

Versioning and Sharing

Data and Models

Lesson 4

DVC Tools for Data Scientists & Analysts

2021



Course lessons

Lesson 1. Course Introduction

Lesson 2. Practices and Tools for Efficient Collaboration in ML projects

Lesson 3. Pipelines Automation and Configuration Management

Lesson 4. Versioning Data and Models

Lesson 5. Visualize Metrics & Compare Experiments with DVC and Studio

Lesson 6. Experiments Management and Collaboration

Lesson 7. Tools for Deep Learning Scenarios

Lesson 8. Review Advanced Topics and Use Cases



Lesson Outline

- ◆ Why does versioning matter?
- ◆ How does data versioning work?
- ◆ Tracking changes
- ◆ Versioning data and models
- ◆ Share and access data



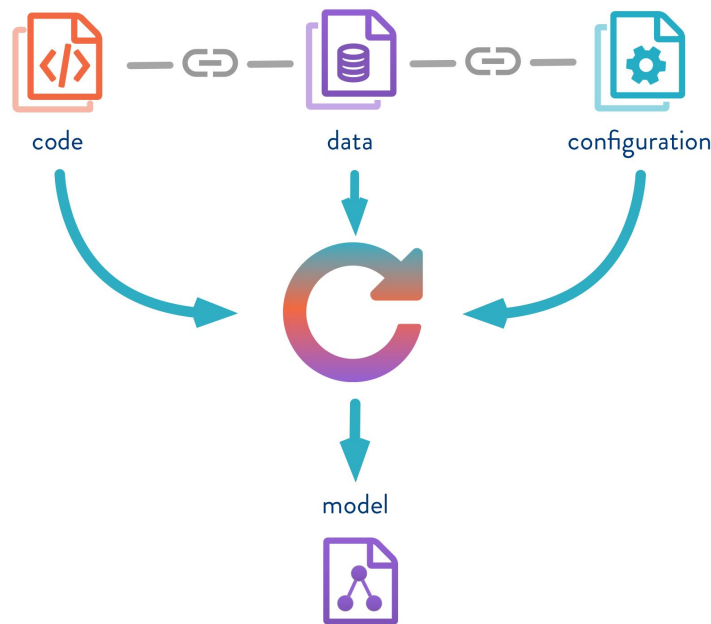


Why does versioning matter?

Reproducibility and ethical requirements



- ◇ Reproducible ML experiments requires versioned data, models and experiment artifacts
- ◇ Meet regulatory compliance and ethical AI requirements (e.g. for health, banking and other industries)

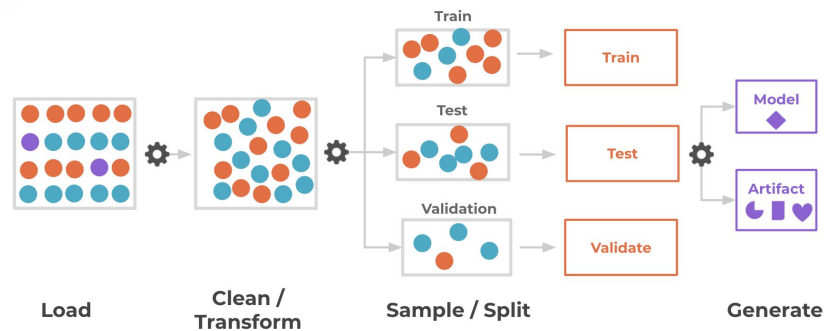


Data Workflow is complicated

need to be deterministic



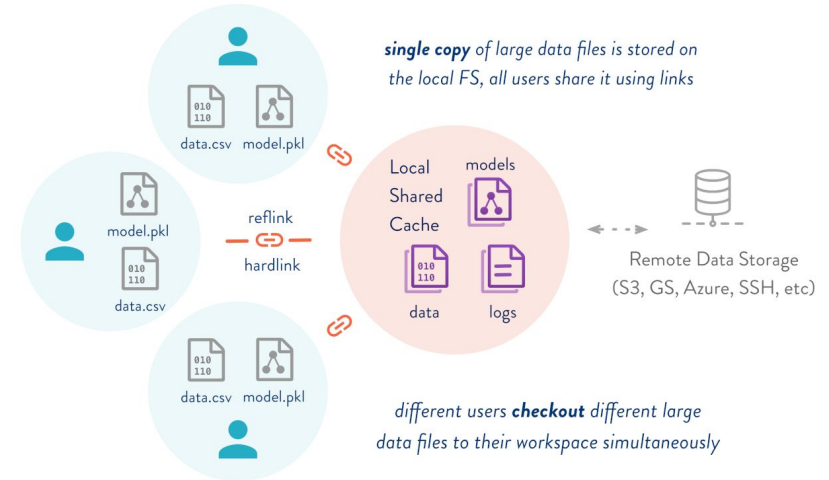
- ◇ Data processing takes a long time
- ◇ Expensive CPU/GPU resources
- ◇ Transferring data over the network is slow
- ◇ Need to be deterministic and reproducible



Improve team collaboration



- ◆ Avoid manual versioning
 - model.pkl
 - model_LogReg.pkl
 - model_LogReg_C1_pL2.pkl
 - model_LogReg_C01_pL2_final.pkl
 - model_SVM_v1.pkl
 -
- ◆ Share results and collaborate with a team



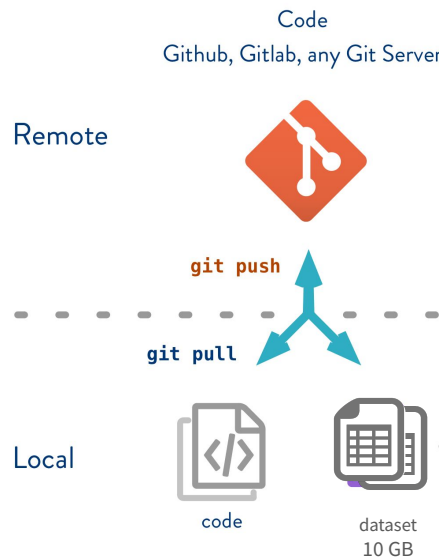


How does data versioning work?

How DVC works with data



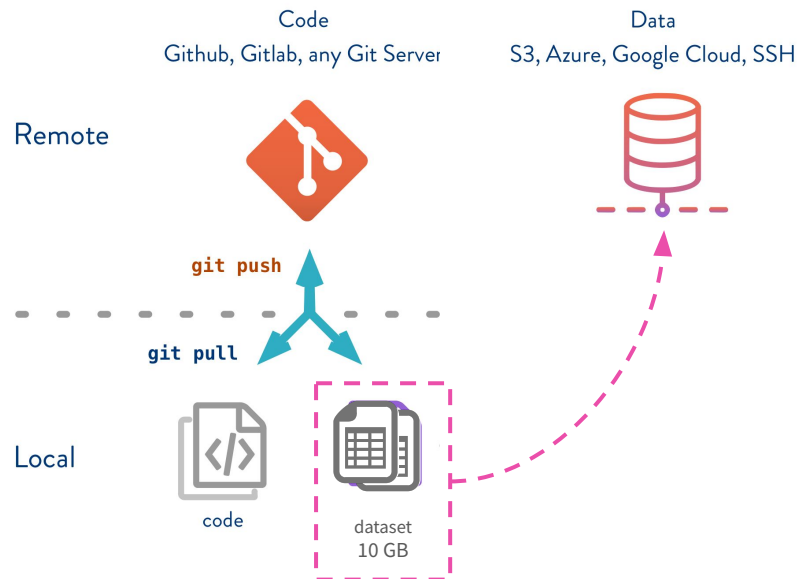
- ◇ We use Git to version the code
- ◇ How to deal with large data set?
 - ~~Add to Git version control~~
 - ~~Do versioning manually~~
 - Use DVC !!!



How DVC works with data



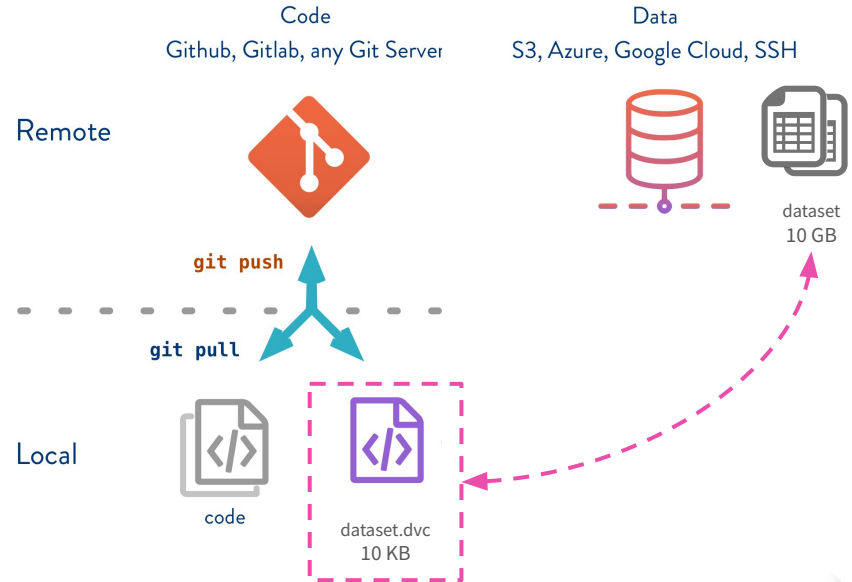
- ◆ Moves the data into cache or DVC storage



How DVC works with data



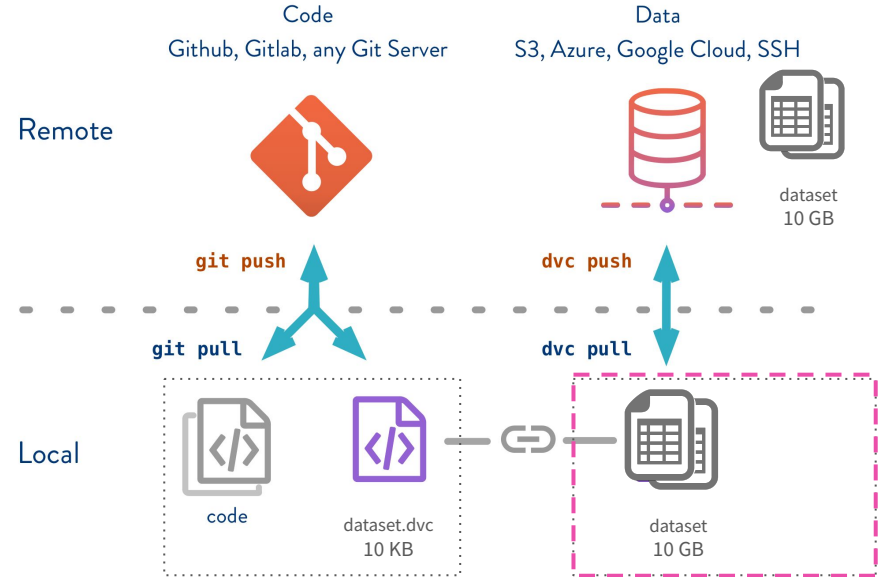
- ◇ Moves the data into cache or DVC storage
- ◇ Creates a metadata file with **.dvc** filename extension
 - a. to keep metadata about original file



How DVC works with data



- ◇ Moves the data into cache or DVC storage
- ◇ Creates a metadata file with **.dvc** filename extension
 - a. to keep metadata about original file
- ◇ Creates a link with the same name as original data file



Add file to version control



run command

```
$ dvc add data/iris.csv
```

output

```
Adding 'data/iris.csv' to 'data/.gitignore'.  
Saving 'data/iris.csv' to '.dvc/cache/57/fce90c81521889c736445f058c4838'.  
Saving information to 'data/iris.csv.dvc'.
```

Add **.dvc** file to Git



run command

```
$ git status
```

output

```
?? data/.gitignore  
?? data/iris.csv.dvc
```

run command

```
$ git add .  
$ git commit -m "Add a source dataset"
```

output

```
Add a source dataset  
2 files changed, 9 insertions(+)  
create mode 100644 data/.gitignore  
create mode 100644 data/iris.csv.dvc
```



Live code example

How data versioning works?



What happens under the hood?

Add file to version control



run command

```
$ dvc add data/data.xml -v
```

output

```
...
Computed stage: 'data/data.xml.dvc' md5: 'None'...
Saving 'data/data.xml' to
'dvc/cache/a3/04afb96060aad90176268345e10355'...
Removing '.../dvc/course/dvc-2-data-versioning/data/data.xml'...
Created 'reflink': .dvc/cache/a3/04afb96060aad90176268345e10355 ->
data/data.xml
Saving information to 'data/data.xml.dvc'
...
To track the changes with git, run:
```

```
git add data/data.xml.dvc
```

**Run command in
verbose mode to
see more details**

Add file to version control



run command

```
$ dvc add data/data.xml -v
```

output

```
...  
[Computed stage: 'data/data.xml.dvc' md5: 'None!...  
Saving 'data/data.xml' to  
'dvc/cache/a3/04afb96060aad90176268345e10355'...  
Removing '.../dvc/course/dvc-2-data-versioning/data/data.xml'...  
Created 'reflink': .dvc/cache/a3/04afb96060aad90176268345e10355 ->  
data/data.xml  
Saving information to 'data/data.xml.dvc'  
...  
To track the changes with git, run:
```

```
git add data/data.xml.dvc
```

Check is *****.dvc** file
already exists

Add file to version control



run command

```
$ dvc add data/data.xml -v
```

output

```
...  
Computed stage: 'data/data.xml.dvc' md5: 'None'...  
Saving 'data/data.xml' to  
'dvc/cache/a3/04afb96060aad90176268345e10355'...  
Removing '.../dvc/course/dvc-2-data-versioning/data/data.xml'...  
Created 'relink': dvc/cache/a3/04afb96060aad90176268345e10355 ->  
data/data.xml  
Saving information to 'data/data.xml.dvc'  
...  
To track the changes with git, run:
```

```
git add data/data.xml.dvc
```

Save file to
.dvc/cache
directory

Add file to version control



run command

```
$ dvc add data/data.xml -v
```

output

```
...  
Computed stage: 'data/data.xml.dvc' md5: 'None'...  
Saving 'data/data.xml' to  
'dvc/cache/a3/04afb96060aad90176268345e10355'...  
Removing '.../dvc/course/dvc-2-data-versioning/data/data.xml'...  
Created 'reflink': .dvc/cache/a3/04afb96060aad90176268345e10355 ->  
data/data.xml  
Saving information to 'data/data.xml.dvc'  
...  
To track the changes with git, run:  
  
git add data/data.xml.dvc
```

**Remove original
file**

Add file to version control



run command

```
$ dvc add data/data.xml -v
```

output

```
...  
Computed stage: 'data/data.xml.dvc' md5: 'None'...  
Saving 'data/data.xml' to  
'dvc/cache/a3/04afb96060aad90176268345e10355'...  
Removing '.../dvc/course/dvc-2-data-versioning/data/data.xml'...  
Created 'relink': .dvc/cache/a3/04afb96060aad90176268345e10355 ->  
data/data.xml  
Saving information to 'data/data.xml.dvc'
```

**Creating reflink to
the cached file**

```
...  
To track the changes with git, run:
```

```
git add data/data.xml.dvc
```

Add file to version control



run command

```
$ dvc add data/data.xml -v
```

output

```
...
Computed stage: 'data/data.xml.dvc' md5: 'None'...
Saving 'data/data.xml' to
'.dvc/cache/a3/04afb96060aad90176268345e10355'...
Removing '.../dvc/course/dvc-2-data-versioning/data/data.xml'...
Created 'reflink': .dvc/cache/a3/04afb96060aad90176268345e10355 ->
data/data.xml
Saving information to 'data/data.xml.dvc'
...
To track the changes with git, run:
```

```
git add data/data.xml.dvc
```

Create *****.dvc**
metafile

What is inside *****.dvc** files?



run command

```
$ cat data/data.xml.dvc
```

output

```
outs:
- md5: a304afb96060aad90176268345e10355
  path: data.xml
```

run command

```
$ du -sh .dvc/cache/**
```

output

```
36M
..dvc/cache/a3/04afb96060aad90176268345e10355
```

dvc add moved the data to
the project's **cache**, and
linked it back in a special
.dvc file

Add a directory to version control



run command

```
$ dvc add datadir
```

output

```
Adding...
!  
...  
95%|██████████| Computing file/dir hashes 1.70k/1.80k [00:00<00:00, 3.46kmd5/s]  
...  
98%|██████████| Saving datadir      1.76k/1.80k [00:02<00:00, 670file/s]  
...  
To track the changes with git, run:  
git add datadir.dvc
```

It works similarly
for data directory

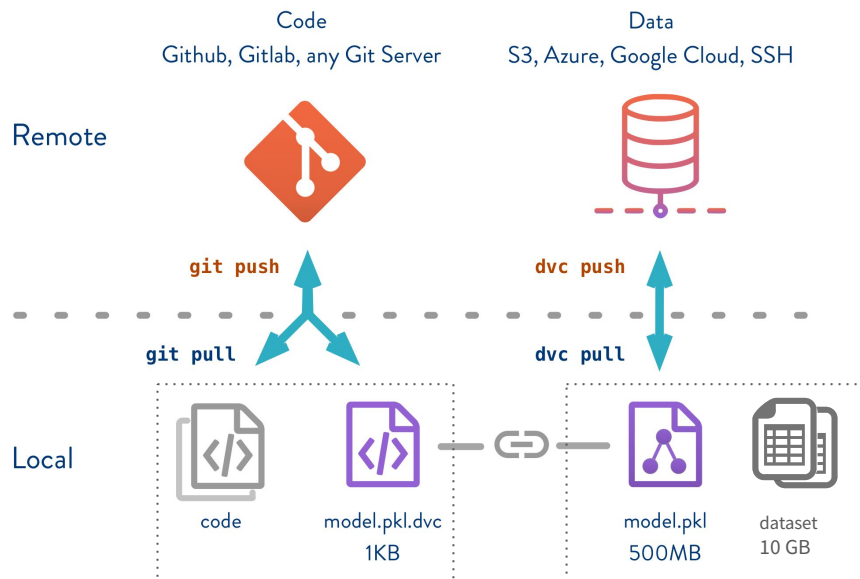


**Version and share
data and models
remote storage**

How DVC works with data



- ◆ Automated versioning of DVC pipeline outputs
 - **dvc.yaml** [deps/outs]
- ◆ Manually add files & folders for DVC version control
 - **dvc add**



Setup your remote storage



run command

```
$ mkdir -p /tmp/dvc  
$ dvc remote add -d myremote /tmp/dvc  
$ git commit .dvc/config -m "Configure local remote"
```

/tmp/dvc used as a local
'remote storage' named
myremote in this example

- ◇ **Remote storage types:** Google Drive, Amazon S3, Azure Blob Storage, Google Cloud Storage, Aliyun OSS, SSH, HDFS, and HTTP

Push data to remote storage



run command

```
$ dvc push
```

output

```
Preparing to upload data to '/tmp/dvc'  
Preparing to collect status from /tmp/dvc
```

```
...
```

```
Collecting information from remote cache...
```

```
Querying 1 hashes via object_exists
```

```
Indexing new .dir 'b6923e1e4ad16ea1a7e2b328842d56a2.dir' with '1800'  
nested files
```

```
Everything is up to date.
```

/tmp/dvc used as a
local 'remote storage'
in this example

Pull data from remote



run command

```
$ dvc pull
```

output

```
Preparing to download data from /tmp/dvc  
Preparing to collect status from /tmp/dvc  
Collecting information from local cache...  
Downloading...
```

```
1801 files fetched
```

**/tmp/dvc used as a
local 'remote storage'
in this example**



Live code example

Version and share data and models



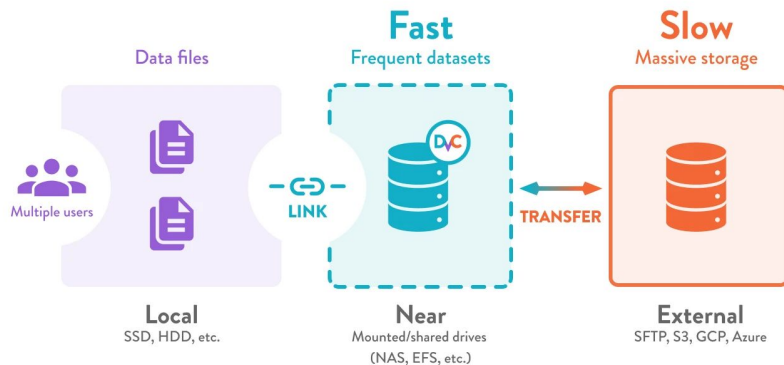
Tracking changes and switching between versions

Tracking changes

and switching between versions



- ◇ Track status of your data and models
 - **dvc status**
 - local workspace vs. `.dvc/cache`
 - local `.dvc/cache` vs. remote storage
- ◇ Switching version
 - **dvc checkout**



Track status of your data and models



run command (no changes)

```
$ dvc status
```

output

Data and pipelines are up to date.

run command (data has changed)

```
$ dvc status
```

output

```
datadir.dvc:
  changed outs:
    modified:    datadir
```

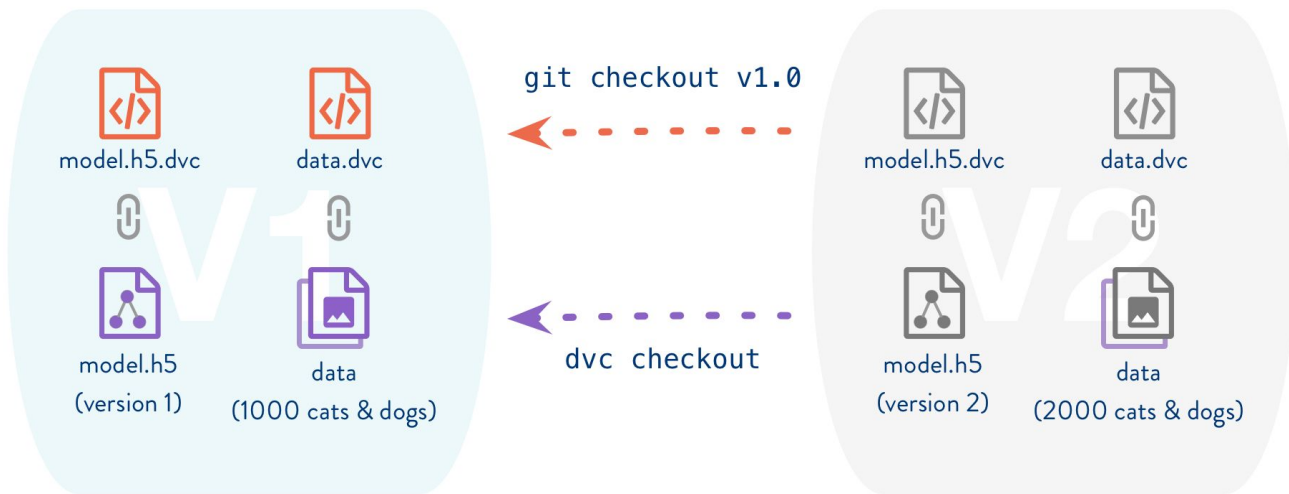
**type and name/hash of
changes detected**

Switching versions



run command

```
$ git checkout v1.0  
$ dvc checkout data.dvc
```





Live code example

Tracking changes and switching between versions



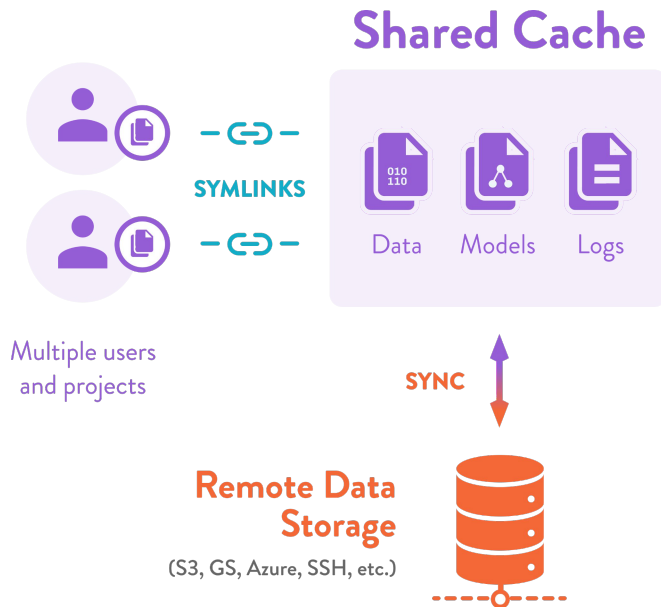
Data Access

Data access features

from CLI and Python interface



- ◇ How do I **download the newest model** to deploy it?
- ◇ How do I **download a specific version** of a model?
- ◇ How do I **reuse datasets** across different projects?
- ◇ **Get features used to train** the model for regulatory compliance?



Find a dataset



run command

```
$ dvc list https://github.com/iterative/dataset-registry use-cases
```

output

```
.gitignore  
cats-dogs  
cats-dogs.dvc
```

.dvc remote data
registry

Get dataset



run command

```
$ dvc get https://github.com/iterative/dataset-registry use-cases
```

output

Download dataset

Import the dataset



run command

```
$ dvc import https://github.com/iterative/dataset-registry use-cases
```

output

Download dataset and
add it under DVC
control



Live code example

Data Access

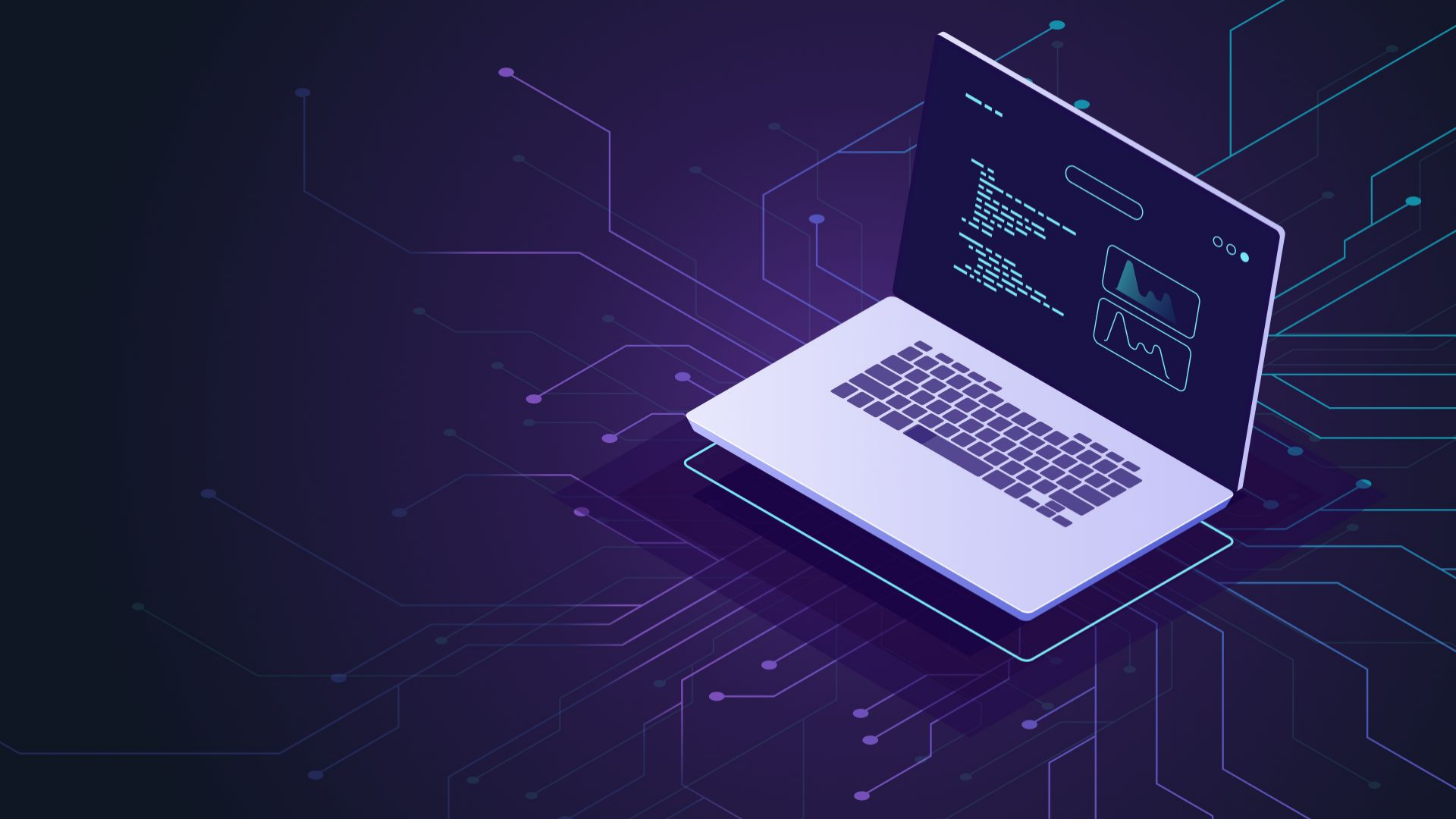


What have we learned?

What have we learned?



1. How DVC does versioning your data and models
2. How to setup versioning for data and models
3. How to share and access data and models
4. How to switch versions





Links

- ◆ Get Started: Data Versioning <https://dvc.org/doc/start/data-and-model-versioning>
- ◆ Get Started: Data and Model Access <https://dvc.org/doc/start/data-and-model-access>
- ◆ Versioning Data and Models <https://dvc.org/doc/use-cases/versioning-data-and-model-files>
- ◆ Sharing Data and Model Files <https://dvc.org/doc/use-cases/sharing-data-and-model-files>