概念

|  |
| --- |
| **🧩 Ingress 概念**  Ingress 是 Kubernetes 集群中的一种资源对象，用于**对外暴露 HTTP/HTTPS 服务**，通过定义域名、路径规则，把外部请求转发到集群内部的 Service。  相当于“七层（HTTP）负载均衡 + 路由规则”。 |

示例

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ingress资源对象定义模板：          单域名多路径示例  定义deployment以及相应的service对象：  nodeport的service  应用：    对外可以访问：    创建第二个deployment和相应service：  nodeport的service  访问：        创建ingress：    这里应该是写错了：第二个路径不能和第一个重复：第一个应该是/api/  第二个应该是/web  rewrite-target: /$1 重写规则  表示当请求路径被匹配后，转发到后端时，会把匹配到的路径重写为 /$1。 比如请求 /api/user 被匹配规则 /api/(.\*)，就会转发到 /user。  spec:  ingressClassName: nginx 指定控制器处理该 Ingress规则  rules:  - host: www.aaa.com  http:  paths: 定义域名和路由规则  第一条路径规则  - path: /api/  pathType: Prefix  backend:  service:  name: api-service  port:  number: 80  访问的时候访问的是：  要你访问的是 **Ingress 定义的域名（Host）** + **Ingress Controller Service 暴露的端口**， 请求就会进入 **Ingress Controller** → 根据规则匹配 → 转发到对应的 **后端 Service / Pod**    到这里老师做的应该都是错的：  看ingress：    kubectl get ingress  输出示例：  NAME CLASS HOSTS ADDRESS（公网ip） PORTS AGE  ingress-test nginx www.aaa.com 80 10m   * ADDRESS 显示的是 **Ingress 对外访问入口的 IP 或域名** * 它来自 **Ingress Controller 的 Service**，具体依赖 Service 类型：   + **LoadBalancer** → EXTERNAL-IP   + **NodePort / ClusterIP** → 如果没有外部负载均衡，ADDRESS 通常为空   **Ingress 的 ADDRESS 字段来源**   | **Ingress Controller Service 类型** | **来源** | **说明** | | --- | --- | --- | | **LoadBalancer** | Service 的 EXTERNAL-IP | 云平台分配的公网或私网负载均衡 IP，会显示在 ADDRESS | | **NodePort / ClusterIP** | Service 的 EXTERNAL-IP（通常是 <none>） | 如果没有外部 IP，ADDRESS 就为空，需要通过 NodeIP:NodePort 或 /etc/hosts 配置访问 |   所以说，ADDRESS 本质上就是 **Ingress Controller 对外入口的 IP 地址**，显示为空只是因为 Service 没有外部 IP 或还未分配。  访问ingress的方式：  ngress Controller 的 Service 类型是 **NodePort** 时  http://<NodeIP>:30080/ 30080是ingress-controller的开放端口  Ingress Controller 是一个独立部署的组件（工具），基于官方或社区镜像，不是 Kubernetes 内置，也不需要你自己手写，只需安装运行即可使用 Ingress 规则。  官方或 Helm 部署的 Ingress Controller 一般自带 Service，无需手动创建；自己部署 Controller 时必须手动创建 Service。  **Ingress 也是 Namespaced**   * Ingress 资源必须指定 namespace，属于某个命名空间：   metadata:  name: api-ingress  namespace: default   * Ingress 只能引用 **同一命名空间内的 Service**   + 也就是说，default 命名空间的 Ingress 不能直接路由到 test 命名空间的 Service   **2️⃣ 命名空间的作用**   * **隔离路由和服务**：   1. 每个命名空间都有自己的 Service 和 Ingress 范围   2. Ingress Controller 会根据 Ingress 资源匹配请求   3. 因为 Ingress 只能引用本命名空间的 Service，所以路由天然被隔离 * **跨命名空间访问**：   1. 默认不允许   2. 如果真的要访问其他命名空间的 Service，需要特殊配置（不常用）   Kubernetes 资源的 **命名空间归属**，分成 **Namespaced** 和 **Cluster级（跨命名空间）** 两类。  **1️⃣ Namespaced 资源（属于某个命名空间）**   * **工作负载**：   + Pod   + Deployment   + ReplicaSet   + StatefulSet   + DaemonSet   + Job / CronJob * **服务与网络**：   + Service   + Ingress   + NetworkPolicy   + Endpoints * **配置类**：   + ConfigMap   + Secret   + PersistentVolumeClaim (PVC) * **RBAC（部分）**：   + Role   + RoleBinding   **特点**：   * 名称在命名空间内唯一 * 跨命名空间默认无法访问或引用 * 可用命名空间隔离资源、路由、权限   **2️⃣ Cluster级资源（不属于任何命名空间，跨命名空间可用）**   * **节点和存储**：   + Node   + PersistentVolume (PV)   + StorageClass * **全局网络和策略**：   + ClusterRole   + ClusterRoleBinding   + CustomResourceDefinition (CRD)   + MutatingWebhookConfiguration / ValidatingWebhookConfiguration   + Namespace（自身就是资源） * **工作负载管理**：   + (部分) PodSecurityPolicy / PodPreset / Admission控制器相关资源   **特点**：   * 全局唯一，不依赖命名空间 * 可以被整个集群访问或管理 |

https配置

|  |
| --- |
|  |

域名路由

|  |
| --- |
|  |

配置ingress controller公网ip

|  |
| --- |
|  |