DATA+AI SUMMIT
BY databricks

RIVER TRASH DETECTION ON DATABRICKS WITH

THE OCEAN CLEANUP

Michael Berk (RSA) & Patrick Leahey (RSA)
6.12.24

1 - Background

Your Presenters

- Data science background
- ~2 years at Databricks
- Studied environmental science
- Passionate about the oceans

- Data engineering background
- ~1 years at Databricks
- Studied math & computer science
- Passionate about the environment

Databricks for Good

Volunteer Databricks Initiative

- 3 verticals
  - Education
  - Foreign Aid
  - Environment
- Pro-bono
- Databricks employees working nights/weekends

What did we do?

1. Scoping
2. Projects
3. Marketing fluff

What did we do?

1. Scoping
    a. Reliability + stability of pipelines
2. Projects
3. Marketing fluff

What did we do?

1. Scoping
   a. Reliability + stability of pipelines
2. Projects
   a. Architecture review and setup
   b. RMS Pipeline
   c. Ingestion framework template
3. Marketing fluff

What did we do?

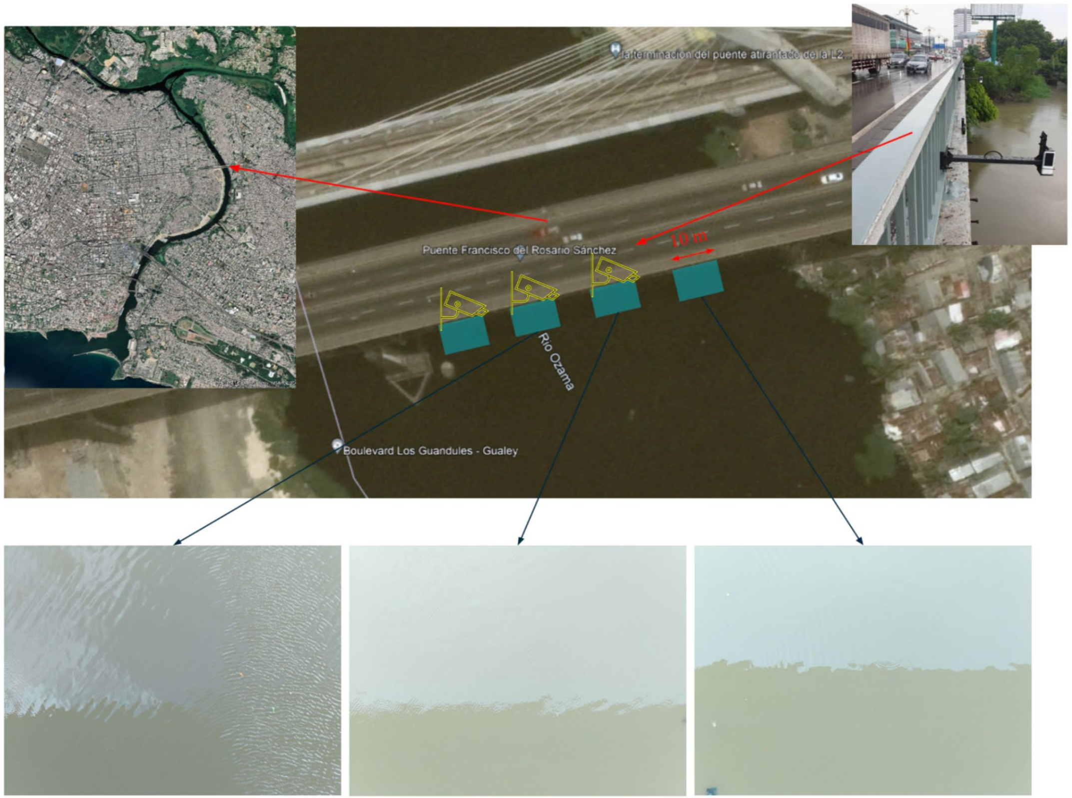
1. Scoping
   a. Reliability + stability of pipelines
2. Projects
   a. Architecture review and setup
   b. RMS Pipeline
   c. Ingestion framework template
3. Marketing fluff
   a. Blog post
   b. Data + AI Summit Talk
   c. Award?

2 - The Ocean Cleanup

SENSORS CAPTURE DATA FROM RIVERS

River Monitoring Systems

- Mounted cameras
    - >50% river width
    - 2-3 images / 15 min
- Water level sensors



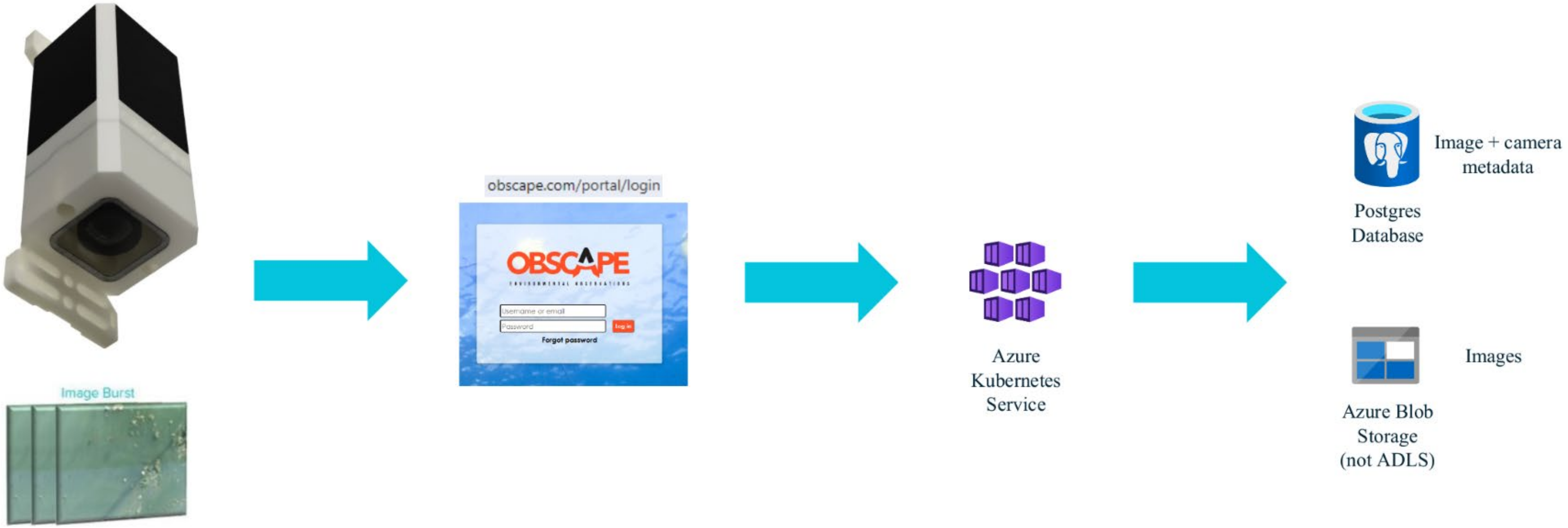
RMS IS STRATEGICALLY DEPLOYED

Goal: Global coverage

Current Footprint

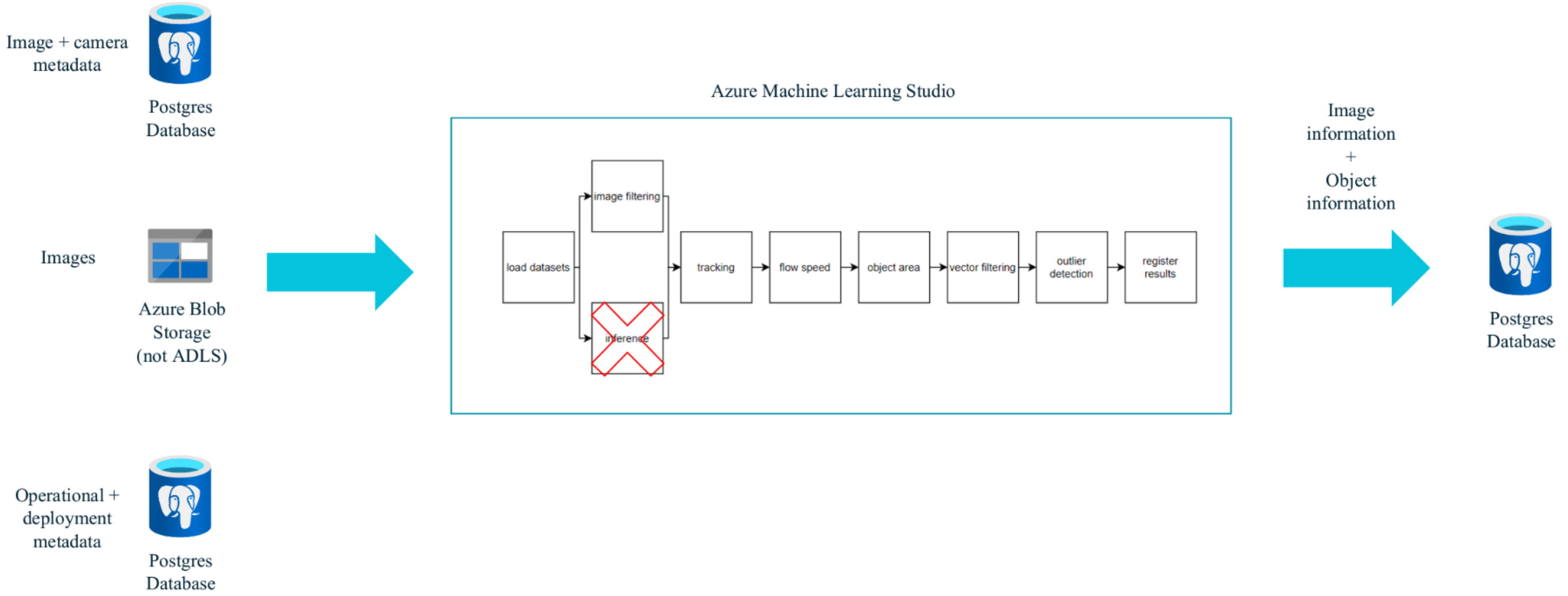
- 12 RMS deployments
- 62 cameras installations
- 130K images/week

IMAGES AND METADATA ARE INGESTED



COMPUTER VISION TO TRACK TRASH

Weekly, per-camera pipelines process images



MIGRATING TO DATABRICKS

BRIDGING THE GAPS

Main objective: Migrate to Databricks

Incremental

- Exactly once
- Reduce cost
- Increase efficiency
- Custom state management solution

Dynamically Distributed

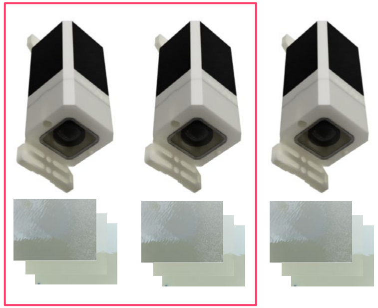
- Parallelize existing Pandas code
- Increase efficiency
- Reduce complexity

INCREMENTAL

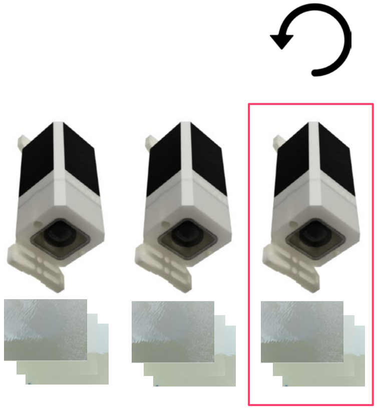
DESIRED PIPELINE BEHAVIOR

Process images exactly once by default, but support:

Process images for a subset of cameras

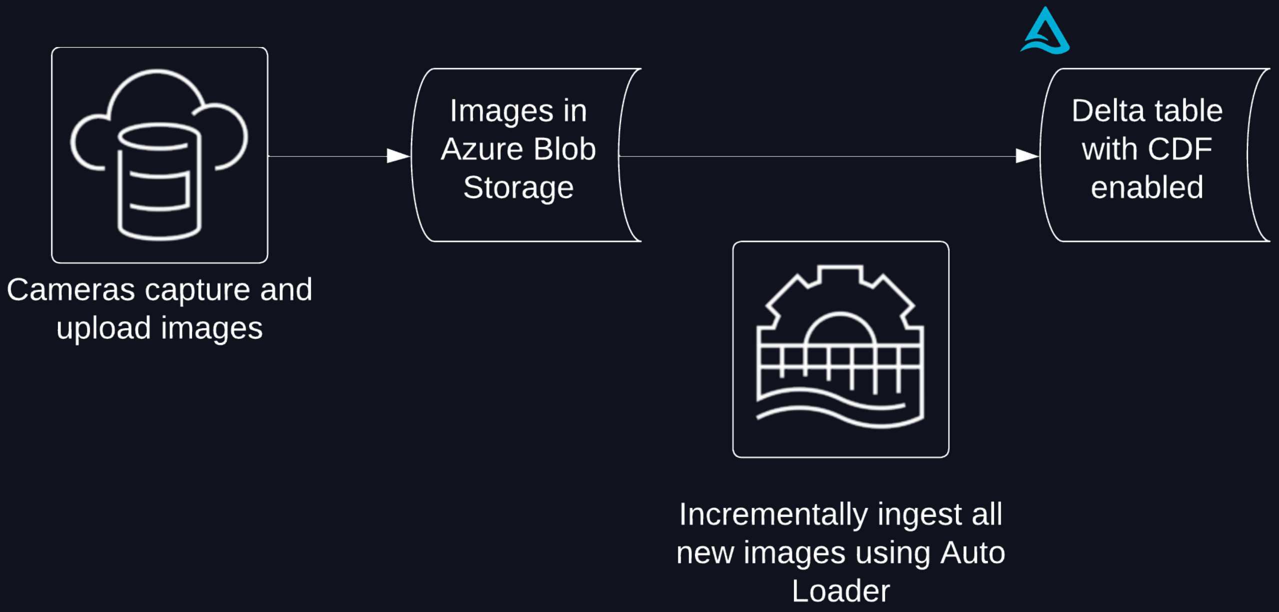


Reprocess images



INCREMENTAL METADATA INGESTION

Using Auto Loader and Change Data Feed (CDF)



INCREMENTAL METADATA INGESTION

Using Auto Loader and Change Data Feed (CDF)

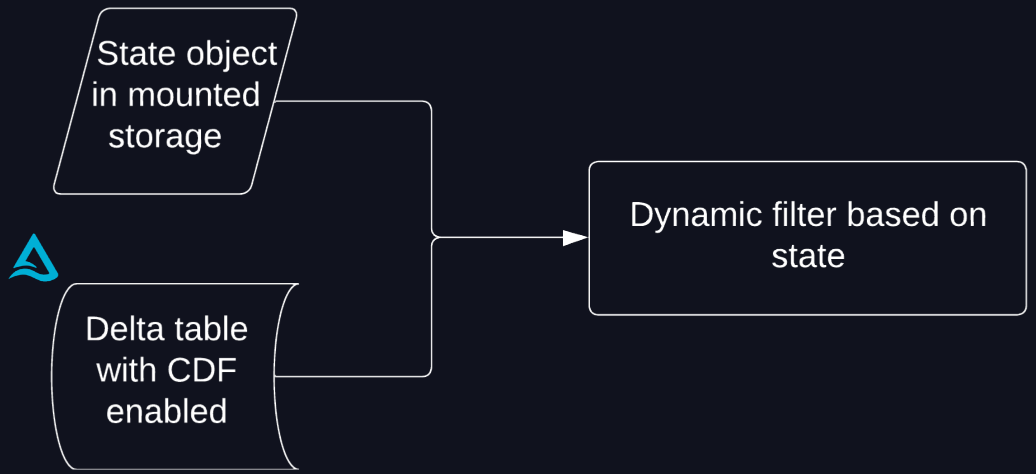
PYTHON

```
(spark
    .readStream
    .format("cloudFiles")
    .option("cloudFiles.format", "binaryFile")
    .option('pathGlobFilter', '*.jpg')
    .option("recursiveFileLookup", "true")
    .load(f"wasbs://{container}@{storage_account}.blob.core.windows.net/{image_path}")
    .drop("content", "length")
    .withColumn("camera_serial", regexp_extract("path", "^.*?/.*?/.*?/.*?/(.*?)/", 1))
    .writeStream
    .option("checkpointLocation", checkpoint_path)
    .trigger(availableNow=True)
    .toTable(image_metadata_path)
)
```

METADATA-DRIVEN FILTERING

With custom state management

```
{
    "id_1": 20,
    "id_2": 20,
    "id_3": 10
}
```



METADATA-DRIVEN FILTERING

With custom state management

PYTHON

```
filter_conds = [
    (col("camera_serial") == camera_serial) & (col("_commit_version") >
    (0 if reprocess else data.get(camera_serial, 0)))
    for camera_serial in camera_serials
]

image_metadata_df = (
    spark
        .read
        .format("delta")
        .option("readChangeFeed", "true")
        .option("startingVersion", 0)
        .table(bronze_image_table_path)
        .filter(reduce(lambda x, y: x | y, filter_conds))
)
```

METADATA READY TO DRIVE PROCESSING

	path	modificationTime	camera_serial	image_name	date
1	› wasbs://attachments...	2024-04-12T08:22:42.000	PTM5165	rms_mella_4_ptm5165_20240301T1100_1.jpg	2024-03-01
2	› wasbs://attachments...	2024-04-12T08:22:47.000	PTM5165	rms_mella_4_ptm5165_20240301T1100_2.jpg	2024-03-01
3	› wasbs://attachments...	2024-04-12T08:22:49.000	PTM5165	rms_mella_4_ptm5165_20240307T1720_1.jpg	2024-03-07
4	› wasbs://attachments...	2024-04-12T08:22:49.000	PTM5165	rms_mella_4_ptm5165_20240305T1450_2.jpg	2024-03-05
5	› wasbs://attachments...	2024-04-12T08:22:38.000	PTM5165	rms_mella_4_ptm5165_20240305T1450_1.jpg	2024-03-05
6	› wasbs://attachments...	2024-04-12T08:22:35.000	PTM5165	rms_mella_4_ptm5165_20240302T1640_2.jpg	2024-03-02
7	› wasbs://attachments...	2024-04-12T08:22:41.000	PTM5165	rms_mella_4_ptm5165_20240303T1740_1.jpg	2024-03-03

DYNAMICALLY DISTRIBUTED

WE CONSIDERED THREE OPTIONS

Ray on Spark	Pandas UDFs	Pandas function APIs
• Distributed framework • Logical partitioning • Parallelize over nay iterable • GA on Databricks • Overhead • Lack of familiarity	• Native Spark • Vectorized Spark-> Pandas transformations via Apache Arrow • Apply Python function to PySpark columns, DataFrames • GROUPED_MAP supports Pandas DataFrame -> Pandas DataFrame on grouped data	• Native Spark • Vectorized Spark-> Pandas transformations via Apache Arrow • Apply Python function to PySpark DataFrames • .applyInPandas() supports Pandas DataFrame -> Pandas DataFrame on grouped data • Higher-level API

INCREMENTAL METADATA INGESTION

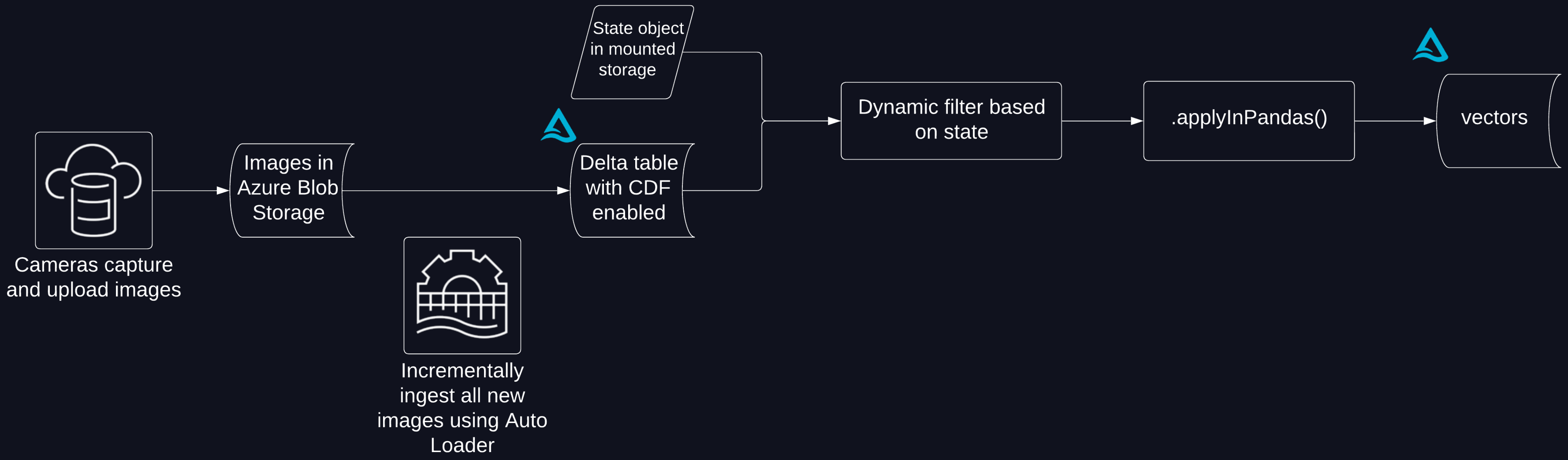
Using Auto Loader and Change Data Feed (CDF)

PYTHON

```python
def apply_pipeline(key, pdf):
    filtered_images = image_filtering(pdf)
    inference_results = batch_inference(filtered_images)
    tracks = object_tracking(inference_results)
    flow_speed = calculate_flow_speed(tracks)
    object_area = calculate_object_area(flow_speed)
    return vector_filtering(object_area)

results = (
    image_metadata_df
        .groupBy("camera_serial", "date")
        .applyInPandas(apply_pipeline, schema)
)
```

END TO END ARCHITECTURE



3 - Results

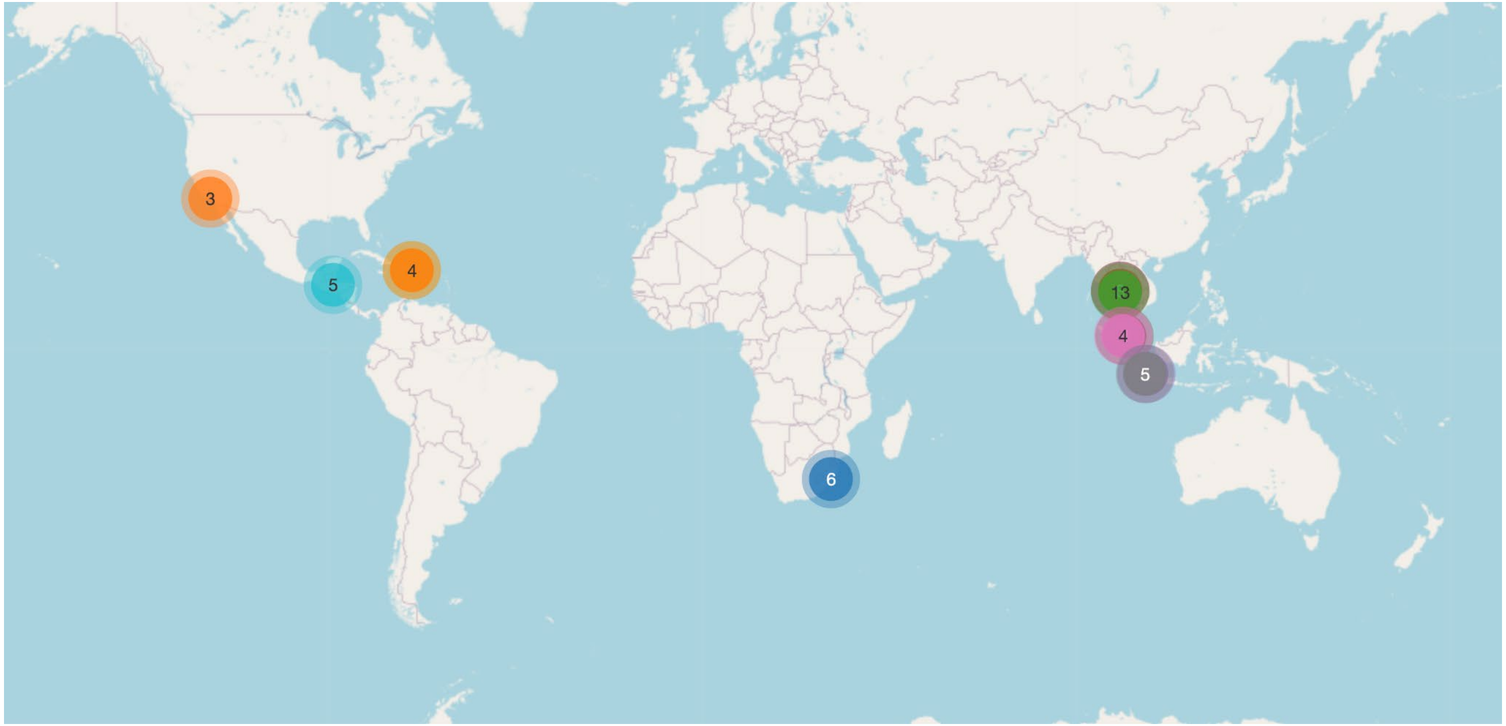
Outcomes

Open source highly-parallelizable ETL

1. Consolidated tooling
2. Incremental processing
3. Databricks stack

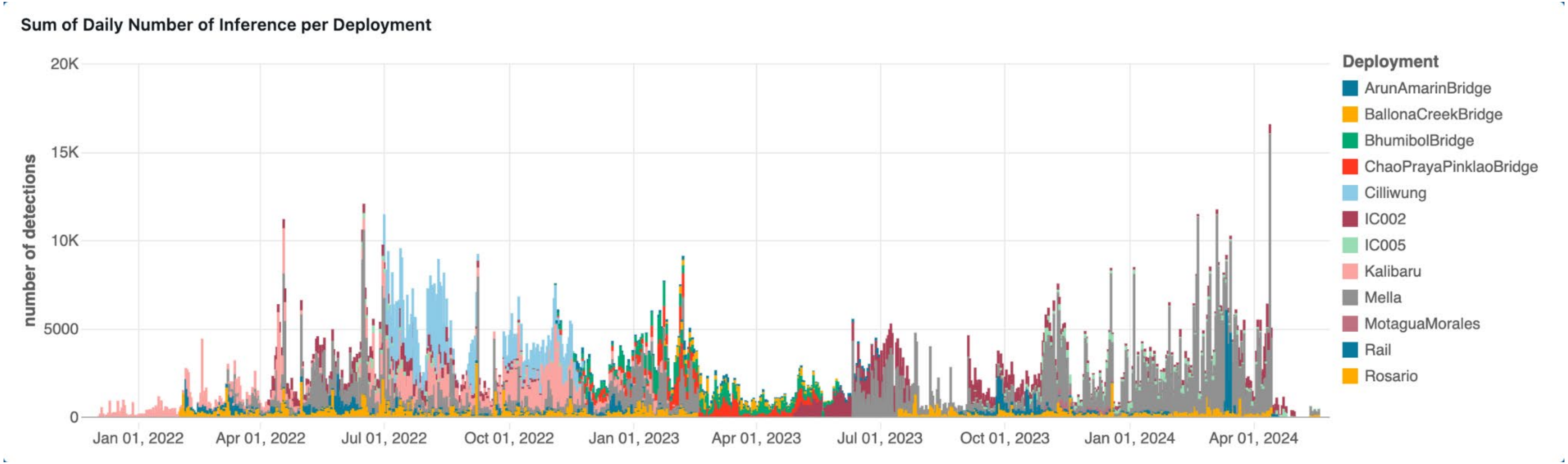
Camera Monitoring Dashboard (KPI)

DBSQL Legacy Dashboard



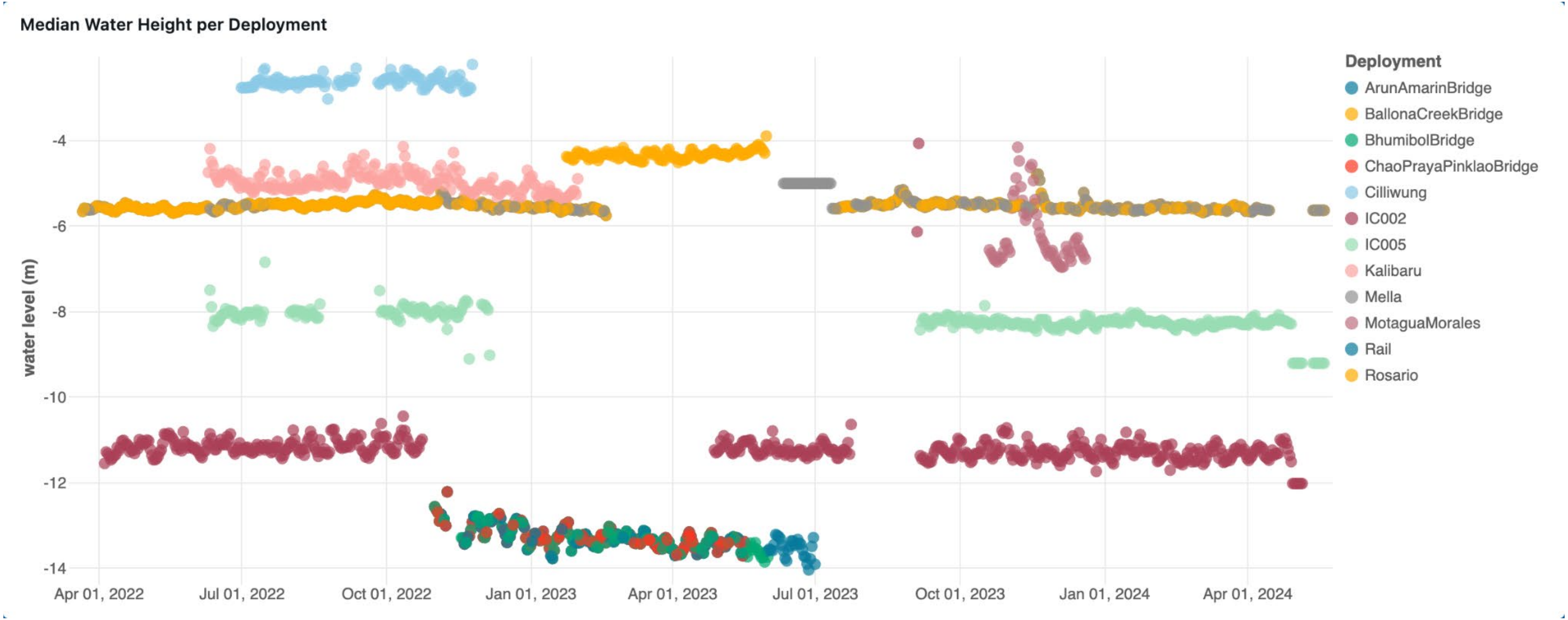
Total Plastic Detections (KPI)

DBSQL Dashboard + GenAI Assistant



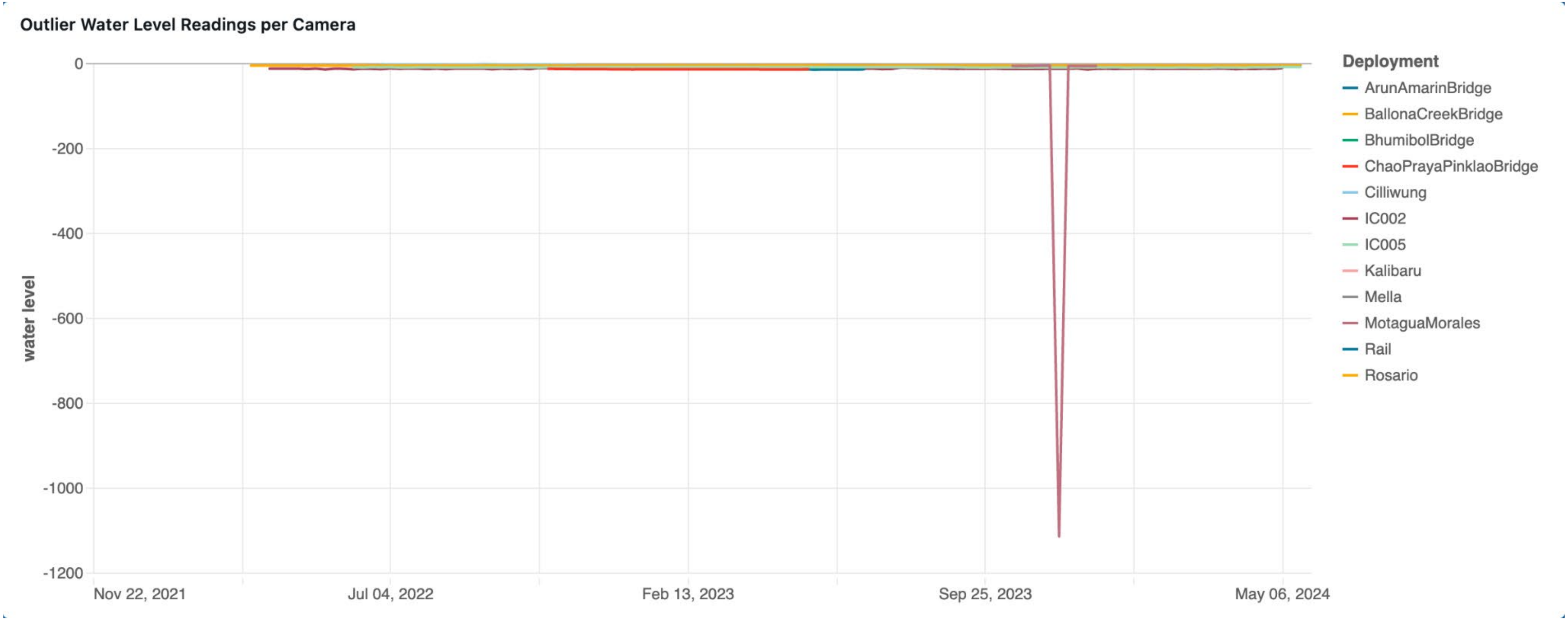
Water Level (Descriptive Stats)

DBSQL Dashboard + GenAI Assistant



Water Level (Outlier Detection)

DBSQL Dashboard + GenAI Assistant



Next Steps

Let the Databricks product speak for itself.

Opportunities

- Unification of tooling
- Power
- UC

Challenges

- Designing organizational policies for scaling
- Migration
- Employee education

Thank You!