[1. Service](https://note.youdao.com/ynoteshare1/iframe.html#9885-1576299905919)

[1.1 创建ClusterIP](https://note.youdao.com/ynoteshare1/iframe.html#5080-1576299911497)

[1.2 查看ClusterIP](https://note.youdao.com/ynoteshare1/iframe.html#8720-1576299943106)

[1.3 创建NodePort](https://note.youdao.com/ynoteshare1/iframe.html#1879-1576300240833)

[1.4 查看NodePort资源](https://note.youdao.com/ynoteshare1/iframe.html#2477-1576300361460)

[1.5 External IPs](https://note.youdao.com/ynoteshare1/iframe.html#4229-1576300642118)

[2. 服务发现](https://note.youdao.com/ynoteshare1/iframe.html#3762-1576326733539)

[2.1 Headless Services](https://note.youdao.com/ynoteshare1/iframe.html#9873-1583746223569)

[3. ExternalName](https://note.youdao.com/ynoteshare1/iframe.html#3989-1583746223569)

[4. Ingress](https://note.youdao.com/ynoteshare1/iframe.html#5574-1576326913329)

[4.1 部署Ingress Controller](https://note.youdao.com/ynoteshare1/iframe.html#1872-1576326956005)

[4.2 创建Ingress](https://note.youdao.com/ynoteshare1/iframe.html#8954-1576326952076)

[5. 会话保持](https://note.youdao.com/ynoteshare1/iframe.html#9097-1576382204752)

**1. Service**

**1.1 创建ClusterIP**

<https://kubernetes.io/zh/docs/concepts/services-networking/service/>

# 方法一：通过命令行创建

kubectl expose deployment <nginx-name> --target-port=80 --port=80 --type=ClusterIP

#方法二：通过yaml文件创建

cat << EOF > clusterip.yaml

kind: Service

apiVersion: v1

metadata:

name: my-service

spec:

type: ClusterIP

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80 #pod的端口

EOF

cat << EOF > endpoint.yaml

apiVersion: v1

kind: Endpoints

metadata:

labels:

app: nginx-deploy

name: my-service #名字保持一样

subsets:

- addresses:

- ip: 10.42.4.126

- ip: 10.42.3.39

- ip: 10.42.4.127

ports:

- port: 80

protocol: TCP

EOF

kubectl create -f clusterip.yaml

**1.2 查看ClusterIP**

curl clusterip:80 可访问

**1.3 创建NodePort**

# 方法一：通过命令行创建

kubectl expose deployment <nginx-name> --target-port=80 --port=80 --type=NodePort

#方法二：通过yaml文件创建

cat << EOF > nodeport.yaml

kind: Service

apiVersion: v1

metadata:

name: my-service

spec:

type: NodePort

ports:

- protocol: TCP

port: 80

targetPort: 80

#可通过设置nodePort指定对外暴露的端口

EOF

kubectl create -f nodeport.yaml

**1.4 查看NodePort资源**

curl node01:32476 可访问

**1.5 External IPs**

apiVersion: v1 kind: Service metadata: name: my-service spec: selector: app: nginx ports: - name: http protocol: TCP port: 18080 targetPort: 9376 externalIPs: - 172.31.53.96

通过kubeproxy,只需暴露一个IP

访问外部：

1。通过external name

2。

clusterIP:none

EP没有selector

clusterIP和EP通过name关联

通过目标端的外网IP访问

kubectl run busybox --image=busybox:1.28 --command sleep 3600

kubectl exec busybox -- nslookup my-externalname

**2. 服务发现**

apiVersion: v1

kind: Pod

metadata:

name: busybox1

labels:

name: busybox

spec:

hostname: busybox-1

subdomain: default-subdomain

containers:

- image: busybox:1.28

command:

- sleep

- "3600"

name: busybox

apiVersion: v1 kind: Pod metadata: name: hostaliases-pod spec: restartPolicy: Never hostAliases: - ip: "127.0.0.1" hostnames: - "foo.local" - "bar.local" - ip: "10.1.2.3" hostnames: - "foo.remote" - "bar.remote" containers: - name: cat-hosts image: busybox command: - cat args: - "/etc/hosts"

**2.1 Headless Services**

apiVersion: v1

kind: Service

metadata:

name: nginx

labels:

app: nginx

spec:

ports:

- port: 80

name: web

clusterIP: None

selector:

app: nginx

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: web

spec:

selector:

matchLabels:

app: nginx # has to match .spec.template.metadata.labels

serviceName: "nginx"

replicas: 3 # by default is 1

template:

metadata:

labels:

app: nginx # has to match .spec.selector.matchLabels

spec:

terminationGracePeriodSeconds: 10

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

name: web

**3. ExternalName**

apiVersion: v1 kind: Service metadata: name: my-service namespace: prod spec: type: ExternalName externalName: my.database.example.com

**4. Ingress**

**4.1 部署Ingress Controller**

git clone https://github.com/kubernetes/ingress-nginx.git

cd /root/ingress-nginx/deploy/static

vim mandatory.yaml

在containers:上面添加

hostNetwork: true

nodeSelector:

app: ingress-controller

kubectl label nodes node01 app=ingress-controller

kubectl apply -f mandatory.yaml

kubectl get namespaces

kubectl -n ingress-nginx get pod

**4.2 创建Ingress**

kubectl run --image=nginx test

kubectl run --image=nginx test2

kubectl expose deployment test --port=18080 --target-port=80

kubectl expose deployment test2 --port=18080 --target-port=80

创建

kubectl create deployment test --image=nginx

kubectl exec -it podname bash

创建service

kubectl expose deployment test2 --port=18080 --target-port=80

编写ingress yaml文件

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: simple-fanout-example

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- host: foo.bar.com

http:

paths:

- path: /foo

backend:

serviceName: service1

servicePort: 18080

- path: /bar

backend:

serviceName: service2

servicePort: 18080

hosts表添加node01的ip地址+域名

vim /etc/hosts

172.31.53.73 node01 foo.bar.com

**5. 会话保持**

创建Service编辑sessionAffinity

kubectl edit service/test

将sessionaffinity参数改为 sessionAffinity: ClientIP，保存 在此查看ipvs规则

root@wan-node1:~# ipvsadm -ln -> RemoteAddress:Port Forward Weight ActiveConn InActConn TCP 192.168.250.200:31389 rr persistent 10800 -> 10.244.0.21:80 Masq 1 0 0 -> 10.244.1.7:80 Masq 1 0 0 -> 10.244.1.8:80 Masq 1 0 0