Go Programming Pattern in Kubernetes Philosophy

Harry Zhang @resouer

Contents

- **What I will talk?**
  - Kubernetes basic in 1 min
  - **For Kubernetes developers:**
    - The Golang programming patterns of Kubernetes (Controller, codegen etc)
    - Write your own Controller
    - gPRC based interface design in Kubernetes (CRI as example)
  - **For Kubernetes users:**
    - Effective pattern of programming based on Kubernetes
  - 广告（Don’t worry, it’s not **that** kind of AD）

- **What I will not talk?**
  - Kubernetes usage and under the hood
  - Internal systems or commercial software

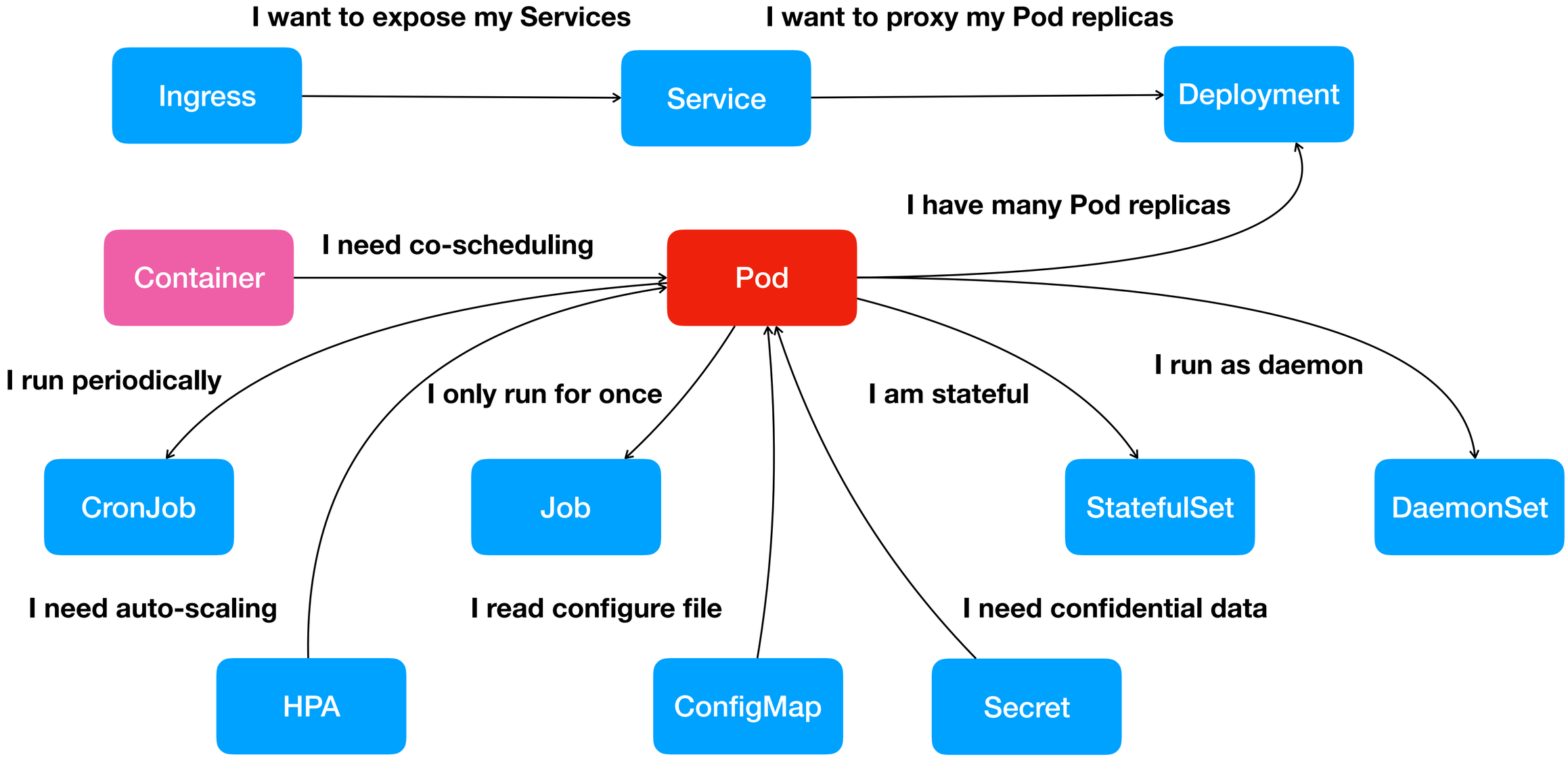
Kubernetes

- The **container** orchestration and management project created by Google
- Successor of Google **Borg/Omega** system
- One of the **most popular** open source projects in this world
- Written by, and heavily depends on Golang

Again: Why Go?

- All about **community**
- A sad story:
  - https://github.com/google/lmctfy
- Now think about a C/C++ based Kubernetes?
- And, well designed **programming patterns** with powerful **extensibility**

Understand Kubernetes in 1 min



Understand Kubernetes in 2 min

- kubectl run nginx —image=nginx:1.7.9 —replicas=3
- kubectl create -f deployment.yaml
- kubectl create -f hpa.yaml
- **API Object Oriented Programming**

```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

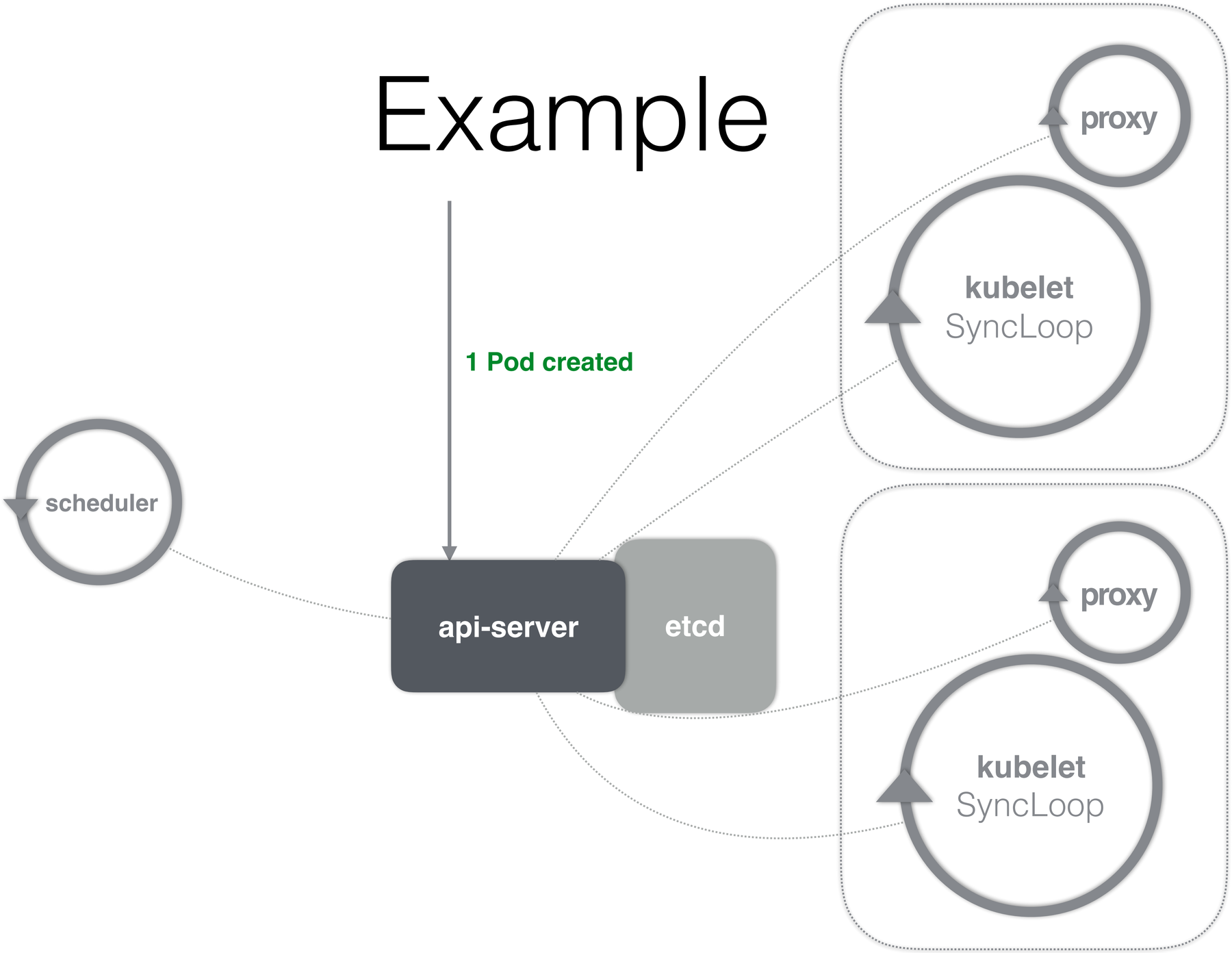
```yaml
apiVersion: autoscaling/v2alpha1
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
  namespace: default
spec:
  scaleTargetRef:
    apiVersion: apps/v1beta1
    kind: Deployment
    name: nginx-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 50
```

Core of API “OO”

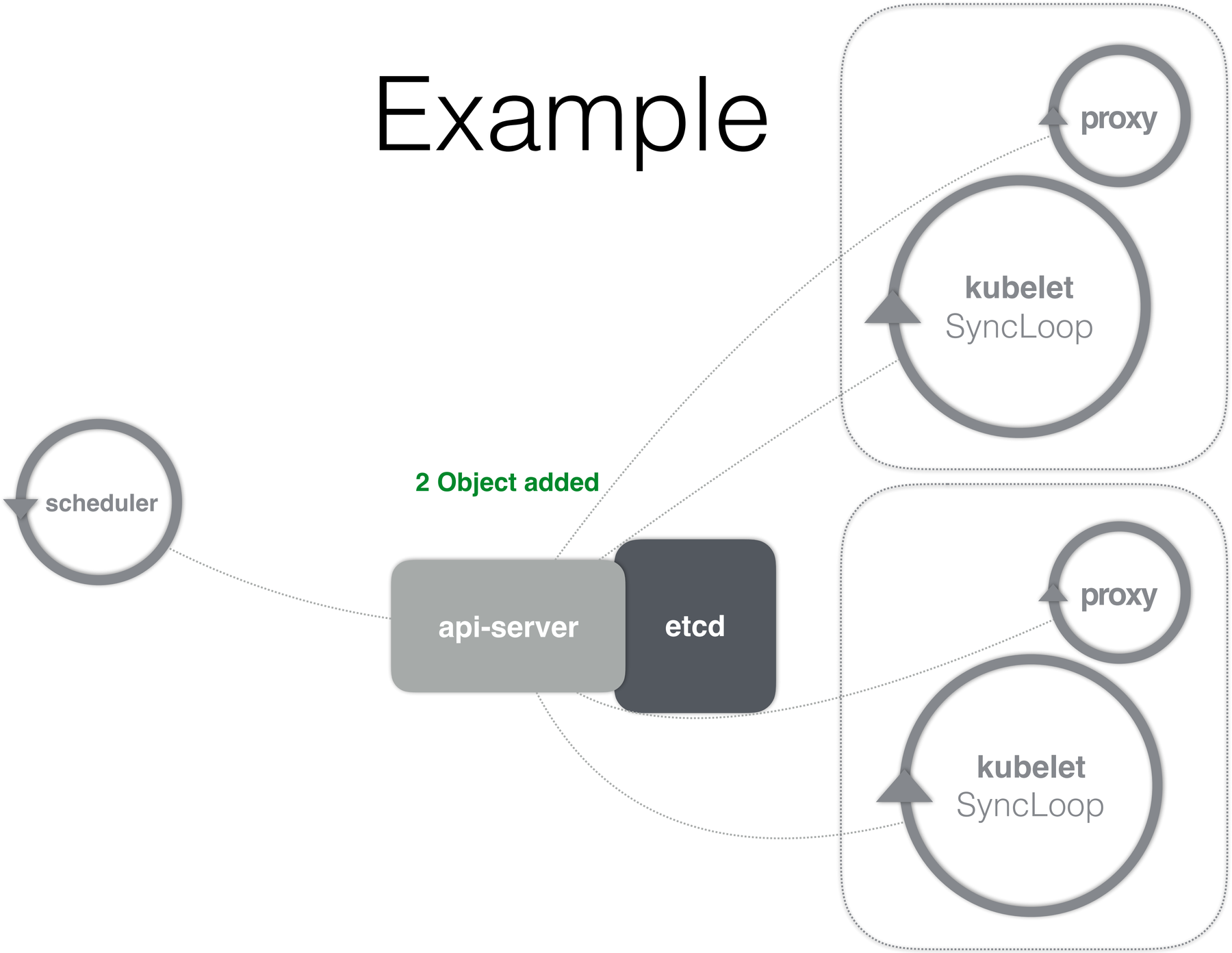
1. API objects stores in etcd

2. Control loops (Sync Loop) to reconcile API objects

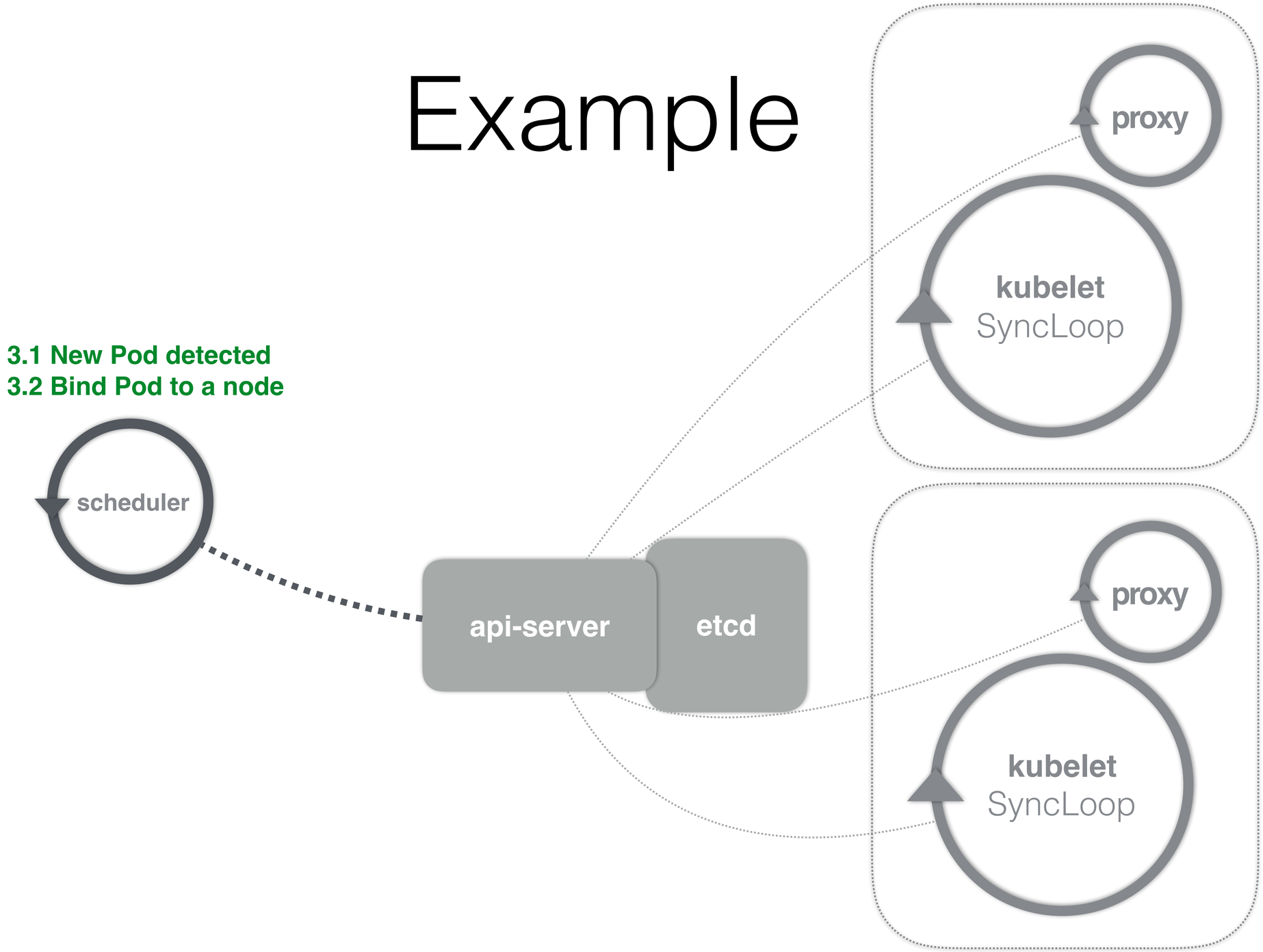
Example


1 Pod created
scheduler
api-server
etcd
proxy
kubelet
SyncLoop
proxy
kubelet
SyncLoop

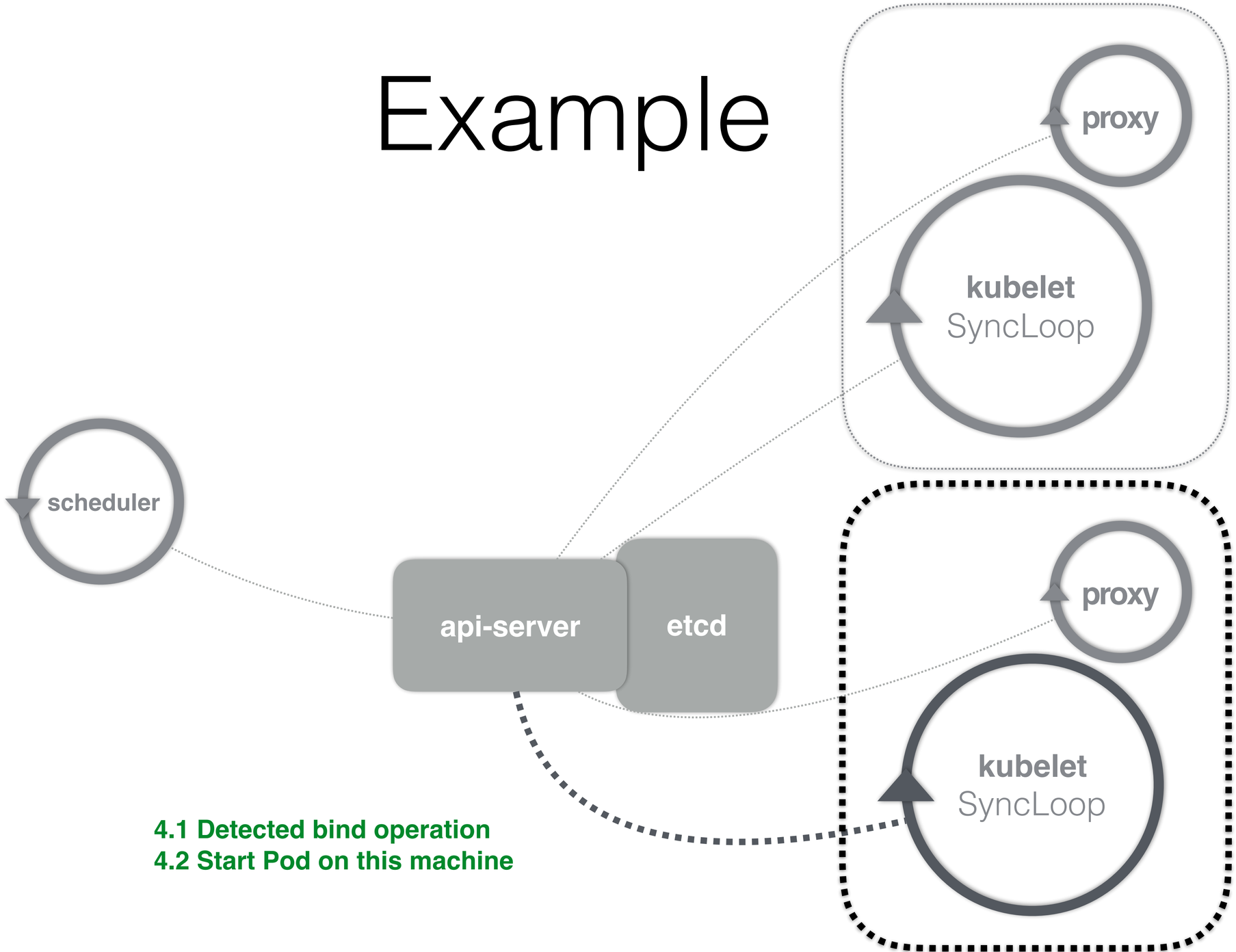
Example


proxy
kubelet
SyncLoop
scheduler
2 Object added
api-server
etcd
proxy
kubelet
SyncLoop

Example


3.1 New Pod detected
3.2 Bind Pod to a node
scheduler
api-server
etcd
proxy
kubelet
SyncLoop
proxy
kubelet
SyncLoop

Example



4.1 Detected bind operation
4.2 Start Pod on this machine

Pattern 1: Controller

- Control everything by **Controller**
  - **Level driven, not edge driven**

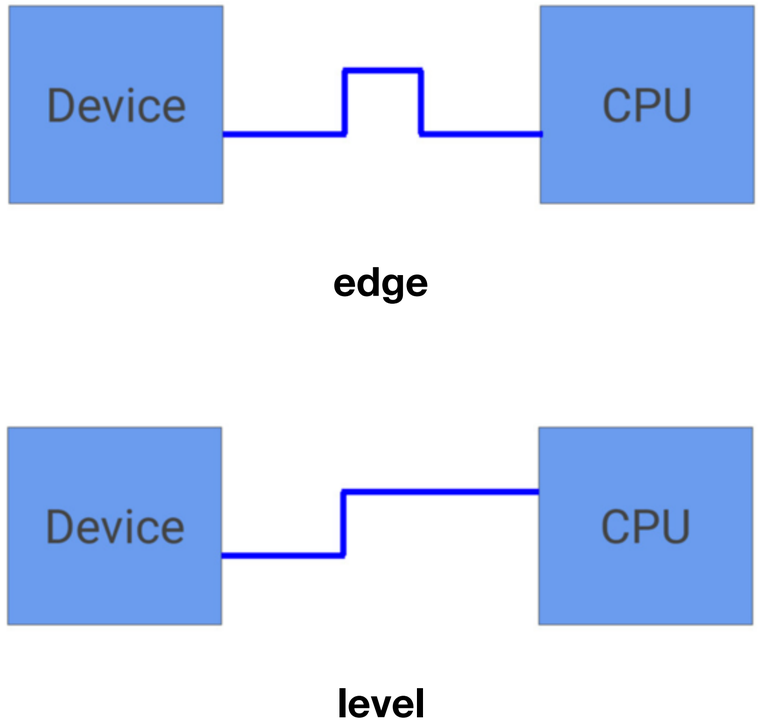


Image: https://speakerdeck.com/thockin/edge-vs-level-triggered-logic

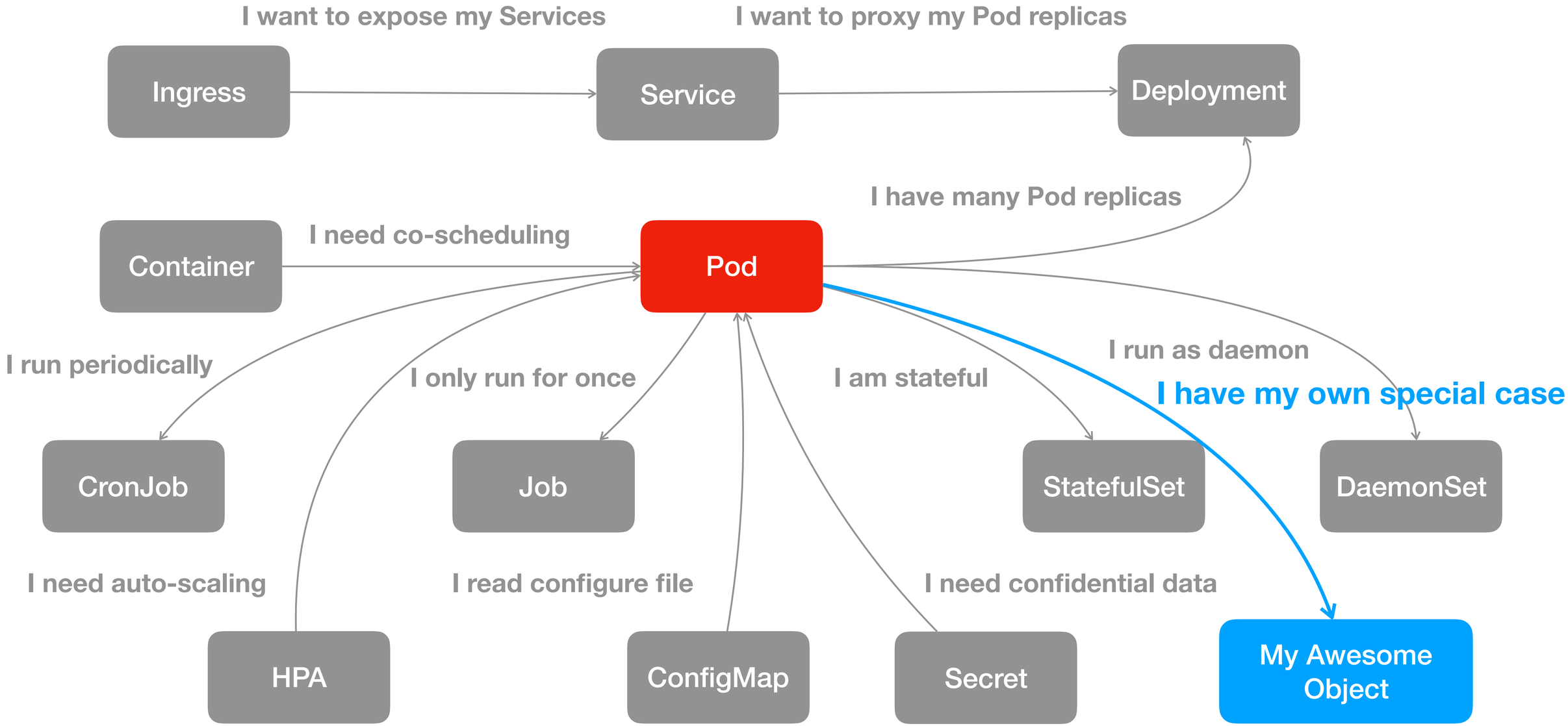
Controller

- The heart of Kubernetes orchestrator
  - drives the cluster state based on the changes to the API objects

- **Write your own controller!**

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

Why DIY?


I want to expose my Services
I want to proxy my Pod replicas
Ingress
Service
Deployment
I have many Pod replicas
Container
I need co-scheduling
Pod
I run periodically
I only run for once
I am stateful
I run as daemon
I have my own special case
CronJob
Job
StatefulSet
DaemonSet
I need auto-scaling
I read configure file
I need confidential data
HPA
ConfigMap
Secret
My Awesome Object

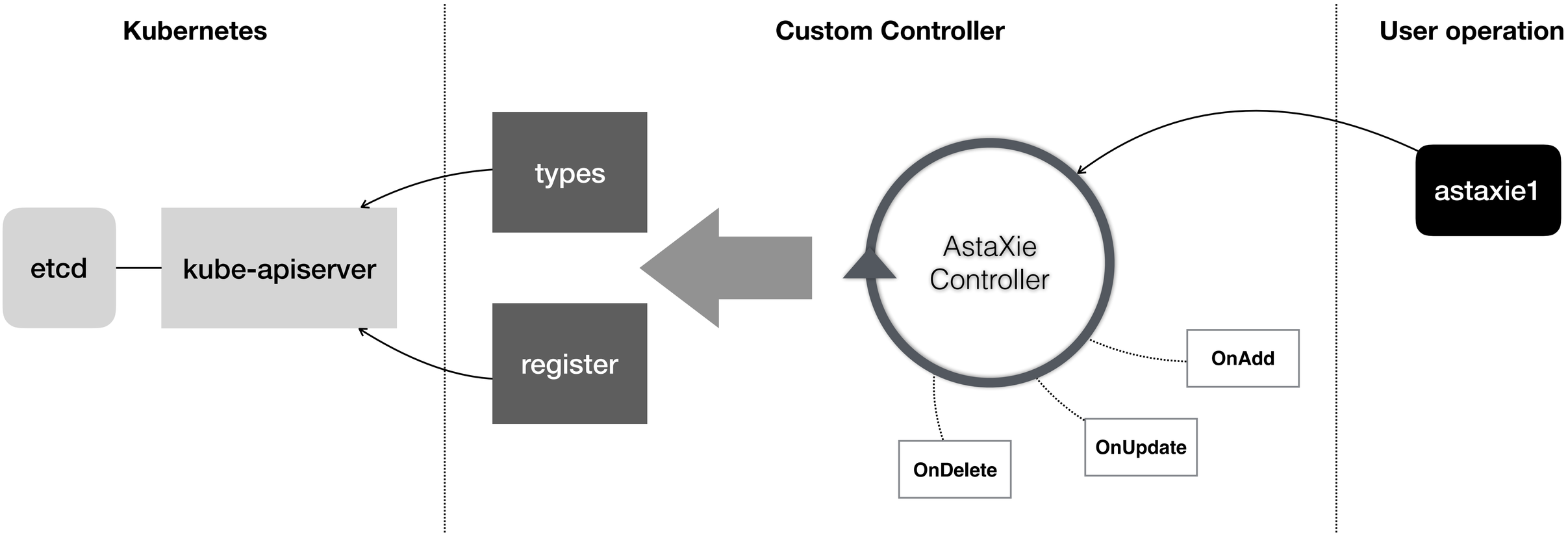
Demo

- I want to have a **Asta Xie** object into k8s API
- I want a controller to handle **add/update/delete** of all Asta Xie instances

```
$ kubectl get astaxie

NAME        KIND
astaxie1    AstaXie.v1.cr.client-go.k8s.io
```

My AstaXie Object & Controller



A Real World Example

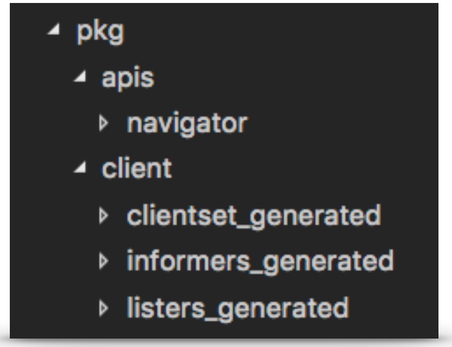
- I want to have a **Network** object into k8s API
- I want a controller to handle **add/update/delete** of all Network instances
  - onAdd: create Neutron network
  - onDelete: delete Neutron network
  - onUpdate: update Network object status
- https://github.com/openstack/stackube/blob/master/pkg/network-controller/network_controller.go

Pattern 2: Gode Generator

- **client-gen**: generate typed Kubernetes AP client for type
  - client.Pod.Get().Resource(…).Do()
- **conversion-gen**: seamless upgrades between API versions
  - apiVersion: k8s.io/v1alpha1 -> apiVersion: k8s.io/v1beta1
- **deepcopy-gen:** deepcopy
  - go get k8s.io/kubernetes/vendor/k8s.io/kube-gen/cmd/deepcopy-gen
  - deepcopy-gen -i ./pkg/apis/v1
- **defaulter-gen:** set default values for fields
- **go-to-protobuf:** generate protobuf messages for your types
- **informer-gen:** generate informers that can be used to watch for updates to your types
- **openapi-gen:** generate openapi compatible API documentation

```
func autoConvert_api_Affinity_To_v1_Affinity(in *api.Affinity, out *v1.Affinity, s conversion.Scope)
    cut.NodeAffinity = (*v1.NodeAffinity)(unsafe.Pointer(in.NodeAffinity))
    cut.PodAffinity = (*v1.PodAffinity)(unsafe.Pointer(in.PodAffinity))
    cut.PodAntiAffinity = (*v1.PodAntiAffinity)(unsafe.Pointer(in.PodAntiAffinity))
    return nil
}
```

```
1  // Deployment enables declarative updates for Pods and ReplicaSets.
2  message Deployment {
3    // Standard object metadata.
4    // +optional
5    optional k8s.io.apimachinery.pkg.apis.meta.v1.ObjectMeta metadata = 1;
6
7    // Specification of the desired behavior of the Deployment.
8    // +optional
9    optional DeploymentSpec spec = 2;
10
11   // Most recently observed status of the Deployment.
12   // +optional
13   optional DeploymentStatus status = 3;
14 }
15
```



More Reference

- github.com/kubernetes/gengo
- github.com/kubernetes/kubernetes/tree/master/cmd/libs/go2idl

Pattern 3: gRPC based Interface

- **Decouple** Kubernetes from external dependencies
- kubelet -> **gRPC** -> dockershim -> dockerd
- **go2idl**: gogoprotobuf based protobuf gen

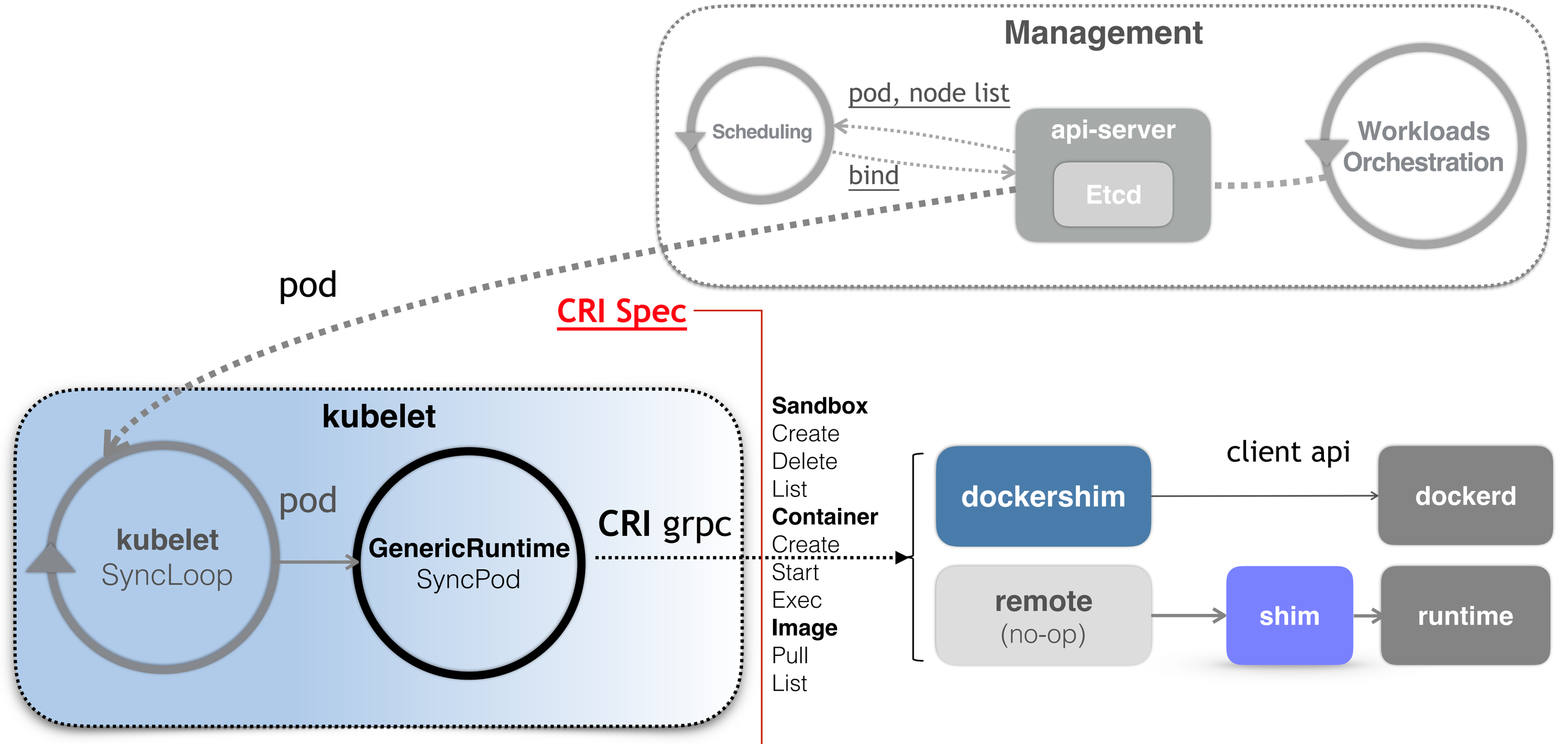
```go
// PodSandboxManager contains basic operations for sandbox.
type PodSandboxManager interface {
    Create(config *PodSandboxConfig) (string, error)
    Delete(id string) (string, error)
    List(filter PodSandboxFilter) []PodSandboxListItem
    Status(id string) PodSandboxStatus
}

// ContainerRuntime contains basic operations for containers.
type ContainerRuntime interface {
    Create(config *ContainerConfig, sandboxConfig *PodSandboxConfig, PodSandboxID string) (string, error)
    Start(id string) error
    Stop(id string, timeout int) error
    Remove(id string) error
    List(filter ContainerFilter) ([]ContainerListItem, error)
    Status(id string) (ContainerStatus, error)
    Exec(id string, cmd []string, streamOpts StreamOptions) error
}

// ImageService contains image-related operations.
type ImageService interface {
    List() ([]Image, error)
    Pull(image ImageSpec, auth AuthConfig) error
    Remove(image ImageSpec) error
    Status(image ImageSpec) (Image, error)
    Metrics(image ImageSpec) (ImageMetrics, error)
}

type ContainerMetricsGetter interface {
    ContainerMetrics(id string) (ContainerMetrics, error)
}
```

CRI


Management
Scheduling
pod, node list
bind
api-server
Etcd
Workloads
Orchestration
pod
CRI Spec
kubelet
kubelet
SyncLoop
pod
GenericRuntime
SyncPod
CRI grpc
Sandbox
Create
Delete
List
Container
Create
Start
Exec
Image
Pull
List
dockershim
client api
dockerd
remote
(no-op)
shim
runtime

Deployment

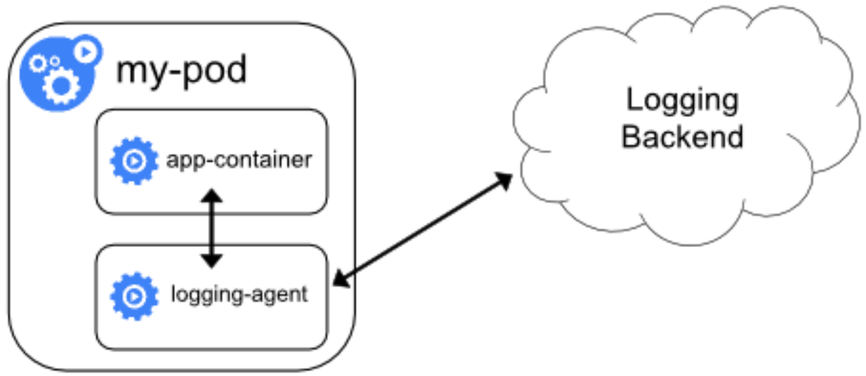
- yum install -y kubelet kubeadm kubectl
- sed -i '2 i\Environment="KUBELET_EXTRA_ARGS=--container-runtime=remote **--container-runtime-endpoint=/var/run/xxx.sock** --feature-gates=AllAlpha=true"' /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
- kubeadm init
- kubeadm join --token $token ${master_ip:port}
- **Done!**

But that’s only one part …

- **Kubernetes** is also about **design pattern** in **container** world
  - decoupling containers
  - re-use images
  - well-designed architecture for your container workloads
- **“How can I build distributed micro-services with container?”**

Programming Pattern

- Sidecar




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-app
spec:
  containers:
  - name: app-container
    image: gcr.io/google_containers/testapp:v1
    volumeMounts:
    - name: varlog
      mountPath: /var/log
  - name: logging-agent
    image: gcr.io/google_containers/fluentd:1.30
    env:
    - name: FLUENTD_ARGS
      value: -c /etc/fluentd-config/fluentd.conf
    volumeMounts:
    - name: varlog
      mountPath: /var/log
    - name: config-volume
      mountPath: /etc/fluentd-config
  volumes:
  - name: varlog
    emptyDir: {}
  - name: config-volume
    configMap:
      name: fluentd-config
```

Programming Pattern

- InitContainer

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: init-demo
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
    - name: workdir
      mountPath: /usr/share/nginx/html
  # These containers are run during pod initialization
  initContainers:
  - name: install
    image: busybox
    command:
    - wget
    - "-O"
    - "/work-dir/index.html"
    - http://kubernetes.io
    volumeMounts:
    - name: workdir
      mountPath: "/work-dir"
  dnsPolicy: Default
  volumes:
  - name: workdir
    emptyDir: {}
```

Programming Pattern

- 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/hightowerlabs/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
      env:
      - name: FLUENTD_ARGS
        value: -c /etc/fluentd-config/fluentd.conf
      volumeMounts:
      - name: varlog
        mountPath: /var/log
      - name: config-volume
        mountPath: /etc/fluentd-config
  volumes:
  - name: varlog
    emptyDir: {}
  - name: config-volume
    configMap:
      name: fluentd-config
```

Automatically Inject

Summary

1. How Kubernetes is using Golang?
    1. What is Kubernetes and how it works (1 mins)
    2. The heart of Kubernetes orchestration: **Controller**
    3. Write **your own Controller** with CRD
    4. code gen for deep copy, API conversion, API doc, encoding/decoding etc
    5. gRPC based interface **(e.g. CRI)**
2. How we can do better to use Kubernetes?
    1. **Programming Patterns** in Kubernetes
        1. this is the main difference of Kubernetes with others
        2. think about why everyone loves Borg

Come and Join, Gophers!

广告时间

END

Harry Zhang @resouer