Для начала установим git, выбирая соответствующие настройки

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Зарегистрируемся на гитхабе

Инструкция по созданию репозитория на Github

1. Регистрируемся на сайте: github.com под именем nikname (может быть любым другим).   
   
2. Нажимаем кнопочку "+" и вводим название репозитория.   
   
3. Выбираем тип Public (Private доступен только в платной версии).   
   
4. Нажимаем Create.   
В результате создан репозиторий в Github (на экране видим инструкцию, по соедининению локального репозитория со вновь созданным).   
   
5. Добавляем удаленный репозиторий (по протоколу SSH) под именем origin (желательно использовать его, но можно любое другое, главное – не master – оно используется в момент релиза версии).

Изображение выглядит как текст

Автоматически созданное описание

После установки Git  можно его запустить в нужной папке, щелкнув ее правой кнопкой мыши и выбрав пункт “Git Bash Here”



Изображение выглядит как текст

Автоматически созданное описание

Выполняем команду get init - то есть инициализируем репозиторий. В рабочей папке появится папка .git, что означает, что репозиторий инициализирован.



 Для того чтобы начать отслеживать (добавить под версионный контроль) новый файл, используется команда git add.

Склонировать удаленный репозиторий в  локальную папку можно с помощью команды clone

git clone git://github.com/schacon/grit.git

Коммит (commit) — это информация об измененных файлах. Коммит состоит из автора коммита, измененных файлов, HEAD и времени. Для примеров мы будем использовать репозиторий и сделаем первый коммит, который отправим на сервер. Он нужен для того, чтобы разграничивать задачки. Так будет понятно, какой код в истории относится к той или иной задаче, чтобы потом мы могли быстро понять суть.  С помощью команды git commit мы сохраним содержимое индекса как неизменяемый снимок в области HEAD. Хорошей практикой считается коротко описать, что было изменено, для этого используется флаг -m

При помощи команды git log в консоли мы можем отслеживать историю коммитов в ветке.

Если коммиты еще не опубликованы в удаленном репозитории, таком как GitHub, то можно  по существу удалить предыдущие коммиты с помощью команды reset. Иначе, если коммиты уже были опубликованы в удаленном репозитории, то рекомендуется использовать команду revert. Эта команда работает, отменяя изменения, которые были внесены в указанный коммит, создавая новый коммит и фактически не удаляя предыдущие коммиты. Это идеально для опубликованных изменений, потому что тогда реальная история репозитория сохраняется

**Контрольные вопросы**

**Опишите своими словами значение следующих терминов:**

**Рабочий каталог**- каталог файловой системы, который используется для нахождения файлов, указанных только по имени либо по относительному пути.

Рабочая директория Git - это директория, которая содержит в себе то, с чем пользователь работает или то, что он извлекает из истории проекта в данный момент. Файлы в этой директории часто удаляются или изменяются Git-ом когда пользователь переключается между ветками.   Вся история  проекта хранится в директории Git; рабочая директория - это просто временное место, где пользователь можете модифицировать файлы, а затем выполнить коммит.

**Репозиторий**-  это все файлы, находящиеся под контролем версий, вместе с историей их изменения и другой служебной информацией.

Репозиторий Git можно создать, либо выбрав любую папку на компьютере, либо клонировав себе уже существующий репозиторий.

**Коммит -**это зафиксированный набор изменений, который показывает, какие файлы изменились и что именно в них изменилось.

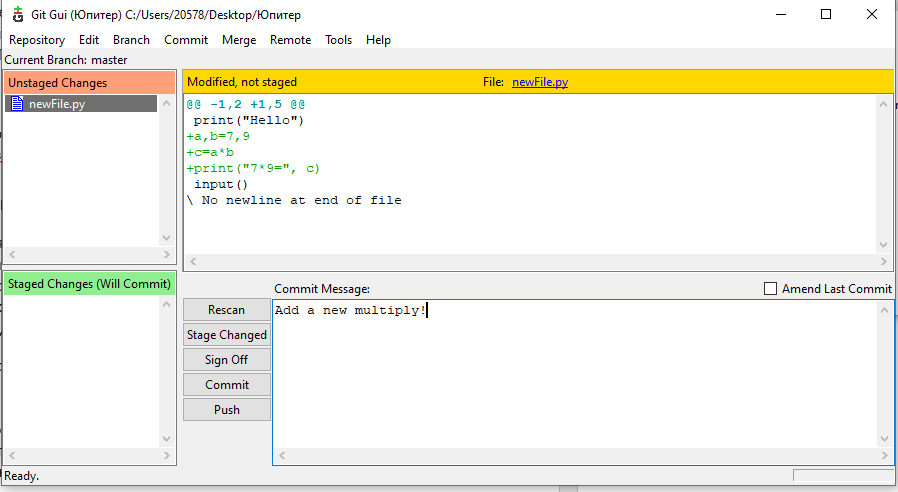
**Ветка в  Git**-  это подвижный указатель на один из коммитов. Обычно ветка указывает на последний коммит в цепочке коммитов. Ветка берет свое начало от какого-то одного коммита. 

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

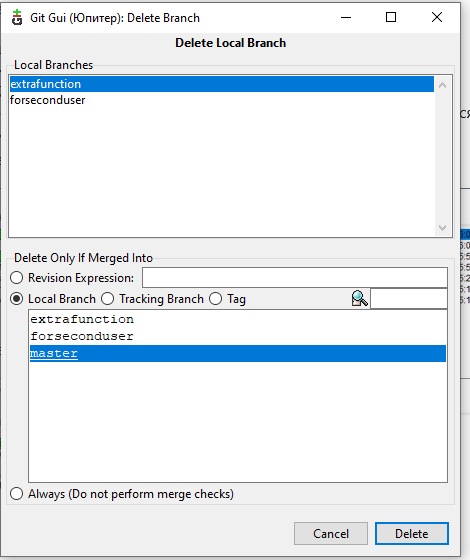
Автоматически созданное описание

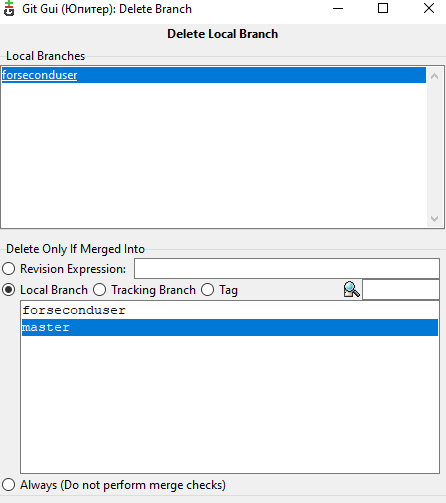
Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание





Изображение выглядит как текст

Автоматически созданное описание

**Скрины проделанной работы в графическом режиме**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, монитор, черный

Автоматически созданное описание



Изображение выглядит как текст, монитор, снимок экрана, экран

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, монитор, черный

Автоматически созданное описание

Изображение выглядит как текст, монитор, снимок экрана

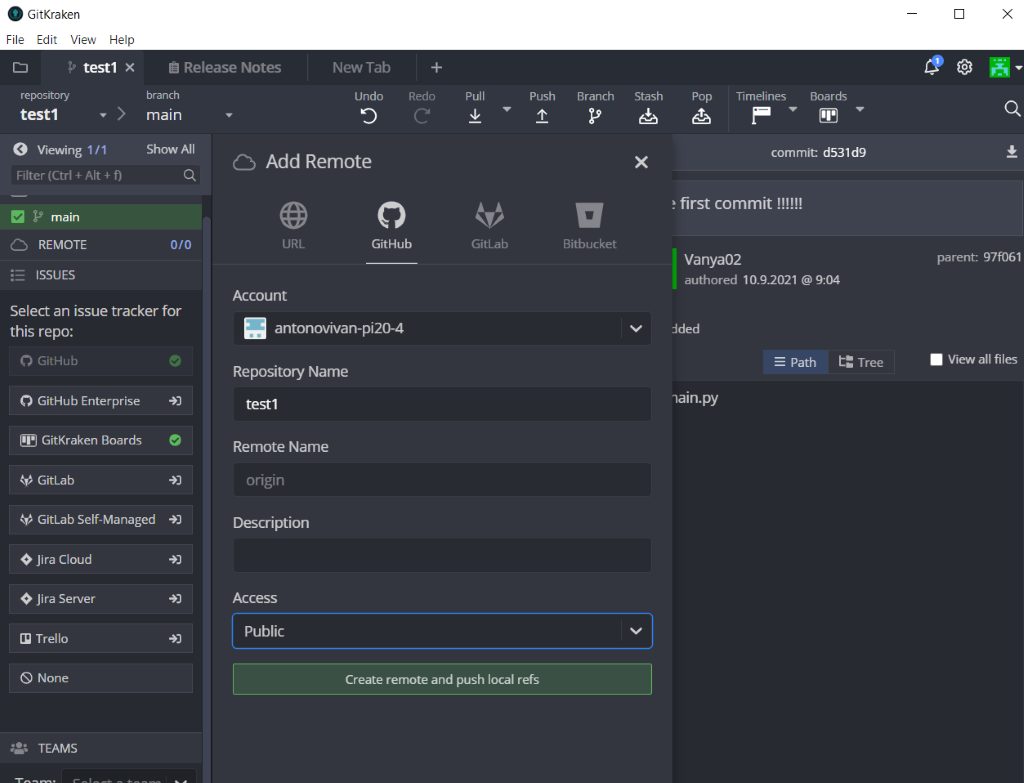
Автоматически созданное описание

Изображение выглядит как текст, монитор, снимок экрана, экран

Автоматически созданное описание

Изображение выглядит как текст, монитор, снимок экрана, экран

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание

Скрины по проделанной работе в терминале и с удаленными репозиториями

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

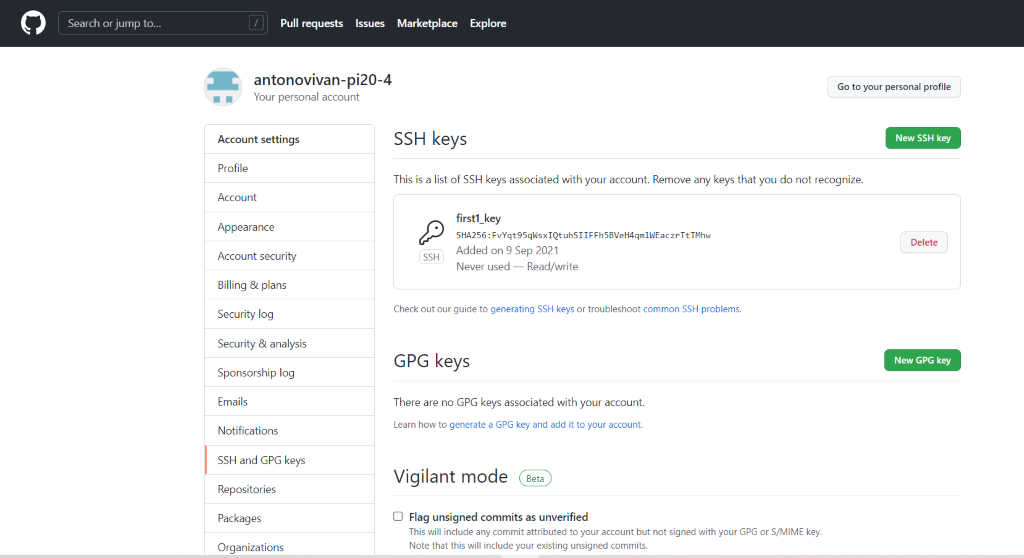
Автоматически созданное описание

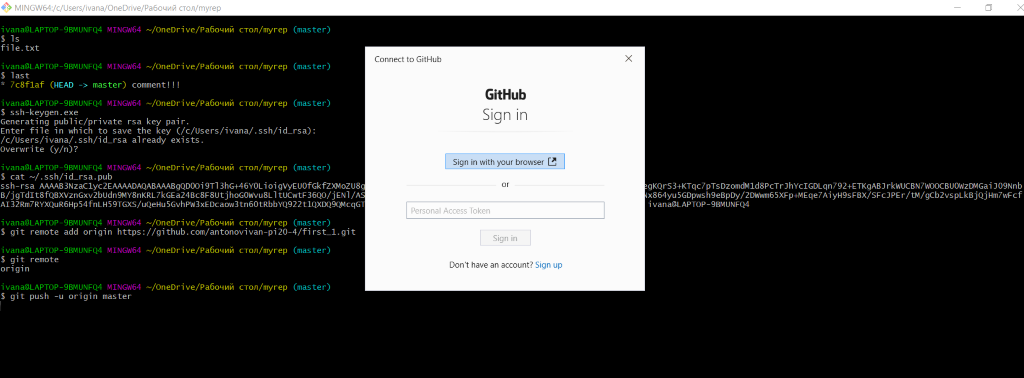
Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание





Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, внутренний

Автоматически созданное описание

Контрольные вопросы

1. Что такое удаленный репозиторий?

Удалённые репозитории — это модификации проекта, которые хранятся в интернете или ещё где-то в сети. Их может быть несколько, каждый из которых, как правило, доступен для вас либо только на чтение, либо на чтение и запись.

1. Где нужно вводить команды git?

Команды git вводятся в терминале git bash

1. Для чего нужны ветки в системах контроля веток?

Ветки позволяют командам разработчиков без труда вести совместную работу с одной централизованной базой кода. Когда разработчик создает ветку, система контроля версий создает копию базы кода, актуальную на текущий момент времени.

1. Как возникают конфликты слияния?

Системы контроля версий предназначены для управления дополнениями, вносимыми в проект множеством распределенных авторов (обычно разработчиков). Иногда один и тот же контент могут редактировать сразу несколько разработчиков. Если разработчик A попытается изменить код, который редактирует разработчик B, может произойти конфликт. Для предотвращения конфликтов разработчики работают в отдельных изолированных ветках.

 Слияние и конфликты являются неотъемлемой частью работы с Git. В других инструментах управления версиями, например SVN, работа с конфликтами может быть дорогой и времязатратной. Git позволяет выполнять слияния очень просто. В большинстве случаев Git самостоятельно решает, как автоматически интегрировать новые изменения.

Обычно конфликты возникают, когда два человека изменяют одни и те же строки в файле или один разработчик удаляет файл, который в это время изменяет другой разработчик. В таких случаях Git не может автоматически определить, какое изменение является правильным. Конфликты затрагивают только того разработчика, который выполняет слияние, остальная часть команды о конфликте не знает. Git помечает файл как конфликтующий и останавливает процесс слияния. В этом случае ответственность за разрешение конфликта несут разработчики.

1. Как разрешать конфликты слияния?

Самый простой способ разрешить конфликт — отредактировать конфликтующий файл.

После редактирования файла выполнить команду git add merge.txt, чтобы добавить новое объединенное содержимое в раздел проиндексированных файлов. Для завершения слияния создать новый коммит, выполнив следующую команду:

git commit -m "merged and resolved the conflict in merge.txt" ( merge.txt-имя файла)

Git обнаружит, что конфликт разрешен, и создаст новый коммит слияния для завершения процедуры слияния.

Команды Git, с помощью которых можно разрешить конфликты слияния

Общие инструменты

git status

Команда status часто используется во время работы с Git и помогает идентифицировать конфликтующие во время слияния файлы.

git log --merge

При передаче аргумента --merge для команды git log будет создан журнал со списком конфликтов коммитов между ветками, для которых выполняется слияние.

git diff

Команда diff помогает найти различия между состояниями репозитория/файлов. Она полезна для выявления и предупреждения конфликтов слияния.

Инструменты для случаев, когда Git прерывает работу в самом начале слияния

git checkout

Команда checkout может использоваться для отмены изменений в файлах или для изменения веток.

git reset --mixed

Команда reset может использоваться для отмены изменений в рабочем каталоге или в разделе проиндексированных файлов.

Инструменты для случаев, когда конфликты Git возникают во время слияния

git merge --abort

При выполнении команды git merge с опцией --abort процесс слияния будет прерван, а ветка вернется к состоянию, в котором она находилась до начала слияния.

git reset

Команду git reset можно использовать для разрешения конфликтов, возникающих во время выполнения слияния, чтобы восстановить заведомо удовлетворительное состояние конфликтующих файлов.

Контрольные вопросы

1. Зачем нужен облачный хостинг репозиториев?

Облачный хостинг это тип веб-хостинга, который использует несколько разных серверов для того, чтобы распределять нагрузку и максимально увеличивать время безотказной работы.

Основные преимущества облачного хостинга в том, что основное внимание направлено на увеличение времени безотказной работы, выделенные ресурсы, простое масштабирование и выделенный IP-адрес.

1. Какими основными функциями обладает сайт github.com?

GitHub — сервис онлайн-хостинга репозиториев, обладающий всеми функциями распределённого контроля версий и функциональностью управления исходным кодом — всё, что поддерживает Git и даже больше. Также GitHub может похвастаться контролем доступа, багтрекингом, управлением задачами и вики для каждого проекта.

1. Как организовать командную работу над открытым проектом?

Управление проектом — это то, что делают команды для создания рабочих процессов и достижения целей проекта, держа в уме все ограничения по ресурсам или времени

В процессе командной работы над открытым проектом каждый разработчик использует команды pull и push, обращаясь к origin. Помимо централизованных отношений pull-push, каждый разработчик может также извлекать изменения от других членов команды. Например, это может быть полезно для совместной работы с 2 или более разработчиками над большой новой функциональностью. Также они не смогут отправить origin команду push преждевременно.