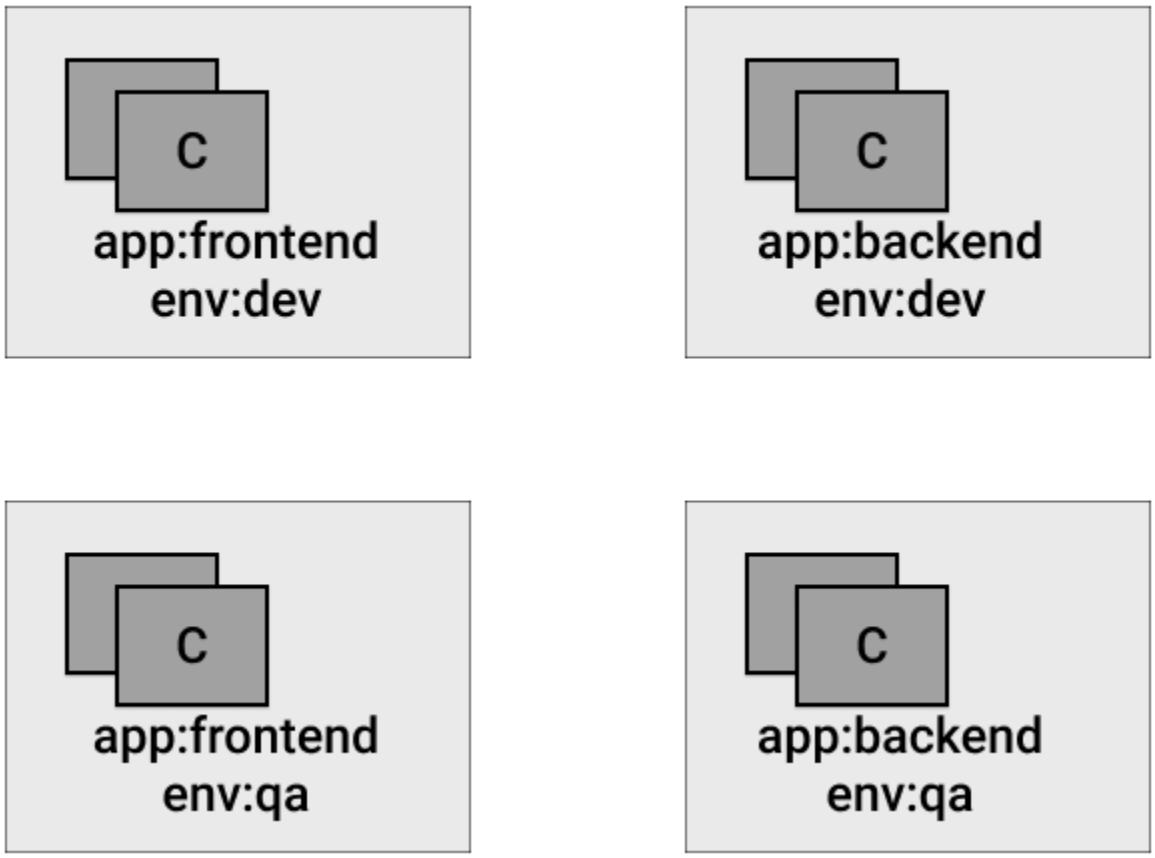
KUB 14: Параметры манифестов

Описание:

Мы уже использовали много параметров при создании deployments и подов, но важно более детально разобраться с параметрами манифестов, чтобы использовать их более правильно.
Labels
Метки (labels) — это пара key-value, которая назначается объектам Kubernetes, например, Pod. Они используются для организации групп объектов или для более удобного управления объектами. Метки не должны быть уникальными.
Например, можно поставить метки на Pod.



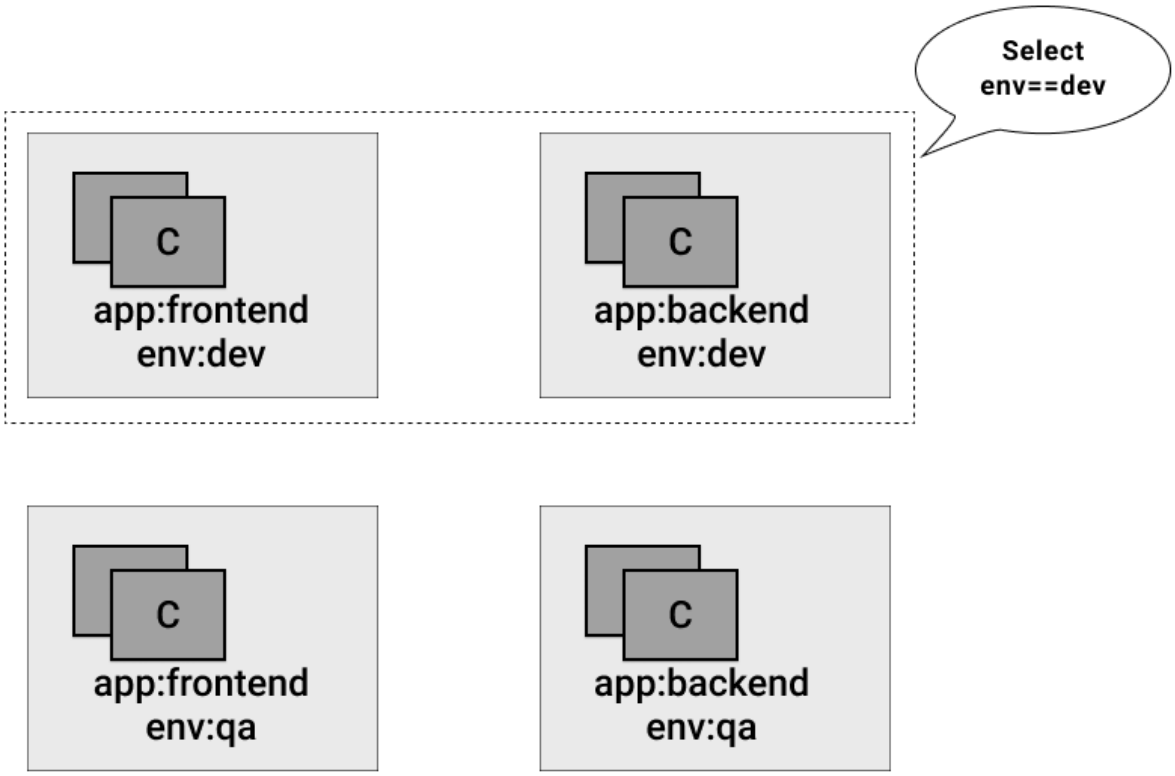
Kubectl позволяет выводить ресурсы, содержащие определенную метку, при помощи флага -l.
К примеру, для вывода всех компонентов с меткой app: nginx можно выполнить команду kubectl get all -l app=nginx.
Выбор объектов по меткам (labels) на основе правил производится очень просто. Есть два типа правил. Это то, что мы прописывали в spec.selector.

Выборка на основе равенства

Эта выборка сопоставляет метки на заданном объекте. Можно использовать операторы =, ==, !=. Например, env=dev выберет объекты с ключом env и значением dev.

Выборка на основе множества

Множества позволяют более точно фильтровать объекты. Можно использовать операторы in, notin, exists. Например, env in (dev,qa) выберет несколько объектов с меткой env и значениями dev или qa.



Пока мы применяли метки только в Service (выбор подов, на которые он должен балансировать нагрузку) и Deployment, но на этом список не заканчивается, и мы с вами еще встретимся с их использованием в последующих задачах.
Annotations
Аннотации достаточно похожи на метки, но их не получится использовать для выборки или группировки объектов. Они представляют собой произвольное текстовое поле в формате key=value. Например, в описании объекта можно указать, что это такое, зачем это нужно.

```
apiVersion: v1
kind: Pod
metadata:
  name: annotations-demo
  annotations:
```

```
    imageregistry: "https://hub.docker.com/"
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```

Некоторые контроллеры получают информацию из annotations, например ingress, helm, cert-manager.
Кроме того, некоторые облачные провайдеры через аннотации получают конфигурацию для сервисов вне самого Kubernetes. Посмотреть аннотации объекта можно через kubectl get pod $pod -o yaml, а добавить — kubectl annotate pod $pod mypod="super".
Liveness Probe
Проверки работоспособности приложения внутри контейнера очень важны. Это поможет Kubernetes своевременно рестартовать приложение. Liveness probe проверяет приложение на работоспособность. Есть три типа встроенных проверок:

- liveness command — команда выполняется внутри контейнера. Если результат выполнения отличен от нуля, то проверка считается проваленной.
- liveness http request — данная проверка выполняет HTTP GET на указанный адрес, ожидает ответ 200 в случае успеха.
- tcp liveness probe — проверка на установление TCP-соединения на указанный порт. Не подразумевает передачу данных, только проверяет, установлено ли соединение.

Пример liveness command:

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-exec
spec:
  containers:
  - name: liveness
    image: k8s.gcr.io/busybox
    args:
    - /bin/sh
    - -c
    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep
600
    livenessProbe:
      exec:
```

```
      command:
      - cat
      - /tmp/healthy
    initialDelaySeconds: 5
    periodSeconds: 5
```

Команда проверяет наличие файла /tmp/healthly. Заметьте дополнительные параметры:

- initialDelaySeconds — параметр задержки первого запуска проверки. Например, если приложение долго запускается, а данный параметр не задан, то kubelet убьет pod раньше, чем приложение будет готово. В такой конфигурации первая проверка будет запущена спустя 5 секунд после запуска контейнера.
- periodSeconds — период запуска проверки, в данном случае 5 секунд.

Пример liveness HTTP Request:

```
livenessProbe:
    httpGet:
      path: /healthz
      port: 8080
      httpHeaders:
      - name: X-Custom-Header
        value: Awesome
    initialDelaySeconds: 3
    periodSeconds: 3
```

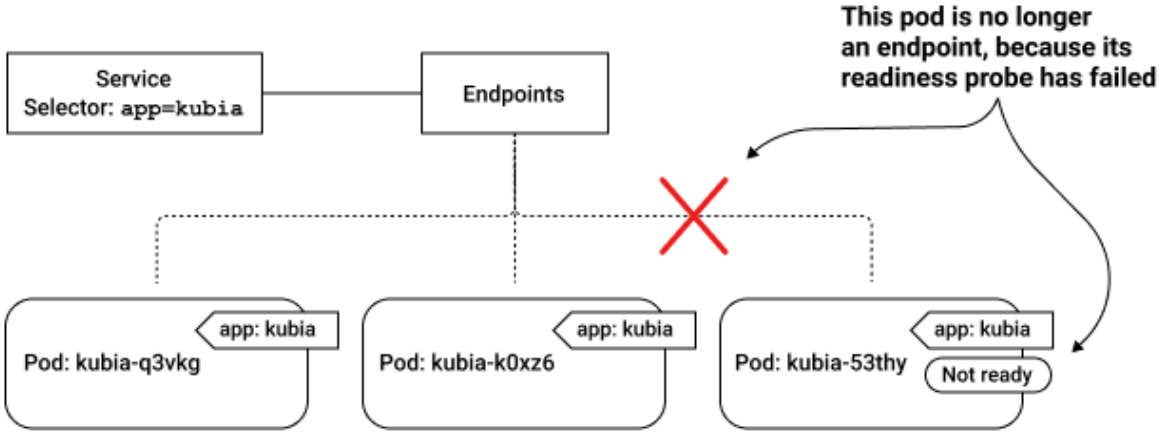
В этой конфигурации будет выполнен HTTP GET на /healthz, порт 8080 с дополнительными заголовками.

Пример TCP liveness probe:

```
livenessProbe:
    tcpSocket:
      port: 8080
    initialDelaySeconds: 15
    periodSeconds: 20
```

В этом варианте проверяется соединение на порт 8080/TCP.

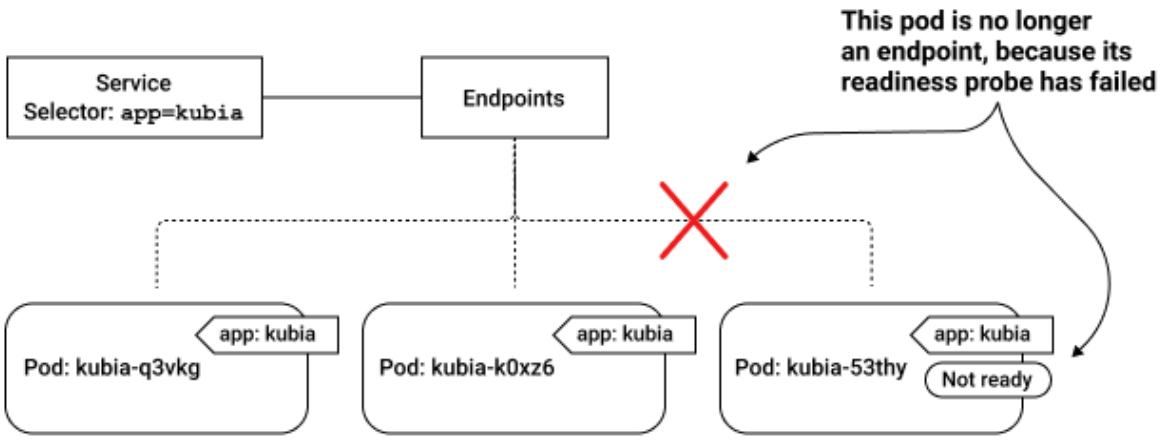
Readiness Probe



Это проверка готовности приложения к работе. Например, приложение долго запускалось, прошло проверку доступности, но еще не готово обрабатывать трафик пользователей, так как загружает кеши в redis. Данный тип проверок помогает kubelet понять, в какой момент времени можно начать направлять трафик на приложение.
Те же типы проверок, как в Liveness Probe, только описываются в spec.containers.readinessProbe, пример:

```
readinessProbe:
  exec:
    command:
    - cat
    - /tmp/healthy
  initialDelaySeconds: 5
  periodSeconds: 5
```

Startup Probe



Это проверка выполняется для того, чтобы удостовериться, что приложение запустилось. Часто бывает, что приложение долго стартует — например, оно читает много файлов с диска, чтобы наполнить кеш. И при этом liveness probes проваливаются, поскольку

приложение не успело запуститься. Чтобы не выставлять большие initDelaySeconds, и существует startupProbe. Работает он также, как и другие проверки:

```
startupProbe:
  httpGet:
    path: /healthz
    port: liveness-port
  failureThreshold: 30
  periodSeconds: 10
```

В данном случае приложению будет дано 300 секунд на запуск, после чего начнут выполняться проверки liveness & readiness probes.

Полезные ссылки:

- Kubernetes Liveness and Readiness Probes
- Liveness probes в Kubernetes могут быть опасны (habr)
- kubernetes annotations
- Labels and Annotations in Kubernetes
- Setting up HTTP Load Balancing with Ingress -- Step 5: (Optional) Configuring a static IP address
- Labels and Selectors (official docs)
- Configure Liveness, Readiness and Startup Probes (official docs)
- Labels and Annotations in Kubernetes

Задание:

1. Создайте деплоймент с именем nginx-mf в namespace default со следующими правилами:
- Образ должен быть nginx:latest
- Добавьте label env=prod.
- Добавьте аннотацию prometheus.io/scrape=true.
- Добавьте liveness probe http на порт 80 с произвольными значениями period & delay.
- Добавьте readiness probe command с вызовом curl http://127.0.0.1:80.

2. Отправьте задание на проверку.