Некоторое время назад, лет 10-20 назад, такой профессии как DevOps не существовало в принципе. Процессами сборки и доставки обычно занимались сами программисты, которые разрабатывали программное обеспечение. Сами приложения были очень простыми и не требовали большого количества интеграций, содержали в себе все необходимые библиотеки, редко общались с интернетом, да и вообще представляли из себя монолит, который уже содержит всё необходимое. Обслуживанием же «железа» обычно занимались сисадмины, часто DevOps сегодня в шуточном ключе так и называют. Обе команды работали довольно разрозненно и не зависели друг от друга. Разработчик передавал достаточно простой исполняемый файл сисадмину, сисадмин хостит исполняемый файл (ну, или сайт, но в целом процесс был очень прост). И тем не менее, бизнесу потребовалось выделить в отдельную роль специалиста, который работает на стыке ранее независимых сфер программирования и обслуживания.

Потребность в таким специалисте были продиктованы в первую очередь изменениями в трендах разработки. Так или иначе, главной причиной возникновения этой специальности является разбиение приложений на микросервисы и вследствие появившаяся необходимость эти микросервисы между собой связать. Разработчику уже не получится просто передать монолит, который содержит в себе всё. Ему эти монолиты придётся между собой связывать. Например, в экосистеме продукта компании может запускаться микросервис авторизации, через который должен идти абсолютно каждый внутренний продукт компании. Само собой, это удобно – везде пользователь один, не надо постоянно регистрироваться. Если бы мы содержали в одном монолите абсолютно все продукты компании с авторизацией в том числе, такое приложение было бы очень трудно поддерживать, вносить в него быстрые изменения, оно было бы очень неадаптивным, оно было бы жутко медленным и так далее, и так далее. Вообщем, микросервисы – это круто, но сложно, потому что привычная схема ломается. Раньше движение продукта было односторонним, разработчик

максимум получал фидбек и грозные тирады сисадмина. Но теперь разработчику придётся общаться и договариваться с сисадмином, спрашивать, как ему подключиться к микросервису и переводить сисадминский эльфийский на программистский эльфийский. А что, если этих микросервисов не одна сотня? Тут и появляется человек, который имеет системные знания в одной и второй сфере одновременно, то есть работает на стыке разработки (Dev), предоставляя им необходимые интеграции и подключения к микросервисам компании, так и на стороне операций и непосредственной работы продукта (Ops), также предоставляя им продукт в удобном виде, позволяющем запуститься и корректно подключиться к другим интеграциям. В эти задачи входят управление исполняемыми файлами продуктов и также их конфигурация.

Ещё одной задачей DevOps часто становится автоматизацией этих процессов. По природе, у программиста две потребности. Потребность в лени и в повышении зарплаты. И так как DevOps наследовал от программистов приставку Dev, работать муторно и долго вручную он, естественно, не захочет, тем самым удовлетворяя свою потребность в лени. Появление CI/CD позволяет автоматизировать процесс доставки и развёртывания программных продуктов. Кроме того, автоматизация позволяет уменьшить человеческий фактор и, следовательно, повышает надёжность полученного продукта.

Соседствующей задачей с настройкой конфигурацией при сборке становится настройка репозиториев и окружения для разработчиков. DevOps может иметь представление о конфигурации «железа», на котором будет запускаться продукт и следовательно рекомендовать соответствующий стек технологий, библиотек. И так как у DevOps есть ещё вторая потребность в повышении зарплаты, часто он выполняет ещё дополнительную работу по настройке окружения для разработки. То есть, добавляет необходимые линтеры, подключает плагины к среде разработки, добавляет анализаторы, прописывает инструкции по локальному запуску программного обеспечения.

DevOps, как следует из своего названия, является прослойкой между операционными действиями, то есть непосредственному запуску приложений, и разработке. Такая необходимость появилась из-за повышенной сложности разработки, появления экосистем. Без DevOps мир был бы сильно проще на самом деле, так как компании не смогли бы себе позволить разработку крупных сложных удобных и безопасных продуктов. Без DevOps процесс доставки приложения соседствовал бы с крайне неприятным «сайт обновляется, прийдите через полчаса» и большим простоям. То, что мы сегодня принимаем за данность, несколько лет назад было роскошью.