



OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование



Меня хорошо видно && слышно?

Ставьте +, если все хорошо
Напишите в чат, если есть проблемы

Проверить, идет ли запись!





Custom Resource Definitions. Operators

Игнатенко Филипп

Преподаватель



Игнатенко Филипп

Руководитель направления
безопасной разработки

#devops #devsecops #k8s

Правила вебинара



Активно участвуем



Задаем вопросы в чат или голосом



Off-topic обсуждаем в Slack #канал группы или #general



Вопросы вижу в чате, могу ответить не сразу

Маршрут вебинара

**Основные компоненты
кластера k8s**



K8s operators



**Разработка собственного
оператора**



LIVE

Цели и смысл вебинара | На занятии вы сможете

1

Освежить в памяти назначение основных компонентов кластера k8s

2

Изучить k8s operators и CRD, понять для чего они применяются

3

Понять, как создавать собственные операторы k8s



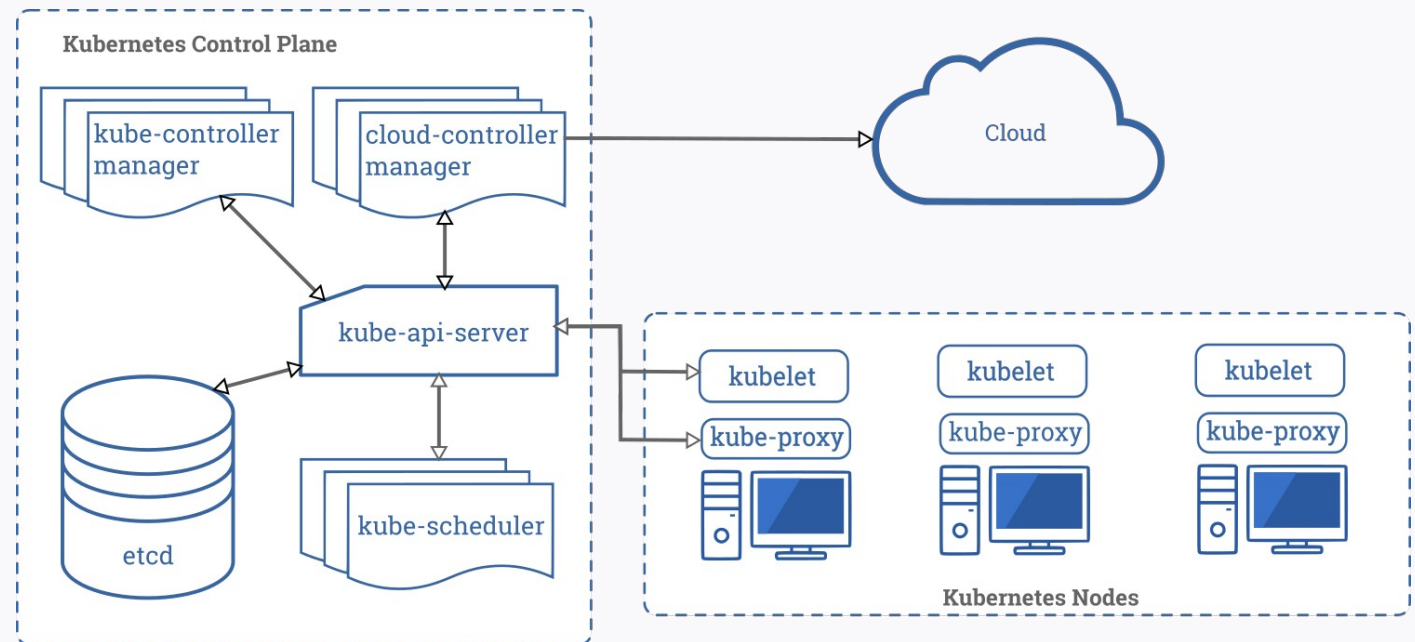
Основные компоненты кластера k8s

Основные компоненты кластера k8s

Из каких компонентов состоит кластер k8s?

Основные компоненты:

etcd
api-server
scheduler
controller-manager



Основные компоненты кластера k8s

etcd

etcd – key/value база данных для хранения конфигурации кластера

- Работает по алгоритму **raft** (он обеспечивает надежность за счет поддержки кворума)
- Единственная база данных для хранения конфигурации, которую поддерживает k8s
- Единственный stateful-компонент
- На каждую master-ноду устанавливается по ноде **etcd**

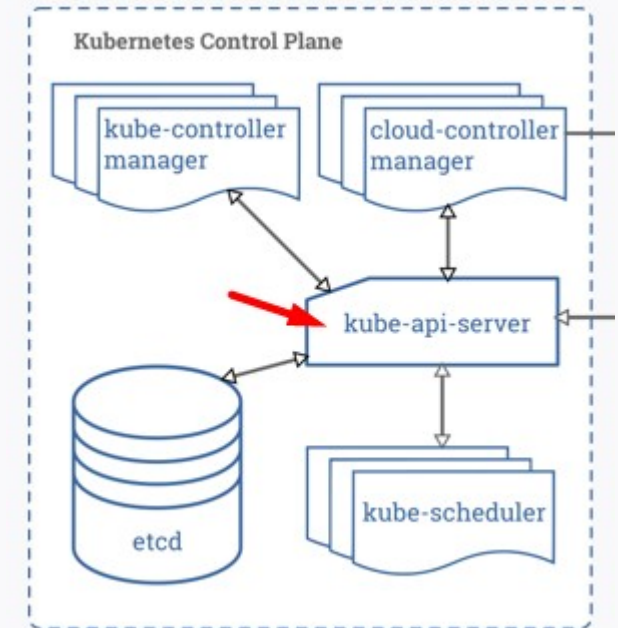


Основные компоненты кластера k8s

api-server

api-server – центральный, главный компонент k8s

- Stateless (в отличие от etcd)
- Взаимодействие через kubectl (но можно работать и просто curl'ом)
- Единственный компонент, который общается с etcd
- Работает по REST API
- Обеспечивает авторизацию и аутентификацию (разграничение прав доступа до содержимому кластера)



Основные компоненты кластера k8s

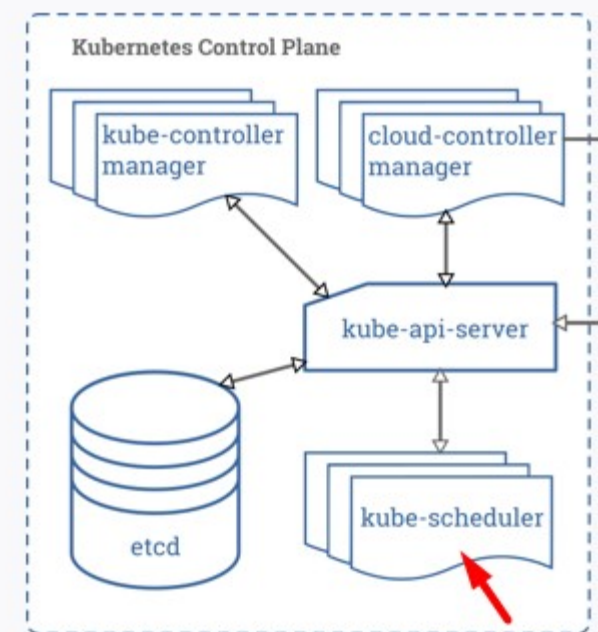
scheduler

scheduler назначает поды на ноды с учетом множества факторов

controller-manager генерирует манифесты подов, записывает данные в api-server, а **scheduler** назначает их на ноды, но учитывает важные параметры:

- QoS (quality of service)
- Affinity и Anti-affinity
- Requests и Limits
- Priority Class (preemption)

scheduler обычно запускают по одному на каждой мастер-ноде (**lease**)



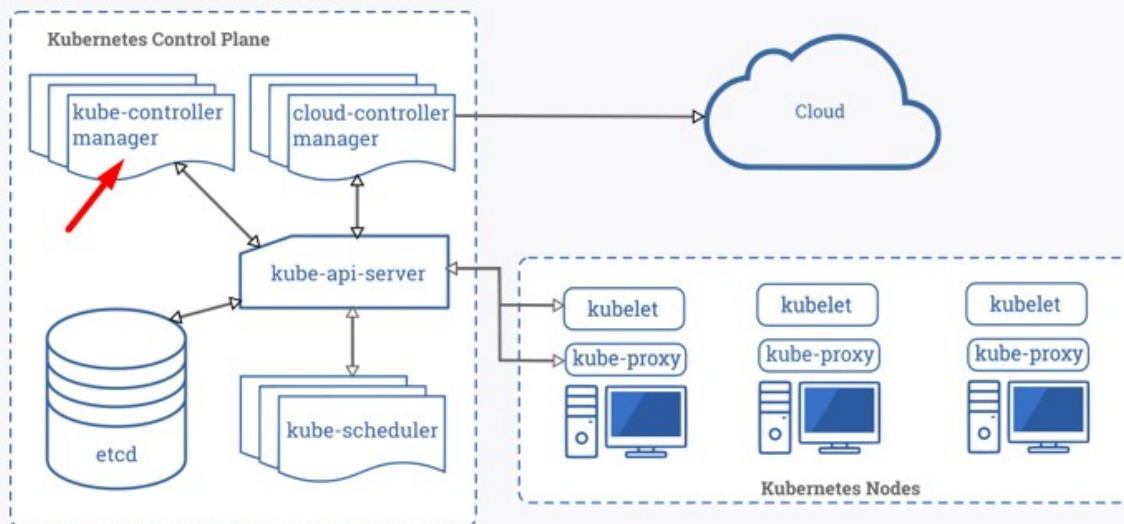
Основные компоненты кластера k8s

controller-manager

controller-manager – запускает процессы набора контроллеров

В состав controller-manager'а входят следующие контроллеры:

- node-controller
- replicaset-controller
- endpoints-controller
- account-controller
- token-controller
- И многие другие...



Основные компоненты кластера k8s

controller-manager (Подробнее о его наборе контроллеров)

- **node controller** - держит связь с нодами кластера, если нода не отвечает, переносит нагрузки на другие ноды

*kubelet общается с api server, **node controller** через controller manager общается тоже с api server, но (!) не с самим kubelet напрямую*

- **replicaset controller** - смотрит в api-server кластера, видит созданные replicaset'ы, реализует процедуру их создания
- **endpoints controller** - автоматизация создания эндпоинтов для сервисов (связь между подом и его сервисом)
- **account controller** - создание стандартных учетных записей
- **token controller** - создание токенов для доступа API

Основные компоненты кластера k8s

Итог - из каких компонентов состоит кластер k8s?

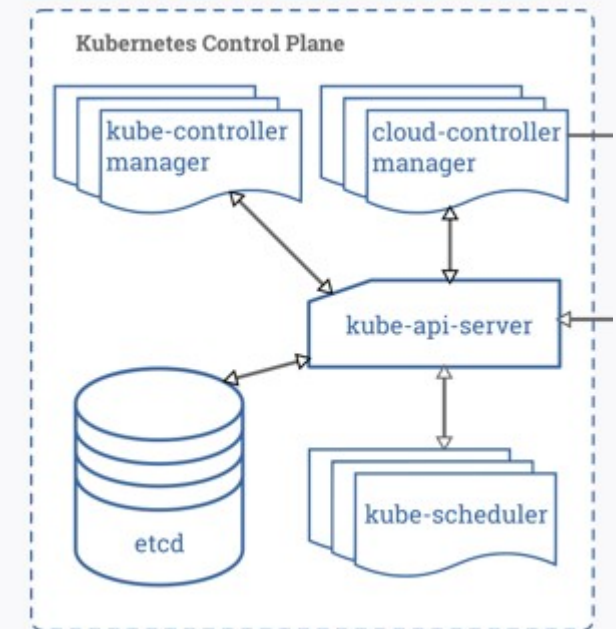
Основные компоненты k8s-кластера:

etcd – хранилище конфигурации

api-server – основной компонент API

controller-manager – запуск набора контроллеров и сборка мусора

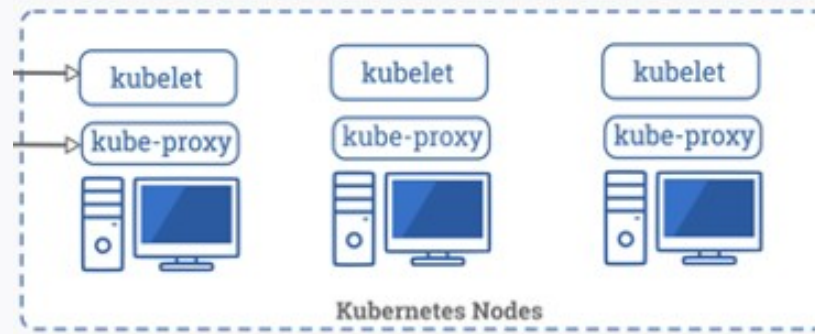
scheduler – назначение подов на ноды с учетом множества факторов



Дополнительные компоненты кластера k8s

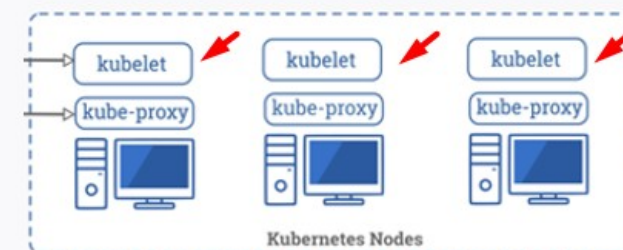
Помимо основных компонентов, установленных на мастер-нодах, для работы кластера необходимы дополнительные компоненты, которые управляются на всех нодах (мастеры и воркеры):

- kubelet
- kube-proxy



Дополнительные компоненты кластера k8s

kubelet



kubelet – агент, работающий на узле кластера

- Работает на каждой ноде (и мастера и воркеры)
- Не запускается в докере, работает как процесс на хосте (systemctl status kubelet)
- Отдает команды docker daemon через docker api (docker run, напр.)
- фактически реализует запуск подов на узле
- Обеспечивает проверки liveness probe, readiness probe, startup probe

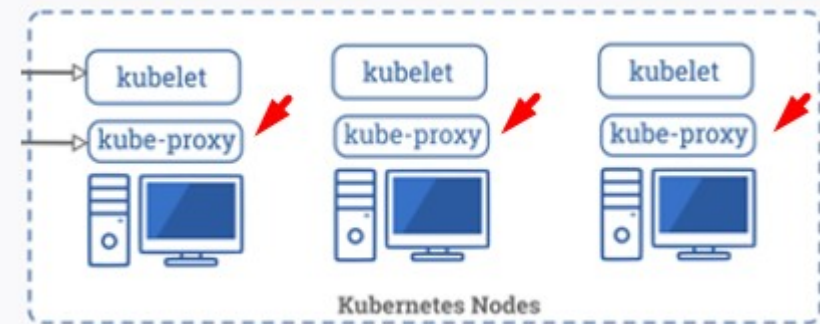
Дополнительные компоненты кластера k8s

kube-proxy

kube-proxy - сетевой прокси, работающий на каждом узле в кластере

- Взаимодействует с api-server
- Устанавливается на всех нодах
- Управляет сетевыми правилами на нодах
- Запускается в контейнере

*Некоторые cni-плагины забирают работу **kube-proxy** себе*



Дополнительные компоненты кластера k8s

Второстепенные компоненты, с которыми вы столкнетесь при работе с кластером k8s:

- **cri** (движок процесса контейнеризации)
- **cni** (сетевые плагины)
- **dns** (k8s-совместимые dns-серверы)
- **ccm** (controller-manager для облачных решений)



Дополнительные компоненты кластера k8s

Подведем итог

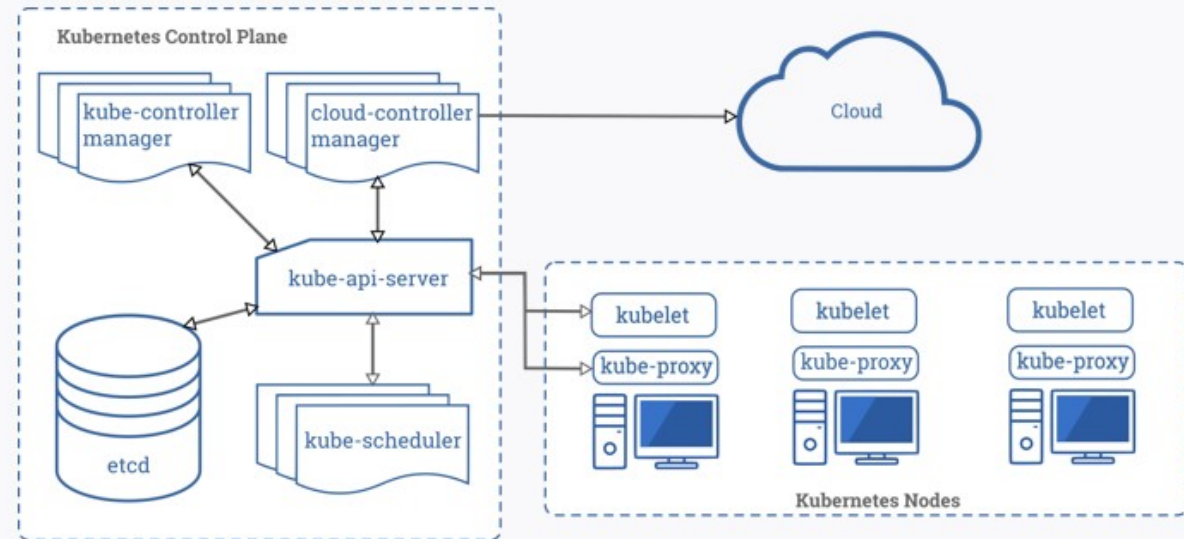
Для работоспособности кластера необходимы следующие компоненты:

На мастер-нодах:

- **etcd**
- **api-server**
- **controller manager**
- **scheduler**

На мастер- и воркер-нодах:

- **kubelet**
- **kube-proxy**



При разворачивании кластера вы можете также столкнуться и с другими неосновными компонентами: **cri, cni, dns, ccm**

The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this image is a semi-transparent network diagram consisting of numerous small blue dots connected by thin white lines, creating a web-like pattern across the center of the slide.

k8s operators

k8s operators

Операторы (operators) позволяют расширить имеющийся функционал в k8s

- В работе операторов скрыты две главные сущности в k8s: **объекты** (pod, service, replicaset и тд) и **контроллеры** (replication controller, account controller и тд)
- **Контроллеры** управляют состоянием кластера, основываясь на описании **объектов** (как делает replication controller, поддерживая количество реплик подов, описанных в replicaset'е)

k8s operators

В чем операторы могут быть полезны?

Операторы позволяют **автоматизировать** рутинные операции в k8s – развертывания, запуск рабочих нагрузок, обновление, удаление и проч. **не меняя** исходного кода самого k8s

k8s operators

Какие бывают операторы?

- Несколько примеров полезных операторов:
- **Развертывание** приложения или связки приложений по запросу
- Создание и восстановление **резевных** копий
- Обработка **обновлений** исходного кода (напр., схем баз данных)
- **Тестирование** кластера (напр., на отказоустойчивость)
- Выбор **лидера** для распределенных систем

k8s operators

Где мне могут встретиться операторы?

Операторы позволяют существенно упростить развертывание и управление многими ресурсами и приложениями.

- Многие вендоры k8s-приложений предоставляют свои собственные операторы:
- Jenkins operator
- Gitlab operator
- Prometheus operator
- Minecraft operator



Репозитории операторов: operatorhub.io, kubedex.com

k8s operators

Где я точно встречу с операторами?

Сервисная сетка (service mesh) на базе [Istio](#) использует собственный оператор как один из доступных способов установки

Однако, Использование оператора для новых установок Istio **не рекомендуется** в пользу методов установки **Istioctl** и **Helm**



Use of the operator for new Istio installations is discouraged in favor of the [Istioctl](#) and [Helm](#) installation methods. While the operator will continue to be supported, new feature requests will not be prioritized.

k8s operators

В чем разница между Helm и оператором?

Helm и операторы — это удобные инструменты для автоматизации задач, и хоть они и делают похожие вещи, но совсем **не взаимозаменяемы** (но **взаимодополняемы**)

Helm можно сравнить с пакетным менеджером, и хорошо подходит для быстрого развертывания больших и сложных приложений всего за несколько команд

Оператор тоже это может, но помимо установки, обновления и удаления приложений, но и включают в себя большое количество сложных конфигурируемых данных

k8s operators

Когда использовать helm, а когда операторы?

Вы просто устанавливаете приложение?

- Используйте **Helm**

Требуется ли вам специфичная, сложно описываемая конфигурация? (гейтвей, виртуальный сервис, распределение трафика и тд)

- Используйте **оператор**

Вы хотите развернуть множество приложений в кластере и более точно настраивать сложные кастомные конфигурации?

- Совмещайте **helm и операторы**

The background of the slide is a high-angle, blue-tinted aerial photograph of a city skyline, likely New York City, showing numerous skyscrapers and buildings. A semi-transparent blue band with a white geometric network pattern of dots and lines runs horizontally across the middle of the image, serving as a backdrop for the title text.

Разработка собственного оператора

Разработка собственного оператора

Как написать собственный оператор?

Для написания собственного оператора придется программировать желаемое поведение **объектов** k8s и логику **контроллера** (операторы оперируют контроллерами, управляющими состоянием кластера, основываясь на описании объектов)

Оператор = Контроллер + Объекты

Разработка собственного оператора

Как написать собственный оператор?

В написании собственного оператора вам могут помочь специальные фреймворки, библиотеки и инструменты, позволяющие без труда реализовать свой оператор на том или ином языке программирования:

- [Charmed Operator Framework](#)
- [kubebuilder](#)
- [KubeOps \(.NET operator SDK\)](#)
- [KUDO \(Kubernetes Universal Declarative Operator\)](#)
- [Metacontroller](#) along with WebHooks that you implement yourself
- [Operator Framework](#)
- [shell-operator](#)

3rd-party клиенты: clojure, dotnet, elixir, go, java, lisp, nodejs, perl, php, python, ruby, rust, scala, swift (поддерживается авторами, а не сообществом k8s)

И отдельно стоит отметить **Operator SDK**

Разработка собственного оператора

Как написать собственный оператор?

Разберем **этапы** написания собственного оператора с помощью фреймворка kubebuilder и языка go:

- 1) Создание проекта контроллера
- 2) Реализация логики работы контроллера
- 3) Создание манифестов для кастомных ресурсов (**CRD**)
- 4) Установка кастомных ресурсов (**CRD**)
- 5) Запуск оператора (локально или в кластере)

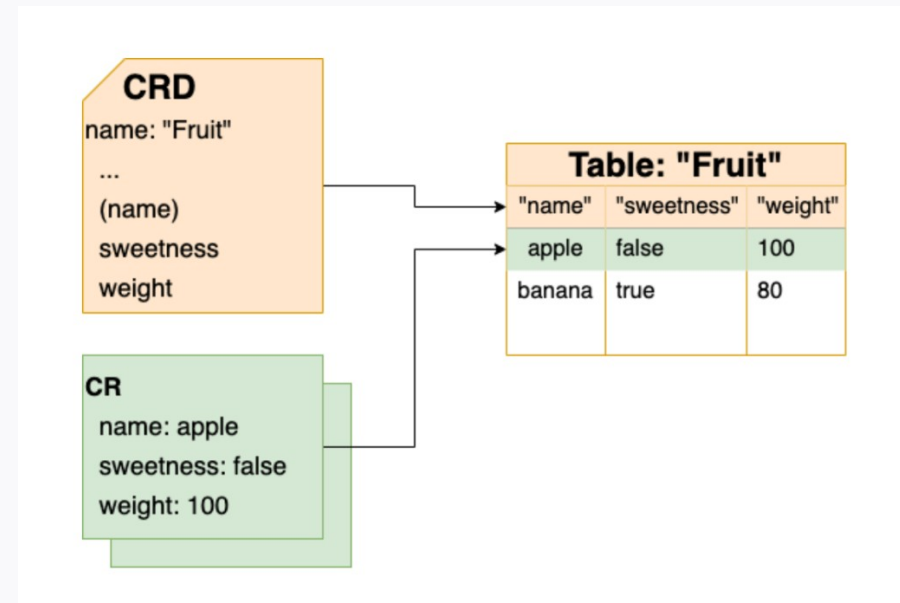
Разработка собственного оператора

Что такое кастомные ресурсы (CRD)

Операторы используют особый (кастомный) тип объектов в k8s для управления другими объектами (основными объектами, такими как pod, ns, rolebinding, quotas и тд)

Эти кастомные объекты называются CRD и используются для управления разными объектами k8s

Простыми словами, CRD — это **особенная таблица** в общей базе данных k8s-кластера, которая содержит записи о различных ресурсах, которыми оперирует **оператор** в своей работе



Разработка собственного оператора

Что такое кастомные ресурсы (CRD)

HelloApp на картинке ниже — это определение кастомного ресурса (CRD), которое обрабатывается нашим оператором

Логика работы в составе оператора k8s:

- 1) Генерируем манифест CRD и применяем его
- 2) Контроллер нашего оператора создаст деплоймент для kind HelloApp (и будет отслеживать его наличие)
- 3) Образ контейнера указан в spec.image
- 4) Число подов равно тому, что мы задали в spec.size (size: 1)

```
demo-operator > config > samples > ! apps_v1_hel
1  apiVersion: apps.anupam.com/v1
2  kind: HelloApp
3  metadata:
4    | name: helloapp-sample
5  spec:
6    | image: "anupamgogoi/mock-app"
7    | size: 1
8
```




LIVE



Задание. Проверка достижения целей занятия

- 1** Повторили назначение основных компонентов кластера k8s
- 2** Изучили что такое операторы и CRD, и где они применяются
- 3** Получили понимание, как написать собственный оператор k8s



индивидуально
или в группе



тайминг



где выполняем



ГОТОВЫ

Цели и смысл вебинара | На занятии вы СМОГЛИ

1

Повторили основные компоненты k8s

2

Поняли, для чего применяются контроллеры в k8s

3

Узнали о назначении k8s операторов

4

Узнали, как взаимодействуют CRD и операторы k8s

5

Получили понимание, как написать собственный оператор k8s

Следующий вебинар

Тема: «Мониторинг компонентов кластера и приложений, работающих в нем»



01.02.22 (Вторник, 01 февраля)




Ссылка на вебинар будет в ЛК за 15 минут




Материалы к занятию в ЛК — можно изучать



Обязательный материал обозначен красной лентой

The background of the image is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer. A network of white lines and dots, resembling a digital or social network, is visible across the blue overlay. The text is centered in this area.

Заполните, пожалуйста,
опрос о занятии по ссылке в чате



Спасибо за внимание!
Приходите на следующие вебинары

Игнатенко Филипп