И если хочется, соответственно, дальше разобраться с операторами и как-то проанализировать их, рекомендую оператор Azure Databricks Operator, который устанавливает Databricks-овские data engineering и machine learning task-и. То есть устанавливает сервер api-шный, в котором общаются уже оператор. И соответственно, может ему что-то передать на выполнение и соответственно, непосредственно это всё зааркистрировать и выполнить.

Оператор написан с помощью Kube builder-а. То есть они, в принципе, все эти тулзы, кроме того, самого Meta Controller-а, они похожи, то есть используют один и тот же концепт reconciliation и менеджера, но здесь у нас всё точно так же, как в нашем проекте. Открываем main.go, у нас здесь какие-то есть флаги для метрик, (HP3 00:53)…, супер знакомый код, не так ли? Вот создается менеджер и создается apiClient, это уже штука непосредственно характерна для (HP3 01:01). Вы здесь можете, например, в вашем случае свои какие-то параметры дополнительные создать, и создаются их все Reconciler-ы. Там, значит, SecretScope'ы, джобы, RunReconciler-ы, кластерные и так далее.

То есть те все самые Reconciler-ы, которые нужны для всякого там Data Science. То есть мы видим, что… Мы, вернее, не видим, но я, сразу забегая вперед скажу, что один Reconciler отвечает за конкретный один тип CRD, на который мы сейчас посмотрим. И вот сколько у нас есть этих типов, столько у нас и будет этих Reconciler-ов. И все они подвязаны, естественно, на одного менеджера. Он в (HP301:43) будет каждого опрашивать по его ресурсам и всё. И они будут как-то устанавливать состояние кластера. Это совершенно стандартная практика для операторов и котроллеров.

В папочке, соответственно, API мы можем видеть наши все типы как раз, которые принадлежат нашим контроллерам. Здесь ребята пошли другим путём. Если я, например, в своем тесте проверял, что у нас объект создан и у него все эти поля.

Здесь они тесты эти вынесли отдельно и проверяют, что, значит, соответственно, поля у нас созданы, клиенты их успешно добавляют и они, соответственно, успешно удаляются. Какой подход при этом использовать, решать вам. Мне нравится подход, когда поля и все эти тесты проходят у нас там, где в этом есть смысл. Опять же, ребята из Data Bricks считают иначе. Я подозреваю, что они умнее или что-то знают, или куда-то уже напоролись, куда я еще не напарывался, ну и соответственно, применяют каким-то образом, эти тесты разделяют для своего удобства.

Соответственно, внутри каждого котроллера есть логика, специфичная для этого котроллера. У нас здесь все так же примерно, как и у меня в коде. То есть здесь создаются event-ы, тут они добавили концепт finalizer-ов, которые удаляют instance-ы после того, как они отработали. Естественно, это всё там разнесено по файликам. Это уже стандартный у нас пошел enterprise подход. К написанию контроллеров это не имеет никакого… Вернее, это имеет, конечно, отношение, но не имеет непосредственно к тому, о чем я хочу сказать. И, соответственно, тесты лежат точно также в этой папке.

В целом, если вы вот так пишите операторам, то у вас скорее всего оно в приложении будет больше одного контролера, и вот так их можно в папочки сложить и спокойненько ими, соответственно, пользоваться.