

SKYSCANNER'S JOURNEY OF ENABLING PRACTICAL DATA & AI GOVERNANCE

Michael Ewins, Director of Engineering & JMLaplante, Principal Software Engineer

About Skyscanner




A global leader in travel and technology connecting millions of people every day with trusted travel providers.

We reach more than 1 billion users every month

110M

A silhouette of a person wearing a cap stands on a rocky mountain peak, looking out over a vast landscape of white clouds and blue sky. The person is positioned behind the large text '110M'.

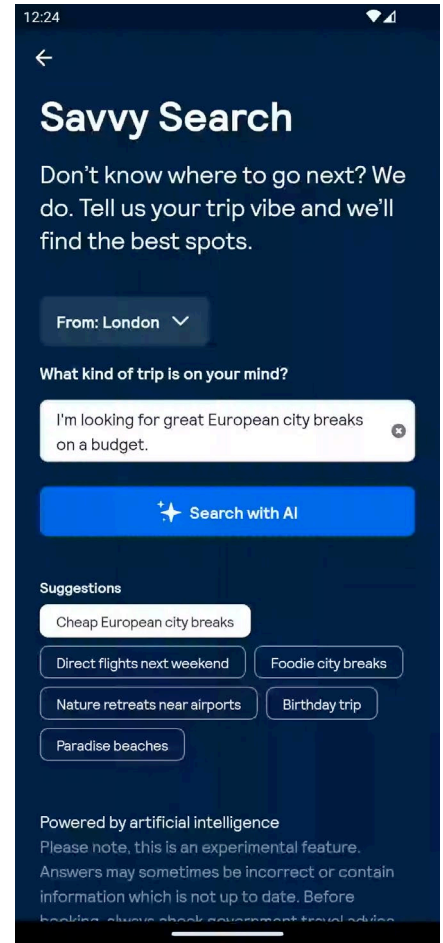


5,000

search requests per second

Data fuels Skyscanner

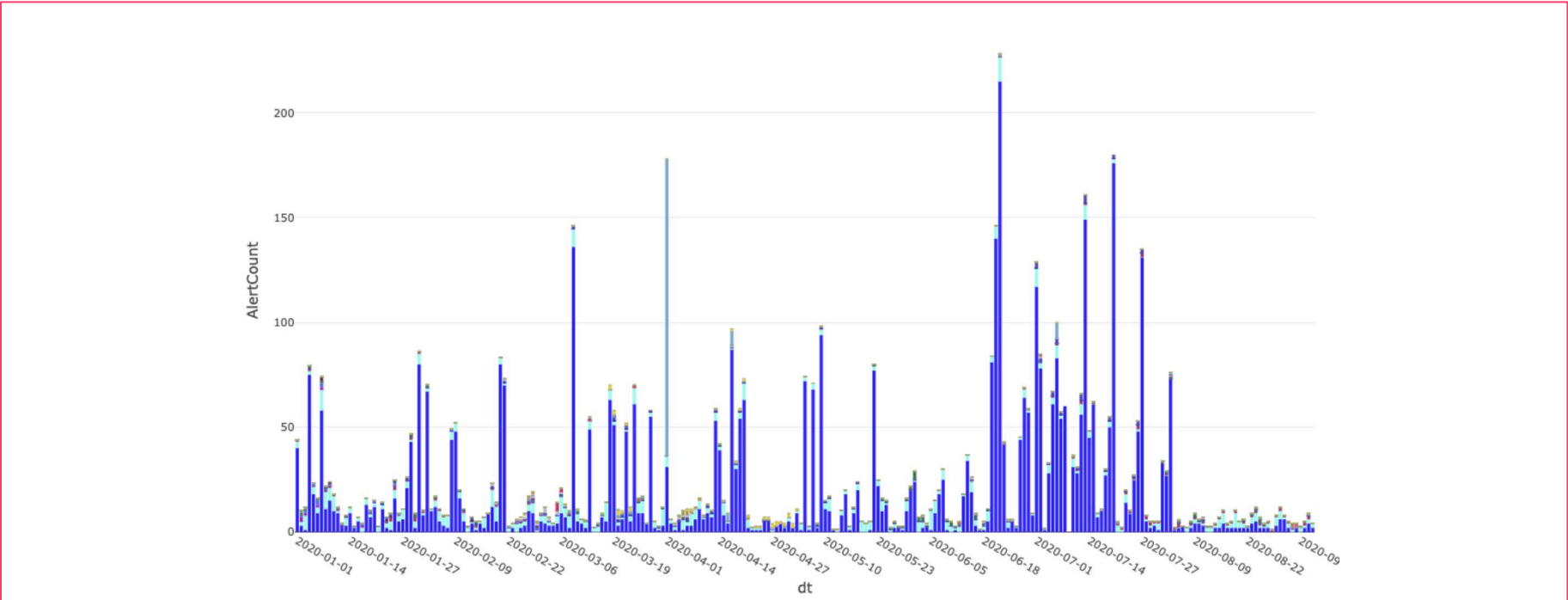
- Analytical data is used to optimize the business and user experience.
- We use Machine Learning models to rank and recommend travel options
- AI Search – answer traveller questions to inspire new experiences.



Data journey: where we started

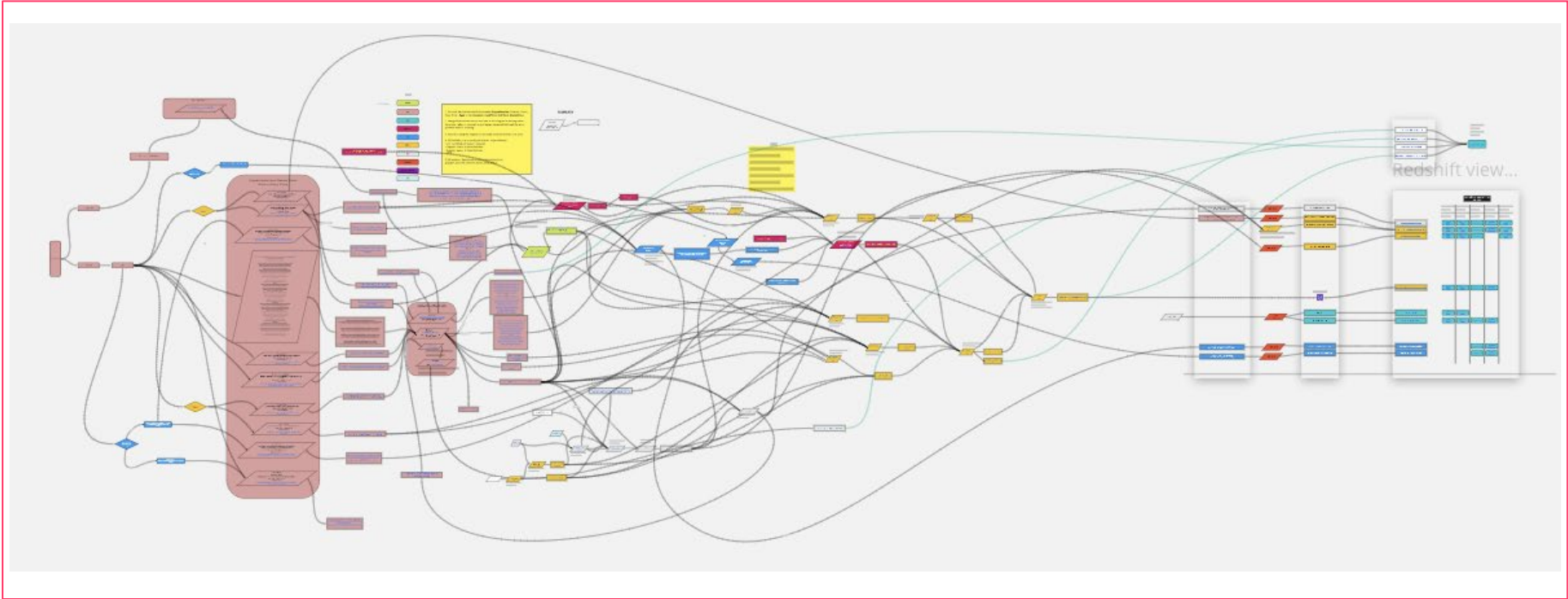
Firefighting to keep data flowing

We were fixing symptoms not underlying issues



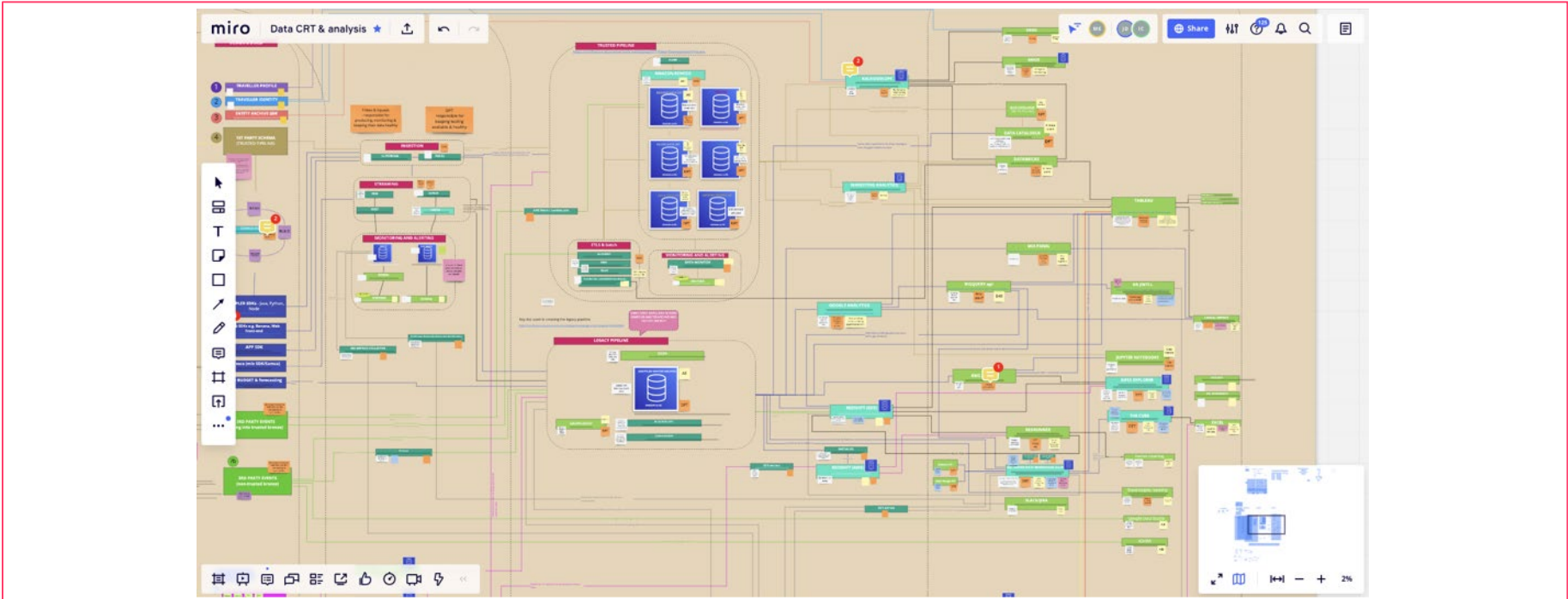
Complex Data Pipelines

Late arriving data, inconsistent data quality, unhappy data customers



Multiple Data Technology Stacks

Complexity, duplicate systems, expensive to operate



Data Tribe

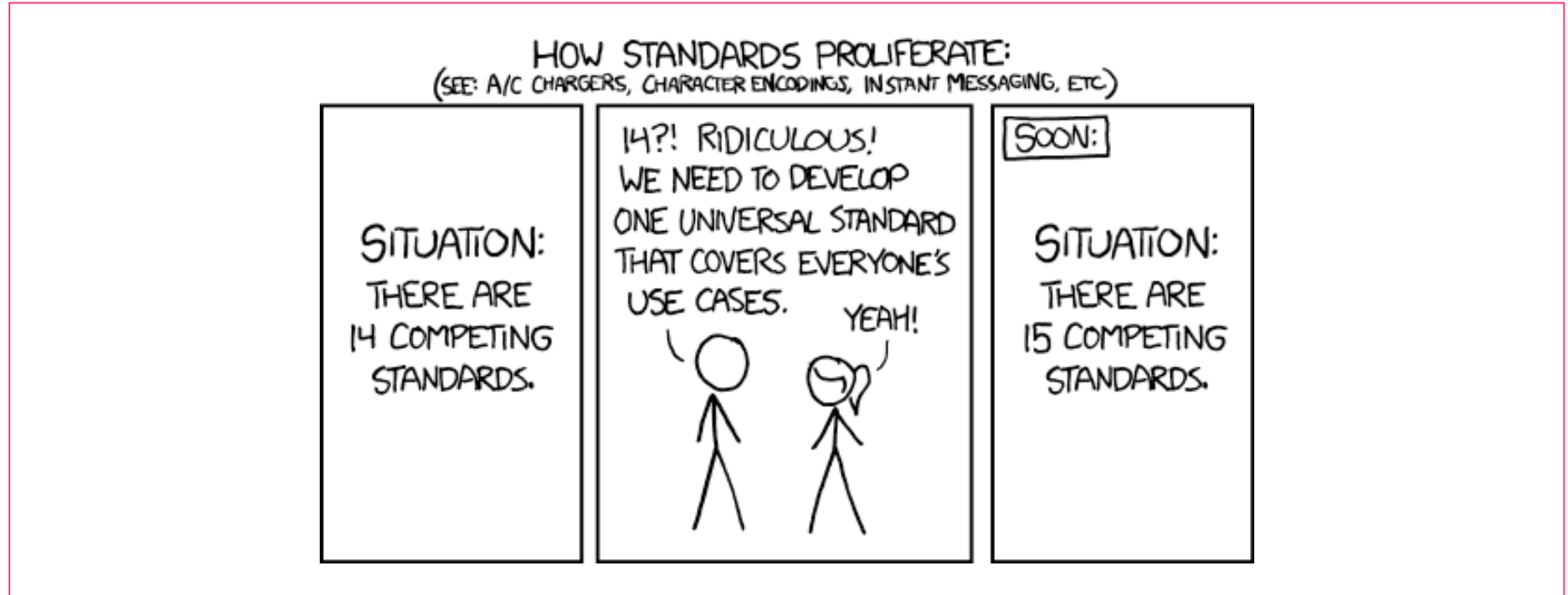
Our mission...

Skyscanner needs
reliable & trustworthy data... so
that we enable smart decisions
quickly using data

Direction of travel

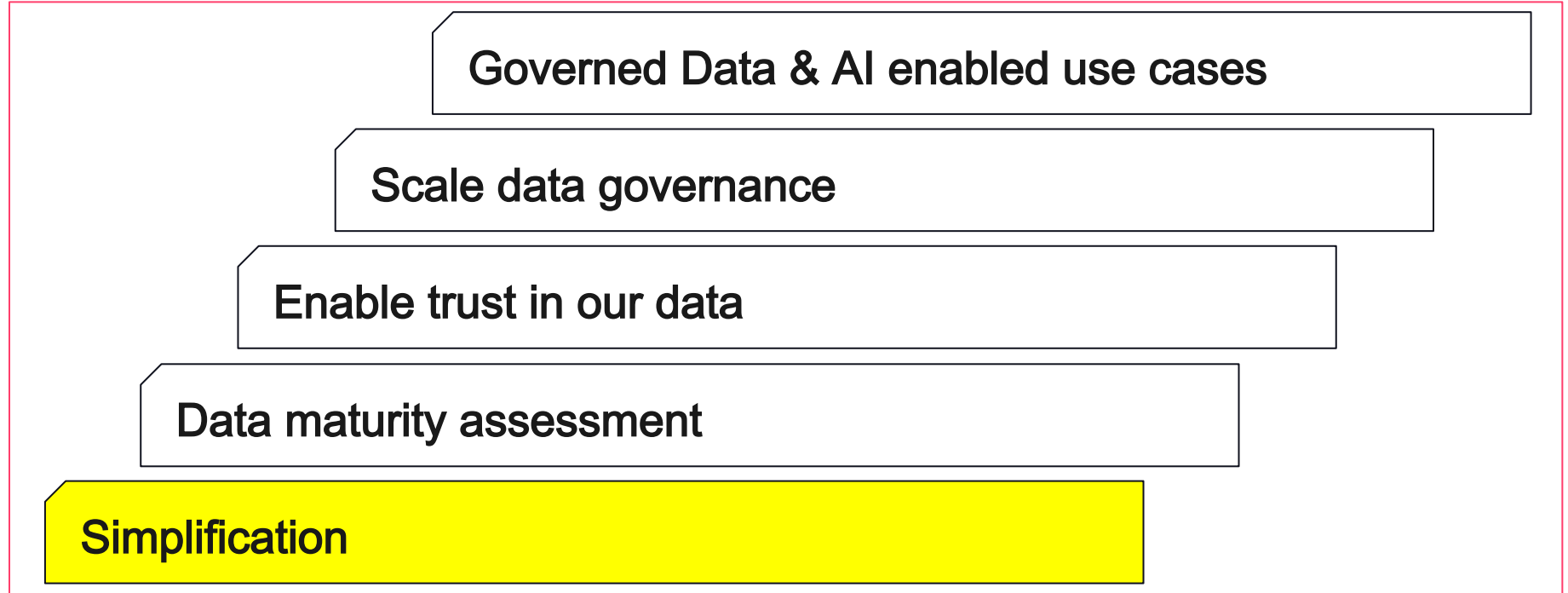
The need for simplification

This XKCD comic sums up where we were...



Skyscanner data journey

Strategic staircase



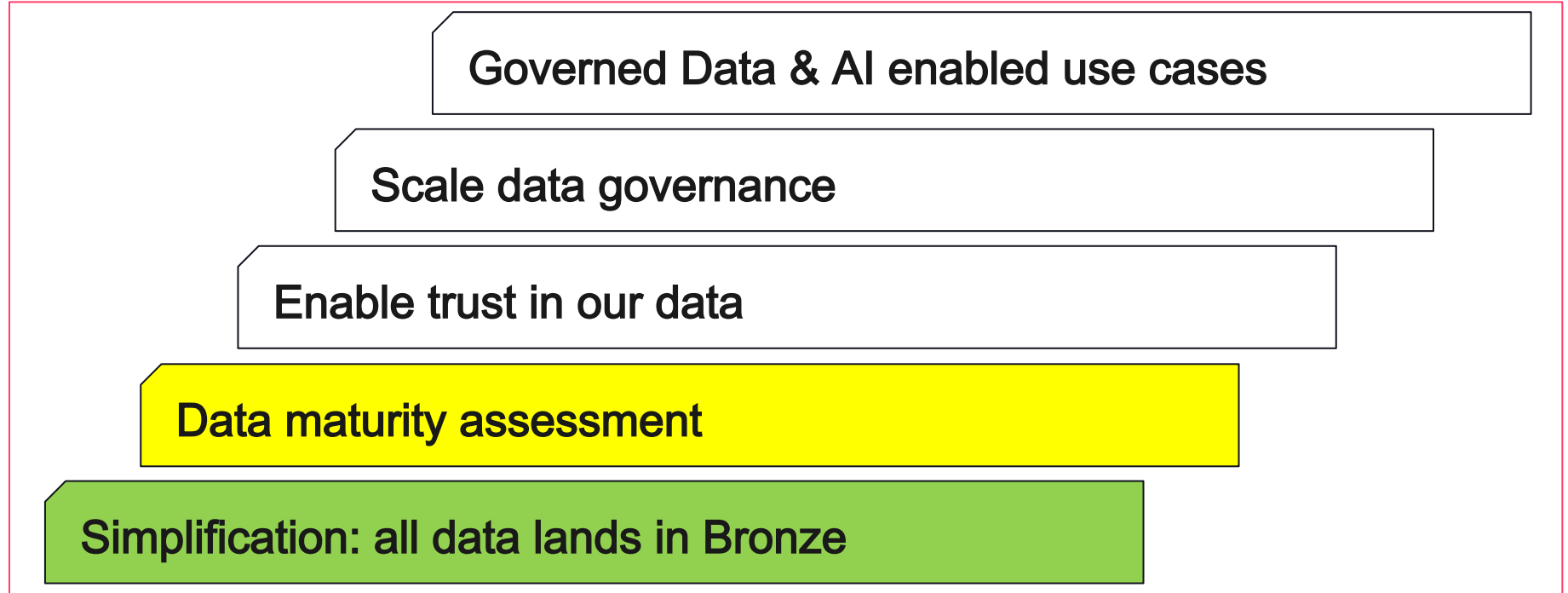
Simplification: from ETL to ELT

Investing in improvements around Medallion architecture

- All data ingestion lands in Bronze
- Transform data using PySpark / Databricks
- Consume data from Medallion

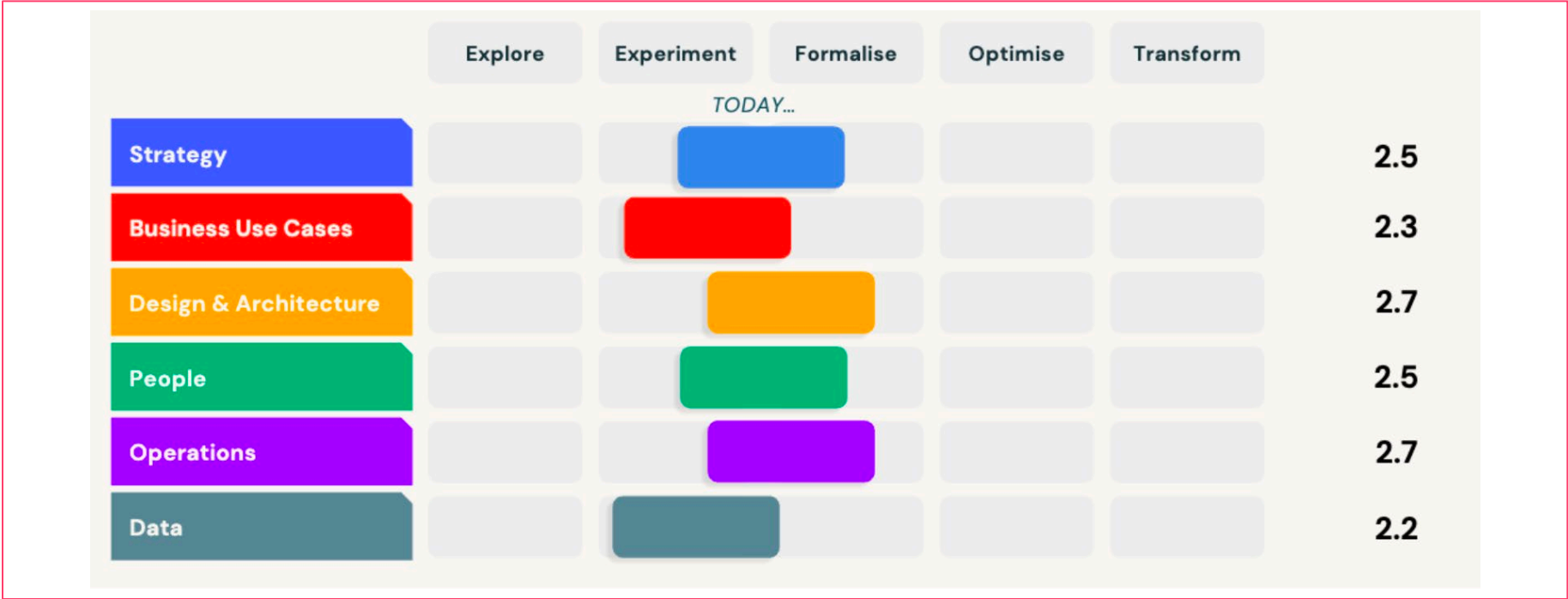
Skyscanner data journey

Strategic staircase



Data Maturity Assessment

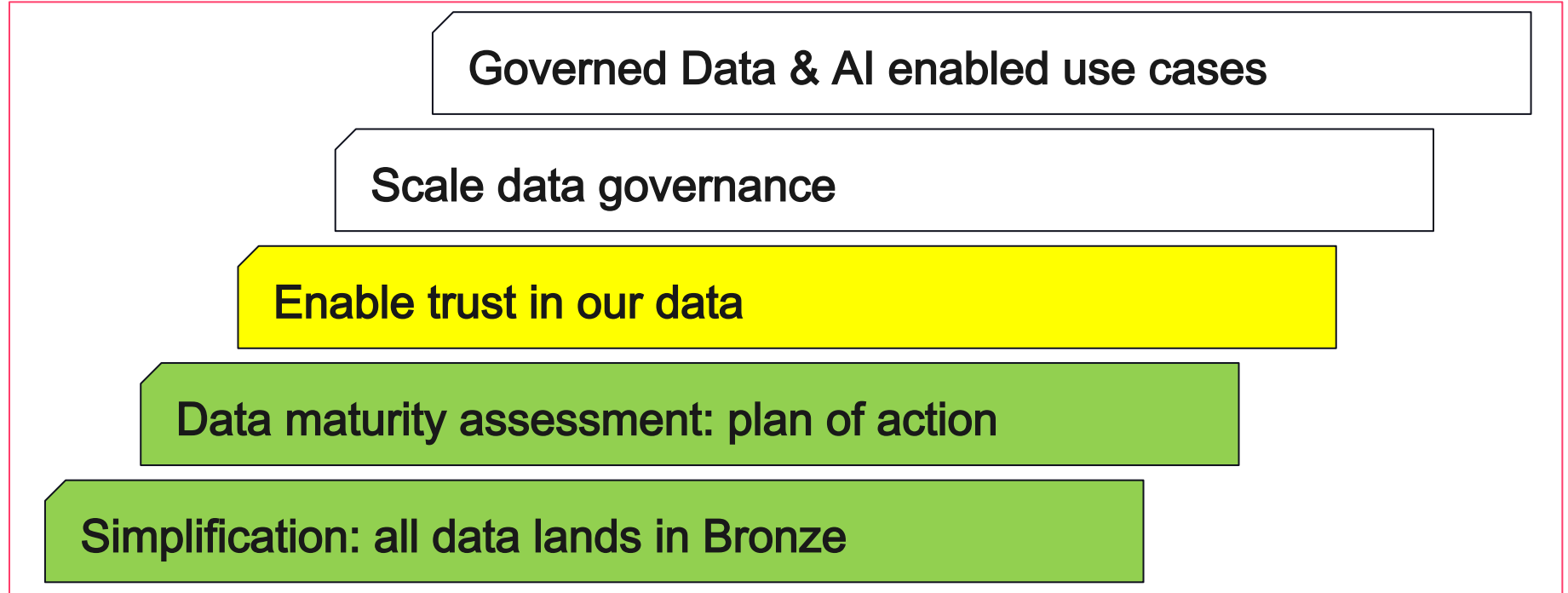
Learning from others...



Enable
trust
in
our
data

Skyscanner data journey

Strategic staircase



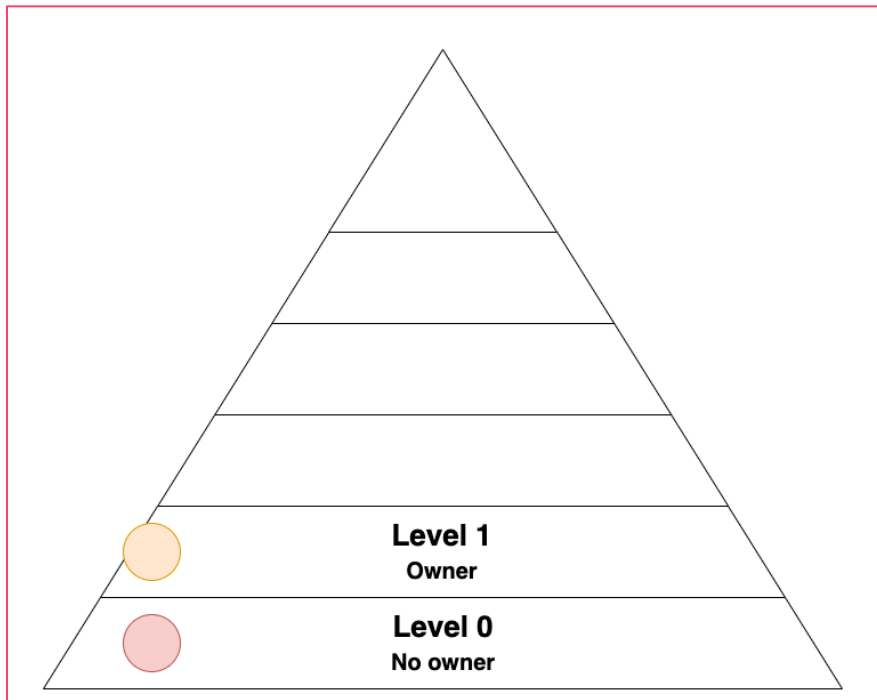
Introducing Data Governance

What is our business critical data?

- Hive meta store contained ~30,000 tables
- Not all data is equal
- Define our business-critical use cases
- We identified ~350 business critical data sets

Measurable Data Governance

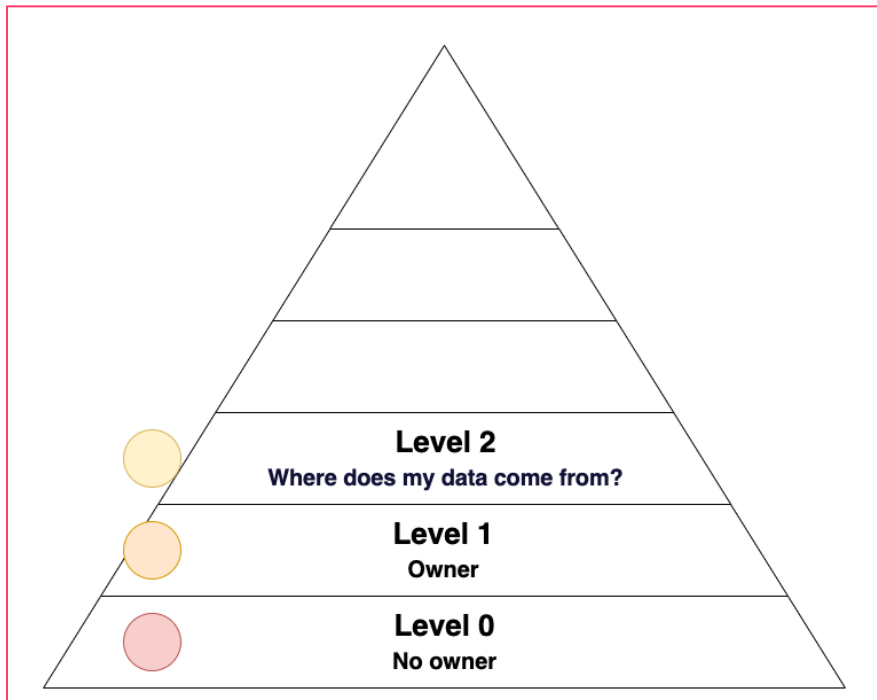
Ownership



- Who owns this data?
- Ownership by teams

Measurable Data Governance

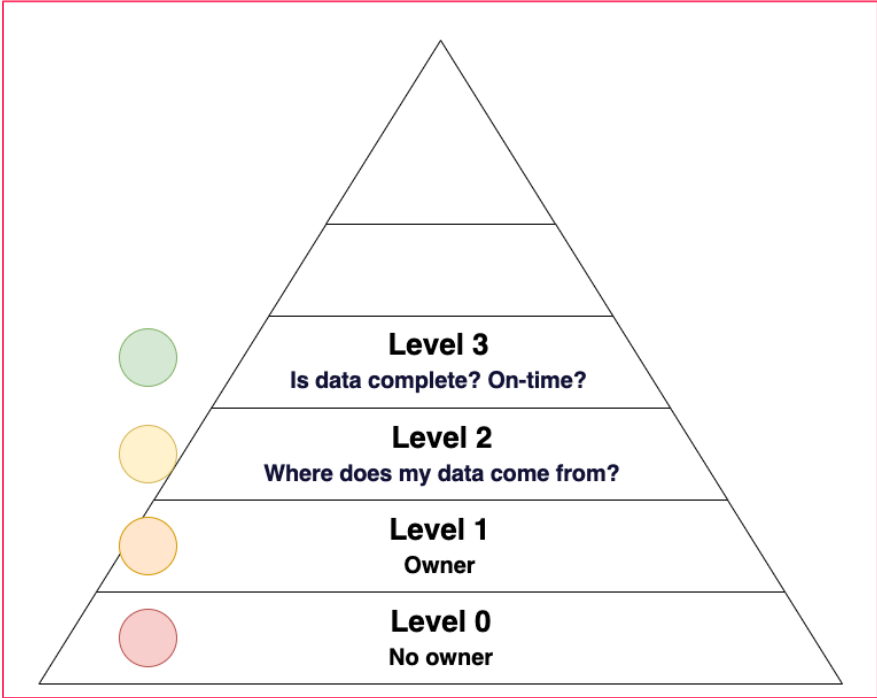
Lineage & access control



- Where does my data come from?
- Who uses my data?

Measurable Data Governance

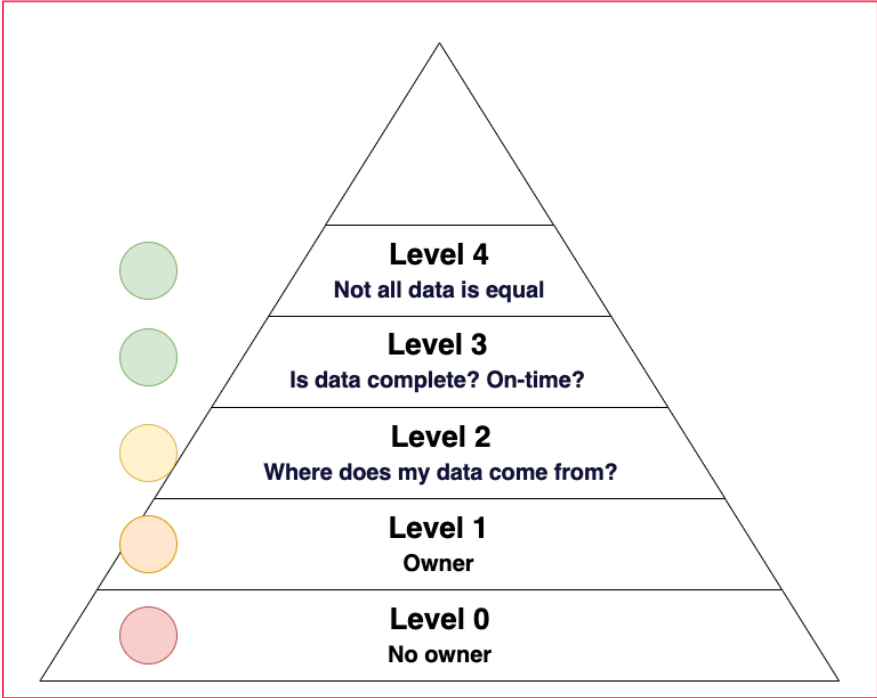
Table level data quality checks



- Is my data complete?
- Is my data on-time?

Measurable Data Governance

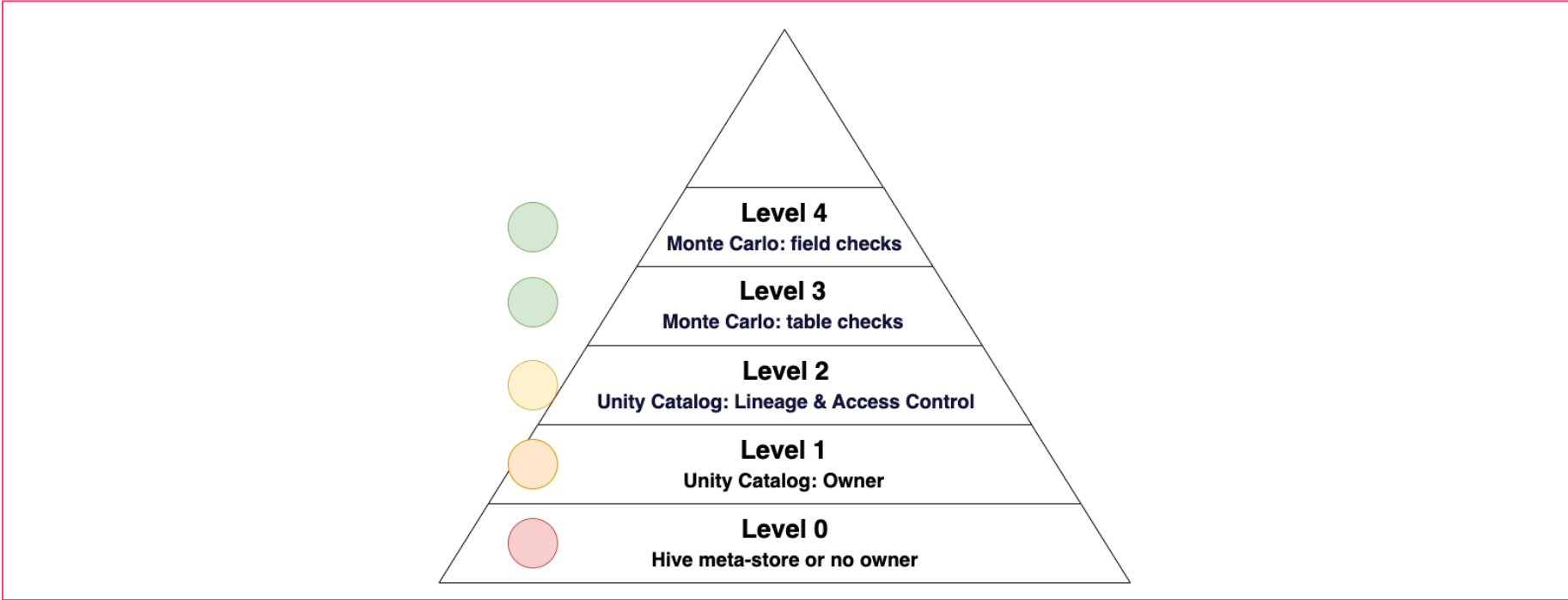
Field level data quality checks



- Not all data is equal
- Extra checks for important data

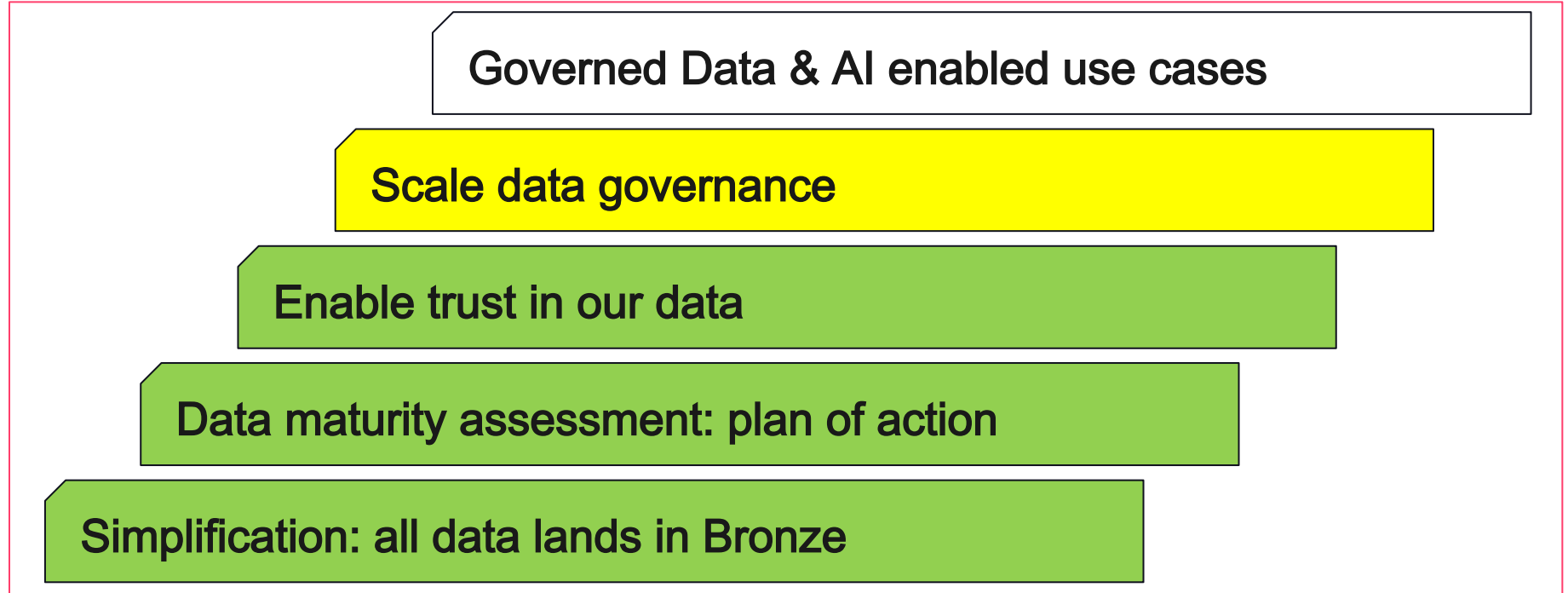
Measurable Data Governance

Coming up... deeper dive on implementation decisions



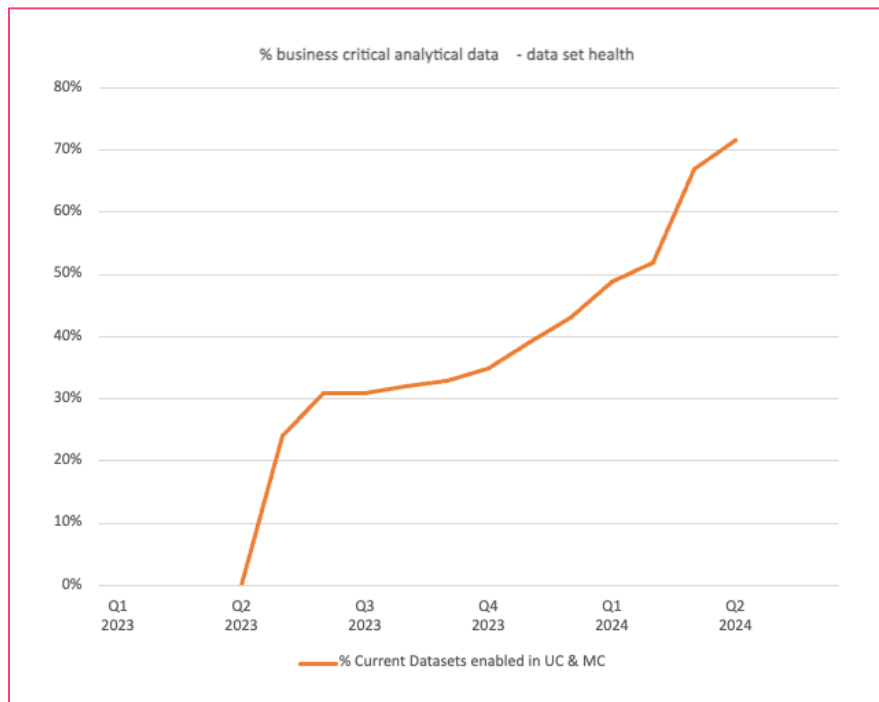
Skyscanner data journey

Strategic staircase



Tracking data set health enablement

Initial focus on ~350 business critical datasets



- Calculate progress:
- 25% ownership
- 25% enabled in UC
- 25% table quality checks
- 25% field quality checks

Unity Catalog benefits now visible

Discoverability, data domains, lineage, access control, etc.

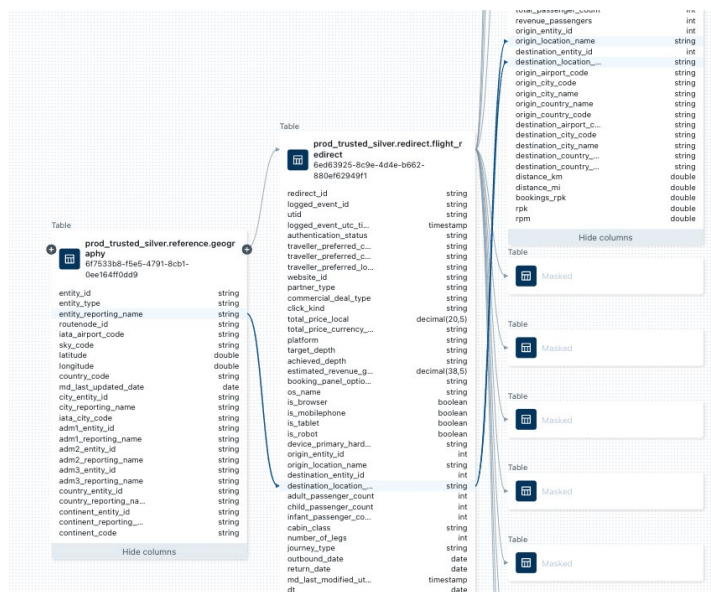
Catalogs > prod_trusted_silver > search_request >

prod_trusted_silver.search_request.flight_search_request ☆

Overview Sample Data Details Permissions History Lineage Insights Quality

Filter columns...

Column	Type	Comment	Tags
dt	string	Date partition - this is the date in which the event was emitted.	
search_request_id	string	Unique identifier of a search event.	
utid	string	Unique Traveller Identifier - identify and track Anonymous and Authenticated users on the Skyscanner platform.	
logged_event_utc_tim...	timestamp	The timestamp of the event. The same as grappler receive timestamp (not client timestamp).	
device_guid	string	Unique identifier for a users device.	

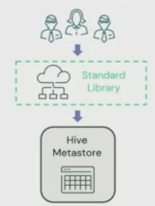


But how could we go faster?

Inspiration came from talks at Data & AI Summit 2023

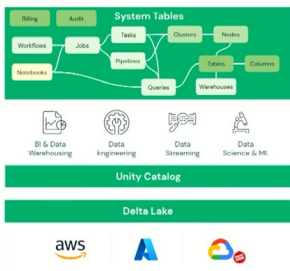
Migration Strategy

Populate Unity Catalog in the background through standard libraries



- Teams used spark to create external tables in Hive metastore
- Executed via a standard library function, `create_table`

```
def create_table(database: str, table: str, df: DataFrame):  
  (  
    df.write.format("delta")  
    .mode("overwrite")  
    .saveAsTable(f"{database}.{table}")  
  )
```



System Tables

Unified operational data plane for AI driven insights on the Lakehouse Platform

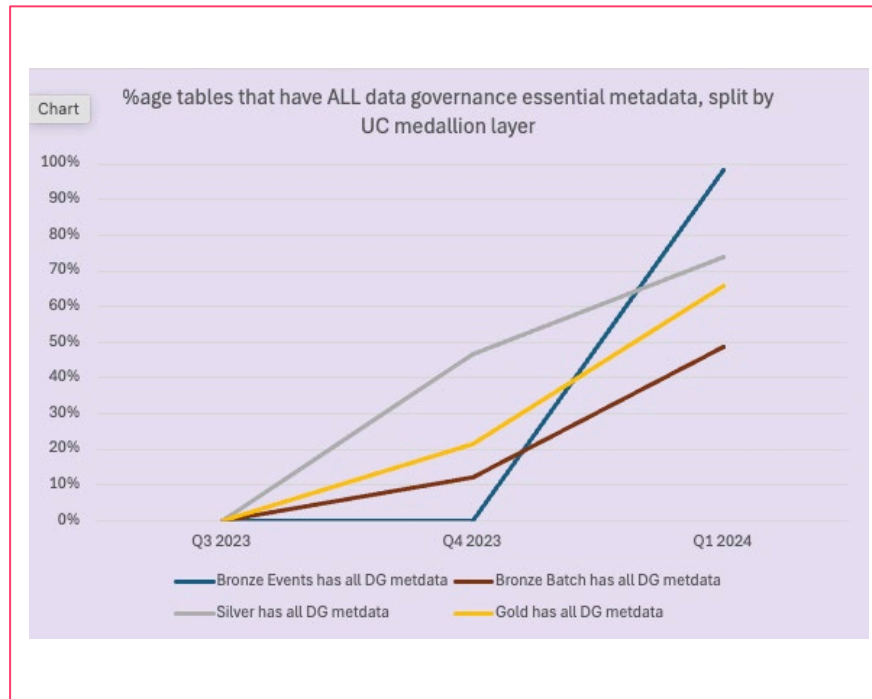
- Consistent
- Connected
- Comprehensive

- YipitData – *“how can we get Unity Catalog without teams modifying their code?”*
- System Tables - see reads & writes for tables outside of Unity Catalog
- We learned... ~1500 active tables not 30000

Tracking Metadata adoption

Automating data governance

- Data owner
- Data Classification
- Data Category
- Data retention period
- Business criticality
- Table and Column descriptions



Evolving Data Governance

New criteria + clearer asks

Criteria	Score 0	Score 1	Score 2	Score 3	Score 4
Ownership	No owner OR invalid owner	Valid owner in legacy catalog	Valid owner defined in Unity Catalog ✓		
Cataloging	Uncatalogued	AWS Glue Catalog	Unity Catalog (not necessarily Medallion)	Unity Catalog + Medallion ✓	
Metadata	No metadata	Incomplete mandatory metadata in Unity Catalog	All mandatory metadata in Unity Catalog	All recommended metadata in Unity Catalog ✓	
Data Reliability	Dataset not monitored	Monitoring enabled < 100% Status Update Rate	Table level checks + 100% Status Update Rate	Field level or custom checks + 100% Status Update Rate	

Deeper Dive: Enable trust in our data

Enabling Unity Catalog

Re-platform & improve data architecture

Architecture

- Review Databricks Workspace topology

Enabling Unity Catalog

Key decisions

Architecture

- Review Databricks Workspace topology
- UC Metastore structure aligned with Medallion & data domains

Enabling Unity Catalog

Key decisions

Architecture

- Review Databricks Workspace topology
- UC Metastore structure aligned with Medallion & data domains
- Catalog visibility across workspaces
- Access control model

Enabling Unity Catalog

Key decisions

Architecture

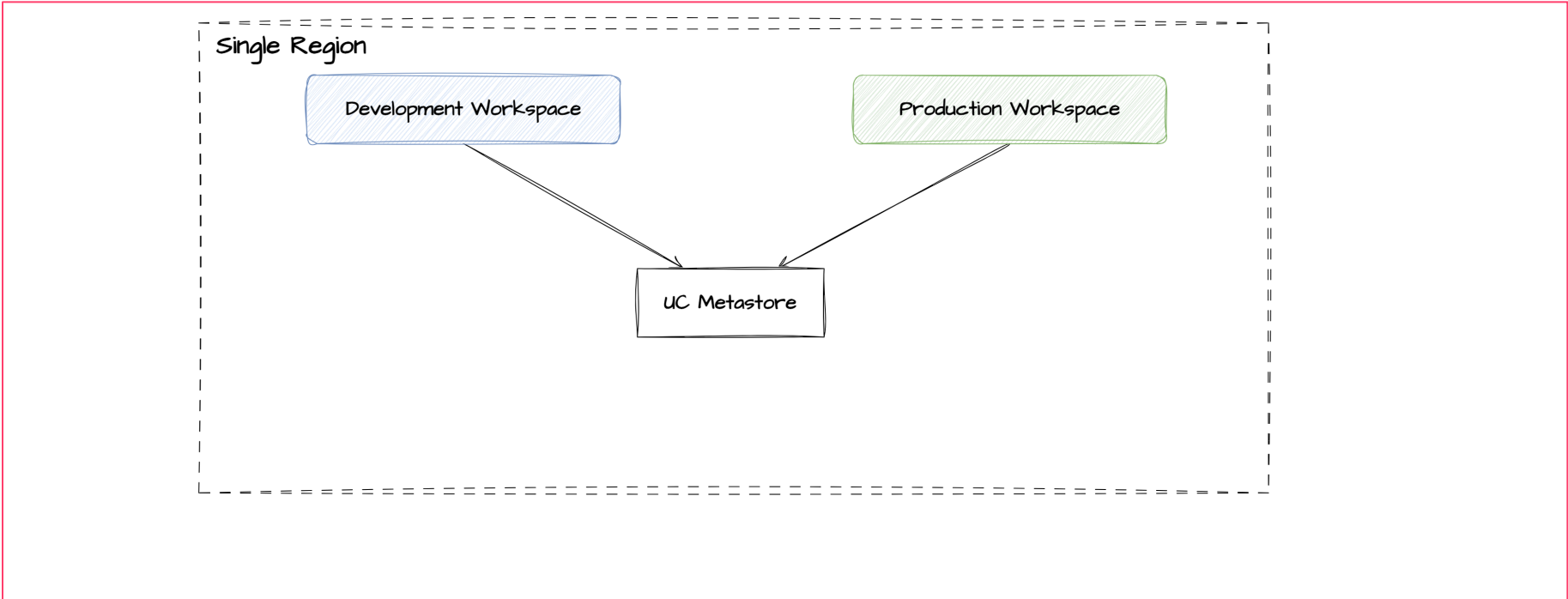
- Review Databricks Workspace topology
- UC Metastore structure aligned with Medallion & data domains
- Catalog visibility across workspaces
- Access control model

Technology

- Storage layer infrastructure
- Creation of UC catalogs and enabling it alongside hive
- Self-serve implementation

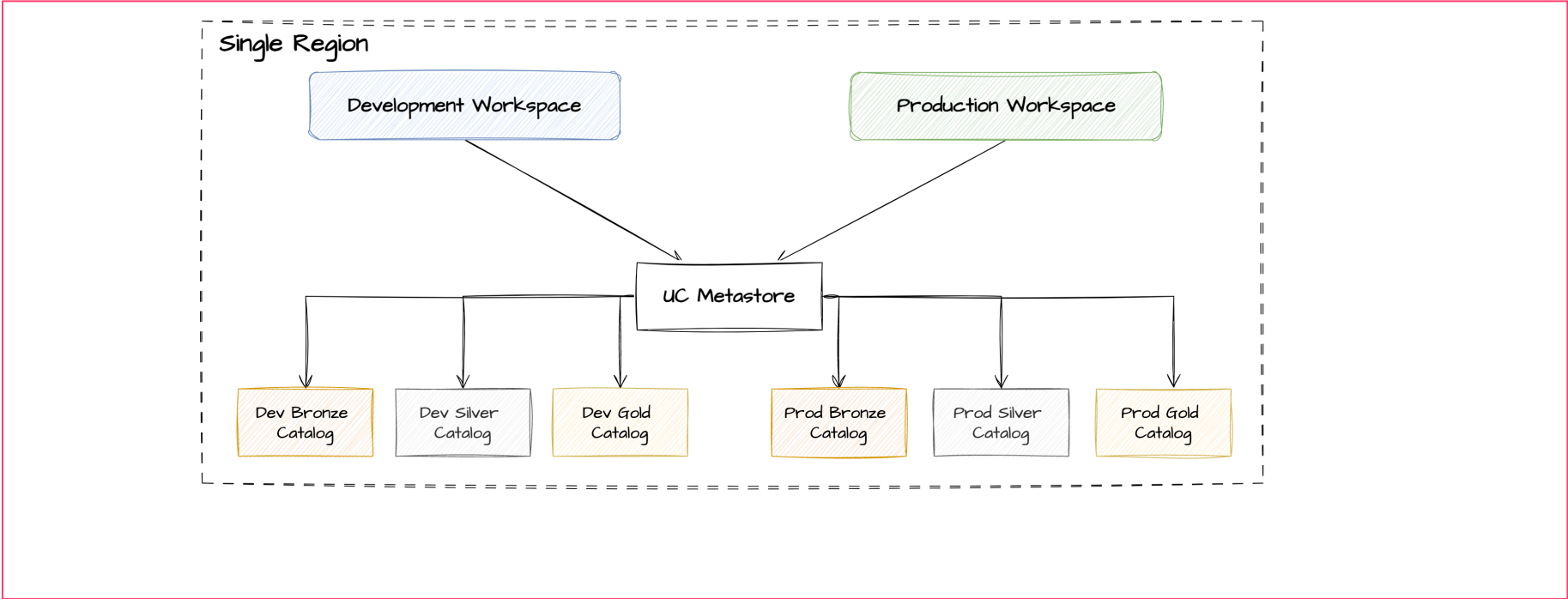
Unity Catalog Architecture

2 workspaces connected to UC, single region/metastore



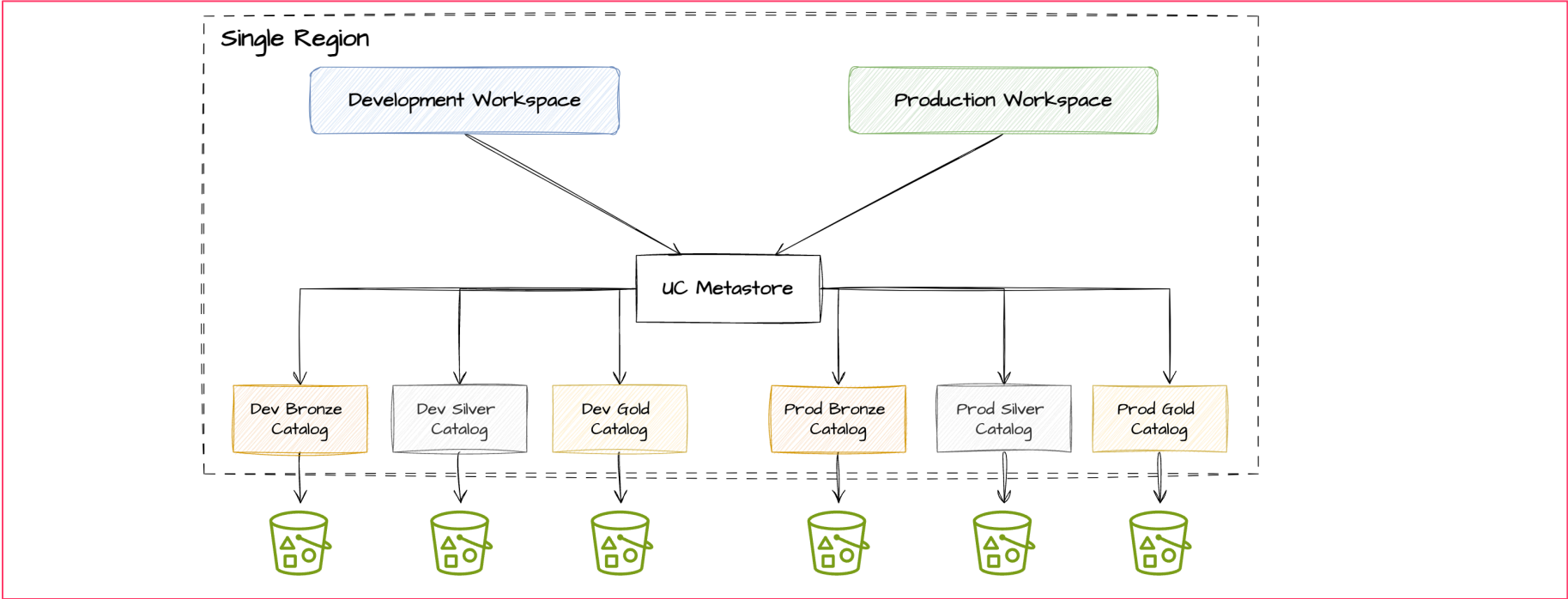
Unity Catalog Architecture

3 Medallion-aligned catalogs per workspace



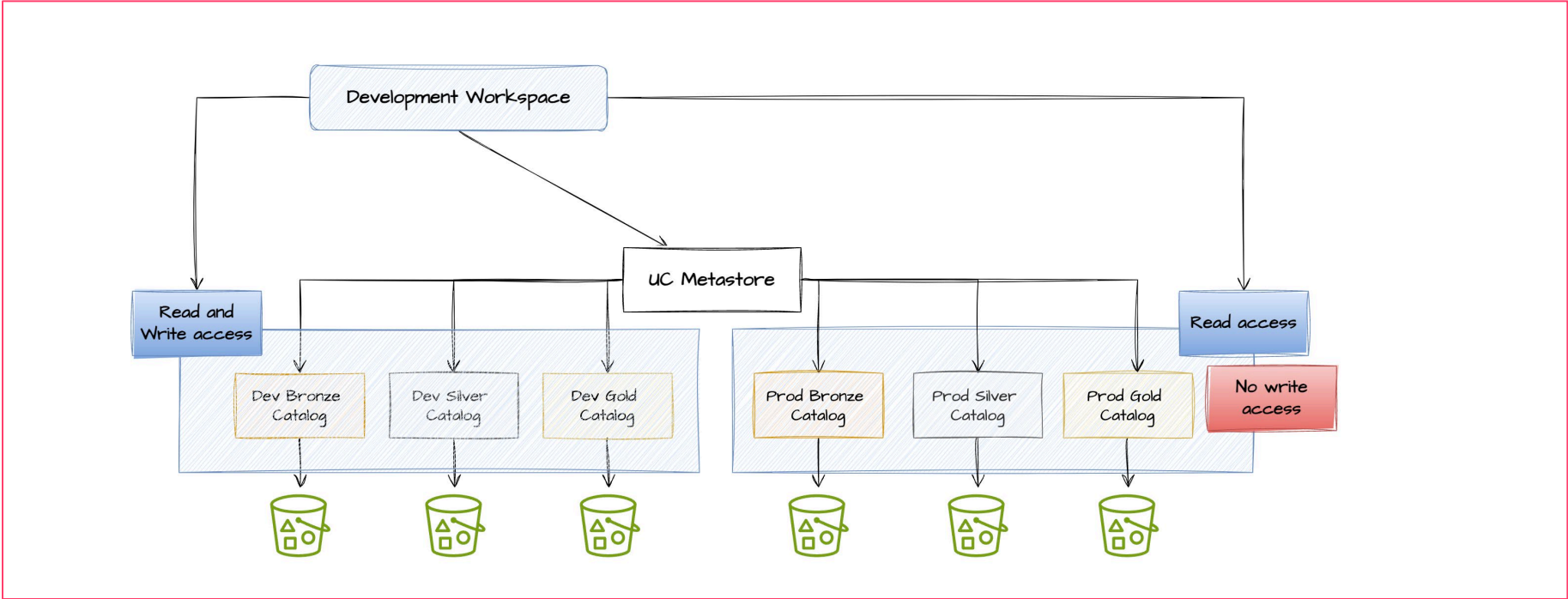
Unity Catalog Architecture

One S3 bucket per catalog



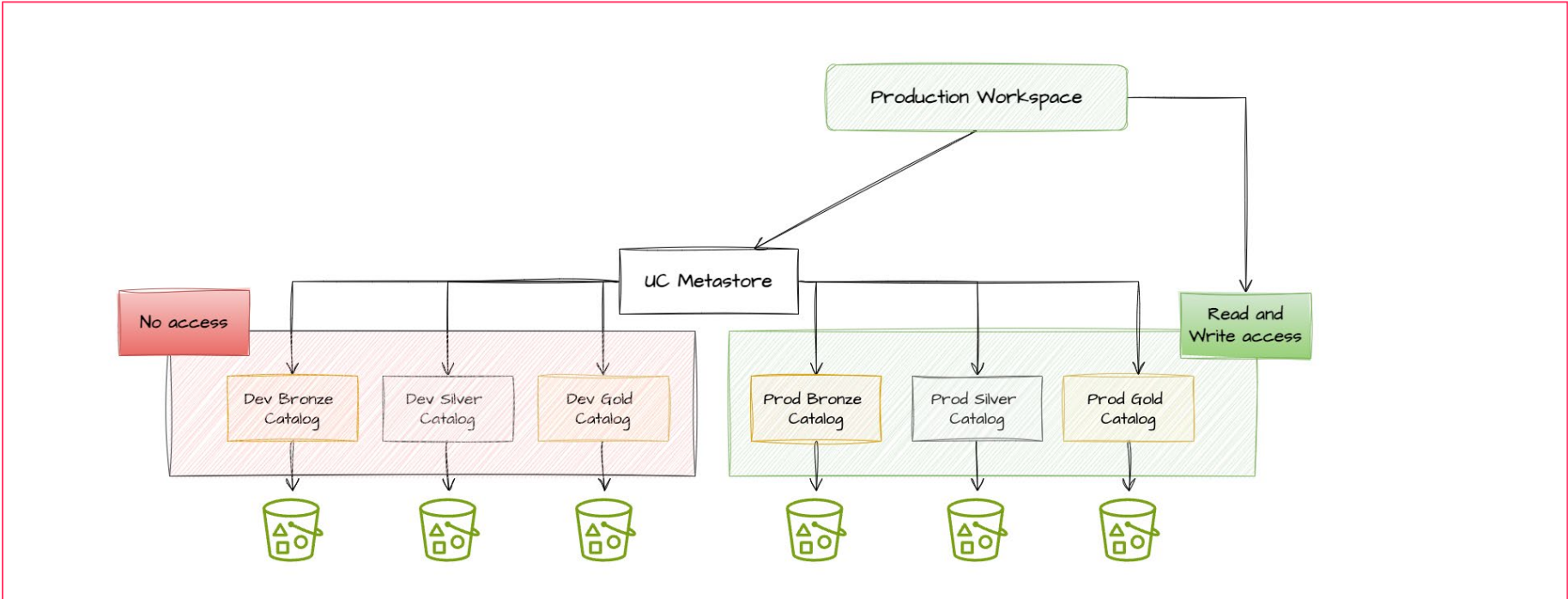
Unity Catalog Architecture

Catalog visibility - Development workspace



Unity Catalog Architecture

Catalog visibility - Production workspace



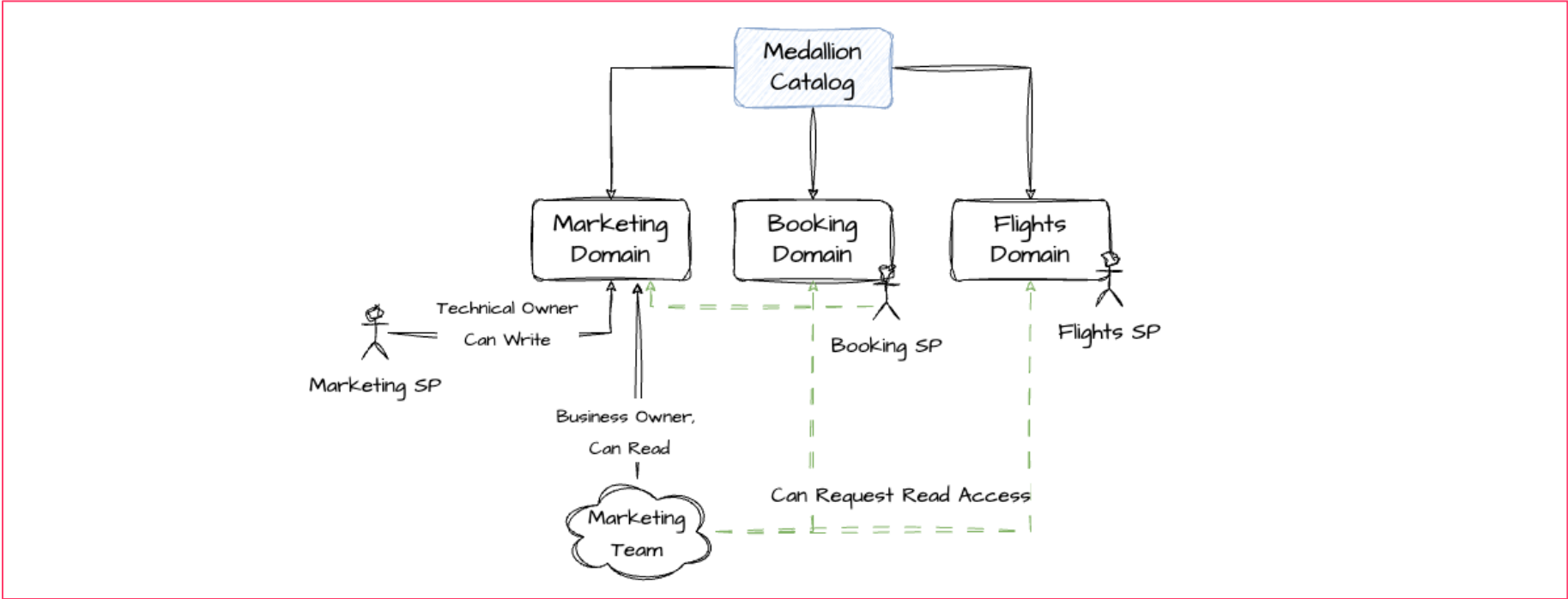
Access Control

Write access via service principals & Read access locked by default

- Technical vs. Business ownership
- Only service principals can be table owners
- Business owners get read access by default
- Teams and SPs can request read-only access to other domains

Access Control

Example



Self serve domains

Set of templates based on Databricks Terraform provider

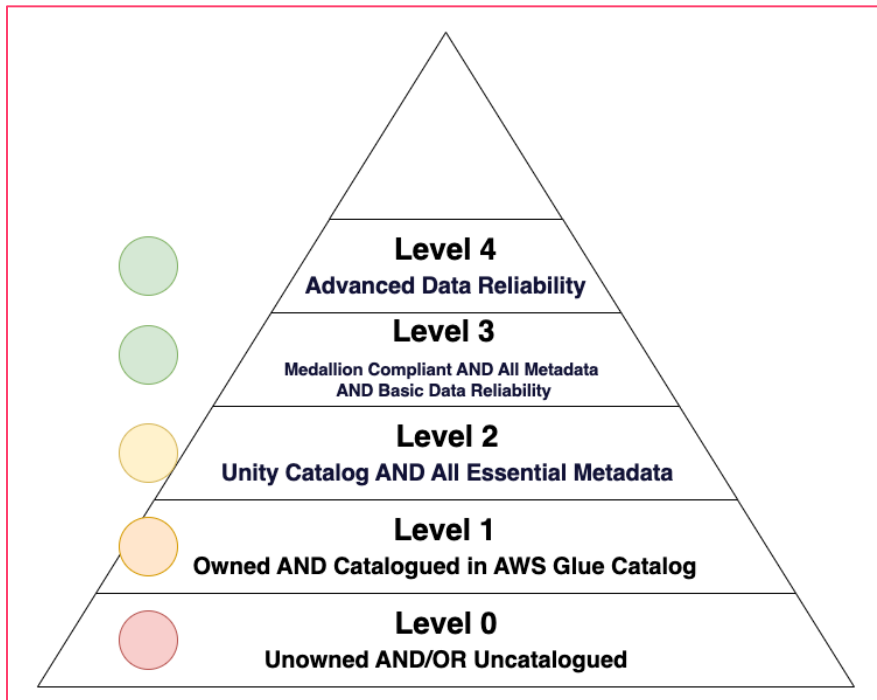
Terraform code for domain creation

```
module "create-prod-price-accuracy-domain" {
  name          = "price_accuracy" # domain name
  owner         = "peregrine" # this will become "owner" schema tag
  description   = "This domain contains data for prices accuracy"
  catalogs      = [
    catalog {
      name = "silver"
    }
  ]

  # Other teams and SPs, which want to read tables in this domain
  table_readers = {
    "silver" = {
      "calendar_cell_quote" = ["prod-price-coverage-domain", "Weathervane squad"],
      "inaccurate_calendar_cell" = ["prod-price-coverage-domain"]
    }
  }
}
```

Unity Catalog enabled

Unlocks level 2 of our data governance framework



- **Essential metadata**

- **Ownership**

- **Data Category and classification**

- **Descriptions**

- **SLOs**

- **Lineage & access control**

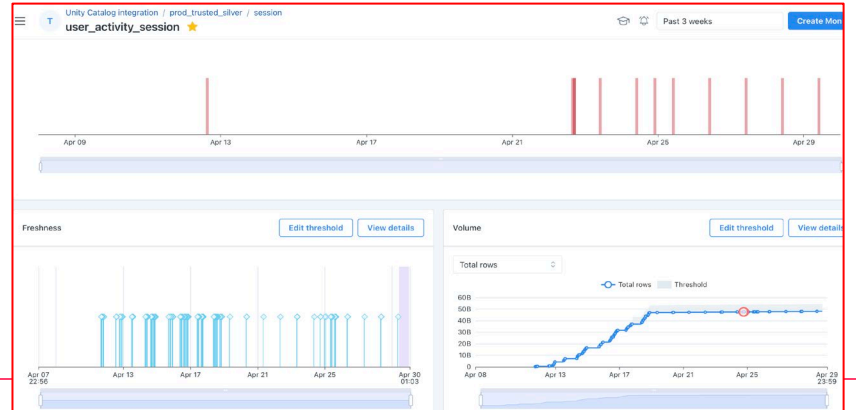
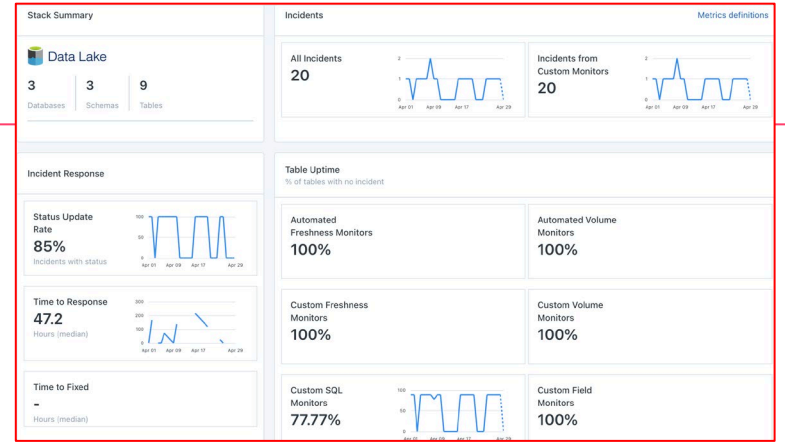
- Table level quality checks

- Field level quality checks

Introducing Monte Carlo

Data Observability Solution

- Enhanced data lineage
- ML-based table level monitors
- Field-level quality metrics
- Basic anomaly detection
- Alerting and incident management



Measurable data set health

Automation – are we adhering to the data health framework?

- Is owner valid?
- Is metadata filled?
- Are required validations added?
- Is monitoring in place?
- How many incidents were there and were they actioned?

Criteria	Score 0	Score 1	Score 2	Score 3	Score 4
Ownership	No owner OR invalid owner	Valid owner in legacy catalog	Valid owner defined in Unity Catalog ✓		
Cataloging	Uncatalogued	AWS Glue Catalog	Unity Catalog (not necessarily Medallion)	Unity Catalog + Medallion ✓	
Metadata	No metadata	Incomplete mandatory metadata in Unity Catalog	All mandatory metadata in Unity Catalog	All recommended metadata in Unity Catalog ✓	
Data Reliability	Dataset not monitored	Monitoring enabled < 100% Status Update Rate	Table level checks + 100% Status Update Rate	Field level or custom checks + 100% Status Update Rate	

Table Name	Ownership Score	Cataloging Score	Metadata Score	Data Reliability Score
bronze.internal.delivery_service_adclickdetails	0	3	1	1
bronze.internal.android_element_impression	2	3	2	2
bronze.internal.android_element_interaction	2	3	2	2
bronze.internal.ios_element_interaction	2	3	2	2
bronze.internal.ios_element_impression	2	3	2	2
bronze.internal.redirects	2	3	2	1
silver.advertising.click	2	3	3	2
silver.advertising.impression	2	3	3	2
silver.redirect.redirect	2	3	3	1
gold.travel_insight.redirect	2	3	2	1
gold.travel_insight.search	2	3	2	1

Deeper Dive: Scale data governance

Step 1: Stop the growth of hive

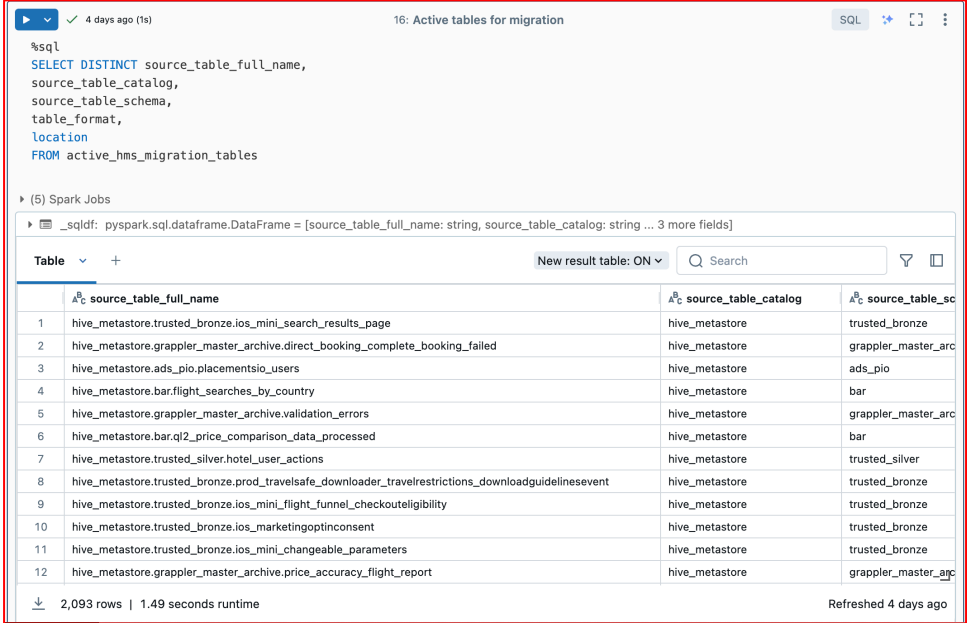
- Production standards
- Provide new UC catalog for ad-hoc & test data with easy access
- Block creation of new databases in hive
- Block creation of new tables in hive



Be careful with different modes of using and accessing data

Step 2: Define migration scope

- Databricks System tables
- hms_to_uc_migration schema provides access information for S3 locations and tables in Hive
- Scope reduction from 30k tables to 1.5k active tables
- Unlocking benefits of UC vs. Medallion compliance



```
%sql
SELECT DISTINCT source_table_full_name,
source_table_catalog,
source_table_schema,
table_format,
location
FROM active_hms_migration_tables
```

(5) Spark Jobs

↳ _sqlidf: pyspark.sql.dataframe.DataFrame = [source_table_full_name: string, source_table_catalog: string ... 3 more fields]

Table + New result table: ON Search

	source_table_full_name	source_table_catalog	source_table_sc
1	hive_metastore.trusted_bronze.ios_mini_search_results_page	hive_metastore	trusted_bronze
2	hive_metastore.grappler_master_archive.direct_booking_complete_booking_failed	hive_metastore	grappler_master_arc
3	hive_metastore.ads_pio.placementsio_users	hive_metastore	ads_pio
4	hive_metastore.bar.flight_searches_by_country	hive_metastore	bar
5	hive_metastore.grappler_master_archive.validation_errors	hive_metastore	grappler_master_arc
6	hive_metastore.bar.q12_price_comparison_data_processed	hive_metastore	bar
7	hive_metastore.trusted_silver.hotel_user_actions	hive_metastore	trusted_silver
8	hive_metastore.trusted_bronze.prod_travelsafe_downloader_travelrestrictions_downloadguidelineevent	hive_metastore	trusted_bronze
9	hive_metastore.trusted_bronze.ios_mini_flight_funnel_checkouteligibility	hive_metastore	trusted_bronze
10	hive_metastore.trusted_bronze.ios_marketingoptinconsent	hive_metastore	trusted_bronze
11	hive_metastore.trusted_bronze.ios_mini_changeable_parameters	hive_metastore	trusted_bronze
12	hive_metastore.grappler_master_archive.price_accuracy_flight_report	hive_metastore	grappler_master_arc

2,093 rows | 1.49 seconds runtime Refreshed 4 days ago

Step 3: Migration

Challenges

- Decentralized physical storage
 - Multiple S3 buckets
 - Different owners
 - Permissions
- Data not always stored in Delta format
- DBR versions incompatible with Unity Catalog
 - Must be ≥ 11.3
- Code-free migration for data consumers

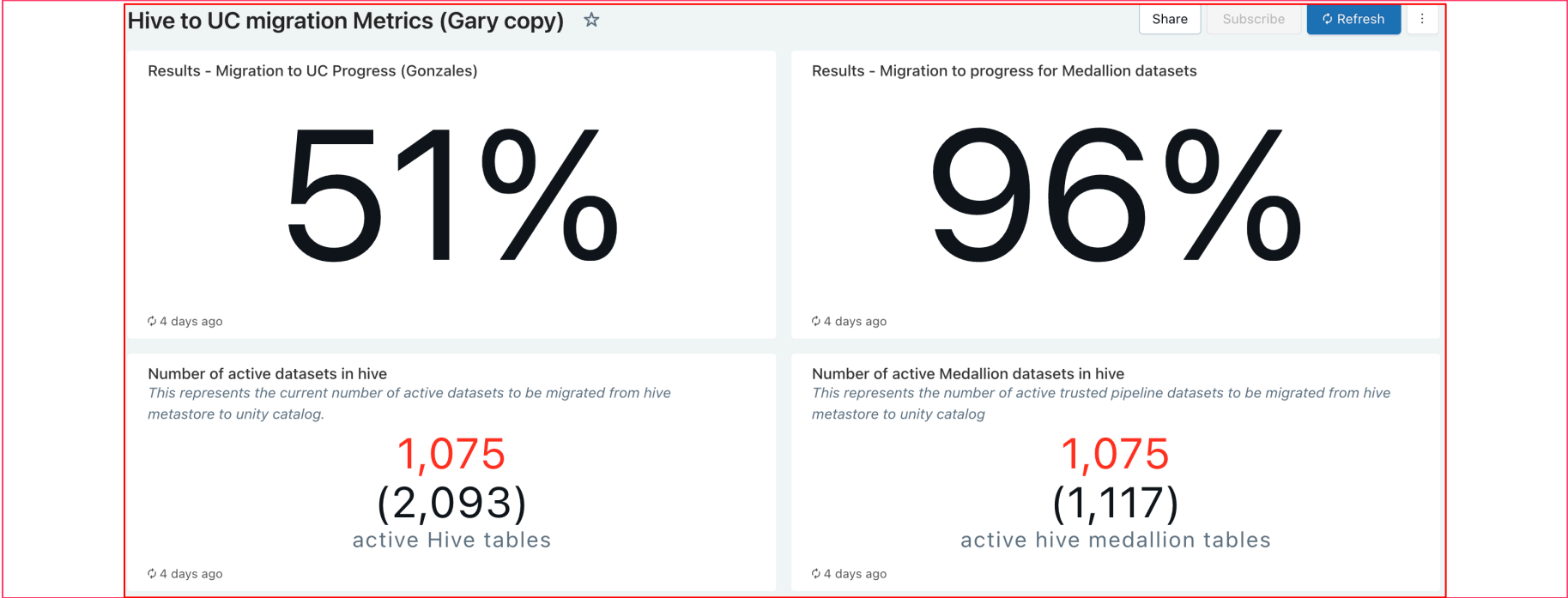
Step 3: Migration

Strategy

- DBR version upgrade across the organization
- Per table:
 - Update UC IAM role with read/write permissions
 - Create external location in UC
 - Create external table in UC
 - Setup permissions for access via Unity Catalog
- Update default catalog for everyone

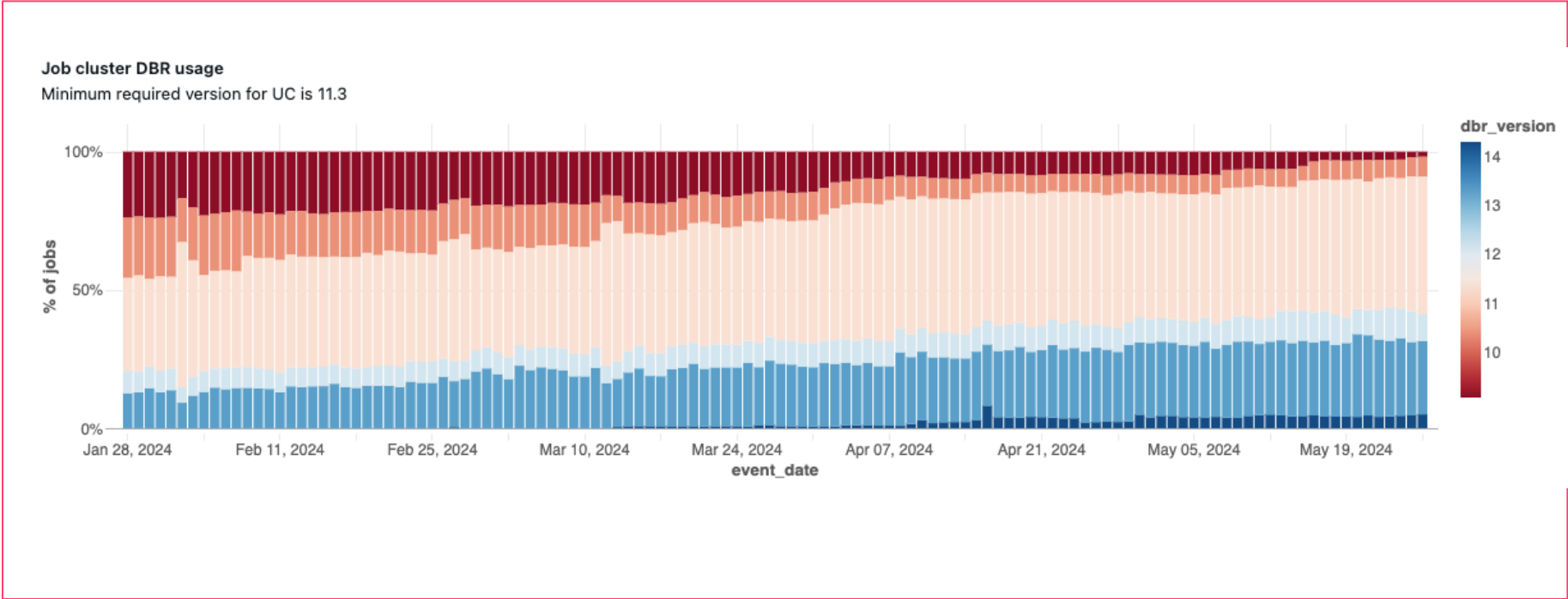
Make our progress visible

Customized hms_to_uc_migration dashboard



Make our progress visible

Databricks Runtime Upgrade progress

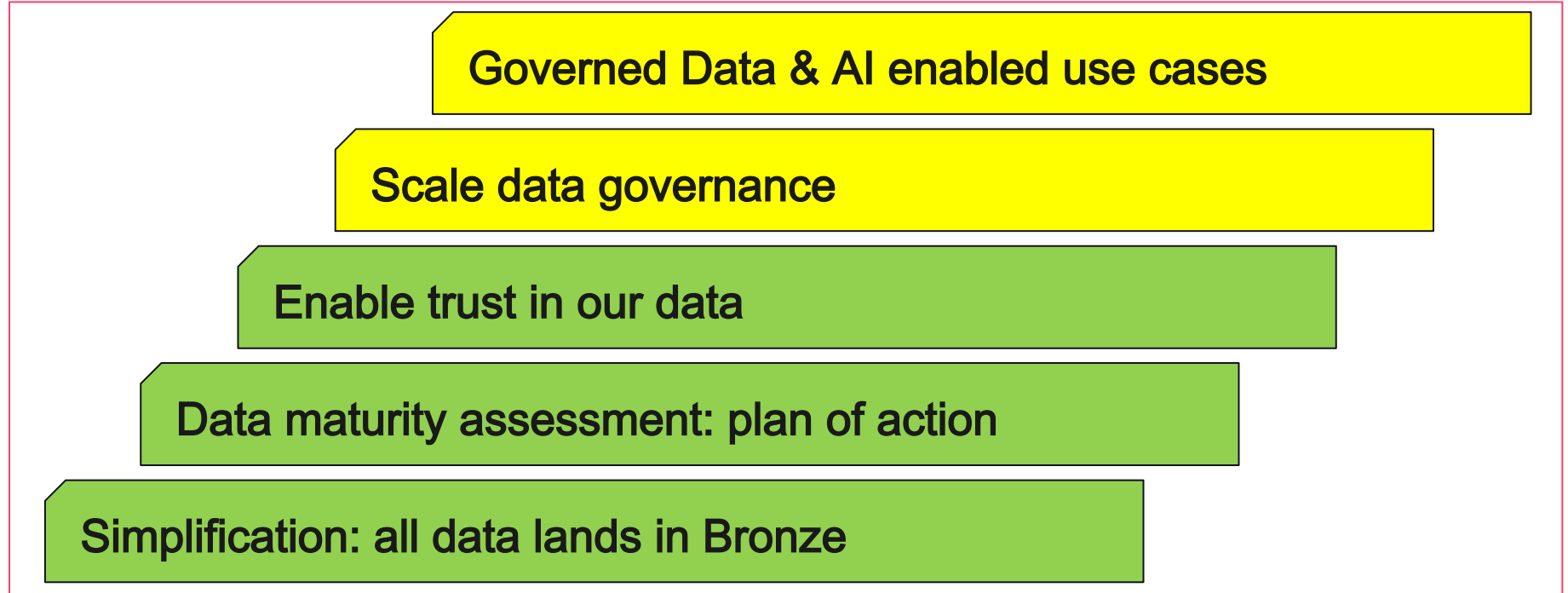


Are
we
there
yet?



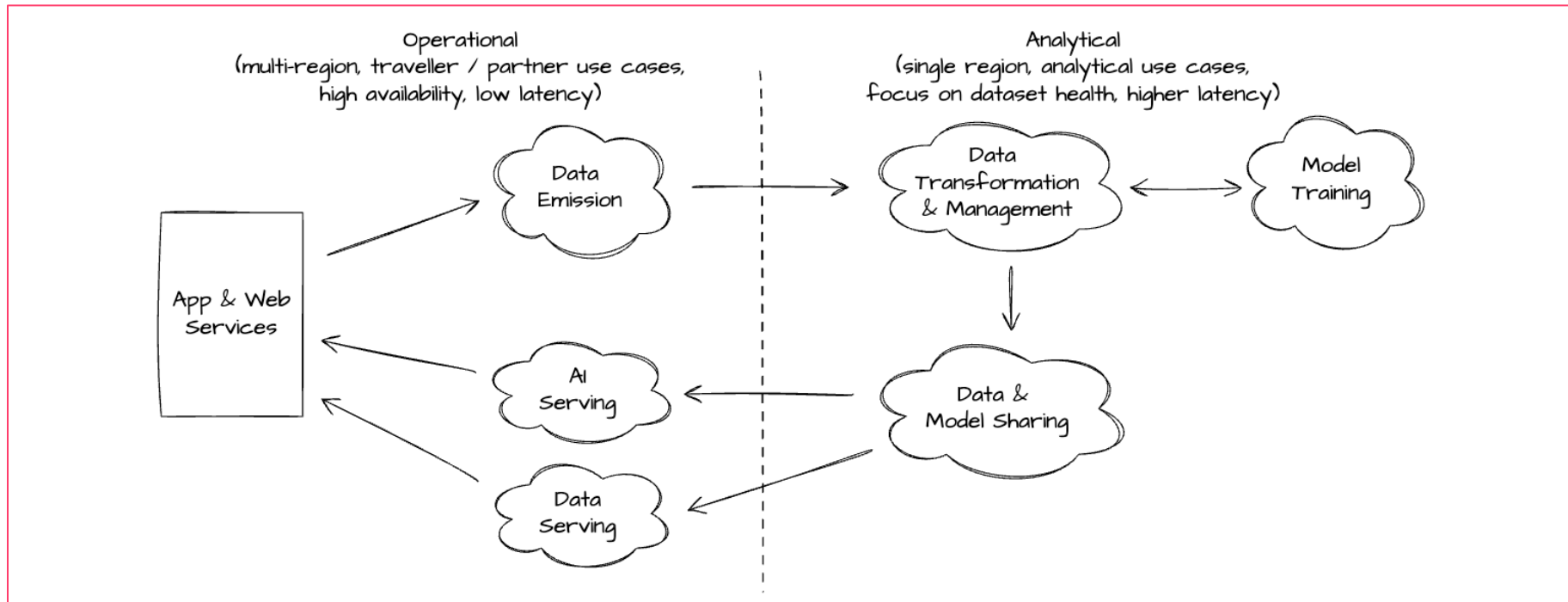
Skyscanner data journey

Change takes time so we need to find ways to accelerate



The Data and AI flywheel

Data is powering Analytical and Operational use cases



What did we find useful?

Practical data governance at scale

- Learn from others. What does good look like?

What did we find useful?

Practical data governance at scale

- Learn from others. What does good look like?
- Not all data is equal. What is business critical?

What did we find useful?

Practical data governance at scale

- Learn from others. What does good look like?
- Not all data is equal. What is business critical?
- Unity Catalog v Medallion

What did we find useful?

Practical data governance at scale

- Learn from others. What does good look like?
- Not all data is equal. What is business critical?
- Unity Catalog v Medallion
- Measure and make progress visible

What did we find useful?

Practical data governance at scale

- Learn from others. What does good look like?
- Not all data is equal. What is business critical?
- Unity Catalog v Medallion
- Measure and make progress visible
- Be clear with priorities and asks of non-data teams

Thank you

