

Kubernetes: 通过无头服务(Headless Service)实现客户端负载均衡

发布于 2023-01-30 15:14:46

👁 6.8K

💬 0

🚩 举报

写在前面

- 分享一些 `K8s` 中 `Headless Service` 的笔记
- 博文内容涉及：
 - `Headless Service` 的简单介绍
 - `Headless Service` 创建
 - 集群内/外 获取 `Headless Service` 对应 `Pod` 列表的 Demo
- 理解不足小伙伴帮忙指正

Headless Service 简单介绍

在某些场景中, 如果我们希望自己控制 Pod 实例的 `负载均衡` 策略, 或者希望直接和 Pod 交互但是又不希望通过端口映射的方式, 比如 [数据库](#) 根据情况做一些读写分离, 或者一些应用在客户端做流量控制等, 不使用 `Service` 提供的由 `Kube-proxy` 代理实现的 `默认负载均衡` 的功能。希望明确是由那几个 `pod` 提供能力, 即直接通过 `Pod` 发布服务, 而不是只有一个 集群 IP `Cluster IP` 或者使用 `NodePort` 、 `LoadBalancer` 、 `ExternalName` 来发布服务。

这个时候, K8s 提供了 `Headless Service`, 即不为 `Service` 设置 `ClusterIP(入口IP地址)`, 也叫 无头服务, 这里分两种情况

有选择器

第一种是有对应的服务能力提供者, 即通过标签选择器选择了对应的后端能力, 比如 `pod` , `deployment` , `statefulset` 等

在这种情况下, 会通过 `Label Selector` 将被选择的后端 Pod 列表返回给调用的客户端, K8s 不会为这样的 Service 分配任何 IP, `DNS` 会为这些 `Service 的 Name` 添加一系列的 A(AAA)记录(IP 地址指向), 直接指向后端映射的 Pod。当然前提是 `通过 标签选择器选择到了对应的 pod。`

`Kube-proxy` 不会处理这类型的 `Service` , 没有 `负载均衡` 机制也没有请求映射, 这里 `Endpoint Controller` 任然会创建 `pod` 对应的 `Endpoint` , 同时 `Kubernetes` 控制平面会在 Kubernetes `API` 中创建 `EndpointSlice` 对象

`EndpointSlices` 表示针对服务的后端网络端点的子集(切片), 这是在 1.21 版本才出现的, 提供了一种简单的方法来跟踪 Kubernetes 集群中的网络端点(network endpoints)。EndpointSlices 为 Endpoints 提供了一种可扩缩和可拓展的替代方案。

在 Kubernetes 中，EndpointSlice 包含对一组网络端点的引用。控制面会自动为设置了选择算符的 Kubernetes Service 创建 EndpointSlice,EndpointSlice 将包含对与 Service 选择算符匹配的所有 Pod 的引用。EndpointSlice 通过唯一的协议、端口号和 Service 名称将网络端点组织在一起

Headless Service 通过暴露的 Endpoints 列表 应用可以通过编码实现客户端的负载均衡。

没有选择器

第二种是没有对应的服务能力提供者，即没有通过选择运算符来获取当前 集群的能力，这个时候，系统不会创建对应 Endpoint ，也不会创建对应的 EndpointSlice 。

这种情况下，DNS 系统会查找和配置以下之一

- 对于 type: ExternalName 服务，查找和配置其 CNAME 记录
- 对所有其他类型的服务，针对 Service 的就绪端点的所有 IP 地址，查找和配置 DNS A / AAAA 条记录
- 对于 IPv4 端点，DNS 系统创建 A 条记录。
- 对于 IPv6 端点，DNS 系统创建 AAAA 条记录。

Headless Service 创建

定义一个 Service 里面的 ClusterIP 为 None ，并且拥有 Selector 标签选择器，这样的 Service 为 Headless Service

查看当前 k8s 中是否存在 Headless

代码语言: javascript 复制

```
1 | [root@vms81.liruilongs.github.io]-[~]
2 | └─$kubectl get svc -A | grep None
3 | awx                                awx-demo-postgres-13          ClusterIP   None          <none>          5432/TCP
4 | kube-system                       liruilong-kube-prometheus-kubelet ClusterIP   None          <none>          10250/TCP,10255/TCP,4194/TCP
```

一般情况下 SatefulSet 需要 Headless Service 来实现 Pod 的网络的一致性(必须创建此服务)，为客户端返回多个服务端地址。可以看到当前的集群中有两个 Headless Service ，一个是有状态应用(SatefulSet) postgres 数据库创建，一个是搭建 prometheus 集群监控创建的。

代码语言: javascript 复制

```
1 | [root@vms81.liruilongs.github.io]-[~]
2 | └─$kubectl get svc awx-demo-postgres-13 -o yaml
3 | apiVersion: v1
4 |
```

```

5  kind: Service
6  metadata:
7      .....
8      name: awx-demo-postgres-13
9      namespace: awx
10 spec:
11     clusterIP: None
12     clusterIPs:
13     - None
14     internalTrafficPolicy: Cluster
15     ipFamilies:
16     - IPv4
17     ipFamilyPolicy: SingleStack
18     ports:
19     - port: 5432
20       protocol: TCP
21       targetPort: 5432
22     selector:
23       app.kubernetes.io/component: database
24       app.kubernetes.io/instance: postgres-13-awx-demo
25       app.kubernetes.io/managed-by: awx-operator
26       app.kubernetes.io/name: postgres-13
27       app.kubernetes.io/part-of: awx-demo
28     sessionAffinity: None
29     type: ClusterIP
30 status:
    loadBalancer: {}

```

这里我们可以大概的看到一个 `Headless` 资源文件定义，对这样的 Service 进行访问，得到的就是一个 `符合选择器的全部的 Pod 列表`，然后客户端去自行的处理这些 Pod 列表。上面的 Service 中，客户端访问 `postgres` 数据库，会返回符合当前选择器的所有 postgres pod。

下面我们来看几个实际的 Demo


有状态 Headless 服务

对于有状态服务来讲，需要创建 `StatefulSet` 为其提供能力，资源文件定义,只是一个 Demo，所以我们这里没有定义 存储卷相关。

```
1  apiVersion: apps/v1
2  kind: StatefulSet
3  metadata:
4    creationTimestamp: null
5    labels:
6      app: web-headless
7    name: web
8  spec:
9    serviceName: web-headless
10   replicas: 3
11   selector:
12     matchLabels:
13       app: web-headless
14   template:
15     metadata:
16       creationTimestamp: null
17     labels:
18       app: web-headless
19     spec:
20       containers:
21         - image: nginx
22           name: nginx-web
23           ports:
24             - containerPort: 80
25               name: nginx-web
26           resources: {}
```

通过资源文件创建有状态的 pod

代码语言: javascript

 复制

```
[root@vms81.liruilongs.github.io]~[~/ansible]
└─$kubectl get statefulsets.apps web -o wide
NAME    READY   AGE    CONTAINERS   IMAGES
web     3/3     96s    nginx-web    nginx
[root@vms81.liruilongs.github.io]~[~/ansible]
└─$kubectl get pods -o wide | grep web*
```

7	web-0	1/1	Running	0	2m13s	10.244.217.10	vms155.liruilongs.github.io	<none>
8	web-1	1/1	Running	0	2m11s	10.244.194.67	vms156.liruilongs.github.io	<none>
9	web-2	1/1	Running	0	114s	10.244.217.11	vms155.liruilongs.github.io	<none>

之后我们需要创建对应的 `Headless Service` ,这里需要注意的是 `clusterIP: None` ,选择器: `app: web-headless`

代码语言: javascript

复制

```

1  └─[root@vms81.liruilongs.github.io]-[~/ansible]
2  └─$cat headless.yaml
3  apiVersion: v1
4  kind: Service
5  metadata:
6    name: web-headless
7    labels:
8      app: nginx_headless
9  spec:
10   ports:
11     - port: 30088
12       targetPort: 80
13       name: nginx-web-headless
14   clusterIP: None
15   selector:
16     app: web-headless

```

创建对应的 Headless SVC

代码语言: javascript

复制

```

1  └─[root@vms81.liruilongs.github.io]-[~/ansible]
2  └─$kubectl apply -f headless.yaml
3  service/web-headless configured
4  └─[root@vms81.liruilongs.github.io]-[~/ansible]
5  └─$kubectl get svc web-headless
6  NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE

```

```
7 | web-headless ClusterIP None <none> 30088/TCP 24h
```

可以看到，他对每个 pod 都创建了对应的 `Endpoint`

代码语言: javascript

复制

```
1 | [root@vms81.liruilongs.github.io]-[~/ansible]
2 | └─$kubectl describe svc web-headless
3 | Name: web-headless
4 | Namespace: awx
5 | Labels: app=nginx_headless
6 | Annotations: <none>
7 | Selector: app=web-headless
8 | Type: ClusterIP
9 | IP Family Policy: SingleStack
10 | IP Families: IPv4
11 | IP: None
12 | IPs: None
13 | Port: nginx-web-headless 30088/TCP
14 | TargetPort: 80/TCP
15 | Endpoints: 10.244.194.67:80,10.244.217.10:80,10.244.217.11:80
16 | Session Affinity: None
17 | Events: <none>
```

可以通过不同的方式获取 Headless Service 的 Pod 列表。

集群外获取 Headless Service 的 Pod 列表

可以直接通过调用 Rest 接口的方式获取 Headless 对应的 Endpoints，这里为了方便暴露 Rest 服务，通过 `kubect1 proxy` 做一个内部代理。

代码语言: javascript

复制

```
1 | [root@vms81.liruilongs.github.io]-[~]
2 | └─$nohup kubect1 proxy --port=30021 &
3 | [1] 109103
4 | [root@vms81.liruilongs.github.io]-[~]
5 |
```

└─\$nohup: 忽略输入并把输出追加到"nohup.out"

测试一下

代码语言: javascript

复制

```
1 └─[root@vms81.liruilongs.github.io]-[~]
2 └─$curl http://localhost:30021/api/ -s
3 {
4   "kind": "APIVersions",
5   "versions": [
6     "v1"
7   ],
8   "serverAddressByClientCIDRs": [
9     {
10      "clientCIDR": "0.0.0.0/0",
11      "serverAddress": "192.168.26.81:6443"
12    }
13  ]
14 }
```

对于 1.21 之前的版本获取 endpoints, 可以通过 `Endpoints` 的方式获取

代码语言: javascript


复制

```
1 └─[root@vms81.liruilongs.github.io]-[~/ansible]
2 └─$kubectl describe endpoints web-headless
3 Name:          web-headless
4 Namespace:     awx
5 Labels:        app=nginx_headless
6                service.kubernetes.io/headless=
7 Annotations:   endpoints.kubernetes.io/last-change-trigger-time: 2022-12-09T07:13:15Z
8 Subsets:
9   Addresses:    10.244.194.67,10.244.217.10,10.244.217.11
10  NotReadyAddresses: <none>
11  Ports:
```

```
12      Name                Port  Protocol
13      ----                -
14      nginx-web-headless  80    TCP
15
16  Events:  <none>
```

通过 curl 调用对应的 REST 接口

代码语言: javascript

 复制

```
1  [root@vms81.liruilongs.github.io]-[~/ansible]
2  └─$curl -s http://localhost:30021/api/v1/namespaces/awx/endpoints/web-headless | jq .subsets
3  [
4    {
5      "addresses": [
6        {
7          "ip": "10.244.194.67",
8          "hostname": "web-1",
9          "nodeName": "vms156.liruilongs.github.io",
10         "targetRef": {
11           "kind": "Pod",
12           "namespace": "awx",
13           "name": "web-1",
14           "uid": "d71722db-1c41-44ee-a55d-0042c7d2086e",
15           "resourceVersion": "14843070"
16         }
17       },
18       {
19         "ip": "10.244.217.10",
20         "hostname": "web-0",
21         "nodeName": "vms155.liruilongs.github.io",
22         "targetRef": {
23           "kind": "Pod",
24           "namespace": "awx",
25           "name": "web-0",
26           "uid": "7fa21492-d0f5-4840-8517-a05ed04651a4",
27           "resourceVersion": "14843008"
28         }
29       }
30     ]
31   }
32 ]
```



```

29     }
30   },
31   {
32     "ip": "10.244.217.11",
33     "hostname": "web-2",
34     "nodeName": "vms155.liruilongs.github.io",
35     "targetRef": {
36       "kind": "Pod",
37       "namespace": "awx",
38       "name": "web-2",
39       "uid": "7b262352-b12c-4ad2-8d83-8143c71e8c27",
40       "resourceVersion": "14843135"
41     }
42   }
43 ],
44 "ports": [
45   {
46     "name": "nginx-web-headless",
47     "port": 80,
48     "protocol": "TCP"
49   }
50 ]
51 }
]

```

如果使用的 1.21 以及之后的版本，我们可以通过 `EndpointSlicp` 来获取对应的 pod 列表

代码语言: javascript

 复制

```

1  [root@vms81.liruilongs.github.io]-[~/ansible]
2  └─$kubectl describe endpointslices web-headless-888xr
3  Name:          web-headless-888xr
4  Namespace:     awx
5  Labels:        app=nginx_headless
6                 endpointslice.kubernetes.io/managed-by=endpointslice-controller.k8s.io
7                 kubernetes.io/service-name=web-headless
8                 service.kubernetes.io/headless=
9

```

```

10 Annotations: endpoints.kubernetes.io/last-change-trigger-time: 2022-12-09T07:13:15Z
11 AddressType: IPv4
12 Ports:
13   Name          Port  Protocol
14   ----          -
15   nginx-web-headless 80    TCP
16 Endpoints:
17 - Addresses: 10.244.217.10
18   Conditions:
19     Ready: true
20     Hostname: web-0
21     TargetRef: Pod/web-0
22     NodeName: vms155.liruilongs.github.io
23     Zone: <unset>
24 - Addresses: 10.244.194.67
25   Conditions:
26     Ready: true
27     Hostname: web-1
28     TargetRef: Pod/web-1
29     NodeName: vms156.liruilongs.github.io
30     Zone: <unset>
31 - Addresses: 10.244.217.11
32   Conditions:
33     Ready: true
34     Hostname: web-2
35     TargetRef: Pod/web-2
36     NodeName: vms155.liruilongs.github.ioweb
37     Zone: <unset>
Events: <none>

```

通过 curl 调用对应的 REST 接口

代码语言: javascript

 复制

```

1 [root@vms81.liruilongs.github.io]-[~/ansible]
2 $curl -s http://localhost:30021/apis/discovery.k8s.io/v1/namespaces/awx/endpointslices/web-headless-888xr | jq .endpoints
3 [
4

```

```
5 {
6   "addresses": [
7     "10.244.217.10"
8   ],
9   "conditions": {
10    "ready": true,
11    "serving": true,
12    "terminating": false
13  },
14  "hostname": "web-0",
15  "targetRef": {
16    "kind": "Pod",
17    "namespace": "awx",
18    "name": "web-0",
19    "uid": "7fa21492-d0f5-4840-8517-a05ed04651a4",
20    "resourceVersion": "14843008"
21  },
22  "nodeName": "vms155.liruilongs.github.io"
23 },
24 {
25   "addresses": [
26     "10.244.194.67"
27   ],
28   "conditions": {
29    "ready": true,
30    "serving": true,
31    "terminating": false
32  },
33  "hostname": "web-1",
34  "targetRef": {
35    "kind": "Pod",
36    "namespace": "awx",
37    "name": "web-1",
38    "uid": "d71722db-1c41-44ee-a55d-0042c7d2086e",
39    "resourceVersion": "14843070"
40  },
41  "nodeName": "vms156.liruilongs.github.io"
42 },
43 {
```

```
44     "addresses": [  
45         "10.244.217.11"  
46     ],  
47     "conditions": {  
48         "ready": true,  
49         "serving": true,  
50         "terminating": false  
51     },  
52     "hostname": "web-2",  
53     "targetRef": {  
54         "kind": "Pod",  
55         "namespace": "awx",  
56         "name": "web-2",  
57         "uid": "7b262352-b12c-4ad2-8d83-8143c71e8c27",  
58         "resourceVersion": "14843135"  
59     },  
60     "nodeName": "vms155.liruilongs.github.io"  
61 }  
]
```

集群内获取 Headless Service 的 Pod 列表

对于无头服务，客户端可以通过连接到服务的 DNS 名称来连接到其 pod，就像使用常规服务一样，因为 DNS 返回 pod 的 IP，客户端直接连接到 pod，所以不是通过服务代理。这里通过 [DNS 解析](#) 获取的 Pod 列表，Headless 服务仍然提供跨 Pod 的负载均衡，但这仅仅是通过 DNS 循环机制实现的负载均衡。而不是 `sessionAffinity` 相关配置

可以通过 对 服务的 DNS 解析来获取 POD 列表。

创建一个测试 Pod

代码语言: javascript

 复制

```
1  [root@vms81.liruilongs.github.io]~[~/ansible]  
2  $kubectl run tmp01 --image=tutum/dnsutils -- sleep infinity  
3  pod/tmp01 created
```

同一命令空间获取 headless Service 的 Pod 列表

代码语言: javascript

复制

```
1  [root@vms81.liruilongs.github.io]-[~/ansible]
2  └─$kubectl exec tmp01 -it -- /bin/bash
3  root@tmp01:/# nslookup web-headless
4  Server:      10.96.0.10
5  Address:     10.96.0.10#53
6
7  Name:   web-headless.awx.svc.cluster.local
8  Address: 10.244.194.67
9  Name:   web-headless.awx.svc.cluster.local
10 Address: 10.244.217.10
11 Name:   web-headless.awx.svc.cluster.local
12 Address: 10.244.217.11
```

不同命名空间获取 headless Service 的 Pod 列表

代码语言: javascript

复制

```
1  [root@vms81.liruilongs.github.io]-[~/ansible]
2  └─$kubectl exec tmp01 -it -- /bin/bash
3  root@tmp01:/# nslookup web-headless.awx
4  Server:      10.96.0.10
5  Address:     10.96.0.10#53
6
7  Name:   web-headless.awx.svc.cluster.local
8  Address: 10.244.217.11
9  Name:   web-headless.awx.svc.cluster.local
10 Address: 10.244.194.67
11 Name:   web-headless.awx.svc.cluster.local
12 Address: 10.244.217.10
13
14 root@tmp01:/# nslookup web-headless.awx.svc.cluster.local.
15 Server:      10.96.0.10
16 Address:     10.96.0.10#53
17
18 Name:   web-headless.awx.svc.cluster.local
```

```
19 Address: 10.244.217.11
20 Name: web-headless.awx.svc.cluster.local
21 Address: 10.244.194.67
22 Name: web-headless.awx.svc.cluster.local
23 Address: 10.244.217.10
```

无状态的 Headless 服务

关于无状态的 Headless Service 这里我们也简单介绍，和，有状态的没什么区别，对应的 SVC 还是用之前的 `web-headless`，在 `StatefulSet` 的基础上，我们创建一个 `deployment` 提供服务能力

代码语言: javascript

 复制

```
1 [root@vms81.liruilongs.github.io]-[~/ansible]
2 └─$cat deploy.yaml
3 apiVersion: apps/v1
4 kind: Deployment
5 metadata:
6   creationTimestamp: null
7   labels:
8     app: web-headless
9   name: web
10 spec:
11   replicas: 3
12   selector:
13     matchLabels:
14       app: web-headless
15   strategy: {}
16   template:
17     metadata:
18       creationTimestamp: null
19     labels:
20       app: web-headless
21     spec:
22       containers:
23       - image: nginx
24         name: nginx-web
25       ports:
```

```
26     - containerPort: 80
27     name: nginx-web
28     resources: {}
29 status: {}
```

通过 DNS 获取IP, 可以发现, 对应的 A 记录增加到 6个

代码语言: javascript

 复制

```
1  [root@vms81.liruilongs.github.io]--[~/ansible]
2  └─$kubectl exec tmp01 -it -- /bin/bash
3  root@tmp01:/# nslookup web-headless.awx
4  Server:      10.96.0.10
5  Address:     10.96.0.10#53
6
7  Name:   web-headless.awx.svc.cluster.local
8  Address: 10.244.194.67
9  Name:   web-headless.awx.svc.cluster.local
10 Address: 10.244.217.11
11 Name:   web-headless.awx.svc.cluster.local
12 Address: 10.244.217.10
13 Name:   web-headless.awx.svc.cluster.local
14 Address: 10.244.194.69
15 Name:   web-headless.awx.svc.cluster.local
16 Address: 10.244.194.70
17 Name:   web-headless.awx.svc.cluster.local
18 Address: 10.244.217.12
19
20 root@tmp01:/# exit
21 exit
```

关于 [Headless Service](#) 和小伙伴分享到这里, 通过 无头服务, 我们可以通过 Servcie 来动态感知 Pod 副本的变化, 监听 Pod 的状态, 实现部分分布式集群的动态构建, 同时在有状态应用中都会涉及 [Headless Service](#) 。

博文参考

<https://kubernetes.io/zh-cn/docs/concepts/services-networking/service/#headless-services>

<https://stackoverflow.com/questions/52707840/what-is-a-headless-service-what-does-it-do-accomplish-and-what-are-some-legiti>

<https://cloud.tencent.com/developer/article/1638722>

<https://kubernetes.io/docs/reference/kubernetes-api/service-resources/endpoints-v1/>

<https://kubernetes.io/docs/reference/kubernetes-api/service-resources/endpoint-slice-v1/>

本文参与 [腾讯云自媒体同步曝光计划](#)，分享自微信公众号。

原始发表：2022-12-27，如有侵权请联系 cloudcommunity@tencent.com 删除



dns

tcp/ip

https

网络安全

负载均衡