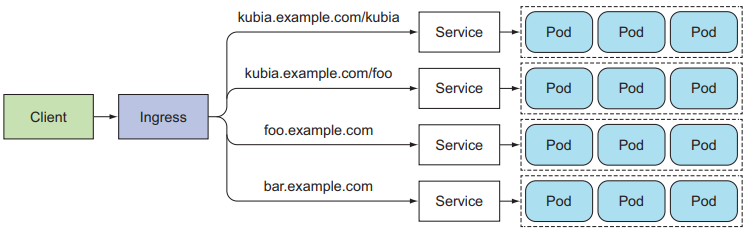
28.Ingress

上一节我们学习了将服务暴露给集群外部客户端的两种方式（NodePort和LoadBalancer），本节我们来学习另外一种方式——Ingress。

为什么需要Ingress

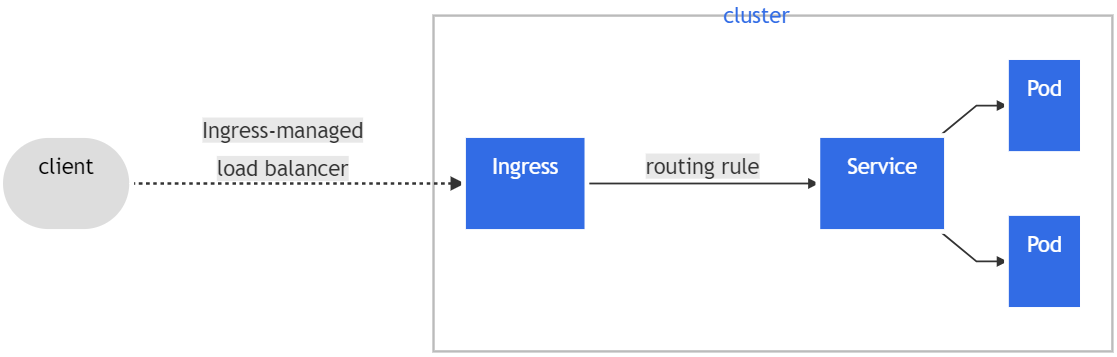
一个很重要的原因是每个LoadBalancer服务都需要属于它自己的负载均衡器以及独立的公共IP地址，而Ingress只需要一个公网IP就能为许多服务提供访问。当客户端发送HTTP请求到Ingress时，Ingress会根据请求中的主机名和路径决定将请求转发到哪个服务，如下图所示：



通过一个Ingress就可以暴露多个服务。
Ingress在网络堆栈的应用层上工作，可以提供诸如基于cookie的会话绑定等service无法提供的特性。

什么是Ingress

Ingress是Kubernetes中的一种资源对象，它使我们能够为运行在Kubernetes集群中的应用程序配置一个HTTP负载均衡器，该应用程序由一个或多个服务表示。为了将这些应用程序交付给Kubernetes集群之外的客户端，这种负载均衡器是必要的。Ingress公开从集群外部到集群内服务的HTTP和HTTPS路由。流量路由由Ingresss资源上定义的规则控制。
下面是一个简单的例子，Ingress将其所有的流量发送到一个服务:



Ingress资源支持如下特性：

- 基于内容的路由
  - 基于主机的路由。例如，将带有主机报头foo.example.com的请求路由到一组服务，带有主机报头bar.example.com的请求路由到另一组服务
  - 基于路径的路由。例如，将以/service-a开头的URI请求路由到服务a，以/service-b开头的URI请求路由到服务b。
- 针对每个主机名的TLS/SSL terminiation代理，比如foo.example.com

什么是Ingress控制器

Ingress控制器一个运行在集群中的应用，它会根据Ingress资源配置一个HTTP负载均衡器。该负载均衡器可以是一个运行在集群中的软件负载均衡器，也可以是一个在集群外部运行的硬件或者云负载均衡器。不同的负载均衡器需要不同的Ingress控制器实现方式。

如果想要Ingress能正常工作，首先必须确保在集群中运行Ingress控制器。不同的Kubernetes环境使用不同的控制器实现方式，但有些根本不提供默认的控制器。

例如，GKE（Google Kubernetes Engine）使用Google Cloud Platform自己的HTTP负载均衡特性来提供Ingress功能。对于Minikube来说，最初并没有提供开箱即用的控制器，但现在它包含一个可以启用的插件便于Ingress的功能。

本文我们使用Nginx Ingress控制器来演示如何使用Ingress控制器。

安装Nginx Ingress控制器

在Master节点上下载deploy.yaml文件：

wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.41.2/deploy/static/provider/baremetal/deploy.yaml

```
[root@k8s-master test]# wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.41.2/deploy/static/provider/baremetal/deploy.yaml
--2021-01-07 02:47:14--  https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.41.2/deploy/static/provider/baremetal/deploy.yaml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 199.232.96.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|199.232.96.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 18331 (18K) [text/plain]
Saving to: 'deploy.yaml'

100%[======================================================================================================================>] 18,331      --.-K/s   in 0.007s

2021-01-07 02:47:15 (2.39 MB/s) - 'deploy.yaml' saved [18331/18331]
```

之所以先下载这个文件，而不是直接通过kubectl apply –f 命令执行，是因为国内无法访问该文件中指定的k8s.gcr.io下的镜像。所以将文件下载下来后，还需要使用我们事先准备好的阿里云镜像替换该镜像：

```
---
# Source: ingress-nginx/templates/controller-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    helm.sh/chart: ingress-nginx-3.10.1
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/version: 0.41.2
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/component: controller
  name: ingress-nginx-controller
  namespace: ingress-nginx
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: ingress-nginx
      app.kubernetes.io/instance: ingress-nginx
      app.kubernetes.io/component: controller
  revisionHistoryLimit: 10
  minReadySeconds: 0
  template:
    metadata:
      labels:
        app.kubernetes.io/name: ingress-nginx
        app.kubernetes.io/instance: ingress-nginx
        app.kubernetes.io/component: controller
    spec:
      dnsPolicy: ClusterFirst
      containers:
        - name: controller
          image: k8s.gcr.io/ingress-nginx/controller:v0.41.2@sha256:1f4f402b9c14f3ae92b11ada1dfe9893a88f0faeb0b2f4b903e2c67a0c3bf0de
          imagePullPolicy: IfNotPresent
          lifecycle:
            preStop:
              exec:
                command:
                  - /wait-shutdown
```

sed –i 's#k8s.gcr.io/ingress–nginx/controller:v0.41.2@sha256:1f4f402b9c14f3ae92b11ada1dfe9893a88f0faeb0b2f4b903e2c67a0c3bf0de#registry.cn–shanghai.aliyuncs.com/pmx/ingress–nginx:v0.41.2@sha256:8aa4fda472ec83ae59fe0ce9720684d769ed277ff9bdcbb0169178dc9d1f8e85#g' deploy.yaml

```
[root@k8s-master test]# sed -i 's#k8s.gcr.io/ingress-nginx/controller:v0.41.2@sha256:1f4f402b9c14f3ae92b11ada1dfe9893a88f0faeb0b2f4b903e2c67a0c3bf0de#registry.cn-sha
nghai.aliyuncs.com/pmx/ingress-nginx:v0.41.2@sha256:8aa4fda472ec83ae59fe0ce9720684d769ed277ff9bdcbb0169178dc9d1f8e85#g' deploy.yaml
[root@k8s-master test]#
```

```yaml
# Source: ingress-nginx/templates/controller-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    helm.sh/chart: ingress-nginx-3.10.1
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/version: 0.41.2
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/component: controller
  name: ingress-nginx-controller
  namespace: ingress-nginx
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: ingress-nginx
      app.kubernetes.io/instance: ingress-nginx
      app.kubernetes.io/component: controller
  revisionHistoryLimit: 10
  minReadySeconds: 0
  template:
    metadata:
      labels:
        app.kubernetes.io/name: ingress-nginx
        app.kubernetes.io/instance: ingress-nginx
        app.kubernetes.io/component: controller
    spec:
      dnsPolicy: ClusterFirst
      containers:
        - name: controller
          image: registry.cn-shanghai.aliyuncs.com/pmx/ingress-nginx:v0.41.2@sha256:8aa4fda472ec83ae59fe0ce9720684d769ed277ff9bdcbb0169178dc9d1f8e85
          imagePullPolicy: IfNotPresent
          lifecycle:
            preStop:
              exec:
                command:
                  - /wait-shutdown
          args:
```

另外，对外开放的时候，需要固定Node上的端口，如果不想Kubernetes帮我们随机生成，可以修改该文件。找到Source: ingress-nginx/templates/controller-service.yaml，在ports上添加nodePort，设置http端口为32080，https端口为32443：

```yaml
# Source: ingress-nginx/templates/controller-service.yaml
apiVersion: v1
kind: Service
metadata:
  annotations:
  labels:
    helm.sh/chart: ingress-nginx-3.10.1
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/version: 0.41.2
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/component: controller
  name: ingress-nginx-controller
  namespace: ingress-nginx
spec:
  type: NodePort
  ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: http
      nodePort: 32080
    - name: https
      port: 443
      protocol: TCP
      targetPort: https
      nodePort: 32443
  selector:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/component: controller
```

然后创建Nginx Ingress控制器：

kubectl apply –f deploy.yaml

```
[root@k8s-master test]# kubectl apply -f deploy.yaml
namespace/ingress-nginx unchanged
serviceaccount/ingress-nginx configured
configmap/ingress-nginx-controller configured
clusterrole.rbac.authorization.k8s.io/ingress-nginx configured
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx configured
role.rbac.authorization.k8s.io/ingress-nginx configured
rolebinding.rbac.authorization.k8s.io/ingress-nginx configured
service/ingress-nginx-controller-admission configured
service/ingress-nginx-controller configured
deployment.apps/ingress-nginx-controller created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission configured
serviceaccount/ingress-nginx-admission configured
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission configured
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission configured
role.rbac.authorization.k8s.io/ingress-nginx-admission configured
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission configured
```

检查Ingress控制器pod是否已经启动：

kubectl get pods –n ingress–nginx \

–l app.kubernetes.io/name=ingress–nginx ––watch

```
[root@k8s-master test]# kubectl get pods -n ingress-nginx \
>   -l app.kubernetes.io/name=ingress-nginx --watch
NAME                                        READY   STATUS              RESTARTS   AGE
ingress-nginx-controller-55bcfcfd89-xfvmv   0/1     ContainerCreating   0          6s
ingress-nginx-controller-55bcfcfd89-xfvmv   0/1     Running             0          28s
ingress-nginx-controller-55bcfcfd89-xfvmv   1/1     Running             0          45s
```

查看生成的service：

```
[root@k8s-master ~]# kubectl get svc -n ingress-nginx
NAME                                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx-controller             NodePort    10.108.239.250   <none>        80:32080/TCP,443:32443/TCP   18h
ingress-nginx-controller-admission   ClusterIP   10.98.223.134    <none>        443/TCP                      18h
[root@k8s-master ~]#
```

到此处就可以直接通过nodeIP:nodePort的方式来访问这个服务。

这个服务是Nginx Ingress 控制器对外访问的一个入口，接收集群外部流量：

```
[root@k8s-master test]# curl 192.168.188.132:32080
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

说明Nginx Ingress控制器部署成功。

通过如下命令查看Ingress 控制器的版本：

POD_NAMESPACE=ingress-nginx

POD_NAME=$(kubectl get pods -n $POD_NAMESPACE -l app.kubernetes.io/name=ingress-nginx --field-selector=status.phase=Running -o jsonpath='{.items[0].metadata.name}')

kubectl exec -it $POD_NAME -n $POD_NAMESPACE -- /nginx-ingress-controller --version

接下来我们就可以创建我们的第一个Ingress资源了。

创建后端服务

创建一个后端服务来演示如何通过Ingress进行访问。

vim test-tomcat-svc.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: test-tomcat-svc
spec:
  selector:
    app: tomcat
  ports:
  - name: http
    targetPort: 8080
    port: 8080
```

apiVersion: v1

kind: Service

metadata:

  name: test-tomcat-svc

spec:

  selector:

    app: tomcat

  ports:

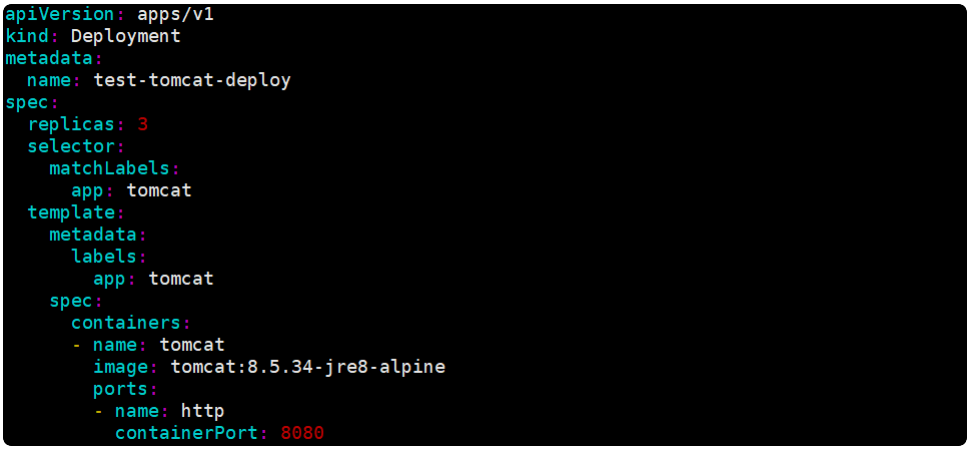
  - name: http

    targetPort: 8080

    port: 8080

```
[root@k8s-master test]# kubectl apply -f test-tomcat-svc.yaml
service/test-tomcat-svc created
[root@k8s-master test]# kubectl get svc --all-namespaces
NAMESPACE       NAME                                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
default         kubernetes                           ClusterIP   10.96.0.1        <none>        443/TCP                      9d
default         test-nodeport                        NodePort    10.97.42.53      <none>        80:30123/TCP                 9d
default         test-svc                             ClusterIP   10.110.132.48    <none>        80/TCP                       9d
default         test-tomcat-svc                      ClusterIP   10.105.75.249    <none>        8080/TCP                     4s
ingress-nginx   ingress-nginx-controller             NodePort    10.108.239.250   <none>        80:32080/TCP,443:32443/TCP   6d2h
ingress-nginx   ingress-nginx-controller-admission   ClusterIP   10.98.223.134    <none>        443/TCP                      6d2h
kube-system     kube-dns                             ClusterIP   10.96.0.10       <none>        53/UDP,53/TCP,9153/TCP       9d
[root@k8s-master test]#
```

vim test-tomcat-deployment.yaml



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-tomcat-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat:8.5.34-jre8-alpine
        ports:
        - name: http
          containerPort: 8080
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-tomcat-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat:8.5.34-jre8-alpine
        ports:
        - name: http
          containerPort: 8080
```

```
kubectl apply -f test-tomcat-deployment.yaml
```

```
[root@k8s-master test]# kubectl apply -f test-tomcat-deployment.yaml
deployment.apps/test-tomcat-deploy created
[root@k8s-master test]# kubectl get pod --all-namespaces
NAMESPACE       NAME                                       READY   STATUS              RESTARTS   AGE
default         test-tomcat-deploy-6b8b5545b6-j45g5        0/1     ContainerCreating   0          9s
default         test-tomcat-deploy-6b8b5545b6-rg9vc        0/1     ContainerCreating   0          9s
default         test-tomcat-deploy-6b8b5545b6-zdddb        0/1     ContainerCreating   0          10s
ingress-nginx   ingress-nginx-controller-55bcfcfd89-xfvmv   1/1     Running             0          5d7h
kube-system     coredns-7ff77c879f-9xkb6                   1/1     Running             1          9d
kube-system     coredns-7ff77c879f-rcxrm                   1/1     Running             1          9d
kube-system     etcd-k8s-master                            1/1     Running             2          9d
kube-system     kube-apiserver-k8s-master                  1/1     Running             2          9d
kube-system     kube-controller-manager-k8s-master         1/1     Running             15         9d
kube-system     kube-flannel-ds-4tb4v                      1/1     Running             2          9d
kube-system     kube-flannel-ds-bds7k                      1/1     Running             1          9d
kube-system     kube-proxy-b7bzc                           1/1     Running             2          9d
kube-system     kube-proxy-v62k4                           1/1     Running             1          9d
kube-system     kube-scheduler-k8s-master                  1/1     Running             16         9d
```

```
[root@k8s-master test]# kubectl get pod --all-namespaces
NAMESPACE       NAME                                       READY   STATUS    RESTARTS   AGE
default         test-tomcat-deploy-6b8b5545b6-j45g5        1/1     Running   0          81s
default         test-tomcat-deploy-6b8b5545b6-rg9vc        1/1     Running   0          81s
default         test-tomcat-deploy-6b8b5545b6-zdddb        1/1     Running   0          82s
```

kubectl get deploy

```
[root@k8s-master test]# kubectl get deploy
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
test-tomcat-deploy   3/3     3            3           112s
[root@k8s-master test]#
```

创建Ingress资源

vim test-tomcat-ingress-nginx.yml

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-tomcat-ingress-nginx
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - host: test.pmx.com
    http:
      paths:
      - path:
        backend:
          serviceName: test-tomcat-svc
          servicePort: 8080
```

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

  name: test-tomcat-ingress-nginx

  annotations:

    kubernetes.io/ingress.class: "nginx"

spec:

  rules:

  - host: test.pmx.com

```
    http:
      paths:
      - path:
        backend:
          serviceName: test-tomcat-svc
          servicePort: 8080
```

kubectl apply -f test-tomcat-ingress-nginx.yml

```
[root@k8s-master test]# kubectl apply -f test-tomcat-ingress-nginx.yml
ingress.extensions/test-tomcat-ingress-nginx created
[root@k8s-master test]# kubectl get ingress
NAME                        CLASS    HOSTS          ADDRESS   PORTS   AGE
test-tomcat-ingress-nginx   <none>   test.pmx.com             80      10s
[root@k8s-master test]#
```

进入nginx ingresss控制器pod，查看更新后的nginx配置文件：

kubectl exec -it ingress-nginx-controller-55bcfcfd89-xfvmv -n ingress-nginx bash

cat nginx.conf

```
## start server test.pmx.com
server {
        server_name test.pmx.com ;

        listen 80  ;
        listen 443  ssl http2 ;

        set $proxy_upstream_name "-";

        ssl_certificate_by_lua_block {
                certificate.call()
        }

        location / {

                set $namespace      "default";
                set $ingress_name   "test-tomcat-ingress-nginx";
                set $service_name   "test-tomcat-svc";
                set $service_port   "8080";
                set $location_path  "/";

                rewrite_by_lua_block {
                        lua_ingress.rewrite({
                                force_ssl_redirect = false,
                                ssl_redirect = true,
                                force_no_ssl_redirect = false,
                                use_port_in_redirects = false,
                        })
                        balancer.rewrite()
                        plugins.run()
                }

                # be careful with `access_by_lua_block` and `satisfy any` directives as satisfy
                # will always succeed when there's `access_by_lua_block` that does not have any
                # other authentication method such as basic auth or external auth useless - all
                #access_by_lua_block {
                #}

                header_filter_by_lua_block {
                        lua_ingress.header()
                        plugins.run()
                }

                body_filter_by_lua_block {
                }

                log_by_lua_block {
                        balancer.log()

                        monitor.call()
```

访问后端服务

在本地Windows系统上配置hosts文件，添加一条hosts记录：

192.168.188.132 test.pmx.com

其中192.168.188.132为节点IP地址。

访问地址：test.pmx.com:32080