27.NodePort和LoadBalancer

到目前为止，我们只讨论了服务如何被集群中的pod消费。但我们也经常需要将某些服务对外暴露出去（比如前端web服务器），以便外部客户端可以访问它们。
从外部访问服务可以有如下几种方式：

- 设置服务类型为NodePort。对于一个NodePort服务来说，每个集群节点都会在节点本身（所以叫做NodePort）上打开一个端口，然后将在该端口上接收到的流量重定向到底层的服务。该服务不能仅通过内部集群IP和端口访问，还可以通过位于所有节点上的专用端口访问。
- 设置服务类型为LoadBalancer。LoadBalancer是NodePort类型的一个扩展，它使得服务可以通过一个专用的负载均衡器来访问。这个负载均衡器由Kubernetes运行的云基础设施提供，跨所有节点将流量重定向到节点端口。客户端通过负载均衡器的IP连接到服务。
- 创建一个Ingress资源。这是一个完全不同的机制，通过一个IP地址暴露多个服务。它运行在HTTP层（网络协议第7层，即应用层），因此能提供比工作在第4层的服务更多的功能。这一点我们将在后面专题讨论。

NodePort服务

将一组pod暴露给外部客户端的第一种方式就是创建一个类型为NodePort的服务。通过创建一个NodePort服务，可以让Kubernetes在所有节点上保留一个端口（所有节点都使用相同的端口号），并将传入的连接转发到作为服务一部分的pods。
这与常规服务（类型为ClusterIP的服务）类似，但是NodePort服务不但可以通过服务的内部集群IP访问，还可以通过任何节点的IP以及预留节点端口访问。

创建NodePort服务

现在我们来创一个NodePort服务，YAML文件如下：

```
apiVersion: v1
kind: Service
metadata:
  name: test-nodeport
spec:
  type: NodePort
  ports:
  - port: 80
```

targetPort: 8080

nodePort: 30123

selector:

app: test1

```
apiVersion: v1
kind: Service
metadata:
  name: test-nodeport
spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 8080
    nodePort: 30123
  selector:
    app: test1
```

在上面的YAML文件中，我们将服务的类型设置为NodePort，服务内部集群IP端口设置为80，服务后面相应的pod的端口为8080。服务可以通过每个集群节点上的30123端口访问到。如果不指定NodePort，Kubernetes就会选择一个随机的端口。

执行如下命令创建这个服务：

kubectl create –f test–svc–nodeport.yaml

```
[root@k8s-master test]# kubectl create -f test-svc-nodeport.yaml
service/test-nodeport created
```

查看NodePort服务：

kubectl get svc test–nodeport

```
[root@k8s-master test]# kubectl get svc test-nodeport
NAME            TYPE       CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
test-nodeport   NodePort   10.97.42.53   <none>        80:30123/TCP   102s
[root@k8s-master test]#
```

EXTERNAL–IP列显示为<nodes>，表示该服务可以通过任意集群节点的IP地址访问到。PORT(S)列显示了集群IP的内部端口号（80）和节点端口（30123）。

服务可以通过如下地址访问到：

- 10.97.42.53:80

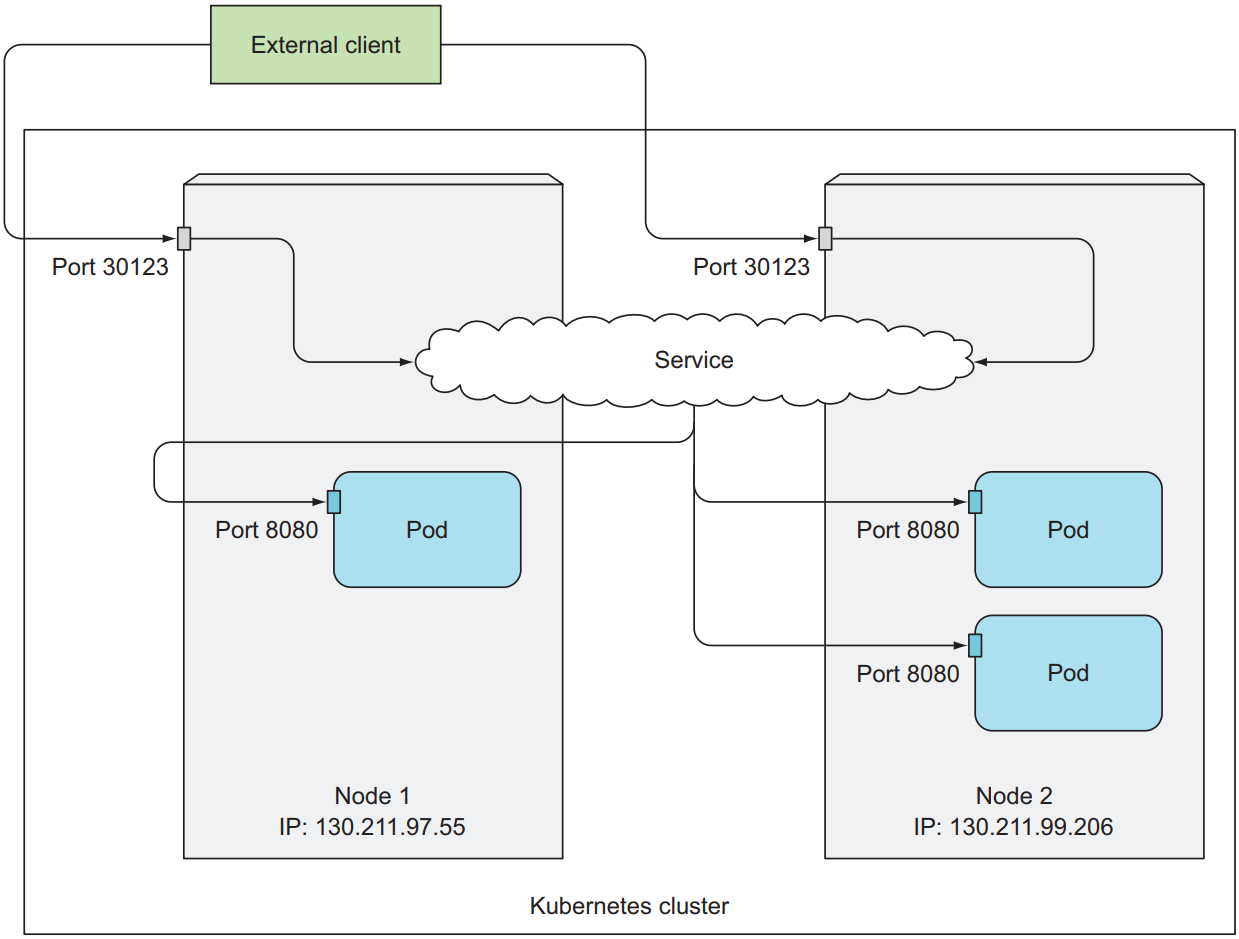
```
[root@k8s-master test]# kubectl exec test-rc-lk84p -- curl 10.97.42.53:80
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0已发送消息至： test-rc-wrr5p
100    36    0    36    0     0  12418      0 --:--:-- --:--:-- --:--:-- 18000
```

- <节点1的IP>:30123，依次类推

```
[root@k8s-master test]# curl 192.168.188.132:30123
已发送消息至： test-rc-rkczt
```

如下图所示，服务暴露在集群中所有节点的30123端口上。传入到其中一个节点30123端口上的连接会被重定向到一个随机选择的pod，这个pod所在节点可能与连接传入的端口所在的节点不是同一个。外

部客户端通过Node1或者Node2连接到NodePort服务。



从上面可以看出，整个Internet可以通过任意节点上的30123端口访问pod，不管客户端请求发送到哪一个节点。但是如果我们让外界只通过第一个节点访问pod，当该节点出故障时，客户端就再也不能访问服务。这就是为什么要在节点之前放置一个负载均衡器，以便能够将请求分发到健康的节点，而不是发送到一个离线的节点。

如果Kubernetes集群支持负载均衡器（对于部署在云平台上的Kubernetes集群来说通常是这样），可以通过创建一个LoadBalancer而不是NodePort服务自动生成负载均衡器。

LoadBalancer服务

运行在云供应商平台上的Kubernetes集群通常都支持从云基础设施自动提供负载均衡器（Load Balancer）。我们只需要设置服务类型为LoadBalancer即可。负载均衡器拥有独立的、可公共访问的IP地址并将所有连接重定向到服务。

可以通过负载均衡器的IP地址访问服务。

如果Kubernetes的运行环境不支持LoadBalancer类型的服务，则不会提供负载均衡器，但是该服务仍然会表现得像一个NodePort服务，因为LoadBalancer服务是NodePort服务的一个扩展。

现在我们来创建一个LoadBalancer服务：

vim test-svc-loadbalancer.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: test-loadbalancer
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
```

apiVersion: v1

kind: Service

metadata:

  name: test-loadbalancer

spec:

  type: LoadBalancer

  ports:

  - port: 80

    targetPort: 8080

  selector:

    app: test1

在上面的YAML文件中，我们指定type为LoadBalancer，这种类型的服务会自动从Kubernetes集群所在的环境的基础设施获取负载均衡器。另外，我们也并未特别指定节点端口，虽然可以这样做，但是此处我们让Kubernetes自动选择一个。

创建服务之后，云基础设施需要花些时间来创建负载均衡器并将其IP地址写入到该Service对象中。该IP地址会显示在EXTERNAL-IP列，如下图，由于当前实验环境不支持负载均衡器，所以一直显示为<peinding>。

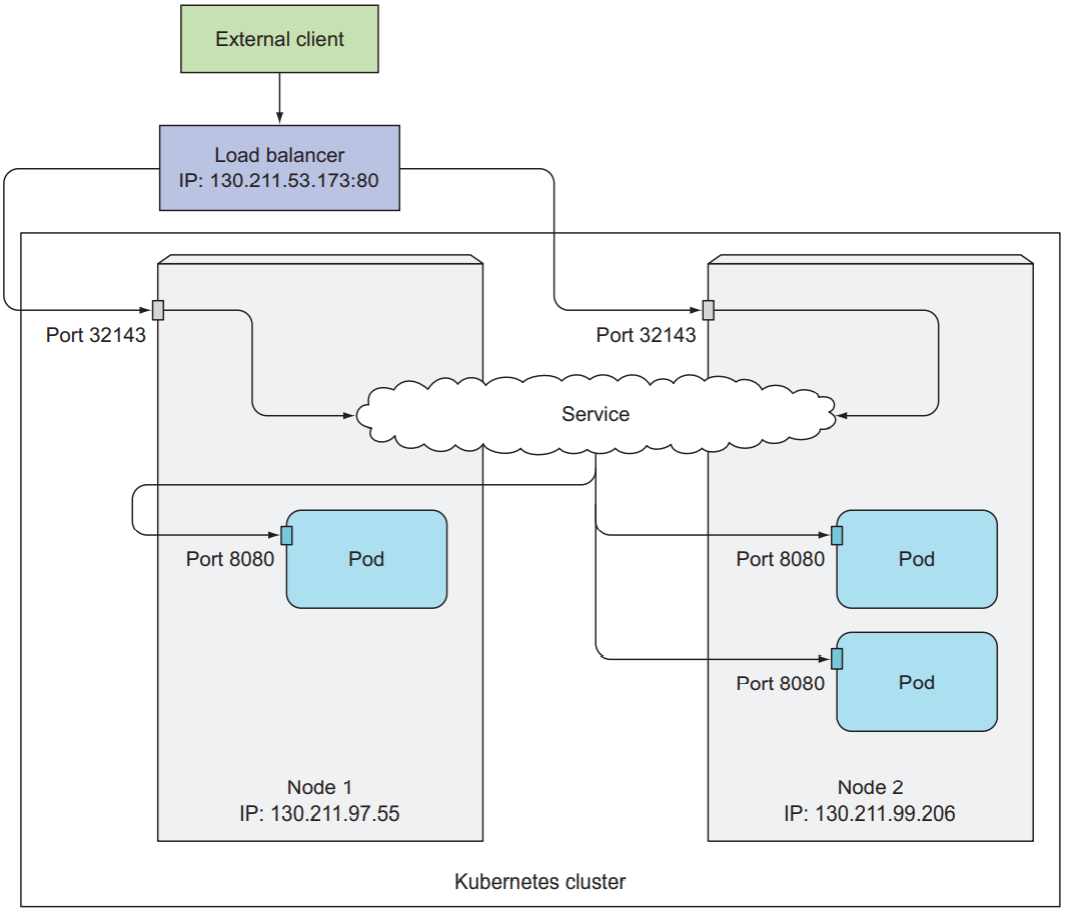
kubectl get svc test-loadbalancer

```
[root@k8s-master test]# kubectl get svc test-loadbalancer
NAME               TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
test-loadbalancer   LoadBalancer   10.106.131.85   <pending>     80:31922/TCP   12m
[root@k8s-master test]#
```

对于支持负载均衡器的环境，EXTERNAL-IP列会显示负载均衡器的IP地址，然后我们就可以通过这个地址访问该服务了：

curl http://<Load Balancer's IP>

下图展示了外部客户端请求是通过LoadBalancer服务被转发到pod上的：



如上图所示，外部客户端连接到负载均衡器的80端口，然后路由到一个随机分配到的节点端口上，之后该连接被转发到某个pod实例（该pod实例所在的节点与收到连接的节点可能不是同一个）。

由于LoadBalancer服务就是在NodePort服务的基础上提供了一个负载均衡器，所以如果我们使用kubectl describe命令查看该服务的信息时，可以看到Kubernetes为该服务自动选择了一个节点端口。如果我们为该端口打开防火墙，就可以像之前访问NodePort服务时通过节点IP访问到该服务。