《 Kubernetes 原理剖析与实战应用》

正范

— 拉勾教育出品 —

11 | K8s Service 轻松搞定服务发现和负载均衡

前言

通过 PV 持久化地保存数据

通过 Deployment 或 Statefulset 这类工作负载来管理多实例

从而保证服务的高可用

为什么需要服务发现?

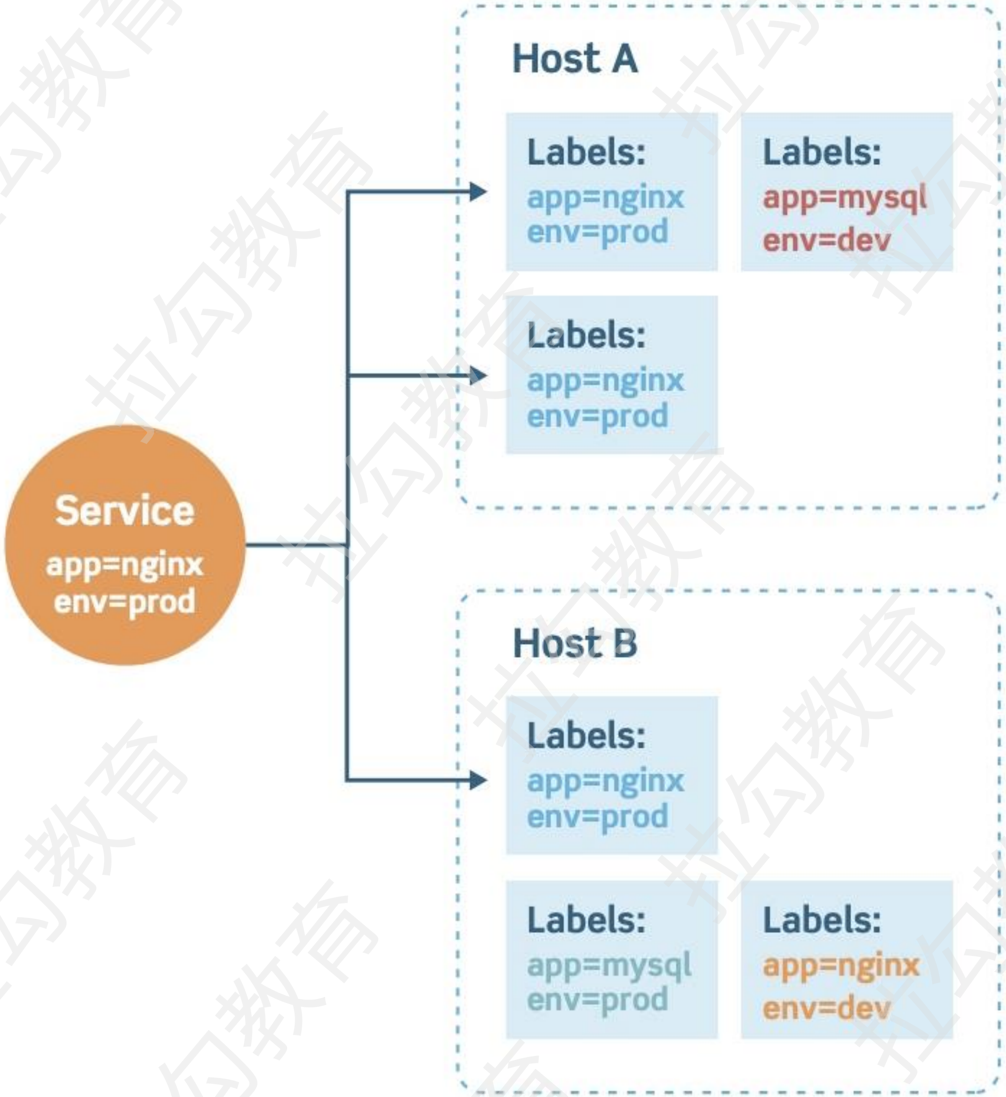
传统的应用部署，服务实例的网络位置是固定的

在 Kubernetes 中

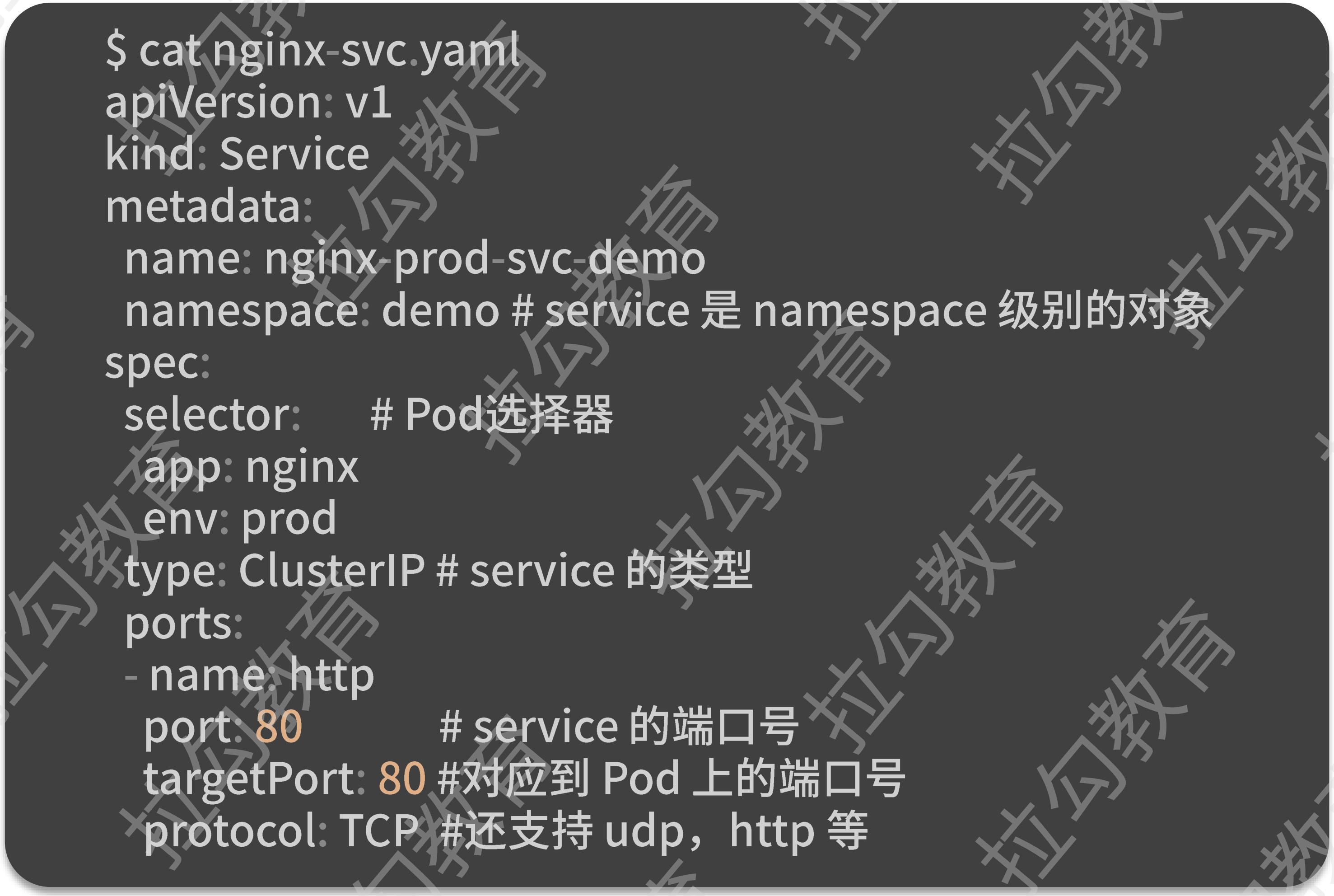
业务都是通过 Pod 来承载的

每个 Pod 的生命周期又很短暂，**用后即焚**，IP 地址也都是随机分配，**动态变化**的

Kubernetes 中的 Service

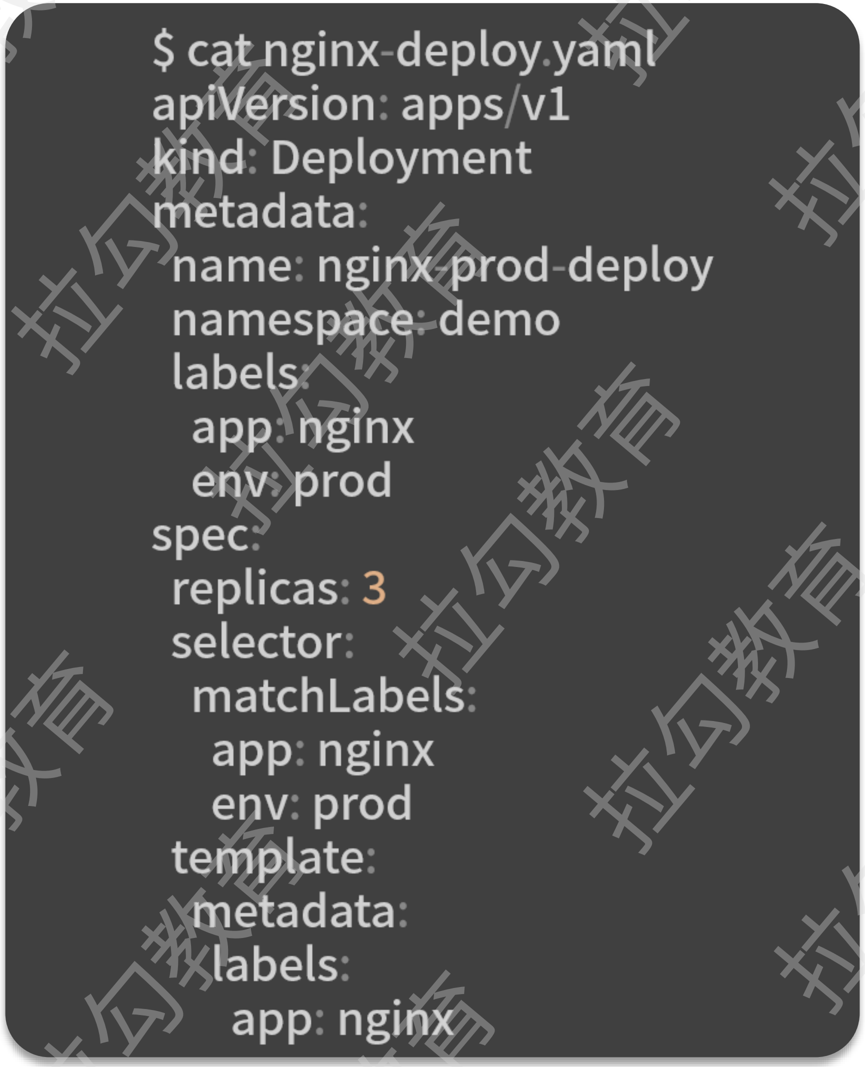


Kubernetes 中的 Service



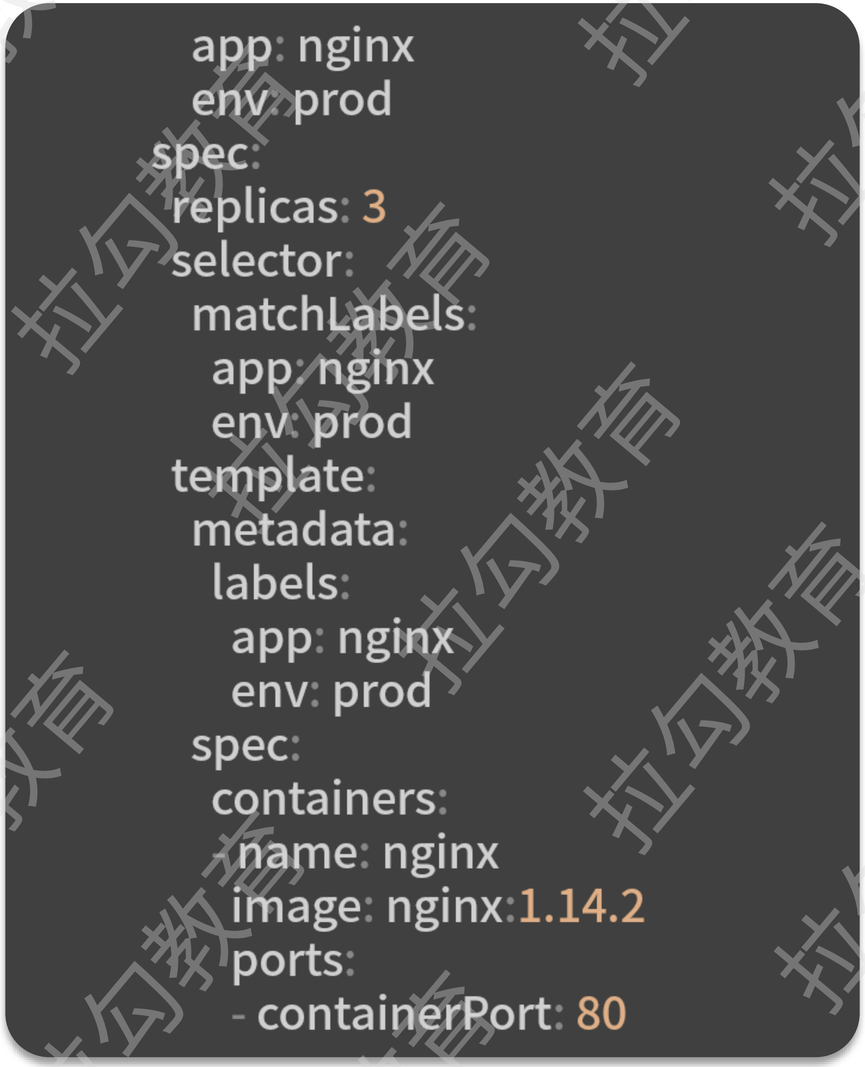
```
$ cat nginx-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-prod-svc-demo
  namespace: demo # service 是 namespace 级别的对象
spec:
  selector:      # Pod选择器
    app: nginx
    env: prod
  type: ClusterIP # service 的类型
  ports:
  - name: http
    port: 80          # service 的端口号
    targetPort: 80 #对应到 Pod 上的端口号
    protocol: TCP  #还支持 udp，http 等
```

Kubernetes 中的 Service



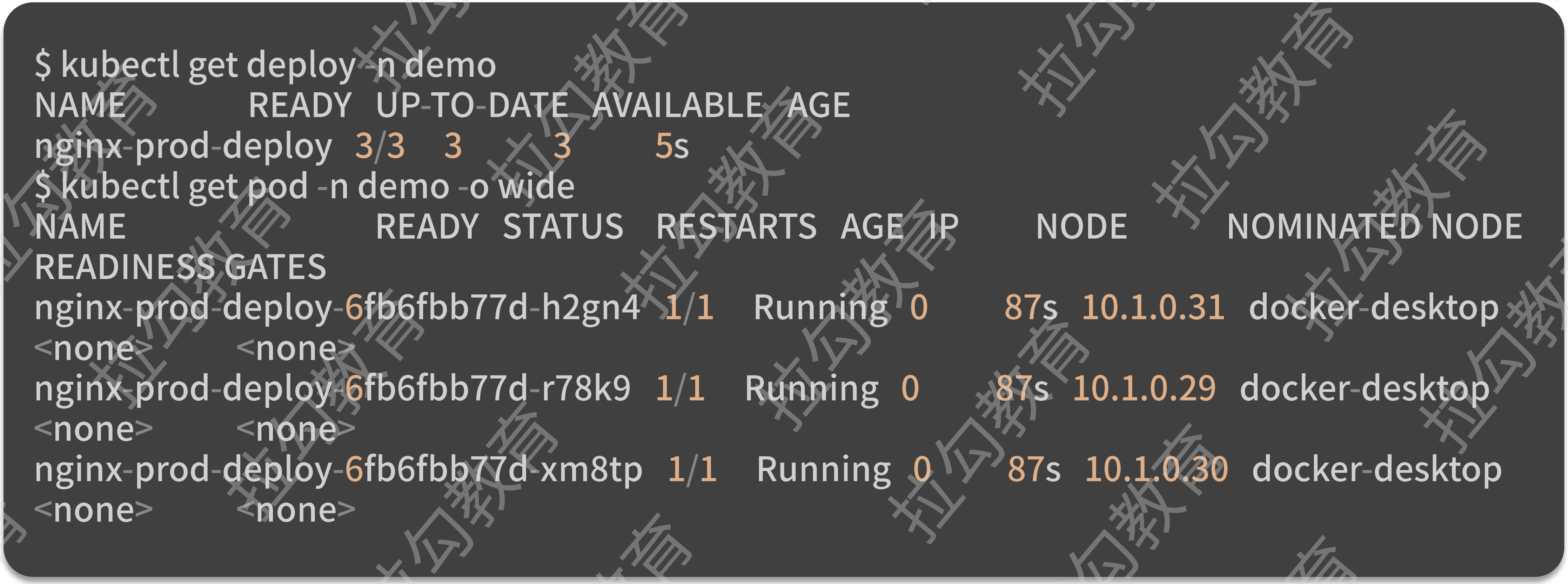
```
$ cat nginx-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-prod-deploy
  namespace: demo
  labels:
    app: nginx
    env: prod
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  template:
    metadata:
      labels:
        app: nginx
```

Kubernetes 中的 Service



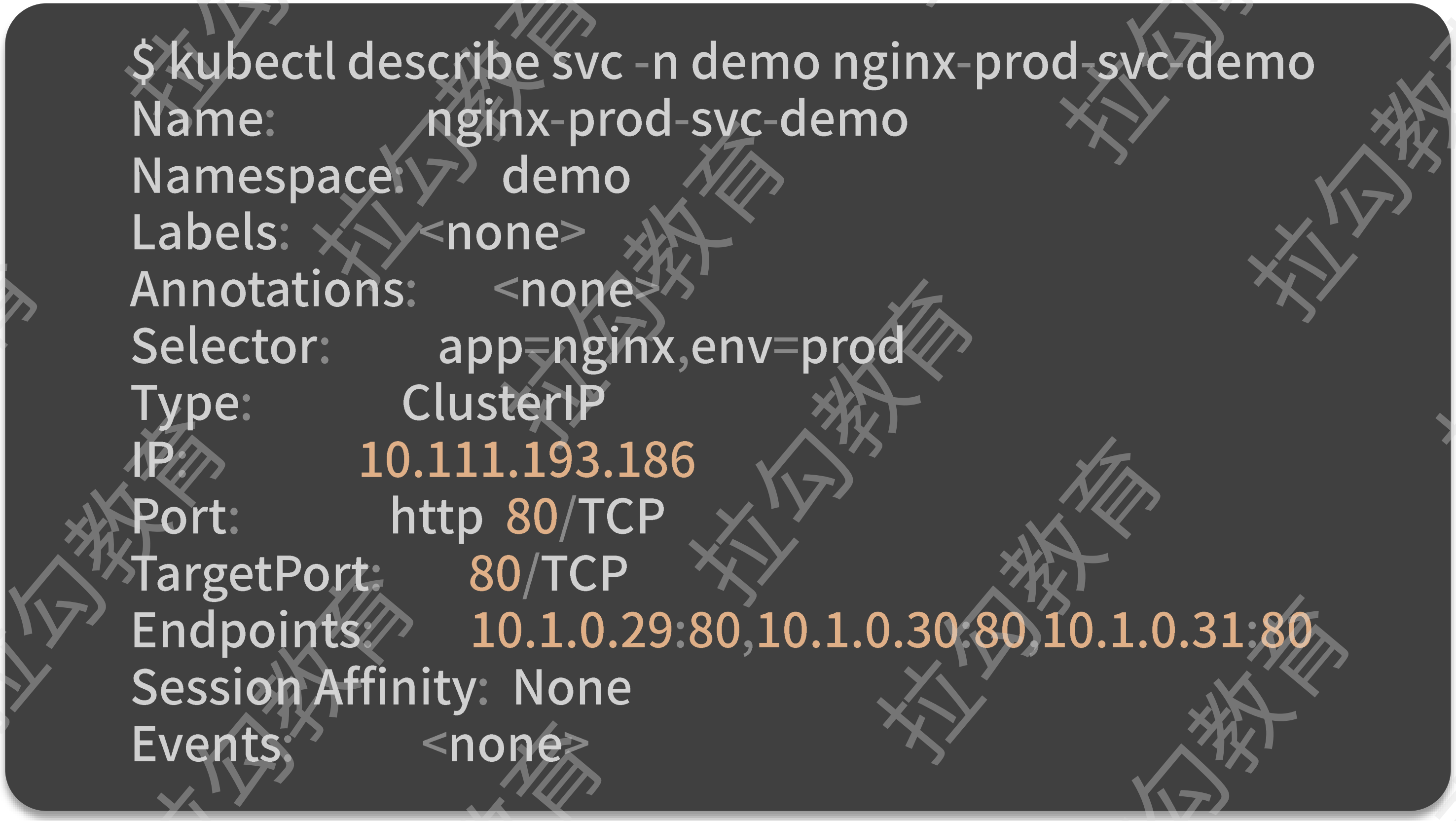
```
    app: nginx
    env: prod
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  template:
    metadata:
      labels:
        app: nginx
        env: prod
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Kubernetes 中的 Service



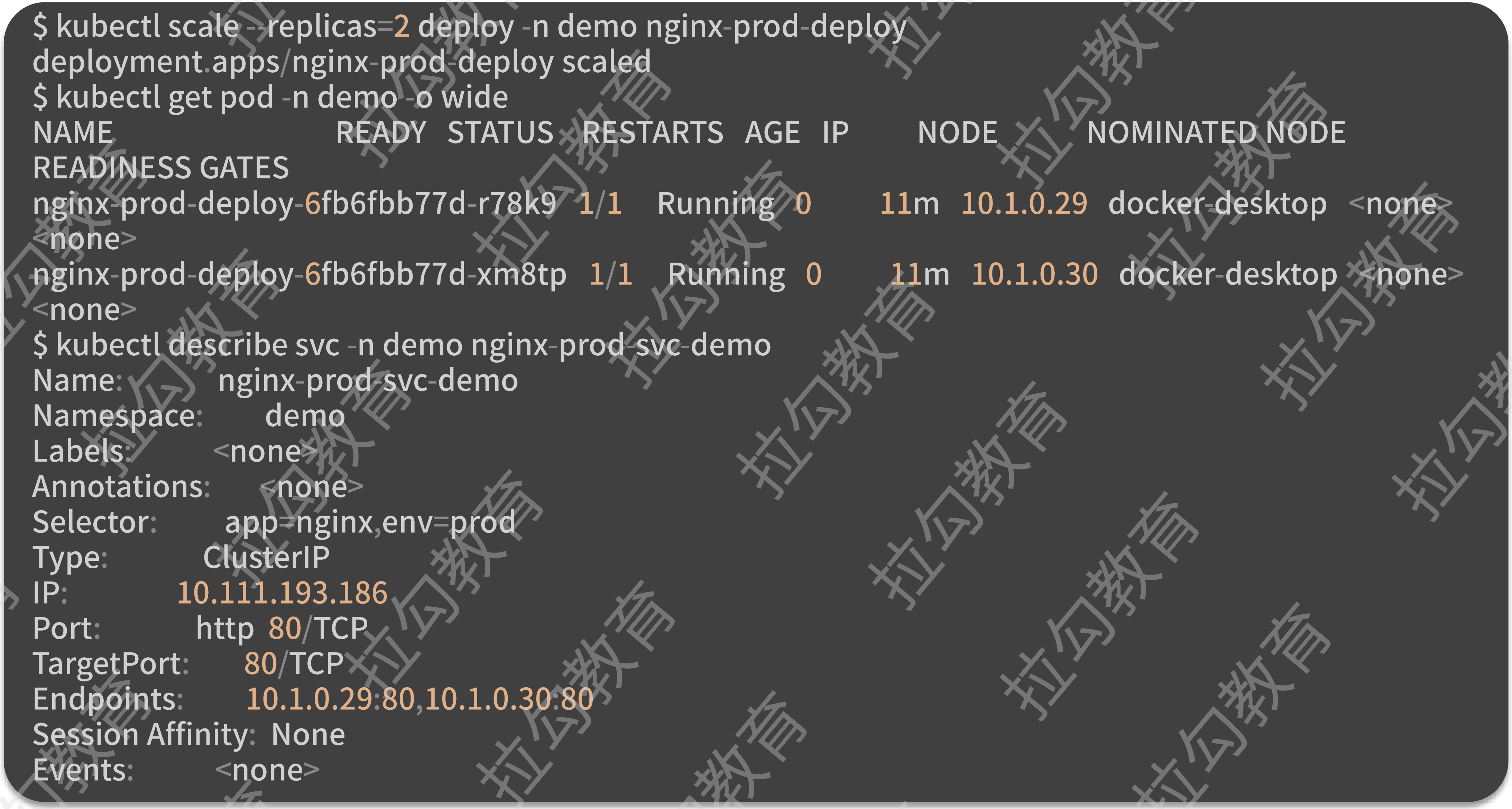
```
$ kubectl get deploy -n demo
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
nginx-prod-deploy   3/3     3            3           5s
$ kubectl get pod -n demo -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE             NOMINATED NODE
READINESS GATES
nginx-prod-deploy-6fb6fbb77d-h2gn4   1/1     Running   0          87s   10.1.0.31   docker-desktop
<none>           <none>
nginx-prod-deploy-6fb6fbb77d-r78k9   1/1     Running   0          87s   10.1.0.29   docker-desktop
<none>           <none>
nginx-prod-deploy-6fb6fbb77d-xm8tp   1/1     Running   0          87s   10.1.0.30   docker-desktop
<none>           <none>
```

Kubernetes 中的 Service



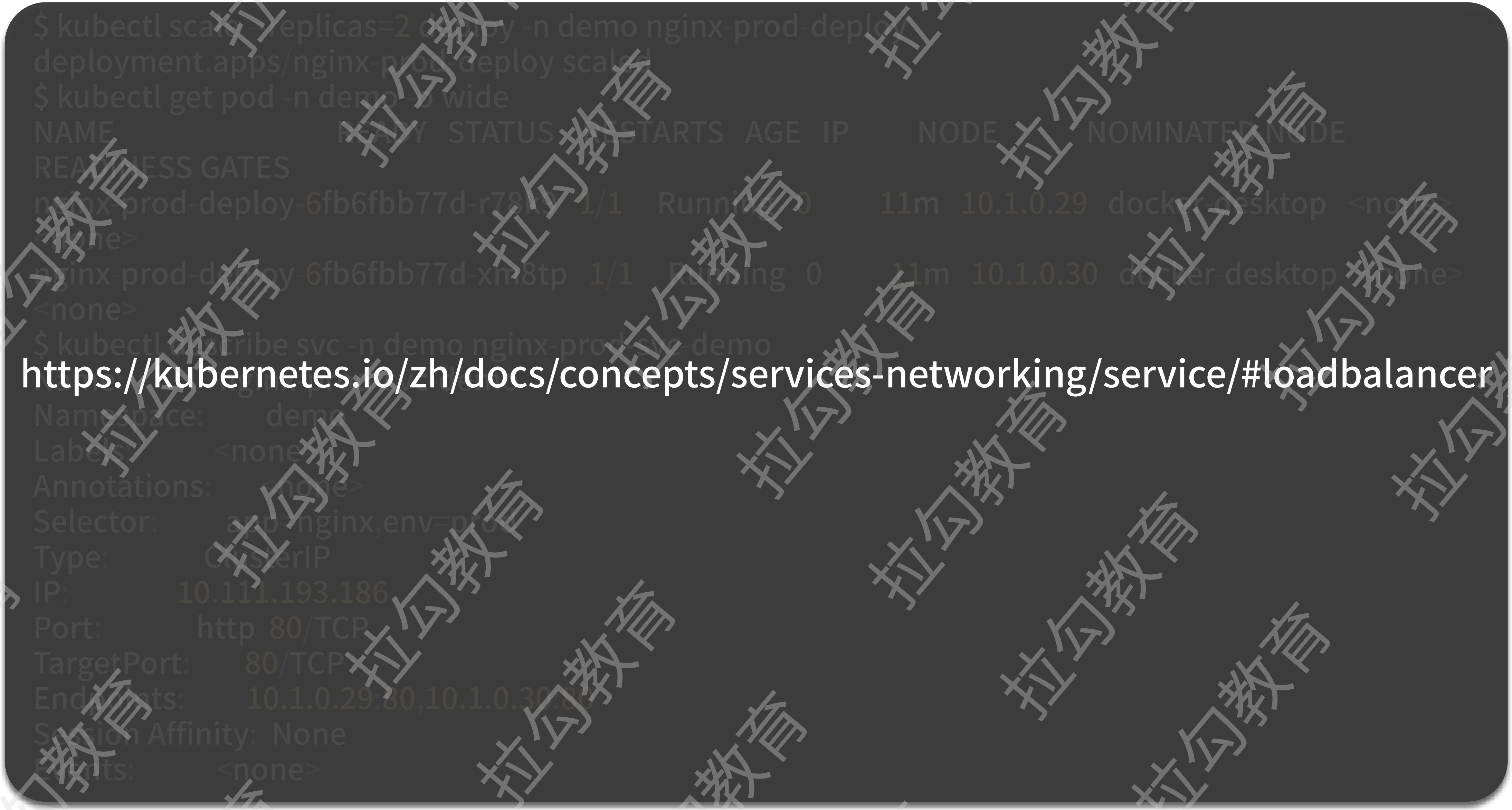
```
$ kubectl describe svc -n demo nginx-prod-svc-demo
Name:              nginx-prod-svc-demo
Namespace:         demo
Labels:            <none>
Annotations:       <none>
Selector:          app=nginx,env=prod
Type:              ClusterIP
IP:                10.111.193.186
Port:              http  80/TCP
TargetPort:        80/TCP
Endpoints:         10.1.0.29:80,10.1.0.30:80,10.1.0.31:80
Session Affinity:  None
Events:            <none>
```

Kubernetes 中的 Service



```
$ kubectl scale --replicas=2 deploy -n demo nginx-prod-deploy
deployment.apps/nginx-prod-deploy scaled
$ kubectl get pod -n demo -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP        NODE          NOMINATED NODE
READINESS GATES
nginx-prod-deploy-6fb6fbb77d-r78k9   1/1     Running   0          11m   10.1.0.29   docker-desktop   <none>
<none>
nginx-prod-deploy-6fb6fbb77d-xm8tp   1/1     Running   0          11m   10.1.0.30   docker-desktop   <none>
<none>
$ kubectl describe svc -n demo nginx-prod-svc-demo
Name:              nginx-prod-svc-demo
Namespace:         demo
Labels:            <none>
Annotations:       <none>
Selector:          app=nginx,env=prod
Type:              ClusterIP
IP:                10.111.193.186
Port:              http  80/TCP
TargetPort:        80/TCP
Endpoints:         10.1.0.29:80,10.1.0.30:80
Session Affinity:  None
Events:            <none>
```

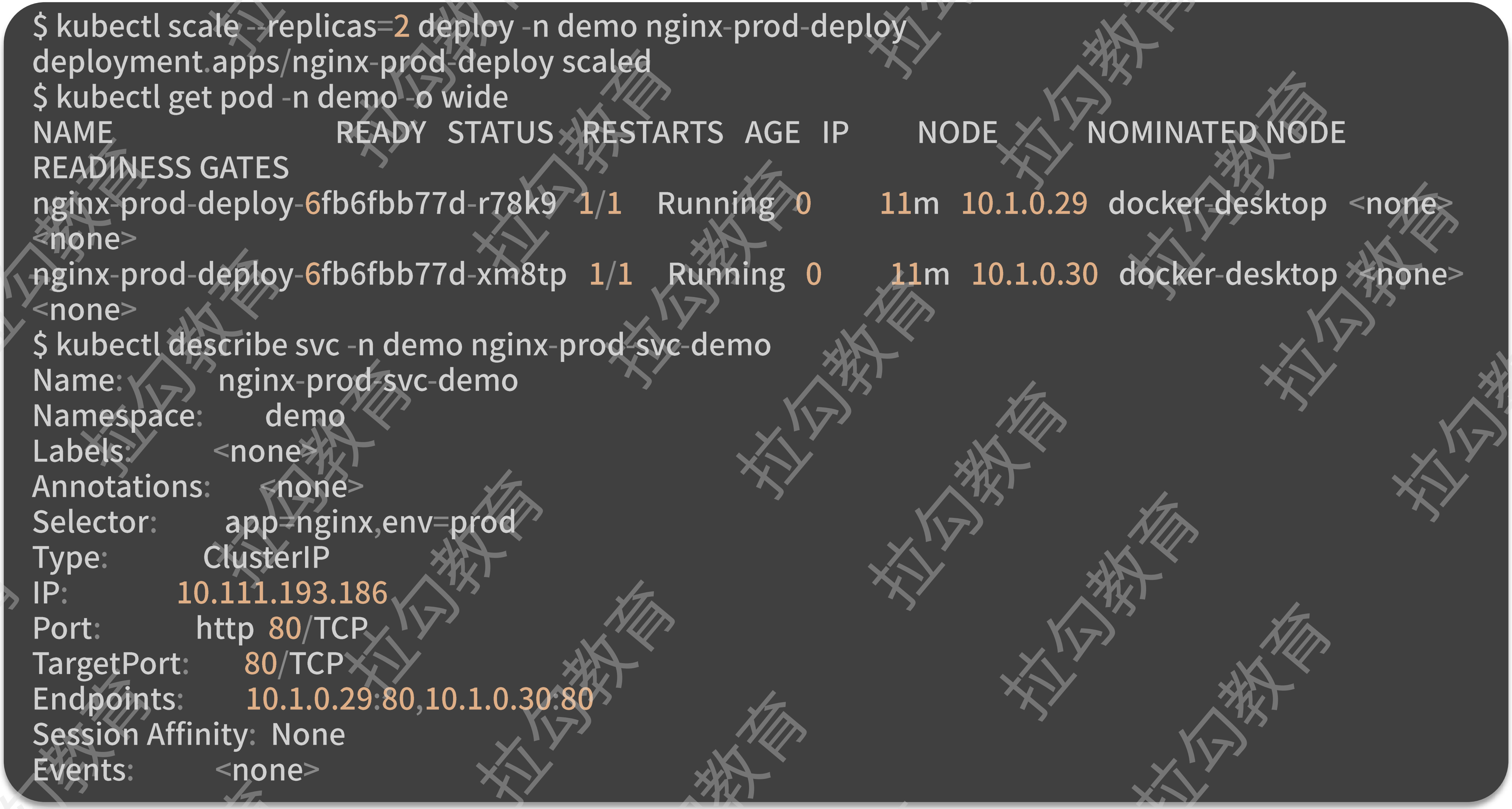
Kubernetes 中的 Service



```
$ kubectl scale --replicas=2 deploy -n demo nginx-prod-deploy
deployment.apps/nginx-prod-deploy scaled
$ kubectl get pod -n demo -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP          NODE             NOMINATED NODE
READINESS GATES
nginx-prod-deploy-6fb6fbb77d-r78k9   1/1     Running   0          11m   10.1.0.29   docker-desktop   <none>
<none>
nginx-prod-deploy-6fb6fbb77d-xm8tp   1/1     Running   0          11m   10.1.0.30   docker-desktop   <none>
<none>
$ kubectl describe svc -n demo nginx-prod-svc-demo
Name:              nginx-prod-svc-demo
Namespace:         demo
Labels:            <none>
Annotations:       <none>
Selector:          app=nginx,env=prod
Type:              ClusterIP
IP:                10.111.193.186
Port:              http  80/TCP
TargetPort:        80/TCP
Endpoints:         10.1.0.29:80,10.1.0.30:80
Session Affinity:  None
Events:            <none>
```

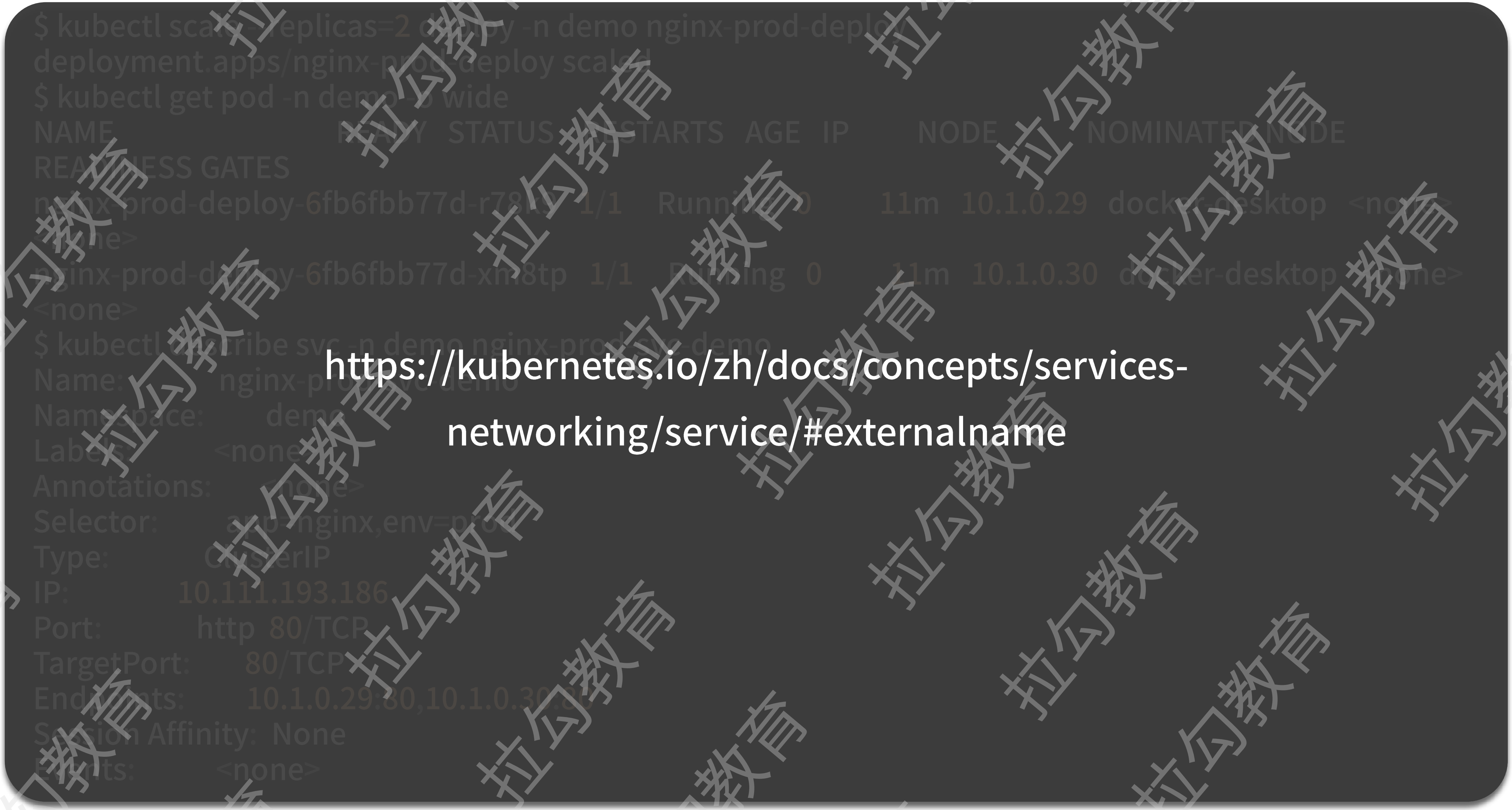
https://kubernetes.io/zh/docs/concepts/services-networking/service/#loadbalancer

Kubernetes 中的 Service



```
$ kubectl scale --replicas=2 deploy -n demo nginx-prod-deploy
deployment.apps/nginx-prod-deploy scaled
$ kubectl get pod -n demo -o wide
NAME                           READY   STATUS    RESTARTS   AGE   IP          NODE             NOMINATED NODE
READINESS GATES
nginx-prod-deploy-6fb6fbb77d-r78k9   1/1     Running   0          11m   10.1.0.29   docker-desktop   <none>
<none>
nginx-prod-deploy-6fb6fbb77d-xm8tp   1/1     Running   0          11m   10.1.0.30   docker-desktop   <none>
<none>
$ kubectl describe svc -n demo nginx-prod-svc-demo
Name:              nginx-prod-svc-demo
Namespace:         demo
Labels:            <none>
Annotations:       <none>
Selector:          app=nginx,env=prod
Type:              ClusterIP
IP:                10.111.193.186
Port:              http  80/TCP
TargetPort:        80/TCP
Endpoints:         10.1.0.29:80,10.1.0.30:80
Session Affinity:  None
Events:            <none>
```

Kubernetes 中的 Service



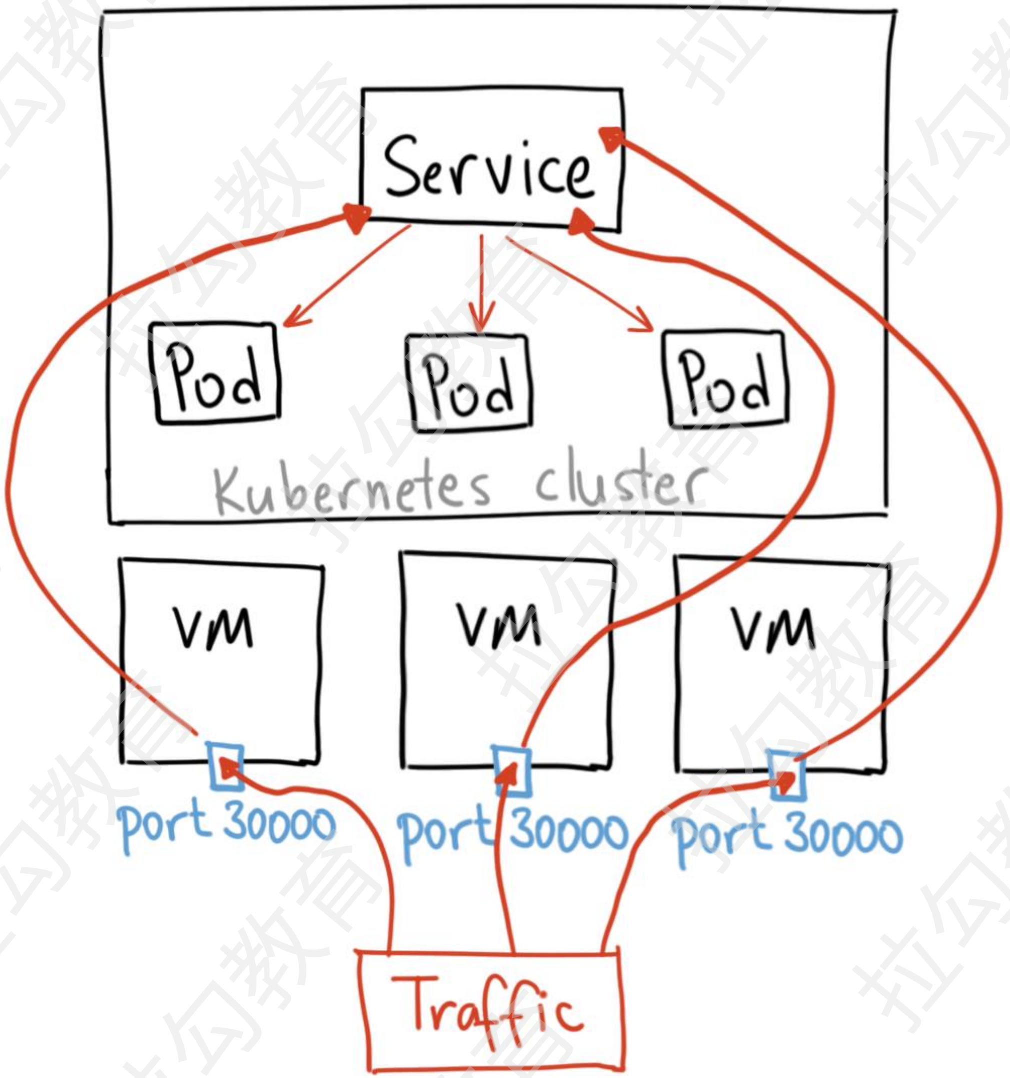
```
$ kubectl scale --replicas=2 deploy -n demo nginx-prod-deploy
deployment.apps/nginx-prod-deploy scaled
$ kubectl get pod -n demo -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP          NODE             NOMINATED NODE
READINESS GATES
nginx-prod-deploy-6fb6fbb77d-r78k9   1/1     Running   0          11m   10.1.0.29   docker-desktop   <none>
<none>
nginx-prod-deploy-6fb6fbb77d-xm8tp   1/1     Running   0          11m   10.1.0.30   docker-desktop   <none>
<none>
$ kubectl describe svc -n demo nginx-prod-svc-demo
Name:              nginx-prod-svc-demo
Namespace:         demo
Labels:            <none>
Annotations:       <none>
Selector:          app=nginx,env=prod
Type:              ClusterIP
IP:                10.111.193.186
Port:              http  80/TCP
TargetPort:        80/TCP
Endpoints:         10.1.0.29:80,10.1.0.30:80
Session Affinity:  None
Events:            <none>
```

https://kubernetes.io/zh/docs/concepts/services-networking/service/#externalname

Kubernetes 中的 Service

```
apiVersion: v1
kind: Service
metadata:
  name: my-nodeport-service
  namespace: demo
spec:
  selector:
    app: my-app
  type: NodePort #这里设置类型为 NodePort
  ports:
  - name: http
    port: 80
    targetPort: 80
    nodePort: 30000
    protocol: TCP
```

Kubernetes 中的 Service



集群内如何访问 Service?

如果该 Service 有 ClusterIP
可以直接用这个虚拟 IP 去访问

比如nginx-prod-svc-demo 这个 Service
通过kubectl get svc nginx-prod-svc-demo -n
dmeo或kubectl get svc nginx-prod-svc-demo -n
dmeo 就可以看到其 Cluster IP 为 10.111.193.186
端口号为 80
那么通过 http(s)://10.111.193.186:80
就可以访问到该服务

集群内如何访问 Service?

如果该 Service 有 ClusterIP
可以直接用这个虚拟 IP 去访问

比如nginx-prod-svc-demo 这个 Service
通过kubectl get svc nginx-prod-svc-demo -n dmeo或kubectl get svc nginx-prod-svc-demo -n dmeo 就可以看到其 Cluster IP 为 10.111.193.186
端口号为 80
那么通过 http(s)://10.111.193.186:80
就可以访问到该服务

当然也可以使用该 Service 的域名
依赖于集群内部的 DNS 即可访问

同 namespace 下的 Pod 可以直接通过
nginx-prod-svc-demo 这个 Service 名去访问
如果是不同 namespace 下的 Pod 则需要加上该
Service 所在的 namespace 名
即nginx-prod-svc-demo.demo去访问

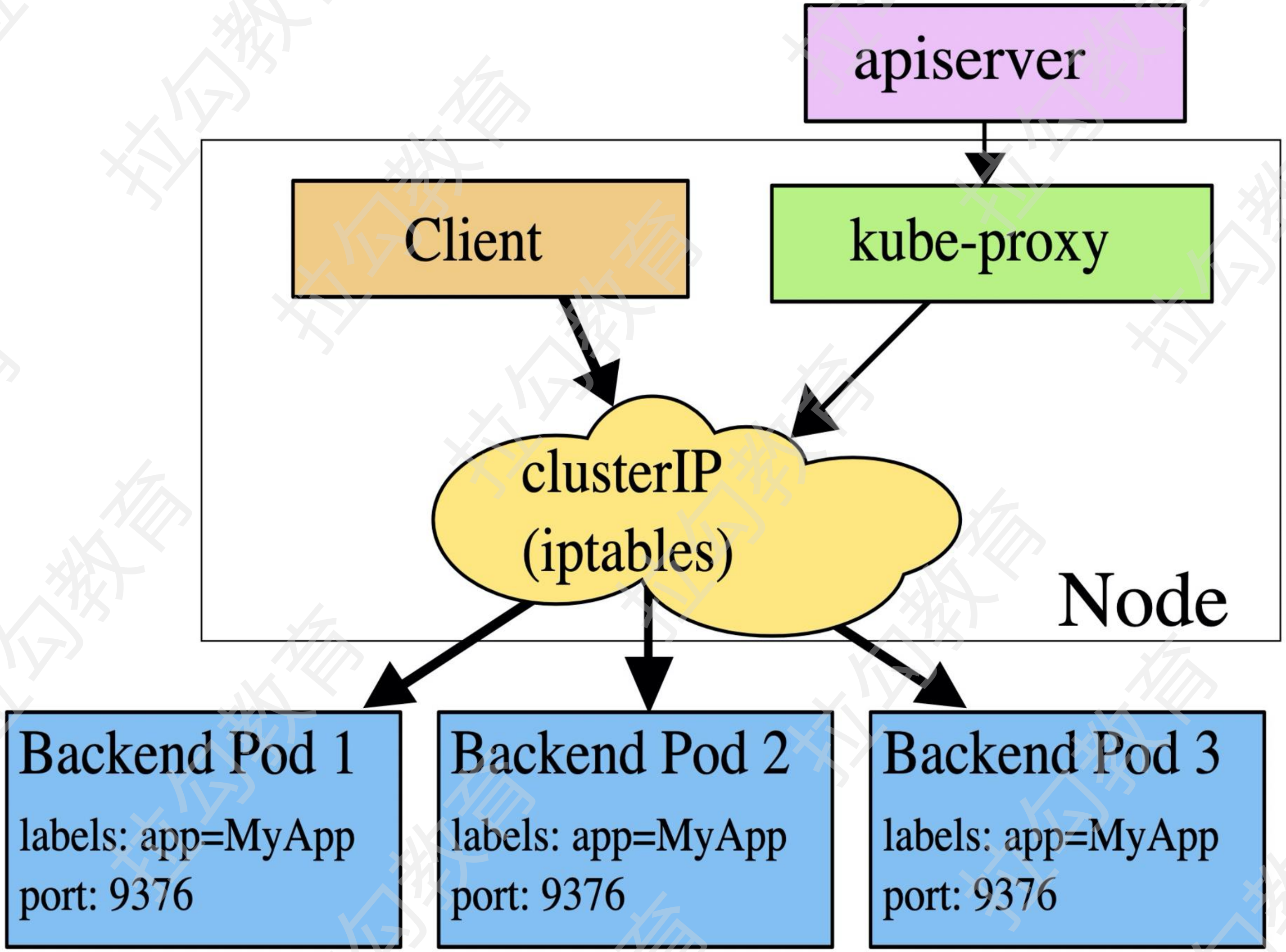
集群内如何访问 Service?

```
# env
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_SERVICE_PORT=443
HOSTNAME=nginx-prod-deploy2-68d8fb9586-4m5hr
NGINX_PROD_SVC_DEMO_SERVICE_PORT_HTTP=80
NGINX_PROD_SVC_DEMO_SERVICE_HOST=10.111.193.186
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
NGINX_PROD_SVC_DEMO_SERVICE_PORT=80
NGINX_PROD_SVC_DEMO_PORT=tcp://10.111.193.186:80
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
NGINX_PROD_SVC_DEMO_PORT_80_TCP_ADDR=10.111.193.186
NGINX_PROD_SVC_DEMO_PORT_80_TCP_PORT=80
NGINX_PROD_SVC_DEMO_PORT_80_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_SERVICE_HOST=10.96.0.1
NGINX_PROD_SVC_DEMO_PORT_80_TCP=tcp://10.111.193.186:80
...
```

集群内部的负载均衡如何实现?

https://kubernetes.io/zh/docs/concepts/services-networking/service/#proxy-mode-ipvs

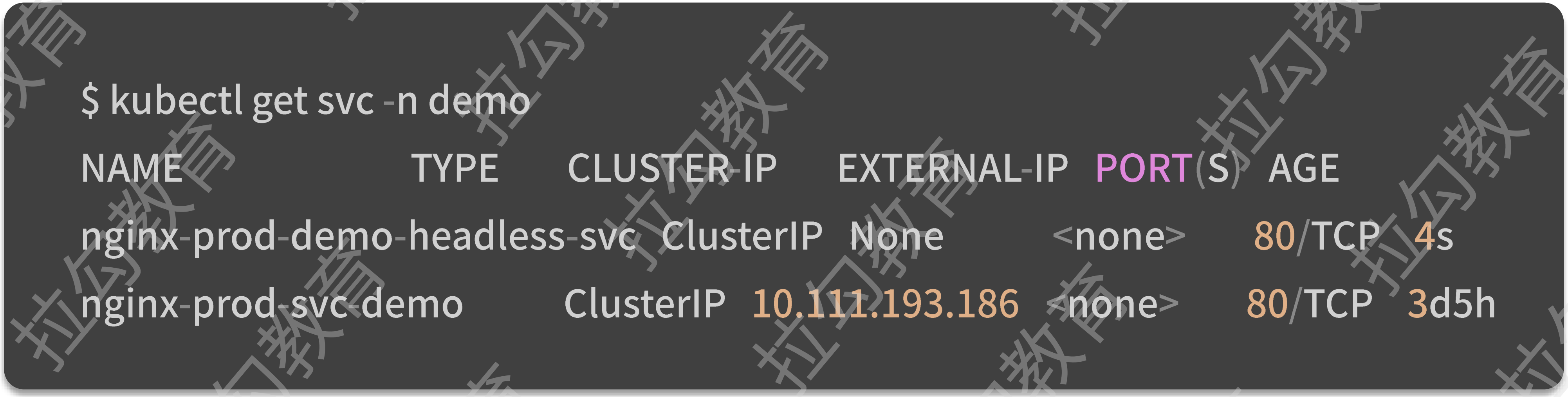
集群内部的负载均衡如何实现?



Headless Service

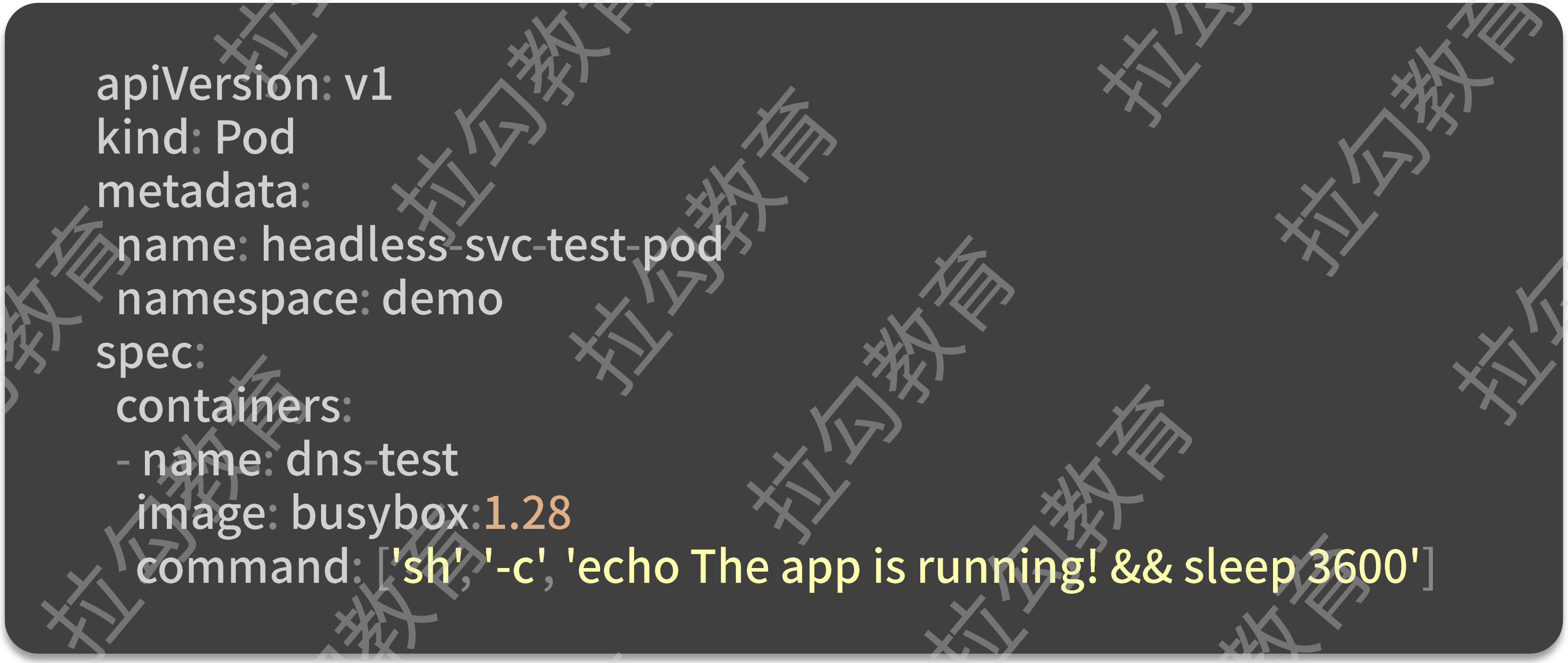
```
apiVersion: v1
kind: Service
metadata:
  name: nginx-prod-demo-headless-svc
  namespace: demo
spec:
  clusterIP: None
  selector:
    app: nginx
    env: prod
  type: ClusterIP
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
```

Headless Service



```
$ kubectl get svc -n demo
NAME                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
nginx-prod-demo-headless-svc  ClusterIP   None             <none>        80/TCP    4s
nginx-prod-svc-demo           ClusterIP   10.111.193.186   <none>        80/TCP    3d5h
```

Headless Service



```yaml
apiVersion: v1
kind: Pod
metadata:
 name: headless-svc-test-pod
 namespace: demo
spec:
 containers:
 - name: dns-test
   image: busybox:1.28
   command: ['sh', '-c', 'echo The app is running! && sleep 3600']
```

Headless Service

```
$ kubectl exec -it -n demo headless-svc-test-pod sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use
kubectl kubectl exec [POD] -- [COMMAND] instead.
/ # nslookup nginx-prod-demo-headless-svc
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local


Name:      nginx-prod-demo-headless-svc
Address 1: 10.1.0.32 10-1-0-32.nginx-prod-demo-headless-svc.demo.svc.cluster.local
Address 2: 10.1.0.33 10-1-0-33.nginx-prod-demo-headless-svc.demo.svc.cluster.local
/ # nslookup nginx-prod-svc-demo
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local


Name:      nginx-prod-svc-demo
Address 1: 10.111.193.186 nginx-prod-svc-demo.demo.svc.cluster.local
```

Headless Service

2

可用于部署有状态服务

每个 StatefulSet 管理的 Pod 都有一个单独的 DNS 记录，且域名保持不变

即 <PodName>.<ServiceName>.<NamespaceName>.svc.cluster.local

这样 Statefulset 中的各个 Pod 就可以直接通过 Pod 名字解决相互间身份以及访问问题

1

用户可以自己选择要连接哪个 Pod

通过查询 Service 的 DNS 记录来获取后端真实负载的 IP 地址，自主选择要连接哪个 IP

写在最后

Service 是 Kubernetes 很重要的对象

主要负责为各种工作负载暴露服务，方便各个服务之间互访

Next：《12 | Helm Charts：如何在生产环境中释放部署生产力？》

拉勾教育

—互联网人实战大学—

关注拉勾「教育公众号」
获取更多课程信息

L / A / G / O / U