

Использование Github Pages

Github – сервер удаленных репозиторий для распределенной системы контроля версий (СКВ) Git – предоставляет возможность бесплатного хостинга статических ресурсов, GitHub Pages. Для работы потребуется ПК с установленной СКВ Git. Графический клиент не нужен, достаточно воспользоваться консолью (в зависимости от настроек при инсталляции – консолью гита, гит башем или просто консолью самой ОС).

Для начала работы требуется перейти в папку проекта, после чего последовательно указать следующие команды:

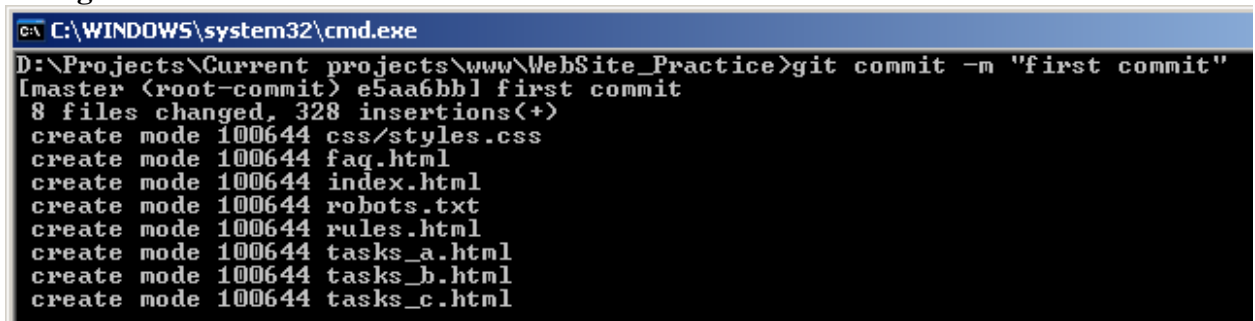
git init

```
D:\Projects\Current projects\www\WebSite_Practice>git init
Initialized empty Git repository in D:/Projects/Current projects/www/WebSite_Practice/.git/
```

git add .

На этом этапе инициализирован локальный репозиторий (хранилище данных о версиях вашего проекта) и добавлены все файлы из рабочей папки в контролируемые. Т.е. СКВ их отслеживает, но пока ничего не хранит. Для заполнения репозитория необходимо выполнить коммит (фиксацию изменений в рабочей папке):

git commit -m "first commit"



```
C:\WINDOWS\system32\cmd.exe
D:\Projects\Current projects\www\WebSite_Practice>git commit -m "first commit"
[master (root-commit) e5aa6bb] first commit
8 files changed, 328 insertions(+)
create mode 100644 css/styles.css
create mode 100644 faq.html
create mode 100644 index.html
create mode 100644 robots.txt
create mode 100644 rules.html
create mode 100644 tasks_a.html
create mode 100644 tasks_b.html
create mode 100644 tasks_c.html
```

Здесь **-m** – ключ, указывающий, что пояснения к коммиту (в данном примере – лаконичное «first commit») будут заданы без использования внешнего редактора, прямо при использовании команды commit.

Если гит использовался на данном ПК впервые, то на этом этапе коммита не произойдет, а будет выведено сообщение об ошибке – с запросом данных того, кто делает этот коммит. И предложено решение проблемы – путем указания двух команд, заполняющих конфигурационный файл СКВ.

git config --global user.name "ваше_имя"

git config --global user.email "ваша_почта"

Чтобы задать настройки только для данного проекта, нужно убрать ключ **--global**. Иначе указанные вами настройки будут использоваться по умолчанию для всех проектов на данном ПК – т.е. при фиксации изменений (выполнении команды commit) на всех проектах, а не только ваших, будут по умолчанию фигурировать ваши данные.

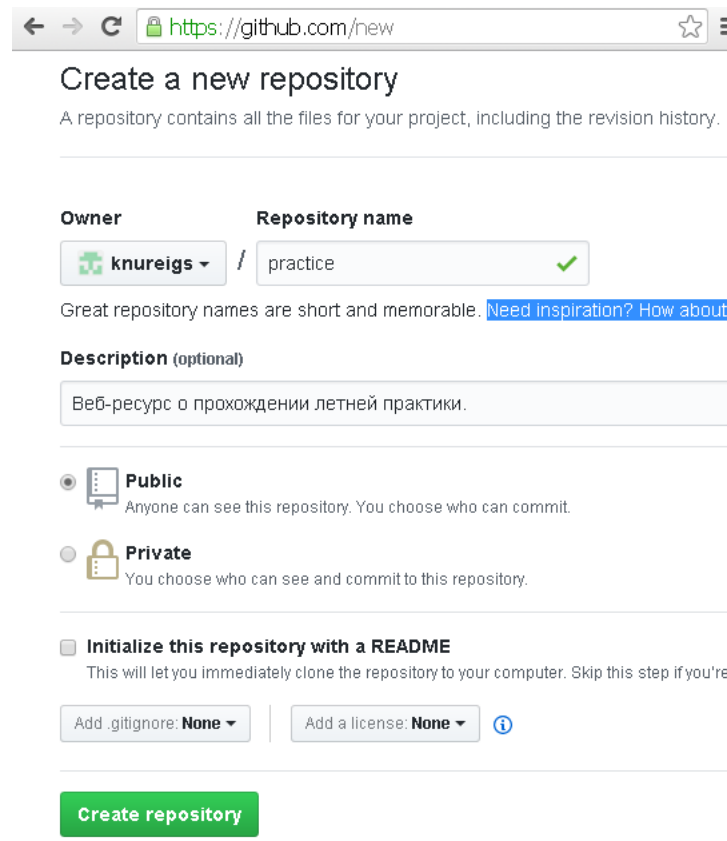
На этом этапе – после успешного коммита – текущее состояние проекта сохранено в локальном репозитории.

Теперь надо задать **удаленный репозиторий**, используемый командой push. Поскольку изначально он отсутствует (нельзя передать данные локального репозитория в удаленный – в то, чего еще нет), нужно его сначала создать, воспользовавшись одним из хостингов репозиторий (Github, Bitbucket и др.). В рамках лабораторной используется Github, вернее, его возможность использовать прямые ссылки на ресурсы репозитория – сервис Github Pages. Если нет аккаунта на Github – **регистрируйтесь**. Обязательно запомните параметры входа, понадобятся в будущем.

После регистрации можно создать пустой репозиторий – необходимо указать только его имя (Repository name). На скриншоте приведен пример создания публичного репозитория (приватные репозитории являются платными на данном сервисе) для проекта, посвященного

прохождению летней практики. В качестве владельца (Owner) системой используется логин, указанный при регистрации на данном сервисе. Назначение проекта указывается в необязательном поле Description.

Использование файла **.gitignore**, содержащего перечень игнорируемых файлов, для данной лабораторной не является необходимым. Хорошим тоном при создании репозитория является указание лицензии – но это также не обязательно.



Теперь указываем данный репозиторий, расположенный на Github, в качестве удаленного репозитория нашего проекта:

```
git remote add origin https://github.com/knureigs/practice.git
```

Здесь **origin** – просто традиционное короткое название удаленного репозитория. Можно указать любое другое название. Сам адрес репозитория указан для примера (в соответствии со скриншотом), узнать его можно при создании нового репозитория на Github.

Выполняем пуш – отправку своих изменений (из локального репозитория, куда был выполнен коммит – фиксация изменений вашей рабочей папки) на удаленный репозиторий, в основную ветку проекта – master (она есть по умолчанию, а других пока нет и не нужно):

```
git push -u origin master
```

Если сделать **git push** без ключа **-u**, Git не свяжет локальную ветку с веткой удалённого репозитория. Поэтому ключ нужен – во избежание проблем с будущим использованием команды **git pull**.

Последует запрос логина и пароля, причем пароль вводится вслепую. Возможна авторизация средствами самой установленной СКВ – с помощью привычной формы ввода логина и пароля. Альтернативное решение – позволяющее избежать проблем с авторизацией при отправке изменений в удаленный репозиторий – предполагает использование SSH.

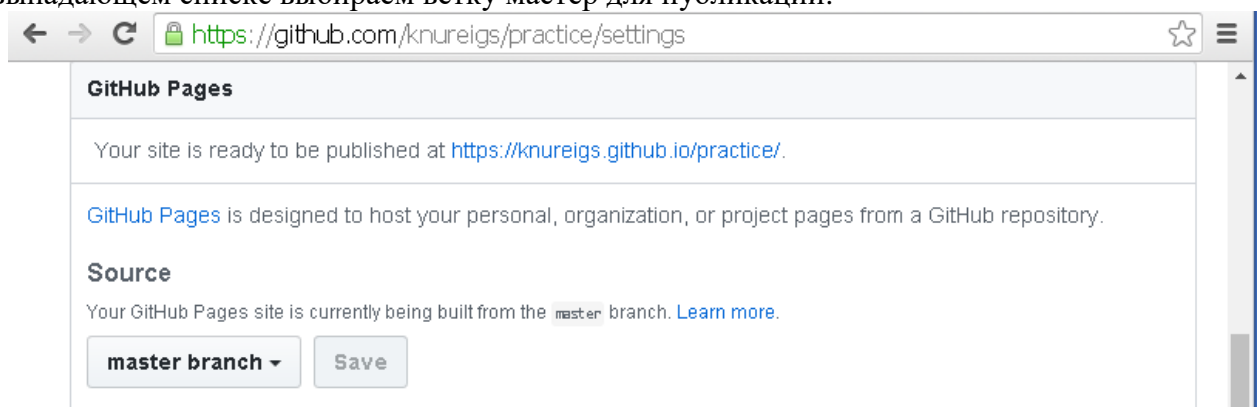
```
C:\WINDOWS\system32\cmd.exe
D:\Projects\Current projects\www\WebSite_Practice>git push -u origin master
Username for 'https://github.com': knureigs
Password for 'https://knureigs@github.com':
Counting objects: 11, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 17.34 KiB | 0 bytes/s, done.
Total 11 (delta 0), reused 0 (delta 0)
To https://github.com:knureigs/practice.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Проверить, актуальна ли версия локального репозитория по сравнению с удаленным, можно командой

git remote show origin

Данная команда возвращает много полезной информации, но главное в скобках в конце – «up to date», если локальный репозиторий актуален по сравнению с удаленным (что не означает, что он актуален по сравнению с вашей рабочей папкой – для этого воспользуйтесь командой **git status**) или «Out of date» – если ваша локальная версия успела устареть.

Теперь репозиторий есть, он заполнен, и в контексте лабораторной работы – требуется открыть доступ по ссылке, используя GitHub Pages. Возвращаемся на <https://github.com> в свой репозиторий, далее – вкладка *settings*, находим настройки GitHub Pages и в выпадающем списке выбираем ветку мастер для публикации.



В результате получен доступ к ресурсу (для данного примера) по следующей ссылке:

<https://knureigs.github.io/practice/>

Проверяем доступ. Ничего страшного, если при первой проверке вместо сайта вы видите ошибку 404 – обновление происходит с некоторой задержкой, попробуйте проверить через пару минут.