Ingress Nginx

Ingress-nginx和Nginx-ingress的区别

```
Ingress-nginx：kubernetes官方维护的ingress

Nginx-ingress：nginx官方维护的ingress
# Ingress-nginx的官方文档：
  https://kubernetes.github.io/ingress-nginx/user-guide/nginx-
configuration/annotations/#rewrite

# Nginx-ingress的官方文档：
  https://docs.nginx.com/nginx-ingress-controller/configuration/ingress-
resources/advanced-configuration-with-annotations/
# Ingress-nginx源码地址
  https://github.com/kubernetes/ingress-nginx

# Nginx-ingress源码地址
  Github：https://github.com/nginxinc/kubernetes-ingress/blob/master/docs/nginx-ingress-
controllers.md
```

部署建议：

```
DaemonSet安装，找几台专门的服务器进行配置ingress （如果没有的足够的资源，就设置QoS，保证ingress最后删除的那个策略）
hostNetwork: true  # 这个设置为true
```

Ingress Nginx 安装

创建一个简单的ingress实例

```
apiVersion: extensions/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: ratel-test1
spec:
  rules:
  - host: ingress.test.com
    http:
      paths:
      - backend:
          serviceName: ingress-test    # 代理名字为ingress-test
          servicePort: 80              # port为80的svc
```

```
        path: /

# 创建ingress
kubectl create -f ingress-demo.yaml
```

Ingress Nginx Redirect

```
apiVersion: v1
items:
- apiVersion: extensions/v1
  kind: Ingress
  metadata:
    annotations:
      nginx.ingress.kubernetes.io/permanent-redirect: https://www.baidu.com   # 重定向到想去的url
    name: ingress-test
    namespace: ratel-test1
  spec:
    rules:
    - host: ingress.test.com
      http:
        paths:
        - backend:
            serviceName: ingress-test
            servicePort: 80
          path: /
kind: List
metadata:
  resourceVersion: ""
  selfLink: ""
```

```
# 创建ingress
kubectl edit ingress-demo.yaml

# entry 容器查看
kubectl get po -n ingress-nginx
kubectl exec -it nginx-ingress-controller-b2442 -n ingress-nginx -- bash
cat /etc/nginx/nginx.conf
```

Ingress Nginx Rewrite

```
apiVersion: extensions/v1
kind: Ingress
```

```
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
  generation: 4
  name: ingress-test
  namespace: ratel-test1
spec:
  rules:
  - host: rewrite.test.com
    http:
      paths:
      - backend:
          serviceName: ingress-test
          servicePort: 80
        path: /something(/|$)(.*)   #($1)($2)
```

```
# win 配置解析
10.4.7.107 rewrite.test.com

# 浏览器访问
http://rewrite.test.com/
http://rewrite.test.com/something  # 跳转到/
  # 比如，公司前端服务是： www.a.com
  # 后端服务是：www.a.com/api
```

Ingress Nginx https

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

生成本地证书,生产环境，用购买的。

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ${KEY_FILE} -out
${CERT_FILE} -subj "/CN=${HOST}/O=${HOST}"

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls-key -out tls.cert -subj
"/CN=test-tls.test.com/O=test-tls.test.com"

# create the secret in the cluster via
kubectl create secret tls ca-cert --key tls.key --cert tls.cert -n ratel-test1
```

禁用https强制跳转

nginx.ingress.kubernetes.io/ssl-redirect: "false"

```
apiVersion: extensions/v1
kind: Ingress
metadata:
```

```
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "false"   # 禁用https强制跳转
  generation: 1
  name: test-tls
  namespace: ratel-test1
spec:
  rules:
  - host: test-tls.test.com
    http:
      paths:
      - backend:
          serviceName: ingress-test
          servicePort: 80
        path: /
  tls:
  - hosts:
    - test-tls.test.com
    secretName: ca-cert
```

设置默认证书：--default-ssl-certificate=default/foo-tls
更改的ingress-controller的启动参数

Dashboard自定义证书

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: kubernetes-dashboard
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        k8s-app: kubernetes-dashboard
```

```
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: normal
                operator: In
                values:
                - "true"
      containers:
      - args:
        - --auto-generate-certificates=false
        - --tls-key-file=server.key
        - --tls-cert-file=server.pem
        - --token-ttl=21600
        - --authentication-mode=basic,token
        - --namespace=kubernetes-dashboard
        image: kubernetesui/dashboard:v2.0.0-rc5
        imagePullPolicy: Always
        lifecycle: {}
        livenessProbe:
          failureThreshold: 3
          httpGet:
            path: /
            port: 8443
            scheme: HTTPS
          initialDelaySeconds: 30
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 30
        name: kubernetes-dashboard
        ports:
        - containerPort: 8443
          protocol: TCP
        resources: {}
        securityContext:
          privileged: false
          procMount: Default
          runAsNonRoot: false
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /certs
          name: kubernetes-dashboard-new
        - mountPath: /tmp
          name: tmp-volume
      dnsPolicy: ClusterFirst
      restartPolicy: Always
```

```
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: kubernetes-dashboard
      serviceAccountName: kubernetes-dashboard
      terminationGracePeriodSeconds: 30
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/master
        operator: Exists
      volumes:
      - name: kubernetes-dashboard-new
        secret:
          defaultMode: 420
          secretName: kubernetes-dashboard-new
      - emptyDir: {}
        name: tmp-volume
```

Ingress Nginx 黑白名单

黑名单：拒绝某段IP访问

白名单：只允许某段IP访问

Annotations：只对指定的ingress生效

ConfigMap：全局生效

黑名单可以使用ConfigMap去配置，白名单建议使用Annotations去配置

白名单配置（建议使用Annotations）

```
# 官网：https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#whitelist-source-range
annotations:
  nginx.ingress.kubernetes.io/whitelist-source-range 10.0.0.0/24,172.10.0.1  # 后面可以跟一个或者多个IP
```

黑名单设置（建议使用ConfigMap）（这是全局生效的）

```
# 因为黑名单可能会不定时加上去，防止恶意攻击的。所以用ConfigMap（热更新）

# 1、更改ingress-nginx的cm
[root@k8s-master01 ~]#  kubectl edit  cm -n ingress-nginx ingress-nginx-controller -oyaml
apiVersion: v1
data:                          # 加上data
```

```
  block-cidrs: 192.168.1.201   # 加上block-cidrs，后面也可以跟多个IP，隔开
kind: ConfigMap
metadata:
  annotations:
    meta.helm.sh/release-name: ingress-nginx
    meta.helm.sh/release-namespace: ingress-nginx
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/version: 0.43.0
    helm.sh/chart: ingress-nginx-3.20.1
  name: ingress-nginx-controller
  namespace: ingress-nginx

# 2、删除ingress-nginx的pod，重新加载配置
[root@k8s-master01 ~]# kubectl get po -n ingress-nginx
[root@k8s-master01 ~]# kubectl delete po -n ingress-nginx --all  # 生产一个一个删，防止配置错误都挂了

# 3、在192.168.1.201节点上，访问ingress代理的域名，然后403表示配置成功
```

针对某个域名设置黑名单--snippet

```
官网参考：https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/nginx-configuration/annotations.md#canary

# 比如ingress代理了www.test.com这个域名，那么想针对这个域名（www.test.com）设置访问黑名单，就编辑这个ingress即可
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  # 在annotations下面加上这几行配置，有多个IP可以deny多个
  annotations:
    nginx.ingress.kubernetes.io/server-snippet: |-
      deny 192.168.1.101;
      deny 192.168.1.102;
      allow all;
# 然后在deny的主机上访问 www.test.com 就403
[root@k8s-master02 ~]# curl ngdemo.qikqiak.com
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ingress Nginx 匹配请求头

```
annotations:
    nginx.ingress.kubernetes.io/server-snippet: |
        set $agentflag 0;

        if ($http_user_agent ~* "(iPhone)" ){    # 匹配规则设置（设置匹配为iPhone的手机，则重定向到下面的url）
          set $agentflag 1;
        }

        if ( $agentflag = 1 ) {
          return 301 https://www.baidu.com;      # 重定向到指定的url
        }
```

Ingress Nginx 速率限制

```
官网参考：https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/nginx-configuration/annotations.md#canary
https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#rate-limiting
```

```
nginx.ingress.kubernetes.io/limit-connections 1 # 连接数只有一个

nginx.ingress.kubernetes.io/limit-rps 1 # 每秒请求数
```

Ingress Nginx 基本认证

文档：https://kubernetes.github.io/ingress-nginx/examples/auth/basic/

同样我们还可以在 Ingress Controller 上面配置一些基本的 Auth 认证，比如 Basic Auth，可以用 htpasswd 生成一个密码文件来验证身份验证。

```
[root@k8s-master01 ~]# htpasswd -c auth foo   # 账号foo   密码 123456
New password:              # 123456
Re-type new password:      # 123456
Adding password for user foo

# 生成一个auth文件
[root@k8s-master01 ~]# ls auth
auth
```

然后根据上面的 auth 文件创建一个 secret 对象：

```
[root@k8s-master01 ~]# kubectl create secret generic basic-auth --from-file=auth
secret/basic-auth created
[root@k8s-master01 ~]# kubectl get secret basic-auth -o yaml
apiVersion: v1
data:
  auth: Zm9vOiRhcHIxJGZzREw0b0xmJHNuUUNDTFkxbTE2N1BkNUdEMHIwcC8K
kind: Secret
metadata:
  name: basic-auth
  namespace: default
type: Opaque
```

然后对上面的 my-nginx 应用创建一个具有 Basic Auth 的 Ingress 对象：

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-with-auth
  annotations:
    # 认证类型
    nginx.ingress.kubernetes.io/auth-type: basic
    # 包含 user/password 定义的 secret 对象名
    nginx.ingress.kubernetes.io/auth-secret: basic-auth
    # 要显示的带有适当上下文的消息，说明需要身份验证的原因
    nginx.ingress.kubernetes.io/auth-realm: 'Authentication Required - foo'
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /
        backend:
          serviceName: my-nginx
          servicePort: 80
```

直接创建上面的资源对象，然后通过下面的命令或者在浏览器中直接打开配置的域名：

```
➜ curl -v http://k8s.qikqiak.com -H 'Host: foo.bar.com'
* Rebuilt URL to: http://k8s.qikqiak.com/
*   Trying 123.59.188.12...
* TCP_NODELAY set
* Connected to k8s.qikqiak.com (123.59.188.12) port 80 (#0)
> GET / HTTP/1.1
> Host: foo.bar.com
> User-Agent: curl/7.54.0
> Accept: */*
```

```
>
< HTTP/1.1 401 Unauthorized
< Server: openresty/1.15.8.2
< Date: Sun, 08 Dec 2019 06:44:35 GMT
< Content-Type: text/html
< Content-Length: 185
< Connection: keep-alive
< WWW-Authenticate: Basic realm="Authentication Required - foo"
<
<html>
<head><title>401 Authorization Required</title></head>
<body>
<center><h1>401 Authorization Required</h1></center>
<hr><center>openresty/1.15.8.2</center>
</body>
</html>
```

我们可以看到出现了 401 认证失败错误，然后带上我们配置的用户名和密码进行认证：

```
➜ curl -v http://k8s.qikqiak.com -H 'Host: foo.bar.com' -u 'foo:foo'
* Rebuilt URL to: http://k8s.qikqiak.com/
*   Trying 123.59.188.12...
* TCP_NODELAY set
* Connected to k8s.qikqiak.com (123.59.188.12) port 80 (#0)
* Server auth using Basic with user 'foo'
> GET / HTTP/1.1
> Host: foo.bar.com
> Authorization: Basic Zm9vOmZvbw==
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: openresty/1.15.8.2
< Date: Sun, 08 Dec 2019 06:46:27 GMT
< Content-Type: text/html
< Content-Length: 612
< Connection: keep-alive
< Vary: Accept-Encoding
< Last-Modified: Tue, 19 Nov 2019 12:50:08 GMT
< ETag: "5dd3e500-264"
< Accept-Ranges: bytes
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
```

```
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

可以看到已经认证成功了。当然出来 Basic Auth 这一种简单的认证方式之外，NGINX Ingress Controller 还支持一些其他高级的认证，比如 OAUTH 认证之类的。

Ingress Nginx 灰度发布/金丝雀发布

3.1、准备2个svc，用于演示

```
[root@k8s-master01 app]# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
my-nginx     ClusterIP   10.104.87.14    <none>        80/TCP     91m
my-nginx1    ClusterIP   10.103.175.67   <none>        80/TCP     2m12s
[root@k8s-master01 app]# curl 10.104.87.14
v1
[root@k8s-master01 app]# curl 10.103.175.67
v2
```

3.2、开启基于ingress的灰度发布

```
官网参考：https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/nginx-configuration/annotations.md#canary
# 开启了灰度发布，才能在同一个ns下创建2个同样域名
# 先创建一个普通的ingress，通过canary.test.com就可以访问到svc my-nginx，版本是v1
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-nginx
  annotations:
    kubernetes.io/ingress.class: "nginx"
```

```
spec:
  rules:
  - host: canary.test.com  # 将域名映射到 my-nginx 服务
    http:
      paths:
      - path: /
        backend:
          serviceName: my-nginx  # 将所有请求发送到 my-nginx 服务的 80 端口
          servicePort: 80
```

3.2.1、基于权重的流量调度

基于权重：基于权重的流量切分的典型应用场景就是蓝绿部署，可通过将权重设置为 0 或 100 来实现。例如，可将 Green 版本设置为主要部分，并将 Blue 版本的入口配置为 Canary。最初，将权重设置为 0，因此不会将流量代理到 Blue 版本。一旦新版本测试和验证都成功后，即可将 Blue 版本的权重设置为 100，即所有流量从 Green 版本转向 Blue。

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-nginx1
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - host: canary.test.com
    http:
      paths:
      - path: /
        backend:
          serviceName: my-nginx1
          servicePort: 80
# 这个是代理v2的ingress，如果使用相同域名，创建会报错
[root@k8s-master01 app]# kubectl apply -f  bbb.yaml
Error from server (BadRequest): error when creating "bbb.yaml": admission webhook
"validate.nginx.ingress.kubernetes.io" denied the request: host "canary.test.com" and
path "/" is already defined in ingress default/my-nginx

# 基于权重
annotations:
  nginx.ingress.kubernetes.io/canary: "true"         # 要开启灰度发布机制，首先需要启用 Canary
  nginx.ingress.kubernetes.io/canary-weight: "30"    # 切30%的流量到v2去，设置100就全部切过去
了

# 创建后查看ingress，可以看到创建了2个代理相同域名的ingress
[root@k8s-master01 app]# kubectl get ingress
NAME         CLASS    HOSTS              ADDRESS          PORTS   AGE
my-nginx     <none>   canary.test.com    10.101.29.125    80      20m
my-nginx1    <none>   canary.test.com    10.101.29.125    80      3m6s
```

验证是否配置成功：

```
[root@k8s-master02 ~]# for i in $(seq 1 3); do curl -s -H canary.test.com; done
v1
v2
v1
```

3.2.2、基于Request Header（还有个never、always不进行演示）

基于 Request Header: 基于 Request Header 进行流量切分的典型应用场景即灰度发布或 A/B 测试场景

注意：当 Request Header **设置为 never 或 always 时，请求将不会或一直被发送到 Canary 版本**，对于任何其他 Header 值，将忽略 Header，并通过优先级将请求与其他 Canary 规则进行优先级的比较。

```
# 基于 Request Header
annotations:
  kubernetes.io/ingress.class: nginx
  nginx.ingress.kubernetes.io/canary: "true"                    # 要开启灰度发布机制，首先需要
启用 Canary
  nginx.ingress.kubernetes.io/canary-by-header-value: canary  # 基于header的流量切分 value
  nginx.ingress.kubernetes.io/canary-by-header: user          # 基于header的流量切分  key
  nginx.ingress.kubernetes.io/canary-weight: "30"             # 会被忽略，因为配置了
canary-by-headerCanary版本
```

验证：

```
➜ for i in $(seq 1 10); do curl -s -H "canary: never" echo.qikqiak.com | grep
"Hostname"; done
Hostname: production-856d5fb99-d6bds
Hostname: production-856d5fb99-d6bds
Hostname: production-856d5fb99-d6bds
Hostname: production-856d5fb99-d6bds
Hostname: production-856d5fb99-d6bds
Hostname: production-856d5fb99-d6bds
Hostname: production-856d5fb99-d6bds
Hostname: production-856d5fb99-d6bds
Hostname: production-856d5fb99-d6bds
Hostname: production-856d5fb99-d6bds

# 流量全部切到v2了，从而实现灰度发布
[root@k8s-master02 ~]# for i in $(seq 1 5); do curl -s -H "user: canary"
canary.test.com; done
v2
v2
v2
v2
v2
```

3.2.3、基于 Cookie

基于 Cookie：与基于 Request Header 的 annotation 用法规则类似。例如在 A/B 测试场景下，需要让地域为北京的用户访问 Canary 版本。那么当 cookie 的 annotation 设置为 `nginx.ingress.kubernetes.io/canary-by-cookie: "users_from_Beijing"`，此时后台可对登录的用户请求进行检查，如果该用户访问源来自北京则设置 cookie `users_from_Beijing` 的值为 `always`，这样就可以确保北京的用户仅访问 Canary 版本。

```
annotations:
  kubernetes.io/ingress.class: nginx
  nginx.ingress.kubernetes.io/canary: "true"    # 要开启灰度发布机制，首先需要启用 Canary
  nginx.ingress.kubernetes.io/canary-by-cookie: "users_from_Beijing"  # 基于 cookie
  nginx.ingress.kubernetes.io/canary-weight: "30"  # 会被忽略，因为配置了 canary-by-cookie
→ for i in $(seq 1 10); do curl -s -b "users_from_Beijing=always" echo.qikqiak.com |
grep "Hostname"; done
Hostname: canary-66cb497b7f-48zx4
Hostname: canary-66cb497b7f-48zx4
Hostname: canary-66cb497b7f-48zx4
Hostname: canary-66cb497b7f-48zx4
Hostname: canary-66cb497b7f-48zx4
Hostname: canary-66cb497b7f-48zx4
Hostname: canary-66cb497b7f-48zx4
Hostname: canary-66cb497b7f-48zx4
Hostname: canary-66cb497b7f-48zx4
Hostname: canary-66cb497b7f-48zx4
```

Ingress Nginx 自定义错误页

```
nginx.ingress.kubernetes.io/server-snippet:
  error_page 404 https://www.baidu.com # 如果访问不存在，跳转到baidu
  # 和nginx的配置没有区别
```

官方错误页面

https://github.com/kubernetes/ingress-nginx/blob/master/docs/examples/customization/custom-errors/custom-default-backend.yaml

Ingress Nginx 监控

```
步骤：https://kubernetes.github.io/ingress-nginx/user-guide/monitoring/
源码地址：https://github.com/kubernetes/ingress-nginx/tree/master/deploy
配置grafana：https://github.com/kubernetes/ingress-
nginx/tree/master/deploy/grafana/dashboards
```

演示

1、背景和环境概述

本文中涉及到的环境中、`prometheus`监控和`grafana`基本环境已部署好。 在`nginx ingress controller`的官方文档中对监控有相应描述 https://kubernetes.github.io/ingress-nginx/user-guide/monitoring/

2、修改prometheus配置

修改`prometheus`的配置，增加对`ingress nginx`的监控配置，可按照官方yaml 进行修改：

```
vim prometheus-configmap.yaml
- job_name: 'ingress-nginx-endpoints'
  kubernetes_sd_configs:
  - role: pod
    namespaces:
      names:
      - ingress-nginx
  relabel_configs:
  - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
    action: keep
    regex: true
  - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scheme]
    action: replace
    target_label: __scheme__
    regex: (https?)
  - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
    action: replace
    target_label: __metrics_path__
    regex: (.+)
  - source_labels: [__address__, __meta_kubernetes_pod_annotation_prometheus_io_port]
    action: replace
    target_label: __address__
    regex: ([^:]+)(?::\d+)?;(\d+)
    replacement: $1:$2
  - source_labels: [__meta_kubernetes_service_name]
    regex: prometheus-server
    action: drop
```

重新apply一下configmap

```
kubectl apply -f prometheus-configmap.yaml
```

3、检查是否生效

打开`prometheus`界面，查看`target`中是否有`ingress nginx`的相关记录

检查查询取值

4、配置grafana图形

在grafana图形中导入模板，模板可以按照官方给出的json文件操作，下载此`json`文件，在`grafana`中导入即可

查看图形