KUBESPHERE x CLOUD NATIVE COMPUTING FOUNDATION

Kubernetes中的流量管理和服务发现

张晋涛

API7.ai云原生技术专家

Apache APISIX PMC

个人介绍

张晋涛

- Apache APISIX PMC
- Kubernetes ingress-nginx reviewer
- 微软MVP
- 『K8S生态周报』发起人和维护者
- zhangjintao@apache.org

KUBESPHERE

Agenda

- Kubernetes中的流量管理和服务发现
- 使用KubeSphere部署Apache APISIX Ingress controller
- 使用Apache APISIX Ingress controller作为K8s集群的南北流量网关
- 使用OpenELB搭配使用

KUBESPHERE

Kubernetes中的流量管理和服务发现

Kubernetes中Pod间通信

- 创建两个Deployment
- 获取Pod的IP地址
- 尝试进行Pod间的ping操作

```
➜  ~ kubectl create deploy redis-0 --image="ghcr.io/moelove/redis:alpine"
deployment.apps/redis-0 created
➜  ~ kubectl create deploy redis-1 --image="ghcr.io/moelove/redis:alpine"
deployment.apps/redis-1 created
```

```
➜  ~ kubectl get pod -l app=redis-0 --output jsonpath='{.items[0].status.podIP}'
10.244.1.9
➜  ~ kubectl get pod -l app=redis-1 --output jsonpath='{.items[0].status.podIP}'
10.244.1.10
```

```
➜  ~ kubectl exec deploy/redis-0 -- ping -c 1 `kubectl get pod -l app=redis-1 \
--output jsonpath='{.items[0].status.podIP}'`
PING 10.244.1.10 (10.244.1.10): 56 data bytes
64 bytes from 10.244.1.10: seq=0 ttl=63 time=0.087 ms

--- 10.244.1.10 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.087/0.087/0.087 ms
```

Kubernetes中Pod间通信

删除Pod后重复进行ping操作

```
→  ~ kubectl get pod -l app=redis-1 --output jsonpath='{.items[0].status.podIP}'
10.244.1.10
→  ~ kubectl delete pods -l app=redis-1
pod "redis-1-5987df6b97-hpnt5" deleted
→  ~ kubectl get pod -l app=redis-1 --output jsonpath='{.items[0].status.podIP}'
10.244.1.11
→  ~ kubectl exec deploy/redis-0 -- ping -c 1 `kubectl get pod -l app=redis-1 --output
jsonpath='{.items[0].status.podIP}'`
PING 10.244.1.11 (10.244.1.11): 56 data bytes
64 bytes from 10.244.1.11: seq=0 ttl=63 time=0.072 ms

--- 10.244.1.11 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.072/0.072/0.072 ms
```

Kubernetes中Pod间通信

- Kubernetes集群中每个Pod有自己独立的IP（通常情况）
- Pod间可通过IP直接进行通信
- Pod的IP一般情况下不能提前获取

Kubernetes中的服务发现

基于CoreDNS的服务发现机制

- <svc>.<ns>.svc.cluster-domain.example
- <Pod-IP>.<ns>.pod.cluster-domain.example

```
➜  ~ kubectl get pods -l app=redis-0 -o jsonpath='{.items[0].status.podIP }'
10.244.2.6%

➜  ~ kubectl exec  deploy/redis-0 -- nslookup 10-244-2-6.default.pod.cluster.local
Server:         10.96.0.10
Address:        10.96.0.10:53

Name:   10-244-2-6.default.pod.cluster.local
Address: 10.244.2.6
```

Kubernetes中的Service类型

- ClusterIP
- NodePort
- LoadBalancer
- ExternalName

Pod可通过Service进行通信

```
→  ~ kubectl exec deploy/redis-0 -- sh -c 'nslookup redis-1 | grep "^[^*]"'
Server:         10.96.0.10
Address:        10.96.0.10:53
Name:   redis-1.default.svc.cluster.local
Address: 10.96.22.135
→  ~ kubectl exec deploy/redis-0 -- sh -c 'redis-cli -h redis-1 -p 6379 ping'
PONG
```

Kubernetes集群的南北向流量

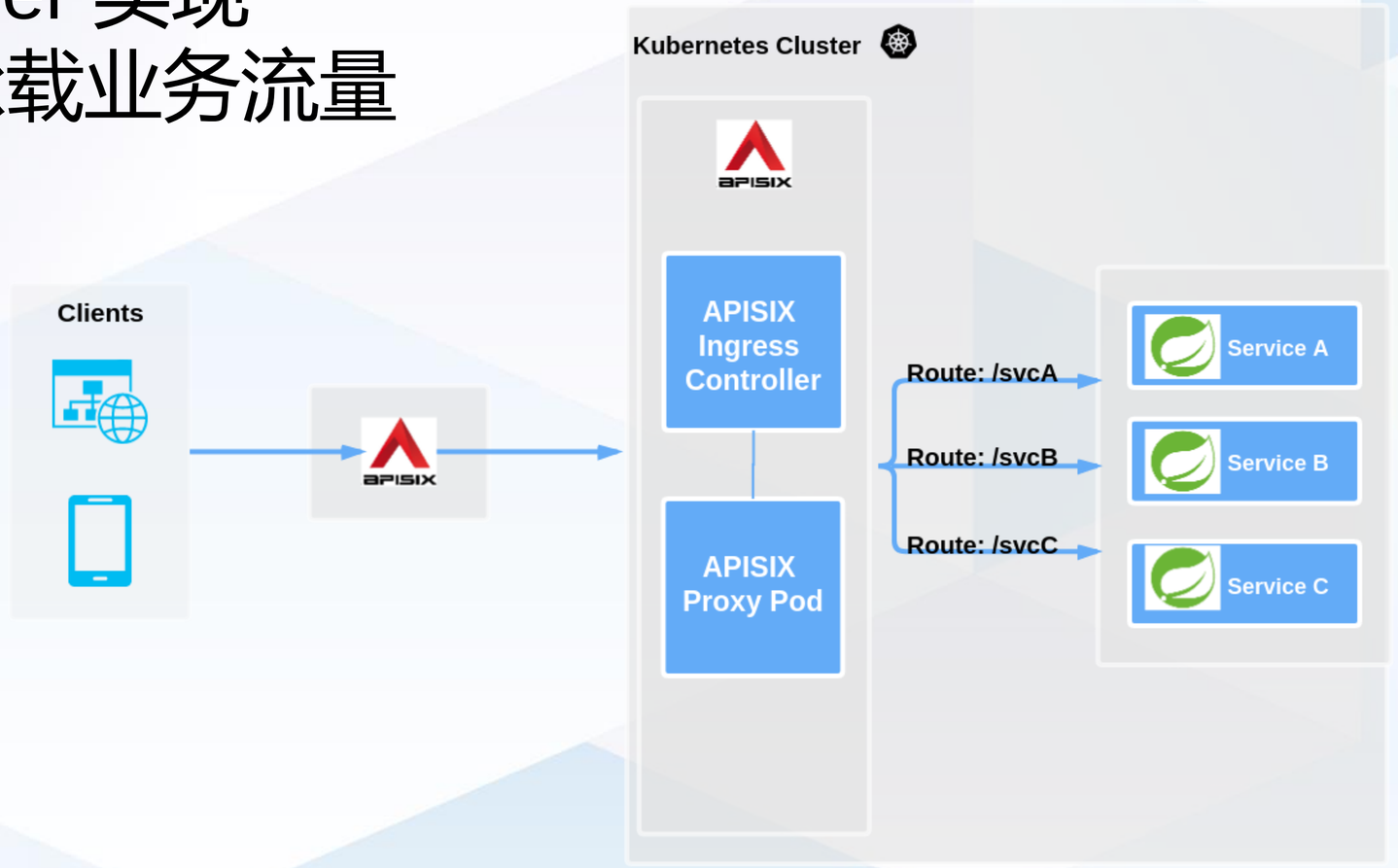
- Service(从集群外访问)
- Ingress

KUBESPHERE

在KubeSphere中部署Apache APISIX Ingress controller

Apache APISIX Ingress controller是什么

- 另一种 Ingress Controller 实现
- 使用 Apache APISIX 承载业务流量
- 通过 CRD 进行实现
- 支持原生 Ingress 资源



https://github.com/apache/apisix-ingress-controller/

Apache APISIX Ingress controller特性

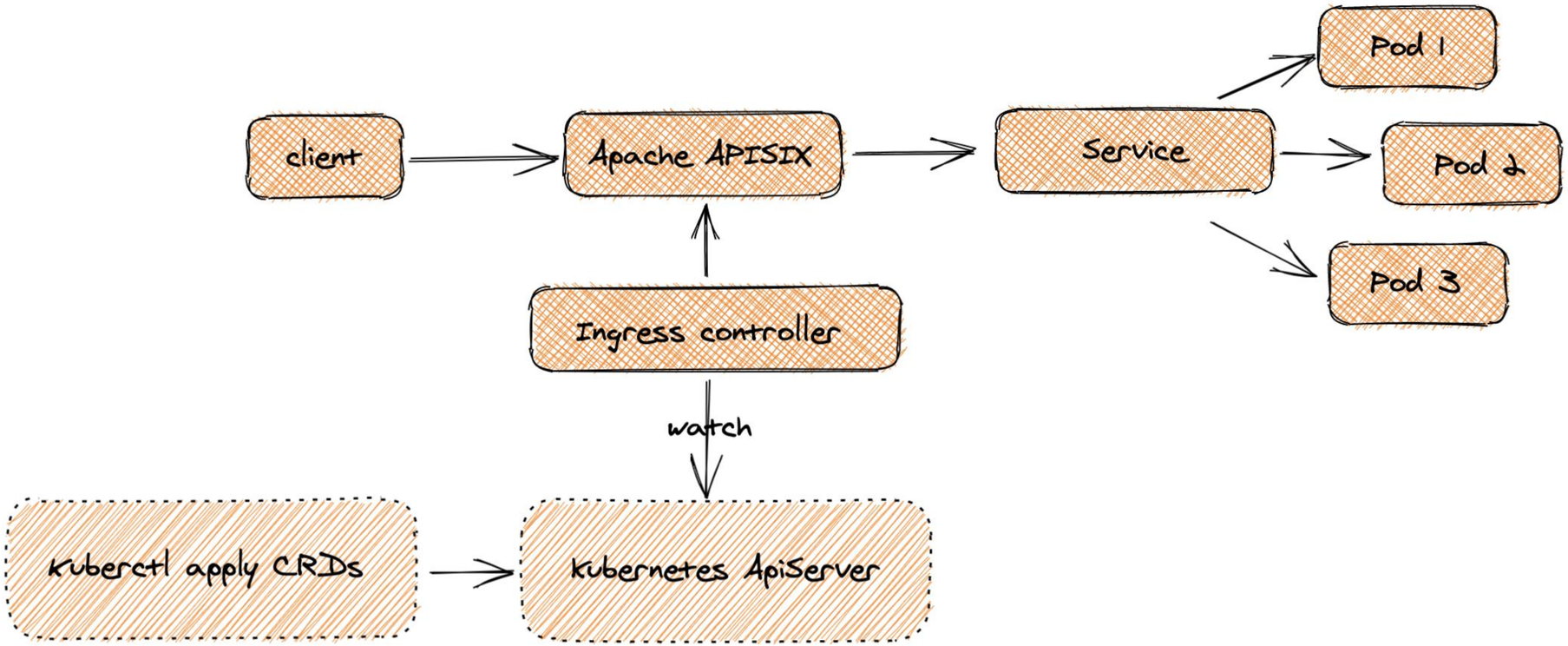
- 全动态：路由、SSL 证书、上游、插件…
- 支持 Custom Resource Definitions，更容易理解的声明式配置
- 支持 K8S 原生 Ingress 配置 （v1 / v1beta1 ）
- 通过 annotation 的方式对 Ingress 能力进行扩展
- 服务自动注册发现，无惧扩缩容
- 更灵活的负载均衡策略
- 健康检查开箱即用

Apache APISIX Ingress controller特性

- 支持高级路由匹配规则
- 支持流量切分
- 支持 Apache APISIX 官方 50 多个插件 & 客户自定义插件
- gRPC plaintext 支持
- TCP 4 层代理
- 状态检查：快速掌握声明配置的同步状态
- ...

Apache APISIX Ingress controller概览

通过CRD进行扩展



Apache APISIX Ingress中的CRD

- ApisixRoute: 路由
- ApisixUpstream： 上游
- ApisixConsumer：消费者
- ApisixTls：证书相关
- ApisixClusterConfig：集群公共配置
- ApisixPluginConfig：插件通用配置

在KubeSphere 中部署Apache APISIX Ingress

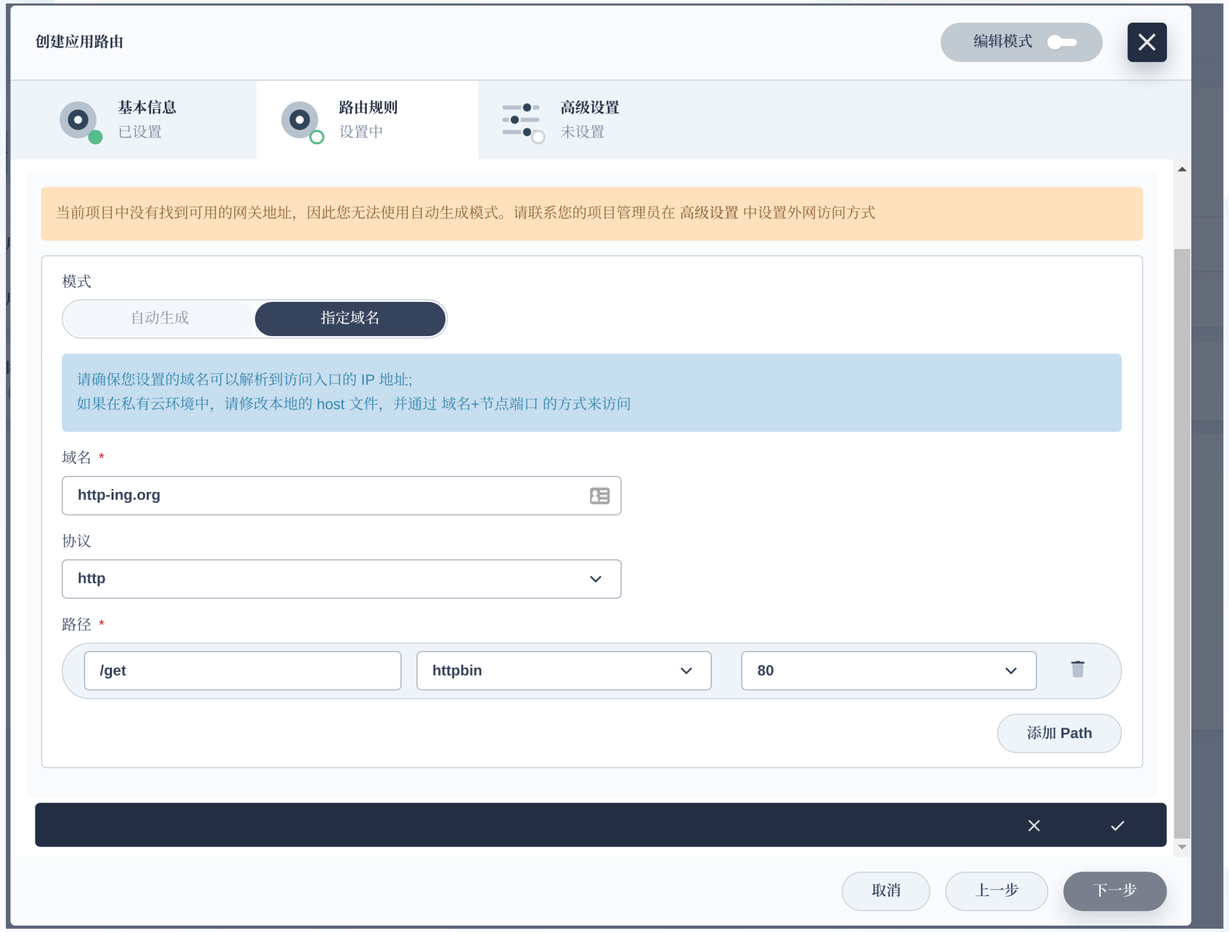
- 添加应用仓库
- 使用应用模板进行创建

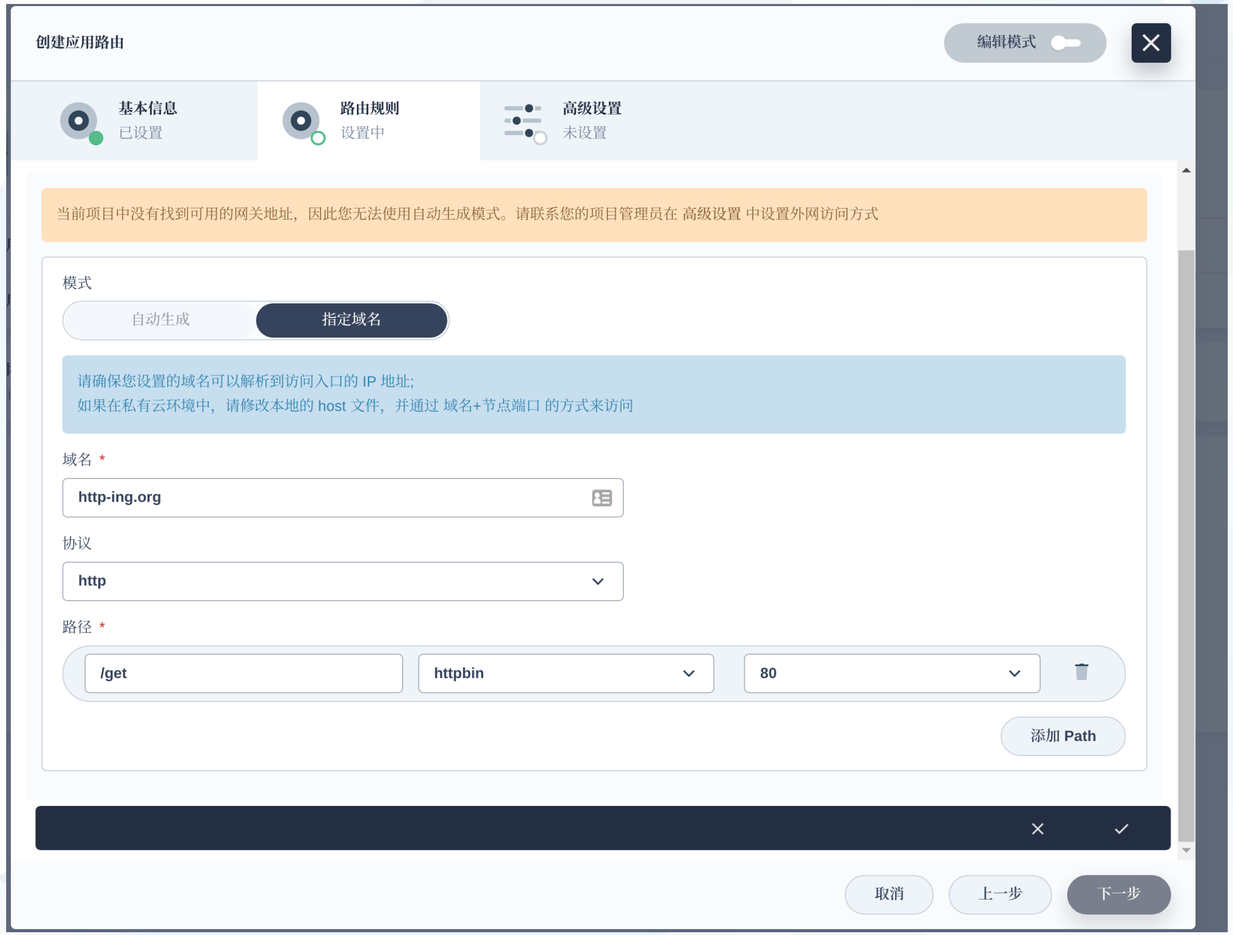
Apache APISIX Ingress controller也可用Helm部署

```
$ helm install apisix apisix/apisix --set gateway.type=NodePort --set ingress-controller.enabled=true --set
ingress-controller.config.apisix.serviceNamespace=apisix --namespace apisix --create-namespace --set ingress-
controller.config.apisix.serviceName=apisix-admin
NAME: apisix
LAST DEPLOYED: Tue Mar 22 08:36:40 2022
NAMESPACE: apisix
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace apisix -o jsonpath="{.spec.ports[0].nodePort}" services apisix-
gateway)
  export NODE_IP=$(kubectl get nodes --namespace apisix -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
```

使用Apache APISIX Ingress代理集群内服务

应用路由实际会创建Ingress资源





KUBESPHERE

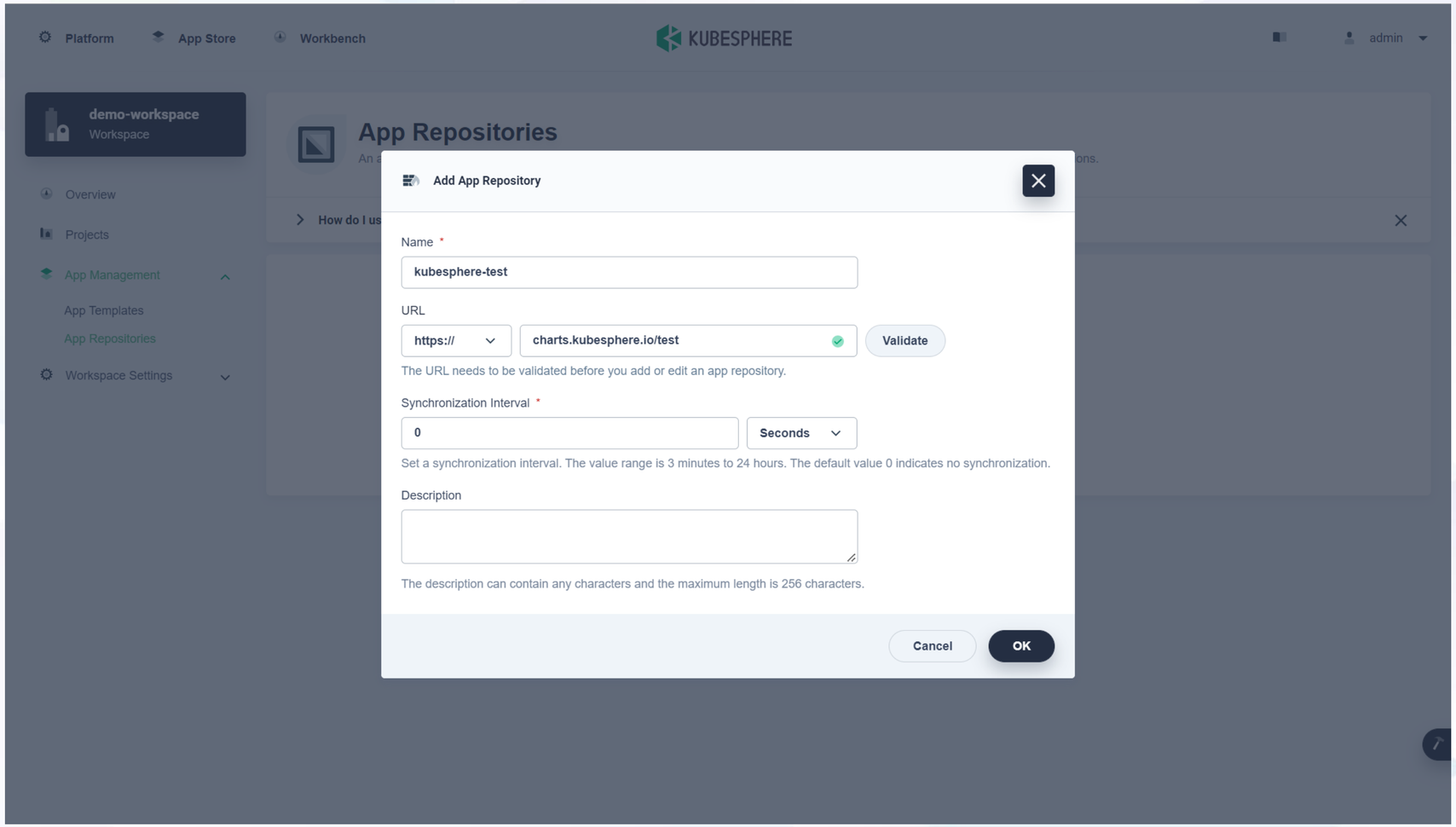
配合OpenELB使用

OpenELB是什么

- 可为多种Kubernetes环境提供LoadBalancer类型的service
- CNCF sandbox项目
- 可使用 CRD 进行 BGP 配置

KubeSphere 中部署OpenELB

- 添加应用仓库
- 使用应用模板进行创建
- 或通过 Helm 完成部署



Apache APISIX Ingress controller与OpenELB结合

更新 APISIX gateway 的 Service类型为LoadBalancer

```
tao@moelove:~$ kubectl -n apisix get svc -l app.kubernetes.io/name=apisix
NAME             TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
apisix-admin     ClusterIP      10.96.209.98    <none>        9180/TCP       14m
apisix-gateway   NodePort       10.96.127.236   <none>        80:32018/TCP   14m

tao@moelove:~$ helm upgrade apisix apisix/apisix --set gateway.type=LoadBalancer --set ingress-
controller.enabled=true --set ingress-controller.config.apisix.serviceNamespace=apisix --namespace
apisix --create-namespace --set ingress-controller.config.apisix.serviceName=apisix-admin

Release "apisix" has been upgraded. Happy Helming!
NAME: apisix
LAST DEPLOYED: Wed Apr  6 09:03:37 2022
NAMESPACE: apisix
STATUS: deployed
REVISION: 2
TEST SUITE: None
NOTES:
1. Get the application URL by running these commands:
     NOTE: It may take a few minutes for the LoadBalancer IP to be available.
           You can watch the status of by running 'kubectl get --namespace apisix svc -w apisix-
gateway'
  export SERVICE_IP=$(kubectl get svc --namespace apisix apisix-gateway --template "{{ range (index
.status.loadBalancer.ingress 0) }}{{.}}{{ end }}")
  echo http://$SERVICE_IP:80

tao@moelove:~$ kubectl -n apisix get svc -l app.kubernetes.io/name=apisix
NAME             TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
apisix-admin     ClusterIP      10.96.209.98    <none>          9180/TCP       14m
apisix-gateway   LoadBalancer   10.96.127.236   172.18.255.200  80:32018/TCP   14m
```

总结

- Kubernetes中基于DNS的服务发现
- Kubernetes中的service和Ingress
- Apache APISIX Ingress
- OpenELB

KUBESPHERE

Follow us!

github.com/kubesphere

twitter.com/kubesphere

UP/KubeSphere

kubesphere.io

kubesphere.slack.com