

All Things You Should Know after Kubernetes 1.8

Harry Zhang @resouer

主办：



Contents



- 1.8之后Kubernetes以及容器社区的现状
- 我们（社区参与者）应该着重关注的主线在哪？
- 这些关注点的设计与实现又是怎样的？

Kubernetes渐成主流



In the last year, however, the growth of Kubernetes has far outpaced other orchestrators. Rancher users are increasingly demanding a better user experience and more functionality on top of Kubernetes. We have, therefore, decided to reengineer Rancher 2.0 to take advantage of the power of Kubernetes by rebasing the popular Rancher experience (known as Cattle) on Kubernetes. With Rancher 2.0:

中国工商银行股份有限公司云计算技术支持服务项目(PaaS)国内公开招标评标结果公示

招标编号: 0747-1660SITCD139

招标内容: 云计算技术支持服务项目(PaaS)

经过评标委员会评定, 推荐中标候选人及非序公示如下:

第一中标候选人: 国际商业机器(中国)有限公司

第二中标候选人: 红帽软件(北京)有限公司

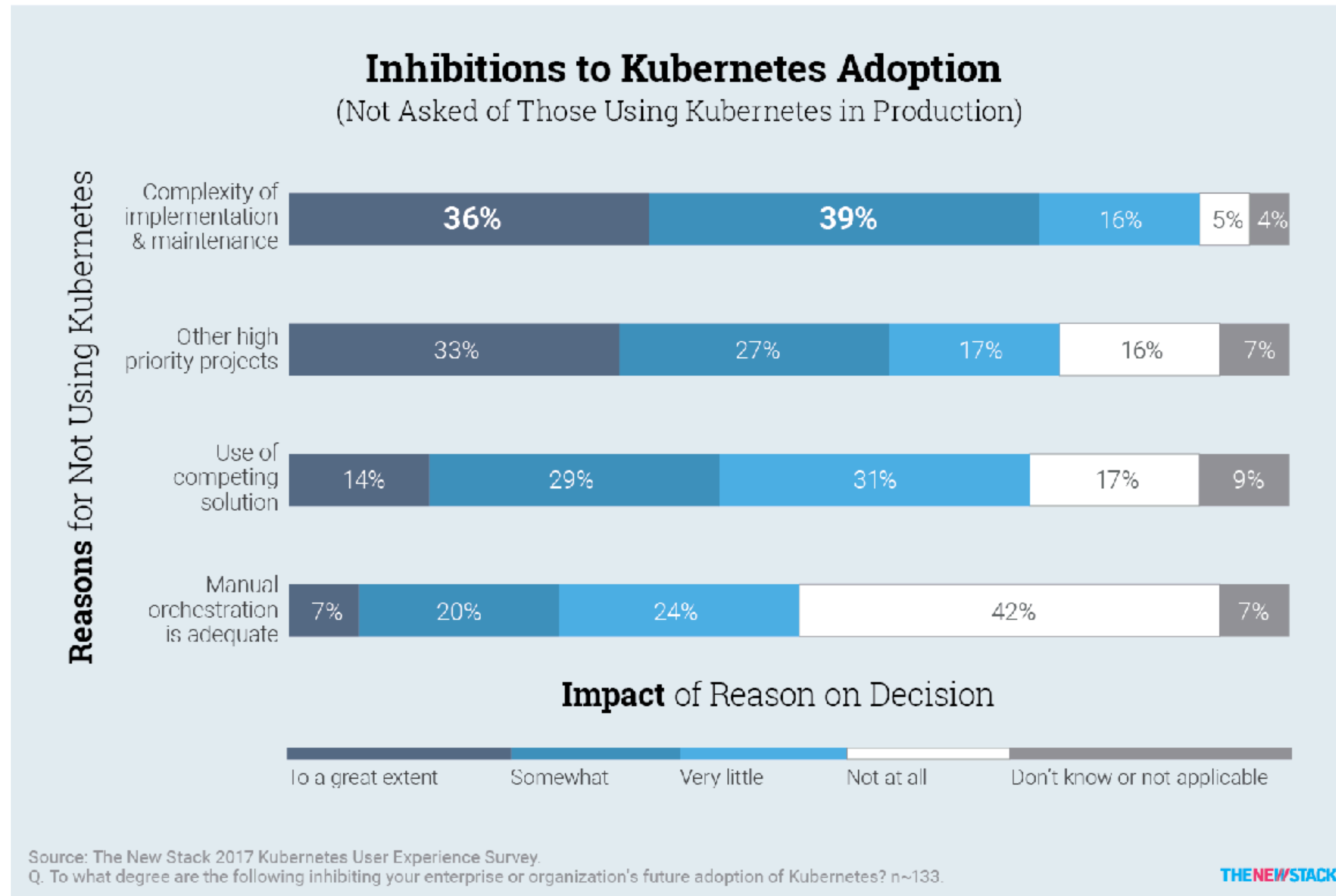
第三中标候选人: 华为技术有限公司

中化国际招标有限责任公司

2017年4月17日



Kubernetes任重道远



曾经的两个核心问题

(build once, run everywhere)

- 代码/二进制 -> 最小可运行单元

- App Engine:

- Intrusive plugin/library (侵入性插件)

- 经典PaaS:

- buildpack: war + tomcat

- Docker: **胜出**

- **王炸**: Docker Image

- war + tomcat + OS libs/bins/deps

- 最小可运行单元 -> 最终用户服务

- Others:

得了，咱们也学Kubernetes吧

- DEA, docker run, IaaS etc.

- 其实并不知道该咋整 ...

- Kubernetes:

胜出

- **王炸**: 抄Borg!

- Pod, Deployment (replica), StatefulSet, Job, CronJob, DaemonSet, Service, Controller

- CRI, CNI, CSI, DevicePlugin ...

1.8之后的发展主线

- 主线：受到社区重点关注，具有技术和战略价值
 1. Runtime逐渐趋于稳定，直面用户的sig-app将体现出强大的吸引力
 2. 生态地位逐步确立，“可扩展性”便成为社区的主战场
- 这两点将是Kubernetes 1.8之后主旋律

1. Control Panel

Pod! Pod! Pod!

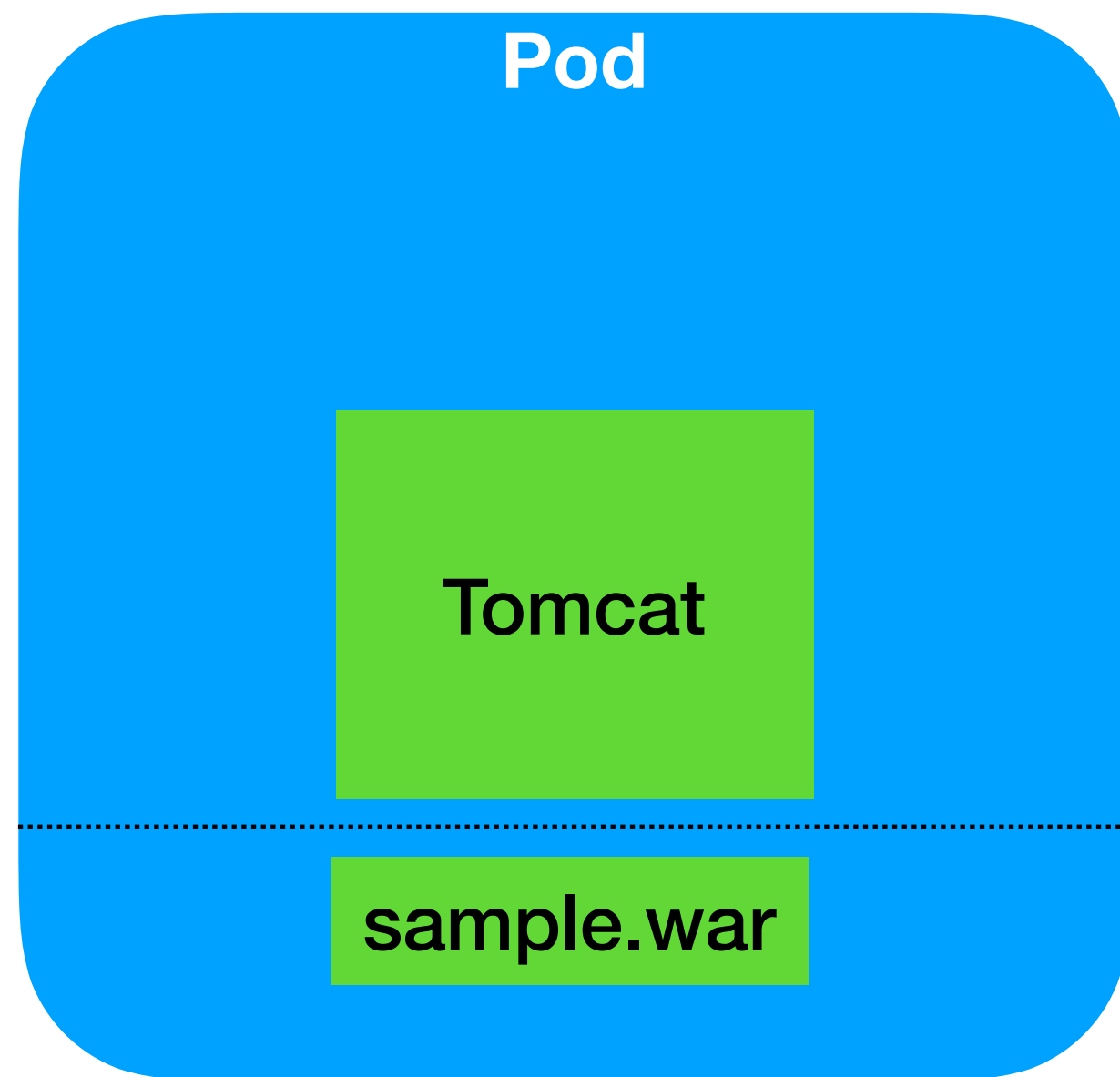
- **解耦**容器关系
- 原子**调度**单位
- Shared namespace
 - network, IPC etc
- Shared volume

Process group in container cloud

Pod: stop creating fat container

抑制制作“胖”容器的冲动

Best Practice: Tomcat + war



- 两个容器组成一个Pod:
 - war容器
 - Tomcat容器

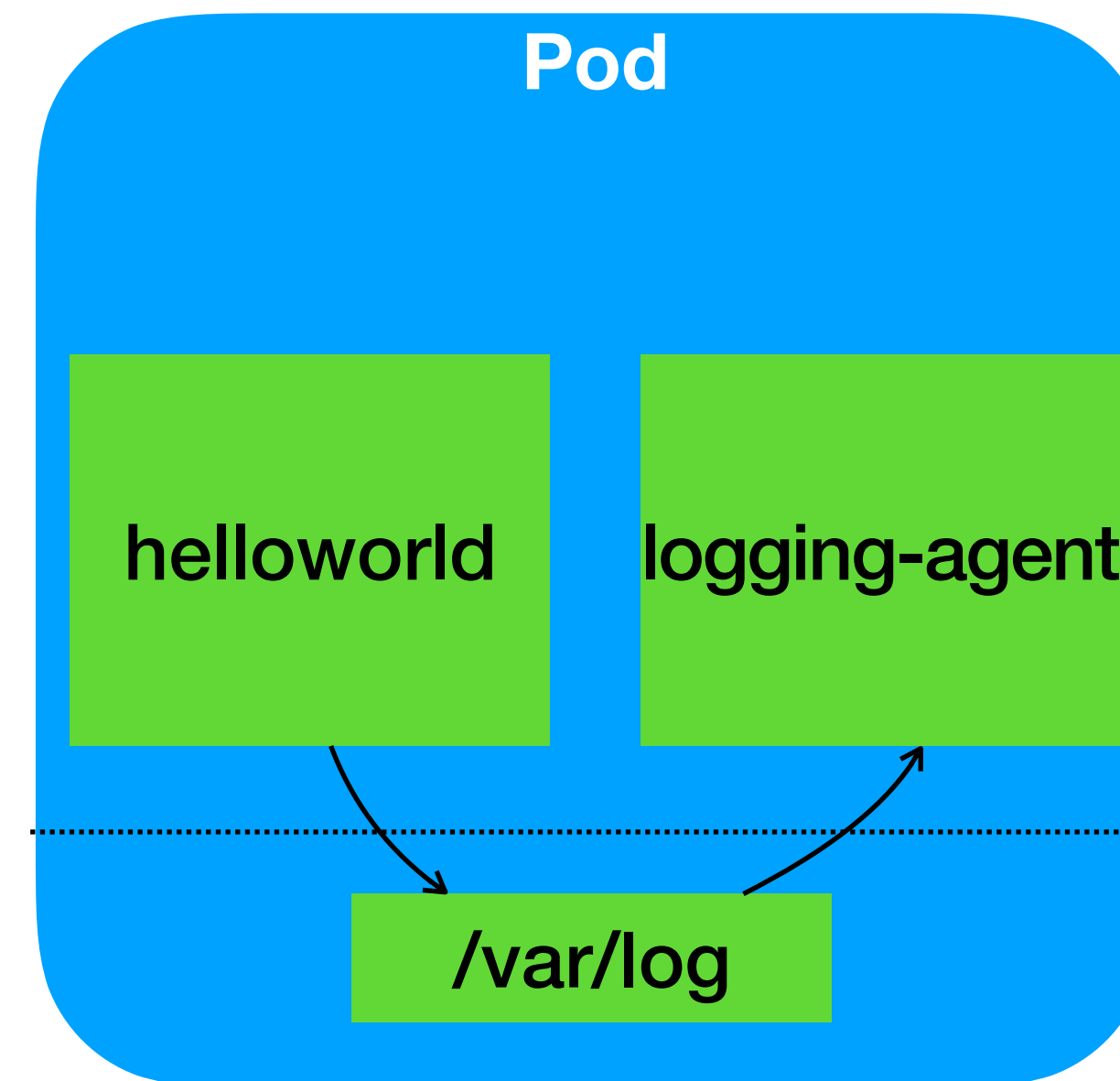
```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: javaweb-2
5  spec:
6    initContainers:
7      - name: war
8        image: resouer/sample:v2
9        command: ["cp", "/sample.war", "/app"]
10       volumeMounts:
11         - mountPath: /app
12           name: app-volume
13   containers:
14     - name: tomcat
15       image: resouer/mytomcat:7.0
16       command: ["sh", "-c", "/root/apache-tomcat-7.0.42-v2/bin/start.sh"]
17       volumeMounts:
18         - mountPath: /root/apache-tomcat-7.0.42-v2/webapps
19           name: app-volume
20     ports:
21       - containerPort: 8080
22         hostPort: 8001
23   volumes:
24     - name: app-volume
25       emptyDir: {}
```

Pod: container design pattern

“容器设计模式”

入门级：sidecar

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: helloworld
spec:
  replicas: 1
  template:
    metadata:
      name: helloworld
    spec:
      containers:
        - name: helloworld
          image: gcr.io/resouer/helloworld:0.0.1
          volumeMounts:
            - name: varlog
              mountPath: /var/log
        - name: logging-agent
          image: gcr.io/google_containers/fluentd:1.30
          volumeMounts:
            - name: varlog
              mountPath: /var/log
      volumes:
        - name: varlog
          emptyDir: {}
```



进阶级： initializer

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  annotations:
    "initializer.kubernetes.io/logging-agent": "true"
  name: helloworld-with-annotation
spec:
  replicas: 1
  template:
    metadata:
      name: helloworld-with-annotation
    spec:
      containers:
        - name: helloworld
          image: gcr.io/resouer/helloworld:0.0.1
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: logging-agent-initializer
data:
  config: |
    - name: logging-agent
      image: gcr.io/google_containers/fluentd:1.30
      volumeMounts:
        - name: varlog
          mountPath: /var/log
      volumes:
        - name: varlog
          emptyDir: {}
```

Automatically Inject



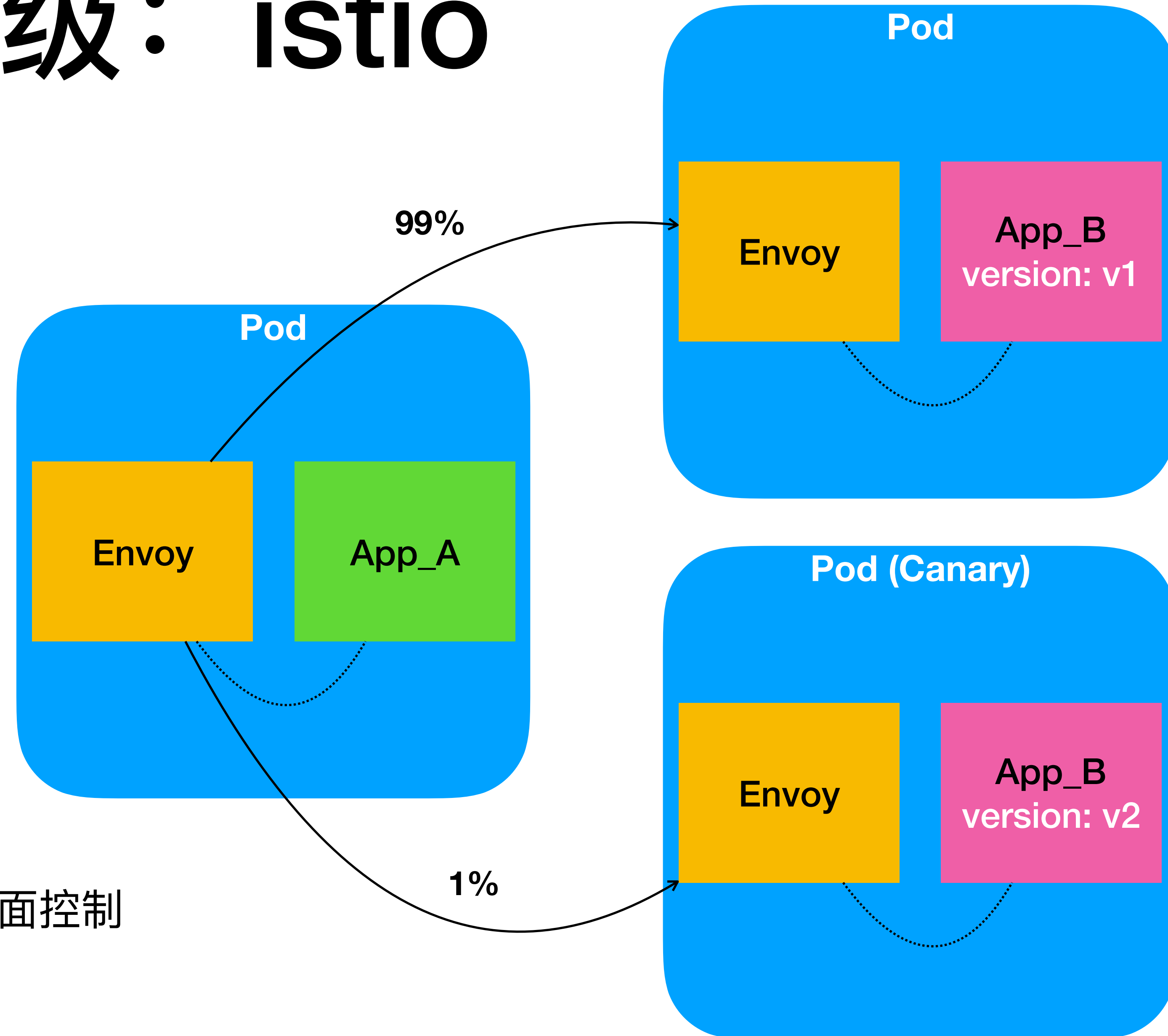
大师级：istio

- 面向容器的非侵入式微服务框架

- 服务拆分 (Pod)
- 流量控制 (Routing & LB)
- 访问授权 (Auth)
- 策略控制 (切面/Aspect)

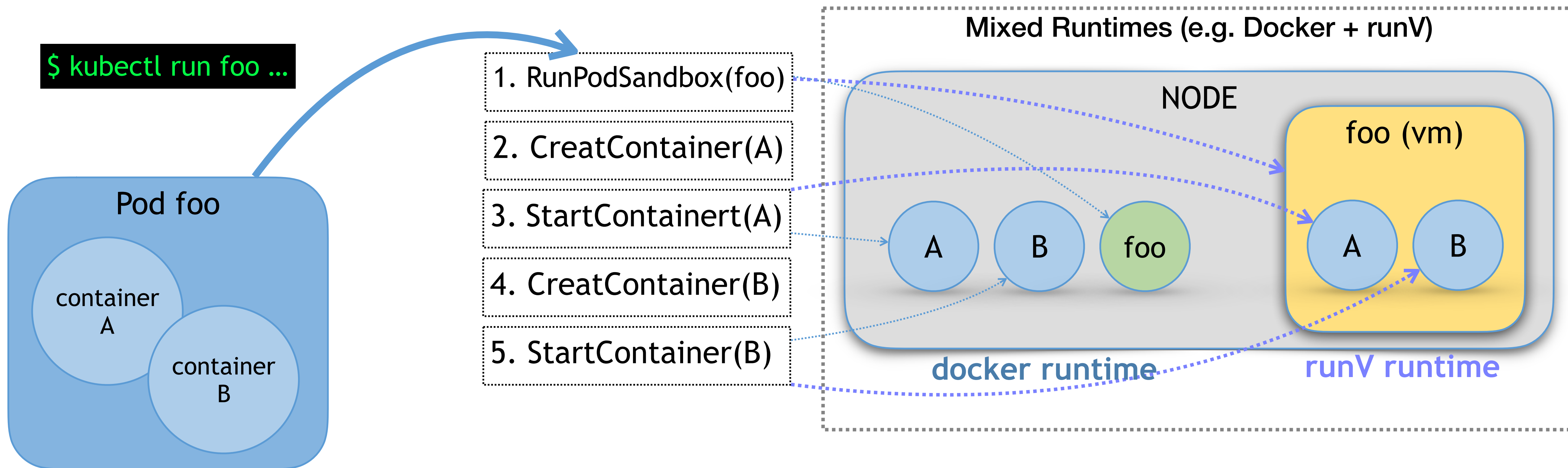
- 实现原理：

- 向每个Pod注入Envoy容器来负责流量切面控制



Pod的实现：CRI

Leads: Google, Hyper, CoreOS, RedHat



Controller: Kubernetes的心脏

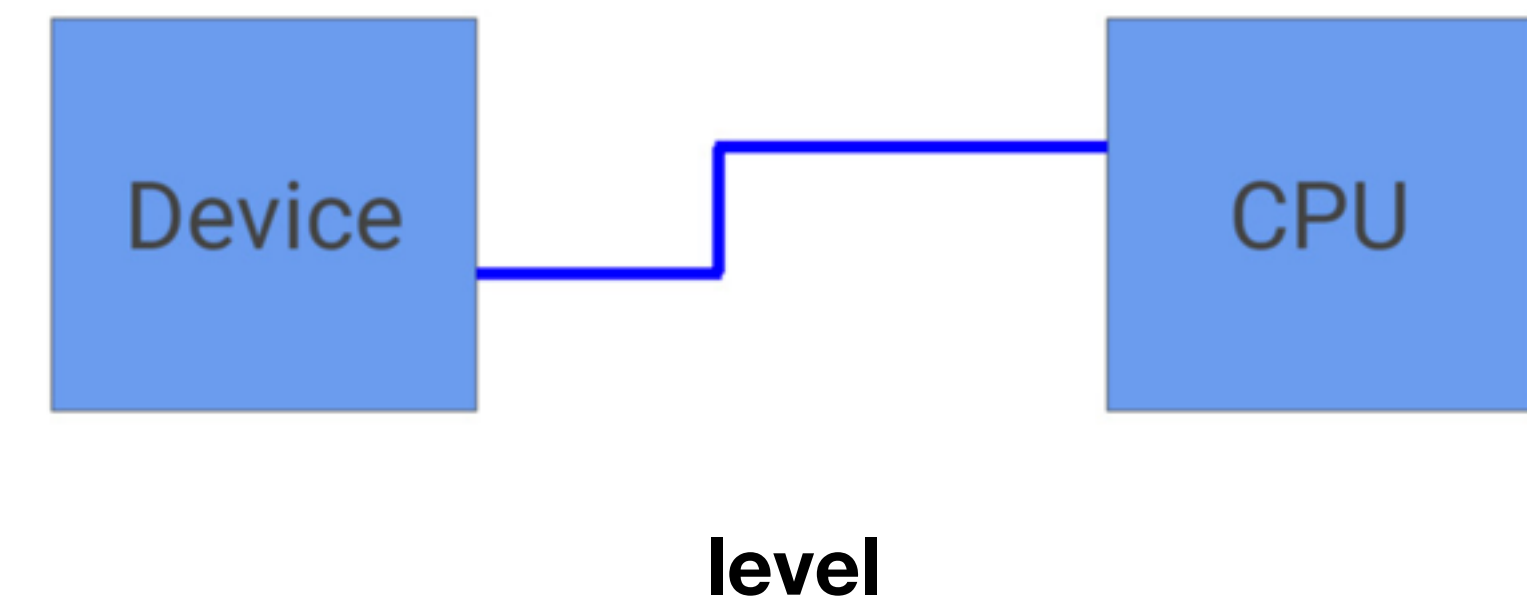
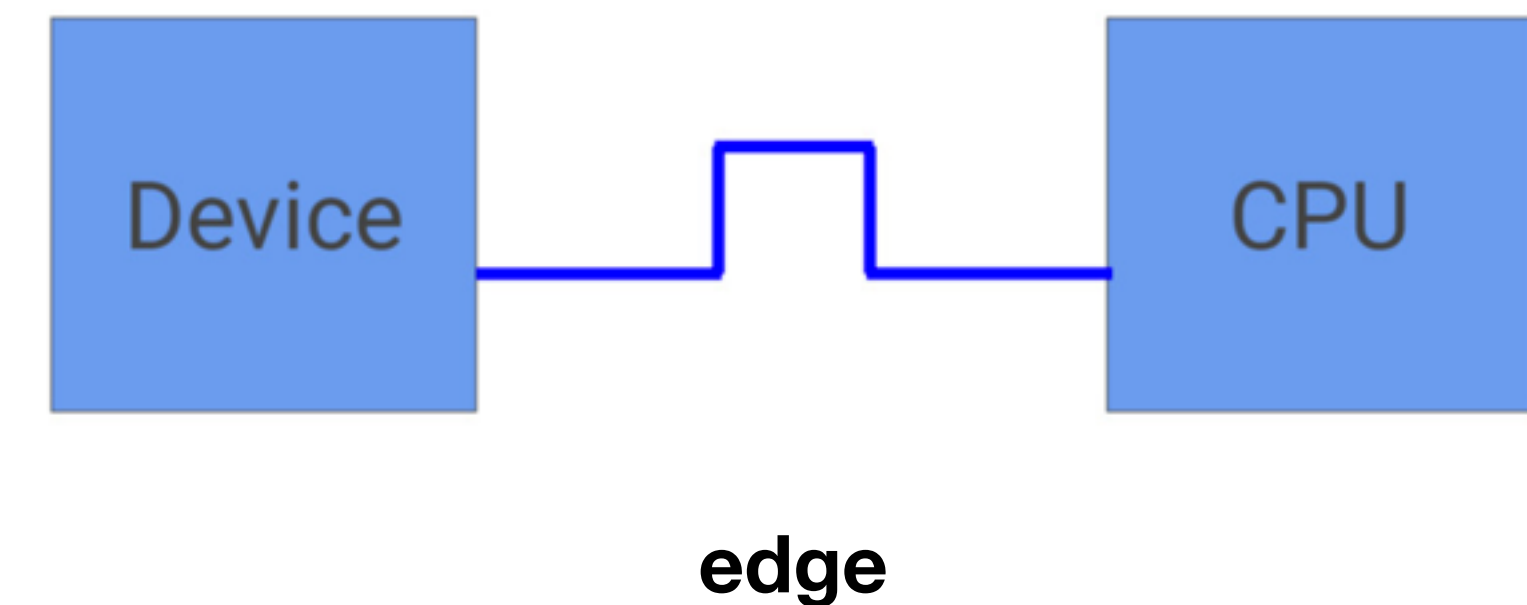
1.“Watch” object change

2.Decide next step based on state change

- not edge driven (event)
- level driven (state)

- **Goals**

- loose coupling
- high performance
- customization and extensibility

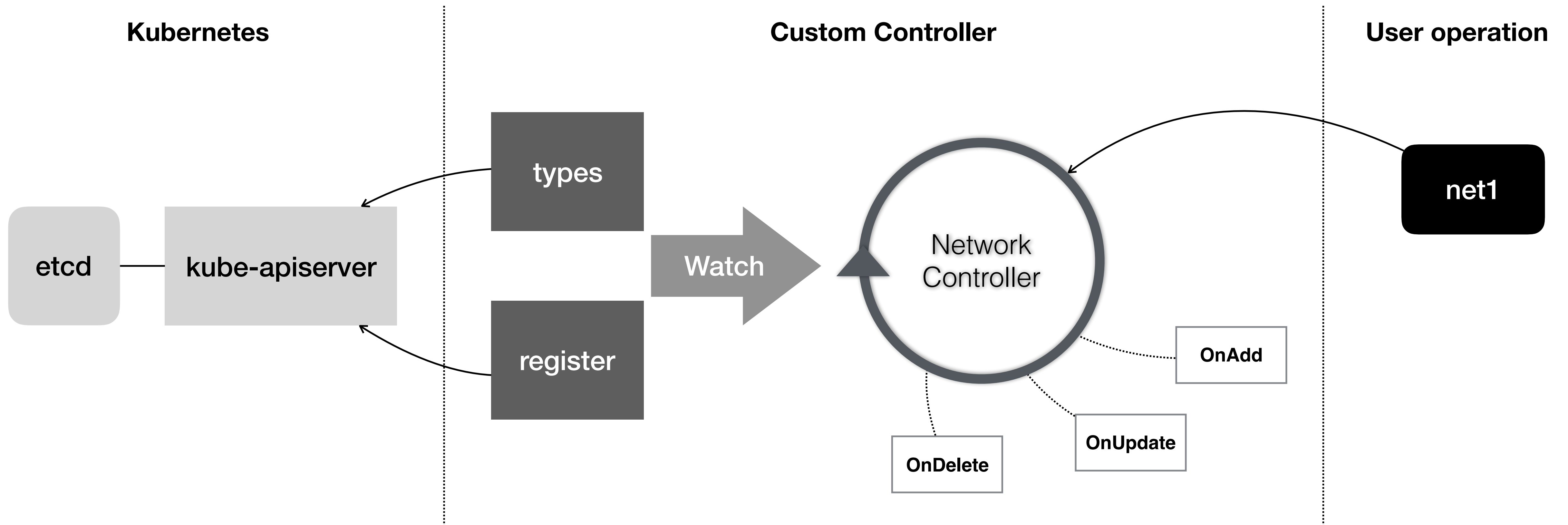


Write Your Own API Object!

- I want to have a **Network** object into k8s API
- I want a **Network Controller** to handle **add/update/delete** of all Network instances

```
$ kubectl get network  
  
NAME          KIND  
net1          network.v1.cr.client-go.k8s.io
```


My Network CRD & Controller



A Real World Example

```
70 func NewNetworkController(kubeClient kubernetes.Interface, osClient openstack.Interface) {
71     // initialize CRD if it does not exist
72     _, err := kubecrd.CreateNetworkCRD(kubeExtClient)
73     if err != nil && !apierrors.IsAlreadyExists(err) {
74         return nil, fmt.Errorf("failed to create CRD to kube-apiserver: %v", err)
75     }
76
77     source := cache.NewListWatchFromClient(
78         osClient.GetCRDClient().Client(),
79         crv1.NetworkResourcePlural,
80         apiv1.NamespaceAll,
81         fields.Everything())
82
83     networkController := &NetworkController{
84         kubeClient: kubeClient,
85         osClient: osClient,
86         source: source,
87         networkInformer := cache.NewInformer(
88             source,
89             &crv1.Network{},
90             0,
91             cache.ResourceEventHandlerFuncs{
92                 AddFunc: networkController.onAdd,
93                 UpdateFunc: networkController.onUpdate,
94                 DeleteFunc: networkController.onDelete,
95             })
96     networkController.networkInformer = networkInformer
97
98     return networkController, nil
99 }
```

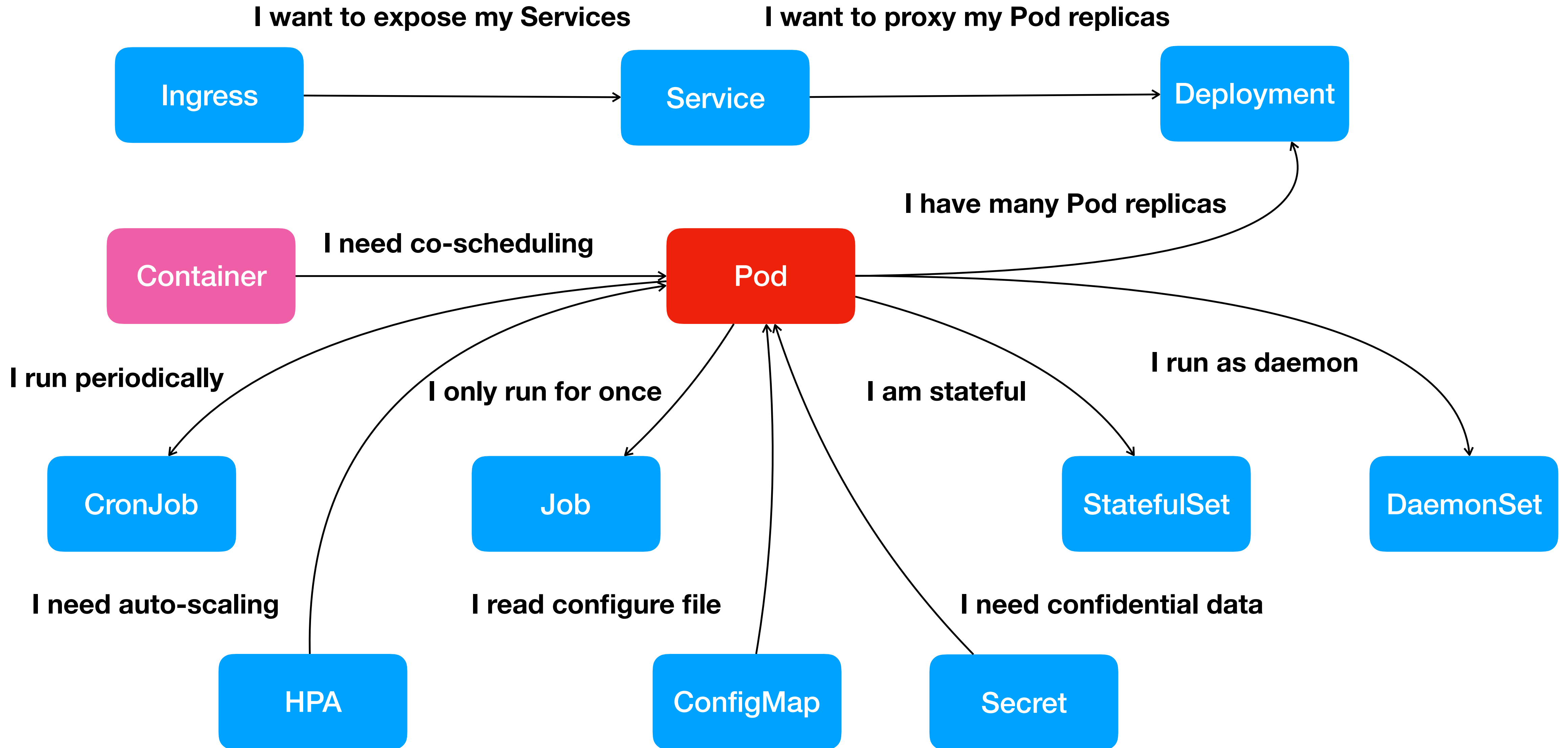
```
101 func (c *NetworkController) onAdd(obj interface{}) {
102     network := obj.(*crv1.Network)
103     // glog.Infof("[NETWORK CONTROLLER] OnAdd %v", network.ObjectMeta.SelfLink)
104     glog.Infof("[NETWORK CONTROLLER] OnAdd %v", network)
105
106     // NEVER modify objects from the store. It's a read-only, local cache.
107     // You can use networkScheme.Copy() to make a deep copy of original object and modify this copy
108     // Or create a copy manually for better performance
109     copyObj, err := c.kubeCRDClient.Scheme().Copy(network)
110     if err != nil {
111         glog.Errorf("ERROR creating a deep copy of network object: %v", err)
112         return
113     }
114 }
```

https://github.com/openstack/stackube/blob/master/pkg/network-controller/network_controller.go

```
82
83     networkController := &NetworkController{
84         kubeClient: kubeClient,
85         osClient: osClient,
86         source: source,
87         networkInformer := cache.NewInformer(
88             source,
89             &crv1.Network{},
90             0,
91             cache.ResourceEventHandlerFuncs{
92                 AddFunc: networkController.onAdd,
93                 UpdateFunc: networkController.onUpdate,
94                 DeleteFunc: networkController.onDelete,
95             })
96     networkController.networkInformer = networkInformer
97
98     return networkController, nil
99 }
```

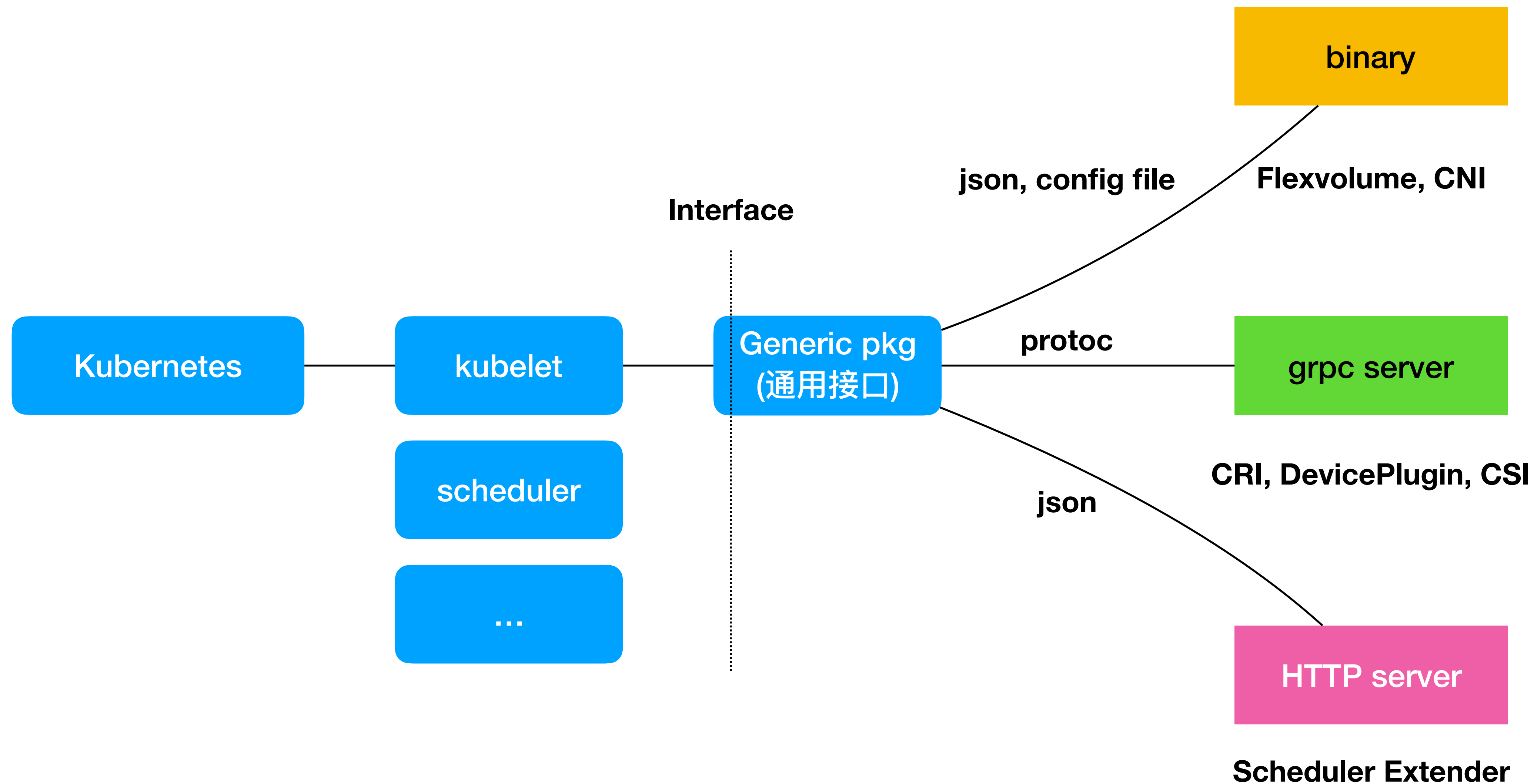
```
118 // 1. Create Network in Neutron
119 // 2. Update Network CRD object status to Active or Failed
120 err = c.addNetworkToDriver(networkCopy)
121 if err != nil {
122     glog.Errorf("Add network to driver failed: %v", err)
123     return
124 }
125
126 // create kube-dns in this namespace.
127 namespace := networkCopy.Namespace
128 if err := c.createKubeDNSDeployment(namespace); err != nil {
129     glog.Errorf("Create kube-dns deployment failed: %v", err)
130     return
131 }
132
133 if err := c.createKubeDNSService(namespace); err != nil {
134     glog.Errorf("Create kube-dns service failed: %v", err)
135     return
136 }
137 }
```

Control Panel



3. Extensibility

Kubernetes扩展通用模型



CNI

Leads: CoreOS, Tigera

1. 通用接口:

1. ADD: e.g. `networkPlugin.addToNetwork(...)`
2. DELETE: e.g. `networkPlugin.deleteFromNetwork(...)`

2. 实际操作:

1. kubelet创建network namespace (i.e. infra container的net ns)
2. kubelet调用CNI插件配置该network namespace

- e.g. ADD:

1. 将network interface (e.g. veth的一端) 插入到ns
2. 配置宿主机 (e.g. veth的另一端加入到bridge)
3. 配置net ns中的IP (e.g. 调用IPAM) 、路由等信息

3. 其他同Pod的容器共享这个network namespace

Flexvolume

Leads: Google, RedHat

1. 通用接口:

1. volume controller: `reconcile(actualWorld, desireWorld)`

1. attach(): volume provider -> host machine path

2. mount(): host machine path -> container path
(bind mount)

2. 实际操作 (e.g. my_driver是二进制文件) :

1. `./my_driver attach <json options> <node name>`

2. `./my_driver isattached <json options> <node name>`

3. `./my_driver mount <mount dir> <json options>`

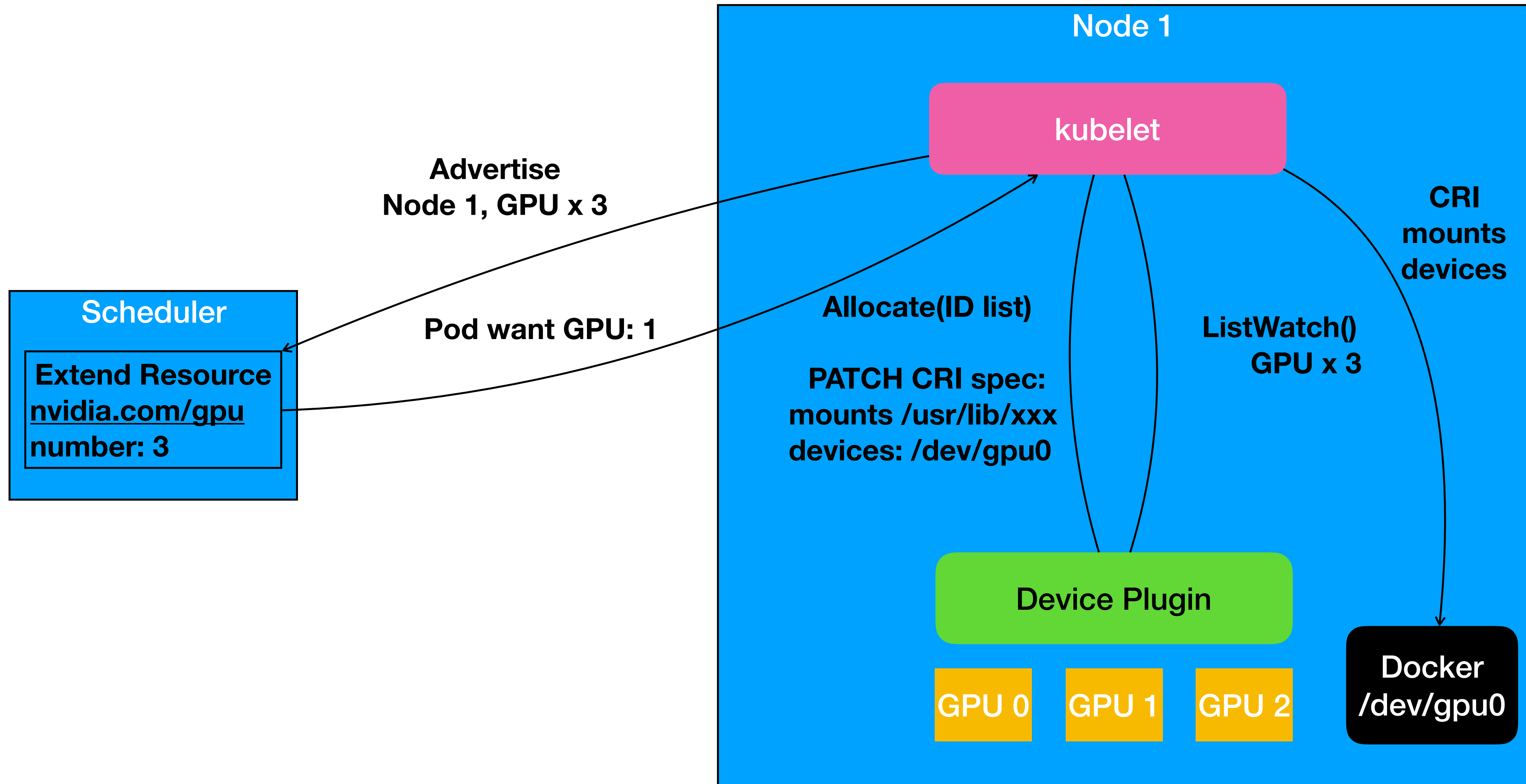
```
volumes:
- name: my-volume
  flexVolume:
    driver: "cinder/flexvolume_driver"
    fsType: ext4
    options: # data will be passed to binary as json
      cinderConfig: /etc/kubernetes/cinder.conf
      volumeID: daa7b4e6-1792-462d-ad47-78e900fed429
```


Device Plugin

Leads: NVIDIA, Intel, Google, RedHat

- 划重点:
 - Deprecate:
 - GPU support based on dockertools (coupled with Docker)
 - Prefer:
 - CRI + Device Plugin
 - Goals:
 - GPUs, High-performance NICs, FPGAs, InfiniBand, Storage devices etc

Device Plugin



Write Your Own Device Plugin

```
// Implements DevicePlugin service functions
func (ngm *nvidiaGPUManager) ListAndWatch(empty *pluginapi.Empty, stream pluginapi.Stream) (resp *pluginapi.ListAndWatchResponse, err error) {
    fmt.Printf("device-plugin: ListAndWatch start\n")
    changed := true
    for {
        for id, dev := range ngm.devices {
            state := ngm.GetDeviceState(id)
            if dev.Health != state {
                changed = true
                dev.Health = state
                ngm.devices[id] = dev
            }
        }
        if changed {
            resp.Devices = append(resp.Devices, ngm.devices)
        }
    }
}
```

Maintain device state

```
func (ngm *nvidiaGPUManager) Allocate(ctx context.Context, reqt *pluginapi.AllocateRequest) (*pluginapi.AllocateResponse, error) {
    resp := new(pluginapi.AllocateResponse)
    for _, id := range reqt.DevicesIDs {
        dev, ok := ngm.devices[id]
        if !ok {
            return nil, fmt.Errorf("Invalid allocation request with non-existing device ID: %s", id)
        }
        if dev.Health != pluginapi.Healthy {
            return nil, fmt.Errorf("Invalid allocation request with unhealthy device: %s", id)
        }
        devRuntime := new(pluginapi.DeviceRuntimeSpec)
        devRuntime.Devices = append(devRuntime.Devices, &pluginapi.DeviceSpec{
            HostPath:      "/dev/" + id,
            ContainerPath: "/dev/" + id,
            Permissions:    "mrw",
        })
        for _, d := range ngm.defaultDevices {
            devRuntime.Devices = append(devRuntime.Devices, &pluginapi.DeviceSpec{
                HostPath: d,
            })
        }
    }
    resp.Spec = append(resp.Spec, devRuntime)
    return resp, nil
}
```

Allocate devices

<https://github.com/GoogleCloudPlatform/container-engine-accelerators>

```
    }
    fmt.Printf("ListAndWatch: send devices %v", resp)
    if err := stream.Send(resp); err != nil {
        fmt.Printf("device-plugin: cannot update device state: %v", err)
        ngm.grpcServer.Stop()
        return err
    }
    changed = false
    time.Sleep(5 * time.Second)
}
```

```
    }
    devRuntime.Mounts = append(devRuntime.Mounts, &pluginapi.Mount{
        ContainerPath: path.Join(ContainerPathPrefix, "lib64"),
        HostPath:      path.Join(HostPathPrefix, "lib"),
        ReadOnly:      true,
    })
    devRuntime.Mounts = append(devRuntime.Mounts, &pluginapi.Mount{
        ContainerPath: path.Join(ContainerPathPrefix, "bin"),
        HostPath:      path.Join(HostPathPrefix, "bin"),
        ReadOnly:      true,
    })
    resp.Spec = append(resp.Spec, devRuntime)
    return resp, nil
}
```

Allocate mounts

Tips

- Next goal: generic API for heterogeneous environment
 - GPU memory, NVLink etc in Device description (instead of qValue).
 - More device types are coming.
- 拥抱变化，但不要走偏！

Summary

1. Control Panel

- How k8s describe my workloads in container way?
- Pod, 容器设计模式, Control Panel (sig-app)
- **Use!**

2. Plugins & extensibility

- How to make k8s works in my way?
- 扩展性设计: Controller, CNI, CRI, CSI, Device Plugin
- **Hack!**
- Enjoy!

Thanks!

Harry (Lei) Zhang

Twitter/Github/Wechat ID: @resouer
Kubernetes | Hyper | ZJU-SEL
harryzhang@zju.edu.cn

主办：

