Unity Catalog Architecture Patterns

Bernhard Walter
2023-09-28

Table of Contents

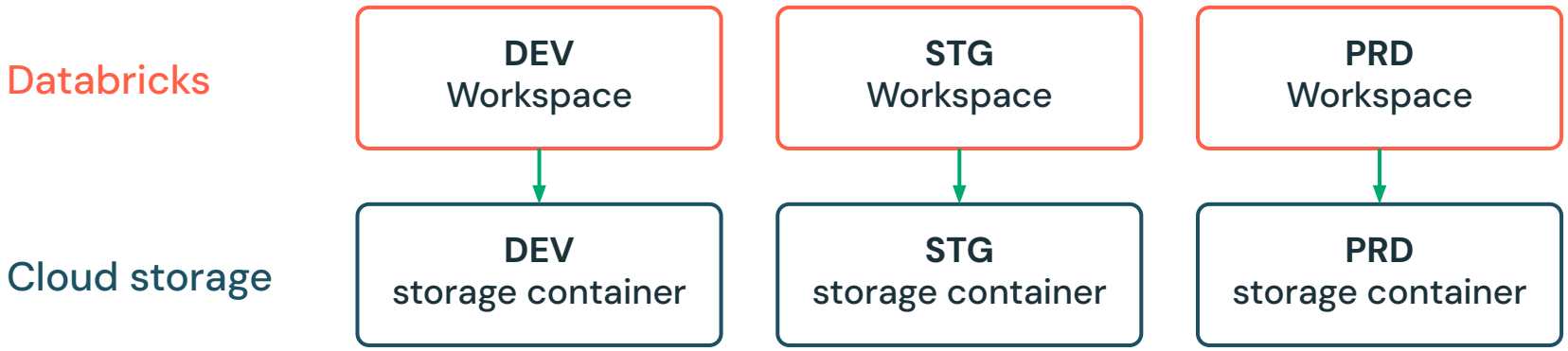
Patterns

Pattern 1:

Software Development Life Cycle (SDLC) setup

Scenario: Software Development Lifecycle

- Customers use different environments for the different stages of development. The minimal setup is to have two environments “Development & Test” and “Production”
- Most customers also split “Development & Test” into two or more environments: “Development” and quality assurance environments being called “Test”, “QA”, “Staging”, “Integration”, …
- This scenario describes the recommended “DEV” - “STG” - “PRD” setup but the approach will also work for more or less environments depending on customer’s setup
- Many enterprise customers also isolate these environments from a storage and compute perspective using different storage containers, VNets/VPCs and Databricks Workspaces



Unity Catalog Isolation Options (SDLC)

Delegation of Management (admin isolation)

Each SDLC environment has its own admin

Workspace to catalog binding

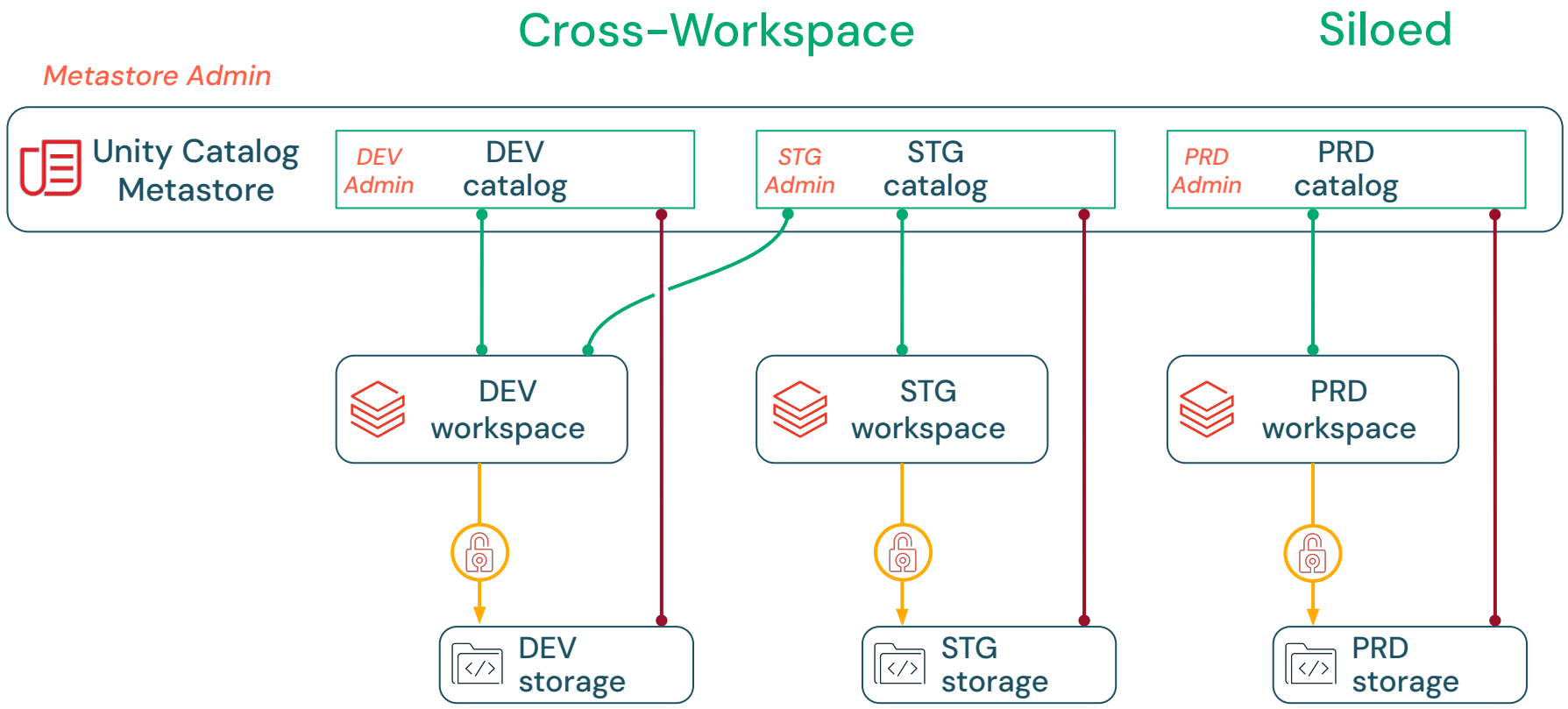
The PRD workspace is fully isolated and has its own catalog. DEV workspace has access to DEV and STG catalog

Storage isolation

This example separates the storage locations on catalog level (typically sufficient)

UC Access Control

Users should only gain access to data/ metadata based on agreed access rules

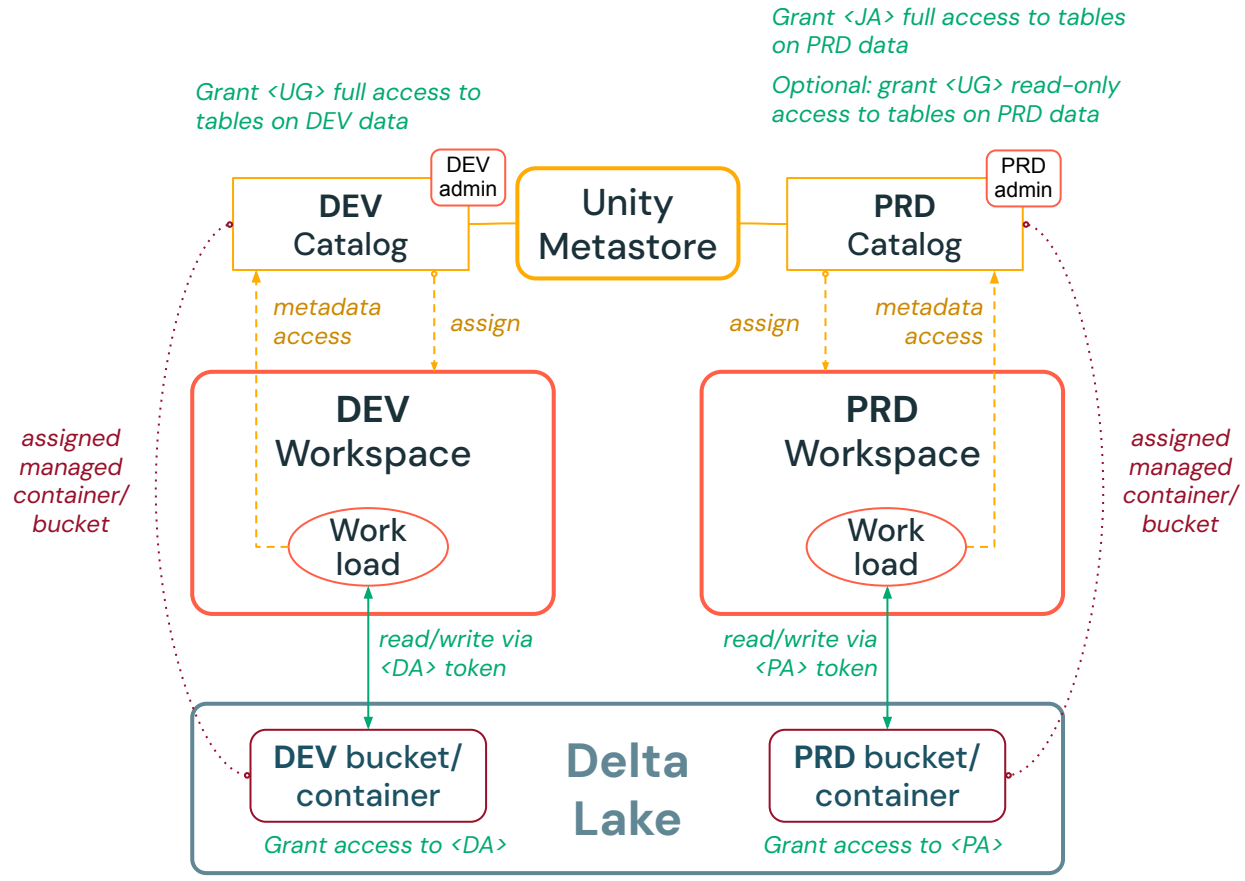


Software Development Lifecycle setup w/ UC

Approach *

- There will only be one Metastore per cloud region, which will be used for all related SDLC workspaces (DEV, STG, PRD, ...)
  - Isolate the environments on the Catalog level of the 3-level namespace of Unity Catalog. Assign DEV, STG, PRD workspaces to their respective catalog only.
  - Isolate the DEV, STG, PRD data locations by assigning dedicated managed buckets/containers to the catalogs
  - Isolate admin scope by delegating administration of the catalogs to different admins for DEV, STG, PRD
- Catalog names can be combinations of SDLC and business / organizational unit names, e.g. *sales_dev*, *sales_prd*, *engineering_dev*
- Access to workspaces, clusters and endpoints needs to be configured accordingly

As a best practice, use at minimum 2 workspaces (others/prod), but ideally 3 workspaces (dev/stg/prod) *



<DA> DEV System Account for UC
<PA> PRD System Account for UC
<JA> System Account to execute jobs in PRD
<UG> User Group working in DEV and Prod

* To simplify the diagram, only DEV and PRD are shown

Pattern 2:

Cross cloud region data access

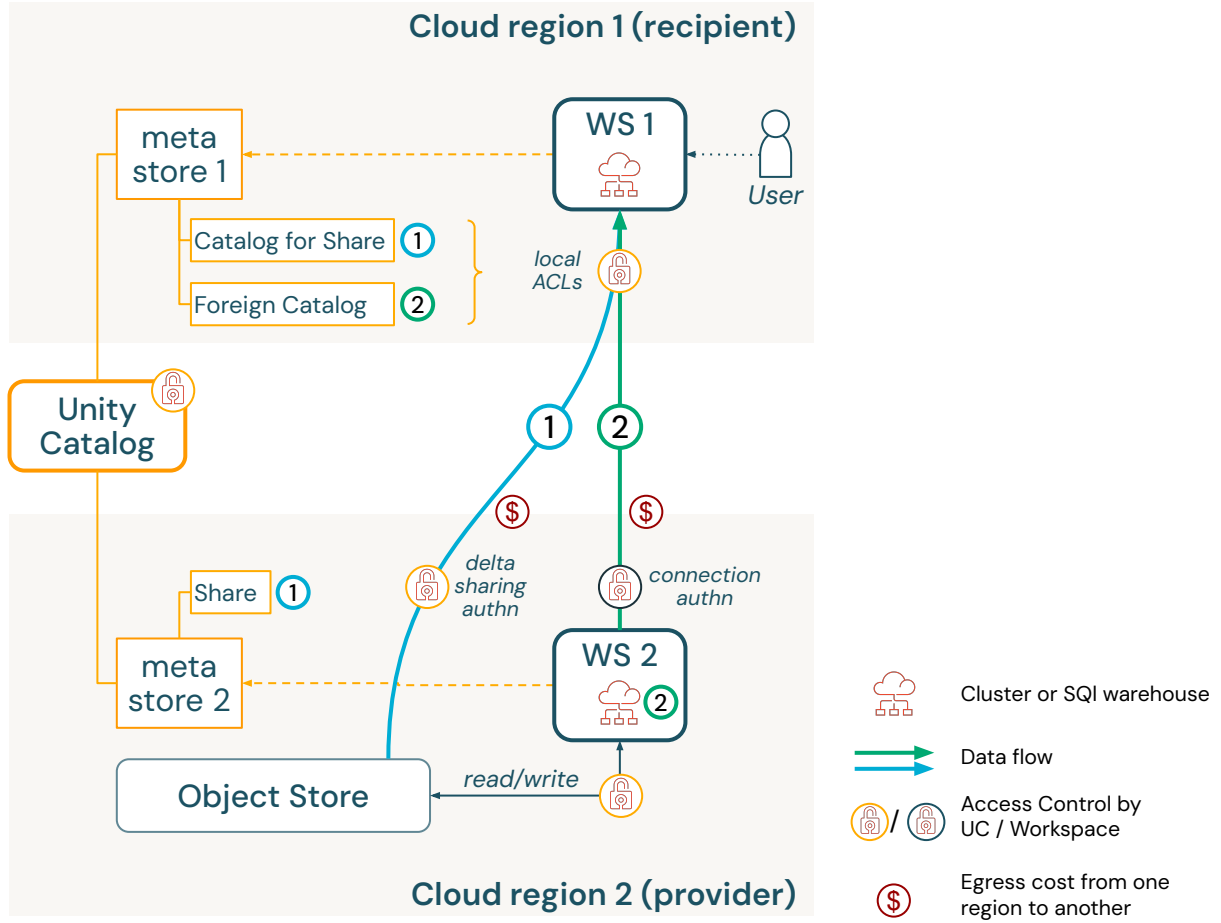
Cross region access – UC integrated

1. **Databricks-to-Databricks Delta Sharing** (provider provides data)
   - Region 2 will share tables with the metastore 1 of region 1
   - Region 1 will create catalogs in metastore 1 to hold the tables from the share of region 2
   - Users in region 1 can then discover and access the shared data from region 2 via metastore 1
2. **Lakehouse Federation** (provider provides compute)
   - Region 1 will connect to Databricks in region 2 represented in metastore 1 by a Foreign Catalog
   - Users in region 1 can then access the shared data from region 2 via metastore 1

In both cases

- region 1 can additionally control access for their users on top in metastore 1
- egress costs arise from data leaving region 2 (region 1 reading)

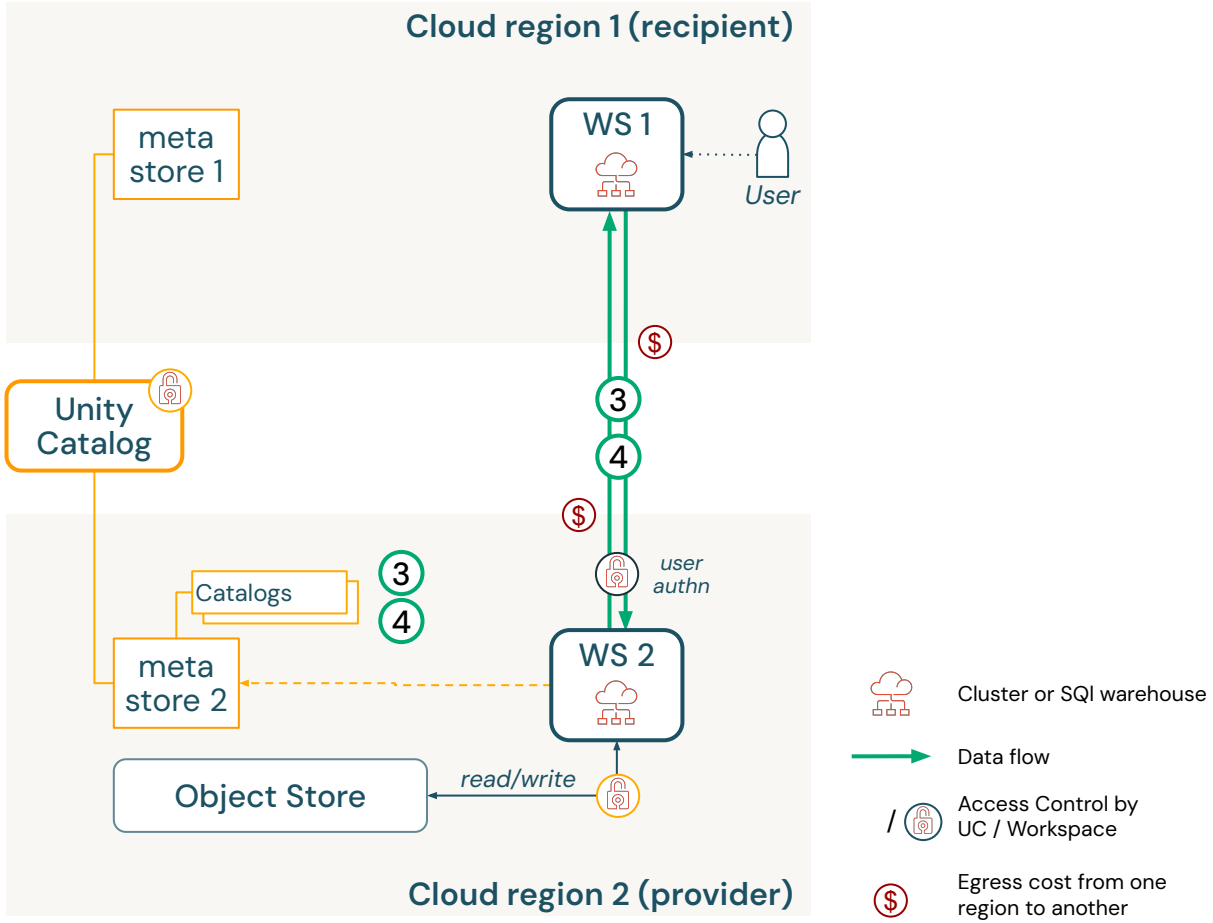
Note: We strongly do not recommend registering common tables as External Tables in more than one metastore



Cross region access - Remote compute

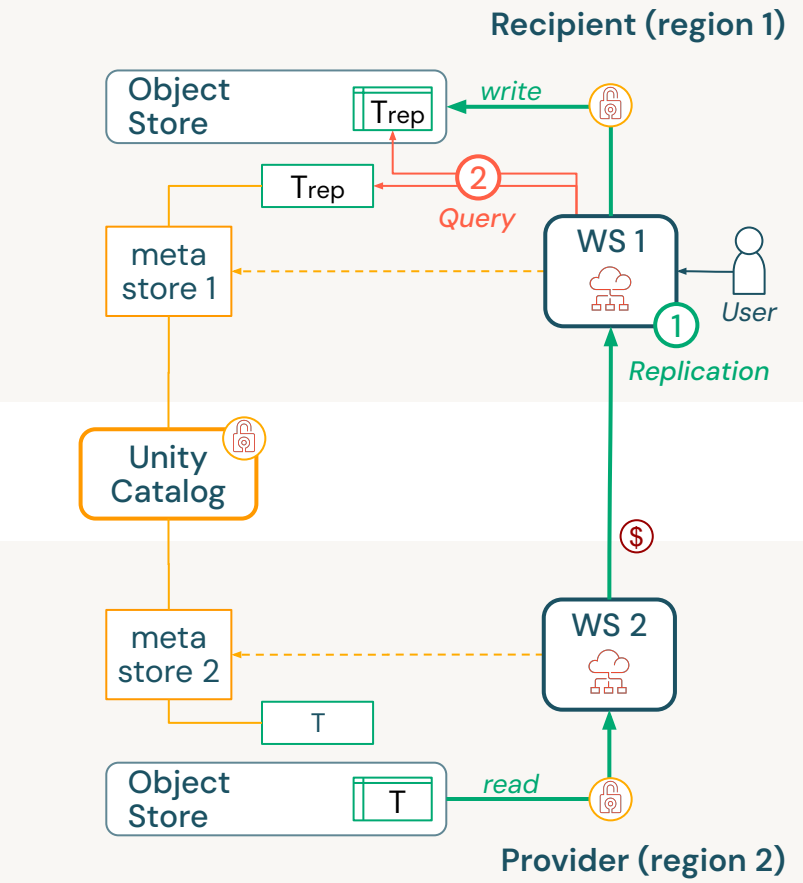
③ **JDBC / ODBC** (provider provides compute) or
④ **Databricks SQL Connector for Python** (provider provides compute)

- Users in region 1 will connect to specific compute resources (clusters or SQL Warehouses) in region 2
- Queries will be executed on the compute resource in Workspace 2
- No integration into metastore of region 1
- egress costs arise from data leaving region 2 (region 1 reading) or data leaving region 1 (region 1 writing)



Managed replication

Relevant in cases when egress costs get too high



Approach:

1. Replicate data in intervals from provider to recipient (driven by the recipient in region 1)
2. Query locally

Replication Types:

- Managed full replication (only reasonable for small data)
- Managed incremental replication (check the used technology whether incremental replication is supported)

Impact:

- Data freshness depends on refresh interval time (business impact)
- Higher maintenance effort for managing the replication pipeline
- Egress cost impact
  - Full replication: Full egress cost each time it will be replicated
  - Incremental replication: One time cost for the whole data set

Pattern 3:

BI for multiple Business Units

Five core processes

To be taken into account when defining the architecture patterns

Data Production

Creation of data via Ingest & ETL processes or by different business oriented teams.

What is a meaningful data set?

Data Publishing

A deliberate step to provide access to a data set for other consumers.

Where to publish which data set?

Data Consumption

Use own data or published data by others for analysis, reporting, ML, ...

Generate value from data

Data Governance

Metadata management and access control.

Whom to provide which access at which point of time?

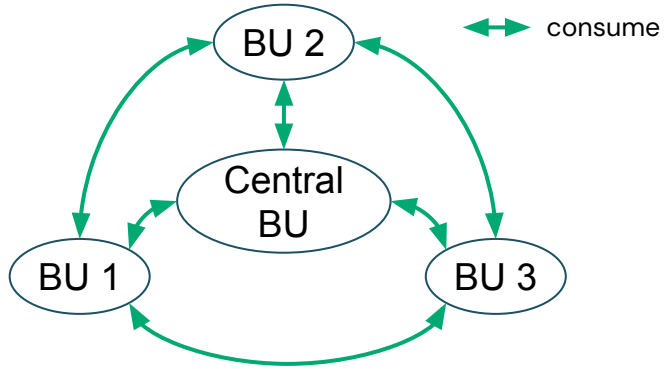
Platform Operations

Data teams will work on one or more platforms. Depending on the enterprise, data teams can operate their own platform, or there is a central team that comes up with a blueprint for the setup.

Distributed vs Centralized Publishing

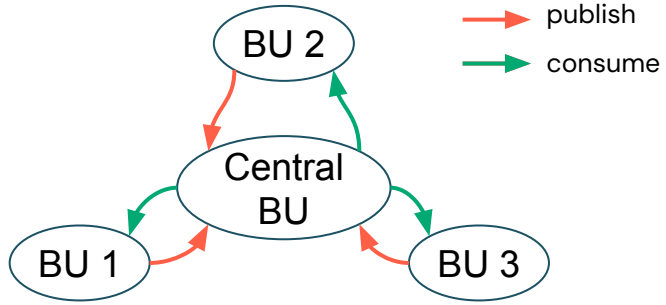
Distributed Publishing

- BUs share the data they created themselves
- Each BU is responsible for data quality (treat published data as a “data product”) and access permissions
- BUs are not isolated from an infrastructure perspective (e.g. no network restrictions)
- Usually there is still a central team to provide platform operations. It is responsible to define secure and compliant blueprints, setup infrastructure, etc. This team also provides capabilities to ensure private data can be protected (e.g. by managing access from a network perspective). However, it does not play a role for publishing and configuring data access.
- The Central BU can also publish common data to other BUs



Centralized Publishing

- A central BU serves as a Hub
- BUs are isolated from each other, e.g. by network access restrictions
- The central BU provides
  - central data storage and the central metastore to publish (data, metadata & ACLs) datasets for all BUs and usually
  - central data governance and applies quality assurance on all published data
  - platform operations (secure and compliant blueprints, infrastructure setups, …)
- Often the central BU publishes own data (via a central data engineering team)

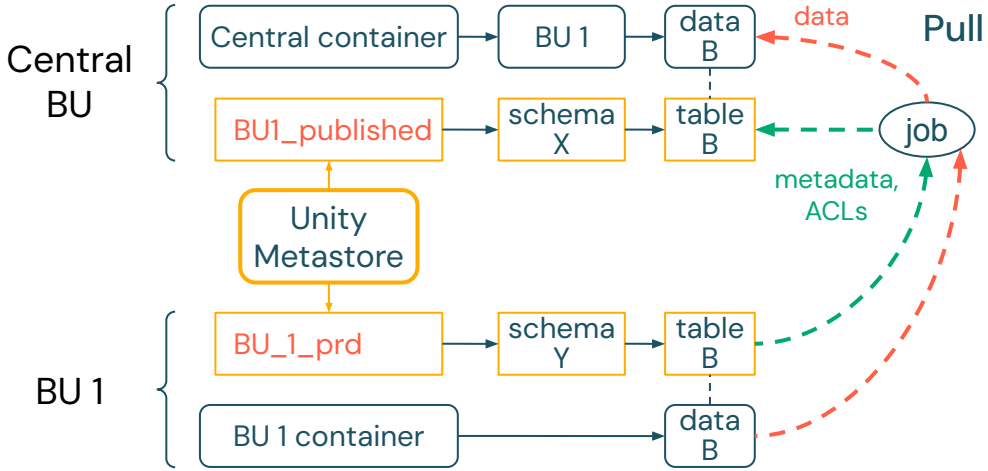


Flavors of Centralized Publishing

Strictly governed (PULL)

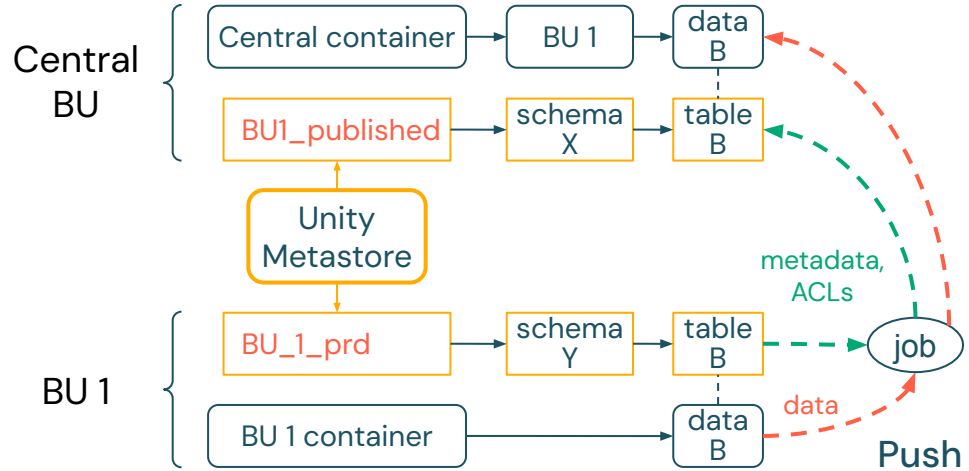
- BU notifies Central BU that a table should be published
- Central BU validates data quality and compliance of metadata to the defined rules
- Central BU copies data from BU container to the BU folder in the central container
- Central BU creates schema in central UC catalog
- Central BU sets up a job to regularly update the centrally published data (batch or streaming)

Note: This pattern also applies to scenarios where BUs are not skilled enough to properly publish data, i.e. central team providing a publishing service instead of primarily applying governance



Loosely governed (PUSH)

- BU copies data to BU folder in the central container (each BU does only have access to its folder)
- BU creates schema in its central catalog for published data (each BU does only have access to its catalog)
- BU defines ACLs on the new table
- BU sets up a job to regularly update the central data (batch or streaming)
- Central team mainly provides publishing infrastructure and applies lightweight data governance (monitoring data quality, validating data is kept up to date, ...)



Pattern 3.1:

BI for multiple Business Units Single cloud-region

Scenario: Single cloud-regions BI

Organization & roles:

Central BU	- Provides blueprints for BU platform setups - Provides central platform management - Provides data centrally (global and possibly BU specific data)
BU 1:	- Works with local data combined with central data - Can provide created data sets to other BUs (publish data)
BU 2:	. . .
. . .	

Cloud setup: All BUs in one cloud-region

Use Case: BI use case focussed on Reporting and Analysis via BI Tools

High level blueprint

Single cloud-region

Central Business Unit (Central BU)

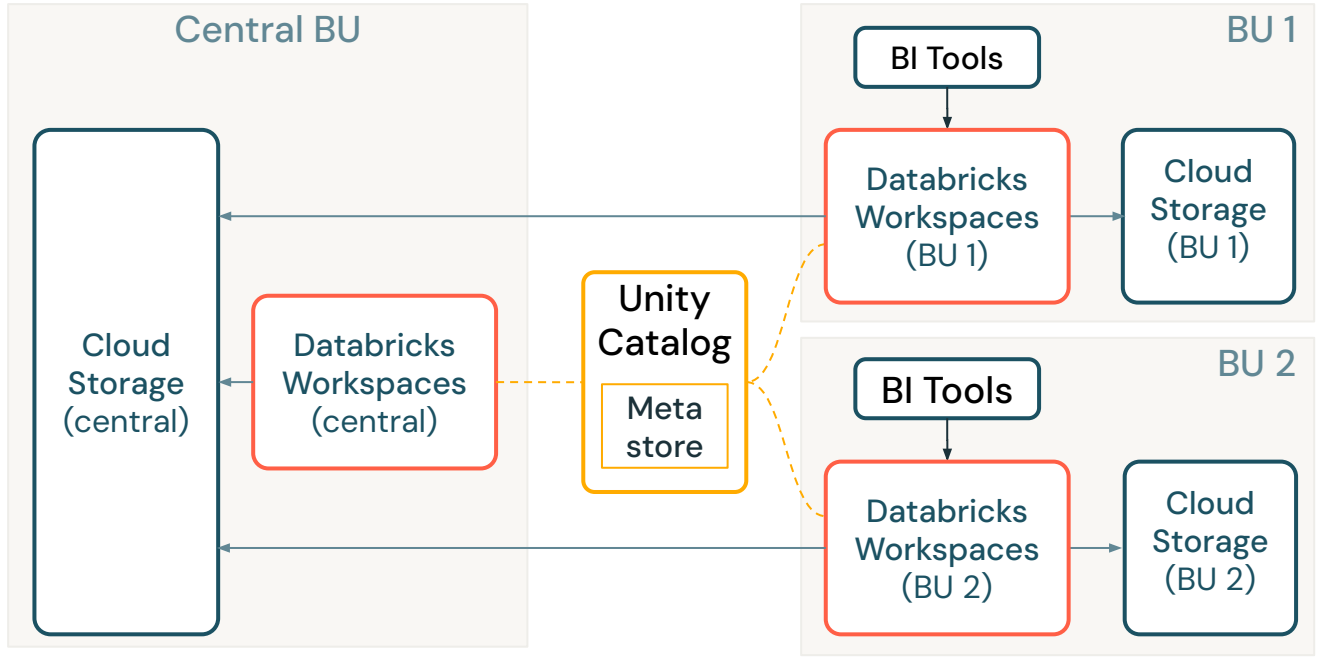
- Maintains central data pipelines
- Provides access to central data for each BU

Other Business Units (BU)

- Use own cloud storage for BU data
- Mainly use BI tools and Databricks SQL to work with own data and centrally provided data
- Can share data on a central cloud storage or their own storage - always governed by Unity Catalog (details see next slides)

Unity Catalog

- Since this setup is in one cloud region, there is a single metastore covering all BUs (including the Central BU)
- Consider Unity Catalog's workspace binding, and storage and admin isolation features when setting up central BU and other BUs.



Option 1: Distributed Publishing

Single cloud-region

Data Production

- Central Ingest & ETL by the central BU
- Other BUs create (business) data sets

Data Publishing

- Central BU publishes data to the central PRD storage and into the catalog of the Central BU in UC
- BUs publish to their PRD storage and into BU catalogs of UC (see metadata mgmt later)

Data Governance (distributed)

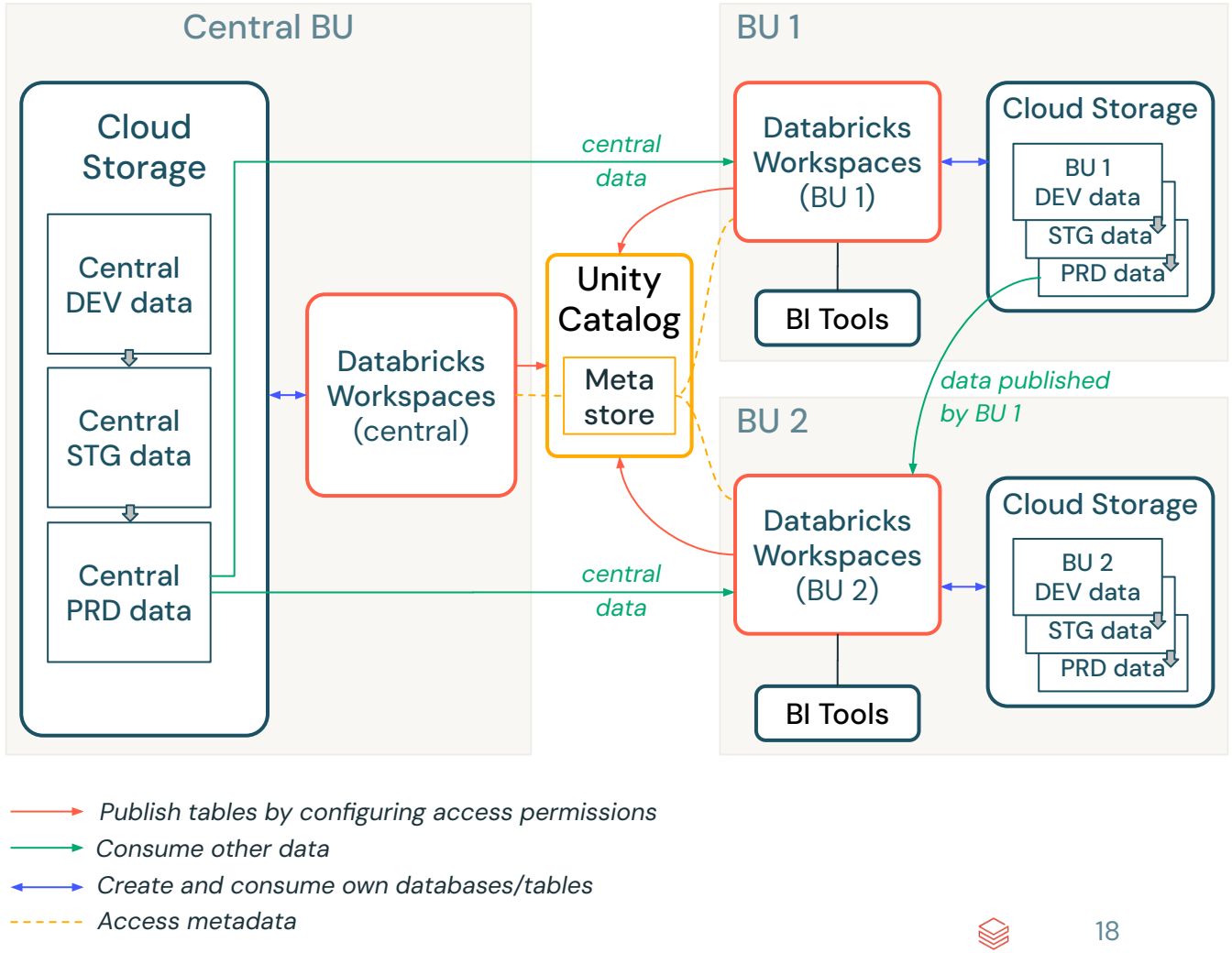
- Central team and each BU can work independently on their catalogs (publishing and setting ACLs)

Data Consumption

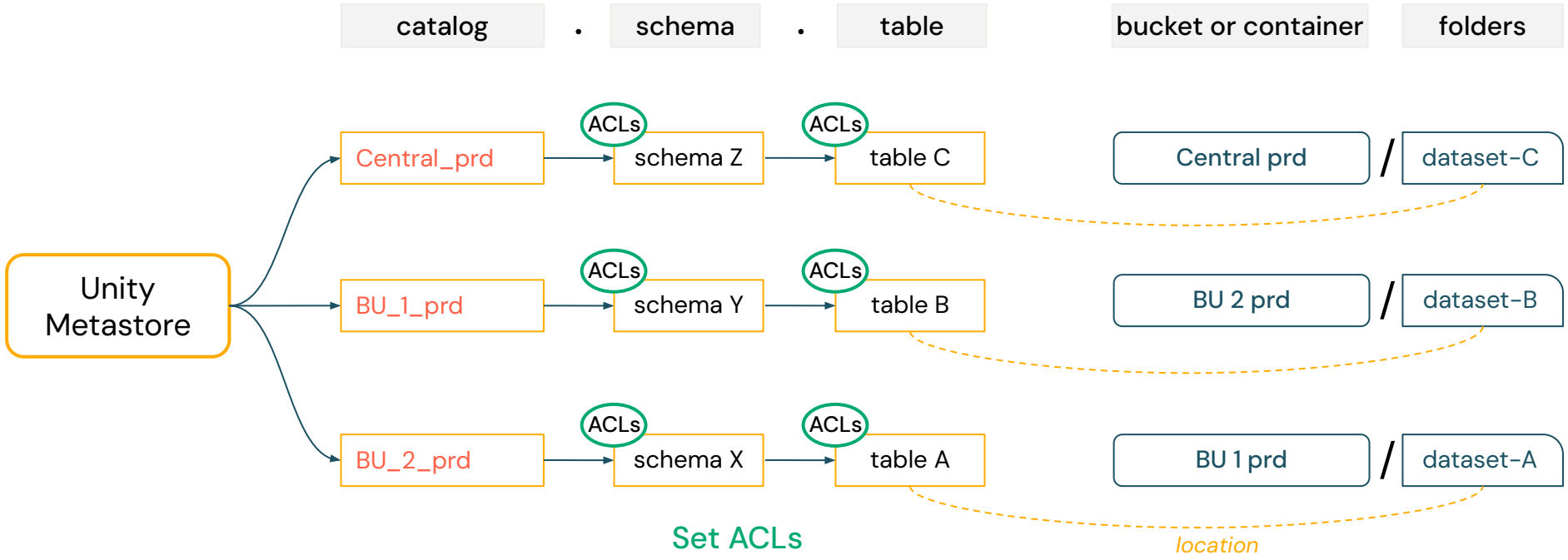
- Published data will be discovered in the metastore and consumed from PRD storage of the BUs (central or other)

Platform operations

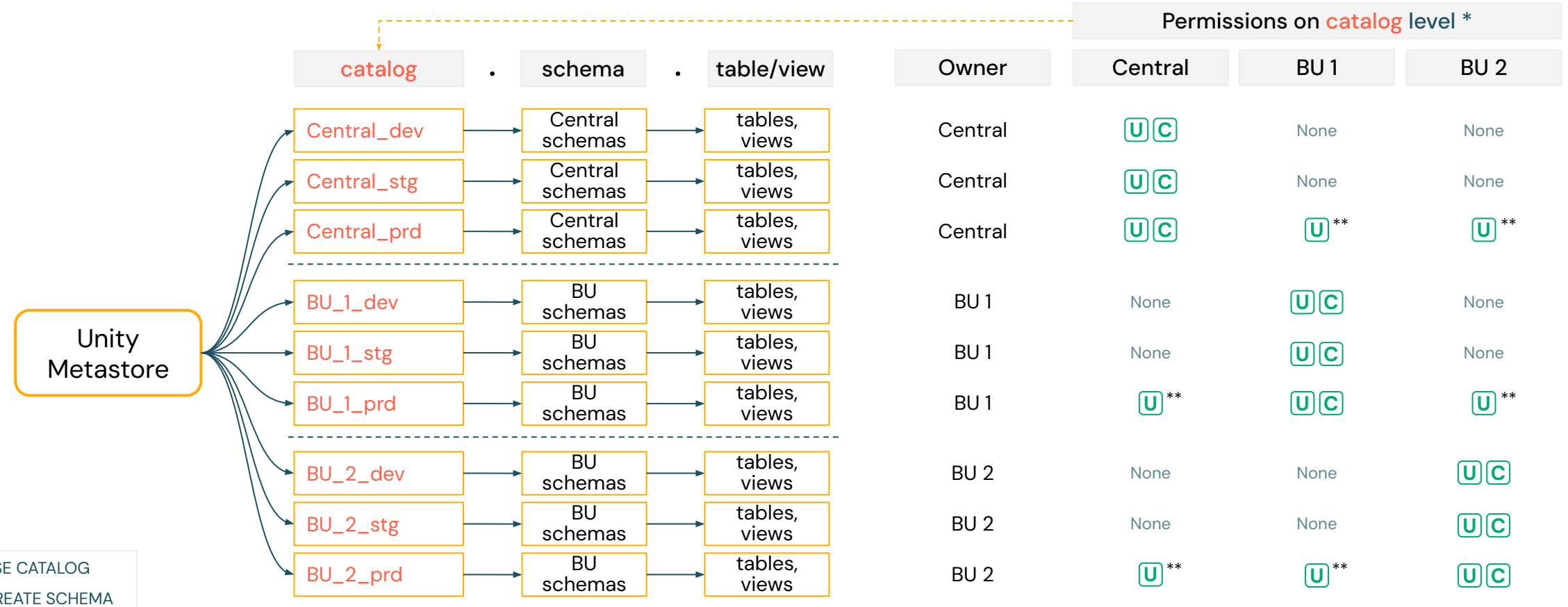
- Central team provides platform blueprints and creates environments for BUs (automated)



Option 1: Distributed Publishing Process



Option 1: Metadata Management



* To access tables, USE SCHEMA and SELECT also need to be provided on schema and table level
** Catalogs where other BUs can be given access to schemas and tables/views by Central team

Option 2: Centralized Publishing

Single cloud-region

Data Production

- Central Ingest & ETL by the central BU
- Other BUs create (business) data sets

Data Publishing

- Central BU publishes data in central PRD storage and into the PRD catalog of the Central BU in UC
- BUs requests from Central team to publish from their PRD storage to central PRD storage and into the BU catalog in UC that is maintained by Central

Data Governance (centralized)

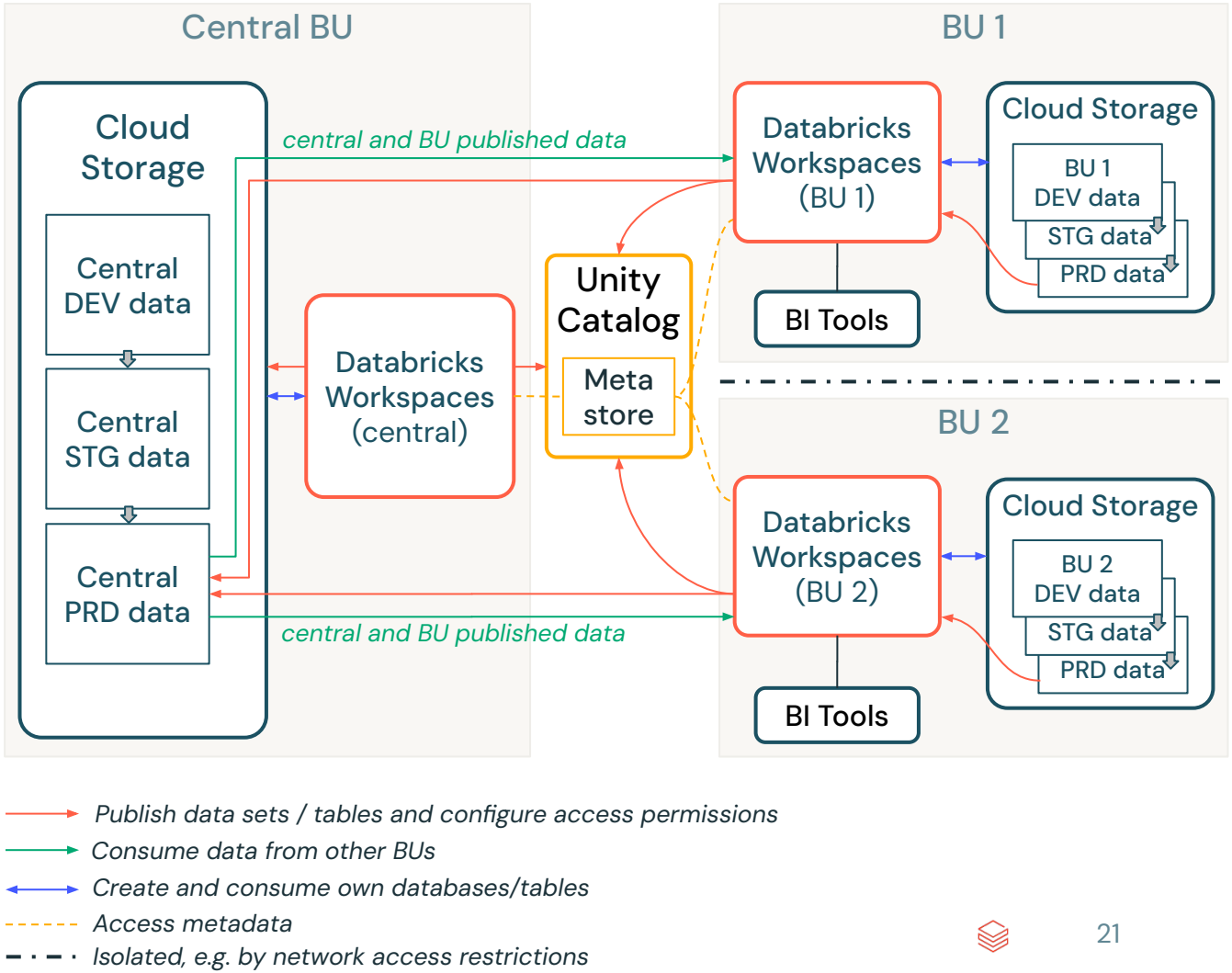
- Central team and each BU (for non published) data can work independently on their catalogs
- Central team applies additional quality assurance and maintains ACLs in the central BU catalog

Data Consumption

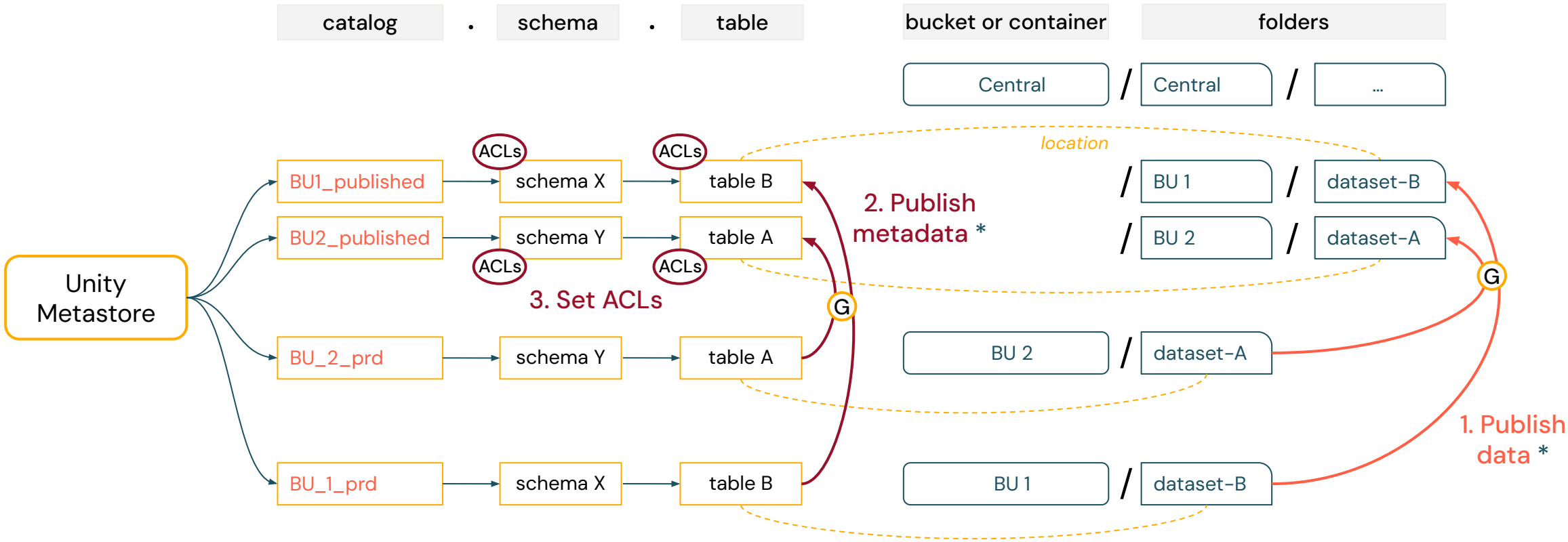
- Published data will be discovered in the Central catalog and consumed from the central PRD storage

Platform operations

- Central team provides platform blueprints, creates environments for BUs (automated)
- Central team could provide common data services



Option 2: Centralized Publishing Process

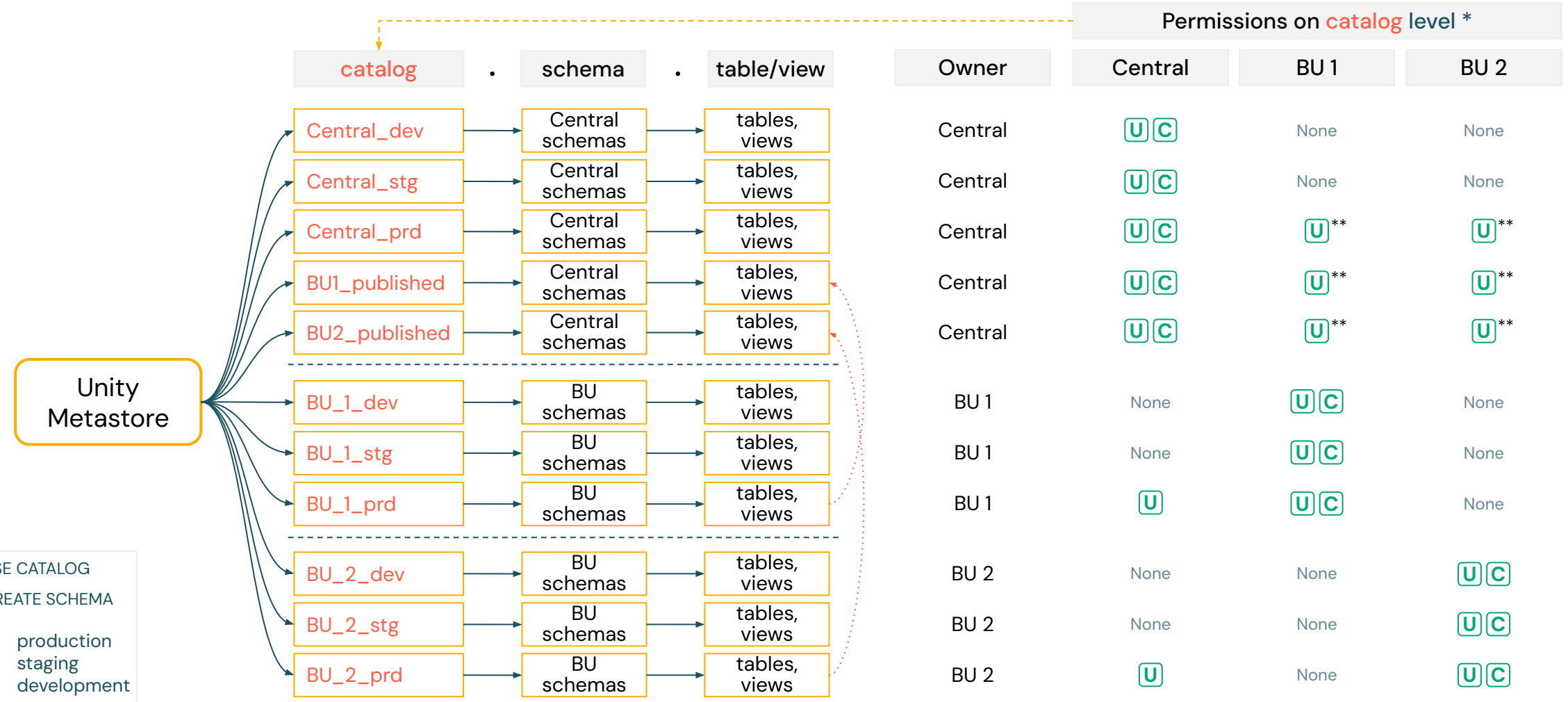


G Central data governance:

- Control data quality and compliance with publishing rules (naming conventions, ...)
- Control data access (set ACLs)

* For details see push/pull discussion earlier

Option 2: Metadata Management



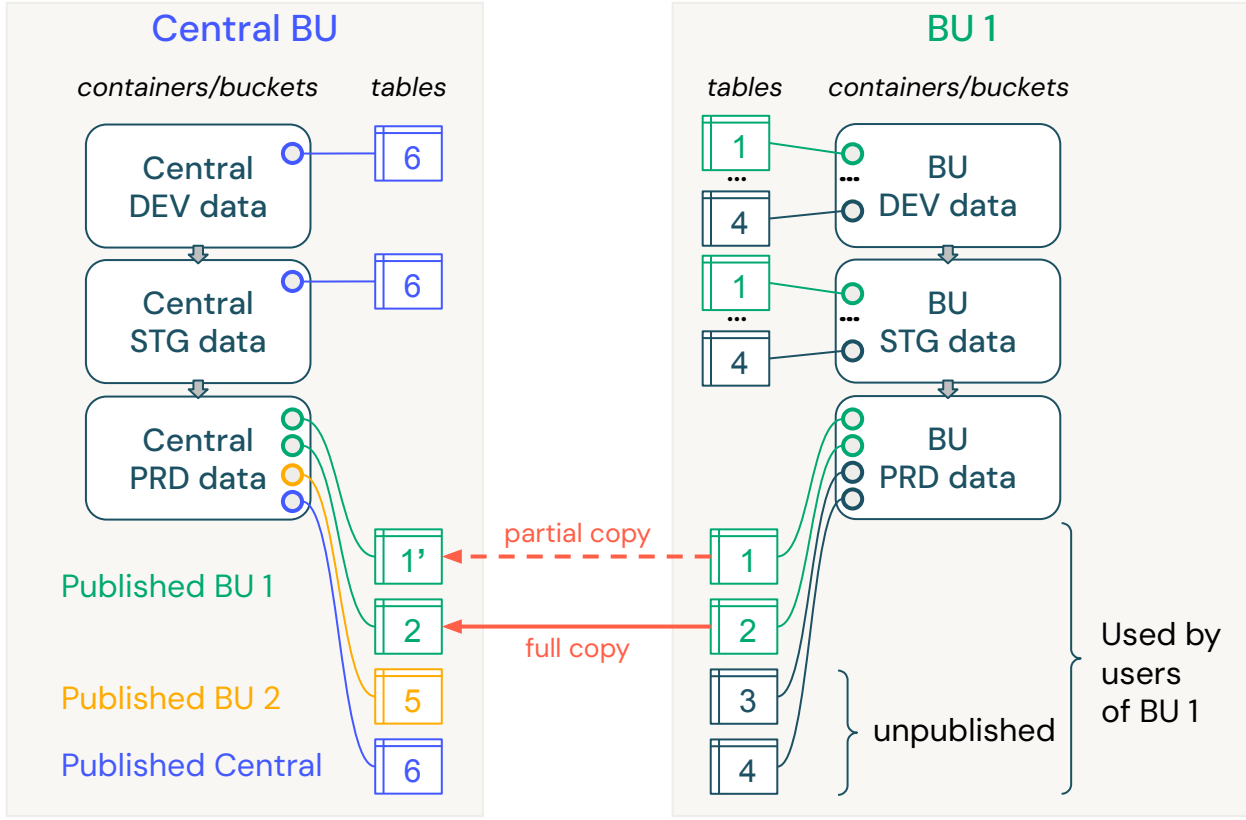
* To access tables, USE SCHEMA and SELECT also need to be provided on schema and table level
** Catalogs where other BUs can be given access to schemas and tables/views by Central team

Granularity of Centralized Publishing

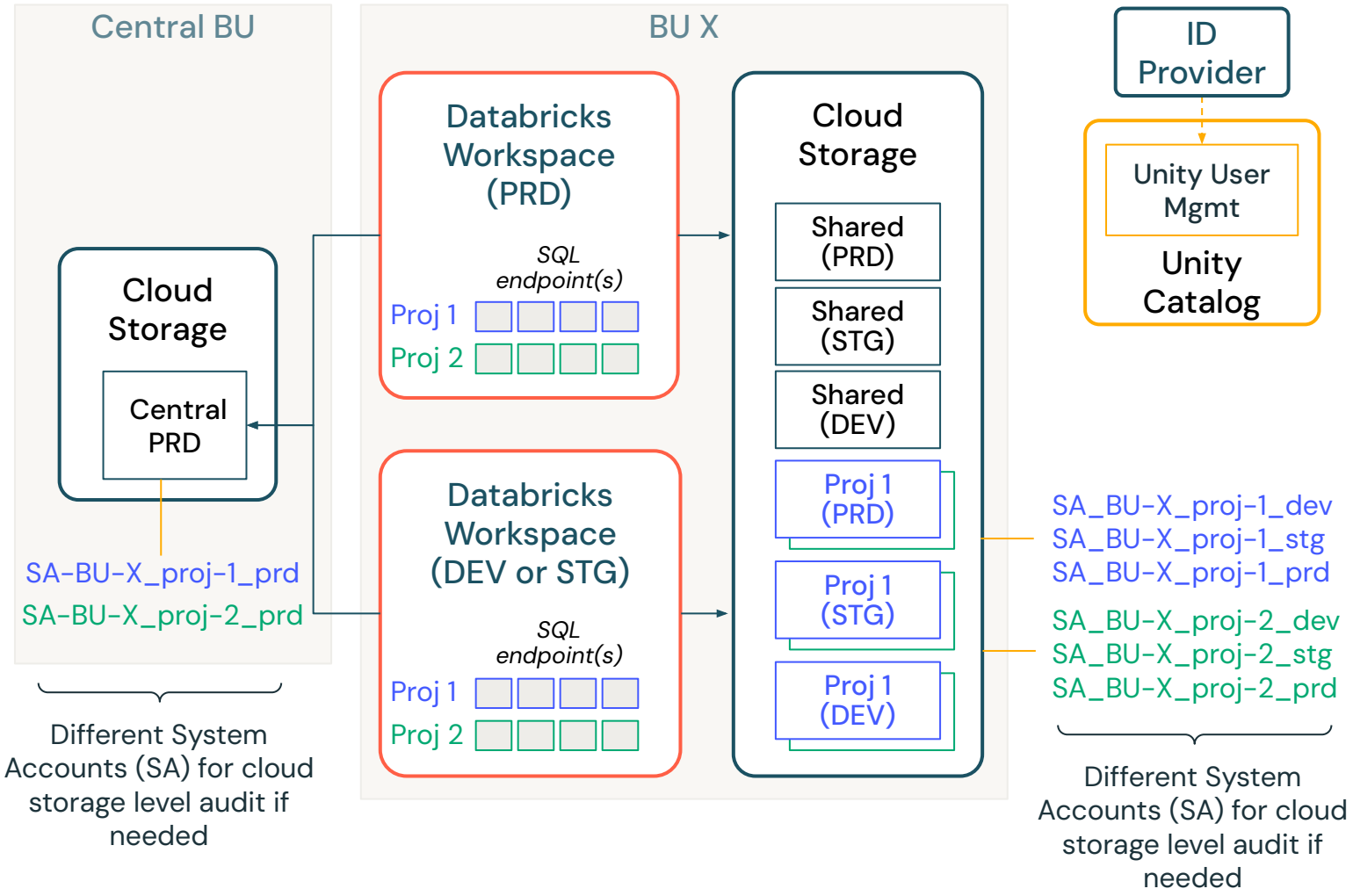
Centralized Publishing does not mean that all production data from a participating BU will be copied over to the central storage location:

- In the centralized publishing model, BUs will still maintain their own Dev/Stg/Prd environments
- BUs initially create production tables for internal consumption
- Some of these tables might be relevant for other BUs, hence worth publishing
  - as a full copy of the original
  - as a partial copy (filtered columns or rows, masking, …)

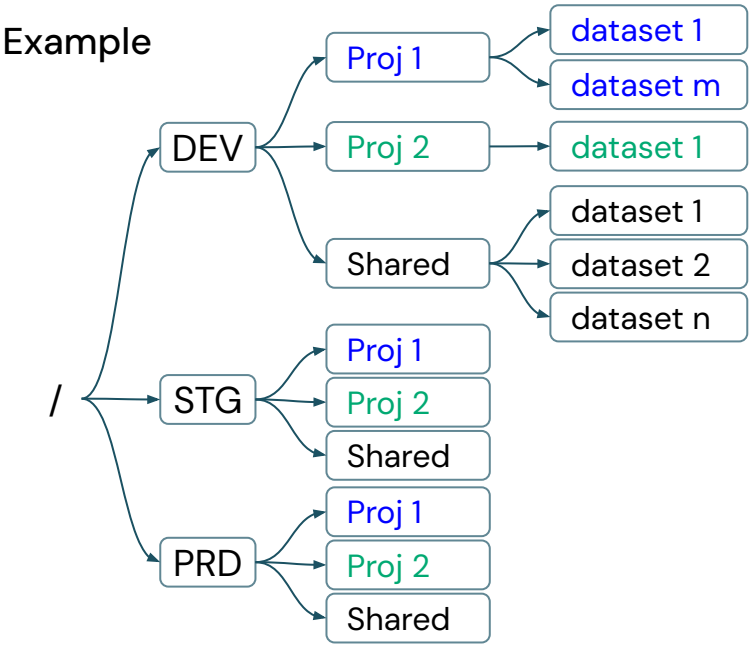
  Note: Choose push / pull publishing according to customer requirements



System Account Mappings



Example



Data Access Pattern

- Read/write
  - 'SA_BU-X_proj-y_dev' to '/DEV/Proj y'
  - 'SA_BU-X_proj-y_stg' to '/STG/Proj y'
  - 'SA_BU-X_proj-y_prd' to '/PRD/Proj y'
- Either read/write or read only
  - 'SA_BU-X_proj-y_dev' to '/DEV/Shared'
  - ...

Pattern 3.2:

BI for multiple Business Units Multiple cloud-regions

Scenario: Multiple cloud-regions BI

Organization & roles:

Central BU	- Provides blueprints for BU platform setups - Provides central platform management - Provides data centrally (global and BU specific data)
BU 1:	- Works with local data combined with central data - Can provide created data sets to other BUs (publish data)
BU 2:	. . .
. . .	

Cloud setup: BUs are in multiple cloud-regions

Use Case: BI use case focussed on Reporting and Analysis via BI Tools

High level blueprint

Multiple cloud-regions

Central Business Unit (BU)

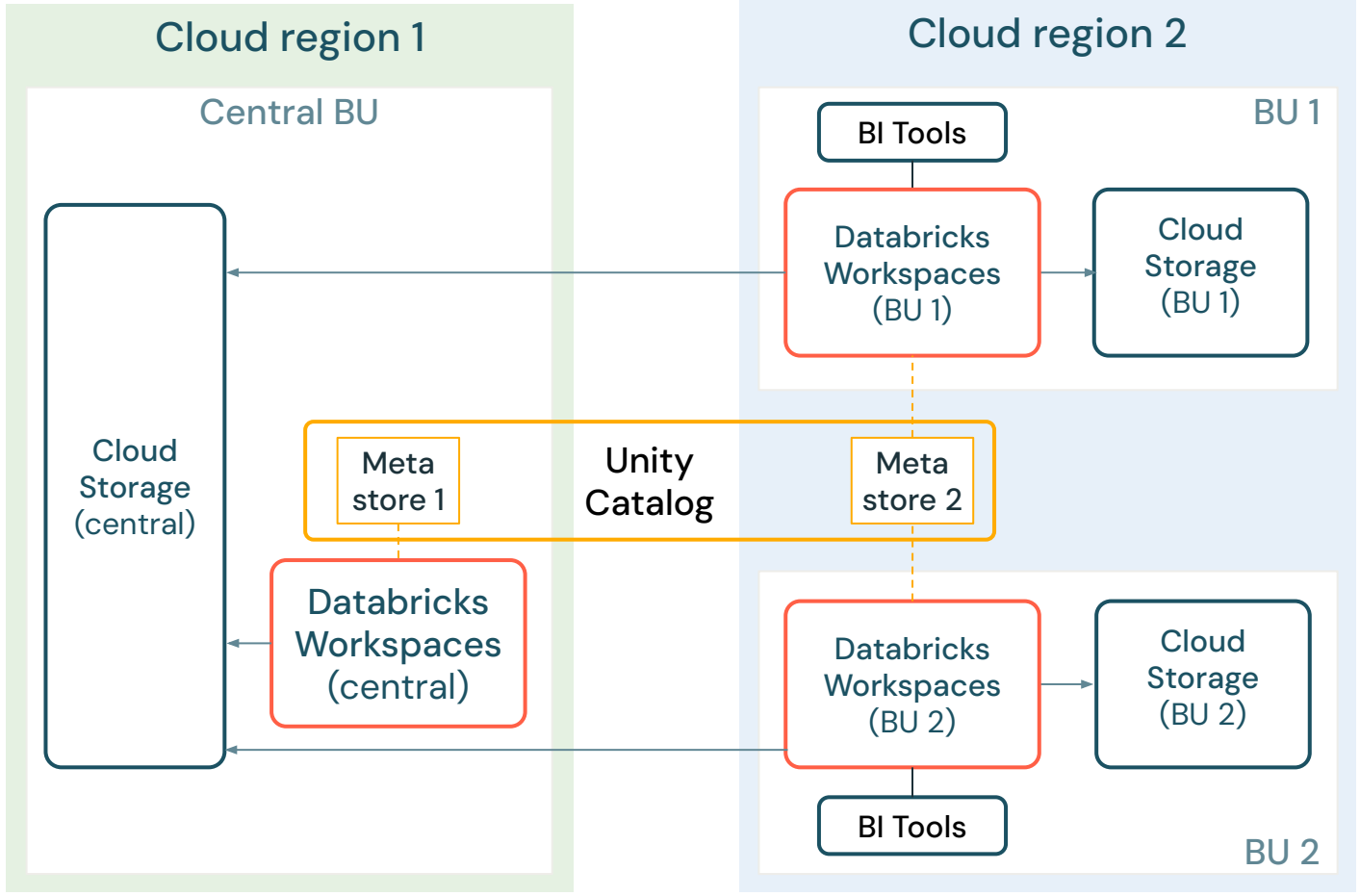
- Maintains central data pipelines
- Provides access to central data for each BU

Other Business Units (BU)

- Use own cloud storage for BU data
- Mainly use BI tools and Databricks SQL to work with own data and centrally provided data
- Can share data on a central cloud storage or their own storage - always governed by Unity Catalog (details see next slides)

Unity Catalog

- Since this setup is in multiple cloud regions, there is one metastore per region.
- Consider Unity Catalog's workspace binding, and storage and admin isolation features when setting up central BU and other BUs.



Option 1: Distributed Publishing

Multiple cloud-regions

Data Production

- Central Ingest & ETL by the central BU
- Other BUs create (business) data sets

Data Publishing

- Central team publishes data to central PRD storage and into UC Metastore 1
- BUs publish to their PRD storage and into their Metastore (here Metastore 2)
- For Delta Sharing, Metastore 2 will be a recipient of Metastore 1 to share central metadata with region 2
- If BUs exist in separate cloud regions and want to share data, Databricks-to-Databricks Sharing needs to be set up for their Metastores (not shown)

Data Governance (distributed)

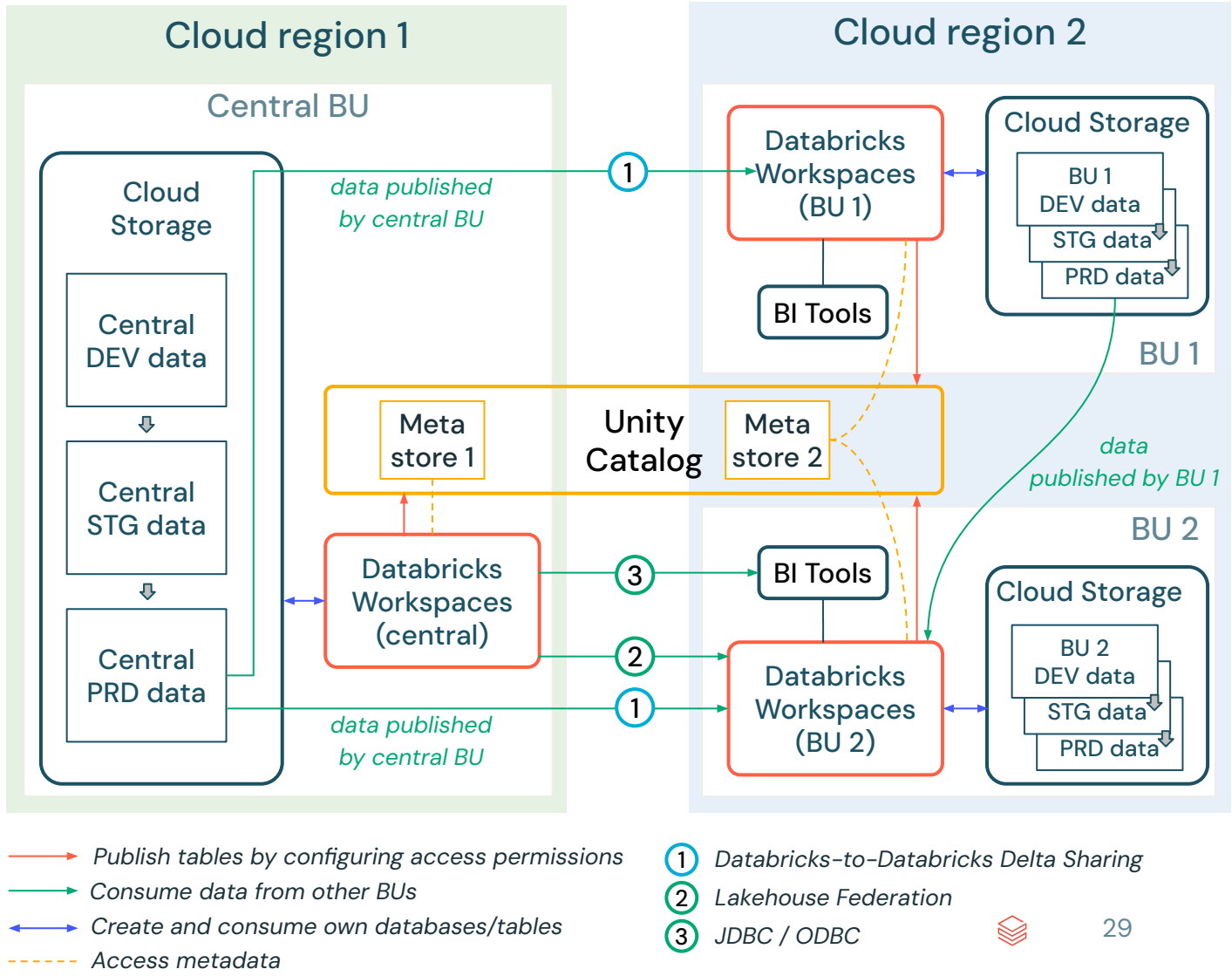
- All BUs set ACLs for their own and delta-shared data

Data Consumption

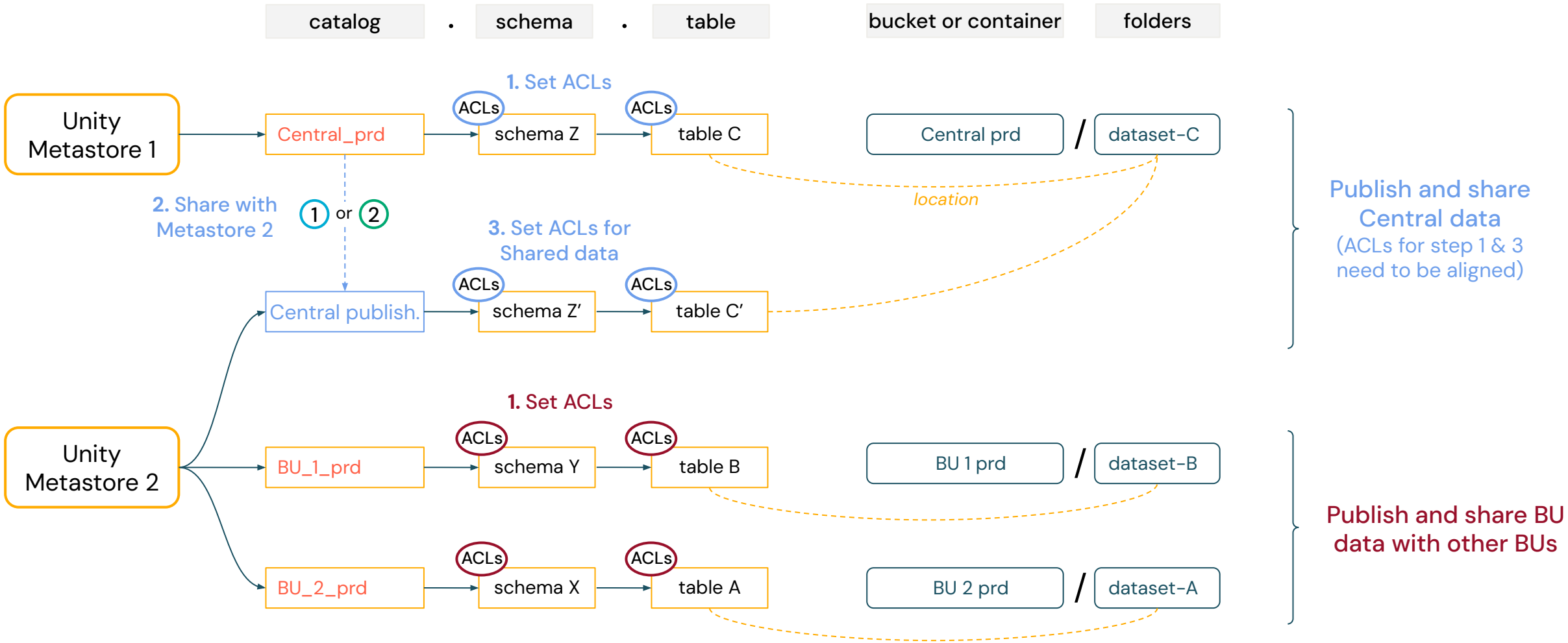
- Published data will be discovered in BUs Metastore and consumed from BUs PRD storage

Platform operations

- Central team to provide platform blueprint and create environments for BUs (automated)

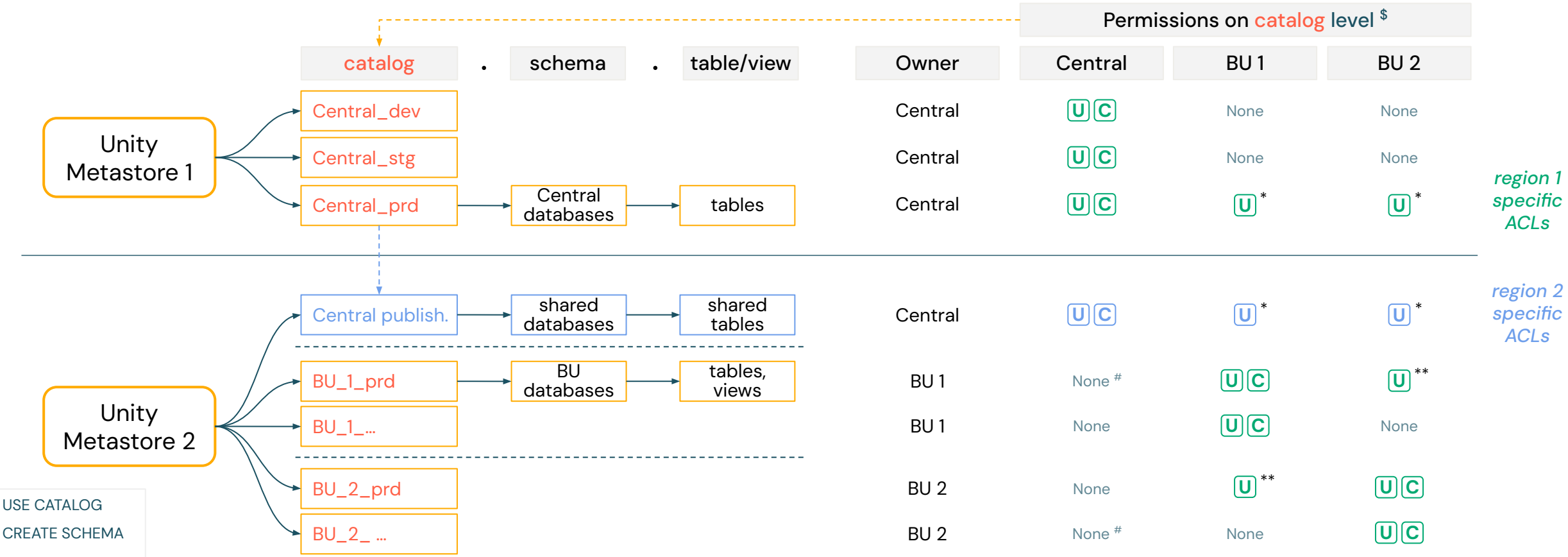


Option 1: Distributed Publishing Process



1 *Databricks-to-Databricks Delta Sharing*
2 *Lakehouse Federation*

Option 1: Metadata Management



$ to access tables, USE SCHEMA and SELECT also need to be provided on schema and table level
* catalog where other BUs can be given access to schemas and tables/views from Central team
** catalogs where other BUs can be given access to schemas and tables/views
# if Central needs access to BU data, Delta Sharing or Lakehouse Federation need to be set up

Option 2: Centralized Publishing

Multiple cloud-regions

Data Production

- Central Ingest & ETL by the central BU
- Other BUs create (business) data sets

Data Publishing

- Central team publishes data in central PRD storage and into the catalog of the Central BU in Metastore 1
- BUs publish by requesting from Central team to publish from their PRD storage to central PRD storage and into the BU catalog in Metastore 1
- Central BU will configure access control on delta-shared data in Metastore 2
- Metastore 2 will be a recipient of Metastore 1 and all published metadata will be shared with Metastore 2

Data Governance (centralized)

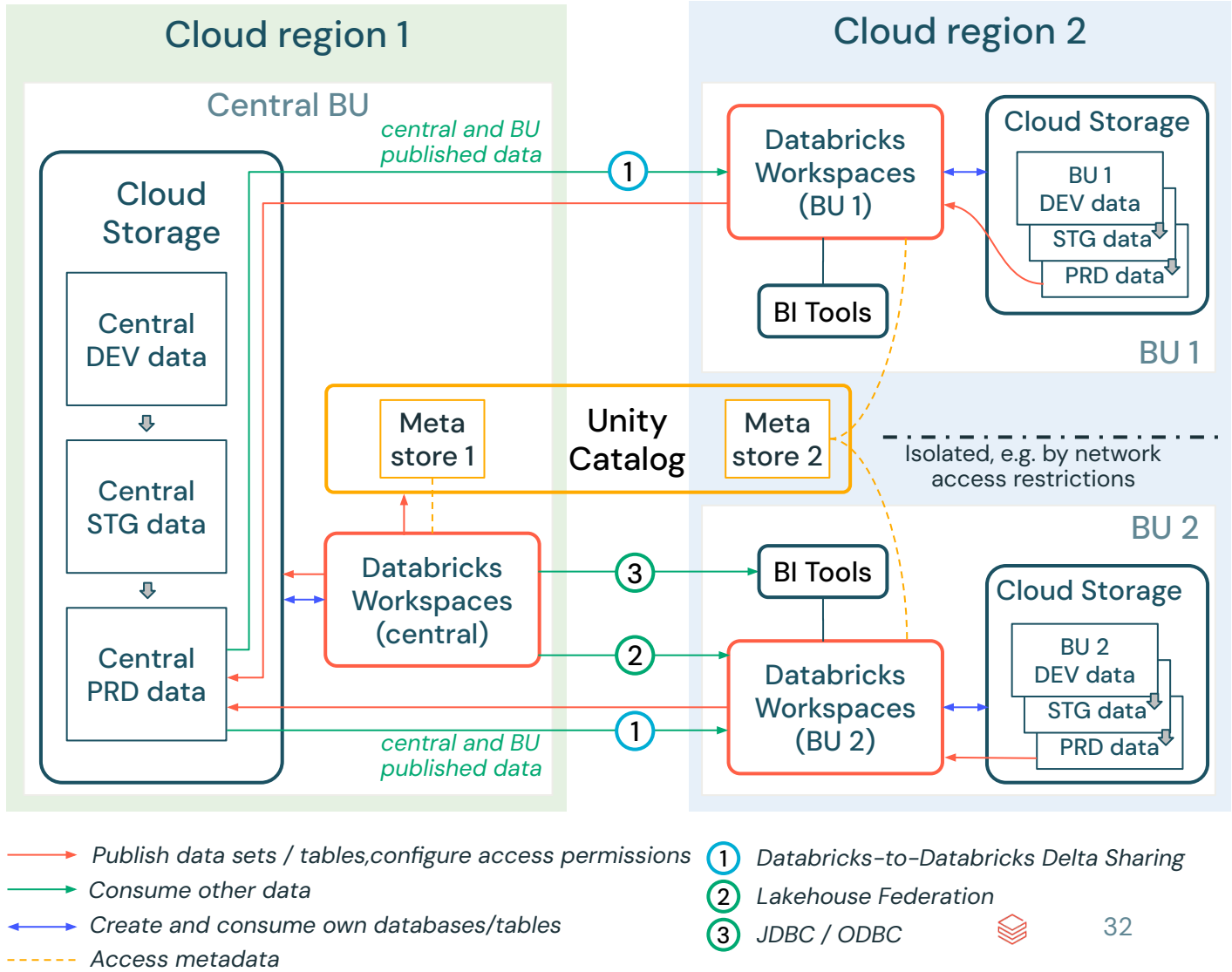
- Access permissions are defined in Unity Catalog. Central team will maintain access control for all published data (with input from BUs)
- Central team might apply additional quality control

Data Consumption

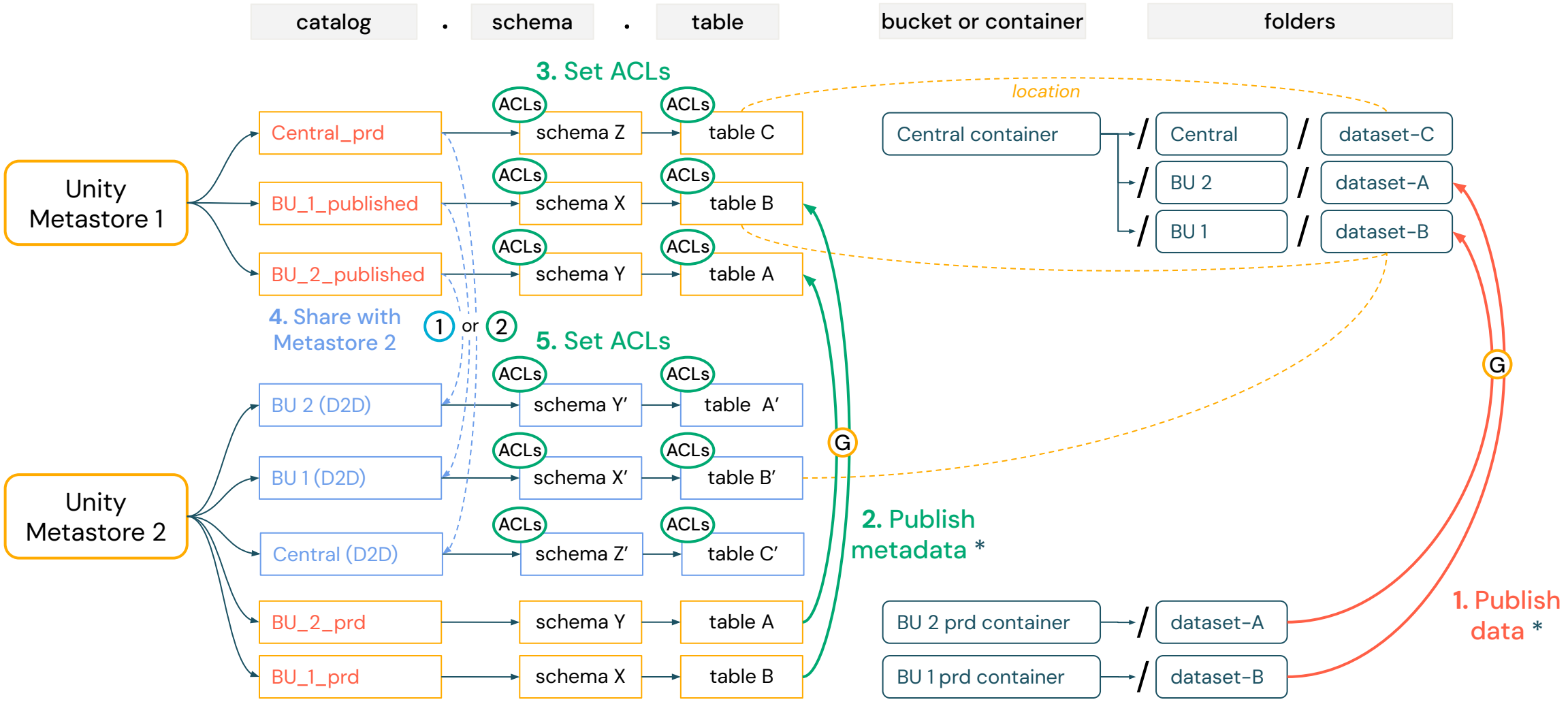
- Published data will be discovered in Metastore 2 by the BUs and consumed from the central PRD storage

Platform operations

- Central team provides platform blueprints and creates environments for BUs (automated)



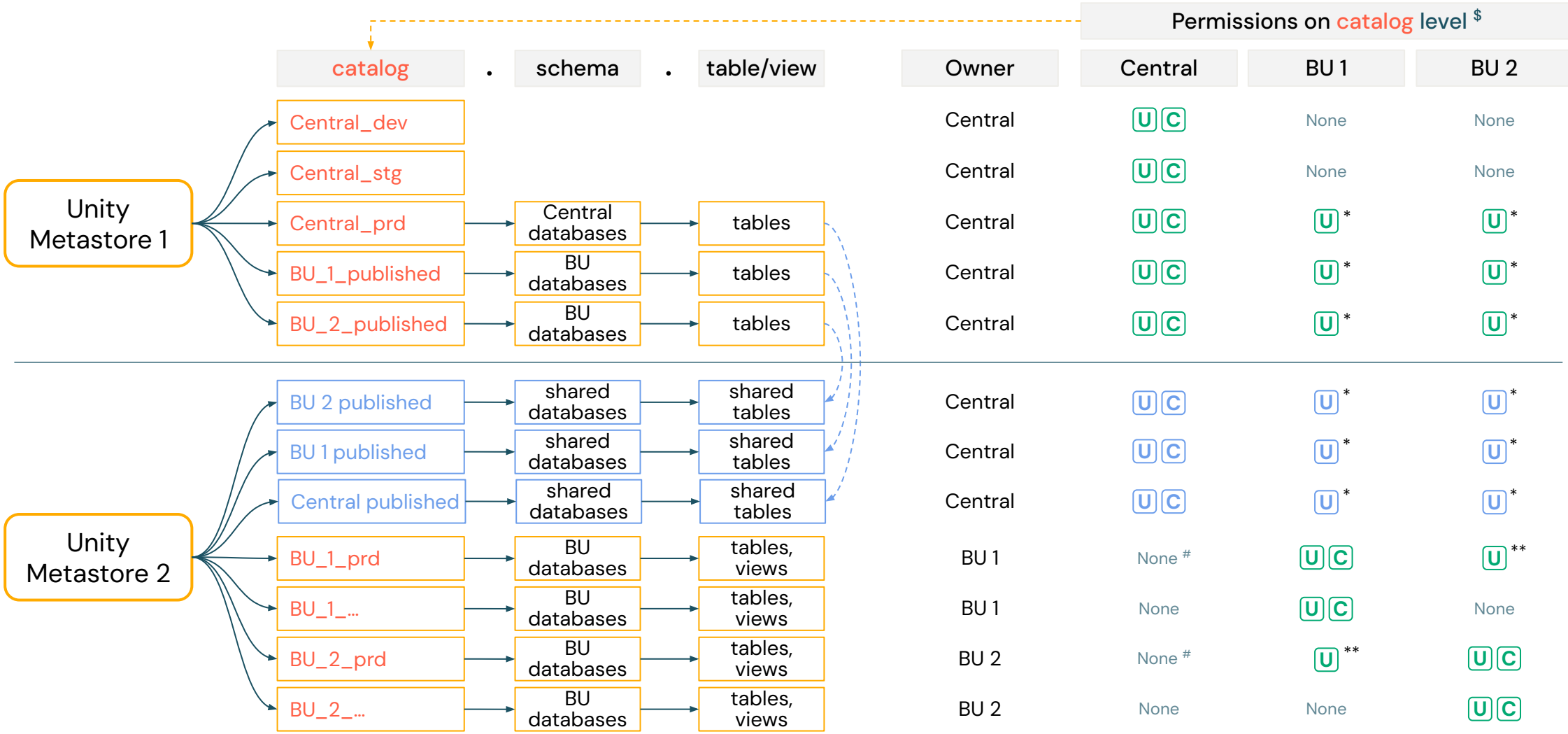
Option 2: Centralized Publishing Process



* For details see push/pull discussion earlier

Option 2: Metadata Management

U USE CATALOG
C CREATE SCHEMA
_prd: production
_stg: staging
_dev: development



$ to access tables, USE SCHEMA and SELECT also need to be provided on schema and table level
* catalog where other BUs can be given access to schemas and tables/views from Central team
** catalogs where other BUs can be given access to schemas and tables/views
\# if Central needs access to BU data, Delta Sharing or Lakehouse Federation need to be set up

System Account Mappings

Same as in Scenario 4.1

Pattern 4:

Phased Migration to UC

Migration approach with 2 security zones

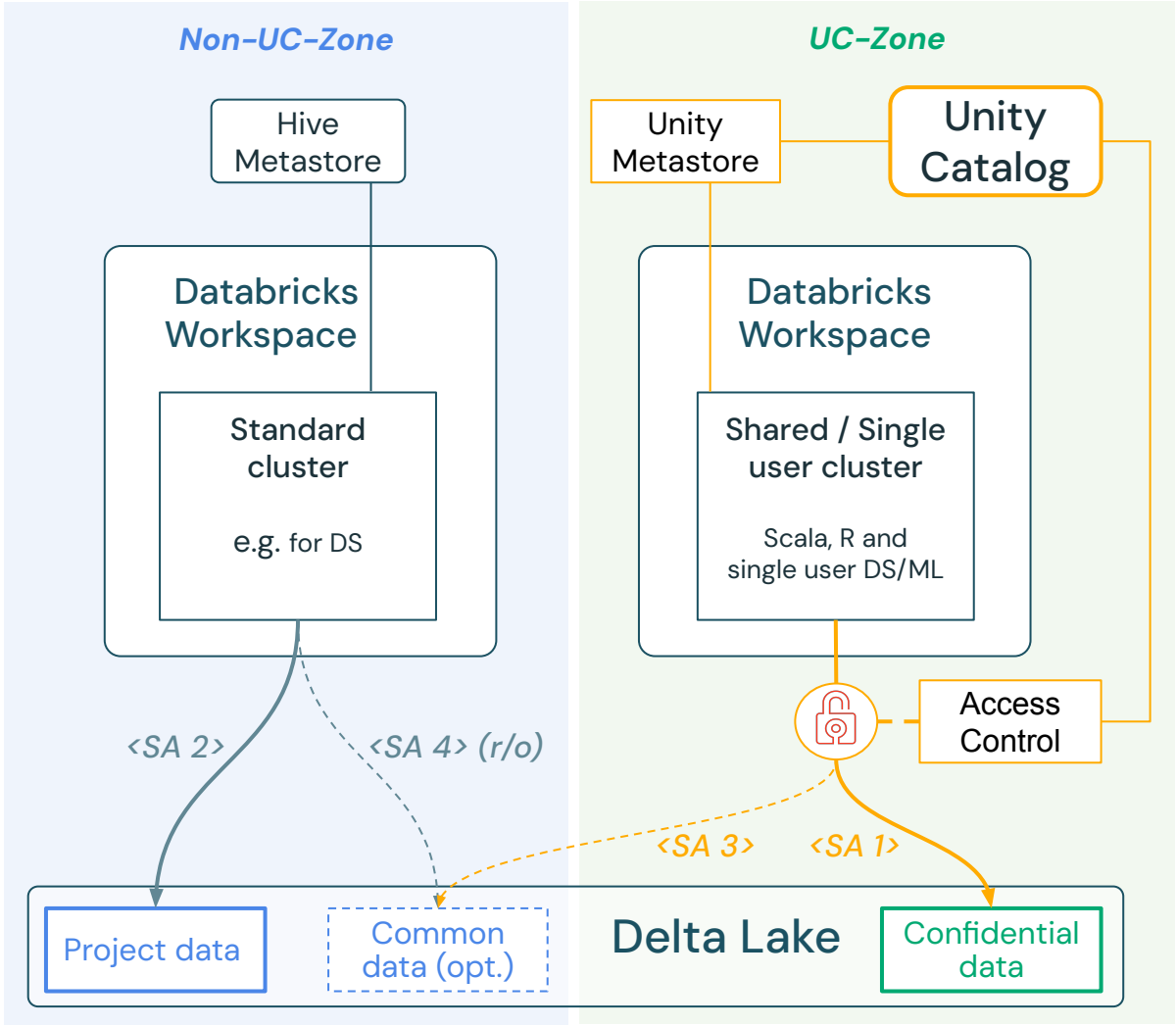
Current situation

Large customers (especially on Azure when they have 10s to 100s of workspaces) won't do a big bang migration independent of workload type

Two security zones

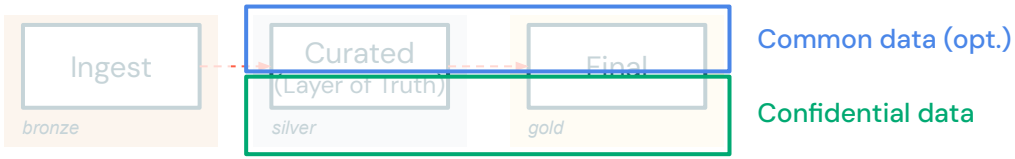
- UC Zone
  All access is managed by Unity Catalog with fine grained permissions.
- Non-UC Zone
  Projects that can't be migrated due to business or technical reasons can be kept in the well know Standard cluster world and migrated later

High level Two-Zone setup



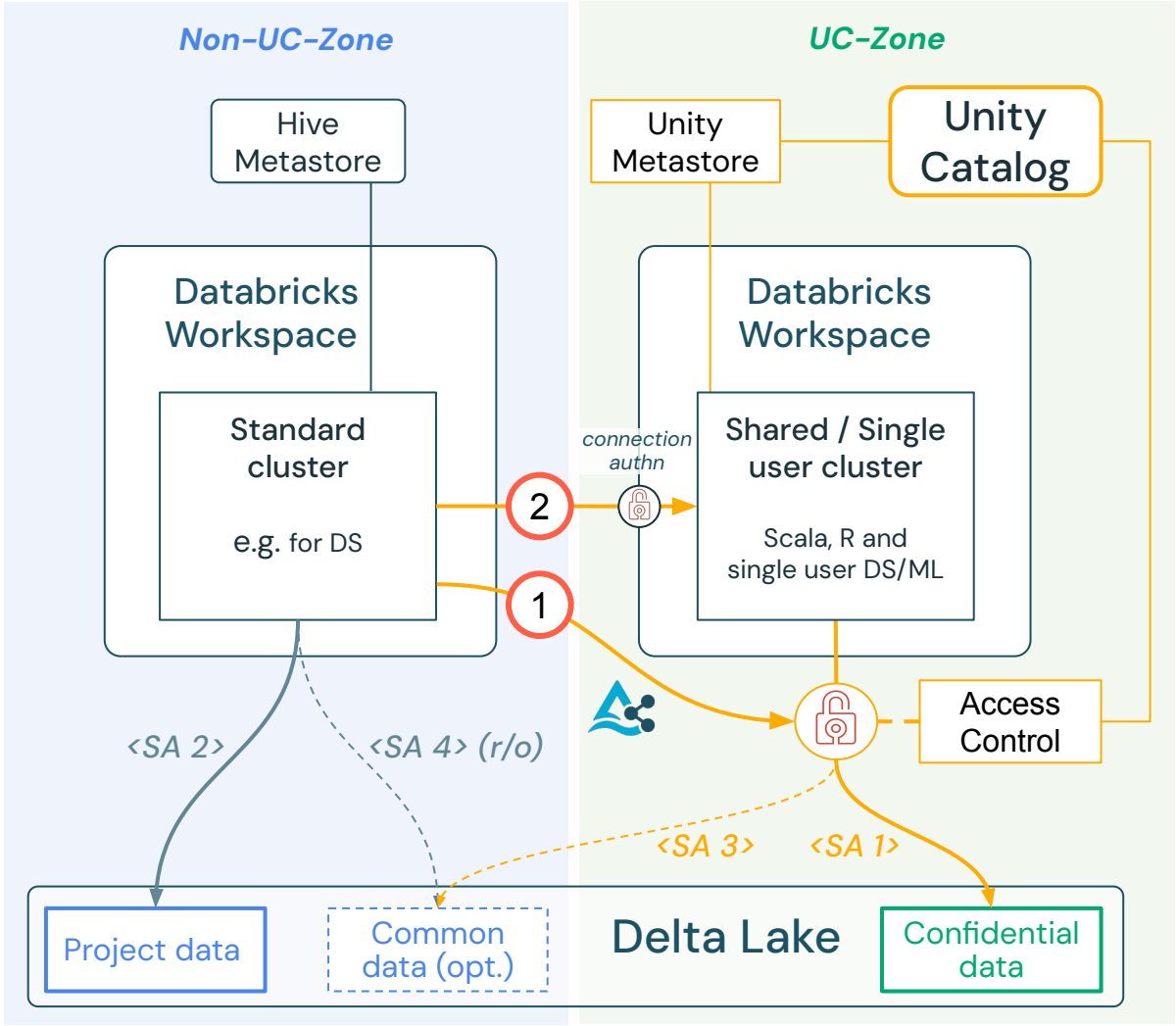
Approach

1. Split data into buckets/containers for
   - **Project data:** Location for the data of non-migrated projects
   - **Common data (optional):** Examples could be public data (weather, sociodemographic data, …) or company data with a low level of confidentiality
   - **Confidential data:** No direct access by Non-UC-Zone Workspaces and available with fine grained security for UC-Zone users



2. Create two zones
   - One secured by Unity catalog for all migrated use cases and projects (data migrated to “Confidential data” if necessary and metadata migrated from Hive Metastore to UC)
   - One for non migrated projects based on “classical” workspaces and security
3. Use different System Accounts for the different data locations

Two Zone setup: Access to UC secured data



Projects might need data from the the UC-Zone, i.e. "Confidential data". However, due to security reasons, this cannot be provided directly.

Options (not ordered by priority):

1. Open Delta Sharing (not Databricks-to-Databricks)
2. Lakehouse Federation, JDBC or Databricks Python connector against DB SQL in the UC-Zone.

Notes:

- To avoid that confidential data will be played back to the UC-Zone via "Common data" (if exists), consider making "Common data" read only for Non-UC-Zone users
- On Azure customers prefer AAD token. While the connector works with AAD token, creating a user AAD token is a involved process and usually no option for end users

Pattern 5:

ML Setup

Using MLR and Unity Catalog