Добавление данных в DVC

Для добавления данных в DVC используется команда `dvc add`, после чего данные оказываются под версионированием DVC, а в рабочей директории появляется файл с расширением `.dvc`, отражающий текущее состояние отслеживаемых данных.[1] [2] [3]

Основные шаги по добавлению данных в DVC

1. Поместить нужные файлы или директории в проект (например, в папку `data/`).
2. Выполнить команду:

   ```
   dvc add <путь_к_файлу_или_директории>
   ```

   Например:

   ```
   dvc add data/raw.csv
   ```

   или для директории:

   ```
   dvc add data/
   ```

   DVC создаст файл слепка (например, `raw.csv.dvc`), пропишет хэш, размер данных, количество файлов и путь в этот файл, а также поместит данные в локальный кэш.[2] [4] [5]

3. Добавить созданные `.dvc`-файлы и `.gitignore` в Git:

   ```
   git add data/raw.csv.dvc .gitignore
   git commit -m "Добавил данные под контроль DVC"
   ```

   Таким образом, Git будет хранить только метаинформацию, а сами объемные данные будут под контролем DVC.[3] [5] [6]

Пример структуры после добавления

```
data/
  ├── raw.csv
  └── raw.csv.dvc
.gitignore
```

`.gitignore` будет автоматически обновлен, чтобы исключить сами данные из Git[5].

Работа с директориями

Команда `dvc add` поддерживает добавление как отдельных файлов, так и целых директорий, в том числе с флагом `--recursive` для рекурсивного добавления вложенных файлов.[4]

Итог

Добавление данных в DVC — это просто: достаточно выполнить `dvc add <путь>` для файлов или папок данных, после чего `.dvc`-файлы попадают в Git, а оригинальные файлы попадают под версионирование DVC, оставаясь вне зоны видимости Git.[5] [2] [3]

⁂

Настроить удаленное хранилище для данных в DVC

Удаленное хранилище в DVC настраивается командами dvc remote add для добавления URL и dvc remote modify для указания параметров/аутентификации, после чего данные синхронизируются командами dvc push и dvc pull соответственно. Для удобства стоит назначить remote по умолчанию флагом -d и хранить секреты в .dvc/config.local через флаг --local.[20] [21] [22] [23] [24] [25]

Быстрый рецепт

- Установить поддержку нужного провайдера при установке через pip, например dvc[s3], dvc[gs], dvc[azure], dvc[ssh], dvc[gdrive], dvc[oss].[23]
- Добавить удаленное хранилище и сразу сделать его дефолтным: dvc remote add -d myremote s3://my-bucket/path (для S3 пример), что зафиксирует URL в .dvc/config и пометит myremote как core.remote.[26] [21] [20]
- При необходимости задать параметры/секреты: dvc remote modify --local myremote option value, используя --local для записи в Git-игнорируемый .dvc/config.local.[27] [22] [23]
- Отправить артефакты и датасеты: dvc push, а для получения на другой машине использовать dvc pull.[24] [25]

Примеры по провайдерам

```
- Amazon S3: dvc remote add -d myremote s3://<bucket>/<key>; аутентификация использует ст
```

```
- Google Cloud Storage: dvc remote add -d myremote gs://<bucket>/<path>; по умолчанию DVC
```

- Azure Blob Storage: dvc remote add myremote azure://<container> и при необходимости задать account_name и другие опции через dvc remote modify.[28]

- SSH/SFTP: dvc remote add -d myremote ssh://user@host:port/path; требуется доступ по SSH и SFTP, креды берутся из SSH-конфигурации или задаются в DVC.[29]
- HTTP/WebDAV: dvc remote add myremote https://...; схемы auth basic/custom задаются через dvc remote modify, секреты записывать с --local.[30] [27]

Управление remotes

- Просмотреть список и дефолтный remote: dvc remote list, где дефолт отмечен и хранится как core.remote.[21]
- Назначить/сменить remote по умолчанию: использовать dvc remote add -d или указать --remote в командах push/pull, а при отсутствии — берется core.remote.[24] [21]
- Для S3-совместимых сторажей настроить кастомный endpoint: dvc remote modify myremote endpointurl https://object-storage.example.com.[21]

Синхронизация данных

- Выгрузка в удаленное хранилище: dvc push загружает данные из локального DVC-кэша в указанный dvc remote.[24]
- Загрузка на рабочую машину: dvc pull скачивает данные из dvc remote в кэш и делает их видимыми в рабочем каталоге (эквивалент dvc fetch + dvc checkout).[25]

Практические советы

- Не коммитить секреты: использовать dvc remote modify --local, чтобы чувствительные параметры оказались в .dvc/config.local, который игнорируется Git.[22] [27]
- DVC часто использует уже настроенные локальные креденшалы провайдеров (AWS, Azure, GCP), поэтому нередко достаточно только dvc remote add.[22]
- Для ускорения передачи используйте параллелизм: ключ -j в dvc push/dvc pull или настройку jobs в конфиге remote.[24]
- При установке через pip не забыть extras для нужного провайдера хранения (например, dvc[s3], dvc[azure], dvc[gs]).[23]

⁂

Работа с версиями данных после добавления в DVC

После добавления данных в DVC с помощью команды `dvc add` работа с версиями данных осуществляется через связку команд Git и DVC, где Git управляет метафайлами `.dvc`, а DVC — самими данными в кэше.[67] [68] [69]

Основной workflow версионирования

Создание новой версии данных:

1. Изменить данные в рабочей директории
2. Выполнить `dvc add <файл>` для обновления метафайла
3. Закоммитить изменения в Git: `git add <файл>.dvc && git commit -m "Обновил данные"`
4. При необходимости отправить данные в удаленное хранилище: `dvc push`[68] [67]

Переключение между версиями:

- Использовать `git checkout <коммит/ветка/тег>` для переключения метафайлов
- Затем `dvc checkout` для синхронизации данных с текущими метафайлами[69] [70] [71]

Пример возврата к предыдущей версии:

```
git checkout HEAD~1  # или конкретный хэш коммита
dvc checkout         # восстановить данные из кэша
```

Работа с ветками для экспериментов

DVC отлично интегрируется с ветвлением Git для проведения экспериментов :[72] [73]

Создание экспериментальной ветки:

```
git checkout -b experiment-new-model
# Изменить данные или параметры
dvc add data/
git add data/.dvc && git commit -m "Эксперимент с новыми данными"
```

Слияние успешных экспериментов:

```
git checkout main
git merge experiment-new-model
dvc checkout  # получить обновленные данные
```

При создании новых веток DVC не копирует все файлы данных — он только меняет метафайлы, что делает ветки очень легковесными.[72]

Использование тегов для важных версий

Для маркировки важных моментов (релизов моделей, завершенных экспериментов) используются Git-теги :[74] [68]

```
git tag -a v1.0 -m "Первая версия модели"
git push --tags  # отправить теги в удаленный репозиторий
```

Возврат к тегированной версии:

```
git checkout v1.0
dvc checkout
```

Команды для отслеживания изменений

Проверка статуса данных:

- `dvc status` — показывает изменения в данных, отслеживаемых DVC[75] [76]
- `git status` — показывает изменения в метафайлах `.dvc`
- `dvc diff` — показывает различия между версиями данных[77]

Просмотр истории изменений:

- `git log <файл>.dvc` — история изменений метафайла
- `dvc metrics show` — отображение метрик по версиям (если настроено)[67]

Восстановление данных

Если данные были случайно удалены из рабочей директории:

```
dvc checkout  # восстановить все отслеживаемые файлы
```

Или для конкретного файла:

```
dvc checkout <файл>.dvc
```

Если данные отсутствуют в локальном кэше:

```
dvc pull  # эквивалент dvc fetch + dvc checkout
```

Эксперименты с DVC Experiments

Для более продвинутой работы с экспериментами DVC предоставляет команды `dvc exp`: [73] [78]

```
dvc exp run --name "experiment-1"  # запуск именованного эксперимента
dvc exp show                       # сравнение экспериментов
dvc exp branch <exp-id> <branch>   # создание ветки из эксперимента
```

Основные принципы

- **Git управляет метаданными** — файлы `.dvc` содержат хэши и пути к данным
- **DVC управляет данными** — хранит файлы в кэше и удаленном хранилище
- **Синхронизация** — всегда нужно выполнять `dvc checkout` после `git checkout`

- **Легковесность** — ветки содержат только метафайлы, не дублируя данные[68] [69] [67]

Такой подход позволяет эффективно работать с большими наборами данных, сохраняя все преимущества версионирования Git для метаданных и обеспечивая масштабируемое хранение самих данных через DVC.

⁂

Подключение DVC с Git для совместной работы

Интеграция DVC с Git создает мощную систему для совместной работы команды с кодом и данными, где Git управляет кодом и метафайлами, а DVC — большими данными и моделями.[104] [105] [106]

Базовая настройка для команды

Инициализация проекта

1. Настройка Git-репозитория:

```
git init
git remote add origin <URL_вашего_репозитория>
```

2. Инициализация DVC:

```
dvc init
dvc config core.analytics false  # отключить аналитику
```

3. Настройка удаленного хранилища для данных:

```
# Добавление удаленного хранилища (например, S3)
dvc remote add -d myremote s3://bucket-name/path
# Или локальная папка для тестирования
dvc remote add -d local_remote /path/to/shared/storage
```

Автоматизация с Git Hooks

Установка автоматических хуков:

```
dvc install
```

Эта команда устанавливает три ключевых Git хука :[107]

- `post-checkout` — автоматически запускает `dvc checkout` после `git checkout`
- `pre-commit` — выполняет `dvc status` перед `git commit` для проверки состояния данных
- `pre-push` — автоматически запускает `dvc push` перед `git push`

Рабочий процесс команды

Добавление нового участника

1. Клонирование репозитория:

```
git clone <URL_репозитория>
cd <project_name>
```

2. Настройка DVC:

```
dvc install  # установка хуков
dvc pull     # получение актуальных данных
```

Работа с данными в команде

Основной workflow для команды:

1. **Получение обновлений:**

```
git pull
dvc pull  # синхронизация данных
```

2. **Внесение изменений:**

```
# Изменение данных
dvc add data/new_dataset.csv
git add data/new_dataset.csv.dvc .gitignore
git commit -m "Добавил новый датасет"
```

3. **Отправка изменений:**

```
dvc push  # загрузка данных в удаленное хранилище
git push  # отправка метафайлов в Git
```

Конфигурация для многопользовательской среды

Персонализированные настройки

Для работы нескольких пользователей на одной машине используйте локальные конфиги : [108]

```
# Каждый пользователь настраивает свои креденшалы
dvc remote modify --local myremote user <username>
dvc remote modify --local myremote password <password>
```

Структура конфигурационных файлов

DVC поддерживает иерархию конфигураций :[109]

1. `.dvc/config` — общие настройки проекта (трекается в Git)
2. `.dvc/config.local` — локальные настройки (игнорируется Git)
3. **Глобальные конфиги** — пользовательские настройки системы

Безопасность креденшалов

Хранение секретов в локальном конфиге:

```
# Секреты НЕ попадают в Git
dvc remote modify --local myremote access_key_id <key>
dvc remote modify --local myremote secret_access_key <secret>
```

Продвинутые практики командной работы

Работа с ветками и экспериментами

Создание экспериментальных веток:

```
git checkout -b experiment-new-model
# Изменение данных или кода
dvc add data/
git add data/.dvc && git commit -m "Эксперимент с новыми данными"
dvc push
git push --set-upstream origin experiment-new-model
```

Слияние успешных экспериментов:

```
git checkout main
git merge experiment-new-model
dvc checkout  # синхронизация данных
```

Использование тегов для релизов

Создание версионированных релизов:

```
git tag -a v1.0 -m "Релиз модели v1.0"
dvc push  # убедиться, что данные загружены
git push --tags
```

Автоматизация с pre-commit

Для более продвинутой автоматизации можно использовать pre-commit framework :[107]

```
# .pre-commit-config.yaml
repos:
- repo: https://github.com/iterative/dvc
  rev: 3.56.0
  hooks:
  - id: dvc-pre-commit
    language_version: python3
    stages: [pre-commit]
  - id: dvc-pre-push
    language_version: python3
    stages: [pre-push]
  - id: dvc-post-checkout
    language_version: python3
    stages: [post-checkout]
```

Лучшие практики для команд

Организация проекта

Рекомендуемая структура :[110]

```
project/
├── data/
│   ├── raw/           # исходные данные
│   ├── processed/     # обработанные данные
│   └── models/        # модели ML
├── src/               # исходный код
├── .dvc/
├── .git/
└── dvc.yaml          # определение пайплайнов
```

Протокол работы команды

1. **Синхронизация перед работой:** всегда выполнять `git pull && dvc pull`
2. **Атомарные коммиты:** связывать изменения кода и данных в одном коммите[106]
3. **Описательные сообщения:** четко описывать изменения в данных
4. **Регулярное обновление:** частая синхронизация с командой
5. **Тестирование пайплайнов:** проверка воспроизводимости перед отправкой

Решение конфликтов

При конфликтах в DVC-файлах:

```
git pull  # получить изменения
dvc checkout  # синхронизировать данные
# Решить конфликты в .dvc файлах вручную
git add . && git commit -m "Решение конфликтов DVC"
```

Мониторинг и отладка

Проверка состояния

```
dvc status    # проверить состояние данных
git status    # проверить состояние кода
dvc diff      # сравнить версии данных
```

Отключение хуков при необходимости

Если хуки мешают (например, при проблемах с удаленным хранилищем):

```
rm .git/hooks/pre-push  # временно отключить pre-push хук
# или использовать --no-verify при коммите
git push --no-verify
```

Такая интеграция DVC с Git обеспечивает эффективную совместную работу команды, автоматизирует рутинные операции и гарантирует синхронизацию как кода, так и данных между всеми участниками проекта.[111] [104] [106]

⁂

1. https://dvc.org/doc/command-reference/add
2. https://habr.com/ru/companies/funcorp/articles/548942/
3. https://habr.com/ru/companies/raiffeisenbank/articles/461803/
4. https://900913.ru/tldr/common/en/dvc-add/
5. https://habr.com/ru/articles/761072/
6. https://matakov.com/treking-eksperimentov-s-mlflow-i-dvc/
7. https://learn.microsoft.com/ru-ru/windows/win32/termserv/dvc-plug-in-registration
8. https://alexott.net/ru/writings/emacs-vcs/EmacsDVC.html
9. https://dvc.org/doc/user-guide/project-structure/dvc-files
10. https://www.youtube.com/watch?v=qBZsS9Jgloo
11. https://techrocks.ru/2020/12/12/git-for-data-science-using-dvc/
12. https://github.com/iterative/dvc/issues/9352

13. https://proglib.io/p/git-dlya-data-science-kontrol-versiy-modeley-i-datasetov-s-pomoshchyu-dvc-2020-12-02
14. https://dvc.org/doc/start
15. https://dvc.org/doc/command-reference/remote/add
16. https://ods.ai/tasks/mlops24-hw-8
17. https://www.reddit.com/r/mlops/comments/19e4zfk/downloading_dataset_from_remote_with_dvc/
18. https://temofeev.ru/info/articles/dvc-git-dlya-dannykh-na-primere-ml-proekta/
19. https://bigdataschool.ru/blog/mlflow-vs-dvc-for-data-version-control-in-mlops/
20. https://dvc.org/doc/command-reference/remote/add
21. https://dvc.org/doc/command-reference/remote
22. https://dvc.org/doc/user-guide/data-management/remote-storage
23. https://dvc.org/doc/command-reference/remote/modify
24. https://dvc.org/doc/command-reference/push
25. https://dvc.org/doc/command-reference/pull
26. https://dvc.org/doc/user-guide/data-management/remote-storage/amazon-s3
27. https://dvc.org/doc/user-guide/data-management/remote-storage/http
28. https://dvc.org/doc/user-guide/data-management/remote-storage/azure-blob-storage
29. https://dvc.org/doc/user-guide/data-management/remote-storage/ssh
30. https://dvc.org/doc/user-guide/data-management/remote-storage/webdav
31. https://ritza.co/articles/dvc-s3-set-up-s3-as-dvc-remote/
32. https://apxml.com/courses/data-versioning-experiment-tracking/chapter-2-versioning-data-dvc/dvc-remote-storage-config
33. https://stackoverflow.com/questions/76336953/how-to-add-existing-data-via-dvc
34. https://dagshub.com/blog/configure-a-dvc-remote-without-a-devops-degree/
35. https://campus.datacamp.com/courses/introduction-to-data-versioning-with-dvc/dvc-configuration-and-data-management?ex=7
36. https://jisap.tecnalia.com/docs/datasets/cli/
37. https://stackoverflow.com/questions/65771384/data-version-control-dvc-cannot-push-to-remote-storage-because-querying-cache
38. https://mlops.community/learn/metadata-storage-and-management/dvc/
39. https://dvc.org/doc/user-guide/project-structure/configuration
40. https://dvc.org/doc/command-reference
41. https://dvc.org/doc/command-reference/exp/push
42. https://dvc.org/doc/command-reference/install
43. https://dvc.org/doc/command-reference/exp/pull
44. https://dvc.org/doc/start
45. https://dvc.org/doc/command-reference/get
46. https://mapattacker.github.io/ai-engineer/dvc/
47. https://cfl.readthedocs.io/en/latest/more_info/dvc_intro.html

48. https://campus.datacamp.com/courses/introduction-to-data-versioning-with-dvc/dvc-configuration-and-data-management?ex=10
49. https://www.kdnuggets.com/2022/07/16-essential-dvc-commands-data-science.html
50. https://dvc.org/doc/command-reference/status
51. https://discuss.dvc.org/t/how-to-automate-dvc-pull-request-for-a-single-file/666
52. https://dagshub.com/blog/solving-dvcs-failed-to-push-data-errors-by-adding/
53. https://dvc.org/doc/command-reference/init
54. https://dvc.org/doc/command-reference/fetch
55. https://dvc.org/blog/using-gcp-remotes-in-dvc
56. https://stackoverflow.com/questions/77108504/using-dvc-with-gcloud-build
57. https://blog.devgenius.io/how-to-connect-dvc-to-gcp-bucket-remote-storage-to-store-and-version-your-data-faf98292b3f4
58. https://anderfernandez.com/en/blog/dvc-tutorial-mlops-data-version-control/
59. https://mlops.swiss-ai-center.ch/part-2-move-the-model-to-the-cloud/chapter-22-move-the-ml-experiment-data-to-the-cloud/
60. https://towardsdatascience.com/large-data-versioning-with-dvc-and-azure-blob-storage-a-complete-guide-b97344827c81/
61. https://github.com/iterative/dvc/issues/7881
62. https://dvc.org/blog/azure-remotes-in-dvc
63. https://discuss.dvc.org/t/dvc-remote-cannot-be-setup/1984
64. https://github.com/iterative/dvc/issues/2200
65. https://stackoverflow.com/questions/73651050/dvc-imports-authentication-to-blob-storage
66. https://discuss.dvc.org/t/how-use-dvc-with-azure-blob-storage/158
67. https://habr.com/ru/companies/raiffeisenbank/articles/461803/
68. https://proglib.io/p/git-dlya-data-science-kontrol-versiy-modeley-i-datasetov-s-pomoshchyu-dvc-2020-12-02
69. https://dvc.org/doc/command-reference/checkout
70. https://dvc.org/doc/start
71. https://discuss.dvc.org/t/get-older-version-of-data-files/18
72. https://www.dasca.org/world-of-data-science/article/effortless-data-and-model-versioning-with-dvc
73. https://www.ridgerun.ai/post/experiment-tracking-with-dvc
74. https://wiki.merionet.ru/articles/cto-takoe-tegi-versii-v-git-i-kak-imi-polzovatsia
75. https://dvc.org/doc/command-reference/status
76. https://gitlab.deepschool.ru/a.kravchuk/cvr-homeworks/-/tree/main/01-dvc
77. https://dvc.org/doc/command-reference/diff
78. https://dvc.org/doc/command-reference/exp/branch
79. https://pcnews.ru/blogs/data_version_control_dvc_versionirovanie_dannyh_i_vosproizvodimost_eksperimentov-915122.html
80. https://matakov.com/treking-eksperimentov-s-mlflow-i-dvc/
81. https://stackoverflow.com/questions/76199034/dvc-checkout-without-git

82. https://discuss.dvc.org/t/cant-go-to-former-version-of-dataset-with-dvc-checkout/615
83. https://bigdataschool.ru/blog/mlflow-vs-dvc-for-data-version-control-in-mlops/
84. https://dvc.org/doc/command-reference/commit
85. https://github.com/iterative/dvc/issues/599
86. https://www.youtube.com/watch?v=qBZsS9Jgloo
87. https://www.reddit.com/r/MachineLearning/comments/mrb096/discussion_should_i_be_using_dvc_data_version/
88. https://stackoverflow.com/questions/75945488/i-tried-dvc-checkout-and-ran-into-an-error
89. https://habr.com/ru/articles/761072/
90. https://apxml.com/courses/data-versioning-experiment-tracking/chapter-2-versioning-data-dvc/switching-dvc-data-versions
91. https://www.reddit.com/r/datascience/comments/aqkg59/does_anyone_use_data_version_control_dvc_thoughts/
92. https://teletype.in/@dt_analytic/dFj8672mBDc
93. https://www.youtube.com/watch?v=9ff4xU9zHGw
94. https://habr.com/ru/companies/funcorp/articles/548942/
95. https://kolodezev.ru/mlsysd11.html
96. https://alexott.net/ru/writings/emacs-vcs/EmacsDVC.html
97. https://git-scm.com/book/ru/v2/Основы-Git-Работа-с-тегами
98. https://git-scm.com/book/ru/v2/Основы-Git-Запись-изменений-в-репозиторий
99. https://www.nic.ru/help/chto-takoe-git-tegi-i-kak-s-nimi-rabotat6-sozdanie-udalenie-ispol6zovanie_11657.html
100. https://discuss.dvc.org/t/dvc-experiments-multiple-branches-workflow/1124
101. https://www.atlassian.com/ru/git/tutorials/inspecting-a-repository/git-tag
102. https://stackoverflow.com/questions/71338160/dvc-experiment-management-workflow
103. https://temofeev.ru/info/articles/dvc-git-dlya-dannykh-na-primere-ml-proekta/
104. https://proglib.io/p/git-dlya-data-science-kontrol-versiy-modeley-i-datasetov-s-pomoshchyu-dvc-2020-12-02
105. https://habr.com/ru/companies/funcorp/articles/548942/
106. https://habr.com/ru/articles/664946/
107. https://dvc.org/doc/command-reference/install
108. https://stackoverflow.com/questions/70928144/multiple-users-in-dvc
109. https://dvc.org/doc/user-guide/project-structure/configuration
110. https://arious.uk/ai/dvc-best-practice-config-files-best-practices
111. https://dvc.org/blog/collaborative-experiments
112. https://bigdataschool.ru/blog/mlflow-vs-dvc-for-data-version-control-in-mlops/
113. https://dvc.org/doc/studio/user-guide/team-collaboration
114. https://wiki.merionet.ru/articles/kak-ispolzovat-git-hooks-dlia-avtomatizacii-kommitov
115. https://www.youtube.com/watch?v=qBZsS9Jgloo
116. https://habr.com/ru/companies/dins/articles/584562/

117. https://www.reddit.com/r/MachineLearning/comments/mrb096/discussion_should_i_be_using_dvc_data_version/
118. https://www.codecentric.de/en/knowledge-hub/blog/remote-training-gitlab-ci-dvc
119. https://blog.openreplay.com/ru/автоматизация-проверок-кода-git-pre-commit-hooks/
120. https://mandarinshow.ru/news/id/30
121. https://dvc.org/doc/user-guide/data-management/remote-storage
122. https://vital.lib.tsu.ru/vital/access/services/Download/vital:14373/SOURCE01
123. https://pcnews.ru/blogs/data_version_control_dvc_versionirovanie_dannyh_i_vosproizvodimost_eksperimentov-915122.html
124. https://github.com/iterative/dvc/issues/1614
125. https://devopsway.ru/posts/day-05-git/
126. https://wiki.merionet.ru/articles/kak-ispolzovat-git-i-github-osnovy-kontrolya-versij
127. https://github.com/iterative/dvc/issues/9075
128. https://dvc.org/doc/command-reference/checkout
129. https://stackoverflow.com/questions/75047065/how-to-disable-dvc-git-hooks
130. https://www.python4data.science/en/24.3.0/productive/git/advanced/hooks/advanced.html
131. https://pre-commit.com
132. https://dvc.org/doc/command-reference/commit
133. https://git-scm.com/book/ms/v2/Customizing-Git-Git-Hooks
134. https://data-ai.theodo.com/en/technical-blog/automate-dvc-pipelines-reproduction-with-makefile
135. https://lakefs.io/data-version-control/dvc-best-practices/
136. https://dvc.org/doc/user-guide
137. https://github.com/iterative/dvc/issues/792
138. https://apxml.com/courses/data-versioning-experiment-tracking/chapter-4-integrating-dvc-mlflow/best-practices-integrated-workflows
139. https://git-scm.com/book/ru/v2/Приложение-C:-Команды-Git-Совместная-работа-и-обновление-проектов