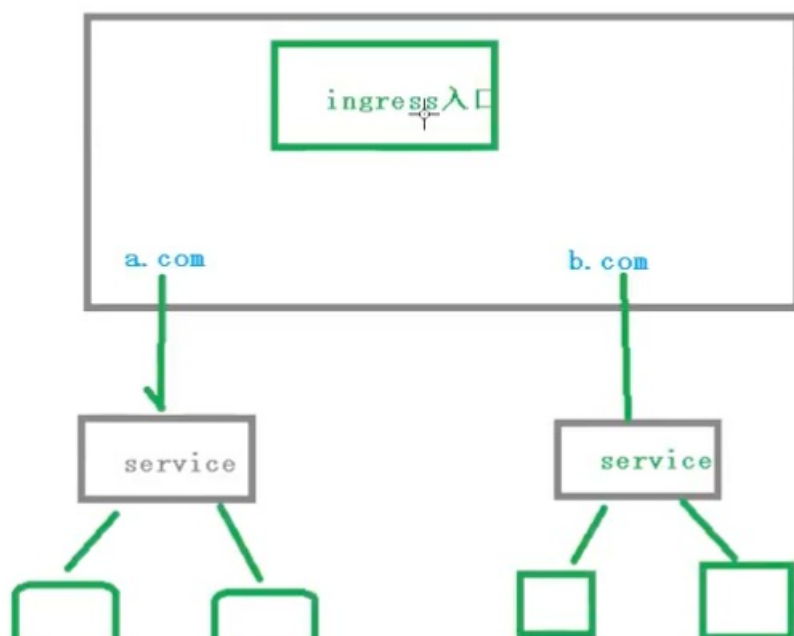


# Ingress 概述



## Ingress

### 1. 將端口號對外暴露，通過 IP + 端口號進行訪問

- 使用 Service 裡面的 NodePort 實現

### 2. NodePort 缺陷

- 在每個節點上都會起到端口，在訪問時可以通過任何節點，通過節點 IP + 暴露端口號實現訪問。
- 意味著每個端口只能使用一次，一個端口對應一個應用。
- 實際訪問中都是用域名，根據不同域名跳轉到不同端口服務中。

### 3. Ingress 和 Pod 的關係

- Pod 和 Ingress 是通過 Service 關聯的。
- Ingress 作為統一入口，由 Service 關聯一組 Pod。

### 說明：

- Ingress 提供基於域名的訪問路由能力，實現集群內外通信的靈活控制。
- 它比 NodePort 更靈活且節約端口資源，是 Kubernetes 中常用的負載均衡工具之一。

以下是圖片內容的文字辨識及翻譯成繁體中文：

## 5. 使用 Ingress

- **第一步：** 部署 Ingress Controller
- **第二步：** 創建 Ingress 規則

我們在這裡選擇官方維護的 NGINX 控制器，實現部署。

以下是圖片中的文字辨識及翻譯成繁體中文：

## 6. 使用 Ingress 對外暴露應用

### (1) 創建 Nginx 應用，對外暴露端口使用 NodePort

```
kubectl create deployment web --image=nginx
```

```
kubectl expose deployment web --port=80 --target-port=80 --type=NodePort
```

### (2) 部署 Ingress Controller

```
[root@k8smaster ~]# vi ingress-con.yaml
[root@k8smaster ~]# kubectl apply -f ingress-con.yaml
namespace/nginx-ingress created
configmap/nginx-configuration created
configmap/tcp-services created
configmap/udp-services created
serviceaccount/nginx-ingress-serviceaccount created
clusterrole.rbac.authorization.k8s.io/nginx-ingress-clusterrole created
role.rbac.authorization.k8s.io/nginx-ingress-role created
rolebinding.rbac.authorization.k8s.io/nginx-ingress-role-nisa-binding created
```

### 說明：

1. **NodePort 暴露應用：**
  - 通過 `NodePort` 將應用對外暴露，允許外部訪問特定端口。
2. **Ingress Controller 部署：**
  - 使用配置文件 `ingress-con.yaml` 部署 Ingress 控制器，啟用基於域名的路由功能。

🔄 修訂版本 #1

★ 由 treeman 建立於 20 🕒 2025 11:12:54

✍ 由 treeman 更新於 20 🕒 2025 11:47:25