Перед тем как мы уже действительно пощупаем operator-ы, давайте разберемся с системой Kubernetes-а, с системой авторизации и доступа к ресурсам. Она довольно сложная, она…, вообще эта тема, на мой взгляд, для отдельного курса лекций – авторизация Kubernetes-а. Я пройдусь по основным моментам, которые нам позволят читать файлы авторизации operator-ов попроще. И погнали.

Итак, Kubernetes использует policie, которое называется: «разрешено, что указано, всё остальное – запрещено». То есть настолько, что даже при указании свойств в авторизации вы не можете сказать «запрещено», потому что, если вы укажете, что оно разрешено, оно не будет разрешено.

Соответственно, у каждого запроса к Kubernetes-овскому API существует 1 из…, или несколько параметров: пользователь, группа, экстра мы пропустим – это не важно. API, который определяет, есть ли этот доступ к ресурсу API или так называемый запрос с нересурсному endpoint-у, соответственно, мы к нему придем попозже. Путь для нересурных endpoint-ов как раз, глагол API и глагол HTTP – это разные вещи. Ресурс, подресурс Namespace и группа API.

Всё сложно, давайте будем разбираться с тем, что нам нужно. Разбираться мы будем с глаголами API, с HTTP глаголами, ресурсами, подресурсами и Namespace-ами, потому что это то, с чем будут работать наши controller-ы. Пользователи, группы и так далее – это для запросов от пользователей для запросов снаружи в кластер. Ваш пользователь будет иметь эти атрибуты, но нас интересует больше глаголы.

Для доступа к различным ресурсам Kubernetes-а, а мы помним, что pods, statefull sets, daemon sets, config maps, что угодно – это ресурсы в Kubernetes-е. Мы можем определить либо глаголы самого Kubernetes-а: get, list, create, update, patch, watch, delete и deletecollection. Либо мы можем определить глаголы HTTP. И здесь

существует mapping, то есть POST, например, у нас map-ится в глагол create. GET, HEAD в get, list. PUT в update, PATCH в patch и DELETE в delete и в deletecollection. И соответственно, также мы указываем ресурсы, ресурсы – это наши pod-ы и прочее.

У ресурсов могут существовать вложенные ресурсы и в таком случае мы можем определить и для них права доступа. Через ссылающихся наших pod-ов будет существовать ресурс health, то pods/health – мы к нему можем также определить набор глаголов. То есть ресурсы определить через слеши и к нему отдельно определить набор глаголов.

Kubernetes предоставляет несколько типов авторизации, авторизация Node-ы, авторизация attribute-based access control, когда вы определяете атрибуты, соответственно, ваших ресурсов… Вернее, ваших пользователей, и они получают доступ. Но нас он не интересует, нас интересует RBAC – Role based access control, то есть, когда вы определяете роль и определяете какой-то список доступов, глаголов и прочего, и заходите через роль.

Для RBAC-а вы должны определить либо роль, либо кластерную роль. Разница между ними в том, что роль всегда ставит permission-ы для какого-то конкретного namespace-а. Если namespace не указан, соответственно, namespace – default. Кластерная роль ставит доступы для всего кластера. Соответственно, в роли вы указываете API группы, потому что Kubernetes так и не смог разобраться, какие ресурсы принадлежат каким группам, поэтому вы их указываете эксплицитно. API группа если не указана, подразумеваются default-ныа группа, в которой находятся у нас pod-ы. И вот здесь вы определяете глаголы, что может делать с pod-ами эта роль. Больше эта роль с pod-ами не может делать ничего. И только, соответственно, в определенном namespace-е. Кластерная роль работает для

всего кластера в целом, точно также cast и API группу и указываете там secret-ы и, например, глаголы.

Можно точно также указывать всё вместе в роли, но опять же secret-ы имеют смысл только на…, например, кластерные роли, потому что они принадлежат кластерам. После этого вы можете забиндить, связать роль с каким-либо ресурсом – это будет определяться субъектом этой роли. То есть в нашем случае пользователь по имени jane присваивается в этой группе вот такая роль: read-pods.

И, в принципе, то же самое для кластерных ролей. Почему это раздельно? – Честно говоря, не знаю, но, тем не менее, это разделено, кластерные роли и роли присваиваются отдельно. Соответственно, вы можете эти роли совмещать, вы можете, опять же, одному пользователю присвоить много ролей, вы можете агрегировать кластерные роли в роли. И соответственно, здесь есть ещё пример, как давать доступ к конкретным ресурсам, например у pod-ов есть ресурс log-и, и вы точно также можете отдельно давать доступ именно к логам этих pod-ов. На этом, в принципе, у меня всё здесь, довольно исчерпывающая документация. То есть никаких проблем не должно возникнуть. Давайте просто разберем эти кластерные роли и роли на примере нашего operator-а из предыдущей лекции.

Вот и давайте разберемся соответственно, с ролью здесь. Создается роль точно также, как и всё остальное в Kubernetes-е через описание, через control. У неё есть имя, есть первая группа – она default-ная, в которой pod-ы, сервисы, endpoint-ы. То есть operator реально требует довольно-таки дохренища доступов. И вот глаголы тоже практически все кроме deletecollection.

Следующая группа API-ев apps, мы перечисляем deployment-ы, daemonset-ы, replicaset-ы. То есть со всем этим наш operator работает, и глаголы,

соответственно, тоже туда же применяются. Потом здесь есть мониторинг и прочее. То есть мы видим, что довольно-таки много api-ашных групп применяется в роль. И потом у нас ещё есть кластерная роль, в которой делаются node-ы, persistentvolumes, namespaces, в них можно смотреть, перечислять и удалять, и так далее. То есть мы довольно большой набор ролей даём operator-у и это оправдано, потому что как правило, operator-ы работают с каким-то довольно большим набором API-шных функций. Как минимум, они что-то там просматривают, что-то получают, на что-то смотрят и что-то создают. То есть как минимум уже у вас будут роли под это.

И роли биндятся, биндятся они для operator-ов на сервисные account-ы – это такая специальная штука, которую вы можете указать в deployment и pod с этим ServiceAccount-ом будет иметь все эти уровни доступа. То есть данный случай, это наши, получается, пользователи. То есть роль биндится на ServiceAccount mariadb-operator, и кластерная роль точно также биндится на ServiceAccount mariadb-operator. И сервисный account просто создается.

И сам биндинг сервисного account-а у нас происходит в deployment-е operator-а. То есть в deployment-е описывается operator, его реплики, его selector-ы. А у спецификации deployment-а перед контейнерами мы указываем ServiceAccountName. Соответственно, все контейнеры с deploy-еные под этим ServiceAccountName будут иметь этот ServiceAccount. Без него наш operator не сможет получить доступ к Kubernetes-овскому API и будет, естественно, fail-ить. На самом деле, даже не sadeploy-ится, потому что Kubernetes скажек: «Не знаю такого ServiceAccountName». А если вы его создадите и не привяжите роли – будет вот так.

На этом всё. Давайте уже переходить к созданию чего-то.