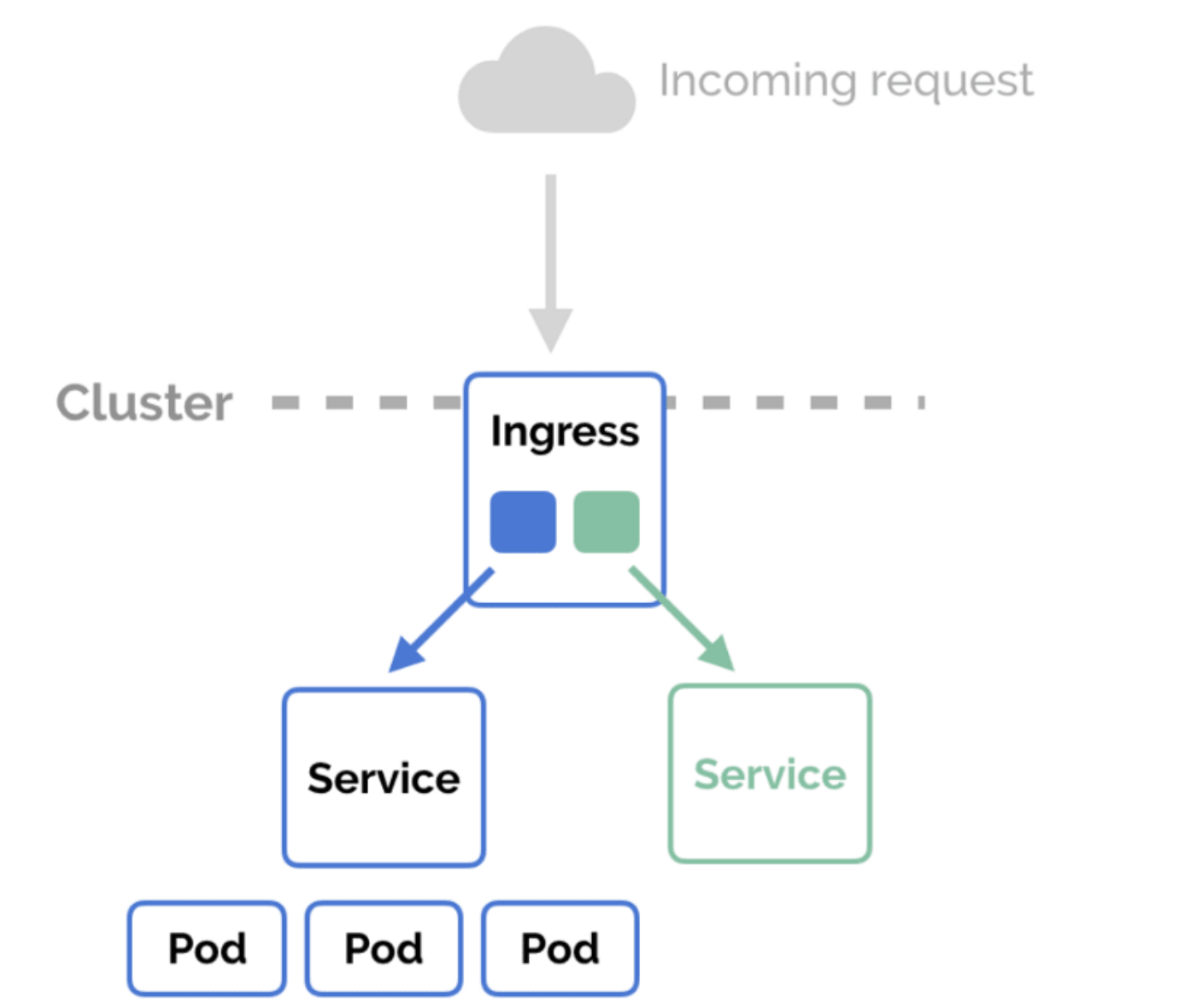


Ingress 与 Service

Service 是 Kubernetes 中用于暴露 Pod 的 API。Ingress 是 Kubernetes 中用于暴露 Service 的 API。Ingress 可以配置为通过 NodePort 或 LoadBalancer 暴露 Service，NodePort 暴露 Service 的端口，LoadBalancer 暴露 Service 的端口。



YAML:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
```

```
name: ingress-example
spec:
  rules:
    - host: example.com
      http:
        paths:
          - pathType: Prefix
            path: /
            backend:
              name: hostname-service
              port: 80
```

```
host:  path: .
```

[illegible]

我们使用 Nginx 作为反向代理服务器，用于处理静态资源请求。Nginx 配置如下：

Nginx Documentation - Installing Nginx Ingress Controller

YAML is a human-readable data serialization format.


```
git clone https://github.com/nginxinc/kubernetes-ingress.git --branch v3.1.1
```

```
# 编译并安装二进制文件
```

```
cd kubernetes-ingress/deployments
```

```
# kubectl apply -f ingress.yaml
```

```
cd kubernetes-ingress/deployments
```

□□ □□□□ □□ □□□□ □□ Nginx □□□□ □□□□ □□/□□ □□ □□

```
# Namespace
kubectl apply -f common/ns-and-sa.yaml
```

```
# [[ ]] cluster role [[ ]] cluster role binding [[ ]]
kubectl apply -f rbac/rbac.yaml
```

```
# Nginx configMap
kubectl apply -f common/nginx-config.yaml

# IngressClass (Nginx)
kubectl apply -f common/ingress-class.yaml

# CRDs
kubectl apply -f common/crds/k8s.nginx.org_virtualservers.yaml
kubectl apply -f common/crds/k8s.nginx.org_virtualserverroutes.yaml
kubectl apply -f common/crds/k8s.nginx.org_transportservers.yaml
kubectl apply -f common/crds/k8s.nginx.org_policies.yaml

## VirtualServer, VirtualServerRoute, TransportServer Policy
##

# TCP/UDP
kubectl apply -f common/crds/k8s.nginx.org_globalconfigurations.yaml
```

部署 Ingress 控制器

```
# 部署 1 replica
kubectl apply -f deployment/nginx-ingress.yaml

# LoadBalancer
kubectl apply -f service/loadbalancer.yaml

## NodePort
kubectl create -f service/nodeport.yaml
```

验证部署

```
kubectl get all -n nginx-ingress
```

NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-ingress-bcd99bfb9-w6kn7	1/1	Running	0	152m

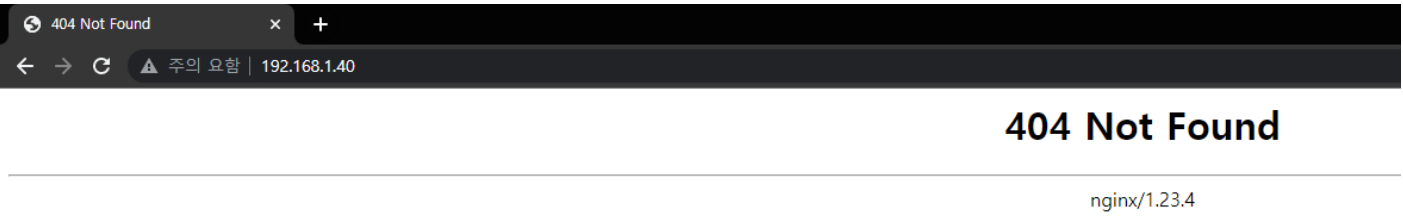
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
------	------	------------	-------------	---------	-----

service/nginx-ingress LoadBalancer 10.233.3.51 192.168.1.40 80:30276/TCP,443:30815/TCP 150m

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx-ingress	1/1	1	1	152m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/nginx-ingress-bcd99bfb9	1	1	1	152m

우리는 Ingress 컨트롤러를 배포하고, External-IP를 주어진 IP로 설정하여 외부에서 접근할 수 있도록 합니다.



우리는 Ingress 컨트롤러를 배포하고, External-IP를 주어진 IP로 설정하여 외부에서 접근할 수 있도록 합니다.

우리는 Ingress 컨트롤러를 배포하고, External-IP를 주어진 IP로 설정하여 외부에서 접근할 수 있도록 합니다. Ingress 컨트롤러는 HTTP, HTTPS, SSL, TLS 등을 처리할 수 있습니다. Ingress 컨트롤러는 Kubernetes 클러스터 내에서 배포된 애플리케이션을 외부에서 접근할 수 있도록 합니다. Ingress 컨트롤러는 Kubernetes 클러스터 내에서 배포된 애플리케이션을 외부에서 접근할 수 있도록 합니다. Ingress 컨트롤러는 Kubernetes 클러스터 내에서 배포된 애플리케이션을 외부에서 접근할 수 있도록 합니다.

```
# 1. my-app.yaml을 생성합니다
vi my-app.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
```

```
spec:
  containers:
    - name: my-app
      image: waji97/ecommerce:v2
      ports:
        - containerPort: 8000
```

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
```

```
spec:
  type: ClusterIP
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

:wq

2个空行

vi my-app2.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
```

```
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
```

```
template:
  metadata:
    labels:
      app: my-app
```

```
spec:
  containers:
```

```
- name: my-app
  image: waji97/django:v2
  ports:
    - containerPort: 8000
```

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  type: ClusterIP
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

:wq

📁 📁 📁 📁 Deployment 📁 📁 📁

```
# 1📁 📁 📁 📁
vi my-app.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
```

```
spec:
  containers:
    - name: my-app
      image: waji97/ecommerce:v2
      ports:
        - containerPort: 8000
```

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  type: ClusterIP
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

:wq

2个pod

vi my-app2.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app2
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-app2
  template:
    metadata:
      labels:
        app: my-app2
    spec:
      containers:
```

```
- name: my-app2
  image: waji97/django:v2
  ports:
    - containerPort: 8000
```

```
apiVersion: v1
kind: Service
metadata:
  name: my-app2-service
spec:
  type: ClusterIP
  selector:
    app: my-app2
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

:wq

保存

```
kubectl apply -f my-app.yaml
deployment.apps/my-app created
service/my-app-service created
```

```
kubectl apply -f my-app2.yaml
deployment.apps/my-app2 created
service/my-app2-service created
```

確認 Deployment

NAME	READY	STATUS	RESTARTS	AGE
pod/my-app-7d7b9b8d94-92qwc	1/1	Running	0	3m48s
pod/my-app-7d7b9b8d94-xhlvd	1/1	Running	0	3m45s
pod/my-app2-764587cdfc-fc7df	1/1	Running	0	3m54s
pod/my-app2-764587cdfc-l9h85	1/1	Running	0	3m54s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.233.0.1	<none>	443/TCP	2d
service/my-app-service	ClusterIP	10.233.46.112	<none>	80/TCP	169m

service/my-app2-service ClusterIP 10.233.27.197 <none> 80/TCP 3m54s

📄 Ingress 📄 📄 📄 📄

YAML 📄 📄 📄 📄

vi ingress.yaml

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
  name: my-ingress
```

```
  annotations:
```

```
    kubernetes.io/ingress.class: "nginx"
```

```
    kubernetes.io/ssl-redirect: "false"
```

```
spec:
```

```
  rules:
```

```
  - host: myapp.ingtest.com
```

```
    http:
```

```
      paths:
```

```
      - pathType: Prefix
```

```
        path: /📄📄📄📄📄📄📄
```

```
        backend:
```

```
          service:
```

```
            name: my-app-service
```

```
            port:
```

```
              number: 80
```

```
  - host: myapp2.ingtest.com
```

```
    http:
```

```
      paths:
```

```
      - pathType: Prefix
```

```
        path: /
```

```
        backend:
```

```
          service:
```

```
            name: my-app2-service
```

```
            port:
```

```
              number: 80
```

:wq

```
# 验证安装
```

```
kubectl apply -f ingres.yaml
```

```
ingress.networking.k8s.io/my-ingress created
```

```
# 查看安装结果
```

```
kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
my-ingress	<none>	myapp.ingtest.com,myapp2.ingtest.com		80	169m

我们使用 `curl` 命令来验证 Ingress 是否正常工作，并检查 DNS 解析是否正确。

我们可以在 PC 的 `'hosts'` 文件中添加 IP 到 DNS 解析。我们可以在 `hosts` 文件中添加，`myapp.ingtest.com` 和 `myapp2.ingtest.com` 指向 IP `192.168.1.40`。DNS 解析是否正确。

Windows 的 `'hosts'` 文件

`C:\Windows\System32\drivers\etc\hosts`

Linux 的 `'hosts'` 文件

`/etc/hosts`

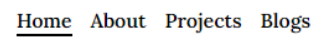
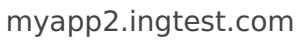
`'hosts'` 文件 IP 地址

```
.
.
192.168.90.73 host.docker.internal
192.168.90.73 gateway.docker.internal

# 添加 External-IP 地址
192.168.1.40 myapp.ingtest.com
192.168.1.40 myapp2.ingtest.com
```

验证结果：

`myapp.ingtest.com`



```
//SSL 配置 在 /myapp /myapp2 目录下
```

Updated 2 June 2023 06:14:42 by