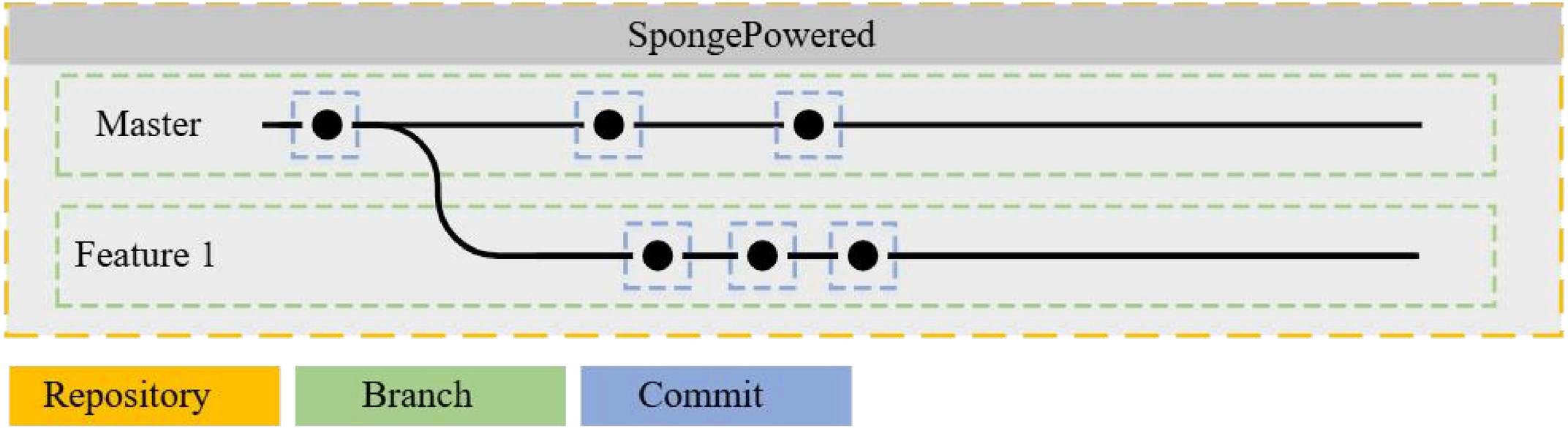
Контроль версии, ведение репозитория. Git, Github, Gitlab

“ИТ-класс”. Бэкенд-разработка на Python.
Бусыгин Дмитрий, 2025 год.

Почему контроль версий важен?

- Вы сохраняете предыдущую версию проекта, а потом можете легко “откатиться” и вернуть всё как было, в случае чего.
- Вы храните не версии проекта, а его обновления.
- Вся история работы храниться структурно и не приходится вручную искать, где что поменяли.
- Особенно это важно, когда работа ведётся в команде. Не надо каждый раз перебрасывать сжатые проекты через сторонние мессенджеры.

Немного терминологии



Репозиторий — место, где хранится весь проект: папки, файлы, изображения, код, записки и.т.д.

Коммит — это снимок проекта, который сохраняется в репозитории. Его делают разработчики каждый раз, когда что-то меняют в проекте, будь-то фикс бага, добавление нового функционала или изменение структуры.

Ветка — буквально ответвление от проекта, где, например, разрабатывается другой функционал.

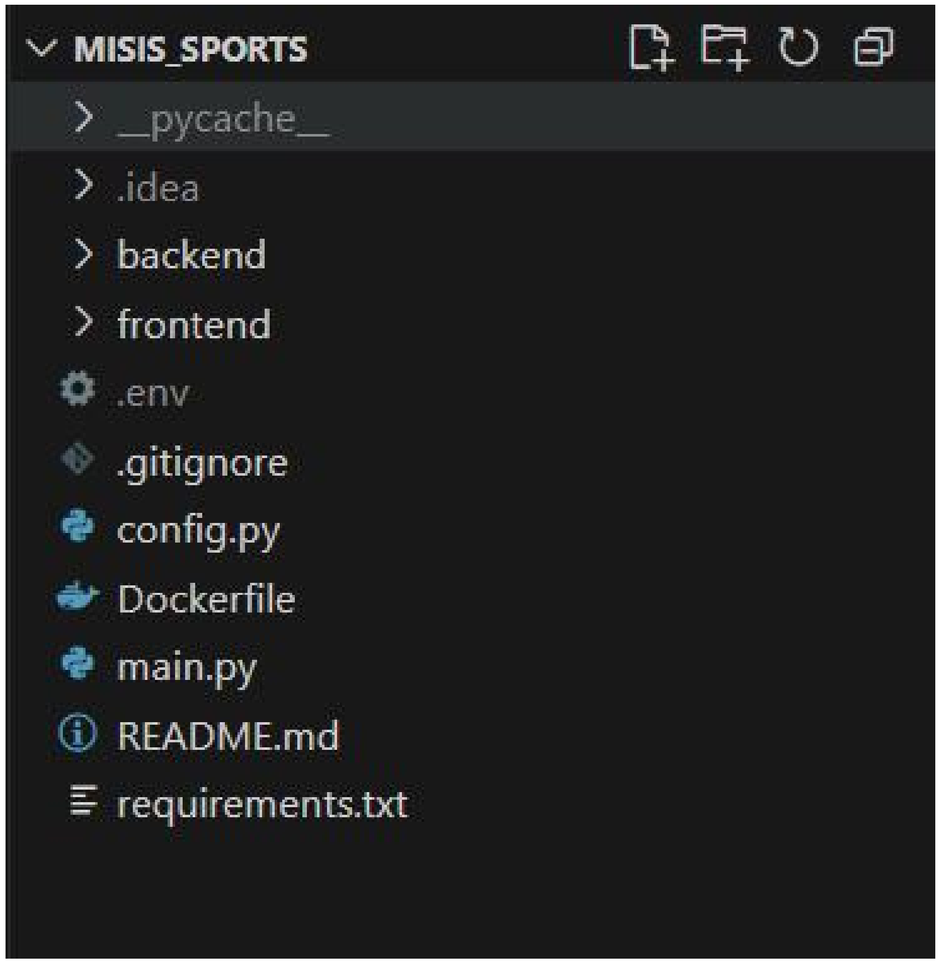
И пришел Линус Торвальдс (да-да, тот самый)...

Git — самая популярная система контроля версий на данный момент.
Она помнит всё: кто, когда и где именно были внесены изменения в проект.
И имеет всё то, что было описано ранее.

Github — облачный сервис для хранения своих проектов. Без него разработка проекта в команде невозможна (точнее ОООООЧЕНЬ неудобна).
Её суть в том, что туда можно выгрузить свои правки в проект и, наоборот, загрузить обновления от другого человека.

НЕ ПУТАТЬ! Git != Github

Из чего состоит любой проект



Сам код — бэкенд, фронтенд или другая архитектура приложения

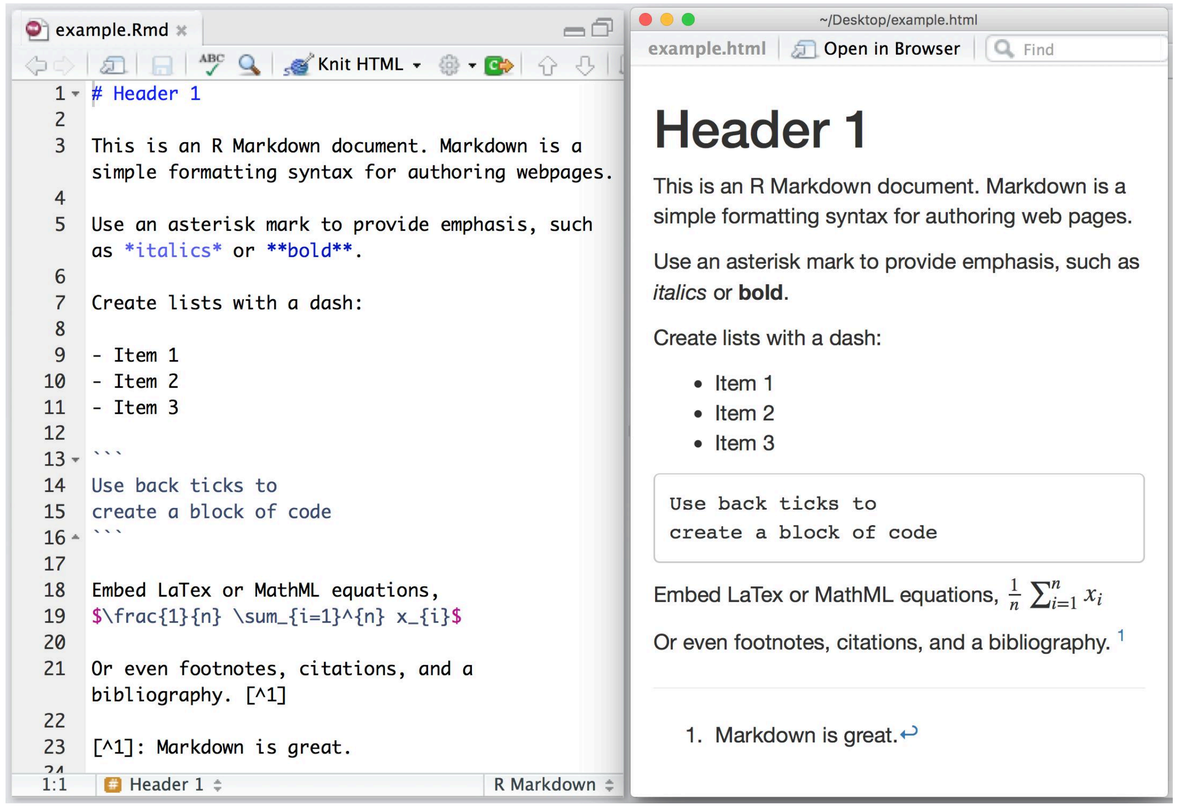
Что добавить в проект:
README.md — краткое описание проекта. О чём он, какой используется технический стек, как запустить на компьютере.

.gitignore — файл, в котором прописано, за какими другими файлами системе не нужно следить, например...

Что нельзя коммитить:
“Мусорные” папки — те, которые создаются при запуске и нужны для корректной работы (__pycache__, .idea, .venv и.т.д.)

“Секреты” — файлы (зачастую .env), хранящие в себе приватную информацию (ключи и пароли для доступа к разным внешним сервисам)

Пару слов про Markdown



На этом языке разметки пишут документацию к проектам.

Он проще чем HTML и остается читаемым даже без форматирования.

Подробнее про него вы можете прочитать здесь: https://www.markdownguide.org/

Ну или отдайте текст нейросетям и они напишут README.md за вас :)

Команды в Git Bash

```
Dima@DESK MINGW64 ~/find_a_walk (main)
$ git pull origin main
From https://github.com/DrPepper18/Findy
 * branch            main       -> FETCH_HEAD
Already up to date.

Dima@DESK MINGW64 ~/find_a_walk (main)
$ git add LICENSE

Dima@DESK MINGW64 ~/find_a_walk (main)
$ git commit -m "update LICENSE"
[main 56e21b3] update LICENSE
 1 file changed, 12 insertions(+), 17 deletions(-)

Dima@DESK MINGW64 ~/find_a_walk (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 669 bytes | 167.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DrPepper18/Findy.git
   219d3cf..56e21b3  main -> main
```

Пример работы в терминале

git init — создать локальный репозиторий

git remote add origin <repository_url> — привязать удаленный репозиторий

git pull — загрузить последнюю версию проекта

git clone <repository_url> = git init + git remote + git pull

git add * — выделить все файлы в проекте

git commit -m «message» — сделать коммит

git push — загрузить коммиты в удаленный репозиторий

git checkout «branch_name» — сменить ветку

git log — посмотреть историю действий

Больше команд тут: https://git-scm.com/docs

Что можете почитать дома

- https://thecode.media/git/ — “Что такое Git и как с ним работать”
- https://git-scm.com/book/en/v2 — “Pro Git book”
- https://git-scm.com/docs — Список команд в Git