Маша, привет,

Решил написать для тебя и для себя небольшой курс по Git.

Git используется практически везде в профессиональной разработке. Он позволяет совместно работать над одним проектом командам любого размера.

Нужен ли Git в индивидуальных проектах, когда разработчик всегда работает один, и нет необходимости интегрировать код, написанный другими программистами?

На мой взгляд, очень нужен:

* Git в сочетании с облачным хранилищем таким как Github (есть и другие) обеспечивает безопасное и структурированное хранение кода. Вы не потеряете свои разработки при краже ноутбука или сбое жесткого диска
* в ходе разработки вы не боитесь что-нибудь сломать неудачными изменениями в самый неподходящий момент. С Git всегда можно откатиться на любую предыдущую стадию проекта
* вы можете создавать рабочие версии – релизы, непрерывно продолжая работу над разработкой
* у вас есть вся история проекта, вы не боитесь чистить код, удаляя куски, от которых вы отказались. Иногда бывает нужно вернуться к старым идеям, они все хранятся в истории проекта в git
* можно подключить к проекту других разработчиков, если потребуется

Git - большой и сложный инструмент, с множеством команд и опций. В больших проектах часто существует специальная роль по интеграции и поддержке исходного кода. Индивидуальному разработчику или маленькой команде обычно нужно знать всего несколько команд и понимать основные принципы работы Git.

Разберем в этой статье, как можно работать одному разработчику в Git.

# Начало проекта

В командной строке из папки проекта:

$ git init

Далее зайти в Github, создать новый проект и следовать инструкции по подключению нового репозитория.

# Ветки master и development

Ветви (branches) в git это способ организации независимых историй развития кода. Код между ветвями может объединяться (merge), новая ветвь может начинаться от любого состояния другой ветви или с нуля.

В индивидуальной разработке обычно достаточно иметь 2 ветви:

master – это ветвь релизов. В ней содержится код, который можно выпускать наружу. В момент релиза там должны быть исправлены все критические ошибки и он должен иметь минимально завершенный вид. В начале проекта это может быть, например, всего одна работающая функция.

Ветвь master создается по умолчанию по команде “git init”

Ветвь без файлов невозможно комитить, поэтому если никаких файлов в проекте пока нет создаем файл readme.md:

$ echo “#” >> README.md

$ git add .

$ git commit -m "project start"

development – это ветвь разработки. Разработчик создает в ней код и регулярно добавляет изменения. Код в этой ветке не обязательно должен быть рабочим в каждом коммите.

Создаем ветвь development:

$ git checkout -b development master

После этого вы в ветке development

## Что помещать в git, а что исключить

Гит может хранить любые файлы, в том числе не текстовые. Рекомендуется хранить там только те файлы, которые вы создаете в процессе работы над проектом.

Иногда в программе используются какие-нибудь внешние данные, например, таблицы excel. Они могут лежать в рабочей директории и по умолчанию попадут в коммит по команде “git add .”

Чтобы этого избежать существует файл .gitignore .

В .gitignore перечисляем в текстовом виде те файлы или шаблоны, которые не должны попасть в репозиторий.

Пример:

<<

data/

venv/

.idea/

output.xlsx

!\*\*/\_\_pycache\_\_/

\*.pyc

\*.pptx

>>

Если какой-то ненужный файл (например, data.txt) попал в кэш коммита после команды “git add”, его можно удалить:

git rm data.txt --cached

Пусть мы хотим удалить все файлы с раширением xlsx из кэша и добавить их в .gitignore:

git rm -r --cached

echo “.xlsx” >> .gitignore

git add .