Закрывая тему наших изображений, работы с нашим изображением. Последние две команды, которые хотел бы рассмотреть – это команда сохранения и команда загрузки изображения.

Мы помним, что Docker работает не только с репозиториями. Вы можете экспортировать изображение в файл, и потом где-то на другом компьютере этот файл вгрузить в свой Docker. И запустить это изображение.

Опять же, в client-е это делается довольно-таки тривиально: client предоставляет команду ImageSave, в которую вы можете передать ID-шники изображений, и соответственно…, тут даже команда супер-простая. Тут, соответственно, изображение передается в query, и из query у вас получается ReadCloser. То есть здесь важно обратить внимание, что в данном случае Docker…, именно (НРЗ 00:51) из этого изображения. То есть это не какой-то там json с ответом, это именно уже само изображение. Поэтому этот ReadCloser надо передать или вывести, соответственно, вы его можете. То есть вы можете с помощью команды Docker save сохранить, например, вот myserver. И вывести его в std, std out put. Опять же, даже тут говорит, что: «Не хочу выводить такое огромное изображение в терминал. Давай-ка ты его лучше сохранишь…, сохранишь файл».

Опять же, да, здесь, в принципе, нету у меня возможности это показать, но тем не менее…, опять же, зачем вам это выводить? То есть вы получаете поток этих байтов, перенаправляете их в файл. Опять же, не забываем все закрыть. И эти файлы я, с помощью команды io.Copy, копирую просто, вернее, эти байты я копирую просто в файл, и файл у меня будет архивом .tar. То есть это такой стандартный формат Docker. Это просто надо знать. Это указано в, опять же, спецификации: Docker save, который является клиентской оберткой, над вот этой самой командой, и сохраняет его в tar архив.

И соответственно, если мы запустим нашу команду, мы получим 28- миллионно битный файл, то есть где-то 27 мегабайт. Я довольно плохо ориентируюсь в размерах файлов в системе. Попробуем его открыть. Вот он тут открылся. Вот 28 тысяч, 164 килобайта. Ну, в общем, как-то так. Здесь больше даже никаких особо настроек у него и нет.

Ну и последнее, что мы будем делать с изображениями для того, чтобы понять, так сказать, контекст Docker-а – это загрузим вот это изображение из того самого файла, который я до этого сохранил. Здесь маленький Disclaimer. Я понимаю, что еще изображение можно build-ить, изображение можно удалять и так далее. Но моя задача дать концепцию. То есть все, в принципе, в этом Docker SDK клиенте работает, на самом деле, абсолютно одинаково. То есть они все такие простые команды: ImageBuild, ImageTag, ImageRemove. То есть они все отражают команды самого Docker SDK. То есть отражает команды…, не отражает, вернее, а клиент Docker-а, он отражает, в принципе, SDK. То есть там никаких особо фишек поверх нету.

То есть вы можете, пользуясь вот этим Docker-ом, в принципе, отриверсить Docker-а SDK. Либо пользуясь, соответственно, исходниками Docker SDK. То есть зайдя сюда, тут все это есть. Здесь, в принципе, какие-то там ну…, по типам можно понять. И спокойно этим пользоваться. Опять же, давать все примеры для всех endpoint-ов – это, наверное, увеличит лекцию раз в 50. И не очень будет полезно. Потому что это не справочная информация, это донесение концепции. Я надеюсь, что она уже на этом этапе, в принципе, понятна.

Соответственно, что мы здесь делаем? Здесь у нас в обратном порядке все происходит, мы читаем файл, вгружаем reader из этого файла. То есть через NewReader вгружаем в ImageLoad. Соответственно, ImageLoad нам вгрузит изображение. И вернет тот самый ответ от Docker SDK. Соответственно, я удалил

myserver, мое изображение, myserver. И давайте посмотрим, что у нас, соответственно, случится. Ну вот, у нас загрузилось изображение, сразу дался его ID-шник. Если мы сейчас вызовем команду DockerList, Docker Images List, то увидим – вот это наше изображение, а также видим, что у него нет репозиторий. Это (НРЗ 05:08) изображение. Почему? Потому что информация о репозитории не сохраняется при выгрузке изображения, сохраняется только сам контент.

Вы можете, опять же, установить tag, с помощью команды cli.ImageTag, у нее тоже довольно все просто. Опять же, context есть, source. Source – это соответственно, ID-шник в репозитории, либо tag в репозитории, которого у нас нет. И вы можете ему поставить нужный, соответственно…, соответственно, tag. В принципе, здесь, наверное, все.

Теперь давайте уже перейдем к остальным функциям Docker SDK.