IDATHA MEMBER OF pyxis ECOSYSTEM

2022

</>

MLOps

Versioning of models with DVC

Author

EMILIANO VIOTTI

Lead Machine Learning Engineer & Technical Leader @ IDATHA

eviotti@idatha.com

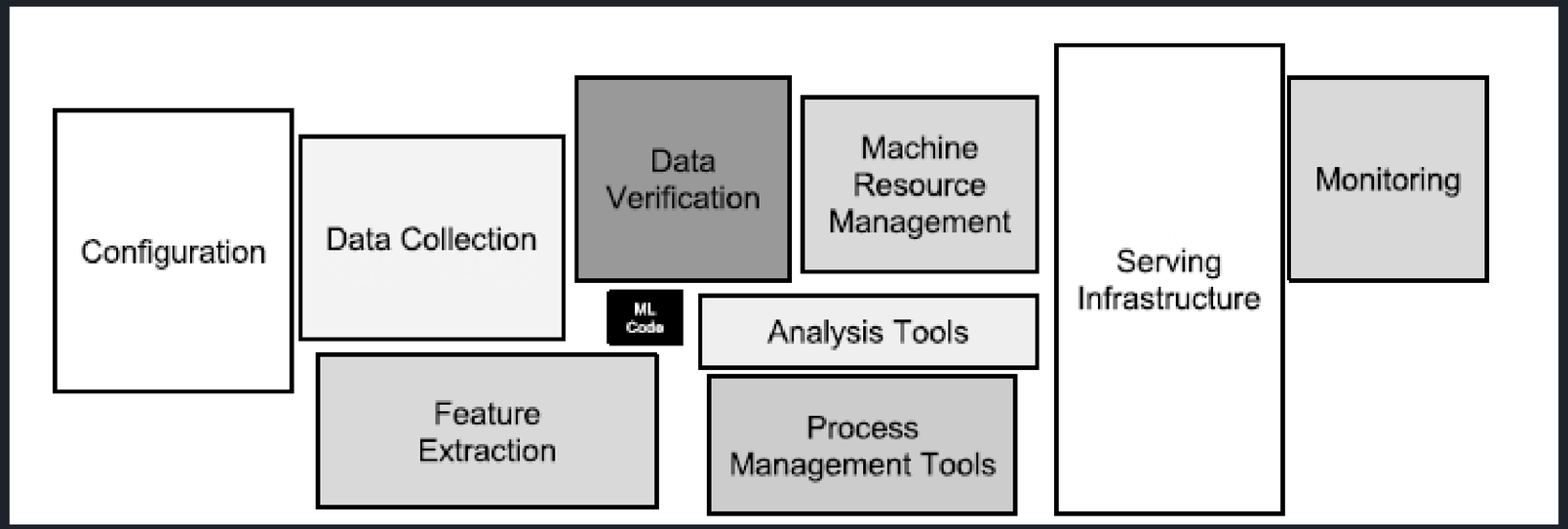
Agenda

- Quick recap
- Motivation
- What is DVC and what is not for
- How it works?
- Hands-On
- Extras

A quick recap

LET'S RECAP THE MOST IMPORTANT THING FROM THE LAST SESSION

ML is way more than just code



Sculley, D. et al. (2015) - Hidden technical debt in machine learning systems

Model life-cycle

1

Scoping

Define project,

business goals and metrics to reach success.

2

Data

Define data and establish baseline, label and organize data:

- different formats
- different sources

3

Modeling

Select and train model, perform error analysis:

- feature selection/generation
- algorithm selection
- hyper-parameter tuning
- train

4

Deployment

Deploy in production, monitor & maintain system

REPEAT

- Rate the impact of model in the business

- Version the source data and attributes

- Learn from mistakes
- Track model metrics
- Source control the code
- Checkpoints in the pipeline

- Automate validation/staged/deployment
- Deploy strategy: canary, blue/green, etc
- Monitor model:
  - Fairness
  - Model Drift

</>

Motivation

WHY DO WE NEED A VERSION CONTROL SYSTEM FOR MODELS?

Needs

DOING MACHINE LEARNING EVERY DAY

- Track datasets: Version Control
- Track models: Idem
- Track experiments: hyper-parameters, results, etc.
- Collaboration is often a requirement
- We are good at using Git to solve this problem
  - Files to big are not properly handled/supported by Git (>100MB needs Git LFS)
  - Binary files is a problem

</>

DVC

WHAT IS DVC AND WHAT IS NOT FOR

What is DVC?

An Open-source Version Control System for Machine Learning Projects, built to make ML models shareable and reproducible. It is designed to handle large files, data sets, machine learning models, and metrics as well as code.

Features

- Principal OS supported: macOs, Windows, Linux
- Git-compatible: DVC runs on top of any Git repository and is compatible with any standard Git server or provider (GitHub, GitLab, etc).
- Storage agnostic: Use Amazon S3, Microsoft Azure Blob Storage, Google Drive, Google Cloud Storage, Aliyun OSS, SSH/SFTP, HDFS, HTTP, network-attached storage, or disc to store data. The list of supported remote storage is constantly expanding.
- ML pipeline framework: DVC has a built-in way to connect ML steps into a DAG and run the full pipeline end-to-end.

See complete list of features in https://dvc.org/features

What is for?

- Versioning data and models: Handle large files, data sets, machine learning models
- CI/CD for Machine Learning: DVC takes care of handling data and models while other tools like CML take care of the orchestration.
- Data Caching Hub: built-in data caching lets you implement a simple and efficient storage layer globally.
- Experiment Tracking: Experiment management features out-of-the-box, captures relevant changesets automatically (input data, source code, hyperparameters, artifacts, etc.).
- Model and Data Registry: Import model and datasets and keep in the loop for new versions.

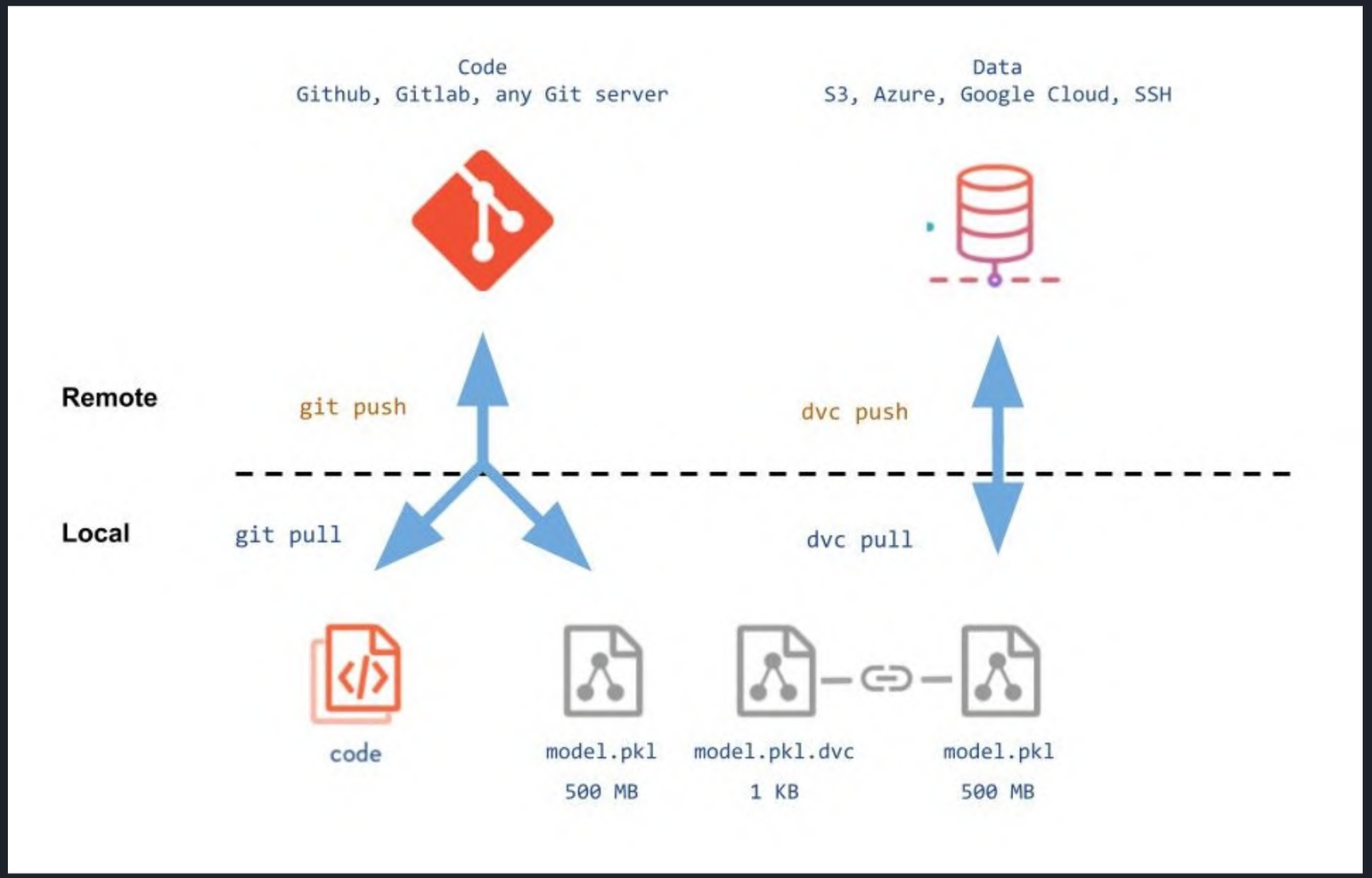
What is not for?

- Data Catalog: Metadata management tool to use to inventory and organize the data.
- Data Governance: everything you do to ensure data is secure, private, accurate, available, and usable.

DVC

HOW IT WORKS

Git vs. Git+ DVC



Hands-On

TIME TO WATCH DVC IN REAL ACTION

Installation

1. Install DVC and only the extra package you need for your remote storage (i.e. s3 or azure). Keep in mind you also need git.

```
pip install dvc [s3|gdrive|azure|…|all]
```

2. Add remote storage to your DVC configuration as default remote.

```
dvc remote add -d storage [MY_STORAGE_ADDRESS_URL]
```

Read more installation: https://dvc.org/doc/install

Integrate DVC (first time)

1. Init DVC configuration on a existing project:

```
dvc init
```

2. Add assets we want to start tracking using DVC instead of Git:

```
git rm -r --cached 'data'
git commit -m "stop tracking data"
git add data
git add data.dvc data/ .gitignore
git commit -m "Add raw data into DVC"
```

3. Push changes:

```
git push
dvc push
```

Read more: https://dvc.org/doc/start/data-management/data-versioning

Pull/Clone DVC project (eventually)

1. Clone existing project / Pull changes

```
git clone https://github.com/efviodo/dvc-hello-world.git
git pull
```

2. Pull DVC files

```
dvc pull
```

Making changes (every day)

When you make a change to a file or directory, run dvc add again to track the latest version.

1. Add new version:

```
dvc add data
```

2. Git-commit changes because data.dvc file changed. Then DVC-push latest version:

```
git commit data.dvc -m "Dataset updates"
dvc push
```

EXTRA

To switch between branches use git checkout + dvc checkout. DVC will read .dvc files to fetch the correct version of assets.

```
git checkout <…>
dvc checkout
```

Extras

A LIST OF ADDITIONAL RESOURCES TO CONTINUE EXPLORING

Extras

- DVC-Hello-World project

  https://github.com/efviodo/dvc-hello-world
- DVC Getting started guide

  https://dvc.org/doc/start
- DVC in minutes

  https://youtu.be/UbL7VUpv1Bs}
- DVC Hands-On

  https://youtu.be/kLKBcPonMYw
- Iterative.ai courses

  https://learn.iterative.ai/

CONTACT

- mlops.uruguay@idatha.com
- https://github.com/mlops-uruguay
- https://www.meetup.com/mlops-uruguay/

Request for talks

https://gobaldia.typeform.com/to/Sh28XWZc