从外部访问Kubernetes中的Pod

前面几节讲到如何访问kubneretes集群，本文主要讲解访问kubenretes中的Pod和Serivce的集中方式，包括如下几种：

- hostNetwork
- hostPort
- NodePort
- LoadBalancer
- Ingress

说是暴露Pod其实跟暴露Service是一回事，因为Pod就是Service的backend。

hostNetwork: true

这是一种直接定义Pod网络的方式。

如果在Pod中使用`hostNotwork:true`配置的话，在这种pod中运行的应用程序可以直接看到pod启动的主机的网络接口。在主机的所有网络接口上都可以访问到该应用程序。以下是使用主机网络的pod的示例定义：

```
apiVersion: v1
kind: Pod
metadata:
  name: influxdb
spec:
  hostNetwork: true
  containers:
    - name: influxdb
      image: influxdb
```

部署该Pod：

```
$ kubectl create -f influxdb-hostnetwork.yml
```

访问该pod所在主机的8086端口：

```
curl -v http://$NODE_IP:8086/ping
```

将看到204 No Content的204返回码，说明可以正常访问。

注意每次启动这个Pod的时候都可能被调度到不同的节点上，所有外部访问Pod的IP也是变化的，而且调度Pod的时候还需要考虑是否与宿主机上的端口冲突，因此一般情况下除非您知道需要某个特定应用占用特定宿主机上的特定端口时才使用`hostNetwork: true`的方式。
这种Pod的网络模式有一个用处就是可以将网络插件包装在Pod中然后部署在每个宿主机上，这样该Pod就可以控制该宿主机上的所有网络。

hostPort

这是一种直接定义Pod网络的方式。
`hostPort`是直接将容器的端口与所调度的节点上的端口路由，这样用户就可以通过宿主机的IP加上来访问Pod了，如:。

```
apiVersion: v1
kind: Pod
metadata:
  name: influxdb
spec:
  containers:
    - name: influxdb
      image: influxdb
      ports:
        - containerPort: 8086
```

```
      hostPort: 8086
```

这样做有个缺点，因为Pod重新调度的时候该Pod被调度到的宿主机可能会变动，这样就变化了，用户必须自己维护一个Pod与所在宿主机的对应关系。
这种网络方式可以用来做 nginx Ingress controller。外部流量都需要通过kubenretes node节点的80和443端口。

NodePort

NodePort在kubenretes里是一个广泛应用的服务暴露方式。Kubernetes中的service默认情况下都是使用的ClusterIP这种类型，这样的service会产生一个ClusterIP，这个IP只能在集群内部访问，要想让外部能够直接访问service，需要将service type修改为 nodePort。

```
apiVersion: v1
kind: Pod
metadata:
  name: influxdb
  labels:
    name: influxdb
spec:
  containers:
    - name: influxdb
      image: influxdb
      ports:
        - containerPort: 8086
```

同时还可以给service指定一个nodePort值，范围是30000-32767，这个值在API server的配置文件中，用--service-node-port-range定义。

```
kind: Service
```

```
apiVersion: v1
metadata:
  name: influxdb
spec:
  type: NodePort
  ports:
    - port: 8086
      nodePort: 30000
  selector:
    name: influxdb
```

集群外就可以使用kubernetes任意一个节点的IP加上30000端口访问该服务了。kube-proxy会自动将流量以round-robin的方式转发给该service的每一个pod。
这种服务暴露方式，无法让你指定自己想要的应用常用端口，不过可以在集群上再部署一个反向代理作为流量入口。

LoadBalancer

`LoadBalancer` 只能在service上定义。这是公有云提供的负载均衡器，如AWS、Azure、CloudStack、GCE等。

```
kind: Service
apiVersion: v1
metadata:
  name: influxdb
spec:
  type: LoadBalancer
  ports:
    - port: 8086
```

```
  selector:
    name: influxdb
```

查看服务：

```
$ kubectl get svc influxdb
NAME       CLUSTER-IP      EXTERNAL-IP      PORT(S)
AGE
influxdb   10.97.121.42    10.13.242.236    8086:30051/TCP
39s
```

内部可以使用ClusterIP加端口来访问服务，如19.97.121.42:8086。
外部可以用以下两种方式访问该服务：

- 使用任一节点的IP加30051端口访问该服务
- 使用EXTERNAL-IP来访问，这是一个VIP，是云供应商提供的负载均衡器IP，如10.13.242.236:8086。

Ingress

Ingress是自kubernetes1.1版本后引入的资源类型。必须要部署Ingress controller才能创建Ingress资源，Ingress controller是以一种插件的形式提供。Ingress controller 是部署在Kubernetes之上的Docker容器。它的Docker镜像包含一个像nginx或HAProxy的负载均衡器和一个控制器守护进程。控制器守护程序从Kubernetes接收所需的Ingress配置。它会生成一个nginx或HAProxy配置文件，并重新启动负载平衡器进程以使更改生效。换句话说，Ingress controller是由Kubernetes管理的负载均衡器。Kubernetes Ingress提供了负载平衡器的典型特性：HTTP路由，粘性会话，SSL终止，SSL直通，TCP和UDP负载平衡等。目前并不是所有的Ingress controller都实现了这些功能，需要查看具体的Ingress controller文档。

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: influxdb
spec:
  rules:
    - host: influxdb.kube.example.com
      http:
        paths:
          - backend:
              serviceName: influxdb
              servicePort: 8086
```

外部访问URL http://influxdb.kube.example.com/ping 访问该服务，入口就是80端口，然后Ingress controller直接将流量转发给后端Pod，不需再经过kube-proxy的转发，比LoadBalancer方式更高效。

总结

总的来说Ingress是一个非常灵活和越来越得到厂商支持的服务暴露方式，包括Nginx、HAProxy、Traefik，还有各种Service Mesh，而其它服务暴露方式可以更适用于服务调试、特殊应用的部署。

参考

- Accessing Kubernetes Pods from Outside of the Cluster - alesnosek.com