Сетевая подсистема в Kubernetes

Не забудь включить запись!

План

- Немного теории о том, как устроен Kubernetes
- Механизм Services
- DNS
- Механизм Ingress

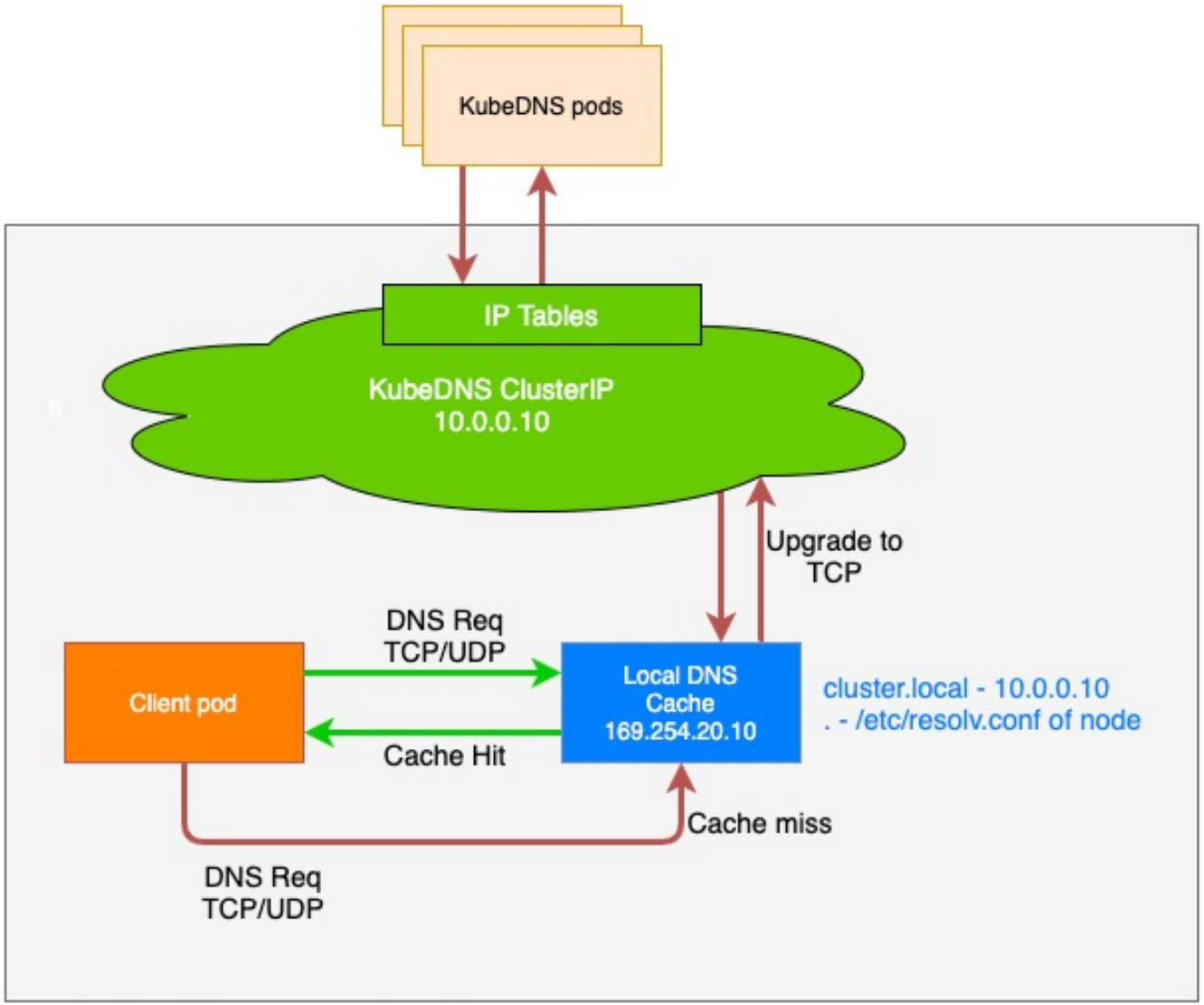
Service

Создает имя, которое можно запросить при помощи DNS

Виды сервисов

- ClusterIP
- NodePort
- LoadBalancer

DNS


KubeDNS pods
IP Tables
KubeDNS ClusterIP
10.0.0.10
Upgrade to
TCP
DNS Req
TCP/UDP
Client pod
Local DNS
Cache
169.254.20.10
cluster.local - 10.0.0.10
. - /etc/resolv.conf of node
Cache Hit
Cache miss
DNS Req
TCP/UDP

Как устроен резолв?

`cat /etc/resolv.conf`

```
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- Откуда это взялось? Кто это положил?
- Почему cluster.local?
- Почему `10.96.0.10`? А почему один?

Волнующая роль kubelet

- Из файла `/var/lib/kubelet/config.yaml`:
  - Адреса резолверов - параметр **clusterDNS**
  - Домен кластера - параметр **clusterDomain**
- Через параметры запуска `kubelet` (deprecated)
  - Адреса резолверов - `--cluster-dns`
  - Домен кластера - `--cluster-domain`

dnsConfig

- Мы в любой момент можем прописать дополнительные опции к `/etc/resolv.conf` в Pod при помощи секции `dnsConfig`
- Она работает с любым `dnsPolicy`, но становится обязательной, если мы выставляем `dnsPolicy: None`

Огласите же список, пожалуйста

Внутри кластера возможно обращение по следующим именам

- Обращение к сервису внутри namespace: `service`
- Обращение к сервису внутри кластера: `service.namespace`
- FQDN сервиса: `service.namespace.svc.<домен кластера>.`
- FQDN пода: `10-0-0-1.ns.pod.cluster.local**` (Deprecated, но пока работает)

> Описание текущей схемы DNS

F...amous ndots problem

По умолчанию, в `resolv.conf` добавляется опция `options ndots:5`

- Это значит, что DNS-запросы, где меньше 5 точек (или не FQDN) обрабатываются так:
  - дописываем домен из списка `search` и пробуем резолвить
  - если не вышло, берем следующий домен, снова пробуем
- Обычно, список такой:

```
default.svc.cluster.local svc.cluster.local cluster.local
```

F...amous ndots problem

- Alpine часто используется в роли базового образа в Kubernetes
- А там вместо glibc с ее кучей известных особенностей применяется musl libc
    - Краткий разбор, полный разбор
    - Рассказ о том, как это может быть больно

Вернемся к Service

Напомню

Виды сервисов

- **ClusterIP**
- NodePort
- LoadBalancer

Каждый Pod имеет свой собственный IP
При пересоздании (может быть очень часто) Pod-а IP меняется

Сервис

- имеет постоянный IP
- обеспечивает балансировку
- облегчает взаимодействие внутри и вне кластера

ClusterIP

Тип сервиса по-умолчанию

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

ClusterIP

Допускается задавать несколько портов (multi-port)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 9376
    - name: https
      protocol: TCP
      port: 443
      targetPort: 9377
```

Сервис под капотом…

```
# kubectl get svc
NAME          TYPE         CLUSTER-IP     EXTERNAL-IP   PORT(S)      AGE
my-service    ClusterIP    10.104.92.240  <none>        80/TCP       5h43m

# iptables-save | grep "my-service"
-A KUBE-SEP-74M76R3MOH5BJ7FU -s 10.0.2.51/32 -m comment --comment "default/my-service" -j KUBE-MARK-MASQ
-A KUBE-SEP-74M76R3MOH5BJ7FU -p tcp -m comment --comment "default/my-service" -m tcp -j DNAT --to-destination
10.0.2.51:9376

-A KUBE-SEP-JPAADZTWUNYJQEJV -s 10.0.1.30/32 -m comment --comment "default/my-service" -j KUBE-MARK-MASQ
-A KUBE-SEP-JPAADZTWUNYJQEJV -p tcp -m comment --comment "default/my-service" -m tcp -j DNAT --to-destination
10.0.1.30:9376

-A KUBE-SEP-KVEHKWBOWX3KCL2I -s 10.0.2.167/32 -m comment --comment "default/my-service" -j KUBE-MARK-MASQ
-A KUBE-SEP-KVEHKWBOWX3KCL2I -p tcp -m comment --comment "default/my-service" -m tcp -j DNAT --to-destination
10.0.2.167:9376

-A KUBE-SERVICES -d 10.104.92.240/32 -p tcp -m comment --comment "default/my-service cluster IP" -m tcp --dport 80 -j
KUBE-SVC-FXIYY6OHUSNBITIX

-A KUBE-SVC-FXIYY6OHUSNBITIX -m comment --comment "default/my-service" -m statistic --mode random --probability
0.33333333349 -j KUBE-SEP-JPAADZTWUNYJQEJV
-A KUBE-SVC-FXIYY6OHUSNBITIX -m comment --comment "default/my-service" -m statistic --mode random --probability
0.50000000000 -j KUBE-SEP-KVEHKWBOWX3KCL2I
-A KUBE-SVC-FXIYY6OHUSNBITIX -m comment --comment "default/my-service" -j KUBE-SEP-74M76R3MOH5BJ7FU
```

А что за NodePort?

- Итак, следующим идет `type: NodePort`
  - ClusterIP
  - **NodePort**
  - LoadBalancer

Ничего особенного

- На **каждой** ноде открывается один и тот же порт из особого диапазона
- Который делает DNAT на адреса Pods

А что за диапазон портов?

- За это отвечает параметр `kube-apiserver` под именем `--service-node-port-range`
- По умолчанию это 30000 - 32767
    - Не самая лучшая идея этот диапазон менять, если что

Пример NodePort-сервиса

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app: MyApp
  ports:
    - port: 80
      targetPort: 8081
      nodePort: 30007
```

LoadBalancer

- Итак, следующим идет `type: LoadBalancer`
  - ClusterIP
  - NodePort
  - **LoadBalancer**

Итак, давайте побалансируем?

- Этот вид сервиса предназначен только для облачных провайдеров
- Которые своими программными средствами реализуют балансировку нагрузки
- В Bare Metal это делается чуть иначе - понадобится MetalLB

Пример LoadBalancer-сервиса

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: LoadBalancer
  selector:
    app: MyApp
  ports:
    - port: 80
      targetPort: 8081
```

Ingress

Ingress

Объект управляющий внешним доступом к сервисам внутри кластера. Обеспечивает:

- Организацию единой точки входа в приложения снаружи
- Балансировку трафика
- Терминацию SSL
- Виртуальный хостинг на основе имен и т.д.

Работает на L7 уровне.

Ingress Controller

Ingress - набор правил внутри кластера Kubernetes.

Для применения данных правил нужен **Ingress Controller** - плагин который состоит из 2-х функциональных частей:

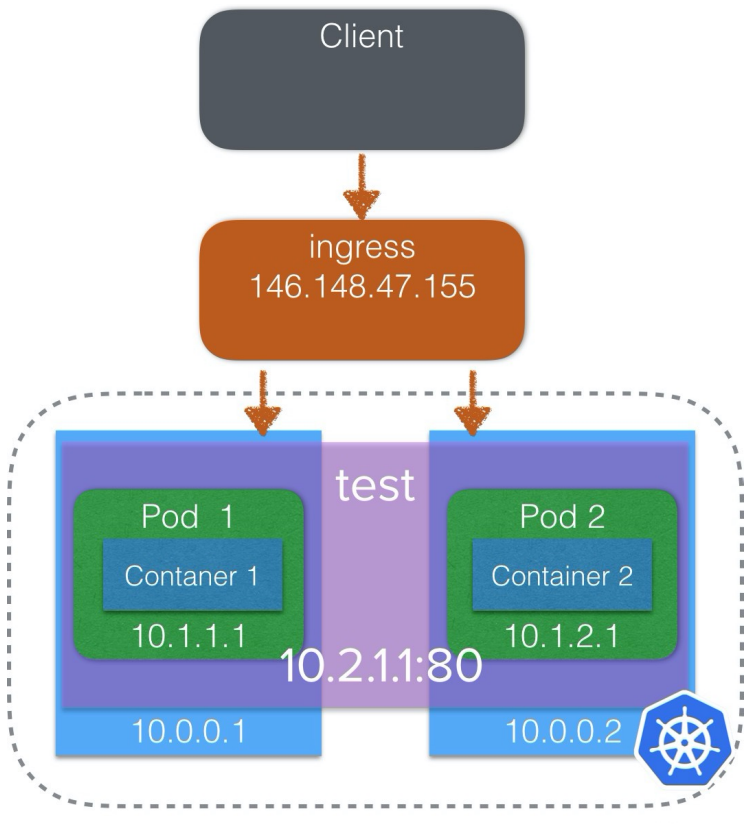
- Приложение, которое отслеживает через Kubernetes API новые объекты Ingress и обновляет конфигурацию балансировщика
- Сам балансировщик (nginx,HAProxy, traefik, ...), который отвечает за управление сетевым трафиком

Разные виды Ingress Controller и их возможности

Хороший обзор про затаскивание внешнего трафика

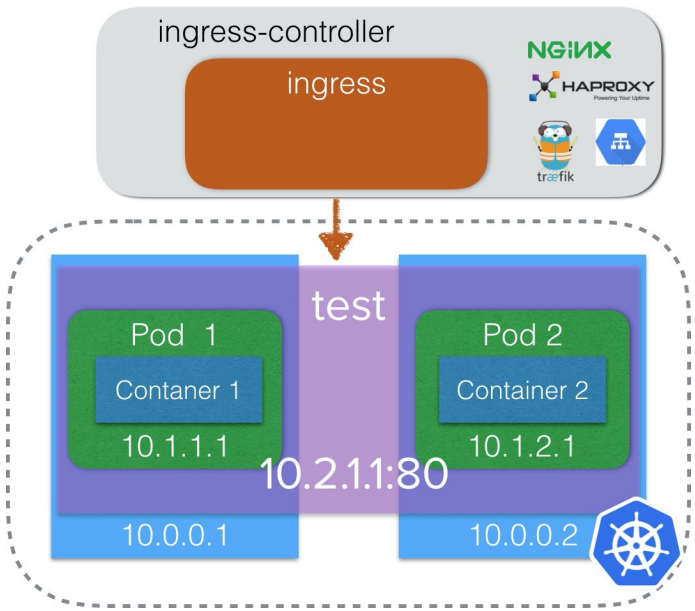
Ingress

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```



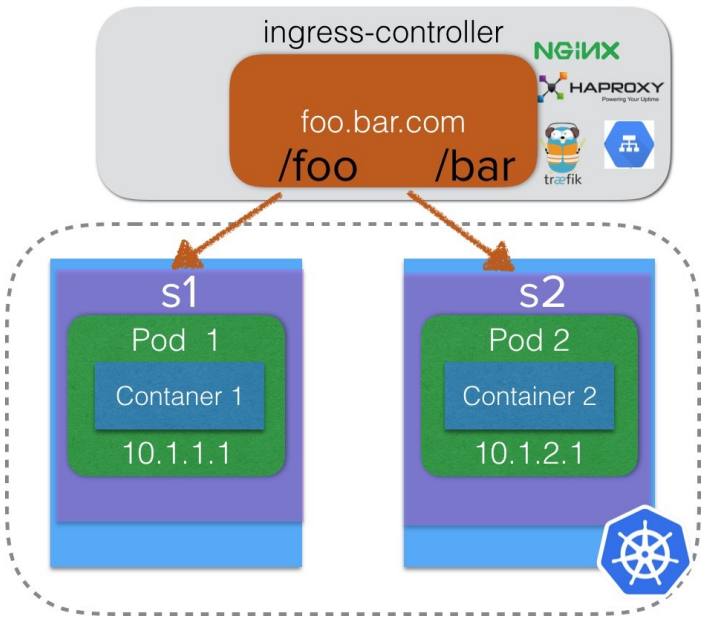
Single Service Ingress

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress
spec:
  backend:
    serviceName: test
    servicePort: 80
```



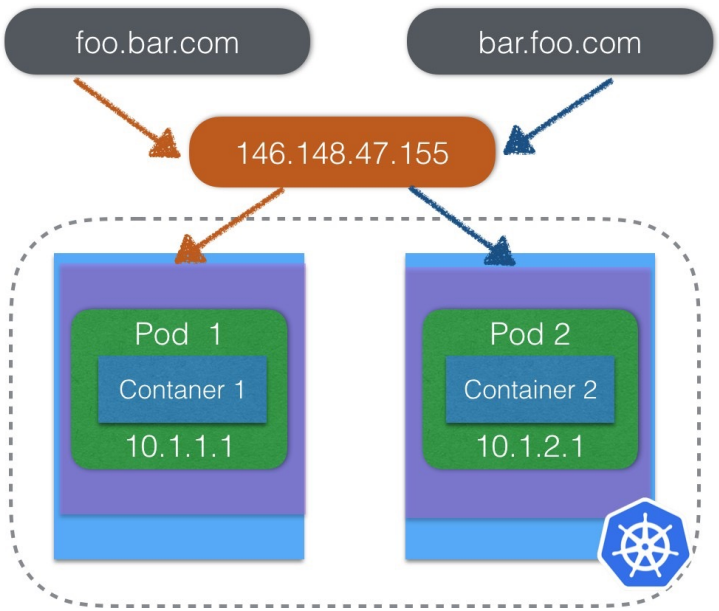
Simple Fanout

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: s1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: s2
          servicePort: 80
```



Name Based Virtual Hosting

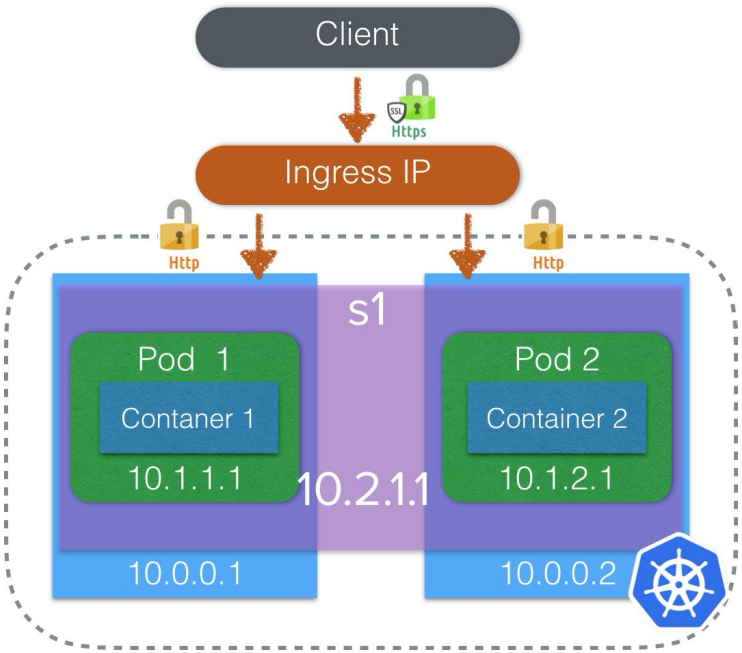
```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: s1
          servicePort: 80
  - host: bar.foo.com
    http:
      paths:
      - backend:
          serviceName: s2
          servicePort: 80
```



TLS termination

```
apiVersion: v1
kind: Secret
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
metadata:
  name: testsecret-tls
  namespace: default


apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: tls-example-ingress
spec:
  tls:
  - hosts:
    - sslexample.foo.com
    secretName: testsecret-tls
  rules:
  ...
```



На посошок

`hostNetwork: true`

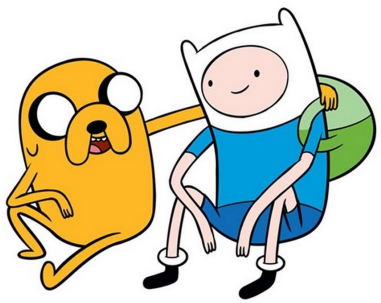
- Если я указываю для Pod это свойство, то сетевой Namespace не создается
    - Pod напрямую видит сетевые адаптеры ноды
- Так работают Pod, которые реализуют сетевую подсистему, например
- Никто не мешает сделать так Ingress Pod с ingress-nginx

hostPort для контейнера

- Я могу указать для контейнера порт, который будет открыт на ноде
  - На ноде, где в итоге оказался Pod
- Так не могут делать несколько контейнеров по понятной причине
- Опять же, для Ingress Pod почему бы и нет

externalIP для Service

- Я могу указать конкретные внешние сетевые адреса для сервиса
- Тогда в iptables будут созданы необходимые пробросы внутрь
  - Мы уже их видели



Спасибо за внимание!

Время для ваших вопросов!