Kubernetes and Democratizing Infrastructure

Kubernetes项目与基础设施“民主化”的探索

Lei Zhang

Kubernetes Community

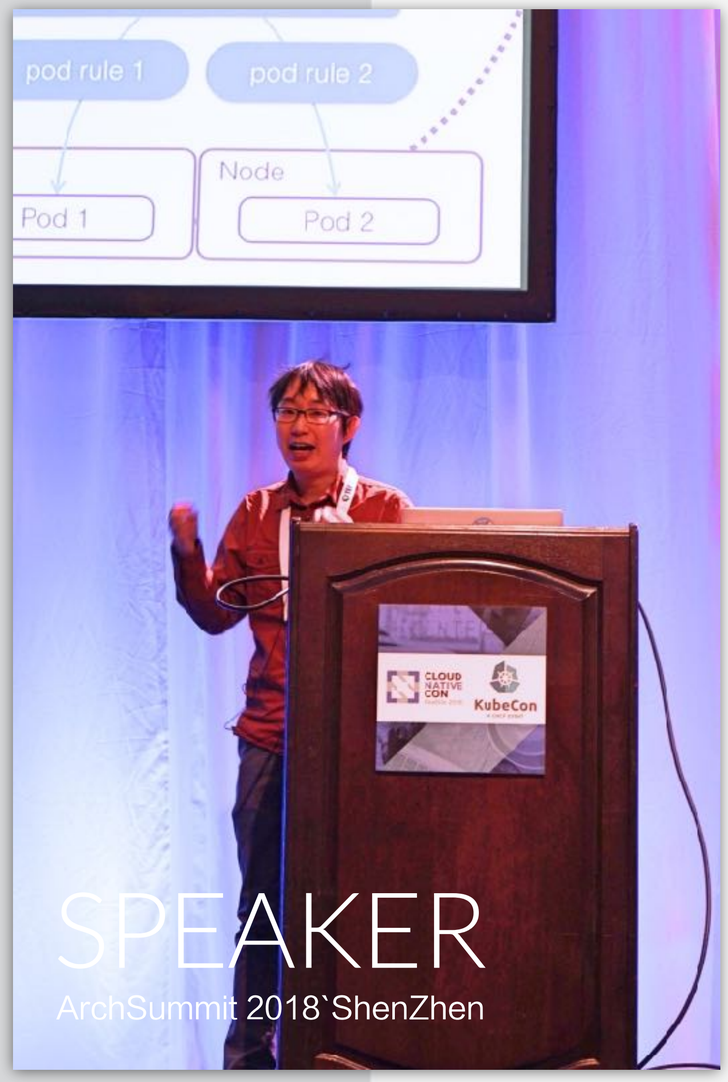
Geekbang. 极客邦科技 | 极客时间

SPEAKER

INTRODUCE

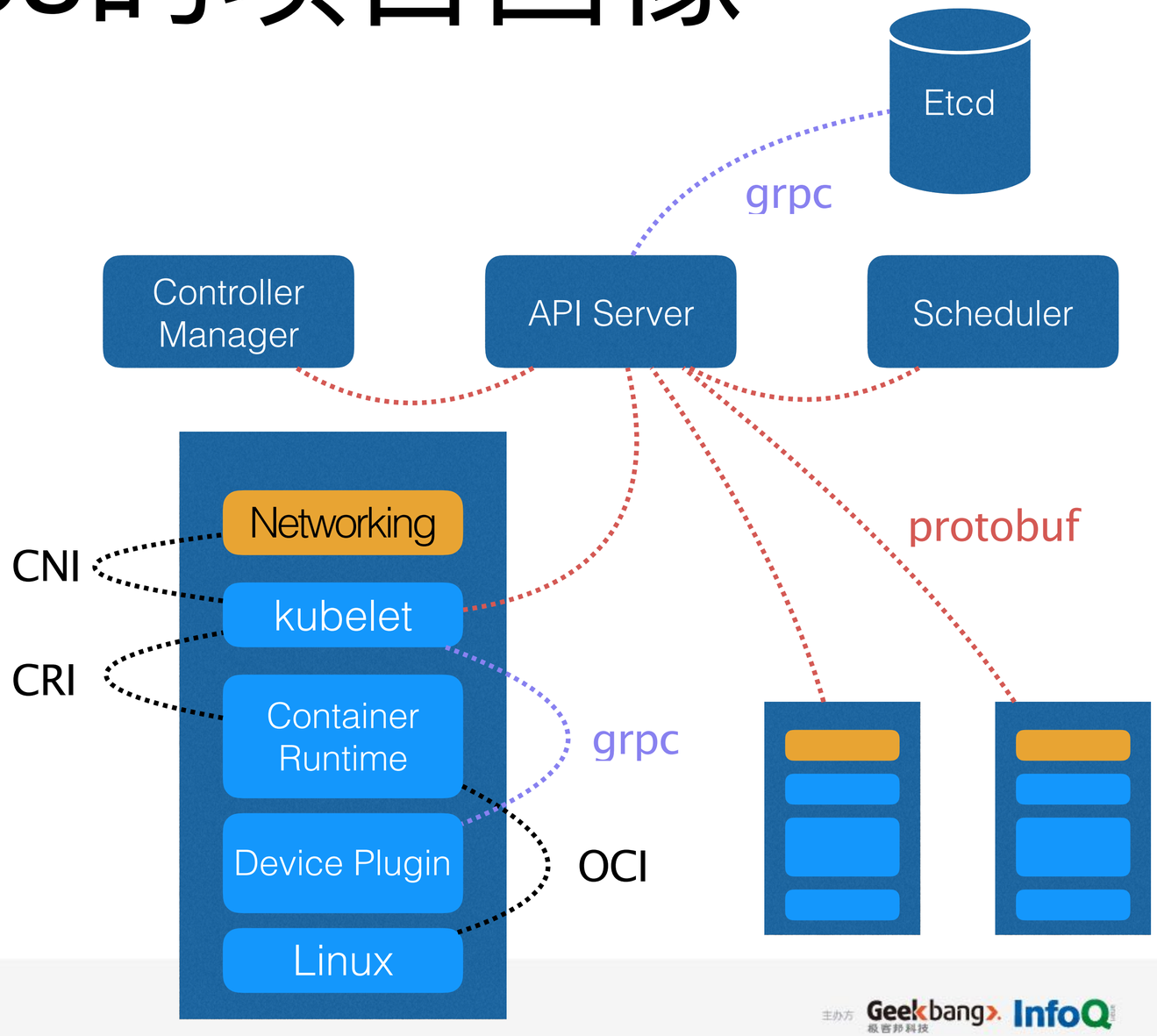
Lei Zhang Kubernetes Community

- 浙江大学, HyperHQ, Microsoft Research (MSR)
- 阿里巴巴集团 技术顾问 & 高级技术专家



Kubernetes的项目画像

- 目标用户：分布式应用的开发者
- 目标集群规模：5000，舒适区：100~1000
- 核心能力：提供基于容器的设计模式与编程规范
- 功能/性能取舍：功能性需求 > 性能性需求
- 用户评价：
  - **“太复杂了！”**



Kubernetes中最复杂的是：yaml文件

yaml文件 = 声明式API

- 什么是声明式API?
- docker service create --name nginx --replicas 2 nginx
- docker service update --image nginx:1.7.9 nginx
  - **命令式**命令行操作
- 有一个与yaml文件对应的API实体（Entity）
  - kubectl create -f **nginx.yaml**
  - kubectl replace -f **nginx.yaml**
    - **命令式**配置文件操作
- 别逗，到底什么才是**声明式API**?

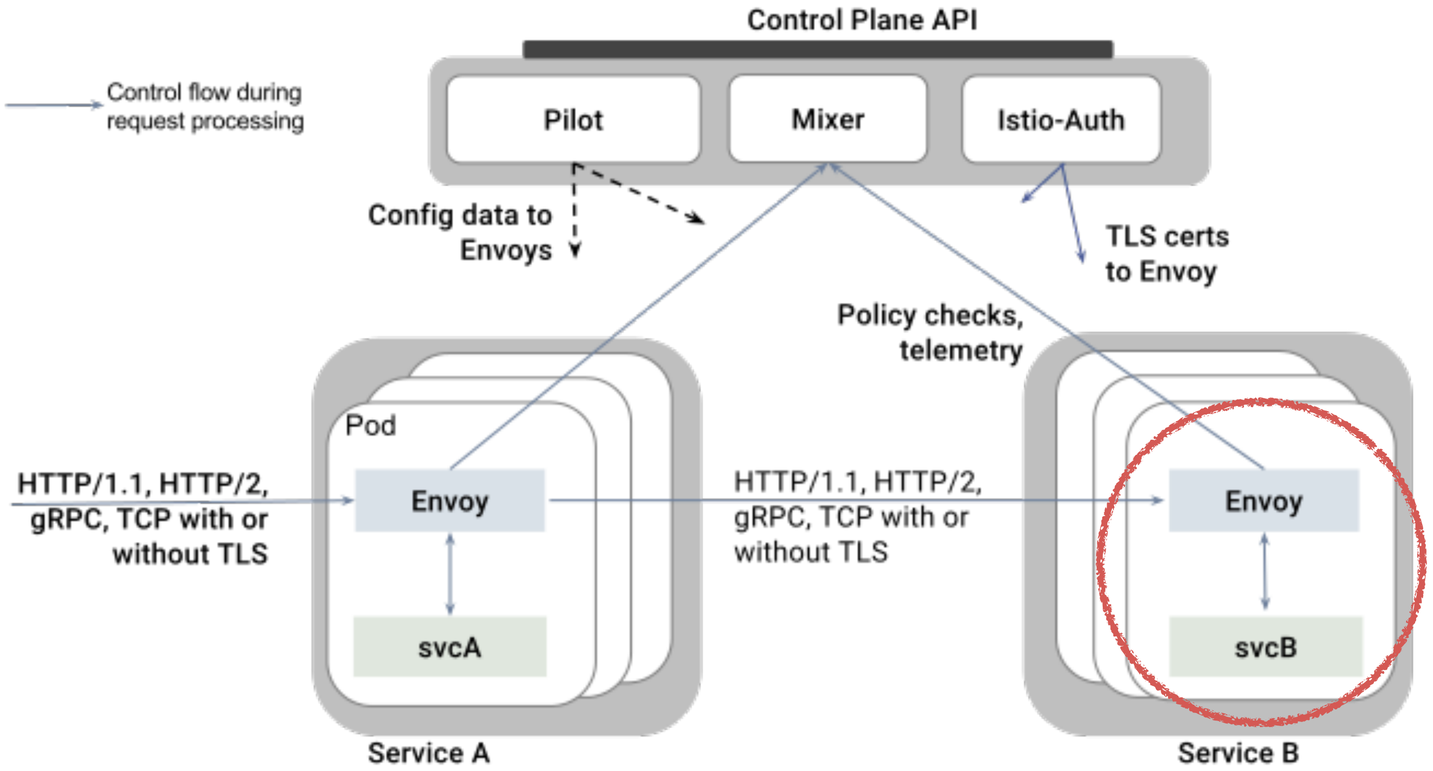
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
```

声明式API

- kubectl **apply** -f nginx.yaml
- // 将image改成nginx:1.7.9
- kubectl **apply** -f nginx.yaml
- **区别?**
  - kubectl replace = **replace**
    - i.e. 1 writer
  - kubectl apply = **patch**
    - i.e. 1+ writers

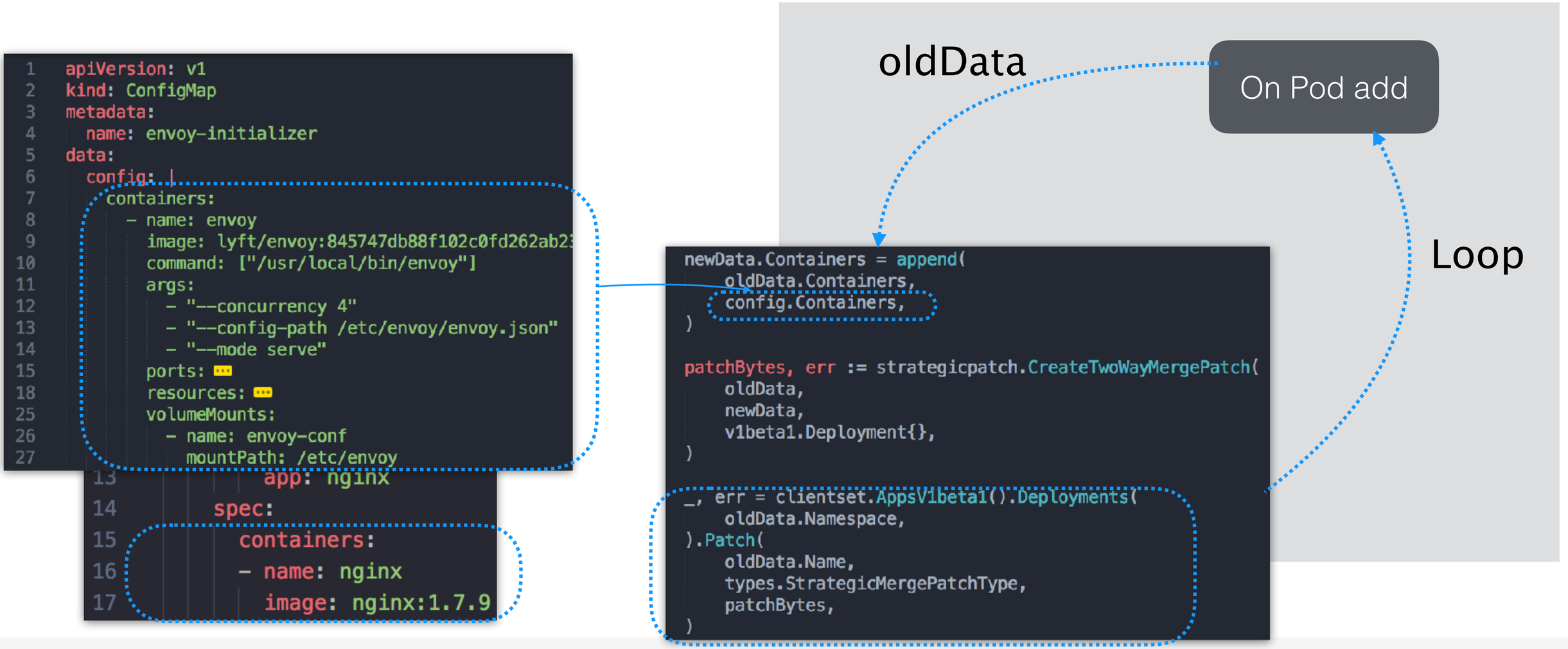
- **Kubernetes声明式API的本质：**
  - 在Etcd中维护(CRUD)了一个API实体
  - 允许有**多个API写端**，以**PATCH**的方式对API实体进行修改，而无需关心本地的原始yaml文件

Istio



自动化的proxy容器注入，是Service Mesh的“灵魂”所在

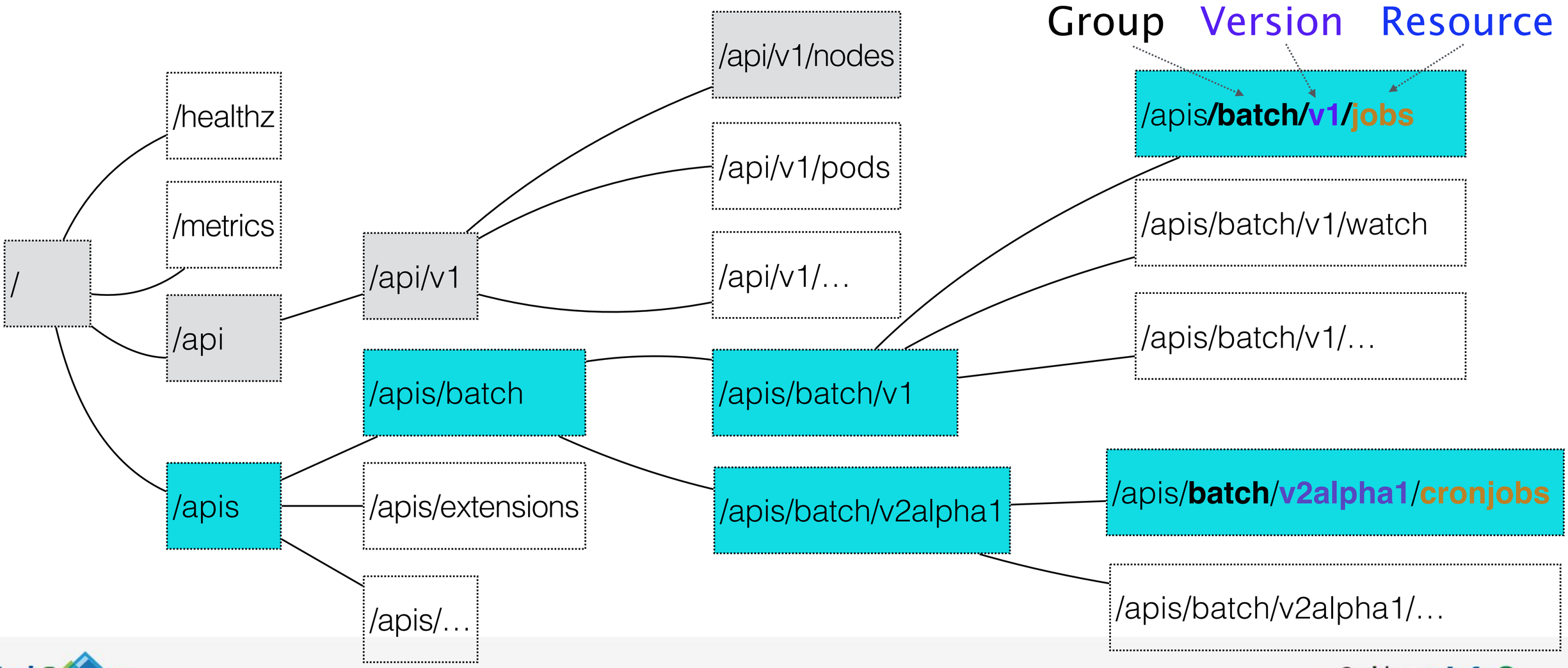
Initializer



声明式API的意义

- 编程友好，而非最终用户友好
- 提供了一套**面向API实体**的编程模型
- 原因：**生态！生态！生态！**
- 这正是Kubernetes项目“复杂性”的根本来源!

Kubernetes的API资源模型



Kubernetes API Workflow

Filters: server/config.go
MaxInflightLimit
Timeout
Panic Recovery
Auth
Audit
…

MUX: server/mux/container.go
Routes: server/routes/*.go

API Groups: endpoints/*.go
API GroupVersion Installer
API Installer

POST /apis/batch/v2alpha1/cronjobs

/

Find Handler

/apis/**batch**

/apis/batch/**v2alpha1**

/apis/batch/v2alpha1/**cronjobs**

Create Resource:
endpoints/hanlders/rest.go
ReadAll
Decode
Convert
Admit

Create Store:
registry/generic/registry/*.go
Validate

Serialization/Encode:
storage/etcd3/store.go
Convert
Encode

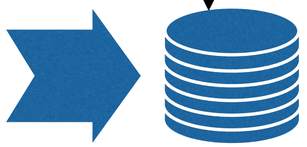
v1beta1

super version

Admission()

Validation()

v1



Etcd
JSON / protpbuf

典型的用户诉求

- “如何在Kubernetes中添加新的API”?
- “如何在Etcd里保存一个我自己的API对象”?

Custom Resource Definition (CRD)

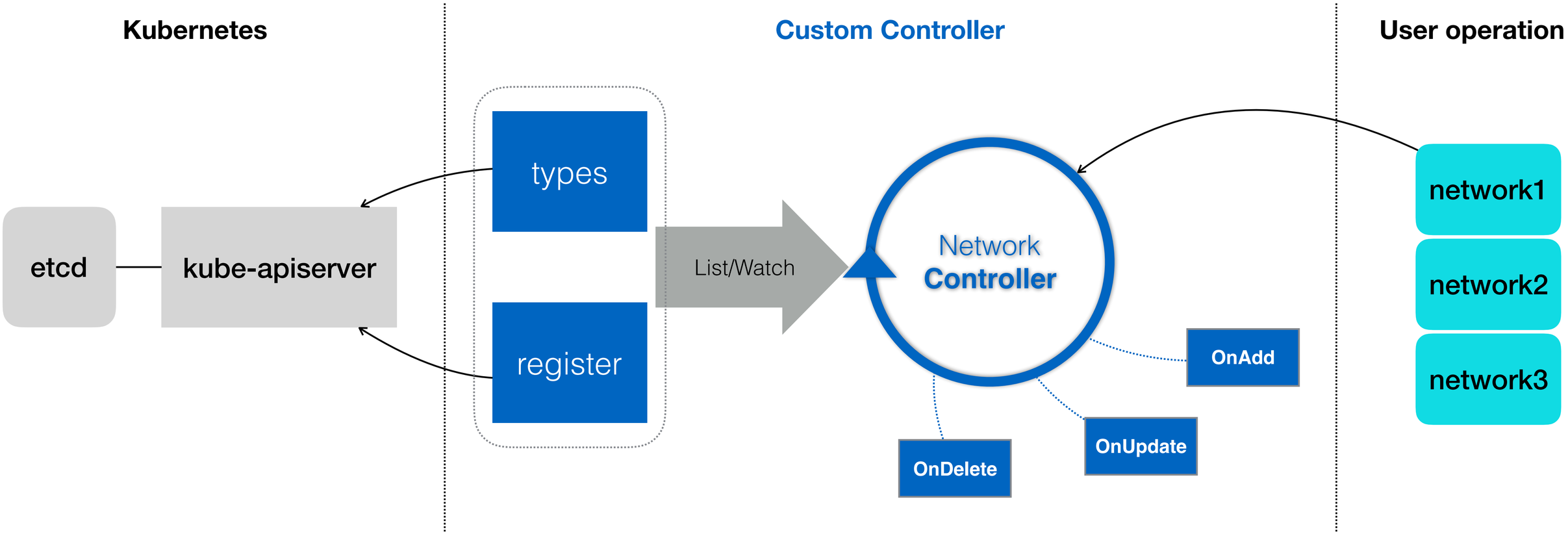
- 自定义API资源模型（CRD）举例：自定义Network
  - kubectl apply -f network-crd.yaml
  - kubectl apply -f my-network.yaml
  - kubectl get network

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  # name must match the spec fields below,
  # and be in the form: <plural>.<group>
  name: network.crd.archsummit.com
spec:
  # group name to use for
  # REST API: /apis/<group>/<version>
  group: crd.archsummit.com
  # version name to use for
  # REST API: /apis/<group>/<version>
  version: v1
  # either Namespaced or Cluster
  scope: Namespaced
```

```yaml
apiVersion: "crd.archsummit.com/v1"
kind: Network
metadata:
  name: my-new-network-object
```

```
NAME                        AGE
my-new-network-object       6s
```

Kubernetes的API编程范式



Controller Pattern

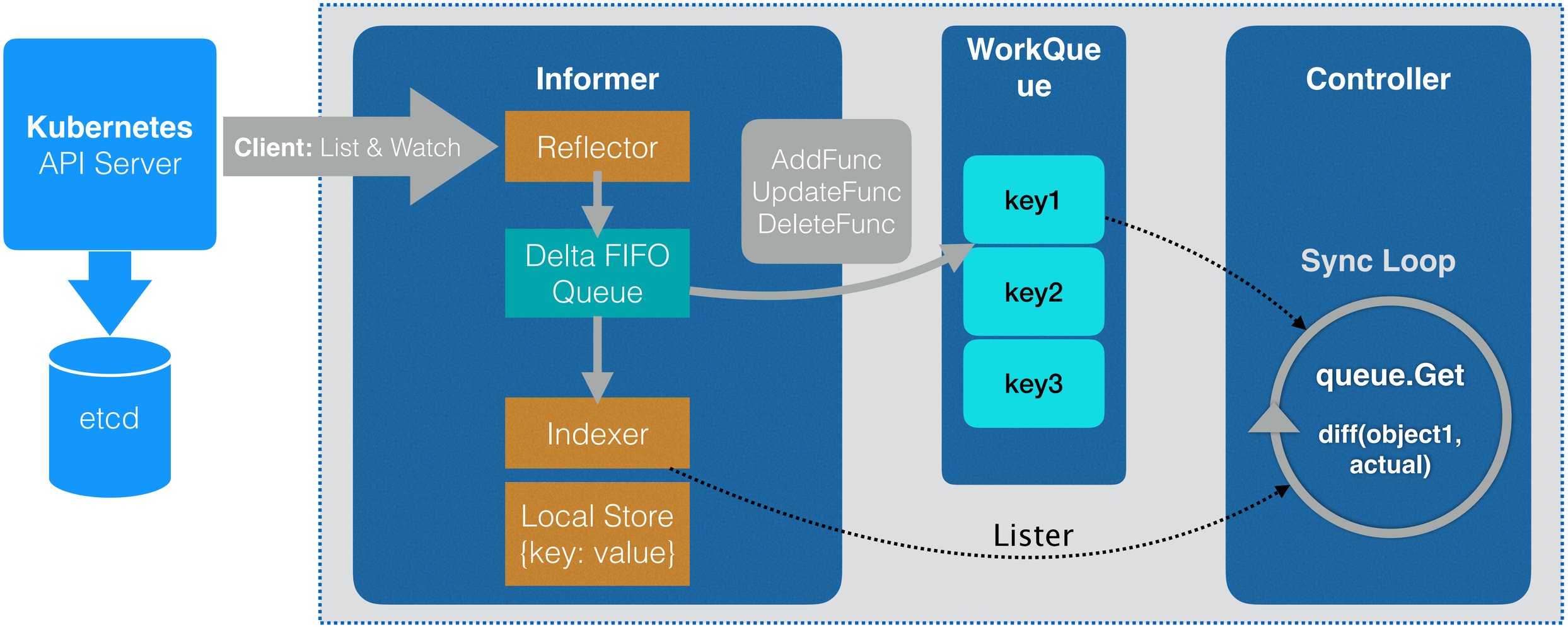
- 即：Reconcile Loop，Kubernetes项目最基本的控制模型

声明式API实体

```
for {
  desired := getDesiredState()
  current := getCurrentState()
  makeChanges(desired, current)
}
```

主动获取
心跳/反馈
Metrics
…

Controller Deep Dive



Kubernetes API Programming

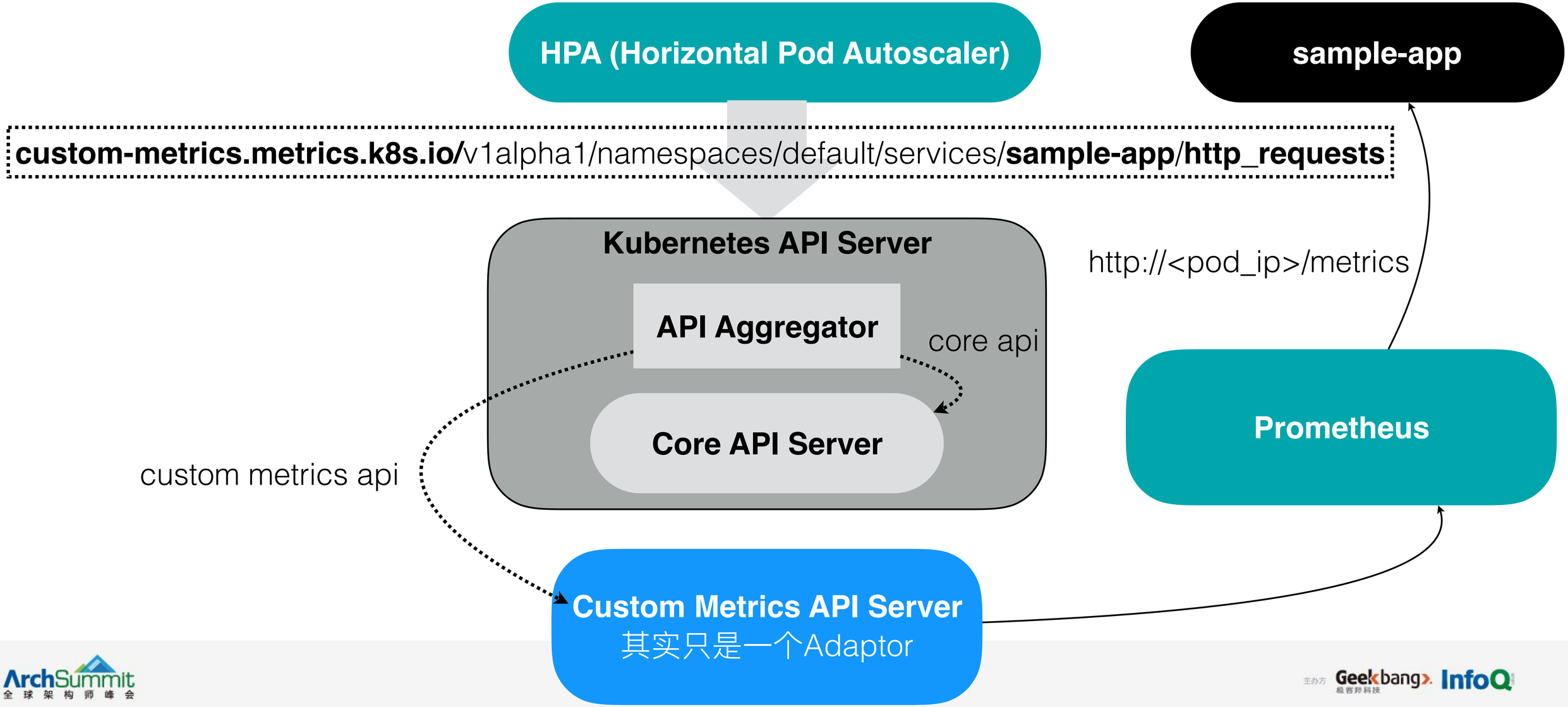
- **对于给定的API Object**
- Kubernetes会为用户自动生成
  - client, informer, lister
  - 并提供多个work queue的实现
- 用户需要编写的代码
  - controller，以及controller中的sync()

```
// +genclient
// +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/

// Network is a specification for a Network resource
type Network struct {
    metav1.TypeMeta   `json:",inline"`
    metav1.ObjectMeta `json:"metadata,omitempty"`

    Spec   NetworkSpec   `json:"spec"`
    Status NetworkStatus `json:"status"`
}
```

CRD不是万能的

- 不应该被当作数据库使用
- CRD有性能瓶颈
  - 暂不支持protobuf
    - API Object数量>1K，或者单个对象>1KB
    - 高频请求（e.g. > 10 req/s）
- 不支持命令式API操作
  - 需要等待API最终返回，或者检查API完成状态
  - API没办法用API Object的方式建模
- 需要实现**完整**的Kubernetes API功能
  - **API Aggregation**

API Aggregation (AA)

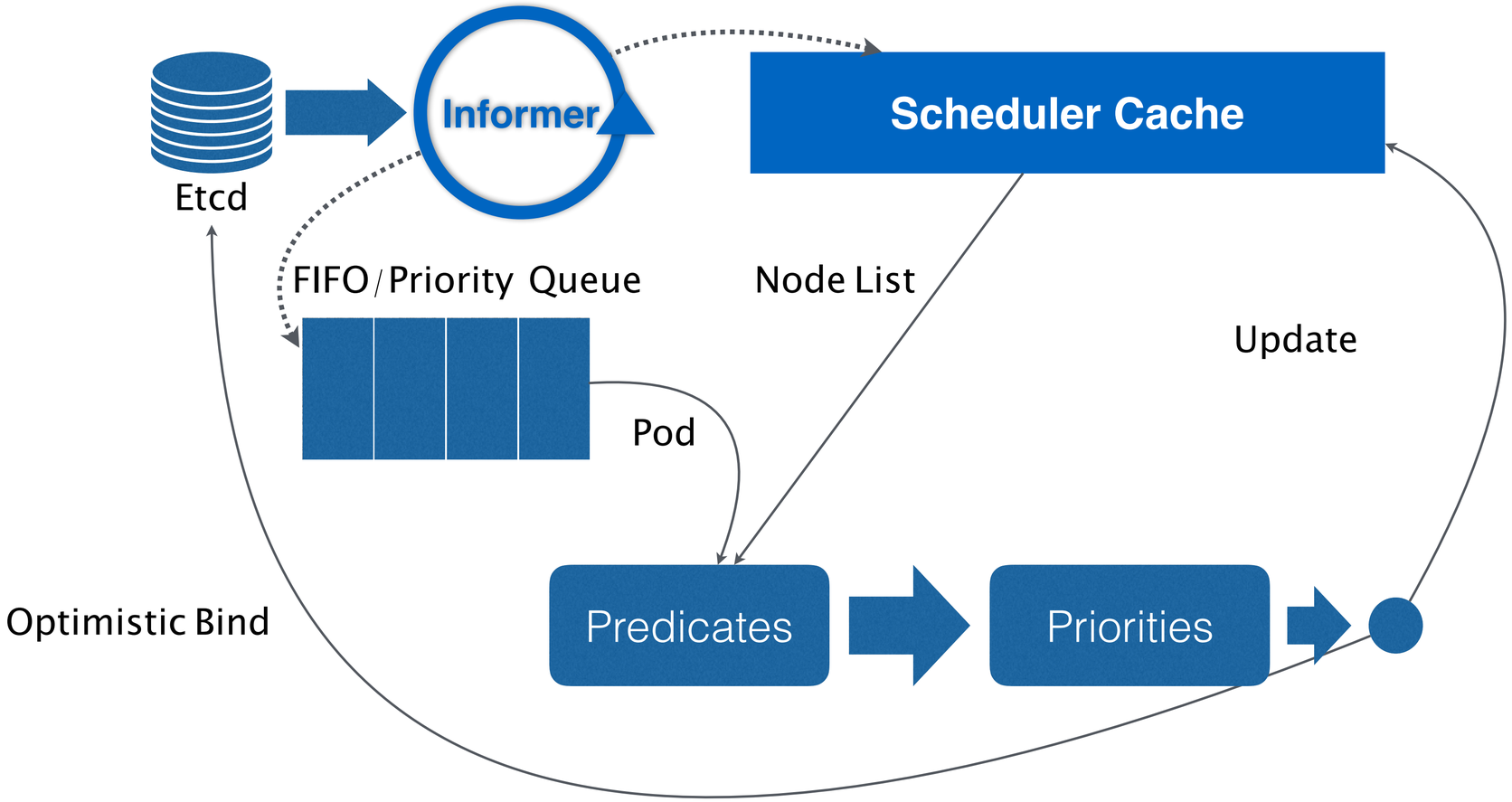


Kubernetes的API层的演化

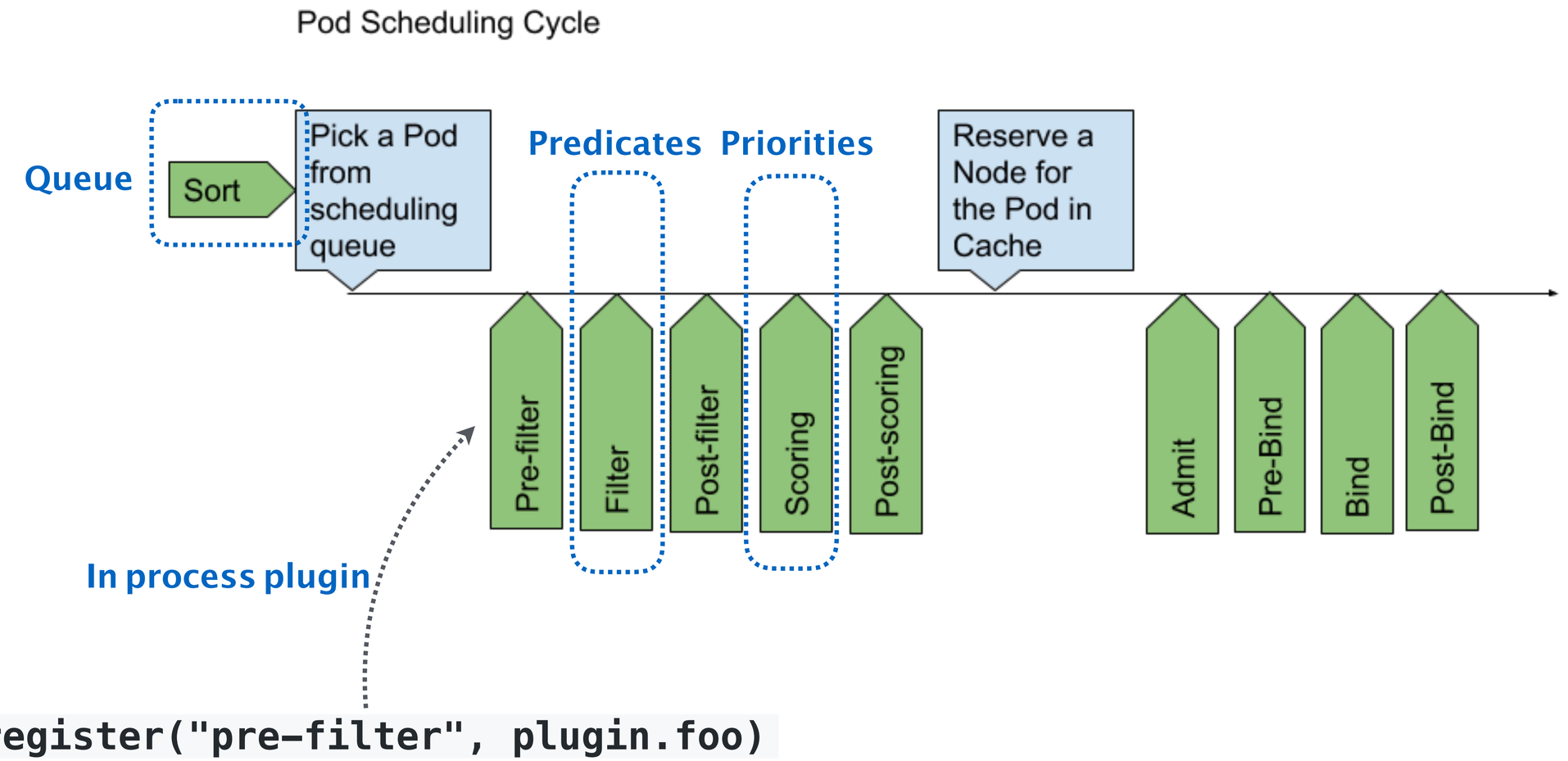
- **集中式的API组件 $\Longrightarrow$ SDK化的API框架**
- 催生出以Kubernetes API为基础的、容器化分布式应用生态
  - e.g. Operator, CRD, Istio, Serverless, Rook.io, Vitness, Google Cloud工具集 ~= **整个CNCF**

kube-scheduler

- **Cache化**
  - 调度全部基于缓存来决策
  - 使用Informer机制来同步缓存
  - 无锁
  - Optimistic Bind
  - Http Extender
- **Framework化**
  - Status: Draft



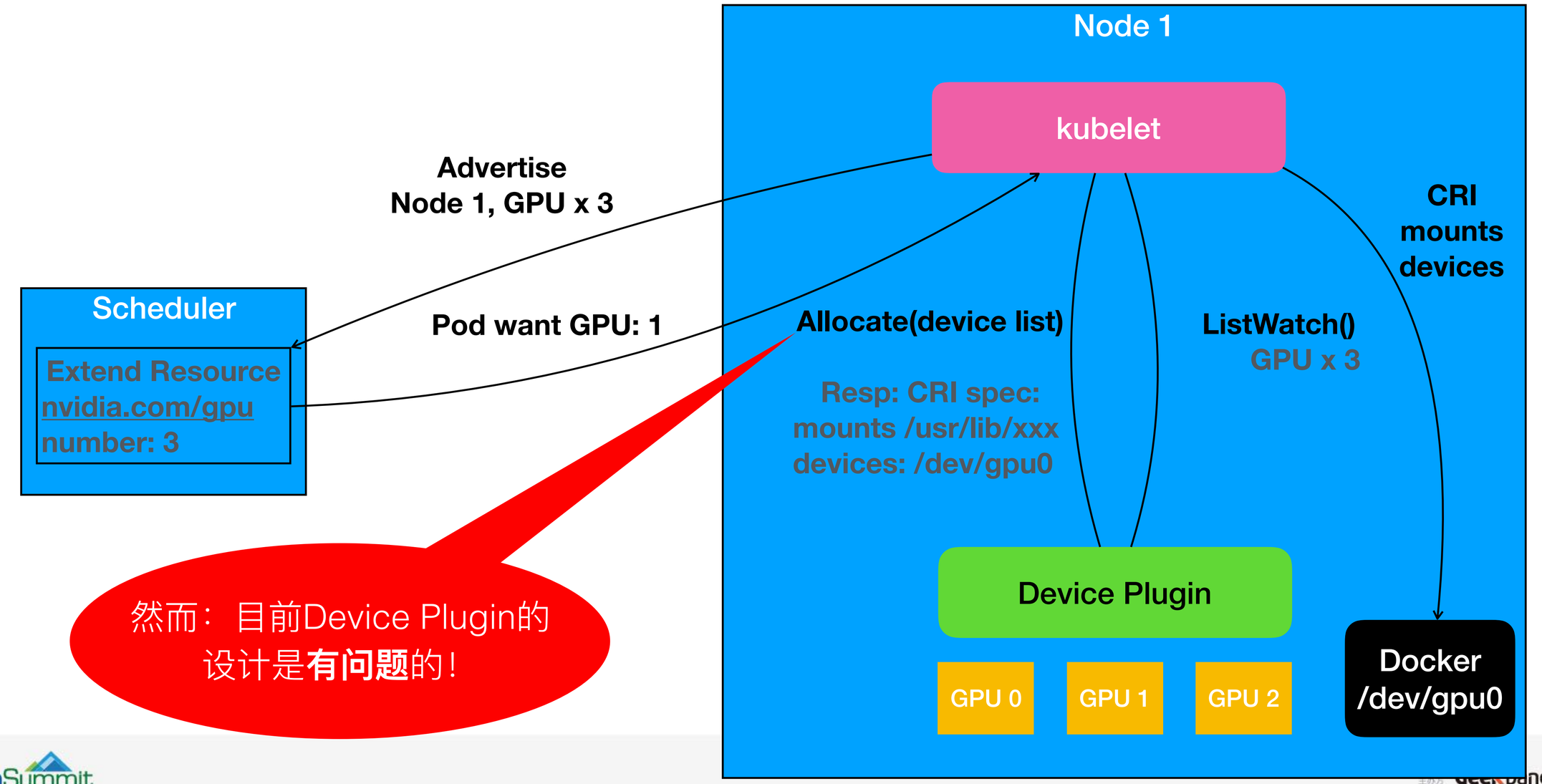
kube-scheduler的Framework化



DevicePlugin

- 以GPU支持为例
- < v1.9：直接跟Docker绑定
  - kubelet直接hack dockershim，在发起Docker API之前inject注入的GPU device和mount目录
- v1.9+：使用CRI和grpc接口
  - Inject CRI spec

DevicePlugin的设计



DevicePlugin的问题出在哪?

-

```
struct ComputeResource {
  metav1.ObjectMeta
  Spec ComputeResourceSpec
  Status ComputeResourceStatus
}
struct ComputeResourceSpec {
     // For node-level resource, this is the name of the Node that owns this ComputeResource.
    // For cluster-level resource, this field is empty.
    NodeName string
     // Raw resource name. E.g.: nvidia.com/gpu
    ResourceName string
    // gpuType: k80
    // zone: us-west1-b
   // Note Kubelet adds a special property corresponding to the above ResourceName field.
   // This will allow a single ResourceClass (e.g., "gpu") to match multiple types of
   // resources (e.g., nvidia.com/gpu and amd.com/gpu) through general set selector.
   Properties map[string]string
}
struct ComputeResourceStatus {
   Capacity resource.Quantity
}
```

-

```
// +nonNamespaced=true
// +genclient=true

type ResourceClass struct {
        metav1.TypeMeta
        metav1.ObjectMeta
        Spec ResourceClassSpec
        // +optional
}

type ResourceClassSpec struct {
        // defines general resource property matching constraints.
        // e.g.: zone in { us-west1-b, us-west1-c }; type: k80
        MetadataRequirements metav1.LabelSelector
        // used to compare preference of two matching ResourceClasses
        // The purpose to introduce this field is explained more later
        Priority int
```

che

```
kind: ResourceClass
metadata:
  name: nvidia.high.mem
spec:
  labelSelector:
    - matchExpressions:
        - key: "Kind"
          operator: "In"
          values:
            - "nvidia-gpu"
        - key: "memory"
          operator: "GtEq"
          values:
            - "30G"
```

- 正在讨论中的解决方案:
  - Resource API

Kubernetes生态的发展方向

- 头重
  - API层的扩展与插件能力的进一步提升
  - 基于扩展型API的Serverless、CI/CD等周边项目涌现
- 脚轻
  - 核心组件进一步插件化、框架化

结语

Kubernetes社区的第一次繁荣已经进入尾声

以扩展型API、插件化核心组件、各类标准化接口为代表的第二次繁荣，正呼之欲出

开源基础设施项目的“民主化”趋势

Democratizing Infrastructure

ArchSummit
全 球 架 构 师 峰 会

关注 ArchSummit 公众号

获取国内外一线架构设计

了解上千名知名架构师的实践动向

北京站：2018年12月7-10日

极客邦企业培训与咨询

帮助企业与技术人成长

精品课程

Excellent Course

- ☑《互联网大规模分布式架构设计与实践》
- ☑《基于大数据的企业运营与精准营销》
- ☑《大数据和人工智能在金融领域的应用》
- ☑《区块链应用与开发技术高级培训》
- ☑《通往卓越管理的阶梯》

扫码关注官方微信服务号
了解更多课程详细信息

Geekbang
极客邦科技

THANKS