Всем привет и добро пожаловать на лекцию «Операторы Kubernetes», которые рассмотрим. Анатомия операторов, их использование в production-е и напишем свой оператор сами.

Если рассматривать назначения операторов Kubernetes, назначение custom-ных controller-ов Kubernetes, то по сути, это расширения Kubernetes-овского API для того, чтобы отвечать запросам именно вашего домена, потому что в общем случае вы, в принципе, можете пользоваться совершенно стандартными Kubernetes-овскими ресурсами и вообще никогда в жизни не видеть этих controller-ов. Но операторы и их custom-ные controller-ы вам помогают, во-первых, категоризировать ресурсы специфические для вашей системы, если у вас есть какие-то приложения специфические или если вам нужно выделить какую-то отдельную группу приложений со своими атрибутами и свойствами – вы можете использовать custom-ные операторы, custom-ные ресурсы соответственно.

Для управления этими custom-ными ресурсами для того, чтобы структурировать…, для того, чтобы вынести общий набор атрибутов для ваших групп тех самых и упросить себе управление, в том числе и тем, что вы уже заранее знаете весь список атрибутов для вашего приложения.

И третье – это расширение Kubernetes-овского API для того, чтобы если у вас есть кластер, у которого есть свои, допустим, клиенты – они могли бы пользоваться вашими ресурсами, и ваш кластер будет отличен от других стандартных кластеров Kubernetes вашими уникальными ресурсами.

Давайте разберемся с анатомией оператора и прежде, чем мы перейдем к самим операторам, поймем, что такое controller-ы. Под капотом у Kubernetes-а (НРЗ 01:45) controller-ов, то есть Kubernetes-овских backend-ов, которые обрабатывают запросы…, декларативные запросы на создание каких-то ресурсов. Мы помним,

что Kubernetes декларативный – ты говоришь ему, как это должно выглядеть, а что ему при этом делать, чтобы это выглядело вот так, это не наша забота. И вот как раз эта не наша забота обеспечивается вот этой внутренней логикой – (НРЗ 02:07) controller-ов внутри Kubernetes-а, который анализирует все наши Daemon Set-ы, Stateful Set-ы, Deployment-ы и так далее, и spawn-ит объекты.

Операторы, controller-ы все работают по принципу observe, analyze, act – «Наблюдай, анализируй, действуй». То есть controller смотрит на состояние системы, анализирует это состояние, сравнивает его в тем, что сейчас есть, поскольку он про это знает. И если оно расходится – предпринимает какие-то усилия для того, чтобы состояние системы было конвергентным, сходилось с тем, что написано в декларации. И это происходит непрерывно, бесконечно – controller-ы основные Kubernetes-овские, которые уже встроены, находятся в Kubernetes-овских Master node-ах и следят за декларацией, берут системные ресурсы и превращают их в Kubernetes-овские объекты, по сути.

И мы имеем возможность написать свои controller-ы custom-ные, которые тоже будут что-то делать со стандартными ресурсами Kubernetes-а, имплементируя нужную нам логику. Либо мы можем сделать custom-ные ресурсы. То есть, когда мы определяем, например, сервис pod, deployment, daemon set – мы определяем ресурс Kubernetes-а.

И мы можем сделать свой custom-ный ресурс, как я показывал в предыдущей лекции, может сделать какой-то свой ресурс. Вот тут, например, есть ресурс flunders и tunnels. Положить его в своё api и опять же, apiservices – это тоже custom-ный ресурс. И мы можем из своих controller-ов управлять этими custom-ными ресурсами. И вот это соединение controller-а и custom-ного ресурса, которым мы управляем, соединение вместе называется как раз Kubernetes-овским оператором. Controller, который работает со стандартными ресурсами – это

controller. Controller, который работает с нашими ресурсами, которые мы сами определили Kubernetes-у, называется operator. За исключением того, что, например, operator можно ещё определить путем передачи ему данных в конфигурацию без custom-ных ресурсов, но это, тем не менее, тоже будет operator-ом.

Существует такое негласное деление между controller-ами и operator-ами, что мол controller – они, значит, только управляют стандартными ресурсами и пользуются ConfigMap-ами, то есть можно им что-то в конфигурацию что-то написать. Operator-ы пользуются custom-ными ресурсами. Это довольно, такое вязкое деление в том смысле, что оно не очень…, оно очень размытое, потому что operator, опять же, я могу сделать operator, который пользуется конфигурационными файлами. И он будет создавать мне controller и использовать в конфигурационной фауне. Ничего такого в этом нет, так делают. Особенно, если у вас нет прав на кластер, чтобы создавать custom-ные ресурсы, это потому, что это самый верхний уровень Kubernetes-овского API, и далеко не у всех разработчиков и далеко не у всех даже DevOps-ов будет доступ к этому самому верхнему уровню.

Но глобально, если мы рассматриваем operator-ы в целом, у нас operator всегда работает с Custom Resource Definitions, custom-ными ресурсами, которые мы создаем и загружаем в Kubernetes, и оператор работает исключительно с ними, это его зона ответственности. Каждый ресурс должен…, в смысле должен? Если у вас есть ресурс, у которого нет controller-а – ничего страшного в этом не произойдет, но он…, толку не будет от него. То есть он будет болтаться, просто занимать место, вы будете его видеть, но не будете видеть какой controller им управляет, потому что его не будет в системе, поэтому он должен иметь какой-то controller, который управляет его состоянием. И всё это вместе называется operator-ом.

Есть следующий уровень безумия здесь, то есть custom-ные ресурсы – это декларативное объявление Kubernetes-у, то, что вы хотите работать с custom-ными ресурсами. И ещё для custom-ного ресурса вы можете, например, определить своё API, через которое можно управлять им с помощью Kubectl-а или чего-то ещё. И это API будет добавлено в соответствующую группу вашего Kubernetes-а и будет доступно для внешних клиентов. И называется такая схема API Aggregation, когда вы добавляете вызовы API в соответственно сам Kubernetes. Техника, используемая довольно редко, используемая во всяких Enterprise кластерах, у которых есть свои какие-то фишки, которые они продают в дальнейшем. Именно, если вы поддержите именно инфраструктуру компании, вряд ли вы этим будете пользоваться. И API Aggregation, вам нужен API сервер, который также крутится внутри Kubernetes-а, все controller-ы крутятся внутри Kubernetes-а. Это просто deployment-ы, это просто pod-ы внутри, существующие, как правило, в одном экземпляре.

Но и по поводу использования в реальности. На самом деле, operator-ы используются довольно много, где, вы можете, например, поставить тот же самый Prometheus через operator-а, для того чтобы удобно объявлять instance-ы Prometheus-а, загружать его себе внутрь систему, просто использовать…, определять ресурс Prometheus-а. Он поднимается удобно, потому что всё собрано в одном месте, нет этих 500 файлов deployment-а Prometheus-а.

Используется STO, который мы уже видели. Там используется controller, который по label-ам и injected sidecar-ы. Это тоже ответственность controller-ов. Во всяких tool-ах compliance-а, во всяких сборах метрик, в каких-нибудь deployment-ах каких-нибудь баз данных. Например, мы в той лекции предыдущей загружали Марию. Используется в tool-инге, всяком сетевом tool-инге. Для серверов авторизации, но и в принципе, вы можете так-то, на самом деле, что угодно обернуть в operator, если в этом есть смысл. Если у вас довольно сложное

приложение, а вы хотите его закрыть от мира его deployment и оставить только несколько ручек, которые пользователи могут контролировать сообразной их ситуации. И со всем этим давайте уже посмотрим, как эти операторы пишутся.