**#SERVICES - доступ к группе подов. Служба имеет не меняющиеся ip и порт с которых маршрутизируются запросы в один подов этой службы (с балансировкой между всеми подами)**

**-Статический IP**

**-DNS имя в kube-dns на этот IP**

**-правила в iptables для роутинга**

**С сервисами создаются EndPoints (ip адреса подов (get ep)) куда пересылаются пакеты, при удалении/добавлении подов меняются EP и не менятеся адрес сервиса**

**IP адрес сервиса «виртуальный» это NAT преобразование - правила как в iptables (если приходит запрос на адрес сервиса:порт то надо переправить на адрес пода:порт)**

**Виды сервисов:**

***ClusterIP*** IP только ВНУТРИ k8s cluster (default)

***NodePort*** для службы NodePort каждый узел кластера открывает порт на самом узле (отсюда и название) и перенаправляет трафик, получаемый на этом порту, в базовую службу. Служба доступна не только через внутренний IP-адрес и порт кластера, но и через выделенный порт на всех узлах;

apiVersion: v1

kind: Service

metadata:

name: x3m55-nodeport

spec:

type: NodePort #тип службы

ports:

- port: 80 #порт внутреннего кластерного IP службы

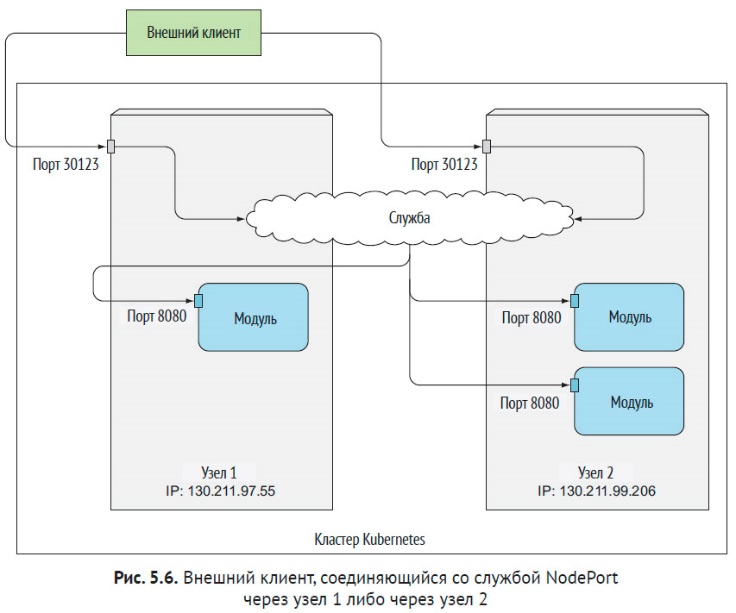
targetPort: 8080 #целевой порт модулей

nodePort: 30123 #служба доступна через порт 30123 каждого узла, можно зайти снаружи на ip нода и этот порт, можно не указывать k8s сам выдаст

selector:

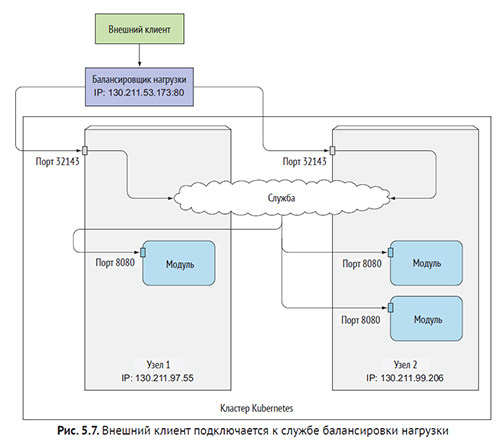
app: prod

Подключение, полученное на порту 30123 первого узла, могло быть перенаправлено либо к модулю, работающему на первом узле, либо к одному из модулей, работающих на втором узле.

******

***LoadBalancer*** Доступ извне в балансировщик - только в облачных кластерах AWS, GCP, Azure. Балансировщик нагрузки перенаправляет трафик на порт узла во всех узлах. Клиенты подключаются к службе через IP-адрес балансировщика нагрузки

Служба с типом LoadBalancer – это служба NodePort с дополнительной подсистемой балансировки нагрузки



Cлучайно выбранный модуль может не работать на том же узле, который получил подключение. Для достижения модуля потребуется дополнительный сетевой переход (hop), но это не всегда желательно.

Этот дополнительный переход можно предотвратить, настроив службу для перенаправления внешнего трафика только на те модули, которые работают на узле, получившем подключение :

spec:

externalTrafficPolicy: Local

Политика внешнего трафика Local, влияет на сохранение исходного IP-адреса, так как между узлом, принимающим подключение, и узлом, на котором размещен целевой модуль, нет дополнительного перехода (SNAT не выполняется)

***ExternalName*** Подключение к внешней службе по доменному имени через DNS CNAME запись (api.somecompany.com)

spec:

type: ExternalName #тип службы

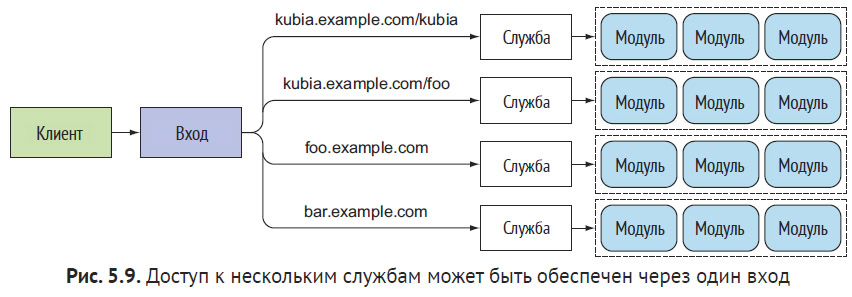
externalName: someapi.somecompany.com #полное доменное имя службы

ports:

– port: 80

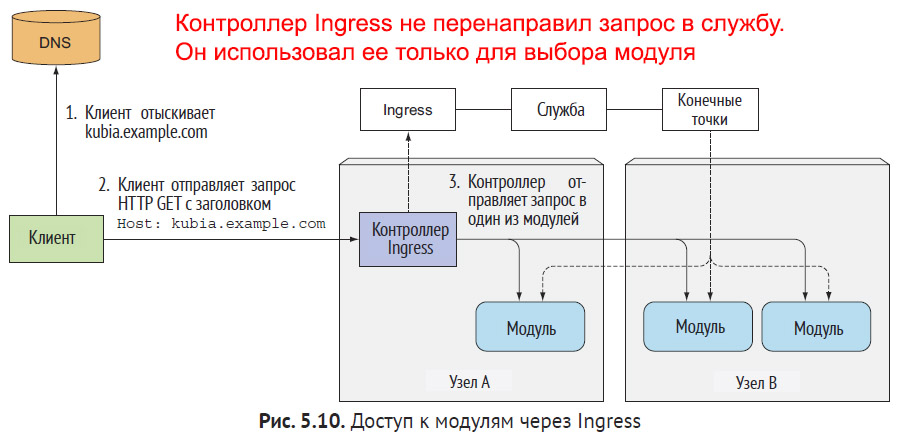
**Ingress**

***Ingress*** Радикально отличающийся механизм предоставления доступа к нескольким службам через единый IP-адрес – он работает на уровне HTTP (сетевой уровень 7) и, следовательно, может предложить больше возможностей, чем службы уровня 4



- В манифесте Ingress правила обработки запросов при запросе имени хоста (можно добавить путь) привязка на имя и порт сервиса который обработает запрос, в манифесте сервиса селектор подов на которые придет запрос

- Клиент обращается к ДНС - возвращает IP адрес ингресс контроллера, затем отправляет HTTP-запрос с заголовком имени сайта, из этого заголовка инг. контроллер определяет к какой службе обращается клиент, находит IP-адрес пода через связанный со службой Endpoints и перенаправляет запрос клиента на под



**Liveness, Readines, Startup:**

**Startup Probe:** проверяет запустилось ли приложение, выполняется при старте. После прохождения этой проверки начинаются **readiness** и **liveness**

Проверка готовности **Readines**: при непрохождении исключается из Endpoints в службе, сам под не удаляется (как если удалить label из пода), выполняется постоянно

Проверка живучести **Liveness:** контроль за состоянием во время жизни пода, при непрохождении перезапускается под, выполняется постоянно

Три механизма проверки:

HTTP GET - запрос на ip, порт и путь контейнера, если отклик без ошибки, проверка проходит (если код отклика 2хх или 3хх)

TCP сокет - проверка подключение по TCP

Exec - выполнение произвольной команды в контейнере, если код состояния 0, проверка проходит, ост коды - непроходит

**HEADLESS (Безголовая служба) использование службы без точки входа**

Обычно, когда вы выполняете DNS-запрос,DNS-сервер возвращает единственный кластерный IP-адрес службы. Но если вы сообщите системе Kubernetes, что для вашей службы вам не нужен кластерный IP-адрес (это можно сделать, присвоив полю clusterIP значение None в спецификации службы), то вместо единственного IP-адреса службы DNS-сервер будет возвращать IP-адреса модулей.

- Селектор меток службы применяется ко всей службе, не может быть настроен для каждого порта индивидуально, если надо чтобы разные порты увязывались с разными множествами подов, надо делать две службы

- ***Обнаружение служб с помощью переменных среды*** : k8s добавляет переменные в поды с адресами и портом службы (команда env)

- ***Обнаружение служб через DNS*** : k8s запускает под с DNS-сервером, для использования которого автоматически настраиваются ост поды в кластере (k8s изменяет /etc/resolv.conf каждого контеёнета) Любой DNS-запрос, выполняемый процессом, запущенным в модуле, будет обрабатываться собственным DNS-сервером Kubernetes, который знает все службы, работающие в вашей системе

- ***Подключение к службам, находящимся за пределами кластера:***При создании службы по селектору меток создаются конечные точки (Endpoints) ip-адреса подов внутри кластера. Можно создать службу с доступом за пределы кластера, для этого создается служба без селектора подов (external-service.yaml) и ресурс kind: Endpoints со вписком конечных точек и порта(external-service-endpoints.yaml). Имя объекта Endpoints должно совпадать с именем службы

*apiVersion: v1*

*kind: Service*

*spec:*

*ports:*

*– name: http*

*port: 80*

*targetPort: http* #ранее в манифесте созданного пода указано имя порта http, можно ссылатся не на цифру а на имя порта, при изменении цифры не нужно менять имя

-----

*spec:*

*sessionAffinity: ClientIP* #заставляет служебный прокси перенаправлять все запросы, исходящие от того же клиентского IP-адреса **на один и тот же** модуль

-----