DVC

장점

1. 대용량의 데이터를 git 과 명세파일을 통해 버전관리 하고 **캐싱**을 통해 빠르게 로드할 수 있다.
2. .dvc 파일만 git repo에 관리하며 실 데이터 저장 공간 ↔ .dvc 를 분리할 수 있다.
3. NAS 스토리지를 share repo로 생성하여 다수 user가 동일한 데이터셋을 사용할 수 있다.

단점

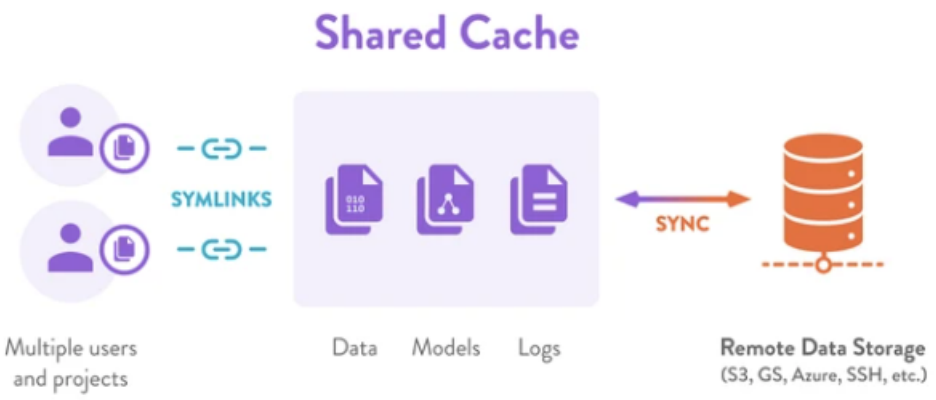
1. GIt CLI 사용에 대한 러닝커브 약간 존재
2. diff를 확인할 GUI 가 있어야 더 직관적으로 확인 가능
3. 커밋 메시지 작성에 대한 규칙 필요

DVC

data scientists still lack best practices for organizing their projects and collaborating effectively. This is a critical challenge: while ML algorithms and methods are no longer tribal knowledge, they are still difficult to develop, reuse, and manage.

- VS Code Extension, CLI, Python API 방식으로 DVC API 호출 가능
- 인프라나 플랫폼에 상관 없이 사용 가능
    - On-prem, Cloud 상관 없음
    - OS나 특정 framework에 종속적이지 않음
- **캐싱 기능이 제공되어 데이터 크기에 상관없이 빠르게 Switch 가능**

  → 해시 값을 통한 캐싱으로 중복캐싱 고려할 필요 없음

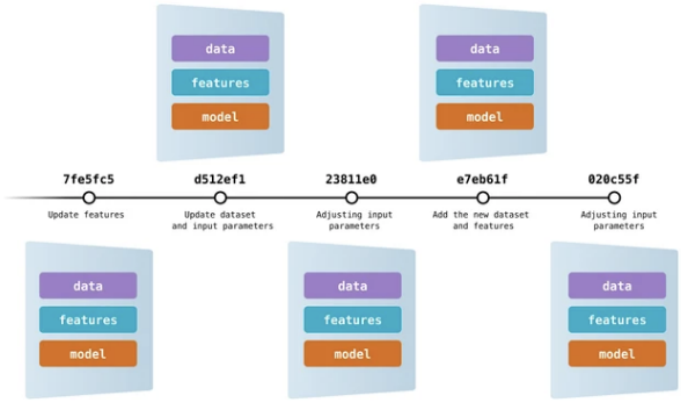


Data storage shared by DVC projects

호환 가능 Storage

- S3, Blob Storage, GCP Storage, SSH, SFTP, **HDFS, webHDFS . . .**
- HDFS 사용 시, libhdfs JNI 설치 필요

DVC File versioning 방식



> 💡 File Versioning을 위해서는 **GIT 필수**

- Git 컨셉과 동일한 방식으로 Data Versioning 사용 가능
  - Git COMMIT 해시 값이 해당 데이터 버전의 고유한 식별값으로 사용

- 대용량 파일에 대한 명세 (`.dvc` , `dvc.yaml`) 파일을 git에 트래킹 하여 파일 버전 관리.

Flow

1. 대용량의 파일 local로 Load.
2. `dvc add` 를 통해 버전관리할 파일 .dvc 파일 생성
   → 내부적으로 cache, .gitignore 에 원본파일 경로 추가

3. GIT 에 명세파일을 commit, tag 함으로 써 버전등록 가능

```
dvc add data model.h5

git add data.dvc model.h5.dvc metrics.csv
git commit -m "Second model, trained with 2000 images"
git tag -a "v2.0" -m "model v2.0, 2000 images"
```

Switch version

```


# Full workspace checkout
git checkout v1.0
dvc checkout

# Specific dataset checkout
git checkout v1.0 data.dvc
dvc checkout data.dvc
```

Automating Capturing

자동 파이프라인을 구성한 후, `dvc repro` 명령어를 사용하여 지정해둔 파이프라인 수행 가능

Makefile 생성하는 것으로 생각하면 됨

- 파이프라인 생성 CLI

```
dvc stage add -n train -d train.py -d data \
          -o model.h5 -o bottleneck_features_train.npy \
          -o bottleneck_features_validation.npy -M metrics.csv \
          python train.py
dvc repro
```

- 생성된 dvc.yaml

```
stages:
  train:
    cmd: python train.py
    deps:
    - data
    - train.py
    outs:
    - bottleneck_features_train.npy
    - bottleneck_features_validation.npy
    - model.weights.h5
    metrics:
    - metrics.csv:
        cache: false
```

> To make things a little simpler: `dvc add` and `dvc checkout` provide a basic mechanism for model and large dataset versioning. `dvc stage add` and `dvc repro` provide a build system for machine learning models, which is similar to Make in software build automation.

버전 별 diff 비교

- commit message 별 diff를 통해서 확인 가능

1. git diff / dvc diff 사용

```
dvc diff -R <commit hash1 > <commit hash 2>

git diff <commit1> <commit 2>
```

```
~/workspace/de/dvc-work/example-versioning (master*) » dvc diff 5dd6b38da286ff0c412ad70b5b861e5b2f622700
Added:
    bottleneck_features_train.npy
    bottleneck_features_validation.npy

Modified:
    model.weights.h5

files summary: 2 added, 1 modified
```

2. GUI 도구 사용