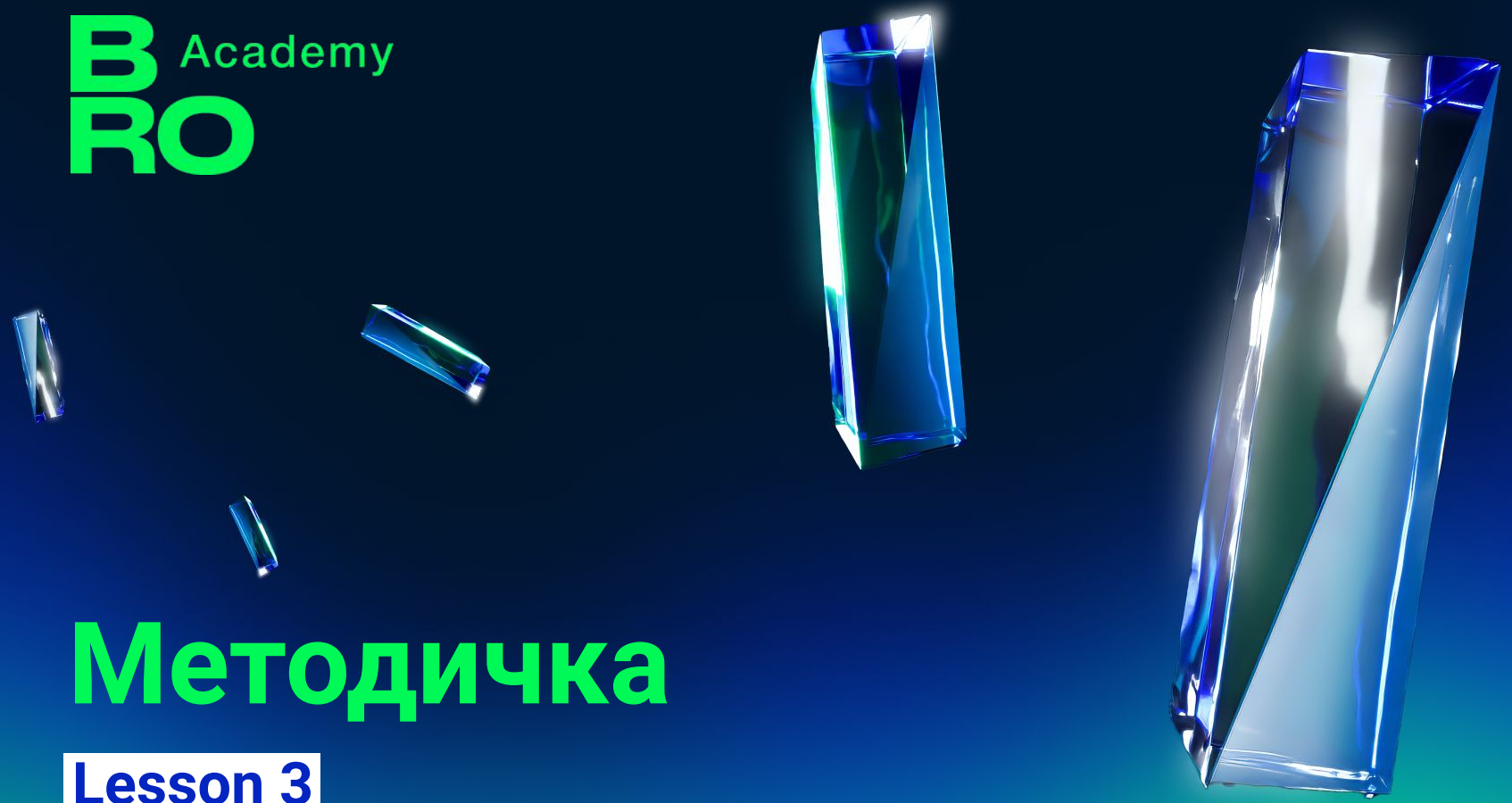
BRO Academy



Методичка

Lesson 3

Система контроля версий

Если вы работаете над проектом не в одиночку, то вам нужно **обмениваться кодом** с коллегами. Даже если вы работаете один, то полезно будет **хранить историю изменений кода**, чтобы всегда можно было понять когда и какие изменения были внесены.

Часто разработчики трудятся в команде над одним проектом, а значит, сразу несколько человек могут изменять один файл одновременно. Чтобы избежать путаницы, в таких случаях используют **систему контроля версий**, которая позволяет **хранить историю** изменений проекта и при необходимости помогает вернуться к предыдущей версии. Одна из самых популярных — Git.

Система контроля версий **(VCS)** - это как "машина для путешествий во времени" для программистов. Когда вы работаете над кодом, VCS позволяет вам сохранять разные версии, возвращаться назад в прошлое и смотреть, какие изменения были сделаны. Она помогает учёным, программистам и всем, кто создает что-либо, следить за изменениями и управлять процессом разработки.

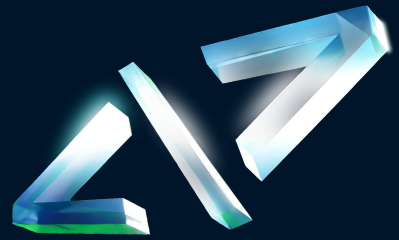
Представьте ситуацию: геймер доходит до финала, проигрывает и возвращается к началу уровня — попадает в ближайшую контрольную точку игры, где разработчики разрешили сохраниться. Если мы уберём контрольные точки, после каждого проигрыша придётся начинать игру заново.

Или представьте, что вы пишете книгу. Вы делаете разные версии - редактируете, добавляете новые страницы и исправляете ошибки. Вам было бы удобно сохранять каждую версию отдельно, чтобы всегда можно было вернуться назад или посмотреть, как менялись ваши мысли. В мире программирования используется что-то похожее - и это система контроля версий.

Зачем использовать git

- Хранение полной истории изменений
- Описание причин всех сделанных изменений
- Откат изменений, если что-то пошло не так
- Поиск причины и ответственного за появления ошибок
- Совместная работа команды над одним проектом
- Возможность изменять код, не мешая работе коллег

GIT



Git — это распределённая и децентрализованная система управления версиями файлов.

Децентрализованная система означает, что у каждого разработчика есть личный репозиторий проекта с полным набором всех версий. А все необходимые для работы файлы находятся на компьютере. При этом постоянное подключение к сети не требуется, поэтому система работает быстро. При командной разработке нужна синхронизация репозиториев, так как проект — один и его состояние должно быть у всех одинаковым.

Изначально была разработана для разработки линукса.

Git – это инструмент, который позволяет программистам сохранять, управлять и отслеживать изменения в коде. Он позволяет создавать "снимки" кода (называемые коммитами) и следить за тем, как код меняется со временем.

И одновременно git — это программа, которую нужно установить и подключить к проекту для управления системой контроля версий.

Из чего состоит git

Репозиторий (repository)

— каталог файловой системы, в котором находятся: файлы конфигурации, файлы журналов операций, выполняемых над репозиторием, индекс расположения файлов и хранилище, содержащее сами контролируемые файлы.

Локальный репозиторий (local)

— репозиторий, расположенный на локальном компьютере разработчика в папке. Именно в нём происходит разработка и фиксация изменений, которые отправляются на удалённый репозиторий.

Удалённый репозиторий (remote)

— репозиторий, находящийся на удалённом сервере. Это общий репозиторий, в который приходят все изменения и из которого забираются все обновления.

У нас удаленные репозитории ваших проектов будут лежать на github

Из чего состоит git

Форк (Fork)

— копия репозитория. Его также можно рассматривать как внешнюю ветку для текущего репозитория. Копия вашего открытого репозитория на Гитхабе может быть сделана любым пользователем, после чего он может прислать изменения в ваш репозиторий через пулреквест.

Хранилище

— это содержимое скрытой папки .git. В этой папке хранятся все версии рабочей области и служебная информация. Этим версиям система автоматически даёт название, состоящее из букв и цифр. Например, это bea0f8e и d516600. Не стоит проводить манипуляции с папкой .git вручную. Вся работа с системой производится командами через специальные приложения или консоль.

Клонирование (Clone)

— скачивание репозитория с удалённого сервера на локальный компьютер в определённую папку для дальнейшей работы с этой папкой как с репозиторием.

Из чего состоит git

Ветка (Branch)

— это параллельная версия репозитория. Она включена в этот репозиторий, но не влияет на главную версию, тем самым позволяя свободно работать в параллельной. Когда вы внесли нужные изменения, то вы можете объединить их с главной версией.

Мастер (Master)

— главная или основная ветка репозитория.

Коммит (Commit)

Точно так же, как и в игре, в системе контроля версий Git можно сохранить текущее состояние проекта. Для этого есть специальная команда — commit. Она делает так, что новая версия проекта сохраняется и добавляется в хранилище. В файле с сохранением отображаются: все изменения, которые происходили в рабочей области, автор изменений и краткий комментарий, описывающий суть изменений. Каждый коммит хранит полное состояние рабочей области, её папок и файлов проекта.

Из чего состоит git

Pull

Если работа над проектом ведётся в команде, то перед тем как начать писать код, нужно получить последнюю версию проекта. Для этого нужно выполнить команду pull. Так мы забираем все изменения, которые были совершены со времени последней синхронизации с удалённым репозиторием. Теперь они у нас в репозитории на локальном компьютере.

Push

Чтобы отправить коллегам последнюю версию проекта выполняем команду push. Если в удалённом репозитории с момента последней синхронизации не было никаких изменений, то все сохранённые изменения успешно загрузятся в облако, и коллеги получат последнюю версию проекта, выполнив команду pull. Если же были изменения, то Git попросит вас перед отправкой подтянуть последние версии, сделав pull.

Пулреквест (Pull Request)

— запрос на слияние форка репозитория с основным репозиторием. Пулреквест может быть принят или отклонён вами, как владельцем репозитория.

Из чего состоит git

Мёрдж (Merge)

— слияние изменений из какой-либо ветки репозитория с любой веткой этого же репозитория. Чаще всего слияние изменений из ветки репозитория с основной веткой репозитория.

Кодревью (code review)

— процесс проверки кода на соответствие определённым требованиям, задачам и внешнему виду.

В итоге проект работает так:

Репозиторий хранит все версии проекта. В случае передачи этого проекта другому человеку, он увидит всё, что с ним происходило до этого.

Ничего не теряется и не удаляется бесследно. При удалении файла в новой версии добавляется запись о том, что файл был удалён.

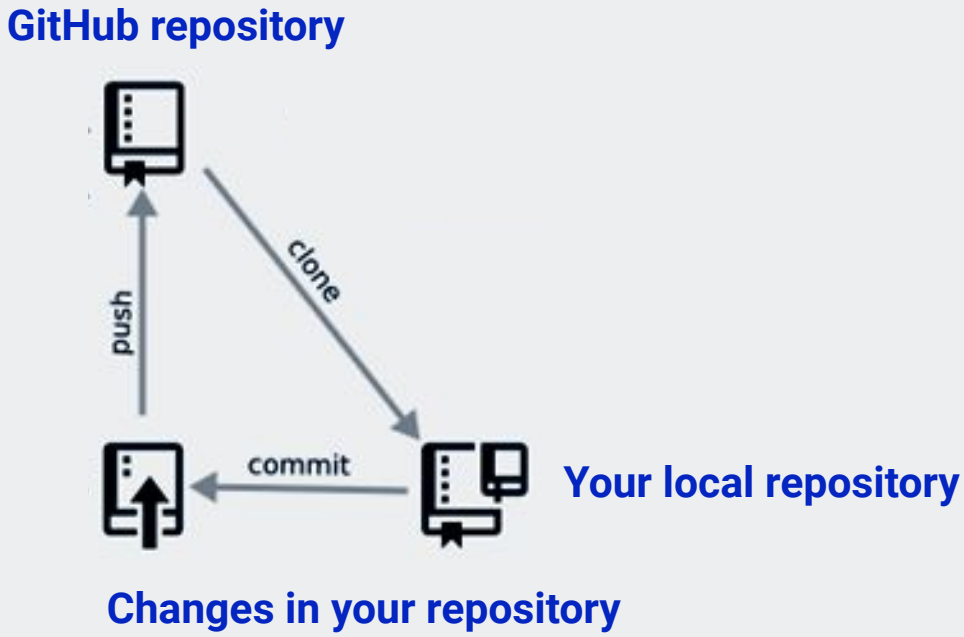
Всегда можно вернуться к любой из версий проекта, загрузив её из хранилища в рабочую область.

Как синхронизовать данные репозиториев между разработчиками?

Изначально Git репозитории сами могут синхронизироваться от пользователя к пользователю. Дополнительные программы для этого не нужны. Есть специальные команды в консоли, позволяющие передавать данные из одного репозитория в другой.

Этот способ сложный и редко используется. Чаще всего разработчики синхронизируют локальный репозиторий с удалённым. Удалённый репозиторий — это тот же репозиторий, только его данные находятся в облаке.

Процесс работы с репозиторием на курсе



Как использовать Git с GitHub

01. Создание репозитория

Репозиторий - это место, где вы будете хранить свой код. На GitHub вы можете создать репозиторий, который будет представлять ваш проект.

02. Клонирование репозитория

Когда репозиторий создан на GitHub, вы можете **склонировать** его себе на компьютер. Это как скачивание книги с интернета, чтобы вы могли работать над ней локально.

03. Добавление и изменение кода

Вы редактируете свой код и каждый раз, когда вы делаете какое-то изменение, вы сохраняете его с помощью "**коммита**". Это, как будто вы делаете фотографию текущего состояния вашей книги.

04. Отправка изменений на GitHub

После того как вы внесли изменения в свой код и сделали коммит, вы можете **отправить** эти изменения на GitHub. Это как будто вы делились своей книгой с другими читателями.

05. Совместная работа

Теперь другие люди могут видеть ваш код, комментировать его и даже предлагать свои изменения. Вы можете обсуждать изменения и вносить их в свой код.

06. Обновление кода

Когда кто-то вносит изменения в ваш код, вы можете **получить** эти изменения на свой компьютер. Это как будто вы загружаете новую главу книги, которую кто-то другой написал.

GitHub Flow

GitHub Flow - это способ работы с Git и GitHub, который подходит для небольших и средних проектов. Он позволяет эффективно совместно работать над кодом и вносить изменения. Вот как это работает:

Создание ветки

Когда вы начинаете работу над новым кодом или исправлением, вы создаете новую **ветку** в репозитории. В этой ветке вы будете делать свои изменения.

Добавление изменений

Вы добавляете, редактируете и удаляете файлы в вашей ветке. Каждое изменение называется **коммитом**. Вы можете делать много коммитов, пока не закончите работу.

Отправка пулл-реквеста

Когда вы готовы поделиться своими изменениями, вы создаете **пулл-реквест** - это как предложение сделать изменения в основную ветку кода. Другие люди могут посмотреть, обсудить и оценить ваши изменения.

Обсуждение и внесение правок

В процессе обсуждения пулл-реквеста вы можете получить комментарии и предложения по улучшению. Вы можете **внести правки**, пока все не будет готово.

Слияние изменений

Когда ваш пулл-реквест готов, вы можете **смерджить** ваши изменения с основной веткой кода. Таким образом, ваш код становится частью проекта.