

Виртуализация: Kubernetes ч.1. (Обзор, запуск приложений)



Артур
Сагутдинов



Артур Сагутдинов

Инженер департамента
голосовых цифровых
технологий

Banks Soft Systems



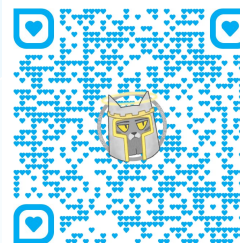
15+ лет в сфере ИТ



Разрабатываю и внедряю
линуксовую инфраструктуру



[Сисадминский блог](#)

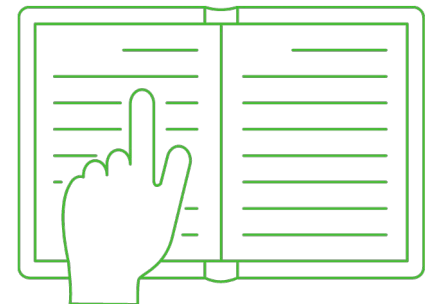


Предисловие

На этом занятии мы поговорим о:

- основах объектов Kubernetes;
- работе с minikube и k3s;
- запуске приложений в Kubernetes.

По итогу занятия вы получите представление о работе Kubernetes и научитесь использовать его для деплоя приложений.



План занятия

1. [Обзор Kubernetes](#)
2. [Структура объектов](#)
3. [Примитивы Kubernetes](#)
4. [Minikube и k3s](#)
5. [Работа с кластером](#)
6. [Документация](#)
7. [Итоги](#)
8. [Домашнее задание](#)



Обзор Kubernetes

Обзор

Kubernetes (k8s) — фреймворк для запуска и управления приложениями в среде контейнеров.

- Объединяет несколько серверов в кластер с единым управлением и единым хранилищем конфигураций.
- Для запуска контейнеров использует docker (или другие).
- Основные принципы — это декларативный подход к описанию и идемпотентность.



Структура объектов

Объекты

Все объекты в kubernetes могут быть представлены в виде yaml или json файлов.

Yaml — (*Yet Another Markup Language*) язык разметки, совместимый с json. Использует отступы и отсюда считается более читабельным.

```
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    env: production
    version: 1.0
spec:
  selector:
    app: nginx
  ports:
    - name: http
      port: 80
      targetPort: 80
```


Объекты

Основные параметры
верхнего уровня:

- **apiVersion** версия апи,
- **kind** тип объекта,
- **metadata** метаданные для идентификации объекта,
- **spec** параметры объекта,
- **status** хранит текущий статус объекта (не указывается при создании).

```
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    env: production
    version: 1.0
spec:
  selector:
    app: nginx
  ports:
    - name: http
      port: 80
      targetPort: 80
```

Объекты

Namespace — пространство имен. Используется для разделения объектов по неким критериям. Например, по окружению (prod, staging, ...) или по принадлежности к проекту (frontend, backend, ...)

Объекты условно можно разделить на:

- **namespaced** — привязанные к namespace,
- **cluster-level** — глобальные объекты.

Labels — лейблы. С их помощью реализован поиск объектов. И как следствие, их модификация. Если имя объекта динамическое и заранее невозможно его знать, то поиск таких объектов осуществляется с помощью лейблов.



Примитивы Kubernetes

Примитивы: Pod

Pod — минимальная единица рабочей нагрузки.

В большинстве случаев:

pod = контейнер

Однако, возможно создание нескольких контейнеров в рамках одного пода.

```
apiVersion: v1
kind: Pod
metadata:
  name: static-web
  labels:
    role: myrole
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
          protocol: TCP
```

Примитивы: ReplicaSet

ReplicaSet задает желаемую конфигурацию репликации и шаблон пода.

Kubernetes будет стараться поддерживать это состояние и в случае удаления или остановки пода.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: redis
  labels:
    tier: redis
spec:
  replicas: 1
  selector:
    matchLabels:
      tier: redis
  template:
    metadata:
      labels:
        tier: redis
    spec:
      containers:
        - name: redis
          image: redis
```

Примитивы: Jobs

Job — разовый запуск пода.
Обычно используется для
бутстрапа или миграций.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
      - name: pi
        image: yii
        command: yii migrate
        restartPolicy: Never
      backoffLimit: 4
```

Примитивы: CronJobs

CronJob — Запуск Job по расписанию.

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: hello
              image: busybox
              imagePullPolicy:
                IfNotPresent
              command:
                - /bin/sh
                - -c
                - date; echo Hello
```

Примитивы: Deployment

Deployment задает шаблон replicaset и занимается выкаткой приложений. При изменении объекта создается новый replicaset, и одновременно со старым они начинают апскейлиться и даунскейлиться соответственно.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```


Примитивы: DaemonSet

DaemonSet ведет себя аналогично деплою. Но вместо количества реплик daemonset поднимает по 1 поду на каждой ноде. Используется для приложений, которые необходимо держать на каждой ноде, например, node-exporter и.т.п.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluent-bit
spec:
  selector:
    matchLabels:
      name: fluent-bit
  template:
    metadata:
      labels:
        name: fluent-bit
    spec:
      containers:
        - name: fluent-bit
          image: fluent-bit:1.6
```

Примитивы: StatefulSet

StatefulSet — используется для деплоя кластерных приложений. Каждый под в нем имеет фиксированное имя, на которое можно сослаться в конфиге.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
          name: web
```

Примитивы: ConfigMap

ConfigMap — используется для хранения конфигурации. Можно хранить как переменные окружения, так и файлы конфига целиком.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: backend-conf
data:
  config.yaml: |
    db_host: mysql
    db_user: root
```

Примитивы: Secret

Secret — используется для хранения любой чувствительной информации (пароли, ключи и.т.п). В etcd хранится в зашифрованном виде. Может иметь один из нескольких типов:

- Opaque
- kubernetes.io/service-account-token
- kubernetes.io/dockercfg
- kubernetes.io/dockerconfigjson
- kubernetes.io/basic-auth
- kubernetes.io/ssh-auth
- kubernetes.io/tlsdata
- bootstrap.kubernetes.io/token

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-basic-auth
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: t0p-Secret
```

Примитивы: Service

Service — используется для направления трафика на под и балансировки в случае, если подов несколько.

По умолчанию сетевой доступ будет обеспечен только внутри кластера. Имя сервиса будет использоваться в качестве DNS.

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

Примитивы: Ingress

Ingress — объект, обрабатываемый ingress контроллером, реализует внешнюю балансировку (обычно по HTTP).

Популярные контроллеры:

- nginx-ingress
- traefik

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```



Minikube и k3s

Minikube

Minikube — компактная версия k8s, созданная для тестирования и разработки в среде k8s. А также в учебных целях.

Установка minikube зависит от операционной системы и системы виртуализации.

Полная инструкция доступна на официальном сайте:

<https://minikube.sigs.k8s.io/docs/start/>

Запуск: `minikube start`

k3s

k3s — аналог minikube, эмулирующий k8s. Запускается с помощью одного бинарного файла.

Установка возможна на linux с помощью bash скрипта:

```
curl -sfL https://get.k3s.io | sh -
```

Полная инструкция по установке доступна по ссылке:

<https://rancher.com/docs/k3s/latest/en/quick-start/>

Запуск: `service k3s start`



Работа с кластером

Работа с кластером

Kubectl — основная утилита для работы с k8s кластером.

Использует конфигурацию, расположенную в файле ~/.kube/config

Переопределить расположение этого файла можно с помощью переменной KUBECONFIG.

k3s имеет встроенный kubectl и может использоваться как:

```
k3s kubectl ...
```

Отдельно эту утилиту можно установить по инструкции:

<https://kubernetes.io/ru/docs/tasks/tools/install-kubectl/>

Работа с кластером

Общий синтаксис:

```
kubectl действие объект флаги
```

Например:

```
kubectl get pods -n kube-system
```

<p>Основные действия:</p> <ul style="list-style-type: none">• <code>get</code>• <code>create</code>• <code>edit</code>• <code>delete</code>• <code>describe</code>• <code>apply</code>	<p>Объекты:</p> <ul style="list-style-type: none">• <code>pods</code>• <code>deploy</code>• <code>svc</code>• <code>secrets</code>• <code>...</code>	<p>Флаги могут различаться для различных команд, из основных необходимо указывать namespace:</p> <ul style="list-style-type: none">• <code>-n</code>• <code>--all-namespaces</code>
---	--	--

Работа с кластером: деплой

Создаем файл: `nginx.yaml`

Деплоим с помощью:

```
kubectl apply -f nginx.yaml
```

Смотрим за статусом:

```
kubectl get po
```

Смотрим подробный статус
пода:

```
kubectl describe po имя_пода
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

Логи и ехес

Аналогом docker logs в k8s является команда **kubectl logs**

Пример:

```
kubectl logs --tail 100 имя_пода
```

Для того чтобы выполнить внутри пода какую-либо команду, используется:

```
kubectl exec -it имя_пода команда
```

Пример:

```
kubectl exec -it nginx-ans2l bash
```



Документация



Документация k8s

Документация на русском:

<https://kubernetes.io/ru/docs/home/>

Api reference:

<https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/>

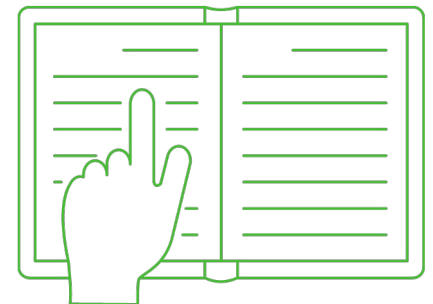


Итоги

Итоги

Сегодня мы рассмотрели примитивы kubernetes, а также утилиты для локальной разработки в среде k8s и теперь:

- умеем работать с кластером;
- можем деплоить и отлаживать приложения.





Домашнее задание

Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате**.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Артур Сагутдинов

