Нужна ли вам непрерывная доставка

Первый вопрос, который вы, возможно, себе задаете, — стоит ли тратить время на изучение непрерывной доставки, и даже если да, стоит ли применять ее в проектах. Ответ — *сто́ит*, если:

- вы профессионально занимаетесь разработкой программного обеспечения;
- в вашем проекте участвует больше одного человека.

Если справедливы оба этих условия, вам непременно следует вложиться в непрерывную доставку. *Даже если выполняется только одно из них* (например, вы работаете над проектом с друзьями ради интереса или разрабатываете профессиональное ПО в одиночку), вы не пожалеете об использовании непрерывной доставки.

> *Но ведь вы даже не спросили, чем я занимаюсь. Что, если я работаю над драйверами ядра, прошивкой или микросервисами? Вы уверены, что мне нужна непрерывная доставка? — спросите вы.*

Это неважно! Какой бы продукт вы ни создавали, принципы, изложенные в этой книге, будут вам полезны. В их основе лежат идеи, которые сформировались еще на заре программной разработки. Это не модные тренды, которые со временем исчезают или теряют популярность; это основы, которые останутся неизменными, независимо от того, создаете ли вы микросервисы, монолитные приложения, распределенные сервисы на основе контейнеров или что-то еще.

> ***Но мне не нужно ничего развертывать!***
>
> Совершенно верно! Развертывание и связанная с ним автоматизация применимы не ко всем видам ПО, однако непрерывная доставка — это гораздо больше, чем просто развертывание. Мы поговорим об этом позже в этой главе.

В этой книге рассматриваются основные принципы непрерывной доставки и приводятся примеры их практического использования. Конкретные детали реализации в проекте наверняка будут уникальными, и возможно, вы не найдете здесь их точного описания, но что вы точно найдете, так это компоненты, необходимые для автоматизации непрерывной доставки, и принципы, соблюдая которые вы добьетесь наибольшего успеха.

Зачем нужна непрерывная доставка

Итак, что же это за штука такая? Начну с того, что непрерывная доставка (CD) означает для меня и почему я считаю ее такой важной:

Непрерывная доставка — это один из процессов современной профессиональной программной разработки.

Разберем это определение по частям:

- *Современная* — профессиональная разработка существует гораздо дольше, чем непрерывная доставка, хотя ребята, которые работали с перфокартами, были бы в восторге от CD! Одна из причин, по которой мы можем сейчас использовать непрерывную доставку, а тогда не могли, заключается в том, что CD отнимает много ресурсов процессора. Непрерывная доставка требует выполнения большого объема кода!

> Я даже не могу себе представить, сколько перфокарт потребовалось бы для описания типичного рабочего процесса CD!

- *Профессиональная* — если вы пишете программы для интереса, то вопрос о том, стоит ли вам возиться с непрерывной доставкой, остается открытым. Как правило, ее используют, когда действительно важно, чтобы программный продукт работал. Чем важнее создаваемое ПО, тем тщательнее следует продумывать непрерывную доставку. Кроме того, говоря о профессиональной разработке, мы вряд ли будем иметь в виду одного программиста, самостоятельно пишущего код. Обычно над продуктом работает несколько человек, а иногда даже сотни.
- *Программная разработка* — в других инженерных областях существуют своды стандартов и сертификатов, которые в разаботке, как правило, отсутствуют. Итак, проще говоря, программная разработка — это написание программ. Когда мы добавляем слово «*профессиональная*», мы имеем в виду профессиональное занятие разработкой ПО.
- *Процесс* — профессиональная разработка требует соблюдения определенных подходов, чтобы написанный нами код делал то, что мы задумали. Эти процессы касаются не столько способов, которыми отдельно взятый программист пишет код (хотя и это важно), сколько способности этого человека сотрудничать с другими разработчиками, чтобы создавать продукт профессионального уровня.

Непрерывная доставка — это совокупность процессов, которые необходимы группе профессиональных разработчиков, чтобы создавать программный продукт, соответствующий исходным целям его создателей.

Подождите, вы хотите сказать, что CD означает «непрерывная доставка»? Я думал, что это непрерывное развертывание!

Некоторые действительно расшифровывают эту аббревиатуру именно так (continuous deployment), и тот факт, что оба термина появились примерно в одно и то же время, вносит большую путаницу. В большинстве известных мне публикаций (не говоря уже о сообществе Continuous Delivery Foundation!) авторы предпочитают употреблять аббревиатуру CD для обозначения непрерывной доставки, поэтому в этой книге мы будем делать то же самое.

Непрерывная доставка

Непрерывная доставка — это совокупность процессов, которые необходимы группе профессиональных разработчиков, чтобы создавать программный продукт, соответствующий исходным целям его создателей.

Мое определение отражает то, что я считаю действительно крутым функционалом CD, но оно слишком далеко от стандартной формулировки, которую вы можете встретить. Посмотрим на определение, данное организацией Continuous Delivery Foundation (CDF) (http://mng.bz/YGXN):

Практика программной разработки, при которой команды выпускают релизы изменений ПО для пользователей безопасным, быстрым и устойчивым образом благодаря:

- **способности осуществлять выпуски в любое время;**
- **автоматизации выпуска релизов.**

Заметьте, что CD подразумевает два основных условия. Вы занимаетесь непрерывной доставкой, если:

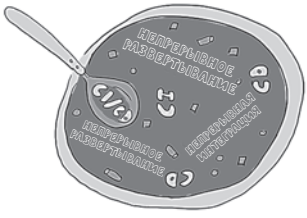
- выпускаете релизы ПО безопасно и в любое время;
- делаете выпуск максимально просто, буквально одним нажатием кнопки.

В этой книге подробно описаны необходимые действия и средства автоматизации, которые помогут вам добиться этих двух целей, а именно:

- Чтобы выпускать изменения безопасно и в любое время, ПО должно всегда находиться в состоянии готовности к релизу (в состоянии

Важным принципиальным изменением, произошедшим в CD по сравнению с CI, является переопределение значения фразы «функция готова». В CD «готова» означает «выпущена» (то есть осуществлен релиз). А процесс перехода от реализации изменений к выпуску автоматизирован, прост и быстр.

История «непрерывных» терминов



- **1994:** *термин «Непрерывная интеграция» (Continuous integration) вводится в книге Грейди Буча (Grady Booch) Object-Oriented Analysis and Design with Applications* [1] *(Addison-Wesley).*
- **1999:** *методика непрерывной интеграции впервые описана в книге Кента Бека (Kent Beck) Extreme Programming Explained*[2] *(Addison-Wesley).*
- **2007:** *дальнейшее развитие методика непрерывной интеграции получила в книге Пола Дюваля (Paul M. Duvall) Continuous Integration*[3] *(Addison-Wesley).*
- **2007:** *термин «непрерывное развертывание» определен в той же книге Дюваля.*
- **2009:** *принципы непрерывного развертывания популярно описаны в блоге Тимоти Фитца (Timothy Fitz) (http://mng.bz/2nmw).*
- **2010:** *методика непрерывной доставки, вдохновленная agile-манифестом*[4]*, описана в книге Continuous delivery*[5] *Джеза Хамбла (Jez Humble) и Дэвида Фарли (David Farley) (Addison-Wesley).*
- **2014:** *первая статья, дающая определение технологии CI/CD, Test Automation and Continuous Integration & Deployment (CI/CD) («Автоматизация тестирования и непрерывная интеграция и развертывание (CI/CD)»), опубликована сообществом Ravello (http://mng.bz/1opR).*
- **2016:** *в Википедию добавлена статья «CI/CD» (http://mng.bz/J2RQ).*

Вы, должно быть, думаете: «О'кей, Кристи, это все хорошо и замечательно, но что же на самом деле означает *доставлять*»? А *непрерывное развертывание*? И что такое *CI/CD*?

Мы действительно вынуждены оперировать множеством терминов! И что еще хуже, разные люди используют их по-разному. В их защиту можно сказать одно: это происходит потому, что некоторые из таких терминов даже не имеют определений!

Остановимся вкратце на эволюции этих терминов, чтобы лучше понять их значение. Непрерывная интеграция, непрерывная доставка и непрерывное развертывание — это термины, которые были созданы намеренно (или, в случае непрерывной интеграции, эволюционировали), а их авторы придавали им совершенно конкретные значения.

CI/CD — особый случай: похоже, этот термин никто не создавал и он появился на свет только потому, что люди, говорившие одновременно обо всех «непрерывных» операциях, нуждались в кратком термине (при этом полная аббревиатура CI/CD/CD почему-то не прижилась!).

Термин CI/CD в том виде, в каком он используется сегодня, означает инструменты и средства автоматизации, применяемые во всех непрерывных операциях — интеграции, доставке и развертывании.

---

[1] Буч Г. и др. «Объектно-ориентированный анализ и проектирование с примерами приложений».

[2] Бек К. «Экстремальное программирование». Санкт-Петербург, издательство «Питер».

[3] Дюваль Пол М. и др. «Непрерывная интеграция: улучшение качества программного обеспечения и снижение риска».

[4] Манифест разработки программного обеспечения по методологии agile. — *Примеч. пер.*

[5] Хамбл Д., Фарли Д. «Непрерывное развертывание ПО. Автоматизация процессов сборки, тестирования и внедрения новых версий».

релиза). В этом нам поможет непрерывная интеграция (continuous integration, CI).
- После проверки изменений методами CI процесс выпуска изменений должен проходить автоматически и его должно быть легко повторить.

Прежде чем я начну подробно рассказывать о том, как достичь этих целей, разберемся в терминологии.

Непрерывная доставка — это набор целей, к которым мы стремимся; способ их достижения может меняться от проекта к проекту. Тем не менее достигать этих целей наиболее эффективно помогает определенная последовательность действий, и именно им посвящена моя книга!

Интеграция

Непрерывная интеграция (Continuous Integration, CI) — старейшее из понятий, с которыми мы познакомились, однако это по-прежнему ключевой элемент непрерывной доставки. Начнем с простого — рассмотрим пока только интеграцию.

Что значит *интегрировать* программу? На самом деле часть фразы опущена: интегрировать объект нужно во что-то другое. В разработке интегрируемый объект — это изменение программного кода. Когда мы говорим об интеграции программного обеспечения, фактически мы имеем в виду

интеграцию изменений кода в существующее программное обеспечение.

Это именно то, чем в основном каждый день занимаются разработчики: изменяют код существующих частей программы. Этот процесс особенно интересен в команде: ее члены постоянно вносят изменения в код, и зачастую в одну и ту же часть продукта. Объединение этих изменений в единое целое и есть *интеграция*.

Интеграция программного обеспечения — это объединение изменений кода, сделанных несколькими людьми.

Иногда вам приходится создавать программу с нуля, но после первой успешной компиляции вы раз за разом будете интегрировать новые изменения в уже существующий программный продукт.

Как вы, вероятно, знаете по своему опыту, иногда этот процесс действительно идет криво. Например, когда я изменяю ту же строку кода, что и вы, и мы пытаемся объединить наши изменения, возникает конфликт и нам приходится вручную решать, как их интегрировать.

И еще одна маленькая деталь. Когда мы интегрируем изменения кода, мы не только просто соединяем их вместе; *мы еще и проверяем работоспособность измененного кода.* Можно сказать, что в аббревиатуре CI не хватает буквы *V* (Verification) —

верификация! Верификация уже заложена в процесс интеграции, поэтому, говоря об интеграции ПО, мы имеем в виду следующее:

Интеграция программного обеспечения — это объединение изменений кода, сделанных несколькими людьми, и проверка того, что получившийся код делает именно то, для чего он предназначен.

Кого волнуют все эти определения? Покажите мне уже наконец код!

Трудно выполнять свою работу последовательно и методично, если ей даже нельзя дать четкое определение. Уделить время тому, чтобы прийти к общему пониманию (через определения), а затем вернуться к основным принципам — это самый эффективный способ подняться на новый уровень!

Непрерывная интеграция

Посмотрим, как придать *интеграции* «*непрерывность*», на примере, не связанном с разработкой. Холли, шеф-повар, готовит соус для макарон. Она начинает с подбора продуктов: лука, чеснока, помидоров, специй. Чтобы приготовить соус, ей нужно *интегрировать* эти продукты вместе в правильном порядке и в нужном количестве.

Для этого каждый раз, когда она добавляет новый ингредиент, *она быстро пробует соус на вкус*. Исходя из вкуса, она может добавить больше какого-то продукта либо понимает, что забыла какой-то нужный ингредиент.

Дегустация помогает ей менять блюдо, проводя его через серию интеграций. Интеграция в данном случае подразумевает две вещи:

- объединение ингредиентов;
- проверку для подтверждения результата.

И это именно то, что означает слово «*интеграция*» в словосочетании *непрерывная интеграция*: объединение изменений кода, а также проверка его работоспособности, то есть объединение и верификация.

Холли повторяет эти шаги во время готовки. Если бы она попробовала соус только в конце, она гораздо меньше контролировала бы процесс и было бы уже поздно вносить необходимые изменения. Именно здесь вступает в дело «*непрерывность*». Вы должны интегрировать (объединять и верифицировать) свои изменения так часто, как только можете, и делать это как можно быстрее.

А как часто вы можете объединять и верифицировать программный продукт? Каждый раз, как только вы внесете изменения!

Непрерывная интеграция — это процесс, при котором объединение изменений кода происходит постоянно и каждое изменение проверяется при внесении.

Объединение изменений кода означает, что разработчики, использующие непрерывную интеграцию, фиксируют и переносят код в общую систему контроля версий каждый раз, когда вносят изменения, и проверяют их корректную работу, применяя автоматические средства верификации, такие как тесты и линтинг[1].

Автоматическая верификация? Линтинг? Не волнуйтесь, если вы не знаете, что это, эта книга поможет вам разобраться! Позже мы рассмотрим, как создавать автоматические проверки, благодаря которым работает непрерывная интеграция.

Что мы доставляем

От непрерывной интеграции мы переходим к непрерывной доставке, и для этого нужно вернуться немного назад. Почти в каждом определении, которое мы рассматриваем, упоминается доставка какой-то программы (например, я собираюсь начать рассказ об интеграции и доставке изменений в программу). Неплохо бы убедиться, что мы все имеем в виду одно и то же, когда говорим о программах (software), ведь в зависимости от проекта этот термин может обозначать совершенно разные вещи.

Доставляемые программы могут иметь различные формы (интеграция и доставка для каждой из них тоже будут проходить по-разному):

- *Библиотека* — если программа ничего не делает сама по себе, а предназначена только для использования в составе другого ПО, то это, скорее всего, библиотека.
- *Двоичный файл* — если программа предназначена для выполнения какой-то задачи, вероятно, это двоичный исполняемый файл. Это может быть сервис, или приложение, или инструмент, который выполняет определенные действия и затем завершает работу, или приложение для мобильного устройства, например планшета или телефона.
- *Конфигурация* — это понятие относится к информации, которую можно передать в двоичный файл, чтобы изменить его работу без перекомпилирования.

Словарик

Термин «*программное обеспечение*» (software) возник в противоположность термину «*аппаратное обеспечение*» (hardware). Аппаратное обеспечение — это собственно физические части компьютеров. Мы производим действия с этими частями, снабжая их инструкциями. Инструкции могут быть встроены непосредственно в аппаратное обеспечение или переданы ему во время работы с помощью программного обеспечения.

[1] Линтинг — проверка кода на соответствие стандартам. — *Примеч. пер.*

Как правило, оно обозначает средства, доступные системному администратору для внесения изменений в работающее ПО.

- *Образ* — образы контейнеров представляют собой особый тип двоичных файлов, ставший чрезвычайно популярным форматом совместного использования и распространения сервисов вместе с их конфигурацией, чтобы они могли выполняться независимо от операционной системы.
- *Сервис* — как правило, сервисы — это двоичные файлы, которые постоянно находятся в рабочем состоянии, ожидая запросов, на которые они отвечают, выполняя какие-либо действия или возвращая информацию. Иногда их также называют *приложениями*.

На разных этапах карьеры вы можете работать с отдельными видами программ или со всеми сразу. Но вне зависимости от того, с какой программой вы будете иметь дело, для ее создания вам потребуется осуществлять *интеграцию* и *доставку* вносимых в нее изменений.

Доставка

Что означает *доставить* изменения в программу, зависит от того, какой программный продукт вы создаете, кто его использует и как. Обычно доставка изменений имеет отношение либо только к одному из процессов сборки, релиза и развертывания ПО, либо сразу ко всем:

- *Сборка* — ряд действий, направленных на получение кода (включая его изменения) и преобразование его в пригодную для использования форму. Обычно это означает компиляцию кода, написанного на языке программирования, в машинный язык. Иногда это также означает помещение кода в пакет, например в образ контейнера или похожий, который менеджер пакетов сможет распознать (например, в пакет PyPI для Python).
- *Публикация* — копирование программного обеспечения в репозиторий (место хранения программ), например, путем загрузки образа или библиотеки в реестр пакетов.
- *Развертывание* — копирование ПО в место, где оно должно быть запущено, и приведение его в рабочее состояние.

Сборка осуществляется в рамках процесса интеграции, чтобы проверить, что все внесенные изменения корректно работают вместе.

Можно проводить развертывание без выпуска релиза: например, развернуть новую версию программы, но не направлять на нее трафик. Однако чаще всего развертывание все же предполагает релиз; все зависит от того, где оно происходит. Если вы развертываете ПО в производственной среде (в продакшене), то выпуск релиза происходит одновременно с развертыванием. Подробнее о развертывании см. в главе 10.

- *Выпуск релиза* — предоставление продукта пользователям. Релиз можно выпускать путем загрузки образа или библиотеки в репозиторий или установки параметров конфигурации для направления определенного процента трафика на развернутый экземпляр программы.

Словарик

Мы осуществляем *сборку* программного обеспечения с тех пор, как у нас появились языки программирования. Это настолько распространенный процесс, что первые системы, которые выполняли то, что мы сейчас называем *непрерывной доставкой*, именовались *системами сборки* (build systems). Этот термин стал таким привычным, что даже сегодня вы слышите слово *сборка* (или *билд*), хотя при этом имеется в виду последовательность задач в пайплайне развертывания, которая выполняется для преобразования ПО (подробнее об этом см. в главе 2).

Непрерывная доставка/непрерывное развертывание

Теперь вы знаете, что значит доставлять изменения в программное обеспечение, но что значит делать это непрерывно? В контексте CI мы узнали, что «*непрерывно*» означает «*как можно быстрее*». Так ли это в случае CD? И да и нет. Непрерывность в CD лучше представить в виде диапазона:

Создаваемое вами ПО, должно находиться в состоянии, при котором в любое время можно выполнить его сборку, релиз и/или развертывание. Но как часто вы решите доставлять это ПО, зависит только от вас.

- 2009: статья в блоге, описывающая принципы непрерывного развертывания
- 2010: методика непрерывной доставки описана в книге с аналогичным названием

«А как насчет непрерывного развертывания?» — спросите вы. Отличный вопрос. Если снова обратиться к истории, то можно заметить, что эти два термина, *непрерывная доставка* и *непрерывное развертывание*, получили широкое распространение практически одновременно. Что же происходило в то время, когда эти термины вошли в оборот?

Это был переломный этап в разработке программного обеспечения: старые способы создания программных продуктов, когда все зависело от людей и ручных операций, жесткое разделение разработки и эксплуатации программ (интересно, что термин *DevOps* (development&operations, разработка и эксплуатация) появился примерно в то же время) и четко определенные процессы разработки ПО (например, *этап тестирования*) — все это начало меняться (со сдвигом влево). Непрерывное развертывание и непрерывная доставка совместно легли в основу практик, появившихся в это время. *Непрерывное развертывание* означает, что:

выпуск релиза рабочей версии ПО для пользователей осуществляется автоматически при каждом коммите.

Словарик

Сдвиг влево — это процесс поиска дефектов на начальных этапах создания программного продукта.

Непрерывное развертывание — это дополнительный этап непрерывной доставки. Непрерывная доставка позволяет осуществлять также и непрерывное развертывание. Постоянное пребывание ПО в состоянии готовности к релизу и автоматизация процесса доставки освобождают вас от необходимости выбирать, что будет лучше для проекта.

Если непрерывное развертывание на самом деле относится к выпуску релизов, почему бы не назвать его непрерывным релизом?

Отличная мысль! *Непрерывный релиз* — более точное название, и оно прояснило бы, как эту методику можно применять к ПО, которое не нужно развертывать. Однако прижился именно термин «*непрерывное развертывание*». Пример непрерывного релиза см. в главе 9.