Оглавление

Введение

Индустрия разработки программного обеспечения переживает медленную, но реальную трансформацию.

Программное обеспечение все чаще становится частью всего, и мы, разработчики, пытаемся справиться с этой растущей потребностью за счет большей автоматизации. Я полагаю, вы читаете эту книгу, потому что являетесь частью данной трансформации.

Чтобы помочь вам в этом преобразовании, Jenkins сам переживает серьезные изменения – от мира «классического» Jenkins, где вы настраиваете его через серию заданий из графического интерфейса на стороне сервера, до мира «современного» Jenkins, где вы настраиваете Jenkins через файлы Jenkinsfile в Git-репозиториях и просматриваете результаты в симпатичном одностраничном приложении.

По мере развития современного Jenkins в сообществе и развертывания этих новых функций я продолжаю сталкиваться с данной проблемой. Большинство пользователей просто не знает о трансформации, которая происходит в Jenkins. Люди продолжают использовать Jenkins, так как они делали это годами!

И чтобы быть справедливым, это имело полный смысл. С одной стороны, это инерция людей и это огромный массив информации и знаний, накопленных в Google, Stack Overflow, наших списках рассылки, средствах отслеживания ошибок и т. д., которые рассказывают людям, как эффективно использовать Jenkins «классическим» способом. С другой – у нас есть сообщество, которое, вообще-то говоря, слишком занято созданием «современного» Jenkins; и в целом недостаточно усилий было потрачено на то, чтобы рассказать людям, как эффективно использовать Jenkins современным способом.

Поэтому я был очень рад услышать об этой книге, которая действительно принимает данный вызов.

В своей книге Брент делает шаг назад и забывает все, что мы узнали о Jenkins за последнее десятилетие. Затем он продолжает воссоздавать то, как Jenkins должен использоваться сегодня.

В отличие от Google, Stack Overflow и т. д., где знания собираются по частям, эта книга дает вам систематизированный маршрут для изучения всего ландшафта, что делает ее действительно ценной.

Это идеальная книга для тех, кто плохо знаком с непрерывной интеграцией и непрерывным развертыванием, а также для тех, кто использует Jenkins в течение многих лет. Она поможет вам узнать и заново открыть для себя Jenkins.

Косукэ Кавагути,
создатель Jenkins, технический директор, CloudBees, Inc.,
февраль 2018

Предисловие

Как использовать эту книгу

Эта книга большая – больше, чем я думал. Я беспокоился об этом в какой-то мере, но решил, что при ее написании есть два пути: я могу либо ограничиться только тем, что необходимо для базового занятия, либо потратить некоторое время на объяснение концепций, создание примеров кода и погрузиться в то, что на самом деле означают терминология, функции и программирование конвейеров в виде кода. Если вы отсканировали книгу, то, возможно, поймете, что я решил сделать последнее.

Мое рассуждение об этом было связано с многолетним опытом обучения людей использованию Jenkins. На коротком занятии или семинаре мы могли затронуть лишь небольшое количество тем. И люди всегда жаждали большего – большего количества деталей и примеров, которые они могли бы применить. В конце выступлений на конференциях я неизменно получал сообщения от людей, которые спрашивали о дополнительных источниках информации, примерах и о том, где найти сведения о том-то и том-то. Зачастую все сводилось к фразам «Поищите в Google» или «Посмотрите этот вопрос на Stack Overflow». В этом нет ничего плохого, но это также и не самый удобный подход.

Данная книга призвана помочь вам найти ответы на вопросы о том, как использовать эту мощную технологию. Конечно, это больше механика, чем DevOps, но есть вероятность того, что если вы читаете это, у вас уже есть некоторое представление о непрерывной интеграции (CI), непрерывном развертывании (CD), DevOps и Jenkins и вы хотите максимально использовать новые возможности Jenkins.

Итак, вот несколько рекомендаций (не стесняйтесь использовать их или игнорируйте их в соответствии с ситуацией):

- не пытайтесь прочитать всю книгу до конца – если только вам не нужно много спать;
- сканируйте разделы, перечисленные в оглавлении. Заголовок главы только намекает на ее полное содержание. Кроме того, не забудьте проконсультироваться с указателем, чтобы найти темы, которые могут вас заинтересовать;

- если вы хотите понять основные идеи и быстро приступить к работе, прочитайте первые две главы, а затем поэкспериментируйте с несколькими базовыми конвейерами. Когда у вас появятся вопросы или проблемы, обратитесь к соответствующим главам книги для изучения отдельных моментов;
- если вы уже знакомы с основами Jenkins и хотите выполнить конвертацию, придерживаясь концепции pipelines-as-code, обратитесь к главе 10, чтобы познакомиться с некоторыми идеями по поводу конверсий, а затем при необходимости обратитесь к другим главам;
- если вы хотите создать более крупный конвейер, обратитесь к главе, посвященной конвертации, и главам, где рассказывается об интеграции с ОС и других технологиях (главы 10–14). И не забывайте о безопасности – об этом тоже есть глава (глава 5);
- если вы хотите автоматизировать Jenkins, посмотрите главу 15;
- если вы столкнулись с проблемами, каждая глава содержит детали, которые могут помочь. Посмотрите на примечания, предупреждения и боковые панели для получения информации о необычных ситуациях или функциях, которые могут сбить вас с толку (или предоставить преимущество, о котором вы даже не думали). В конце книги также есть глава о более общих проблемах.

Я открыто признаю проблему, возникающую при написании любой технической книги в наши дни, а именно: технологии быстро развиваются. В процессе написания глав данной книги я возвращался к ней, пытаясь не отставать от последних изменений и нововведений, и пересматривал главы по мере необходимости. Я твердо убежден, что материал в этой книге обеспечит вам хорошую основу и предоставит справочную информацию для работы с Jenkins 2. Но, конечно, вы всегда должны обращаться к последней документации сообщества для обновлений и новых инноваций.

И наконец, просьба: даже если вам не нужно читать большую часть книги, если вы находите отрывки, которые вы прочитали, полезными, пожалуйста, найдите время и оставьте отзыв. Люди узнают о полезных книгах главным образом через сарафанное радио и онлайн-обзоры. Ваш отзыв может оказать огромное влияние.

Спасибо, и надеюсь увидеть вас на будущих тренингах или конференциях!

Обозначения, используемые в этой книге

В этой книге используются следующие типографские обозначения.

Курсив

Обозначает новые термины, URL-адреса, адреса электронной почты, имена и расширения файлов.

`Моноширинный шрифт`

Используется для листингов программ, а также в абзацах для ссылок на элементы программы, такие как имена переменных или функций, базы данных, типы данных, переменные среды, операторы и ключевые слова.

`Моноширинный полужирный шрифт`

Показывает команды или другой текст, который должен вводиться пользователем буквально.

`<Моноширинный курсив>`

Показывает текст, который должен быть заменен значениями, вводимыми пользователем, или значениями, определенными контекстом.

СОВЕТ

Этот элемент означает подсказку или предложение.

ПРИМЕЧАНИЕ

Этот элемент означает общее примечание.

ПРЕДУПРЕЖДЕНИЕ

Этот элемент указывает на предупреждение или предостережение.

Использование примеров кода

Дополнительный материал (примеры кода, упражнения и т. д.) можно скачать по адресу *https://resources.oreilly.com/examples/0636920064602*.

Эта книга здесь, чтобы помочь вам сделать вашу работу. В общем, если пример кода предлагается с этой книгой, вы можете использовать его в своих программах и документации. Вам не нужно обращаться к нам за разрешением, если вы не воспроизводите значительную его часть. Например, написание программы, которая использует несколько фрагментов кода из этой книги, не требует разрешения. Продажа или распространение CD-ROM с примерами из книг O'Reilly требует разрешения. Чтобы ответить на вопрос, сославшись на эту книгу и приведя пример кода, разрешения не требуется. Включение значительного количества примеров кода из этой книги в документацию вашего продукта требует разрешения.

Мы ценим, но не требуем, установление авторства. Установление авторства обычно включает в себя название, автора, издателя и ISBN. Например: Jenkins 2: Up and Running by Brent Laster (O'Reilly). Copyright 2018 Brent Laster, 978-1-491-97959-4.

Если вы считаете, что использование примеров кода выходит за рамки добросовестного использования или указанных выше полномочий, свяжитесь с нами по адресу *permissions@oreilly.com*.

ВАЖНОЕ ПРИМЕЧАНИЕ О ПРИМЕРАХ КОДА В ЭТОЙ КНИГЕ

Во многих случаях, когда листинги кода встречаются в книге, отдельные строки слишком длинные, чтобы поместиться в печатном пространстве. В этих случаях код оборачивается и продолжается на следующей строке (строках). Как правило, в этих строках нет символов продолжения строки. Тем не менее обычно можно сказать, где, семантикой команды или отступом, код был продолжен из строки выше.

ПРИМЕЧАНИЕ О ЦИФРАХ В ЭТОЙ КНИГЕ

В этой книге было использовано много скриншотов и рисунков, чтобы помочь разъяснить информацию читателю. Качество и масштабирование некоторых визуальных элементов могут различаться в зависимости от методов, используемых для их захвата. Также, так как сообщество Jenkins часто выпускает обновленные версии приложения и его плагинов, приведенные в книге визуальные представления могут быть изменены.

Как с нами связаться

Вопросы и замечания по поводу этой книги отправляйте в издательство:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
707-829-0515 (международный или местный)
707-829-0104 (факс)

Для этой книги создана веб-страница, на которой публикуются сведения о замеченных опечатках, примеры и разного рода дополнительная информация. Адрес страницы *http://bit.ly/agile-application-security*.

Замечания и вопросы технического характера следует отправлять по адресу *bookquestions@oreilly.com*.

Дополнительную информацию о наших книгах, конференциях и новостях вы можете найти на нашем сайте по адресу *http://www.oreilly.com*.

Читайте нас на Facebook: *http://facebook.com/oreilly*.
Следите за нашей лентой в Twitter: *http://twitter.com/oreillymedia*.
Смотрите нас на YouTube: *http://www.youtube.com/oreillymedia*.

Благодарности

Самая большая благодарность за эту книгу – сообществу Jenkins. Jenkins является доказательством того, что разработанное сообществом и поддерживаемое им программное обеспечение может быть невероятно полезным, универсальным и качественным. Спасибо всем, кто внес свой вклад в Jenkins или участвовал в разработке плагинов либо учебных материалов, отвечая на вопросы или выпуская релизы Jenkins.

В отдельности есть много людей, которых можно поблагодарить. Единственный способ, который мне приходит на ум, – сделать это общими категориями.

Спасибо Косукэ Кавагути (Kohsuke Kawaguchi) за создание Hudson, а затем и Jenkins, и за согласие написать предисловие к этой книге. Технический драйв и лидерство, которое вы привносите в Jenkins через сообщество и CloudBees, оказали огромное положительное влияние на то, как мы создаем и поставляем программное обеспечение.

Спасибо техническим редакторам, Патрику Вулфу (Patrick Wolfe), Брайану Доусону (Brian Dawson) и Хаиму Краузе (Chaim Krause).

Они потратили много времени на рецезирование книги – и я ценю это. Содержание стало неизмеримо лучше благодаря их отзывам.

Патрик Вульф сыграл важную роль в предоставлении технических обновлений и дополнительной информации на ранней стадии выхода книги. Это помогло убедиться в том, что в большинстве случаев книга соответствует текущему состоянию Jenkins (по крайней мере, на момент выпуска). Его вклад неоценим, и я ценю то время и открытость, которые он уделил этому проекту.

Брайан Доусон также оказал большую помощь, отмечая изменения и места, где книга может быть улучшена для пользователей Jenkins. Хотя Брайан и Патрик оба работают в CloudBees, они иллюстрируют стремление компании свободно отдавать себя сообществу Jenkins.

Хаим Краузе – один из самых преданных мне людей. Проработав с ним над двумя книгами, я всегда ценил его усилия и внимание к деталям. Он тратит время, чтобы опробовать что-то и указать, где формулировка или примеры нуждаются в обновлении или изначально не имеют смысла. В книге есть ряд деталей, которые обязаны ему своей точностью.

Огромное спасибо персоналу O'Reilly. Во-первых, спасибо Брайану Фостеру (Brian Foster), редактору, который был готов рискнуть и поддерживал эту книгу на протяжении всего пути. Спасибо Анжеле Руфино (Angela Rufino), которая помогла мне быть в курсе процесса, отвечала на все мои вопросы и обеспечивала надзор, чтобы довести книгу до конца. Также спасибо Нэн Барбер Nan Barber за ее своевременную работу по редактированию.

Хочу публично выразить признательность Дуайту Рэмси (Dwight Ramsey) и Рэйчел Хед (Rachel Head), редакторам, за то, что они сделали мой текст читабельным и понятным, Джастину Биллингу (Justin Billing), редактору производства, и Жасмин Квитин (Jasmine Kwityn), корректору, за то, что они собрали все воедино для создания окончательного и отшлифованного продукта.

Большей частью материала, изложенного в этой книге, я впервые поделился и довел до совершенства на занятиях в режиме реального времени, которые я провожу для платформы O'Reilly Safari, и на семинарах во время конференций. Спасибо Сьюзен Конант (Susan Conant) (снова вместе с Брайаном Фостером) за то, что выслушали мои идеи, касающиеся занятий по Jenkins 2, и за помощь в их развитии. Кроме того, спасибо Вирджинии Уилсон (Virginia Wilson) за дополнительные возможности для написания материалов по непрерывной интеграции и непрерывному развертыванию, а также организаторам конференции Рэйчел (Румелиотис Rachel Roumeliotis) и Одре Картер (Audra Carter) за руководство заседаниями.

Наконец, в O'Reilly я хочу поблагодарить обучающий персонал, оказавший поддержку большому количеству занятий в режиме реального времени, которые я проводил по Git и Jenkins. Спасибо Ясмине Гре-

ко, Линдсей Вентимилье, Нурул Ишаку и Шеннон Катт (Yasmina Greco, Lindsay Ventimiglia, Nurul Ishak, Shannon Cutt) за то, что они наблюдали за всеми занятиями и следили за тем, чтобы все было на профессиональном уровне.

Говоря о конференциях, было бы упущением не упомянуть Джея Цимермана (Jay Zimmerman). Джей является основателем и организатором серии конференций No Fluff Just Stuff и впервые предоставил мне возможность выступить на мероприятиях по всей стране, посвященных Jenkins.

Спасибо руководству SAS за поддержку моих инициатив по созданию и проведению корпоративных курсов обучения на протяжении многих лет для сотрудников компании и по всему миру. Я особенно благодарен Гленну Мусиалу, Синди Шнаппер и Энди Диггельманну (Glenn Musial, Cyndi Schnupper, Andy Diggelmann) за поддержку моих усилий.

Спасибо всем, кто посетил один из моих тренингов или семинаров по Jenkins, особенно тем, кто задал вопрос и/или оставил отзыв, чтобы я больше думал над темами и способами улучшения контента.

Спасибо тем, кто работает в CloudBees от имени сообщества Jenkins, чтобы развивать Jenkins, отвечать на вопросы и предоставлять документацию для всех нас как пользователей, мы ценим ваши усилия. Их слишком много, чтобы перечислить их все, но несколько раз появлялись имена, когда я исследовал материал для книги, в том числе Патрик Вулф, Джесси Глик, Эндрю Байер, Джеймс Думей, Лиам Ньюман и Джеймс Браун (Patrick Wolfe, Jessie Glick, Andrew Bayer, James Dumay, Liam Newman, James Brown). Если вы видите что-то, написанное этими парнями, прочитайте это, и вы, вероятно, узнаете что-то полезное. Также спасибо Максу Арбаклу (Max Arbuckle) за координацию конференций Jenkins World, где была впервые представлена большая часть сведений о Jenkins 2.

Самая глубокая благодарность из всех должна быть выражена моей жене Анн-Мари и моим детям. Эта книга писалась в течение длительного периода времени, в основном по ночам и выходным, что отнимало у них время, пока я писал о чем-то, что казалось им чуждым. Тем не менее они всегда поддерживали меня. Анн-Мари, ты была мне самой большой поддержкой и источником силы и воодушевления, как и во всем. Спасибо вам за это и за то, что помогли мне сохранить порядок и баланс между жизнью, мечтами и работой. Вы приносите мне доброту, любовь и вдохновение каждый день нашей совместной жизни, и за это я искренне благодарен.

Наконец, спасибо читателям данной книги. Я искренне надеюсь, что вы извлечете из нее пользу, и это поможет вам добиться прогресса в использовании Jenkins и всех связанных с этим вопросов.

Глава 1

Представляем Jenkins 2

Добро пожаловать в «*Jenkins 2: приступаем к работе*». Независимо от того, являетесь ли вы администратором сборки, разработчиком, тестировщиком или кем-то еще, вы попали в нужное место, чтобы узнать об эволюции Jenkins. С этой книгой вы на пути к использованию возможностей Jenkins 2 для проектирования, реализации и исполнения ваших конвейеров с таким уровнем гибкости, контроля и простоты в обслуживании, который ранее не был возможен в Jenkins. И, независимо от вашей роли, вы быстро увидите преимущества.

Если вы разработчик, написание вашего конвейера в виде кода будет более удобным и естественным. Если вы профессионал DevOps, поддерживать ваш конвейер станет проще, потому что вы можете обращаться с ним как с любым другим набором кода, который управляет ключевыми процессами. Если вы являетесь тестировщиком, то сможете воспользоваться расширенной поддержкой таких функций, как параллелизм, чтобы получить больше возможностей для ваших усилий. Если вы менеджер, то сможете обеспечить качество вашего конвейера так же, как и для исходного кода. Если вы пользователь Jenkins, вы существенно расширите свою базу навыков и будете готовы к новой эволюции концепции «pipelines-as-code».

Достижение этих целей требует понимания и планирования перехода от существующих реализаций. Jenkins 2 представляет собой существенный переход от более старых, более традиционных версий Jenkins на основе форм. И с таким переходом есть чему поучиться.

Но все это управляемо. В качестве первого шага нам нужно заложить прочный фундамент основ Jenkins 2 (что это такое? каковы важнейшие пункты?), включая новые функции, изменения в рабочей среде и понимание новых концепций, на которых он основан. Вот о чем эта и последующие главы. С некоторым из этого вы, возможно, уже знакомы. И если так, то это здорово. Однако я предлагаю, по крайней мере,

сканировать те разделы, которые выглядят знакомыми. Там может быть что-то новое или достаточно изменившееся, чтобы на него можно было обратить внимание.

В этой главе мы рассмотрим на высоком уровне, что отличает Jenkins 2 и как это будет соответствовать тому, к чему вы привыкли. Мы рассмотрим три ключевые области:

- что такое Jenkins 2 с точки зрения значительных новых функций и возможностей, которые он представляет;
- каковы причины (мотивы и движущие силы) перехода в Jenkins;
- насколько совместим Jenkins 2 с предыдущими версиями, каковы соображения совместимости.

Давайте начнем с того, что отличает Jenkins 2 от традиционных версий.

Что такое Jenkins 2?

В этой книге использование термина «Jenkins 2» не совсем точно. В нашем конкретном контексте это способ упоминания более новых версий Jenkins, которые напрямую включают поддержку концепции «pipelines-as-code» и другие новые функции, такие как Jenkinsfiles, о которых мы будем говорить на протяжении всей книги.

Некоторые из этих функций были доступны для версий Jenkins 1.x в течение некоторого времени через плагины (и, чтобы было ясно, Jenkins 2 получает большую часть своей новой функциональности от основных обновлений существующих плагинов, а также совершенно новые плагины.) Но Jenkins 2 представляет нечто большее, а именно переход к фокусированию на этих особенностях в качестве предпочтительного, основного способа взаимодействия с Jenkins. Вместо заполнения веб-форм для определения заданий Jenkins пользователи теперь могут писать программы, используя DSL Jenkins и Groovy для определения своих конвейеров и выполнения других задач.

Под DSL здесь подразумевается *предметно-ориентированный язык* (domain-specific language), «язык программирования» для Jenkins. DSL основан на Groovy и содержит термины и конструкции, которые инкапсулируют специфичную для Jenkins функциональность. Примером является ключевое слово **node** (узел), которое сообщает Jenkins, что вы будете программно выбирать узел (ранее «ведущее устройство» или «ведомое устройство»), на котором хотите выполнить эту часть вашей программы.

JENKINS И GROOVY

На протяжении долгого времени движок Groovy входил в состав Jenkins. Это использовалось для того, чтобы разрешать продвинутые операции сценариев и обеспечивать доступ/функциональность, недоступную через веб-интерфейс.

DSL является основной частью Jenkins 2. Он служит строительным блоком, который делает возможными другие ключевые функции, с которыми сталкивается пользователь. Давайте посмотрим на некоторые из них, чтобы увидеть, как они отличают Jenkins 2 от «традиционного» Jenkins. Мы быстро рассмотрим новый способ отделить ваш код от Jenkins в файлах Jenkinsfile, более структурированный подход к созданию рабочих процессов с помощью декларативных конвейеров и захватывающий новый визуальный интерфейс Blue Ocean.

Jenkinsfile

В Jenkins 2 определение вашего конвейера теперь может быть отделено от самого Jenkins. В предыдущих версиях Jenkins определения заданий были сохранены в файлах конфигурации в домашнем каталоге Jenkins. Это означало, что Jenkins сам должен был видеть, понимать и изменять определения (если вы не хотите работать с XML напрямую, что было непросто). В Jenkins 2 вы можете написать свое определение конвейера как сценарий DSL в текстовой области в веб-интерфейсе. Тем не менее вы также можете взять DSL-код и сохранить его как текстовый файл с вашим исходным кодом. Это позволяет управлять заданиями Jenkins, используя файл, содержащий код, как любой другой исходный код, в том числе отслеживать историю, видеть различия и т. д.

ПЛАГИН JOBCONFIGHISTORY

Для полноты картины следует упомянуть, что для Jenkins существует плагин JobConfigHistory, который отслеживает историю изменений конфигурации XML с течением времени и позволяет смотреть, что менялось каждый раз. Он доступен на Jenkins wiki.

Имя файла, в котором, как ожидает Jenkins 2, будут сохранены определения ваших заданий/конвейеров, – *Jenkinsfile*. У вас может быть мно-

го файлов Jenkins, каждый из которых отличается от других проектом и веткой, в которой он хранится. Вы можете хранить весь свой код в файле Jenkinsfile, или можете вызывать/извлекать другой внешний код через общие библиотеки. Также доступны DSL-операторы, которые позволяют загружать внешний код в ваш сценарий (подробнее об этом в главе 6).

Jenkinsfile может служить маркировочным файлом. Это означает, что если Jenkins видит его как часть исходного кода вашего проекта, он понимает, что это проект/ветка, которую Jenkins может запустить. Он также косвенно понимает, с каким источником системы управления исходным кодом (SCM) и веткой он должен работать. Затем он может загрузить и выполнить код в файле Jenkinsfile. Если вы знакомы с системой сборки Gradle, то это похоже на идею файла *build.gradle*, используемого данным приложением. Мы еще будем говорить подробнее о файлах Jenkinsfile на протяжении всей книги.

На рис. 1.1 показан пример файла Jenkinsfile в системе контроля версий.

Декларативные конвейеры

В предыдущих воплощениях концепции «pipeline as code» в Jenkins код представлял собой в основном сценарий Groovy с включенными специфическими для Jenkins шагами DSL. Там было очень мало навязанной структуры, а программный поток управлялся конструкторами Groovy. Отчеты об ошибках и проверка были основаны на выполнении программы Groovy, а не на том, что вы пытались сделать с Jenkins.

Эта модель является тем, что мы сейчас называем *сценарными конвейерами*. Однако DSL для конвейера продолжил развиваться.

В сценарных конвейерах DSL поддерживал большое количество различных шагов для выполнения задач, но упускал некоторые ключевые метаданные задач, ориентированных на Jenkins, такие как обработка после сборки, проверка ошибок для структур конвейера и возможность легко отправлять уведомления на основе различных состояний. Многое из этого можно эмулировать с помощью механизмов программирования Groovy, таких как блоки `try-catch-finally`. Но для этого требовалось больше навыков программирования на Groovy в дополнение к ориентированному на Jenkins программированию. Файл Jenkins, показанный на рис. 1.1, является примером сценариев конвейера с обработкой уведомлений `try-catch`.

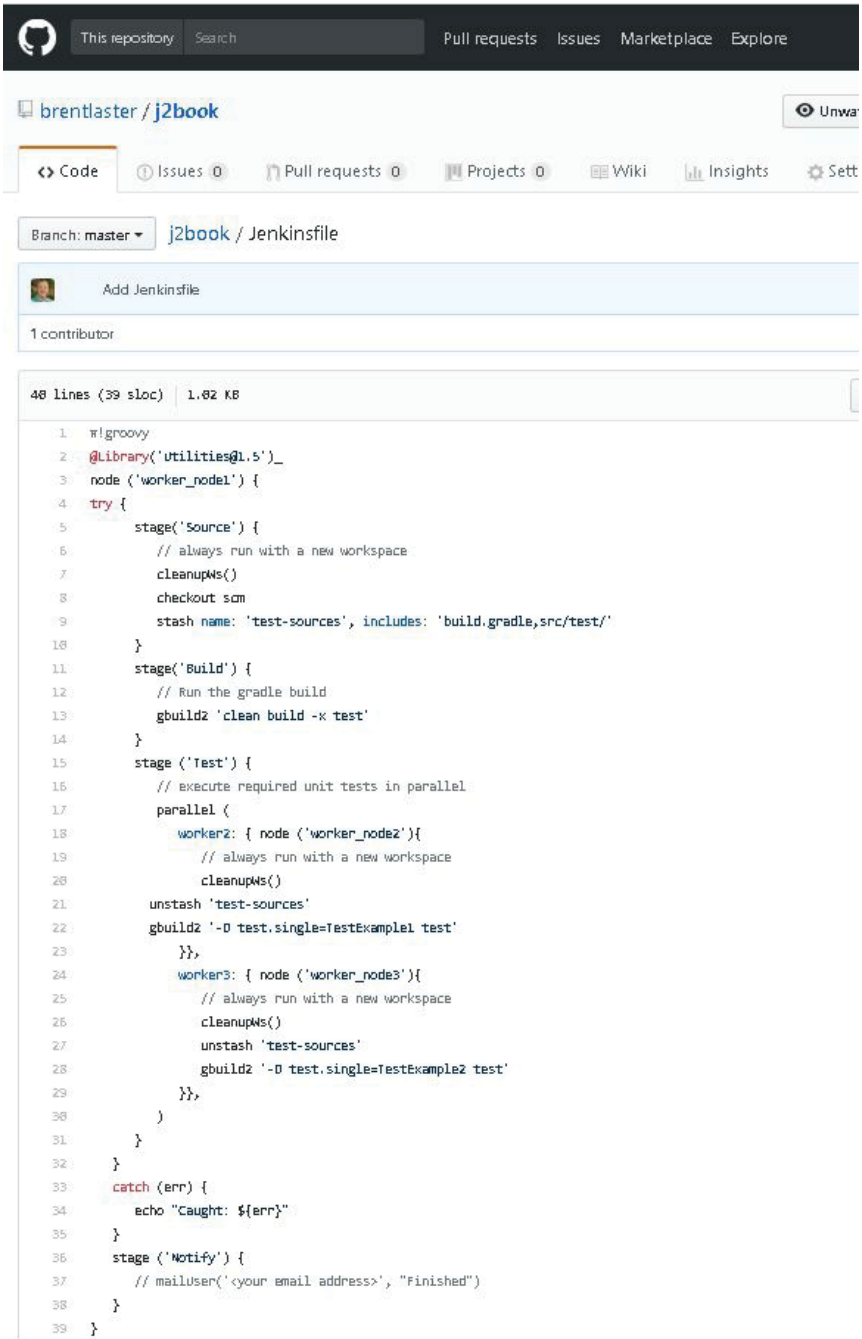

Рис. 1.1. Пример файла Jenkinsfile в системе контроля исходного кода

В 2016 и 2017 годах CloudBees, корпоративная компания, которая является основным участником проекта Jenkins, представила расширенный синтаксис программирования для концепции «pipelines-as-code» под названием *декларативные конвейеры*. Этот синтаксис добавляет ясную, ожидаемую структуру конвейерам, а также улучшенные элементы и конструкции DSL. Результат более тесно напоминает рабочий процесс построения конвейера в веб-интерфейсе (с проектами Freestyle).

Примером здесь является обработка после сборки с уведомлениями, основанными на статусах сборки, которые теперь можно легко определить с помощью встроенного механизма DSL. Это уменьшает необходимость дополнения определения конвейера Groovy-кодом для эмуляции традиционных функций Jenkins.

Более формальная структура декларативных конвейеров обеспечивает более чистую проверку ошибок.

Таким образом, вместо того чтобы сканировать обратные вызовы Groovy при возникновении ошибки, пользователю предоставляется краткое, направленное сообщение об ошибке – в большинстве случаев указывающее непосредственно на проблему. На рис. 1.2 показан фрагмент кода, созданного следующим декларативным конвейером с расширенной проверкой ошибок:

```
pipeline {
     agent any
     stages {
        stae('Source') {
           git branch: 'test', url: 'git@diyvb:repos/gradle-greetings'
           stash name: 'test-sources', includes: 'build.gradle,/src/test'
        }
        stage('Build') {
        }
     }
}
```

Интерфейс Blue Ocean

Структура, которая поставляется с декларативными конвейерами, также служит основой для другого нововведения в Jenkins 2 – Blue Ocean, нового визуального интерфейса Jenkins. В Blue Ocean добавлено графическое представление для каждой стадии конвейера, показывающее индикаторы успеха/неудачи и прогресса и позволяющее щелкать по кнопке доступ к журналам для каждой отдельной части. Blue Ocean

также предоставляет базовый визуальный редактор. На рис. 1.3 показан пример успешного выполнения конвейера с журналами, отображаемыми в Blue Ocean. Глава 9 полностью посвящена новому интерфейсу.


```
Started by user Jenkins Admin
org.codehaus.groovy.control.MultipleCompilationErrorsException: startup failed:
WorkflowScript: 4: Expected a stage @ line 4, column 7.
         stae('Source') {
         ^

WorkflowScript: 4: Stage does not have a name @ line 4, column 7.
         stae('Source') {
         ^

WorkflowScript: 4: Nothing to execute within stage "null" @ line 4, column 7.
         stae('Source') {
         ^

WorkflowScript: 7: Nothing to execute within stage "Build" @ line 7, column 7.
         stage('Build') {
         ^

4 errors
```

Рис. 1.2. Декларативный конвейер с расширенной проверкой ошибок

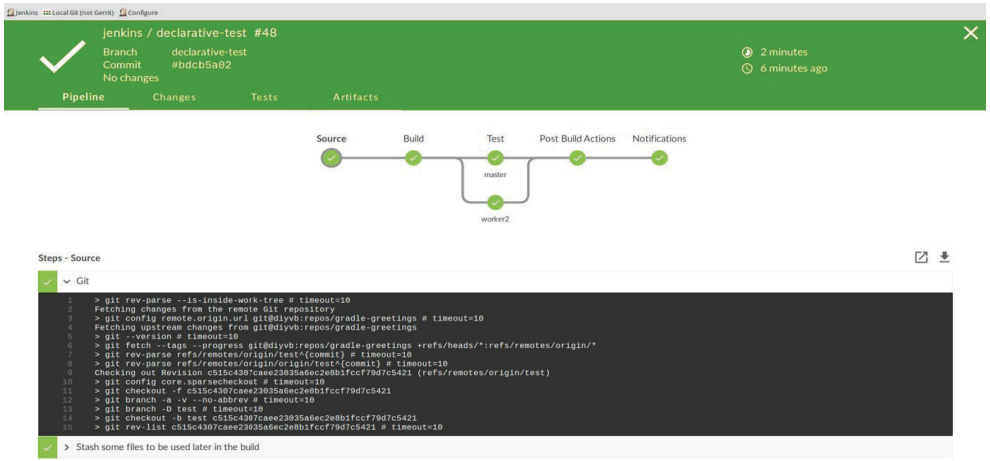

Рис. 1.3. Отображение успешного запуска и проверка журналов с помощью интерфейса Blue Ocean

Новые типы заданий в Jenkins 2

Jenkins 2 поставляется с несколькими новыми типами заданий, в основном разработанными с использованием ключевых функций, таких как pipelines-as-code и файлы Jenkinsfile. Они упрощают автоматизацию создания рабочих мест и конвейеров, а также организацию ваших проектов. Создание каждого нового задания/элемента/проекта начинается одинаково.

НОВЫЕ ТИПЫ ЗАДАНИЙ И ПЛАГИНЫ

Чтобы было ясно, наличие новых типов заданий зависит от наличия необходимых плагинов. Если вы принимаете рекомендуемые плагины во время процесса установки, то получаете типы заданий, которые будут обсуждаться далее.

После установки Jenkins 2 и входа в систему вы можете создавать новые задания, как и раньше. Как показано на рис. 1.4, призыв под баннером **Welcome to Jenkins!** (Добро пожаловать в Jenkins!) предлагает пользователям «создавать новые задания», но пункт меню для этого на самом деле обозначен как **New Item** (Новый элемент). Большинство этих элементов в конечном счете также является своего рода проектом. Для наших целей я буду попеременно использовать термины «задание», «элемент» и «проект» на протяжении всей книги.

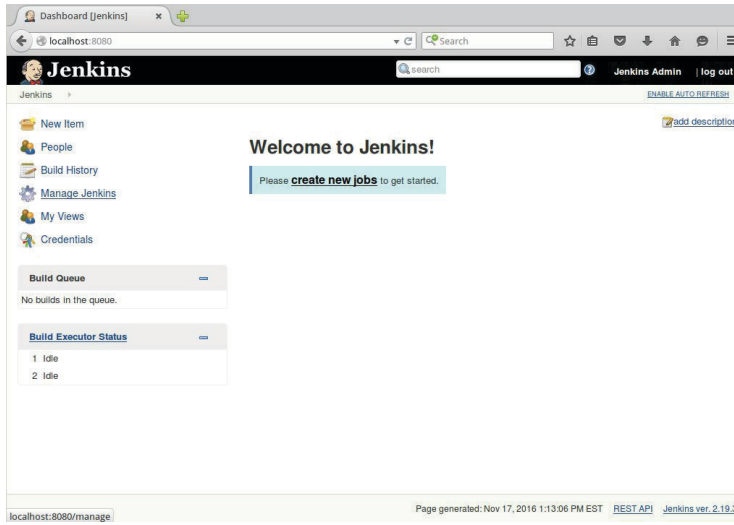

Рис. 1.4. Экран приветствия Jenkins: отправная точка для создания новых заданий, элементов и проектов

Когда вы решите создать новый элемент в Jenkins 2, вам будет представлен экран для выбора типа нового задания (рис. 1.5). Вы заметите знакомые типы, такие как проект Freestyle, а также те, которые вы, возможно, раньше не видели. Я кратко изложу здесь новые типы заданий, а затем объясню каждый из них более подробно в главе 8.

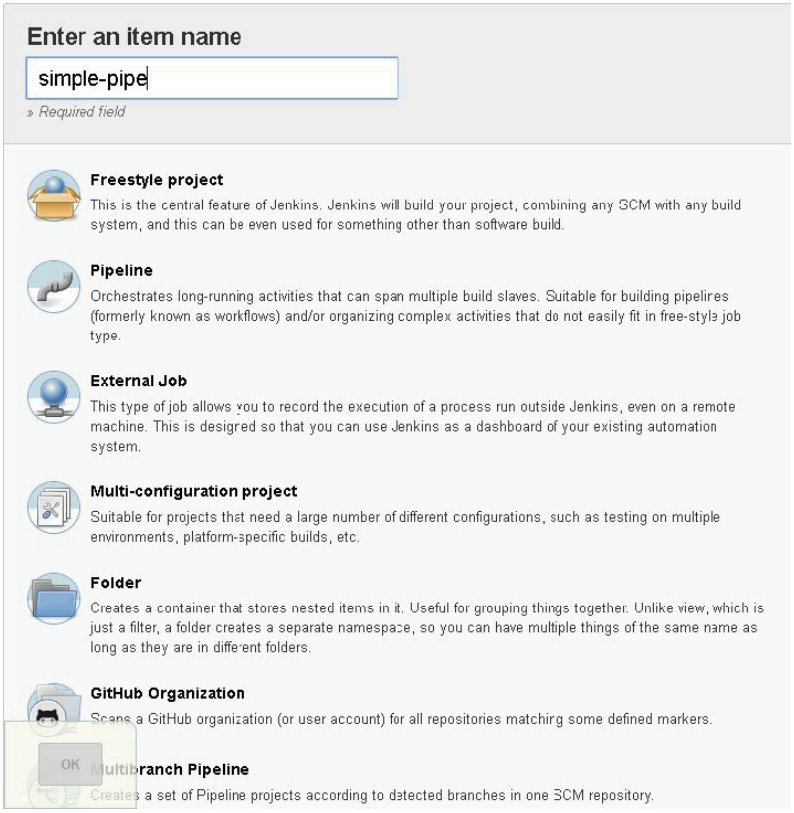

Рис. 1.5. Проекты Jenkins 2

Конвейер

Как следует из названия, тип проекта Pipeline предназначен для создания конвейеров. Это делается путем написания кода в DSL Jenkins. Это основной тип проекта, о котором мы будем говорить на протяжении всей книги.

Как уже отмечалось, конвейеры могут быть написаны либо в «сценарном» синтаксическом стиле, либо в «декларативном» синтаксическом стиле. Конвейеры, созданные в этом типе проекта, также могут быть легко преобразованы в файлы Jenkinsfile.

Папка

Это способ группировать проекты, а не сам тип проекта. Обратите внимание, что это не похоже на традиционные вкладки **Представление** на панели управления Jenkins, которые позволяют фильтровать список проектов. Это скорее похоже на папку в операционной системе. Имя папки становится частью пути проекта.

Организация

Некоторые платформы управления исходным кодом предоставляют механизм для группировки репозиториев в «организации». Интеграции Jenkins позволяют хранить сценарии конвейера Jenkins как файлы Jenkinsfile в репозиториях внутри организации и осуществлять выполнение на их основе.

В настоящее время поддерживаются организации GitHub и Bitbucket, другие запланированы на будущее. Для простоты в этой книге мы будем говорить в основном о проектах организации GitHub в качестве примера.

При условии достаточного доступа Jenkins может автоматически настроить вебхук организации (уведомление от веб-сайта) на стороне хостинга, который будет уведомлять ваш экземпляр Jenkins о любых изменениях в репозитории. Когда Jenkins получает уведомление, он обнаруживает файл Jenkinsfile в качестве маркера в хранилище и выполняет команды в файле для запуска конвейера.

Разветвленный конвейер

В этом типе проекта Jenkins снова использует файл Jenkinsfile в качестве маркера. Если в проекте создается новая ветка с файлом Jenkins, Jenkins автоматически создаст новый проект в Jenkins только для этой ветки. Этот проект может быть применен к любому репозиторию Git или Subversion.

Мы подробнее рассмотрим каждый из этих новых типов проектов в главе 8. Однако стоит также отметить, что Jenkins по-прежнему поддерживает традиционную рабочую лошадку – проекты Freestyle. Вы все еще можете создавать задания, используя формы веб-интерфейса, и выполнять их, как раньше. Но, безусловно, акцент в Jenkins 2 сделан на заданиях для конвейеров.

Легко видно, что Jenkins 2 существенно отличается от традиционной модели Jenkins. Таким образом, стоит потратить несколько минут, чтобы обсудить причины изменений.

Причины перехода

Можно утверждать, что в течение многих лет Jenkins был наиболее плодотворным инструментом управления рабочими процессами и конвейерами. Так что же привело к необходимости совершить переход в сторону Jenkins 2? Давайте рассмотрим несколько потенциальных причин, как внешних, так и внутренних по отношению к Jenkins.

Движение DevOps

Идеи, стоящие за непрерывной интеграцией, непрерывной доставкой и непрерывным развертыванием, существуют уже несколько лет. Но с самого начала они были скорее конечной целью, чем отправной точкой. С повышенным вниманием к DevOps в последние годы пользователи и компании стали ожидать, что инструментарий будет помогать им в реализации DevOps и непрерывных практик из коробки (или, по крайней мере, не усложнять).

Учитывая его место в пространстве автоматизации рабочих процессов, был чем-то ожидаемым (и, возможно, необходимым) тот факт, что Jenkins будет развивать свои возможности для поддержки этих отраслевых драйверов.

Сборка конвейеров

Создание какого-либо одного задания в интерфейсе Jenkins Freestyle не обязательно было проблематичным. Но попытка собрать несколько заданий в конвейер непрерывной доставки программного обеспечения, который может перевести код из фиксации в развертывание, часто могла быть сложной задачей. Основные функции Jenkins позволяли запускать определенное задание после завершения другого, но обмен данными между заданиями, такими как рабочие области, параметры и т. д., нередко был проблематичным, или для этого требовались специальные плагины либо приемы.

Возобновляемость

Ключевая часть функциональности Jenkins 2 зависит от способности конвейеров быть долговечными – это означает, что задания продолжают выполняться на агентах или выбираются с того места, где они остановились, после перезапуска ведущего узла. Фактически одно из требований совместимости плагина с Jenkins 2 – возможность сериализации состояний, чтобы их можно было восстановить в случае переза-

пуска ведущего устройства. С предыдущими версиями Jenkins было не так; пользователей и процессы часто оставляли там, где им нужно было либо просматривать журналы, чтобы выяснить, где что осталось, либо просто начинать процесс заново с самого начала.

Конфигурируемость

Поскольку пользователи были в значительной степени ограничены веб-интерфейсом, для работы с устаревшими версиями Jenkins обычно требовалось найти правильное место на экране, определить кнопки и поля и постараться не сделать опечатку при вводе данных. Для изменения рабочего процесса (например, изменение порядка шагов в задании или изменение порядка выполнения заданий) могло потребоваться многократное взаимодействие кликов, перетаскиваний и ввода текста, в отличие от более простых обновлений, доступных в интерфейсе текстового редактора. В некоторых случаях, когда элементы графического интерфейса пользователя были предоставлены для взаимодействия с инструментарием, способы отправки определенных команд в инструментарий через интерфейс Jenkins были недоступны. Веб-формы, принятые в Jenkins, хорошо подходят для простого, структурированного выбора, но они не настолько удобны, когда речь идет об итеративном управлении потоками или управлении потоками на основе решений.

Совместное использование рабочих пространств

Традиционно в Jenkins у каждого задания было свое рабочее пространство для получения исходного кода, выполнения сборок или любой другой необходимой обработки. Это хорошо работало для отдельных заданий, изолировало их среды и предотвращало перезапись данных. Однако при объединении заданий это могло привести к неэффективному процессу, который сложно преодолеть. Например, если необходимо выполнить несколько заданий в конвейере, обрабатывая собранные артефакты, необходимость каждый раз повторно собирать артефакты была крайне неэффективной. Хранение и извлечение артефактов в хранилище между выполнением заданий требовало добавления нескольких шагов и настройки каждого задания. Более эффективно было бы разделить рабочее пространство между заданиями, но сделать это в устаревших версиях Jenkins было нелегко. Скорее, пользователь должен был определить пользовательские рабочие пространства и использовать параметры, которые указывали на рабочее пространство, или применять специализированный плагин, чтобы заставить его работать.

Специализированные знания

Как показало предыдущее обсуждение общих рабочих областей, пользователям часто нужно было знать «правильные приемы», чтобы реализовать в устаревших версиях Jenkins то, что они могли бы легко сделать в обычной программе или сценарии (передача данных, управление потоками, внешние вызовы и т. д.).

Доступ к логике

Устаревшие версии Jenkins обычно использовали веб-формы для ввода данных и сохраняли их в файлах конфигурации XML в своем домашнем каталоге. При такой реализации не было простого способа взглянуть на логику выполнения нескольких заданий. Для пользователей, незнакомых с ним, понимание настройки Jenkins и определений заданий может потребовать небольшой прокрутки экранов, просмотра значений в формах, перелистывания назад и вперед между глобальными конфигурациями и т. д. Это усложнило поддержку, сотрудничество между несколькими пользователями и понимание многопрофильных конвейеров, особенно в случае существенных изменений, проверки или отладки, которую необходимо было сделать.

Управление источником конвейера

Как подчеркивалось в предыдущем разделе, «источником» заданий старых версий Jenkins был XML-файл. Его было не только сложно прочитать, но и трудно изменить и исправить, не заходя в веб-интерфейс. Для того чтобы существовать в том же месте, что и исходный код, не была разработана конфигурация. Конфигурация и исходный код были двумя отдельными объектами, управляемыми двумя различными способами.

Следствием этого было отсутствие возможности проверки. Хотя были плагины, помогающие отслеживать изменения во времени, это было не так удобно, как отслеживание простых изменений исходного файла, и требовалось, чтобы приложение Jenkins само могло отслеживать изменения в заданиях.

Конкуренция

Еще один фактор, который, несомненно, вступил в игру, заключается в том, что вокруг концепции «pipelines as code» возникли другие приложения. Существуют различные примеры, такие как Pivotal's Concourse, который использует контейнеризацию для выполнения заданий и позволяет описывать конвейеры в файлах YAML.