42 Kubernetes Service 类型

更新时间：2020-11-02 09:58:53

> 衡量一个人的真正品格，是看他在知道没人看见的时候干些什么。——孟德斯鸠

Kubernetes Service 类型

我们上一篇文章简单介绍了 Kubernetes Service 的使用，默认使用了 ClusterIP 的 Service 类型，实际上 Kubernetes 支持的 Service 类型有多种：

- **ClusterIP**：默认的 Service Type，通过集群的内部 IP 暴露服务，只能在集群内部进行访问；
- **NodePort**：通过每个 Node 上面的某个端口 （NodePort）暴露服务。通过该端口的请求会自动路由到后端的 ClusterIP 服务，这个 ClusterIP 服务是自动创建的。通过 NodePort，我们可以在集群外部访问我们的服务，但是，在生产环境上面并不建议使用 NodePort；
- **LoadBalancer**：使用云厂商提供的负载均衡器，可以向外部暴露服务。外部的负载均衡器可以路由到 NodePort 和 ClusterIP 服务；
- **ExternalName**：通过返回 CNAME 将服务映射到 externalName 字段中的内容；
- **Ingress**：严格来说，Ingress 不是一种服务类型，而是用来充当集群的服务的入口点。Ingress 可以将路由规则整合到一个资源中，然后通过同一个 IP 地址暴露多个服务。

1. ClusterIP

ClusterIP 模式是 Kubernetes Service 的默认类型，ClusterIP 类型的 Service 一个重要特点就是只能在集群内部访问。

2. NodePort

使用 NodePort 类型的 Serive 只需要在 spec 中将 type 中指定为 NodePort 即可，下面是一个简单的例子。

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service-nodeport
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 30001
      targetPort: 80
```

类似的我们通过 `kubectl apply` 创建 Service 对象。

```
$ kubectl apply -f nginx-service-nodeport.yaml -n imooc
service/nginx-service-nodeport created
$ kubectl get service -n imooc
NAME                    TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
nginx-service           ClusterIP   10.0.213.149   <none>        80/TCP           7h26m
nginx-service-nodeport  NodePort    10.0.8.178     <none>        30001:31633/TCP  1s
```

下面的 nginx-service-nodeport 就是我们刚刚创建的 NodePort 类型的 Service，我们可以看到 TYPE 显示为了 NodePort。并且 CLUSTER-IP 字段也分配了一个 IP，这个是怎么回事呢？其实这个是 NodePort 类型的 Service 自动创建的 ClusterIP，也就是说 NodePort 类型的 Service 后端还是通过 ClusterIP 来实现的。

然后是 PORT(S) 字段域有两个端口值，前面表示 ClusterIP 对应的端口，也就是 30001；后面的表示 Node 本地对应的端口，是 31633。那么可能有人会问了，我们定义 Service 的时候并没有指定 Node 的本地端口是多少啊？这个端口值是随机的吗？

是的，没错，确实是随机的，只不过是在一个区间内随机。这个区间是在 kubernetes 的 ApiServer 启动的时候，启动参数里面通过指定参数 `--service-node-port-range` 来指定的，默认为 30000 - 32767。

下面是我在阿里云的 ACK 上购买的 Kubernetes 集群内的 ApiServer 的启动参数，我们可以看到参数 `--service-node-port-range` 指定的值为 30000 - 32767，和默认值一样，对于我们启动的 NodePort 的 Service 系统分配的端口值为 31633，也是落在这个区间。

```
ps aux | grep service-node-port
root     6351  2.0  2.9 477832 235040 ?       Ssl  2019 5018:00 kube-apiserver --audit-log-maxbackup=10 --audit-log-maxsize=100 --audit-log-path=/var/log/kubernetes/kubernetes.audit --audit-log-maxage=7 --audit-policy-file=/etc/kubernetes/audit-policy.yml --apiserver-count=500 --endpoint-reconciler-type=lease --enable-aggregator-routing=true --runtime-config=admissionregistration.k8s.io/v1beta1 --profiling=false --advertise-address=172.16.60.185 --allow-privileged=true --authorization-mode=Node,RBAC --client-ca-file=/etc/kubernetes/pki/apiserver-ca.crt --cloud-provider=external --enable-admission-plugins=NodeRestriction --enable-bootstrap-token-auth=true --etcd-cafile=/etc/kubernetes/pki/etcd/ca.pem --etcd-certfile=/etc/kubernetes/pki/etcd/etcd-client.pem --etcd-keyfile=/etc/kubernetes/pki/etcd/etcd-client-key.pem --etcd-servers=https://172.16.60.183:2379,https://172.16.60.184:2379,https://172.16.60.185:2379 --feature-gates=VolumeSnapshotDataSource=true,CSINodeInfo=true,CSIDriverRegistry=true --insecure-port=0 --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key --requestheader-allowed-names=front-proxy-client --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt --requestheader-extra-headers-prefix=X-Remote-Extra- --requestheader-group-headers=X-Remote-Group --requestheader-username-headers=X-Remote-User --secure-port=6443 --service-account-key-file=/etc/kubernetes/pki/sa.pub --service-cluster-ip-range=10.0.0.0/16 --service-node-port-range=30000-32767 --tls-cert-file=/etc/kubernetes/pki/apiserver.crt --tls-private-key-file=/etc/kube
```

NodePort 的值实际上是可以我们自己指定的，指定的时候需要注意的是，一定要保证指定的值处于参数 `--service-node-port-range` 指定的区间内，不然可能会导致 Service 创建失败。下面是指定 NodePort 的示例 yaml。

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service-nodeport
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 30001
      targetPort: 80
      nodePort: 30002
```

我们还是通过 `kubectl describe` 来看一下 NodePort 的 Service 对象。

```
$ kubectl describe service nginx-service-nodeport -n imooc
Name:                     nginx-service-nodeport
Namespace:                imooc
Labels:                   <none>
Annotations:              kubectl.kubernetes.io/last-applied-configuration:
                          {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"nginx-service-nodeport","namespace":"imooc"},"spec":{"ports":[{
"p...
Selector:                 app=nginx
Type:                     NodePort
IP:                       10.0.8.178
Port:                     <unset>  30001/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31633/TCP
Endpoints:                10.1.1.154:80,10.1.2.159:80,10.1.2.31:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

和 ClusterIP 类型的 Service 对象简单对比一下不难发现， NodePort 类型的 Service 对象除了 Type 为 NodePort 外，只多个了一个 NodePort 字段，总结一下 NodePort 类型的 Service 的几种端口：

- NodePort：Node 节点本地启动的用来监听和转发请求的端口，每个节点上都会启动；
- Port：NodePort 类型的 Service 自动创建的 ClusterIP 的端口；
- TargetPort：ClusterIP 转发的目标端口。

所以对于 NodePort 类型的 Service，外部的请求顺序是：**NodePort -> Port -> TargetPort**。

3. LoadBalancer

LoadBalancer 类型的 Service 只需要在 Service 的 spec 中将 type 指定为 LoadBalancer 即可，然后将会异步的创建负载均衡器。这样外部流量将向请求到外部的负载均衡器上，然后转发到后端的真正提供服务的 Pod 上，但是 LoadBalancer 的具体实现要依赖于云提供厂商。下面我们以阿里云的容器服务为例，创建一个 LoadBalancer 类型的 Service，描述文件如下。

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service-lb
spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 30005
      targetPort: 80
```

然后通过 `kubectl apply` 创建该 Service。

```
$ kubectl apply -f nginx-service-lb.yaml -n imooc
service/nginx-service-lb configured
$ kubectl get service -n imooc
NAME                TYPE          CLUSTER-IP     EXTERNAL-IP      PORT(S)          AGE
nginx-service       ClusterIP     10.0.213.149   <none>           80/TCP           11h
nginx-service-lb    LoadBalancer  10.0.13.63     39.102.158.120   30005:30423/TCP  64m
```

这里的 EXTERNAL-IP 就是外部的负载均衡器的 IP，对应的端口是 30005，同时我们可以看到在 PORT(S) 字段域还起了一个本地的 NodePort 端口 30423。

我们下面用 curl 请求一个外部的负载均衡器，返回的信息确实是 nginx 服务器，也就是后端的 Pod 返回的。

```
$ curl 39.102.158.120:30005
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

我们通过 kubectl describe 查看一下 LoadBalancer 类型的 Service 的信息。

```
$ kubectl describe service nginx-service-lb -n imooc
Name:                     nginx-service-lb
Namespace:                imooc
Labels:                   <none>
Annotations:              kubectl.kubernetes.io/last-applied-configuration:
                          {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"nginx-service-lb","namespace":"imooc"},"spec":{"ports":[{"port":
3...
Selector:                 app=nginx
Type:                     LoadBalancer
IP:                       10.0.13.63
LoadBalancer Ingress:     39.102.158.120
Port:                     <unset>  30005/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30423/TCP
Endpoints:                10.1.1.154:80,10.1.2.159:80,10.1.2.31:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

我们可以看到除了 Type 变成 LoadBalancer 类型之外，还多了一个 LoadBalancer Ingress，其实就是外部的负载均衡器的 IP。

4. ExternalName

类型为 ExternalName 的 Service 将服务映射到 DNS 名称，而不是通过 selector 选择器。可以使用 spec.externalName 参数指定 DNS 名称。例如，如下 Service 定义将服务 nginx-service-external-name 服务映射为 www.baidu.com 。

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service-external-name
spec:
  type: ExternalName
  externalName: www.baidu.com
```

我们还是通过 kubectl apply 来创建 Service

```
$ kubectl apply -f nginx-service-ename.yaml -n imooc
service/nginx-service-external-name created
$ kubectl get service nginx-service-external-name -n imooc
NAME                         TYPE           CLUSTER-IP   EXTERNAL-IP     PORT(S)   AGE
nginx-service-external-name  ExternalName   <none>       www.baidu.com   <none>    17s
```

我们可以看到 Service nginx-service-external-name 显示的 TYPE 为 ExternalName，EXTERNAL-IP 为我们的 spec.externalName 字段定义的值。当查找服务 nginx-service-external-name.imooc.svc.cluster.local 时，集群 DNS 服务将返回 CNAME 记录，也就是 www.baidu.com 。访问该 Service 的方式与其他服务的方式相同，但主要区别在于重定向发生在 DNS 级别，而不是通过代理和转发。

当然也可以通过 kubectl describe 来看一下 Service 的详情信息。

```
$ kubectl describe service nginx-service-external-name -n imooc
Name:              nginx-service-external-name
Namespace:         imooc
Labels:            <none>
Annotations:       kubectl.kubernetes.io/last-applied-configuration:
                   {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"nginx-service-external-name","namespace":"imooc"},"spec":{"extern
...
Selector:          <none>
Type:              ExternalName
IP:
External Name:     www.baidu.com
Session Affinity:  None
Events:            <none>
```

5. Ingress

Ingress 严格来说并不是一种 Service 类型，而是 Kubernetes 官方提供的用于对外暴露服务的方式。下面是一个简单的 Ingress 的声明。

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
spec:
  rules:
  - host: www.example.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: nginx-service
          servicePort: 80
```

Ingress 中的最核心的地方是 `spec.rules`，可以在 rule 中定一个多个规则，每个规则下面包含以下信息：

- host：服务暴露的域名；
- http：路由转发协议，可以是 http 或者 https，协议下面包含：
    - path：路由 router；
    - backend：后端服务，主要包括服务名称和服务端口。

6. 总结

本文简单介绍了 Kubernetes 提供的几种服务类型：

- ClusterIP 是一种默认的服务类型，具有较多限制；
- NodePort 是一种可以快速暴露服务的服务类型，一般用来快速调试；
- LoadBalancer 有时候会用于生产环境；
- ExternalName 很少使用；
- Ingress 目前 kubernetes 集群向外暴露服务的最长使用的方式。

}

41 使用 Service 访问一组特定的 Pod