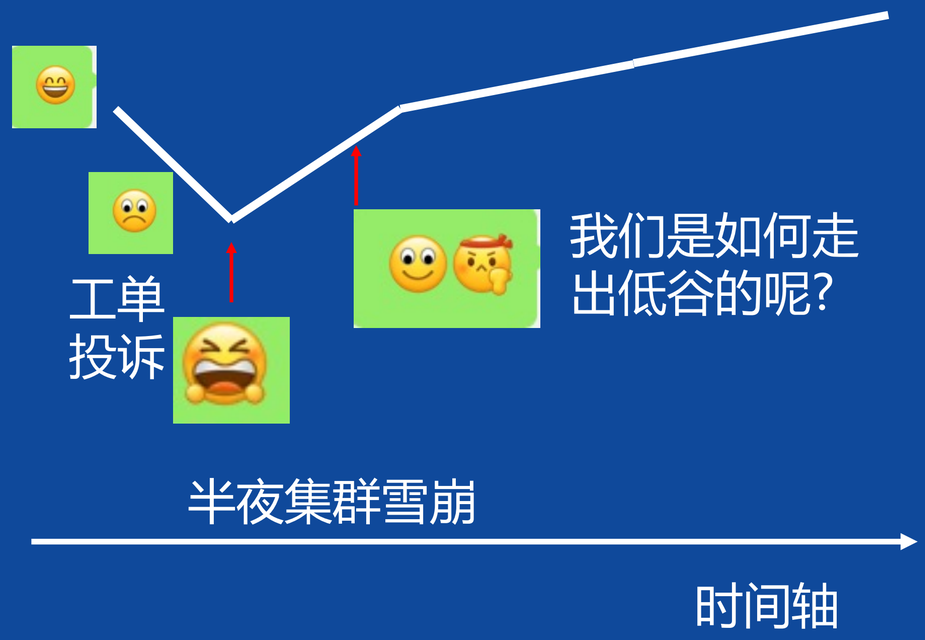
腾讯大规模云原生平台稳定性实践

唐聪

腾讯云TKE稳定性负责人，腾讯云技术专家

自我介绍

- 2014年本科毕业加入腾讯，负责大规模qq后台业务、redis平台开发
- 2018年加入腾讯云容器团队，负责大规模etcd集群治理和优化，master组件稳定性优化
- etcd社区活跃贡献者，2021年热门极客时间专栏《etcd实战课》作者



大纲

- k8s故障案例分析与最佳实践
- 大规模etcd集群治理
- Kubernetes其他master组件治理
- QA

k8s故障案例分析与最佳实践

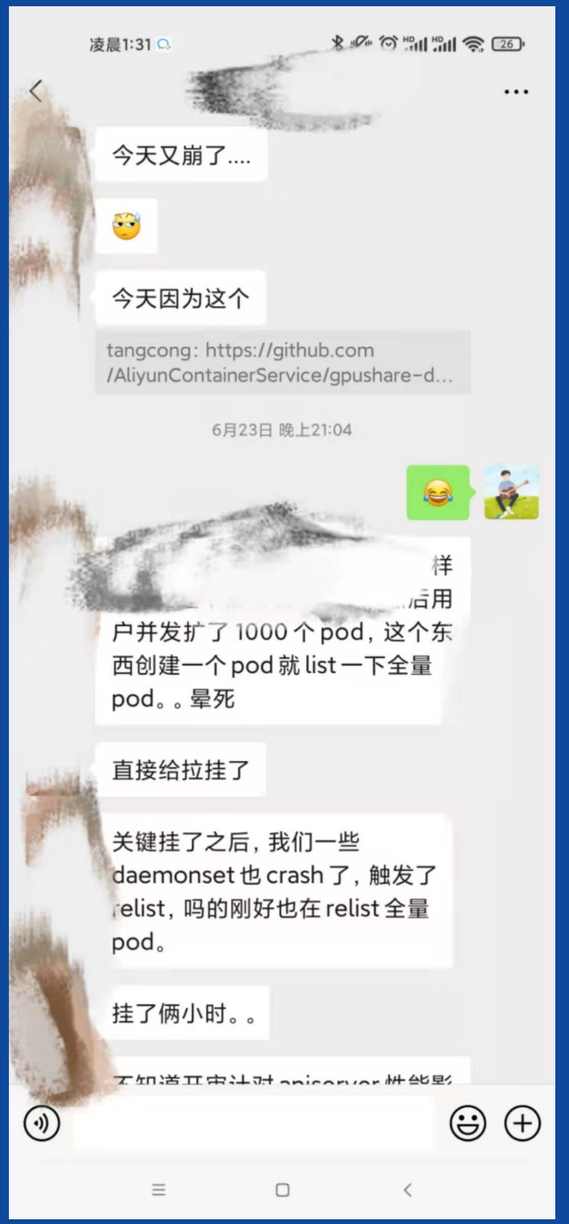
从故障案例说起–k8s稳定性为何不可忽视?

如何保障k8s的稳定性呢？

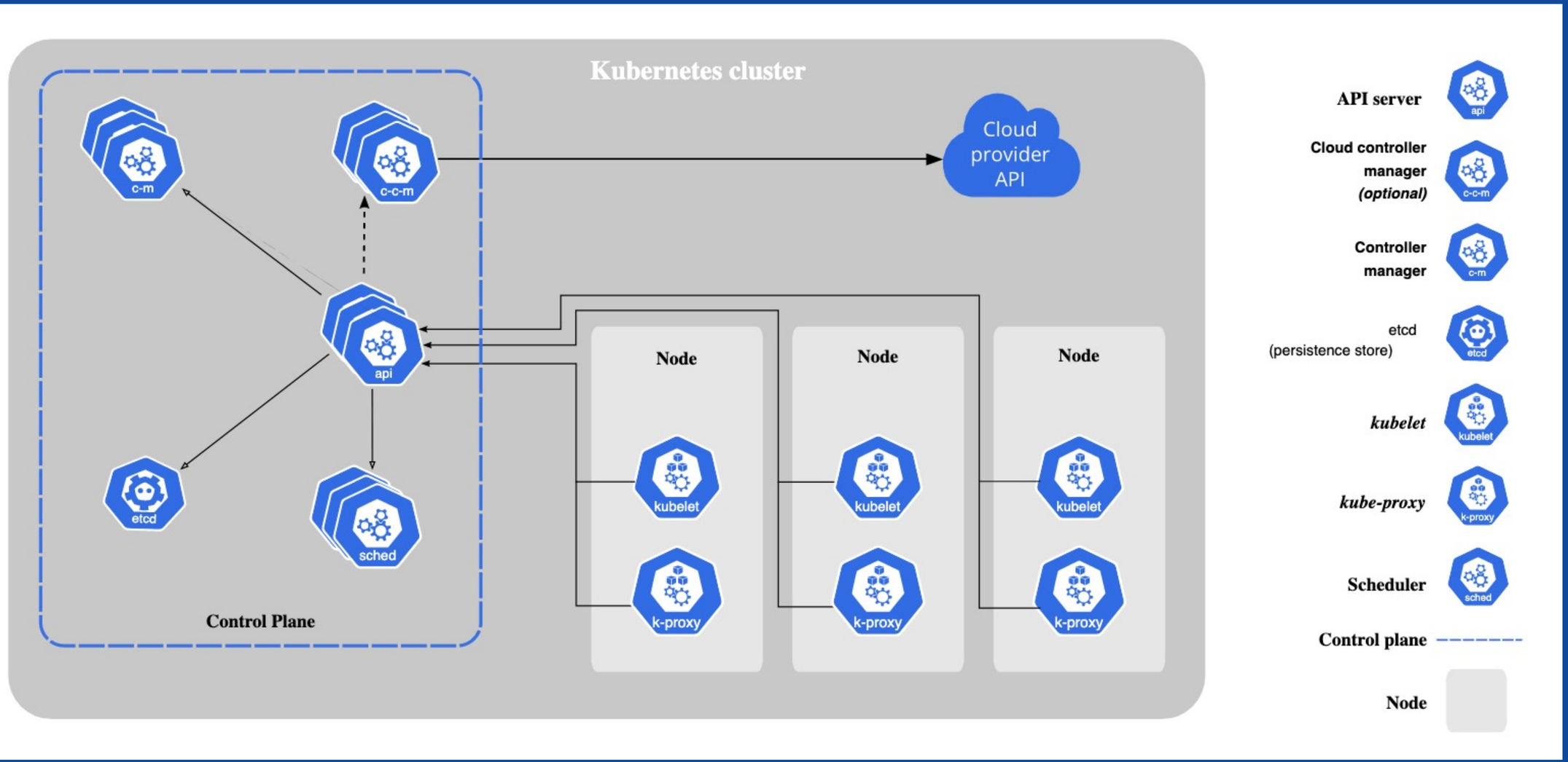
故障案例1:
某大厂业务k8s集群因list请求多次雪崩，其中某次故障6小时，恢复过程中又决策失误，导致业务数据面出现断网，造成最严重的故障

故障案例2:
某剧因太火爆，核心服务部署在k8s集群中，业务自行开发的扩容流程，在流量高峰自动扩容失败，用户大量重试，服务雪崩，最终导致大规模投诉

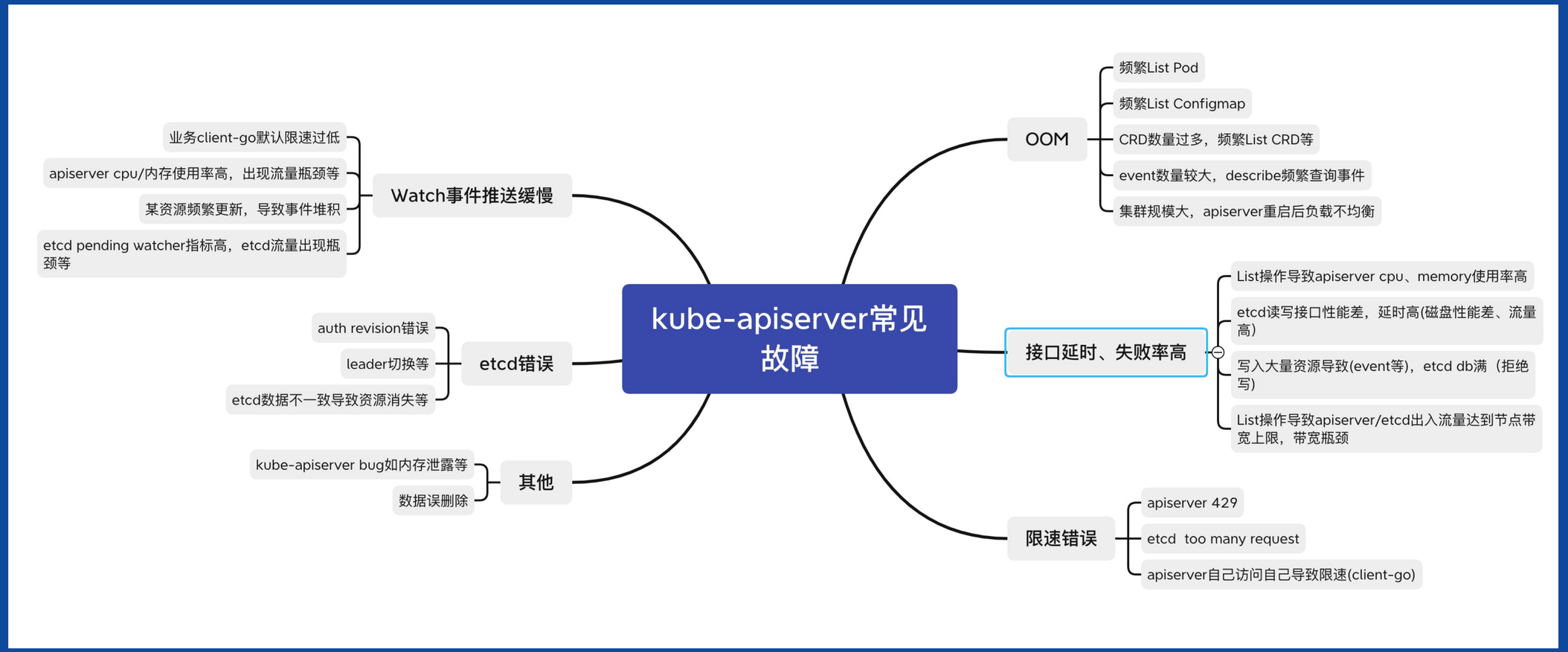
故障案例3:
某业务operator，因依赖方业务变更，返回数据异常，operator快速进行一致性协调操作中，删除了用户的k8s某类资源，最终导致大量用户访问集群异常。



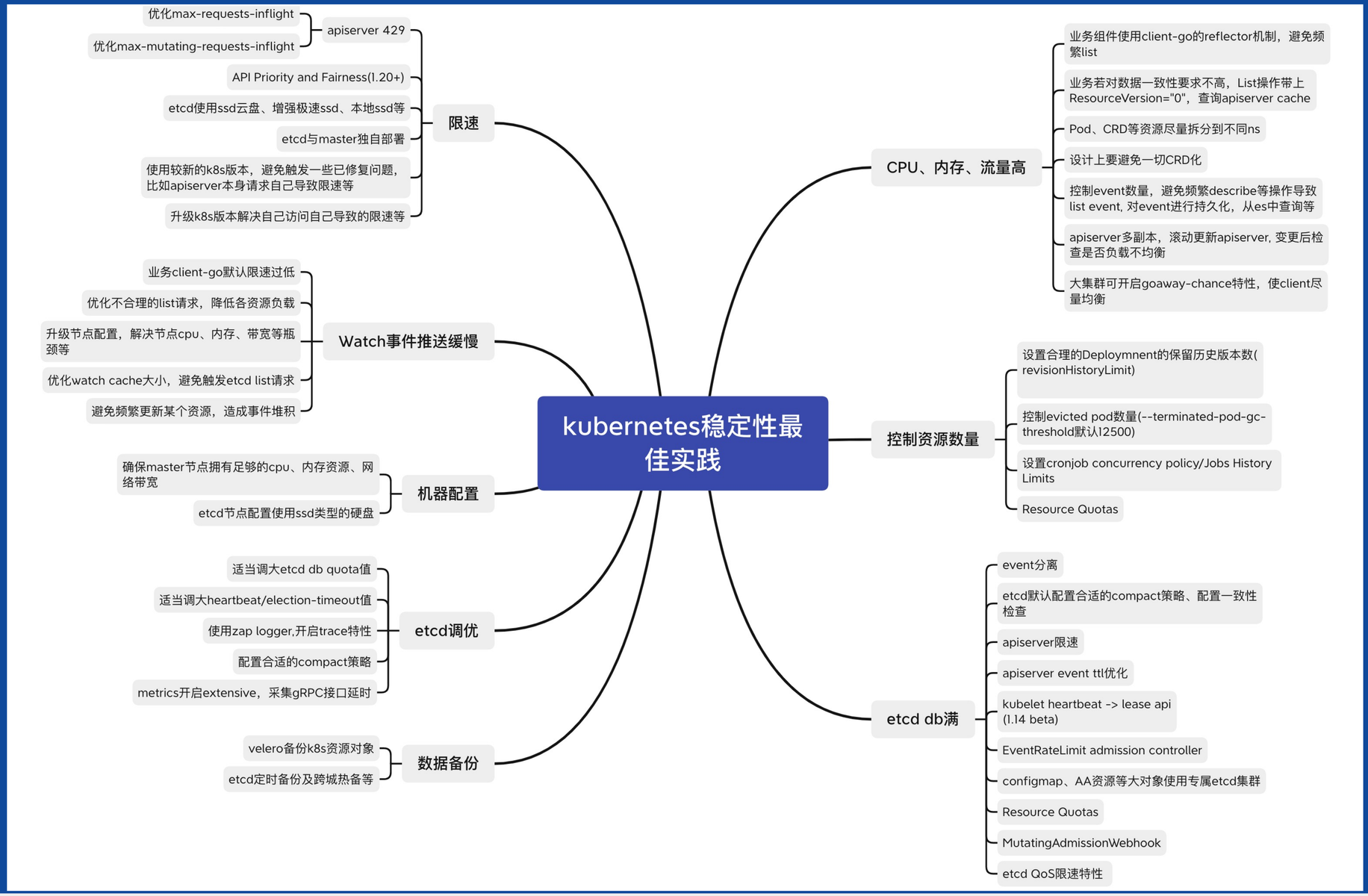
Kubernetes核心架构



Kube-apiserver常见故障



Kubernetes稳定性最佳实践



搞崩Kube-apiserver查询典型案例1-标签查询

default ns下当前有10w个pod，其中apisix pod 2个，标签为app=apisix

kubectl
get pod -l app=apisix

A: 2个

B: 10万个

C: 500个

GET api/v1/namespaces/default/pods?labelSelector=app%3Dapisix&limit=500
200 OK

搞崩Kube-apiserver查询典型案例2-Describe Event查询

kube-system ns下当前有10w条event, 以coredns pod-name开头的event 6条

kubectl –n kube-system describe pod/core-dns-xx

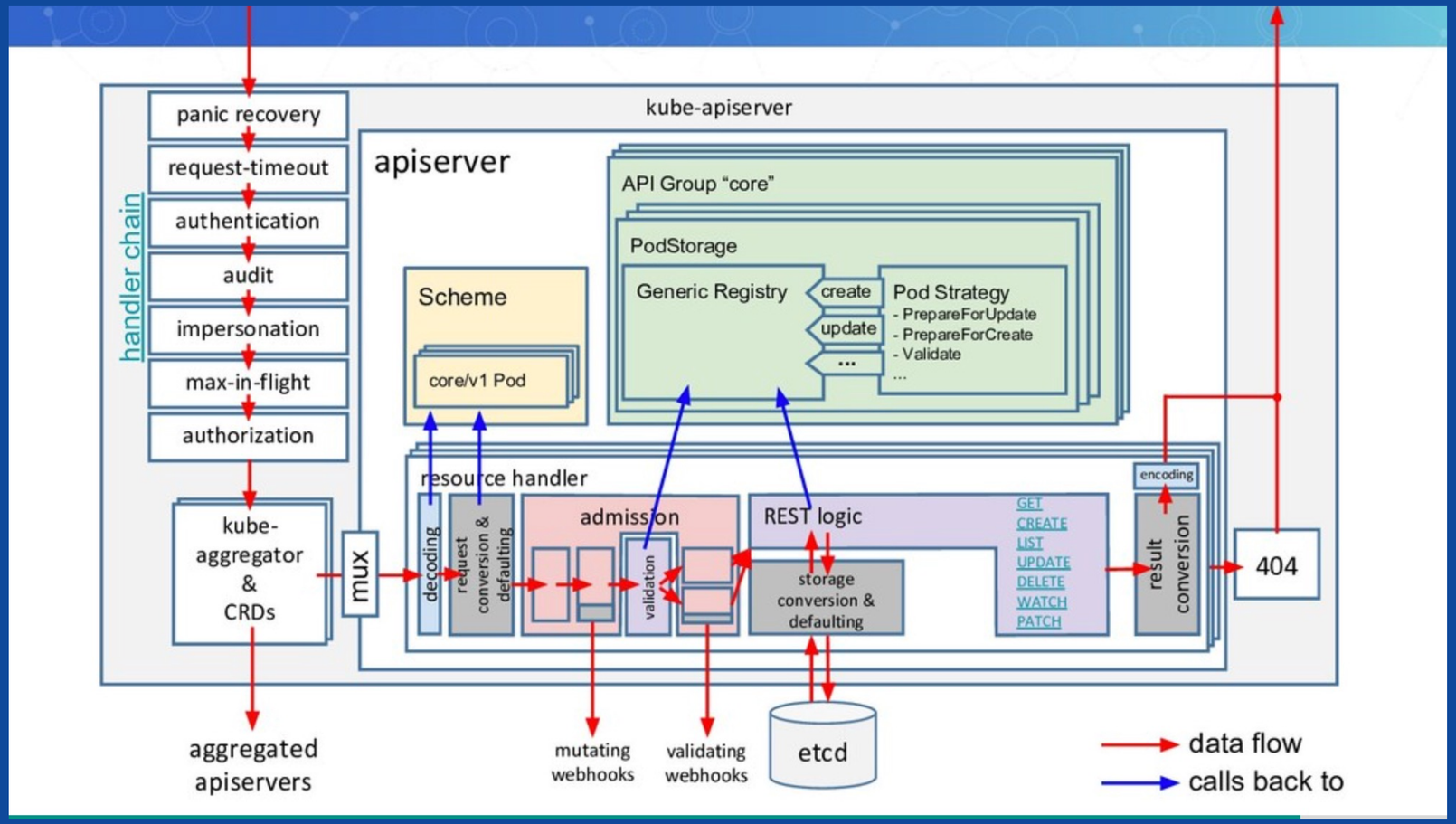
A: 6条

B: 10万条

C: 不确定

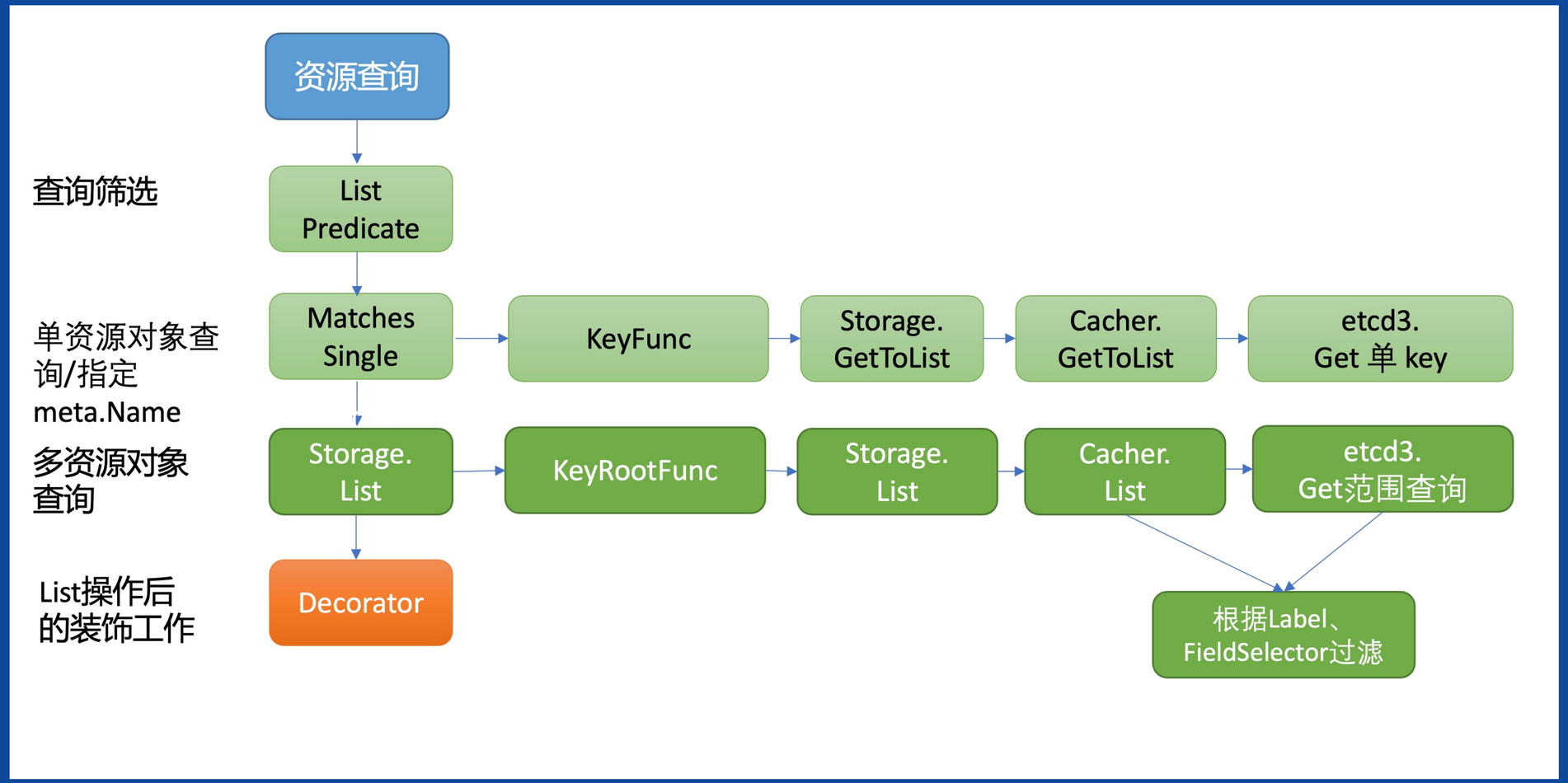
GET /api/v1/namespaces/kube-system/events?fieldSelector=involvedObject.name%3Dcoredns-5d4c6744bf-88lr4%2CinvolvedObject.namespace%3Dkube-system%2CinvolvedObject.uid%3Dd7f81fbe-9133-46ca-ba22-a5772ad2d6ae 200 OK

apiserver如何执行查询get/list等资源的请求呢?



- 鉴权/用户身份是否合法
- 审计/记录用户详细行为
- 限速/请求优先级及公平管理
- 授权/用户是否有权限对其访问的资源进行操作
- 准入机制/提供在访问资源前拦截请求的动态扩展能力
- 匹配resource handler
- 查询etcd存储

Kube-apiserver List查询资源简要流程



k8s数据是如何存储在etcd中

/registry/clusterrolebindings/system:coredns
/registry/clusterroles/system:coredns
/registry/configmaps/kube-system/coredns
/registry/deployments/kube-system/coredns
/registry/events/kube-system/coredns-7fcc6d65dc-6njlg.1662c287aabf742b
/registry/events/kube-system/coredns-7fcc6d65dc-6njlg.1662c288232143ae
/registry/pods/default/apisix-7fcc6d65dc-jvj26
/registry/pods/default/apisix-7fcc6d65dc-mgvtb
/registry/replicasets/kube-system/coredns-7fcc6d65dc
/registry/secrets/kube-system/coredns-token-hpqbt
/registry/serviceaccounts/kube-system/coredns

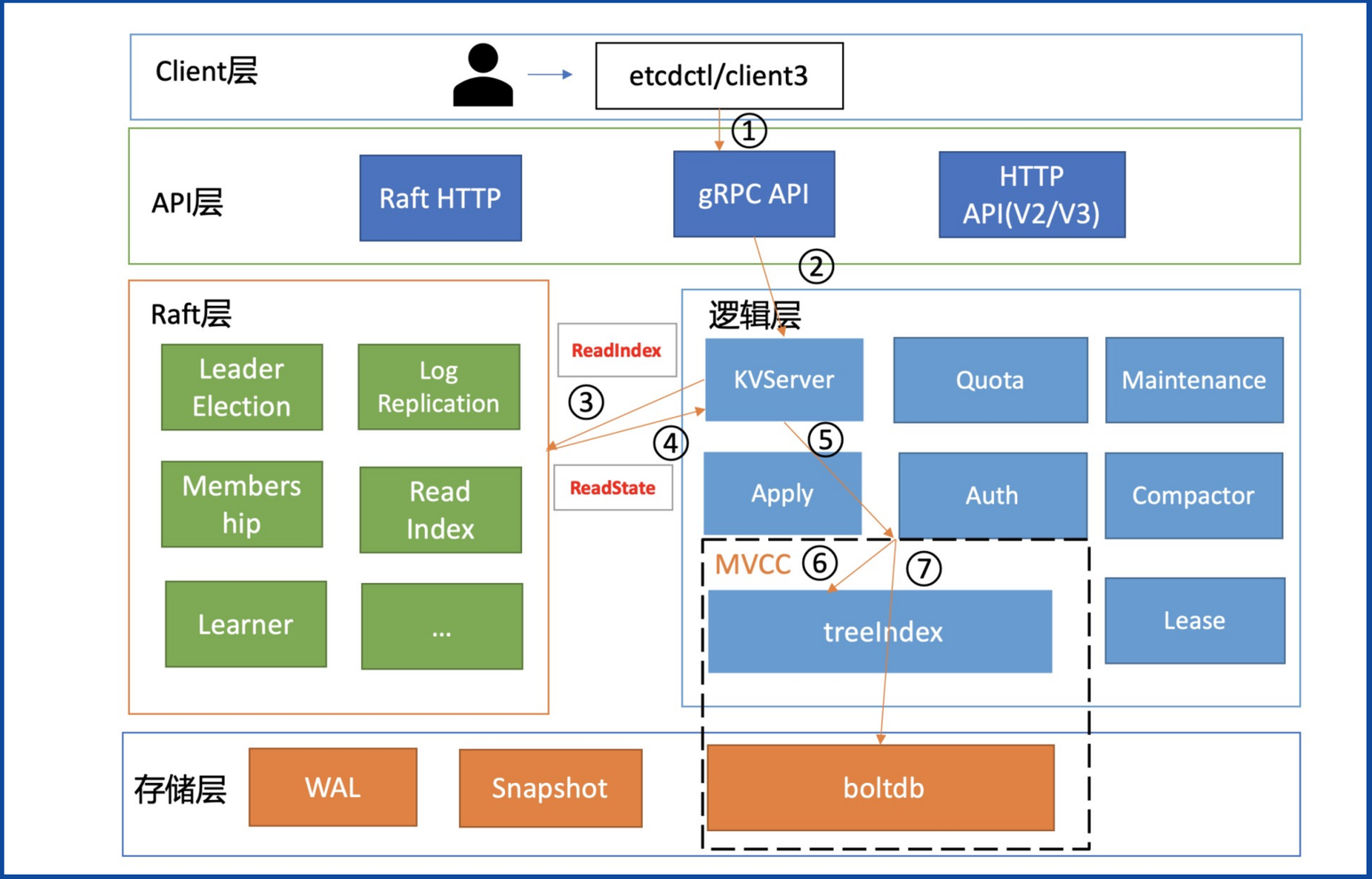
etcd收到apiserver查询pod请求是怎样的呢?

```go
// List implements storage.Interface.List.
func (s *store) List(ctx context.Context, key, resourceVersion string, pred stora
    // ...
    rangeEnd := clientv3.GetPrefixRangeEnd(keyPrefix)
    options = append(options, clientv3.WithRange(rangeEnd))
    // ...
    // loop until we have filled the requested limit from etcd or there are no mo
    var lastKey []byte
    var hasMore bool
    var getResp *clientv3.GetResponse
    for {
        startTime := time.Now()
        getResp, err = s.client.KV.Get(ctx, key, options...)
        metrics.RecordEtcdRequestLatency( verb: "list", getTypeName(listPtr), star
        if err != nil {
            return interpretListError(err, len(pred.Continue) > 0, continueKey, k
        }
        if err = s.ensureMinimumResourceVersion(resourceVersion, uint64(getResp.k
```

- Apiserver分页请求etcd server的gRPCKV模块的Range接口(单key、范围查询等），默认线性读模式，强一致性

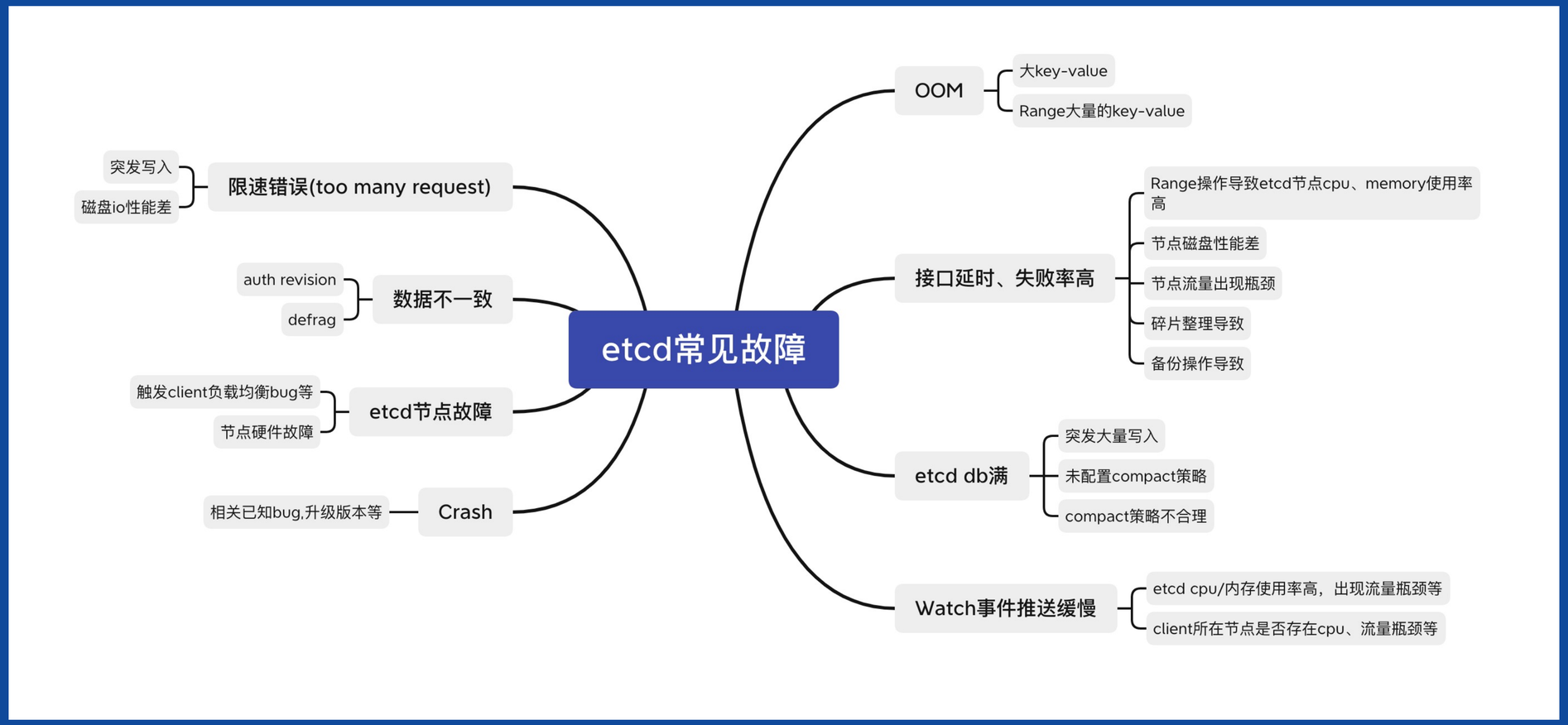
response type = /etcdserverpb.KV/Range, request count = 0, request size = 53, response count = 500, response size = 2505334, request content = key:"/registry/pods/default/" range_end:"/registry/pods/default0" limit:500

etcd读请求流程–kubernetes是如何从etcd查询数据的

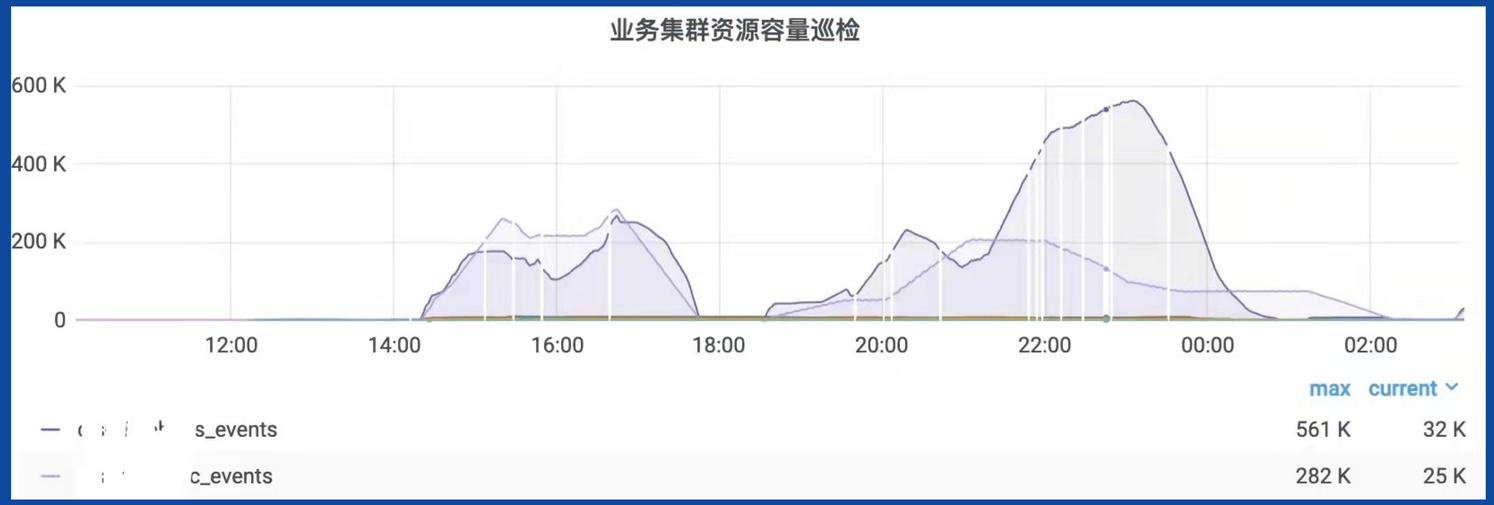


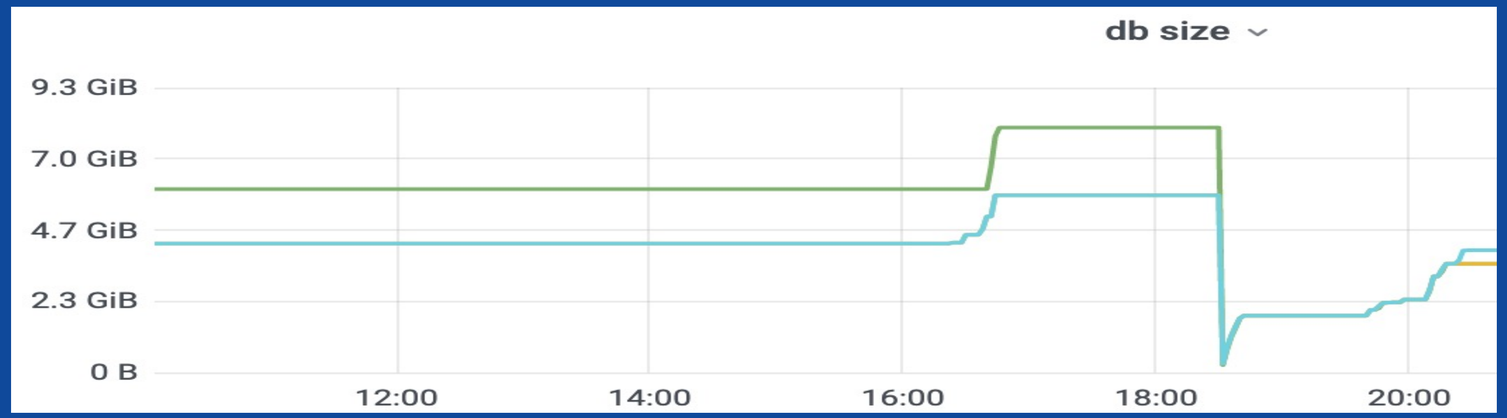
- etcd gRPC KV Server收到请求后，则进入KV模块的Range接口, etcd 3.1版本后线性读实现是readIndex机制，它需要从leader获取最新已提交的日志索引号(commitedIndex)
- 等待本节点已经应用到状态机的最高索引号大于等于leader返回的commitedIndex后才能允许读
- 从索引树中获取key对应的最新版本号
- 然后再优先从cache中获取，若未命中则从boltdb获取

etcd常见故障

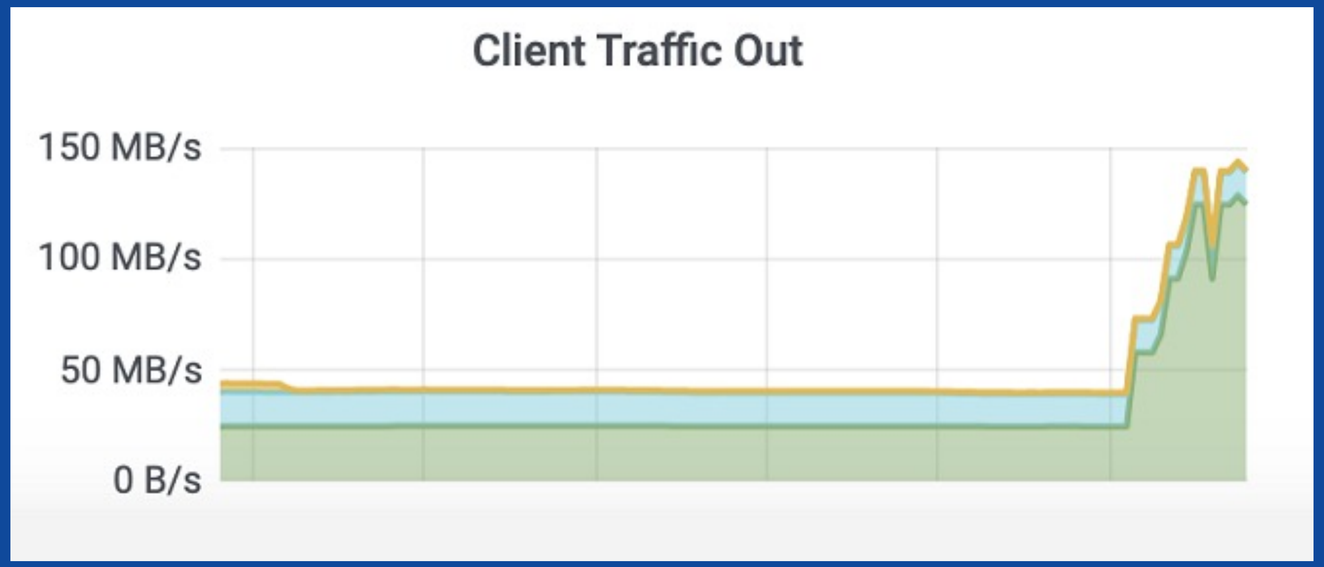

etcd常见故障
OOM
大key-value
Range大量的key-value
接口延时、失败率高
Range操作导致etcd节点cpu、memory使用率高
节点磁盘性能差
节点流量出现瓶颈
碎片整理导致
备份操作导致
etcd db满
突发大量写入
未配置compact策略
compact策略不合理
Watch事件推送缓慢
etcd cpu/内存使用率高，出现流量瓶颈等
client所在节点是否存在cpu、流量瓶颈等
限速错误(too many request)
突发写入
磁盘io性能差
数据不一致
auth revision
defrag
etcd节点故障
触发client负载均衡bug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

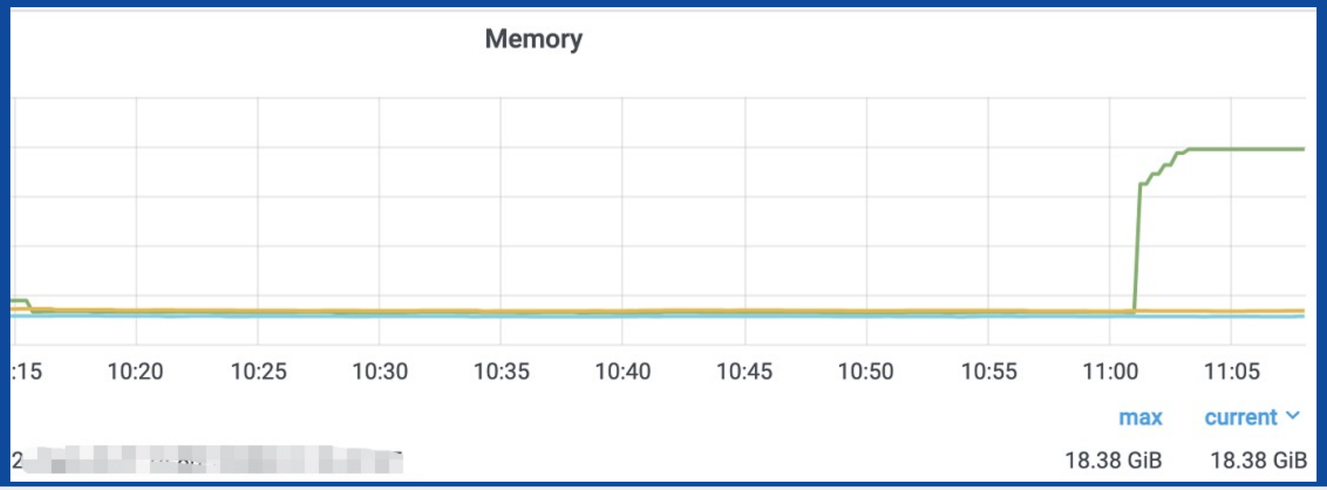
搞崩etcd典型案例分析–event大量写入



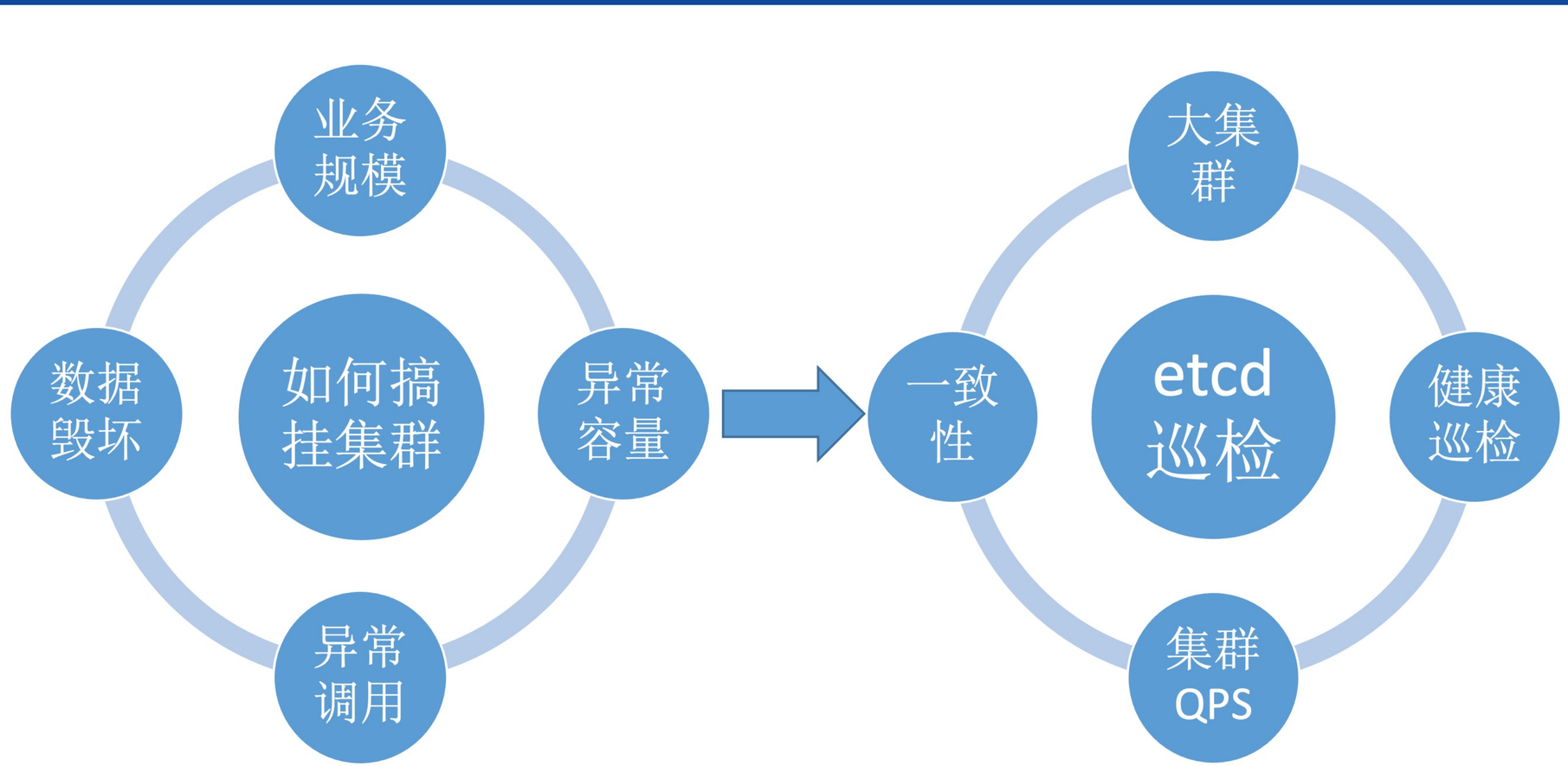


搞崩etcd典型案例分析–大量list查询，流量内存暴涨



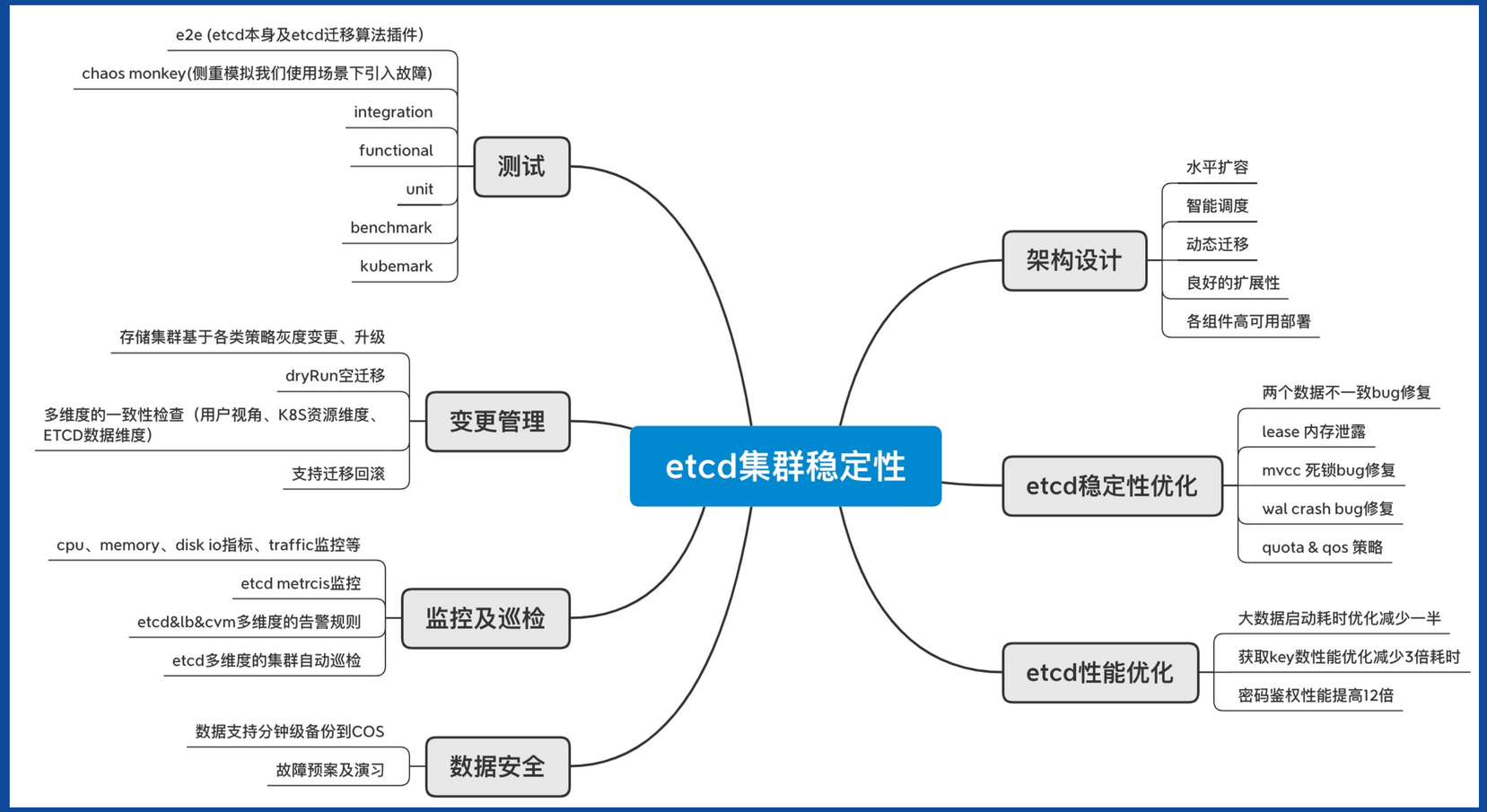


如何发现更多影响etcd稳定性的隐患呢？– 多维度集群巡检



自动化巡检显著降低现网隐患，让工作更加快乐轻松

如何保障及提高etcd稳定性? — 全景图



大规模etcd集群治理

大规模etcd集群治理挑战–业务背景

大规模的etcd集群

对外提供etcd产品化服务，支撑腾讯和斗鱼、微盟等公司的网关、注册中心等场景

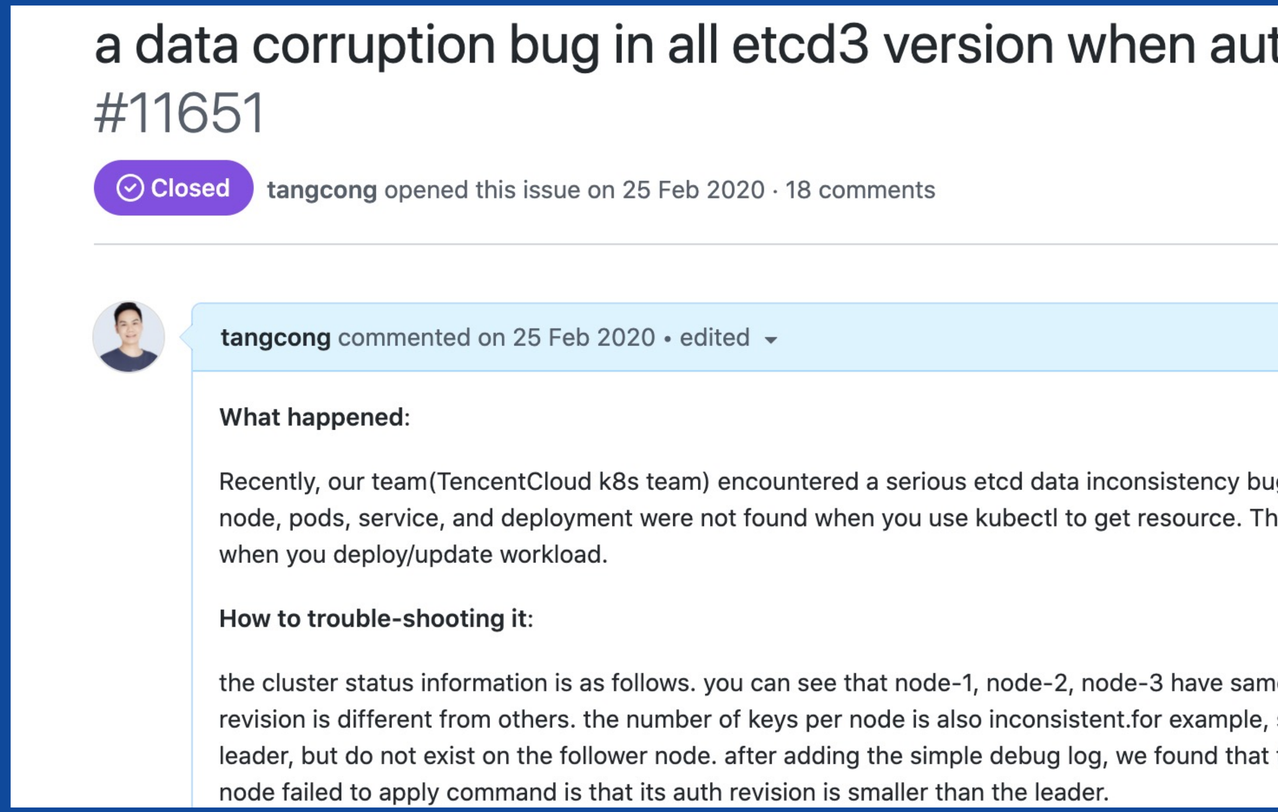
数万的k8s全托管和独立集群，总集群规模核数千万级

上百万开发者和企业用户使用腾讯云的TKE、EKS、EDGE、Mesh等云原生产品服务

大规模etcd集群治理挑战–运维

大规模etcd集群治理挑战–内核

- 数据不一致
- 内存泄露、死锁等
- 性能
- QoS
- 跨城容灾



a data corruption bug in all etcd3 version when aut
#11651

Closed **tangcong** opened this issue on 25 Feb 2020 · 18 comments

tangcong commented on 25 Feb 2020 • edited

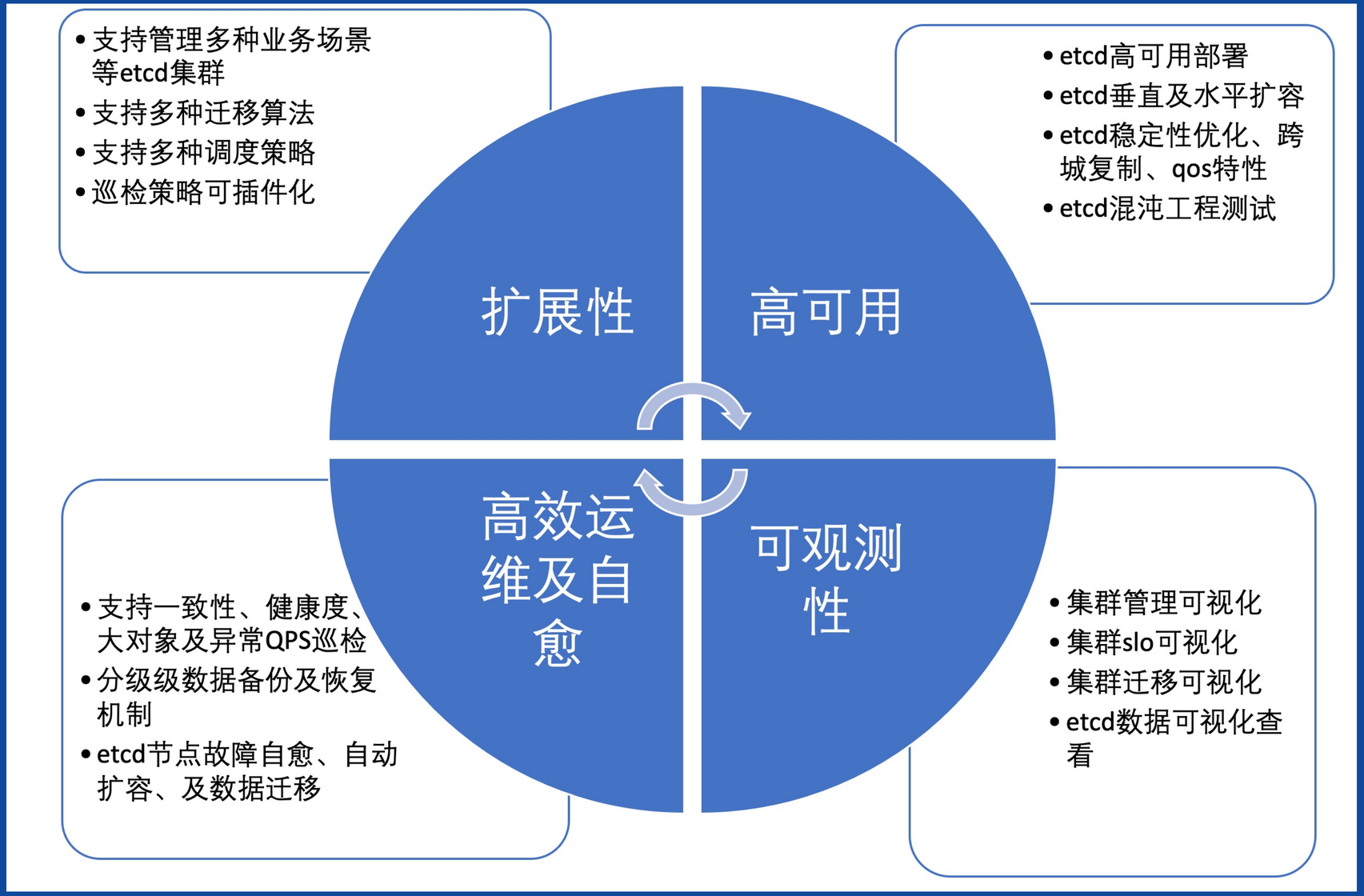
What happened:

Recently, our team(TencentCloud k8s team) encountered a serious etcd data inconsistency bu
node, pods, service, and deployment were not found when you use kubectl to get resource. Th
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 sam
revision is different from others. the number of keys per node is also inconsistent.for example,
leader, but do not exist on the follower node. after adding the simple debug log, we found that
node failed to apply command is that its auth revision is smaller than the leader.

设计目标

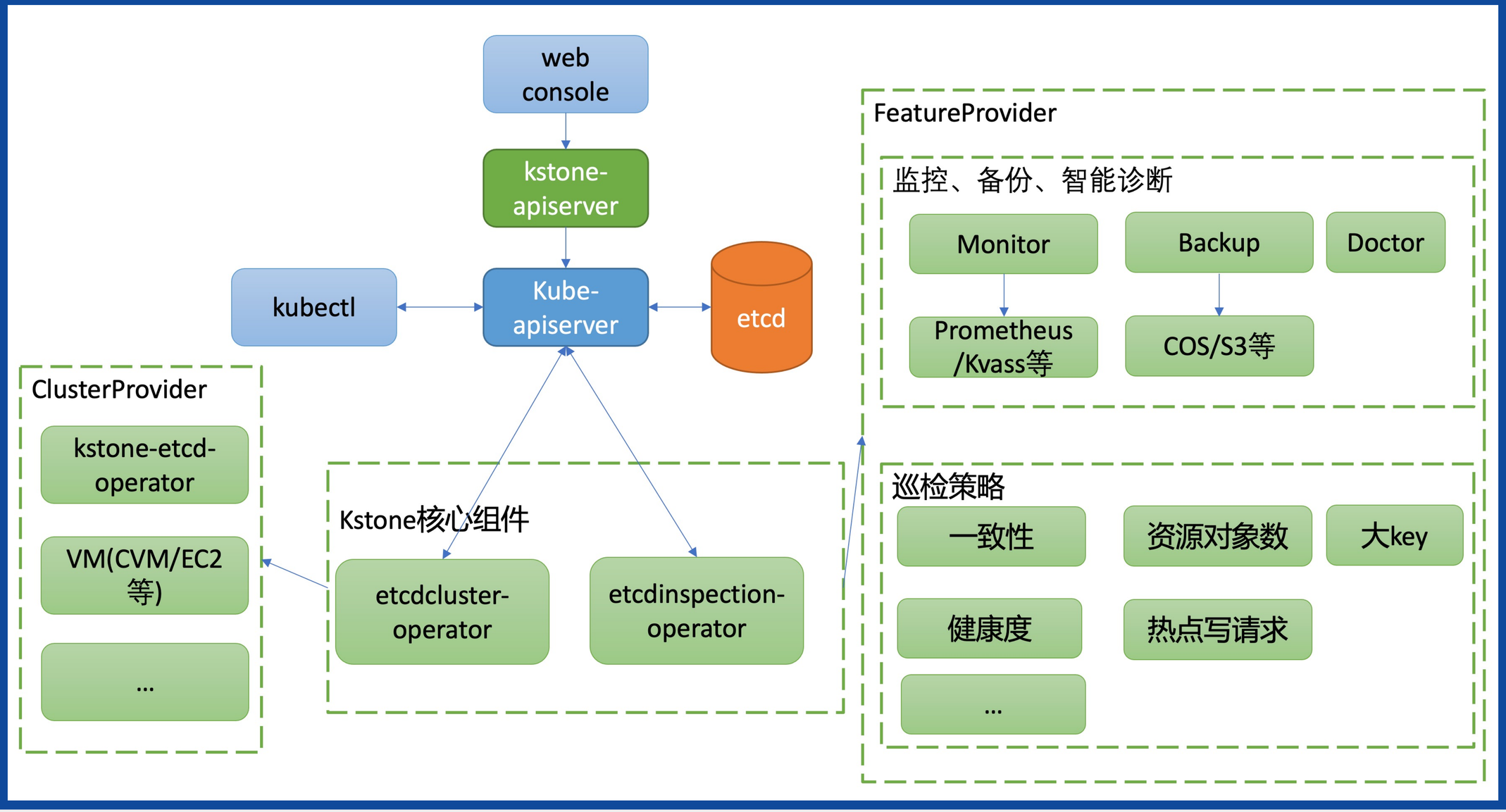

• 支持管理多种业务场景等etcd集群
• 支持多种迁移算法
• 支持多种调度策略
• 巡检策略可插件化
• etcd高可用部署
• etcd垂直及水平扩容
• etcd稳定性优化、跨城复制、qos特性
• etcd混沌工程测试
扩展性
高可用
高效运维及自愈
可观测性
• 支持一致性、健康度、大对象及异常QPS巡检
• 分级级数据备份及恢复机制
• etcd节点故障自愈、自动扩容、及数据迁移
• 集群管理可视化
• 集群slo可视化
• 集群迁移可视化
• etcd数据可视化查看

方案选型

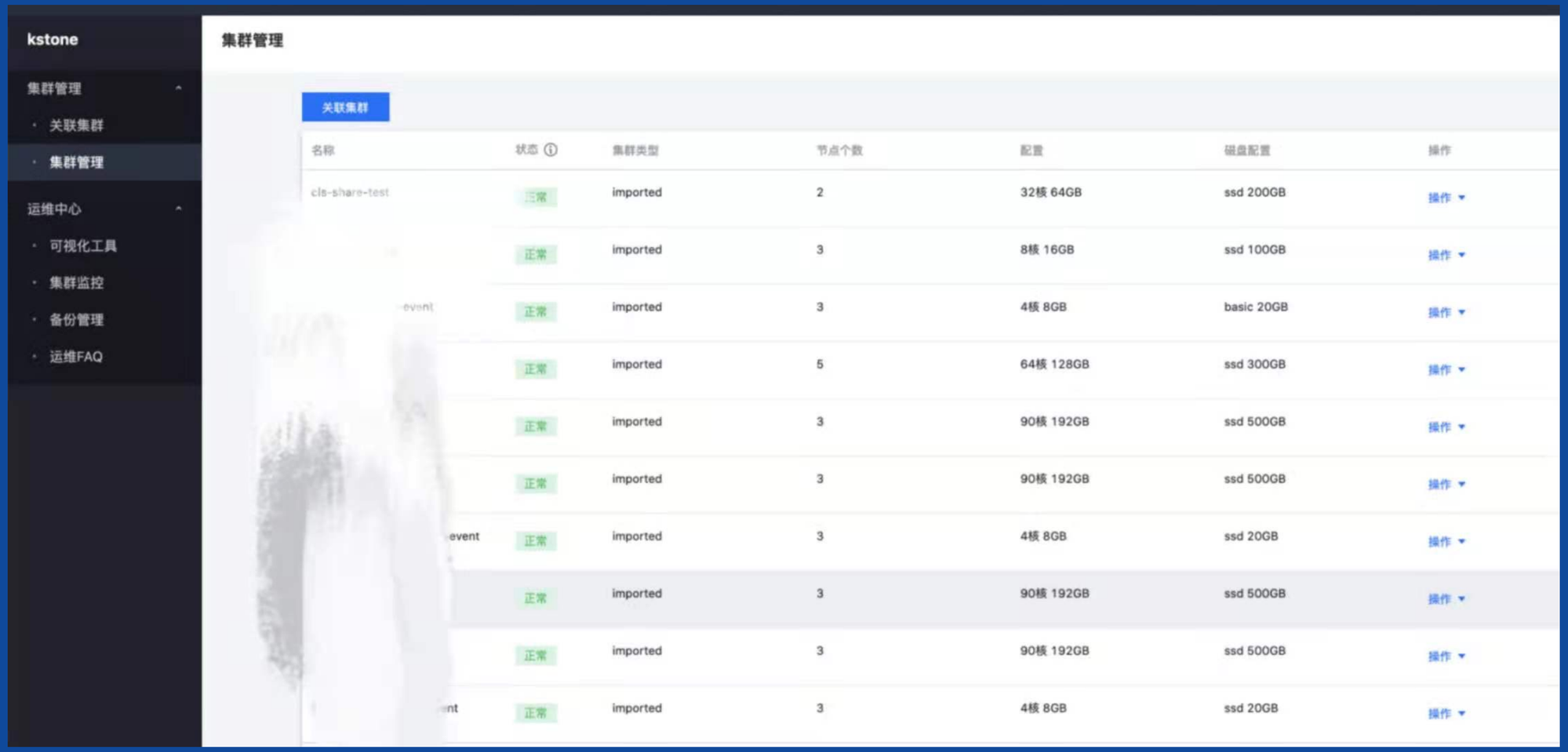
比较项\技术方案	原生HTTP服务	RPC框架	Kubernetes扩展机制
开发效率	低	较高	高
可观测性	自建	高	高/event/kubectl
可运维性	一般	一般	高,operator机制

机制\特性	支持kubectl/kubernetes UI	支持Custom Storage	是否需要运行Additional ApiServer	支持Custorom Advanced Feature	Pick Up BugFix
CRD	YES	NO	NO	NO	NO
Aggregated API	YES	YES	YES	YES(logs/exec,protobuf)	YES

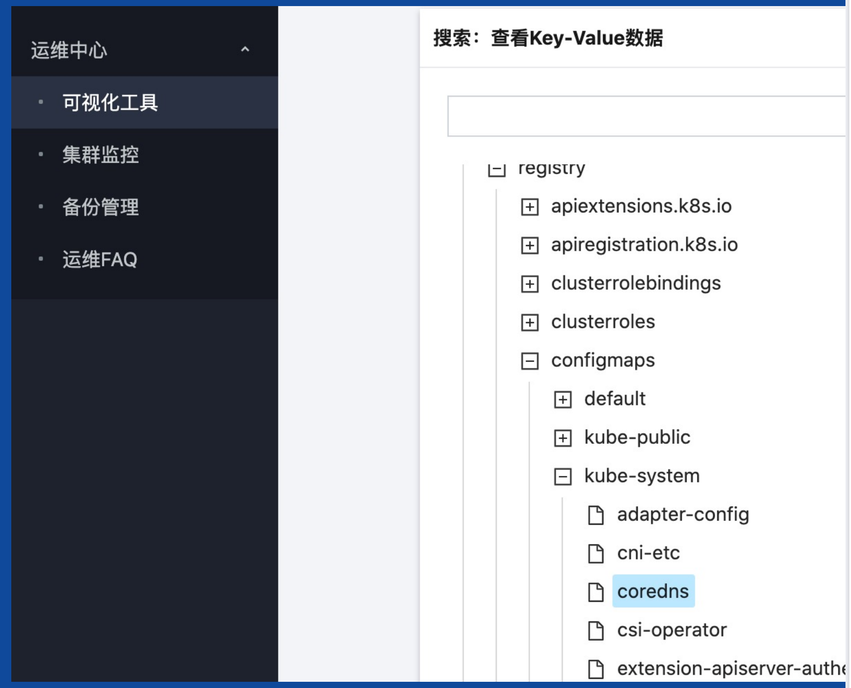
kstone架构

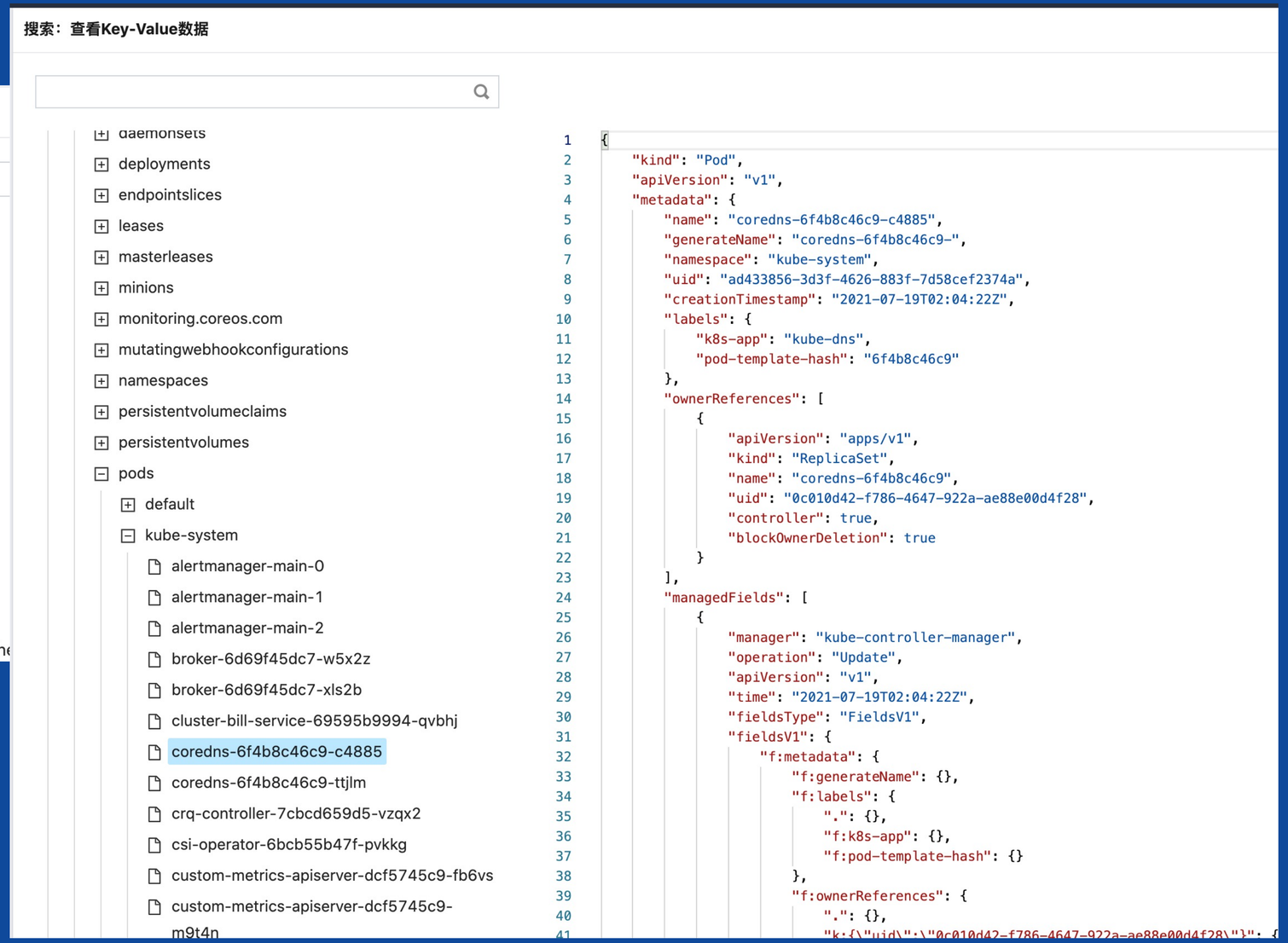

web console
kstone-apiserver
kubectl
Kube-apiserver
etcd
FeatureProvider
监控、备份、智能诊断
Monitor
Backup
Doctor
Prometheus /Kvass等
COS/S3等
ClusterProvider
kstone-etcd-operator
VM(CVM/EC2等)
...
Kstone核心组件
etcdcluster-operator
etcdinspection-operator
巡检策略
一致性
资源对象数
大key
健康度
热点写请求
...

可视化etcd管理平台

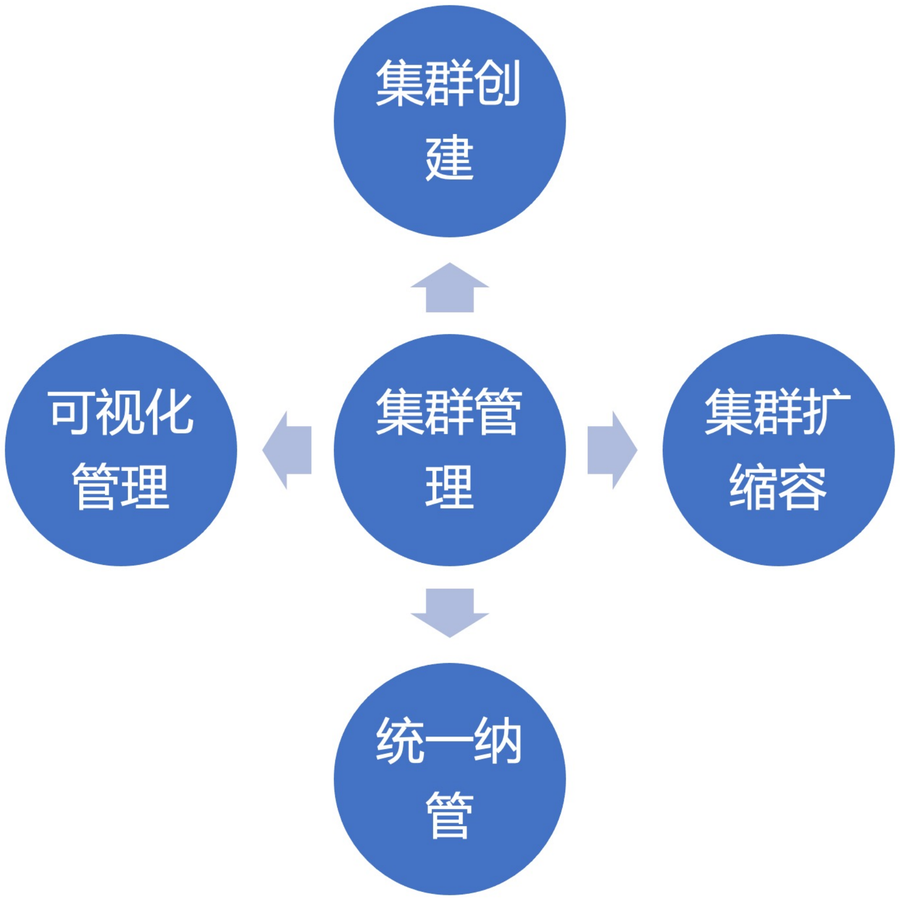


可视化查看k8s对象


运维中心
可视化工具
集群监控
备份管理
运维FAQ
搜索：查看Key-Value数据
registry
apiextensions.k8s.io
apiregistration.k8s.io
clusterrolebindings
clusterroles
configmaps
default
kube-public
kube-system
adapter-config
cni-etc
coredns
csi-operator


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": {
".": {},

基于operator的自动化集群管理



新建Etcd集群

集群名称 gz-tke-vpc-etcd-

标签 请选择 =

地域 广州(华南地区)

所属业务 tke

集群类型 kstone-etcd-operator cvm

集群版本 v3.3 v3.4 v3.5.1-rc.2+qos

集群资源类型 main

节点个数 3 个

CPU 8 核

内存 32 GB

磁盘类型 SSD云硬盘 普通云硬盘 高性能云硬盘

磁盘大小 50 GB

是否独占

是否开启https

Kstone CRD资源之EtcdCluster Anno/Spec

- etcd集群/EtcdCluster，如右图，
- 支持多种clusterType,VM集群、容器化集群、存量集群等
- 通过Anotation控制多个Feature开关，一键开启监控、备份、健康巡检、一致性检查等特性

```yaml
apiVersion: kstone.tkestack.io/v1alpha1
kind: EtcdCluster
metadata:
  annotations:
    certName: kstone/test
    featureGates: monitor=true,consistency=true,healthy=true,request=true,backup=false
    importedAddr: https://1.2.3.4:12846
    kubernetes: 'true'
  labels:
    clusterType: imported
    etcdName: test
  name: test
  namespace: kstone
spec:
  affinity: {}
  authConfig: {}
  clusterType: imported
  description: test
  diskSize: 50
  diskType: ssd
  name: test
  size: 3
  totalCpu: 4
  totalMem: 8
```

Kstone CRD资源之EtcdCluster Status

- Status.FeatureGates，显示各个特性开通状态
- Status.Member显示各个成员节详细信息

```yaml
status:
  featureGatesStatus:
    consistency: done
    healthy: done
    monitor: done
    request: done
  members:
    - clientUrl: https://1.2.3.4:2379
      endpoint: 1.2.3.4
      extensionClientUrl: https://1.2.3.9:12730
      memberId: '71412298143718490'
      name: 1.2.3.4
      port: '2379'
      role: Follower
      status: Running
      version: 3.4.9
    - clientUrl: https://1.2.3.5:2379
      endpoint: 1.2.3.5
      extensionClientUrl: https://1.2.3.9:12861
      memberId: '8991984052334821064'
      name: 1.2.3.5
      port: '2379'
      role: Leader
      status: Running
      version: 3.4.9
    - clientUrl: https://1.2.3.6:2379
      endpoint: 1.2.3.6
      extensionClientUrl: https://1.2.3.9:12105
      memberId: '17985372337265212585'
      name: 1.2.3.6
      port: '2379'
      role: Follower
      status: Running
      version: 3.4.9
  phase: Running
```

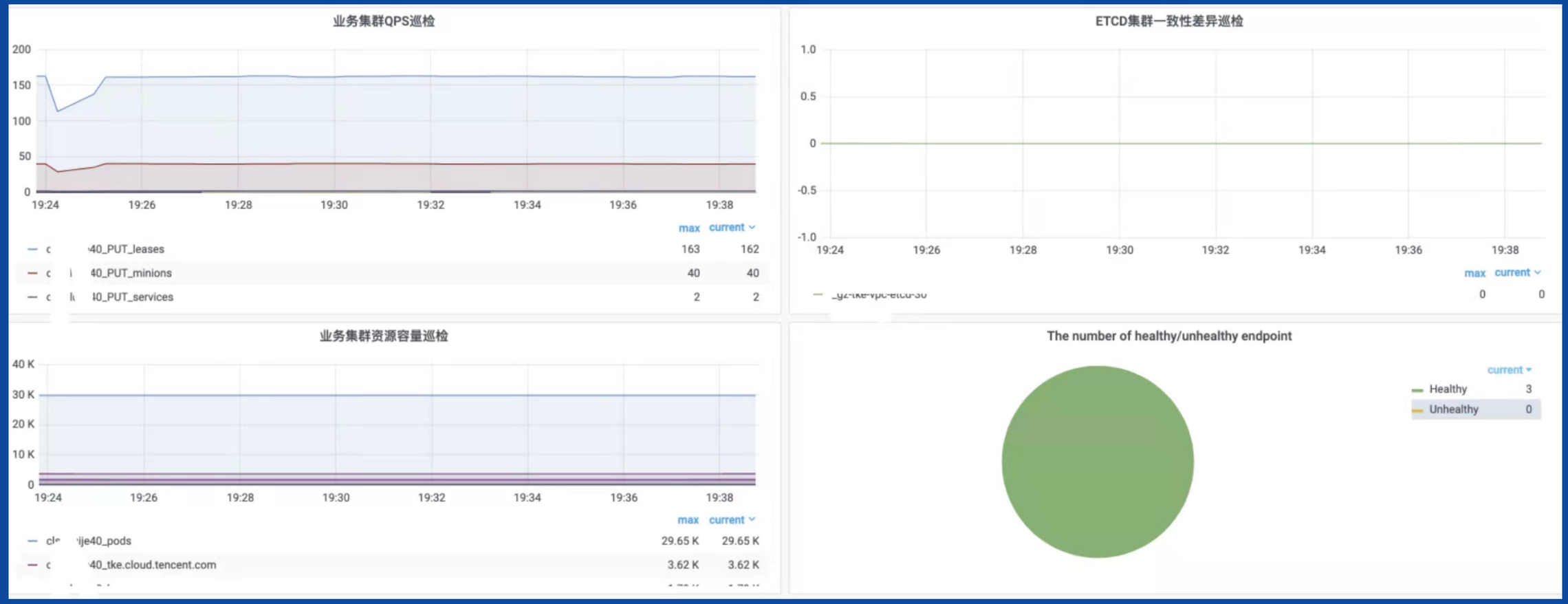
Kstone CRD资源之EtcdInspection

- Etcd巡检任务/EtcdInspection，如右图

- 支持多种巡检策略(监控、备份、健康巡检、一致性检查)，各个巡检策略支持多种后端Prvoider实现


```yaml
apiVersion: kstone.tkestack.io/v1alpha1
kind: EtcdInspection
metadata:
  labels:
    clusterType: imported
    etcdName: test
  name: test-healthy
  namespace: kstone
  ownerReferences:
    - apiVersion: kstone.tkestack.io/v1alpha1
      kind: EtcdCluster
      name: test
      uid: 9e9ddd34-9c92-4aee-bc3d-68d1a223cd13
spec:
  clusterId: test
  inspectIntervalInSecond: 0
  inspectionName: healthy
  provider: default
  name: test-healthy
```

巡检视图


业务集群QPS巡检
200
150
100
50
0
19:24
19:26
19:28
19:30
19:32
19:34
19:36
19:38
max
current
40_PUT_leases
163
162
40_PUT_minions
40
40
40_PUT_services
2
2
ETCD集群一致性差异巡检
1.0
0.5
0
-0.5
-1.0
19:24
19:26
19:28
19:30
19:32
19:34
19:36
19:38
max
current
0
0
业务集群资源容量巡检
40 K
30 K
20 K
10 K
0
19:24
19:26
19:28
19:30
19:32
19:34
19:36
19:38
max
current
29.65 K
29.65 K
40_tke.cloud.tencent.com
3.62 K
3.62 K
The number of healthy/unhealthy endpoint
current
Healthy
3
Unhealthy
0

kstone迁移任务实现

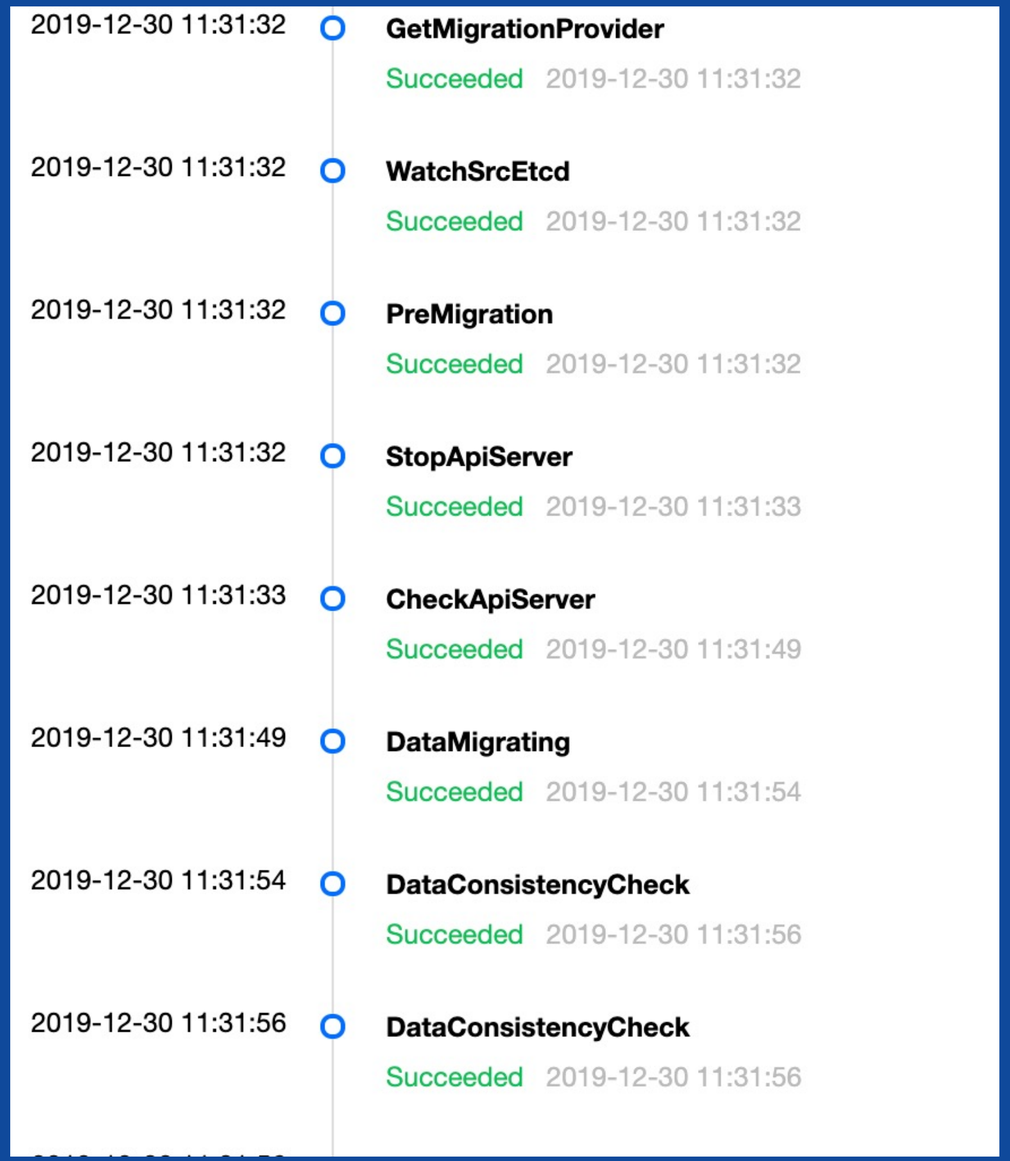
- 新增EtcdMigration迁移任务CRD，描述迁移源etcd、目的etcd集群、迁移算法、一致性检测策略等。
- 迁移算法抽象化，支持多种Provider, 比如etcd v2->v3, v3->v3冷迁移，v3-v3热迁移。
- 支持多种维度的数据一致性检查策略，比如etcd维度的数据一致性检查，k8s应用层的资源对象一致性检查等。
- 针对k8s场景迁移etcd导致的client list-watch hang住问题（迁移后的etcd版本号(apiserver resource version)小于原有etcd，修改k8s版本源码，增加watch操作的timeout机制。

Kstone–etcd数据迁移任务可视化查看

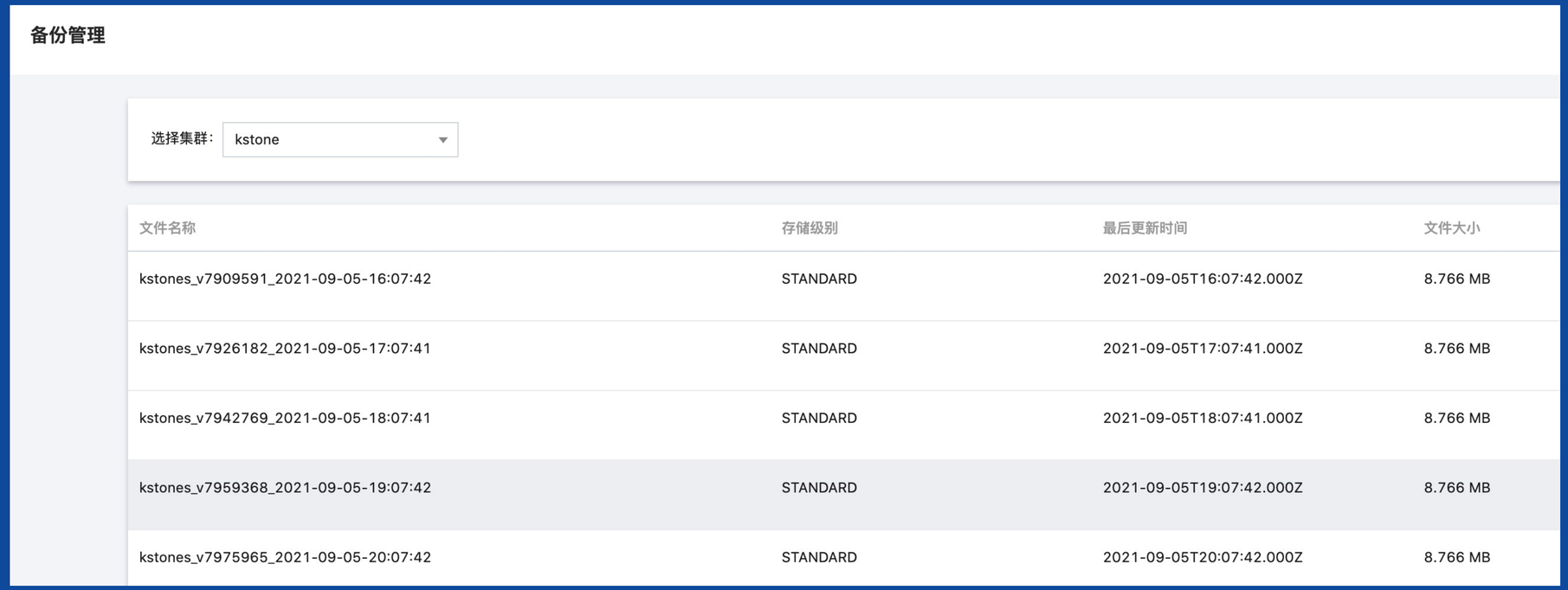
某业务 1178个节点的大规模集群快速迁移etcd效果(整个流程仅耗时34s)

迁移结果

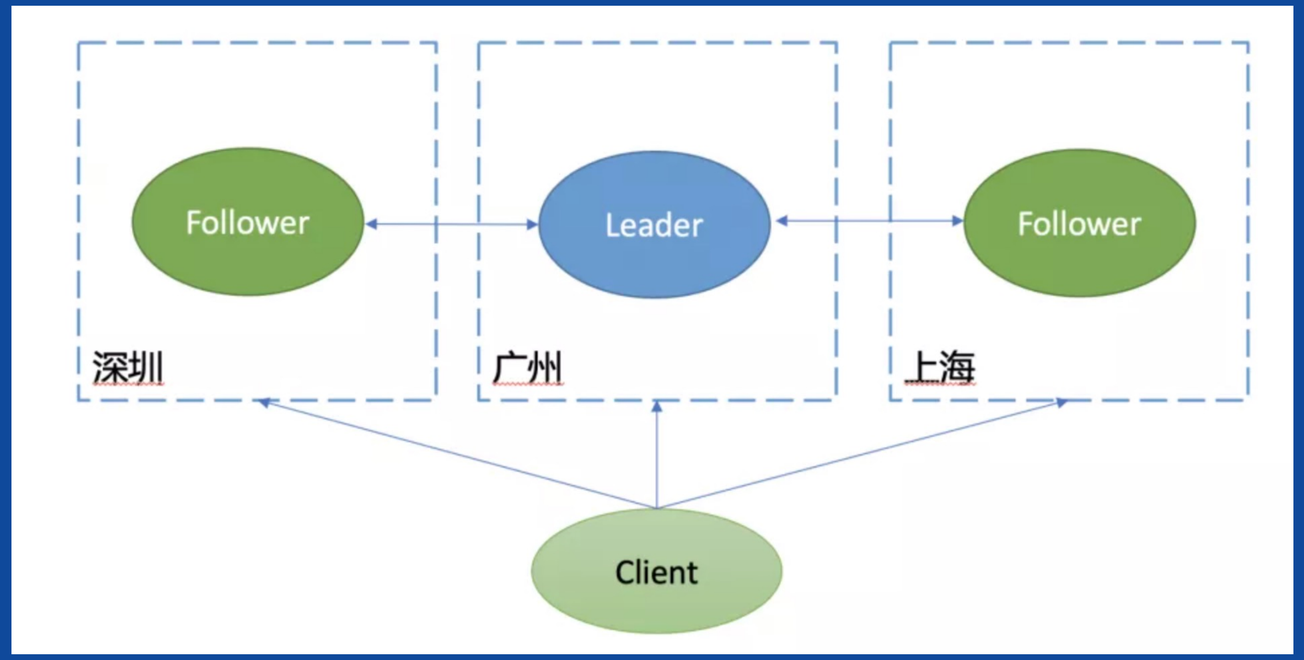
任务状态	Succeeded
节点数量	1178个（迁移前为1178个）
Service数量	3个（迁移前为3个）
Pod数量	19200个（迁移前为19200个）
开始时间	2019-12-30 11:31:25
结束时间	2019-12-30 11:31:59



数据冷备份–分钟级备份到cos等对象存储



跨城数据热备份–方案一



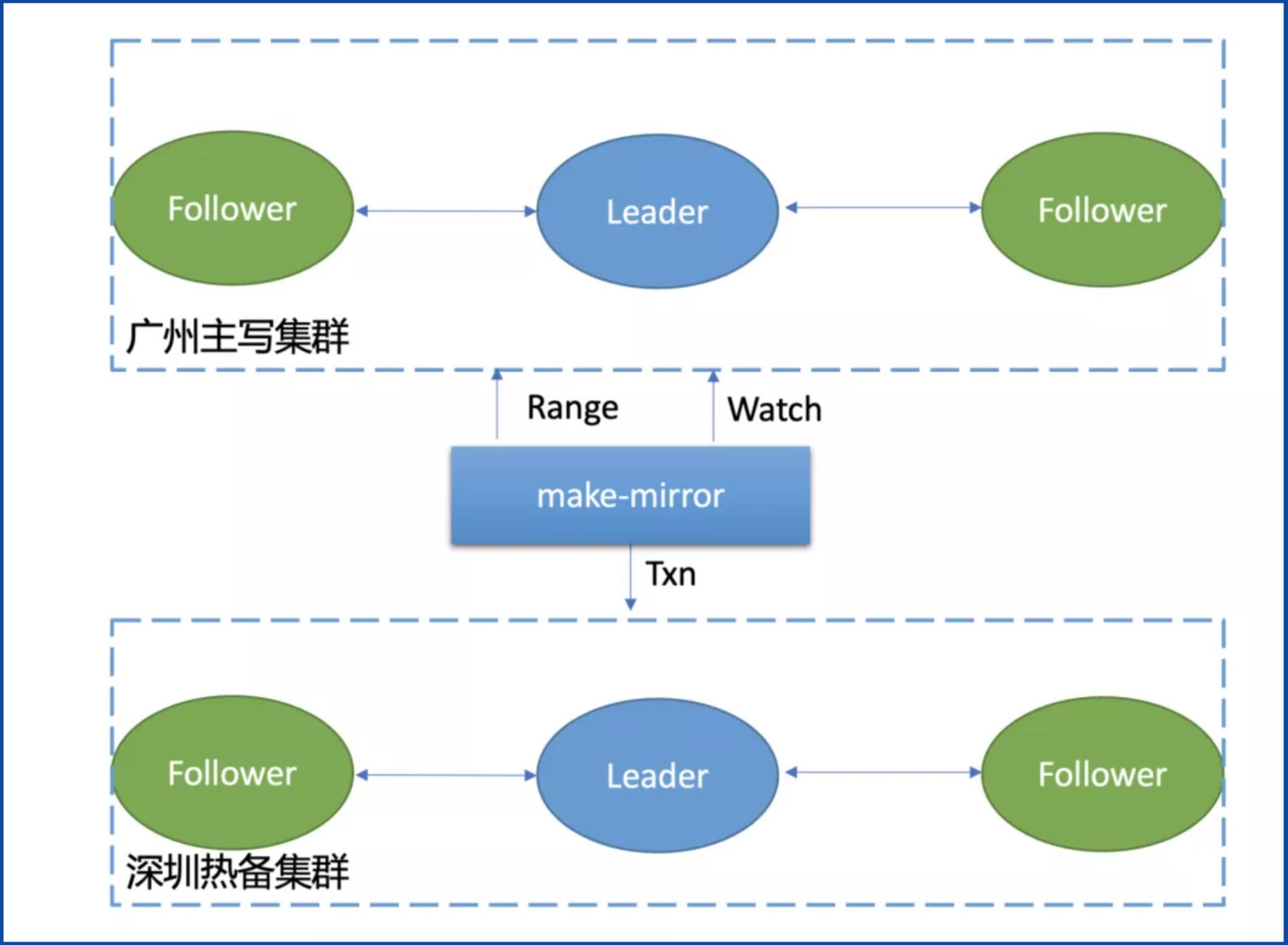
优点：

- 各地域网络互通后，部署简单，无需运维额外组件
- 数据跨城强一致同步，3 节点部署场景中，可容忍任一城市故障，并且不丢失任何数据

缺点：

- 在 3 节点部署的场景下，任意写请求至少需要两个节点应答确认，而不同节点部署在各地，ping 延时会从几毫秒上升到 30ms 左右（深圳 - 上海)，因此会导致写性能急剧下降。
- etcd 默认的读请求是线性读，当 Follower 节点收到 Client 发起的读请求后，它也需要向 Leader 节点获取相关信息，确认本地数据追赶上 Leader 后才能返回数据给 client，避免读取到旧数据等，在这过程中也会导致 etcd 读延时上升、吞吐量下降。
- 跨城网络质量抖动

跨城数据热备份–方案二社区make–mirror机制



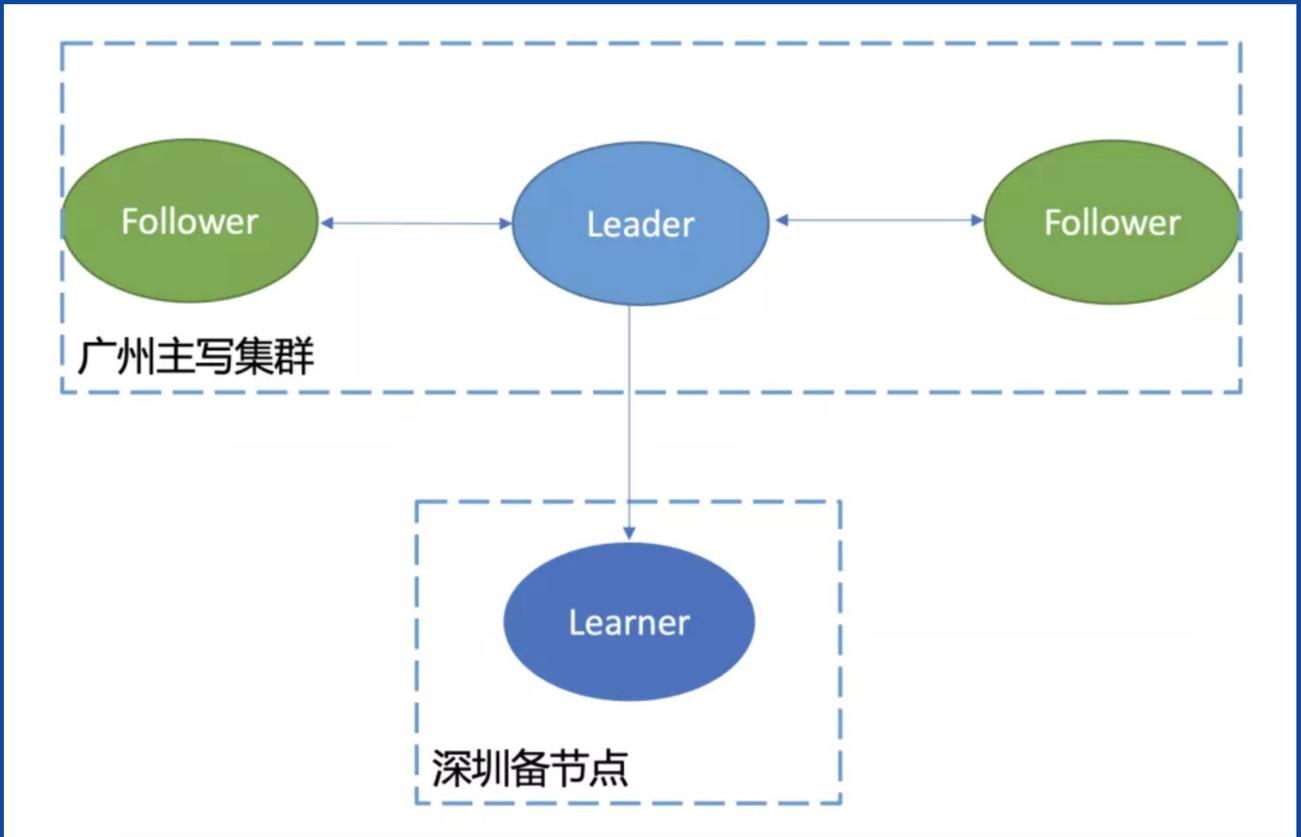
优点：

- 主 etcd 集群读写性能高，整体上不受跨地域网络延时、网络质量波动影响
- 若业务可容忍短暂不一致，可就近访问距离最近的 etcd 集群
- 不依赖高版本 etcd

缺点：

- 当写请求较大的时候，备集群可能存在一定的数据落后，可能读到脏数据。
- 社区自带的 make-mirror 同步链路中断后，退出重启会再次进入全量同步模式，性能较差，无法满足生产环境诉求。
- 社区自带的 make-mirror 工具缺少 leader 选举、数据一致性检测、日志、metrics 等一系列特性，不具备生产环境可用性。
- 不支持同步非 key-value 数据，如 auth 鉴权相关数据、lease 数据等。

跨城数据热备份–方案三learner



优点：

- 各地域网络互通后，部署简单，只需往 etcd 集群中添加一个 Learner 节点，无需运维额外组件
- Learner 节点可同步任意类型数据，如 key-value、 auth 鉴权数据、 lease 数据

缺点：

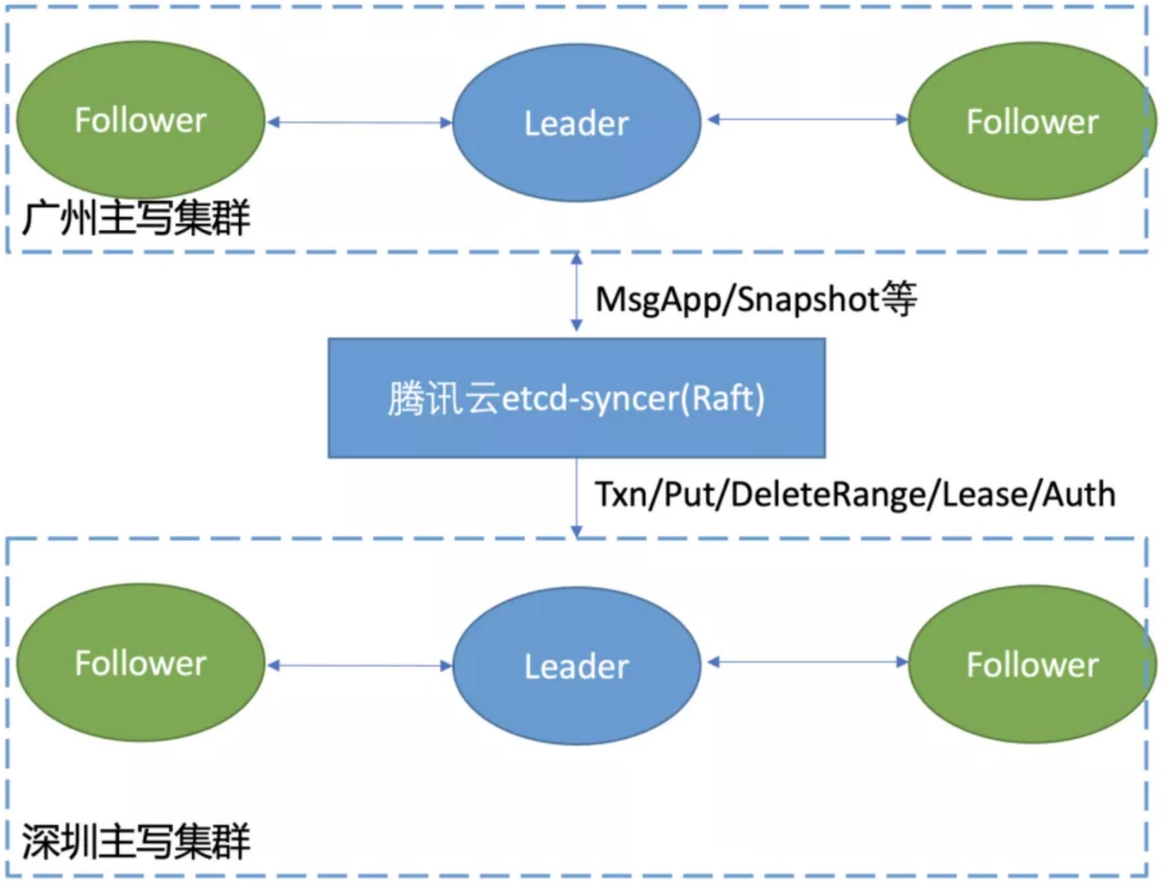
- Learner 节点只允许串行读，也就是业务如果就近读，会读到旧数据。
- 依赖高版本 etcd，etcd 3.4 及以上版本才支持 Learner 特性，并且只允许一个 Learner 节点 .
- 主集群全面故障后，无法快速将 Learner 节点提升为可写的独立 etcd 集群。

跨城数据热备份–etcd同步服务mirror–plus版本

mirror-plus方案make-mirror 的加强版，为了解决 make-mirror 的各种缺陷，它实现了以下特性、优点:

- 支持多种同步模式，全量同步、断点续传，不再担忧专线、公网网络质量抖动
- 高可用，负责同一数据路径复制的实例支持多副本部署, 一副本故障后，其他副本将在 5 秒后获得锁，在之前实例同步的进度基础上，进行快速恢复
- 支持一致性检查（全量数据检查、快照检查)
- 支持多实例并发复制提升性能（不同实例负责不同的路径），建议生产环境配置多实例，每个实例负责不同路径
- 良好的运维能力，基于 k8s deployment 一键部署，丰富的 metrics、日志，完备的 e2e 测试用例覆盖核心场景（http/https 场景，服务异常中断、网络异常等 ）

跨城数据热备份–etcd同步服务raft版本



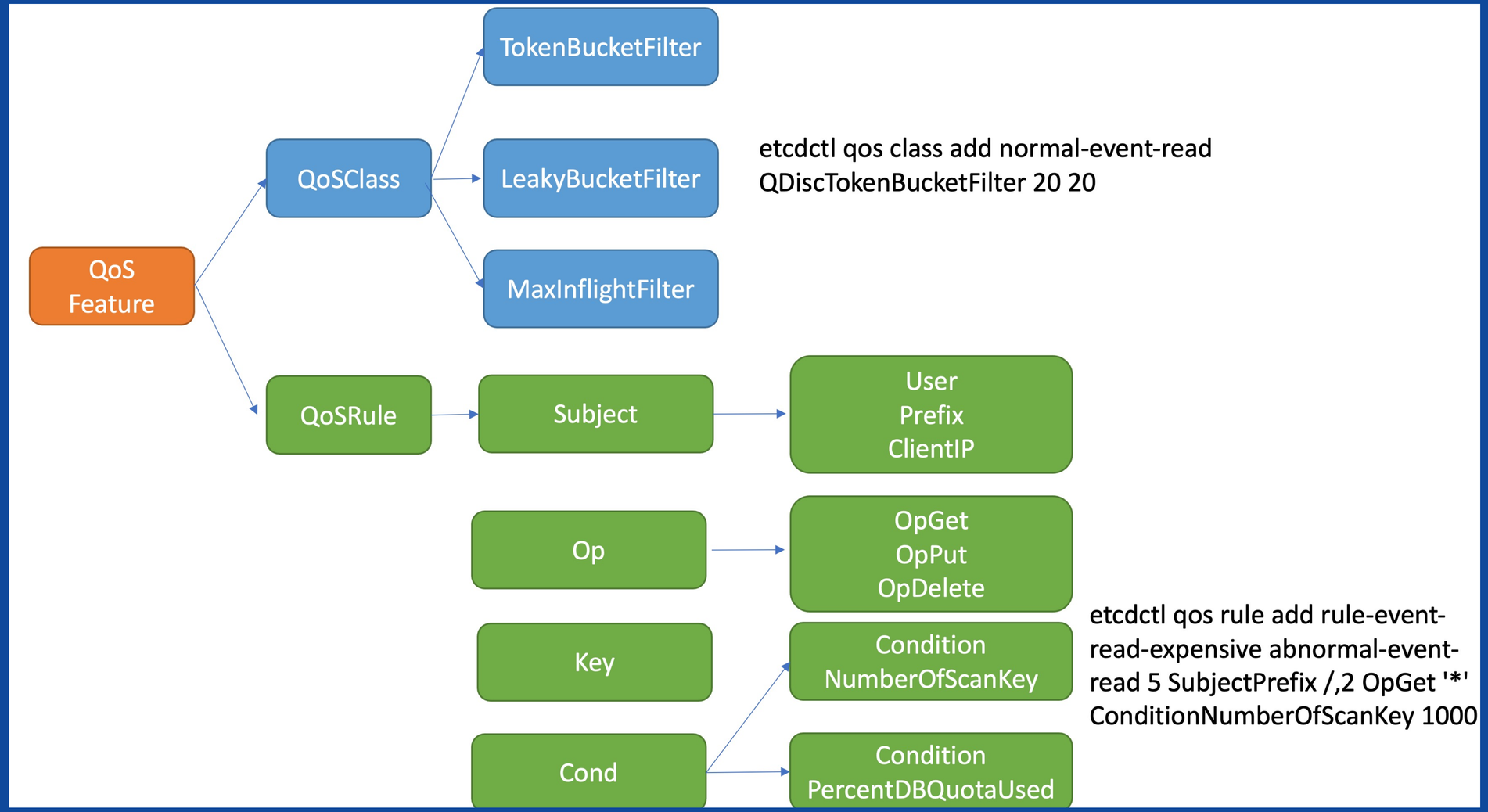
优点：

- 具备 etcd-syncer 之 mirror-plus 版本的所有特性和优点，同时不依赖 etcd mvcc 历史数据。
- 基于 etcd 底层的 Raft 日志同步数据，可以同步 key-value、auth、lease 等各种类型的数据。
- 不依赖高版本的 etcd。

缺点：

- 一定的开发时间。

etcd QoS特性–保障etcd集群稳定性



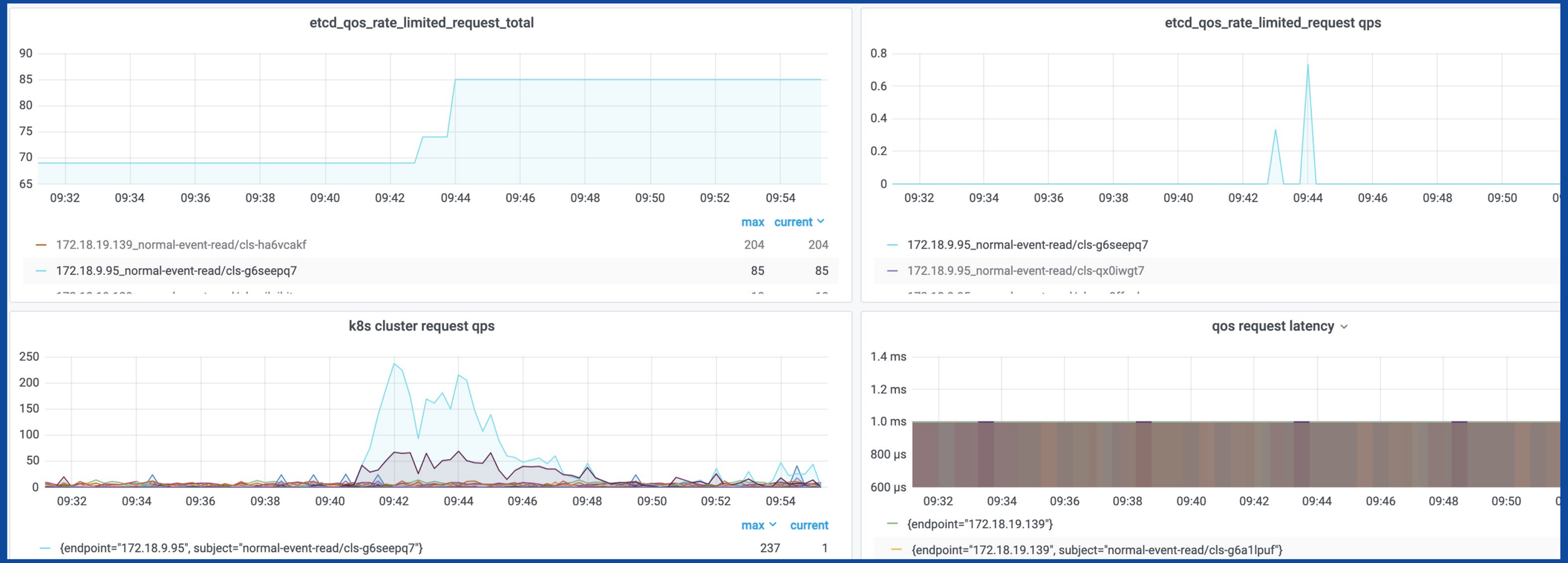
etcd QoS特性–event分场景设置限速器

- etcdctl qos class add normal-event-read QDiscTokenBucketFilter 20 20
- etcdctl qos class add normal-event-write QDiscTokenBucketFilter 20 20
- etcdctl qos class add abnormal-event-read QDiscTokenBucketFilter 1 1
- etcdctl qos class add abnormal-event-write QDiscTokenBucketFilter 1 1

案例–扫描key过多和db即将满异常场景严格限速

- etcdctl qos rule add rule-event-read normal-event-read 1 SubjectPrefix /,2 OpGet
- etcdctl qos rule add rule-event-write normal-event-write 1 SubjectPrefix /,2 OpPut,OpDelete
- etcdctl qos rule add rule-event-read-expensive abnormal-event-read 5 SubjectPrefix /,2 OpGet '*' ConditionNumberOfScanKey 1000
- etcdctl qos rule add rule-event-db-quota abnormal-event-write 8 SubjectPrefix /,2 OpPut '*' ConditionPercentDBQuotaUsed 90

etcd QoS特性–限速效果



Kubernetes master组件治理

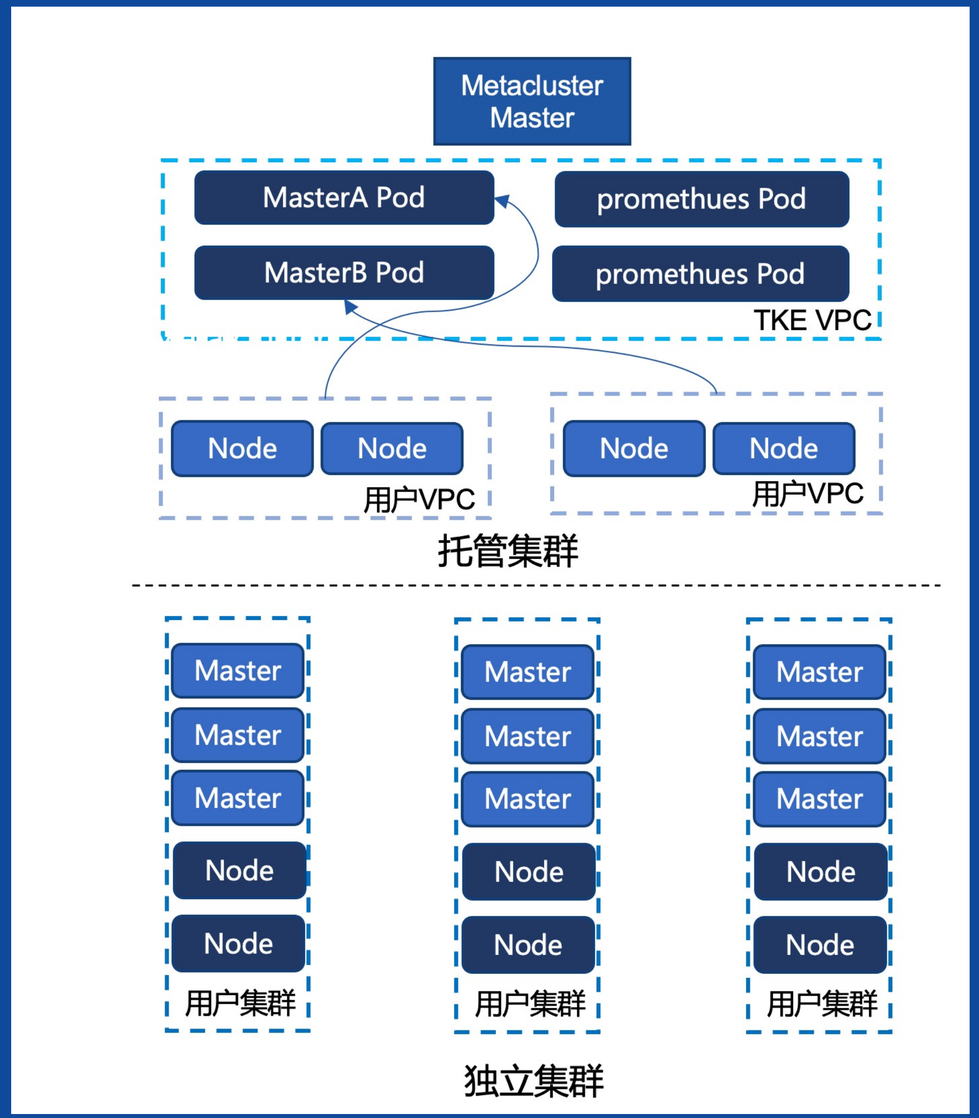
TKE 用户集群管理模式

两种管理模式

- Master 托管模式 —— 所有租户的托管集群的管理面（Master 、etcd ）全部作为服务由一个 Kubernetes 集群维护，稳定性更高、扩展更简单。适合一般业务或集群规模变化比较频繁的场景；
- 独立集群模式 —— 集群的管理面运行在集群本身的 IaaS 资源上，集群隔离性和独立性更好。适合对集群定制化更高的场景；

集群功能

- 自定义参数、多种运行时和发行版
- CVM、黑石节点、节点池、节点自动扩缩容
- 集群版本升级
- Addons机制



master组件治理挑战分析

- 如何解决数万集群的master组件运维问题(用户集群快速创建变更、节点故障等)
- 如何按需扩缩容，实现成本与高可用平衡？（高负载自动扩容、低负载根据画像缩容)
- 如何简化运维复杂度，实现高效运维?(管控集群节点安全下线、监控、巡检、变更)

大规模组件运维–k8s in k8s and master–operator

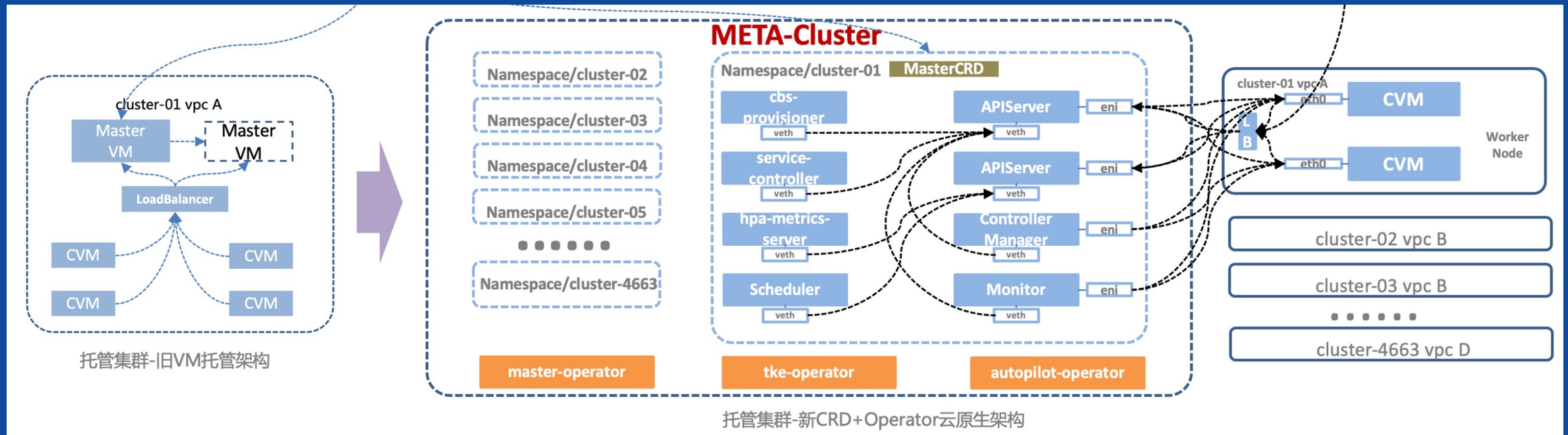
思路

k8s提供了完善的发布、自动扩缩容、监控等能力，是否可以用k8s来管理用户 k8s Master，根本性的解决k8s Master运维问题

Master部署与运维

- 利用k8s CRD机制编写插件 Master-operator
- 声明式API解决一致性问题
- 每个k8s集群在MetaCluster中占用一个namespace
- Deployment支持快速扩容、滚动升级和健康检查
- Pod Pending事件自动触发MetaCluster节点扩容

整体架构



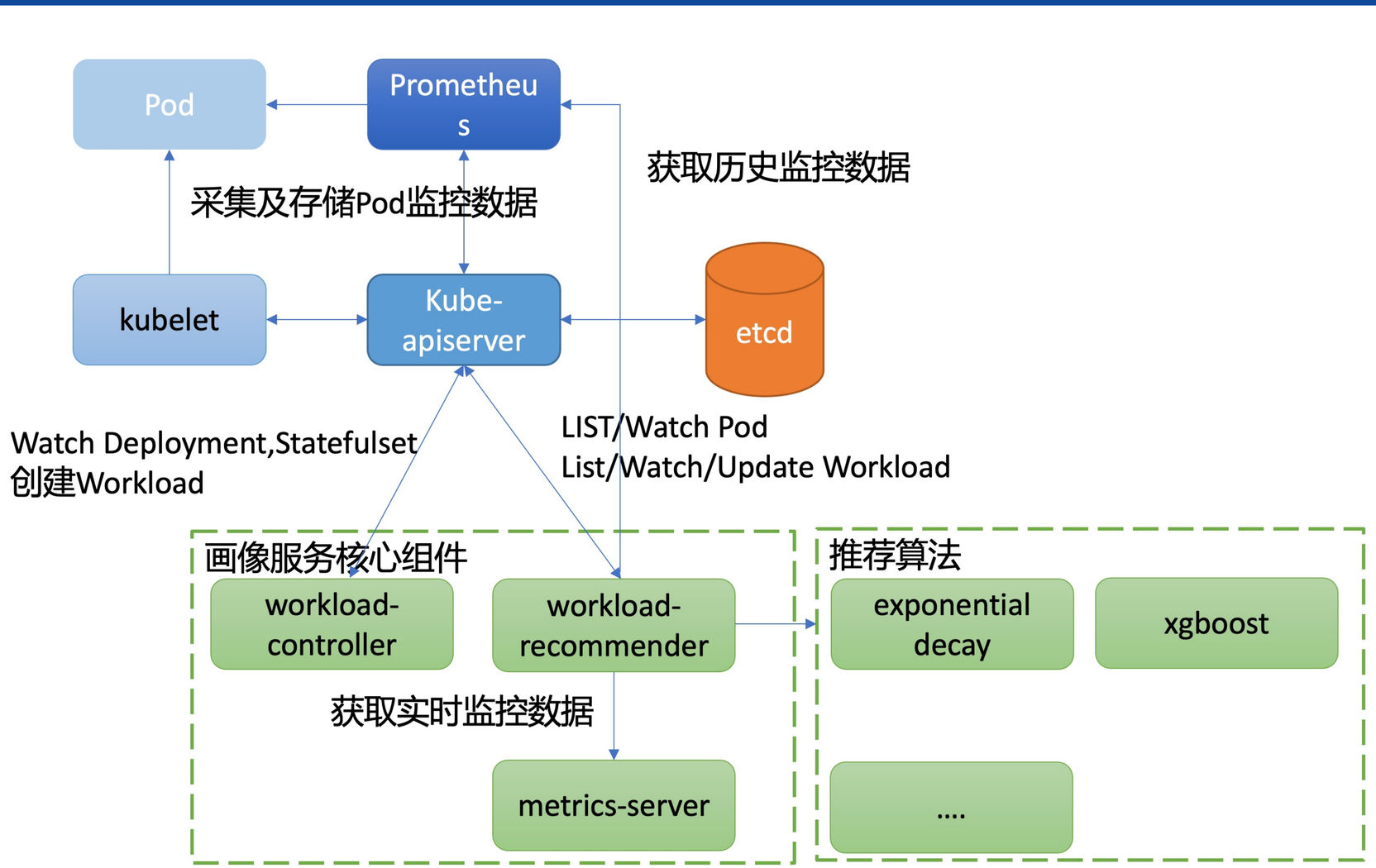
托管集群-旧VM托管架构

托管集群-新CRD+Operator云原生架构

元集群(META-Cluster)：

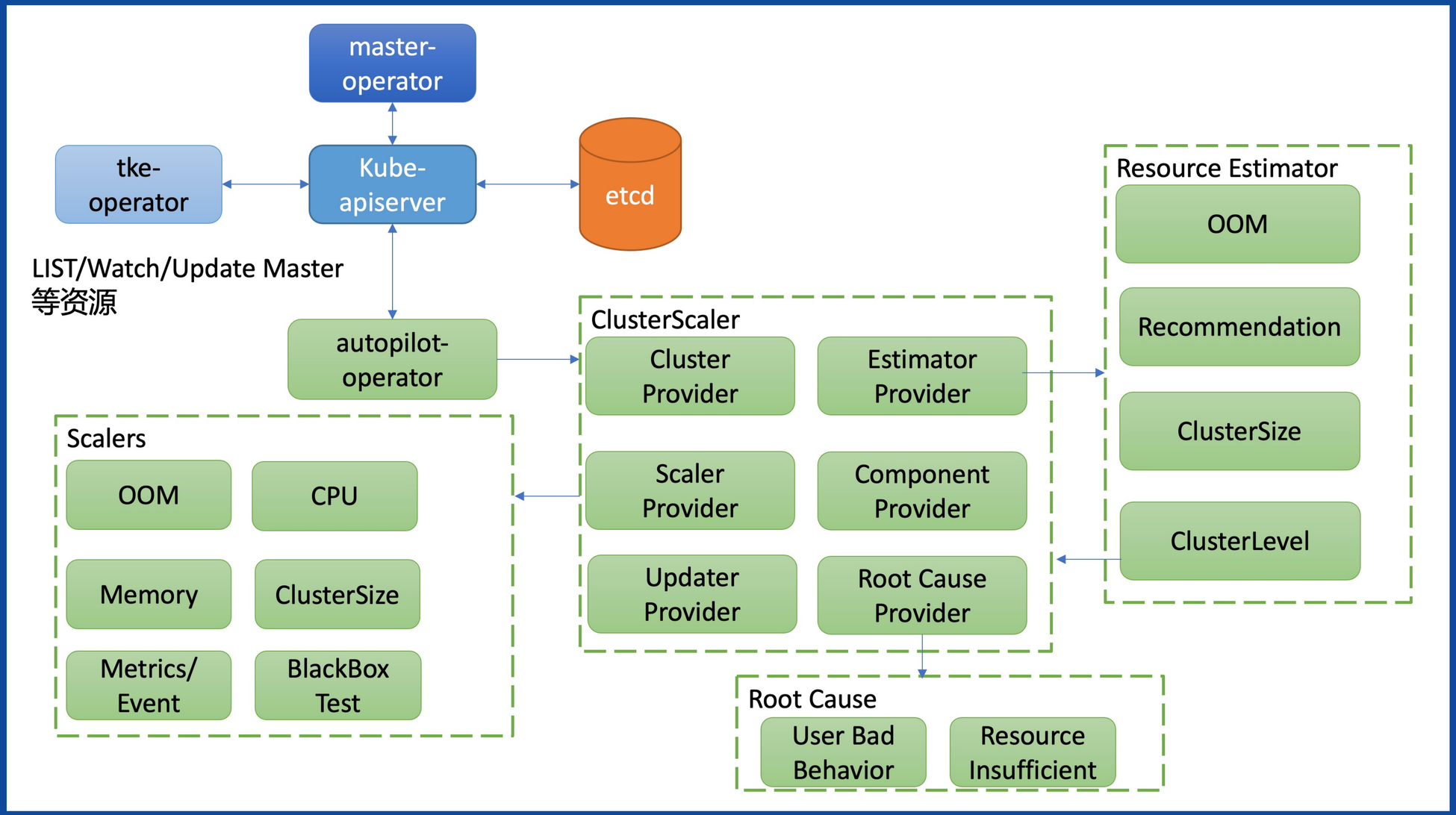
- 作用：专用于部署/管理用户托管Master组件的容器集群，只存在于云产品账号/VPC下；
- 优势：用 kubernetes 解决托管集群Master组件部署/扩容/故障恢复问题，用 operator 解决配置/升级/发布/自动扩缩容/运维等问题

可用性与成本优化–画像服务



- workload–controller负责deployment、statefulset等负载类型的画像CRD生成
- workload–recommender是基于vpa扩展开发的资源推荐组件，支持多种推荐算法，各个workload支持声明使用指定的推荐算法
- 数据源支持metrics–server、prometheus

可用性与成本优化–autopilot组件架构



- ClusterScaler CRD描述了集群各个组件扩缩容触发的事件源和指标、扩缩容冷静期、扩缩容开关等
- 支持多种scaler，比如常用的cpu和内存使用率、oom事件、集群大小等
- 支持多种resource estimator，如基于oom时的内存预估、画像服务推荐的资源配置、集群大小等

可用性与成本优化–基于真实负载的自动缩容

```yaml
apiVersion: autopilot.cloud.tencent.com/v1beta1
kind: ClusterScaler
metadata:
  annotations:
    cloud.tencent.com/tke-cluster-appid: "1233456"
  labels:
    clusterid: cls-123
  name: cls-123
  namespace: cls-123
spec:
  appId: 1233456
  clusterId: cls-123
  clusterLevel: L1
  componentScalerPolicy:
    - hpaPolicy: null
      targetRef:
        name: cls-123-cloud-controller-manager
      vpaPolicy:
        containerPolicies:
          - containerName: cloud-controller-manager
            scaleDownMode: Auto
            scaleUpMode: Auto
        scaleDownStabWindowSeconds: 600
        scaleUpStabWindowSeconds: 300
        scalers:
          scaleDown:
            - LowResourceUtilization
          scaleUp:
            - HighResourceUtilization
            - OOMEvent
    - hpaPolicy:
        maxReplicas: 3
        minReplicas: 1
        targetCPUUtilizationPercentage: 75
        targetRef:
          name: cls-123-apiserver
        vpaPolicy:
```

```yaml
status:
  conditions:
    - component: cloud-controller-manager
      count: 1
      lastEndTime: "2021-11-04T11:36:19Z"
      lastStartTime: "2021-11-04T11:36:19Z"
      message: update master succ
      reason: VPAScaleDownSucc
      status: "True"
      triggers:
        - LowResourceUtilization
      type: VPAScaleDown
    - component: service-controller
      count: 1
      lastEndTime: "2021-11-04T11:37:46Z"
      lastStartTime: "2021-11-04T11:37:46Z"
      message: update master succ
      reason: VPAScaleDownSucc
      status: "True"
      triggers:
        - LowResourceUtilization
      type: VPAScaleDown
    - component: apiserver
      count: 1
      lastEndTime: "2021-11-04T11:38:48Z"
      lastStartTime: "2021-11-04T11:38:48Z"
      message: update master succ
      reason: VPAScaleDownSucc
      status: "True"
      triggers:
        - LowResourceUtilization
```

- 扩缩容过程可视化、结果持久化
- 支持dry run
- 支持按用户、标签、组件等配置不同缩容阈值
- 数千集群存在低负载
- 通过autopilot组件每年节省大量成本费用

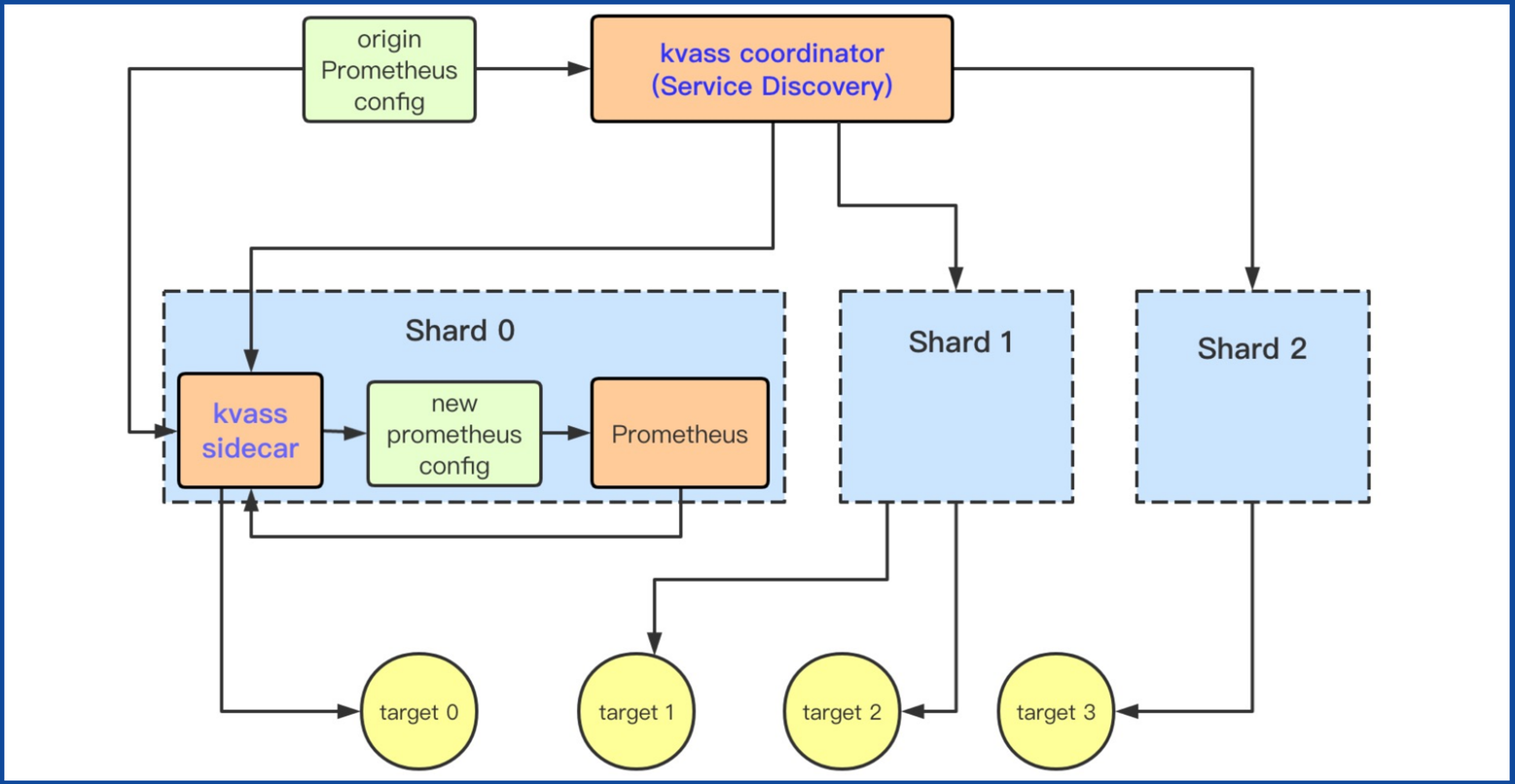
可用性与成本优化–基于OOM事件快速自动扩容

```yaml
apiVersion: autopilot.cloud.tencent.com/v1beta1
kind: ClusterScaler
metadata:
  annotations:
    cloud.tencent.com/tke-cluster-appid: "1233456"
  labels:
    clusterid: cls-123
  name: cls-123
  namespace: cls-123
spec:
  appId: 1233456
  clusterId: cls-123
  clusterLevel: L1
  componentScalerPolicy:
    - hpaPolicy:
        maxReplicas: 3
        minReplicas: 1
        targetCPUUtilizationPercentage: 75
        targetRef:
          name: cls-123-apiserver
        vpaPolicy:
          containerPolicies:
            - containerName: apiserver
              scaleDownMode: Auto
              scaleUpMode: Auto
          scaleDownStabWindowSeconds: 600
          scaleUpStabWindowSeconds: 300
          scalers:
            scaleDown:
              - LowResourceUtilization
            scaleUp:
              - HighResourceUtilization
              - OOMEvent
              - KubeMetricsInfightRequest
              - KubeApiServerHealthy
              - ...
```

```yaml
status:
  conditions:
      type: VPAScaleDown
    - component: apiserver
      count: 1
      lastEndTime: "2021-11-04T11:38:48Z"
      lastStartTime: "2021-11-04T11:38:48Z"
      message: update master succ
      reason: VPAScaleDownSucc
      status: "True"
      triggers:
        - LowResourceUtilization
      type: VPAScaleUp
    - component: apiserver
      count: 1
      lastEndTime: "2021-11-04T12:00:49Z"
      lastStartTime: "2021-11-04T12:00:49Z"
      message: update master succ
      reason: VPAScaleUpSucc
      status: "True"
      triggers:
        - HighResourceUtilization
        - OOMEvent
      type: VPAScaleUp
  lastScaleTime: "2021-11-04T12:00:49Z"
```

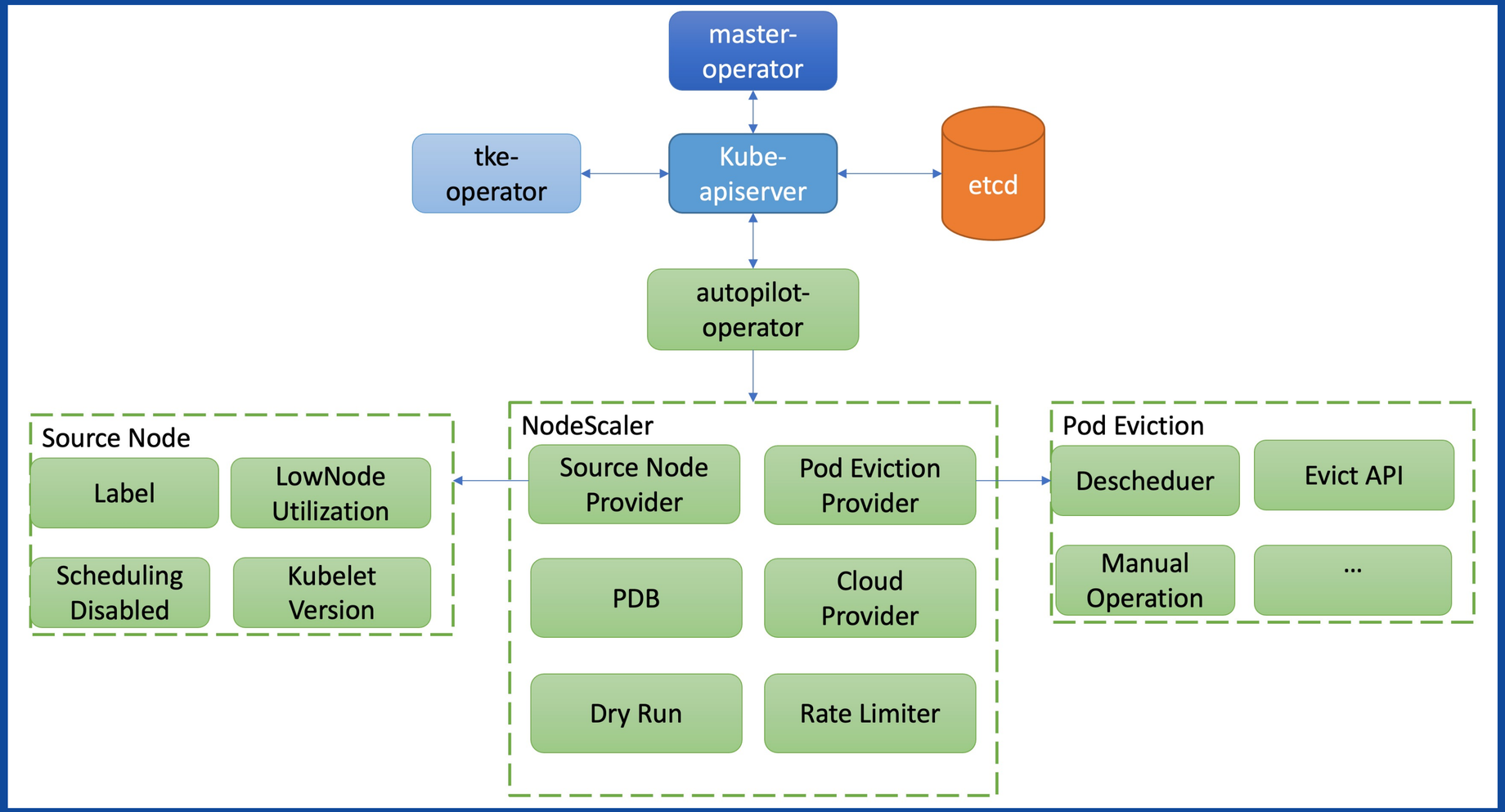
- 支持多种扩容事件源、scaler
- 常用的cpu和内存使用率、oom事件、限速等核心metrics、黑盒连通性测试等
- 支持多种异常场景的根因分析(root cause)
- 扩容过程可视化并推送到运维中心
- 快速速响应扩容，缓慢缩容

metacluster高效运维–kvass解决大集群监控稳定性、扩展性



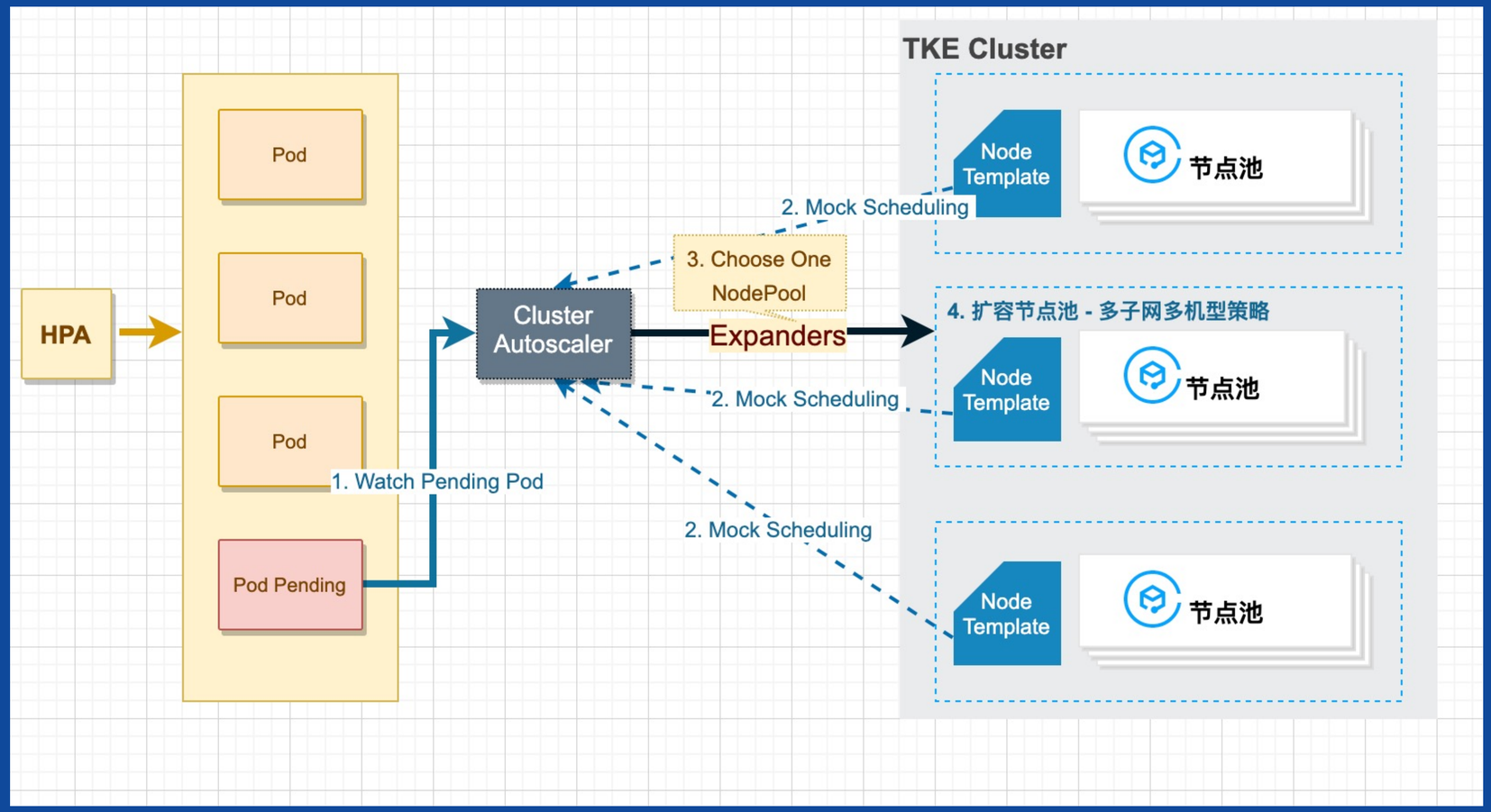
- 轻量，安装方便
- 支持数千万series规模 (数千k8s节点)
- 无需修改Prometheus配置文件，无需加入hash_mod
- target动态调度
- 根据target实际数据规模来进行分片复杂均衡，而不是用hash_mod
- 支持多副本

metacluster高效运维–节点高效治理之缩容



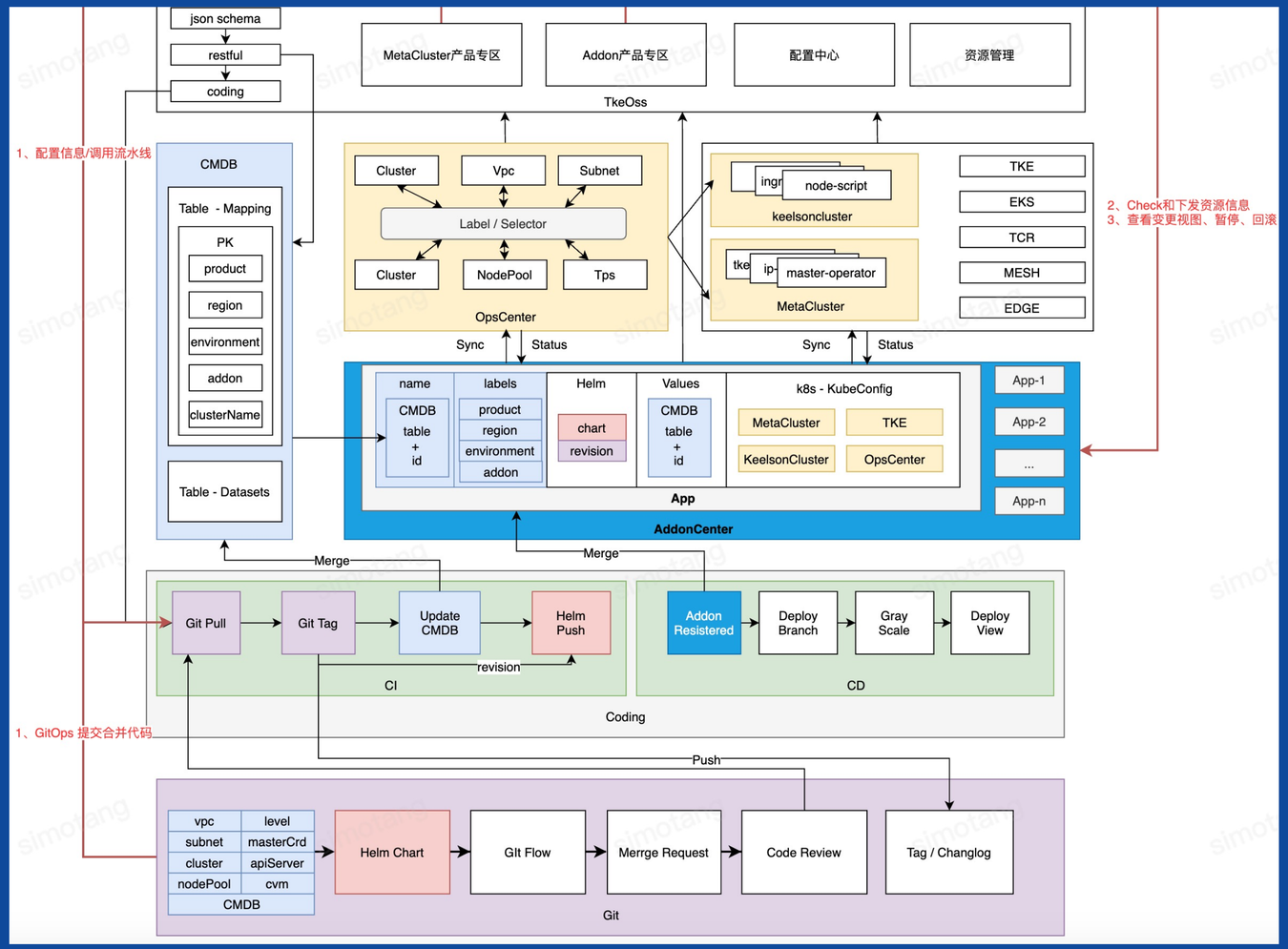
- 支持通过多种策略获取待缩容节点
- 检查及设置每个组件设置pdb策略，确保节点下线时服务可用性不受影响
- 支持通过多种策略来驱逐节点上的Pod
- 支持dry run和限速，节点下线流程更加可靠

metacluster高效运维–节点高效治理之扩容



- 标准化，分组管理（CPU、内存、GPU、AMD/ARM）
- 模板化，自动化，降低操作成本(弹性伸缩，自动扩容)
- 统一的label，taint，支持批量更新（快速扩容，精准调度）
- 自动化、批量维护节点（自动升级、自动恢复）

metacluster高效运维–gitops云原生运维平台



- 基于gitops流程管理多产品、多地域、多环境、多版本、多种类型组件
- 一键开区，高效部署支撑集群与组件，极大避免了人为操作失误

THANKS

Global

Architect Summit