



> Конспект > 2 урок > Версионирование данных (DVC)

> Оглавление

- > [Оглавление](#)
- > [Управление версиями](#)
 - > [SVC](#)
 - > [DVC](#)
- > [DVC](#)
 - > [DVC Команды](#)
 - > [Поддерживаемые Хранилища](#)
- > [Инициализация DVC](#)
 - > [Настройка места хранения](#)
- > [DVC ADD, PUSH/PULL, DIFF](#)
 - > [DVC ADD](#)
 - > [DVC PULL/PUSH](#)
 - > [DVC DIFF](#)
- > [DVC функционал и ошибки использования](#)
 - > [DVC функционал](#)
 - > [DVC ошибки](#)

> Управление версиями

> SVC

SVC (Source Version Control) — это практика отслеживания изменений в коде и управления ими. Подходят для **Software Engineering**.

Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Особенно это становится актуальным когда мы работаем в команде.



Схема работы с кодом

1. **Push** - сделали изменения в коде, отправляем их в удаленный репозиторий для контроля.
2. **Build** - данный этап происходит если требуется. Также здесь может происходить компиляция.
3. **Deploy** - отправляем внесенные изменения в пользование.

Данные системы классно справляются с задачей контроля версий кода, но для контроля версионности данных такие системы не подходят, т.к. хранят избыточное кол-во информации. В случае с большими данными избыточность может быть серьезной проблемой.

> DVC

DVC (Data Version Control) — это инструмент управления экспериментами с данными и машинным обучением. Подходят для **Data/ML Engineering**.

DVC позволяет версионировать данные отдельно от кода, не размещая данные в репозитории.

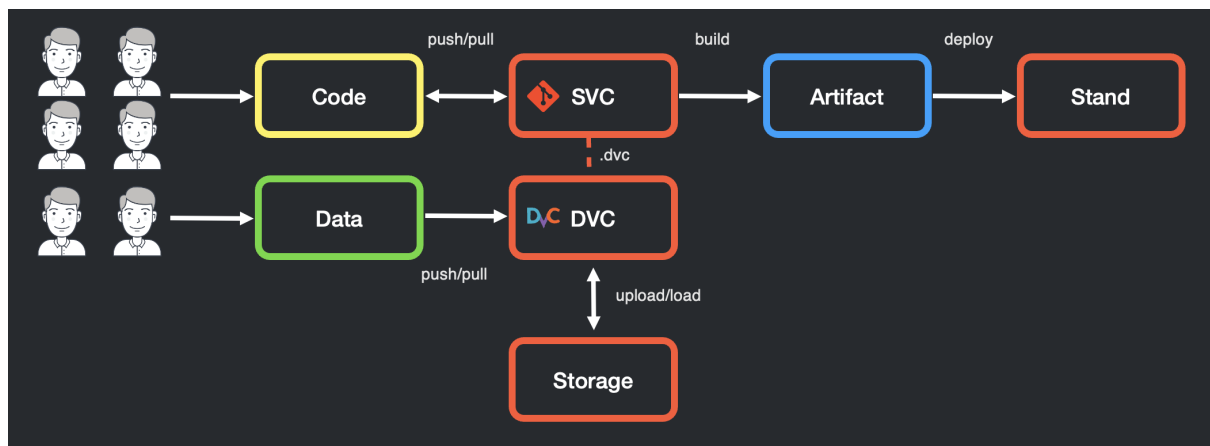
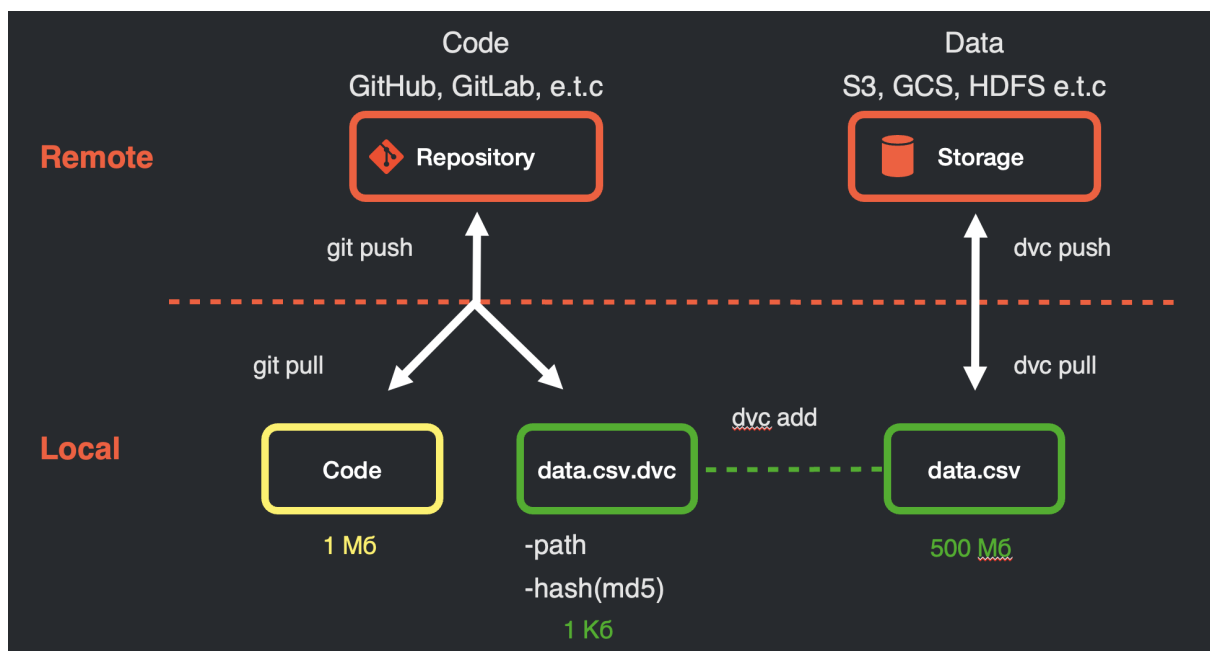


Схема работы DVC

DVC хранит в репозитории метаданные о используемых датасетах. Сами датасеты хранятся в специальных удаленных хранилищах (**storage**).

> DVC



В данном примере у нас есть удаленный git-репозиторий для управления кодом. В который мы можем вносить изменения из локального репозитория используя `git commit` и далее отправляя изменение через `git push` или забирая изменения из удаленного репозитория с помощью `git pull`. Так выглядит типичная работа с кодом.

Для параллельной работы с датасетами необходимо инициировать `dvc` аналогично инициализации `git`, но с дополнительным определением конфигурации удаленного хранилища. Это делается для того чтобы `dvc` понимал, где нужно хранить данные. Далее файл с данными добавляется в отслеживание с помощью команды `dvc add`. В результате создается файл с метаданными, который будет зарегистрирован в `git` (хранится путь к этому файлу и его hash-метка в md5). Для загрузки датасета на локальный репозиторий используется команда `dvc pull`, для загрузки в удаленное хранилище `dvc push`.

> DVC Команды

`dvc init` - инициализация DVC проекта. (регистрирует репозиторий, как dvc-репозиторий, позволяет в дальнейшем регистрировать файлы и выполнять разную конфигурацию)

`dvc add` - добавление файла в DVC. (регистрирует файл, dvc создаст сам создаст по этому файлу метаданные, которую нам нужно будет закоммитить в нашем репозитории исходного кода)

`dvc push` - загрузка файлов в удаленное хранилище.

`dvc pull` - выгрузка файлов из удаленного хранилища.

`dvc get` - выгрузка файла в папку по url. (позволяет достаточно быстро скачивать большие объемы данных, можно использовать, чтобы загрузить какой-то незарегистрированный датасет из внешнего хранилища)

`dvc checkout` - восстанавливает соответствующие версии DVC файлов. (актуализирует информацию из других версий метаданных к нашему комиту)

`dvc status` - отображения несоответствия кэша и хранилища.

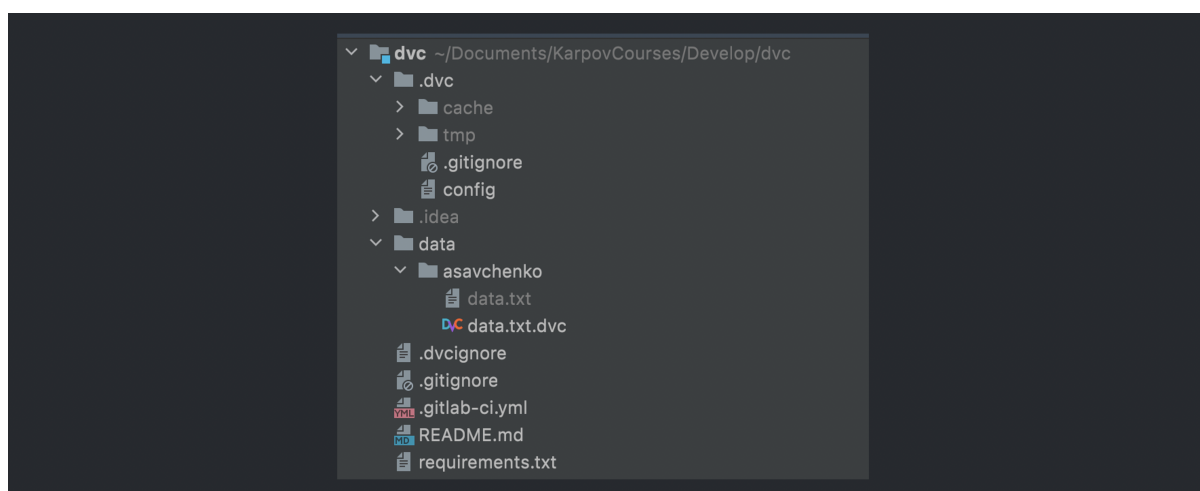
`dvc version` - сводная информация о DVC и настройках проекта. (будет указана информация об удаленном хранилище, которые используем для хранения файлов)

> Поддерживаемые Хранилища

- Amazon S3/S3-compatible storage
- Azure Blob Storage
- Google Cloud Storage
- Google Drive
- HDFS

- HTTP
- SSH
- WebDAV
- Local (Directory)

> Инициализация DVC



Для того, чтобы создать метаданные о **dvc** в репозитории исходного кода, используется команда `dvc init`. **DVC** создаст конфигурацию необходимую ему, так же создаст дополнительные файлы. Появится невидимая папка `.dvc`, где будет вся сводная информация о нашем dvc-репозитории. Так же при необходимости можно будет добавить в `.gitignore` все форматы файлов, которые хотим исключить из репозитория.



Пример заполнения .gitignore

> Настройка места хранения

После регистрации репозитория как dvc-репозитория, выполняется настройка места хранения. Для этого необходимо выполнить команду `dvc remote add`. В

данной команде указываем удаленное хранилище, которое будем использовать как хранилище данных. Например:

```
dvc remote add -d myremote s3://mybucket/path
```

, где
`myremote` - название коннекшена,
`s3://mybucket/path` - url нашего хранилища.

Также можно модифицировать хранилище. С помощью команды `dvc remote modify`. С этой командой нужно передавать имя нашего коннекшена и указывать параметры, которые хотим изменить. Например, для изменения региона можно использовать следующую команду:

```
dvc remote modify myremote region us-east-2
```

Наши команды модификации изменяют конфигурационный файл в нашей папке `.dvc`. Конфигурационный файл будет иметь название `.dvc/config`. Вместо команд, можно редактировать руками конфигурационный файл напрямую.

```
['remote "myremote"]  
url = s3://mybucket/path  
region = us-east-2  
[core]  
remote = myremote
```

Пример содержания конфигурационного файла

После настройки можно посмотреть какие соединения настроены в нашем dvc-хранилище с помощью команд `dvc remote list`. С помощью этой команды можно посмотреть какие коннекшены присутствуют в нашем dvc-репозитории и какие из них, мы можем использовать.

myremote	s3://mybucket/path
myremote2	s3://mybucket2/path

Пример вывода команды `dvc remote list`

> DVC ADD, PUSH/PULL, DIFF

> DVC ADD

Для того, чтобы зарегистрировать датасет в dvc-репозитории, необходимо выполнить команду `dvc add`, где надо указать путь к датасету, который хотим зарегистрировать. DVC создаст дополнительный файл с метаданными. Файл будет точно так же называться как датасет, но иметь суффикс `.dvc`. При этом файл всегда будет расположен рядом с нашим датасетом, в той же директории.

```
> dvc add data/test_data.csv

test_data.csv.dvc

outs:
- md5: d3045877db37345a62e3c90c0f6aacc9
  size: 348747
  path: test_data.csv
```

Пример вывода команды `dvc add`

После того как добавили датасет в dvc-репозиторий выполняются команды `git commit & push`, для того чтобы зафиксировать наши изменения. В изменение должен быть включен файл `.dvc`. Не путать с нашим датасетом, его в коммите быть не должно. После выполняется `push`, чтобы зафиксировать изменения в удаленном репозитории. Далее можно продолжать свою работу. Если мы изменили наш датасет, достаточно обновить метаданные. Для этого необходимо выполнить команду `dvc add`. DVC обновит метаданные. После надо будет выполнить `git commit & push`, чтобы зафиксировать изменения повторно.

> DVC PULL/PUSH

Чтобы разместить датасет в удаленном репозитории, необходимо выполнить команду `dvc push`. DVC отправит все зарегистрированные датасеты в наш удаленный репозиторий. Если нам необходимо выкачать свежий код на новый последний, и хотим подтянуть необходимые нам датасеты, необходимо выполнить команду `dvc pull`. DVC согласно нашим метафайлам по датасету будет выкачивать датасеты из удаленного хранилища. Когда выполняется

команда `dvc pull`, **DVC** читает созданные файлы, в которых присутствует суффикс `.dvc` с метаданной по нашим файлам и скачивает их.

> DVC DIFF

Когда работаем с git-репозиторием, мы можем смотреть изменения между коммитами. Точно так же **dvc** позволяет смотреть diff между коммитами **git**, а также смотреть какие файлы были изменены от одного коммита к другому. Для этого существует команда `dvc diff`, где можно указать нужный файл. В команде, помимо файла, необходимо указать айдишники коммитов. **DVC** будет отображать список файлов или один файл, если он действительно изменялся.

```
> dvc diff --targets data.txt -- 271ee8a ef7876
> dvc diff --targets data.txt -- 271ee8a HEAD

Modified:
  data.txt
files summary: 1 modified
```

Пример вывода команды `dvc diff`

> DVC функционал и ошибки использования







> DVC функционал

- Версионирование датасетов и моделей
 - Версионирование экспериментов
 - Версионирование пайплайнов для создания моделей и датасетов
 - Реестр данных через внешнее хранилище
 - Можно использовать как библиотеку Python
-

> DVC ошибки

DVC в некотором роде следует за git-репозиторием или другим репозиторием. **Git** версионирует наши изменения, а **DVC** лишь работает с

нашими файлами будто это датасеты или модели.

	<pre>> git push > dvc push</pre>		<pre>> dvc push > git push</pre>
	<pre>> dvc pull > git pull</pre>		<pre>> git pull > dvc pull</pre>
	<pre>> dvc add data.txt > [редактирование data.txt] > dvc push</pre>		<pre>> [редактирование data.txt] > dvc add data.txt > dvc push</pre>

Примеры частых ошибок