Введение в ГИТ

Итак, когда вы склонировали себе репозиторий, вы имеете на своём ПК его копию. А значит у вас на ПК теперь есть репозиторий **(ура)**. И в нём храниться всё - от информации о ветках, и коммитах, до самих файлов. Далее описаны несколько команд (которые будут понятны только после полного прочтения тутора)

- `git status` - показывает статус текущего репозитория (нужен **add/commit/push** или всё хорошо)
- `git add .` - добавляет все папки и подпапки текущей директории в следующий коммит (подготавливает их для дальнейшей обработки)
- `git commit [-m "message"] [-S]` - Создаёт новый коммит, при необходимости подписывает его с помощью флага `-S` (о подписывании коммита и будем говорить далее).
  Флаг `-m` "обзывает" коммит.
- `git push` - пушит ваш локальный репозиторий в ветку на сервере (пытается их синхронизировать, сохранить изменения на сервере git). При необходимости в данную команду можно/нужно добавить аргументы `origin <ветка>`, где ветка - это название ветки куда хотите запушить.
  Полная комманда для ветки origin/lab2 будет выглядеть так: `git push origin lab2`
- `git pull` - загружает все изменения и коммиты с сервера.
- `git log` - выводит всю информацию о коммитах, а также на каком находится текущая ветка.
- `git config [--global] --list` - если использовать `--global` - выведет информацию о конфигурации git глобально (для всех репозиториев), а если нет то для текущего. Если не использовать лист а написать (к примеру) `git config user.name "John Doe"`, тогда переменная `user.name` изменится, но только для данного репозитория (чтобы изменить глобальную - напишите флаг `--global`)

Работа с windows

Если у вас стоит **OS windows** - данные команды будет проще всего выполнять в приложении **git bash**. Вот ссылка. https://git-scm.com/download/win.
После установки, все команды, которые будут написаны далее будут работать без особых проблем.

Настройка SSH (опять?)

Начнём с настройки *SSH* для *НАШЕГО ПК* (не для *solaris*, как было в прошлый раз)

1 Перейдём в папку `.ssh` командой `cd ~/.ssh` . Если не работает, то пишем: `cd ~`
`mkdir .ssh`
`cd .ssh *`

- Данный набор команд создаст папку **.ssh**

2 Далее в этой папке создадим пару ключей командой `ssh-keygen -t rsa`. **ОБЯЗАТЕЛЬНО** с паролем (иначе гит будет жаловаться) Запомнили название *(КЛЮЧ)* После чего создали файл **config** командой `touch config` Открываем данный файл для редактирования командой `nano config`

В данном файле прописываем:

Host ИМЯ_КОНФИГА
HostName github.com
IdentityFile ~/.ssh/КЛЮЧ
IdentitiesOnly yes

- где *КЛЮЧ* - название ключа (без расширения pub) а *ИМЯ_КОНФИГА* - любое имя конфига которое захотите (не сильно длинное, его необходимо запомнить)

3 Далее, после создания данной пары ключей идём на гитхаб и там подключаем этот ключ. Сначала, конечно же, скопируем его, введя команду `cat КЛЮЧ.pub` **(ОБЯЗАТЕЛЬНО .pub)** и скопировав в буфер обмена (в **git bash** работает пкм и открывается контекстное меню)

4 Потом идём на гитхаб, открываем настройки, SSH и GPG ключи, add SSH key, далее вставляем ключ и называем его (как-нибудь) ДА, это не ошибка, мы вставляем туда уже **второй** ключ После чего добавляем его же, но в типе ключа (при добавлении) указываем не **authentication key** А **signing key**.

Мы добавили ssh ключ входа + ssh ключ подписи

Работа

5 Чтобы всё собственно заработало (подпись), осталось лишь *пара* штрихов.

```
git config --global user.name "Vasya Pupkin"
git config --global user.email v.pupkin@g.nsu.ru
```

(замените Vasya Pupkin и v.pupkin на ваши человеческие имя и фамилию латиницей и префикс вашего университетского email).

6 После чего пишете (ВНЕ ЗАВИСИМОСТИ ОТ СИСТЕМЫ)

```
git config --global gpg.format ssh
```
`git config --global user.signingKey ~/.ssh/КЛЮЧ` - где *КЛЮЧ* - название вашего ключа

7 Далее, всё очень даже просто.

Клонируем репозиторий в любую удобную для нас папку.

- Чтобы перемещаться по папкам можно использовать `cd` и `ls`

```
git clone git@ИМЯ_КОНФИГА:НИКНЕЙМ/РЕПОЗИТОРИЙ
```

Никнейм - ваш ник на github **Репозиторий** - название репозитория **Имя_конфига** - имя конфига 😃

8 Далее естественно работаем не в основной ветке (создаём ответвление от irtegov/accepted)

```
git branch -a
git checkout irtegov/accepted
git checkout -b lab2
```

Этой последовательностью комманд мы создаём новую ветку под названием lab2 (для каждой новой лабы - своя ветка)

9 Создаём свою рабочу папку

`mkdir -p ГРУППА/ПРЕФИКС/lab2`

ГРУППА - номер группы **ПРЕФИКС** - префикс вашего университетского email (v.pupkin)

10 Далее заходим в корневую папку репозитория. Работаем (создаём любые файлы/папки которые нужны). После этого чтобы залить на сервер изменения:

`git add .`
`git commit -S -m "message"` - в message - любое описание коммита. -S подписывает коммит.
`git push origin lab2` - где lab2 - название ВАШЕЙ ветки

После данных команд в форке репозитория в ветке **lab2** (перейдите в неё, проверьте что на сайте вы находитесь в ней) вы должны увидеть внесённые изменения. **commit** должен быть **ОДИН** (требование Иртегова). Зайдите в коммиты ветки. Если всё хорошо, то справа в коммите будет заветная надпись **Verified**.

КАК объеденить коммиты правильно?

Есть у нас на ПК **N** коммитов, но нам нужно их объеденить? Не проблема!

1 Для этого в рабочем репозитории пишем команду:

`git rebase -i HEAD~N` - где N количество желаемых схлопнутых коммитов

откроется редактор **vim** с файлом, где предложат схлопнуть ветки

Если весь текст синий - это **vim**, и чтобы редактировать нажмите на клавиатуре **esc**, потом **i**
Редактируйте (все подсвеченые синим **pick** меняйте на **squash**, кроме самого верхнего, в него будут сливаться коммиты) После чего **esc**, **:wq** (набираем на клаве) Вышли. Далее снова открылся **vim** (сам). Он предлагает оставить одно из названий. Стираем ненужное, выходим из **vim**.

Далее, у нас есть **commit**. Но есть проблема - он не подписан. Чтобы подписать последний созданный / существующий коммит прописываем:

`git commit --amend --no-edit -S`

коммит подписан, можно пушить. Но что если у вас уже запушены несколько коммитов в 1 ветку на 1 задачу? Тоже не беда. Чтобы это действительно проверить вводим:

`git push origin lab2` - lab2 это ветка куда пушим

ОСТОРОЖНО, ИСПРАВЛЕНИЕ КОТОРОЕ БУДЕТ ДАЛЬШЕ МОЖЕТ ПЕРЕЗАПИСАТЬ И БЕЗВОЗВРАТНО УДАЛИТЬ ФАЙЛЫ В ВАШЕМ РЕПОЗИТОРИИ НА GITHUB

Если вылазит ошибка, из-за склеенного коммита, то пишем волшебное слово `--force` после комманды
`git push origin lab2 --force` - и ВУАЛЯ
Всё сработало. Но главное не забывать, что force перезаписывает **ВСЮ** историю commit-ов сервера, что может повлечь потерю данных в ветке)

Дальше остаётся только зайти на **github**, проверить что всё верно, зайти на свою ветку (к примеру lab2) и там будет кнопка **conribute**. На неё нажимаем, там создаём **pull request**. После перехода появится окно с pull request-ом, и по итогу нам нужно отправлять pull request из ветки **lab2** в ветку

irtegov/accepted. Только **тогда** Иртегов начнёт принимать задачу. (Настройка слияния веток есть сверху)

ЭТО МОЁ ВИДЕНЬЕ И В НЕКОТОРЫХ ВОПРОСАХ Я НЕ ПРЕТЕНДУЮ НА ДОСТОВЕРНОСТЬ)))

Полячков Д.А. 22216