**Git и Github**

**автор: alexKudryavtsev-web**

**Git, основные команды**

**Система контроля версий**

Система контроля версий – это утилита, позволяющая программистам сохранять разные версии программы и переключаться между ними.

Самая главная из них – это Git. В отличие от других она может существовать локально, и притом мало весить из-за того, что она сохраняет только изменения.

**Установка**

Установить Git можно с сайта <https://gitforwindows.org/>.

**Настройка**

Команда config позволяет изменять настройки. Вначале следует указать свое имя, желательно чтоб оно соответствовало имени Github и почту:

git config --global user.name "My Name"

git config --global user.email myEmail@example.com

**Инициализация**

В папке проекта, прежде всего надо инициализировать сам Git.

git init

После этого создастся скрытая папка .git, теперь данную папку можно назвать ***репозиторием***.

**Подготовка файлов и коммиты**

***Коммит*** – это фиксированное содержимое проекта. Можно представить коммит, как версию приложения или скриншот.

Чтоб сделать коммит сначала нужно подготовить файлы, изменения которых зафиксируются в коммите. Такие файлы называются ***подготовленные файлы***.

Сделать их можно командой add.

git add *имя файла*

git add -A

Первая форма подготовить один файл, а вторая подготовит все файлы в папке, т.е. репозитории. Допустим, мы сделали достаточно изменений в проекте и нам нужно это сохранить в коммите.

git commit –m “*сообщение*”

**Команда status**

Одна из самых популярных команд – это status, которая выводит информацию о репозитории.

А точнее ветку, состояние, подготовленные файлы (помечены зеленым цветом) и не отслеживаемые файлы (помечены красным).

git status

**Команда log и id коммита**

Просмотреть список всех коммитов и с информацией об авторе, дате создания и id можно командой log.

git log

Номер id коммита может показаться слишком длинным, но на практике используются лишь 7 первых символов.

**Команда show**

Команда show позволяет посмотреть новое в коммите.

git show *id*

**Команда diff**

Команда diff позволяет посмотреть разницу между изменения двух коммитов.

git diff id1 id2

git diff id1..id2

Обе записи эквиваленты.

**Переключение между коммитами**

Переключаться между коммитами можно командой checkout:

git checkout *id*

git checkout *id имя-файла*

Первая версий заменить всю папку, а вторая лишь указанный файл.

**Файл .gitignore**

Не все в проекте нужно коммить, например:

* node\_modules
* логи
* папки, созданные IDE
* системные файлы
* сборка проекта (build)
* ключи
* и т.д.

Для того чтоб это не попало в коммит при использовании add –A есть файл .gitignore. Например, так он может выглядеть:

\*.log

node\_modules/

build/

Чтоб данный файл заработал, его следует отдельно закоммитить.

**Github, удаленные репозитории**

Утилита Git хранит все локально. Но можно хранить на отдельном сервере, что позволяет делать Github. Это позволяет показать свой код всему миру и работать в команде.

**Создание репозитория на Github**

Сначала нужно зарегистрироваться на сайте и создать на нем новый репозиторий, который называется ***удаленный***.

Репозиторий, который создается на Git надо связать с репозиторием на Github. Для этого есть команда remote add.

git remote add origin [https://github.com/*имя*/*название проекта*.git](https://github.com/имя/название%20проекта.git)

Теперь локальный репозиторий связан с удаленным. Подмечу, что origin – это внутреннее название самого репозитория.

**Отправка изменений**

Теперь нужно отправить изменения с локального репозитория на удаленный командой push:

git push origin master

Для того чтоб отправить изменение на сервер нужно сделать коммит.

**Получение изменений**

Так теперь репозиторий находится на сервере, то другие программисты также захотят получить проекта. Это можно pull:

git pull origin master

Но прежде этого нужно быть подключенным к удаленному репозиторию.

**Клонирование**

Команда clone похожа на pull, но она просто выгружает содержимое и новый локальный репозиторий никак не связан с удаленным. Понятно, что подключение не нужно.

git clone *url репозитория*

**Ветки**

**Принцип веток**

***Ветка*** – это изолированная копия проекта с изменения. Веток бывает много и для каждой фичи создается отдельная ветка. Но, по-умолчанию, есть главная ветка master, которая содержит стабильную версию проекта.

Далее, когда эти фичи создаются, то их добавляют в главную ветку. Это называется **слиянием** (merge).

**Команда branch**

Команда branch позволяет просмотреть список веток, притом та ветка, на которой сейчас пользователь будет дополнена звездочкой.

git branch

Также это команда позволяет создать новую ветку.

git branch *имя*

**Переключение между ветками**

Знакомая уже команда checkout также переключает ветки.

git checkout *имя*

Теперь в новой ветке можно работать, так же как и в главной.

**Слияние**

Команда merge сливает одну в ветку в текущую ветку.

git merge *имя*

**Удаление ветки**

Команда branch –d позволяет удалить ветку.

git branch –d *имя*