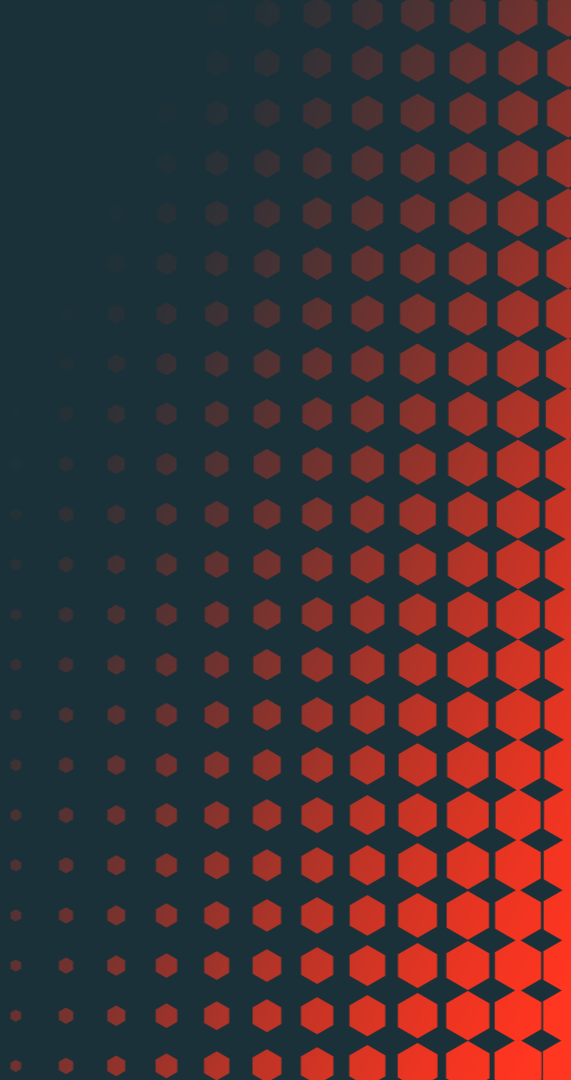
PII & Regulatory Compliance



databricks

GDPR (General Data Protection Regulation)

Intentions of the ACT:

- Know what personal data is collected and how it is used
- Have incorrect personal data updated
- Have personal data erased / “to be forgotten”
- Have personal data exported

Businesses obligations:

- Companies have 30 days to comply with the request
- Penalties up to 4% of annual revenues or €20M, whichever is greater

CCPA (California Consumer Privacy Act)

Intentions of the ACT:

- Right to know what personal data is collected, used, shared or sold
- Right to have personal data deleted by businesses and business's service provider
- Right to opt-out of sale of personal information
- Right to non-discrimination in terms of price or service

Businesses obligations:

- Confirm receipt within 10 business days of request
- Process request within 45 calendar days
- Fines of up to $2,500 per violation and $750 per consumer per incident

Compliance Requirements

- Inform customers what personal information is collected
- Delete, update, or export personal information as requested
- Process request in a timely fashion (30 days)

How Lakehouse Simplifies Compliance

- Reduce copies of your PII
- Find personal information quickly
- Reliably change, delete, or export data
- Use transaction logs for auditing

Manage Access to PII

- Control access to storage locations with cloud permissions
- Limit human access to raw data
- Pseudonymize records on ingestion
- Use table ACLs to manage user permissions
- Configure dynamic views for data redaction
- Remove identifying details from demographic views

Common Approaches to De-identification

Pseudonymization

Protects datasets on record level for **Machine Learning**.

Switches original data point with pseudonym for **later re-identification**, inaccessible to unauthorized users.

A pseudonym is still considered to be personal data according to the GDPR.

Anonymization

Protects entire tables, databases or entire data catalogues mostly for **Business Intelligence**.

Personal data is irreversibly altered in such a way that a data subject can no longer be identified **directly or indirectly**.

Usually a combination of more than one technique used in real-world scenarios.

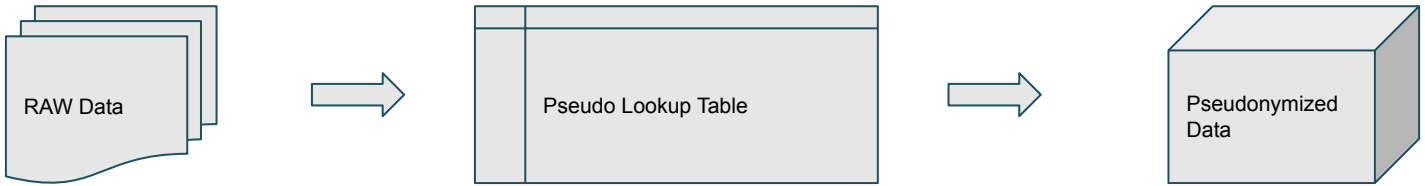
kpkrdiTAnqvfxuyE	**********
1AYrGFCTTRYOw0	**********
72AjraZ8sU9EsNw	**********

Pseudonymization

Replacing personal information with an artificial identifier or pseudonym, prior to the receipt of the data subject request.

Keeping a secured “lookup table” that maps the customer’s personal identifiers (name, email address, etc) to the pseudonym

Delete information then means deleting information from the “look-up” table and all other information becomes unidentifiable



Hashing

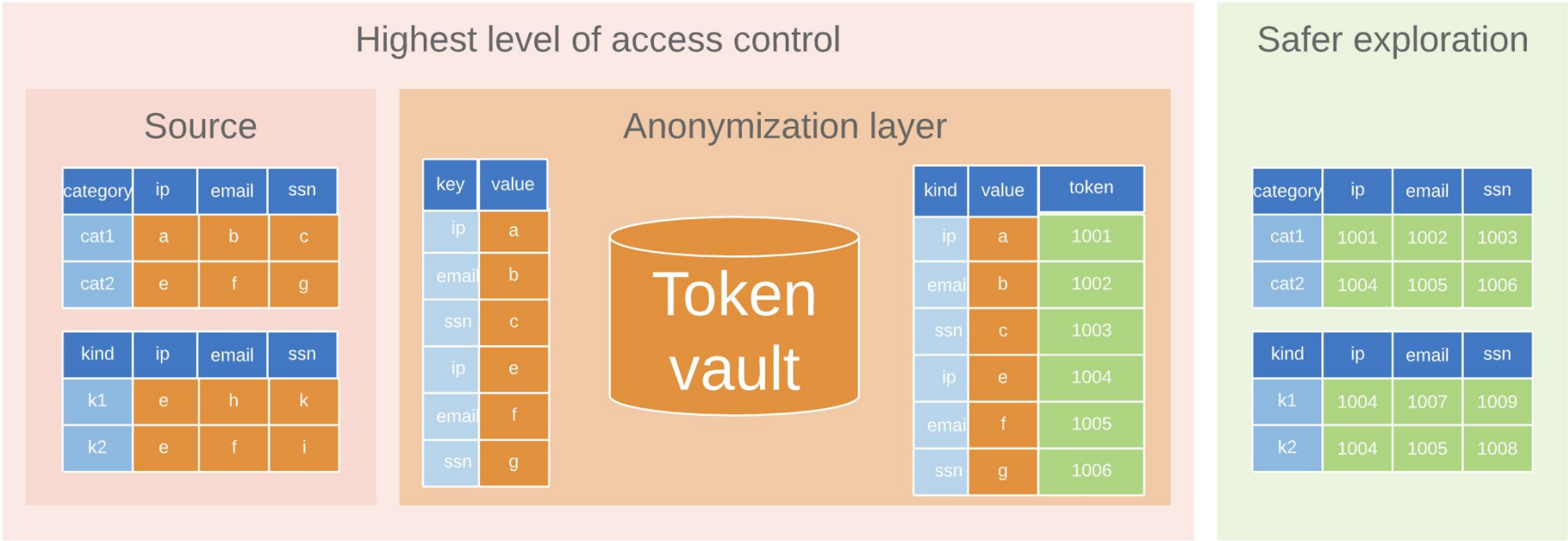
Just SHA512() the sensitive data

- Usefulness: high
- Difficulty: easy
- Schema: same
- Format: different
- Performance: group by is slower
- Re-identification: hashcat, dictionary and/or combinator attacks

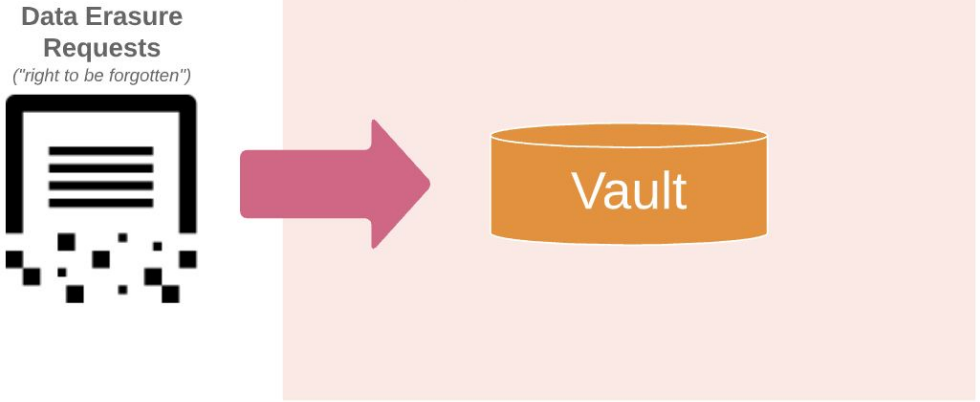
Tokenization

Non-inferrable source-to-destination mapping

- Usefulness: very high
- Difficulty: high
- Schema: almost
  - strings become longs, making performance higher
- Format: different
- Performance: slower to write, faster to read
- Re-identification: depends


Highest level of access control
Source
category
ip
email
ssn
cat1
a
b
c
cat2
e
f
g
kind
ip
email
ssn
k1
e
h
k
k2
e
f
i
Anonymization layer
key
value
ip
a
emai
b
ssn
c
ip
e
emai
f
ssn
g
Token vault
kind
value
token
ip
a
1001
emai
b
1002
ssn
c
1003
ip
e
1004
emai
f
1005
ssn
g
1006
Safer exploration
category
ip
email
ssn
cat1
1001
1002
1003
cat2
1004
1005
1006
kind
ip
email
ssn
k1
1004
1007
1009
k2
1004
1005
1008

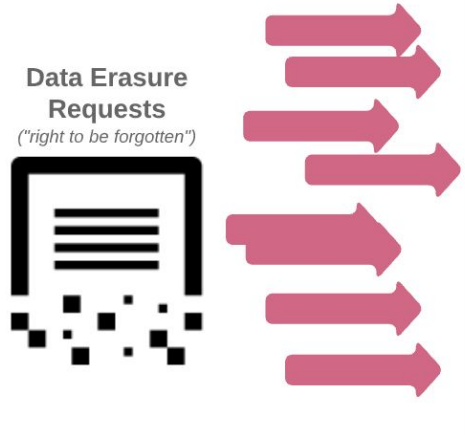
GDPR with Vault



versus

GDPR without any vault

Vault-based systems might be difficult to build, but without it coping with GDPR Data Erasure Requests might be, in fact, more difficult



databricks

Column & Row suppression

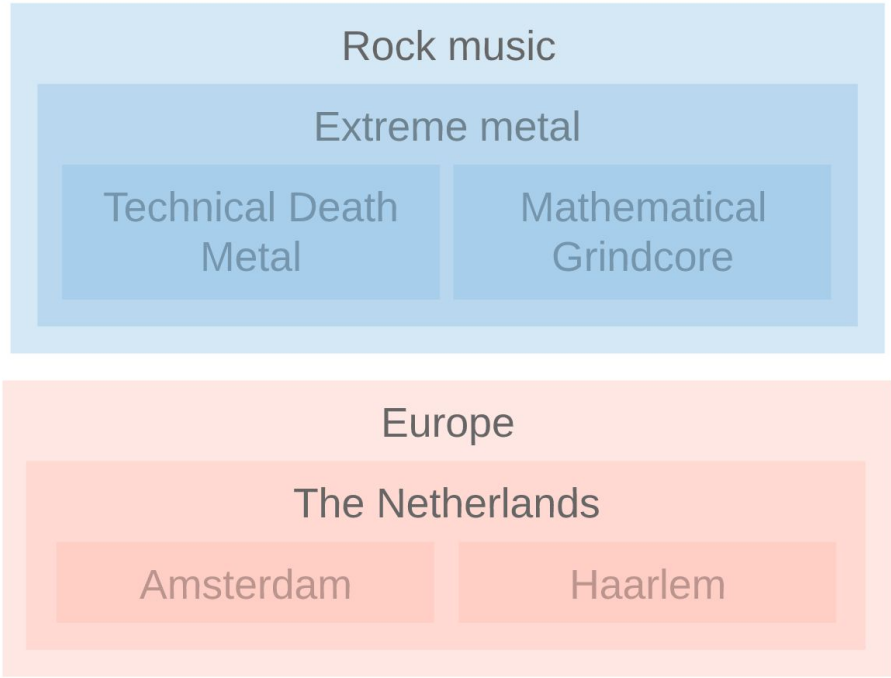
Is not always enough for both data scientists and those, who care about protecting the data, because few outliers may lead to identification, when joined with external sources

- Accuracy: medium value
- Difficulty: easy
- Schema: fewer columns
- Performance: less data
- Re-identification: be aware of outliers

Generalisation

Remove precision from data

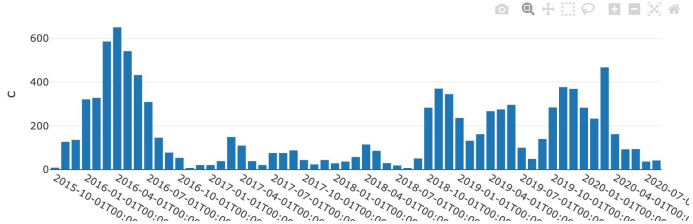
- Accuracy: medium to high
- Difficulty: trivial with rules. Fun with Federated Learning.
- Schema: same
- Format: same
- Performance: minimal impact
- Re-identification: difficult, but possible



Binning

APPROX_PERCENTILE is super performant and generic way to add contextual binning to any existing Spark DataFrame

```
%sql SELECT date_trunc('MONTH', joindate) AS joined, count(*) AS c FROM resellers GROUP BY 1 ORDER BY 1
```




```
1  import org.apache.spark.sql.functions._
2
3  val df = spark.table("resellers").select('joindate)
4    .withColumn("tenure_in_years",
5      round(datediff(current_date(), 'joindate) / 365, 2))
6    .withColumn("ntiles",
7      expr("approx_percentile(tenure_in_years, array(0.33, 0.66, 0.95)) OVER ()"))
8    .withColumn("tenure",
9      when('tenure_in_years <= 'ntiles(0), lit("NEW"))
10     .when('tenure_in_years <= 'ntiles(1), lit("SEASONED"))
11     .when('tenure_in_years <= 'ntiles(2), lit("SENIOR"))
12     .otherwise(lit("EARLY_EMPLOYEE")))
13   .drop("ntiles")
14
15 display(df)
```

Sensitive

Simple clustering

(2) Spark

df: org.apache.spark.sql.DataFrame = [joindate: date, tenure_in_years: double ... 1 more fields]

	joindate	tenure_in_years	tenure
8	2019-12-01	0.91	NEW
9	2020-04-01	0.58	NEW
10	2016-04-17	4.53	SENIOR
11	2017-08-08	3.22	SEASONED
12	2016-04-27	4.51	SENIOR
13	2016-01-26	4.76	SENIOR
14	2016-12-06	3.9	SEASONED
15	2015-11-19	4.95	EARLY_EMPLOYEE
16	2019-12-06	0.9	NEW
17	2017-08-25	3.18	SEASONED

Directly derived

Window [joindate#610172, tenure_in_years#610216, approx_percentile(tenure_in_years#610216, [0.33,0.66,0.95], 10000, 0, 0) windowspecdefinition(specifiedwindowframe(RowFrame, unboundedpreceding$(), unboundedfollowing$())) AS ntiles#610219]

WindowStages: 40629.0

WholeStageCodegen (2)

databricks

Truncating: IP addresses

- Rounding IP address to /24 CIDR is considered anonymous enough, if other properties of a dataset allow.
- IP-geolocation (MaxMind, IP2Location) databases would generally represent it as city or neighbourhood level, where the real IP would represent a street.

```
1  import spark._
2  val df = Seq("10.130.176.215", "10.5.56.45",
3      "10.208.126.183", "10.10.14.84",
4      "10.106.62.87", "10.190.48.173",
5      "10.141.100.19", "10.153.248.211",
6      "10.121.140.220", "10.37.240.84").toDF
7    .withColumn("ip_truncated",
8      concat(substring_index('value, ".", 3), lit(".0/24")))
9  display(df)
```

	value	ip_truncated
1	10.130.176.215	10.130.176.0/24
2	10.5.56.45	10.5.56.0/24
3	10.208.126.183	10.208.126.0/24
4	10.10.14.84	10.10.14.0/24
5	10.106.62.87	10.106.62.0/24
6	10.190.48.173	10.190.48.0/24
7	10.141.100.19	10.141.100.0/24

Showing all 10 rows.

Rounding

Example rounding rules:

- All numbers are rounded to the nearest multiple of 15
- Any number lower than 7.5 is rounded suppressed
- Halves are always rounded upwards (e.g. 2.5 is rounded to 5)

```
1  val modulo = 15
2  val df = spark.table("resellers")
3    .groupBy('city, 'industry)
4    .count()
5    .withColumn("count_rounded",
6      when('count <= modulo/2, null)
7       .otherwise(ceil('count / modulo) * modulo))
8    .where('count_rounded.isNotNull)
9
10 display(df)
```

▸ (2) Spark Jobs

▸ df: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [city: string, industry: string ... 2 more fields]

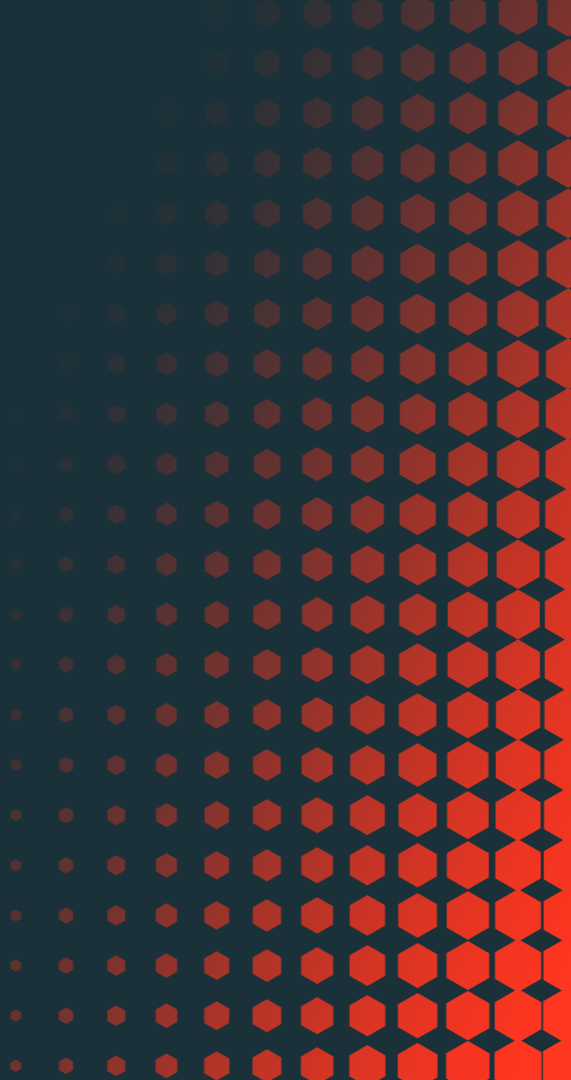
	city	industry	count	count_rounded
1	Rotterdam	Fishing	11	15
2	Tilburg	Wholesale Trade	82	90
3	Tilburg	Finance	90	90
4	Monnickendam	Insurance	9	15
5	Almere	Manufacturing	15	15
6	Rotterdam	Electric, Gas and Sanitary service	243	255
7	Rotterdam	Communications Service	178	180

Showing all 112 rows.

Notebook: Creating a Pseudonymized PII Lookup Table

Notebook: Storing PII Securely

Managing ACLs for the Enterprise Lakehouse



databricks

Challenges with Analytics in the Data Lake

- Difficult to provide access on need-to-know basis
- Unclear governance & ownership of data assets
- Data proliferation resulting in possible compliance issues
- Difficult to ensure integrity of data and reporting

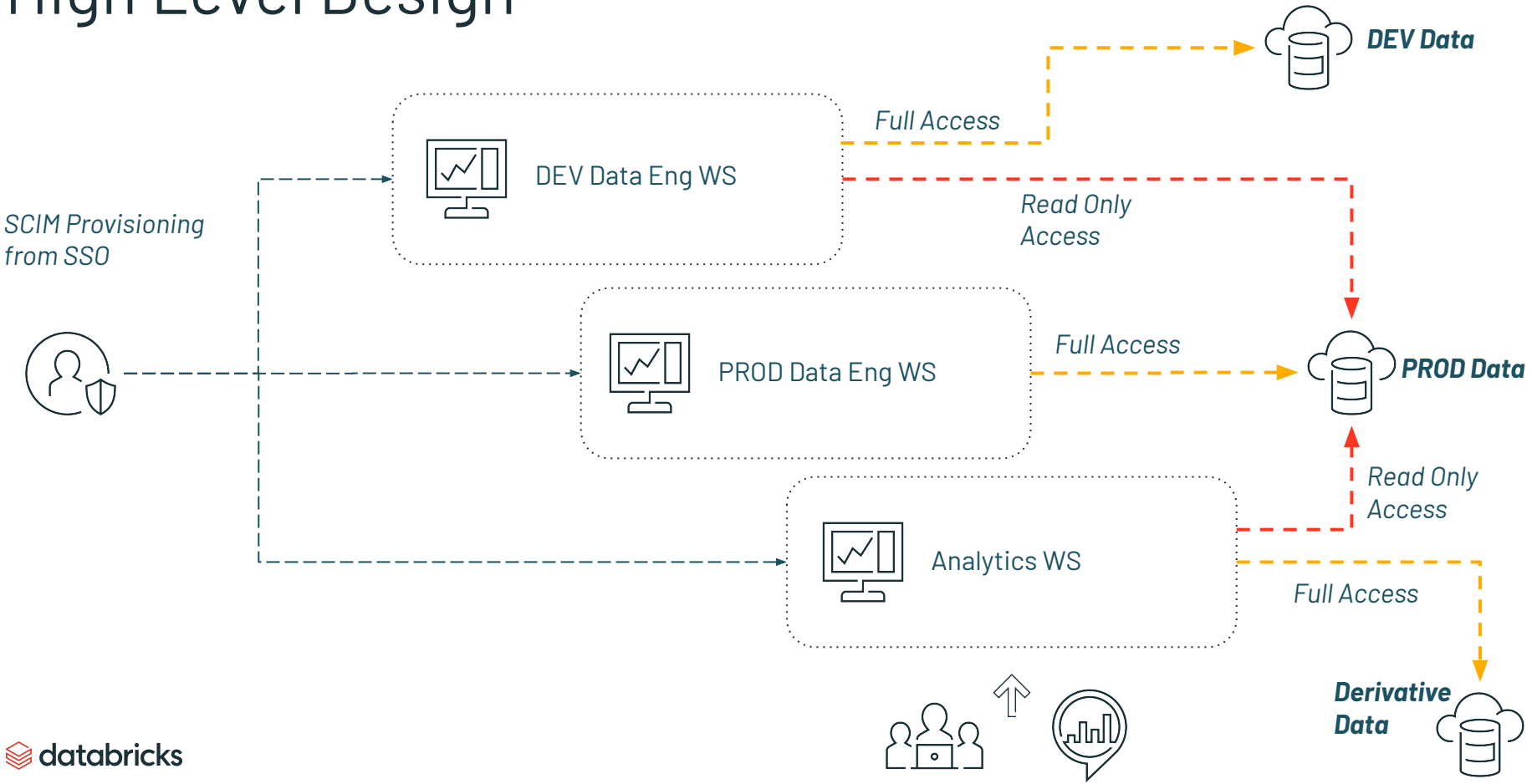
How Can We Answer The Following

“Can we provide access to valuable data to users across the company, in a secure manner?”

“Are users only able to access the data they’re entitled to?”

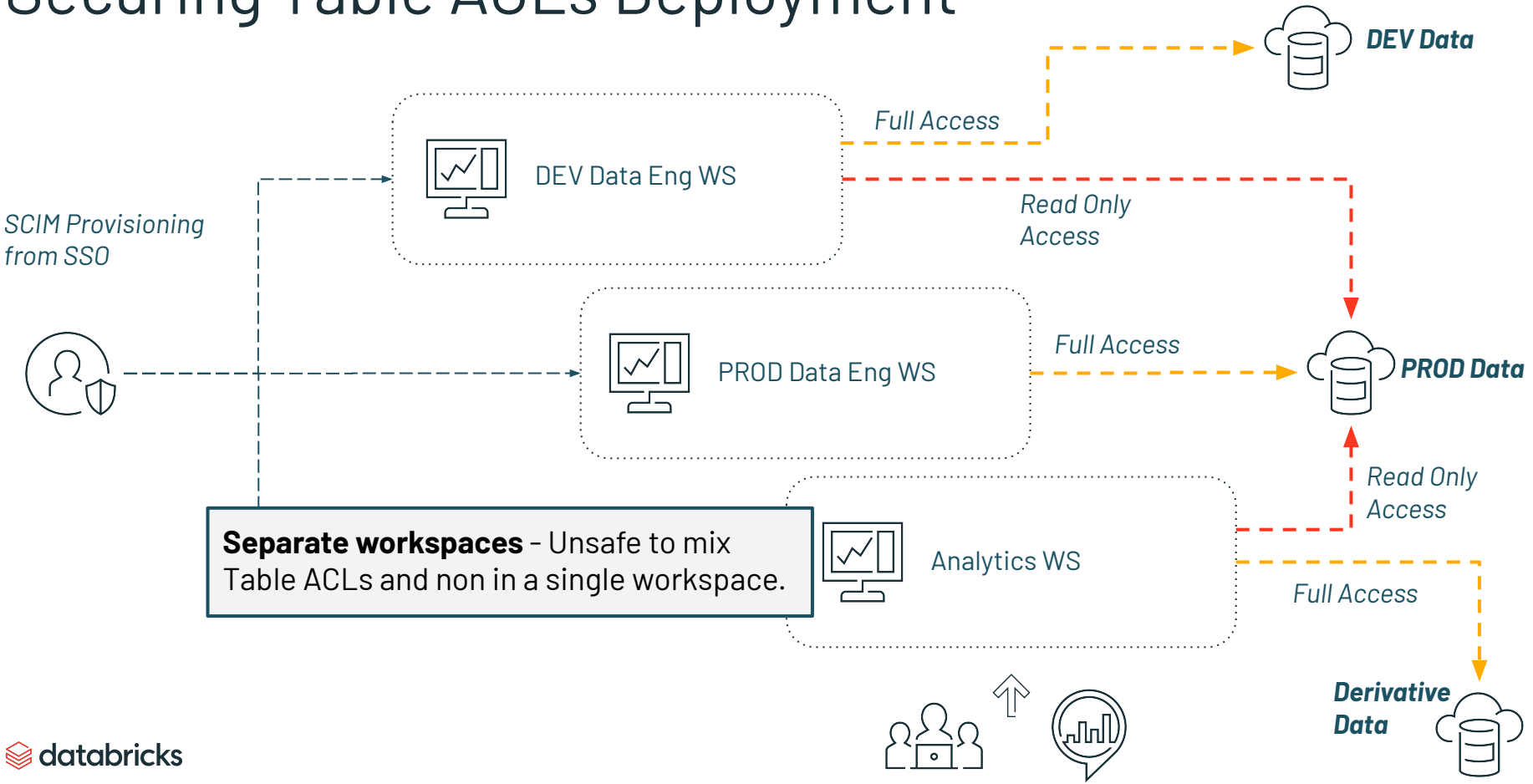
“Has any data been altered or manipulated?”

High Level Design



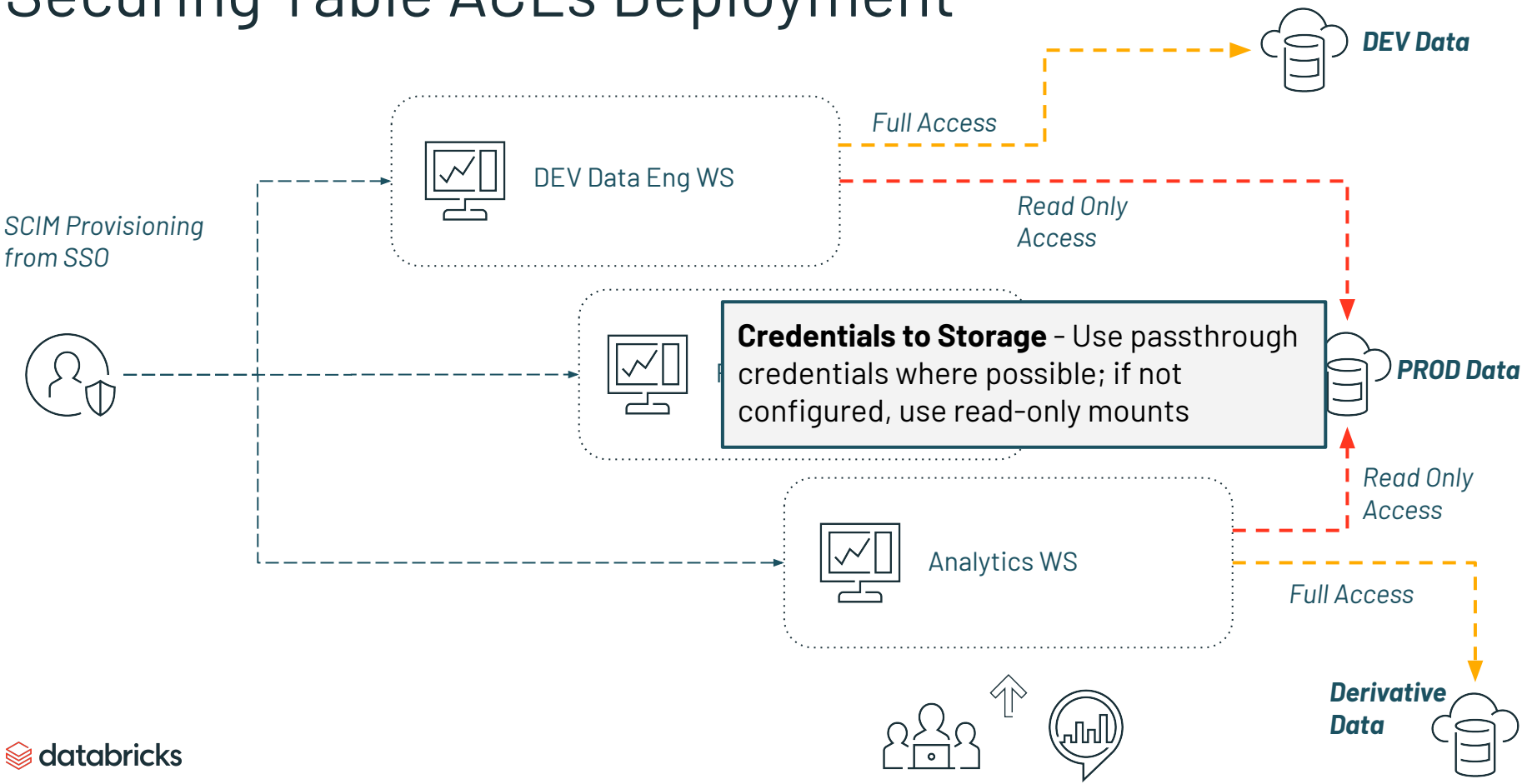
databricks

Securing Table ACLs Deployment



databricks

Securing Table ACLs Deployment



databricks

Securing Table ACLs Deployment

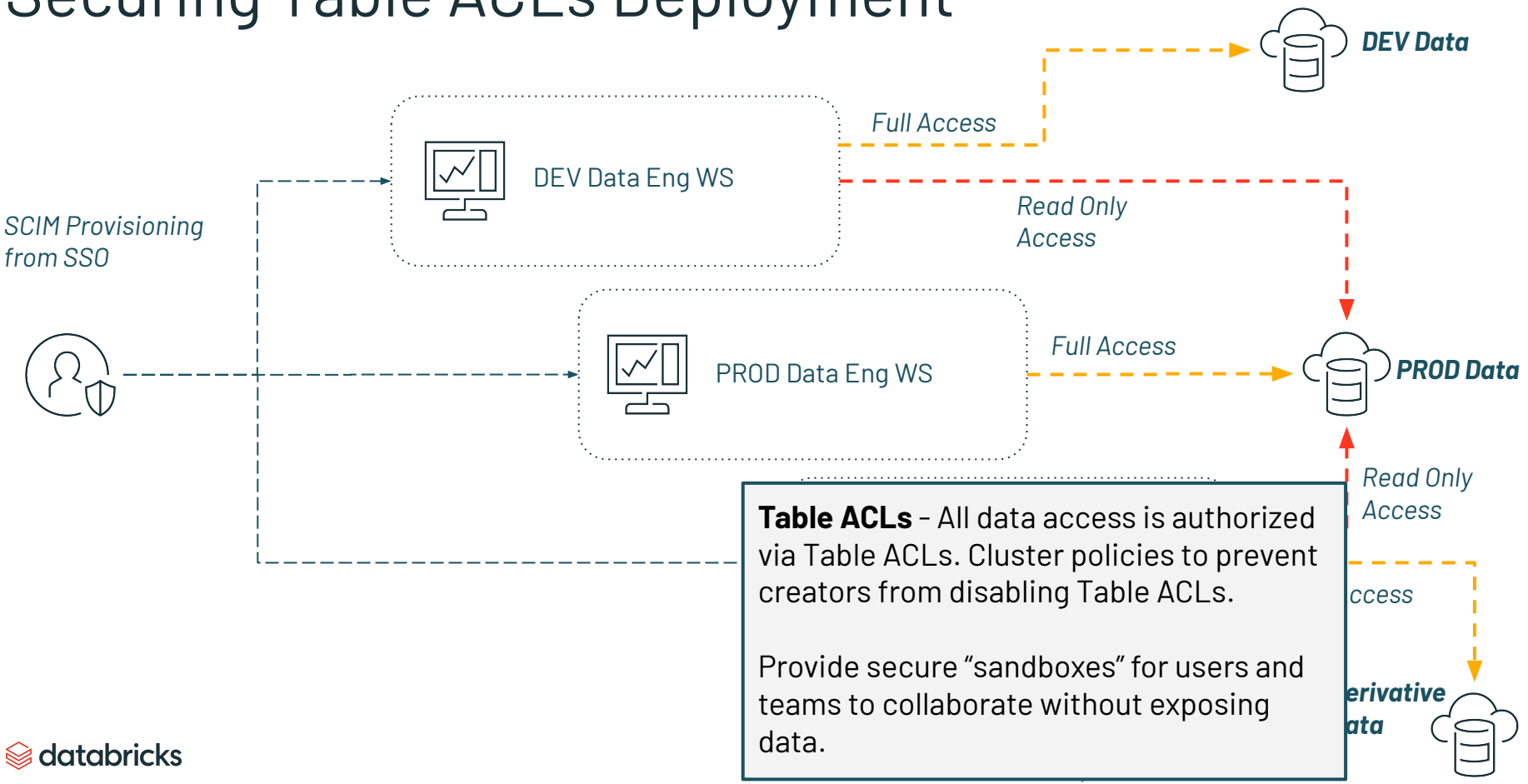
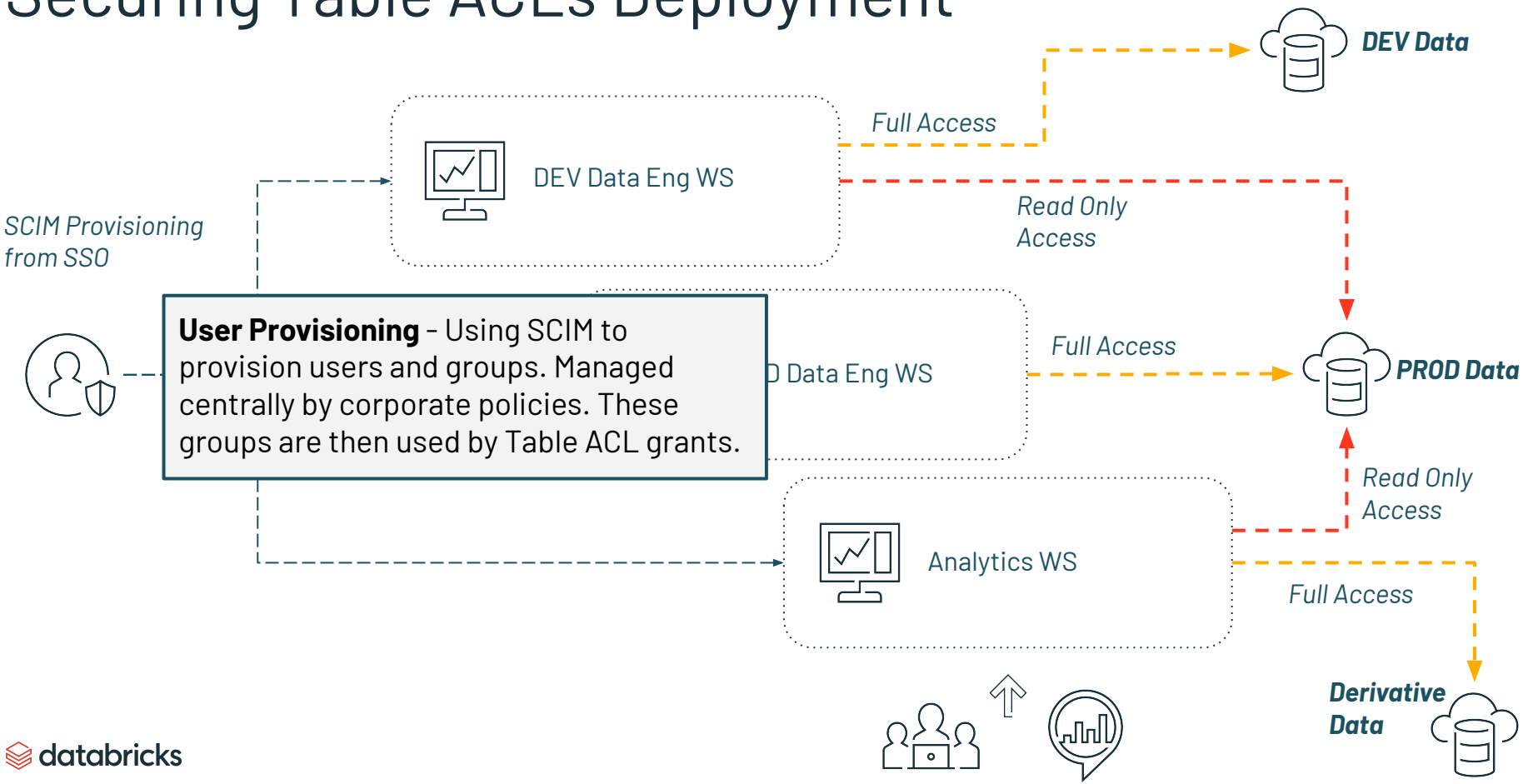


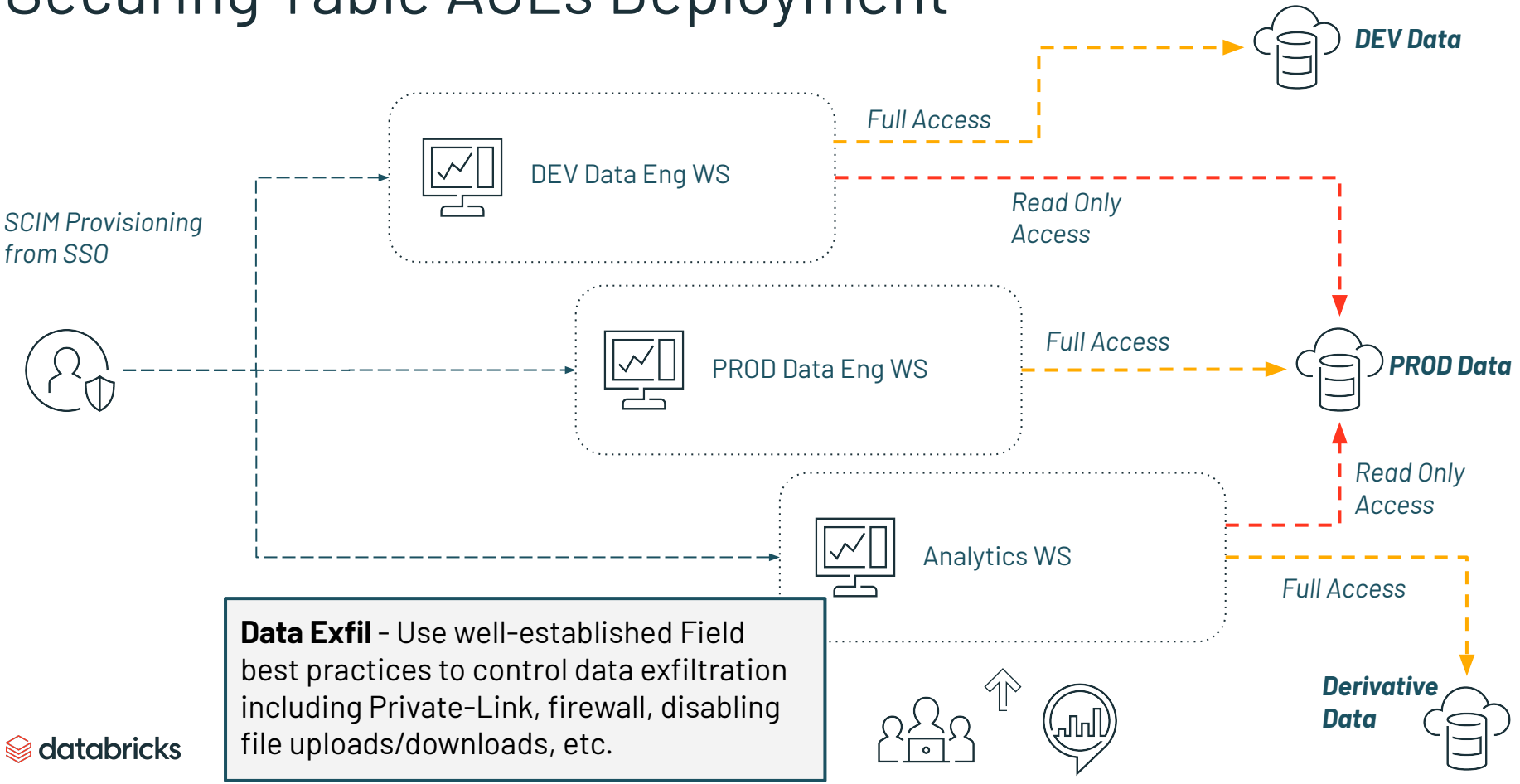
Table ACLs - All data access is authorized via Table ACLs. Cluster policies to prevent creators from disabling Table ACLs.

Provide secure "sandboxes" for users and teams to collaborate without exposing data.

Securing Table ACLs Deployment



Securing Table ACLs Deployment



Data Exfil - Use well-established Field best practices to control data exfiltration including Private-Link, firewall, disabling file uploads/downloads, etc.

databricks

Grant Access to Production Datasets

Assumptions

- End-users need read-only access
- Datasets organized by database

```
GRANT USAGE, SELECT, READ_METADATA ON DATABASE hr TO `HR`;
```

Alternative, grant access on specific tables:

```
GRANT USAGE ON DATABASE hr TO `HR`;
GRANT SELECT, READ_METADATA ON TABLE employees TO `HR`;
GRANT SELECT, READ_METADATA ON TABLE addresses TO `HR`;
```

Enable Secure Data Sharing

Assumptions

- Teams/depts need a private area to collaborate in
- Datasets must not be shared outside team/dept
- New tables are not automatically shared to other members

```
GRANT USAGE, CREATE ON DATABASE project_data TO `Data Analysts`;
```

Alternatively, members automatically see all new tables:

```
GRANT USAGE, SELECT, READ_METADATA, CREATE ON DATABASE project_data TO `Data Analysts`;
```

Enable Private User Sandboxes

Assumptions

- Users need a private area to store derivative datasets
- Datasets must not be shared with other users

```
GRANT USAGE, CREATE ON DATABASE user1 TO
`user1@databricks.com`;
```

Delegate Administration of Access Policies

Assumptions

- Need to delegate management of ACLs to data stewards/owners
- Monitor using SQL Analytics query logs and/or workspace audit logs

```
ALTER DATABASE hr OWNER TO `HR Admins`;
```

Delegate Database Creation to Trusted Users

Assumptions

- Delegate database creation to trusted users
- Database creators manage ACLs
- Monitor grants using SQL Analytics query logs and/or workspace audit logs

```
GRANT CREATE ON CATALOG TO `Data Admins`;
```

Give All Users Read-Only Access To All Datasets

Assumptions

- Simple security model where all users have same level of access
- All new datasets are automatically accessible

```
GRANT USAGE, SELECT, READ_METADATA ON CATALOG TO `users`;
```

Column Masking/Redaction

Assumptions

- Need to redact fields based on user's identity
- Do not give access to underlying table, only view

```
CREATE OR REPLACE VIEW employee_vw
AS
SELECT
  username,
  firstname,
  lastname,
  CASE WHEN
    is_member('HR') OR current_user = username THEN street_address
    ELSE '*******'
  END AS street_address, -- HR can see all addresses, users can see their own address!
  department
FROM employees;

GRANT SELECT ON employees_vw TO users;
```

Row Filtering

Assumptions

- Need to filter data based on user's group membership
- Data includes column matching group name (e.g. "HR", "Sales", etc.)
- Do not give access to underlying table, only view

```
CREATE OR REPLACE VIEW employee_vw
AS
SELECT * FROM
FROM employees
WHERE
  is_member('HR') OR is_member(department); -- HR see all records, users only see records in own department

GRANT SELECT ON employee_vw TO users;
```

Rollout Plan

- SCIM provisioning with groups
- Identify & classify initial set of datasets
- Register in new workspace and apply ACLs
- For Fine-Grained
  - Define access policies (e.g. “Sales only see their own accounts…”, “Support technicians can only see accounts within their region”, etc.)
  - Identify missing attributes needed for dynamic views
- Lifecycle policy for derivative datasets
- Define process to “promote” new or derivative datasets
  - Need to classify data & define access policies
  - Legal & compliance review

Notebook: Deidentified PII Access

Notebook: Propagating Deletes with Change Data Feed

Notebook: Deleting at Partition Boundaries


databricks