GIT

решение конфликтов
ветвление
git fork
github pages
vercel
CI/CD + модели ведения проекта

GIT ветвление

Git ветвление — это одна из ключевых особенностей системы контроля версий Git, которая позволяет пользователям создавать, управлять и сливать различные версии (ветки) своего проекта. Ветки в Git позволяют параллельно работать над разными задачами, такими как разработка новых функций, исправление багов или экспериментирование с новыми идеями, без вмешательства в основную (главную) ветку проекта.

Преимущества ветвления:

- Изолированная работа: Ветки позволяют вам изолировать работу над новыми функциями и исправлениями ошибок, не влияя на основной код.
- Совместная работа: Разные участники команды могут работать над разными ветками одновременно.
- История изменений: Ветки сохраняют историю всех изменений, что упрощает отслеживание и возврат к предыдущим состояниям проекта.

Основные команды для работы с ветками

Создание новой ветки: ***git branch имя_ветки***

Переключение на другую ветку: ***git checkout имя_ветки***

Создание и переключение на новую ветку: ***git checkout -b имя_ветки***

Просмотр всех веток: ***git branch***

Просмотр всех веток, включая локальные и удаленные: ***git branch -a***

Слияние веток: ***git merge имя_ветки***

Удаление ветки: ***git branch -d имя_ветки***

Что такое GIT FORK

Git fork (форк) — это процесс создания копии существующего репозитория, обычно из другого пользователя или организации, на своей учетной записи. Это не команда в Git, а функциональность, предоставляемая хостинг-сервисами для репозиториев, такими как GitHub, GitLab и Bitbucket. Форк позволяет разработчику независимо работать с копией проекта, вносить изменения и предлагать их обратно в исходный репозиторий через pull request (запрос на включение изменений).

Как сделать форк? На примере GitHub:

1. Найти репозиторий: Перейдите к репозиторию, который хотите форкнуть.
2. Нажать кнопку "Fork": В правом верхнем углу страницы репозитория нажмите кнопку "Fork".
3. Выбор аккаунта: Выберите, в какой аккаунт или организацию хотите форкнуть репозиторий.
4. Получение форка: После выполнения этих шагов у вас будет форкнутый репозиторий в вашем аккаунте.

Работа с форкнутым репозиторием

Клонирование форка: Клонируйте форкнутый репозиторий на локальную машину.

Добавление удаленного репозитория: Добавьте исходный репозиторий как удаленный репозиторий с именем upstream, чтобы можно было синхронизировать изменения.

Синхронизация с исходным репозиторием: Периодически подтягивайте изменения из исходного репозитория и вносите их в свой форк.

Работа с ветками: Создайте ветку для работы над новыми функциями или исправлениями.

Коммиты и пуш: Внесите изменения, сделайте коммиты и отправьте их в ваш форк.

Создание pull request: Перейдите на сайт (например, GitHub) и создайте pull request из вашей ветки в форкнутом репозитории в ветку main или другую ветку исходного репозитория.

```
git clone https://github.com/ваш_аккаунт/форкнутый_репозиторий.git
cd форкнутый_репозиторий
```

```
git remote add upstream https://github.com/оригинальный_автор/исходный_репозиторий.git
```

```
git fetch upstream
git checkout main
git merge upstream/main
```

```
git checkout -b new-feature
```

```
git add .
git commit -m "Добавлена новая функция"
git push origin new-feature
```

Конфликты версий

Конфликты в Git возникают, когда изменения из разных веток касаются одного и того же участка кода и не могут быть автоматически объединены. Решение конфликтов — это важный навык для эффективной работы с Git.

Способы решения конфликтов

- Rebase (перебазирование)
- Merge (слияние)

Merge

Merge (слияние) в Git — это процесс объединения изменений из одной ветки в другую. Он создаёт новый коммит, который содержит изменения обеих веток.

Есть две ветки А и Б, при залитии А на Б происходит конфликт:

Актуализируем ветки (git pull; git fetch)

Заливаем ветку А на ветку Б, для этого:

1. git checkout Б - переходим на ветку Б;
2. git merge А - на ветке Б производим слияние с веткой А;
3. решаем конфликты через интерфейс IDE-шки
4. git add . фиксируем все изменения
5. git commit -m “solve conflicts” - создаем коммит с решением конфликта
6. git push origin Б - синхронизируем ветку с удаленным репозиторием

git merge –abort => отмена слияния

Rebase

Git rebase - это команда, которая используется для переноса или перебазирования коммитов на другую ветку. Основная цель git rebase - это изменить историю коммитов, делая её более чистой и логичной, путем применения коммитов из одной ветки на другую.

Основные шаги работы git rebase:

- Выбор базовой ветки: Вы выбираете ветку, на которую хотите перебазировать свою текущую ветку.
- Запуск команды: Вы запускаете git rebase с указанием целевой ветки. Например, git rebase main перебазирует текущую ветку на ветку main.
- Применение коммитов: Git применяет каждый коммит вашей текущей ветки поверх целевой ветки.
- Разрешение конфликтов: Если возникают конфликты между изменениями в ваших коммитах и изменениями в базовой ветке, Git останавливает процесс и требует разрешения конфликтов.
- Завершение ребейза: После разрешения конфликтов Git завершает перебазирование и ваши коммиты теперь лежат поверх целевой ветки.

Важно помнить, что перебазирование изменяет историю коммитов, поэтому следует использовать его с осторожностью, особенно если работаете с общими ветками, доступ к которым есть у других разработчиков.

Различия между merge и rebase

Merge создает новый коммит, который объединяет изменения из двух веток, сохраняя историю обоих веток.

Rebase перемещает коммиты из текущей ветки на новую базу, переписывая историю и создавая более линейную и чистую историю изменений.

Оба метода имеют свои преимущества и могут использоваться в зависимости от требований проекта и предпочтений команды.

GitHub pages

GitHub Pages – это бесплатный сервис хостинга статических сайтов, который является частью платформы GitHub. Он позволяет вам публиковать свои веб-страницы непосредственно из репозитория GitHub.

- Перейдите на страницу своего репозитория на GitHub.
- Нажмите кнопку "Настройки" ("Settings") в правом верхнем углу.
- Выберите "Страницы" ("Pages") в левом меню.
- В разделе "Источник" ("Source") выберите ветку, содержащую файлы вашего сайта (обычно это ветка "main").
- GitHub Pages автоматически сгенерирует URL-адрес для вашего сайта. Вы можете найти его в поле "URL-адрес GitHub Pages" ("GitHub Pages URL").

Что такое vercel

Vercel – это облачная платформа для разработки, развертывания и хостинга веб-приложений и статических сайтов.

Vercel подходит для:

- Веб-приложения: Развертывайте любые веб-приложения, созданные с помощью современных фреймворков.
- Одностраничные приложения (SPA): Vercel идеально подходит для SPA, так как он обеспечивает быструю загрузку и плавную работу.
- Статические сайты: Vercel также может использоваться для хостинга статических сайтов.
- Сервисы: Развертывайте серверные функции с помощью Vercel Functions.

Что такое CI/CD

CI/CD (Continuous Integration/Continuous Delivery) — это практика разработки программного обеспечения, направленная на улучшение процесса разработки, ускорение доставки и повышение качества программного продукта

Проблемы, решаемые CI/CD:

- Частые итерации разработки: CI/CD позволяет быстрее интегрировать и тестировать код, что способствует более частым итерациям разработки и, как следствие, улучшает скорость разработки и внедрения новых функций.
- Уменьшение рисков: Благодаря автоматизированным тестам и процессам, CI/CD снижает риск внесения ошибок в продукт, так как проблемы в коде обнаруживаются раньше и исправляются быстрее.
- Повышение качества ПО: Автоматизированные процессы сборки и тестирования обеспечивают более высокое качество кода и приложения в целом.
- Быстрая доставка изменений: CI/CD ускоряет процесс доставки изменений в продакшн, что особенно важно в условиях частых обновлений и требований рынка.

Continuous Integration (CI)

CI — это практика, при которой разработчики регулярно интегрируют свой код в общий репозиторий. Основные принципы CI включают:

- Автоматизированные сборки и тесты: При каждом коммите кода происходит автоматическая сборка проекта и запуск тестов.
- Раннее обнаружение ошибок: Благодаря автоматическим тестам, проблемы в коде обнаруживаются на ранних стадиях разработки, что позволяет быстрее их исправлять.

Continuous Delivery (CD)

CD — это расширение CI, охватывающее автоматизацию процессов доставки программного обеспечения на производственные серверы или другие целевые среды. Основные аспекты CD включают:

- Автоматизированная сборка и доставка: После успешного прохождения всех тестов код автоматически собирается и развертывается на целевой среде.
- Управление версиями и конфигурациями: При помощи CD можно управлять версиями приложения и его конфигурациями, обеспечивая надежность и консистентность процесса развертывания.

Ресурсы

Vercel - ТЫК

Модели ветвления (статья) - ТЫК

Простой back-end для todo-шки - ТЫК

документация по использованию github pages - ТЫК