Table of contents

Best 7 Data Version Control Tools That Improve Your Workflow With Machine Learning Projects

5 min

Jakub Czakon

14th November, 2022

Machine Learning Tools

Keeping track of all the data you use for models and experiments is not exactly a piece of cake. It takes a lot of time and is more than just managing and tracking files. You need to ensure everybody's on the same page and follows changes simultaneously to keep track of the latest version.

You can do that with no effort by using the right software! **A good data version control tool will allow you to have unified data sets with a strong repository of all your experiments.**

It will also enable smooth collaboration between all team members so everyone can follow changes in real-time and always know what's happening.

It's a great way to systematize data version control, improve workflow, and minimize the risk of

Table of contents

So check out these top tools for data version control that can help you automate work and optimize processes.

Data versioning tools are critical to your workflow if you care about reproducibility, traceability, and ML model lineage.

They help you get a version of an artifact, a hash of the dataset or model that you can use to identify and compare it later. Often you'd log this data version into your metadata management solution to make sure your model training is versioned and reproducible.

How to choose a data versioning tool?

To choose a suitable data versioning tool for your workflow, you should check:

- **Support for your data modality**: how does it support video/audio? Does it provide some preview for tabular data?
- **Ease of use**: how easy is it to use in your workflow? How much overhead does it add to your execution?
- **Diff and compare**: Can you compare datasets? Can you see the diff for your image directory?
- **How well does it work with your stack**: Can you easily connect to your infrastructure, platform, or model training workflow?

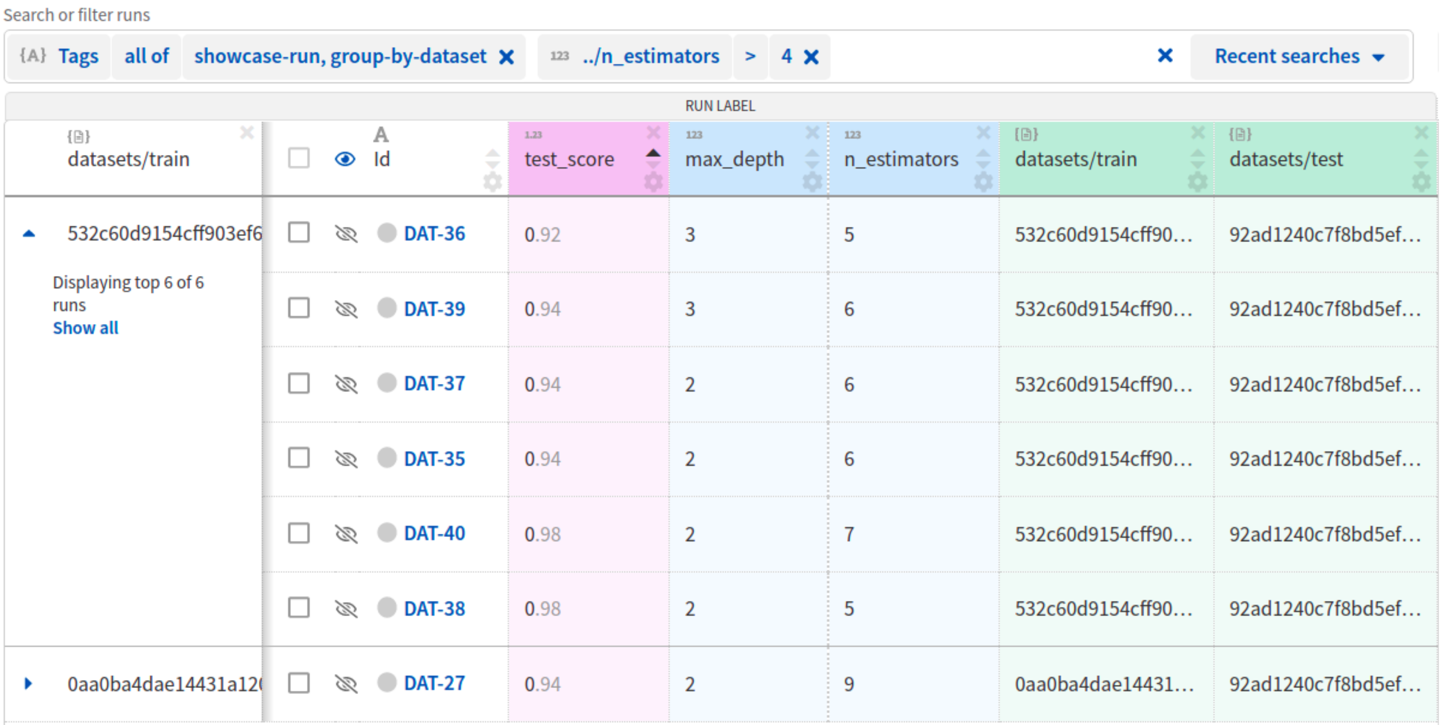
- **Can you get your team on board**: If your team does not adopt it, it doesn't matter how good the tool is. So keep your teammates skillset in mind and preferences in mind.

Here're are a few tools worth exploring.

Table of contents

Best data version control tools

1. Neptune



Search or filter runs

{A} Tags | all of | showcase-run, group-by-dataset ✕ | 123 ../n_estimators | > | 4 ✕ | ✕ | Recent searches

RUN LABEL

datasets/train	□	Id	test_score	max_depth	n_estimators	datasets/train	datasets/test
532c60d9154cff903ef6	□	DAT-36	0.92	3	5	532c60d9154cff90…	92ad1240c7f8bd5ef…
Displaying top 6 of 6 runs Show all	□	DAT-39	0.94	3	6	532c60d9154cff90…	92ad1240c7f8bd5ef…
	□	DAT-37	0.94	2	6	532c60d9154cff90…	92ad1240c7f8bd5ef…
	□	DAT-35	0.94	2	6	532c60d9154cff90…	92ad1240c7f8bd5ef…
	□	DAT-40	0.98	2	7	532c60d9154cff90…	92ad1240c7f8bd5ef…
	□	DAT-38	0.98	2	5	532c60d9154cff90…	92ad1240c7f8bd5ef…
0aa0ba4dae14431a12(	□	DAT-27	0.94	2	9	0aa0ba4dae14431…	92ad1240c7f8bd5ef…

Example dashboard in Neptune

Neptune is an ML metadata store that was built for research and production teams that run many experiments.

You can log and display pretty much any ML metadata from hyperparameters and metrics to videos,

Table of contents

Neptune artifacts let you version datasets, models, and other files from your local filesystem or any S3-compatible storage with a single line of code. Specifically, it saves:

- **Version** (hash) for the file or folder
- **Location** of the file or folder
- Folder **structure** (recursively)
- **Size** of the file or folder

Once logged, you can use Neptune UI to group runs on dataset versions or see how the artifacts changed between runs.

When it comes to data versioning, Neptune is a very lightweight solution, and you can get going quickly. That said, it may not give you everything you need data-versioning-wise.

On the other hand, you get experiment tracking and model registry all in one place and use a flexible metadata structure to organize training and production metadata the way you want to. It is like a dictionary or a folder structure that you create in code and display in the UI.

If you are wondering if it will fit your workflow:

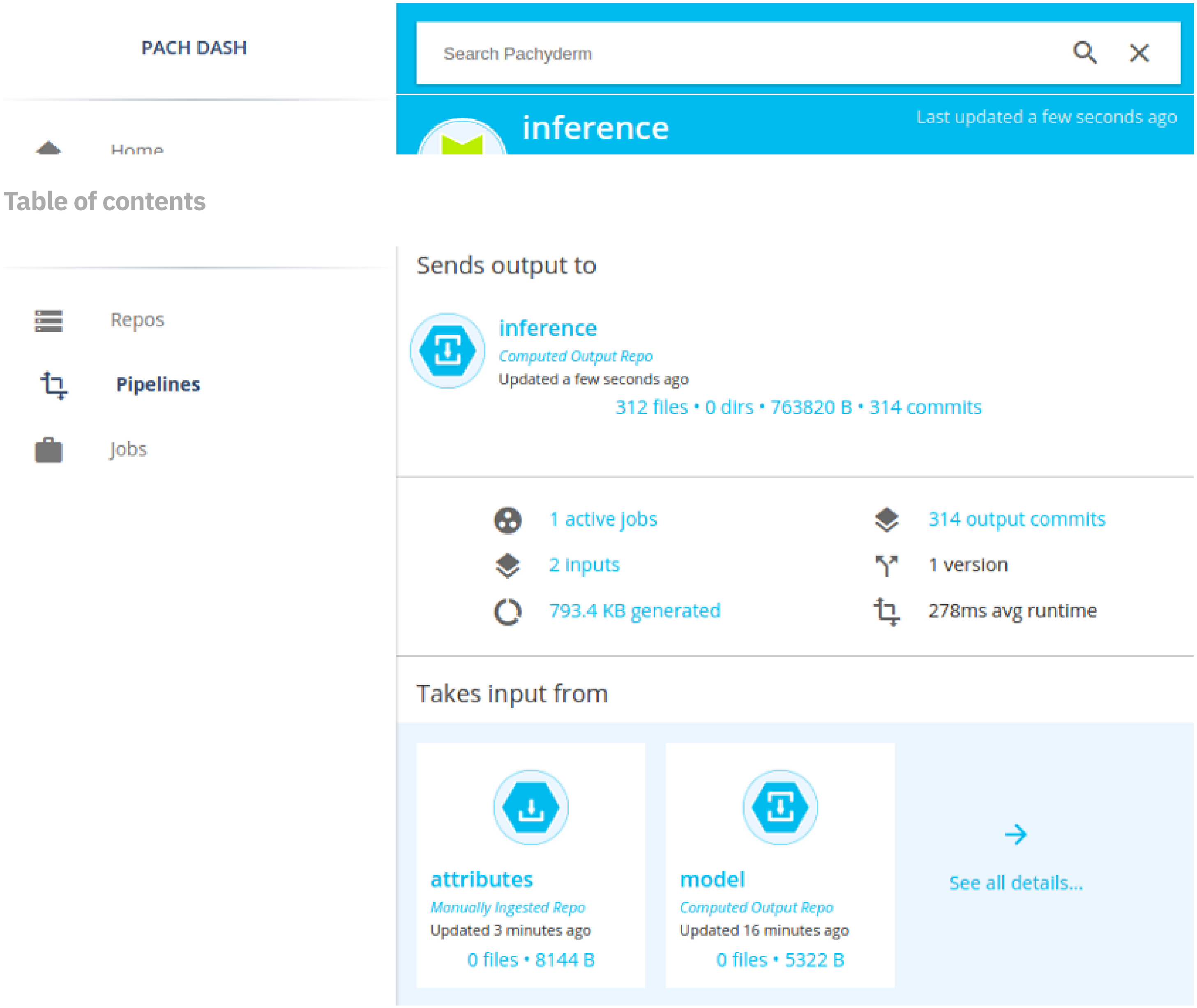
- check out case studies of how people set up their MLOps tool stack with Neptune
- explore an example public project about dataset versioning
- run a hello world or dataset versioning example in Colab and see for yourself

- but if you are like me, you would like to compare it to other tools in the space like DVC, Pachyderm, or wandb. So here are many deeper feature-by-feature comparisons to make the evaluation easier.

Table of contents

2. Pachyderm

Table of contents



inference

Last updated a few seconds ago

Sends output to

inference
Computed Output Repo
Updated a few seconds ago
312 files • 0 dirs • 763820 B • 314 commits

1 active jobs
314 output commits
2 inputs
1 version
793.4 KB generated
278ms avg runtime

Takes input from

attributes
Manually Ingested Repo
Updated 3 minutes ago
0 files • 8144 B

model
Computed Output Repo
Updated 16 minutes ago
0 files • 5322 B

→ See all details...

Pachyderm is a complete version-controlled data science platform that helps to control an end-to-end machine learning life cycle. It comes in three different versions, Community Edition (open-source, with the ability to be deployed anywhere), Enterprise Edition (complete version-controlled platform), and Hub Edition (a hosted version, still in beta).

It's a great platform for **flexible collaboration** on any kind of machine learning project.

Table of contents

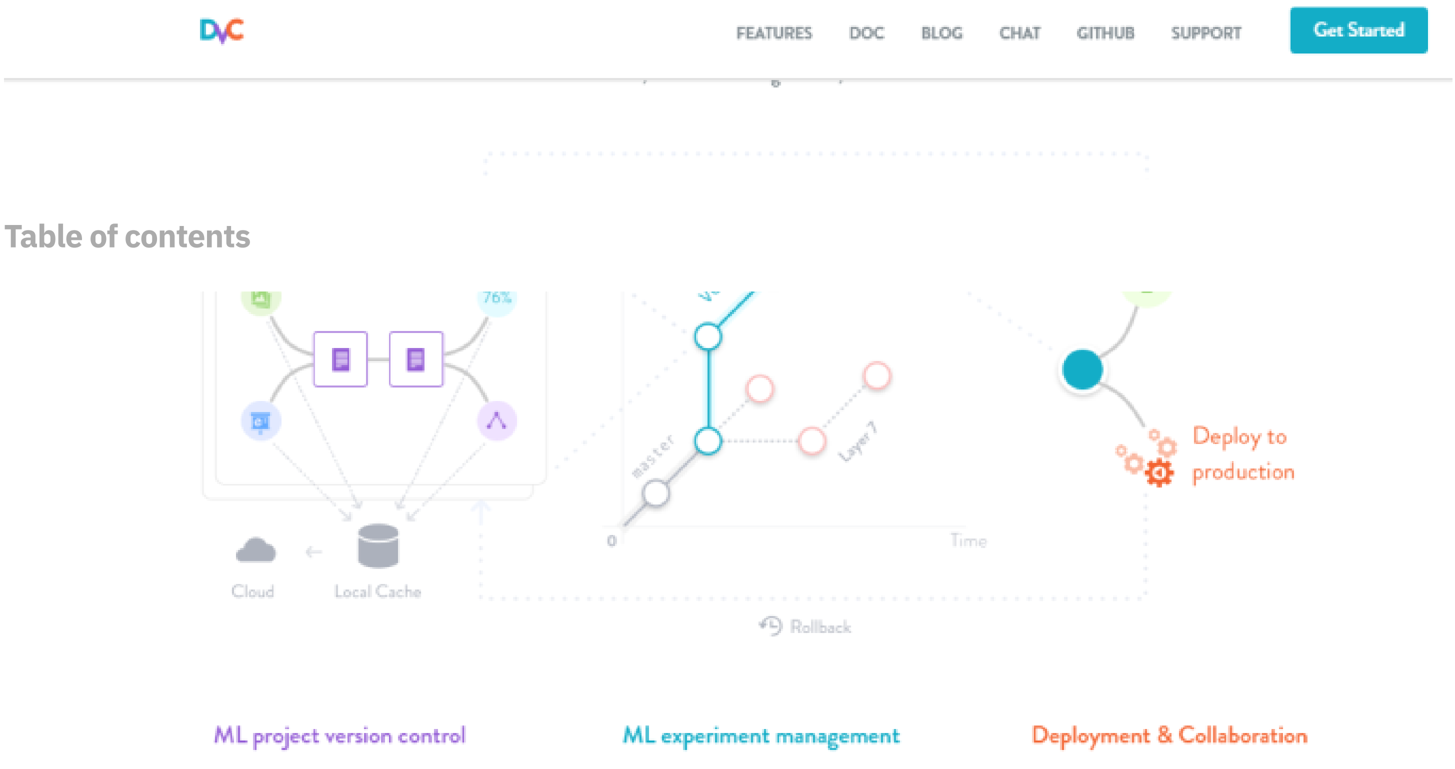
- Pachyderm lets you continuously update data in the master branch of your repo, while experimenting with specific data commits in a separate branch or branches
- It supports any type, size, and number of files including binary and plain text files
- Pachyderm commits are centralized and transactional
- Provenance enables teams to build on each other work, share, transform, and update datasets while automatically maintaining a complete audit trail so that all results are reproducible

Check also

The Best Pachyderm Alternatives

3. DVC

Table of contents



DVC is an open-source version control system for machine learning projects. It's a tool that lets you define your pipeline regardless of the language you use.

When you find a problem in a previous version of your ML model, DVC saves your time by leveraging code data, and pipeline versioning, to give you reproducibility. You can also train your model and share it with your teammates via DVC pipelines.

DVC can cope with versioning and organization of big amounts of data and store them in a well-organized, accessible way. It focuses on data and pipeline versioning and management but also has some (limited) experiment tracking functionalities.

DVC – summary:

- Possibility to use different types of storage— it’s storage agnostic
- Full code and data provenance help to track the complete evolution of every ML model
- Reproducibility by consistently maintaining a combination of input data, configuration, and the code that was initially used to run an experiment

Table of contents

- A built-in way to connect ML steps into a DAG and run the full pipeline end-to-end
- Tracking failed attempts
- Runs on top of any Git repository and is compatible with any standard Git server or provider

See also

DVC vs Neptune comparison

4. Git LFS

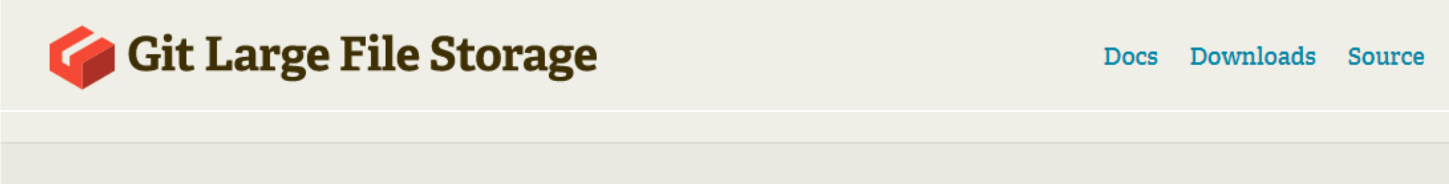
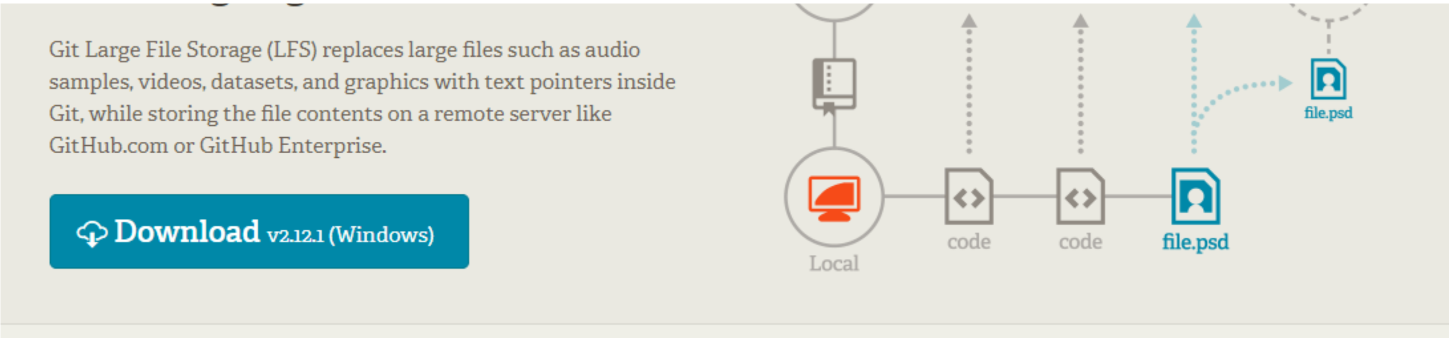


Table of contents

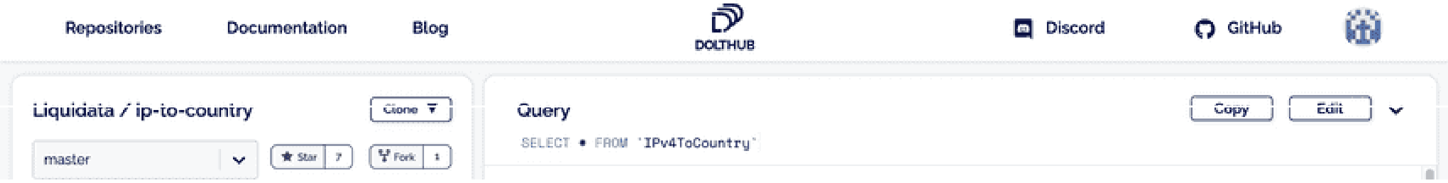


Git Large File Storage (LFS) replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise.

Git Large File Storage (LFS) is an open-source project. It **replaces large files** such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise.

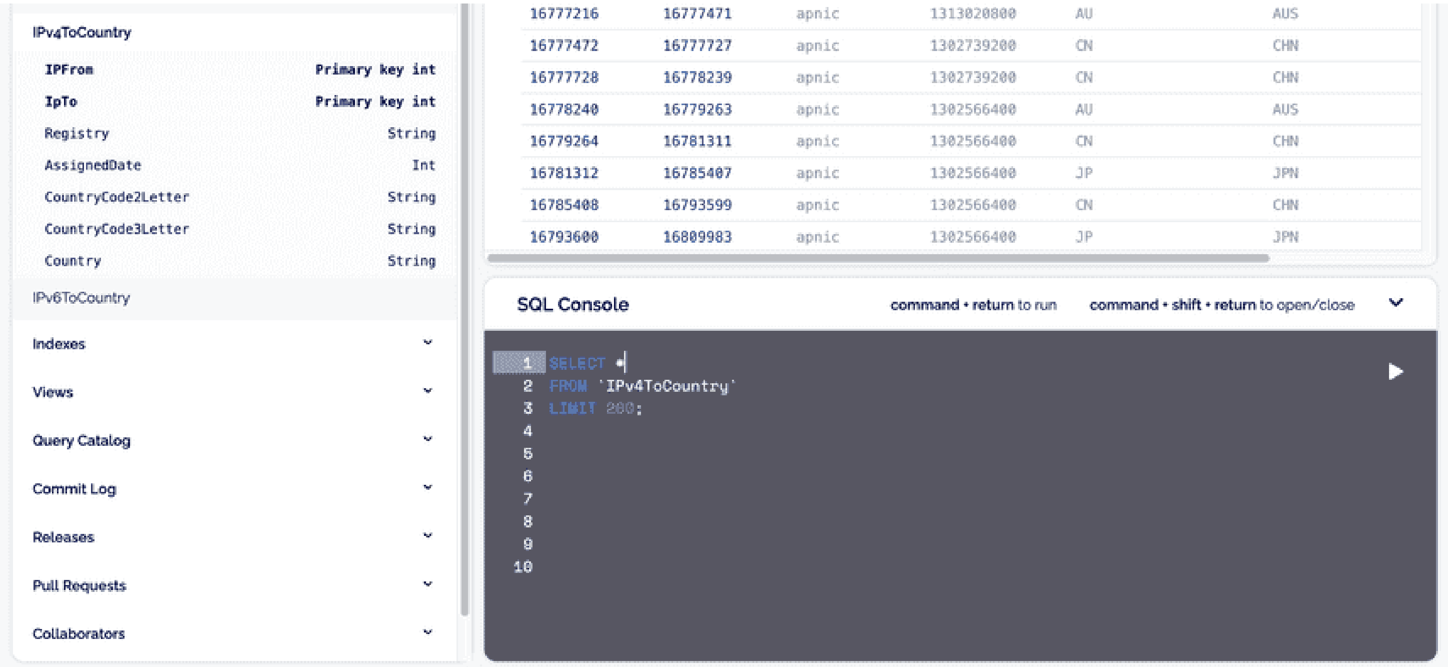
It allows you to **version large files**—even those as large as a couple GB in size—**with Git**, **host more** in your Git repositories with external storage, and to **faster clone and fetch** from repositories that deal with large files.

At the same time, you can keep your workflow and the same access controls and permissions for large files as the rest of your Git repository when working with a remote host like GitHub.



```
SELECT * FROM `IPv4ToCountry`
```

Table of contents



IPFrom	Primary key int
IpTo	Primary key int
Registry	String
AssignedDate	Int
CountryCode2Letter	String
CountryCode3Letter	String
Country	String

16777216	16777471	apnic	1313020800	AU	AUS
16777472	16777727	apnic	1302739200	CN	CHN
16777728	16778239	apnic	1302739200	CN	CHN
16778240	16779263	apnic	1302566400	AU	AUS
16779264	16781311	apnic	1302566400	CN	CHN
16781312	16785407	apnic	1302566400	JP	JPN
16785408	16793599	apnic	1302566400	CN	CHN
16793600	16809983	apnic	1302566400	JP	JPN

```
SELECT *
FROM `IPv4ToCountry`
LIMIT 200;
```

Dolt is a SQL database that you can *fork, clone, branch, merge, push, and pull* just like a git repository. Dolt allows data and schema to evolve together to **make a version control database a better experience**. It's a great tool to collaborate on with your team.

You can freely connect to Dolt just like to any MySQL database to run queries or update the data using SQL commands.

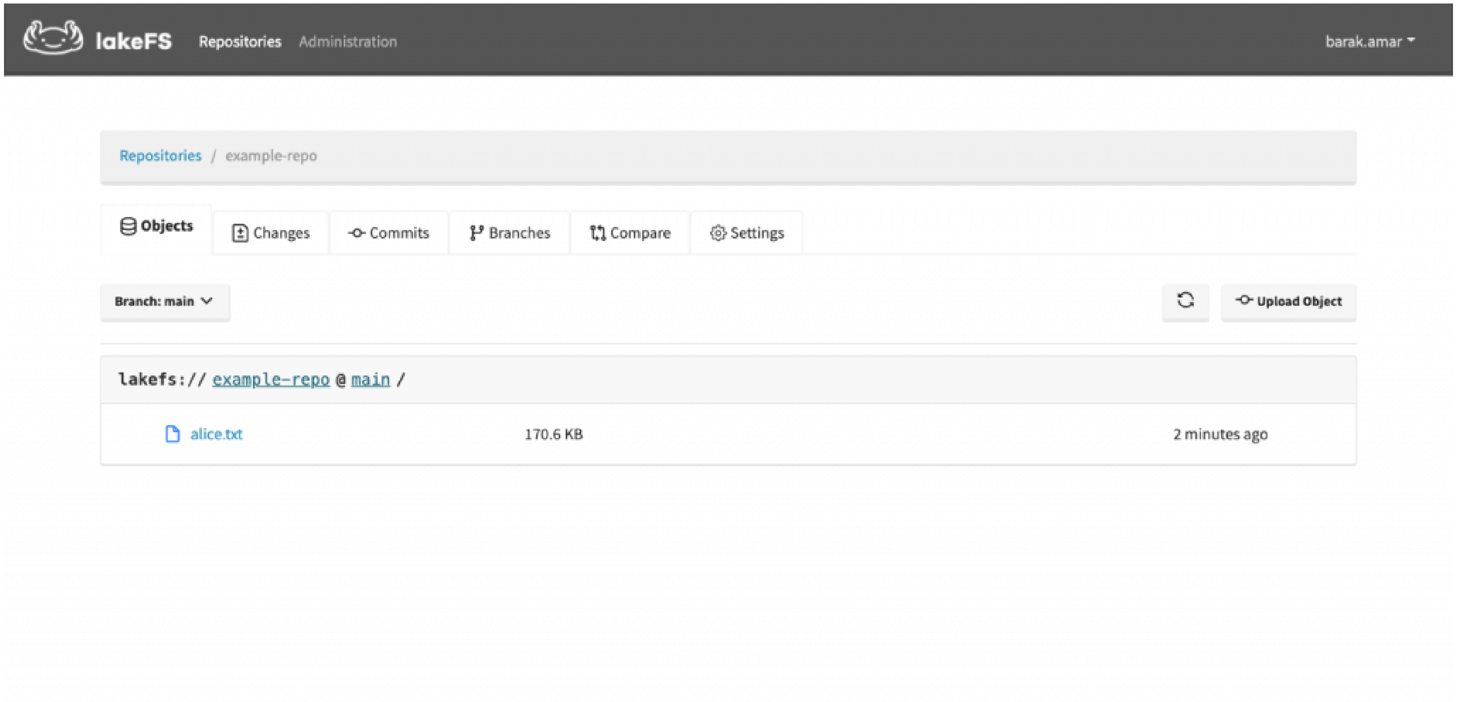
Use the command line interface to import CSV files, commit your changes, push them to a remote, or merge your teammate's changes.

All the commands you know for Git work exactly the same for Dolt. Git versions files, Dolt versions tables.

There’s also **DoltHub** – a place to share Dolt databases.

Table of contents

6. lakeFS



```
lakefs:// example-repo @ main /
```

alice.txt	170.6 KB	2 minutes ago

lakeFS is an open-source platform that provides a Git-like branching and committing model that scales to Petabytes of data by utilizing S3 or GCS for storage.

This branching model makes your data lake ACID-compliant by allowing changes to happen in isolated branches that can be created, merged, and rolled back atomically and instantly.

lakeFS has three main areas that let you focus on differen aspect of your ML models:

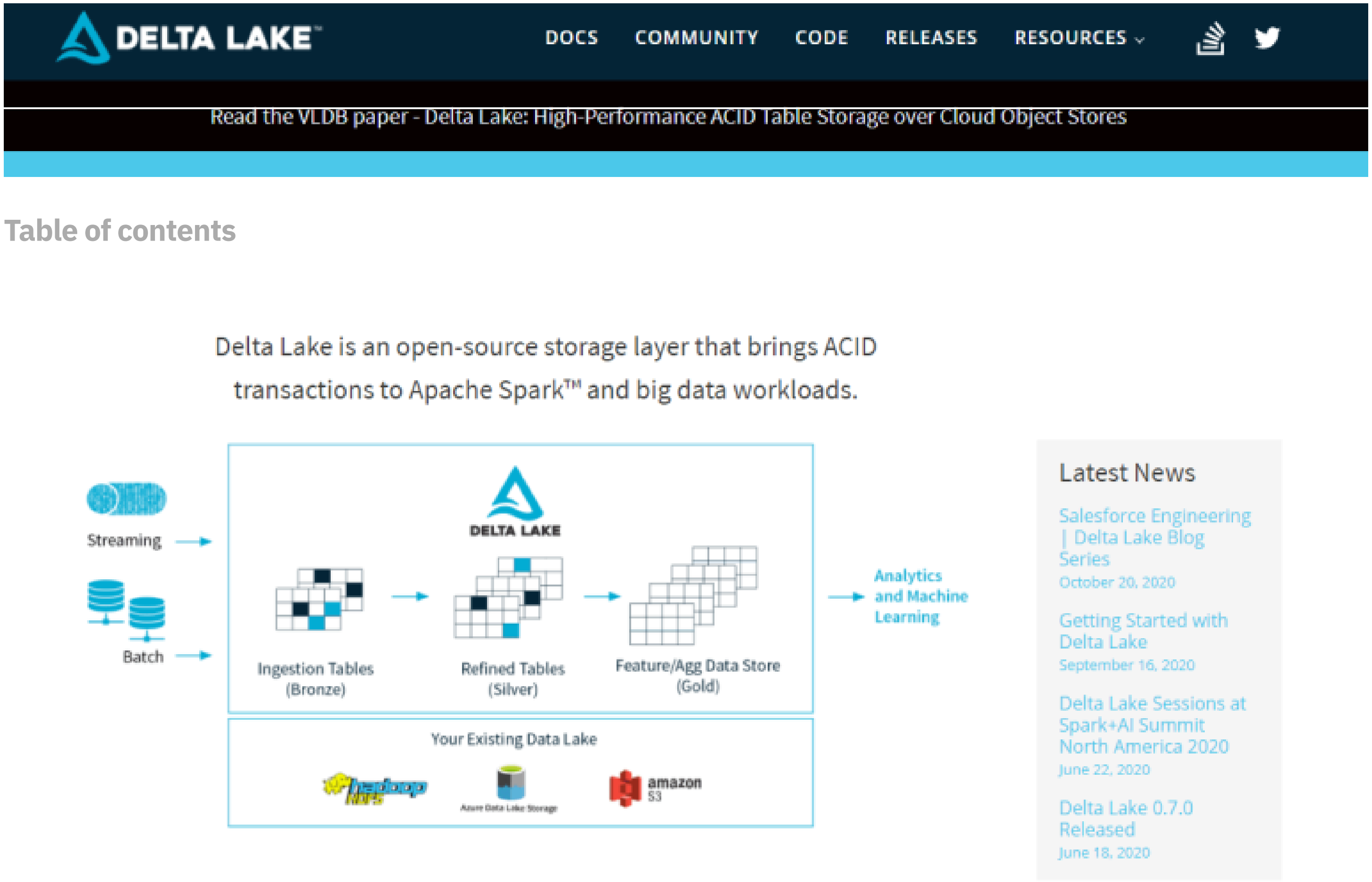
1. **Development Environment for Data:** has tools that you can use to isolate snapshot of the lake you can experiment with while others are not exposed; reproducibility to compare changes and

Table of contents

3. **Continuous Data Deployment:** ability to quickly revert changes to data; providing consistency in your datasets; testing of production data to avoid cascading quality issues

lakeFS is a great tool for focusing on a specific area of your datasets to make ML experiments more consistent.

7. Delta Lake



Read the VLDB paper - Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores

Table of contents

Delta Lake is an open-source storage layer that brings ACID transactions to Apache Spark™ and big data workloads.

Latest News

Salesforce Engineering | Delta Lake Blog Series
October 20, 2020

Getting Started with Delta Lake
September 16, 2020

Delta Lake Sessions at Spark+AI Summit North America 2020
June 22, 2020

Delta Lake 0.7.0 Released
June 18, 2020

Delta Lake is an open-source storage layer that brings **reliability to data lakes**. Delta Lake provides ACID transactions, scalable metadata handling, and unifies streaming and batch data processing. It runs on top of your existing data lake and is fully compatible with Apache Spark APIs.

Delta Lake – summary:

- **Scalable metadata handling**: Leverages Spark's distributed processing power to handle all the metadata for petabyte-scale tables with billions of files at ease.

- **Streaming and batch unification**: A table in Delta Lake is a batch table as well as a streaming source and sink. Streaming data ingest, batch historic backfill, interactive queries all just work out of the box.
- **Schema enforcement**: Automatically handles schema variations to prevent insertion of bad

Table of contents

- **Data versioning enables rollbacks, full historical audit trails, and reproducible machine learning experiments**
- **Supports merge, update,** and delete operations to enable complex use cases like change-data-capture, slowly-changing-dimension (SCD) operations, streaming upserts, and so on.

To wrap it up

Now that you have the list of the best tools for data versioning, you “just” need to figure out how to make it work for you and your team.

That can be tricky.

Some things to consider when choosing a data versioning are:

- **How easy is it to set up**: You may not have the time, needs, or budget to test something heavy right now.
- **Can you get your team onboard**: Sometimes, the solution is great, but you need more software engineering-oriented mindset to use it. Some ML researchers or data scientists may not end up using it.
- **What tool stack are you using today**: Are you using specific tools, infrastructure, or platform that has good integration with a particular data versioning solution. In that case, probably the best

option is to just go with that.

- **Data modality**: Is it images, tables, text, all? Sometimes the tool doesn't support your modality very well as it was built with a different use case in mind.

Table of contents

Reach out to me, and let's see what I can do!

Jakub Czakon

Mostly an ML person. Building MLOps tools, writing technical stuff, experimenting with ideas at Neptune.

Follow me on

Read next

Best AI & ML Tools When You Work With Projects for

Table of contents

Telecom companies have a lot of business and functional divisions that make them tick.

Data scientists that work in telecom can have various tasks to take care of depending on the division. There could be a data science team that improves customer experience, or one that powers the product & engineering division.

In this article, we'll look at popular use cases of data science and related tools in a telecom company, from my perspective as an ex-telecom data scientist.

My typical day as a Data Scientist in telecom
As a data scientist in the customer experience team, there are three main types of tasks that my role involved.

Business As Usual (BAU)
BAU is where you track certain KPIs, or tune/refresh an existing machine learning model.

Honestly, it's the least interesting work for a passionate data scientist, since it involves very little innovation and plenty of redundant tasks.

But it's a very important part of telecom companies, because it helps leadership and executives

Continue reading

Table of contents

Building Visual Search Engines with Kuba Cieślik

by Stephen Oladele, 17 min read

Read more

Deploying ML Models on GPU With Kyle Morris

by Stephen Oladele, 25 min read

Read more

Table of contents

ML Collaboration: Best Practices From 4 ML Teams

by Vidhi Chugh, 7 min read

Read more

Classification in ML: Lessons Learned From Building and Deploying a Large-Scale Model

by Shibsankar Das, 7 min read

Read more

Newsletter

Top MLOps articles, case studies, events (and more) in your inbox every month.

Your e-mail

Get Newsletter

PRODUCT

DOCUMENTATION

Table of contents

COMMUNITY

COMPANY

The Best MLOps Tools MLOps at a Reasonable Scale ML Metadata Store MLOps: What, Why, and How
Experiment Tracking in Machine Learning

Terms of service Privacy policy