# Глава 9 Контроль версий в SAP Web IDE Использование Git

*В этой главе мы познакомим вас с основными концепциями использования Git в качестве системы управления исходным кодом (SCM) и как работать с репозиториями Git с помощью SAP Web IDE.*

Git - это система контроля версий (VCS), наиболее часто используемая для разработки SAPUI5. В этой главе мы рассмотрим наиболее важные рабочие процессы и команды Git, используемые для разработки SAPUI5 в SAP Web IDE. Вы также узнаете, как связать репозиторий Git с проектами SAP Web IDE, и о концепциях работы с различными ветками репозитория Git.

## 9.1 Введение в Git

Система управления исходным кодом (SCM) помогает вам отслеживать все изменения в файлах вашего проекта. SAP Web IDE имеет встроенную поддержку Git, поэтому мы будем использовать Git для наших проектов. Однако вместо того чтобы создавать отдельный репозиторий Git в системе, мы будем создавать репозитории проектов с помощью GitHub (популярный сервис, расширяющий Git) и затем свяжем эти репозитории со своими проектами в SAP Web IDE. В проектной среде использование SCM полезно, когда несколько разработчиков работают над одним проектом. Добавив SCM в проект, вы можете отслеживать, кто изменил файл в проекте, сравнивать исходный код с его ранними версиями, отменять изменения или даже отслеживать, кто сделал последний коммит, который создал ошибку или проблему в проекте, и многое другое. Поскольку мы используем Git в качестве SCM для SAP Web IDE, давайте начнем с основ.

## 9.2 Основы Git

Git будет служить в качестве центрального репозитория, который будет содержать самую последнюю версию проекта. Все разработчики, работающие над проектом, будут клонировать центральный репозиторий в свою SAP Web IDE и работать над копией кода в локальном репозитории. Разработчики могут отправить код обратно в центральный репозиторий и зафиксировать свои изменения. Эти изменения могут быть синхронизированы другими разработчиками в их локальный репозиторий, где они могут продолжить работу над своими частями независимо друг от друга, как показано на рисунке 9.1.

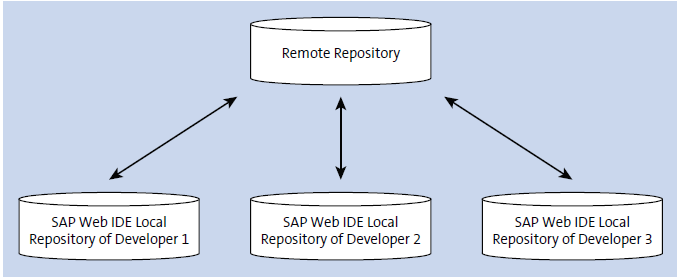


Рисунок 9.1 Концепция общего репозитория для проектов, использующих Git

В отличие от других VCS, Git отслеживает изменения в виде снимков. Другими словами, Git не только отслеживает новые изменения файлов в каждой версии, он хранит снимок состояния всего проекта каждый раз, когда новое изменение вносится в репозиторий. На рисунке 9.2 показано, как Git хранит снимки; прямоугольники с пунктирными линиями обозначают файлы, которые не изменились в данной версии, поэтому Git ссылается на файл в его предыдущей версии.

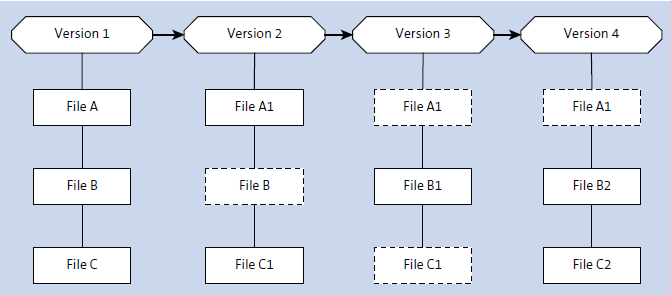


Рисунок 9.2 Подход Git Snapshots для статуса проекта

Теперь, когда у вас есть базовое понимание Git, давайте начнем создавать репозиторий и связывать его с нашими ранее созданными проектами. Альтернативным подходом было бы создать Git-репозитории до создания реальных приложений в предыдущих главах, но для наших примеров мы решили связать приложения в SAP Web IDE с Git-репозиторием после завершения работы над приложениями. Вы можете использовать любой из этих подходов. Если вы будете пользователем, устанавливающим начальную версию приложения, то вы можете создать репозитории позже. Если несколько разработчиков с самого начала проекта, то вам нужно будет сначала создать начальный удаленный репозиторий, затем клонировать пустые репозитории в SAP Web IDE для каждого разработчика проекта и начать работу над проектом. На рисунке 9.3 показан высокоуровневый обзор шагов по настройке Git-репозитория для ваших проектов.

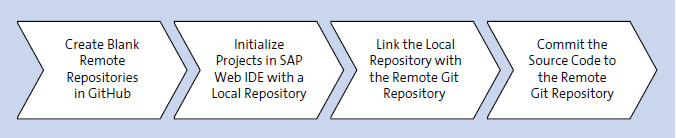


Рисунок 9.3 План настройки репозитория проекта Git

### 9.2.1 Создание начальных репозиториев проекта с помощью GitHub

Сначала вам нужно создать учетную запись на <https://github.com>. На главной странице GitHub нажмите на кнопку **New Repository** в левой части страницы, как показано на рис. 9.4, чтобы создать начальный пустой репозиторий для проекта.

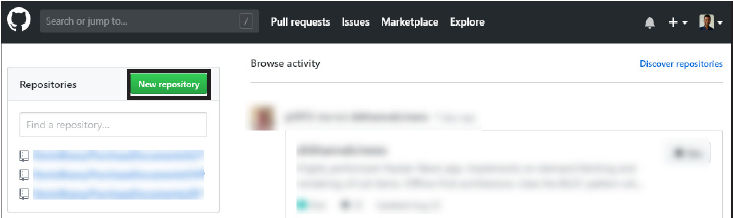


Рисунок 9.4 Создание нового репозитория в GitHub

На следующем экране задайте хранилищу имя (для нашего примера введите "PurchaseDocumentsALPRepo") и описание. Установите флажок **Initialize this repository with a README** и нажмите на **Create repository**, как показано на рисунке 9.5.

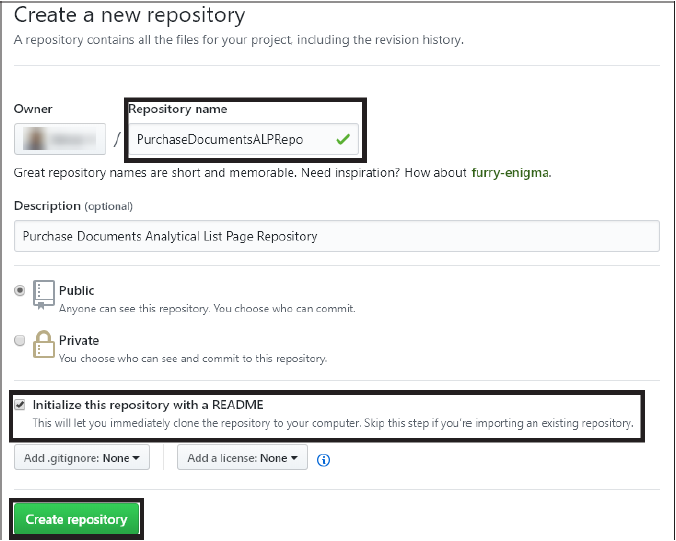


Рисунок 9.5 Настройка и инициализация нового репозитория

На следующем экране вы увидите, что репозиторий был настроен. Теперь нажмите на кнопку **Clone or** **download**, а затем нажмите на значок **Copy**, чтобы скопировать URL, как показано на рисунке 9.6.

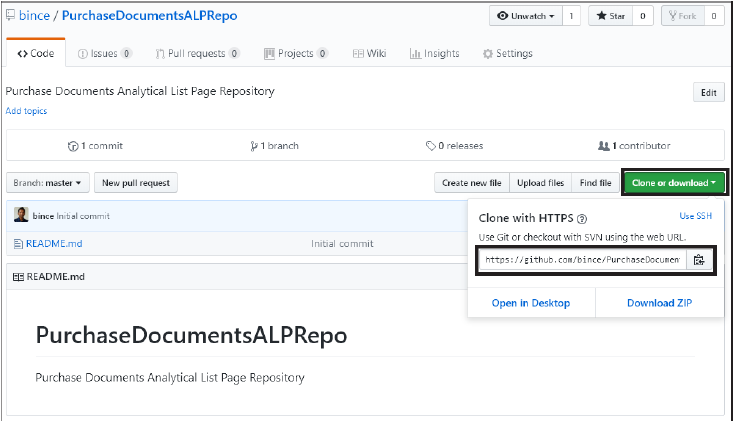


Рисунок 9.6 Копирование URL-адреса репозитория Git

### 9.2.2 Инициализация локального репозитория для проектов в SAP Web IDE

После настройки начального удаленного репозитория откройте SAP Web IDE и щелкните правой кнопкой мыши на проекте. Затем выберите **Git – Initialize Local Repository**, как показано на рисунке 9.7. Будет создан локальный Git-репозиторий с основной веткой по умолчанию для проекта. Этот репозиторий будет действовать как основной репозиторий для дальнейших разработок, пока вы не переместите изменения в удаленный репозиторий.

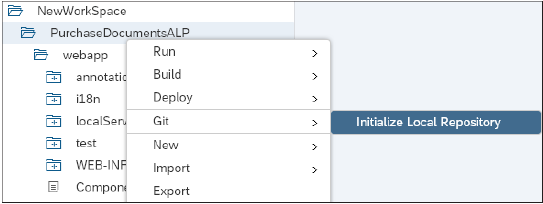


Рисунок 9.7 Инициализация локального репозитория для проекта в SAP Web IDE

### 9.2.3 Связывание локального репозитория с удаленным репозиторием в GitHub

Теперь давайте свяжем локальный репозиторий с удаленным репозиторием, который мы создали с помощью GitHub. Щелкните правой кнопкой мыши на проекте и перейдите к **Git – Set Remote**, как показано на рисунке 9.8.

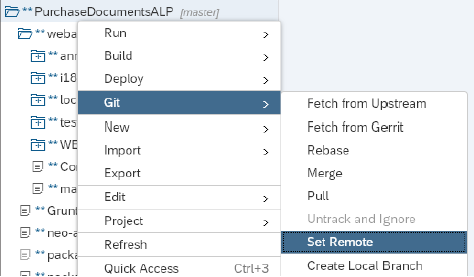


Рисунок 9.8 Связывание локального репозитория с удаленным Git-репозиторием

В следующем всплывающем окне введите URL удаленного репозитория, который мы скопировали из проекта GitHub ранее, а затем нажмите OK, как показано на рисунке 9.9.

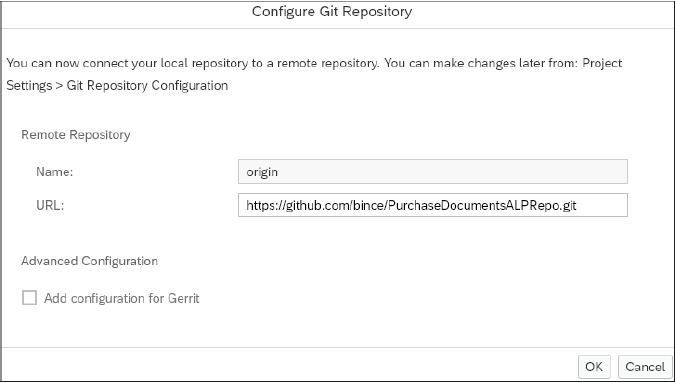


Рисунок 9.9 Настройка Git-репозитория с помощью Git URL в SAP Web IDE

В следующем окне, как показано на рисунке 9.10, нажмите OK для подтверждения изменений, полученных из ветки удаленного репозитория.

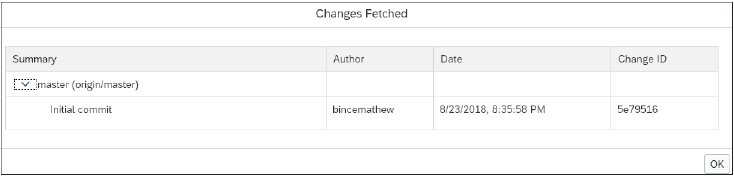


Рисунок 9.10 Изменения, извлекаемые из удаленной ветви в Git

### 9.2.4 Отправка кода в репозиторий (Stage, Commit, Push)

Перед фиксацией изменений в репозитории Git вам необходимо понять некоторые основные понятия в SAP Web IDE, связанные с интеграцией Git. Давайте вкратце рассмотрим некоторые части SAP Web IDE, которые понадобятся вам далее в этой главе для работы с нашими репозиториями.

Как показано на рисунке 9.11, когда вы нажимаете на значок **Git Pane** (Панель Git) в правой части проекта SAP Web IDE, появится множество значков, связанных с Git.

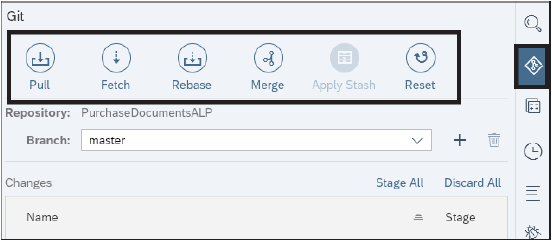


Рисунок 9.11 Панель Git в SAP Web IDE

Давайте вкратце рассмотрим каждый из этих значков:

#### Pull

Являясь комбинацией действий fetch и rebase Git, этот значок извлекает все изменения из удаленной ветки репозитория и объединяет эти изменения с локальной веткой в SAP Web IDE.

#### Fetch

Эта опция извлекает изменения, обнаруженные в удаленном репозитории, и перечисляет их в списке изменения для вас; однако эти изменения не будут применены к вашей локальной ветке.

#### Rebase

В дополнение к remote/master ветви, вам может потребоваться создать дополнительные локальные ветви в зависимости от требований проекта, что часто приводит к конфликтующему коду в разных ветках. В этой ситуации может быть полезна опция Rebase, поскольку эта опция заставит Git временно отменить изменения в локальной ветке, обновить код из удаленной ветки, а затем попытаться применить локальные изменения снова в правильном порядке. Эта опция поможет, когда ваши локальные изменения находятся в конфликте с удаленным репозиторием или когда вы хотите, чтобы новое изменение в удаленной ветке было добавлено в вашу локальную ветку без потери локальных изменений в проекте. Rebase также перепишет историю коммитов проекта для получения более чистой истории проекта, как показано на рисунке 9.12, где различные коммиты в удаленной и локальной ветках представлены квадратами и треугольниками соответственно.

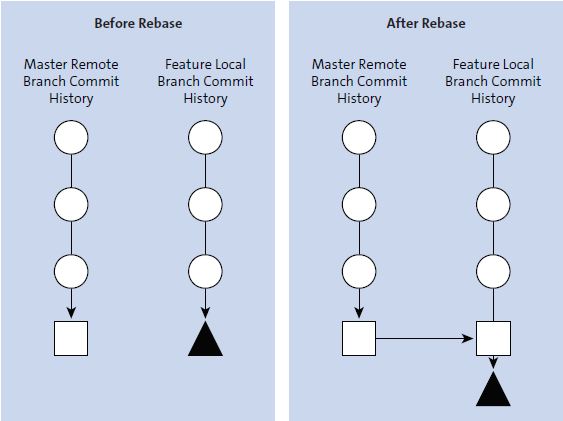


Рисунок 9.12 Концепция Git Rebase

#### Merge

Эта опция работает аналогично опции Rebase, но немного по-другому. Допустим, вам нужно внести некоторые новые изменения в удаленную ветку из локального репозитория для новой функции. Например, вы могли бы создать новую ветку функций в локальной SAP Web IDE для работы над этой функцией, и вам нужно, чтобы некоторые изменения из master/remote ветки для включения в локальную ветку. Опция Merge создает коммит слияния с вашей локальной веткой, объединяя историю обеих веток вместе и тем самым объединяя историю коммитов обеих веток, как показано на рисунке 9.13.

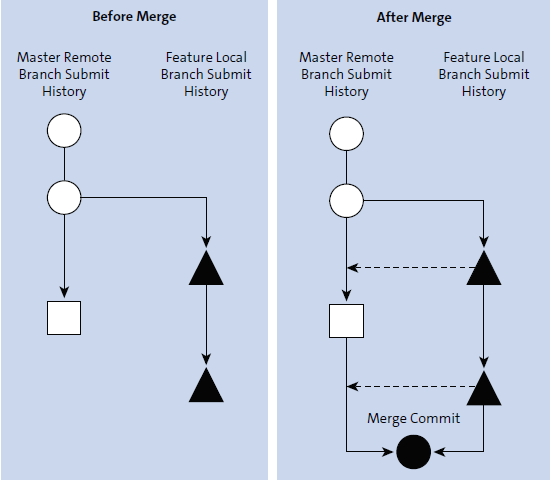


Рисунок 9.13 Концепция слияния Git

#### Apply Stash

Эта опция полезна, когда вы работаете над файлом в локальной ветке и решаете переключиться на другую локальную ветку для работы над отдельной функцией вашего проекта, но вы не хотите фиксировать текущие незавершенные изменения обратно в удаленную ветку. Вместо того чтобы отправлять изменения в удаленную ветвь, разработчик может использовать опцию Stash для временного хранения незафиксированных изменений, а затем переключиться на другую локальную ветвь без потери этих изменений. После того, как разработчик закончит работу в другой локальной ветке и переключится обратно на локальную ветку с нефиксированного кода, код можно извлечь с помощью кнопки Apply Stash на панели панели Git.

#### Reset

Как следует из названия, эта опция сбрасывает все незафиксированные изменения из локальной ветви в SAP Web IDE. Сброс имеет две разновидности: смешанный (mixed) сброс или жесткий (hard) сброс. Смешанный сброс сбрасывает ваши файлы из области постановки и возвращает ваш локальный репозиторий к исходной истории коммитов удаленной ветки, но ваши локальные изменения по-прежнему доступны в файлах. Жесткий сброс сбрасывает историю фиксации, удаляет файлы из области постановки и удаляет все локальные изменения, которые вы сделанные в вашем рабочем пространстве. По сути, ваше рабочее пространство снова синхронизируется с последним зафиксированным состоянием соответствующей удаленной ветки.

Далее вам необходимо ознакомиться с областью Staging панели Git в SAP

Web IDE, как показано на рисунке 9.14. К ним относятся:

1 Область постановки

2 Опции стейджинга

3 Нефиксированные файлы в локальной ветке

Область постановки (staging area) - это место, где локальные изменения в проекте отслеживаются SAP Web IDE, и у пользователя есть возможность зафиксировать файлы в удаленном хранилище из или отменить изменения и вернуться к исходному коду. Посредством установки или снятием флажков для каждого файла, можно выбрать, какие файлы будут зафиксированы, а какие нет.

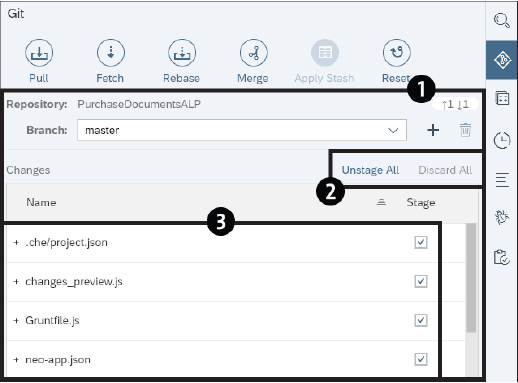


Рисунок 9.14 Область постановки в SAP Web IDE Full-Stack

Чуть ниже списка отслеживаемых изменений в области постановки доступна текстовая область для ввода комментарий для описания фиксации, а также некоторые опции для фиксации, фиксации и переноса, переноса или хранения изменений, как показано на рисунке 9.15. Если вы уже ввели изменения, и вам нужно внести новые изменения в предыдущее введение, то перед нажатием кнопки Push необходимо установить флажок Amend.

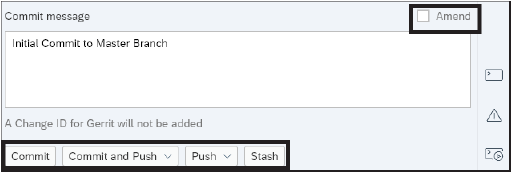


Рисунок 9.15 Опции Staging Area Amend, Commit, Commit и Push, Push и Stash

Теперь, когда мы рассмотрели концепции, связанные с панелью Git, давайте вернемся к шагу где мы связали локальную ветку с удаленной веткой на GitHub. После установления связи с удаленной веткой, нажмите на кнопку Pull (1), как показано на рисунке 9.16, чтобы синхронизировать удаленную ветвь с локальной ветвью (в удаленной ветви не было никакого кода кроме файла Readme, поэтому она не должна создавать конфликтов).

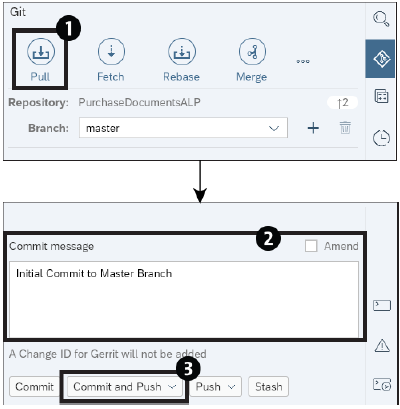


Рисунок 9.16 Выполнение начальной фиксации в основной ветви

После этого шага вы можете отправить весь код в вашей локальной ветке проекта обратно в удаленную ветвь и сделать первый коммит. Введите сообщение о фиксации (2), как показано на рисунке 9.16, а затем нажмите на кнопку Commit and Push (3). Весь код из вашей локальной ветки будет передан в remote ветку. Когда вы нажмете кнопку Commit and Push, обратите внимание, что SAP Web IDE предложит вам два варианта ветви для фиксации, как показано на рисунке 9.17. Опция origin/master указывает на исходную remote мастер-ветвь вашей локальной ветви, а опция Remote Branch дает вам список доступных remote ветвей для выбора. Для нашего сценария выберите из списка origin/master, и SAP Web IDE создаст эту ветвь в remote репозитории.

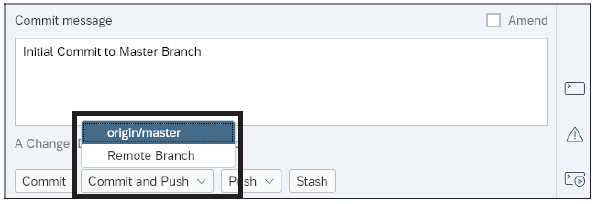


Рисунок 9.17 SAP Web IDE Создание нового remote репозитория для первоначального коммита

Появится всплывающее окно, в котором вы введете свои учетные данные GitHub, чтобы сделать фиксацию. Введите данные вашей учетной записи GitHub и нажмите на кнопку OK, как показано на рисунке 9.18.

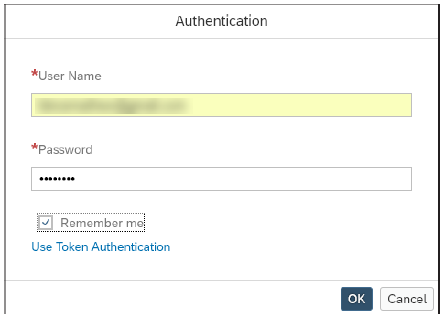


Рисунок 9.18 Ввод учетных данных аккаунта GitHub для фиксации кода

Теперь, если вы войдете в свою учетную запись GitHub, вы должны увидеть весь ваш код, зафиксированный из SAP Web IDE в локальном репозитории, как показано на рисунке 9.19.

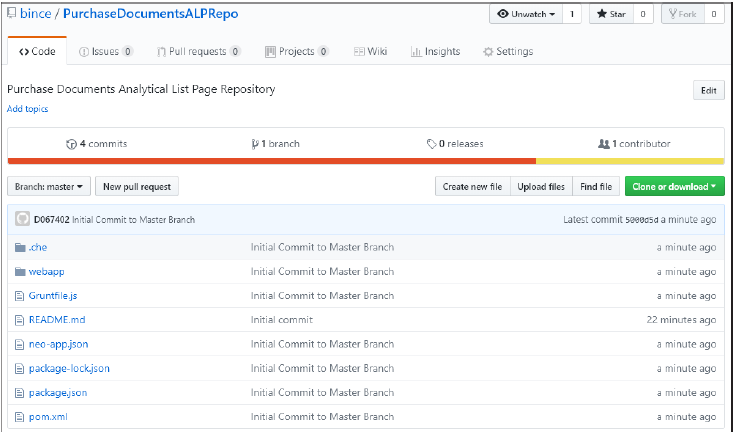


Рисунок 9.19 Зафиксированный код теперь виден на GitHub

### 9.2.5 Клонирование проекта в SAP Web IDE

Чтобы изучить опции Fetch, Merge и Pull с точки зрения другого разработчика, работающего над тем же проектом, давайте клонируем наш новый проект в SAP Web IDE. работающего над тем же проектом в SAP Web IDE, давайте клонируем нашу только что созданную удаленную ветку с GitHub в SAP Web IDE. Чтобы воспроизвести сценарий, в котором два разработчика работают над одним проектом, этот дополнительный клонированный проект в SAP Web IDE будет выступать в качестве второго разработчика. Затем мы отправим код из оригинального проекта (в котором мы инициализировали локальный репозиторий, а затем связали его с удаленным репозиторием GitHub) и retrieve/merge код в клонированном проекте в той же SAP Web IDE для имитации сценария.

Чтобы клонировать проекты из Git, перейдите по ссылке File Git Clone Repository, как показано на рисунке 9.20.

Во всплывающем окне введите URL-адрес репозитория Git (тот же URL-адрес, который мы скопировали ранее в этой главе при создании начального репозитория в GitHub) и нажмите кнопку Clone, как показано на рисунке 9.21.

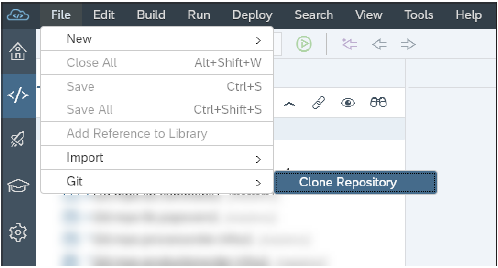


Рисунок 9.20 Клонирование репозитория Git в SAP Web IDE

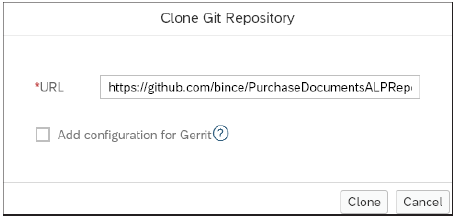


Рисунок 9.21 Ввод URL репозитория Git для клонирования проекта

Теперь давайте создадим новую локальную ветку в SAP Web IDE. Нажмите на значок +, как показано на рисунке 9.22.

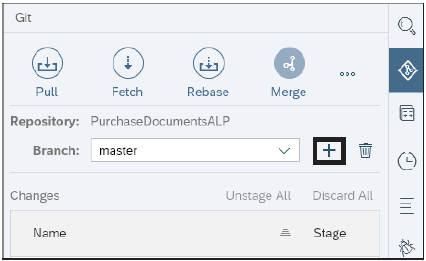


Рисунок 9.22 Добавление новой локальной ветки

Давайте назовем новую ветвь "ALPFeatureBranch", а исходную ветвь "master", как показано на рисунке 9.23, потому что нам нужна копия основной ветви в новую локальную ветвь. Нажмите OK.

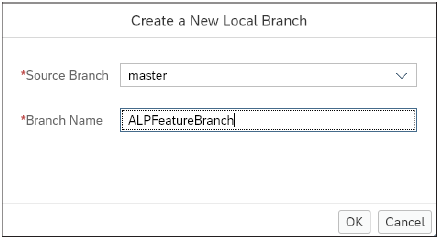


Рисунок 9.23 Создание новой локальной ветви

Теперь новая локальная ветвь создана, и вы можете видеть, что новая ветвь выбрана в панели Git, как показано на рисунке 9.24.

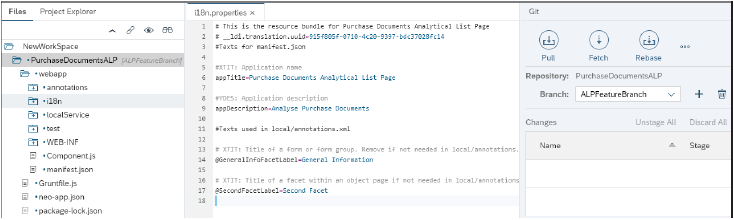


Рисунок 9.24 Новая локальная ветвь, добавленная в клонированный проект в SAP Web IDE

Далее давайте откроем проект, который имеет только одну локальную ветвь (мастер-ветвь) и перенесем некоторые изменения в remote ветвь. Например, допустим, один из наших разработчиков внес несколько изменений в файл i18n и зафиксировал изменения в мастер-ветви.

Как показано на рисунке 9.25, файлы, которые были изменены, появятся в области staging.

К ним относятся:

1 Файлы, выбранные для фиксации из области постановки на хранение

2 Файлы, исключенные из области постановки для фиксации

Разработчик может выбрать или отменить выбор файлов соответствующим образом, а затем сделать коммит.

Обратите внимание, что разработчик выбрал файлы project.json и i18n.properties поэтапно. Файл extended\_runnable\_file.html не был выбран и поэтому не будет перенесен в remote ветвь. Когда вы закоммитите изменения в remote репозитории, код будет обновлен в мастер-ветке, на которую SAP Web IDE сделала первоначальную фиксацию.

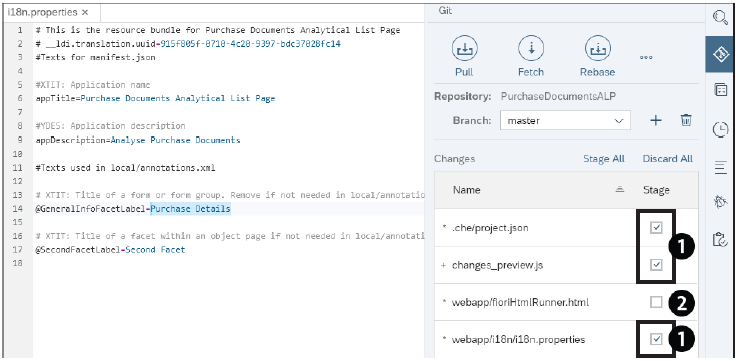


Рисунок 9.25 Область постановки в SAP Web IDE для панели Git

### 9.2.6 Получение кода из remote ветки (Fetch, Merge, Pull)

Теперь давайте извлечем новые изменения из удаленной ветки, которые были перенесены из другого проекта в новую локальную ветку, а затем объединим изменения из удаленной ветки в новую локальную ветку. В данном случае, однако, второй разработчик также редактировал тот же файл i18n.properties. Но сначала разработчику необходимо проверить, были ли новые изменения недавно добавлены в удаленную ветку другим членом команды. Поэтому разработчик нажимает на значок Fetch, как показано на рисунке 9.26.

Действие Fetch обнаруживает изменения в удаленной ветке и перечисляет изменения во всплывающем окне, как показано на рисунке 9.26. всплывающем окне, как показано на рисунке 9.27.

Теперь разработчик пытается выполнить действие Merge, нажав на кнопку Merge, как показано на рисунке 9.28, который попытается объединить код из основной ветки в новую локальную ветвь.

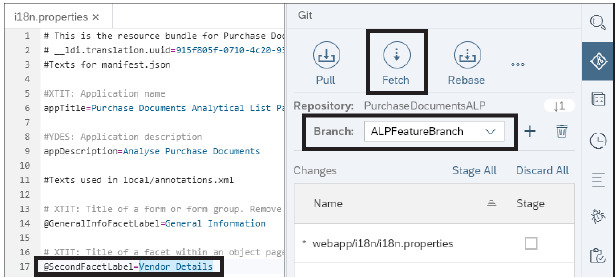


Рисунок 9.26 Инициирование выборки для поиска изменений в удаленной ветви

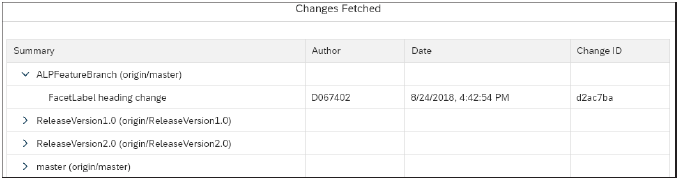


Рисунок 9.27 Действие Fetch при обнаружении изменений из удаленной ветки

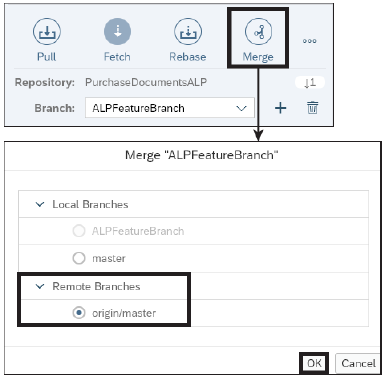


Рисунок 9.28 Слияние новой локальной ветви с удаленной ветвью

Но запрос Merge будет неудачным, как показано на рисунке 9.29, потому что изменения существуют в том же файле. (Если бы в этом новом локальном ответвлении не было изменений в том же файле, то запрос запрос Merge/Rebase сработал бы просто отлично.) Теперь разработчик должен либо отбросить изменения и затем попытаться выполнить слияние, либо использовать опцию Stash. В данном случае, поскольку изменения были незначительными, проблем не возникло. Но давайте предположим, что изменения намного большие, и разработчик просто не захотел делать резервную копию кодов вручную в блокнот, а затем пытаться выполнить запрос Merge после отбрасывания изменений в локальном репозитории. В этом случае разработчикам особенно пригодится функция Stash в Git.

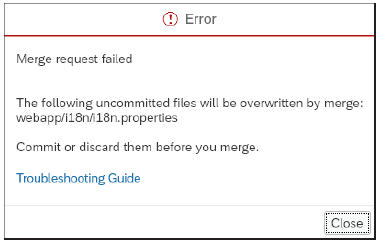


Рисунок 9.29 Отказ запроса на слияние из-за конфликтов файлов

В нижней части области хранения обратите внимание, что кнопка Stash теперь активна. Нажмите на кнопку Stash, как показано на рисунке 9.30. Как упоминалось ранее в разд. 9.2.4, Stash служит в качестве временного хранилища для ваших незафиксированных кодов, которое позволяет извлечь эти изменения позже, чтобы вы могли переключаться между ветвями без потери изменений.

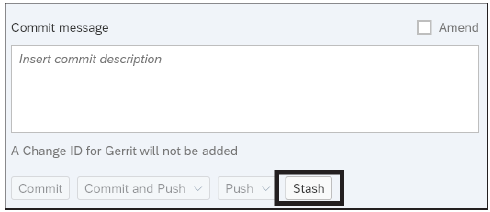


Рисунок 9.30 Опция Stash в панели Git

В следующем окне вы должны дать описание спрятанного кода, чтобы его можно было идентифицировать при попытке извлечения. Введите необходимое описание (в данном случае "Резервное копирование изменений i18n-файла") и нажмите на кнопку Stash, как показано на рис. 9.31.

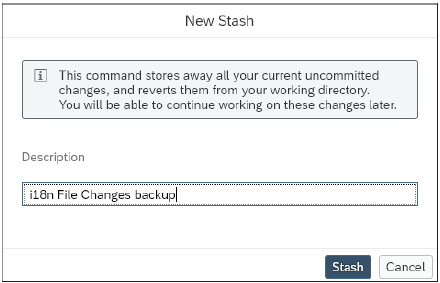


Рисунок 9.31 Именование нового файла для хранения

После того, как локальные изменения были сохранены, разработчик может инициировать действие Merge (1), как показано на рисунке 9.32. На этот раз действие будет выполнено успешно, без каких-либо конфликтов. Новые изменения из основной ветви будут видны и в новой локальной ветви. Теперь разработчик может вернуть спрятанные коды обратно в проект. Для этого нажмите на кнопку Apply Stash (2) в панели Git, как показано на рисунке 9.32. Коды, которые были слиты из основной ветки с новой локальной веткой после нажатия на кнопку Merge (3).

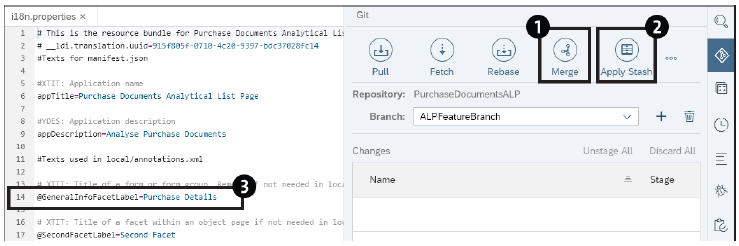


Рисунок 9.32 Слияние и применение Stash в панели Git

Появится новое окно со списком сохраненных stash файлов. Выберите файл Stash; нажмите на второй радиокнопке (разработчик может выбрать, сохранить ли сохраненные файлы для дальнейшего использования или отбросить их); и нажмите кнопку “Продолжить”, как показано на рисунке 9.33.

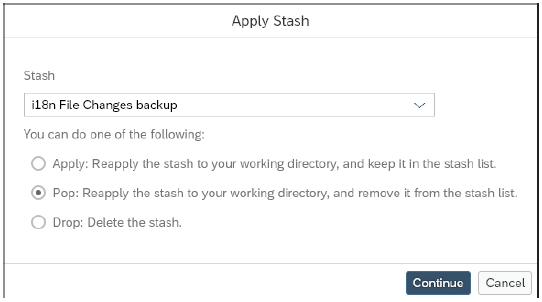


Рисунок 9.33 Выбор сохраненного тайника во всплывающем окне

Теперь ранее не зафиксированные коды снова появятся в файлах, как показано на рисунке 9.34. Разработчик может зафиксировать эти изменения обратно в основную ветвь после того, как разработка в локальной ветви будет завершена.

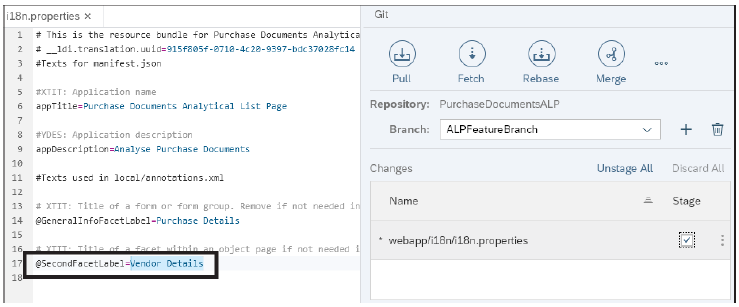


Рисунок 9.34 Спрятанные коды теперь восстановлены в локальной ветви

Предположим, что эти новые изменения в локальной ветви были завершены вторым разработчиком, и он вернул их в основную ветвь. Другие члены команды теперь могут получить последний код из центрального репозитория, нажав на кнопку Pull на панели Git, и все локальные ветки будут синхронизированы с удаленной веткой, как показано на рисунке 9.35.

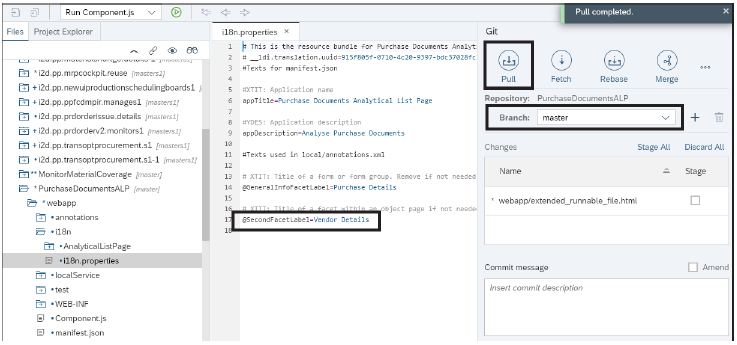


Рисунок 9.35 Инициирование Pull Request для получения последнего кода от удаленной основной ветви (Remote Master Branch)

### 9.2.7 Работа с ветвями

Когда ваш проект разработки должен предоставить несколько версий продукта со временем, разделение удаленных репозиториев на различные итерации, основанные на функциях или версиях продукта, имеет смысл. Но вам всегда будет нужна master/infinity ветвь, содержащая последний код. infinity ветвь, содержащая последний код. Различные ветви версий вашего приложения периодически создаются из этой основной ветви. На рисунке 9.36 показан сценарий с несколькими удаленными и локальными ветвями. Каждая удаленная ветвь может иметь n-ное количество локальных ветвей, в зависимости от требований проекта. Такой подход с использованием нескольких удаленных ветвей помогает разделить, что будет предоставлять каждая версия продукта, и разработчики могут продолжать обеспечивать поддержку старых версий для устранения ошибок в продукте, выпускать исправления и так далее. Разработчики могут даже решить перенести определенную функцию на более старую версию продукта, если это необходимо.

В этом разделе мы создадим две дополнительные удаленные ветки: одну ветку с помощью GitHub и одну удаленную ветку непосредственно из SAP Web IDE. Затем мы создадим локальные ветки в SAP Web IDE из этих удаленных веток.

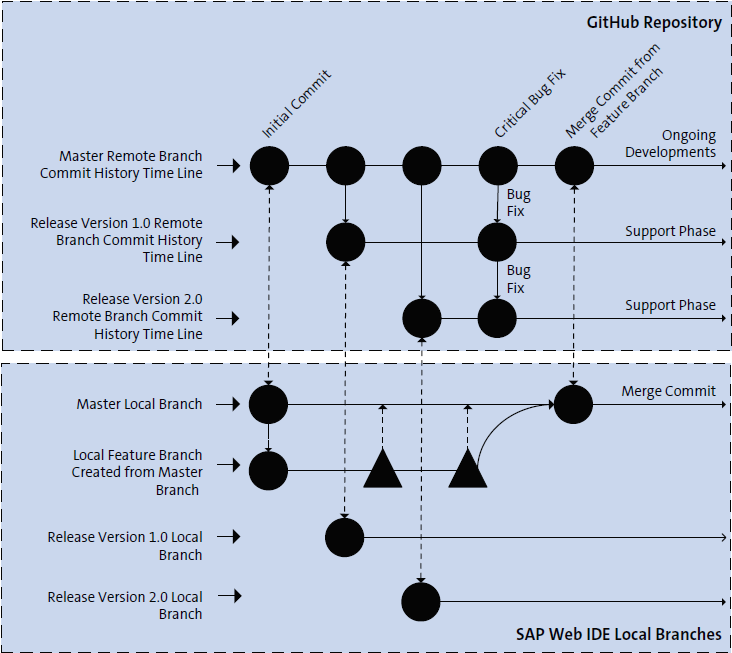


Рисунок 9.36 Работа с несколькими удаленными и локальными ветвями

Сначала давайте создадим новую удаленную ветку под названием ReleaseVersion1.0 на GitHub. Откройте репозиторий проекта, нажмите на кнопку Branch:master, как показано на рисунке 9.37 (1), и введите имя новой ветви (2). Затем нажмите на кнопку Create branch: ReleaseVer sion1.0 (3) в левом нижнем углу экрана.

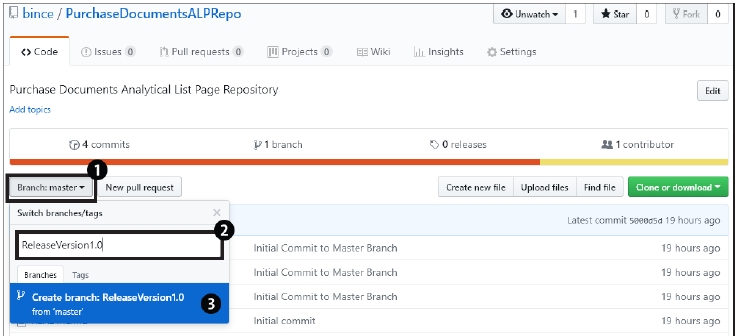


Рисунок 9.37 Создание новой удаленной ветви в GitHub

Теперь давайте создадим новую удаленную ветку под названием ReleaseVersion2.0 с помощью SAP Web IDE. Щелкните правой кнопкой мыши на проекте и выберите Git Create Remote Branch, как показано на рис. 9.38.

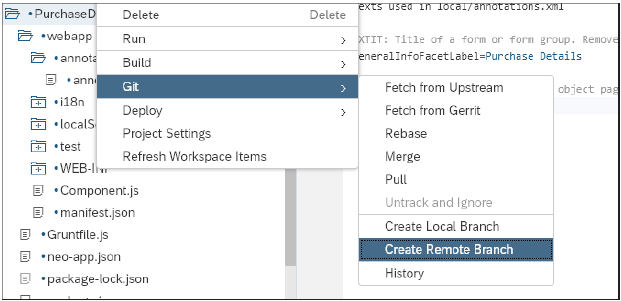


Рисунок 9.38 Создание новой удаленной ветки из SAP Web IDE

В новом всплывающем окне выберите Source Branch в качестве origin/master, как показано на рисунке 9.39. Введите имя филиала как "ReleaseVersion2.0"; будет создана копия с удаленной ветки Master, так что все изменения будут перенесены в новую удаленную ветку).

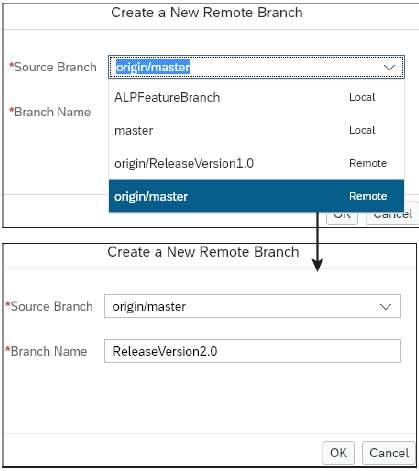


Рисунок 9.39 Создание новой удаленной ветви ReleaseVersion2.0

После создания новых удаленных ветвей можно приступить к созданию локальных ветвей в SAP Web IDE в панели Git (1). Нажмите на значок + (2) рядом с Branch, как показано на рисунке 9.40. Во всплывающем окне выберите Source Branch, из которого вы хотите создать локальную ветвь, и нажмите на кнопку OK (3).

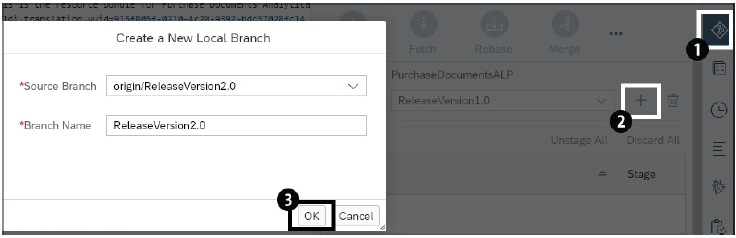


Рисунок 9.40 Создание новой локальной ветви из удаленного ReleaseVersion2.0

Создайте все локальные ветви в соответствии с требованиями вашего проекта. После того, как все ветви будут созданы, вы увидите их в раскрывающемся списке Branch в панели Git в SAP Web IDE, как показано на рисунке 9.41.

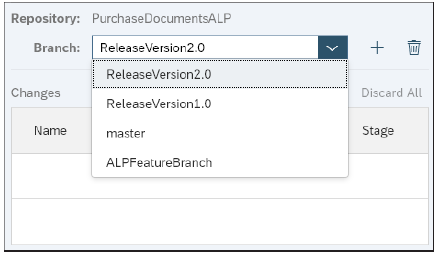


Рисунок 9.41 Обзор всех веток в панели Git в SAP Web IDE

Вы также можете просматривать все свои удаленные ветки непосредственно на странице GitHub, как показано на Рисунок 9.42. При таком подходе разработчики могут легко работать над разными ветками на основе циклов выпуска проекта или релизов функций. Несколько разработчиков в вашей команде могут легче работать над одним и тем же проектом.

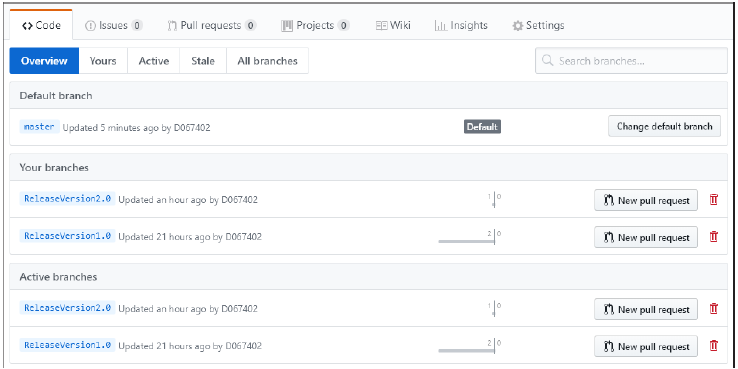


Рисунок 9.42 Обзор всех удаленных веток в GitHub

## 9.3 Резюме

В этой главе вы узнали о репозитории Git и о том, как использовать GitHub в качестве расширения для создания и управления репозиториями в Git. Мы рассмотрели создание локальных репозиториев в SAP Web IDE и связывании их с удаленным репозиторием Git. Мы рассмотрели основные опции GitHub, расположенные на панели Git в SAP Web IDE, включая Pull, Fetch, Rebase, Merge и Reset. Вы также узнали, как фиксировать (commit), перемещать (push) и помещать (stage) коды в удаленные репозитории из SAP Web IDE. Мы обсудили, как создавать несколько локальных и удаленных ветвей, а также как работать с несколькими ветвями для разработки. В следующей главе вы узнаете об автоматизированном тестировании на платформе SAP S/4HANA.