掘金小册

如何快速上手 Kubernetes

张晋涛

个人介绍

- 张晋涛 @API7.AI
- Apache APISIX Committer
- Kubernetes Ingress-nginx reviewer
- containerd/Docker/Helm/Kubernetes/KIND contributor
- 『K8S 生态周报』维护者
- 《Kubernetes上手实践》小册作者
- https://github.com/tao12345666333

Agenda

- Kubernetes 概览
- Kubernetes 环境准备
- Kubernetes 中的应用部署
- Kubernetes 中的流量路由

掘金小册

Kubernetes 概览

Kubernetes 发展历程

- 2014 年由 Google 开源
- 大规模场景下 Docker 容器编排
- 隔离性 / 标准化
- 滚动更新
- 故障自愈
- 动态扩 / 缩容

Kubernetes 发展历程

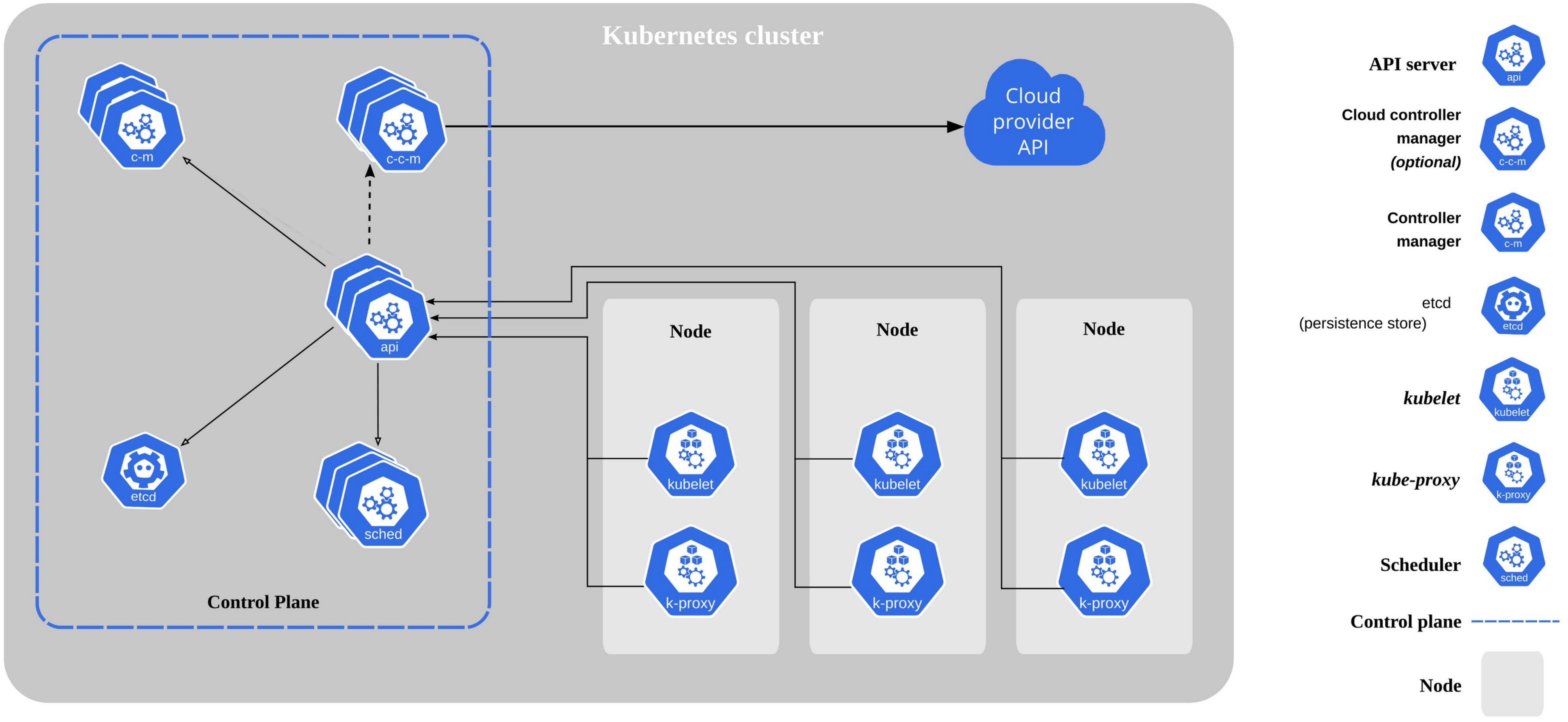
- CNCF 首个毕业项目
- Kubernetes 在生产环境中的应用比例达到 83%
- GitHub star 数 80.9k

注：以上统计数据来自 CNCF 的 2020 调查报告

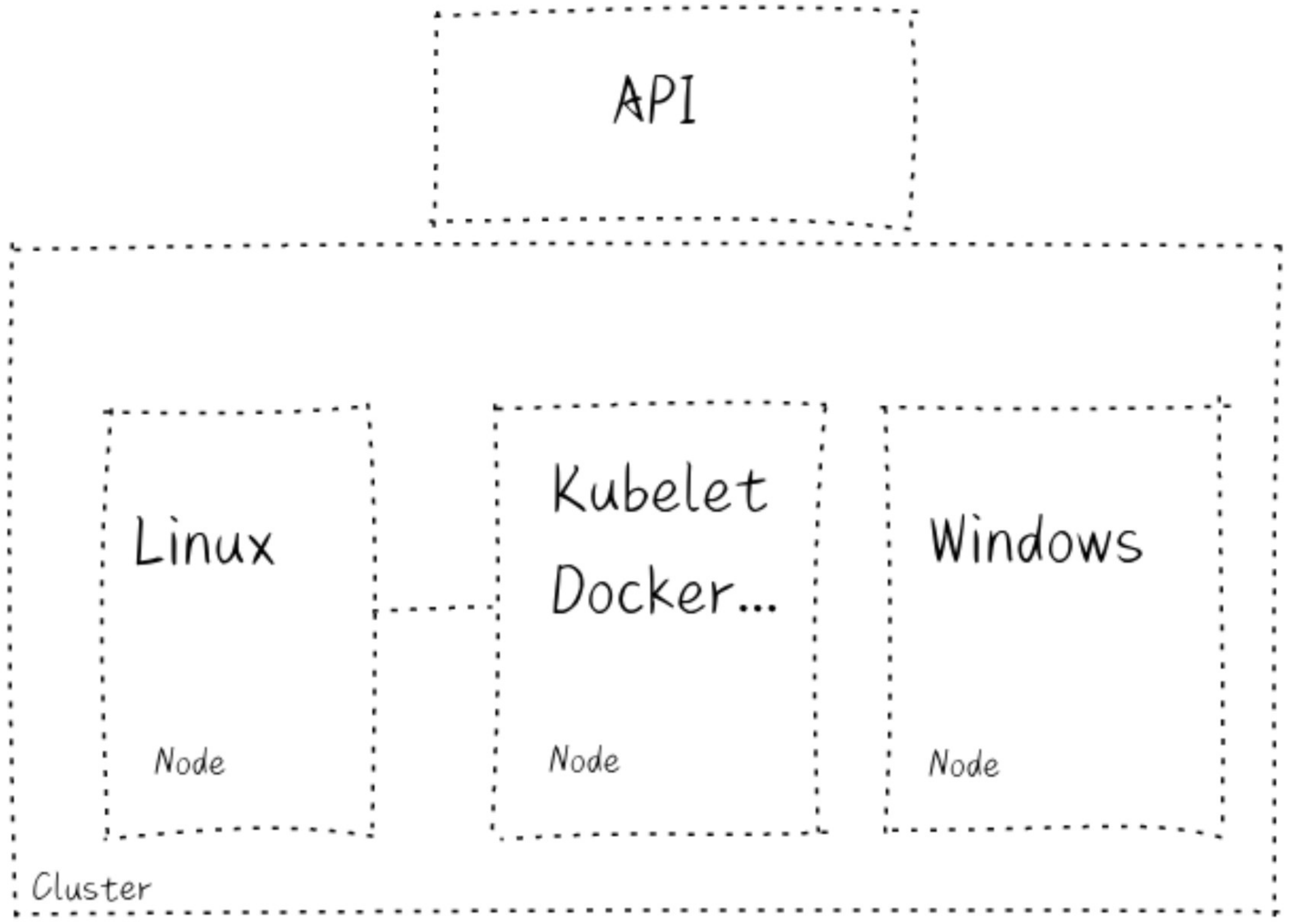
为什么需要 Kubernetes

- 大规模多节点下的容器编排
- 快速扩 / 缩容
- 故障自愈
- 弹性
- 技术趋势
- 一致性
- 不锁定

Kubernetes 架构



理解 Kubernetes



掘金小册

Kubernetes 环境准备

在线环境

- Katacoda
- Play with Kubernetes

本地测试环境

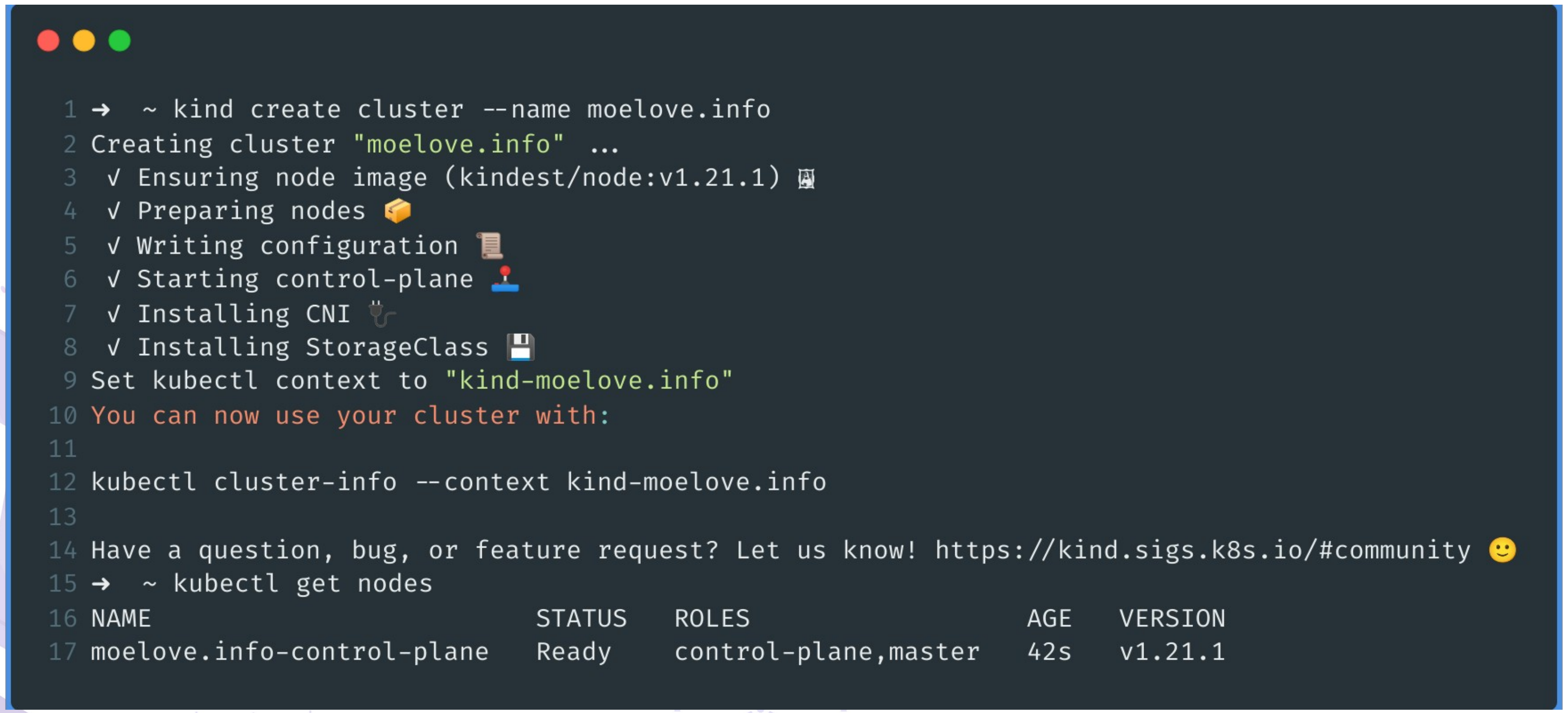
- Docker Desktop （内置）
- KIND （ Kubernetes In Docker ）
- Lima
- Minikube
- Microk8s
- …

生产部署

- kubeadm （推荐）
- kops （ AWS ）
- Kubespray
- ...

使用 KIND 创建本地环境

- 准备 Docker 环境
- 下载 KIND 二进制文件



```
➜  ~ kind create cluster --name moelove.info
Creating cluster "moelove.info" ...
 ✓ Ensuring node image (kindest/node:v1.21.1) 🖼
 ✓ Preparing nodes 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
Set kubectl context to "kind-moelove.info"
You can now use your cluster with:

kubectl cluster-info --context kind-moelove.info

Have a question, bug, or feature request? Let us know! https://kind.sigs.k8s.io/#community 🙂
➜  ~ kubectl get nodes
NAME                         STATUS   ROLES                  AGE   VERSION
moelove.info-control-plane   Ready    control-plane,master   42s   v1.21.1
```

掘金小册

Kubernetes 中的应用部署

Kubernetes 中的核心概念

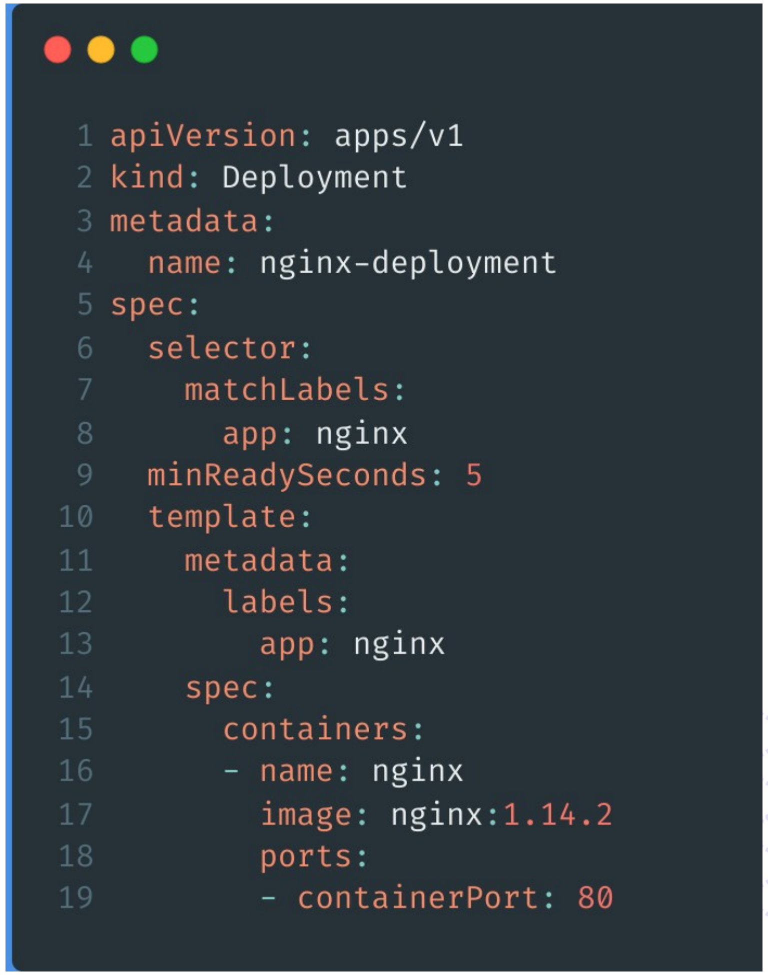
- Resource Object

- Spec
- Status

- Pod
- Deployment
- Daemonset
- Service
- Statefulset
- Job/CronJob
- Ingress
- Node

Kubernetes 核心设计理念

- 声明式
- 无侵入性
- 可移植（提供抽象）
- 显式定义

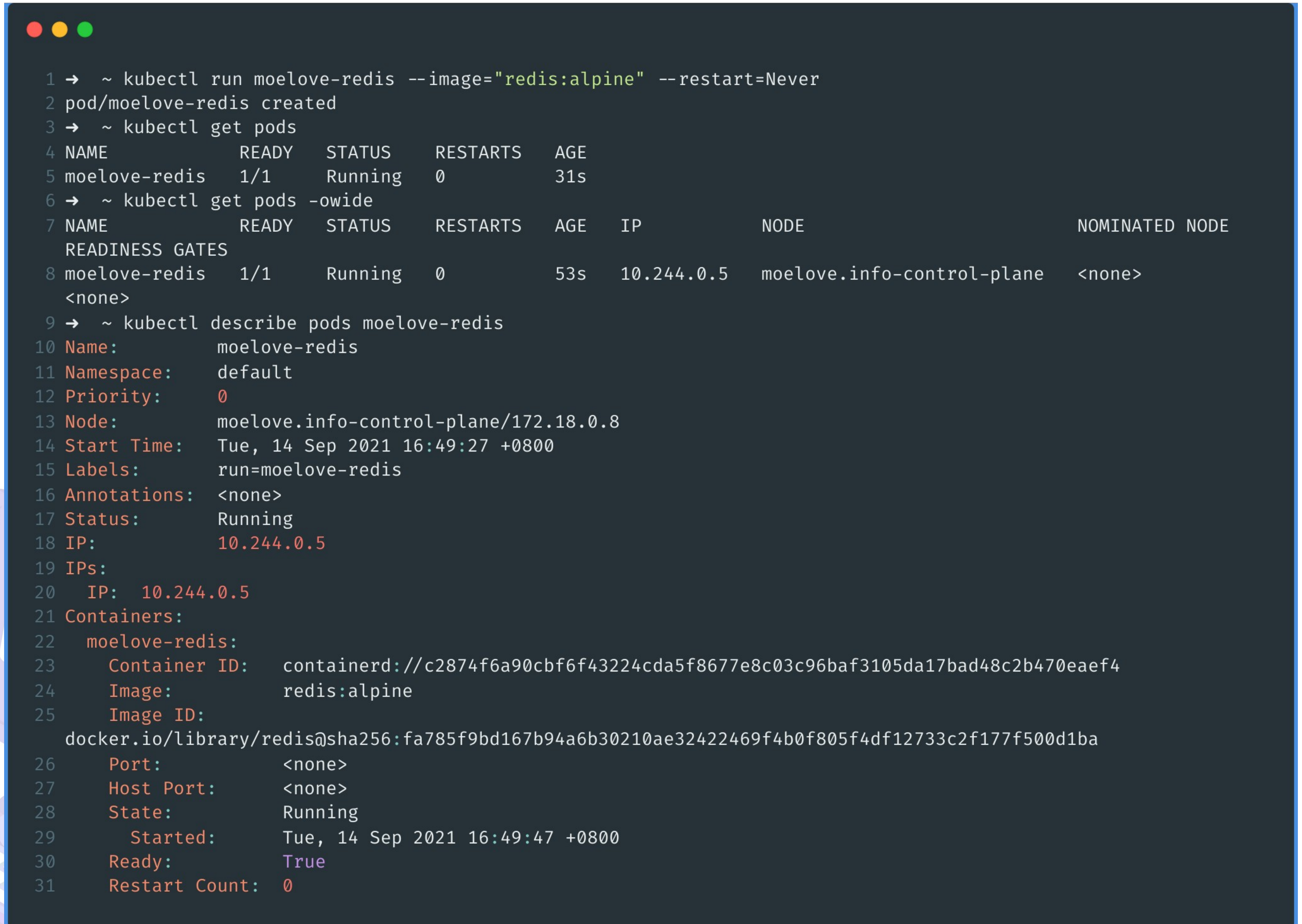


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  minReadySeconds: 5
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Kubernetes 如何管理容器

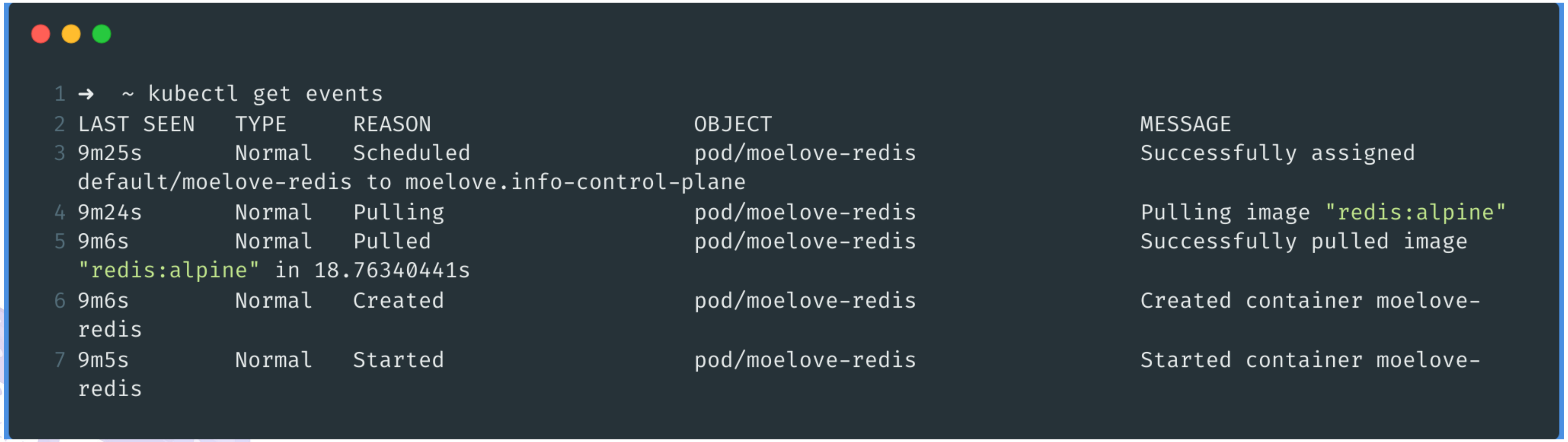
- Kubernetes 中最小调度单元是 Pod
- 每个 Pod 可以包含多个 container
- 每个 Pod 有自己的 IP
- 可通过集群网络与其他 Pod 通信
- Pod 内容器共享网络堆栈

Kubernetes 运行 Pod



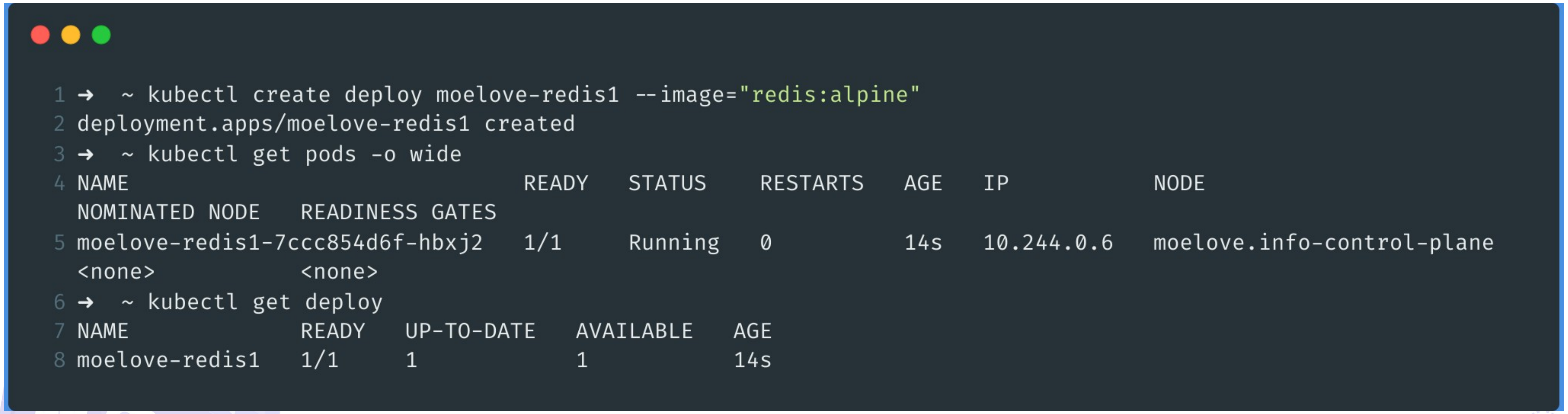
```
➜  ~ kubectl run moelove-redis --image="redis:alpine" --restart=Never
pod/moelove-redis created
➜  ~ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
moelove-redis   1/1     Running   0          31s
➜  ~ kubectl get pods -owide
NAME            READY   STATUS    RESTARTS   AGE   IP           NODE                          NOMINATED NODE
READINESS GATES
moelove-redis   1/1     Running   0          53s   10.244.0.5   moelove.info-control-plane    <none>
<none>
➜  ~ kubectl describe pods moelove-redis
Name:          moelove-redis
Namespace:     default
Priority:      0
Node:          moelove.info-control-plane/172.18.0.8
Start Time:    Tue, 14 Sep 2021 16:49:27 +0800
Labels:        run=moelove-redis
Annotations:   <none>
Status:        Running
IP:            10.244.0.5
IPs:
  IP:  10.244.0.5
Containers:
  moelove-redis:
    Container ID:   containerd://c2874f6a90cbf6f43224cda5f8677e8c03c96baf3105da17bad48c2b470eaef4
    Image:          redis:alpine
    Image ID:
docker.io/library/redis@sha256:fa785f9bd167b94a6b30210ae32422469f4b0f805f4df12733c2f177f500d1ba
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Tue, 14 Sep 2021 16:49:47 +0800
    Ready:          True
    Restart Count:  0
```

Kubernetes 如何启动 Pod



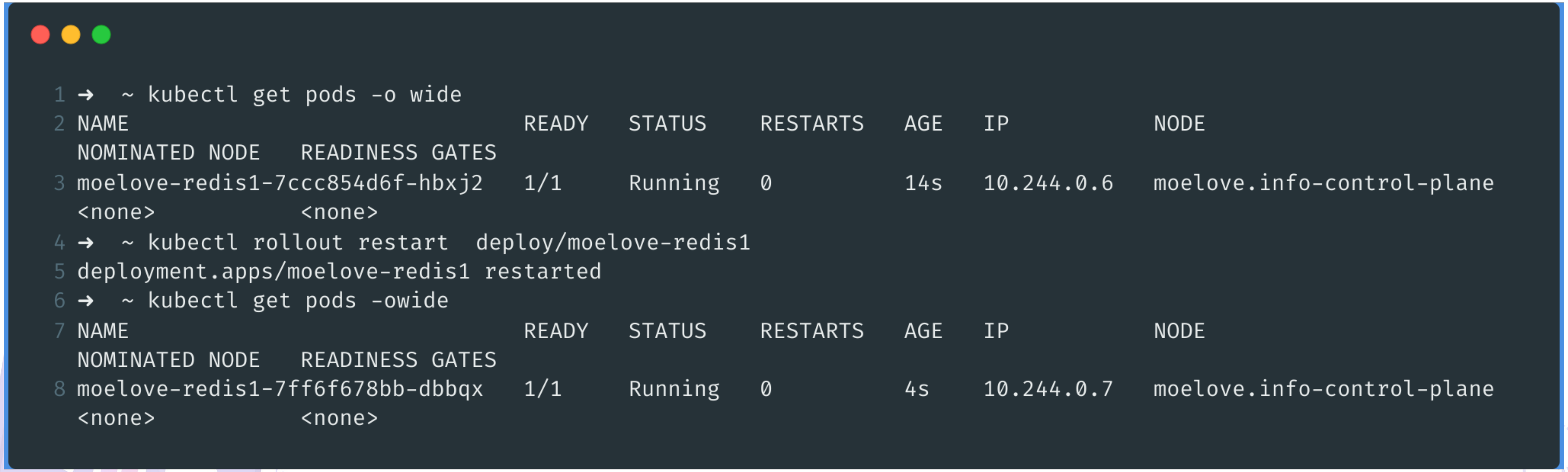
```
➜  ~ kubectl get events
LAST SEEN   TYPE     REASON                       OBJECT                          MESSAGE
9m25s       Normal   Scheduled                    pod/moelove-redis               Successfully assigned
default/moelove-redis to moelove.info-control-plane
9m24s       Normal   Pulling                      pod/moelove-redis               Pulling image "redis:alpine"
9m6s        Normal   Pulled                       pod/moelove-redis               Successfully pulled image
"redis:alpine" in 18.76340441s
9m6s        Normal   Created                      pod/moelove-redis               Created container moelove-
redis
9m5s        Normal   Started                      pod/moelove-redis               Started container moelove-
redis
```

Kubernetes 使用 Deployment 部署



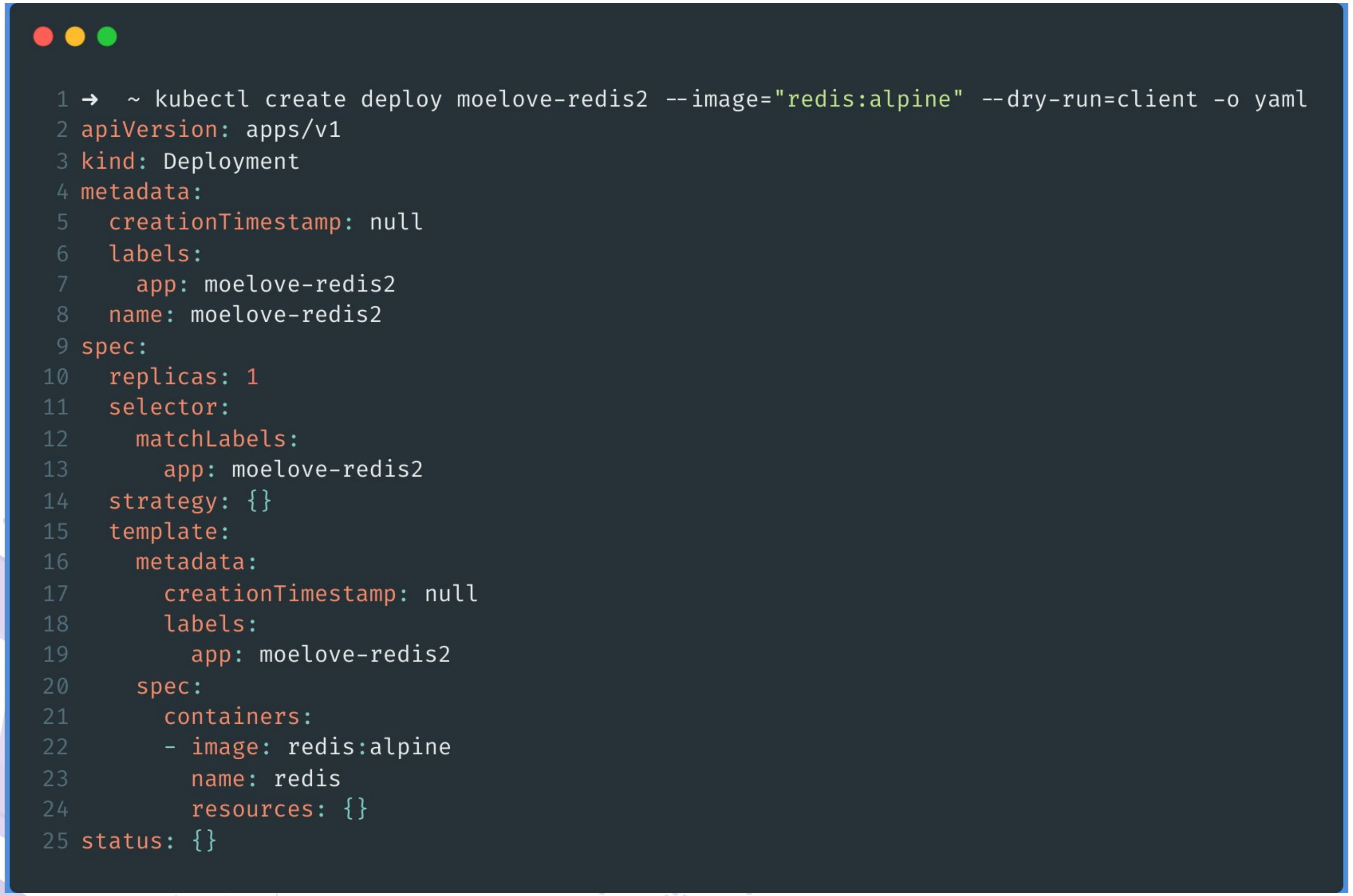
```
➜  ~ kubectl create deploy moelove-redis1 --image="redis:alpine"
deployment.apps/moelove-redis1 created
➜  ~ kubectl get pods -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP            NODE
NOMINATED NODE   READINESS GATES
moelove-redis1-7ccc854d6f-hbxj2   1/1     Running   0          14s   10.244.0.6    moelove.info-control-plane
<none>           <none>
➜  ~ kubectl get deploy
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
moelove-redis1   1/1     1            1           14s
```

Kubernetes 重启 Pod



```
➜  ~ kubectl get pods -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP            NODE
NOMINATED NODE   READINESS GATES
moelove-redis1-7ccc854d6f-hbxj2   1/1     Running   0          14s   10.244.0.6    moelove.info-control-plane
<none>           <none>
➜  ~ kubectl rollout restart  deploy/moelove-redis1
deployment.apps/moelove-redis1 restarted
➜  ~ kubectl get pods -owide
NAME                              READY   STATUS    RESTARTS   AGE   IP            NODE
NOMINATED NODE   READINESS GATES
moelove-redis1-7ff6f678bb-dbbqx   1/1     Running   0          4s    10.244.0.7    moelove.info-control-plane
<none>           <none>
```

Kubernetes 使用 YAML 部署

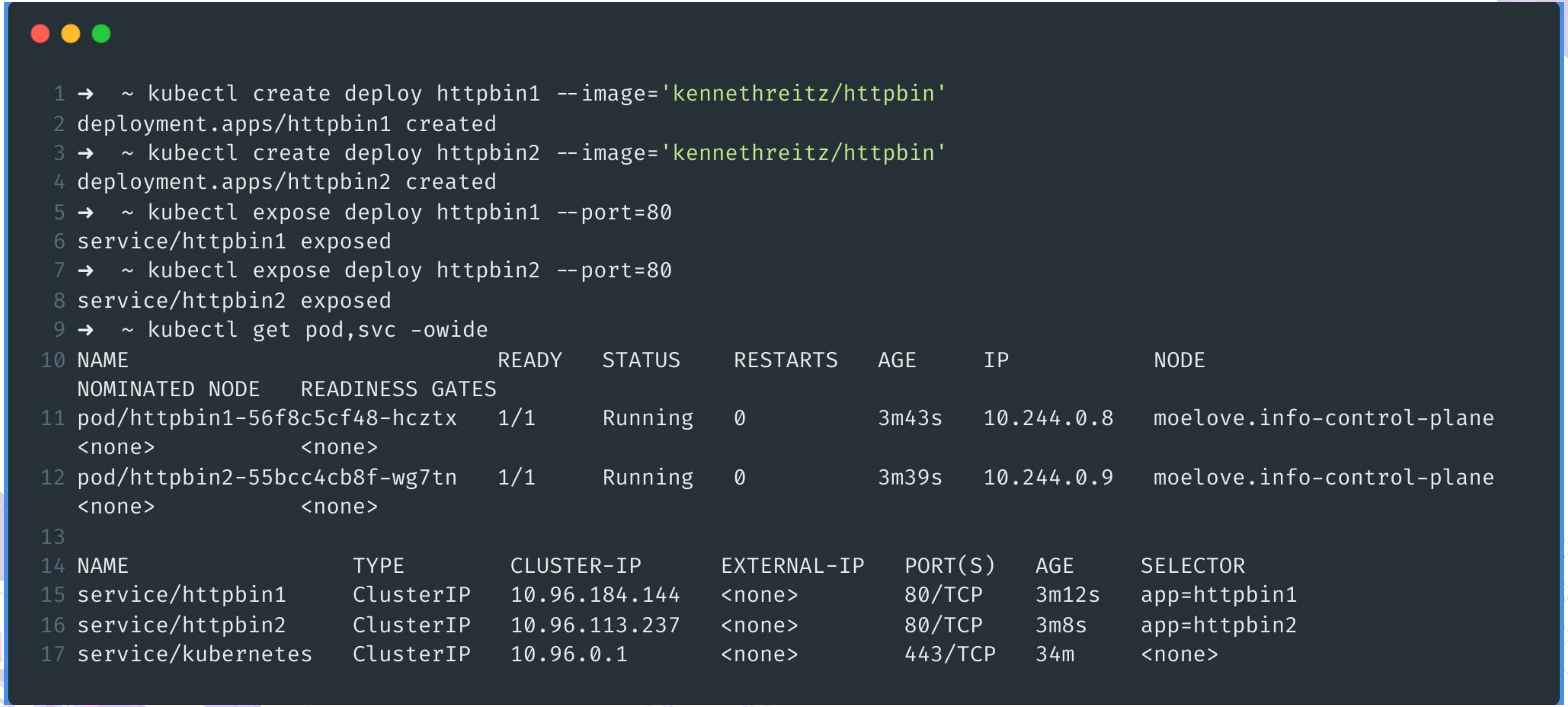


```
➜  ~ kubectl create deploy moelove-redis2 --image="redis:alpine" --dry-run=client -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: moelove-redis2
  name: moelove-redis2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: moelove-redis2
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: moelove-redis2
    spec:
      containers:
      - image: redis:alpine
        name: redis
        resources: {}
status: {}
```

掘金小册

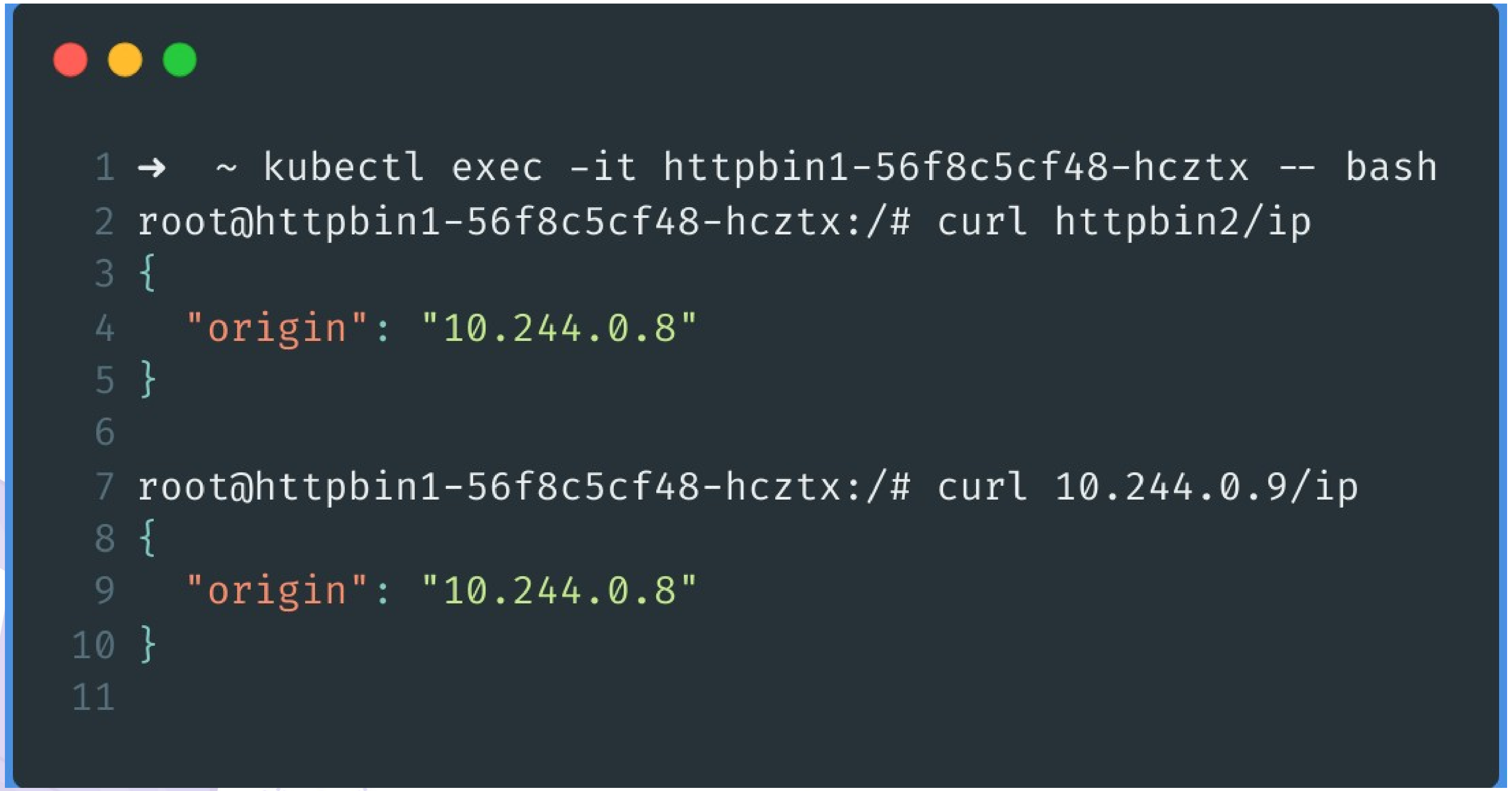
Kubernetes 中的流量路由

Kubernetes 中的 Service



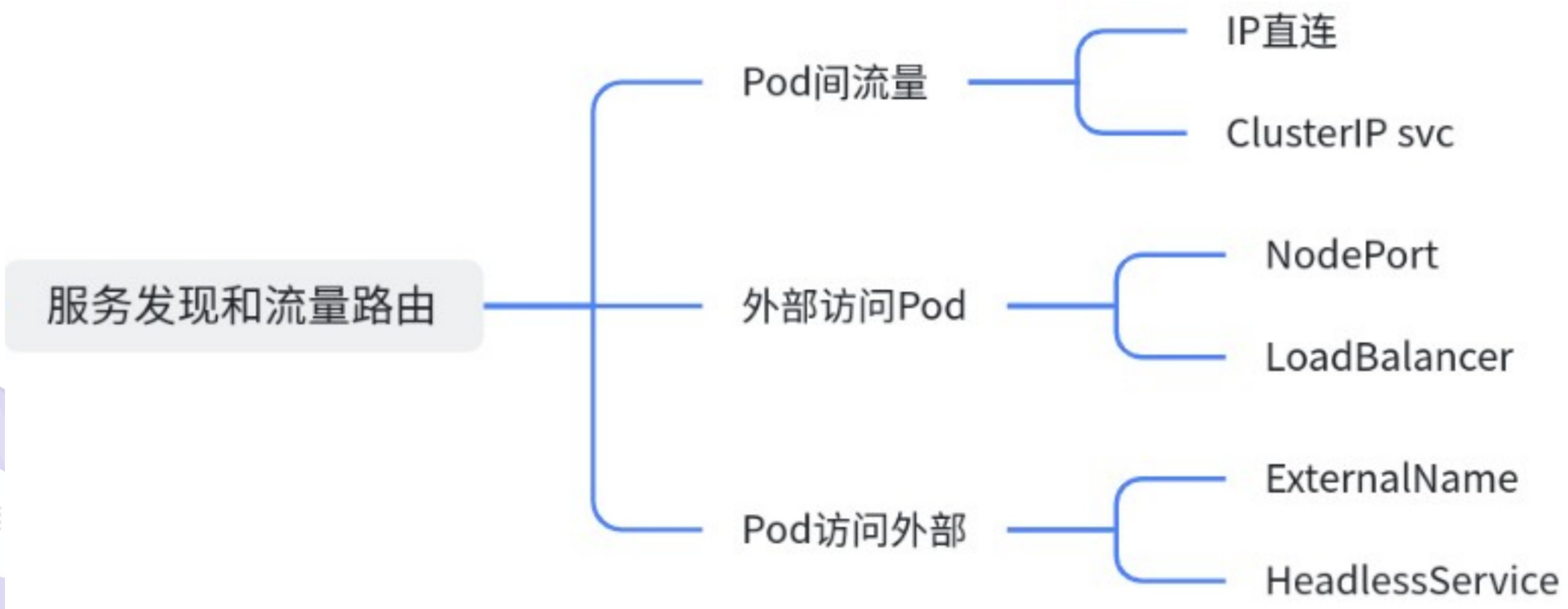
```
➜  ~ kubectl create deploy httpbin1 --image='kennethreitz/httpbin'
deployment.apps/httpbin1 created
➜  ~ kubectl create deploy httpbin2 --image='kennethreitz/httpbin'
deployment.apps/httpbin2 created
➜  ~ kubectl expose deploy httpbin1 --port=80
service/httpbin1 exposed
➜  ~ kubectl expose deploy httpbin2 --port=80
service/httpbin2 exposed
➜  ~ kubectl get pod,svc -owide
NAME                           READY   STATUS    RESTARTS   AGE     IP           NODE
NOMINATED NODE   READINESS GATES
pod/httpbin1-56f8c5cf48-hcztx   1/1     Running   0          3m43s   10.244.0.8   moelove.info-control-plane
<none>           <none>
pod/httpbin2-55bcc4cb8f-wg7tn   1/1     Running   0          3m39s   10.244.0.9   moelove.info-control-plane
<none>           <none>

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE     SELECTOR
service/httpbin1     ClusterIP   10.96.184.144   <none>        80/TCP    3m12s   app=httpbin1
service/httpbin2     ClusterIP   10.96.113.237   <none>        80/TCP    3m8s    app=httpbin2
service/kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP   34m     <none>
```

Kubernetes 中的 Service

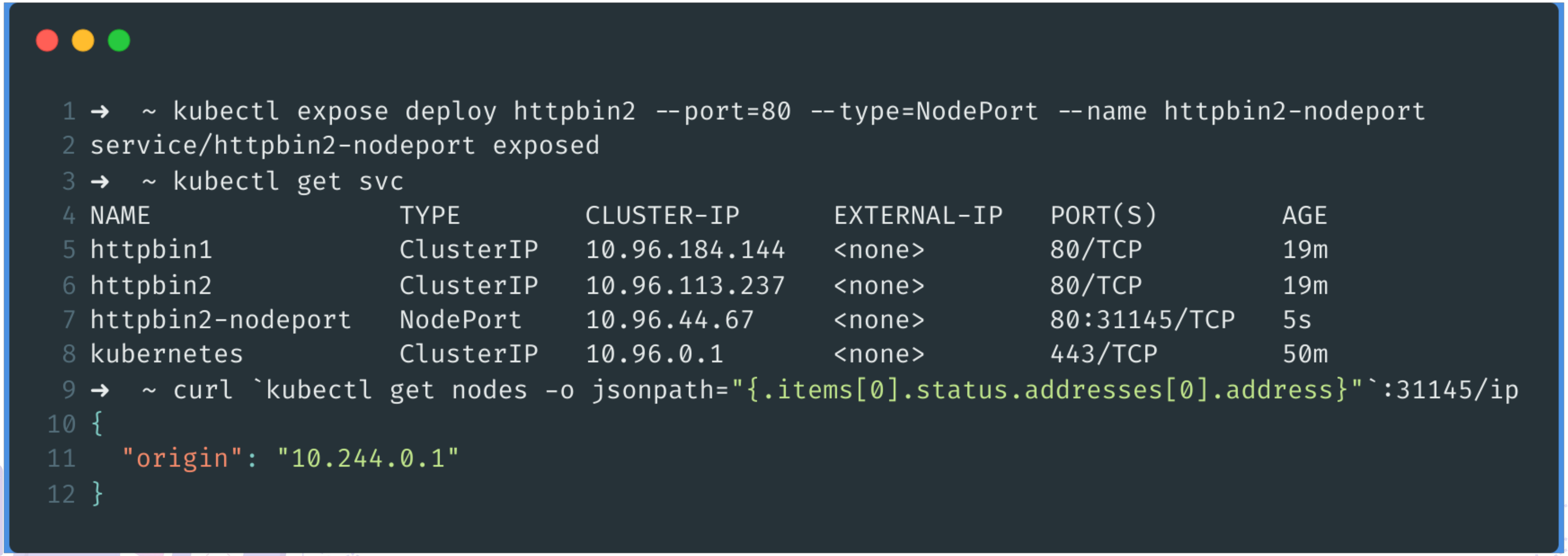


```
➜  ~ kubectl exec -it httpbin1-56f8c5cf48-hcztx -- bash
root@httpbin1-56f8c5cf48-hcztx:/# curl httpbin2/ip
{
  "origin": "10.244.0.8"
}

root@httpbin1-56f8c5cf48-hcztx:/# curl 10.244.0.9/ip
{
  "origin": "10.244.0.8"
}

```

Kubernetes 中的 Service

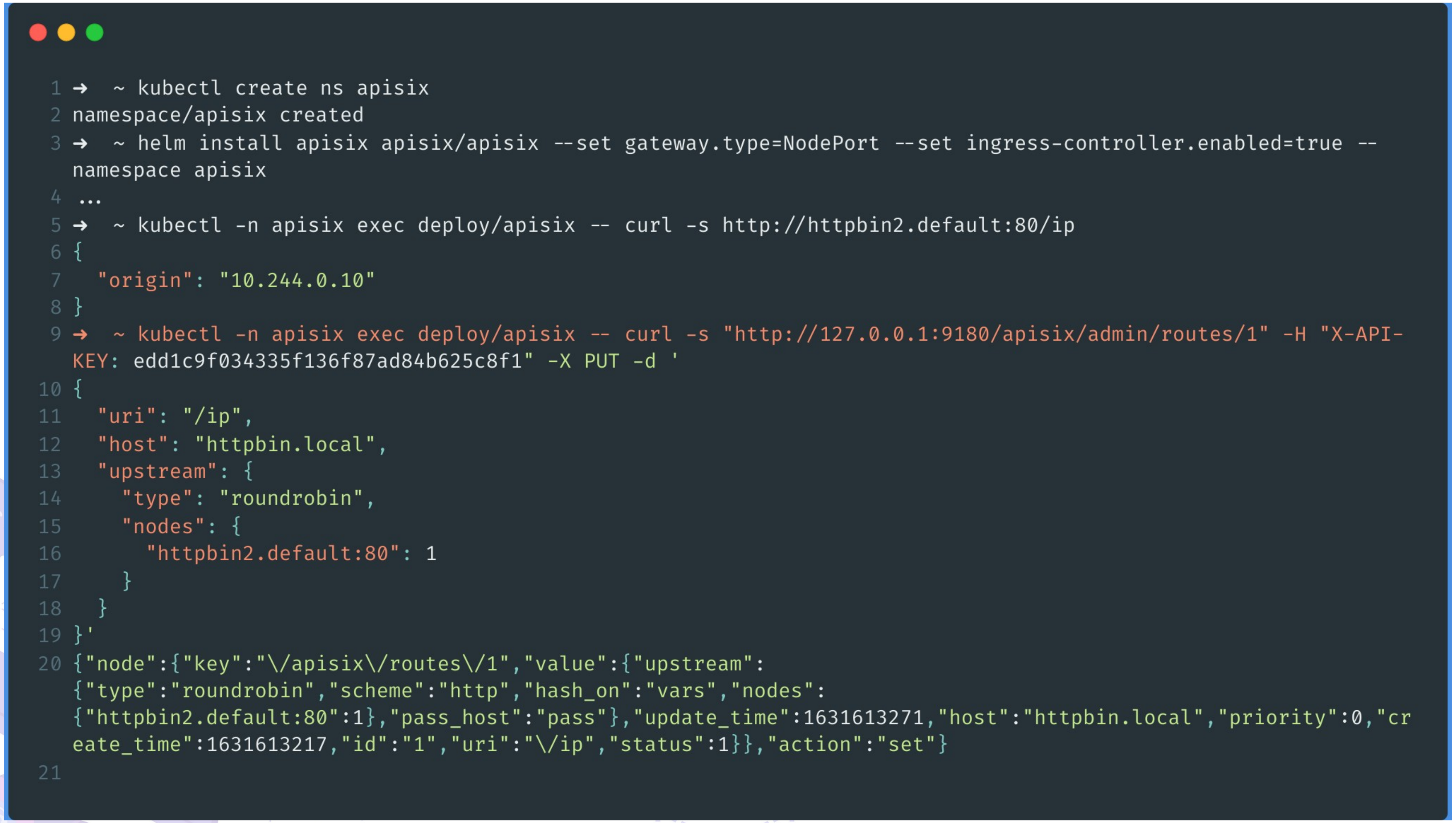


使用 NodePort 在集群外访问服务



```
➜  ~ kubectl expose deploy httpbin2 --port=80 --type=NodePort --name httpbin2-nodeport
service/httpbin2-nodeport exposed
➜  ~ kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
httpbin1            ClusterIP   10.96.184.144   <none>        80/TCP          19m
httpbin2            ClusterIP   10.96.113.237   <none>        80/TCP          19m
httpbin2-nodeport   NodePort    10.96.44.67     <none>        80:31145/TCP    5s
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP         50m
➜  ~ curl `kubectl get nodes -o jsonpath="{.items[0].status.addresses[0].address}"`:31145/ip
{
  "origin": "10.244.0.1"
}
```

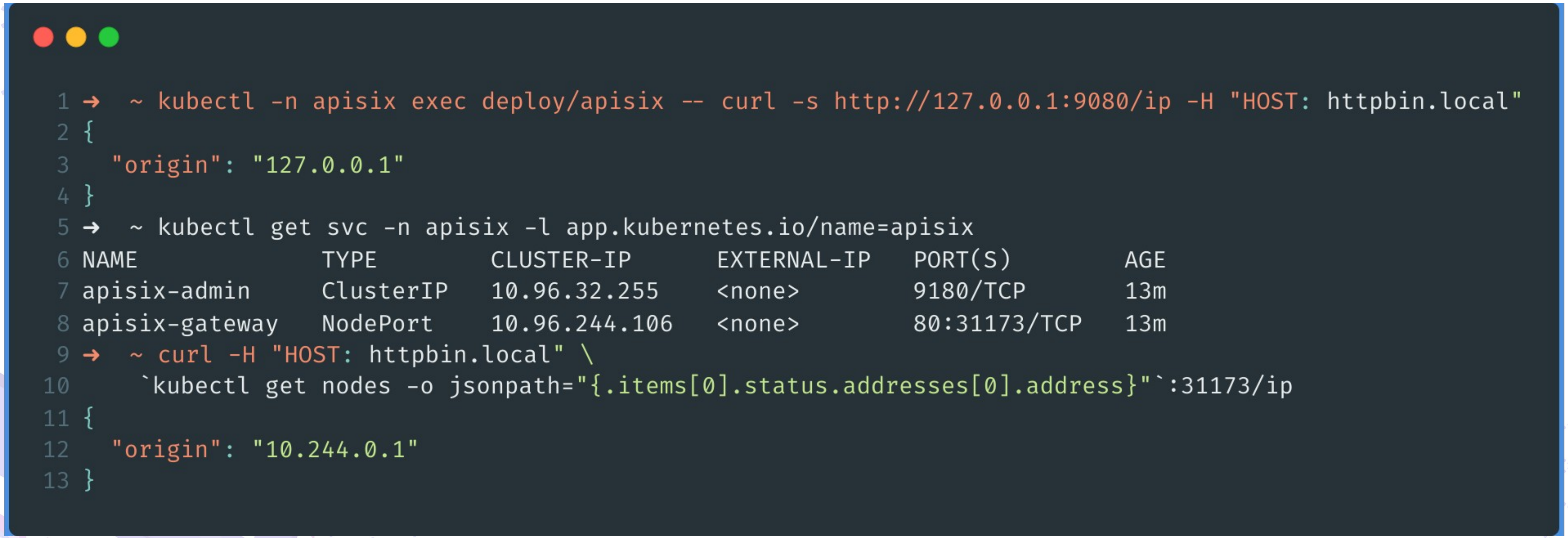
使用 Gateway 在集群外访问服务



```
➜  ~ kubectl create ns apisix
namespace/apisix created
➜  ~ helm install apisix apisix/apisix --set gateway.type=NodePort --set ingress-controller.enabled=true --namespace apisix
...
➜  ~ kubectl -n apisix exec deploy/apisix -- curl -s http://httpbin2.default:80/ip
{
  "origin": "10.244.0.10"
}
➜  ~ kubectl -n apisix exec deploy/apisix -- curl -s "http://127.0.0.1:9180/apisix/admin/routes/1" -H "X-API-KEY: edd1c9f034335f136f87ad84b625c8f1" -X PUT -d '
{
  "uri": "/ip",
  "host": "httpbin.local",
  "upstream": {
    "type": "roundrobin",
    "nodes": {
      "httpbin2.default:80": 1
    }
  }
}'
{"node":{"key":"\/apisix\/routes\/1","value":{"upstream":{"type":"roundrobin","scheme":"http","hash_on":"vars","nodes":{"httpbin2.default:80":1},"pass_host":"pass"},"update_time":1631613271,"host":"httpbin.local","priority":0,"create_time":1631613217,"id":"1","uri":"\/ip","status":1}},"action":"set"}

```

https://apisix.apache.org/

使用 Gateway 在集群外访问服务



```
➜  ~ kubectl -n apisix exec deploy/apisix -- curl -s http://127.0.0.1:9080/ip -H "HOST: httpbin.local"
{
  "origin": "127.0.0.1"
}
➜  ~ kubectl get svc -n apisix -l app.kubernetes.io/name=apisix
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
apisix-admin      ClusterIP   10.96.32.255    <none>        9180/TCP       13m
apisix-gateway    NodePort    10.96.244.106   <none>        80:31173/TCP   13m
➜  ~ curl -H "HOST: httpbin.local" \
     `kubectl get nodes -o jsonpath="{.items[0].status.addresses[0].address}"`:31173/ip
{
  "origin": "10.244.0.1"
}
```

总结

- Kubernetes 整体架构
- Kubernetes 核心概念
- Kubernetes 中的应用部署
- Kubernetes 中的流量路由

链接

- KIND: https://kind.sigs.k8s.io/
- Apache APISIX: https://apisix.apache.org/
- Ingress-nginx: https://github.com/kubernetes/ingress-nginx