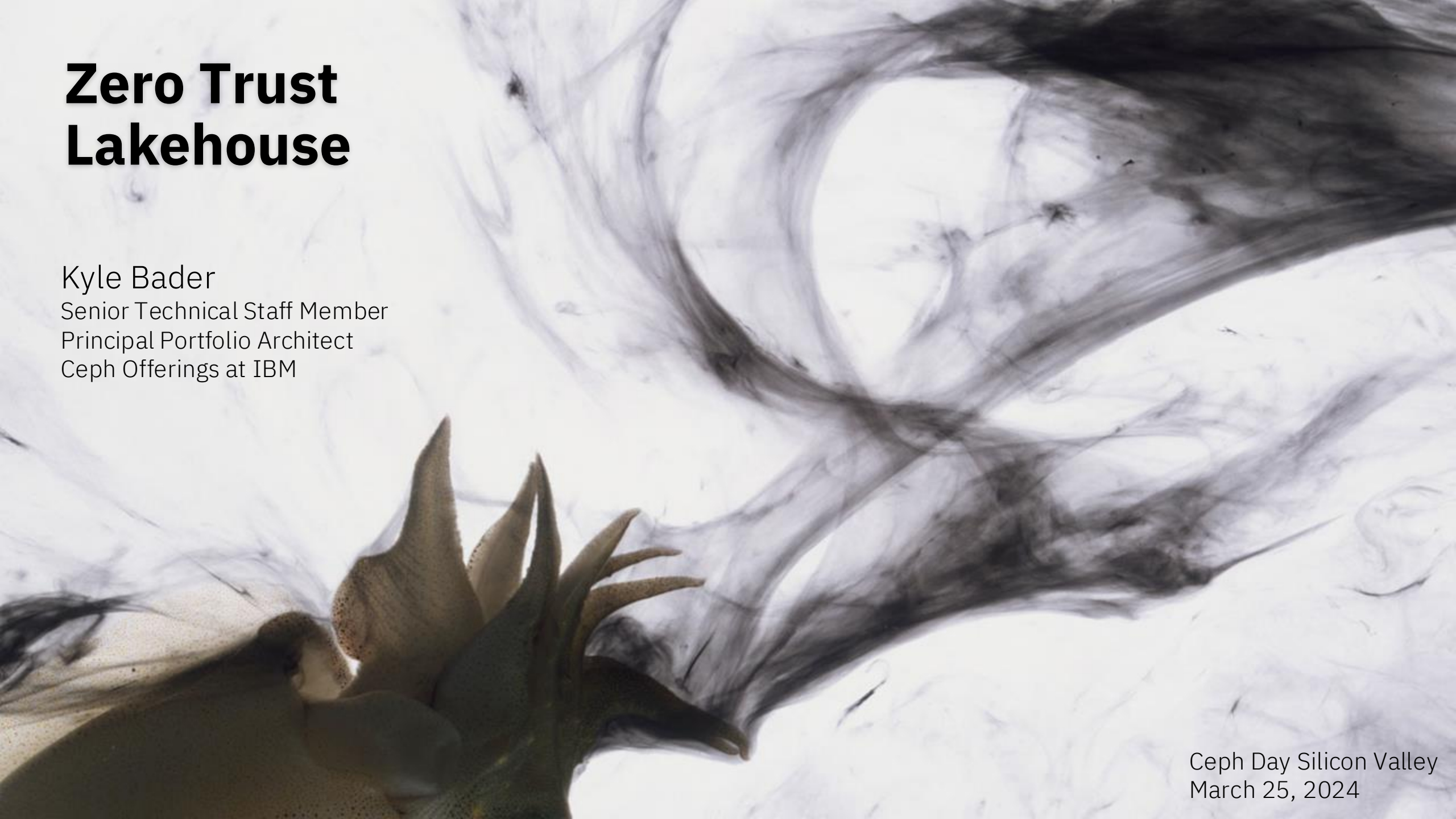
Zero Trust Lakehouse



Kyle Bader
Senior Technical Staff Member
Principal Portfolio Architect
Ceph Offerings at IBM

Ceph Day Silicon Valley
March 25, 2024

Core challenge for data lakehouse

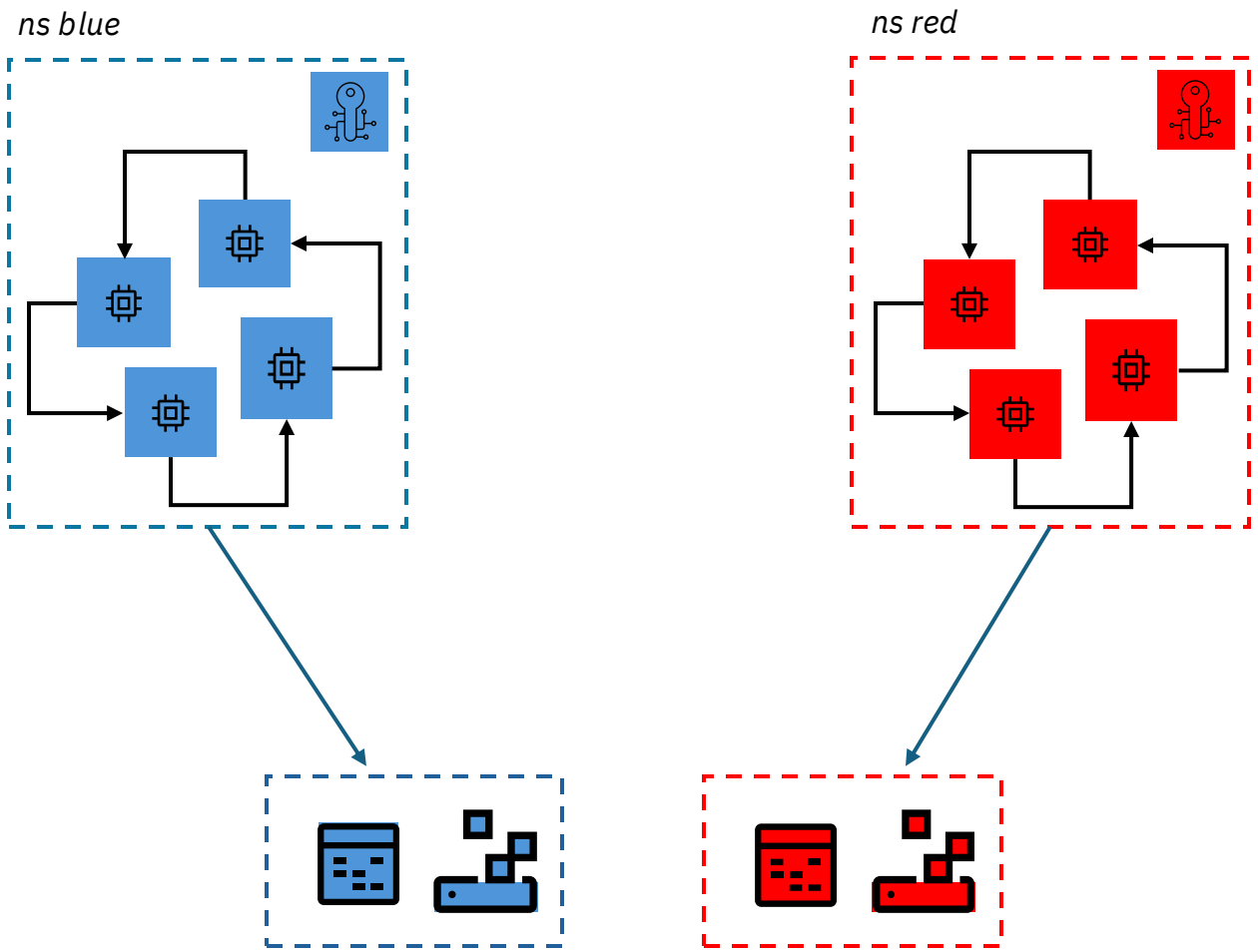
1. Database-level access semantics (what users intuitively expect)
2. Storage-level enforcement (what zero-trust requires)
3. While maintaining direct paths (what performance and scalability demands)

A strategy that incorporates a technical catalog into our product helps us move up the value chain and *keep moving up*.

We also want to push storage-level enforcement to be more granular, instead of stopping at table level access control we could introduce row and column level security, all while pushing work further down the direct path to maximize performance and efficiency.

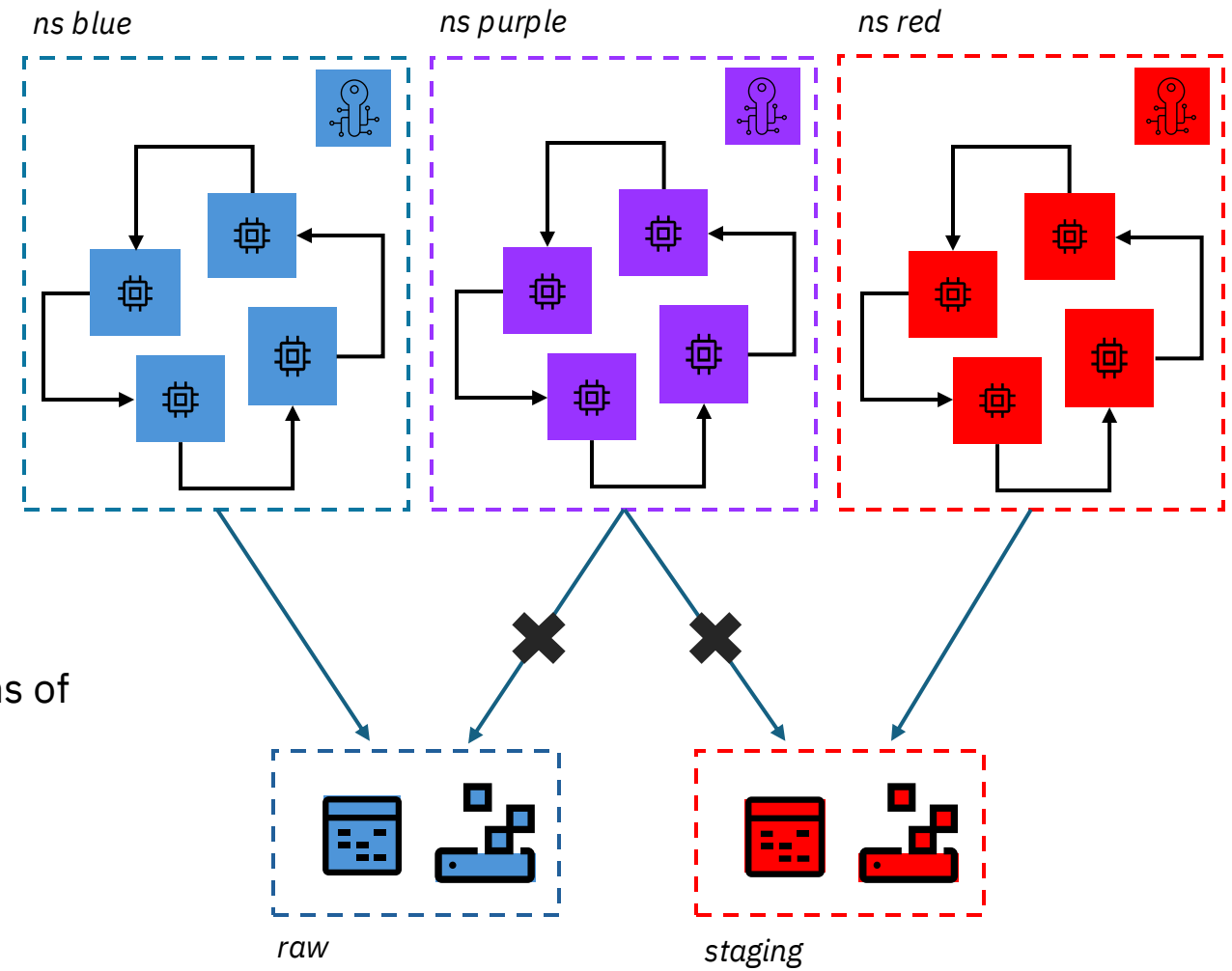
Access Control

Secure and govern: Namespace scoped secrets



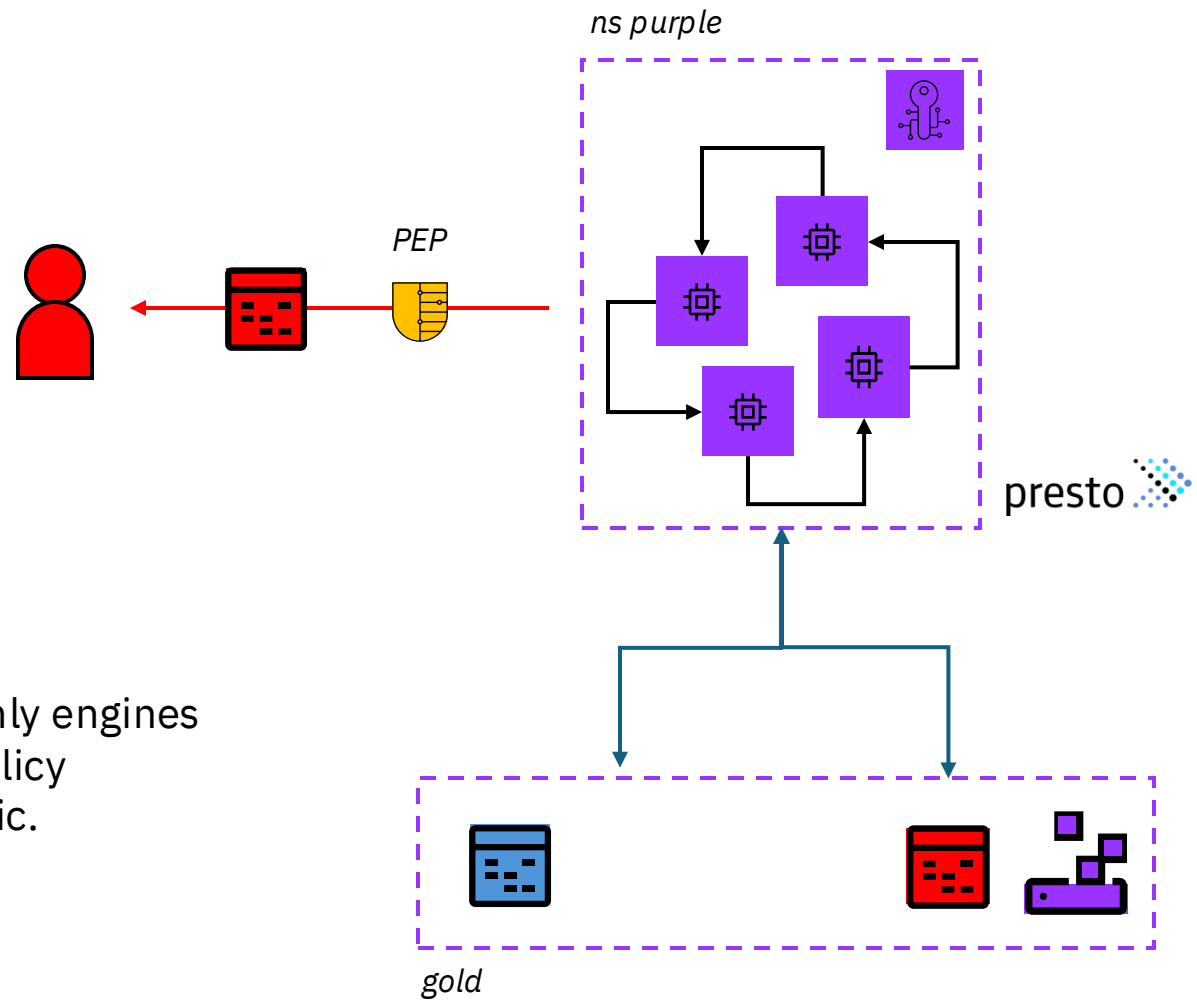
Buckets have 1:1 relationship with namespaces, dataset per bucket.

Secure and govern: Namespace scoped secrets



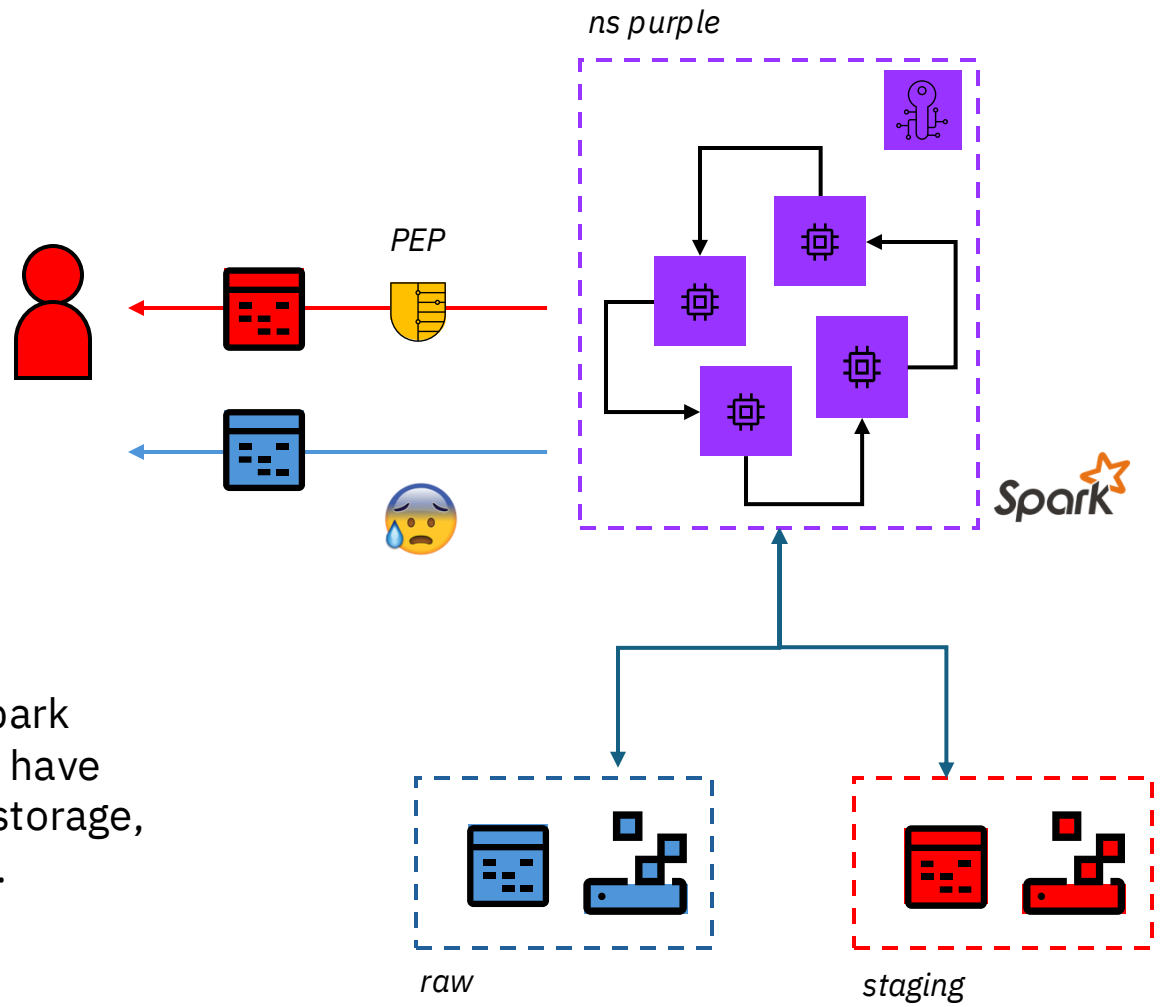
Highlights the limitations of ObjectBucketClaim for Lakehouse workflows.

Secure and govern: Engine policy enforcement



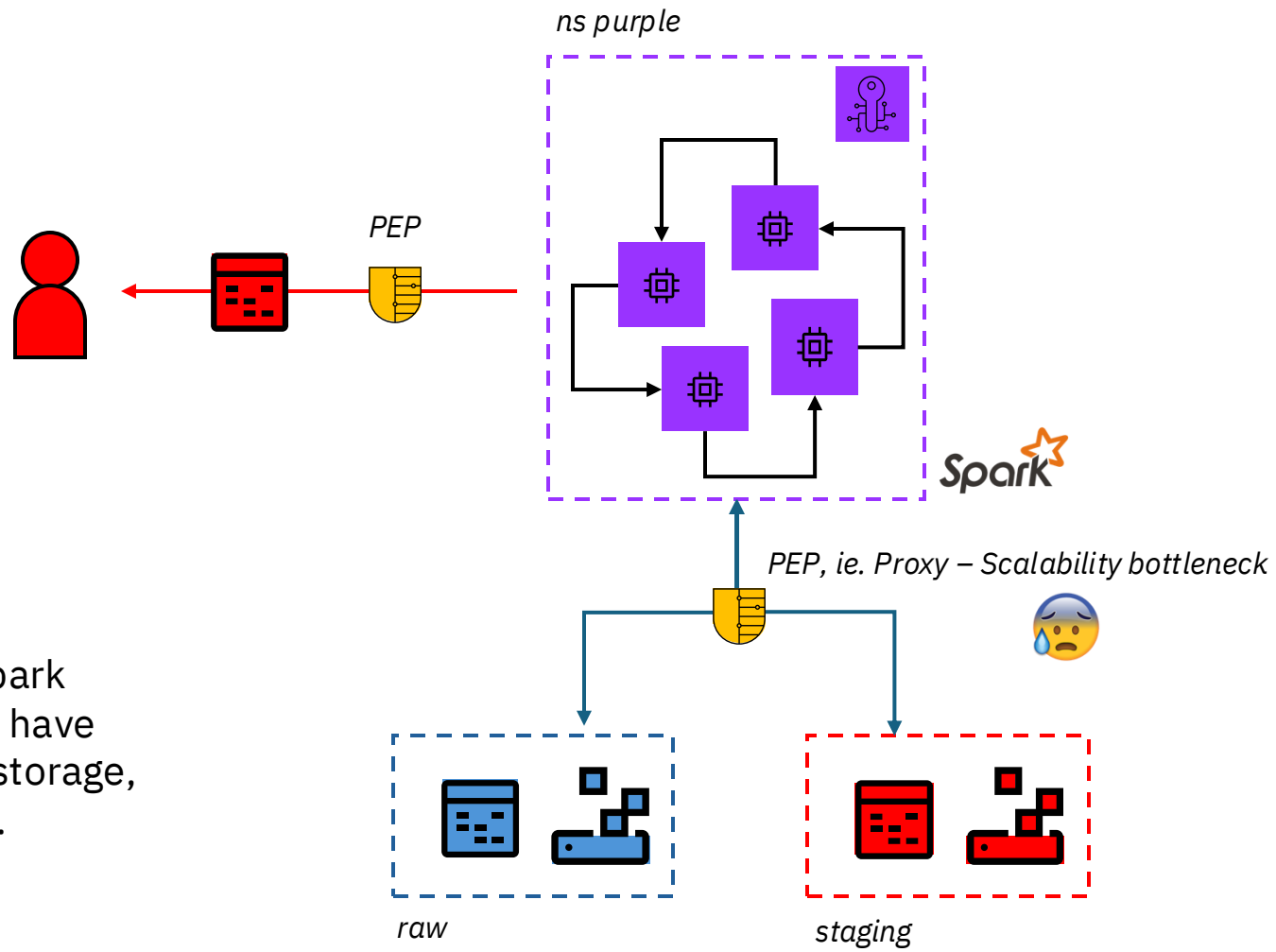
Viable for SQL only engines with common policy enforcement logic.

Secure and govern: Engine policy enforcement



Unsuitable for Spark applications that have direct access to storage, bypassing policy.

Secure and govern: Engine policy enforcement



Unsuitable for Spark applications that have direct access to storage, bypassing policy.

Technical Catalogs

Table data service

Proof of concept Terraform to:

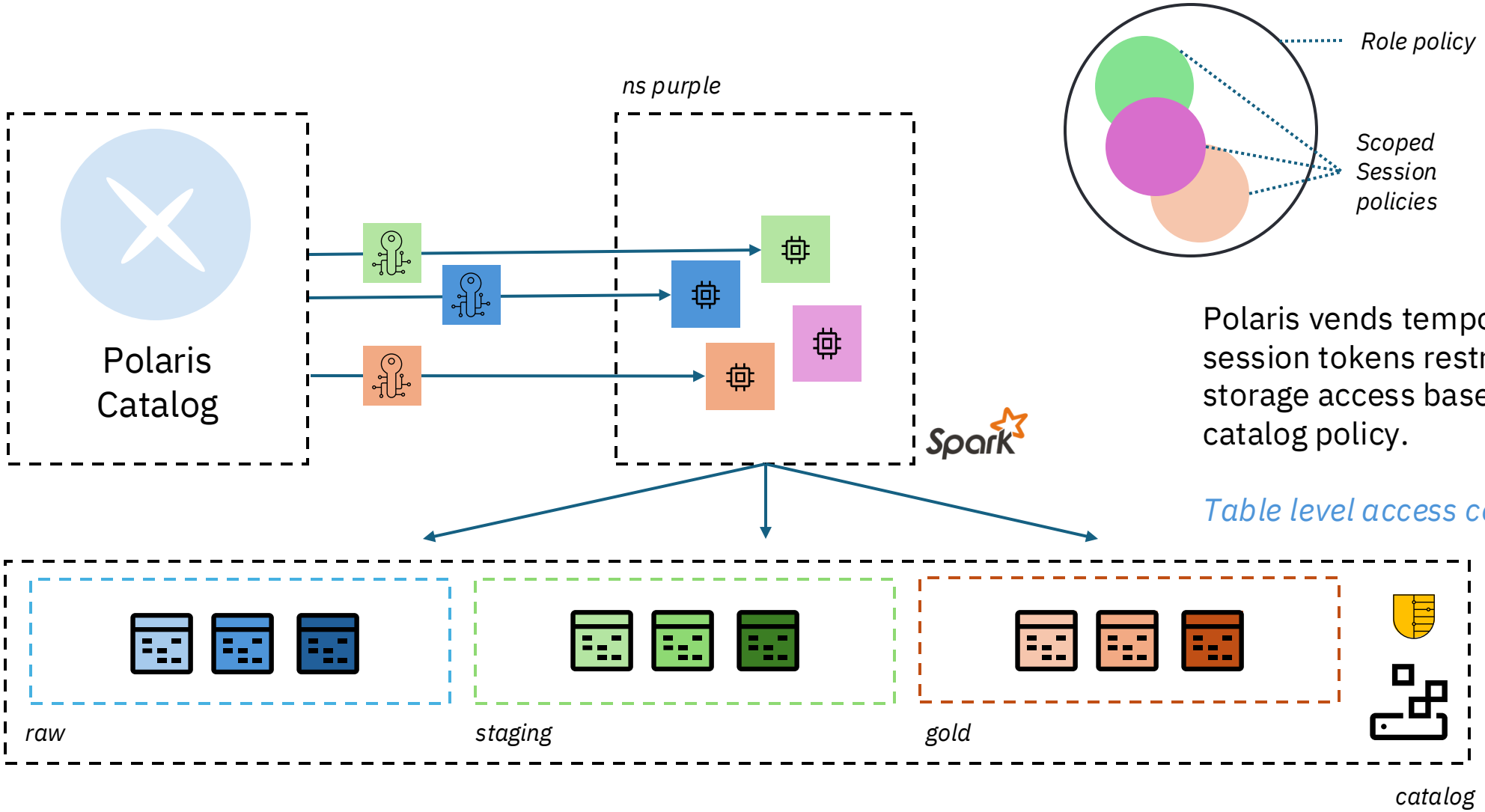
- Deploy Polaris
- Create and manage service dependencies on Ceph resources (S3, IAM)
- Create and manage Polaris resources

Could be productized as a *Polaris operator* delivered by *IBM Fusion*.
Differentiates IBM offerings from community Ceph.

Strong competitive counter to AWS Lake Formation

https://github.com/mmgaggle/polaris-ceph-demo/blob/main/ceph-resources.tf

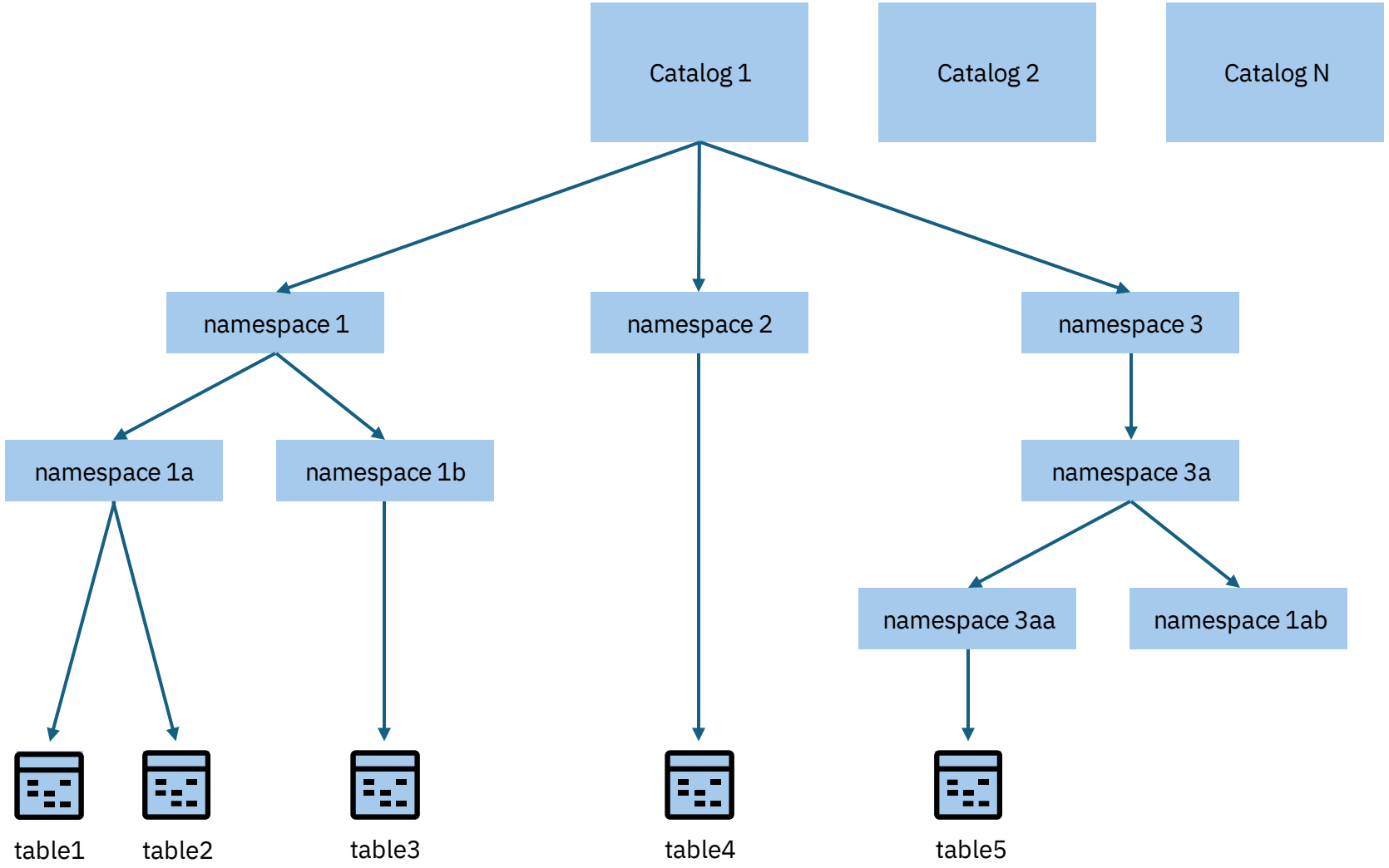
Secure and govern: Catalog credential vending



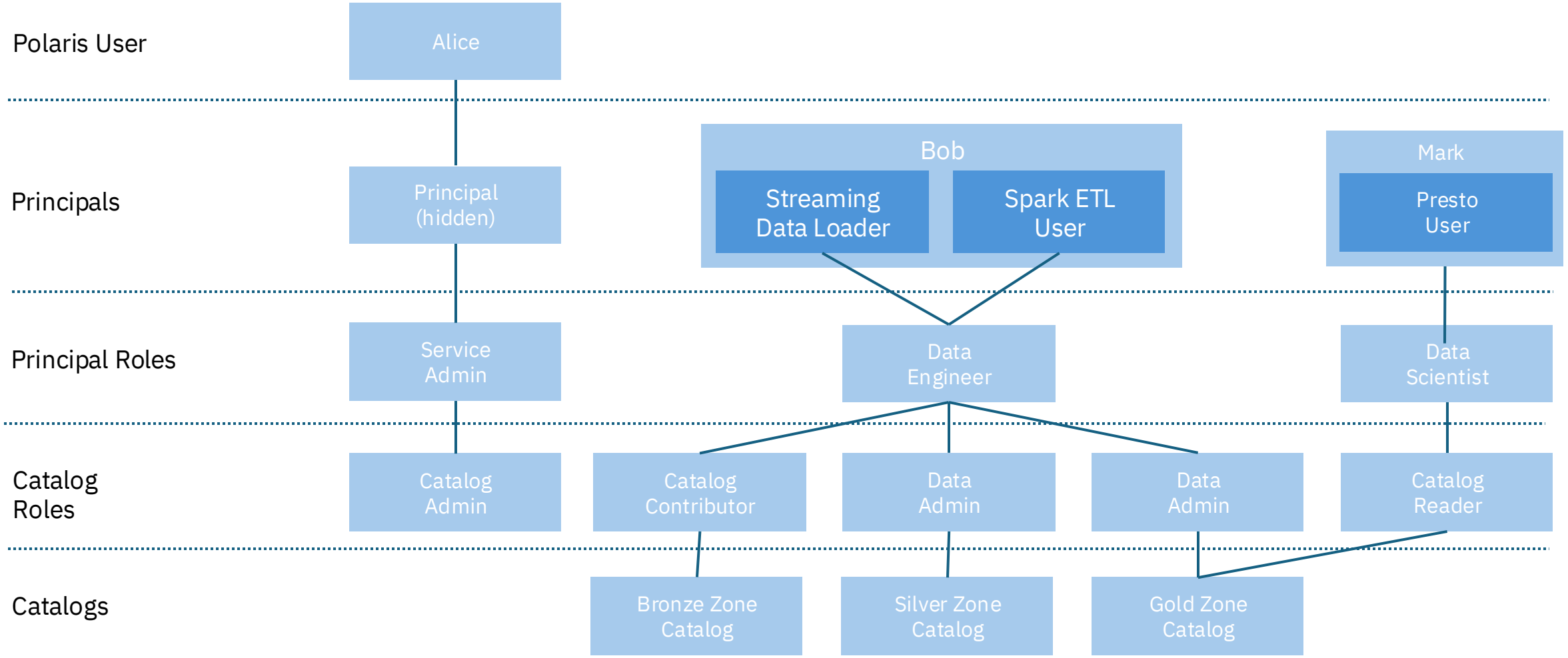
Polaris vends temporary session tokens restricting storage access based on catalog policy.

Table level access control

Secure and govern: Namespaces



Secure and govern: RBAC example



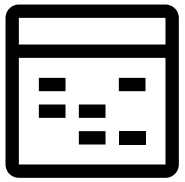


Table optimization

Catalog unlocks background update services such as

- Table compaction
  - Combine small files into fewer larger ones
  - Merging delete files with data files generated by Iceberg merge-on-read
  - *Clustering tables by z-order*
  - *Vector index regeneration*

- Snapshot management
  - Prune old table snapshots

- Unreferenced file removal
  - Expire objects not referenced in any table snapshots

Table Maintenance Integration with Polaris

Secure and govern: RBAC for other tables?

```
foo/                              # Iceberg table ‘foo’
  ├── metadata/                   # Contains all table metadata files
  │   ├── version-hint.text       # Points to current metadata version
  │   └── v[version].metadata.json # Metadata file for each version
  ├── data/                       # Contains data files
  │   └── [partition_spec]/       # Optional partition directories
  │   └── [file_group]/           # Groups of data files
  │   └── [data_file].parquet     # Data files (usually Parquet)
  └── snapshots/                  # Contains snapshot metadata
      └── snap-[id].avro          # Snapshot metadata files
```

s3-tags are applied to these objects to map data files to tables and namespaces, and to use as conditions in session policies

```
s3://foo/                         # LanceDB root location
  ├── bar/                        # Table 'bar’
  │   ├── data/                   # Column data files
  │   ├── indices/                # Index files
  │   ├── _latest.manifest        # Points to current version
  │   └── schema.arrow            # Table schema
  ├── baz/                        # Table 'baz’
  │   ├── data/
  │   ├── indices/
  │   ├── _latest.manifest
  │   └── schema.arrow
```

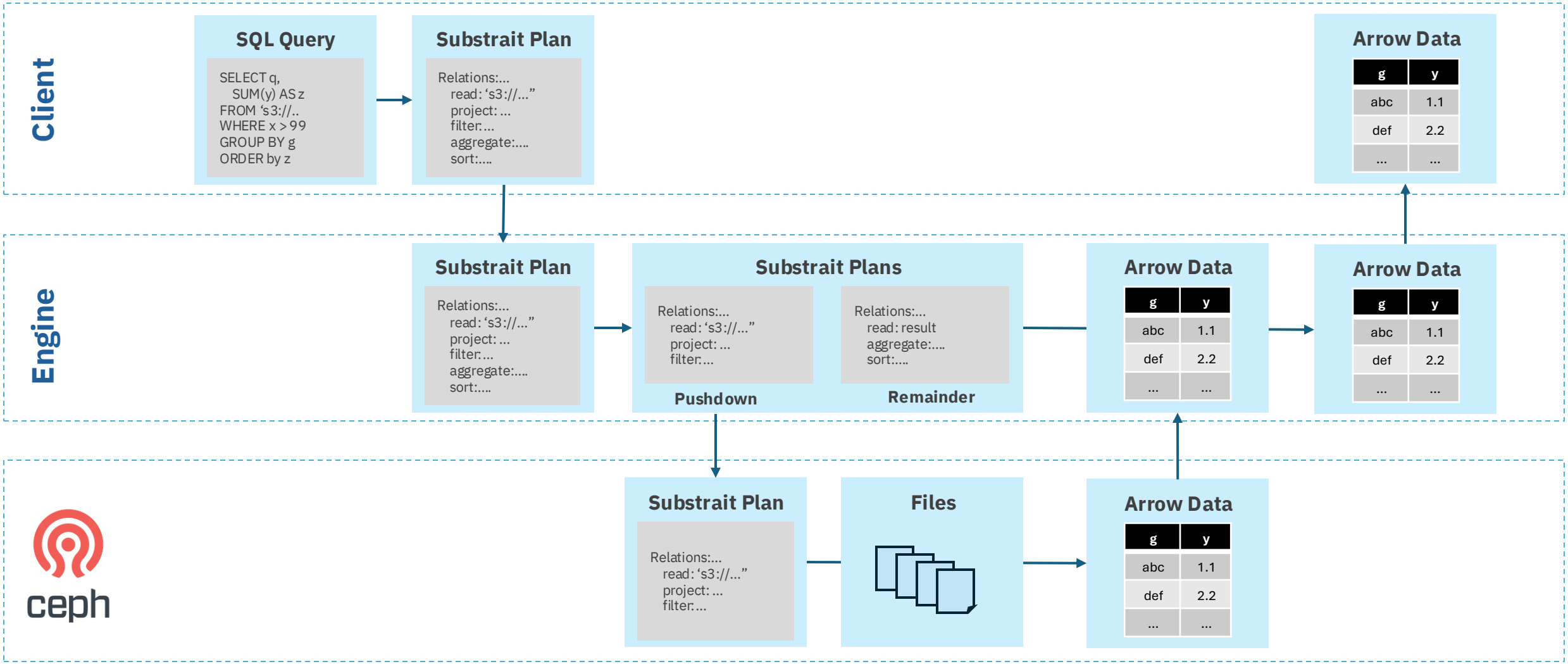
Polaris like access controls could easily be extended to lance tables with a wrapper or native support for s3-tags and client side support for token vendors

Delta (and other format) Table Support in Polaris

Extra

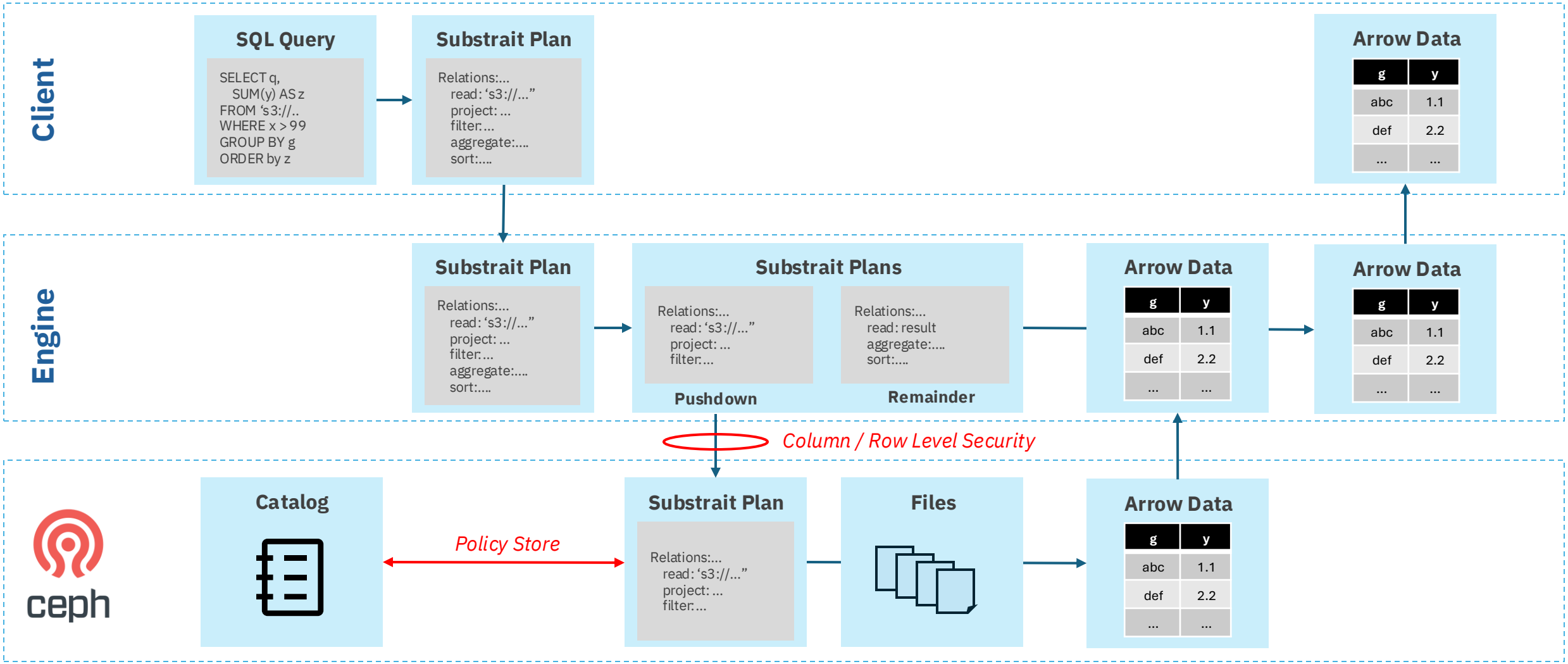
Read API for structured records

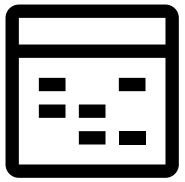
Volley
Read API



Read API for structured records: secure and govern

Volley
Read API





Directory tables

Add notion of Volumes and Directory tables to organize unstructured data

- Secure and govern unstructured like structured data
- Structured information about unstructured data
- AI use cases
  - Features (training), vectors (RAG), and full-text search through external indexes
  - FAISS or DiskANN for vector

- Bucket notifications to pub/sub for ISV software, serverless (Knative), AI extractors

Unstructured Data Support in Polaris