

## Лабораторная работа 2. Система контроля версий Git

Цель работы: научиться управлять версиями проекта с помощью системы Git; освоить работу с репозиториями GitHub.

Git – распределённая система контроля версий, включающая в себя набор консольных программ, нацеленных на фиксирование и отслеживание изменений любых файлов.

Git используется разработчиками по всему миру, оптимизируя и упрощая разработку внутри команды. Работая с VCS GIT вы способны подключаться к удаленным репозиториям (например: github) из вашей рабочей папки.

После успешной синхронизации вам предоставляется возможность скачать репозиторий с github локально на компьютер либо выгрузить данные на удаленный git-репозиторий. Таким образом git является необходимым инструментом для разработчика, облегчая работу над конкретным проектом для нескольких людей.

### ***Основной список git-команд:***

*git init* – Инициализация репозитория

*git status* – Просмотр состояния

*git clone [ссылка на удаленный git репозиторий]* – Скачивание git-репозитория

*git commit -m 'сообщение'* – Фиксация изменений

*git add .* – Добавить все

*git add -A* – Добавить все

*git add [файл с расширением]* – Добавить конкретный файл

*git push origin master* – Отправка изменений в ветку master

*git pull origin master* – Принятие изменений из ветки master

*git checkout [git ветка]* – Перейти на ветку

*git branch* – Посмотреть все имеющиеся ветки

*git branch [git ветка]* – Создать ветку

*git merge [git ветка]* – Слияние веток

*git rm [файл]* – Удаление файла

*git push origin HEAD* – Отправить текущую ветку на удаленный git-репозиторий, не вводя ее название

*git push origin* – Отправить все ветки на удаленный git-репозиторий

*git branch -d [git ветка]* – Удалить ветку (после git merge)

*git branch -D [git ветка]* – Просто удалить ветку (игнорируя git merge)

*git push origin :[git ветка]* – Удалить ветку в удаленном git-репозитории

*git reset --hard d8578edf8458ce06fbc5bb76a58c5ca4a58c5ca4* – Жесткий откат к конкретному коммиту (хэш смотрим в «git log»)

*git reset --soft d8578edf8458ce06fbc5bb76a58c5ca4a58c5ca4* – Мягкий откат к конкретному коммиту (хэш смотрим в «git log»)

*git remote add origin [ссылка на удаленный git репозиторий]* – Подключится к удаленному git репозиторию

## **Установка**

Установка git для Windows.

Рекомендуется официальный дистрибутив, содержащий в себе графическую оболочку и консоль. Скачать git-клиент можно здесь: <https://git-scm.com/download/win>

Установка git для Linux.

Откройте терминал и введите команду *sudo apt-get install git*

Установка git для MAC на OS X.

Графический инсталлятор. Скачать git-клиент можно здесь: <http://sourceforge.net/projects/git-osx-installer/>.

## **Настройка**

Перед началом работы настоятельно рекомендуется произвести инициализацию Вас как пользователя.

```
git config --global user.name "Ваше имя"  
git config --global user.email ваш@mail.ru
```

После указания имени и email ваши данные станут публичными. Все ваши действия будут помечены именем и почтой.

### ***Git init – Создание git-репозитория***

Для инициализации репозитория в консоли необходимо ввести команду `git init`, предварительно выбрав директорию для работы.

Для этой цели создайте папку `site6.local` в корне локального сервера `open server`. Путь до рабочей папки выглядит следующим образом: `C:/OpenServer/domains/site6.local`. Вы можете создать директорию в любом месте и назвать по своему желанию.

Откройте git-консоль и перейдите в рабочую папку:

```
cd C:/OpenServer/domains/site6.local
```

#### ***Примечание!***

В операционной системе Windows, кликнув внутри рабочей папки правой кнопкой мыши, в зависимости от установленного git ПО, вы можете вызвать git-консоль моментально из текущей директории без надобности указания пути.

Как уже упоминалось ранее, для инициализации git-репозитория вводим в терминал команду:

```
git init
```

В ответ мы должны получить `Initialized empty Git repository in [путь]` – это означает, что git-репозиторий был успешно инициализирован и в корне проекта была создана скрытая папка с названием `.git`, в которой хранятся git-настройки и прочие служебные файлы.

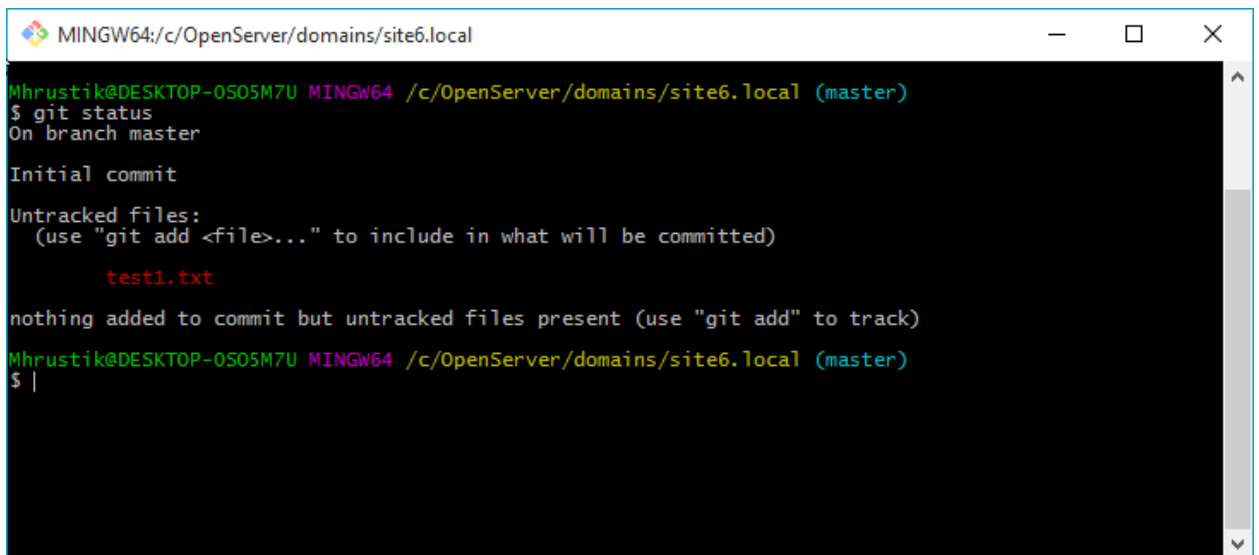
### ***Git status – Определение состояния***

Создадим обычный текстовый файл `test1.txt` с любым содержимым (по желанию).

Вводим в git-консоль команду:

```
git status
```

В ответ мы получим сообщение, представленное на рисунке 1.



```
MINGW64:/c:/OpenServer/domains/site6.local
Mhrustik@DESKTOP-0S05M7U MINGW64 /c:/OpenServer/domains/site6.local (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test1.txt

nothing added to commit but untracked files present (use "git add" to track)
Mhrustik@DESKTOP-0S05M7U MINGW64 /c:/OpenServer/domains/site6.local (master)
$ |
```

Рисунок 1 – Git-статус

Нам необходимо добавить файл и произвести commit для фиксации изменений и указания сообщения (Например: Был создан текстовой файл).

### ***Git add – Подготовка файлов***

Пропишем в терминале git-команду добавления конкретного файла:

```
git add test1.txt
```

Для добавления всех файлов вы можете использовать команды:

```
git add .
git add -A
```

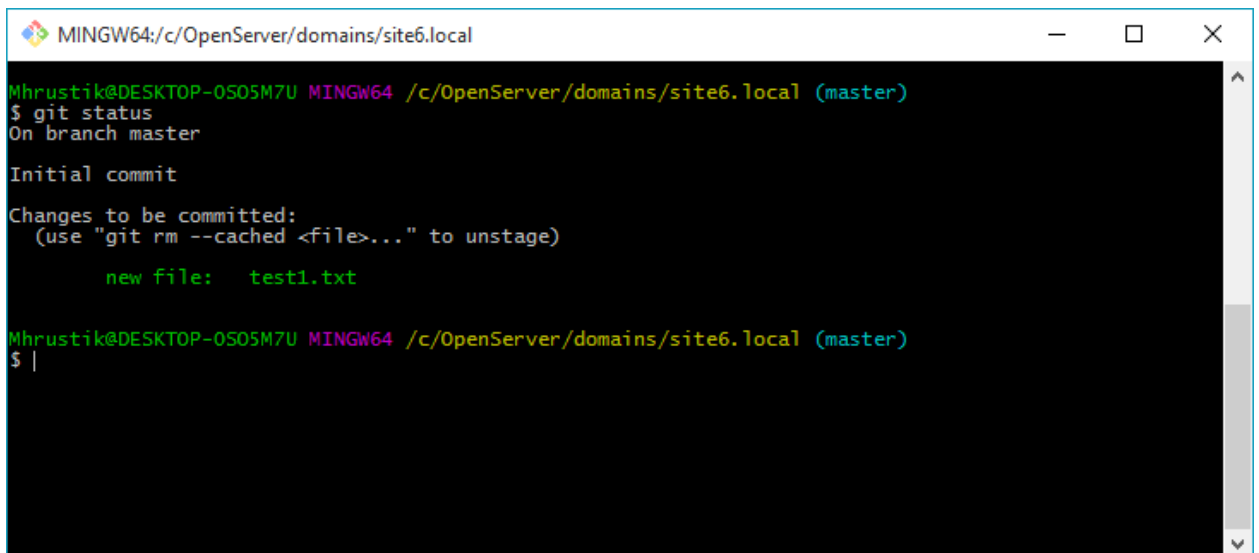
### ***Примечание!***

При всех последующих изменениях файла test1.txt вам каждый раз необходимо добавлять файл заново.

Проверим статус командой:

```
git status
```

Теперь сообщение выглядит иначе (рисунок 2).



```
MINGW64:/c:/OpenServer/domains/site6.local
Mhrustik@DESKTOP-0S05M7U MINGW64 /c:/OpenServer/domains/site6.local (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test1.txt

Mhrustik@DESKTOP-0S05M7U MINGW64 /c:/OpenServer/domains/site6.local (master)
$ |
```

Рисунок 2 – Изменённый Git-статус

### ***Git commit – Фиксация изменений***

Коммиты в Git необходимы для отслеживания изменений. Если Вы делаете commit с указанием ключа -m "в котором даете понять какие изменения вы внесли", то все коммиты моментально вносятся в историю и доступны для публичного просмотра. Более того git-коммиты содержат ключи, обращаясь к которым, Вы с легкостью можете откатиться до нужного вам состояния.

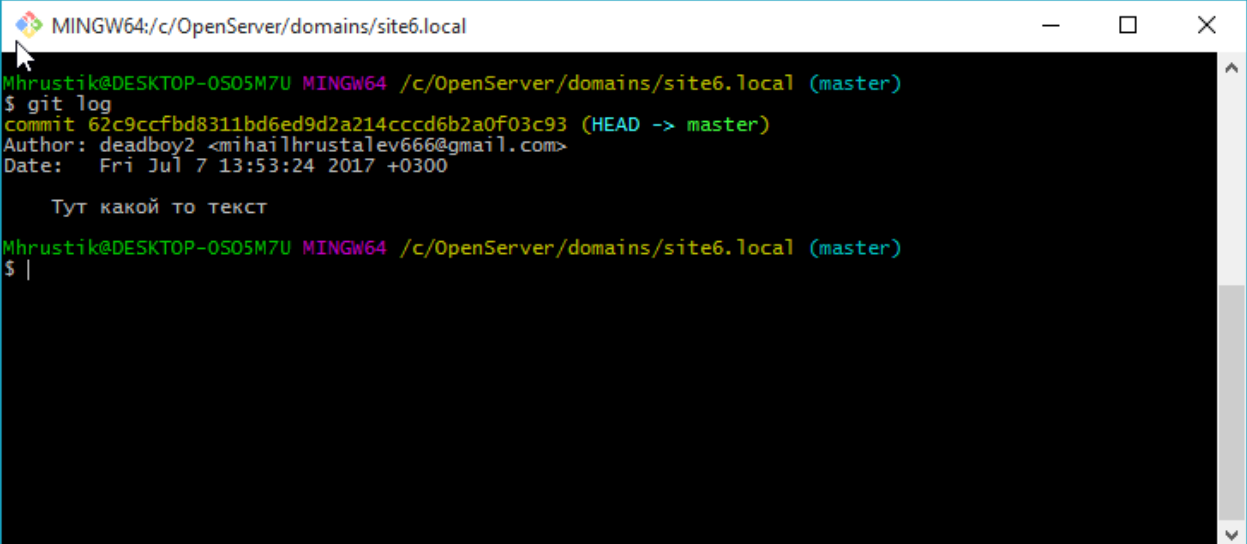
После добавления файла командой *git add* необходимо произвести фиксацию изменений командой:

*git commit -m "сообщение"*

Для просмотра истории коммитов достаточно ввести команду:

*git log*

В результате получаем ответ, представленный на рисунке 3.

A screenshot of a terminal window titled 'MINGW64:/c:/OpenServer/domains/site6.local'. The terminal shows a user running the command 'git log'. The output displays a commit hash '62c9ccfbd8311bd6ed9d2a214cccd6b2a0f03c93' in yellow, followed by '(HEAD -> master)' in green. Below this, the author 'deadboy2 <mihailhrustalev666@gmail.com>' and the date 'Fri Jul 7 13:53:24 2017 +0300' are shown. The commit message 'Тут какой то текст' is displayed in white. The prompt '\$' is visible at the bottom of the terminal.

```
MINGW64:/c:/OpenServer/domains/site6.local
Mhrustik@DESKTOP-0S05M7U MINGW64 /c:/OpenServer/domains/site6.local (master)
$ git log
commit 62c9ccfbd8311bd6ed9d2a214cccd6b2a0f03c93 (HEAD -> master)
Author: deadboy2 <mihailhrustalev666@gmail.com>
Date: Fri Jul 7 13:53:24 2017 +0300

    Тут какой то текст

Mhrustik@DESKTOP-0S05M7U MINGW64 /c:/OpenServer/domains/site6.local (master)
$ |
```

Рисунок 3 – Git-лог

В данном ответе содержится идентификатор состояния, автор, дата создания git-коммита.

#### *Примечание!*

С каждым последующим изменением в файле или добавлением новых вы должны запомнить последовательность действий:

*git add (добавляем файлы) -> git commit -m "" (Фиксируем состояние).*

#### ***Git reset – Откат изменений***

Если содержимое текстового файла test1.txt пустое, запишем в него: hello world!!!

После успешного сохранения документа вводим в консоль:

```
git status
git add .
git commit -m "Написали привет мир!"
git log
```

Скопируйте полностью хэш самого первого состояния в формате: *def8d5cad2193801239d08f5fa3d1d6b1ed033f1*.

Для отката изменений к конкретному коммиту выполним команду:

```
git reset --soft def8d5cad2193801239d08f5fa3d1d6b1ed033f1
```

#### ***Важно!***

Откат изменений с ключом *--soft* не удаляет состояния, а лишь смещает индекс. Наиболее рекомендуемый способ для отката.

Откат изменений с ключом *--hard* удаляет полностью все состояния и индексы.

### ***Gitignore – игнорирование файлов и папок***

Создадим папку с названием `ignore_files`. Внутри папки создадим текстовый файл `file.txt`. Если по каким-либо причинам требуется игнорировать данную папку и не добавлять ее в git-репозиторий, необходимо в корне проекта создать `gitignore`-файл.

Если система ругается на создание файла с именем `.gitignore`, создайте простой текстовый документ и пересохраните его с именем `.gitignore`.

Откроем файл `gitignore` в текстовом редакторе и добавим название папки, которую хотим исключить: `ignore_files` (рисунок 4).

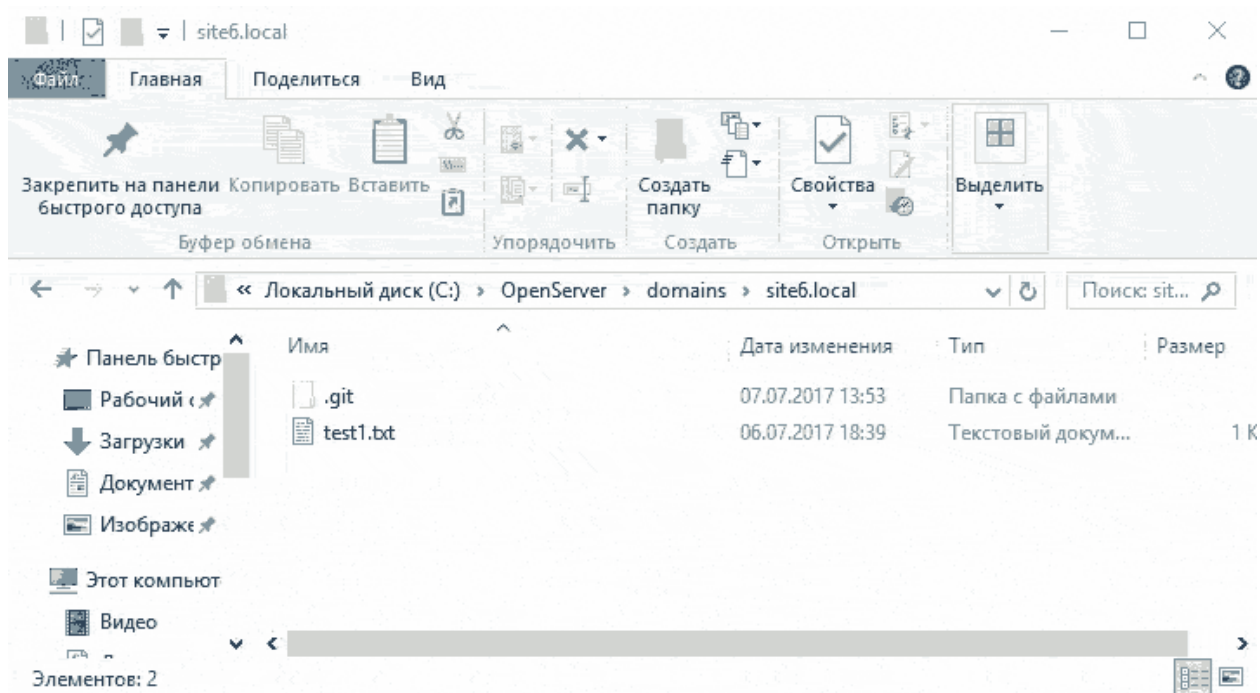
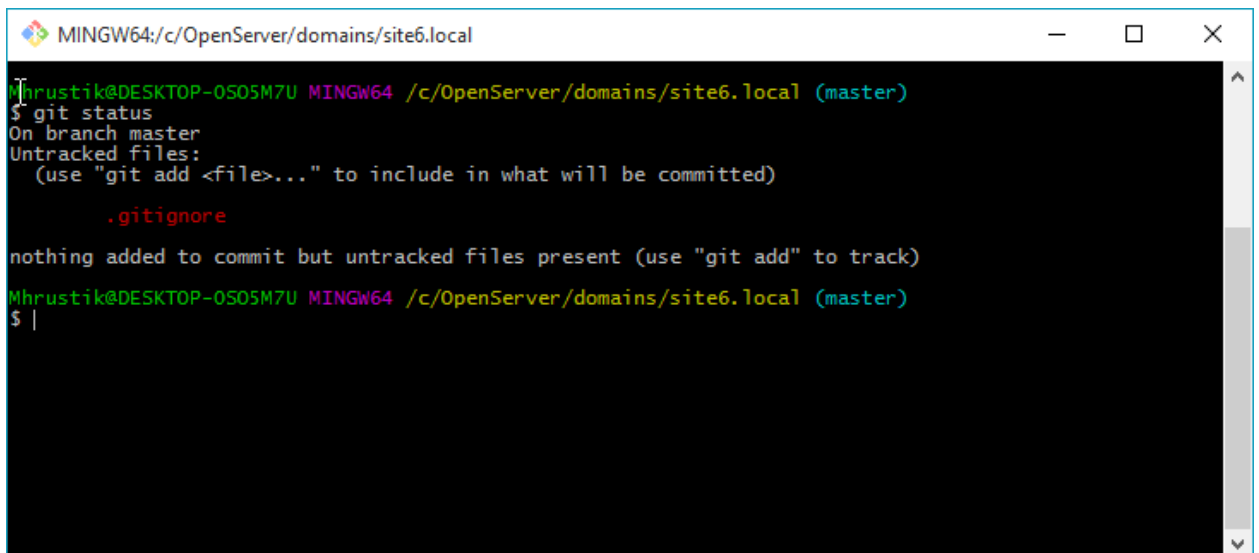


Рисунок 4 – Gitignore

Сохраняем документ и командой `git status` проверяем (рисунок 5).



```
MINGW64:/c:/OpenServer/domains/site6.local
Mhrustik@DESKTOP-0S05M7U MINGW64 /c:/OpenServer/domains/site6.local (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
Mhrustik@DESKTOP-0S05M7U MINGW64 /c:/OpenServer/domains/site6.local (master)
$ |
```

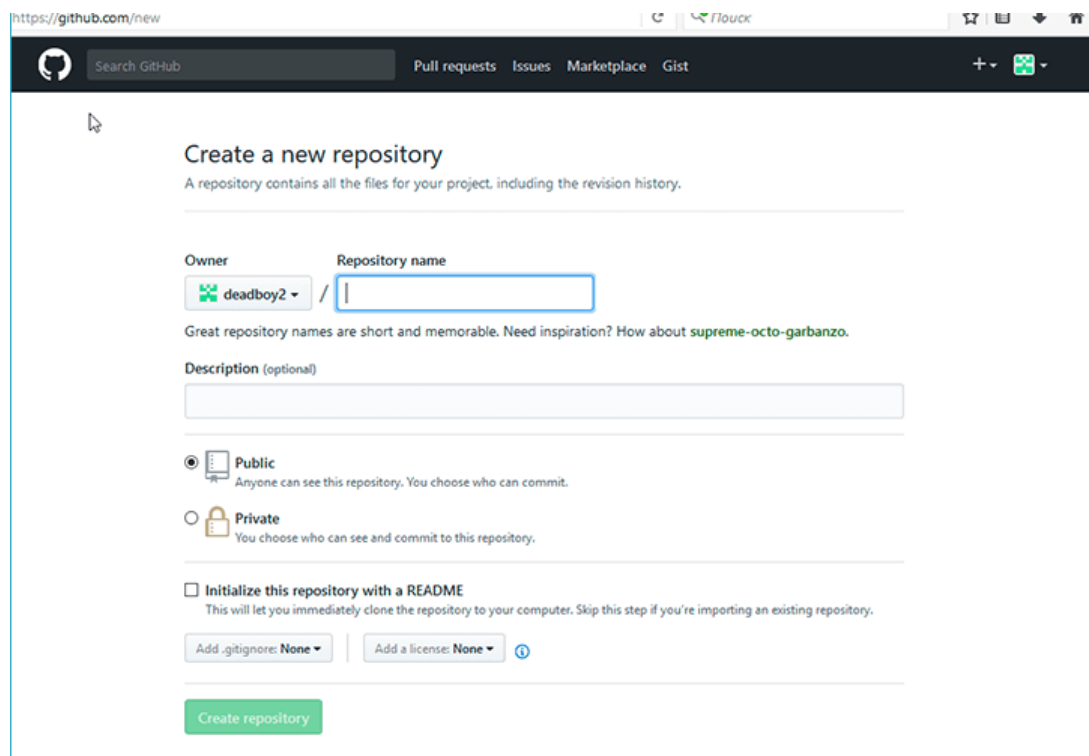
Рисунок 5 – Проверка статуса после выполнения команды gitignore

Если все сделано правильно, в ответ не увидим папки которую исключили.

### *Работа с удаленными git-репозиториями*

Использование Git в реальных условиях почти всегда означает работу с удаленными репозиториями. Перед подключением и отправкой данных в git-репозиторий, необходимо создать учетную запись на [github.com](https://github.com).

Создадим новый репозиторий на github (рисунок 6).



https://github.com/new

Search GitHub Pull requests Issues Marketplace Gist

### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: deadboy2 / Repository name: [input field]

Great repository names are short and memorable. Need inspiration? How about [supreme-octo-garbanzo](#).

Description (optional): [input field]

☒ Public  
Anyone can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

☐ Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: None

Create repository

Рисунок 6 – Создание нового репозитория



1. Задаем имя.
2. Описание (по желанию).
3. Оставляем репозиторий в публичном доступе.
4. Создаем репозиторий.
5. Копируем ссылку на репозиторий.

Переходим в рабочую папку локального репозитория и вводим в консоль команду:

```
git remote add origin https://github.com/deadboy2/test.git
```

Теперь локальный git-репозиторий успешно связан с удаленным.

### ***Отправляем данные в удаленный git-репозиторий***

После успешного подключения к репозиторию на GitHub можно отправлять данные командой:

```
git push origin master,
```

где

origin – имя репозитория по умолчанию;

master – ветка, в которую мы хотим отправить данные (по умолчанию master – главная ветка).

После выполнения команды *git push* в удаленном репозитории Вы должны увидеть все ваши данные. Обратите внимание, что игнорированные файлы не попали в репозиторий на GitHub.

### ***Принимаем данные из удаленного git-репозитория***

Поскольку изменений в удаленный репозиторий никто не вносил, принимать нам нечего. Попробуйте создать новый файл или внести правки в уже имеющийся затем введите команду:

```
git pull origin master
```

### ***Git clone – Клонирование репозитория***

Для скачивания git-репозитория из GitHub достаточно ввести команду:

```
git clone https://github.com/deadboy2/test.git
```

## ***Git branch – работа с ветками***

Представьте, что вы являетесь разработчиком плагина, у которого со временем будут выходить важные обновления. Чтобы не затрагивать старую версию, разумнее создать новую. Итог: две ветки с разными версиями. Кроме основных официальных версий могут быть и подверсии. С git-ветками можно производить манипуляции, а именно: создание, слияние, удаление.

### ***Создание git-ветки***

В терминале введем команду на создание новой git-ветки:

```
git branch 1.0
```

Для просмотра всех имеющихся веток вводим команду:

```
git branch
```

В ответ получаем имена веток где \*звездочкой указана текущая активная ветвь.

Для перехода на ветвь с именем 1.0 необходимо ввести команду:

```
git checkout 1.0
```

Теперь создайте в корне проекта папку 1.0 внутри которой текстовый файл с именем 1.0 и внесите в него любое содержимое. На этом этапе вы должны находиться на ветке с именем 1.0.

Добавьте файлы:

```
git add .
```

Зафиксируйте изменения:

```
git commit -m "Создали папку и файл с именем 1.0 в новой git ветке 1.0"
```

Переключитесь на ветку master командой:

```
git checkout master
```

Наблюдайте за происходящим в корне проекта, выполняя переходы с одной ветки на другую.

## ***Слияние и удаление git-веток***

Чтобы произвести слияние ветки master и 1.0 нужно переключиться на ветку master:

*git checkout master*

Командой git merge производим слияние двух git-веток:

*git merge 1.0*

Затем git-ветку с именем 1.0 можно удалить командой:

*git branch 1.0 -D*

## ***Разрешение конфликтов при слиянии git-веток***

Git конфликты возникают довольно часто, когда несколько разработчиков работают в разных ветках. К примеру, Вы пишете код в ветке master в файле index.txt, а другой разработчик пишет в этом же файле, но уже в иной git-ветке под названием test. Рассмотрим на примере.

Файл index.txt в ветке master с содержимым:

*hello из ветки мастер*

Файл index.txt в ветке test с содержимым:

*hello из ветки тест*

Как только произойдет слияние двух веток, произойдет исключение, и в консоли всплывет предупреждающее сообщение.

Чтобы вручную решить конфликт, необходимо открыть файл index.txt и обсудить с разработчиком, какие изменения из двух следует оставить. После этого произвести добавление и фиксацию коммита, а ненужную ветку удалить.