Итак, вы положили все зависимости в go mod, у вас прописались зависимости в go sum, скачались локально на компьютер и у вас все прекрасно собирается? и работает. Но если приляжет GitHub, уже такое происходило, приляжет (НРЗ 00:15), уже такое происходило, приляжет Google, угадайте, что происходило уже в этом году, вы останетесь без зависимостей. Другие разработчики и система сборки, которая каждый раз сбрасывает все кэши, что угодно, в общем у вас перестанут собираться приложения. Естественно, это может быть, как приемлемо, но если у вас там никакой не жесткий график релиза, может быть неприемлемо, если у вас реально есть там жесткий график релизов и вам нужно все библиотеки сохранять. Плюс, если вы не хотите зависеть от людей, обновляющих свои библиотеки на GitHub, делающих новые релизы, коммитящих какие-то фичеры в ветке, уже релизнутые, такое тоже бывает, вы можете, так сказать, сделать слепок ваших всех библиотек. Физический слепок кода и положить в папочку проекта. Это называется vendoring. Давайте посмотрим, как он работает.

Итак, как делается vendoring? На самом деле, очень просто. Мы уже видели команду в go mod, если мы про нее забыли, можно еще раз посмотреть, кстати, давайте посмотрим, что нам выведет go mod Graph, вон он. Смотрите? как все красиво выводит. Ну ладно, не об этом сейчас. То есть мы можем, соответственно, использовать команду go mod vender и сделать vendored copy of dependencies. То есть мы просто пишем go mod vendor, и в этот момент происходит как бы скачивание или просто копирование папочки в vendor. И вот, соответственно, у вас есть папка vendor, в которой у вас лежат физические все исходники.

То есть, как это работает в go? Если go не видит ничего в vendor или папочки vendor не существует, он идет в Package. Если существует, он берет все оттуда. Соответственно, все, это — слепок, который вы можете загрузить в GitHub, и это у вас останется в проекте. То есть никакой больше внешней зависимости у вас не будет. Соответственно, здесь у вас перечислены все зависимости vendor-а, и

соответственно, у вас теперь ваш проект будет использовать только зависимости с папочки vendor. Вы можете даже их редактировать, чего я делать, естественно, не рекомендую.

В принципе, это все, что я хотел рассказать вам про управление зависимостями и структурой проектов в go. И давайте посмотрим, какие же мы можем сделать для себя выводы?

Я думаю, что сегодня мы-таки узнали, как структурировать свои приложения. Поняли, что, на самом деле, ничего сложного нет, реально. У нас есть исходные файлы, и они объединяются в Package. Package-и у вас лежат по папочкам. Все вот эти вот вместе папочки, объединяющиеся в единое приложение или какую-то библиотеку, являются модулем. Все, больше в go ничего, реально, нет. И мы попользовались модулями, узнали, как, соответственно, работают зависимости в модулях, узнали, что такое vendoring и как его применять.

Всем спасибо и удачи. Увидимся в следующем блоке.