 etcd 集群、迁移算法、一致性检测策略等。
- 迁移算法抽象化，支持多种 Provider, 比如 etcd v2->v3, v3->v3 冷迁移，v3-v3 热迁移。
- 支持多种维度的数据一致性检查策略，比如 etcd 维度的数据一致性检查，k8s应用层的资源对象一致性检查等。
- 针对 k8s 场景迁移 etcd 导致的 client list-watch hang 住问题（迁移后的 etcd 版本号 (apiserver resource version) 小于
- 原有 etcd，修改 k8s 版本源码，增加 watch 操作的 timeout 机制。

## Kstone



迁移结果

| | |
|---|---|
| 任务状态 | Succeeded |
| 节点数量 | 1178个（迁移前为1178个） |
| Service数量 | 3个（迁移前为3个） |
| Pod数量 | 19200个（迁移前为19200个） |
| 开始时间 | 2019-12-30 11:31:25 |
| 结束时间 | 2019-12-30 11:31:59 |



| | | |
|---|---|---|
| 2019-12-30 11:31:32 | GetMigrationProvider | Succeeded 2019-12-30 11:31:32 |
| 2019-12-30 11:31:32 | WatchSrcEtcd | Succeeded 2019-12-30 11:31:32 |
| 2019-12-30 11:31:32 | PreMigration | Succeeded 2019-12-30 11:31:32 |
| 2019-12-30 11:31:32 | StopApiServer | Succeeded 2019-12-30 11:31:33 |
| 2019-12-30 11:31:33 | CheckApiServer | Succeeded 2019-12-30 11:31:49 |
| 2019-12-30 11:31:49 | DataMigrating | Succeeded 2019-12-30 11:31:54 |
| 2019-12-30 11:31:54 | DataConsistencyCheck | Succeeded 2019-12-30 11:31:56 |
| 2019-12-30 11:31:56 | DataConsistencyCheck | Succeeded 2019-12-30 11:31:56 |

APACHE APISIX

感谢聆听
THANKS