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

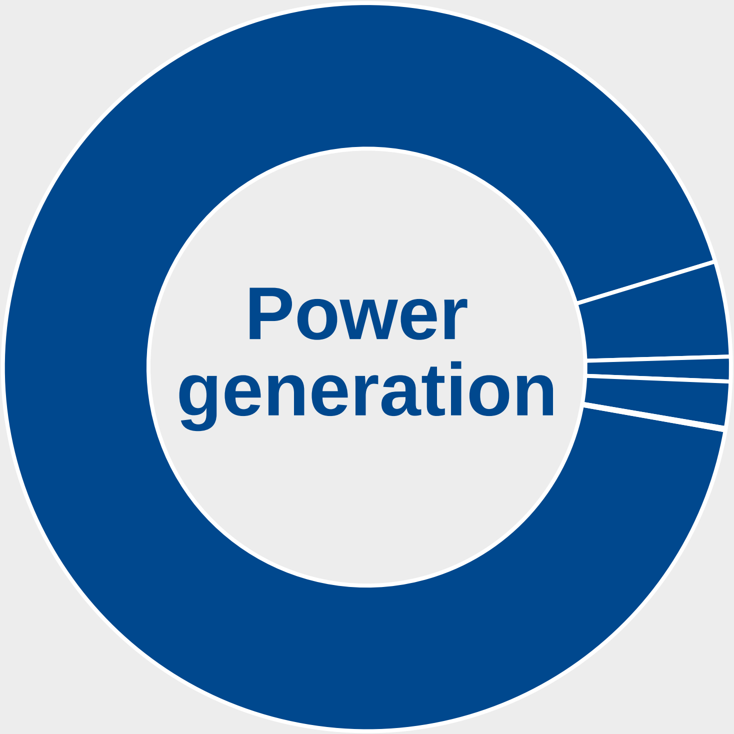
VERBUND AG & VERBUND Green Power at a glance

VERBUND AG	ca. 3,800 employees[1]	12 countries[2]	ca. 33 TWh electricity[3]
VERBUND Green Power GmbH	ca. 170 employees	6 countries	ca. 2 TWh electricity[3]

1 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
2 https://www.verbund.com/en-at/about-verbund/company
3 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

VERBUND generates 98% from renewable energy sources

Hydropower[1]
>92% of electricity generation



Wind power
>4% of electricity generation

Solar power[2]
>1% of electricity generation

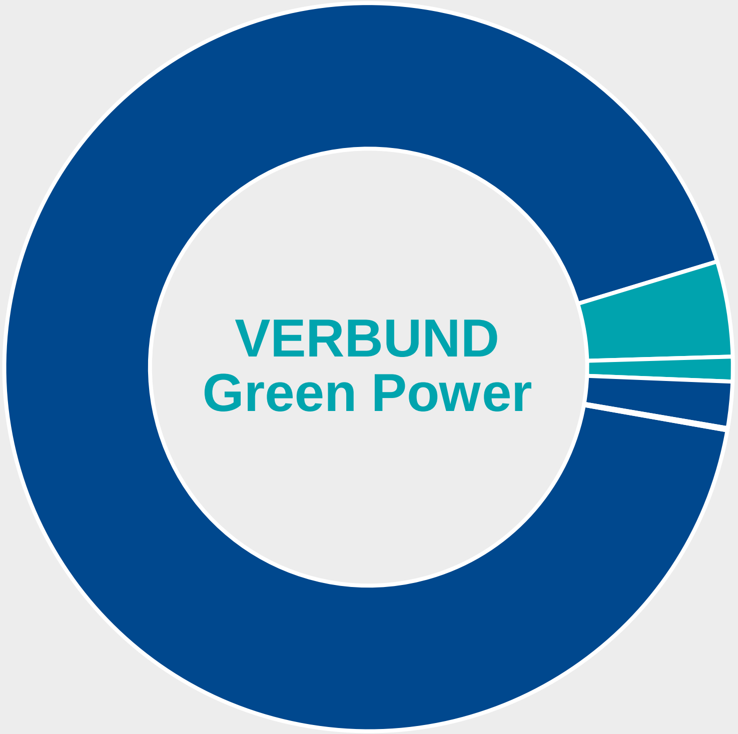
Thermal power
>2% of electricity generation

Battery storage
<1% of electricity generation

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[1]
>92% of electricity generation



Wind power
>4% of electricity generation

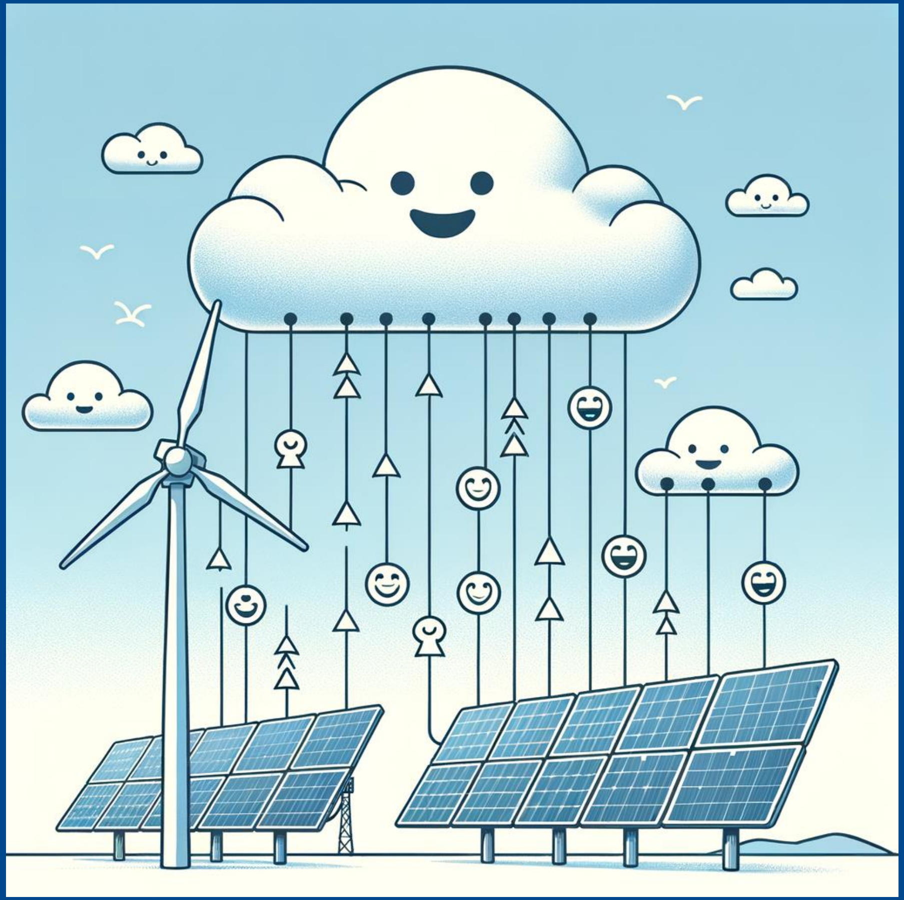
Solar power[2]
>1% of electricity generation

Thermal power
>2% of electricity generation

Battery storage
<1% of electricity generation

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

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

Databricks Asset Bundles (DABs) became generally available in 2024[2]

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

V

1 https://www.databricks.com/blog/2022/04/05/announcing-generally-availability-of-databricks-delta-live-tables-dlt.html
2 https://www.databricks.com/blog/announcing-general-availability-databricks-asset-bundles

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

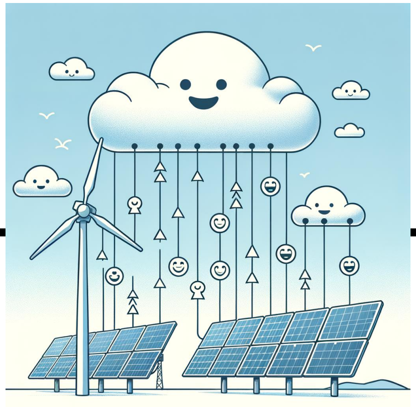
Our use case: Operational plant data for live monitoring and BI reporting

wind turbine

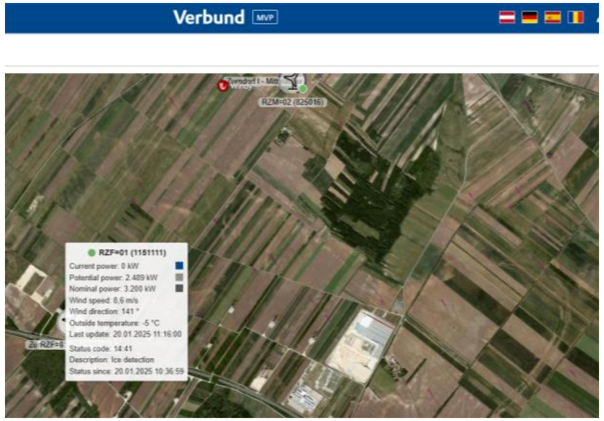
⋮

PV plant

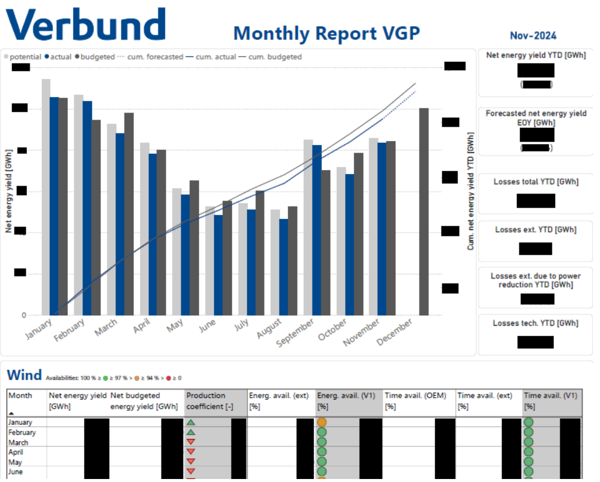
⋮



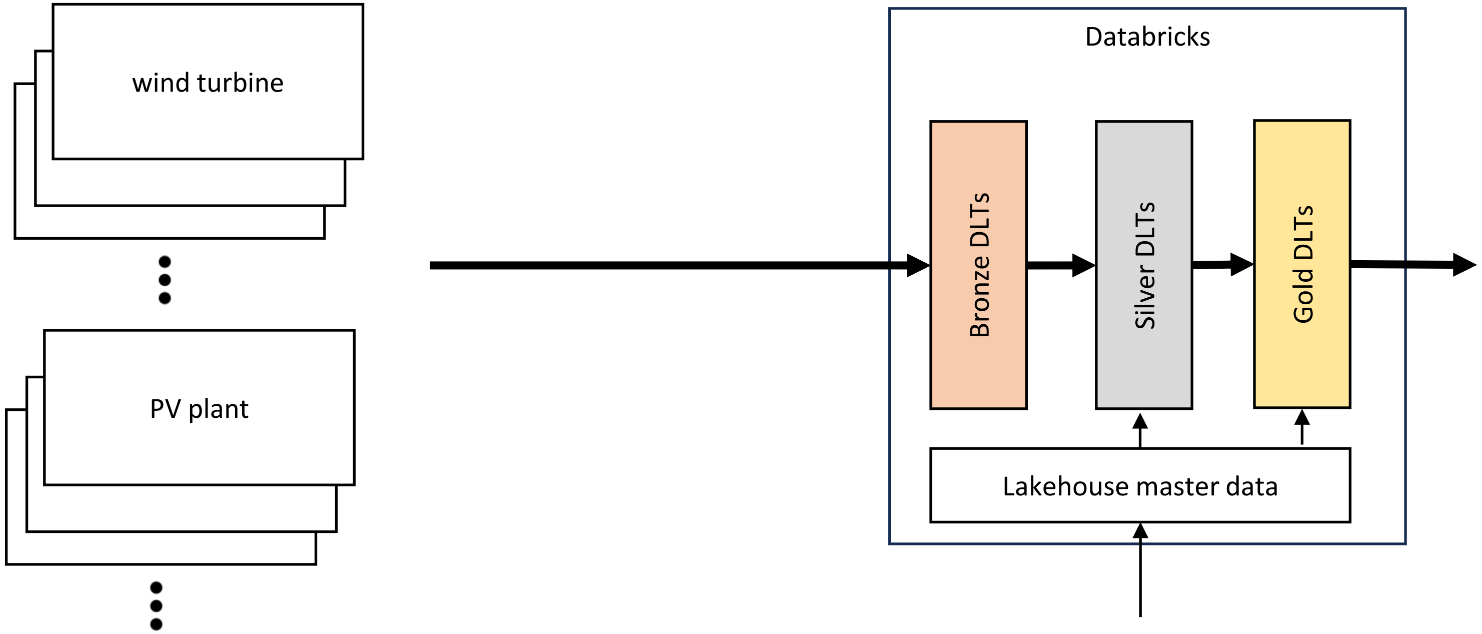
live monitoring



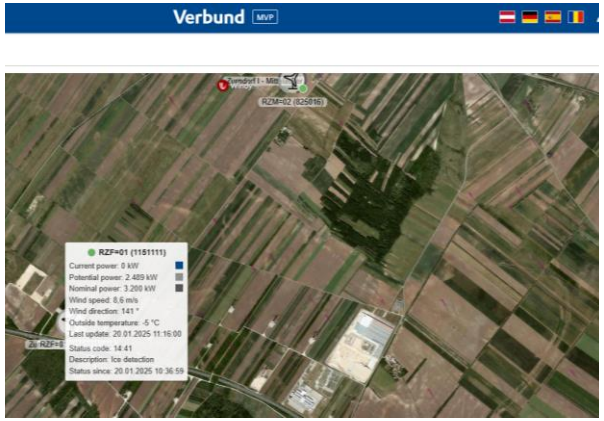
business intelligence



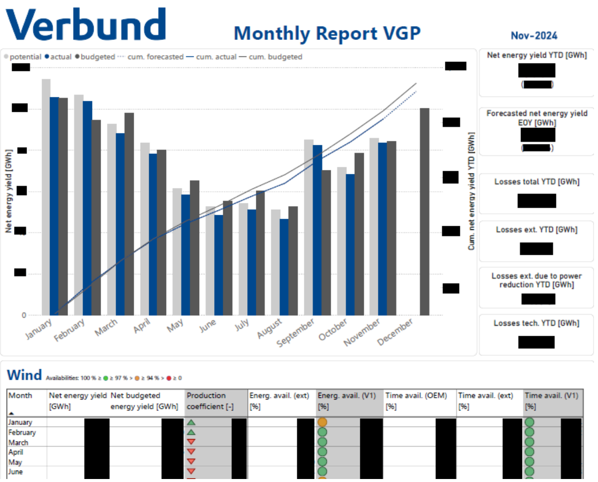
All data are streamed through Databricks



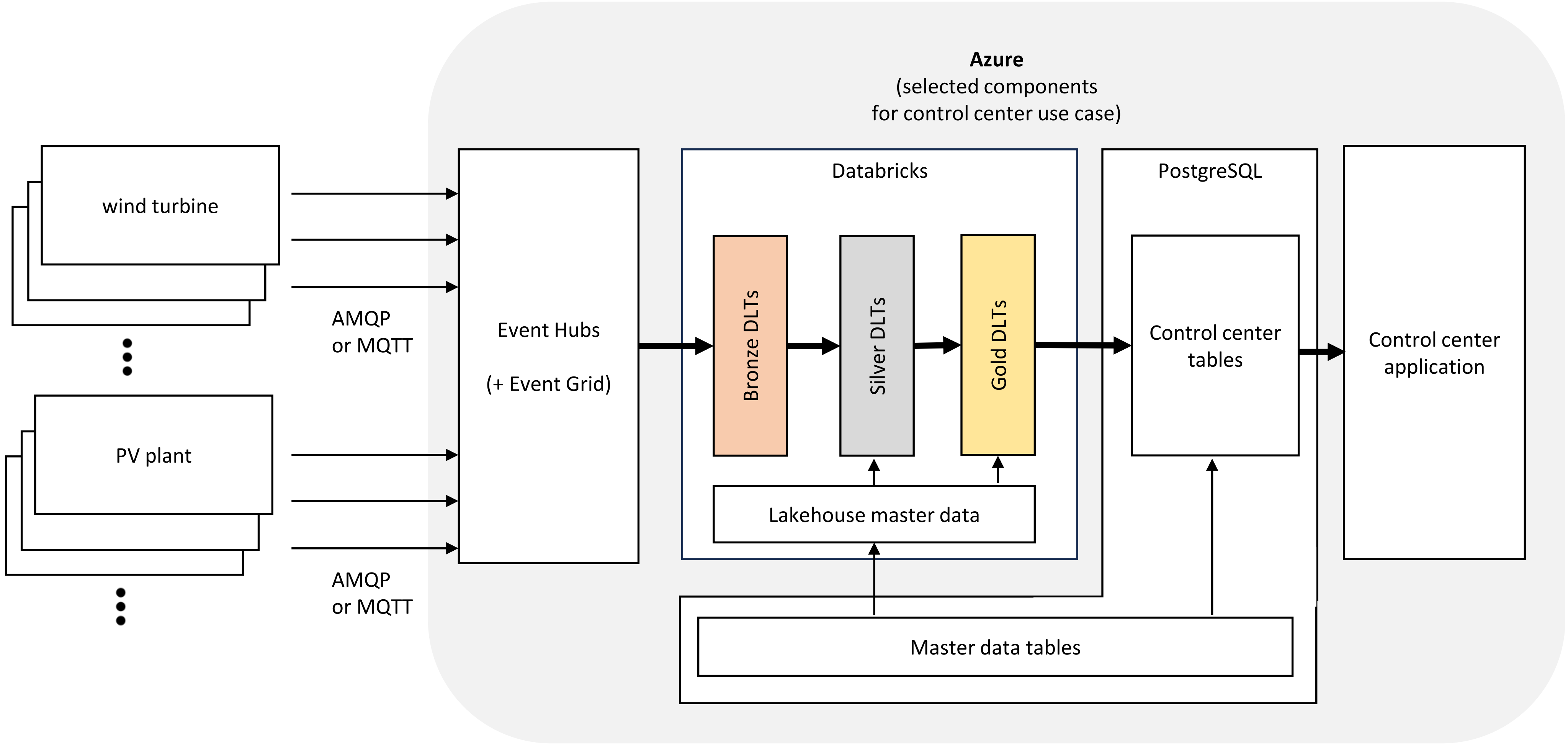
live monitoring



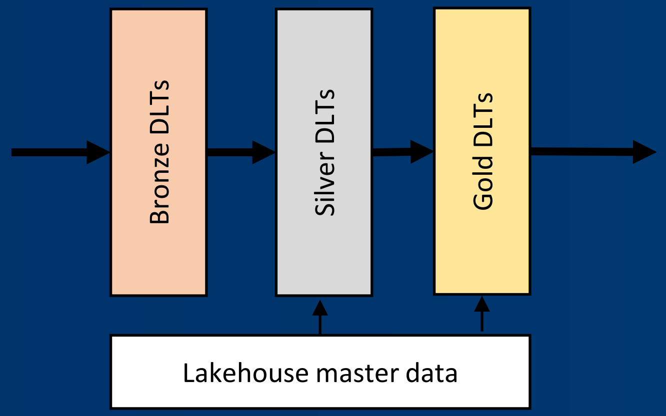
business intelligence



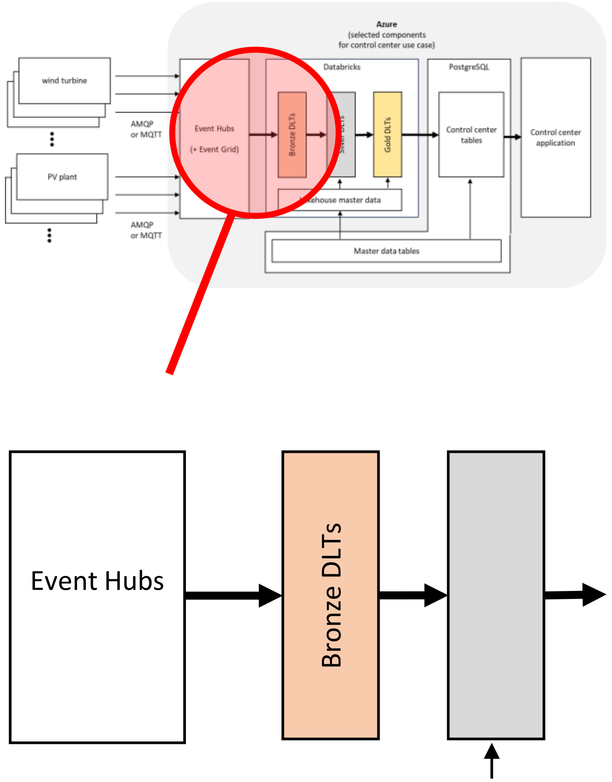
We use message protocols to send data towards the cloud and ingest these data to Databricks from Event Hubs



Delta Live Tables

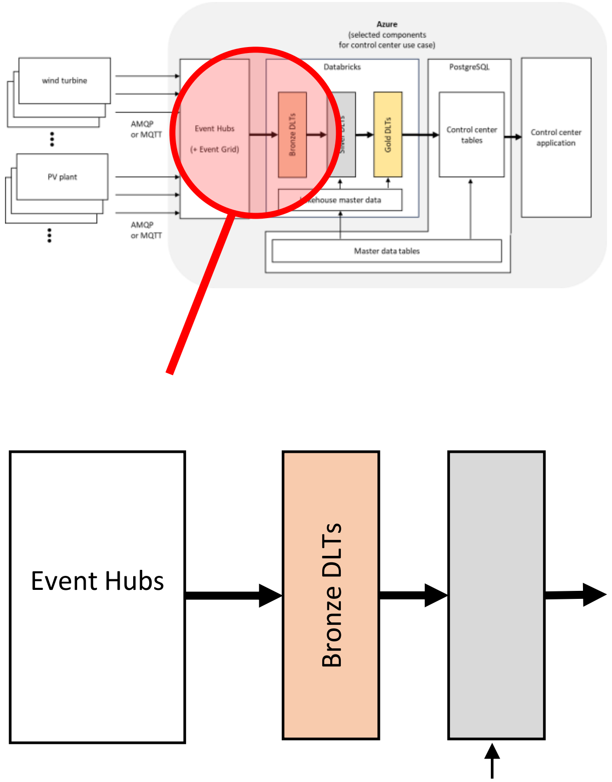


DLT makes ingesting messages into a table straight-forward




```python
@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




```python
@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)
    )
```

```python
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

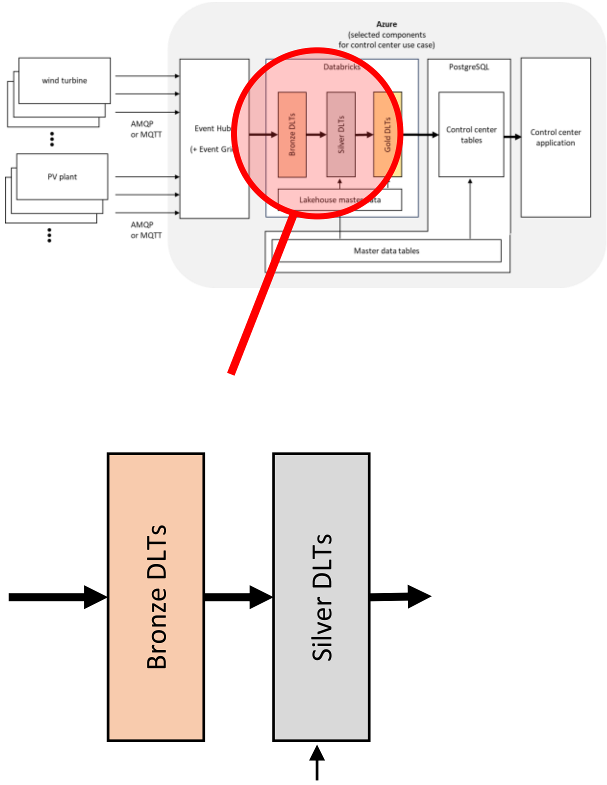
_eh_enqueued_ts[1]	_inserted_at_ts	payload

2025-01-22T17:40:12.371+00:00	2025-01-22T17:40:14.422+00:00	{
		"turbine_id":"RG15",
		"measurement_ts":"2025-01-22T17:40:11.371894Z",
		"data":{
		"WTUR.W":67000,
		"WMET.HorWdSpd":4.3,
		"WMET.HorWdDir":134.5,
		…
		}
		}

2025-01-22T17:40:42.560+00:00	2025-01-22T17:40:44.119+00:00	{
		"turbine_id":"RG15",
		"measurement_ts":"2025-01-22T17:40:41.050264Z",
		"data":{
		"WTUR.W":67000,
		"WMET.HorWdSpd":4.4,
		"WMET.HorWdDir":137.1,
		…
		}
		}

1 Some people like to add an underscore prefix to technical columns: https://towardsdatascience.com/best-practices-for-technical-columns-in-database-design-f3fce9357426

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



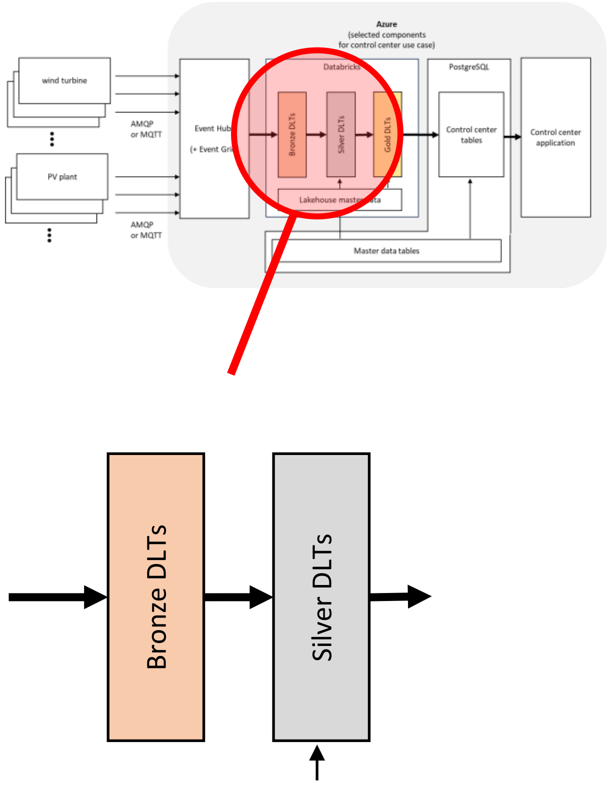

```
@dlt.table
def my_silver_table() -> DataFrame:
    return dlt.read_stream("my_bronze_table").transform(transform_bronze_to_silver)
```

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,
		…
		}
		}

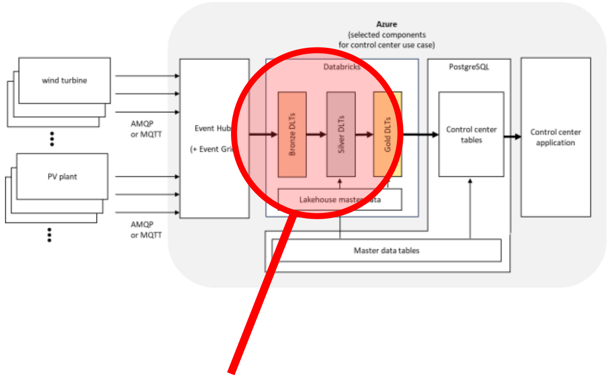
Append-only operations like extracting and casting JSON values work well

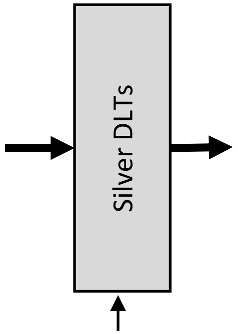



```python
@dlt.table
def my_silver_table() -> DataFrame:
    return dlt.read_stream("my_bronze_table").transform(transform_bronze_to_silver)
```

```python
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



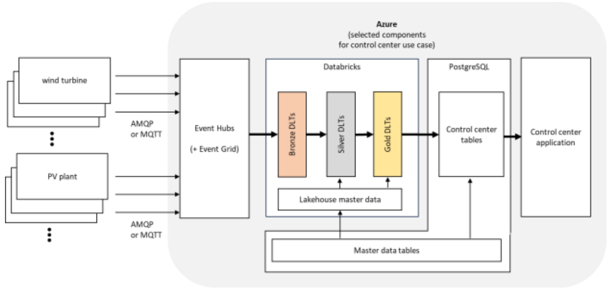



```
@dlt.table
def my_silver_table() -> DataFrame:
    return dlt.read_stream("my_bronze_table").transform(transform_bronze_to_silver)
```

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




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

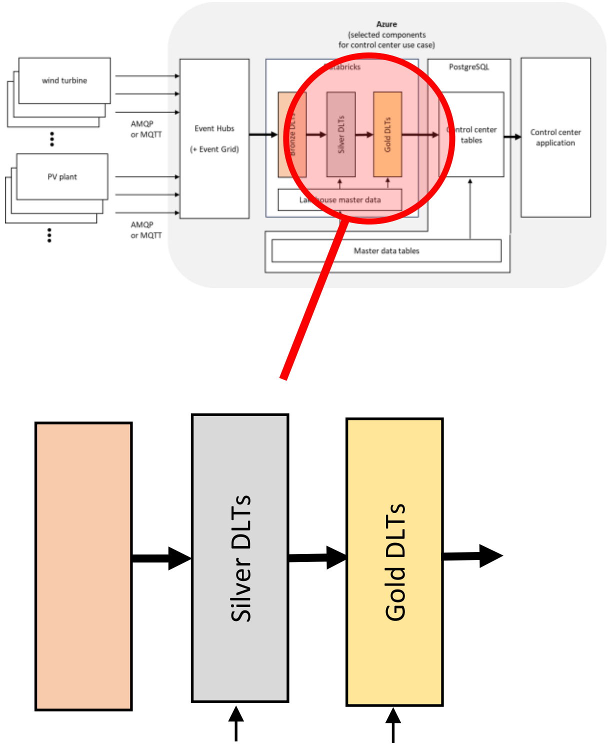
```python
def remove_duplicated_rows(df: DataFrame) -> DataFrame:
    df = df.withWatermark("_inserted_at_ts", "5 seconds")
    df_deduped = df.dropDuplicatesWithinWatermark(
        ["turbine_id", "measurement_ts", "wind_speed_ms"]
    )
    return df_deduped
```

} 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~~ 2025-01-22T17:40:41.050264+00:00	~~4.3~~ 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)"[1]

1 https://docs.databricks.com/en/dev-tools/bundles/index.html

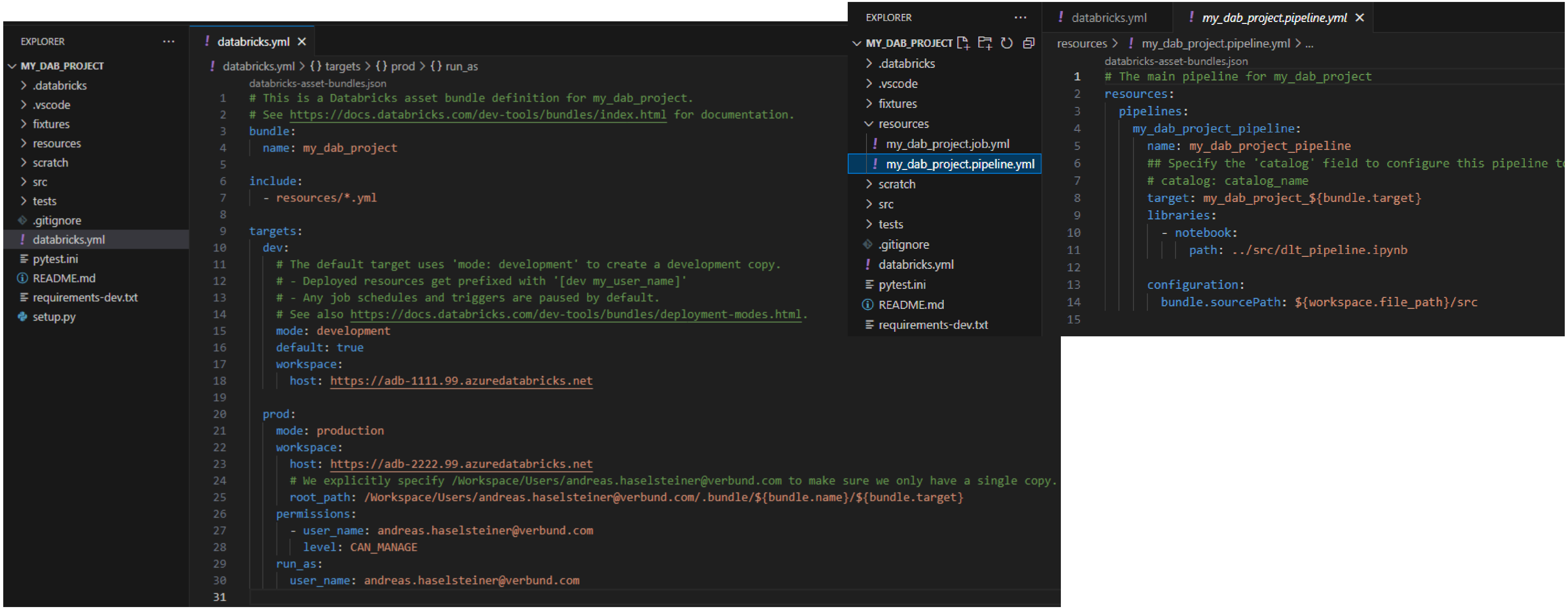
Sounds good.. When should I use Databricks Asset Bundles?

“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”[1]

1 https://docs.databricks.com/en/dev-tools/bundles/index.html

Show me the thing

```
databricks bundle init
```



Our development process with Databricks Asset Bundles

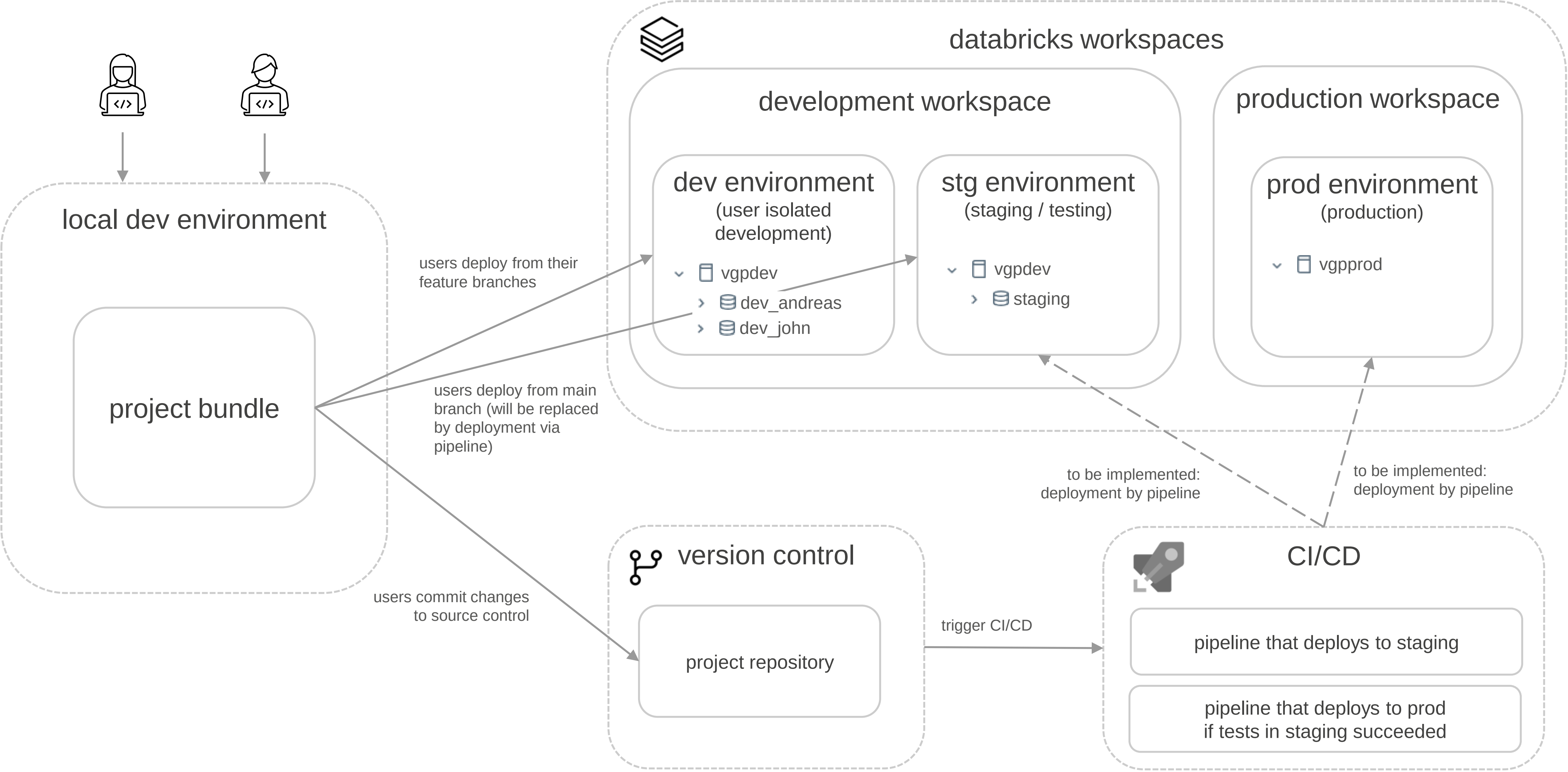


Illustration inspired by https://docs.databricks.com/en/dev-tools/bundles/index.html
Many thanks for setting up our initial DAB to Thomas Totter!

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

- ?

Thank you!

andreas.haselsteiner@verbund.com

Verbund