KUB 22: Управление входящими соединениями — ingress

Описание:

Данный контроллер дает возможность обойтись без NodePort и выставить приложение на просто запоминаемый URL. Также дополнительно предоставляет TLS, virtual-name хостинг, subpath маршрутизацию к приложению, расширенные правила обработки трафика.
Существует большое количество реализаций Ingress Controller, которые «под капотом», как правило, используют открытые решения, начиная от nginx и HAproxy, заканчивая более близкими к контейнерам Traefik и Envoy. Кроме того, облачные поставщики еще дают свои Ingress controllers, управляющие их облачными решениями (к примеру, в случае Google — управляет Google Load Balancer). Мы же в рамках текущего задания будем проходить преимущественно Nginx Ingress Controller, как богатый на возможности.
Необходимо разделять 2 сущности в Kubernetes — Ingress и Ingress Controller.

- Ingress Controller — это приложение, которое обрабатывает трафик согласно манифестам (как правило, генерируя по шаблонам конфиги для определенного приложения).
- Ingress — как раз тот манифест, который описывает, как должен обрабатываться трафик и в какие сервисы запросы должны проксироваться.

Итак, начнем наше знакомство с Ingress Controller.
Для начала работы нам нужно включить Ingress Controller. В будущем мы изучим helm, и вы сможете устанавливать nginx ingress controller через него, но пока просто применим официальный манифест, который описывает все необходимые ресурсы:

```
$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes/ingress-nginx/contr
oller-v0.41.2/deploy/static/provider/cloud/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
configmap/ingress-nginx-controller created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx
created
role.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
service/ingress-nginx-controller-admission created
service/ingress-nginx-controller created
deployment.apps/ingress-nginx-controller created
```

```
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
serviceaccount/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
```

Давайте проверим, что у нас создался сервис с внешним адресом, который будет использоваться nginx ingress controller:

```
$ kubectl -n ingress-nginx get deploy
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
ingress-nginx-controller   1/1     1            1           54s
vozerov@mba:~/work/rebrain/kuber/tasks $ kubectl -n ingress-nginx get svc
NAME                                 TYPE           CLUSTER-IP
EXTERNAL-IP      PORT(S)                      AGE
ingress-nginx-controller             LoadBalancer   10.96.232.7
84.201.133.236   80:32141/TCP,443:30524/TCP   58s
ingress-nginx-controller-admission   ClusterIP
10.96.158.230   <none>           443/TCP
58s
```

Как вы видите — у нас запустились поды, которые будут обрабатывать трафик, а также создался сервис, на который мы сможем настраивать наши домены — 84.201.133.236. В случае с self-hosted решениями внешний адрес (или внутренний) должен выдать metallb. Или же вы можете настроить в nginx директиву proxy_pass на адреса ваших нод kubernetes на порты 32141 и 30524 (взяты из примера выше). Таким образом nginx будет все входящие соединения пробрасывать на порты нод кластера, а те в свою очередь будут приводить к сервису ingress-nginx-controller. По сути это NodePort сервис. Но удобнее конечно же использовать metallb и получать отдельный адрес для ingress, который можно прописать в dns.
Давайте создадим deployment с nginx и сервис к нему:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: nginx
  template:
    metadata:
      labels:
        app.kubernetes.io/name: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
---
apiVersion: v1
kind: Service
metadata:
 name: nginx-svc
spec:
 type: ClusterIP
 selector:
   app.kubernetes.io/name: nginx
 ports:
 - port: 80
   targetPort: 80
```

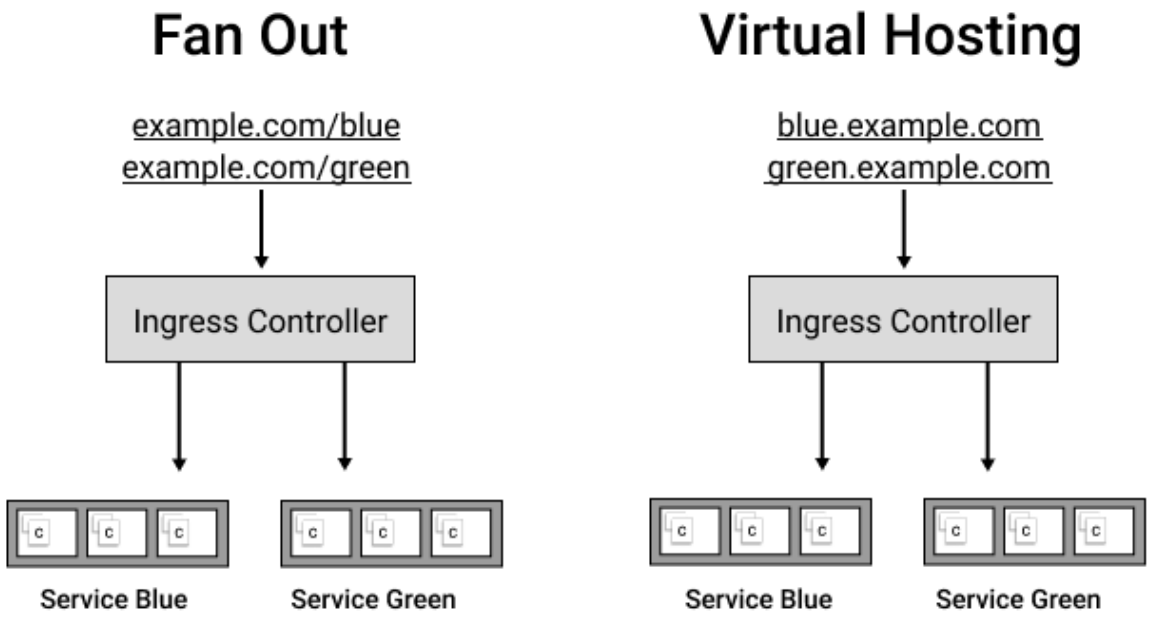
Теперь мы можем создать ingress:

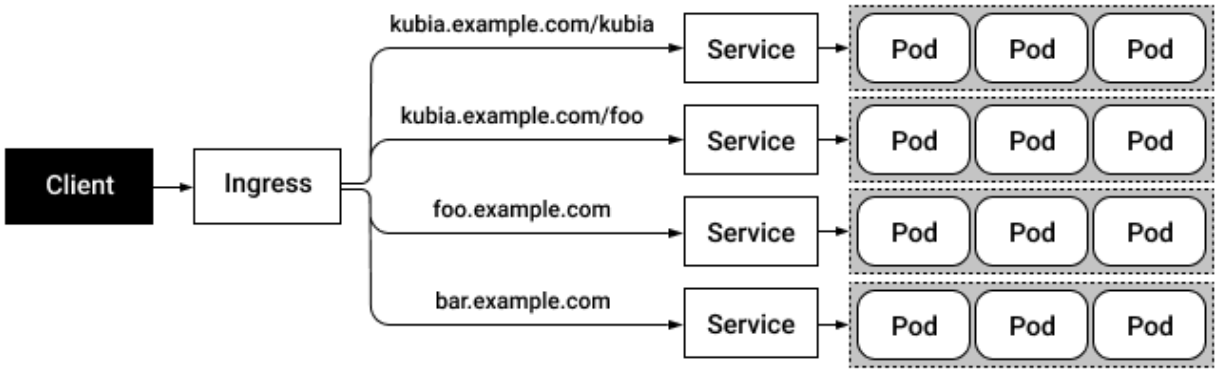
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: web-ingress
  namespace: default
spec:
  rules:
  - host: blue.example.com
    http:
      paths:
      - backend:
          serviceName: nginx-svc
          servicePort: 80
```

Теперь, если мы создадим DNS-запись для адреса blue.example.com, которая смотрит на адрес 84.201.133.236, то мы сможем обращаться по имени к нашему nginx. Давайте попробуем сделать это с помощью curl:

```
$ curl -D - -s -o /dev/null -H 'Host: blue.example.com'
http://84.201.133.236/
HTTP/1.1 200 OK
Date: Fri, 04 Dec 2020 10:12:54 GMT
Content-Type: text/html
Content-Length: 612
Connection: keep-alive
Last-Modified: Tue, 24 Nov 2020 13:02:03 GMT
ETag: "5fbd044b-264"
Accept-Ranges: bytes
```

В текущем виде мы сделали host based virtual hosting, но возможен и другой пример.

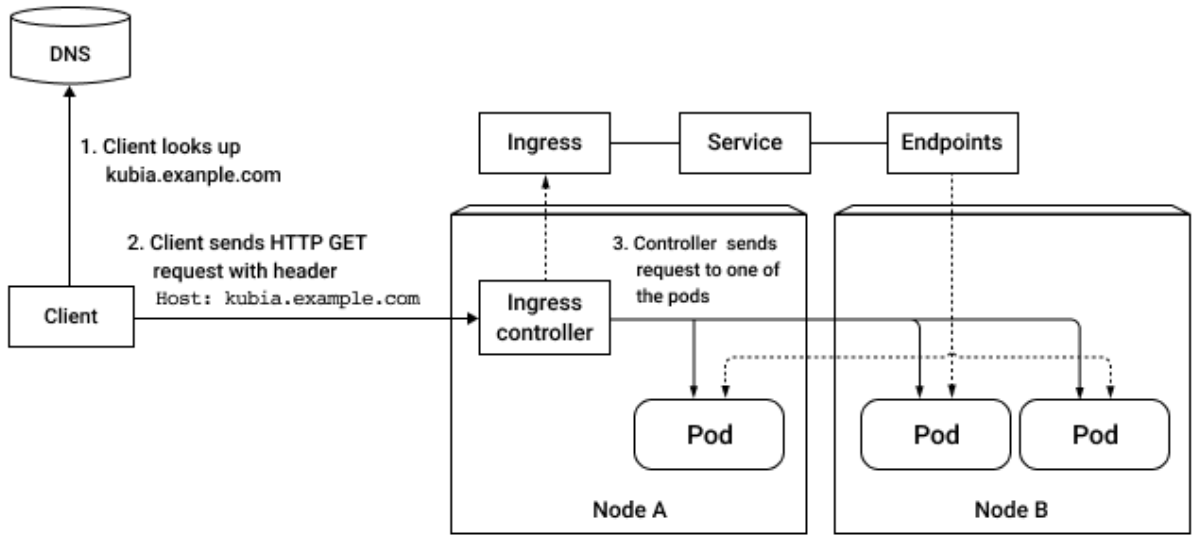




Пример описания ingress для subpath:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-service-ingress
spec:
  rules:
  - host: my.hostname.com
    http:
      paths:
      - path: /my-service
        backend:
          serviceName: my-service
          servicePort: 80
```

В этом примере сервис будет доступен пользователю по адресу my.hostname.com/my-service. Ingress зависит от service, который, в свою очередь, зависит от endpoints.



Некоторые реализации ingress предоставляют дополнительные возможности. Например, такие как автоматическая настройка TLS, авторизация, дополнительные заголовки. Рассмотрим реализацию ingress на nginx. В примере ниже в директиве tls мы добавляем сертификат. Это практически также, как в nginx — при помощи семейства директив ssl*. Для того чтобы это корректно заработало, предварительно нужно создать секрет с сертификатом и ключом.

```
kubectl create secret tls tls-secret --cert=tls.cert
--key=tls.key
```

Затем указать данный сертификат при создании ingress объекта.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: kubia
spec:
  tls:
  - hosts:
    - myhost.example.com
  secretName: tls-secret
  rules:
  - host: myhost.example.com
  http:
    paths:
    - path: /
      backend:
        serviceName: my-service
        servicePort: 80
```

Можете проверить доступность сервиса через 80 и 443 порт вашего ingress.
Множество дополнительных настроек конфигурируется через аннотации ресурса annotations.md.

Полезные ссылки:

- kubernetes annotations
- Labels and Annotations in Kubernetes
- Ingress Controllers (official docs)
- Kubernetes Ingress (comparison)
- Ingress (official docs)

Создание DNS записи в cloudflare через API

P.S. На самом деле мы могли автоматизировать установку ingress контроллера при создании окружения, но мы решили что вы лучше поймете процесс, когда своими руками свяжете DNS имя и IP адрес балансировщика. Если у вас возникнут проблемы на данном шаге - не стесняйтесь и пишите нам в чат.
Для создание DNS записи типа A в cloudflare вы можете использовать curl, пример которого приведен ниже:

```
curl -X POST
"https://api.cloudflare.com/client/v4/zones/9152ec3c08b1a4faeaa9
5353a929fcc5/dns_records" -H "Authorization: Bearer __Bearer
```

```
token__" -H "Content-Type:application/json" --data
'{"type":"A","name":"${domain name}","content":"${ip
address}","proxied":false}'
```

В параметре data надо заменить ${domain name} на параметр, полученный при создании окружения. Так же необходимо заменить параметр ${ip address} на адрес, полученный на 2 шаге.
В ответ на эту команду вы получите уникальный идентификатор записи:

```
{"result":{"id":"f3e4808896173309cbe80254e37225b7","zone_id":"91
52ec3c08b1a4faeaa95353a929fcc5","zone_name":"rbr-kubernetes.com"
,"name":"${domain name}","type":"A","content":"${ip
address}","proxiable":false,"proxied":false,"ttl":1,"locked":fal
se,"meta":{"auto_added":false,"managed_by_apps":false,"managed_b
y_argo_tunnel":false,"source":"primary"},"created_on":"2021-04-2
0T13:58:35.180297Z","modified_on":"2021-04-20T13:58:35.180297Z"}
,"success":true,"errors":[],"messages":[]}
```

В данном случае id - f3e4808896173309cbe80254e37225b7. Он потребуется, если вам придется обновить dns запись на новый адрес (например при пересдаче задания). Для обновления записи воспользуйтесь следующей командой:

```
curl -X PUT
"https://api.cloudflare.com/client/v4/zones/9152ec3c08b1a4faeaa9
5353a929fcc5/dns_records/${dns id}" -H "Authorization:
Bearer__Bearer token__" -H "Content-Type:application/json"
--data '{"type":"A","name":"${domain name}","content":"${ip
address}","proxied":false}'
```

Вместо переменной ${dns id} - подставьте свой идентификатор, который вы получили при создании записи.
После создания записи вы можете проверить созданную запись с помощью dig:

```
dig ${domain name}
```

Если в ответе отображается IP адрес все отлично! Напоминаем, что при изменении записи должно пройти некоторое время для обновления информации в DNS серверах.
После завершения задания запись можно не удалять - мы удалим ее автоматически через некоторое время.

Задание:

1. Установите nginx-ingress контроллер в namespace ingress-nginx
2. Проверьте какой IP адрес получил service LoadBalancer в namespace ingress-nginx
3. Создайте DNS запись: ${domain name}, которая указывает на IP адрес из 2ого шага.

4. Создайте deployment с nginx nginx-dp в namespace default.
5. Создайте сервис для nginx-dp с именем svc-internal в namespace default с типом ClusterIP.
6. Создайте ingress nginx-ingress в namespace default для доменного имени ${domain name}.
7. Отправьте задание на проверку.