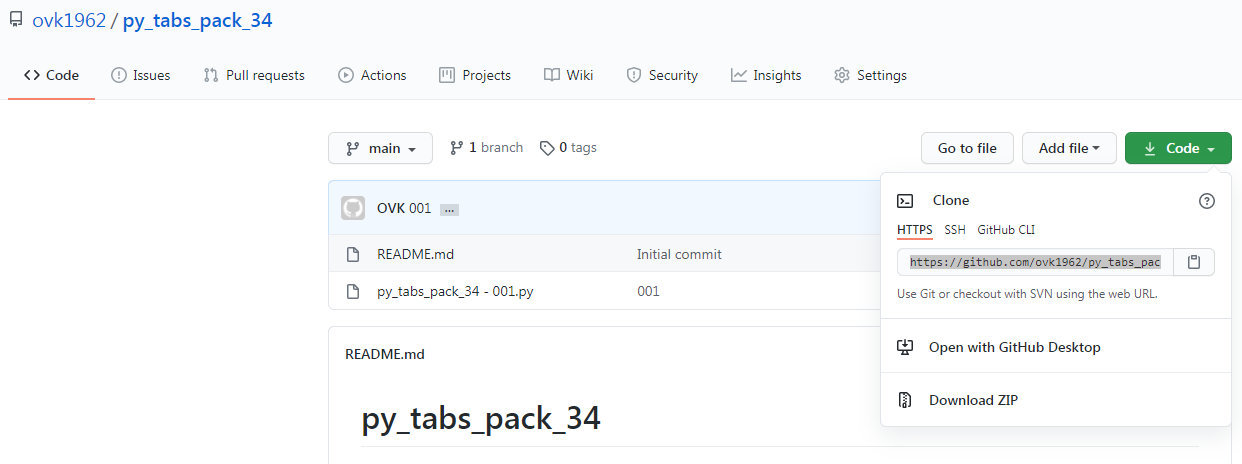
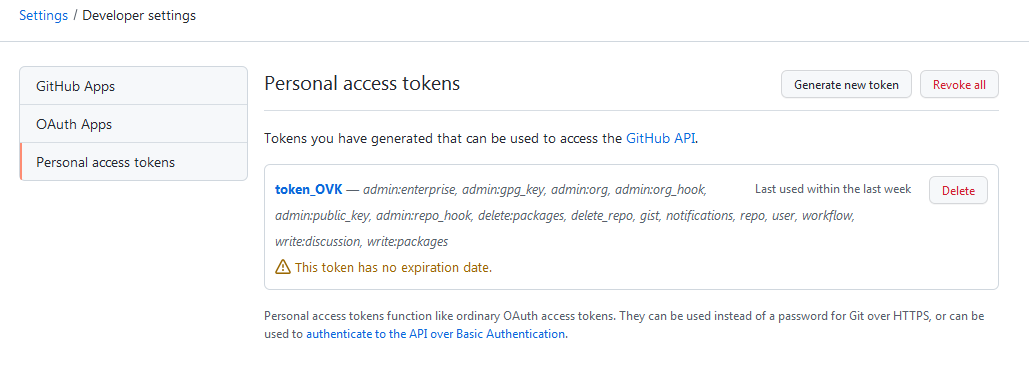
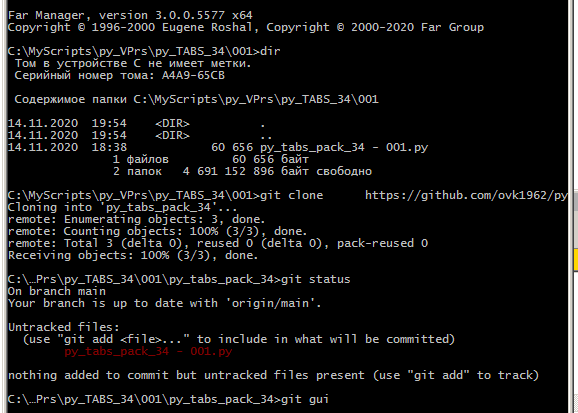
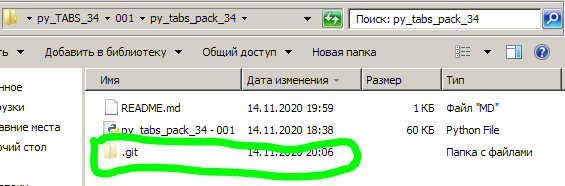
* Git-2.29.2.2-64-bit установка GIT и перезагрузка компe
* Создаем НОВОЕ репо py\_tabs\_pack\_34 на удаленном сервере github.com

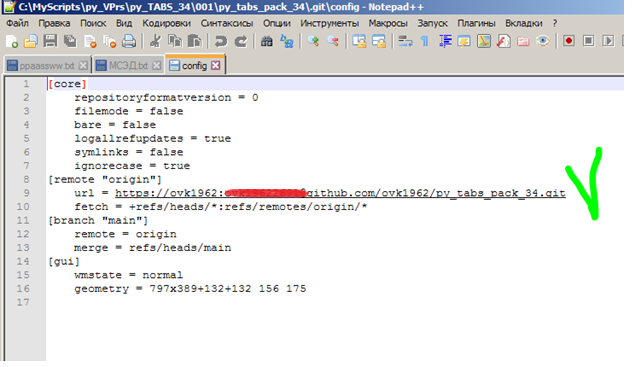


* Вместо пароля начали использовать token
* Token\_OVK = ghp\_Cm3zbWUe45coszbfwrM0z6MvwuClVN34RDY4

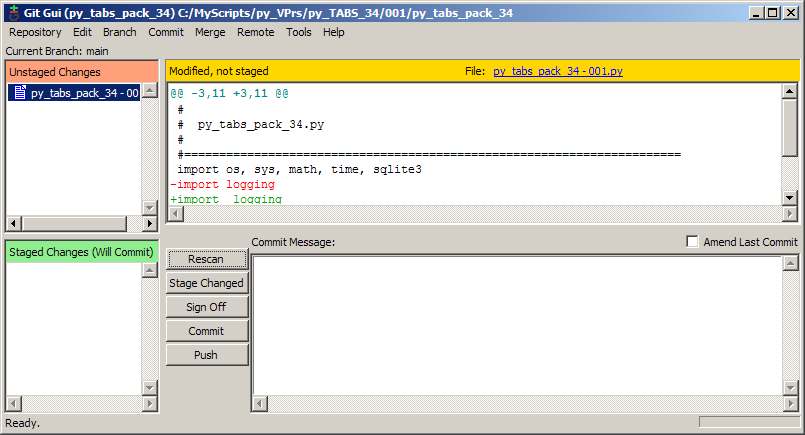


* Клонируем на локальный сервер
* Исправляем config

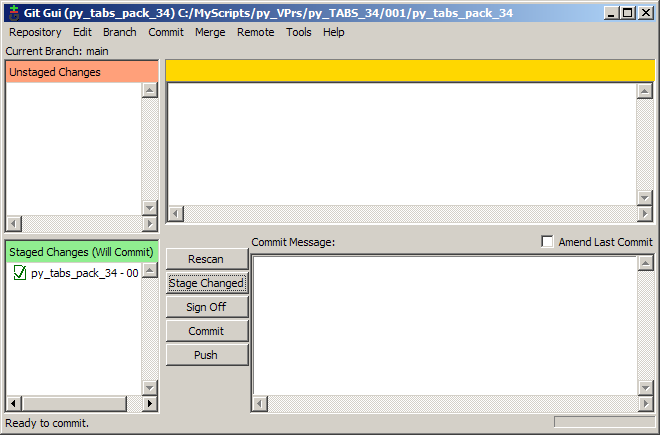
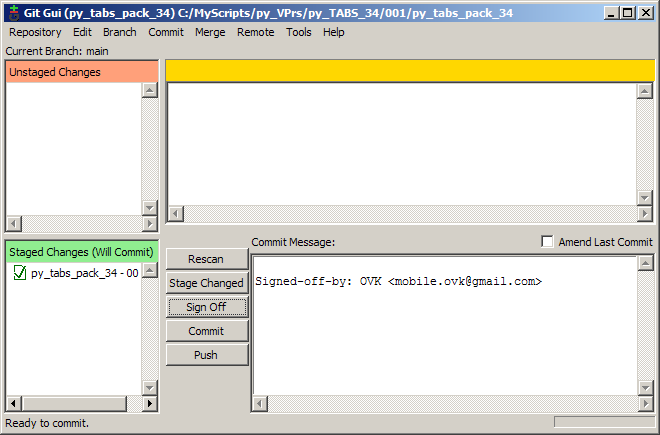


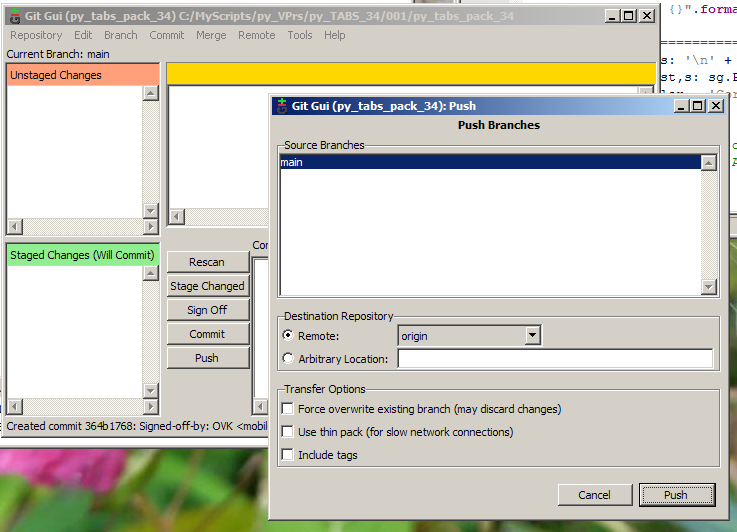
****

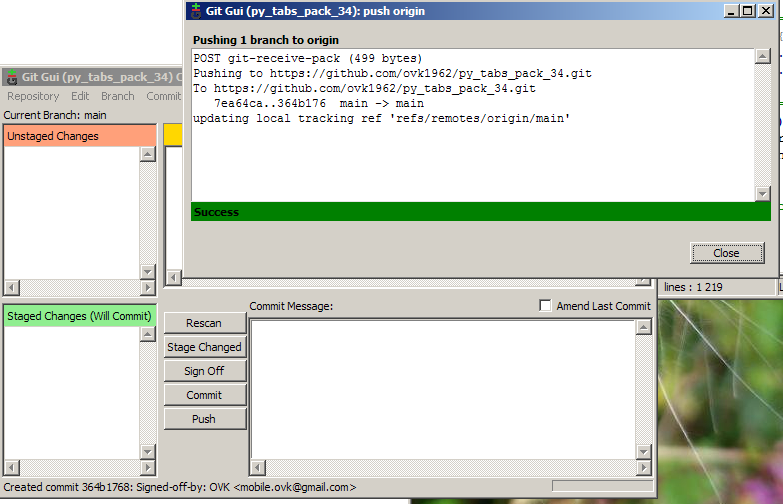
* Запускаем в папке где локальный репо (в консоле или в FAR) команду git gui



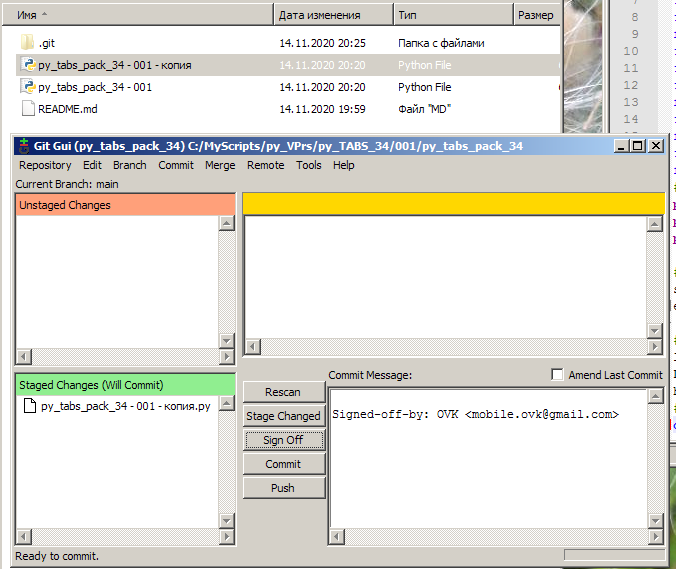
* Rescan / Stage / Sign / Commit / Push

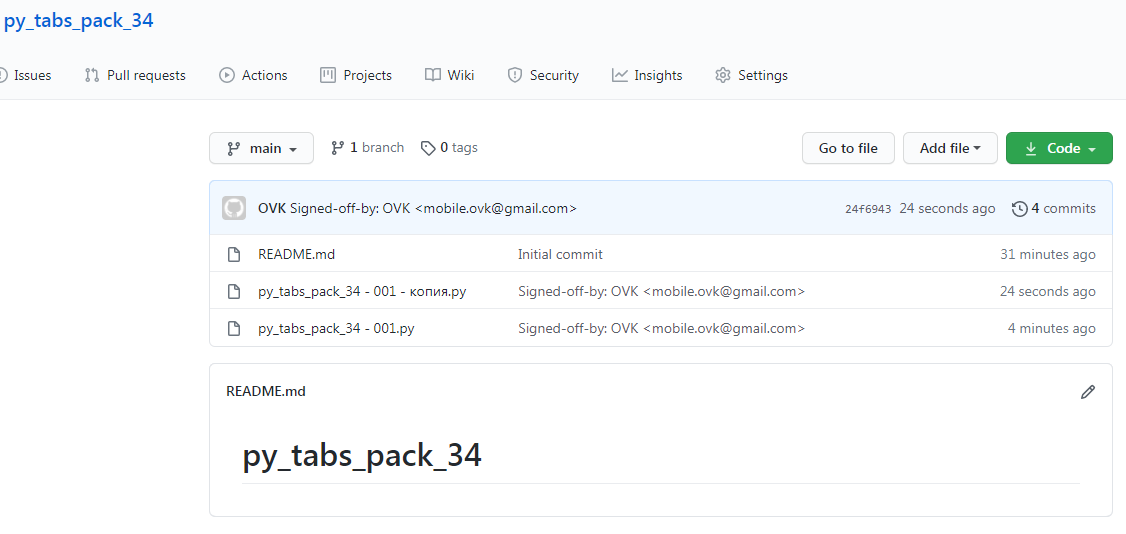
  


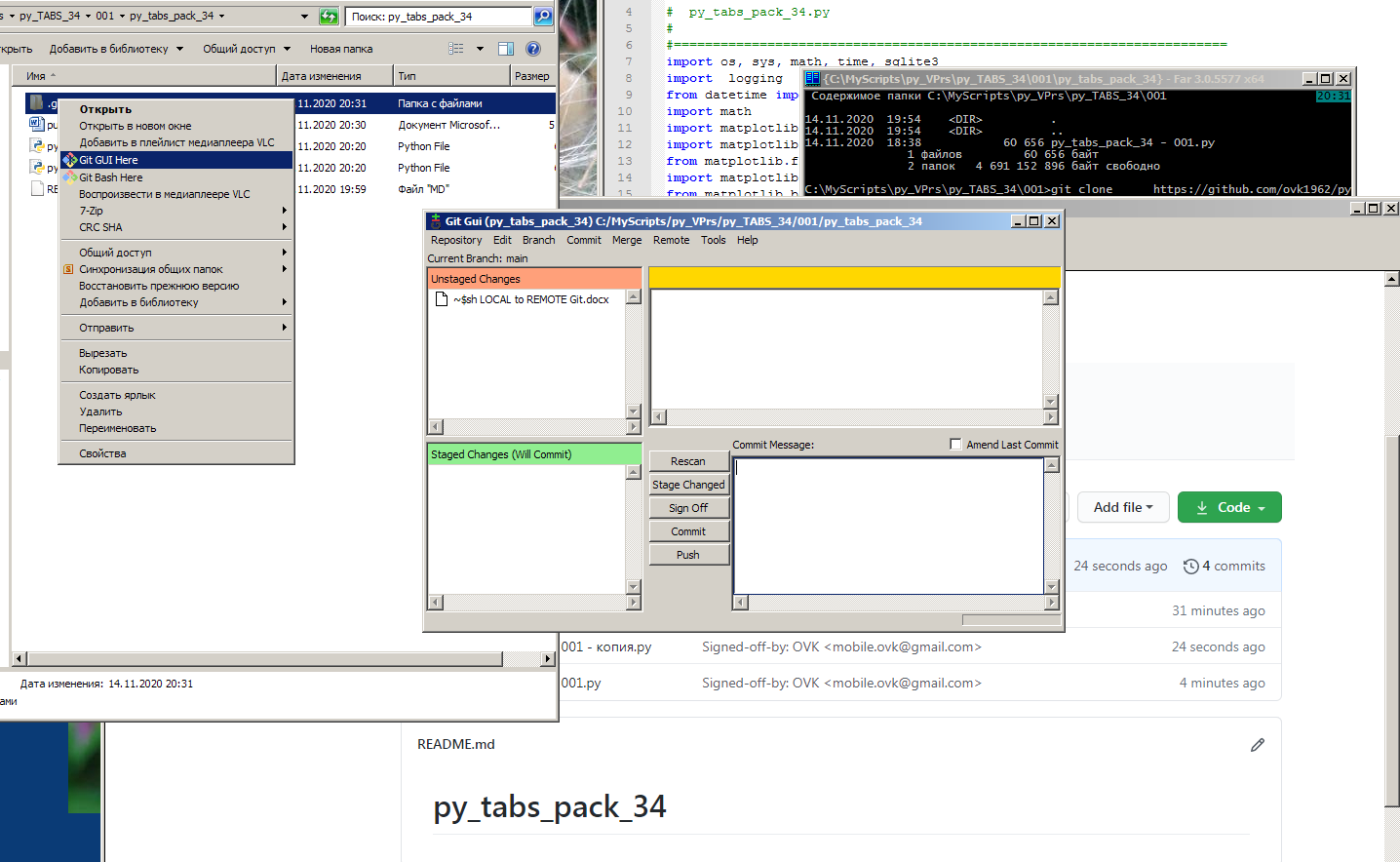




ПРИМЕР – пушим на удаленный сервер файл \*\*\* - 001 - копия







<https://zencoder.ru/git/git-gitignoring-files/>

Два способа игнорирования файлов - локальный и глобальный. Шаблоны для игнорирования файлов в для различных операционных систем, языков программирования, CMS и т. д. Игнорирование уже отслеживаемого файла в Git. Отслеживание пустых директорий.

## Шаблоны правил для .gitignore

Два ресурса на GitHub, в которых собраны шаблоны файла .gitignore для разных случаев - разных операционных систем, разных языков программирования и т. д.

### Общий список правил

По ссылке ниже представлен набор общих правил и рекомендаций при создании файла .gitignore. Помимо этого, в данном разделе показан шаблон для самого общего случая использования .gitignore:

[Общий список правил игнорирования файлов в Git](https://help.github.com/articles/ignoring-files/)

### Детализированный список правил

По ссылке ниже представлен более детальный и полезный список правил, по которым создается файл .gitignore. В этом списке можно найти рекомендации для разных операционных систем (Linux, Mac OS X, Windows, Android), разных языков программирования (Java, Perl и т. д.), различных CMS (к примеру - Joomla, Magento, Jekyll). Даже для README есть свой набор правил!

[Деталированный список правил игнорирования файлов в Git](https://github.com/github/gitignore)

Помимо списка языков программирования, в котором для каждого пункта этого списка представлены рекомендованные правила для файла .gitignore, имеется еще дополнительная директория “Global”.

Как раз в этой директории располагается список рекомендаций применительно к операционным системам. Достаточно найти нужную систему, перейти по ссылке и воспользоваться набором правил.

## Глобальное игнорирование

В Git возможно игнорирование файлов двумя способами:

* локально
* глобально

В последнем случае (глобально) имеется ввиду, что Git будет игнорировать указанные файлы всегда, в любом из проектов.

Ярким примером такого глобального игнорирования являются служебные файлы операционной системы - они должны игнорироваться всегда и в любом проекте Git.

Файл конфигурации с глобальным игнорированием может быть назван как угодно (в отличие от файла .gitignore). Главное - он должен располагаться в домашней директории пользователя, а не в директории с проектом. И это его главное отличие от файла .gitignore.

После того, как файл глобального игнорирования создан, необходимо добавить его в конфигурационный файл Git. То есть, “сказать” Git, что “у тебя есть список файлов, которые ты должен игнорировать; этот список называется так и расположен он здесь”.

Выполняется это командой:

git config --global core.excludesfile ~/.gitignore\_global

Если теперь взглянуть на конфигурационный файл Git, то увидим следующее:

$ cat ~/.gitconfig

[user]

name = g\*e

email = g\*e@gmail.com

[core]

excludesfile = /Users/g\*e/.gitignore\_global

## Игнорирование отслеживаемого файла

Может возникнуть ситуация, когда файл был создан и помещен в репозиторий Git. То есть, теперь он является отслеживаемым.

Но это была ошибка - этот файл не нужно было отслеживать! Как сказать Git, чтобы он не отслеживал этот файл?

Для этого нужно выполнить два шага.

Первый - это внести имя нужного файла в список игнорируемых файлов - отредактировать файл .gitignore.

Однако, этот шаг еще ничего не даст - Git все также будет его отслеживать, так как он уже внесен в буфера обмена.

Для этого нужно “вынуть” данный файл из буфера обмена. Это выполняется командой:

git rm --cached name\_of\_file

Данная команда удалит указанный файл из буфера обмена, однако оставит в рабочей директории и в репозитории. Последующий коммит произведет удаление этого файла и из репозитория.

Осталось только выполнить коммит обоих изменений - добавление файла в игнорируемые и удаление его из репозитория.

Все последующие изменения в этом файле отслеживаться не будут.

## Отслеживание пустых директорий

У Git имеется одна очень не очевидная особенность, которая сбивает с толку прозелитов и в один прекрасный момент может их подвести, если они не знают о данном факте.

Заключается он в том, что Git - это система контроля версий для файлов! Другими словами - Git умеет отслеживать только файлы!

Поэтому, если в проекте имеется пустая директория, то Git ее просто “выкинет”. По той простой причине, что точки зрения Git директория - всего лишь путь к файлу, способ найти отслеживаемый файл. А раз файла нет, то путь ведет вникуда, так как отслеживать нечего.

Кстати сказать, я сам когда-то был таким новичком и столкнулся с этим непонятным для меня фактом. Интуитивно догадался, в чем дело и поступал таким “корявым” (как я считал) способом.

К примеру, для директории psd я создавал файл-заглушку dummy.psd размером 1px\*1px. Главное - чтобы в директории psd что-то лежало - тогда Git не удалит ее! )

Как оказалось, я был прав! И это единственный правильный способ заставить Git отслеживать пустые директории.

Одна только маленькая поправка - в мире Unix такой файл-заглушку можно создать с помощью команды touch и выглядеть такой файл будет так (общепринятое соглашение):

.gitkeep Что же - этот способ более аккуратный и грамотный!