**This is just to help during my lightning talk**

**Basically, a dialog, a script.**

**\*\*Notes:**

***- ACID transactions refer to: Atomicity, entire transactions completes; Consistency, Data follows rules, or it will be rolled back; Isolation, one transaction completed before the other start; Durability, Data is saved in a persistent state once is completed.***

***- Lineage refers to the full journey of data from the source to the destinations. It shows the conversions, transformations, or processes applied to it.***

***- a brief explanation of databricks. Databricks is a unified data analytics platform built on Apache Spark. Basically, it combines data engineering, data science, machine learning and analytics on a single platform powered by Lakehouse architecture and it exists to empower data teams to collaborate, and analyze and innovate at scale.***

***-Yes, it can run automatically, either you set it to run at a certain time intervals or to be triggered when new files are created***

***-*** ***AUTO CDC flows let you ingest change data seamlessly without manual dedupe logic or ordering headaches-***

***-*** ***Streaming tables and materialized views unify batch and streaming workflows***

***-*** ***The new IDE lets you see DAGs, live previews, and debug inline***

***-*** ***You can add declarative data quality expectations directly in pipelines so errors fail early***

***-*** ***Compute is serverless or autoscaling and recovery is automatic—making operations smooth.***

**SLIDE** **1** — Presentation (10 seconds | ~20 words)

Hi everyone, my name is Leonardo Trevizo. And in the next five minutes I’ll show you how Lakeflow declarative Pipelines simplify building and operating ETL workflows.

**SLIDE 2** — Introduction (25 seconds | ~55 words)

To continue with the topics from the previous lightning talks, today I will focus on automating data engineering within databricks. In my first lightning talk we explored Unity Catalog as the foundation for governance and security. In the second, we discussed Delta Lake and Data Lakehouse architecture, this third session introduces Lakeflow declarative pipelines, a technology designed to simplify and automate data engineering workflows.

**SLIDE 3** — Agenda (10 seconds | ~25 words)

Alright, here’s a quick roadmap for what we’ll be covering.

We’ll start by looking at the problems that arise when working with ETL processes with traditional tools.

**SLIDE 4** — What are Lakeflow Declarative Pipelines (35 seconds | ~75 words)

So… Most traditional data pipelines are fragile. You need to write multiple scripts, and connect tasks manually, which makes orchestration hard. Logic is often duplicated—one set for batch, another for streaming—adding complexity. Ordering execution steps like CDC merges becomes a painful, manual process. When jobs fail, debugging and recovery are time‑consuming. And across all this, governance is inconsistent and hard to enforce. In short, pipelines without automation are slow to build, costly to maintain, and error‑prone.

***SLIDE 5*** *— What are Lakeflow Declarative Pipelines? (50 seconds | ~85 words)*

*What are Lakeflow declarative pipelines? Declarative pipelines let you define what you want like a transformation or a CDC target without specifying how to execute each step. This declarative syntax means you can focus on the logic whilst Lakeflow automatically manages orchestration and dependencies. You simply write code in SQL or Python, Lakeflow determine how and when to run, with automated scaling to match the workload. The best part is this method combines batch and streaming logic in a seamless way.*

**SLIDE 6** — What is a declarative pipeline? (1:25 minutes | ~190 words)

When working with ETL pipelines, first you need to figure out… What am I doing with this data?

Is the data going to be streaming?

Is it going to be stored in Delta Lake?

How will it be orchestrated?

Is it intended for data science?

Will it be placed in a data warehouse?

Will it be used for BI?

There are multiple objectives that can be defined for the final data. Then, version control is required, as well as a strategy for deploying the ETL logic.

It is also necessary to establish how the quality of the data will be checked, how governance will be applied, how the data will be discovered and explored, and how backfill will be managed. Dependencies between pipelines or flows must be handled, along with checkpoints and retries for incremental processing, ensuring that only new data is processed.

Essentially, with Lakeflow declarative pipelines, you only need to focus on transforming your data, on building the transformation logic.

**SLIDE 7** — How a data engineer uses traditional pipelines (25 seconds | ~40 words)

In this example you can see that with traditional ETL, every time new files arrive you must re‑run the same command, replacing the entire table. That means reprocessing all data, even records that were already ingested. It is repetitive, inefficient, and costly.

***SLIDE 8*** *— Using declarative Pipelines -Example (25 seconds | ~35 words)*

*Here, you’ll see that declarative pipelines changes this. Instead of manual replacement, new files are automatically detected and processed, while existing data remains untouched. You get incremental ingestion by default, so pipelines scale with your data.*

**SLIDE 9** — Built-in monitoring, etc. -Example (15 seconds | ~45 words)

On top of this, Lakeflow provides built‑in monitoring and metrics. You can see record counts, job status, and lineage directly in the UI.

In short: less wasteful reprocessing, more intelligent streaming, and full visibility—pipelines that keep themselves up to date.

***SLIDE 10*** *— Real benefits + Conclusion (20 seconds | ~50 words)*

*Lakeflow Declarative Pipelines builds on the solid foundation of Delta Lake and Unity Catalog that we discussed earlier. Delta gives you ACID compliance and versioning; Unity Catalog gives you governance and lineage. Lakeflow brings automation, quality, and operational simplicity. Together, they form a truly modern, managed data Lakehouse that’s reliable, traceable, and scalable.*

**SLIDE 11** — Acknowledgements (10 seconds | ~20 words)

Thank you all for your time and attention.

Special thanks to my instructors, colleagues and mentors, for your support and feedback.

• • • •