Мы закончили писать наш котроллер, у нас существует полный, значит, Job для контроллеров. То есть у нас существует полный Reconciliation cycle, у нас есть типы объявленные. У нас всё, в принципе, готово. Вопрос – чего нет? Нет у нас CRD.

Как я уже сказал, CRD генерируется из вот этого вот файлика. На момент, когда мы генерируем наш скелет проекта, у нас не существует никаких, значит, CRD, не существует никаких полей, поэтому CRD генерировать несколько бессмысленно. Естественно, это нужно делать после того, как мы заполнили все эти поля. Вот в данном случае мы уже готовы генерировать наше CRD. Для этого у нас в makefile существует команда make manifests, с которой мы сейчас и запустим. Значит, команда make manifests сгенерирует наши CRD из нашего API и поместит их в папочку CRD bases.

Соответственно, здесь у нас уже есть openAPIV3Schema, у нас здесь есть description и есть enum, есть всякие различные параметры. То есть это все сгенерировано за нас. Нам нужно только вот этот файлик заполнить и, соответственно, запустить make manifests, и наш CRD файлик будет уже готов для того, чтобы его установить в наш Kubernetes-овский кластер и с ним уже спокойненько работать. Как мы тут видим, здесь как раз есть история, что, соответственно, большинство комментариев на английском, и получается такой микс. Поэтому, когда вы пишите свои контроллеры, лучше, естественно, писать их на английском, даже если аудитория, которая будет использовать, она будет русскоязычной. Либо у вас там надо уже тогда переводить всё, то есть прям редактировать этот файлик и переводить всё на русский.

Время наше CRD заинсталлировать для того, чтобы контроллеру было с чем работать, и инсталляция CRD у нас тоже происходит с помощью makefiles, с помощью команды make install. Несколько контринтуитивно, то есть обычно make

install – это сделать установку именно пакета целиком, в данном случае это только у нас наше CRD. Make install использует тот факт, что у нас существует конфигурация в kubectl и спокойненько можно подключиться к кластеру и её использовать. Соответственно, у меня развернут в данном случае minikube. Если у вас, например, вы подключены к какому-то Amazon-овскому кластеру с правами доступа админ, то это, в принципе, будет для вас то же самое, то есть этот (HP3 03:11) абсолютно прозрачен. Не важно, что у вас – локальный deployment или кластеровский deployment.

Мы просто делаем make install, он нам будет скачивать kustomize, который как раз отвечает за инсталляцию всех этих кастомных CRD, и после этого установит наши CRD-шки с помощью kubectl apply. Теперь мы, соответственно, можем посмотреть, можем с нашего кластера получить к наши CronJob-ы, которых в данном случае нету. Ну вот Kubernetes с ней ругнулся, то есть такой вот тип ресурса у нас есть. В данном случае, если бы у нас было что-то другое, то, естественно, Kubernetes бы сказал, что такого у нас нету.

Для тестирования нашего контроллера мы можем воспользоваться командой make run. Что она делает? За счет как раз этого разделения менеджера и контроллера мы можем сделать это прозрачным образом. Задеплоить контроллер от лица как бы пользователя, то есть подсоединиться к Kubernetes-у, слушать нужные нам Endpoint-ы без Deploy-я самого контроллера. Это удобно для, соответственно, отладки, для ловли каких-то багов. То есть вам не нужно ничего загружать, вы просто делаете make run, запускаете и смотрите на логи. Для того, чтобы наш контроллер в данном случае заработал, тут видно, что у нас всё тут, значит, подключилось, есть worker-ы. И мы готовы реконсилировать как раз наши CronJob-ы. Для того, чтобы всё получилось, нам нужно загрузить какую-то CronJob-у. У нас уже есть шаблон CronJob-ы для тестирования, давайте его заполним в соответствии с нашей вот этой схемой в CRD.

Соответственно, наша CronJob-а. У нас здесь определено API, определен, значит, v1, определен вид, что это CronJob-а, все её параметры. Как у нас заведено в схеме, всё у нас здесь соответствует этой схеме и, соответственно, мы готовы к тому, чтобы эту джобу запустить.

Давайте сделаем kubectl apply -f, значит, нашего конфига с CronJob-ами. У нас создалась CronJob-а, и в этот момент можно увидеть, что отработал контроллер и создал джобы. Сам себе (НРЗ 06:00) на следующий (НРЗ 06:04). И вот, опа, создалась джоба. То есть тут важно, что, конечно, очень важно то, что в контроллере нужно писать максимально подробно логи, чтобы было понятно, что у вас там происходит, и очень важно в случае любых ошибок правильно его, соответствнно, (НРЗ 06:25).

Давайте посмотрим на вывод Kubernetes-овской команды. Во-первых, посмотрим на pod-ы. Вот у нас pod создался для CronJob-а. Если мы сейчас посмотрим CronJob-ы – их не будет, поскольку нам нужно смотреть на специальные наши CronJob-ы, определенные в наши CustomResourceDefinition, то есть на вот эти. Вот наша CronJob-а, мы у неё можем взять Description как раз. Опять же, здесь это подресурс вот этого типа. Сделаем вот так и вот, мы всё видим. Мы видим эти вот Managed Fields, мы видим наши, соответственно, операции, видим наш Last Schedule Time, который мы обновляем. В общем, как-то так. Наш контроллер работает, наши джобы обновляются, у нас в целом всё хорошо.

Итак, давайте задеплоим теперь наш оператор, наш контроллер и все сопутствующие ему причиндалы в сам Kubernetes в виде deployment-а, чтобы уж там всё внутри крутилось и никак с нашей машиной, естественно, не работало. В этом нам помогут команды Make file-а docker-build и docker-push, которые под капотом просто содержат docker-build и docker-push с тэгом изображения image.

Оно определит, куда мы хотим засунуть наш контроллер и куда мы хотим его задеплойить.

Я создал изображение на dockerhub. Dockerhub – это бесплатная штука. В ней вы можете создать один приватный репозиторий и сколько хотите публичных репозиториев. И соответственно, я просто делаю make docker-build и docker-push и указываю тэг IMG=nortix/cronjob-controller, потому что это вот такой путь к моему контроллеру, и в (НРЗ 08:42) делаю версионирование один, ноль, ноль. Всё, собираем. И соответственно, задеплоиваем наш контроллер внутрь docker-а, внутрь dockerhub-а. После сборки нашего изображения и его deploy-я, соответственно, в репозитории. Вот оно, наше изображение.

Мы можем теперь заняться deploy-ем всего остального, то есть всех этих файликов, ролей, CRD, базы, всего, кроме samples, в общем, в наш Kubernetes-совский кластер. Для этого используется как раз тот самый kustomize, который соберет из этого всего один deployment файл и, соответственно, команда kubectl apply, которая задеплоит внутрь нашего кластера. На этот счет у нас есть Short Cut в makefile-ах, который называется deploy. Он, собственно говоря, собирает файлы и их просто (НРЗ 09:44).

Давайте запустим его без вот этой команды и посмотрим, что у нас в итоге то собирается. Итак, указом ему IMG, потому что IMG он берет для установки контроллера и смотрим, соответственно, что получается. Вот у нас Deployment, kubectl operator system, у нас здесь подставился наш IMG нашего оператора, здесь есть ещё сервисы, которые принадлежат оператору, всякие конфигурации, роли, (НРЗ 10:19), наши вот эти кастомные как раз CRD и прочее. То есть это один такой большой файлик deployment-а, который, в принципе, можете для внешних пользователей вашего кластера положить в deployment яму и собственно говоря,

так и распространять. Это, в принципе, всё, что нужно для deployment-а оператора.

Вернем команду на место и просто запустим наш Staff внутри Kubernetes кластера. После deploy-я у нас создался NewSpace kubebuilder operator system, и в нём лежит, соответственно, наш deployment нашего оператора. Тут уже всё есть, то есть оператор с proxy для обращения к ресурсам кластера. Мы можем посмотреть, что там внутри. Внутри нам уже должны быть знакомые логи. Давайте задеплоим нашу работу, нашу джобу, kubectl apply -f, значит config/samples и наша джоба. Наша джоба создана. Мы напоминаем, можем посмотреть все эти CRD по имени прям, получить их списочек. Вот у нас cronjob-sample, можем, опять же, на него посмотреть, но тут…, вернее его дискрайбнуть. Тут ничего, в принципе, у нас пока интересного не будет.

Вот наши работы, busybox. Мы это уже, в принципе, видели. Давайте посмотрим ещё раз на логи. Мы видим, что наш менеджер нашёл какие-то работы, но никаких пока scheduler-ов нету. Мы ждем до 31-ой минуты, а потом будем запускать работу.

Вот я чуть-чуть подождал, соответственно, у нас прошли scheduler-ы и создалась джоба. То есть сейчас мы видим, должны увидеть джобу в кластере. В нашем, естественно, дефолтном кластере. Я просто (НРЗ 12:32), да, уже даже два pod-а. Уже два раза работа прошла. То есть наш контроллер в общем-то работает, всё у него хорошо. Мы его задеплоили, опять же, у нас…, мы можем сделать готовый файл deployment-а. И есть еще такая команда undeploy, которая все следы нашего пребывания с кластера, в общем-то, удалит. Удалит все эти метрики, менеджеры, все роли, и сделает кластер таким, каким он был «до». Довольно удобно, когда вы что-то тестируете, надо каждый раз делать это заново, а не просто версию апдейтить.