**Практическая работа № 6.**

РАБОТА В СИСТЕМЕ КОНТРОЛЯ ВЕРСИЙ

***Цель работы:***Регистрация на Github. Создание репозитория, клонирование. Команды Github’а..

Отчет по работе должен содержать:

1. Тему и цель работы.
2. Формулировки практических заданий.
3. Выполненное задание.
4. Ответы на контрольные вопросы

Теоретическая часть

Распределенная система контроля версий

Централизованная система контроля версий (CVCS) использует центральный сервер для хранения всех файлов и позволяет совместную работу. Но основным недостатком CVCSявляется его единственная точка отказа, то есть отказ центрального сервера. К сожалению, если центральный сервер падает на час, то в течение этого часа никто не может взаимодействовать. В худшем случае, если диск центрального сервера поврежден и надлежащее резервное копирование не было выполнено, вы потеряете всю историю проекта. Здесь в систему входит система управления распределенной версией (DVCS).

*Репозиторий*, или проект Git, охватывает весь набор файлов и папок, связанных с проектом, а также историю изменений каждого файла. История файлов отображается как моментальные снимки во времени, называемые *коммитами* , а коммиты существуют как отношения связанных списков и могут быть организованы в *ветви* . Поскольку Gitявляется DVCS, репозитории являются автономными единицами, и каждый, кто владеет копией репозитория, может получить доступ ко всей базе кода и к своей истории. Используя командную строку или другие легкодоступные интерфейсы, репозиторий gitтакже позволяет: взаимодействовать с историей, клонировать, создавать ветви, коммитить, объединять, сравнивать изменения между версиями кода и многое другое.

С помощью таких платформ, как GitHub, **BitBucket, SourceForge**, Gitтакже предоставляет больше возможностей для прозрачности проекта и совместной работы. Публичные хранилища помогают командам работать вместе, чтобы создать наилучший конечный продукт.

Основные команды Git

Для использования Gitразработчики используют определенные команды для копирования, создания, изменения и комбинирования кода. Эти команды могут быть выполнены непосредственно из командной строки или с помощью приложения, такого как GitHubDesktopили GitKraken. Вот несколько общих команд для использования Git:

* gitinitинициализирует новый репозиторий Gitи начинает отслеживать существующий каталог. Он добавляет скрытую подпапку в существующую директорию, в которой размещается внутренняя структура данных, необходимая для контроля версий.
* gitcloneсоздает локальную копию проекта, который уже существует удаленно. Клон включает в себя все файлы проекта, историю и ветви.
* gitaddэтапы изменения. Gitотслеживает изменения в коде проекта разработчика, но необходимо создать и сделать снимок изменений(snapshot), чтобы включить их в историю проекта. Эта команда выполняет первый этап двухэтапного процесса. Любые изменения, которые будут поставлены, станут частью следующего моментального снимка и части истории проекта. Внесение изменений в два этапа дает разработчикам полный контроль над историей своего проекта.
* gitcommitсохраняет моментальный снимок в истории проекта и завершает процесс отслеживания изменений. Короче говоря, коммит функционирует подобно съемке. Все, что было поставлено с gitadd, станет частью моментального снимка с gitcommit.
* git status показывает статус изменений
* gitbranchпоказывает ветви, обрабатываемые локально. Создает ветку.
* gitmergeобъединяет линии развития вместе. Эта команда обычно используется для объединения изменений, выполненных на двух разных ветвях.
* gitpullобновляет локальную линию разработки обновлениями от своего удаленного репозитория.
* gitpushобновляет удаленный репозиторий любыми коммитами, сделанными локально.

Практическая часть

В первую очередь надо установить клиент git: обязательно потребуется консольный клиент, доступный по ссылке http://git-scm.com/downloads

(поддерживаются основные ОС), графический клиент можно установить по желанию, исходя из своих предпочтений. На Unix системах можно воспользоваться менеджером пакетов (yum на fedora и подобных или apt-get на debian, ubuntu и подобных) вместо того, чтобы скачивать установщик с сайта.

Далее работа с git будет объясняться на примере работы с консольным клиентом по следующим причинам:

* Чтобы у вас складывалось понимание происходящего и при возникновении проблем вы могли четко объяснить, что вы делали, и было видно, что пошло не так.
* Все нажатия кнопок в графических клиентах в итоге сводят к выполнению определённых команд консольного клиента, в то же время возможности графических клиентов ограничены по сравнению с консольным
* У тех, кто будет работать в классе на стоящих там компьютерах, не будет другого выбора, кроме как пользоваться консольным клиентом (на сколько мне известно, никаких графических клиентов для git там не установлено)

В данной работе рассмотрим использование для этих целей популярного в настоящее время сервиса Github.

GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git и разработан на RubyonRails и Erlang компанией GitHub,

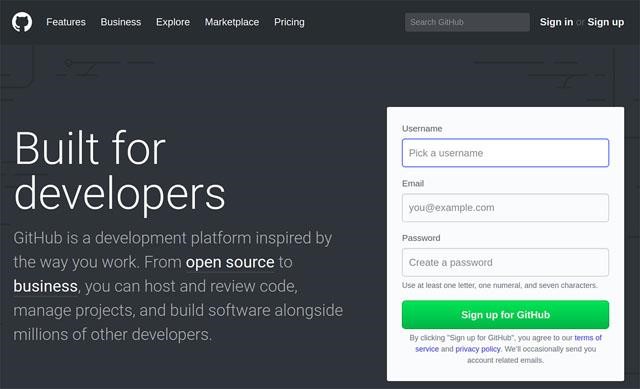
На его примере разберем, как работать с получением изменений и их отправкой.

**Регистрация на GitHub**

**GitHub** — веб-сервис, который основан на системе Git. Это такая социальная сеть для разработчиков, которая помогает удобно вести коллективную разработку IT-проектов. Здесь можно публиковать и редактировать свой код, комментировать чужие наработки, следить за новостями других пользователей. Именно в GitHub работаем мы, команда Академии, и студенты интенсивов.

Чтобы начать работу с GitHub, нужно зарегистрироваться на сайте, если вы ещё этого не сделали. За дело.

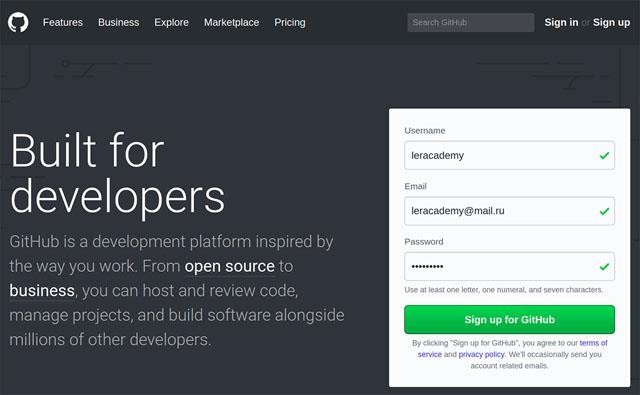
1. Переходим на[сайтGitHub.](https://github.com/)



Cтартовая страница GitHub.

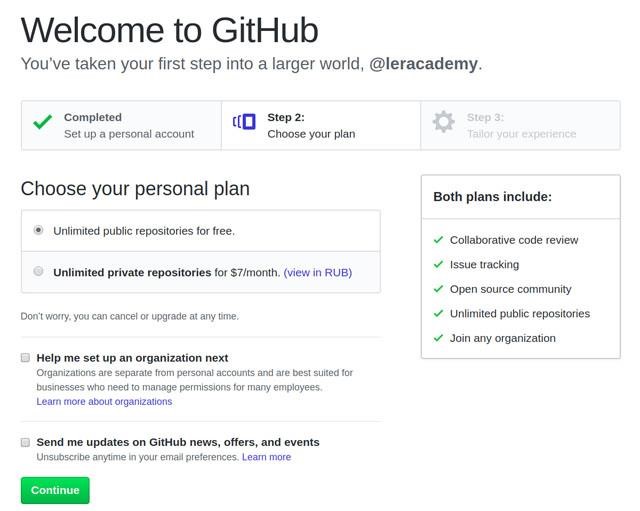
1. Есть два варианта начала регистрации:

* Нажимаем кнопку Sign up (зарегистрироваться), попадаем на страницу регистрации, где вводим обязательные данные: имя пользователя, адрес электронной почты и пароль. После заполнения полей нажимаем Createan account (создать аккаунт).
* Cразу вводим имя, почту и пароль на главной странице GitHub и нажимаем Signup forGitHub (зарегистрироваться на GitHub).



Первый шаг регистрации профиля на стартовой странице GitHub.

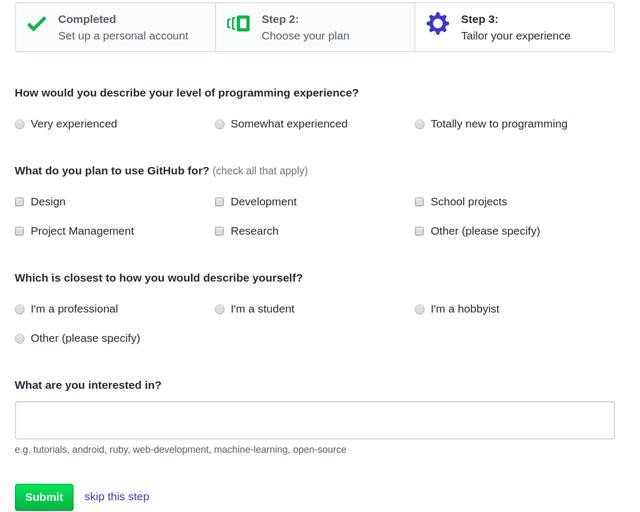
1. На втором этапе нужно выбрать тарифный план. GitHub — бесплатный сервис, но предоставляет некоторые платные возможности. Выбираем тарифный план и продолжаем регистрацию.



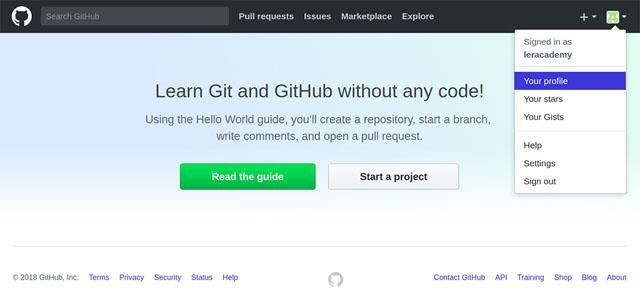
Выбор тарифа на втором шаге регистрации.

Пользование GitHub бесплатно до тех пор, пока ваш код открыт для всех. Платные версии позволяют создавать закрытые, приватные репозитории, которые видите только вы.

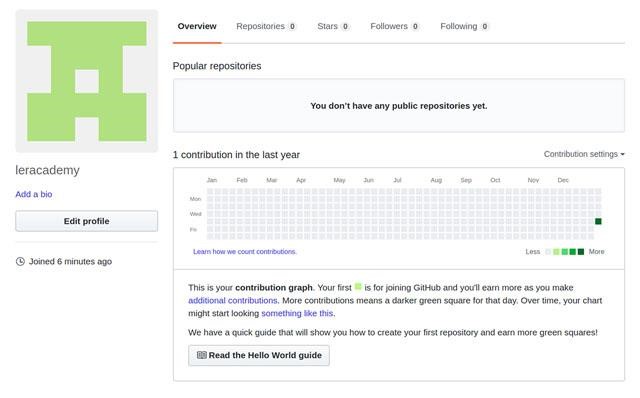
1. Третий шаг — небольшой опрос от GitHub, который вы можете пройти, заполнив все поля и нажать Submit или пропустить, нажав skipthisstep.



1. Опрос на третьем шаге регистрации.
2. После прохождения всех этапов на сайте, на указанный при регистрации ящик вам придёт письмо от GitHub. Откройте его и подтвердите свой почтовый адрес, нажав Verifyemailaddress (подтвердить электронный адрес) или скопируйте вспомогательную ссылку из письма и вставьте её в адресную строку браузера.



Переход в ваш профиль.



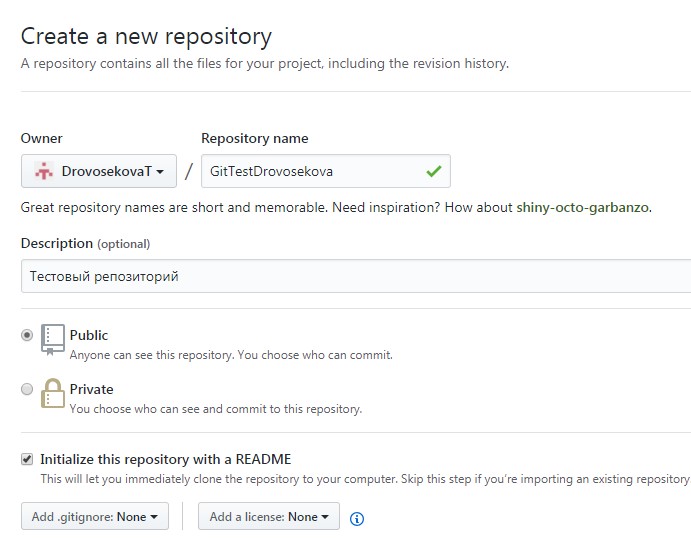
Так выглядит ваш профиль после регистрации. Теперь у вас есть профиль на GitHub.

Сейчас у нас нет ни одного репозитория, и мы можем либо создать новый репозиторий, либо ответвиться (fork) от уже существующего чужого репозитория и вести собственную ветку разработки. Затем, при желании, свои изменения можно предложить автору исходного репозитория

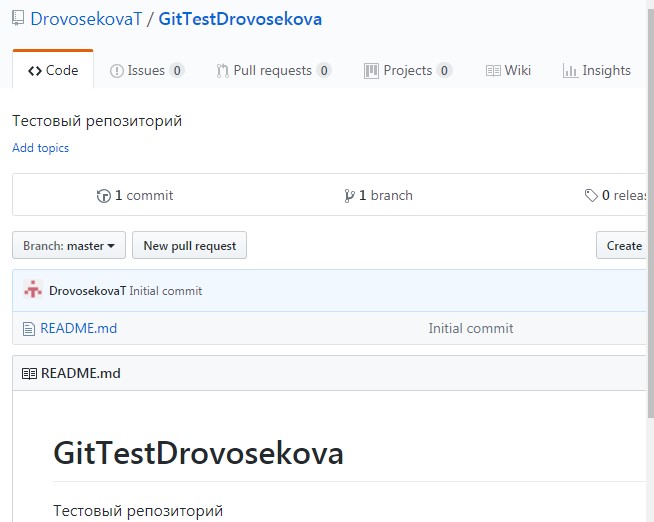
(Pullrequest).

1. Если Вы находитесь в своем профиле, то в верхней части окна видите один из пунктов меню «Repositories 0». Это означает, что у вас в данный момент нет созданных репозиториев. На этой же вкладке есть кнопка «New», которая позволяет создать новый репозиторий. Нажимаем её.
2. Открылось окно создания репозитория. Даем ему имя (например, gittest) можно написать описание. Он должен быть публичным. Также установите галочку там, где предлагается поместить в репозиторий текстовый файл Readme.

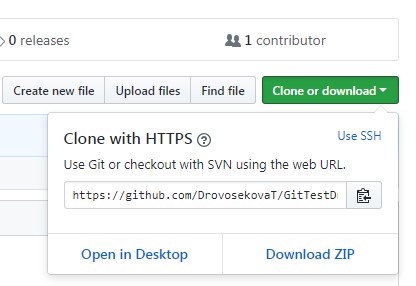
Нажмите кнопку «Create repository».



1. Когда вы создаете новый репозиторий таким образом, на Github появляется папка и для нее выполняется команда gitinit, т.е. происходит то же самое, что мы делали в предыдущих лабораторных работах вручную.
2. После создания репозитория вы в него попадаете и видите, что в нем в данный момент находится один файл – README.md, который был создан автоматически:



1. Далее нам нужно скопировать в буфер обмена адрес нашего репозитория на Github. Для этого нажмите кнопку «Cloneordownload»и в открывшемся окне копку справа от адреса ссылки:

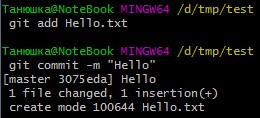


1. Откройте консоль GitBash (она должна быть у вас открыта после лабораторной работы 2, если нет – перейдите в папку tmp, с которой мы работали). Даем команду **git clone**https://github.com/DrovosekovaT/GitTestDrovosekova.git **test**

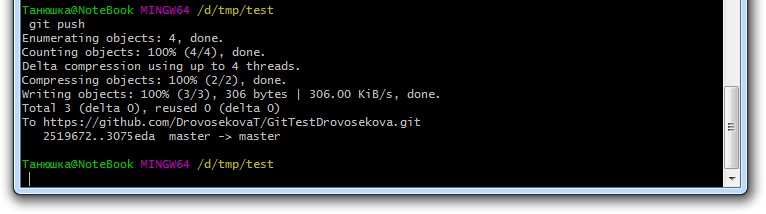
(вставьте из буфера обмена свой адрес репозитория и после него через пробел название папки, где будет создан клон удаленной папки-репозитория, в данном случае test).

Эта команда создаст в папке test копию репозитория с указанного адреса с Github, автоматически Github подключит к локальному репозиторию как удалённый и назовёт его origin, т.е. оригинал (перейдите в папку test и проверьте это командой gitremote -v), автоматически в нашем локальном репозитории создаст локальную ветку master (проверьте это командой gitstatus), которую свяжет с веткой master в удаленном репозитории на Github.

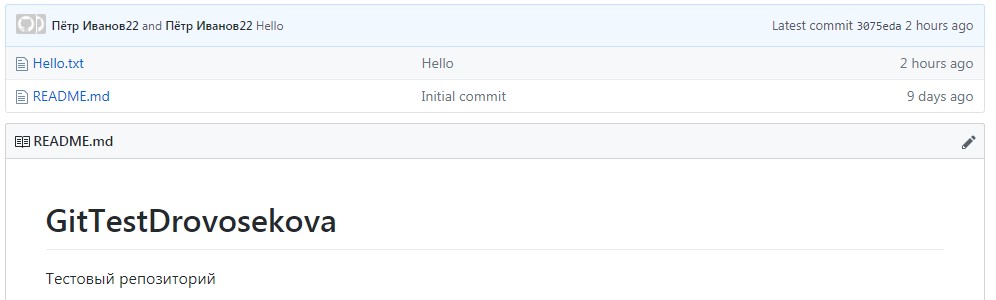
1. Теперь рассмотрим, как синхронизировать локальный репозиторий (клон) с оригиналом, который находится на Github. В клонированной папке test создайте локально файл Hello.txt, напишите в нем текст «Hello!!!», закройте его, сохранив изменения. Теперь у нас репозиторий содержит 2 файла, а на Github только один. Для того, чтобы зафиксировать изменения, выполните команду gitaddHello.txtи создайте коммит:



1. Выполните команду **gitpush**. Эта команда берет имеющиеся локальные изменения и отправляет их в удалённый репозиторий. В ответ на запрос введите своё имя на Github и пароль.



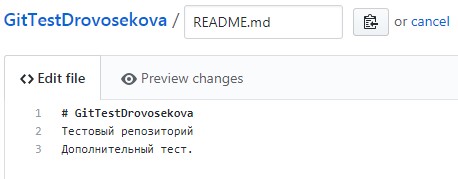
1. Откройте репозиторий на сайте Github, убедитесь, что файл добавился:



Если вы откроете добавившийся файл Hello.txt, то увидите свой текст.

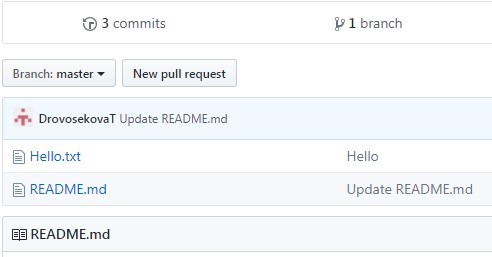
Также вы видите название коммита во втором столбце.

1. Также есть обратная команда. Допустим, у нас на Github появились какие-то изменения, которых нет на локальном компьютере. Для примера откройте и отредактируйте файл README.md. Для этого откройте его на Github и нажмите кнопку с карандашом (Editthisfile), допишите в третьей строке какой-либо текст:

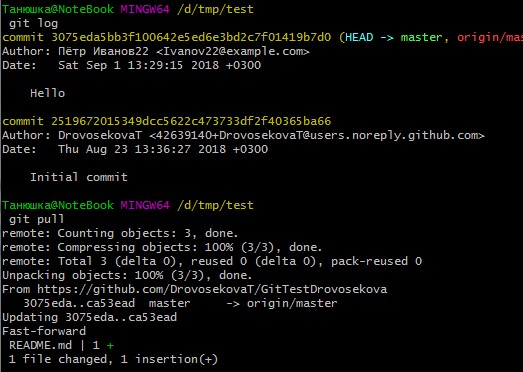


Теперь нажмите кнопку «Commitchanges» в нижней части страницы.

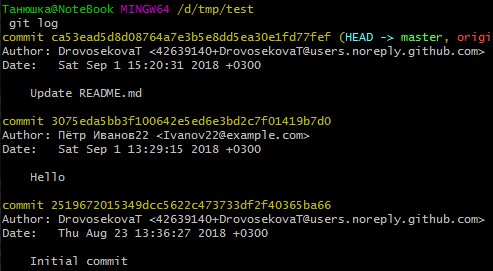
1. Перейдите в папку репозитория, вы увидите, что коммитов стало три.



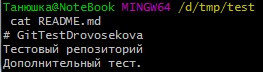
1. Теперь откройте GitBash на вашем локальном компьютере и вызовите команду gitlog. Вы увидите, что коммитов по-прежнему два. Необходимо загрузить изменения с Github.
2. Введите команду gitpull. Эта команда загрузит изменения из удалённого репозитория.



1. Вызовите gitlog и убедитесь, что коммитов стало три.



1. Чтобы посмотреть содержимое обновленного файла, введите команду catREADME.md:



Примечание: Если у вас уже есть репозиторий на компьютере, то вы можетеего клонировать в пустую (без файла README) папку на Github.

**ВЫВОДЫ**:

gitpull – не просто «достает изменения» из удаленного репозитория, но и пытается применить изменения из удалённого репозитория к нашему (влить удалённую ветку в локальную). Ветки будут изучаться далее.

gitpush – отправляет наши наборы изменений на удалёный репозиторий.

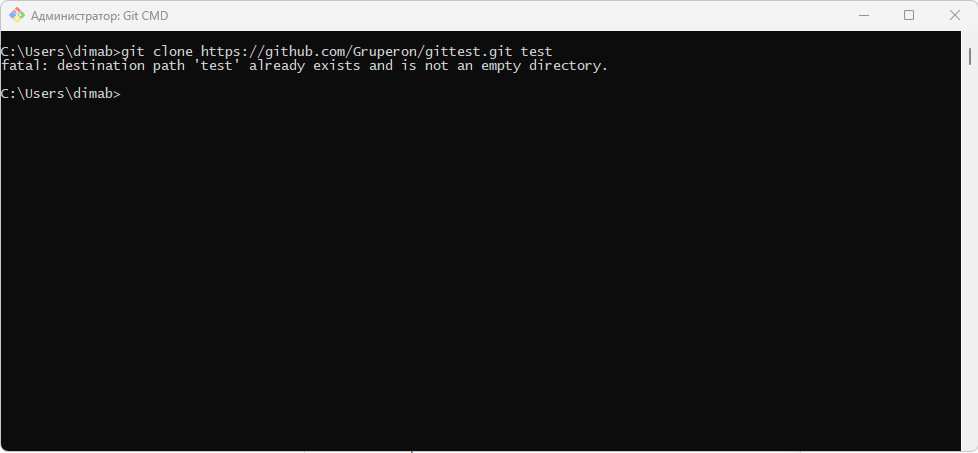
Таким образом, для чего применяется Github в реальной работе?

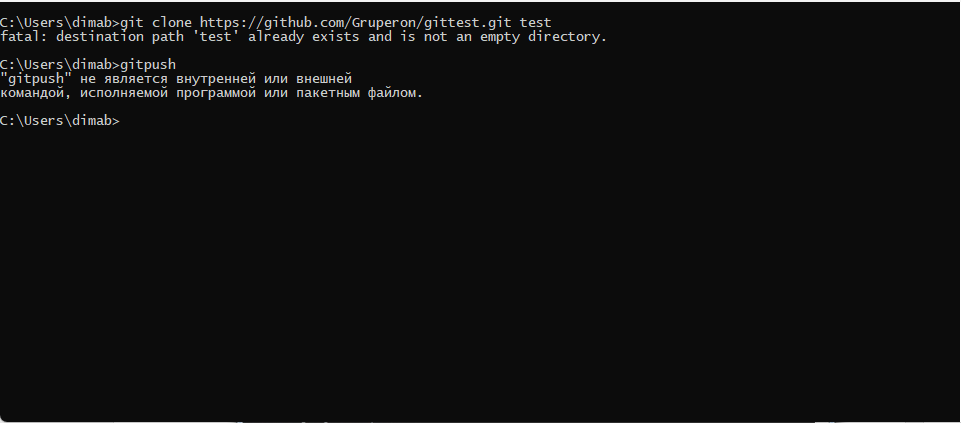
1.Githubочень удобно использовать для быстрого бесплатного создания репозитория в интернете.

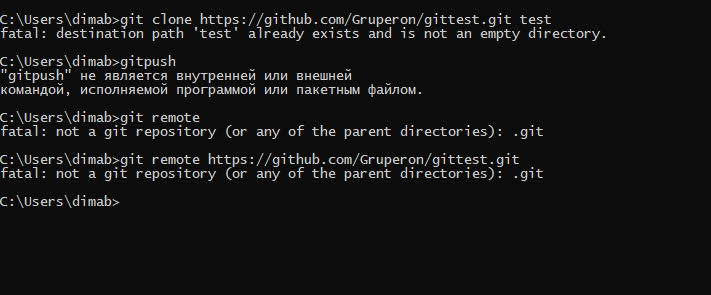
Для чего может понадобиться репозиторий в интернете?

1. Дать кому-то на него ссылку, чтобы этот человек мог удобно посмотреть на код, хранящийся там, удобно посмотрел на историю изменений.
2. Githubможет служить посредником для коллективной работы.
3. Если вы работаете за несколькими компьютерами, можно клонировать на них репозиторий с Github и синхронизировать через него все версии.

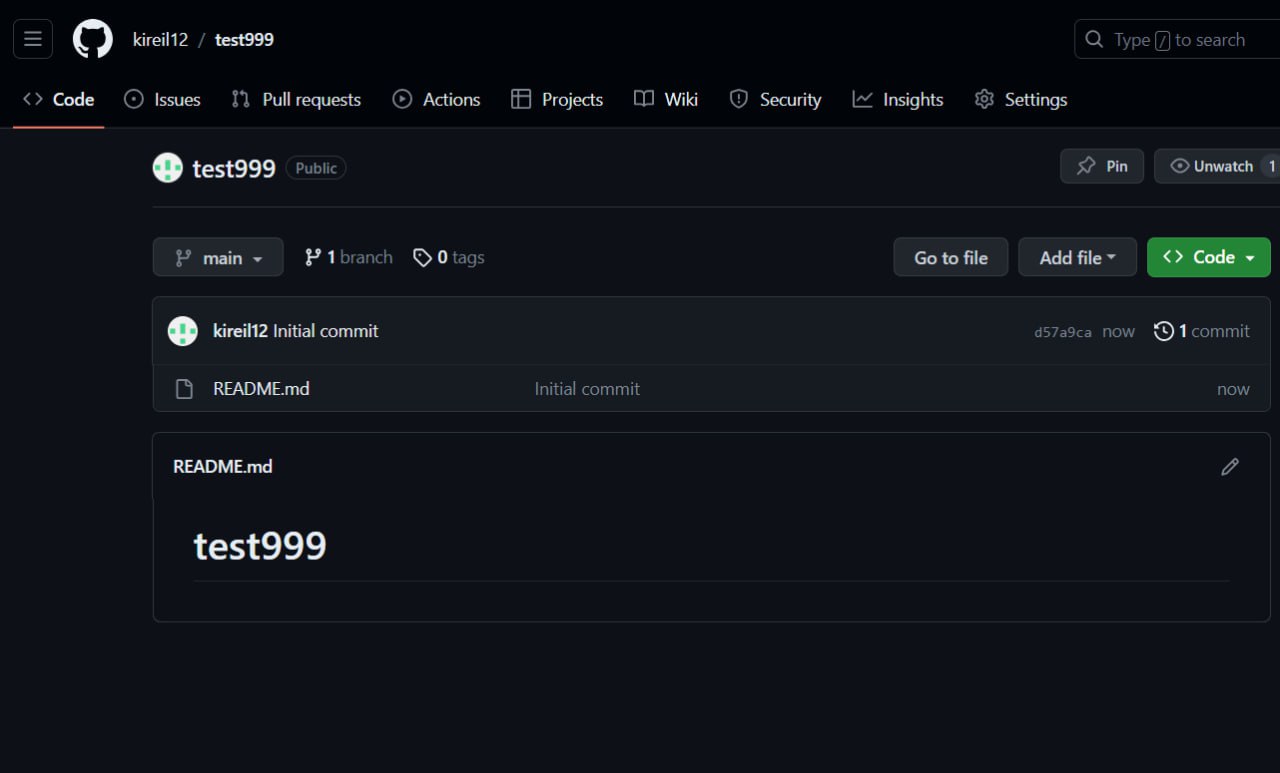
**ДОПОЛНИТЕЛЬНЫЕЗАДАНИЯ**

1. Создайте у себя на компьютере еще один репозиторий ("второй"). Пустой. Свяжите его с уже существующим ("первым") с помощью git remote. Передайте в него изменения из первого. Объясните полученный результат. 
2. Создайте на гитхабе пустой репозиторий ("А"). ВНИМАНИЕ! Не создавайте в нем файл readme!
   * + Установите его в качестве удаленного для репозитория "первый"
     + Передайте из "первого" в "А" изменения
     + Приложите ссылку на "А" к своему ДЗ

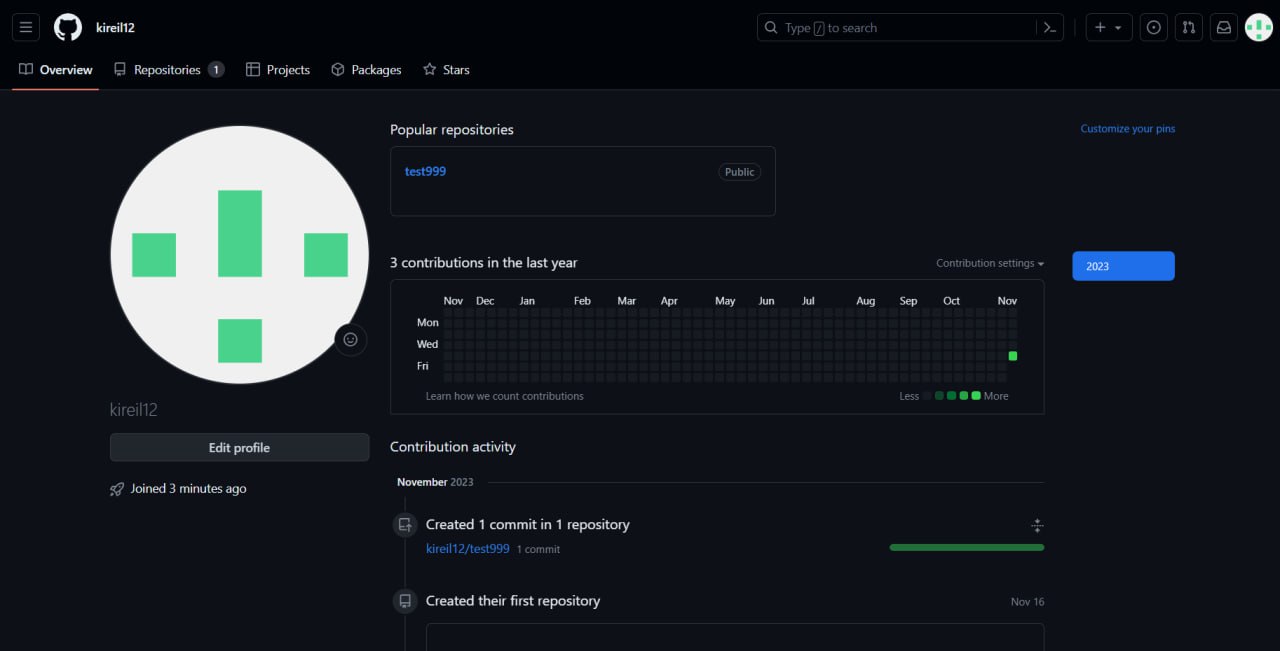


1. Создайте на гитхабе новый непустой (с файлом readme) репозиторий "Б". Склонируйте его к себе ("третий"). Выполните локально несколько коммитов, передайте их в "Б". Приложите ссылку на "Б" 

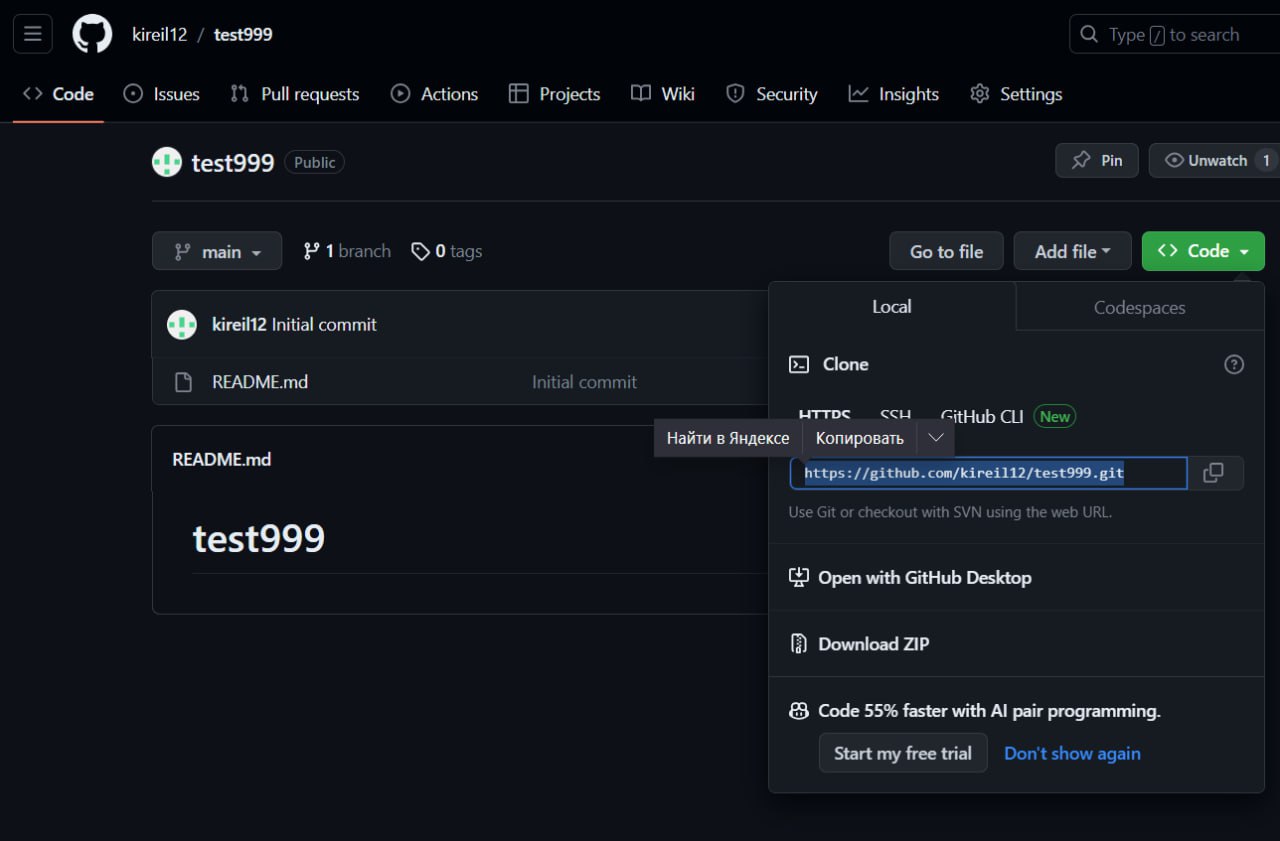
4. Сделайте еще один клон "Б" - "четвертый". Измените один и тот же файл в локальном репозитории "третий", git push, затем в "четвертом" (те же строки, но другие изменения) - и тоже попытка git push. Если возникнет конфликт - попробуйте объяснить его суть.



1.Зарегистрированный профиль



2.Созданный репозиторий



3.Адрес ссылки

Контрольные вопросы

1. Как отправить изменения на удаленный репозиторий?

Чтобы отправить эти изменения ваш удаленный репозиторий, выполните git push origin master. Можно изменить master на любую другую ветвь, чтобы отправить изменения на неё.

1. Как влить изменения в локальную ветку из удаленного репозитория?

для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду git remote. Она выведет названия доступных удалённых репозиториев. Если вы клонировали репозиторий, то увидите как минимум origin — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование

1. Какой командой можно зафиксировать изменения в репозитории?

Командой “git commit” можно зафиксировать изменения в репозитории.

1. Как используется Gihub в реальной работе?

GitHub используется в реальной работе для хранения кода проектов, управления версиями, отслеживания изменений и совместной работы над проектами. Он также предоставляет возможности для обсуждения, оценки и критики кода, что помогает улучшить качество проектов.

1. Опишите принцип коллективной работы с Github

Принцип коллективной работы с GitHub заключается в следующем:

– Каждый участник проекта имеет свой аккаунт на GitHub.  
– Участники проекта совместно работают над проектом, добавляя, удаляя или изменяя файлы в репозитории проекта.  
– Когда участник хочет сохранить изменения, он выполняет команду “git push”, чтобы отправить изменения на сервер GitHub.  
– После отправки изменений сервер GitHub создает новую ветку в репозитории проекта и сохраняет в ней изменения.  
– Другие участники проекта могут просмотреть изменения, выполнив команду “git pull”, чтобы получить последнюю версию проекта с сервера GitHub.  
– Если участник хочет внести изменения в код, который был изменен другим участником, он может создать запрос на слияние (pull request) на GitHub, чтобы получить изменения от других участников проекта.