service mesh那些事儿

kubernetes and istio

- xiaorui.cc

Istio

为什么要微服务?

- 微服务的优点
  - 松耦合，代码结构更加清晰
  - 开发者友好，避免老代码包袱.
  - 独立发布、快速迭代
  - 故障隔离
  - 增加重用，可组合
  - 针对性横向扩展

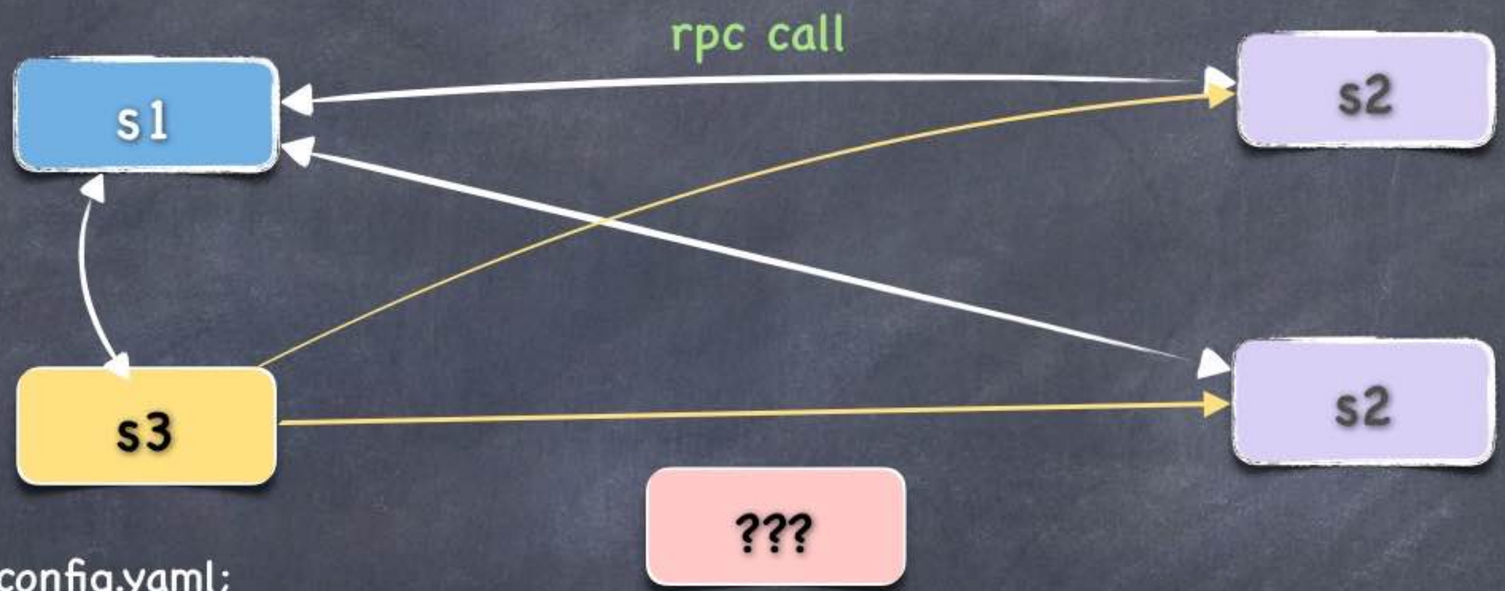
服务发现调度的演变

- 静态配置
- 单体结构

- 中心调度
- 构建sdk
- 消息总线
- service mesh

Istio

服务发现之静态配置



get hosts from config.yaml;

```
upstream:
  s2:
    - 10.x.x.1
    - 10.x.x.2
```

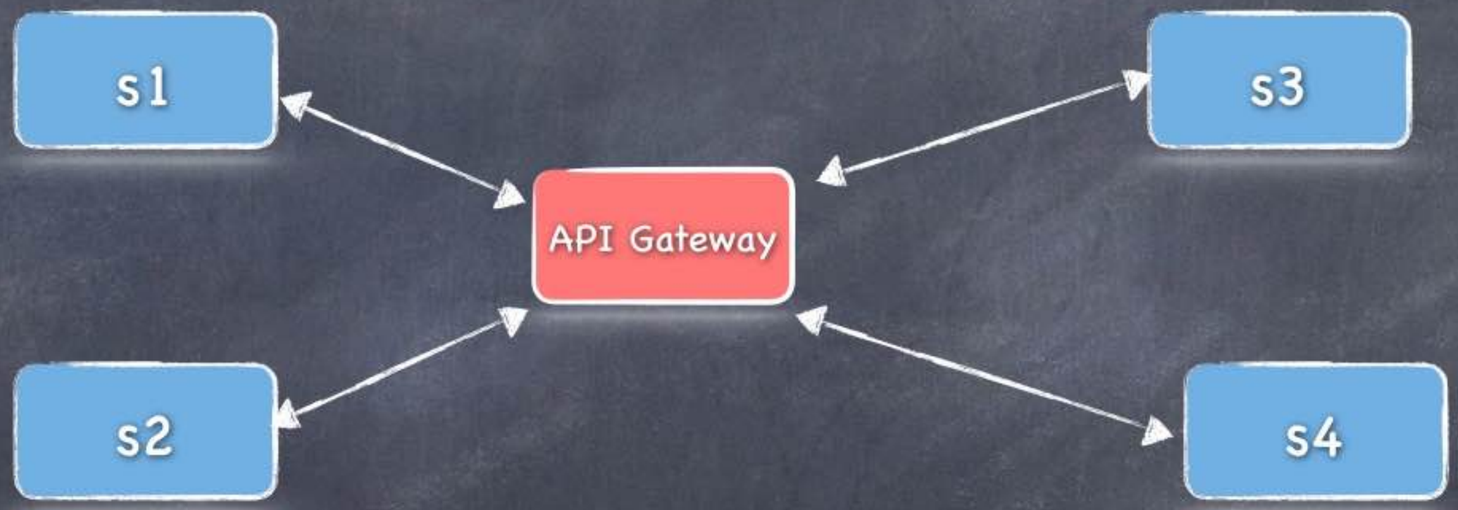
- 频繁上线
- 手动维护配置

Istio

服务发现之负载均衡

static config

```
upstream:
  s1:
    - 10.x.x.1
s2:
    - 10.x.x.2
...
```

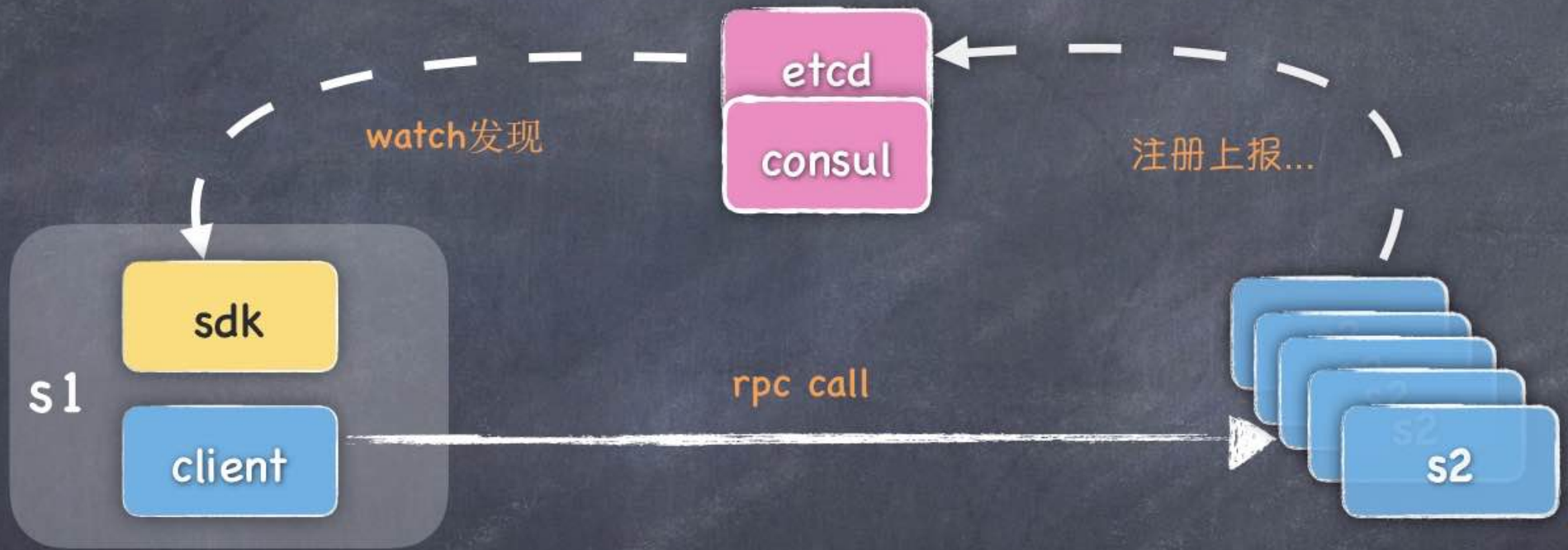


- dynamic config
  - envoy
  - kong
  - openresty upsync

- 中心节点

Istio

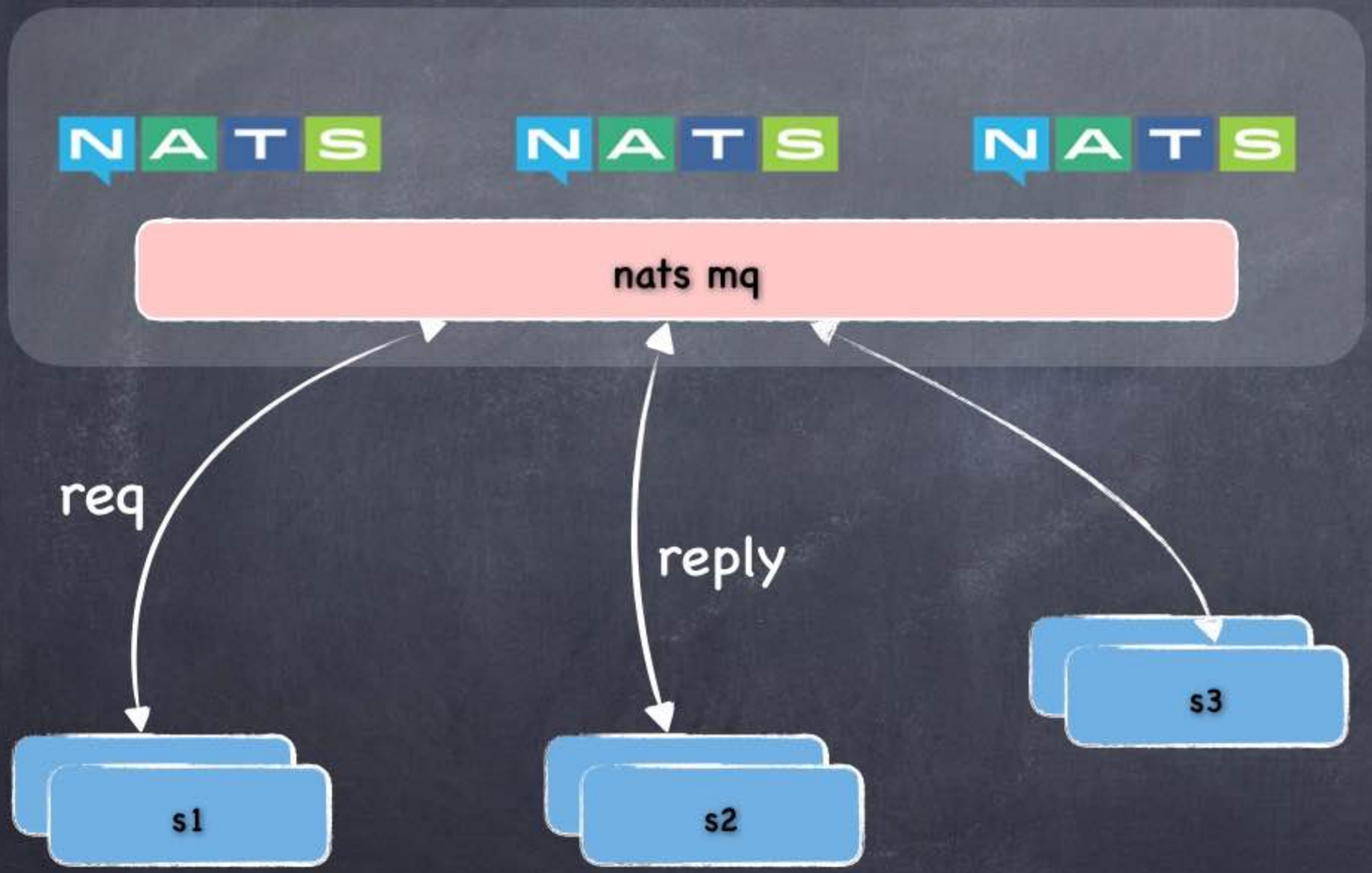
服务发现之sdk



- 开发难度不简单？
- 多种语言重复性开发？

Istio

消息总线



- 优点
  - 开发快, 架构易理解
  - go-micro/go-kit也集成该方案
- 缺点
  - 超时控制
  - topic分区 ?
  - 流量管理
  - 灰度发布

Istio

Service Mesh

Istio

Service mesh

- 帮你注册
- 帮你发现
- 帮你访问
- 帮你策略

- 轻量级代理，各类决策，负载调度
- 应用程序无感知
- 解耦应用程序的限频，熔断，重试/ 超时
- 灰度上线
- ...

还存在的问题?

- 自己开发servcie mesh ?
- 找一个service mesh ?
  - Istio
  - Spring cloud
  - Dubbo
  - Alipay SofaMesh
  - ...

k8s <-> istio

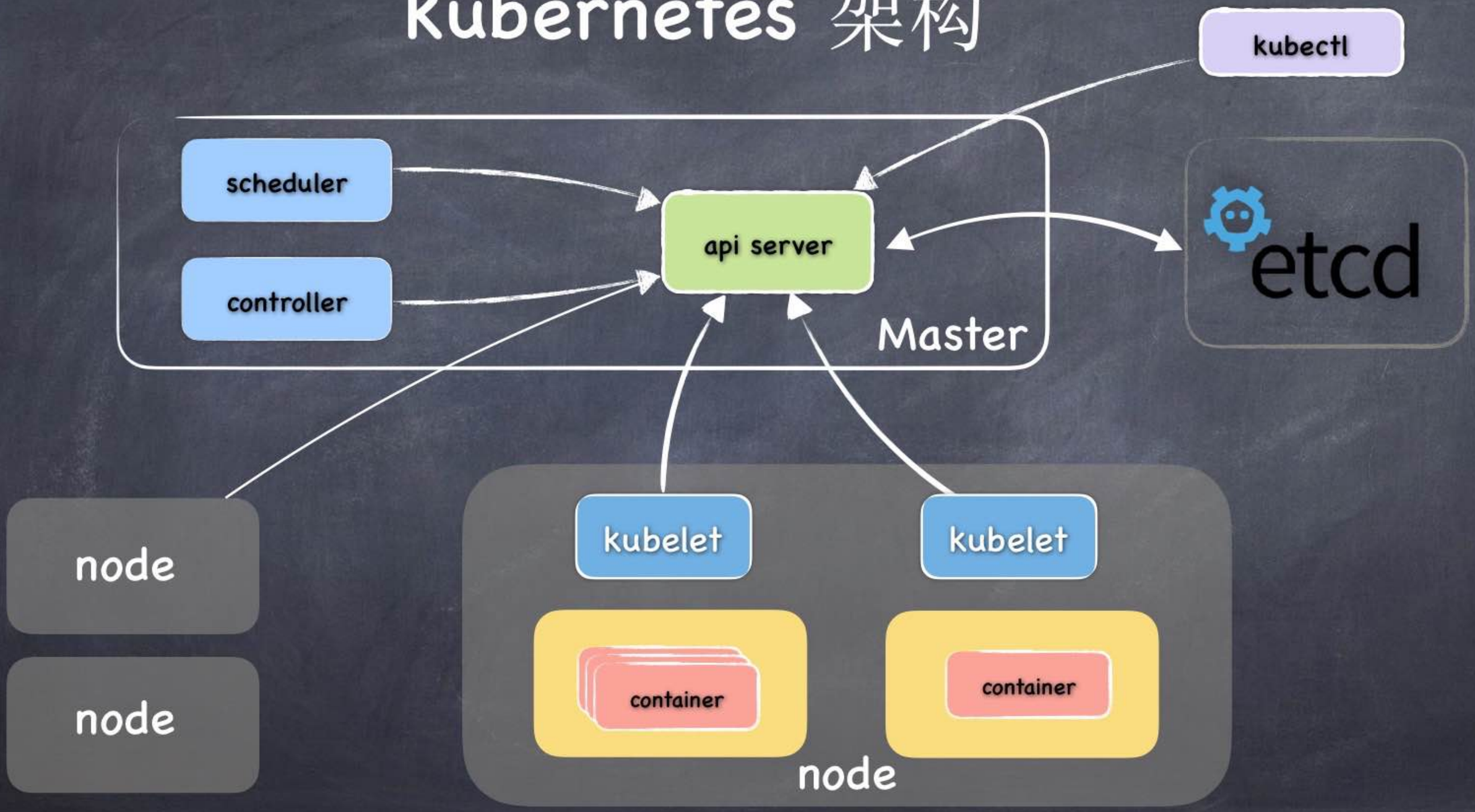
- 部署及管理
  - 大规模部署
  - 灵活部署
  - 集群式管理
  - ...

what is kubernetes ?

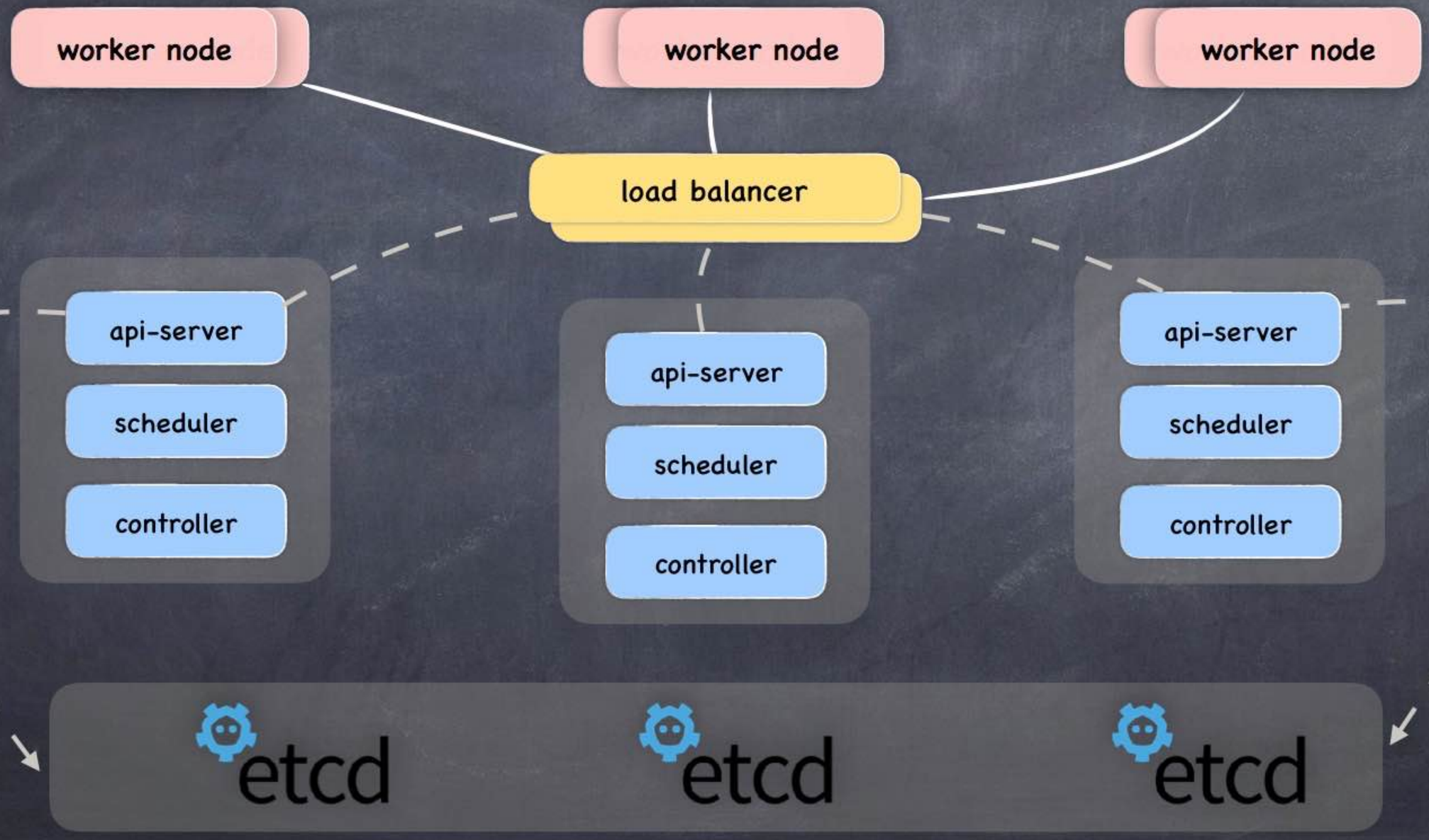
- 基于容器的集群编排引擎

- 扩展集群
- 滚动升级回退
- 弹性伸缩服务
- 自动治愈
- 服务发现
- 资源配额
- 灵活扩展API

kubernetes 架构



kubernetes ha



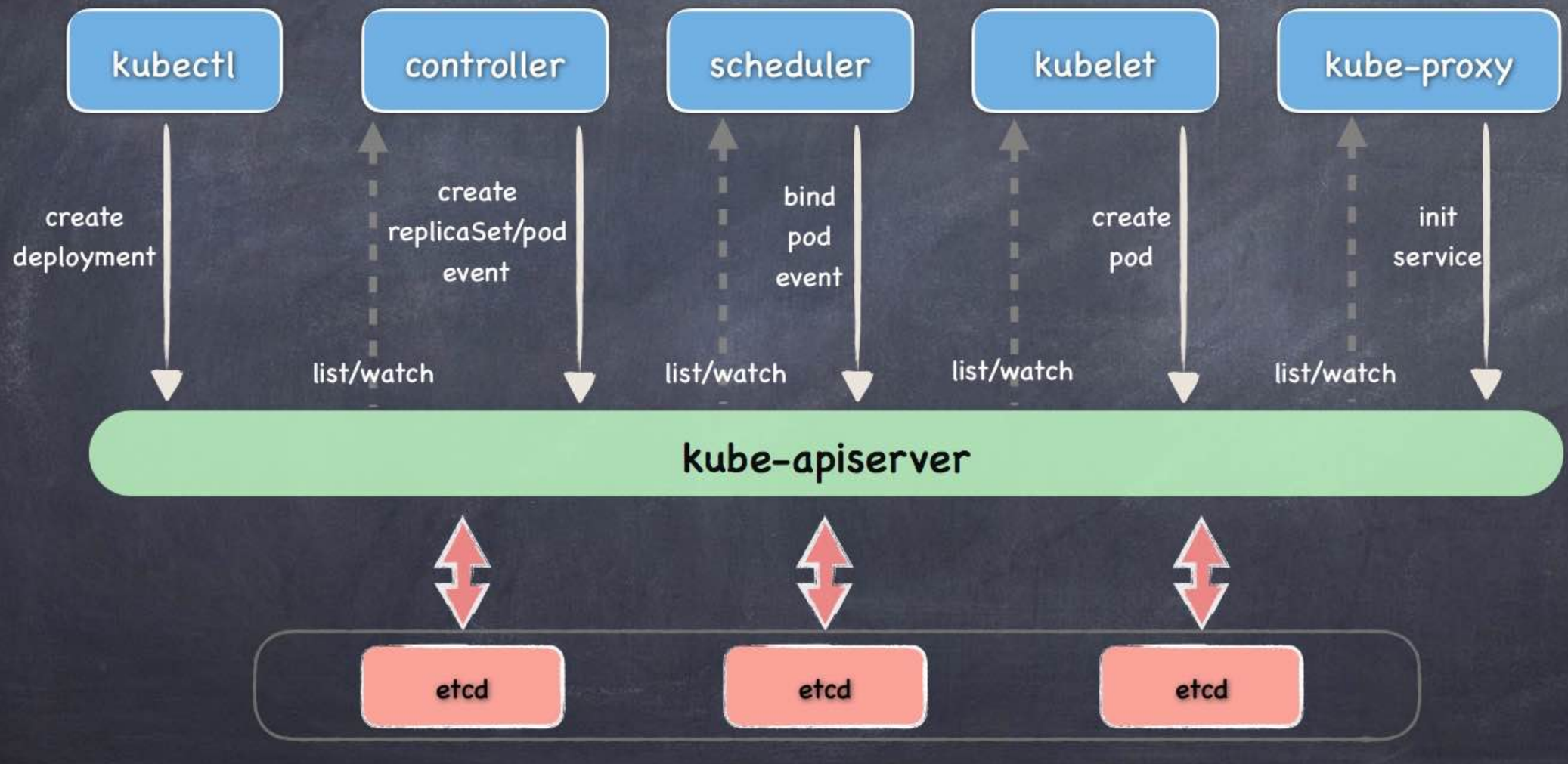
kubernetes 架构

- maser
  - api server
    - 总操作入口
  - controller
    - 控制中心
  - scheduler
    - pod调度器
- node
  - kubelet
    - 管理容器的生命周期
    - 监控
    - 上报节点状态
  - kube-proxy
    - 管理service

kubernetes notion

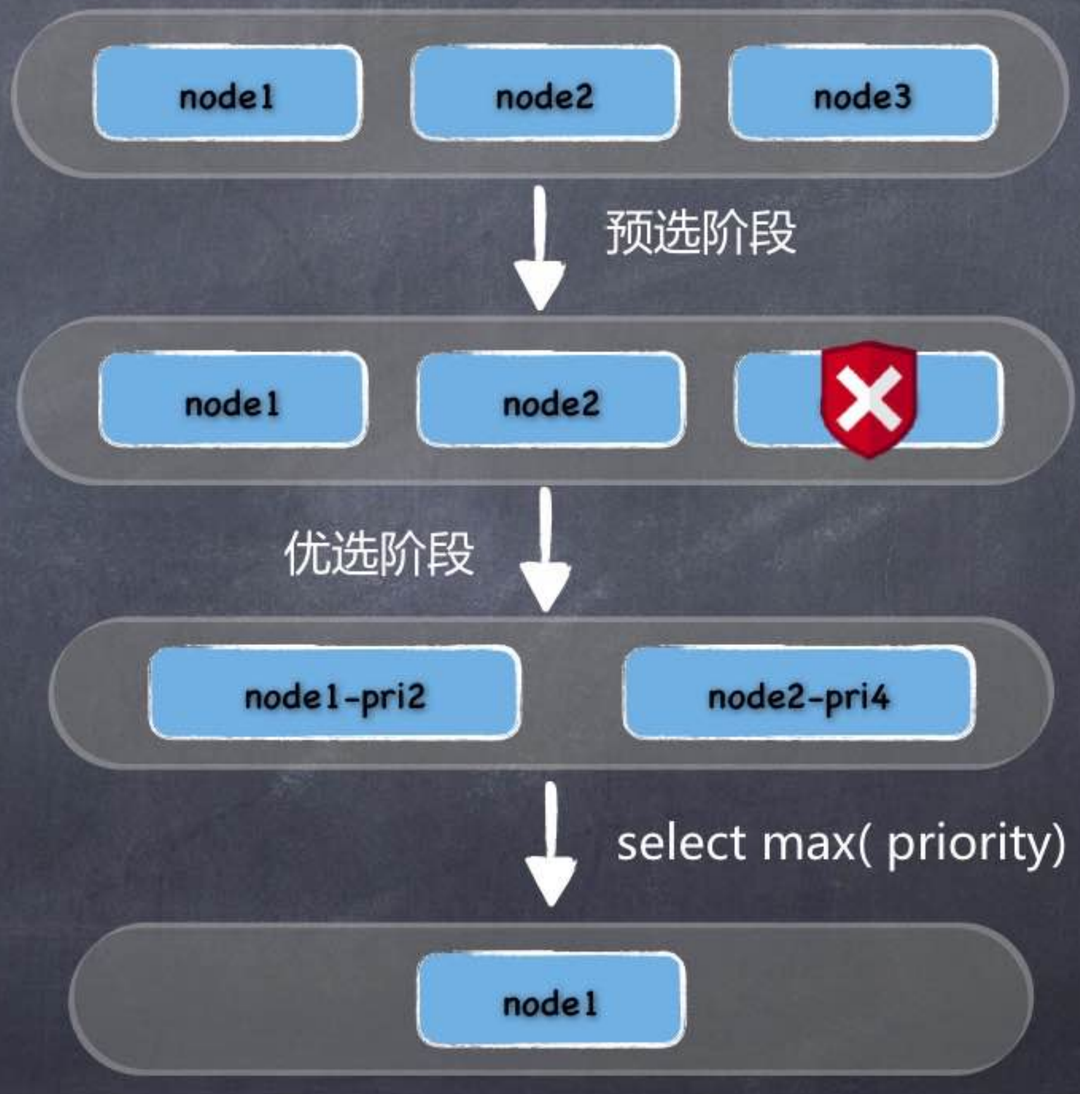
- Pod 最小单位
- Deployment
- Service
- RepliaSet
- StatefulSet
- DaemonSet
- Crontab
- Job
- ConfigMap
- Label
  - node
    - disktype=ssd
    - gpu=true
  - pod
    - app
    - version
  - ...

cretae deployment process



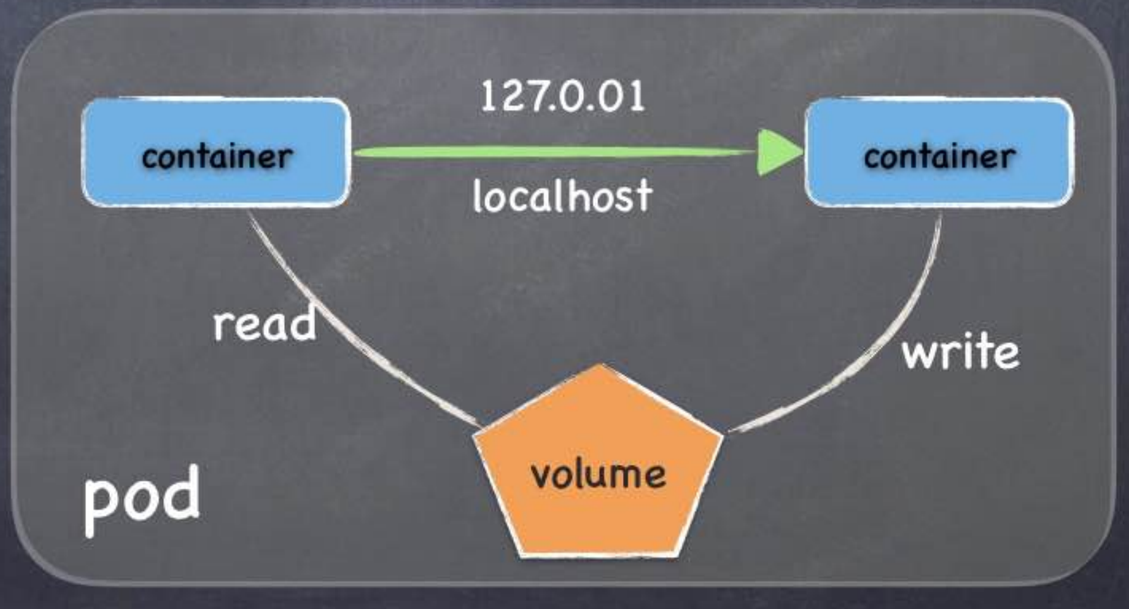
scheduler

- predicates 预选过程
  - 过滤掉不满足条件的节点
  - PodFitsResources
  - PodFitsHostPorts
  - PodSelectorMatches
  - CheckNodeDiskPressure
  - CheckNodeMemoryPressure
- priorities 优选过程
  - 对节点按照优先级排序
  - LeastRequestedPriority
  - SelectorSpreadPriority
  - ImageLocalityPriority
  - NodeAffinityPriority
- algorithmprovider
  - 选择优先级最高的节点



pod

- 一个pod可以有多个容器
- pod之间容器共享网络namespace (127.0.0.1)
- pod之间容器通过Volume来共享目录 (emptyDir and hostPath)



Service Detail

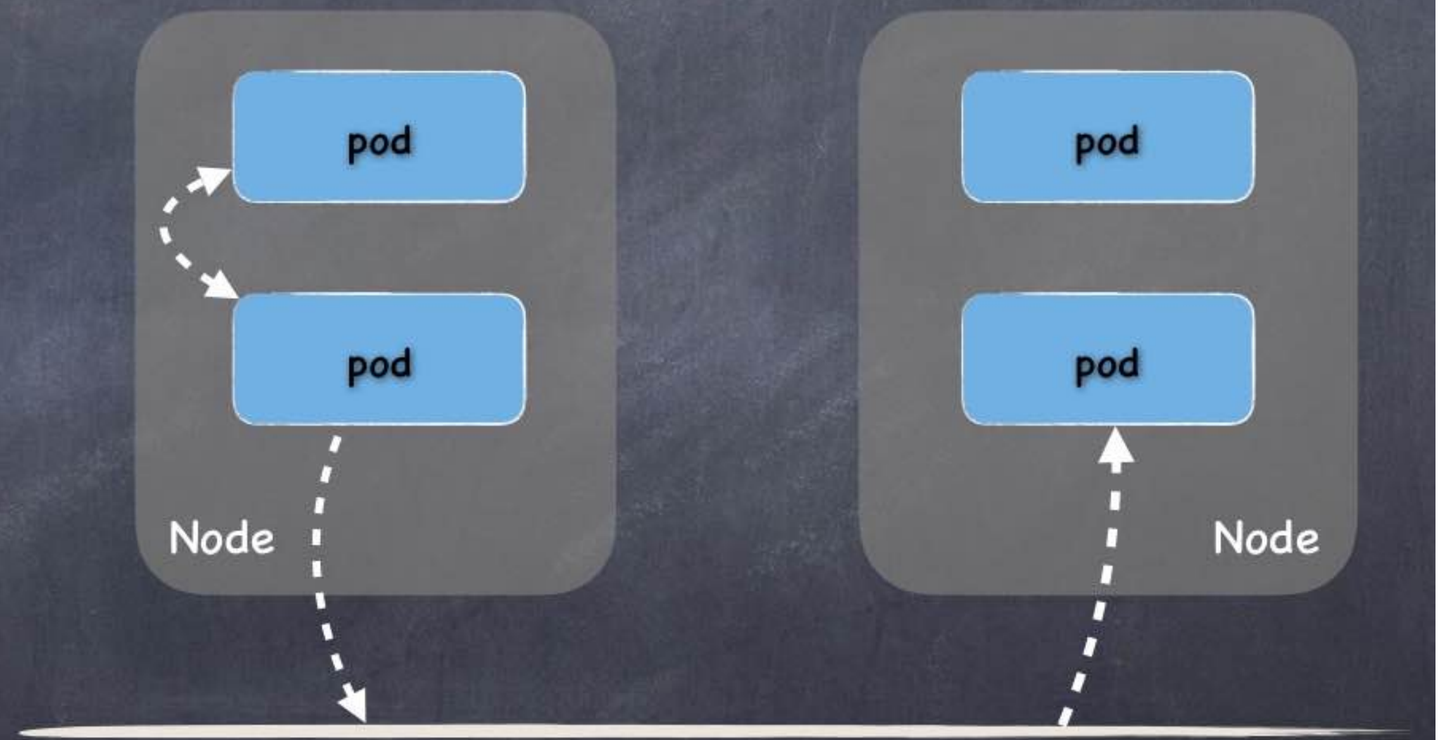
- type
  - clusterIP
  - nodePort
  - HeadLess
    - clusterIP: None
  - lb
  - ...

- iptables做转发
  - 匹配延迟
    - 线性匹配
  - 更新延迟
    - 不能增量

- ipvs做转发
  - 算法更灵活
    - 最小负载
    - 最少连接
    - session
  - hash 匹配
  - 可控的更新延迟

kubernetes network

- Pod
  - 宿主机到pod可以通
  - 宿主机的pod之间可以互通
    - docker/cni0 网桥
  - 不同node的pod也可以互通
    - cni 接口



kubernetes cni

flannel

pod

container

veth0 10.1.11.x/24

cni0

flannel.1-tun

flanneld-8285

Node1

10.244.0.11 eth0

route table

etcd cluster

udp socket

pod

container

veth0 10.1.12.x/24

cni0

flanneld-8285

flannel.1-tun

Node2

10.244.0.12 eth0

Packet	mac	outer-ip	udp	inner-ip	payload

服务发现

- 环境变量 env
  - Get ClusterIP, Port
- 使用service name
  - 经过coredns解析拿到clusterIP

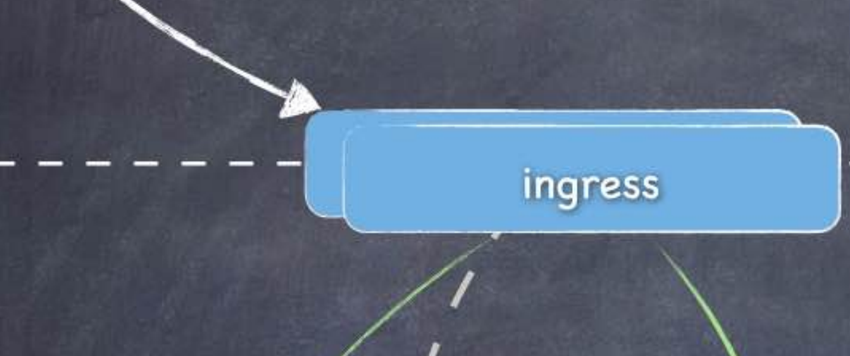
```
BACKEND_SERVICE_PORT_HTTP=80
HTTPBIN_PORT_8000_TCP=tcp://10.100.63.123:8000
BGATEWAY_PORT_9009_TCP_ADDR=10.100.60.183
RATINGS_PORT_9080_TCP_ADDR=10.109.134.221
KUBERNETES_PORT=tcp://10.96.0.1:443
PRODUCTPAGE_PORT_9080_TCP=tcp://10.101.32.36:9080
KUBERNETES_SERVICE_PORT=443
NGINX_SRV_PORT_80_TCP=tcp://10.99.95.82:80
HTTPBIN_SERVICE_PORT=8000
BGATEWAY_PORT_9009_TCP_PORT=9009
HTTPBIN_PORT=tcp://10.100.63.123:8000
RATINGS_PORT_9080_TCP_PORT=9080
HOSTNAME=backend-v1-97ddfb4db-tlnqs
DETAILS_PORT_9080_TCP=tcp://10.101.184.231:9080
ASSET_SERVICE_HOST=10.109.122.107
RATINGS_PORT_9080_TCP_PROTO=tcp
BGATEWAY_PORT_9009_TCP_PROTO=tcp
BGATEWAY_PORT=tcp://10.100.60.183:9009
BGATEWAY_SERVICE_PORT=9009
ASSET_PORT_8090_TCP_ADDR=10.109.122.107
REVIEWS_SERVICE_PORT_HTTP=9080
SLEEP_SERVICE_PORT_HTTP=80
```

对于业务来说, 使用 Service Name 就可以了

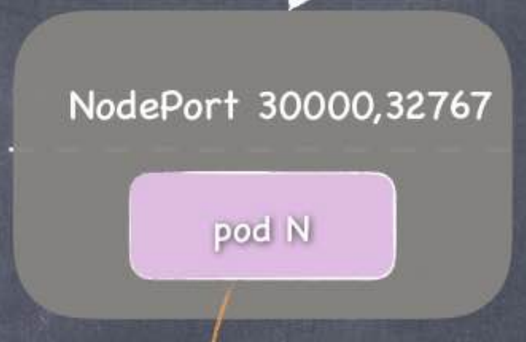
kubernetes 外部访问

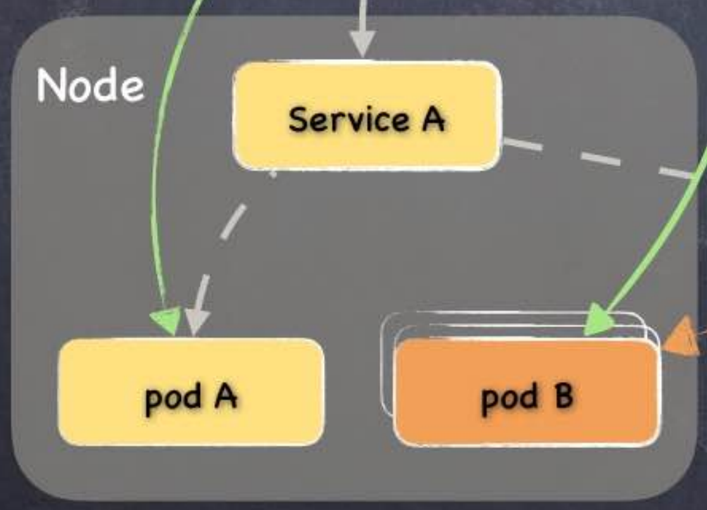
- hostNetwork = true
- hostPort
- Ingress (nginx, haproxy, traefix, envoy)
- NodePort (iptables nat)
- 公有云Load Balancer (aws, azure, gce ...)

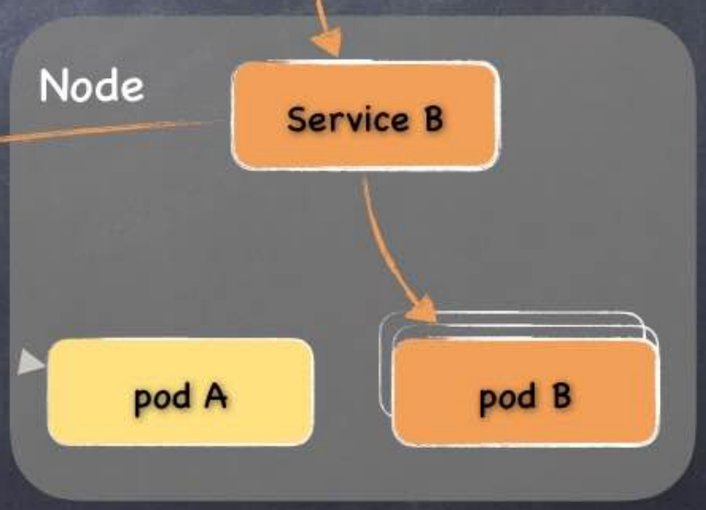
kubernetes 外部访问



inside kubernets cluster







ingress design

- skip kube-proxy
  - direct upstream endpoint
- hostPort
  - bind node port
- daemonSet
  - one pod each node

```
for {
        rateLimiter.Accept()
        ingresses, err := ingClient.List(api.ListOptions{})
        if err != nil {
            continue
        }
        if reflect.DeepEqual(ingresses.Items, known.Items) {
            continue
        }
        known = ingresses
        os.Create("/etc/nginx/nginx.conf")
        tmpl.Execute(w, ingresses)
        shellOut("nginx -s reload")
}
```

Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-v2
  labels:
    app: backend
    version: v2

spec:
  replicas: 3
  selector:
    matchLabels:
      app: backend
      version: v2
  template:
    metadata:
      labels:
        app: backend
        version: v2
    spec:
      containers:
      - name: backend
        image: xiaorui/backend
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 3000
```

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: backend
  labels:
    app: backend

spec:
  selector:
    app: backend

  ports:
  - name: http
    port: 80
    targetPort: 3000
```

Ingress

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: traefik-ingress
  namespace: default
spec:
  rules:
  - host: 163.com
    http:
      paths:
      - path: /
        backend:
          serviceName: backend
          servicePort: 80
```

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx-ingress
spec:
  rules:
  - host: xiaorui.cc
    http:
      paths:
      - backend:
          serviceName: backend
          servicePort: 80
```

快速扩容

```
→ kubectl get pods|grep backend-v2
backend-v2-66578dbbdb-j22gg       2/2     Running   0          4d11h
backend-v2-66578dbbdb-j5sdt       2/2     Running   0          4d11h
backend-v2-66578dbbdb-ks64n       2/2     Running   0          4d11h

→ kubectl scale deployment backend-v2 --replicas 10
deployment.extensions/backend-v2 scaled

→ kubectl get pods|grep backend-v2
backend-v2-66578dbbdb-2cnzf       2/2     Running   0          9s
backend-v2-66578dbbdb-557vt       2/2     Running   0          9s
backend-v2-66578dbbdb-5dpxk       2/2     Running   0          9s
backend-v2-66578dbbdb-bksmp       2/2     Running   0          10s
backend-v2-66578dbbdb-cc727       2/2     Running   0          10s
backend-v2-66578dbbdb-j22gg       2/2     Running   0          4d11h
backend-v2-66578dbbdb-j5sdt       2/2     Running   0          4d11h
backend-v2-66578dbbdb-ks64n       2/2     Running   0          4d11h
backend-v2-66578dbbdb-ks8cm       2/2     Running   0          9s
backend-v2-66578dbbdb-xkvzv       2/2     Running   0          10s
```

升级回滚

```
# rolling update
kubectl set image deployment/backend backend=xiaorui/backend:v2

# roll back
kubectl rollout undo deployment/backend
```

- maxUnavailable:
  - 更新过程中不可用的pod数量
  - default: 25%
- maxSurge:
  - 更新中pod总数的最大值
  - default: 25%

也可使用servcie selector version规避

升级回滚

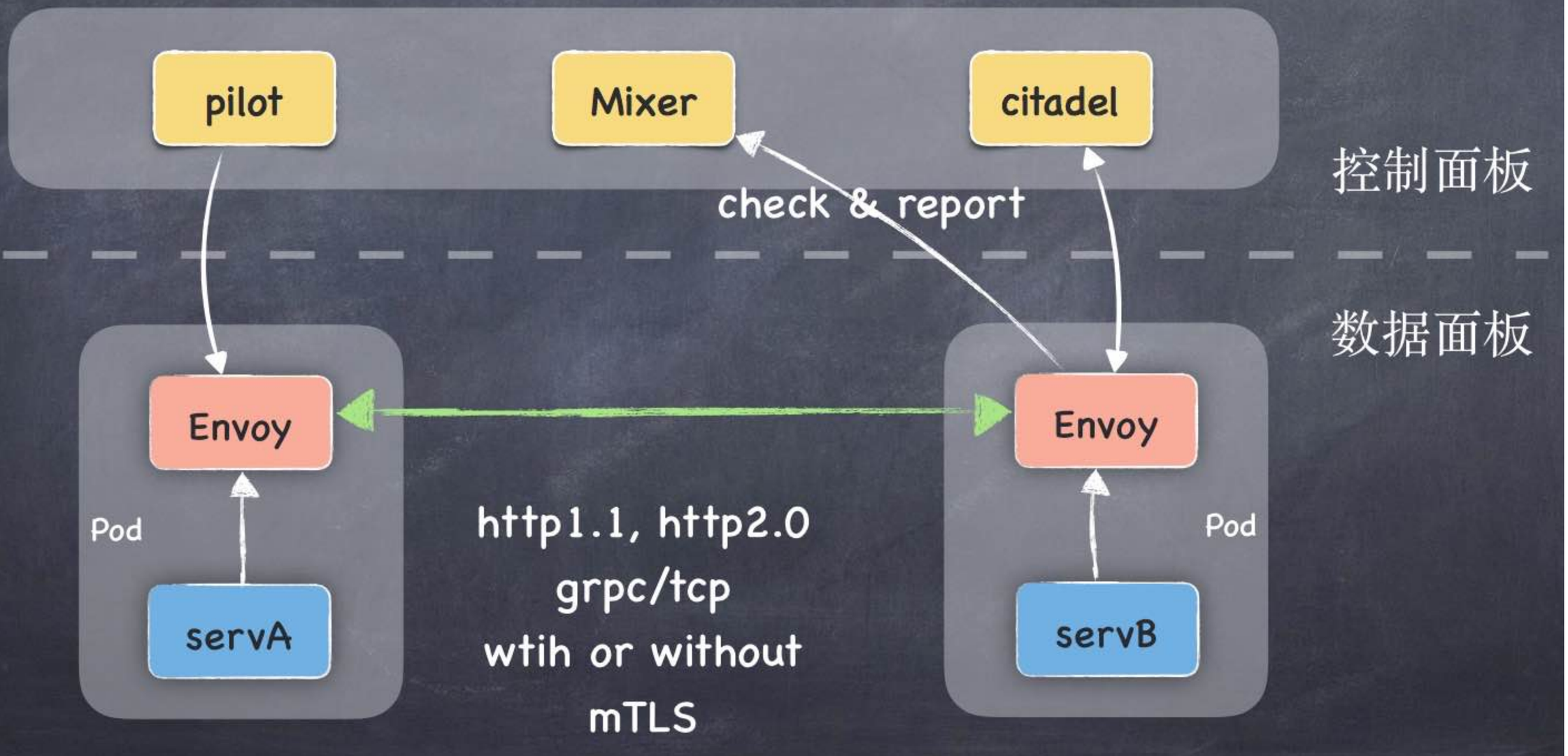

Deployment
Deployment
Deployment
Deployment
Rs (old)
Rs (old)
Rs (new)
Rs (old)
Rs (new)
Rs (new)
app-v1
app-v1
app-v2
app-v1
app-v2
app-v2

为什么还需要istio？

- k8s
  - 服务发现
  - 4层负载均衡
  - 滚动升级
  - ...
- istio
  - 规避长连接的“坑”
    - gpc client、net/http
  - 更细致的7层负载均衡
  - 更细致的流量管理
  - 更细致的灰度发布
  - ...

istio

Istio


pilot
Mixer
citadel
控制面板
check & report
数据面板
Envoy
Envoy
Pod
Pod
servA
servB
http1.1, http2.0
grpc/tcp
wtih or without
mTLS

istio 组件

Istio

- Pilot-x 服务发现
- Mixer
  - istio-policy 检查权限, 配额
  - istio-telemetry 收集调用metrics
- citadel 证书
- galley 校验正确性
- ingressgateway 网关

- jaeger
- prometheus
- zipkin
- fluented
- ...

istio pilot

kubectl/istioctl

kube-apiserver

Etcd-Cluster

list/watch

Pilot

XDS Protocol

XDS

Pod

Envoy

Service

Pod

Envoy

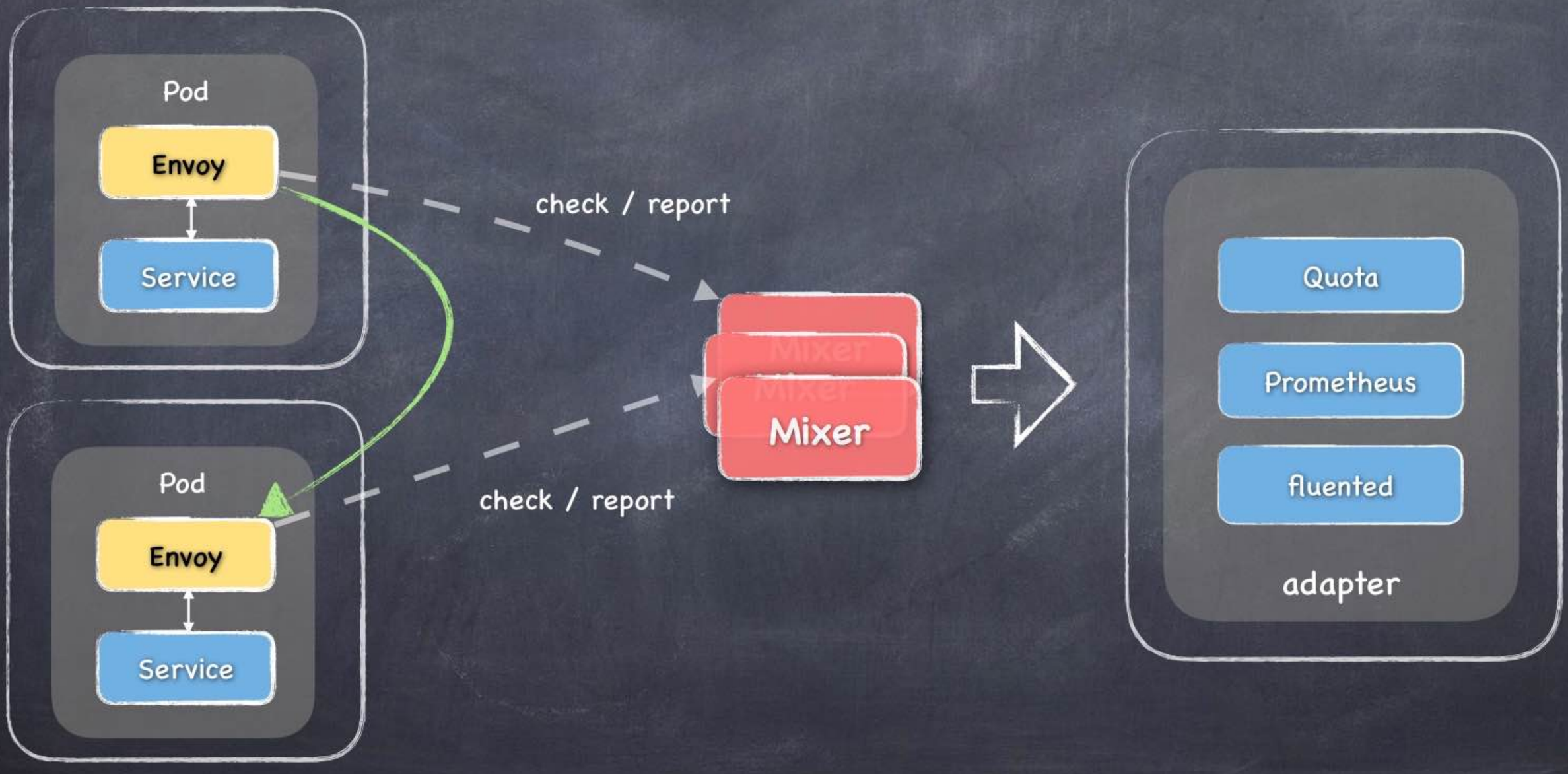
Service

Pod

Envoy

Service

istio mixer adapter

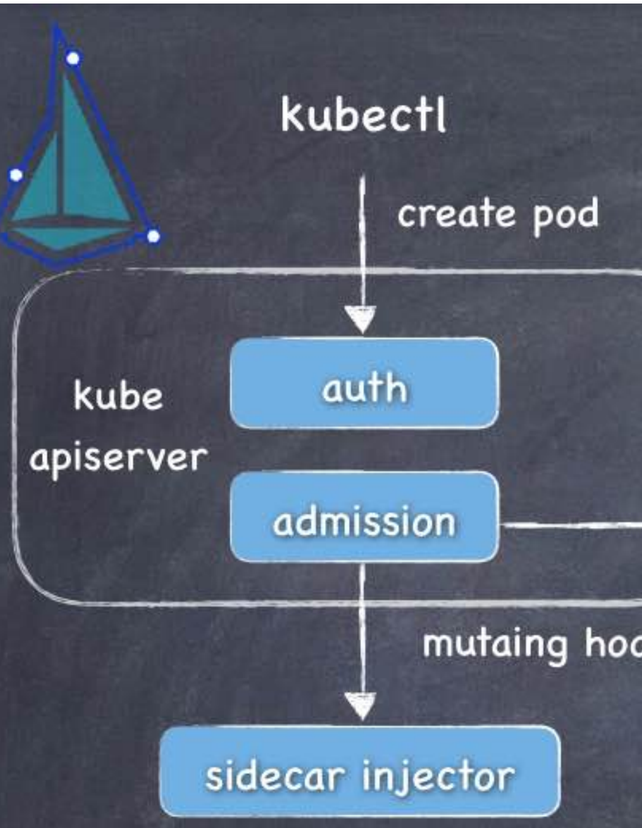


istio 流量治理流程

- 控制面板流程：
  - 管理员通过kubectl/istioctl或者API创建流量规则
  - Pilot从kubernets apiserver获取数据，并规则转换为Envoy xds
  - Pilot将xds推送给envoy
- 数据面板流程：
  - Envoy动态载入xds配置，并初始化新的资源监听
  - Envoy 拦截 pod上的本地容器的Inbound和Outbound流量
  - 在流量经过Envoy时执行对应的流量规则，执行流量治理

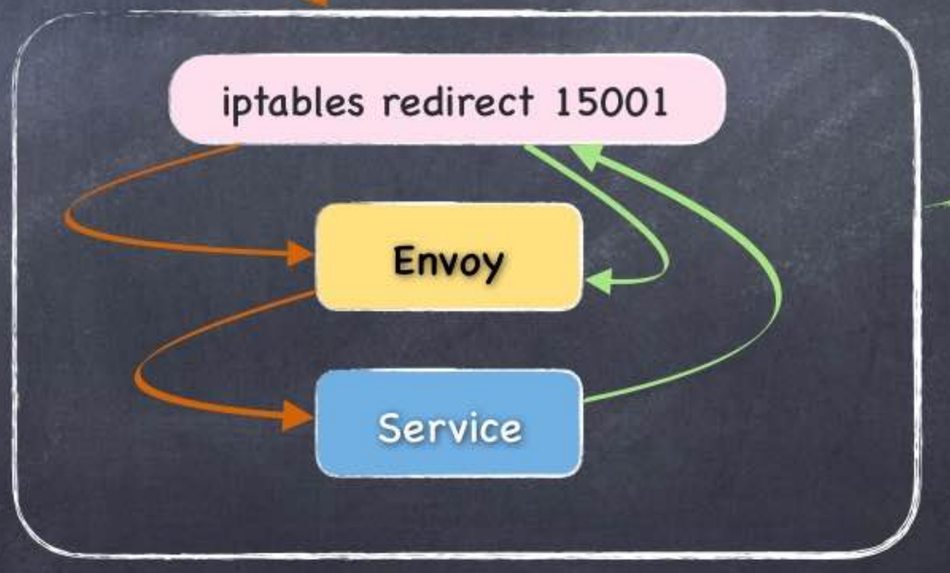
istio inject

Istio



input

- pilot-proxy
  - pilot-agent
    - 生成envoy初始化配合
    - 管理envoy的生命周期
  - envoy sidecar



ouput

- pilot-init
  - ./prepare_proxy.sh -p 15001 -u 1337
  - init exit !

only tcp !!!

istio crd resource

Istio

- VirtualService
  - 定义路由规则
  - 流量管理
- DestinationRule
  - 定义可路由的目的服务的子集
  - 目的服务的策略 (断路器/负载均衡/TLS ...)
- ServiceEntry
  - 定义网格之外的资源
- Gateway
  - 为网格配置网关
- Sidecar
  - 服务隔离
- EnvoyFilter
  - 集成Lua可自定义envoy的过滤链规则

一个实例

Istio

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: backend
spec:
  hosts:
  - backend
  http:
  - route:
    - destination:
        host: backend
        subset: v1
      weight: 50
    - destination:
        host: backend
        subset: v2
      weight: 50
```

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: backend
spec:
  host: backend
  subsets:
  - name: v1
    labels:
      version: v1
    trafficPolicy:
      loadBalancer:
        simple: ROUND_ROBIN

  - name: v2
    labels:
      version: v2
    trafficPolicy:
      loadBalancer:
        simple: LEAST_CONN
```

rules

Istio

SVC-A

条件:
uri, scheme, method, cookie,
headers, post, sourceLables,
gateways

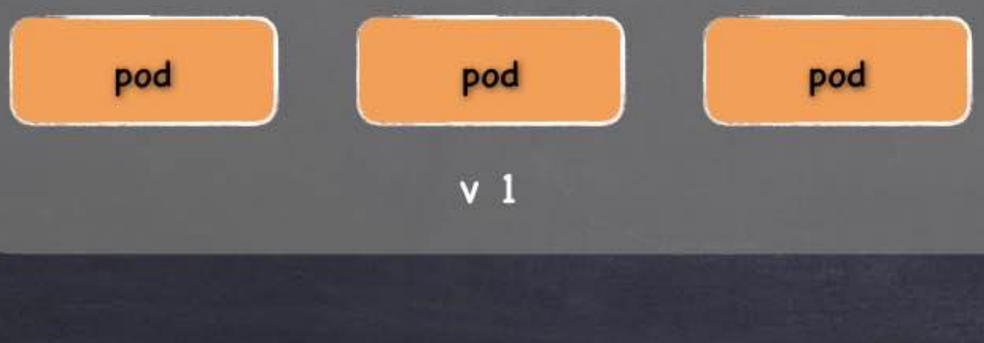
- 重定向
- 重写
- 重试
- 故障注入
- 流量镜像
- more

规则匹配

Istio

path: /v1/login
header[user]: rui

default



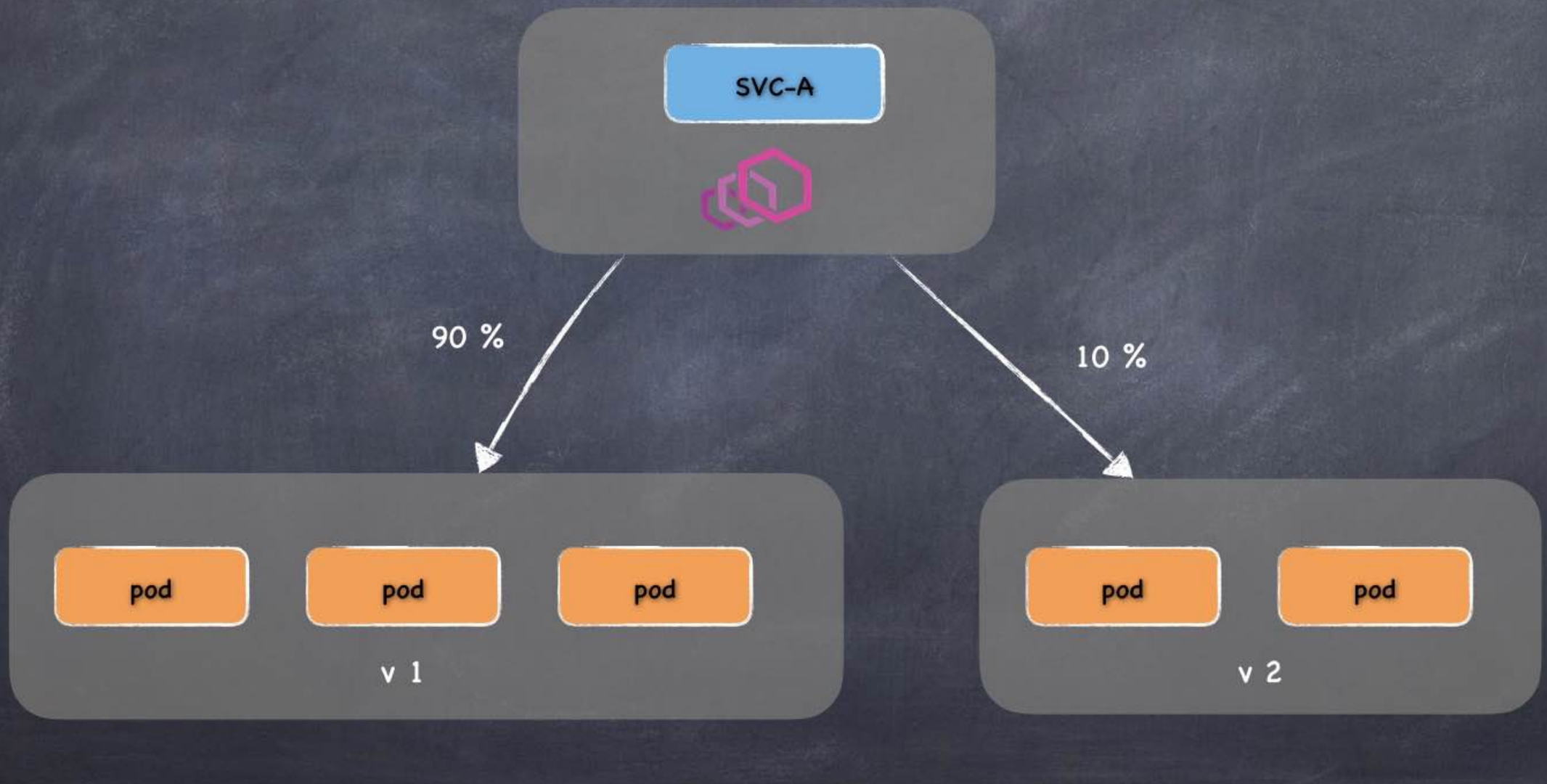
规则匹配

Istio

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: backend
spec:
  hosts:
  - backend
  http:
  - match:
    - uri:
        prefix: /v1/
    - uri:
        regex: ^.*?info\?v1.*$
    - headers:
        user:
          exact: rui
    route:
    - destination:
        host: productpage
        subset: v1

  - route:
    - destination:
        host: backend
        subset: v2
```

权重

Istio



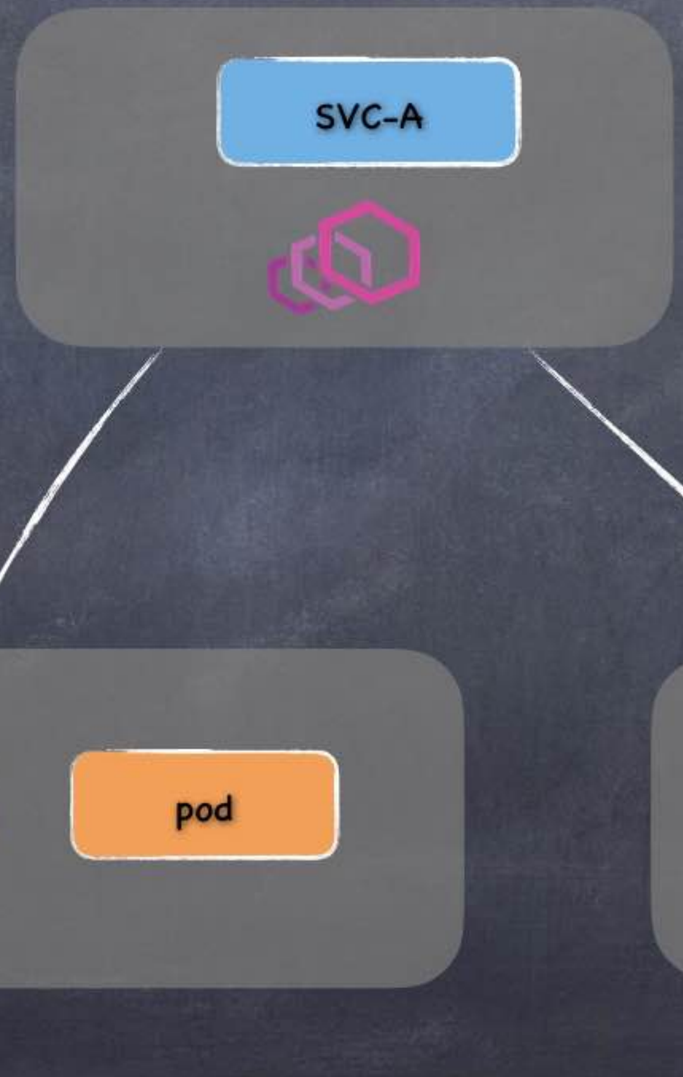
权重

Istio

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: backend
spec:
  hosts:
  - backend
  http:
  - route:
    - destination:
        host: backend
        subset: v1
      weight: 90
    - destination:
        host: backend
        subset: v2
      weight: 10
```

镜像

Istio



as goreplay, tcpcopy

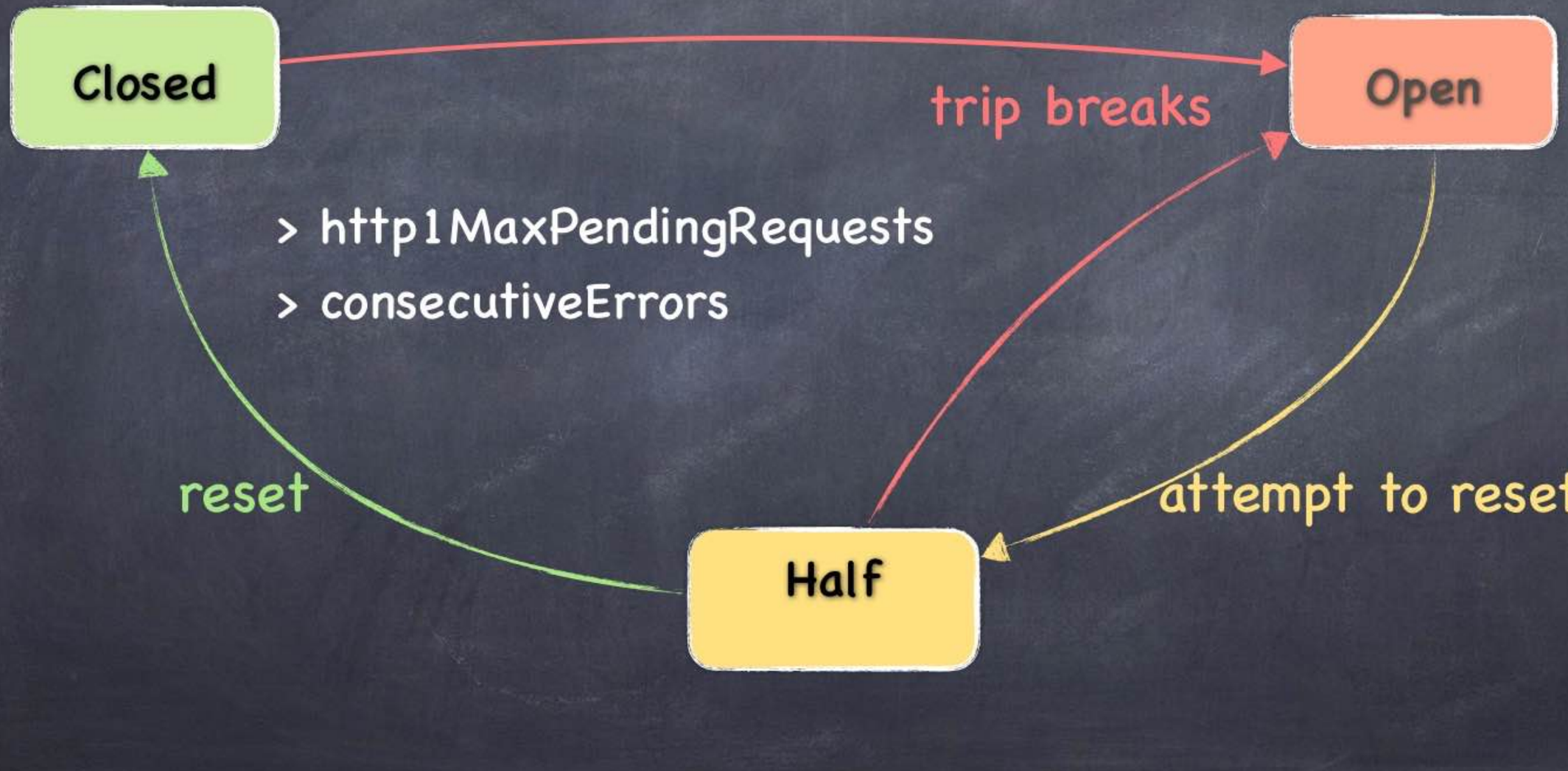
镜像

Istio

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: backend
spec:
  hosts:
  - backend
  http:
  - route:
      - destination:
          host: backend
          subset: v1
        weight: 100
    mirror:
        host: backend
        subset: test
```

熔断

Istio


Closed
trip breaks
Open
> http1MaxPendingRequests
> consecutiveErrors
reset
attempt to reset
Half

熔断

Istio

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: backend-circuit-breaker
spec:
  host: backend
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 50  # contain http
        connectTimeout: 5s
      http:
        http1MaxPendingRequests: 50
        http2MaxRequests: 1000
        maxRequestsPerConnection: 20
    outlierDetection:
      consecutiveErrors: 50
      interval: 5s
      baseEjectionTime: 30s
      maxEjectionPercent: 50
```

other

Istio

- ratelimit
- timeout
- session affinity
- ...

- retry
- rewrite
- redirect
- ...

- fault inject
- delay
- abort
- denier
- attribute
- iplist

Istio

例子

https://github.com/rfyiamcool/istio-http-lb

- k8s
- istio
- 涵盖大多数功能测试

before


biss envoy
biss.proto
biss gateway
busproto.SrvReqMesg
busproto.SrvRespMesg
nats mq
busproto.SrvReqMesg
busproto.SrvRespMesg
s1
s2
s3

after

Istio

biss envoy

biss.proto

Envoy

biss gateway

busproto.SrvReqMesg

busproto.SrvRespMesg

Envoy

s1

Envoy

s1

Envoy

s1

Istio

平滑迁移

业务只需要更换srvFrame库包即可 !!!

- 自定义grpc编码
- grpc.UnknownServiceHandler
- grpc invoke raw
- grpc reflect & protobuf reflect

Istio

control script

- 通过env和template来生成k8s/istio配置
- 通过namespace来隔离每个developer的环境
- 可配置服务组启动，跳过注入及启动顺序等
- 通过control来管理mesh各资源的生命周期
  - gen; start; stop; restart; logs; pods; ...

https://github.com/rfyiamcool/k8s-istio-control

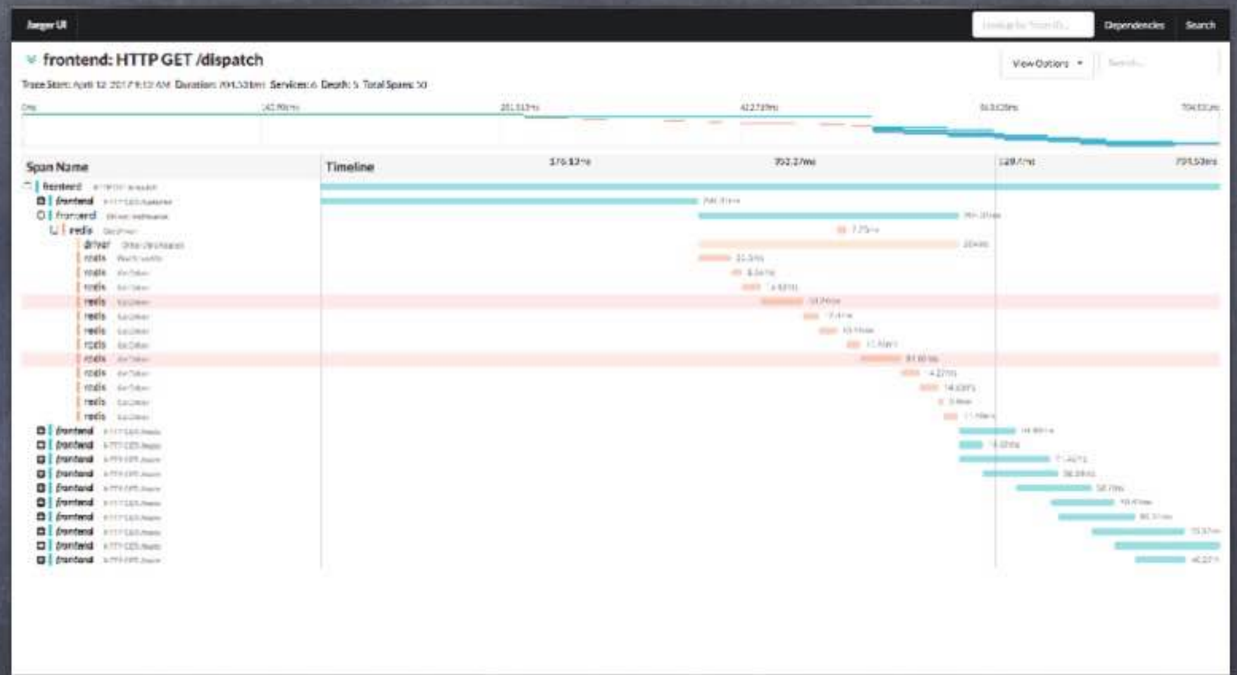
```yaml
# 生成的配置的路径
output_path: ./output

# 映射到模板里的变量
vars:
  run_env: TEST # PROD, DEV
  benvoy_hostnet: false
  namespace: ruifengyun
  nfs_server: 10.10.10.10
  ...

# 跳过istio注入的服务
skip_inject_service:
  - postgres
  - command-bus
  ...

# 服务列表
service:
  - benvoy
  - website
  ...

# 强制依赖
must_deps:
  - postgres

# 高优先级依赖，强制启动顺序
high_priority_deps:
  - postgres
  ...

mid_priority_deps:

# 服务组
service_group:
  middleware:
    - postgres
    - scylla
    ...
  trade:
    - trade-bus
    - trade-server
    - me
    ...

# 启动的服务
enable:
  service:
    - ...
  service_group:
    - ...
```

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: benvoy
  namespace: {{ .namespace }}

spec:
    ...
    spec:
      {{ if eq .benvoy_hostnet "true" }}
      hostNetwork: true
      {{ end }}
      dnsPolicy: ClusterFirstWithHostNet
      containers:
      - name: benvoy
        image: {{ .benvoy_image }}
        imagePullPolicy: {{ .pull_mode }}
        livenessProbe:
          tcpSocket:
            port: 80
        ...

        ports:
        - containerPort: 80    # HTTP_PORT

        ...

      volumes:
      - name: benvoy
        nfs:
          path: /biss-dep/dep
          server: {{ .nfs_server }}
```

other

Istio

- 分布式链路追踪
  - opentracing
    - jaeger
    - zipkin
- design
  - trace ID
  - span ID



- 分布式日志追踪
  - 使用trace id追溯上下文

" Q&A "

- xiaorui.cc