

# A lakehouse for photovoltaic and wind data: Developing with Delta Live Tables and Databricks Asset Bundles

Andreas Haselsteiner

Databricks User Group Vienna #3  
January 22, 2025



# VERBUND AG & VERBUND Green Power at a glance

VERBUND AG

ca. **3,800**  
employees<sup>1</sup>

**12**  
countries<sup>2</sup>

ca. **33**  
TWh electricity<sup>3</sup>

VERBUND Green  
Power GmbH

ca. **170**  
employees

**6**  
countries

ca. **2**  
TWh electricity<sup>3</sup>

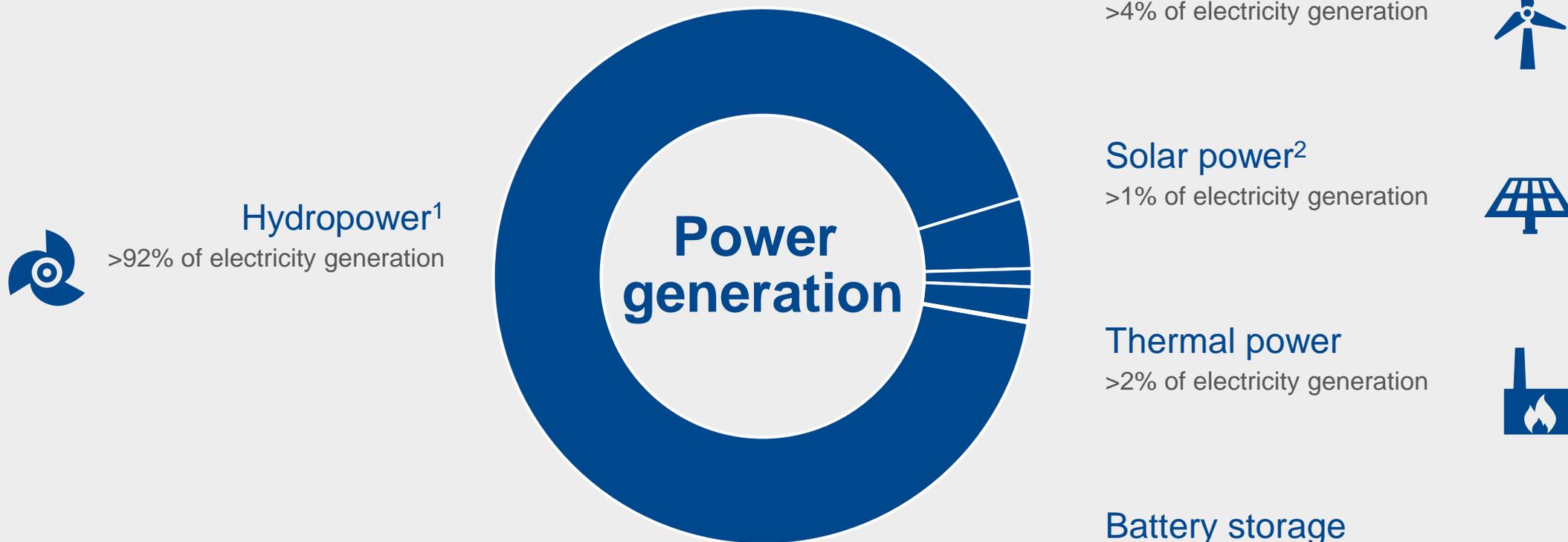


<sup>1</sup> Verbund AG, Integrierter Geschäftsbericht 2023, page 3, [https://www.verbund.com/-/media/verbund/ueber-verbund/investor-relations/finanzpublikationen/de/2024/verbund-integrierter\\_geschaeftsbericht\\_2023\\_deutsch.ashx](https://www.verbund.com/-/media/verbund/ueber-verbund/investor-relations/finanzpublikationen/de/2024/verbund-integrierter_geschaeftsbericht_2023_deutsch.ashx)

<sup>2</sup> <https://www.verbund.com/en-at/about-verbund/company>

<sup>3</sup> Verbund AG, Integrierter Geschäftsbericht 2023, page 4, [https://www.verbund.com/-/media/verbund/ueber-verbund/investor-relations/finanzpublikationen/de/2024/verbund-integrierter\\_geschaeftsbericht\\_2023\\_deutsch.ashx](https://www.verbund.com/-/media/verbund/ueber-verbund/investor-relations/finanzpublikationen/de/2024/verbund-integrierter_geschaeftsbericht_2023_deutsch.ashx)

# VERBUND generates 98% from renewable energy sources



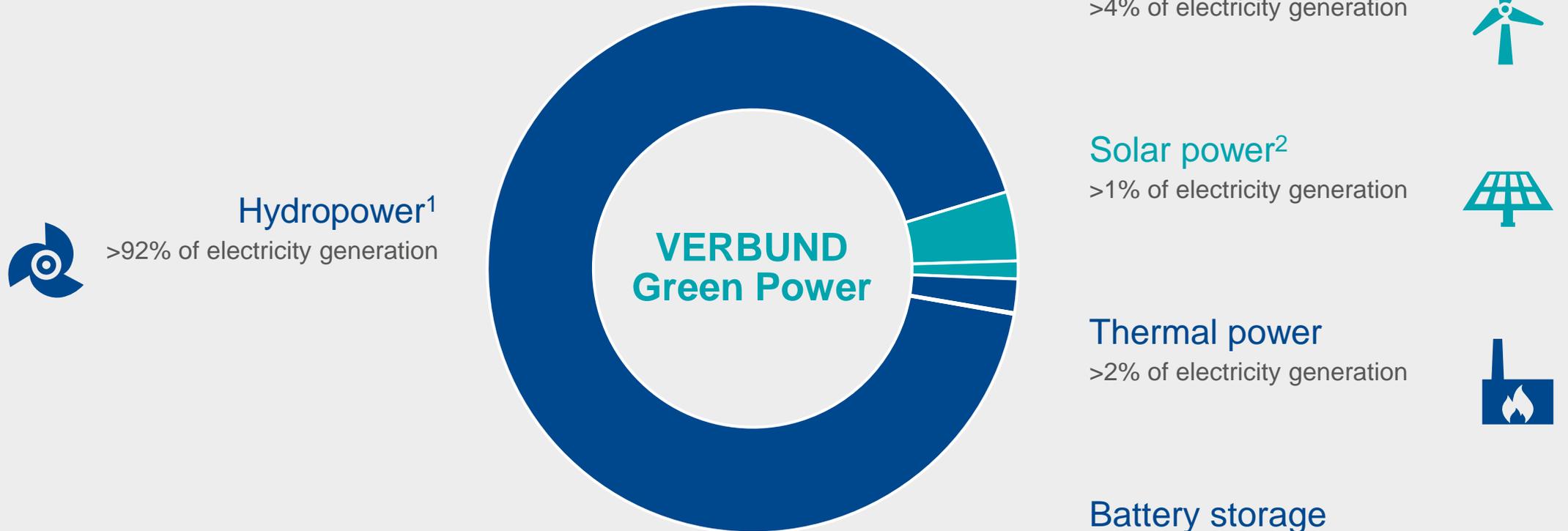
1 including options; without fully-consolidated plants (Ashta 1&2 and Nussdorf)

2 without leasing/contracting plants

All figures actual generation 2023



# VERBUND generates 98% from renewable energy sources



 **Hydropower<sup>1</sup>**  
>92% of electricity generation

**Wind power**  
>4% of electricity generation



**Solar power<sup>2</sup>**  
>1% of electricity generation



**Thermal power**  
>2% of electricity generation

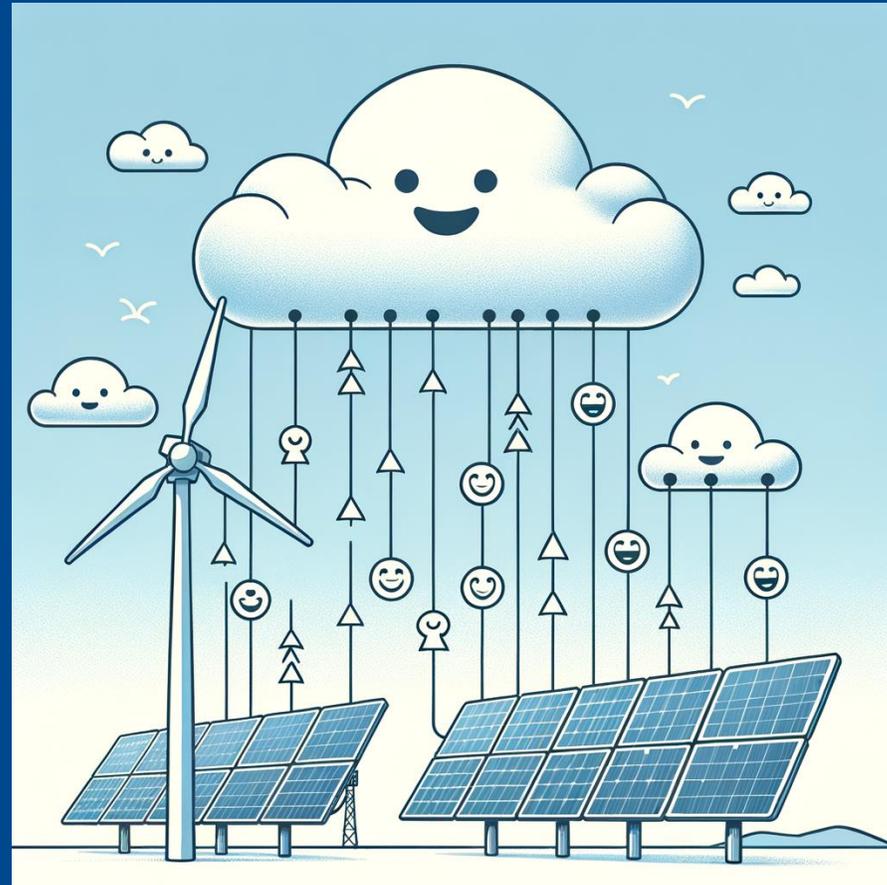


**Battery storage**  
<1% of electricity generation



<sup>1</sup> including options; without fully-consolidated plants (Ashta 1&2 and Nussdorf)  
<sup>2</sup> without leasing/contracting plants  
All figures actual generation 2023

Wind turbines and PV plants produce plenty of interesting data ...



... that need a modern lakehouse somewhere up in the clouds

# For our lakehouse version 2, we use two new Databricks features

Delta Live Tables (DLTs) became generally available in 2022<sup>1</sup>

Databricks Asset Bundles (DABs) became generally available in 2024<sup>2</sup>

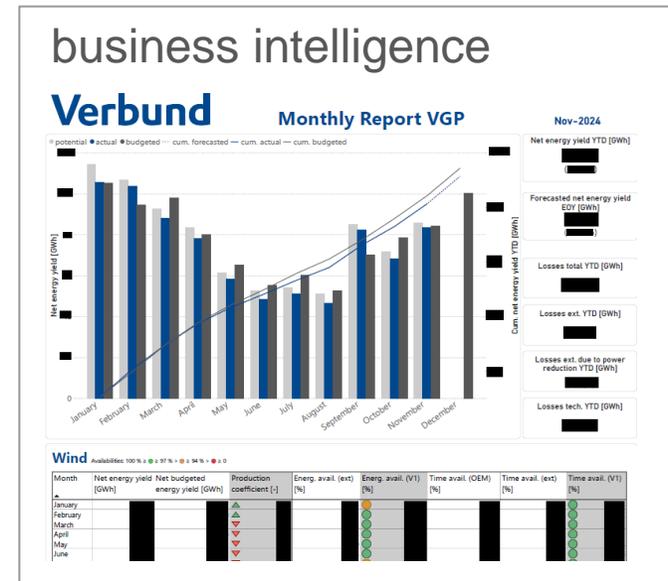
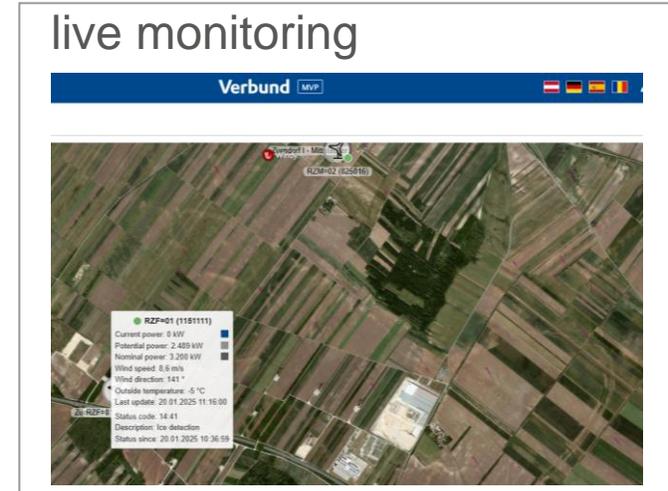
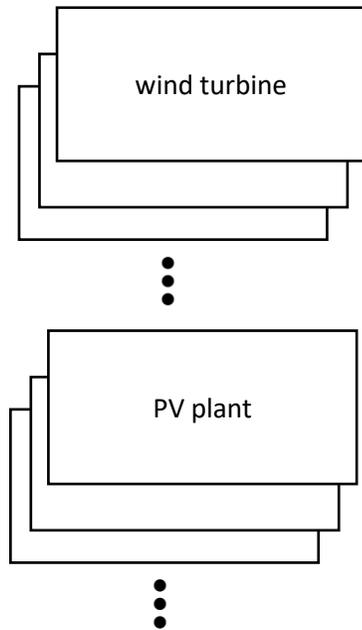
```
@dlt.table
def my_silver_table() -> DataFrame:
    return (
        dlt.read_stream("my_bronze_table").
        transform(transform_to_silver)
    )
```

```
databricks bundle deploy -t dev
```

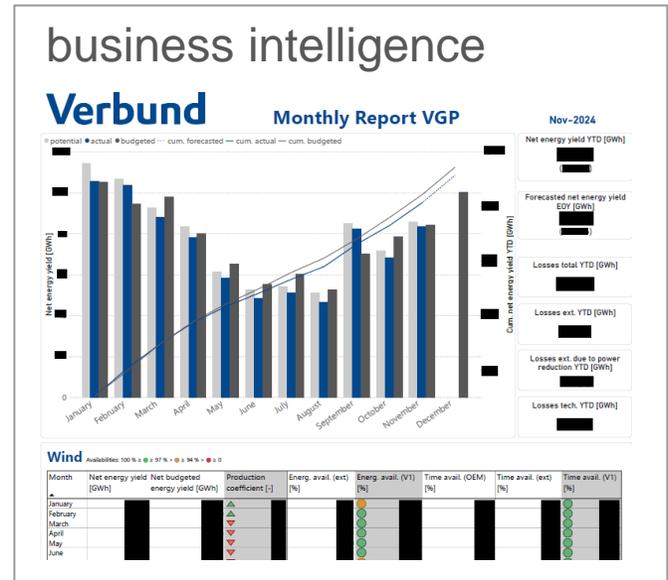
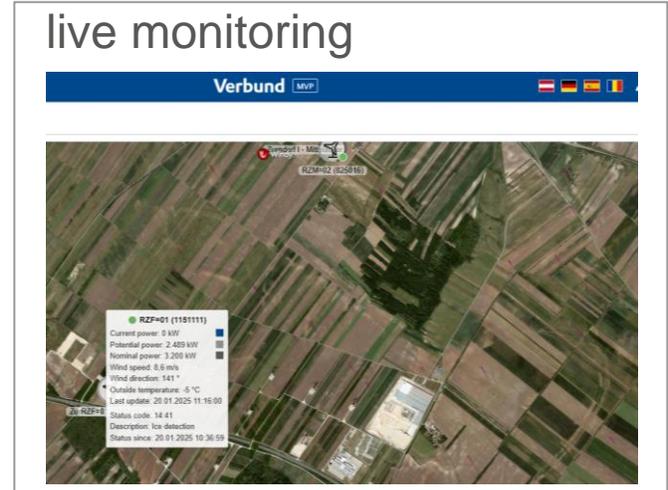
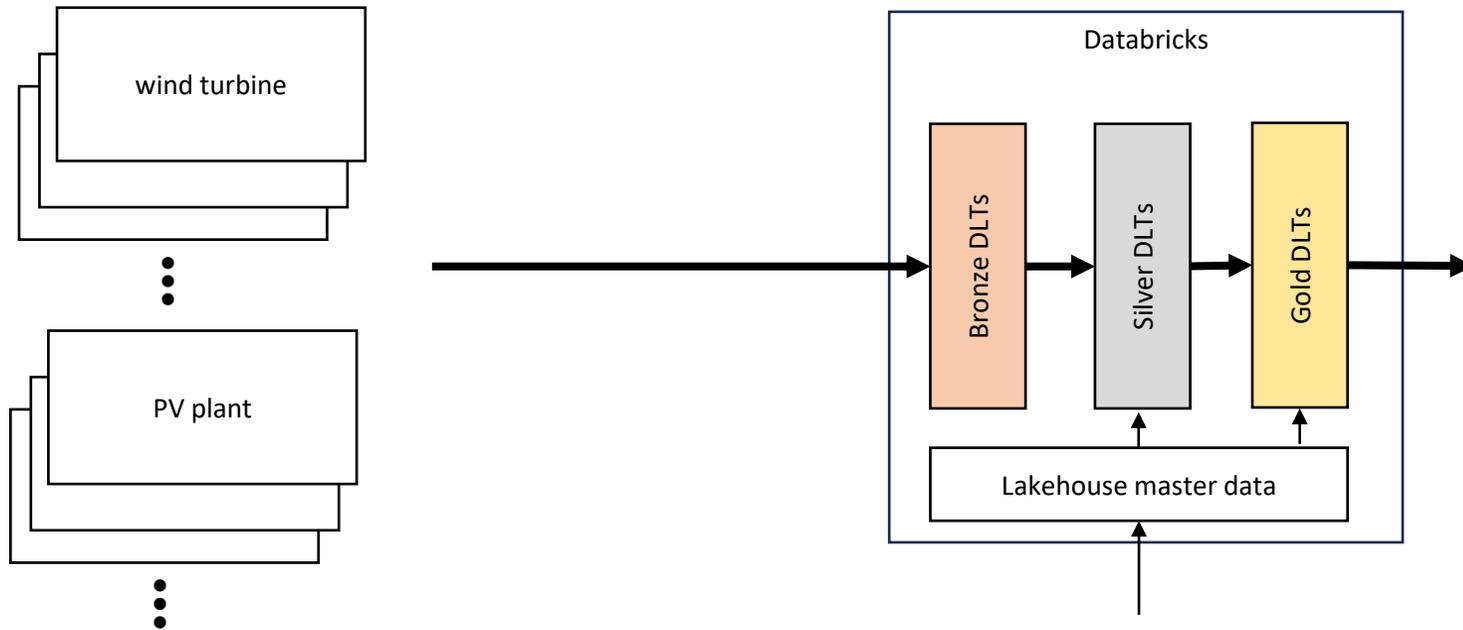
# Agenda

1. Our use case: Operational plant data for live monitoring and BI reporting
2. Delta Live Tables: How we use it to stream through bronze, silver, and gold layers
3. Databricks Asset Bundles: How we use it in our development process
4. What's great and what isn't

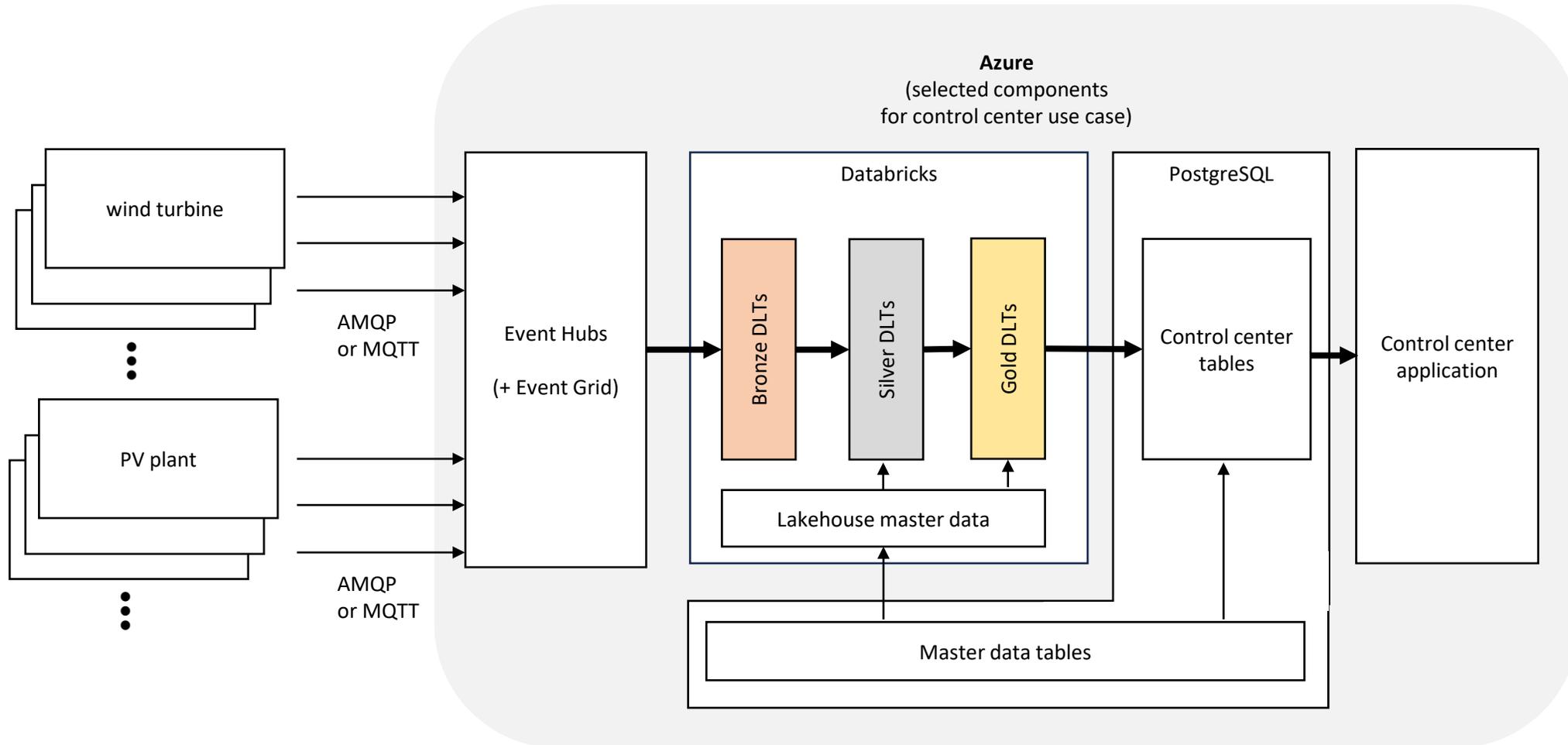
# Our use case: Operational plant data for live monitoring and BI reporting



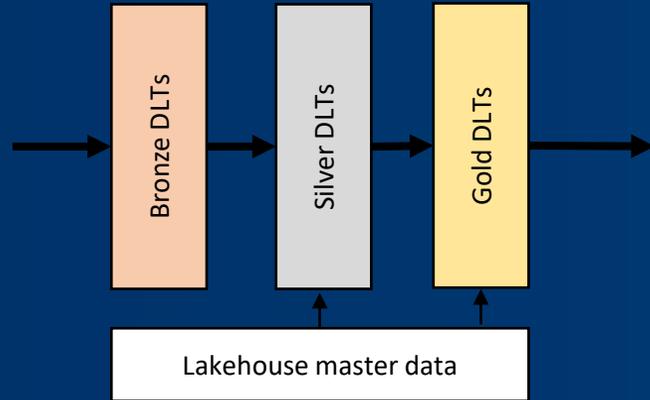
# All data are streamed through Databricks



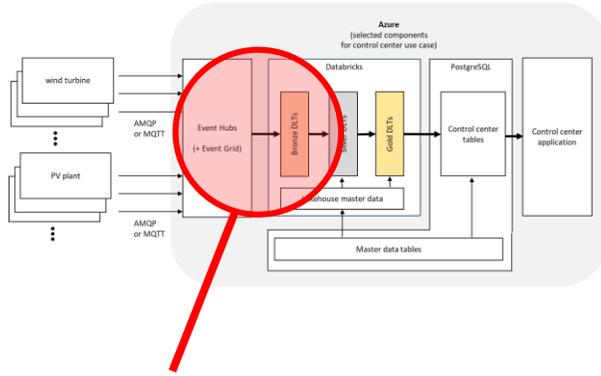
# We use message protocols to send data towards the cloud and ingest these data to Databricks from Event Hubs



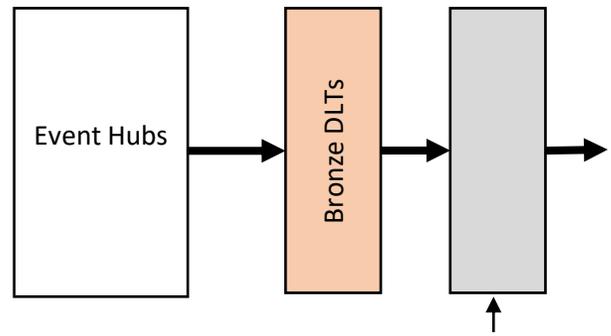
# Delta Live Tables



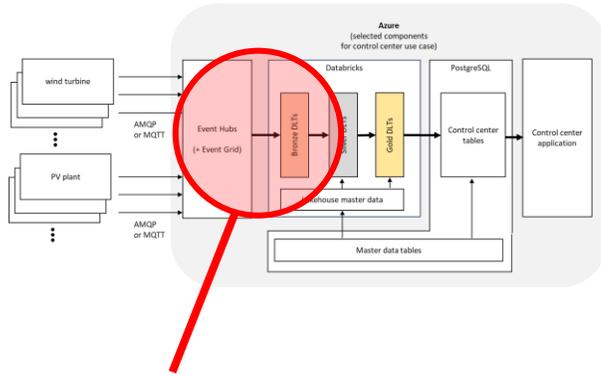
# DLT makes ingesting messages into a table straight-forward



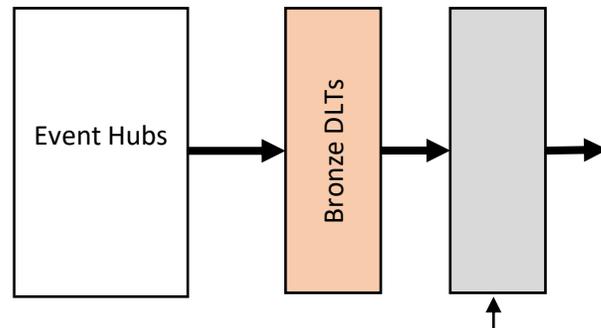
```
@dlt.table
def my_bronze_table() -> DataFrame:
    return (
        spark.readStream.format("kafka")
            .options(**kafka_options) # Contains Event Hub name and credentials
            .load()
            .transform(parse_message)
    )
```



# DLT makes ingesting messages into a table straight-forward



```
@dlt.table
def my_bronze_table() -> DataFrame:
    return (
        spark.readStream.format("kafka")
            .options(**kafka_options) # Contains Event Hub name and credentials
            .load()
            .transform(parse_message)
    )
```



```
def parse_message(df: DataFrame) -> DataFrame:
    return df.selectExpr(
        "cast(timestamp as timestamp) as _eh_enqueued_ts",
        "current_timestamp() as _inserted_at_ts",
        "cast(value as string) as payload", # Actual message
    )
```

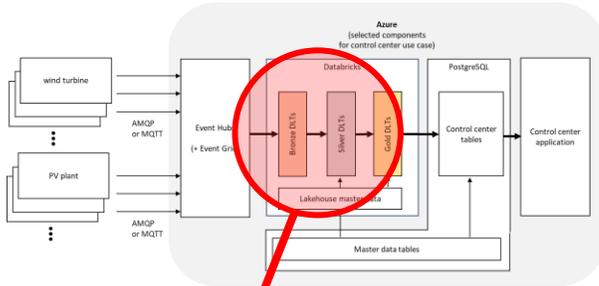
# Our bronze table holds raw JSON messages

my\_bronze\_table

<code>_eh_enqueued_ts<sup>1</sup></code>	<code>_inserted_at_ts</code>	<code>payload</code>
2025-01-22T17:40:12.371+00:00	2025-01-22T17:40:14.422+00:00	<pre>{   "turbine_id": "RG15",   "measurement_ts": "2025-01-22T17:40:11.371894Z",   "data": {     "WTUR.W": 67000,     "WMET.HorWdSpd": 4.3,     "WMET.HorWdDir": 134.5,     ...   } }</pre>
2025-01-22T17:40:42.560+00:00	2025-01-22T17:40:44.119+00:00	<pre>{   "turbine_id": "RG15",   "measurement_ts": "2025-01-22T17:40:41.050264Z",   "data": {     "WTUR.W": 67000,     "WMET.HorWdSpd": 4.4,     "WMET.HorWdDir": 137.1,     ...   } }</pre>



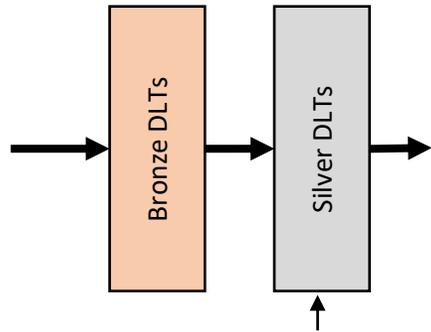
# Append-only operations like extracting and casting JSON values work well



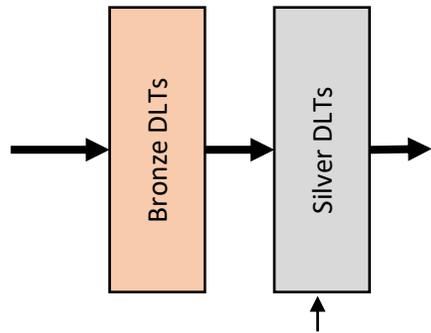
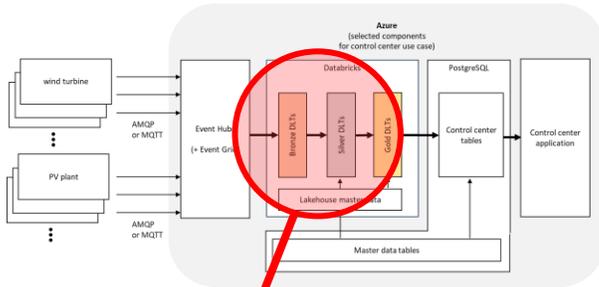
```
@dlt.table
def my_silver_table() -> DataFrame:
    return dlt.read_stream("my_bronze_table").transform(transform_bronze_to_silver)
```

my\_bronze\_table

_eh_enqueued_ts	_inserted_at_ts	payload
		{ "turbine_id": "RG15", "measurement_ts": "2025-01-22T17:40:11.371894Z", "data": { "WTUR.W": 67000, "WMET.HorWdSpd": 4.3, "WMET.HorWdDir": 134.5, ... } }



# Append-only operations like extracting and casting JSON values work well



```
@dlt.table
def my_silver_table() -> DataFrame:
    return dlt.read_stream("my_bronze_table").transform(transform_bronze_to_silver)
```

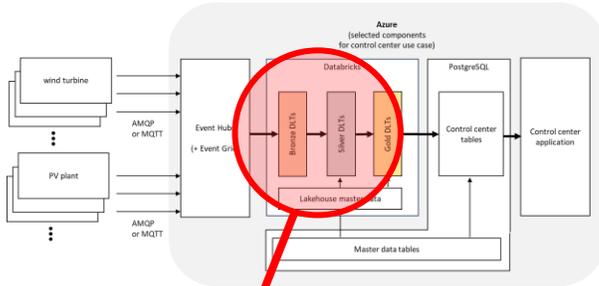
```
from pyspark.sql import functions as F
```

```
def transform_bronze_to_silver(df: DataFrame) -> DataFrame:
    """Transforms the DataFrame to [turbine_id, measurement_ts, wind_speed_ms]"""

    expression_for_id = F.get_json_object("payload", "$['turbine_id']").alias(
        "turbine_id"
    )
    expression_for_timestamp = (
        F.get_json_object("payload", "$['measurement_ts']")
        .cast(TimestampType())
        .alias("measurement_ts")
    )
    expression_for_wind_speed = (
        F.get_json_object("payload", "$['data']['WMET.HorWdSpd']")
        .cast(DoubleType())
        .alias("wind_speed_ms")
    )

    return df.select(
        expression_for_id, expression_for_timestamp, expression_for_wind_speed
    )
```

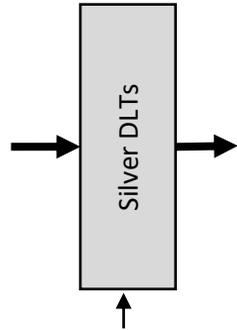
# Append-only operations like extracting and casting JSON values work well



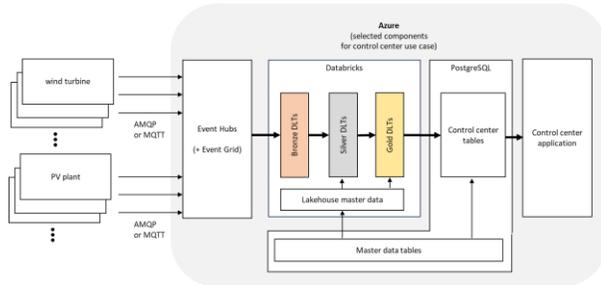
```
@dlt.table  
def my_silver_table() -> DataFrame:  
    return dlt.read_stream("my_bronze_table").transform(transform_bronze_to_silver)
```

my\_silver\_table

turbine_id	measurement_ts	wind_speed_ms
RG15	2025-01-22T17:40:11.371894+00:00	4.3
RG15	2025-01-22T17:40:41.050264+00:00	4.4



# Removing, updating, and aggregating rows can be tricky and slow



```
@dlt.table
def my_output_table() -> DataFrame:
    return dlt.read_stream("my_input_table").transform(remove_duplicated_rows)
```

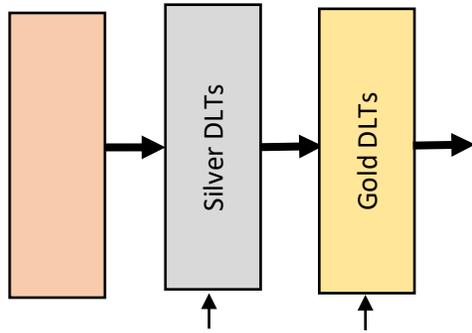
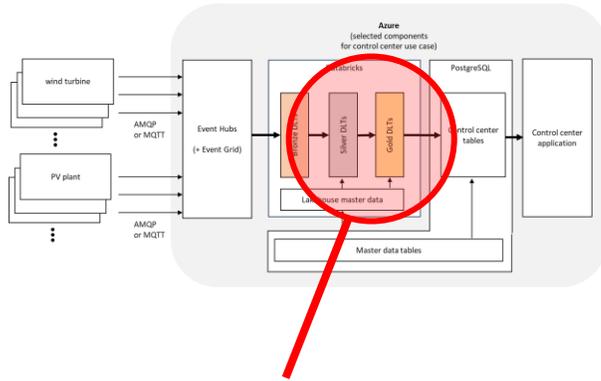
```
def remove_duplicated_rows(df: DataFrame) -> DataFrame:
    df = df.withWatermark("_inserted_at_ts", "5 seconds")
    df_deduped = df.dropDuplicatesWithinWatermark(
        ["turbine_id", "measurement_ts", "wind_speed_ms"]
    )
    return df_deduped
```

} watermarking required

my\_output\_table

turbine_id	measurement_ts	wind_speed_ms
RG15	2025-01-22T17:40:11.371894+00:00	4.3
RG15	2025-01-22T17:40:11.371894+00:00	4.3
RG15	2025-01-22T17:40:41.050264+00:00	4.4

# Removing, **updating**, and aggregating rows can be tricky and slow



```
dlt.create_streaming_table(
    name="latest_wind_speed_by_turbine",
)
dlt.apply_changes(
    target="latest_wind_speed_by_turbine",
    source="my_silver_table",
    keys=["turbine_id"],
    sequence_by=F.col("measurement_ts"),
    stored_as_scd_type=1,
)
```

latest\_wind\_speed\_by\_turbine

turbine_id	measurement_ts	wind_speed_ms
RG15	2025-01-22T17:40:11.371894+00:00	4.3
	2025-01-22T17:40:41.050264+00:00	4.4

# Delta Live Tables



Impressively  
simple for some  
streaming use  
cases



Tricky for some  
transformations

# What are Databricks Asset Bundles?

„Databricks Asset Bundles (DABs) are a tool to facilitate the adoption of software engineering best practices, including source control, code review, testing, and continuous integration and delivery (CI/CD)”<sup>1</sup>

# Sounds good.. When should I use Databricks Asset Bundles?

“Use them when you want to manage complex projects where multiple contributors and automation are essential, and continuous integration and deployment (CI/CD) are a requirement”<sup>1</sup>

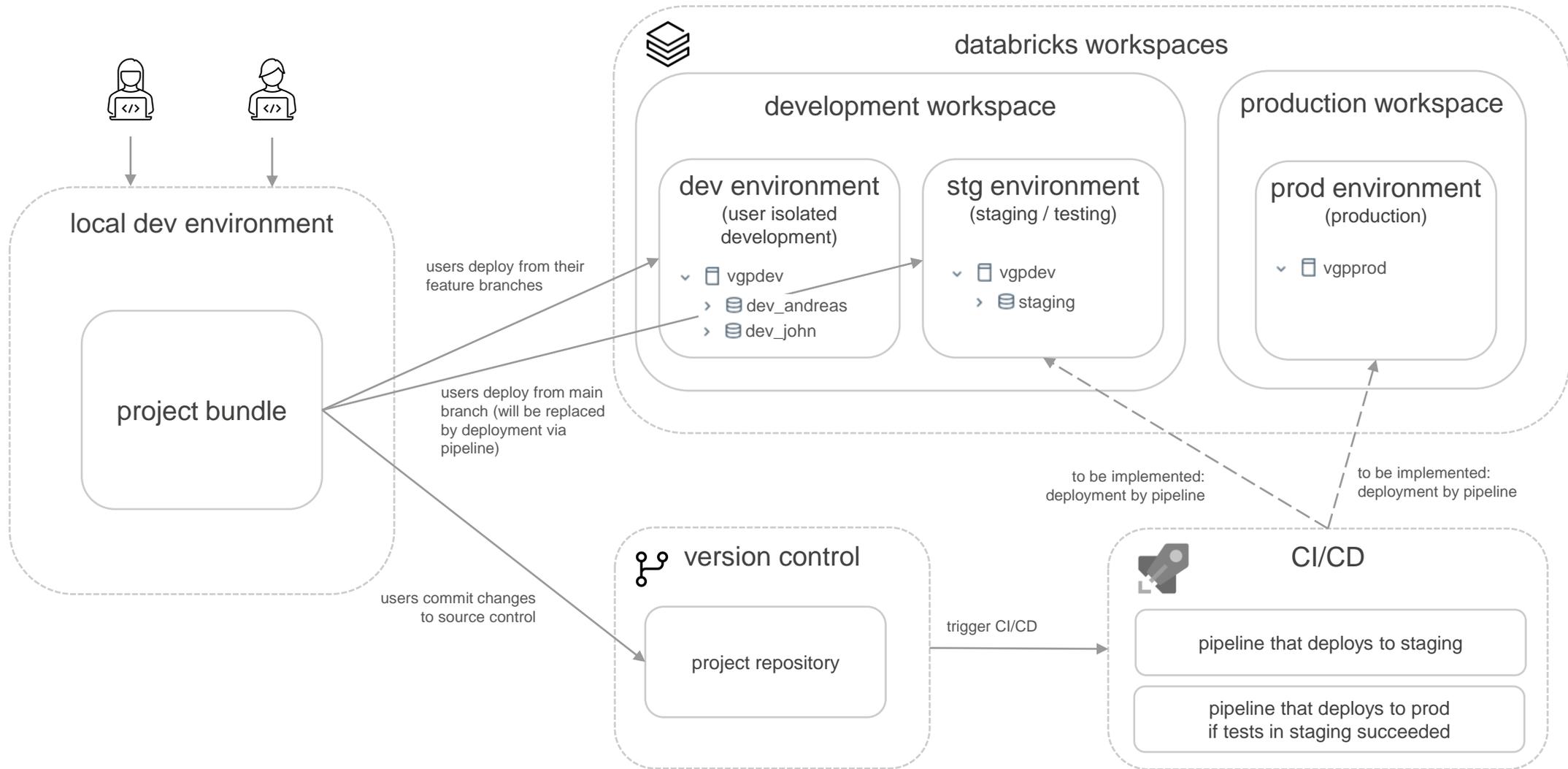
# Show me the thing

databricks bundle init

The image shows a VS Code editor with two files open. The left pane shows the Explorer view with a project structure including .databricks, .vscode, fixtures, resources, scratch, src, tests, .gitignore, databricks.yml, pytest.ini, README.md, requirements-dev.txt, and setup.py. The main editor shows the content of databricks.yml, which defines a bundle named 'my\_dab\_project' with 'dev' and 'prod' targets. The 'dev' target is set to 'development' mode and uses a workspace at 'https://adb-1111.99.azure.databricks.net'. The 'prod' target is set to 'production' mode and uses a workspace at 'https://adb-2222.99.azure.databricks.net'. The 'prod' target also includes a 'root\_path' and 'permissions' for the user 'andreas.haselsteiner@verbund.com'. The right pane shows the content of my\_dab\_project.pipeline.yml, which defines a pipeline named 'my\_dab\_project\_pipeline' with a 'notebook' library pointing to 'dlt\_pipeline.ipynb' in the 'src' directory.

```
! databricks.yml > {} targets > {} prod > {} run_as
databricks-asset-bundles.json
1 # This is a Databricks asset bundle definition for my_dab_project.
2 # See https://docs.databricks.com/dev-tools/bundles/index.html for documentation.
3 bundle:
4   name: my_dab_project
5
6 include:
7   - resources/*.yml
8
9 targets:
10  dev:
11    # The default target uses 'mode: development' to create a development copy.
12    # - Deployed resources get prefixed with '[dev_my_user_name]'
13    # - Any job schedules and triggers are paused by default.
14    # See also https://docs.databricks.com/dev-tools/bundles/deployment-modes.html.
15    mode: development
16    default: true
17    workspace:
18      host: https://adb-1111.99.azure.databricks.net
19
20  prod:
21    mode: production
22    workspace:
23      host: https://adb-2222.99.azure.databricks.net
24    # We explicitly specify /Workspace/Users/andreas.haselsteiner@verbund.com to make sure we only have a single copy.
25    root_path: /Workspace/Users/andreas.haselsteiner@verbund.com/.bundle/${bundle.name}/${bundle.target}
26    permissions:
27      - user_name: andreas.haselsteiner@verbund.com
28        level: CAN_MANAGE
29    run_as:
30      user_name: andreas.haselsteiner@verbund.com
31
! my_dab_project.pipeline.yml > ...
databricks-asset-bundles.json
1 # The main pipeline for my_dab_project
2 resources:
3   pipelines:
4     my_dab_project_pipeline:
5       name: my_dab_project_pipeline
6       ## Specify the 'catalog' field to configure this pipeline to
7       # catalog: catalog_name
8       target: my_dab_project_${bundle.target}
9       libraries:
10        - notebook:
11          path: ../src/dlt_pipeline.ipynb
12
13 configuration:
14   bundle.sourcePath: ${workspace.file_path}/src
15
```

# Our development process with Databricks Asset Bundles



# Delta Live Tables & Asset Bundles: What's great and what isn't

## Delta Live Tables

- + Reading a streaming source is straightforward
- + Append-only transformations over multiple tables
- Things that are straightforward in batch like removing, updating, and aggregating records can be tricky

# Delta Live Tables & Asset Bundles: What's great and what isn't

## Delta Live Tables

- + Reading a streaming source is straightforward
- + Append-only transformations over multiple tables
- Things that are straightforward in batch like removing, updating, and aggregating records can be tricky

## Databricks Asset Bundles

- + The best thing since sliced bread
- ?

**V** Thank you!

[andreas.haselsteiner@verbund.com](mailto:andreas.haselsteiner@verbund.com)