

Kubernetes 网络和负载均衡

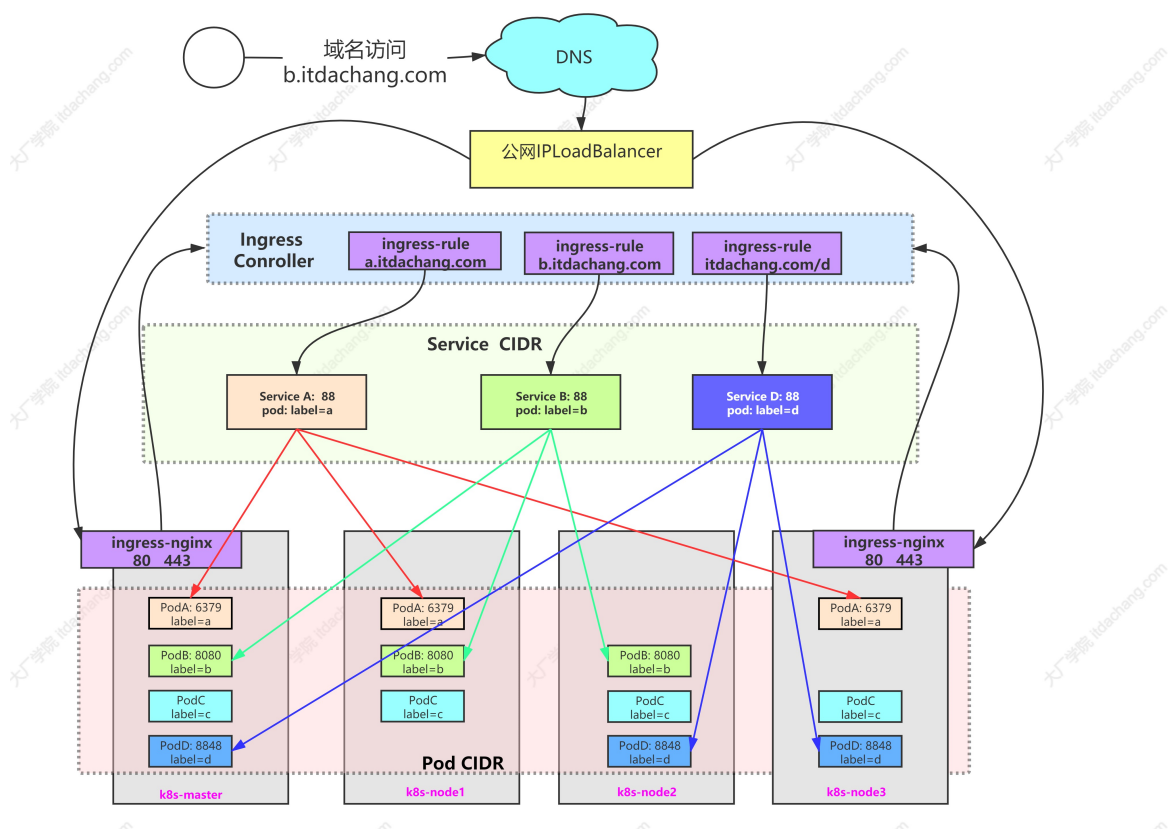
一、Kubernetes网络

Kubernetes 网络解决四方面的问题：

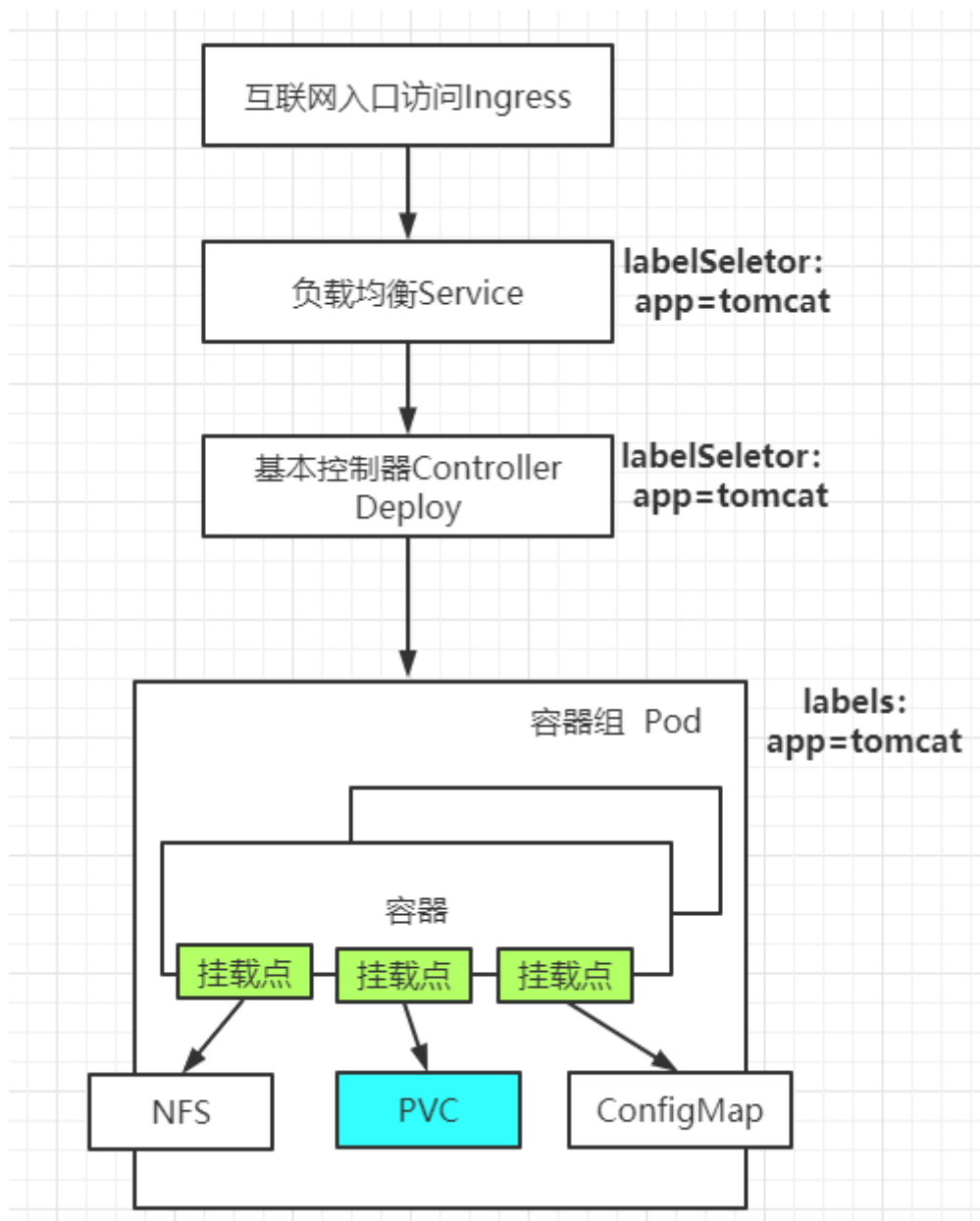
- 一个 Pod 中的容器之间通过**本地回路 (loopback) 通信**。
- 集群网络在不同 pod 之间提供通信。Pod和Pod之间互通
- Service 资源允许你对外暴露 Pods 中运行的应用程序，以支持来自于集群外部的访问。Service和 Pod要通
- 可以使用 Services 来发布仅供集群内部使用的服务。

1、k8s网络架构图

1、架构图

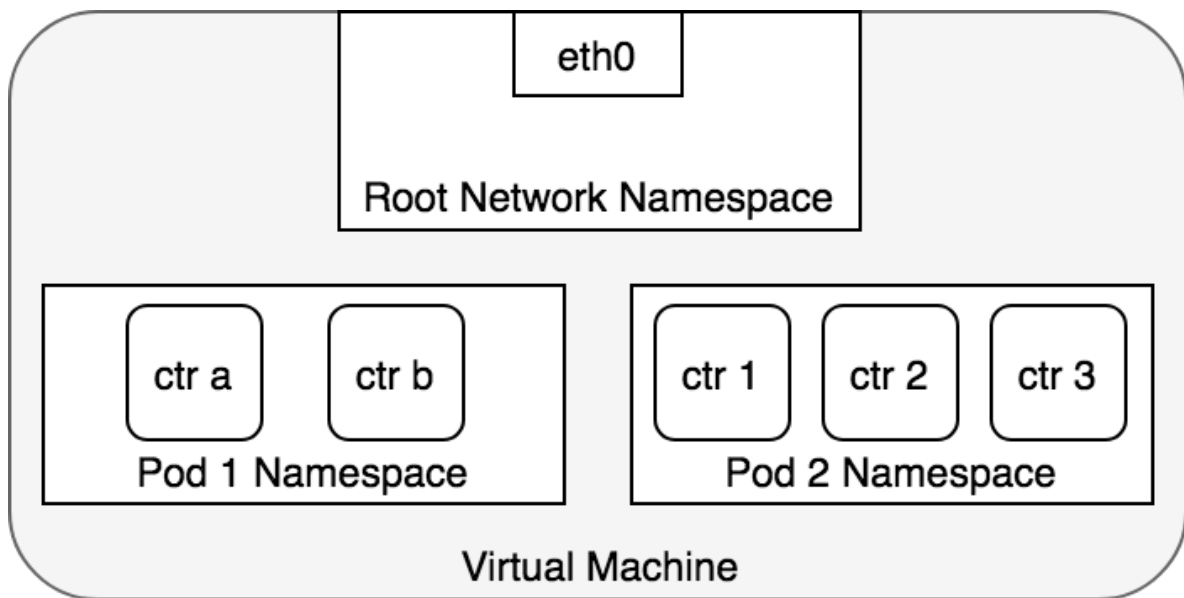


2、访问流程



2、网络连通原理

1、Container To Container



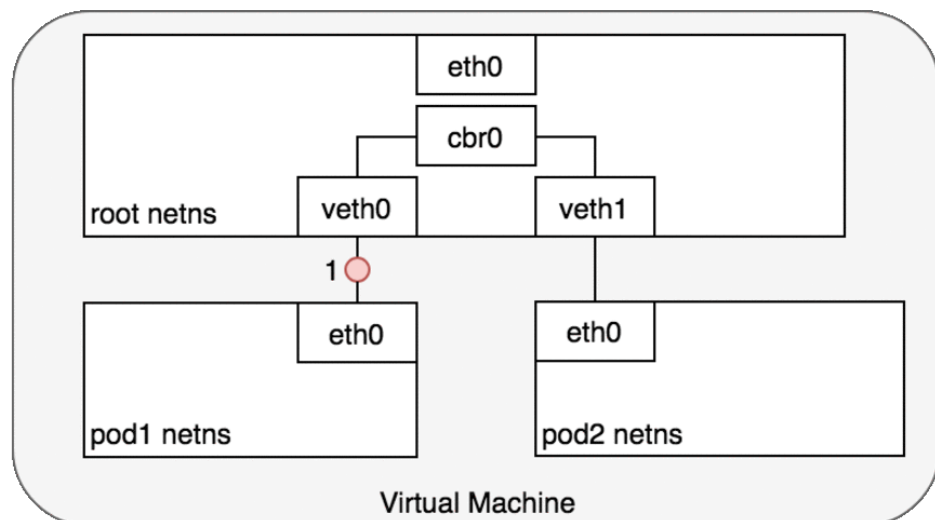
```

1  ip netns add ns1 #添加网络名称空间
2  ls /var/run/netns #查看所有网络名词空间
3  ip netns        #查看所有网络名词空间
4  # Linux 将所有的进程都分配到 root network namespace, 以使得进程可以访问外部网络
5  # Kubernetes 为每一个 Pod 都创建了一个 network namespace

```

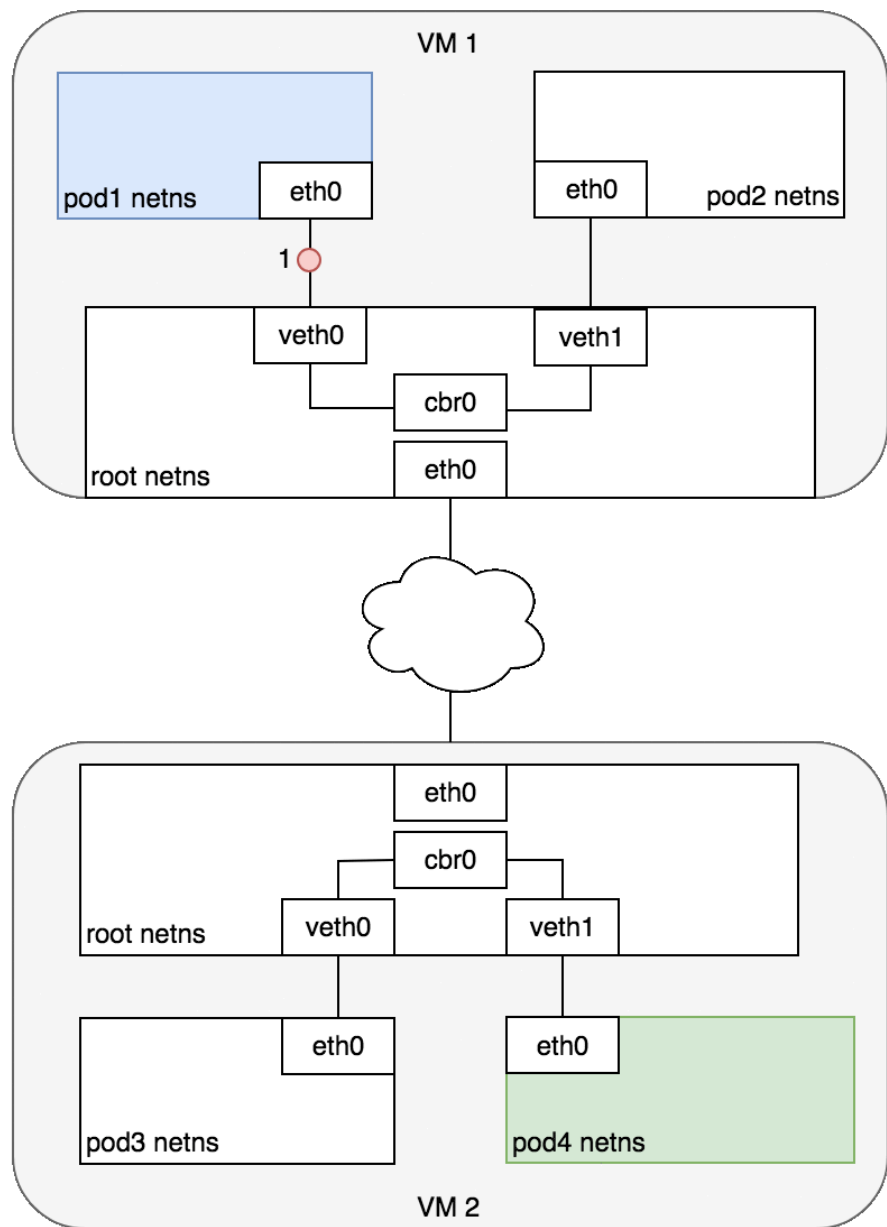
2、Pod To Pod

1、同节点



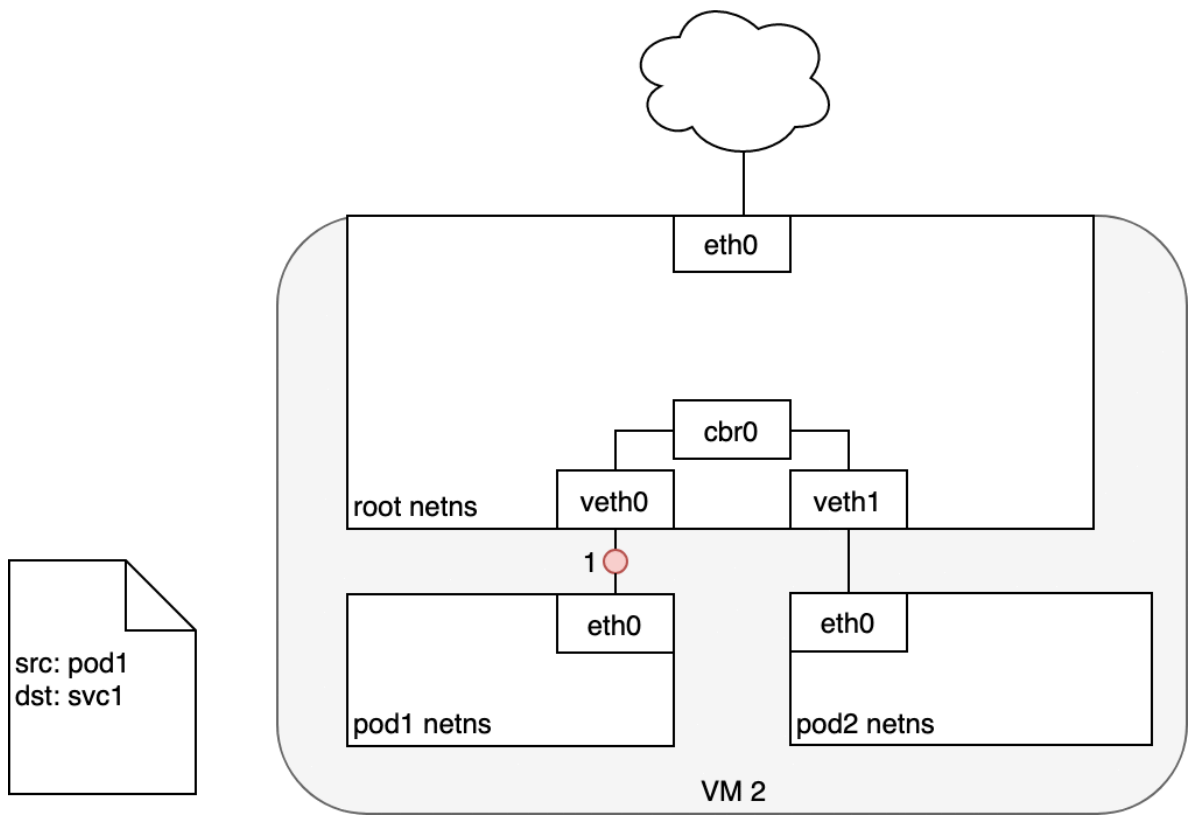
2、跨节点

src: pod1
dst: pod4

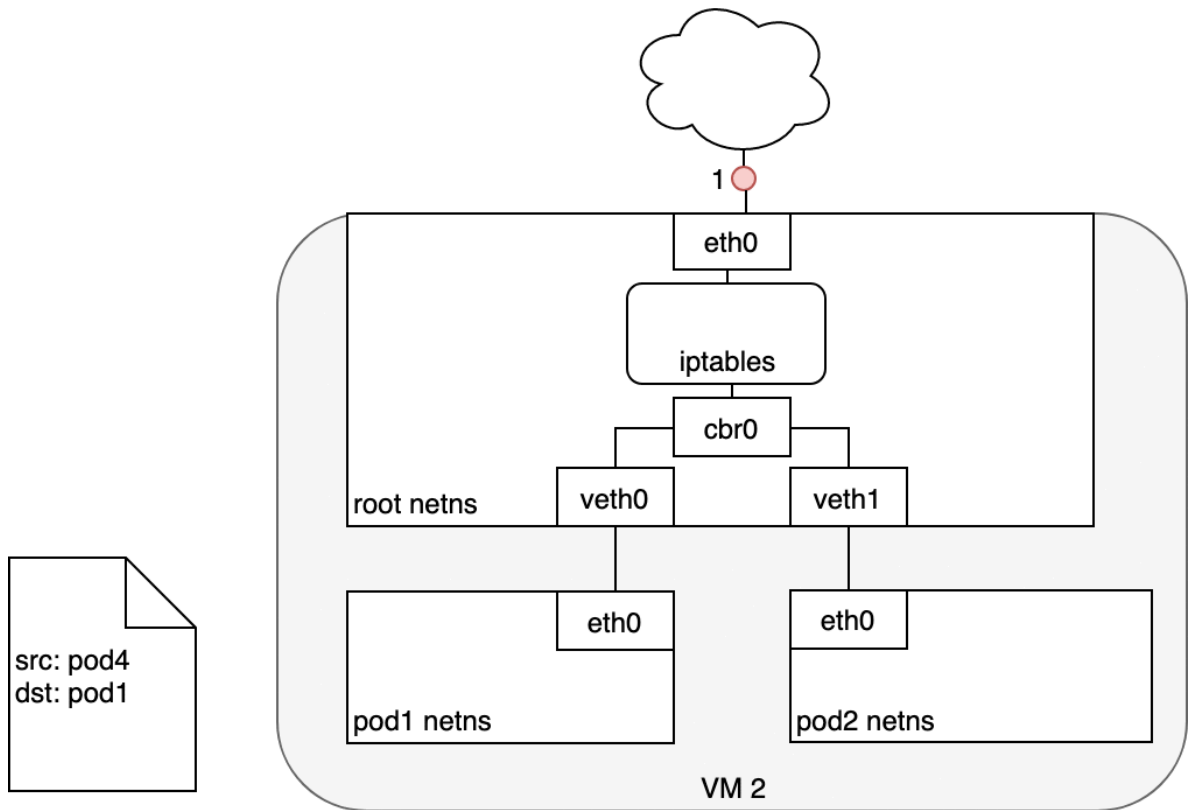


3、 Pod-To-Service

1、 Pod To Service

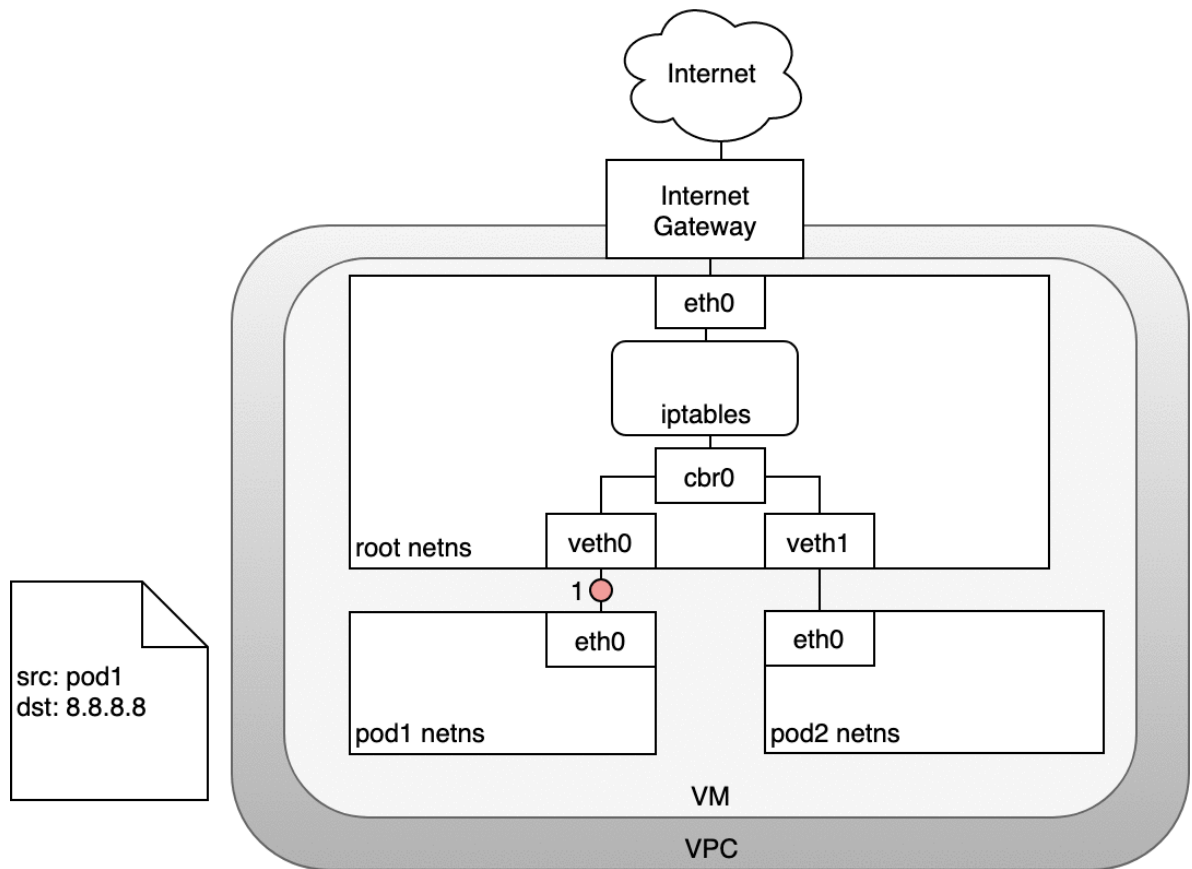


2、Service-To-Pod

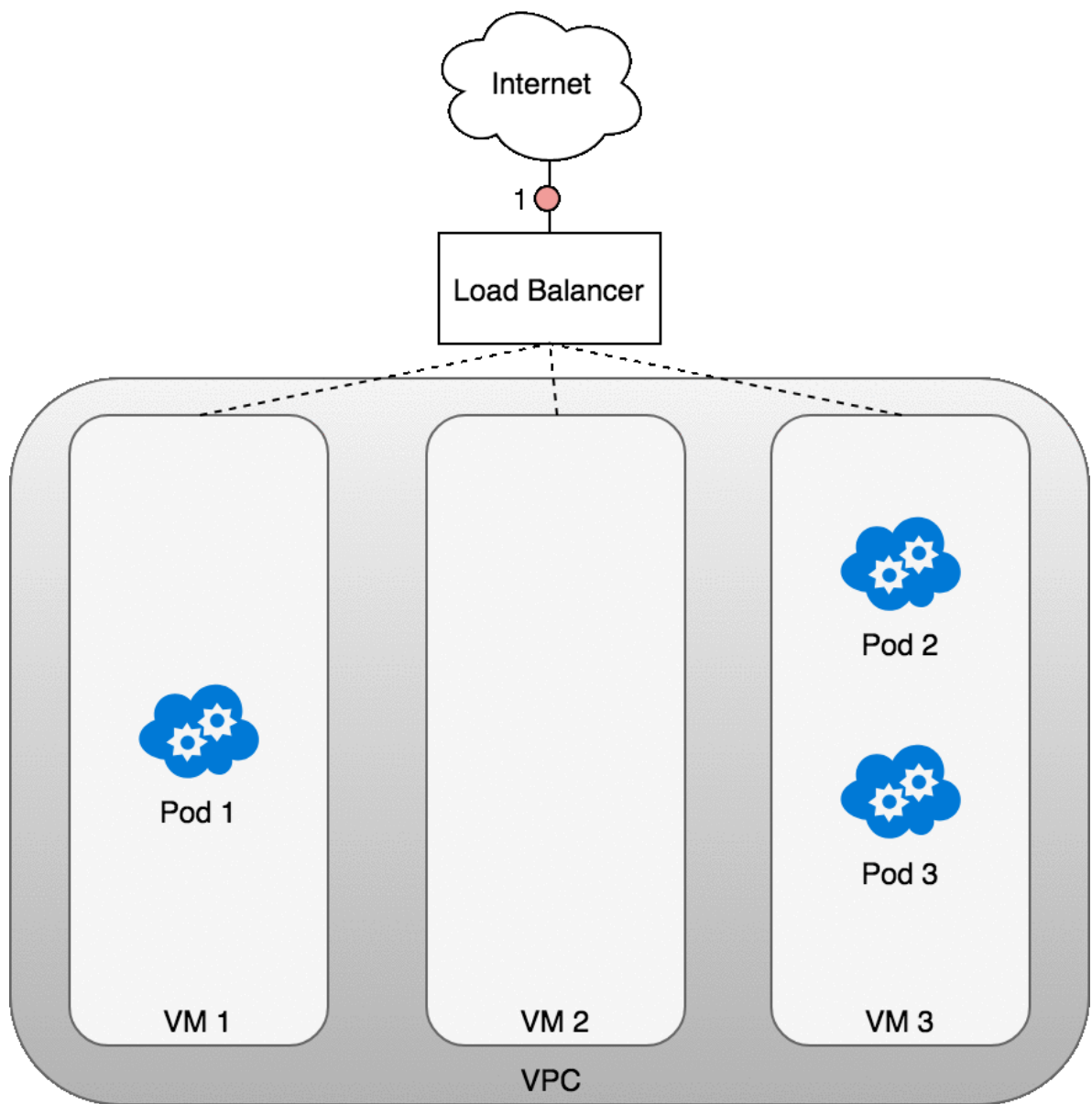


4、Internet-To-Service

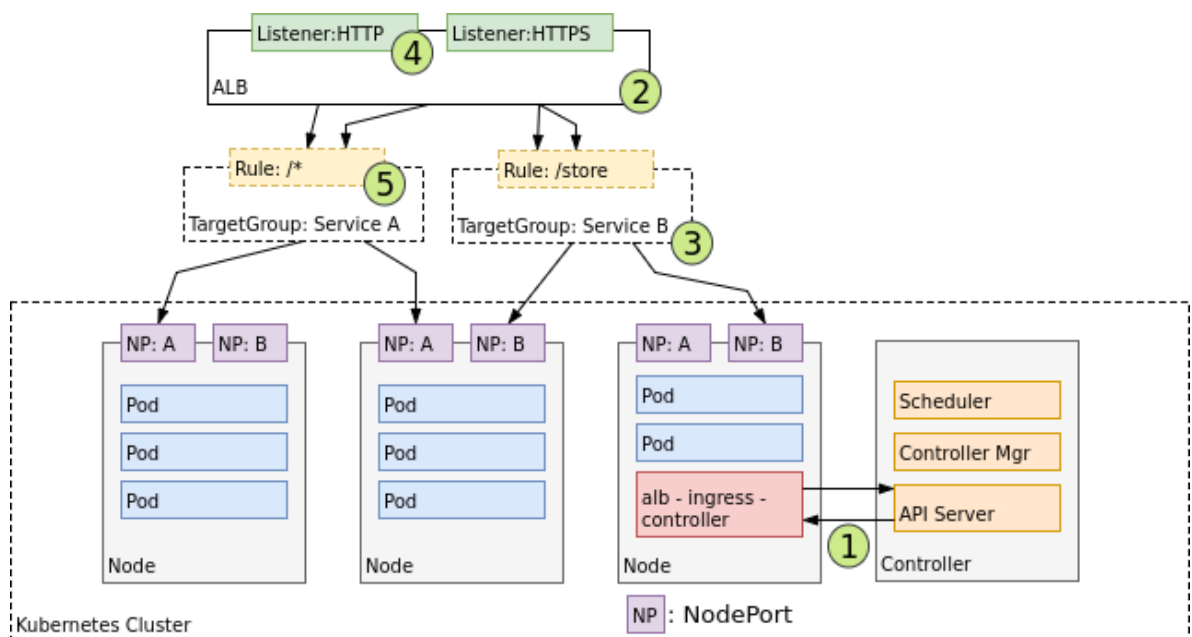
1、Pod-To-Internet



2、Internet-To-Pod (LoadBalancer -- Layer4)



3、Internet-To-Pod (Ingress-- Layer7)



二、Service

负载均衡服务。让一组Pod可以被别人进行服务发现。

Service ---> 选择一组Pod

别人只需要访问这个Service。Service还会基于Pod的探针机制（ReadinessProbe：就绪探针）完成Pod的自动剔除和上线工作。

- Service即使无头服务。别人（Pod）不能用ip访问，但是可以用service名当成域名访问。
- Service的名字还能当成域名被Pod解析

1、基础概念

将运行在一组 Pods 上的应用程序公开为网络服务的抽象方法。

云原生服务发现

service中的type可选值如下，代表四种不同的服务发现类型

- ExternalName
 - ClusterIP: 为当前Service分配或者不分配集群IP。负载均衡一组Pod
 - NodePort: 外界也可以使用机器ip+暴露的NodePort端口 访问。
 - nodePort端口由kube-proxy开在机器上
 - 机器ip+暴露的NodePort 流量先来到 kube-proxy
 - LoadBalancer.
- **ClusterIP** : 通过集群的内部 IP 暴露服务，选择该值时服务只能够在集群内部访问。这也是默认的 **ServiceType** 。
 - **NodePort** : 通过每个节点上的 IP 和静态端口（**NodePort**）暴露服务。**NodePort** 服务会路由到自动创建的 **ClusterIP** 服务。通过请求 **<节点 IP>:<节点端口>**，你可以从集群的外部访问一个 **NodePort** 服务。
 - **LoadBalancer** : 使用云提供商的负载均衡器向外部暴露服务。外部负载均衡器可以将流量路由到自动创建的 **NodePort** 服务和 **ClusterIP** 服务上。
 - **ExternalName** : 通过返回 **CNAME** 和对应值，可以将服务映射到 **externalName** 字段的内容（例如，**foo.bar.example.com**）。无需创建任何类型代理。

1、创建简单Service


```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: my-service
5  spec:
6    selector:
7      app: MyApp    ## 使用选择器选择所有Pod
8    # type: ClusterIP  ##type很重要，不写默认是ClusterIP
9    ports:
10     - protocol: TCP
11       port: 80
12       targetPort: 9376

```

- Service 创建完成后，会对应一组EndPoint。可以kubectl get ep 进行查看
- type有四种，每种对应不同服务发现机制
- Service可以利用Pod的就绪探针机制，只负载就绪了的Pod。自动剔除没有就绪的Pod

2、创建无Selector的Service

- 我们可以创建Service不指定Selector
- 然后手动创建EndPoint，指定一组Pod地址。
- 此场景用于我们负载均衡其他中间件场景。

```

1  # 无selector的svc
2  apiVersion: v1
3  kind: Service
4  metadata:
5    name: my-service-no-selector
6  spec:
7    ports:
8     - protocol: TCP
9       name: http  ###一定注意，name可以不写，
10       ###但是这里如果写了name，那么endpoint里面的ports必须有同名name才能绑定
11       port: 80 # service 80
12       targetPort: 80 #目标80
13 ---
14 apiVersion: v1
15 kind: Endpoints
16 metadata:
17   name: my-service-no-selector  ### ep和svc的绑定规则是：和svc同名同名称空间，port同名
18   namespace: default
19   subsets:
20   - addresses:
21     - ip: 220.181.38.148
22     - ip: 39.156.69.79
23     - ip: 192.168.169.165
24     ports:
25     - port: 80
26       name: http  ## svc有name这里一定要有
27     protocol: TCP

```

原理：kube-proxy 在负责这个事情

```
1  ## 实验
2  apiVersion: v1
3  kind: Service
4  metadata:
5    name: cluster-service-no-selector
6    namespace: default
7  spec:
8    ## 不选中Pod而在下面手动定义可以访问的EndPoint
9    type: ClusterIP
10   ports:
11     - name: abc
12       port: 80  ## 访问当前service 的 80
13       targetPort: 80  ## 派发到Pod的 80
14   ---
15   apiVersion: v1
16   kind: Endpoints
17   metadata:
18     name: cluster-service-no-selector  ## 和service同名
19     namespace: default
20   subsets:
21     - addresses:
22       - ip: 192.168.169.184
23       - ip: 192.168.169.165
24       - ip: 39.156.69.79
25     ports:
26       - name: abc  ## ep和service要是一样的
27         port: 80
28         protocol: TCP
```

场景：Pod要访问 MySQL。MySQL单独部署到很多机器，每次记ip麻烦

集群内创建一个Service，实时的可以剔除EP信息。反向代理集群外的东西。

2、ClusterIP

```
1  type: ClusterIP
2  ClusterIP: 手动指定/None/""
```

- 手动指定的ClusterIP必须在合法范围内
- None会创建出没有ClusterIP的headless service（无头服务），Pod需要用服务的域名访问

3、NodePort

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: my-service
5    namespace: default
6  type: NodePort
7  ports:
8    - protocol: TCP
9      port: 80 # service 80
10     targetPort: 80 #目标80
11     nodePort: 32123 #自定义

```

- 如果将 `type` 字段设置为 `NodePort`，则 Kubernetes 将在 `--service-node-port-range` 标志指定的范围内分配端口（默认值：30000-32767）
- k8s集群的所有机器都将打开监听这个端口的数据，访问任何一个机器，都可以访问这个service对应的Pod
- 使用 `nodePort` 自定义端口

4、ExternalName

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: my-service-05
5    namespace: default
6  spec:
7    type: ExternalName
8    externalName: baidu.com

```

- 其他的Pod可以通过访问这个service而访问其他的域名服务
- 但是需要注意目标服务的跨域问题

5、LoadBalancer

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    creationTimestamp: null
5    labels:
6      app.kubernetes.io/name: load-balancer-example
7    name: my-service
8  spec:
9    ports:
10     - port: 80
11       protocol: TCP
12       targetPort: 80

```

```
13     selector:
14         app.kubernetes.io/name: load-balancer-example
15     type: LoadBalancer
```

6、扩展 - externalIP

在 Service 的定义中，`externalIPs` 可以和任何类型的 `.spec.type` 一通使用。在下面的例子中，客户端可通过 `80.11.12.10:80` (`externalIP:port`) 访问 `my-service`

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4      name: my-service-externalip
5  spec:
6      selector:
7          app: canary-nginx
8      ports:
9          - name: http
10            protocol: TCP
11            port: 80
12            targetPort: 80
13      externalIPs: ### 定义只有externalIPs指定的地址才可以访问这个service
14          - 10.170.0.111
```

7、扩展 - Pod的DNS

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4      name: default-subdomain
5  spec:
6      selector:
7          name: busybox
8      clusterIP: None
9      ports:
10         - name: foo # 实际上不需要指定端口号
11           port: 1234
12           targetPort: 1234
13  ---
14  apiVersion: v1
15  kind: Pod
16  metadata:
17      name: busybox1
18      labels:
19          name: busybox
20  spec:
21      hostname: busybox-1
22      subdomain: default-subdomain
```

```

23     ## 指定必须和svc名称一样，才可以 podName.subdomain.名称空间.svc.cluster.local访问。否
    则访问不同指定Pod
24     containers:
25     - image: busybox:1.28
26       command:
27         - sleep
28         - "3600"
29       name: busybox
30     ---
31     apiVersion: v1
32     kind: Pod
33     metadata:
34       name: busybox2
35       labels:
36         name: busybox
37     spec:
38       hostname: busybox-2
39       subdomain: default-subdomain
40       containers:
41       - image: busybox:1.28
42         command:
43           - sleep
44           - "3600"
45       name: busybox

```

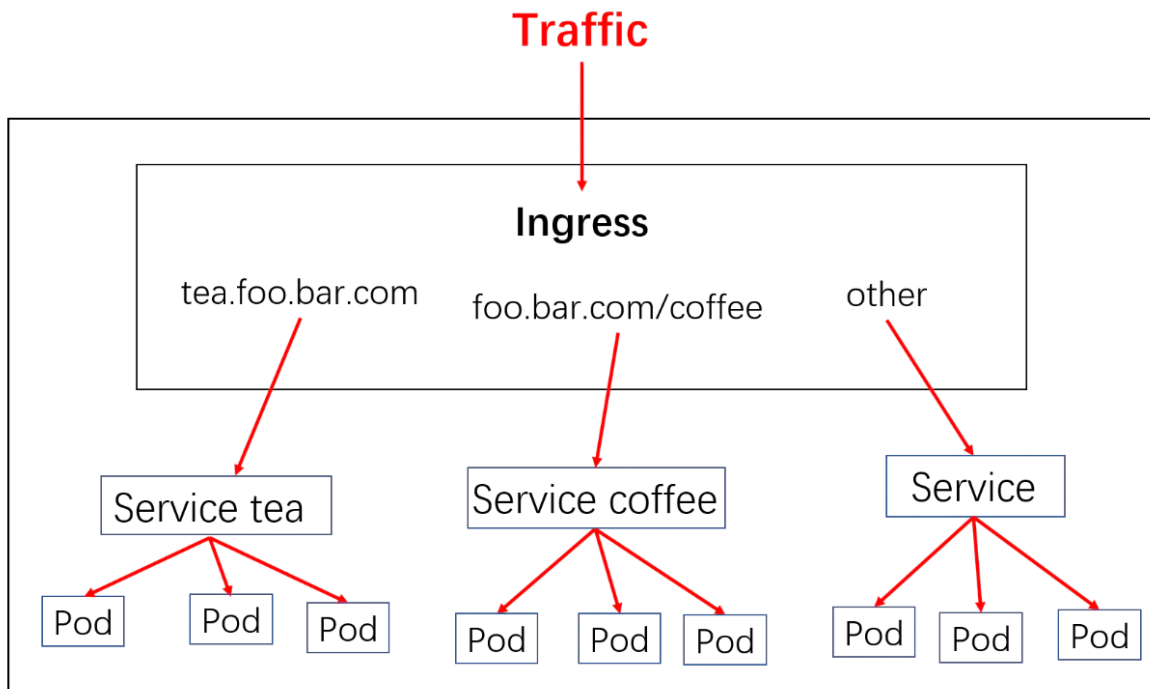
- 访问 `busybox-1.default-subdomain.default.svc.cluster.local` 可以访问到busybox-1。
- 访问Service
 - 同名称空间
 - ping service-name 即可
 - 不同名称空间
 - ping service-name.namespace 即可
- 访问Pod
 - 同名称空间
 - ping pod-host-name.service-name 即可
 - 不同名称空间
 - ping pod-host-name.service-name.namespace 即可

三、Ingress

为什么需要Ingress?

- Service可以使用NodePort暴露集群外访问端口，但是性能低下不安全
- 缺少Layer7的统一访问入口，可以负载均衡、限流等

- **Ingress** 公开了从集群外部到集群内 **服务** 的 HTTP 和 HTTPS 路由。流量路由由 Ingress 资源上定义的规则控制。
- 我们使用 Ingress 作为整个集群统一的入口，配置 Ingress 规则转到对应的 Service

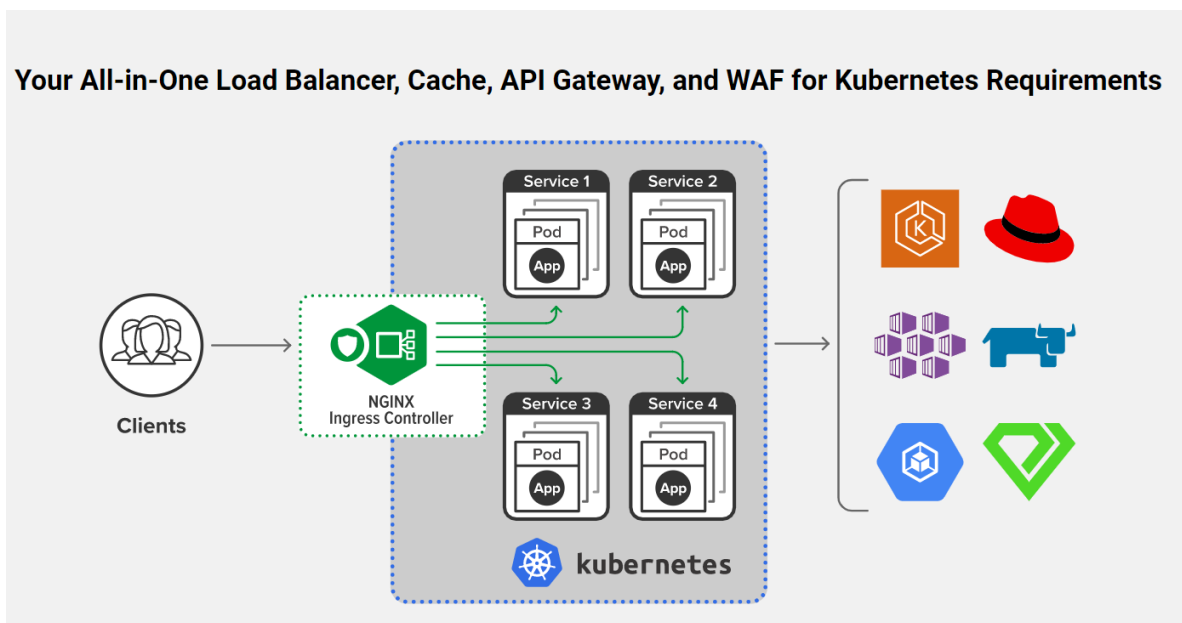


1、Ingress nginx和Nginx ingress

1、nginx ingress

这是nginx官方做的，适配k8s的，分为**开源版**和**nginx plus版（收费）**。 [文档地址](https://www.nginx.com/products/nginx-ingress-controller)

<https://www.nginx.com/products/nginx-ingress-controller>



2、ingress nginx

<https://kubernetes.io/zh/docs/concepts/services-networking/ingress/#ingress-%E6%98%AF%E4%BB%80%E4%B9%88>

这是k8s官方做的，适配nginx的。这个里面会及时更新一些特性，而且性能很高，也被广泛采用。[文档地址](#)

```
1  ## 默认安装使用这个镜像
2  registry.cn-hangzhou.aliyuncs.com/lfy_k8s_images/ingress-nginx-controller:v0.46.0
```

2、ingress nginx 安装

1、安装

自建集群使用 [裸金属安装方式](#)

需要如下修改：

- 修改ingress-nginx-controller镜像为 registry.cn-hangzhou.aliyuncs.com/lfy_k8s_images/ingress-nginx-controller:v0.46.0
- 修改Deployment为DaemonSet比较好
- 修改Container使用主机网络，直接在主机上开辟 80,443端口，无需中间解析，速度更快
- Container使用主机网络，对应的dnsPolicy策略也需要改为主机网络的
- 修改Service为ClusterIP，无需NodePort模式了
- 修改DaemonSet的nodeSelector: `ingress-node=true` 。这样只需要给node节点打上 `ingress-node=true` 标签，即可快速的加入/剔除 ingress-controller的数量

修改好的yaml如下。大家直接复制使用

```
1
2  apiVersion: v1
3  kind: Namespace
4  metadata:
5    name: ingress-nginx
6    labels:
7      app.kubernetes.io/name: ingress-nginx
8      app.kubernetes.io/instance: ingress-nginx
9
10 ---
11 # Source: ingress-nginx/templates/controller-serviceaccount.yaml
12 apiVersion: v1
13 kind: ServiceAccount
14 metadata:
15   labels:
16     helm.sh/chart: ingress-nginx-3.30.0
17     app.kubernetes.io/name: ingress-nginx
18     app.kubernetes.io/instance: ingress-nginx
19     app.kubernetes.io/version: 0.46.0
20     app.kubernetes.io/managed-by: Helm
21     app.kubernetes.io/component: controller
22   name: ingress-nginx
```

```

23     namespace: ingress-nginx
24     automountServiceAccountToken: true
25     ---
26     # Source: ingress-nginx/templates/controller-configmap.yaml
27     apiVersion: v1
28     kind: ConfigMap
29     metadata:
30       labels:
31         helm.sh/chart: ingress-nginx-3.30.0
32         app.kubernetes.io/name: ingress-nginx
33         app.kubernetes.io/instance: ingress-nginx
34         app.kubernetes.io/version: 0.46.0
35         app.kubernetes.io/managed-by: Helm
36         app.kubernetes.io/component: controller
37       name: ingress-nginx-controller
38       namespace: ingress-nginx
39     data:
40     ---
41     # Source: ingress-nginx/templates/clusterrole.yaml
42     apiVersion: rbac.authorization.k8s.io/v1
43     kind: ClusterRole
44     metadata:
45       labels:
46         helm.sh/chart: ingress-nginx-3.30.0
47         app.kubernetes.io/name: ingress-nginx
48         app.kubernetes.io/instance: ingress-nginx
49         app.kubernetes.io/version: 0.46.0
50         app.kubernetes.io/managed-by: Helm
51       name: ingress-nginx
52     rules:
53       - apiGroups:
54         - ''
55         resources:
56           - configmaps
57           - endpoints
58           - nodes
59           - pods
60           - secrets
61         verbs:
62           - list
63           - watch
64       - apiGroups:
65         - ''
66         resources:
67           - nodes
68         verbs:
69           - get
70       - apiGroups:
71         - ''
72         resources:
73           - services
74         verbs:
75           - get
76           - list
77           - watch
78       - apiGroups:
79         - extensions
80         - networking.k8s.io # k8s 1.14+

```



```

81     resources:
82     - ingresses
83     verbs:
84     - get
85     - list
86     - watch
87 - apiGroups:
88   - ''
89     resources:
90     - events
91     verbs:
92     - create
93     - patch
94 - apiGroups:
95   - extensions
96   - networking.k8s.io # k8s 1.14+
97     resources:
98     - ingresses/status
99     verbs:
100    - update
101 - apiGroups:
102   - networking.k8s.io # k8s 1.14+
103     resources:
104     - ingressclasses
105     verbs:
106     - get
107     - list
108     - watch
109 ---
110 # Source: ingress-nginx/templates/clusterrolebinding.yaml
111 apiVersion: rbac.authorization.k8s.io/v1
112 kind: ClusterRoleBinding
113 metadata:
114   labels:
115     helm.sh/chart: ingress-nginx-3.30.0
116     app.kubernetes.io/name: ingress-nginx
117     app.kubernetes.io/instance: ingress-nginx
118     app.kubernetes.io/version: 0.46.0
119     app.kubernetes.io/managed-by: Helm
120   name: ingress-nginx
121 roleRef:
122   apiGroup: rbac.authorization.k8s.io
123   kind: ClusterRole
124   name: ingress-nginx
125 subjects:
126 - kind: ServiceAccount
127   name: ingress-nginx
128   namespace: ingress-nginx
129 ---
130 # Source: ingress-nginx/templates/controller-role.yaml
131 apiVersion: rbac.authorization.k8s.io/v1
132 kind: Role
133 metadata:
134   labels:
135     helm.sh/chart: ingress-nginx-3.30.0
136     app.kubernetes.io/name: ingress-nginx
137     app.kubernetes.io/instance: ingress-nginx
138     app.kubernetes.io/version: 0.46.0

```

```
139     app.kubernetes.io/managed-by: Helm
140     app.kubernetes.io/component: controller
141     name: ingress-nginx
142     namespace: ingress-nginx
143 rules:
144   - apiGroups:
145       - ''
146     resources:
147       - namespaces
148     verbs:
149       - get
150   - apiGroups:
151       - ''
152     resources:
153       - configmaps
154       - pods
155       - secrets
156       - endpoints
157     verbs:
158       - get
159       - list
160       - watch
161   - apiGroups:
162       - ''
163     resources:
164       - services
165     verbs:
166       - get
167       - list
168       - watch
169   - apiGroups:
170       - extensions
171       - networking.k8s.io # k8s 1.14+
172     resources:
173       - ingresses
174     verbs:
175       - get
176       - list
177       - watch
178   - apiGroups:
179       - extensions
180       - networking.k8s.io # k8s 1.14+
181     resources:
182       - ingresses/status
183     verbs:
184       - update
185   - apiGroups:
186       - networking.k8s.io # k8s 1.14+
187     resources:
188       - ingressclasses
189     verbs:
190       - get
191       - list
192       - watch
193   - apiGroups:
194       - ''
195     resources:
196       - configmaps
```

```

197     resourceName:
198       - ingress-controller-leader-nginx
199     verbs:
200       - get
201       - update
202   - apiGroups:
203     - ''
204     resources:
205       - configmaps
206     verbs:
207       - create
208   - apiGroups:
209     - ''
210     resources:
211       - events
212     verbs:
213       - create
214       - patch
215 ---
216 # Source: ingress-nginx/templates/controller-rolebinding.yaml
217 apiVersion: rbac.authorization.k8s.io/v1
218 kind: RoleBinding
219 metadata:
220   labels:
221     helm.sh/chart: ingress-nginx-3.30.0
222     app.kubernetes.io/name: ingress-nginx
223     app.kubernetes.io/instance: ingress-nginx
224     app.kubernetes.io/version: 0.46.0
225     app.kubernetes.io/managed-by: Helm
226     app.kubernetes.io/component: controller
227   name: ingress-nginx
228   namespace: ingress-nginx
229 roleRef:
230   apiGroup: rbac.authorization.k8s.io
231   kind: Role
232   name: ingress-nginx
233 subjects:
234   - kind: ServiceAccount
235     name: ingress-nginx
236     namespace: ingress-nginx
237 ---
238 # Source: ingress-nginx/templates/controller-service-webhook.yaml
239 apiVersion: v1
240 kind: Service
241 metadata:
242   labels:
243     helm.sh/chart: ingress-nginx-3.30.0
244     app.kubernetes.io/name: ingress-nginx
245     app.kubernetes.io/instance: ingress-nginx
246     app.kubernetes.io/version: 0.46.0
247     app.kubernetes.io/managed-by: Helm
248     app.kubernetes.io/component: controller
249   name: ingress-nginx-controller-admission
250   namespace: ingress-nginx
251 spec:
252   type: ClusterIP
253   ports:
254     - name: https-webhook

```

```
255     port: 443
256     targetPort: webhook
257     selector:
258       app.kubernetes.io/name: ingress-nginx
259       app.kubernetes.io/instance: ingress-nginx
260       app.kubernetes.io/component: controller
261 ---
262 # Source: ingress-nginx/templates/controller-service.yaml: 不要
263 apiVersion: v1
264 kind: Service
265 metadata:
266   annotations:
267   labels:
268     helm.sh/chart: ingress-nginx-3.30.0
269     app.kubernetes.io/name: ingress-nginx
270     app.kubernetes.io/instance: ingress-nginx
271     app.kubernetes.io/version: 0.46.0
272     app.kubernetes.io/managed-by: Helm
273     app.kubernetes.io/component: controller
274   name: ingress-nginx-controller
275   namespace: ingress-nginx
276 spec:
277   type: ClusterIP ## 改为clusterIP
278   ports:
279     - name: http
280       port: 80
281       protocol: TCP
282       targetPort: http
283     - name: https
284       port: 443
285       protocol: TCP
286       targetPort: https
287   selector:
288     app.kubernetes.io/name: ingress-nginx
289     app.kubernetes.io/instance: ingress-nginx
290     app.kubernetes.io/component: controller
291 ---
292 # Source: ingress-nginx/templates/controller-deployment.yaml
293 apiVersion: apps/v1
294 kind: DaemonSet
295 metadata:
296   labels:
297     helm.sh/chart: ingress-nginx-3.30.0
298     app.kubernetes.io/name: ingress-nginx
299     app.kubernetes.io/instance: ingress-nginx
300     app.kubernetes.io/version: 0.46.0
301     app.kubernetes.io/managed-by: Helm
302     app.kubernetes.io/component: controller
303   name: ingress-nginx-controller
304   namespace: ingress-nginx
305 spec:
306   selector:
307     matchLabels:
308       app.kubernetes.io/name: ingress-nginx
309       app.kubernetes.io/instance: ingress-nginx
310       app.kubernetes.io/component: controller
311   revisionHistoryLimit: 10
312   minReadySeconds: 0
```

```

313     template:
314         metadata:
315             labels:
316                 app.kubernetes.io/name: ingress-nginx
317                 app.kubernetes.io/instance: ingress-nginx
318                 app.kubernetes.io/component: controller
319         spec:
320             dnsPolicy: ClusterFirstWithHostNet    ## dns对应调整为主机网络
321             hostNetwork: true    ## 直接让nginx占用本机80端口和443端口，所以使用主机网络
322             containers:
323                 - name: controller
324                   image: registry.cn-hangzhou.aliyuncs.com/lfy_k8s_images/ingress-
nginx-controller:v0.46.0
325                   imagePullPolicy: IfNotPresent
326                   lifecycle:
327                       preStop:
328                           exec:
329                               command:
330                                   - /wait-shutdown
331                   args:
332                       - /nginx-ingress-controller
333                       - --election-id=ingress-controller-leader
334                       - --ingress-class=nginx
335                       - --configmap=$(POD_NAMESPACE)/ingress-nginx-controller
336                       - --validating-webhook=:8443
337                       - --validating-webhook-certificate=/usr/local/certificates/cert
338                       - --validating-webhook-key=/usr/local/certificates/key
339                   securityContext:
340                       capabilities:
341                           drop:
342                               - ALL
343                           add:
344                               - NET_BIND_SERVICE
345                       runAsUser: 101
346                       allowPrivilegeEscalation: true
347                   env:
348                       - name: POD_NAME
349                         valueFrom:
350                             fieldRef:
351                                 fieldPath: metadata.name
352                       - name: POD_NAMESPACE
353                         valueFrom:
354                             fieldRef:
355                                 fieldPath: metadata.namespace
356                       - name: LD_PRELOAD
357                         value: /usr/local/lib/libmimalloc.so
358                   livenessProbe:
359                       httpGet:
360                           path: /healthz
361                           port: 10254
362                           scheme: HTTP
363                       initialDelaySeconds: 10
364                       periodSeconds: 10
365                       timeoutSeconds: 1
366                       successThreshold: 1
367                       failureThreshold: 5
368                   readinessProbe:
369                       httpGet:

```

```

370         path: /healthz
371         port: 10254
372         scheme: HTTP
373         initialDelaySeconds: 10
374         periodSeconds: 10
375         timeoutSeconds: 1
376         successThreshold: 1
377         failureThreshold: 3
378     ports:
379     - name: http
380       containerPort: 80
381       protocol: TCP
382     - name: https
383       containerPort: 443
384       protocol: TCP
385     - name: webhook
386       containerPort: 8443
387       protocol: TCP
388     volumeMounts:
389     - name: webhook-cert
390       mountPath: /usr/local/certificates/
391       readOnly: true
392     resources:
393       requests:
394         cpu: 100m
395         memory: 90Mi
396     nodeSelector:
397       node-role: ingress #以后只需要给某个node打上这个标签就可以部署ingress-nginx到
这个节点上了
398       #kubernetes.io/os: linux ## 修改节点选择
399     serviceAccountName: ingress-nginx
400     terminationGracePeriodSeconds: 300
401     volumes:
402     - name: webhook-cert
403       secret:
404         secretName: ingress-nginx-admission
405 ---
406 # Source: ingress-nginx/templates/admission-webhooks/validating-webhook.yaml
407 # before changing this value, check the required kubernetes version
408 # https://kubernetes.io/docs/reference/access-authn-authz/extensible-admission-
controllers/#prerequisites
409 apiVersion: admissionregistration.k8s.io/v1
410 kind: ValidatingWebhookConfiguration
411 metadata:
412   labels:
413     helm.sh/chart: ingress-nginx-3.30.0
414     app.kubernetes.io/name: ingress-nginx
415     app.kubernetes.io/instance: ingress-nginx
416     app.kubernetes.io/version: 0.46.0
417     app.kubernetes.io/managed-by: Helm
418     app.kubernetes.io/component: admission-webhook
419   name: ingress-nginx-admission
420 webhooks:
421 - name: validate.nginx.ingress.kubernetes.io
422   matchPolicy: Equivalent
423   rules:
424   - apiGroups:
425     - networking.k8s.io

```

```

426         apiVersions:
427             - v1beta1
428         operations:
429             - CREATE
430             - UPDATE
431         resources:
432             - ingresses
433     failurePolicy: Fail
434     sideEffects: None
435     admissionReviewVersions:
436         - v1
437         - v1beta1
438     clientConfig:
439         service:
440             namespace: ingress-nginx
441             name: ingress-nginx-controller-admission
442             path: /networking/v1beta1/ingresses
443 ---
444 # Source: ingress-nginx/templates/admission-webhooks/job-
445 patch/serviceaccount.yaml
446 apiVersion: v1
447 kind: ServiceAccount
448 metadata:
449     name: ingress-nginx-admission
450     annotations:
451         helm.sh/hook: pre-install,pre-upgrade,post-install,post-upgrade
452         helm.sh/hook-delete-policy: before-hook-creation,hook-succeeded
453     labels:
454         helm.sh/chart: ingress-nginx-3.30.0
455         app.kubernetes.io/name: ingress-nginx
456         app.kubernetes.io/instance: ingress-nginx
457         app.kubernetes.io/version: 0.46.0
458         app.kubernetes.io/managed-by: Helm
459         app.kubernetes.io/component: admission-webhook
460     namespace: ingress-nginx
461 ---
462 # Source: ingress-nginx/templates/admission-webhooks/job-patch/clusterrole.yaml
463 apiVersion: rbac.authorization.k8s.io/v1
464 kind: ClusterRole
465 metadata:
466     name: ingress-nginx-admission
467     annotations:
468         helm.sh/hook: pre-install,pre-upgrade,post-install,post-upgrade
469         helm.sh/hook-delete-policy: before-hook-creation,hook-succeeded
470     labels:
471         helm.sh/chart: ingress-nginx-3.30.0
472         app.kubernetes.io/name: ingress-nginx
473         app.kubernetes.io/instance: ingress-nginx
474         app.kubernetes.io/version: 0.46.0
475         app.kubernetes.io/managed-by: Helm
476         app.kubernetes.io/component: admission-webhook
477 rules:
478     - apiGroups:
479         - admissionregistration.k8s.io
480     resources:
481         - validatingwebhookconfigurations
482     verbs:
483         - get

```

```
483         - update
484     ---
485     # Source: ingress-nginx/templates/admission-webhooks/job-
486     patch/clusterrolebinding.yaml
487     apiVersion: rbac.authorization.k8s.io/v1
488     kind: ClusterRoleBinding
489     metadata:
490       name: ingress-nginx-admission
491       annotations:
492         helm.sh/hook: pre-install,pre-upgrade,post-install,post-upgrade
493         helm.sh/hook-delete-policy: before-hook-creation,hook-succeeded
494       labels:
495         helm.sh/chart: ingress-nginx-3.30.0
496         app.kubernetes.io/name: ingress-nginx
497         app.kubernetes.io/instance: ingress-nginx
498         app.kubernetes.io/version: 0.46.0
499         app.kubernetes.io/managed-by: Helm
500         app.kubernetes.io/component: admission-webhook
501     roleRef:
502       apiGroup: rbac.authorization.k8s.io
503       kind: ClusterRole
504       name: ingress-nginx-admission
505     subjects:
506     - kind: ServiceAccount
507       name: ingress-nginx-admission
508       namespace: ingress-nginx
509     ---
510     # Source: ingress-nginx/templates/admission-webhooks/job-patch/role.yaml
511     apiVersion: rbac.authorization.k8s.io/v1
512     kind: Role
513     metadata:
514       name: ingress-nginx-admission
515       annotations:
516         helm.sh/hook: pre-install,pre-upgrade,post-install,post-upgrade
517         helm.sh/hook-delete-policy: before-hook-creation,hook-succeeded
518       labels:
519         helm.sh/chart: ingress-nginx-3.30.0
520         app.kubernetes.io/name: ingress-nginx
521         app.kubernetes.io/instance: ingress-nginx
522         app.kubernetes.io/version: 0.46.0
523         app.kubernetes.io/managed-by: Helm
524         app.kubernetes.io/component: admission-webhook
525       namespace: ingress-nginx
526     rules:
527     - apiGroups:
528       - ''
529       resources:
530       - secrets
531       verbs:
532       - get
533       - create
534     ---
535     # Source: ingress-nginx/templates/admission-webhooks/job-patch/rolebinding.yaml
536     apiVersion: rbac.authorization.k8s.io/v1
537     kind: RoleBinding
538     metadata:
539       name: ingress-nginx-admission
540       annotations:
```



```

540     helm.sh/hook: pre-install,pre-upgrade,post-install,post-upgrade
541     helm.sh/hook-delete-policy: before-hook-creation,hook-succeeded
542     labels:
543       helm.sh/chart: ingress-nginx-3.30.0
544       app.kubernetes.io/name: ingress-nginx
545       app.kubernetes.io/instance: ingress-nginx
546       app.kubernetes.io/version: 0.46.0
547       app.kubernetes.io/managed-by: Helm
548       app.kubernetes.io/component: admission-webhook
549     namespace: ingress-nginx
550   roleRef:
551     apiGroup: rbac.authorization.k8s.io
552     kind: Role
553     name: ingress-nginx-admission
554   subjects:
555     - kind: ServiceAccount
556       name: ingress-nginx-admission
557       namespace: ingress-nginx
558   ---
559   # Source: ingress-nginx/templates/admission-webhooks/job-patch/job-
createSecret.yaml
560   apiVersion: batch/v1
561   kind: Job
562   metadata:
563     name: ingress-nginx-admission-create
564     annotations:
565       helm.sh/hook: pre-install,pre-upgrade
566       helm.sh/hook-delete-policy: before-hook-creation,hook-succeeded
567     labels:
568       helm.sh/chart: ingress-nginx-3.30.0
569       app.kubernetes.io/name: ingress-nginx
570       app.kubernetes.io/instance: ingress-nginx
571       app.kubernetes.io/version: 0.46.0
572       app.kubernetes.io/managed-by: Helm
573       app.kubernetes.io/component: admission-webhook
574     namespace: ingress-nginx
575   spec:
576     template:
577       metadata:
578         name: ingress-nginx-admission-create
579         labels:
580           helm.sh/chart: ingress-nginx-3.30.0
581           app.kubernetes.io/name: ingress-nginx
582           app.kubernetes.io/instance: ingress-nginx
583           app.kubernetes.io/version: 0.46.0
584           app.kubernetes.io/managed-by: Helm
585           app.kubernetes.io/component: admission-webhook
586       spec:
587         containers:
588           - name: create
589             image: docker.io/jettech/kube-webhook-certgen:v1.5.1
590             imagePullPolicy: IfNotPresent
591             args:
592               - create
593               - --host=ingress-nginx-controller-admission,ingress-nginx-
controller-admission.$(POD_NAMESPACE).svc
594               - --namespace=$(POD_NAMESPACE)
595               - --secret-name=ingress-nginx-admission

```

```

596         env:
597             - name: POD_NAMESPACE
598               valueFrom:
599                 fieldRef:
600                   fieldPath: metadata.namespace
601         restartPolicy: OnFailure
602         serviceAccountName: ingress-nginx-admission
603         securityContext:
604             runAsNonRoot: true
605             runAsUser: 2000
606     ---
607     # Source: ingress-nginx/templates/admission-webhooks/job-patch/job-
608     patchWebhook.yaml
609     apiVersion: batch/v1
610     kind: Job
611     metadata:
612         name: ingress-nginx-admission-patch
613         annotations:
614             helm.sh/hook: post-install,post-upgrade
615             helm.sh/hook-delete-policy: before-hook-creation,hook-succeeded
616         labels:
617             helm.sh/chart: ingress-nginx-3.30.0
618             app.kubernetes.io/name: ingress-nginx
619             app.kubernetes.io/instance: ingress-nginx
620             app.kubernetes.io/version: 0.46.0
621             app.kubernetes.io/managed-by: Helm
622             app.kubernetes.io/component: admission-webhook
623     namespace: ingress-nginx
624     spec:
625         template:
626             metadata:
627                 name: ingress-nginx-admission-patch
628                 labels:
629                     helm.sh/chart: ingress-nginx-3.30.0
630                     app.kubernetes.io/name: ingress-nginx
631                     app.kubernetes.io/instance: ingress-nginx
632                     app.kubernetes.io/version: 0.46.0
633                     app.kubernetes.io/managed-by: Helm
634                     app.kubernetes.io/component: admission-webhook
635             spec:
636                 containers:
637                     - name: patch
638                       image: docker.io/jettech/kube-webhook-certgen:v1.5.1
639                       imagePullPolicy: IfNotPresent
640                       args:
641                         - patch
642                         - --webhook-name=ingress-nginx-admission
643                         - --namespace=$(POD_NAMESPACE)
644                         - --patch-mutating=false
645                         - --secret-name=ingress-nginx-admission
646                         - --patch-failure-policy=Fail
647                       env:
648                           - name: POD_NAMESPACE
649                             valueFrom:
650                               fieldRef:
651                                 fieldPath: metadata.namespace
652                       restartPolicy: OnFailure
653                       serviceAccountName: ingress-nginx-admission

```

```
653     securityContext:
654         runAsNonRoot: true
655         runAsUser: 2000
```

2、验证

访问部署了ingress-nginx主机的80端口，有nginx响应即可。

2、卸载

`kubectl delete -f ingress-controller.yaml` 即可

3、案例实战

1、基本配置

```
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4      name: itdachang-ingress
5      namespace: default
6  spec:
7      rules:
8      - host: itdachang.com
9        http:
10          paths:
11          - path: /
12            pathType: Prefix
13            backend: ## 指定需要响应的后端服务
14              service:
15                name: my-nginx-svc ## kubernetes集群的svc名称
16                port:
17                  number: 80 ## service的端口号
```

- **pathType 详细：**
 - **Prefix**：基于以 / 分隔的 URL 路径前缀匹配。匹配区分大小写，并且对路径中的元素逐个完成。路径元素指的是由 / 分隔符分隔的路径中的标签列表。如果每个 p 都是请求路径 p 的元素前缀，则请求与路径 p 匹配。
 - **Exact**：精确匹配 URL 路径，且区分大小写。
 - **ImplementationSpecific**：对于这种路径类型，匹配方法取决于 IngressClass。具体实现可以将其作为单独的 **pathType** 处理或者与 **Prefix** 或 **Exact** 类型作相同处理。

2、默认后端

```
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
```

```

3  metadata:
4      name: itdachang-ingress
5      namespace: default
6  spec:
7      defaultBackend: ## 指定所有未匹配的默认后端
8          service:
9              name: php-apache
10             port:
11                 number: 80
12     rules:
13     - host: itdachang.com
14       http:
15         paths:
16         - path: /abc
17           pathType: Prefix
18           backend:
19             service:
20               name: my-nginx-svc
21             port:
22                 number: 80

```

效果

- itdachang.com 下的 非 /abc 开头的所有请求，都会到defaultBackend
- 非itdachang.com 域名下的所有请求，也会到defaultBackend

3、路径重写

<https://kubernetes.github.io/ingress-nginx/examples/rewrite/>

Rewrite 功能，经常被用于前后分离的场景

- 前端给服务器发送 / 请求映射前端地址。
- 后端给服务器发送 /api 请求来到对应的服务。但是后端服务没有 /api的起始路径，所以需要 ingress-controller自动截串

```

1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4      annotations:
5          nginx.ingress.kubernetes.io/rewrite-target: /$2
6      name: rewrite
7      namespace: default
8  spec:
9      rules:
10     - host: itdachang.com
11       http:
12         paths:
13         - backend:
14             service:
15                 name: php-apache
16                 port:
17                     number: 80

```

```
18     path: /api(/|$)(.*)
19     pathType: Prefix
```

4、配置SSL

<https://kubernetes.github.io/ingress-nginx/user-guide/tls/>

生成证书：（也可以去青云申请免费证书进行配置）

```
1  $ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ${KEY_FILE:tls.key}
   -out ${CERT_FILE:tls.cert} -subj
   "/CN=${HOST:itdachang.com}/O=${HOST:itdachang.com}"
2
3  kubectl create secret tls ${CERT_NAME:itdachang-tls} --key ${KEY_FILE:tls.key} --
   cert ${CERT_FILE:tls.cert}
4
5
6  ## 示例命令如下
7  openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.cert
   -subj "/CN=itdachang.com/O=itdachang.com"
8
9  kubectl create secret tls itdachang-tls --key tls.key --cert tls.cert
```

配置域名使用证书；

```
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    name: itdachang-ingress
5    namespace: default
6  spec:
7    tls:
8      - hosts:
9        - itdachang.com
10      secretName: itdachang-tls
11  rules:
12    - host: itdachang.com
13      http:
14        paths:
15          - path: /
16            pathType: Prefix
17            backend:
18              service:
19                name: my-nginx-svc
20                port:
21                  number: 80
```

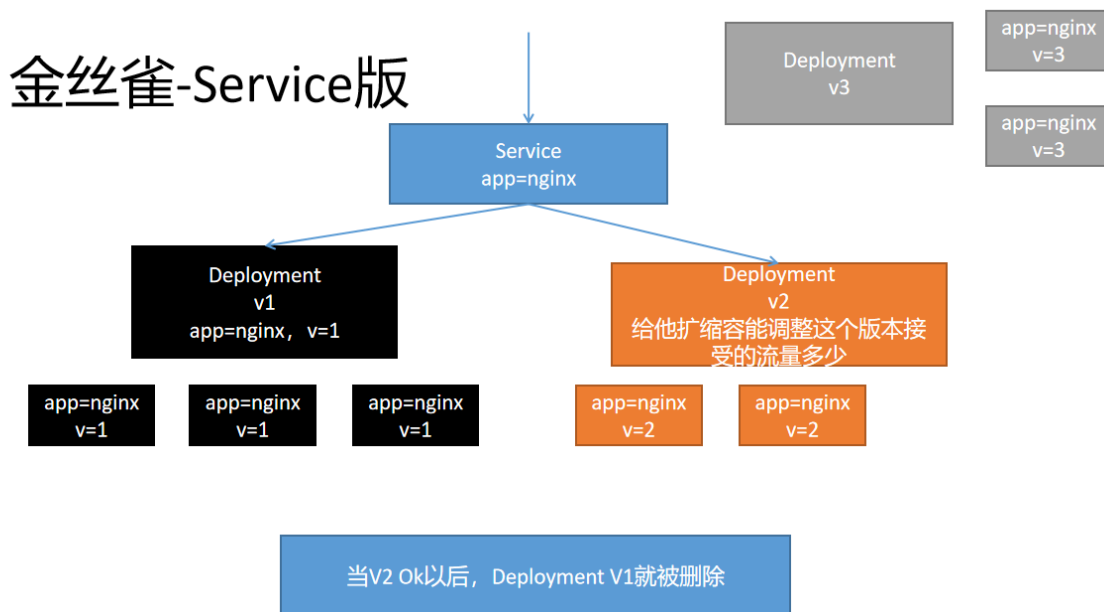
配置好证书，访问域名，就会默认跳转到https；

5、限速

<https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#rate-limiting>

6、灰度发布-Canary

以前可以使用k8s的Service配合Deployment进行金丝雀部署。原理如下

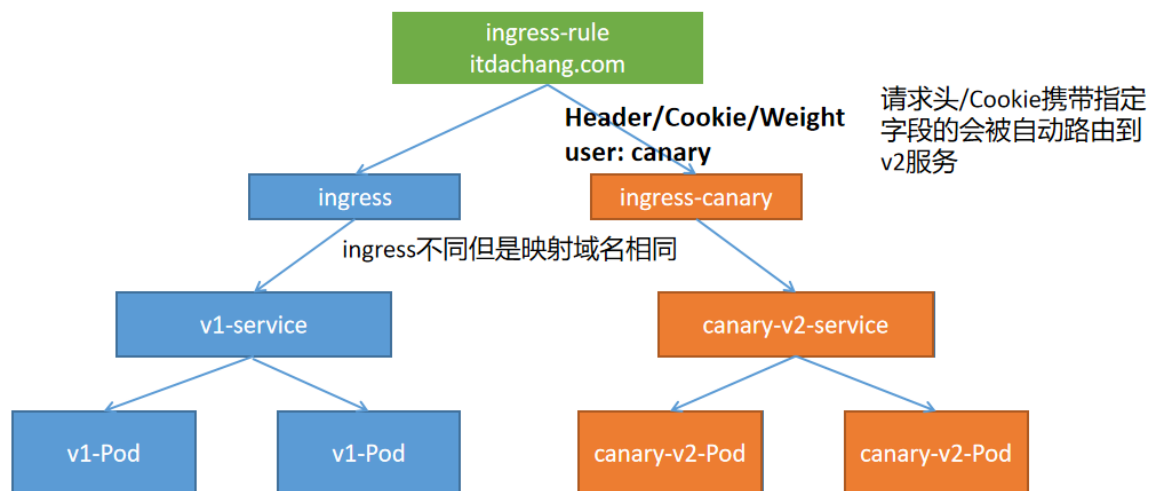


缺点:

- 不能自定义灰度逻辑, 比如指定用户进行灰度

现在可以使用Ingress进行灰度。原理如下

金丝雀-Ingress版



以后新版本上线, 配置新的ingress-canary规则即可。
canary验证通过以后, 移除旧的ingress和service。
取消当前ingress-canary的annotation, 变为普通的ingress

```
1 ## 使用如下文件部署两个service版本。v1版本返回nginx默认页, v2版本返回 11111
2 apiVersion: v1
3 kind: Service
```

```
4  metadata:
5      name: v1-service
6      namespace: default
7  spec:
8      selector:
9          app: v1-pod
10     type: ClusterIP
11     ports:
12     - name: http
13       port: 80
14       targetPort: 80
15       protocol: TCP
16 ---
17 apiVersion: apps/v1
18 kind: Deployment
19 metadata:
20     name: v1-deploy
21     namespace: default
22     labels:
23         app: v1-deploy
24 spec:
25     selector:
26         matchLabels:
27             app: v1-pod
28     replicas: 1
29     template:
30         metadata:
31             labels:
32                 app: v1-pod
33         spec:
34             containers:
35             - name: nginx
36               image: nginx
37 ---
38 apiVersion: v1
39 kind: Service
40 metadata:
41     name: canary-v2-service
42     namespace: default
43 spec:
44     selector:
45         app: canary-v2-pod
46     type: ClusterIP
47     ports:
48     - name: http
49       port: 80
50       targetPort: 80
51       protocol: TCP
52 ---
53 apiVersion: apps/v1
54 kind: Deployment
55 metadata:
56     name: canary-v2-deploy
57     namespace: default
58     labels:
59         app: canary-v2-deploy
60 spec:
61     selector:
```

```

62     matchLabels:
63       app: canary-v2-pod
64   replicas: 1
65   template:
66     metadata:
67       labels:
68         app: canary-v2-pod
69   spec:
70     containers:
71     - name: nginx
72       image: registry.cn-hangzhou.aliyuncs.com/lfy_k8s_images/nginx-test:env-
msg

```

7、会话保持-Session亲和性

<https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#session-affinity>

第一次访问，ingress-nginx会返回给浏览器一个Cookie，以后浏览器带着这个Cookie，保证访问总是抵达之前的Pod；

```

1  ## 部署一个三个Pod的Deployment并设置Service
2  apiVersion: v1
3  kind: Service
4  metadata:
5    name: session-affinity
6    namespace: default
7  spec:
8    selector:
9      app: session-affinity
10   type: ClusterIP
11   ports:
12   - name: session-affinity
13     port: 80
14     targetPort: 80
15     protocol: TCP
16   ---
17   apiVersion: apps/v1
18   kind: Deployment
19   metadata:
20     name: session-affinity
21     namespace: default
22     labels:
23       app: session-affinity
24   spec:
25     selector:
26       matchLabels:
27         app: session-affinity
28     replicas: 3
29     template:
30       metadata:
31         labels:
32           app: session-affinity
33       spec:
34         containers:

```



```
35     - name: session-affinity
36     image: nginx
```

编写具有会话亲和的ingress

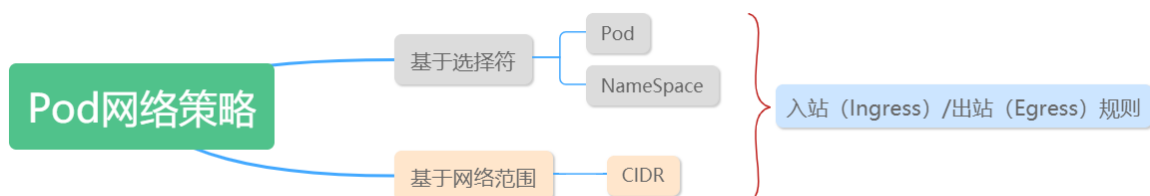
四、NetworkPolicy

<https://kubernetes.io/zh/docs/concepts/services-networking/network-policies/>

指定Pod间的网络隔离策略，默认是所有互通。

Pod 之间互通，是通过如下三个标识符的组合来辩识的：

1. 其他被允许的 Pods（例外：Pod 无法阻塞对自身的访问）
2. 被允许的名称空间
3. IP 组块（例外：与 Pod 运行所在的节点的通信总是被允许的，无论 Pod 或节点的 IP 地址）



1、Pod隔离与非隔离

- 默认情况下，Pod 都是非隔离的（non-isolated），可以接受来自任何请求方的网络请求。
- 如果一个 NetworkPolicy 的标签选择器选中了某个 Pod，则该 Pod 将变成隔离的（isolated），并将拒绝任何不被 NetworkPolicy 许可的网络连接。

2、规约

```
1  apiVersion: networking.k8s.io/v1
2  kind: NetworkPolicy
3  metadata:
4    name: test-network-policy
5    namespace: default
6  spec:
7    podSelector: ## 选中指定Pod
8      matchLabels:
9        role: db
10   policyTypes: ## 定义上面Pod的入站出站规则
11     - Ingress
12     - Egress
13   ingress: ## 定义入站白名单
14     - from:
15       - ipBlock:
```

```

16     cidr: 172.17.0.0/16
17     except:
18     - 172.17.1.0/24
19   - namespaceSelector:
20     matchLabels:
21       project: myproject
22   - podSelector:
23     matchLabels:
24       role: frontend
25   ports:
26   - protocol: TCP
27     port: 6379
28   egress: ## 定义出站白名单
29   - to:
30     - ipBlock:
31       cidr: 10.0.0.0/24
32     ports:
33     - protocol: TCP
34       port: 5978

```

- **基本信息：** 同其他的 Kubernetes 对象一样，`NetworkPolicy` 需要 `apiVersion`、`kind`、`metadata` 字段
- **spec：** `NetworkPolicy` 的 `spec` 字段包含了定义网络策略的主要信息：
 - **podSelector：** 同名称空间中，符合此标签选择器 `.spec.podSelector` 的 Pod 都将应用这个 `NetworkPolicy`。上面的 Example 中的 `podSelector` 选择了 `role=db` 的 Pod。如果该字段为空，则将对名称空间中所有的 Pod 应用这个 `NetworkPolicy`
 - **policyTypes：** `.spec.policyTypes` 是一个数组类型的字段，该数组中可以包含 `Ingress`、`Egress` 中的一个，也可能两个都包含。该字段标识了此 `NetworkPolicy` 是否应用到入方向的网络流量、出方向的网络流量、或者两者都有。如果不指定 `policyTypes` 字段，该字段默认将始终包含 `Ingress`，当 `NetworkPolicy` 中包含出方向的规则时，`Egress` 也将被添加到默认值。
 - **ingress：** `ingress` 是一个数组，代表入方向的白名单规则。每一条规则都将允许与 `from` 和 `ports` 匹配的入方向的网络流量发生。例子中的 `ingress` 包含了一条规则，允许的入方向网络流量必须符合如下条件：
 - Pod 的监听端口为 `6379`
 - 请求方可以是如下三种来源当中的任何一种：
 - `ipBlock` 为 `172.17.0.0/16` 网段，但是不包括 `172.17.1.0/24` 网段
 - `namespaceSelector` 标签选择器，匹配标签为 `project=myproject`
 - `podSelector` 标签选择器，匹配标签为 `role=frontend`
 - **egress：** `egress` 是一个数组，代表出方向的白名单规则。每一条规则都将允许与 `to` 和 `ports` 匹配的出方向的网络流量发生。例子中的 `egress` 允许的出方向网络流量必须符合如下条件：
 - 目标端口为 `5978`
 - 目标 `ipBlock` 为 `10.0.0.0/24` 网段

因此，例子中的 `NetworkPolicy` 对网络流量做了如下限制：

1. 隔离了 `default` 名称空间中带有 `role=db` 标签的所有 Pod 的入方向网络流量和出方向网络流量
2. `Ingress` 规则（入方向白名单规则）：

- 当请求方是如下三种来源中的任意一种时，允许访问 `default` 名称空间中所有带 `role=db` 标签的 Pod 的6379端口：
 - ipBlock 为 `172.17.0.0/16` 网段，但是不包括 `172.17.1.0/24` 网段
 - namespaceSelector 标签选择器，匹配标签为 `project=myproject`
 - podSelector 标签选择器，匹配标签为 `role=frontend`
- 3. Egress规则（出方向白名单规则）：
 - 当如下条件满足时，允许出方向的网络流量：
 - 目标端口为 `5978`
 - 目标 ipBlock 为 `10.0.0.0/24` 网段

3、to和from选择器的行为

NetworkPolicy 的 `.spec.ingress.from` 和 `.spec.egress.to` 字段中，可以指定 4 种类型的标签选择器：

- `podSelector` 选择与 `NetworkPolicy` 同名称空间中的 Pod 作为入方向访问控制规则的源或者出方向访问控制规则的目标
- `namespaceSelector` 选择某个名称空间（其中所有的Pod）作为入方向访问控制规则的源或者出方向访问控制规则的目标
- `namespaceSelector` 和 `podSelector` 在一个 `to` / `from` 条目中同时包含 `namespaceSelector` 和 `podSelector` 将选中指定名称空间中的指定 Pod。此时请特别留意 YAML 的写法，如下所示：

```
1    ...
2    ingress:
3      - from:
4        - namespaceSelector:
5            matchLabels:
6              user: alice
7          podSelector:
8            matchLabels:
9              role: client
10    ...
```

该例子中，`podSelector` 前面没有 `-` 减号，`namespaceSelector` 和 `podSelector` 是同一个 `from` 元素的两个字段，将选中带 `user=alice` 标签的名称空间中所有带 `role=client` 标签的 Pod。但是，下面的这个 NetworkPolicy 含义是不一样的：

```
1    ...
2    ingress:
3      - from:
4        - namespaceSelector:
5            matchLabels:
6              user: alice
7          - podSelector:
8            matchLabels:
9              role: client
10    ...
```

后者，`podSelector` 前面带 `-` 减号，说明 `namespaceSelector` 和 `podSelector` 是 `from` 数组中的两个元素，他们将选中 NetworkPolicy 同名称空间中带 `role=client` 标签的对象，以及带 `user=alice` 标签的名称空间的所有 Pod。

前者是交集关系，后者是并集关系

- **ipBlock** 可选择 IP CIDR 范围作为入方向访问控制规则的源或者出方向访问控制规则的目标。这里应该指定的是集群外部的 IP，因为集群内部 Pod 的 IP 地址是临时分配的，且不可预测。

集群的入方向和出方向网络机制通常需要重写网络报文的 source 或者 destination IP。kubernetes 并未定义应该在处理 **NetworkPolicy** 之前还是之后再修改 source / destination IP，因此，在不同的云供应商、使用不同的网络插件时，最终的行为都可能不一样。这意味着：

- 对于入方向的网络流量，某些情况下，你可以基于实际的源 IP 地址过滤流入的报文；在另外一些情况下，NetworkPolicy 所处理的 "source IP" 可能是 LoadBalancer 的 IP 地址，或者其他地址
- 对于出方向的网络流量，**基于 ipBlock 的策略可能有效，也可能无效**

4、场景

<https://kubernetes.io/zh/docs/concepts/services-networking/network-policies/#default-policies>