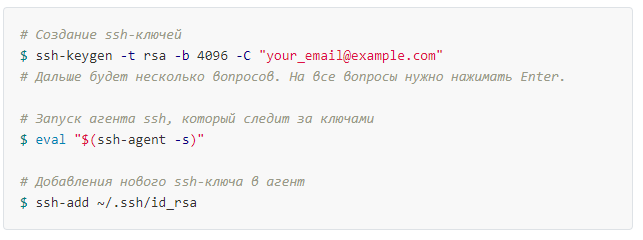
**Алгоритм создания GIT репы + команды для работы:**

1. **Установить GIT и прописать 2 строчки для конфига (чтобы видели кто изменял файлы на git):**
   1. **Git config --global user.name “Demyankov Vladislav”**
   2. **Git config --global user.email** [**demyankov@devopser.by**](mailto:demyankov@devopser.by)
   3. **Cat .gitconfig – для проверки (/c/Users/пользователь/.gitconfig)**
2. **Создаем базу данных git для созданной локальной папки репы и создаем файл .gitignore для игнорирования определенных файлов и директорий:**
   1. **Git init MYproject**
   2. **Git status – проверка git**
   3. **Nano .gitignore**
3. **Добавляем файлы и делаем снимок (commit):**
   1. **Echo “file1” > file.txt**
   2. **Git add file.txt (. – чтобы добавить все файлы)**
   3. **Git commit –m “V1”**
   4. **Git log – для просмотра истории снимков**
   5. **Git checkout имя - для возращения файла в исходное состояние до переноса его на этап stage**
4. **Заходим на github создаем репу,синхронизируем ее с локальной и закидываем изменения локальной копии:**
   1. **git remote add origin url**
   2. **git push -u origin master**
5. **Клонируем репу с github на локальный компьютер, изменяем и закидываем обратно:**
   1. **Git clone url-ссылка репы**
   2. **Изменяем или добавляем файлы, делаем add, commit**
   3. **Git push origin (–u origin master)**
6. **Клонируем репу с github на локальный комп, создаем ветку fix, изменяем в ней файлы и добавляем в ветку master, удаляем ветку fix:**
   1. **Git clone url**
   2. **Git checkout –b fix – создали и перешли в ветку fix**
   3. **Git checkout имя-ветки – переход между ветками**
   4. **Git branch – показывает на какой ветки находимся**
   5. **Git merge fix (объединяем ветки, должны находиться на ветки master)**
   6. **Git branch –d fix (должны находиться на ветки master)**
   7. **Git branch –D имя-ветки (если изменения в ветки уже ненужны)**
7. **Как вернуться на прошлый снимок:**
   1. **Git checkout хэш-снимка (берем с git log)**
   2. **Git checkout master –переходим на самый последний снимок**
8. **Изменение файлов и имя снимка без создания нового:**
   1. **Изменяем файлы**
   2. **Git commit --amend (нажимаем insert, изменяем, ESC, нажимаем : wq!**
9. **Для полного возвращения к прошлому снимку:**
   1. **Git reset --hard HEAD~число-насколько снимку вернуться назад**
10. **Для удаления снимков но с сохранением текущих файлов:**
    1. **Git reset --soft HEAD~число-сколько снимков удалить**
11. **Добавления своей ветки с изменениями для проверки и последующей синхронизацией другим человеком:**
    1. **Git push --set-upstream origin имя-ветки**
    2. **Заходим на github нажимаем compare and pull request**
    3. **Заходим в pull request, оставляем коммент и нажимаем creat**
    4. **После того как другой человек нажал merge, удаляем локальную и удаленную ветку (если он не удалил)**
    5. **Git branch –d имя-ветки (удаление с ветки master) –локальную репу удаляем**
    6. **Git push origin --delete имя-ветки –удаленную репу удаляем**
12. **Обновление локальной репы:**
    1. **Git pull**

**Полезное от курса ru.hexlet.io**

**Создание SSH ключей**



**git pull --rebase -** каждый раз при начале работы

**Для работы по ssh нужно на компе добавить**

В **sudo nano ~/.ssh/config** следующие строки:

*Host github.com*

*Hostname ssh.github.com*

*Port 443*

Проверяем соединение: **ssh -T** [**git@github.com**](mailto:git@github.com)

**git restore 'имя файла' –** восстанавливает файл из последнего коммита в исходном состоянии, **git restore --staged ‘имя файла’ –** позволяет откатиться до шаг stage.

**git diff** – показывает разницу между файлами до stage (если нужно и в stage, то с ключом **--staged**).

**git log -p** - показывает все коммиты, а также ключ -p показывает изменения (diff). **--oneline** – ключ выводит все комииты строчкой хэш и название коммита.

**git show ‘хэш’** – показывает изменение по определенному коммиту, хэш берем из git log и хватает только первые 8 символов.

**git blame ‘имя файла’ –** показывает изменения файла, кто изменял и в каком коммите.

**git grep -i ‘что ищем’** - показывает все файлы в который совпадает искомое. (ключ -i поиск без учета регистра).

**git clean -fd** - очищает все незакомиченные данные и возвращает на состояния последнего коммита.

**git revert ‘хэш’** - отменяет последний коммит.

**git reset --hard HEAD~** - отменяет последний коммит(делать только если этого коммита нет в github) **--hard** означает полное удаление (Без него git reset отменит коммит, но не удалит его, а поместит все изменения этого коммита в рабочую директорию), **HEAD~** означает "один коммит от последнего коммита".

**git commit --amend --no-edit** – если забыли добавить что-то в коммит (--amend), --no-edit – для того чтобы не заходить в редактор.

**git add -i** – переход в интерактивный режим.

**git checkout ‘хэш’ -** перейти в один из коммитов.

**git switch -** -перейти быстро на главную ветку.

**git branch** – показывает в какой мы ветки и коммите находимся. (**git switch branch** – переключение между ветками).

**git stash** - сохраняет измененные файлы в .git (чтобы их вызвать **git stash pop**).