Как создать новую ветку Git профессионально

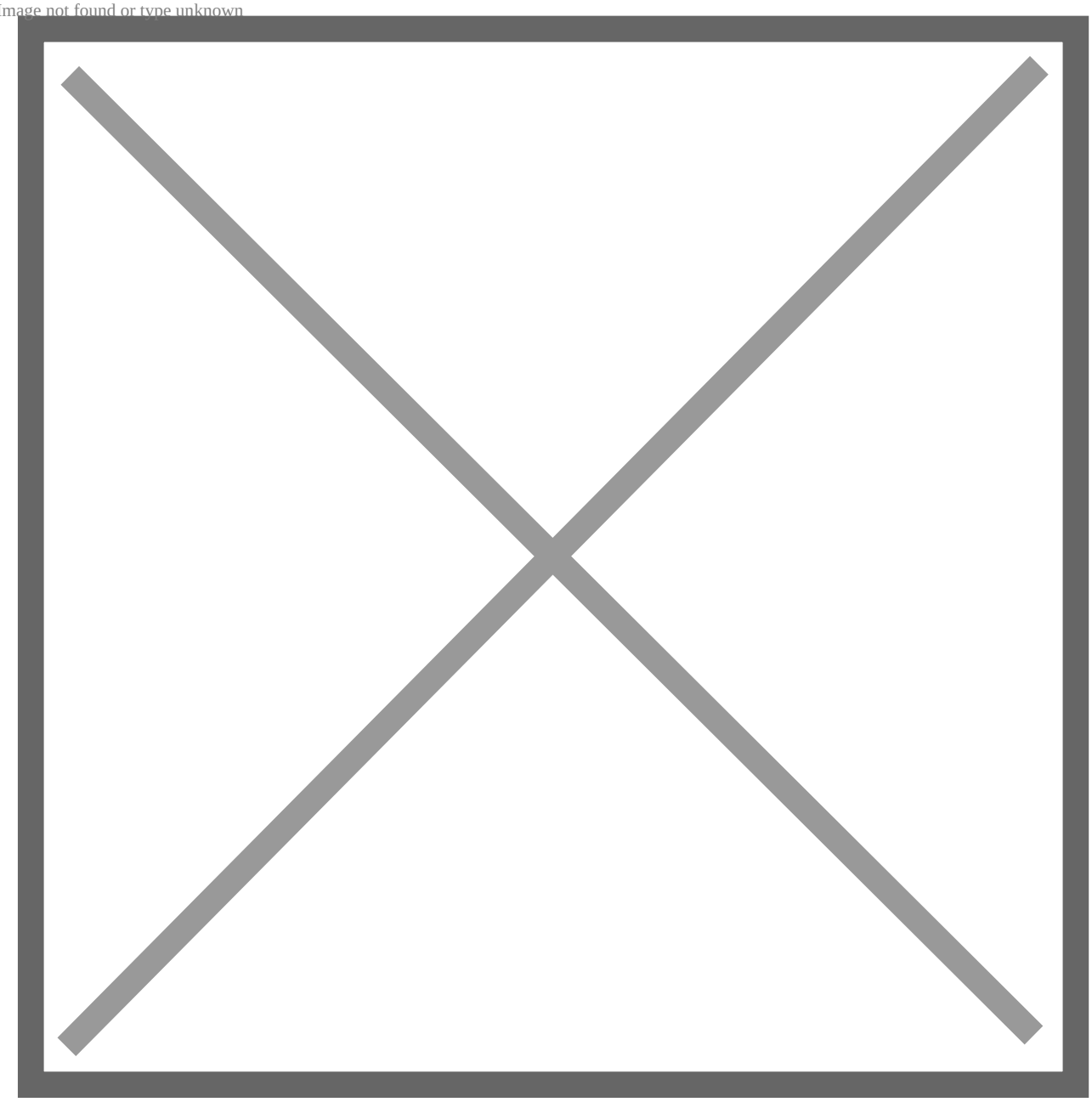
Описание

Ветвление – это основная функция Git’a. С ее помощью вы можете работать над определенной функцией или компонентом программного обеспечения, не ломая свой код. Это возможность для вас как разработчика внести значительные изменения в исходный код и выбрать, что вы хотите сделать с этими изменениями. В любом случае, вы можете объединить их со всем проектом или исключить из него. Если вы работали с Git’ом, то, возможно, уже заметили, что существует несколько способов создания ветвей. Если вы начинающий разработчик и еще не сталкивались с Git, знание того, как использовать инструменты контроля версий, сэкономит вам значительное время, а если нет, то сделает ваш опыт разработчика интересным.

Этот пост тактически ориентирован на то, чтобы помочь вам плавно создавать ветви Git в рамках определенного рабочего процесса. В итоге вы приобретете прочные навыки, которые сможете использовать для обслуживания своих репозиториев GitHub. Примечание: Если вы начинающий разработчик, ознакомьтесь с тем, как настроить учетную запись GitHub, прежде чем двигаться дальше. Если вы уже сделали это, переходите к разделу “Как это сделать”. Однако для закрепления полученных знаний не помешает повторить пройденное.

Что такое ветви Git?

Image not found or type unknown



Ответвление в Git подразумевает версию вашего репозитория, которая отличается от основного проекта (имеется во всех современных системах контроля версий). Проще говоря, вы отходите от основной линии разработки и работаете, не нарушая исходной линии. Многие доступные инструменты контроля версий (VCS) используют эту технику, которая предполагает создание новой копии каталога исходного кода. При работе с большими проектами копирование исходного кода может занять некоторое время.

С другой стороны, это ваш шанс поэкспериментировать с изменениями в коде.

Модель ветвления Git’а считается “убийственной функцией” и ставит Git на один уровень с другими инструментами в сообществе VCS. Модель работы Git проста и делает операции ветвления практически мгновенными, а также переключение через несколько. Если вы освоите модель работы с ветвлениями Git, вы откроете для себя мощные возможности и инструменты, которые могут повысить ваши навыки разработки. Итак, как же обстоят дела с ветвлениями?

Зачем нужны ветви Git

Ветви Git играют ключевую роль в системе контроля версий Git. Вот несколько причин, почему вам нужны ветви Git:

- **Параллельная разработка** – Современное программное обеспечениеявляется сложным, и многие разработчики часто работают вместе над егосозданием. Ветви позволяют разным разработчикам работать над разнымифункциями. В других случаях это может быть исправление ошибок безстолкновений в работе. Каждая ветвь – это ворота в новую линию разработки.Вы можете легко переключаться между ветвями во время работы надопределенными функциями.
- **Сотрудничество** – Ветки Git позволяют работать с другими разработчиками над общей кодовой базой. Вы можете создавать новые ветки, вносить изменения в код и отправлять их в общий репозиторий, где другие разработчики могут просмотреть ваши работы, прежде чем объединить их с основной веткой.
- **Экспериментирование** – Вы можете использовать ветви git для тестирования новых функций до их слияния с основной ветвью и без влияния на нее. Это ваш шанс попробовать новый подход к решению проблемы. И если новые методы в вашем коде работают правильно, вы можете слить их в основную ветку.
- **Управление релизами** – Вы можете использовать ветви для управления релизами. Вы можете создать каждую ветвь, привязанную к определенному релизу в вашем рабочем проекте. Это позволит вам исправлять ошибки и вносить изменения, не затрагивая основную ветвь разработки.
- **Версионирование** – Вы можете использовать ветки Git для версионирования;в этом случае каждая ветка будет представлять собой новую версию вашего программного обеспечения. Как разработчику, лучше всего использовать ветки для каждого выпуска программного обеспечения и отслеживать изменения, которые вы используете для управления различными версиями кодовой базы.

Начало работы с Git – краткая справка

Теперь, когда вы поняли “почему”, пришло время перейти на новый уровень и заняться “как”. Двигаясь вперед, вы уже должны были настроить свой аккаунт на GitHub. Если вам еще нужно это сделать, пожалуйста, сделайте это. Этот раздел является практическим. Команда **git checkout** в Git подразумевает переключение между различными версиями вашей целевой сущности. В некоторых сетевых сообществах разработчиков термин “check out” означает выполнение команды checkout. Эта команда работает с тремя сущностями: ветками, файлами и коммитами.

Проверка ветви

Вы можете использовать команду **git branch** для создания ветвей, по которым вы перемещаетесь с помощью команды **git checkout**. Когда вы проверяете ветку, вы обновляете файлы в вашем рабочем каталоге, чтобы они соответствовали версии, хранящейся там. Другими словами, вы говорите Git’у записывать все ваши коммиты в ветку (изменяя линию разработки). Использование выделенных веток для новых функций в вашей разработке – это переход от старого рабочего процесса subversion (SVN) и облегчает работу с кодом во всех случаях, о которых говорилось в разделе “Зачем нужны ветки”. Команду **git checkout** не следует путать с **git clone**. Первая используется для переключения между версиями кода, а вторая – для получения кода из удаленного репозитория.

Использование существующих ветвей

Если репозиторий, над которым вы работаете, имеет существующие ветви, вы можете визуализировать их в интерфейсе командной строки с помощью команды **git branch**. Доступные ветки будут перечислены, и зеленая ветка – это та, над которой вы работаете в данный момент, если вы используете операционную систему Windows и Visual Studio Code. Чтобы переключаться между ветками, используйте команду **git checkout branchname**. Фраза ‘branchname’ означает имя вашей ветки, и в своей работе вы можете следовать любому соглашению об именовании.

Создание ветвей Git

Предположим, что вы находитесь в середине разработки программного обеспечения и хотите добавить новую функцию. Лучший способ решить эту задачу

– создать новую ветку с помощью ‘git branch’.

Практически, это то, что вы вводите в командной строке:

```
git branch branchname
```

Это означает, что вы создали ответвление от основной ветки **main/master** (в большинстве случаев именно здесь вы запускаете свой живой проект). Имя вашего нового ответвления в данном случае будет ‘branchname’.

Чтобы перейти на новую ветку, вы используете **git checkout** ; см. ниже:

```
git checkout branchname
```

Если вы разработчик, который, как и я, любит экономить время, вы можете создать ответвление и сразу же переключиться на него, используя ‘git checkout’ с аргументом ‘-b’, за которым следует имя вашего ответвления. Практически, вы могли бы просто сделать это, чтобы получить такие же результаты, как и в наших предыдущих шагах, см:

```
git checkout -b branchname
```

Параметр ‘-b’ указывает Git’у запустить **git branch** непосредственно перед ее проверкой. Рассмотрим другие техники, которые можно использовать для создания ответвлений git.

Давайте рассмотрим другие техники, которые вы можете использовать для создания ответвлений в зависимости от ваших потребностей:

Создание ответвления от текущего ответвления

Если вы хотите создать новую ветку на основе текущей, то лучший способ – использовать наши недавно приобретенные навыки:

```
git checkout -b <branchname>
```

Этот метод автоматически создает и переключает вас на новый филиал. Чтобы подтвердить, что вы находитесь в новой ветке, в вашем терминале должно появиться сообщение – switched to a new branch ‘branchname’. Если вы начинающий разработчик, вы должны быть готовы вводить команды на своей консоли без скобок (<>). Они необходимы для иллюстрации и объяснения, их не следует путать или использовать неправильно.

Создание ответвления из другого ответвления

Вы можете создать новую ветку на основе другой существующей ветки, добавив имя этой ветки в качестве отправной точки. Вот команда:

```
git branch <new branch> <base branch>
```

И в практическом случае так и будет:

```
git branch new-branch branchname
```

Это означает, что ‘new branch’ – это наш новый филиал, а ‘branchname’ – это наш базовый (основополагающий) филиал.

Создание ответвления из коммита

Если вы хотите создать новое ответвление на основе коммита (а не ветки), вам нужно указать хэш коммита в качестве отправной точки. А чтобы найти хэш коммита, на основе которого вы создаете ответвление, запустите **git log**.

Хеш коммита обычно представляет собой длинную строку символов, начинающуюся с ‘commit’. Имея хэш коммита, вы можете создать его, выполнив команду:

```
 git branch <branch name> <commit-hash>
```

Затем вы можете перейти на свою новую ветвь, проверив ее.

Создание ответвления от тега

Чтобы создать ответвление от определенного тега, найдите имя тега, от которого вы хотите создать ответвление. Выполните команду **git tag**, чтобы получить список всех доступных тегов в вашем репозитории. Определив имя тега, выполните команду **git branch <new branch> <tag name>**, после чего вы сможете

переключиться на новую ветвь и начать вносить изменения в код.

Создание ответвления с помощью отсоединенного состояния HEAD

Вы можете использовать состояние отсоединенного HEAD для создания новых ветвей, которые начинаются с определенного коммита без немедленного переключения на эту ветвь. Эта техника полезна, когда вы хотите поэкспериментировать с новыми изменениями, не затрагивая ветку, над которой вы работаете. Начните с поиска хэша коммита, который вы хотите использовать для создания ответвления – используйте **git log**. Получив хэш коммита, выполните команду: **git checkout <commit hash>**. Эта команда подразумевает, что вы находитесь в состоянии detached HEAD, то есть вы не находитесь в ветке, а указываете на конкретный коммит. Далее вы можете использовать **git branch < branch name>**, чтобы создать ветку на основе вашего текущего коммита.

Создание ответвления от удаленного ответвления

Начните с создания локальной ветки. К этому моменту вы уже должны уметь использовать: **git checkout -b <branch name>**. Удаленная ветка создается автоматически, когда вы перемещаете локально созданную ветку в удаленный репозиторий. Вы перемещаете ветку в удаленное хранилище командой: **git push origin <branch name>** В этой команде ‘origin’ означает удаленное хранилище, в которое вы перемещаете изменения и в котором создаете ветки. Вы можете заменить его на имя вашего удаленного репозитория; это прекрасно работает.

Создание ответвления в удаленном репозитории

Чтобы создать ответвление в удаленном репозитории, получите последние изменения из вашего удаленного репозитория, выполнив команду **git fetch**. Получив последние обновления, вы можете создать новую ветку, проверив ее. А после проверки вы можете отправить новые изменения:

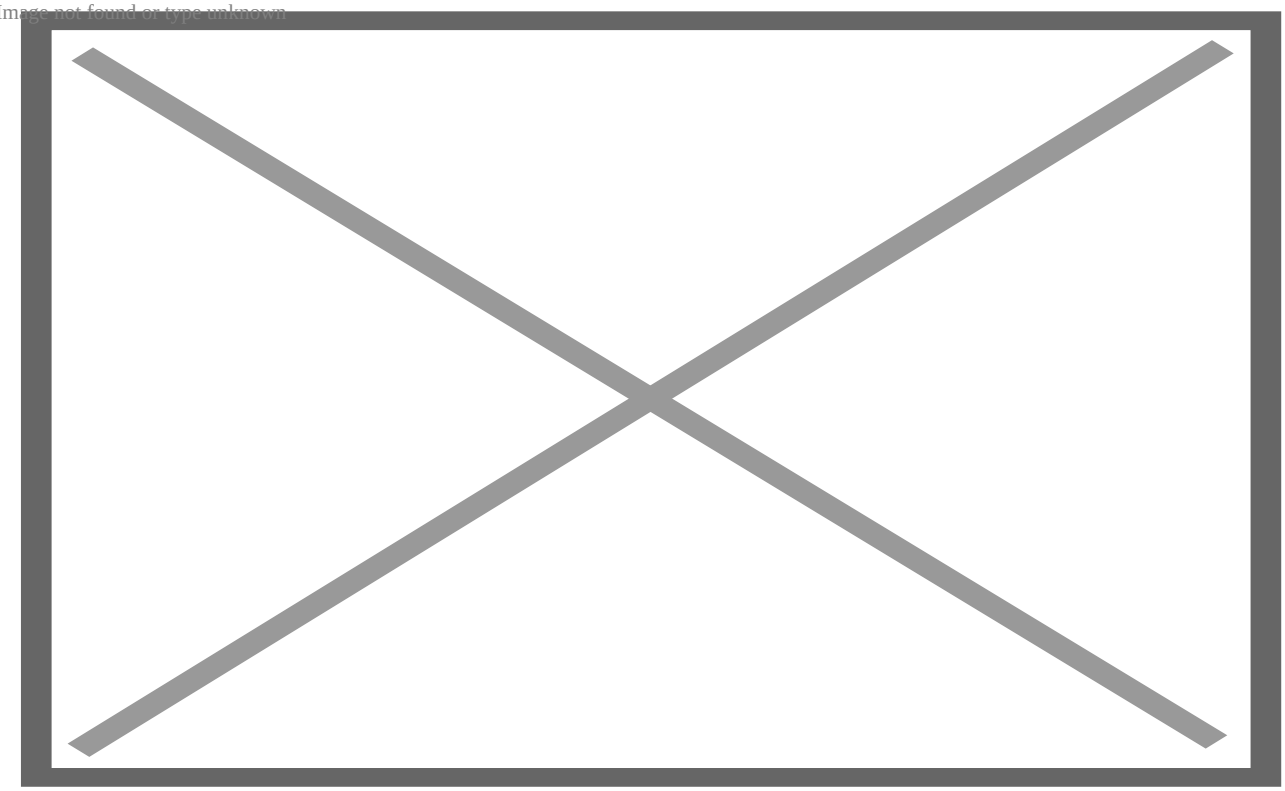
```
git push -u <remote repo> <new branch name>
```

В ваших операциях лучше всего использовать origin (как в случае создания ответвления от удаленного ответвления). Это сэкономит ваше время и снизит вероятность ошибки при вводе имен удаленных репо.

Онлайн-интерфейс GitHub

Все задачи, выполняемые в командной строке, можно повторить с помощью интерфейса GitHub Online. Чтобы создать новую ветку, откройте страницу репозитория проекта и найдите ветки в левом верхнем углу – чаще всего это master/main. Если вы нажмете на него, ниже будет показан список доступных ветвей, а также текстовое поле, которое вы можете использовать для поиска или создания ветви.

Image not found or type unknown



Чтобы создать ветвь, введите ее название в текстовое поле. Онлайн-интерфейс автоматизирован и позволяет создавать ветви из определенных элементов, таких как теги, ветви и коммиты. А если вы новичок в создании ответвлений, ознакомление с документацией GitHub по ответвлениям может избавить вас от проблем в будущих начинаниях разработчика.

Заключительные слова

Изучив несколько методов создания ответвлений в Git, вы теперь можете использовать приобретенные навыки в разработке программного обеспечения с плавным рабочим процессом Git. Лучший вариант создания ответвлений будет во многом зависеть от вашей команды: критериев рабочего процесса и событий. Например, если вы просите соавторов внести свой вклад, они могут создавать ветви удаленно и вносить неоценимый вклад. Вы видели, как разветвление Git предоставляет вам мощный механизм для более эффективной и результативной работы над программными проектами. И хотя существуют и другие варианты управления Git, этот пост показал вам, как ориентироваться в ветвях Git в командной строке, и помог вам чувствовать себя более уверенно при использовании Git.

Дата Создания
10.05.2023