Git

Пример установки и настройки репозитория github

1. Создается репозиторий на github
2. Инициализируется репозиторий

   ```
   git init
   ```

3. Получаем стартовые данные

   ```
   git pull https://github.com/sudaka/sumservice
   ```

4. Добавляем файлы

   ```
   git add data.csv
   ```

5. Делаем первый коммит

   ```
   git commit –m “Initial commit”
   ```

6. Добавляем название удаленного репозитория

   ```
   git remote add master https://github.com/sudaka/sumservice
   ```

7. Отправляем измененное состояние

   ```
   git push master master
   ```

Общие сведения.

Ссылки:

Книга по GIT

GIT клиент для windows

Теоретическая информация

- Git хранит данные в виде набора снимков миниатюрной файловой системы
- Почти все операции выполняются локально
- Три основных состояния файлов: изменён (modified), индексирован (staged) и зафиксирован (committed)
  - К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы.
  - Индексированный — это изменённый файл в его текущей версии, отмеченный для включения в следующий коммит.
  - Зафиксированный значит, что файл уже сохранён в вашей локальной базе
- Секции проекта Git: рабочая копия (working tree), область индексирования (staging area) и каталог Git (Git directory).
  - Рабочая копия - снимок одной версии проекта. Эти файлы извлекаются из сжатой базы данных в каталоге Git и помещаются на диск для использования или редактирования.
  - Область индексирования — это файл с информацией о том, что попадёт в следующий коммит.
  - Каталог Git — это то место, где Git хранит метаданные и базу объектов вашего проекта. Это копируемая часть при клонировании репозитория с другого компьютера.
- Базовый подход в работе с Git:
  - Изменяете файлы вашей рабочей копии.
  - Выборочно добавляете в индекс только те изменения, которые должны попасть в следующий коммит, добавляя тем самым снимки только этих изменений в индекс.
  - Когда вы делаете коммит, используются файлы из индекса как есть, и этот снимок сохраняется в ваш каталог Git.
  - Если определённая версия файла есть в каталоге Git, эта версия считается зафиксированной (committed). Если файл был изменён и добавлен в индекс, значит, он индексирован (staged). И если файл был изменён с момента последнего распаковывания из репозитория, но не был добавлен в индекс, он считается изменённым (modified).

Настройка Git

Просмотр всех установленных настроек

```
git config --list
```

Просмотр места размещения настроек

```
git config --list --show-origin
```

3 уровня настроек: системные, глобальные и репозитория
Значения, общие для всех пользователей системы и для всех их репозиториев. Права суперпользователя.

```
 git config --system
```

Значения текущего пользователя и применяется ко всем репозиториям в текущей системе.

```
git config --global
```

Настраивает значения репозитория, который вы используете в данный момент. Нужно находиться в репозитории Git.

```
git config --local
```

Значение конкретного ключа

```
git config <key>
```

Настройка имени и адреса электронной почты. Каждый коммит в Git содержит эту информацию и не может быть далее изменен.

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

Настройка текстового редактора для набора сообщений

```
git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"
```

Настройка имени ветки при инициализации репозитория командой git init (master по умолчанию). Не обязательно.

```
git config --global init.defaultBranch main
```

Создание репозитория

Создание репозитория в существующем каталоге
Перейти в каталог.

```
cd C:/Users/user/my_project
```

Инициализировать репозиторий. Будет создан подкаталог .git

```
git init
```

Клонирование существующего репозитория
С сервера забирается (pulled) каждая версия каждого файла из истории проекта.

```
git clone <url>
```

Для клонирования репозитория в другой каталог необходимо указать имя

```
git clone https://github.com/libgit2/libgit2 mylibgit
```

В предыдущих командах будет создана папка. Для клонирования в текущую папку (точка через пробел)

```
get clone <url> .
```

Gitverse

Приватный репозиторий

Работать по правильному логину/паролю не будет, если включена 2FA. Нужно сначала сгенерировать новый токен доступа (Настройки - управление токенами). Затем клонировать репозиторий в следующем виде:

```
git clone https://<токен>@<адрес репозитория>
```

Например

```
git clone https://b0...e3@gitverse.ru/bobrobot/projects_ogrinizer.git
```

Затем перейти в директорию и задать локального пользователя и email

```
git config --local user.name "Bobrov Sergey"
git config --local user.email test@test.ru
```

После этого приватный репозиторий начинает корректно работать.

Авторизация VSC

После клонирования способом выше начинает работать автоматически, но может и слететь. Если слетает, то пользователь - как он отображается в профиле (в моем случае bobrobot), пароль - токен доступа.

Ветвления

??????? ?? ?????? ???????? ??? ?????? commit ??????? ??????? ??????

Команда	Доп. параметры	Описание
git branch		?????? ????????? ?????
	-a	?????? ????????? (? ?????? ?? ??????? git) ?????
	<name>	?????????? ?????
	-d hotfix	???????? ???????? ?????
	--merged --no-merged	?????? ?????? ????? ?????? ?? ??????
	--move bad-branch-name corrected-branch-name	??????????????
git checkout	<name>	• переключение на ветку
	-b <name>	???????? ????? ? ???????????? ?? ???
	-- fname	???????????? ????????? ????? fname
git push	--set-upstream origin corrected-branch-name	?????????? ??????????????
	origin --delete master	???????? ????? ?? ????????? ???????
git log --oneline --decorate --graph --all		????? ?????? ???????

??????? ?????

Команда	Описание
git checkout master	???????????? ? ?? ?????, ??????? ????? ????????
git merge hotfix1	??????????? ??????? ????? ? ?????? hostfix1

? ??????, ?????????? ?????? ??????????, ???????? ??????, ???????????? ????????. ????? ???????
??? ????????, ??????? ????? (??? ????? ???????) ?????, ???????? ???? (git add…) ? ???????????.
????? ????????? ???? ?????? ????? ????? ? ????? commit

??????????????

```
git checkout experiment
git rebase master
git checkout master
git merge experiment
```

? ?????? ????????? ?????????? ? ????????? ???? git add (??? ???????!), ????? git rebase --continue

?????????????? ? ?????? 3 ?????, ??????????? 3 ???? ?? 2 ?????, ?? ? ???????? ????? ????? ???????? ????????? 3 ?????, ?????? ???? ??? ?? ???????

```
git rebase --onto master server client
```

Примеры.

Однократное ветвление в VSC

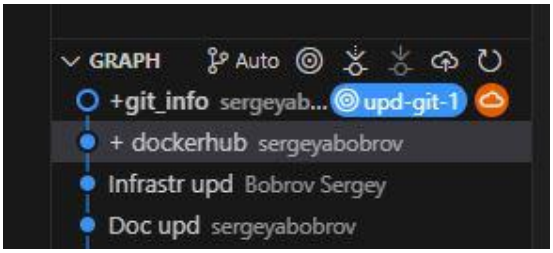
Стартовые условия:

- Два git клиента - ПК для разработки с VSC и тестовый сервер.
- Текущий commit полностью соответствует commit на сервере.

Думаю, что через интерфейс VSC было бы правильнее, но мне пока что проще через консоль.

```
>git checkout -b upd-git-1
Switched to a new branch 'upd-git-1'
>git push -u origin upd-git-1
```

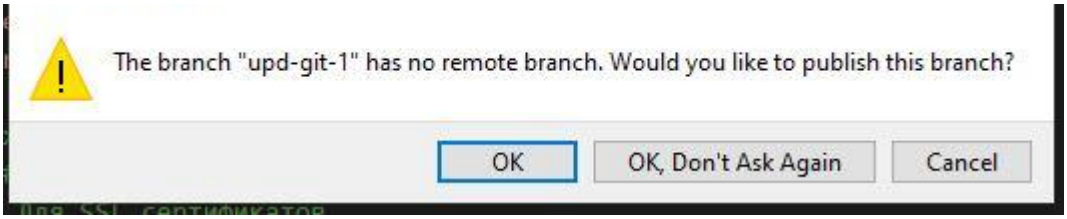
После этого в VSC в разделе git изменилось имя ветки на upd-git-1, все коммиты будут идти в нее.



Клонируем репозиторий на тестовый сервер, смотрим список веток. И не видим ветку upd-git-1, возможно потому что в ней нет ни одного коммита.

```
git branch
* master
```

В VSC сохраняем изменения. Появляется сообщение об отсутствии удаленной ветки.



На gitverse в разделе Ветки появляется имя ветки и отображается commit. Но git branch на тестовом сервере все равно не показывает дополнительные ветки. Смотрим удаленные ветки и переключаемся на нее. Теперь git branch выдает состояние нужной ветки.

```
git branch
* master

git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/upd-git-1

sergey@jenkins:~/projects_ogrinizer$ git checkout remotes/origin/upd-git-1

sergey@jenkins:~/projects_ogrinizer$ git branch
* (HEAD отделён на origin/upd-git-1)
  master
```

Чтобы связать локальную ветку и удаленную ветку, сначала локальную ветку нужно создать. Поэтому на тестовом сервере создаем ветку (можно с другим именем, но проще одинаковые имена)

```
git checkout -b upd-git-1
Switched to a new branch 'upd-git-1'
```

Теперь связываем ветки

```
git branch --set-upstream-to remotes/origin/upd-git-1
```

Теперь работают pull

```
git pull
Обновление 86a0c85..d813524
Fast-forward
```

```
 compose-files/Dockerhub/docker-compose.yaml | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Теперь делаем слияние веток

```
git branch
  master
* upd-git-1

git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

D:\projects\projects_ogrinizer>git pull
Already up to date.

D:\projects\projects_ogrinizer>git merge upd-git-1
Updating 436f79a..c889934
Fast-forward
 compose-files/Dockerhub/.env                | 3 ++-
 compose-files/Dockerhub/docker-compose.yaml | 4 ++--
 2 files changed, 4 insertions(+), 3 deletions(-)

D:\projects\projects_ogrinizer>git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote: . Processing 1 references
remote: Processed 1 references in total
To https://gitverse.ru/bobrobot/projects_ogrinizer.git
   436f79a..c889934  master -> master
```

Предыдущий вариант полностью переносит ветку со всеми коммитами. Однако можно объединить обновление в один большой коммит:

```
git checkout master
git pull
git merge --squash upd-hub-1
git commit -m "Итоговое обновление из ветки upd-hub-1"
git push
```

На удаленной машине обязательно перейти на ветку master перед созданием новой ветки!

Что нужно знать о Git

Локальный репозиторий

- Установка git клиента. Показатель успешной установки - вывод версии при команде git --version
- Бывают локальные и облачные хранилища. Они существуют независимо и синхронизируются вручную. Из-за этого могут быть глюки.
- Создание локального хранилища
- Что такое коммит. Создание коммитов в текущей ветке
- Отслеживаемое и неотслеживаемое состояние файлов
- Возврат на предыдущие состояния для просмотра

Сейчас нужно уметь:

1. Создать у себя директорию C:\projects
2. В ней создать директорию git_local
3. Через консоль инициализировать репозиторий.
4. Создать файл file1.txt с текстом "Первый коммит", файл file2.docx с текстом "Первый коммит"
5. Добавить файлы в git и создать commit с именем First commit
6. Изменить текст в file2.docx на "Второй коммит"
7. Создать commit с именем Second commit
8. Вернуть директорию в состояние, когда в обоих файлах текст "Первый коммит"
9. Вернуть директорию в состояние, когда в в file2.docx текст "Второй коммит"
10. Скопировать директорию git_local в git_local_1
11. В git_local_1 перевести в состояние, когда в обоих файлах текст "Первый коммит"
12. В файле file1.txt изменить текст на "Второй коммит" и закоммитить

Ветвления

- Зачем нужен .gitignore
- Как реально хранятся файлы и что такое ветвления
- Правила именования веток
- Создать несколько веток с разным состоянием файлов
- Слить одну из веток в основную ветку

Удаленный репозиторий

- Зарегистрироваться на GitVerse

Вопросы

- Чем отличается локальный репозиторий от простой папки
- Что будет если удалить папку .git
- Ситуация: локальный репозиторий, некий глюк, но файлы