有道笔记地址：<https://note.youdao.com/s/3EQiAQn8>

**1.Nginx Ingress介绍**

在Kubernetes集群中，Ingress作为集群内服务对外暴露的访问接入点，其几乎承载着集群内服务访问的所有流量。Ingress是Kubernetes中的一个资源对象，用来管理集群外部访问集群内部服务的方式。您可以通过Ingress资源来配置不同的转发规则，从而达到根据不同的规则设置访问集群内不同的Service所对应的后端Pod。

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

**1.1 Nginx Ingress Controller 工作原理**

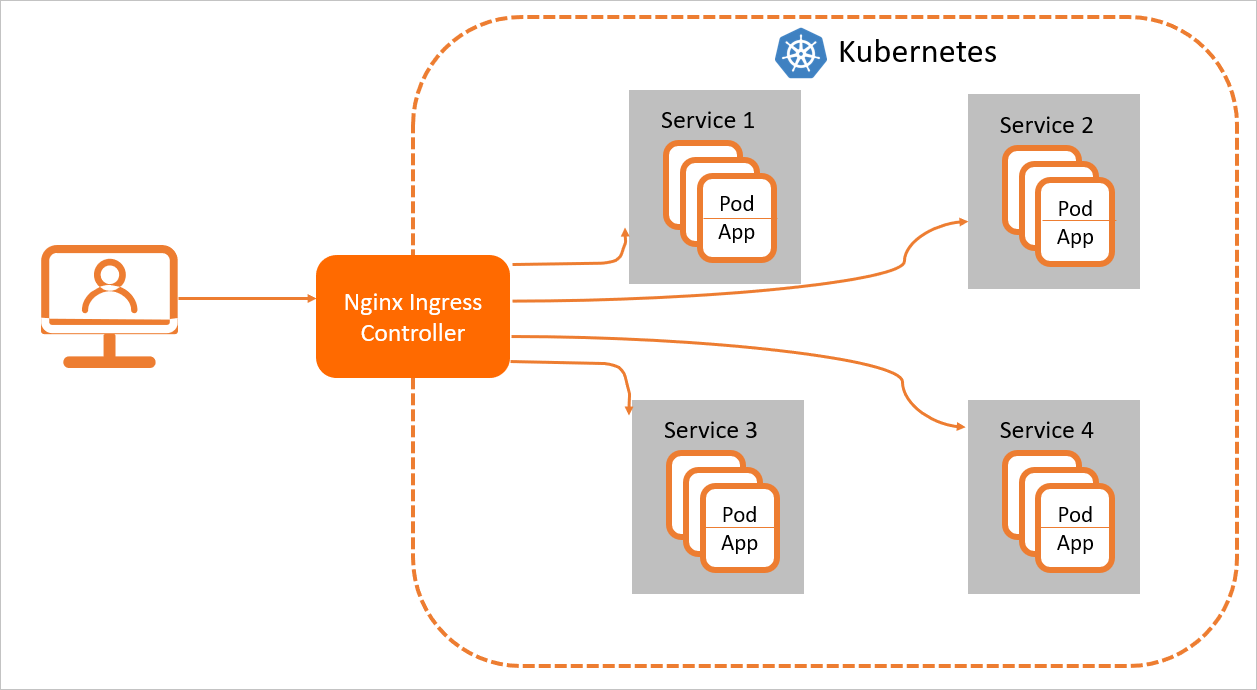
为了使得Nginx Ingress资源正常工作，集群中必须要有个Nginx Ingress Controller来解析Nginx Ingress的转发规则。Nginx Ingress Controller收到请求，匹配Nginx Ingress转发规则转发到后端Service所对应的Pod，由Pod处理请求。Kubernetes中Service、Nginx Ingress与Nginx Ingress Controller有着以下关系：

Service是后端真实服务的抽象，一个Service可以代表多个相同的后端服务。

Nginx Ingress是反向代理规则，用来规定HTTP/HTTPS请求应该被转发到哪个Service所对应的Pod上。例如根据请求中不同的Host和URL路径，让请求落到不同Service所对应的Pod上。

Nginx Ingress Controller是一个反向代理程序，负责解析Nginx Ingress的反向代理规则。如果Nginx Ingress有增删改的变动，Nginx Ingress Controller会及时更新自己相应的转发规则，当Nginx Ingress Controller收到请求后就会根据这些规则将请求转发到对应Service的Pod上。

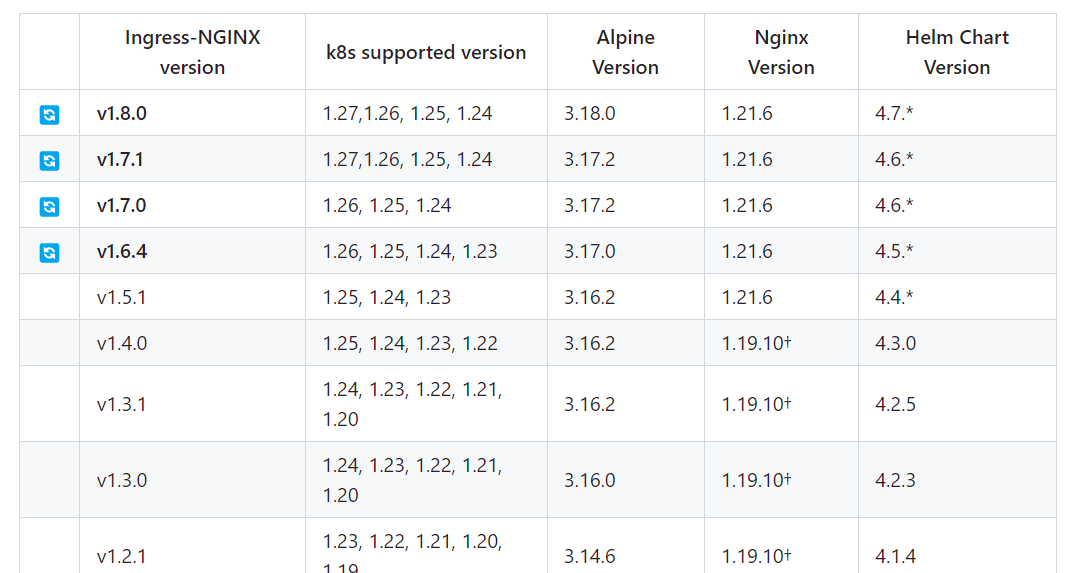
Nginx Ingress Controller通过API Server获取Ingress资源的变化，动态地生成Load Balancer（例如Nginx）所需的配置文件（例如nginx.conf），然后重新加载Load Balancer（例如执行nginx -s load重新加载Nginx）来生成新的路由转发规则。



**1.2 Nginx Ingress安装**

使用 Nginx Ingress 实现灰度发布的集群，需部署 Nginx Ingress 作为 Ingress Controller，并且对外暴露统一的流量入口。

兼容性查看：<https://github.com/kubernetes/ingress-nginx#supported-versions-table>



**下载Ingress**

<https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.8.0/deploy/static/provider/aws/nlb-with-tls-termination/deploy.yaml>

也可以使用我提供的**deploy.yaml （v1.8.0）**

**安装Ingress**

# 查看会用到的镜像， registry.k8s.io拉取不了

grep image: deploy.yaml

image: registry.k8s.io/ingress-nginx/controller:v1.8.1@sha256:e5c4824e7375fcf2a393e1c03c293b69759af37a9ca6abdb91b13d78a93da8bd

image: registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230407@sha256:543c40fd093964bc9ab509d3e791f9989963021f1e9e4c9c7b6700b02bfb227b

image: registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230407@sha256:543c40fd093964bc9ab509d3e791f9989963021f1e9e4c9c7b6700b02bfb227b

#因默认镜像源国内无法访问 ，修改为docker hub上的镜像源

docker pull giantswarm/ingress-nginx-controller:v1.8.1

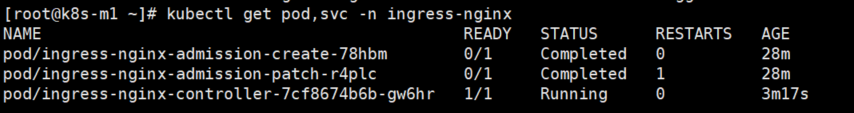
docker pull dyrnq/kube-webhook-certgen:v20230407

# 安装

kubectl apply -f deploy.yaml

#查看安装是否成功

kubectl get pods -n ingress-nginx -owide



**2. 使用 Nginx Ingress 实现灰度发布**

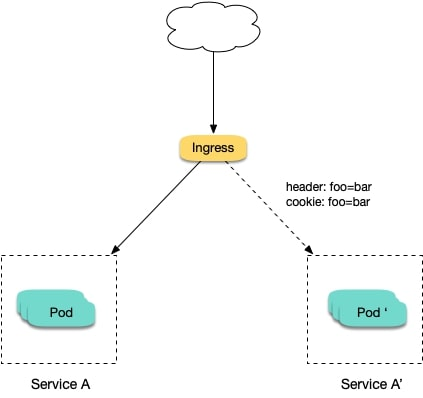
<https://kubernetes.github.io/ingress-nginx/examples/canary/>

**2.1 使用场景**

使用 Nginx Ingress 实现灰度发布适用场景主要取决于业务流量切分的策略，目前 Nginx Ingress 支持基于 Header、Cookie 和服务权重3种流量切分的策略，基于这3种策略可实现以下两种发布场景：

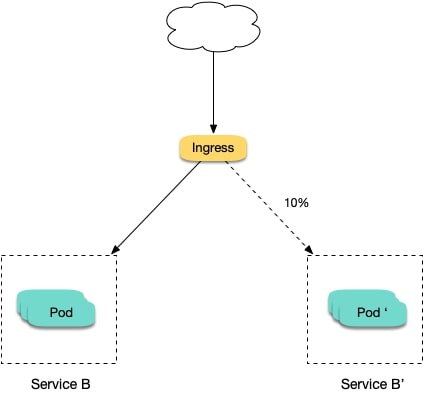
**场景1: 灰度新版本到部分用户**

假设线上已运行了一套对外提供7层服务的 Service A，此时需上线开发的新版本 Service A'，但不期望直接替换原有的 Service A，仅灰度部分用户，待运行一段时间足够稳定后再逐渐全量上线新版本，平滑下线旧版本。针对此场景可使用 Nginx Ingress 基于 Header 或 Cookie 进行流量切分的策略来发布，业务使用 Header 或 Cookie 来标识不同类型的用户，并通过配置 Ingress 来实现让带有指定 Header 或 Cookie 的请求被转发到新版本，其它请求仍然转发到旧版本，从而将新版本灰度给部分用户。示意图如下：



**场景2: 切分一定比例的流量到新版本**

假设线上已运行了一套对外提供7层服务的 Service B，此时修复了 Service B 的部分问题，需灰度上线新版本 Service B'。但不期望直接替换原有的 Service B，需先切换10%的流量至新版本，待运行一段时间足够稳定后再逐渐加大新版本流量比例直至完全替换旧版本，最终平滑下线旧版本。示意图如下：



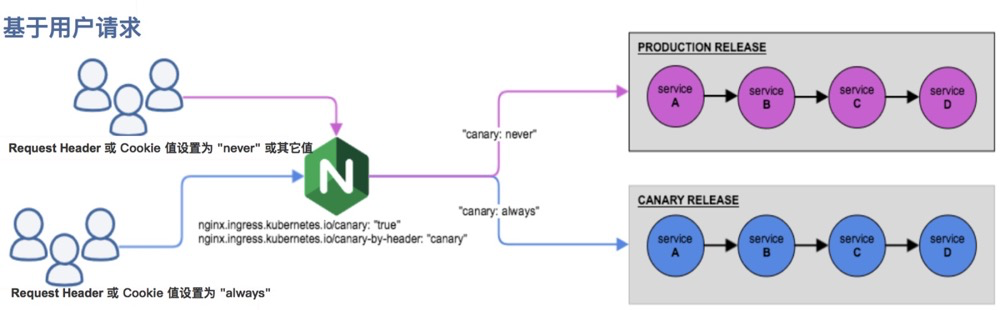
**2.2 注解说明**

通过给 Ingress 资源指定 Nginx Ingress 所支持的 annotation 可实现灰度发布。需给服务创建2个 Ingress，其中1个常规 Ingress，另1个为带 nginx.ingress.kubernetes.io/canary: "true" 固定的 annotation 的 Ingress，称为 Canary Ingress。Canary Ingress 一般代表新版本的服务，结合另外针对流量切分策略的 annotation 一起配置即可实现多种场景的灰度发布。

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

以下为相关 annotation 的详细介绍：

**nginx.ingress.kubernetes.io/canary-by-header**
表示如果请求头中包含指定的 header 名称，并且值为 always，就将该请求转发给该 Ingress 定义的对应后端服务。如果值为 never 则不转发，可以用于回滚到旧版。如果为其他值则忽略该 annotation。

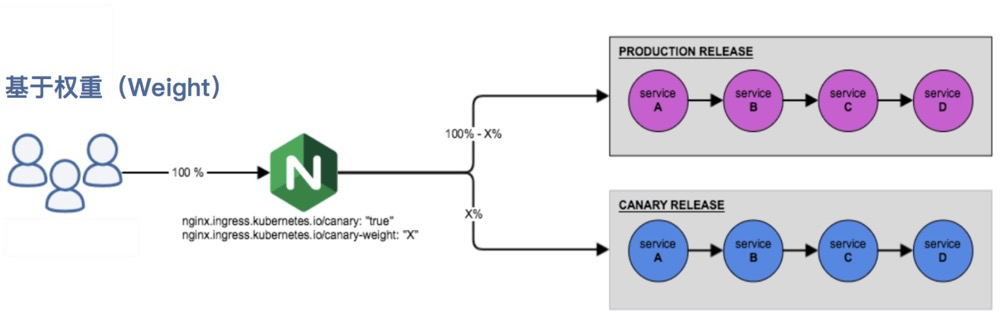


**nginx.ingress.kubernetes.io/canary-by-header-value**
该 annotation 可以作为 canary-by-header 的补充，可指定请求头为自定义值，包含但不限于 always 或 never。当请求头的值命中指定的自定义值时，请求将会转发给该 Ingress 定义的对应后端服务，如果是其它值则忽略该 annotation。

**nginx.ingress.kubernetes.io/canary-by-header-pattern**
与 canary-by-header-value 类似，区别为该 annotation 用正则表达式匹配请求头的值，而不是只固定某一个值。如果该 annotation 与 canary-by-header-value 同时存在，该 annotation 将被忽略。

**nginx.ingress.kubernetes.io/canary-by-cookie**
与 canary-by-header 类似，该 annotation 用于 cookie，仅支持 always 和 never。

**nginx.ingress.kubernetes.io/canary-weight**
表示 Canary Ingress 所分配流量的比例的百分比，取值范围 [0-100]。例如，设置为10，则表示分配10%的流量给 Canary Ingress 对应的后端服务。



以上规则会按优先顺序进行评估，优先顺序为： canary-by-header -> canary-by-cookie -> canary-weight。

当 Ingress 被标记为 Canary Ingress 时，除了 nginx.ingress.kubernetes.io/load-balance 和 nginx.ingress.kubernetes.io/upstream-hash-by 外，所有其他非 Canary 注释都将被忽略。

**2.3 使用示例**

**部署两个版本的服务**

1）在集群中部署第一个版本的 Deployment，本文以 nginx-v1 为例。YAML 示例如下：

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-v1

spec:

replicas: 1

selector:

matchLabels:

app: nginx

version: v1

template:

metadata:

labels:

app: nginx

version: v1

spec:

containers:

- name: nginx

image: "openresty/openresty:centos"

ports:

- name: http

protocol: TCP

containerPort: 80

volumeMounts:

- mountPath: /usr/local/openresty/nginx/conf/nginx.conf

name: config

subPath: nginx.conf

volumes:

- name: config

configMap:

name: nginx-v1

---

apiVersion: v1

kind: ConfigMap

metadata:

labels:

app: nginx

version: v1

name: nginx-v1

data:

nginx.conf: |-

worker\_processes 1;

events {

accept\_mutex on;

multi\_accept on;

use epoll;

worker\_connections 1024;

}

http {

ignore\_invalid\_headers off;

server {

listen 80;

location / {

access\_by\_lua '

local header\_str = ngx.say("nginx-v1")

';

}

}

}

---

apiVersion: v1

kind: Service

metadata:

name: nginx-v1

spec:

type: ClusterIP

ports:

- port: 80

protocol: TCP

name: http

selector:

app: nginx

version: v1

2）再部署第二个版本的 Deployment，本文以 nginx-v2 为例。YAML 示例如下：

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-v2

spec:

replicas: 1

selector:

matchLabels:

app: nginx

version: v2

template:

metadata:

labels:

app: nginx

version: v2

spec:

containers:

- name: nginx

image: "openresty/openresty:centos"

ports:

- name: http

protocol: TCP

containerPort: 80

volumeMounts:

- mountPath: /usr/local/openresty/nginx/conf/nginx.conf

name: config

subPath: nginx.conf

volumes:

- name: config

configMap:

name: nginx-v2

---

apiVersion: v1

kind: ConfigMap

metadata:

labels:

app: nginx

version: v2

name: nginx-v2

data:

nginx.conf: |-

worker\_processes 1;

events {

accept\_mutex on;

multi\_accept on;

use epoll;

worker\_connections 1024;

}

http {

ignore\_invalid\_headers off;

server {

listen 80;

location / {

access\_by\_lua '

local header\_str = ngx.say("nginx-v2")

';

}

}

}

---

apiVersion: v1

kind: Service

metadata:

name: nginx-v2

spec:

type: ClusterIP

ports:

- port: 80

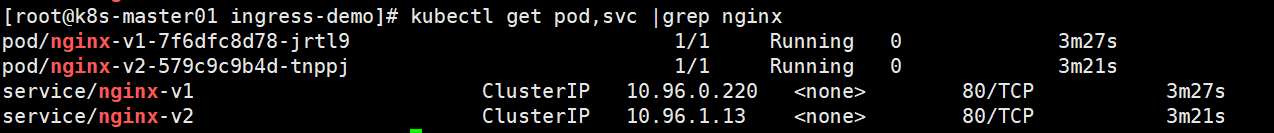
protocol: TCP

name: http

selector:

app: nginx

version: v2



3）创建 Ingress，对外暴露服务，指向 v1 版本的服务。YAML 示例如下：

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: nginx

spec:

ingressClassName: nginx

rules:

- host: canary.example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: nginx-v1

port:

number: 80

4）执行以下命令，进行访问验证

curl -H "Host: canary.example.com" http://EXTERNAL-IP # EXTERNAL-IP 替换为 Nginx Ingress 自身对外暴露的 IP



**基于 Header 的流量切分**

创建 Canary Ingress，指定 v2 版本的后端服务，并增加 annotation。实现仅将带有名为 Region 且值为 cd 或 sz 的请求头的请求转发给当前 Canary Ingress，模拟灰度新版本给成都和深圳地域的用户。YAML 示例如下：

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

annotations:

nginx.ingress.kubernetes.io/canary: "true"

nginx.ingress.kubernetes.io/canary-by-header: "Region"

nginx.ingress.kubernetes.io/canary-by-header-pattern: "cd|sz"

name: nginx-canary

spec:

ingressClassName: nginx

rules:

- host: canary.example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: nginx-v2

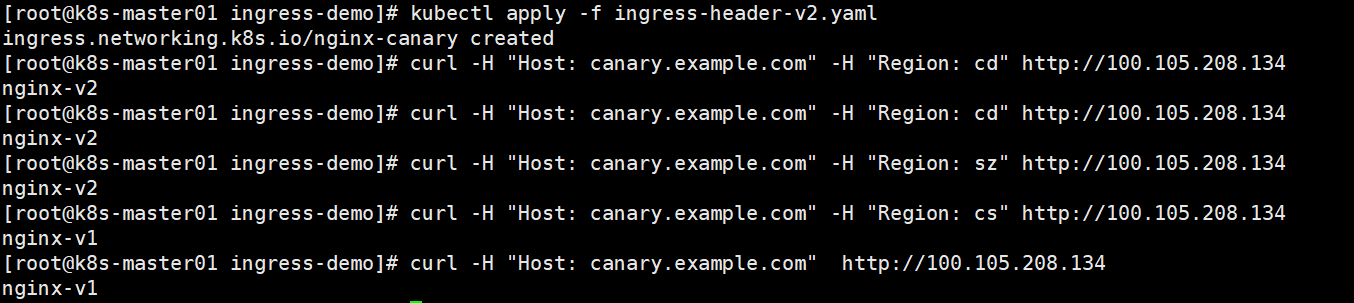
port:

number: 80

执行以下命令，进行访问测试

curl -H "Host: canary.example.com" -H "Region: cd" http://100.105.208.134

可查看当仅有 header Region 为 cd 或 sz 的请求才由 v2 版本服务响应



**基于 Cookie 的流量切分**

使用 Cookie 则无法自定义 value，以模拟灰度成都地域用户为例，仅将带有名为 user\_from\_cd 的 Cookie 的请求转发给当前 Canary Ingress。YAML 示例如下：

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

annotations:

nginx.ingress.kubernetes.io/canary: "true"

nginx.ingress.kubernetes.io/canary-by-cookie: "user\_from\_cd"

name: nginx-canary

spec:

ingressClassName: nginx

rules:

- host: canary.example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: nginx-v2

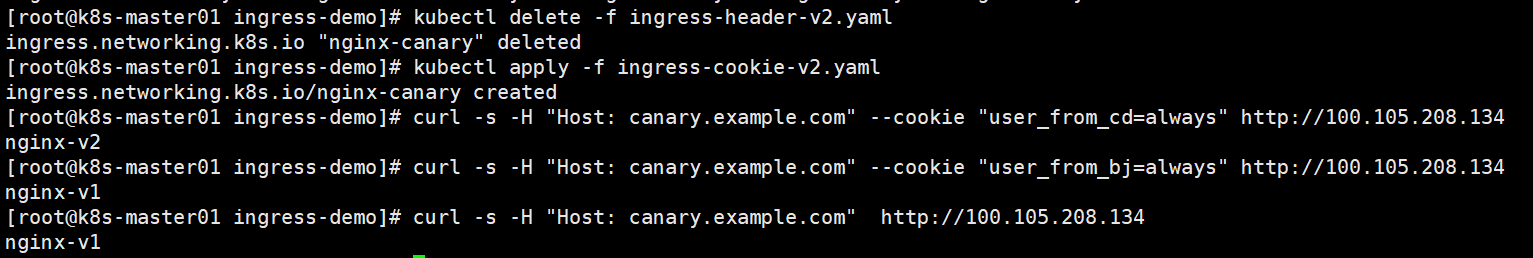
port:

number: 80

注意：部署此配置需要先清除Header的canary ingress 配置

执行以下命令，进行访问测试

curl -s -H "Host: canary.example.com" --cookie "user\_from\_cd=always" http://100.105.208.134



**基于服务权重的流量切分**

使用基于服务权重的 Canary Ingress 时，直接定义需要导入的流量比例即可。以导入10%流量到 v2 版本为例，YAML 示例如下：

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

annotations:

nginx.ingress.kubernetes.io/canary: "true"

nginx.ingress.kubernetes.io/canary-weight: "10"

name: nginx-canary

spec:

ingressClassName: nginx

rules:

- host: canary.example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: nginx-v2

port:

number: 80

执行以下命令，进行访问测试

for i in {1..10}; do curl -H "Host: canary.example.com" http://100.105.208.134; done;

可查看，有十分之一的几率由 v2 版本的服务响应，符合10%服务权重的设置

