СТРПО

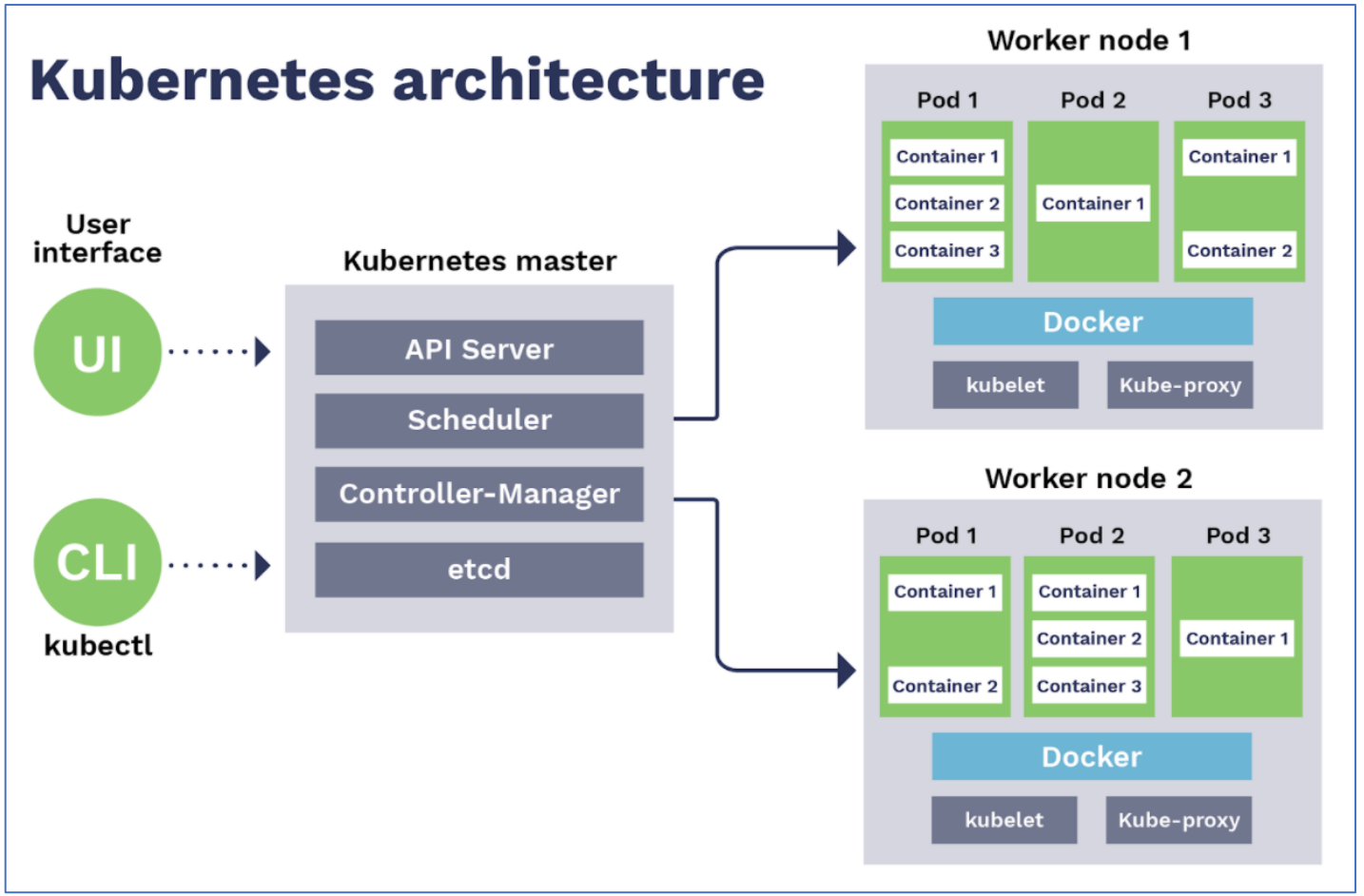
Семинар к лабораторным работам №2

Содержание

- Модель Kubernetes
- Декларативное управление кластером
- Взаимодействие сервисов
- Метки
- Непрерывное обновление

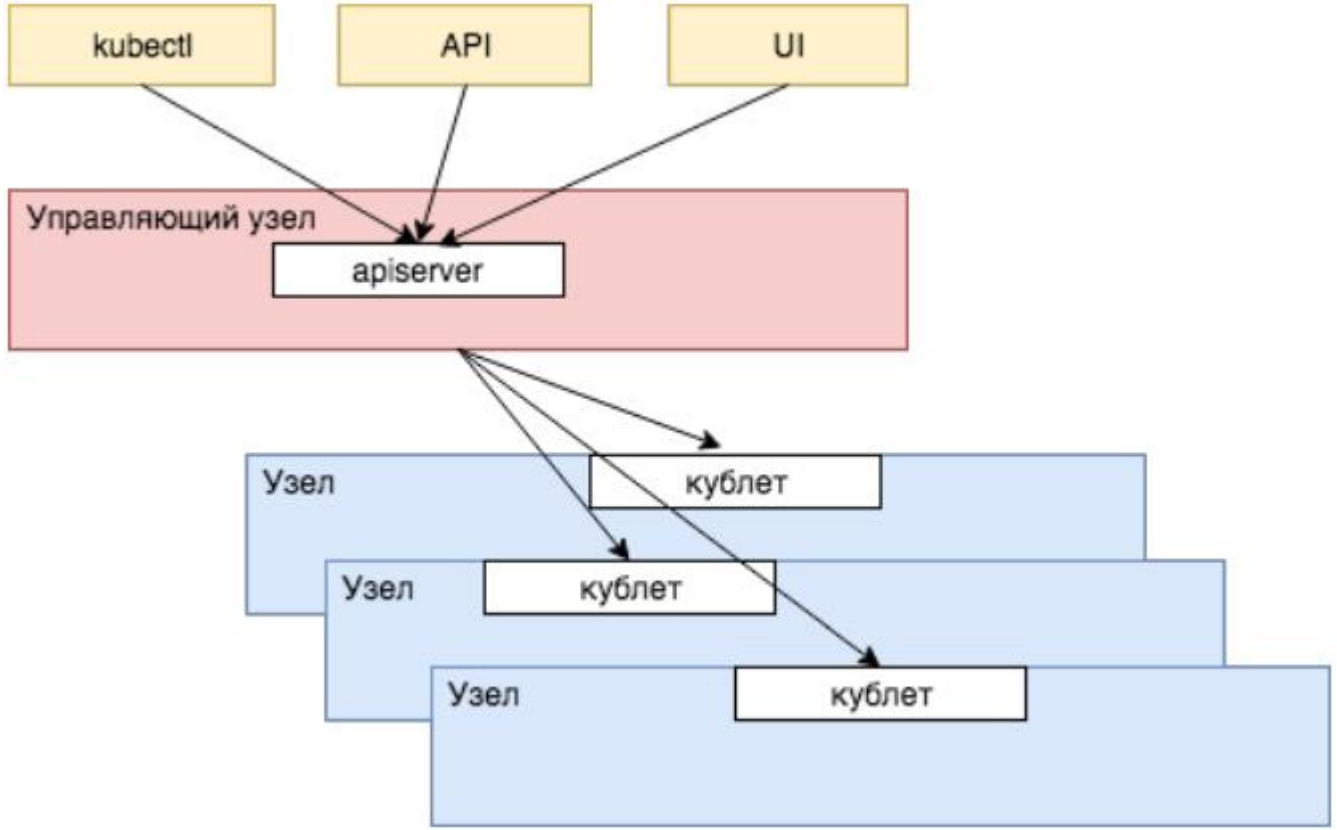
Модель Kubernetes

- Кластер
- Управляющий узел
- Рабочий узел
- Отсек
- Развёртывание
- Служба



Модель Kubernetes: Управление

- На узлах запускаются отсеки
- Кублет – агент Kubernetes
- Формат команд: JSON, YAML

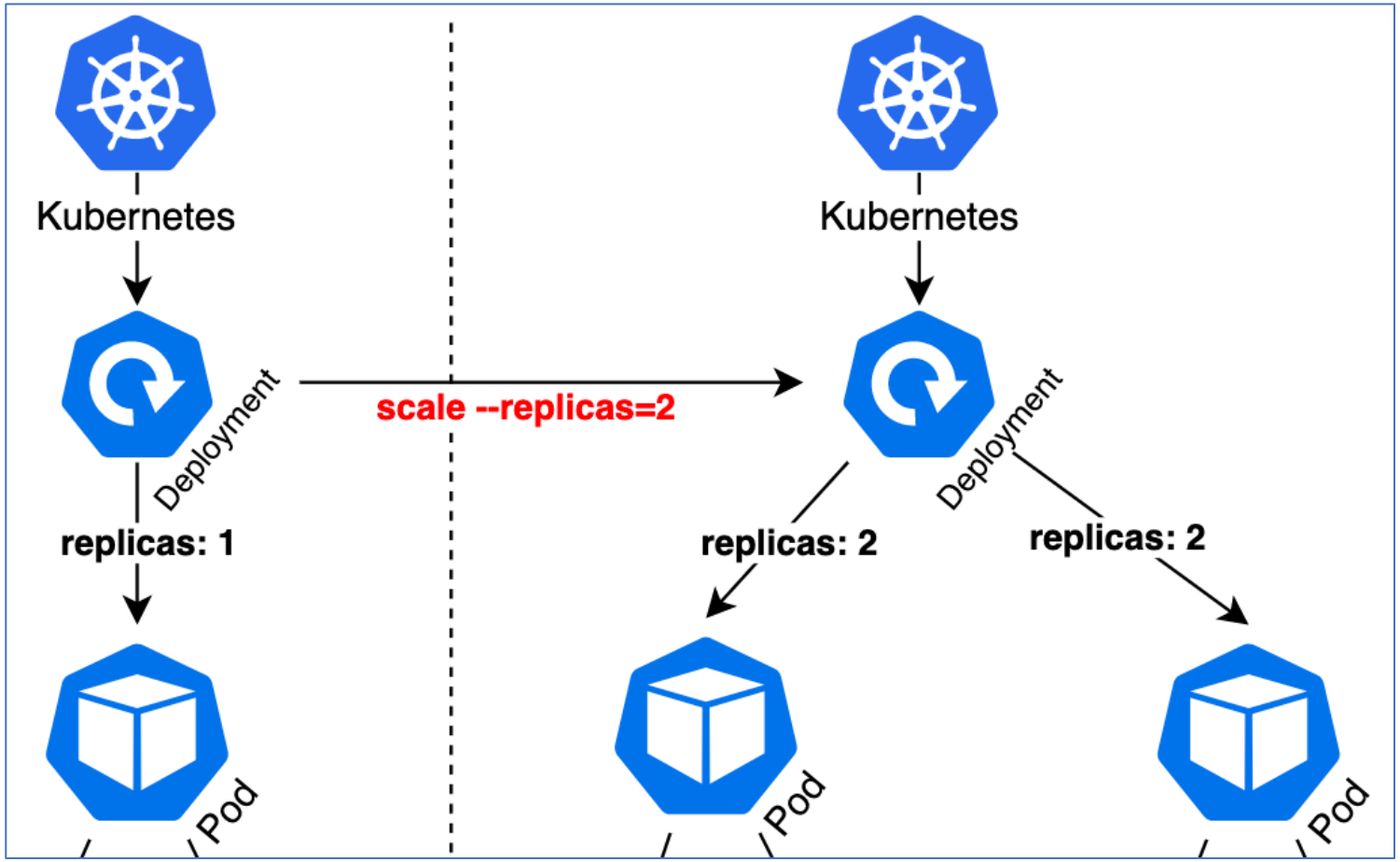


Модель Kubernetes: Отсек

- Наименьшая логическая единица кластера
- Как правило соотношение контейнеров к отсекам 1:1
- Отсеки изолированы так же как контейнеры

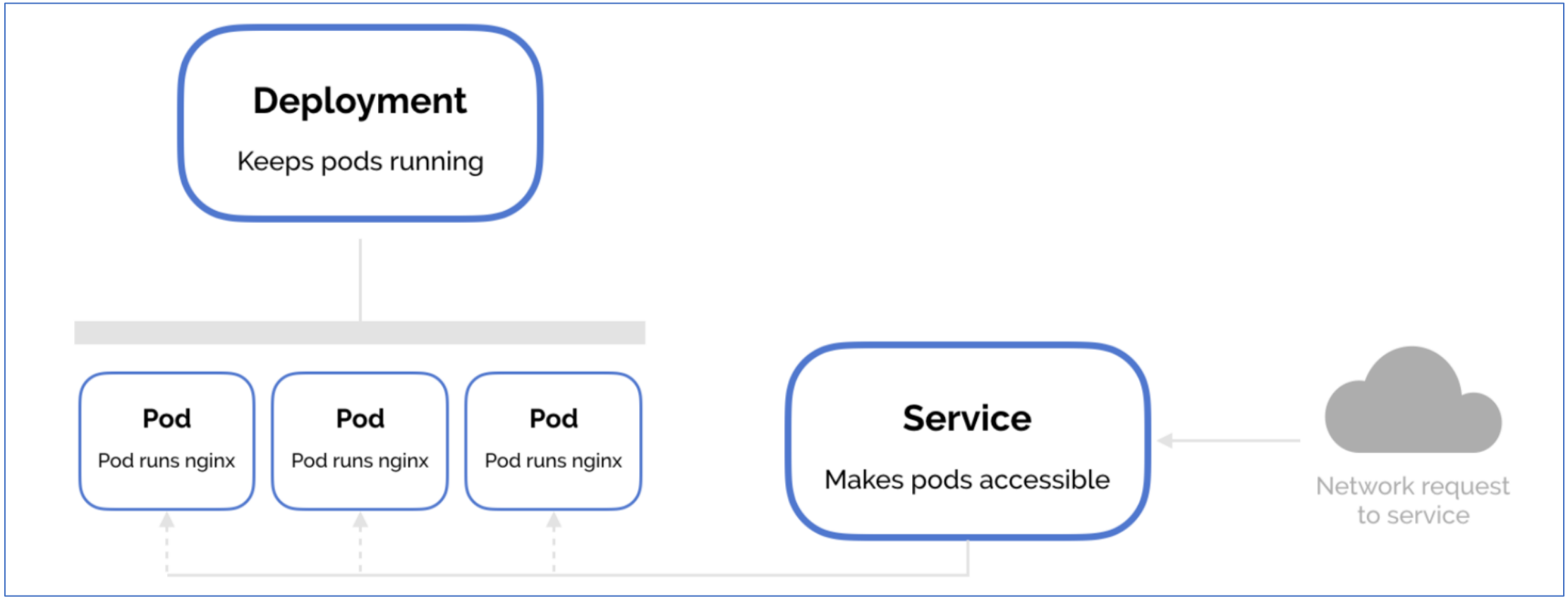
Модель Kubernetes: Развёртывание

- Управление отсеками



Модель Kubernetes: Служба

- Доступ к развёртыванию



Декларативное управление кластером

- REST
- Manifest
- YAML
- Desired State
- `kubectl <command> --dry-run=client -o yaml`
- `kubectl apply -f <path>`

Декларативное управление кластером

Развёртывание:

- Масштабирование
- Шаблон построения
- Образы

```yaml
# Версия программного интерфейса Kubernetes
apiVersion: apps/v1
# Тип объекта
kind: Deployment
# Метаданные нашего объекта, вложенный объект ObjectMeta
metadata:
  creationTimestamp: null
  # список меток самого объекта Deployment
  labels:
    app: time-service
  name: time-service
```

```yaml
# Описание собственно правил развертывания контейнера
# Вложенный объект DeploymentSpec
spec:
  # количество запущенных отсеков pods для масштабирования
  replicas: 1
  selector:
    matchLabels:
      app: time-service
  strategy: {}
  # описание шаблона для создания новых отсеков
  template:
    metadata:
      creationTimestamp: null
      # список меток для нового отсека
      labels:
        app: time-service
    # непосредственно описание контейнера в отсеке
    spec:
      containers:
      - image: time-service
        name: time-service:0.1.0
        resources: {}
status: {}
```

Декларативное управление кластером

Сервис:

- Метки
- Порты
- Тип связи

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: time-service
  name: time-service
spec:
  # список портов. Дополнительно можно указать протокол
  ports:
  - port: 8080
  # по этим меткам идет поиск отсеков, куда отправляются запросы
  selector:
    app: time-service
  # тип сервиса. В облаке можно использовать LoadBalancer
  type: NodePort
```

Взаимодействие сервисов

- DNS
- Переменные окружения

Взаимодействие сервисов: DNS

- Основная задачей системы управления кластером – обнаружение сервисов (service discovery)
- Сетевые адреса отсеков динамичны
- Один сервис – один внутренний DNS.
- Сервисы должны обладать уникальными именами.

Взаимодействие сервисов: Переменные окружения

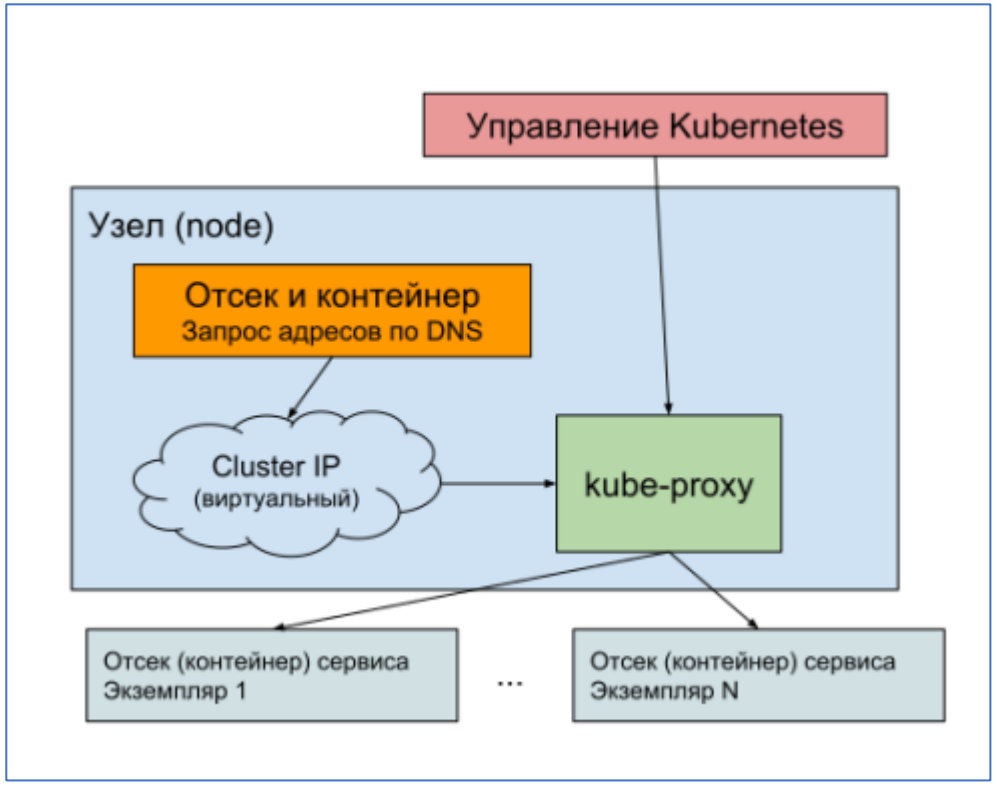
- Добавление переменных окружения только создающимся отсекам
- `TIME_SERVICE_PORT=tcp://10.97.105.149:8080`
  `TIME_SERVICE_SERVICE_PORT=8080`
  `TIME_SERVICE_SERVICE_HOST=10.97.105.149`
- Добавление во все отсеки немедленно после появления нового объекта Service без проверки готовности

Взаимодействие сервисов: Типы

- ClusterIP
- NodePort
- LoadBalancer

Взаимодействие сервисов: Виртуальные IP-адреса

- Kube-proxy
- Точки переадресации на отсеки
- Метки



Взаимодействие сервисов: Проверка готовности сервиса

- Ready
- HTTP 200-399

```yaml
spec:
 selector:
   matchLabels:
     app: weekend-service
 template:
   metadata:
     labels:
       app: weekend-service
   spec:
     containers:
       name: weekend-service
      # секция проверки готовности
       - readinessProbe:
            # проверка готовности с помощью запроса HTTP
            httpGet:
              path: /ready
              port: 5678
            periodSeconds: 2
            initialDelaySeconds: 5
            failureThreshold: 1
            successThreshold: 1
       - image: weekend-service:0.2.0
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
   app: weekend-service
 name: weekend-service
```

Метки

- Поиск сервисов использует метки (label) в объекте из секции metadata
- Метки доступны для любого объекта Kubernetes
- Предикат поиска – конъюнкция наличия всех меток
- `kubectl get pods --selector=<label>=<value> [,<label>=<value>[,…]]`
- Произвольный набор текстовых ключей и их значений
- Эффективная организация объектов по категориям

Метки: «Канареечное» развёртывание

- Сосуществование двух версии одной функциональности
- Дополнительное развёртывание
- Перебор отсеков

Метки: «Сине-зелёное» развёртывание

- Привязка сервиса к конкретной метке
- Ожидание готовности новой версии
- Аналогия с обновлением операционной системы мобильных телефонов

Непрерывное обновление

- Recreate
- RollingUpdate

```
…
# Вложенный объект DeploymentSpec
spec:
  # Количество запущенных отсеков pods для масштабирования
  replicas: 1
  # максимальное количество редакций (revisions)
  revisionHistoryLimit: 5
  # стратегия обновления
  strategy:
    type: RollingUpdate
...
```

```
# стратегия обновления
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxSurge: 100%
    maxUnavailable: 0
```

Непрерывное обновление: ReplicaSet

- `kubectl rollout history deploy time-service`
- `kubectl rollout undo deploy time-service`
- Одна редакция – один объект ReplicaSet
- Масштабирование

```yaml
# Тип объекта
…
kind: Deployment
# Метаданные нашего объекта, вложенный объект ObjectMeta
metadata:
   # список меток самого объекта Deployment
  labels:
    app: time-service
  # аннотации объекта
  annotations:
    kubernetes.io/change-cause: Updated version to 0.2.0
  name: time-service
```

```
$ kubectl get replicasets
NAME                      DESIRED   CURRENT   READY   AGE
time-service-5f59fbf479   0         0         0       42m
time-service-749f577cbc   0         0         0       41m
time-service-7ff577b7bd   1         1         1       41m
```

Непрерывное обновление: Автоматическое масштабирование

- `minikube addons enable metrics-server`
- `kubectl top node`
- `kubectl autoscale deployment/time-service --min=1 --max=3 --cpu-percent=80 --dry-run=client -o yaml`

```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  creationTimestamp: null
  name: time-service
spec:
  maxReplicas: 3
  minReplicas: 1
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: time-service
  targetCPUUtilizationPercentage: 80
```

Непрерывное обновление: Проверка жизнеспособности (liveness)

- По умолчанию сервис готов после запуска контейнера

```
spec:
  containers:
  - image: time-service:0.2.0
    name: time-service
    readinessProbe:
        …
    livenessProbe:
         httpGet:
           path: /time
           port: 8080
         periodSeconds: 2
         initialDelaySeconds: 5
         failureThreshold: 2
         successThreshold: 1
```

Источники

- Документация https://kubernetes.io
- Литература https://github.com/ivanporty/cloud-docker-k8s-book