Get Ready for ML!

Level Up Your Data Lake

With Delta and lakeFS

Data Council – Austin

March 2022

Speakers

Adi Polak

Vice President of Developer Experience | Treeverse

Adi is an open-source technologist who believes in communities and is passionate about building a better world through open collaboration. As Vice President of Developer Experience at Treeverse, Adi helps build lakeFS, git-like interface for the data lakehouse. In her work, she brings her vast industry research and engineering experience to bear in educating and helping teams design, architect, and build cost-effective data systems and machine learning pipelines that emphasize scalability, expertise, and business goals.

Adi is a frequent worldwide presenter and the author of O'Reilly's upcoming book, "Machine Learning With Apache Spark." Adi is also a proud Beacon for Databricks! Previously, she was a senior manager for Azure at Microsoft, where she focused on building advanced analytics systems and modern architectures.

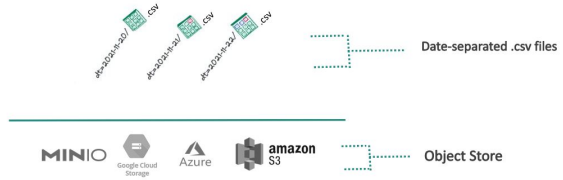
Paul Singman

Developer Advocate | Treeverse

Paul is a developer advocate for the lakeFS project, after several years on the analytics team at Equinox Fitness. His goal is to democratize big data analytics through explaining data architectures that are both user-friendly and cost-effective. He's spoken at various conferences and meetups, including the Postgres Conference NYC and AWS re:Invent. When not working you can find him drinking tea and playing golf

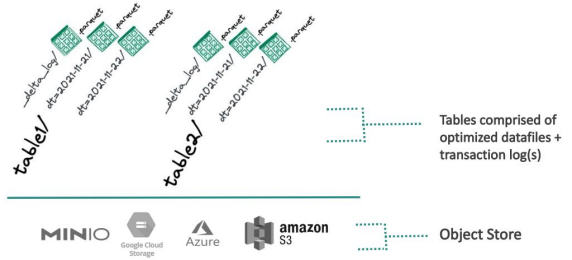
Narrative Flow

Level 0: Basic Data Lake

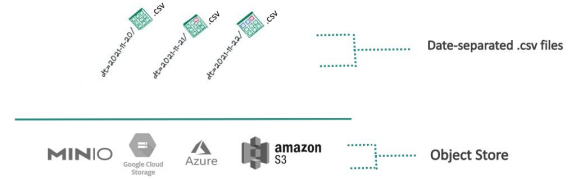


Narrative Flow

Level 1: Table-Format Enhanced

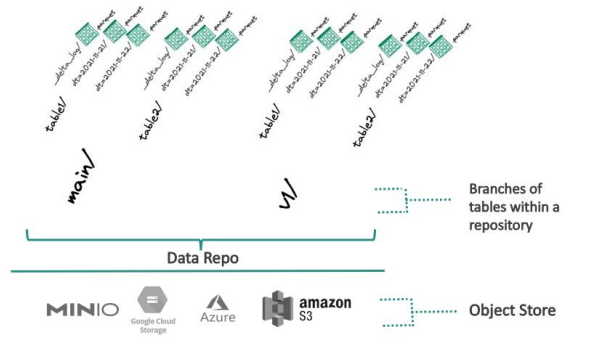


Level 0: Basic Data Lake

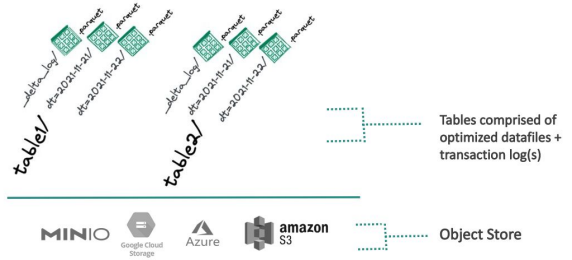


Narrative Flow

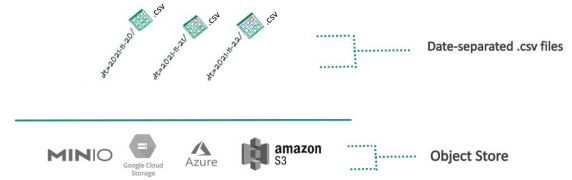
Level 2: Full Data Version Control



Level 1: Table-Format Enhanced



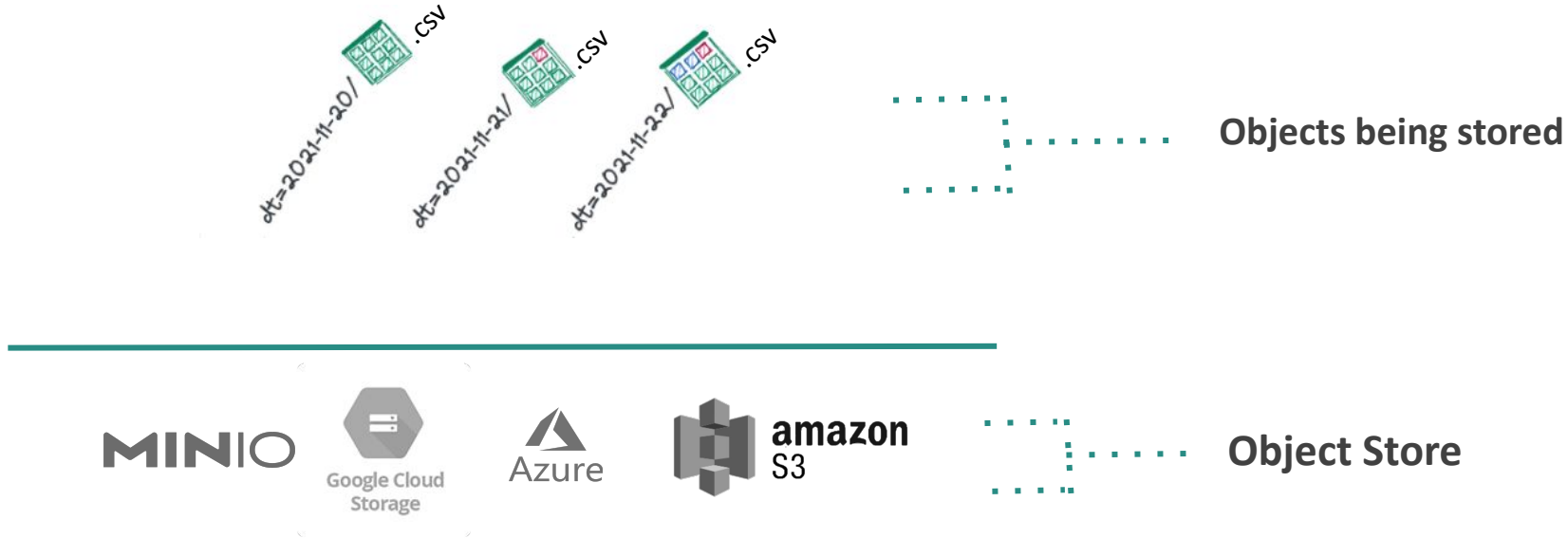
Level 0: Basic Data Lake



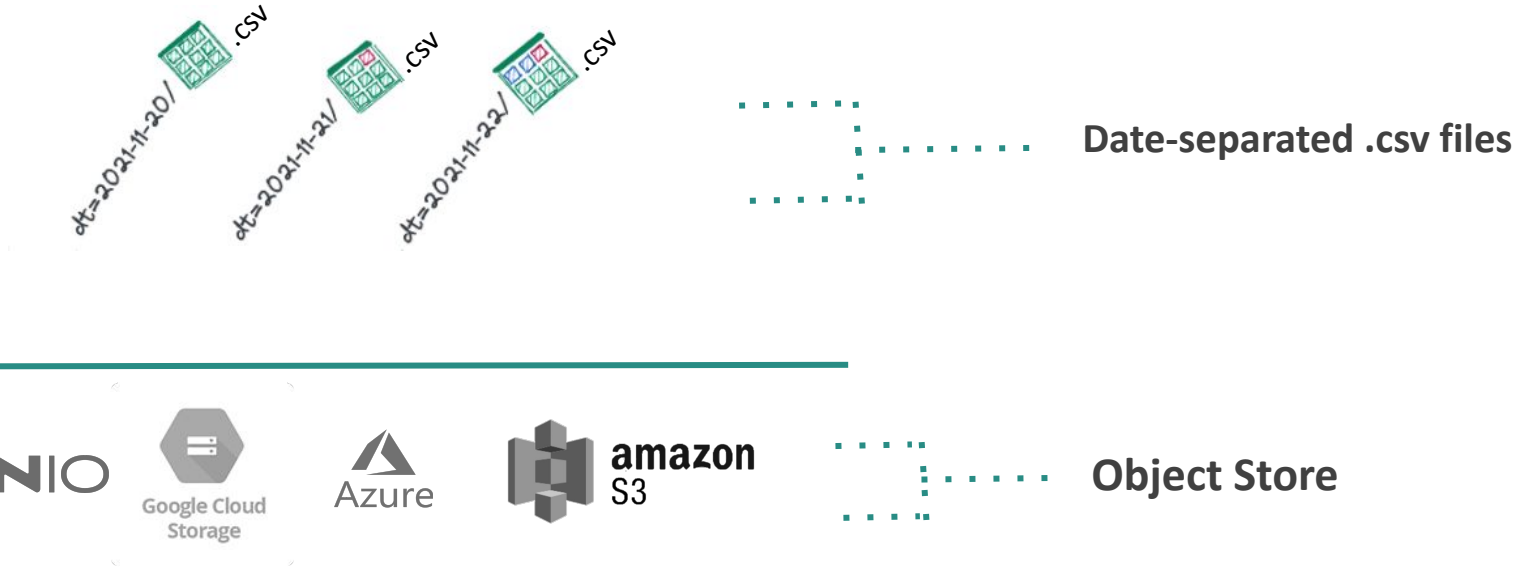
L0: Basic Data Lake

L0: Basic Data Lake

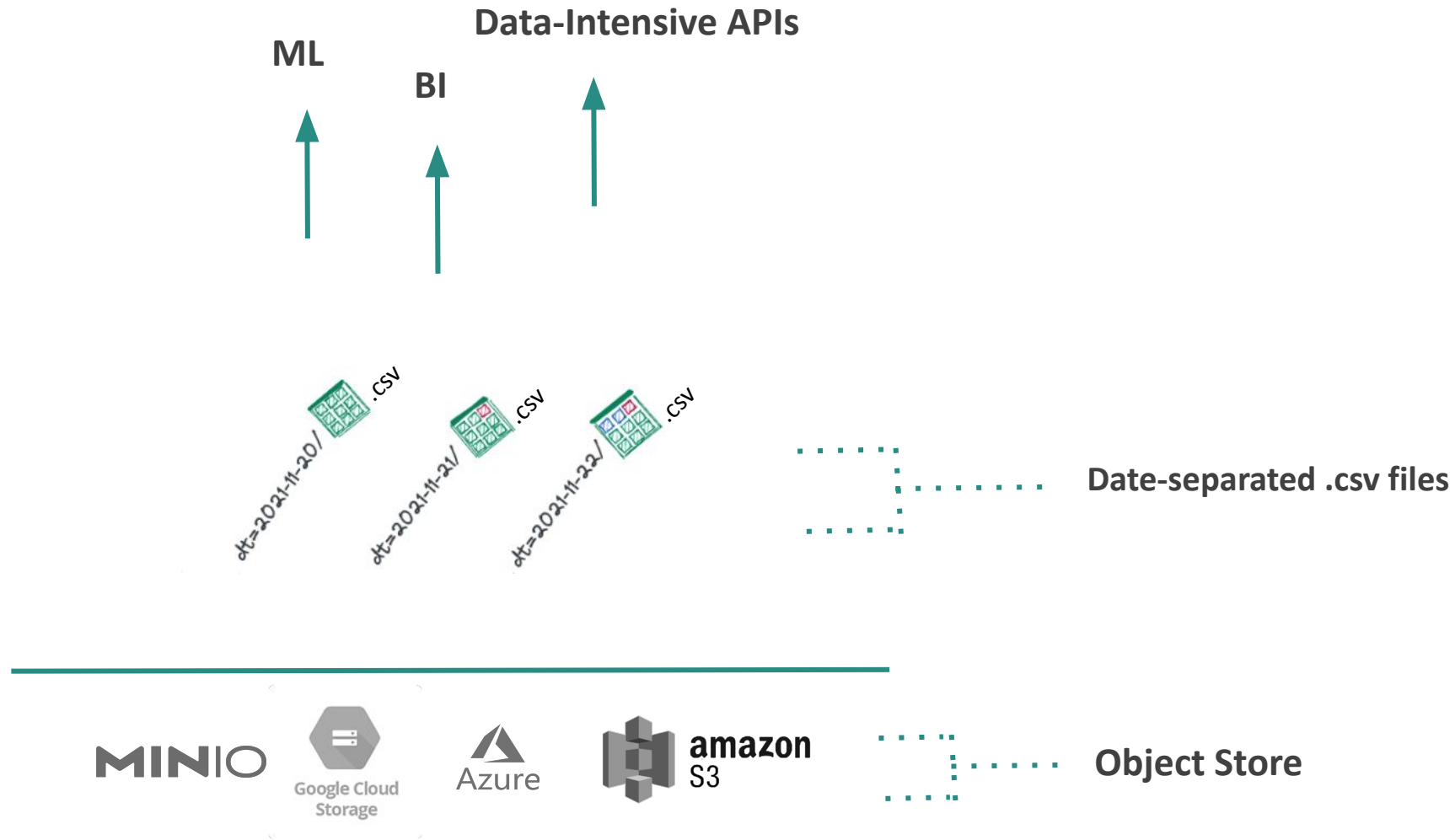
L0: Basic Data Lake



L0: Basic Data Lake



L0: Basic Data Lake



Why Object Storage?

MINIO

Google Cloud Storage

Azure

amazon S3

Why Object Storage?

MINIO Google Cloud Storage Azure amazon S3

are **awesome** in terms of

Why Object Storage?

are **awesome** in terms of

- Performance
- Cost
- Developer Experience
- Connectivity

Why Object Storage?

MINIO Google Cloud Storage Azure amazon S3 are **awesome** in terms of

- **Performance**
- Cost
- Developer Experience
- Connectivity

- Achieve **3.5k** PUT requests per second **per prefix**
- **5.5k** GET requests per second **per prefix**
- Auto-scales to this limit automatically and overall capacity is limitless
- "something like 11 '9's of availability"

Why Object Storage?

MINIO Google Cloud Storage Azure amazon S3 are **awesome** in terms of

- Performance
- **Cost**
- Developer Experience
- Connectivity

- **Storage**: $.023 per GB vs $.10 for RDS or $.12 for EBS
- **Network:**
  - $5 per milllion PUT, $.40 per million GET requests,
  - $0 transfer data in, $.09 per GB for data transfer out
- ~5-8x times cheaper than block storage

Why Object Storage?

are **awesome** in terms of

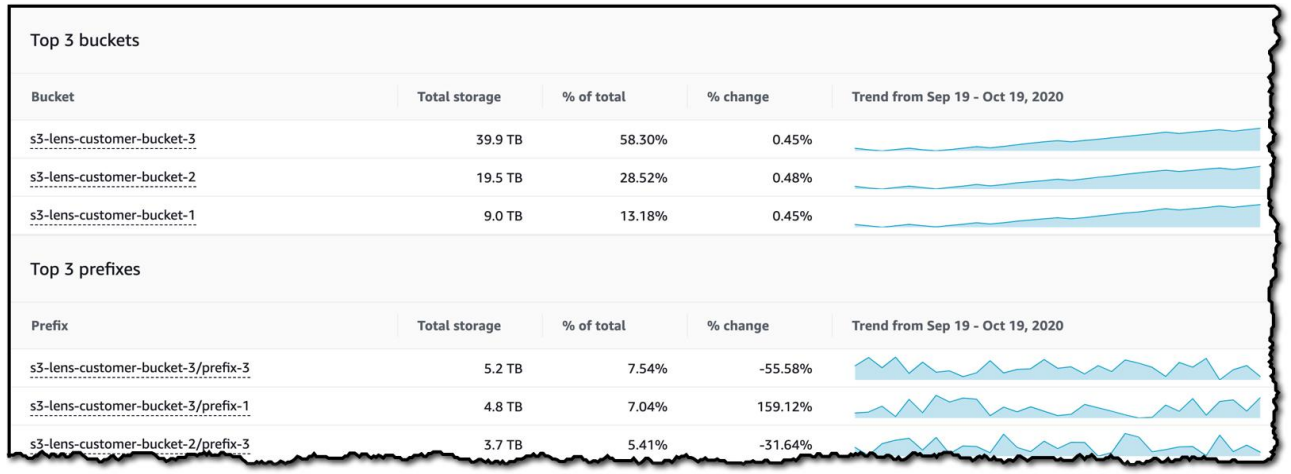
- Performance
- Cost
- **Developer Experience**
- Connectivity

- Mature client SDKs
- Strong Consistency (2020)
- AWS Storage Lens (2020)
- Feature-rich (events, permissions, inventories, replication...)

Why Object Storage?

MINIO Google Cloud Storage Azure amazon S3

are **awesome** in terms of



Top 3 buckets

Bucket	Total storage	% of total	% change	Trend from Sep 19 - Oct 19, 2020
s3-lens-customer-bucket-3	39.9 TB	58.30%	0.45%	
s3-lens-customer-bucket-2	19.5 TB	28.52%	0.48%	
s3-lens-customer-bucket-1	9.0 TB	13.18%	0.45%	

Top 3 prefixes

Prefix	Total storage	% of total	% change	Trend from Sep 19 - Oct 19, 2020
s3-lens-customer-bucket-3/prefix-3	5.2 TB	7.54%	-55.58%	
s3-lens-customer-bucket-3/prefix-1	4.8 TB	7.04%	159.12%	
s3-lens-customer-bucket-2/prefix-3	3.7 TB	5.41%	-31.64%	

- Mature client SDKs
- Strong Consistency (2020)
- **AWS Storage Lens (2020)**
- Feature-rich (events, permissions, inventories, replication...)

Why Object Storage?

MINIO Google Cloud Storage Azure amazon S3 are **awesome** in terms of

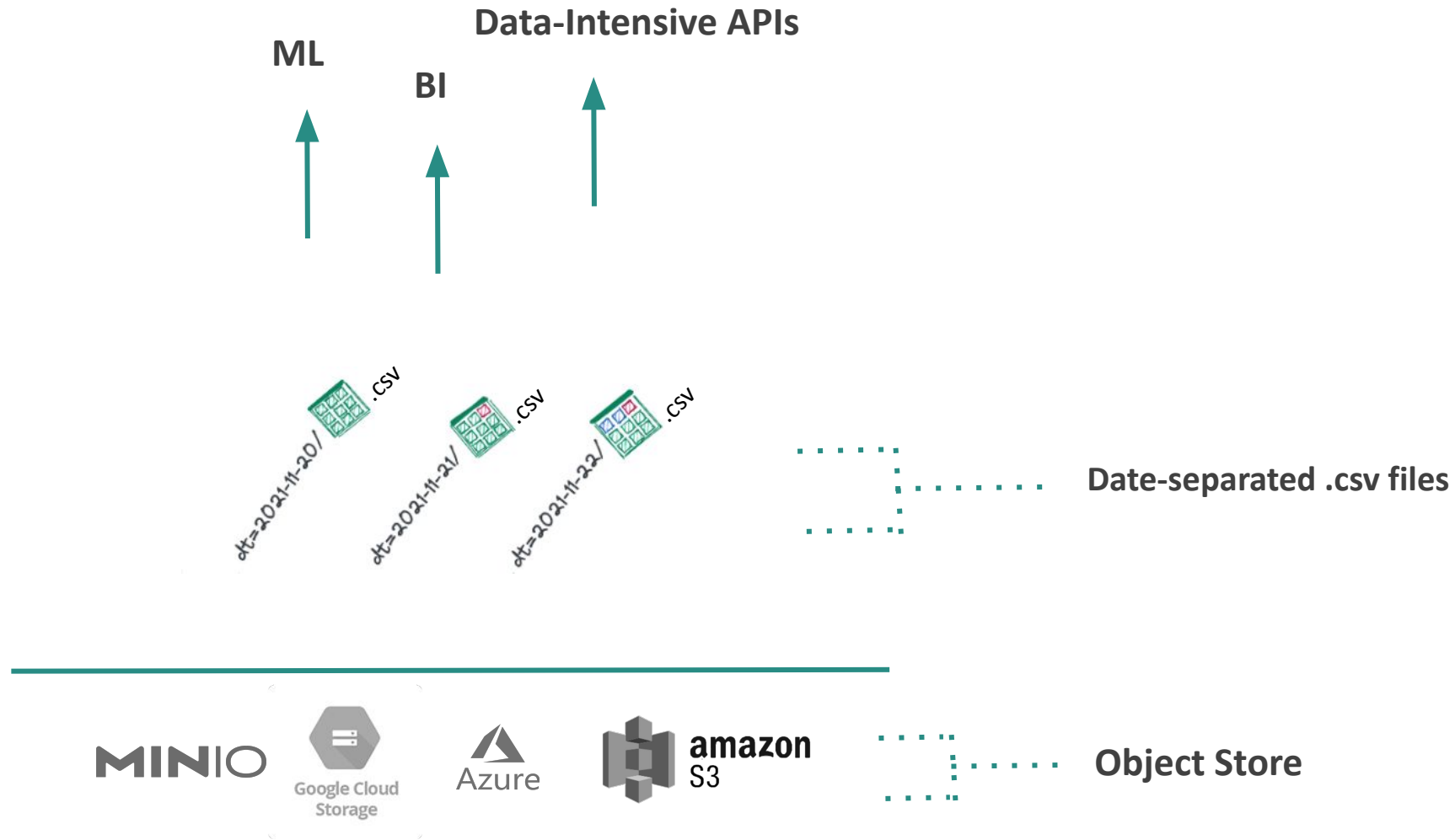
- Performance
- Cost
- Developer Experience
- **Connectivity**

Why Object Storage?

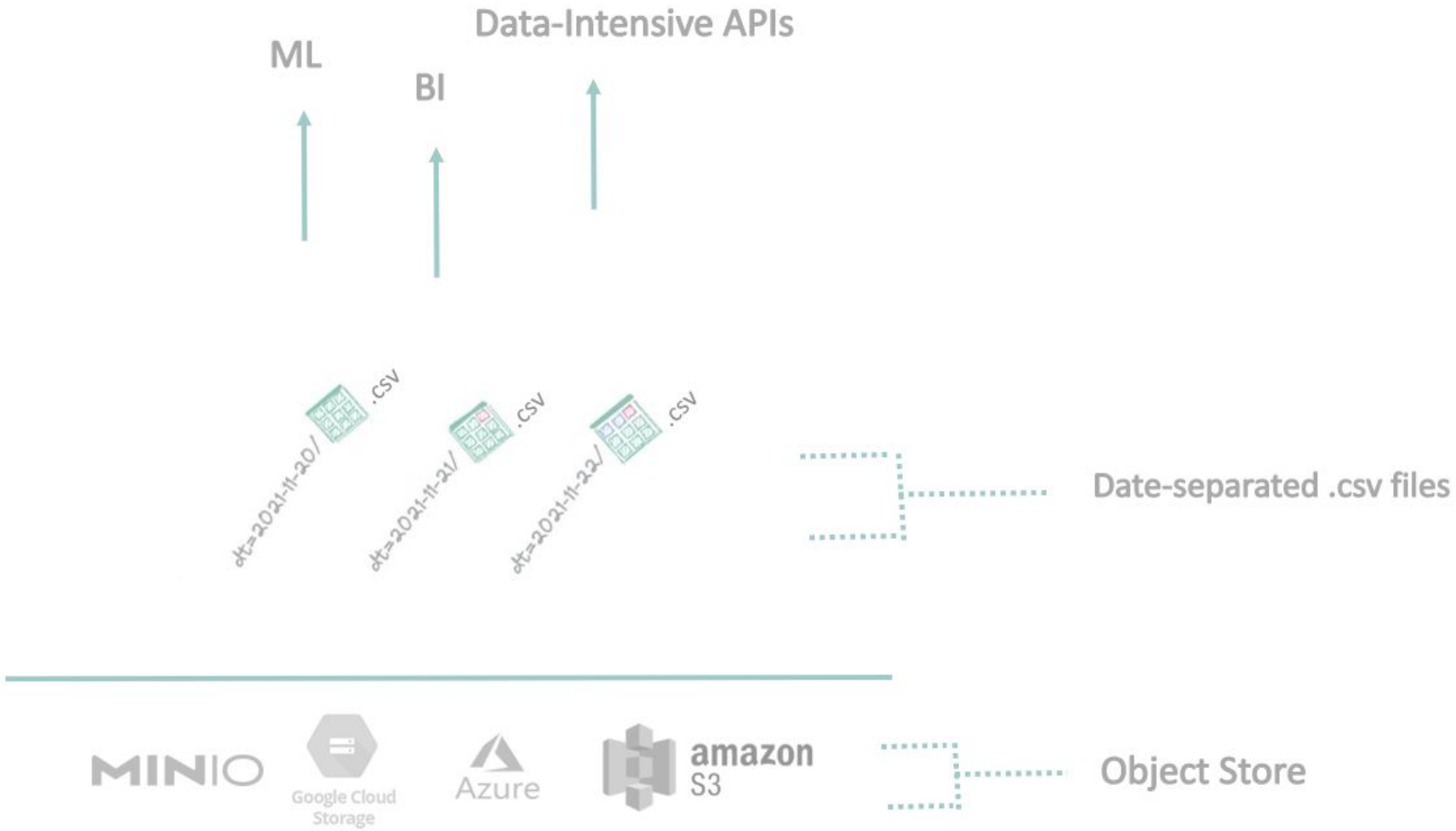
MINIO Google Cloud Storage Azure amazon S3 are **awesome** in terms of

- Performance
- Cost
- Developer Experience
- **Connectivity**

L0: Basic Data Lake



L0: Basic Data Lake



Now let's make object store-specific improvements

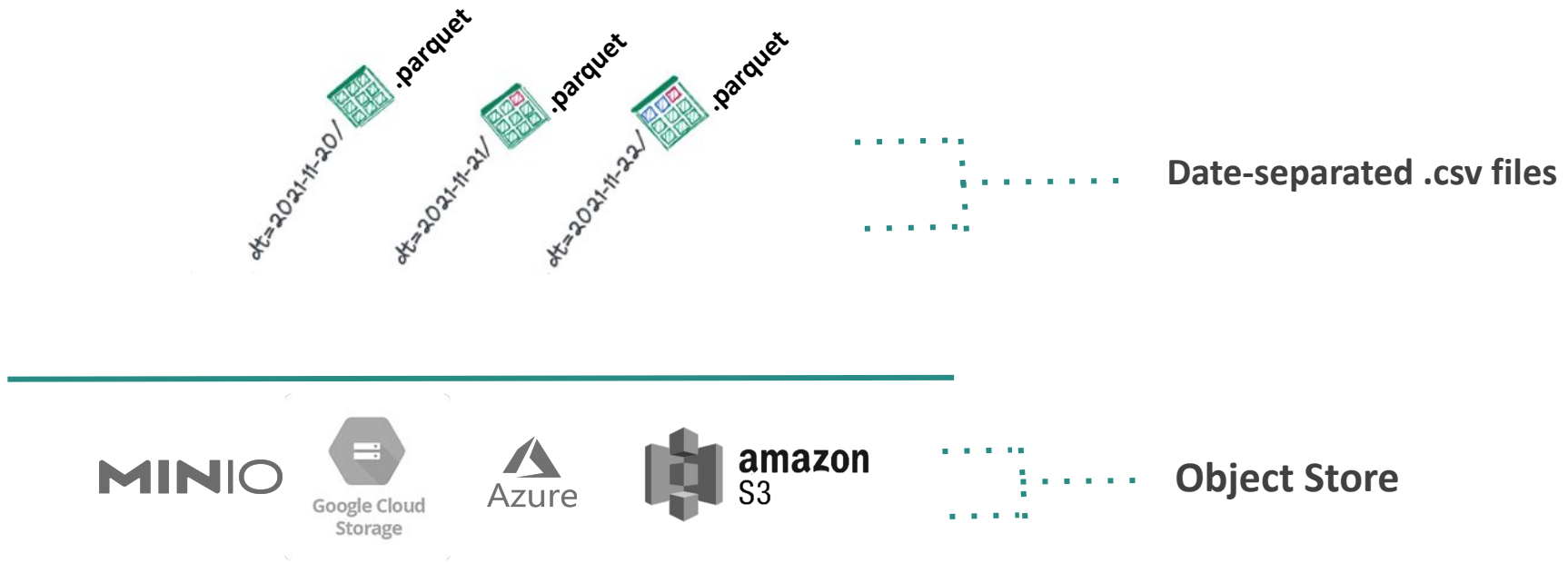
L0.5: Parquet File Format

.parquet

.parquet

.parquet

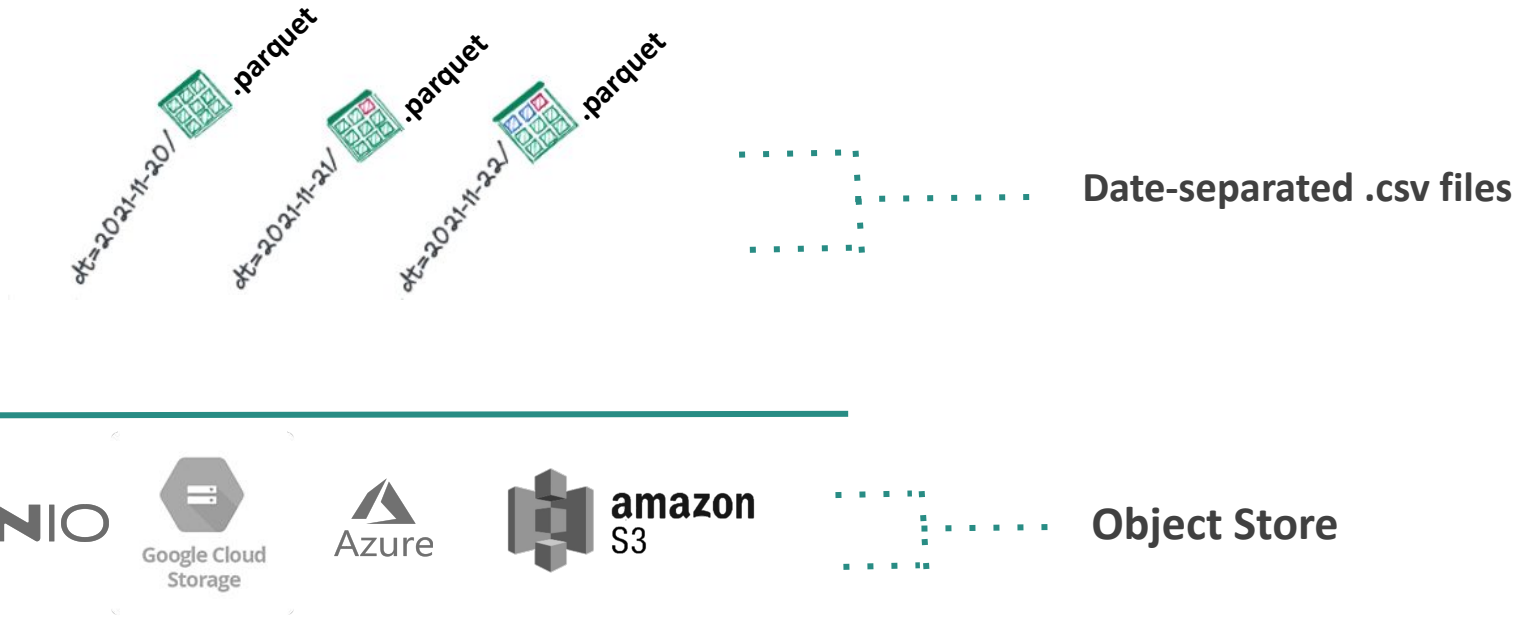
L0.5: Parquet File Format



L0.5: Parquet File Format

Benefits of parquet:

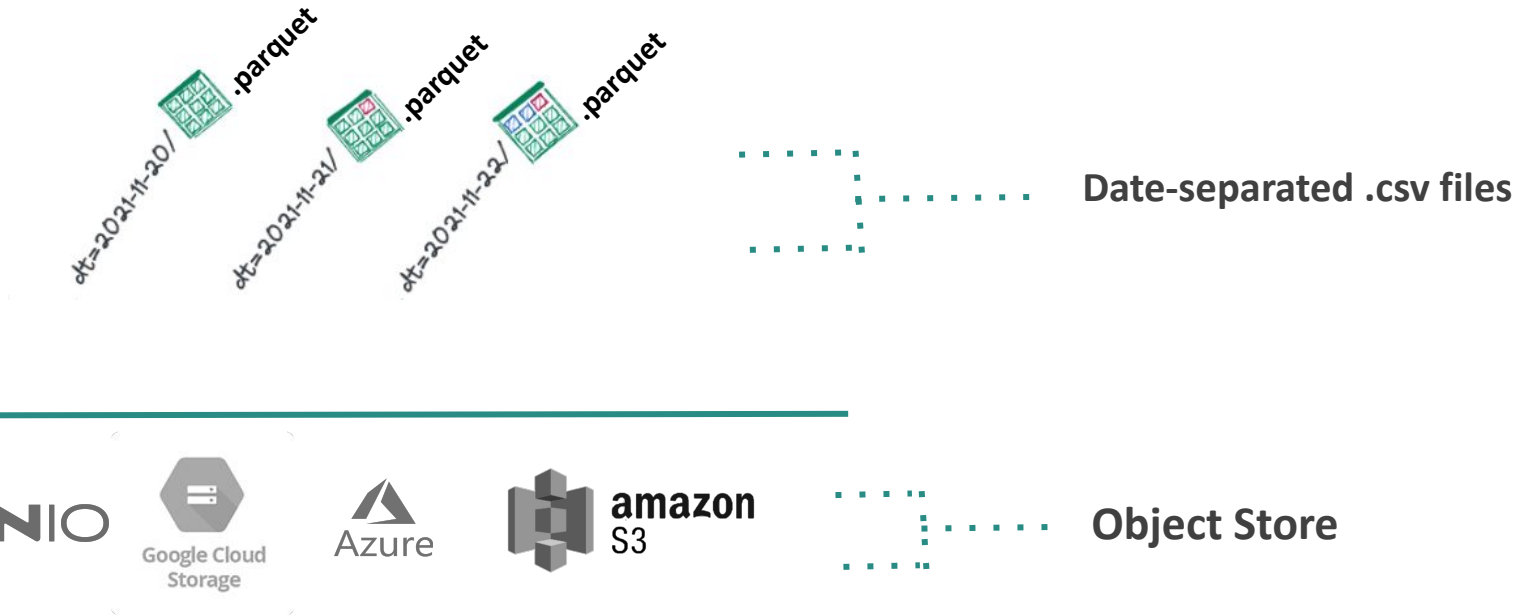
1. Columnar
2. Compressible
3. Complex



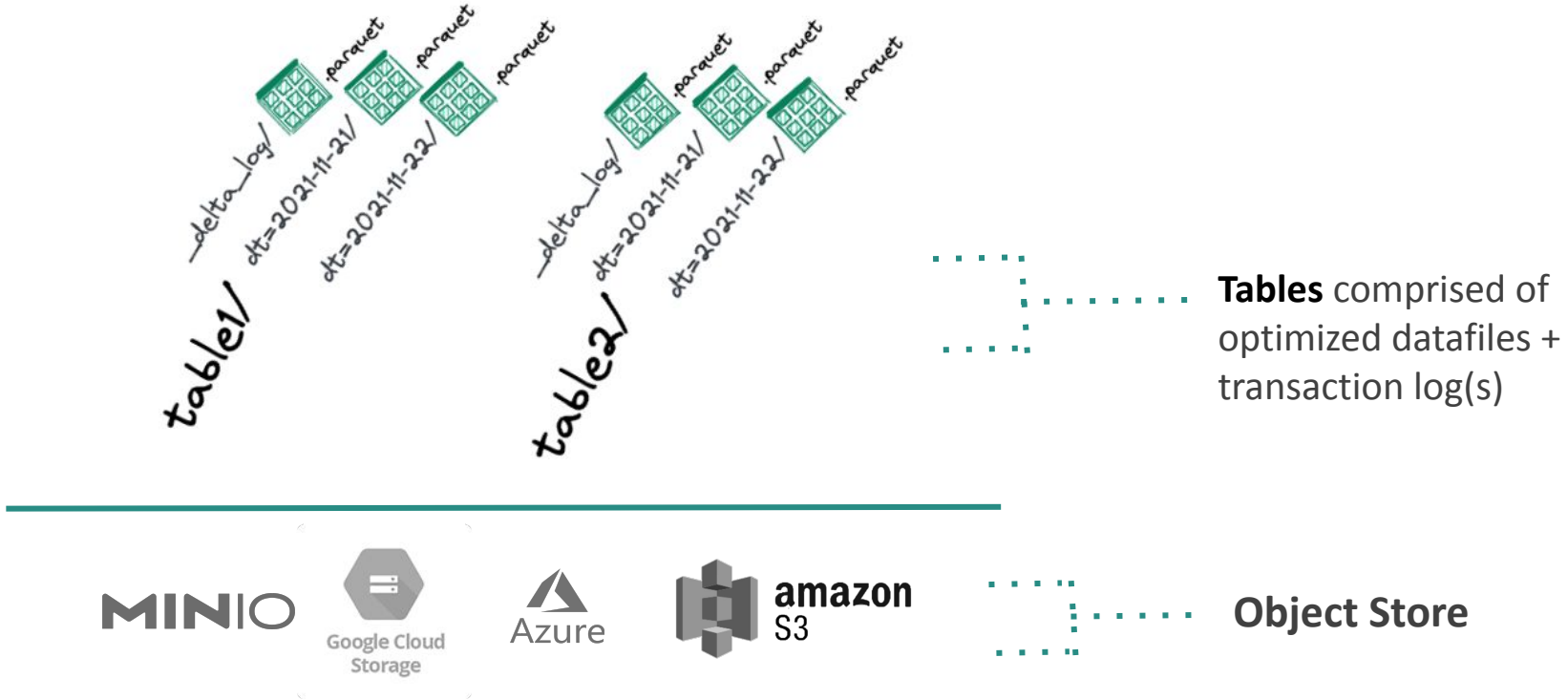
L0.5: Parquet File Format

Challenges with parquet:

1. Operates at the object level



L1: Modern Table Formats



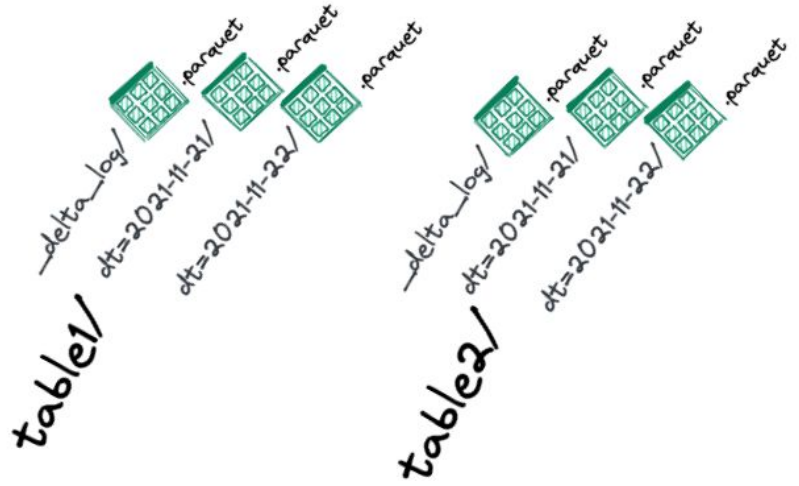
L1: Modern Table Formats

New Operations at the table level

- Define schema
- Traverse versions
- Upsert atomically

Implementations:

- Apache Hudi
- Apache Iceberg
- Delta Lake

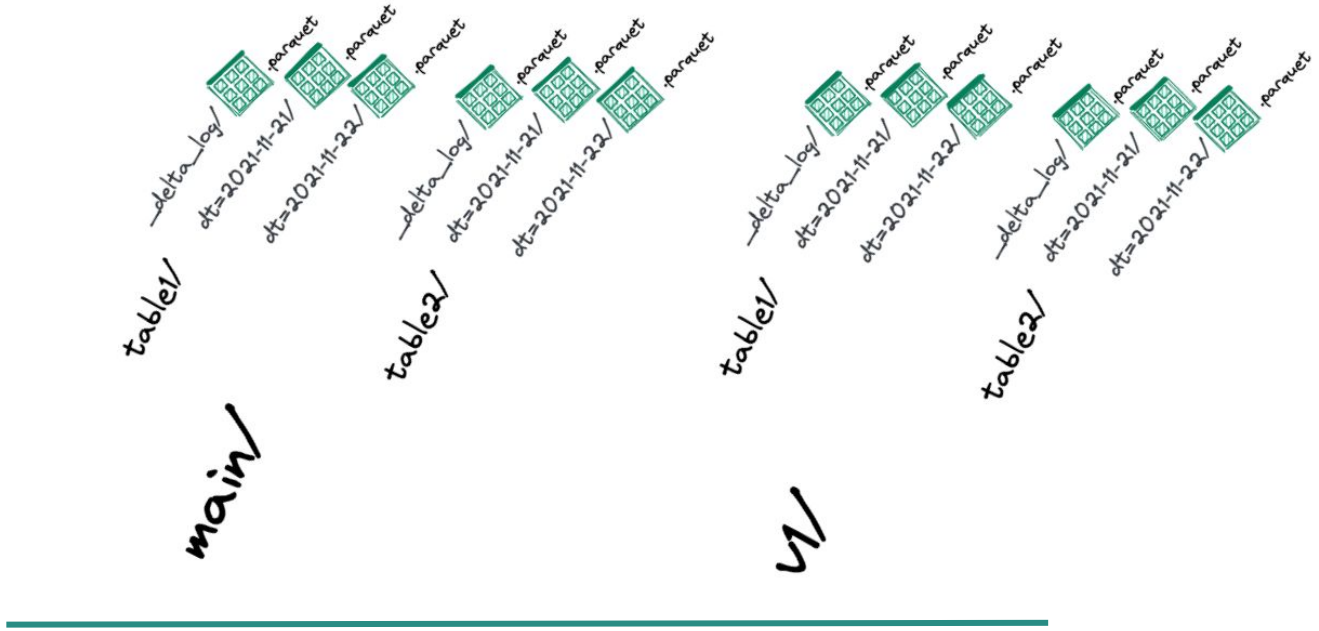


Tables comprised of optimized datafiles + transaction log(s)

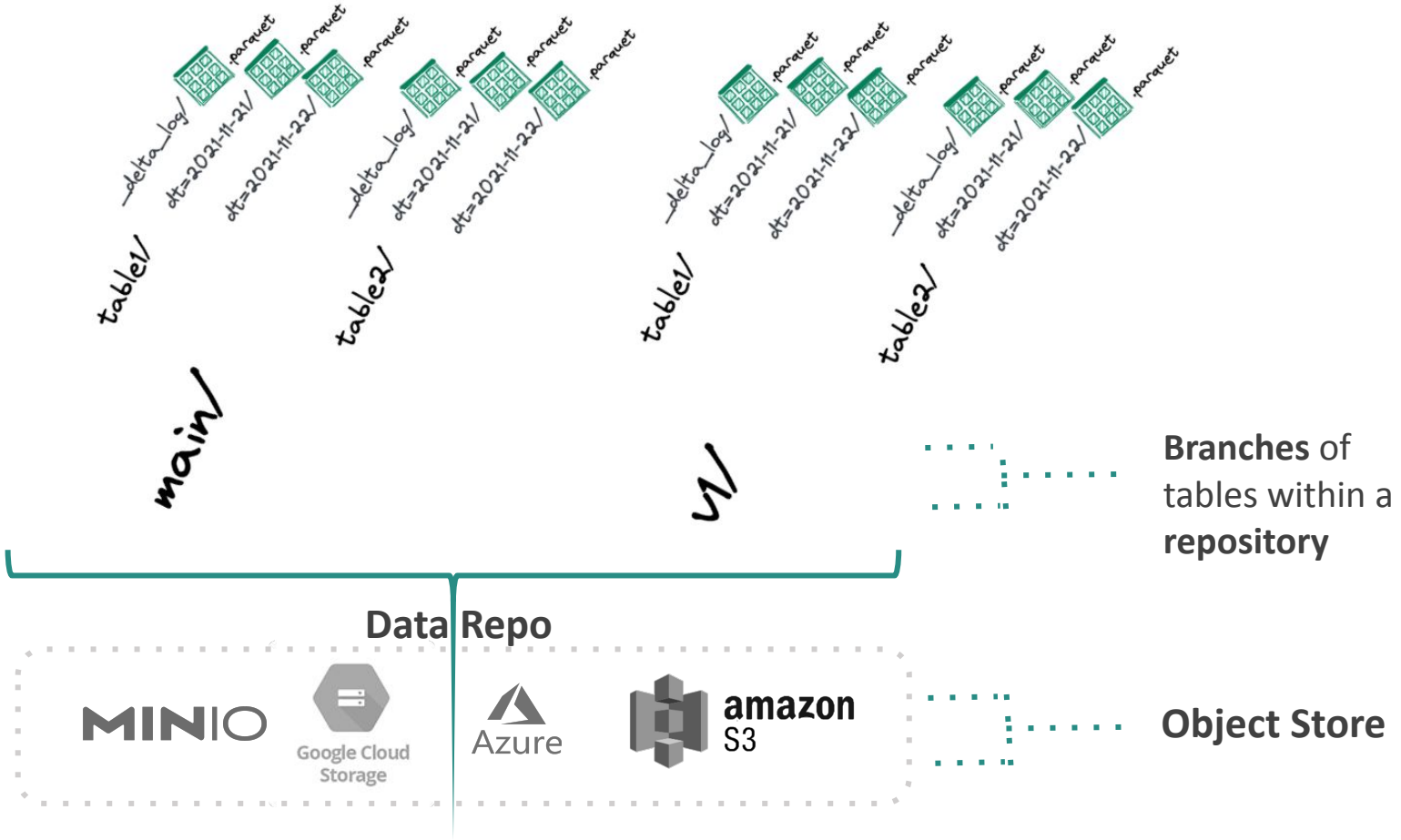
MINIO Google Cloud Storage Azure amazon S3

Object Store

L2: Data Version Control



L2: Data Version Control



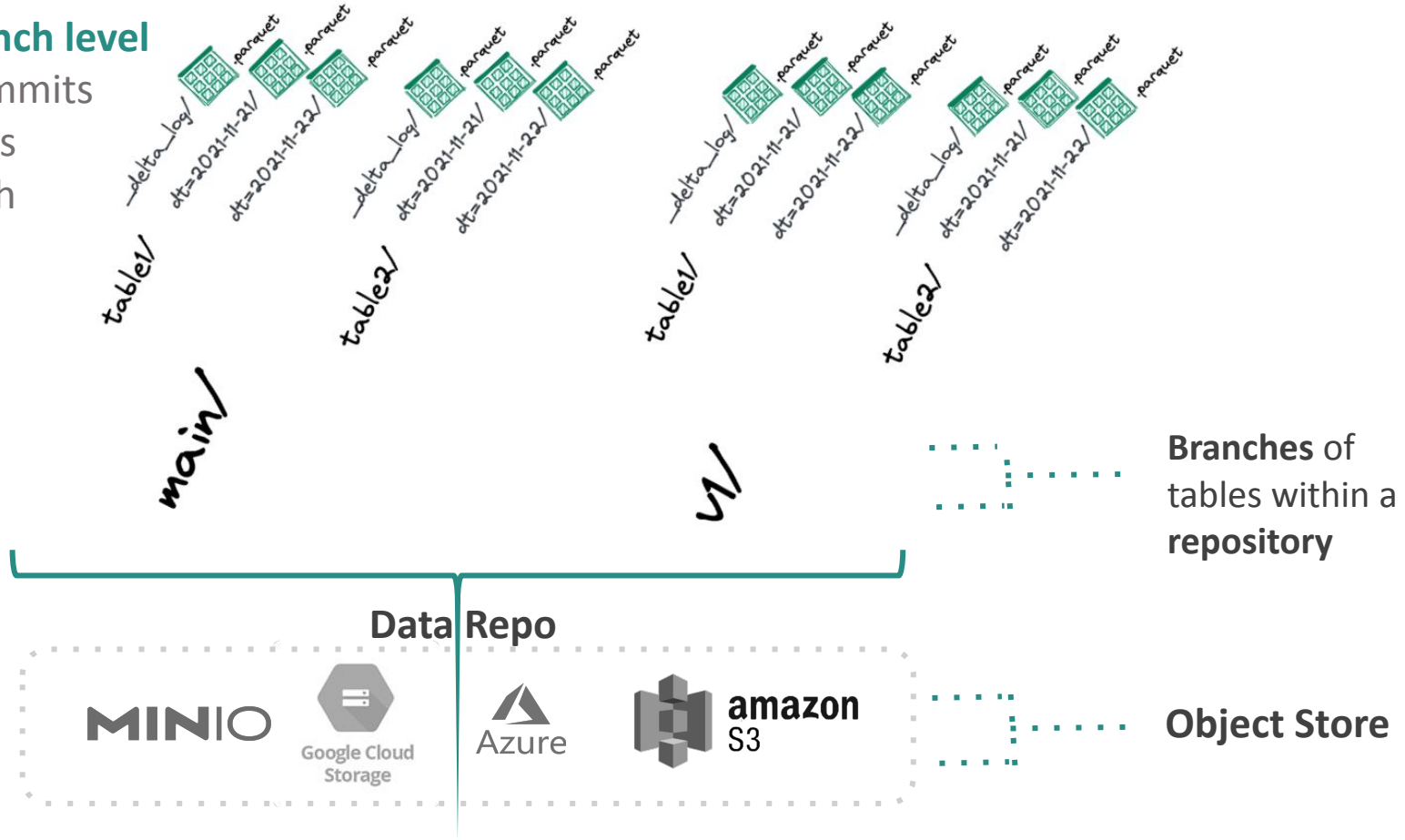
L2: Data Version Control

New Operations at the branch level

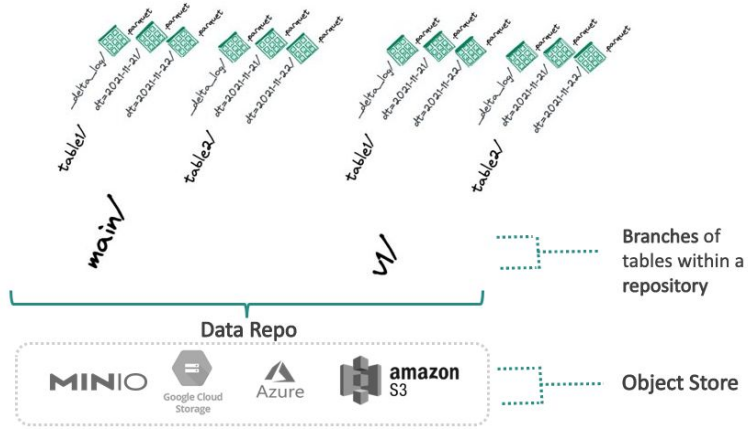
- Traverse among commits
- Merge two branches
- Create a new branch
- Take a commit

Implementations:

- **lakeFS**
- **Proj Nessie**



L2: Data Version Control Applications



New Operations at the branch level

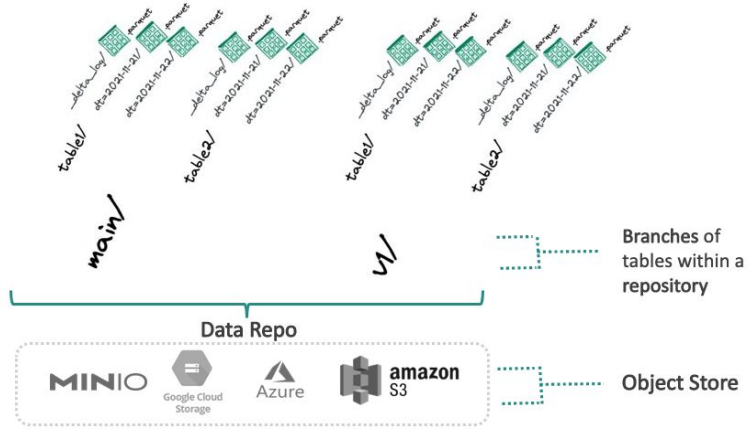
Traverse among commits

Merge two branches

Create a new branch

Take a commit

L2: Data Version Control Applications



New Operations at the branch level

Traverse among commits

Merge two branches

Create a new branch

Take a commit

lakeFS CLI Example

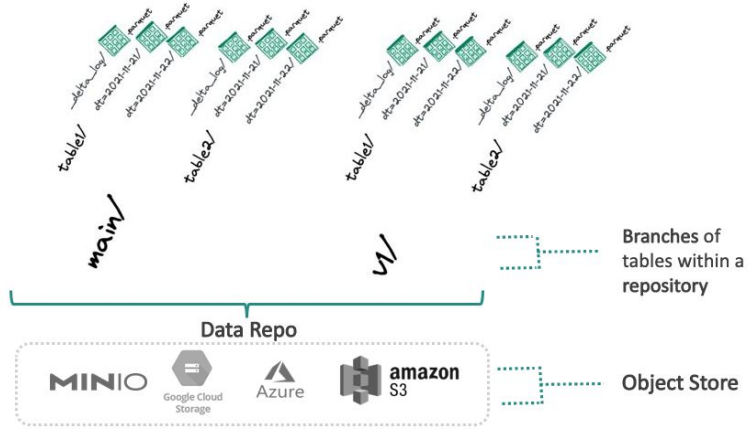
```
$ lakectl revert main^1
```

```
$ lakectl merge my-branch-main
```

```
$ lakectl branch create my-branch
```

```
$ lakectl commit –m "new commit"
my-branch
```

L2: Data Version Control Applications



New Operations at the branch level

Traverse among commits

Merge two branches

Create a new branch

Take a commit

lakeFS CLI Example

```
$ lakectl revert main^1
```

```
$ lakectl merge my-branch-main
```

```
$ lakectl branch create my-branch
```

```
$ lakectl commit –m "new commit"
my-branch
```

Useful for...

Instant recovery from issues

Atomic updates (cross-coll)

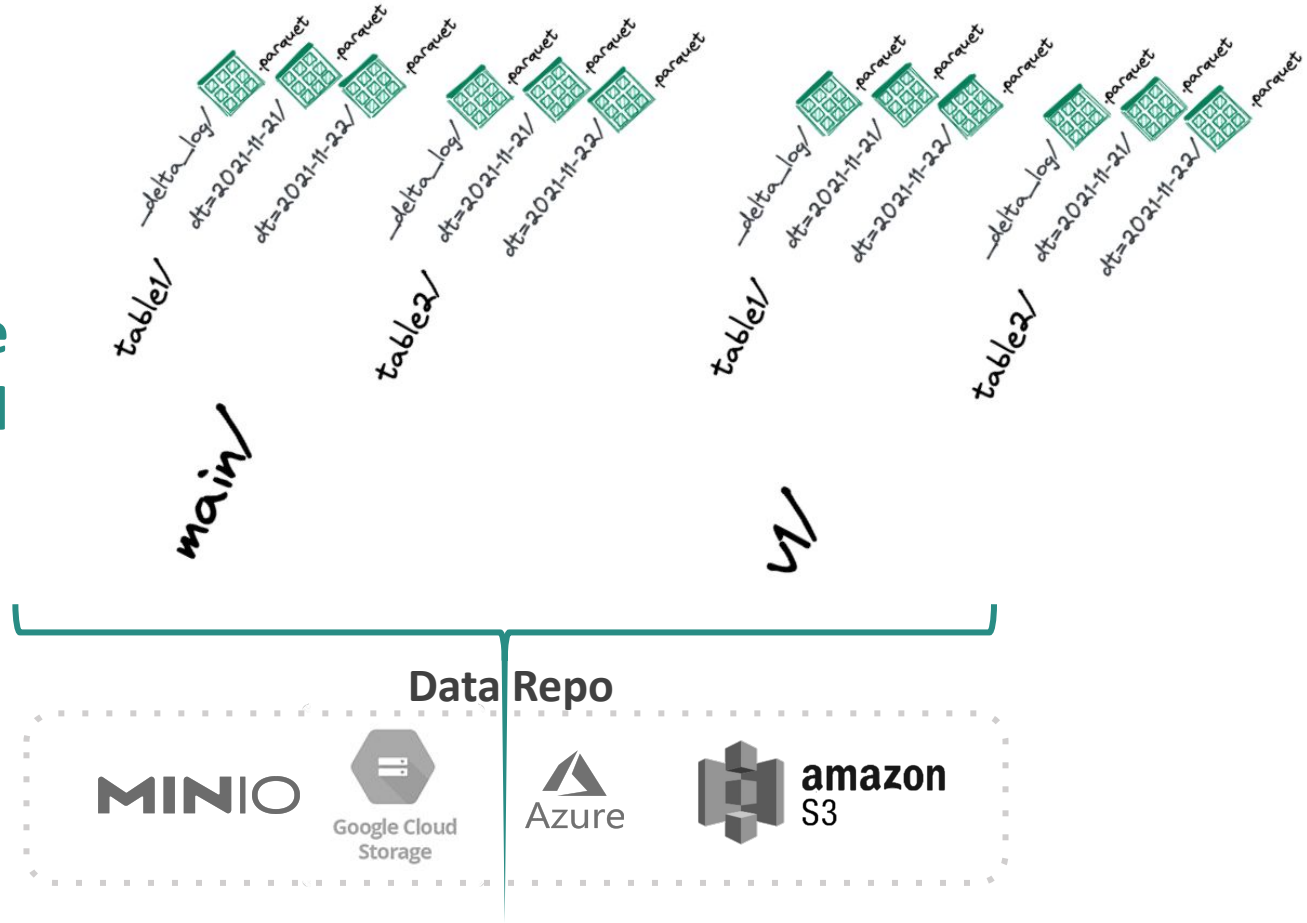
Dev Environment creation

Reproducing ML experiments

Leveling Up Data Lake Takeaways

Stop operating at the **file level**

Start operating at the **table** and **repository level**



lakeFS.io

THANK YOU!