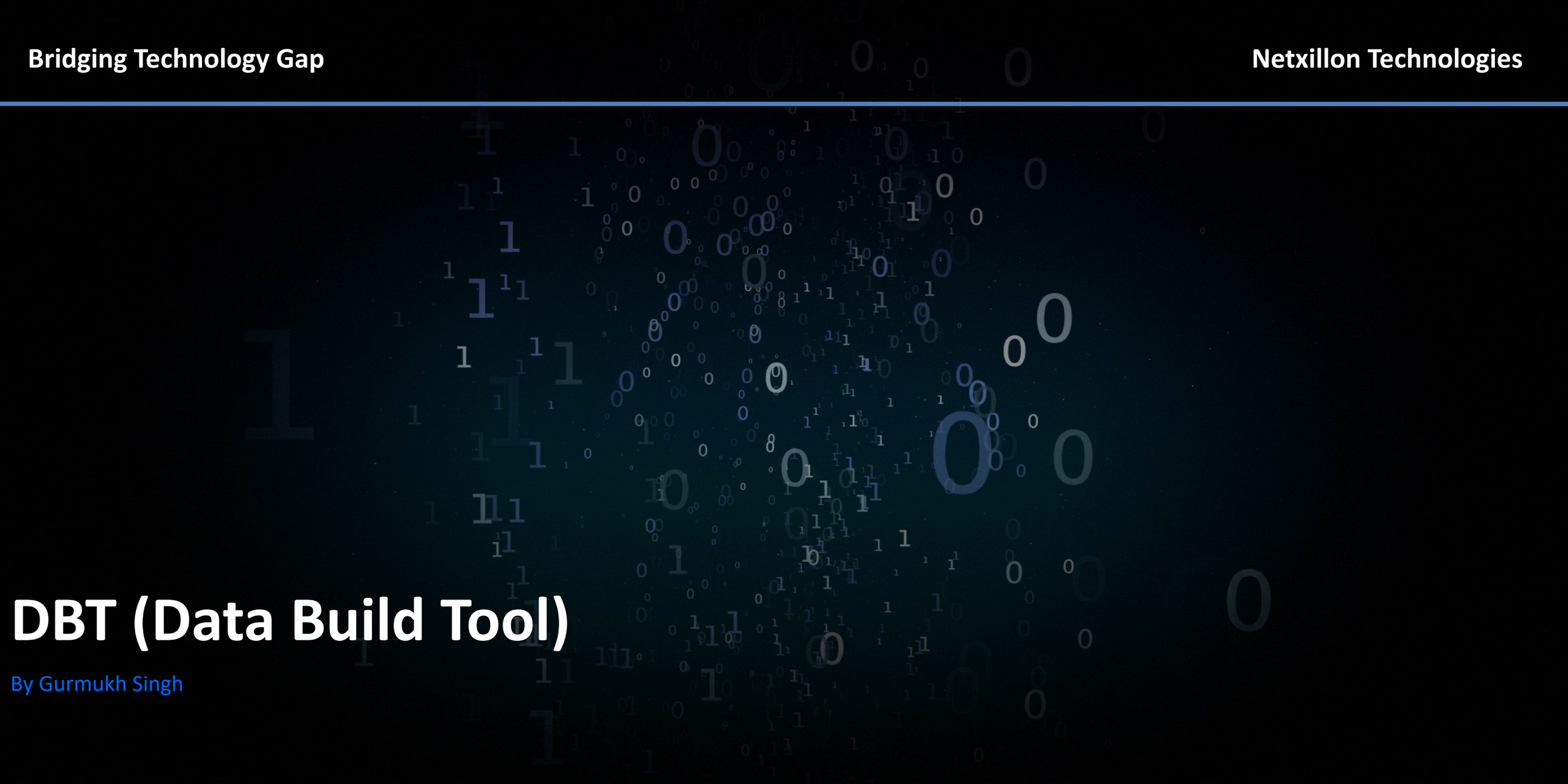


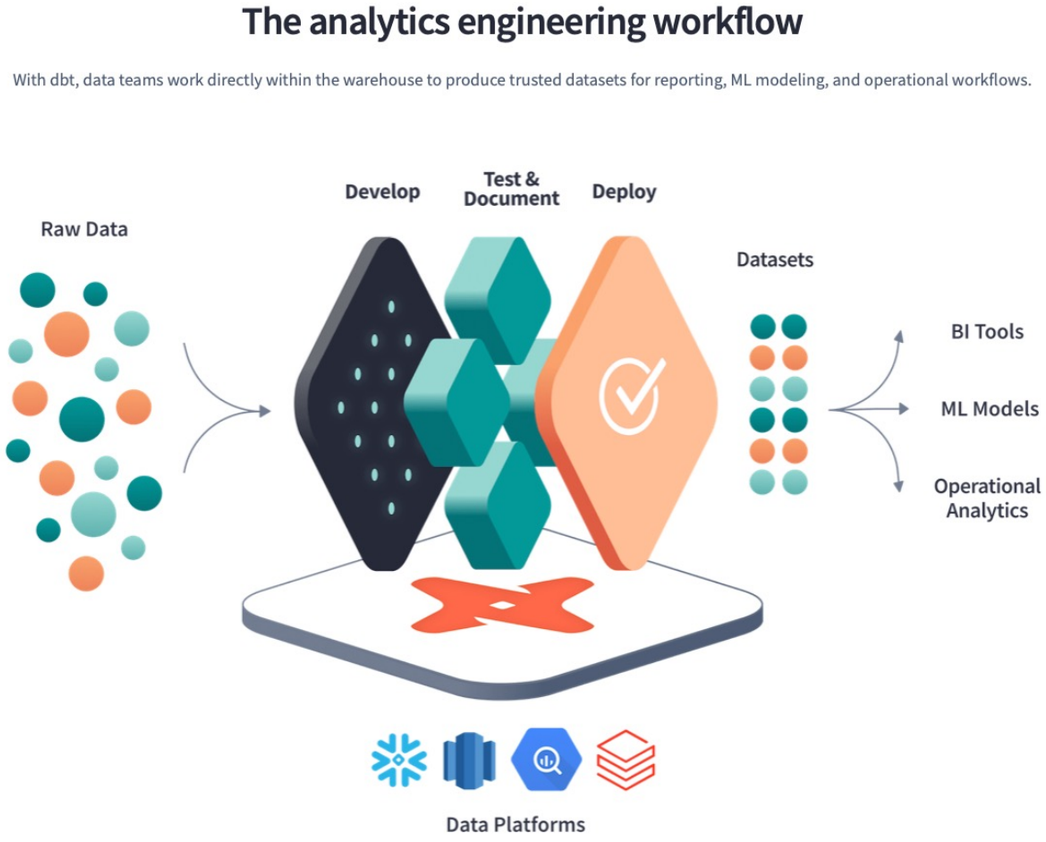
DBT (Data Build Tool)

By Gurmukh Singh

About Me:

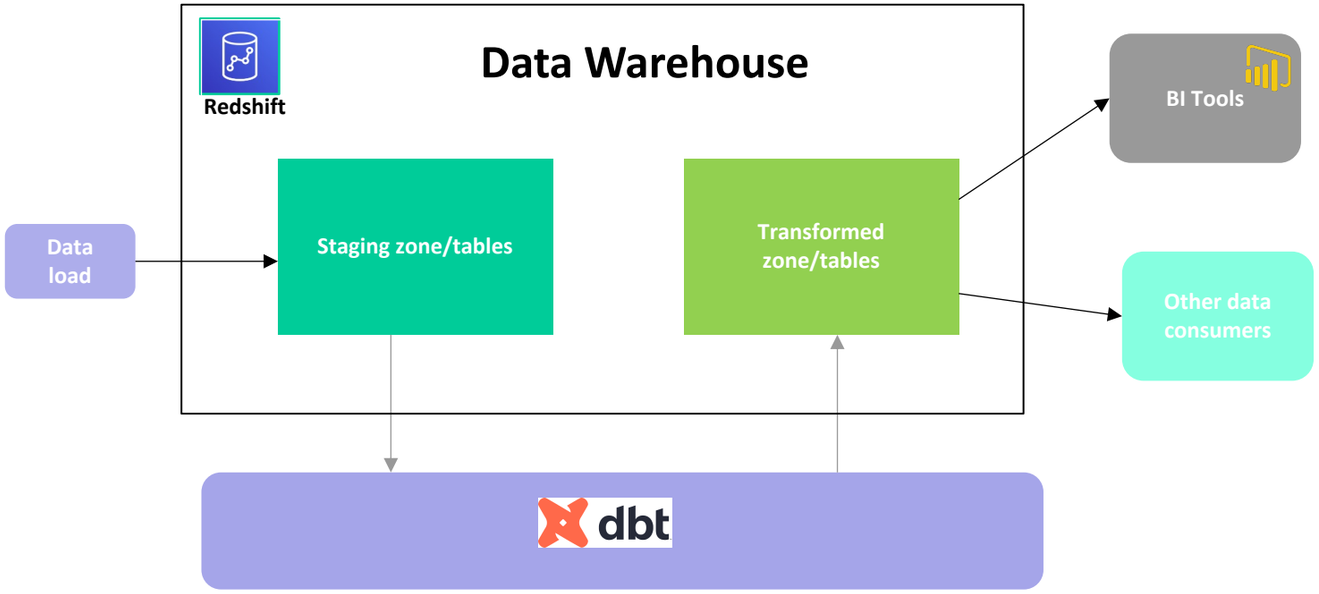
- My Name is **Gurmukh Singh,** you can call me "**Aman"**
- Have 17 + years of Experience, mainly into:
  - Distributed and Big Data Systems
  - Infrastructure Scalability and Performance Tuning.
  - Network Planning and Design.
  - Hadoop Eco-System
- M. Tech, Computer Science
- Red Hat Certified Architect – RHCA
- Certified on Cloudera, Hortonworks and AWS.
- Worked with organization like Yahoo, Amazon, Hortonworks, Cloudera.
- Author of books "**Monitoring Hadoop**" and "**Hadoop 2.x Administration CookBook**"

Dbt is a development framework that combines modular SQL with software engineering best practices to make data transformation reliable, fast, and fun.

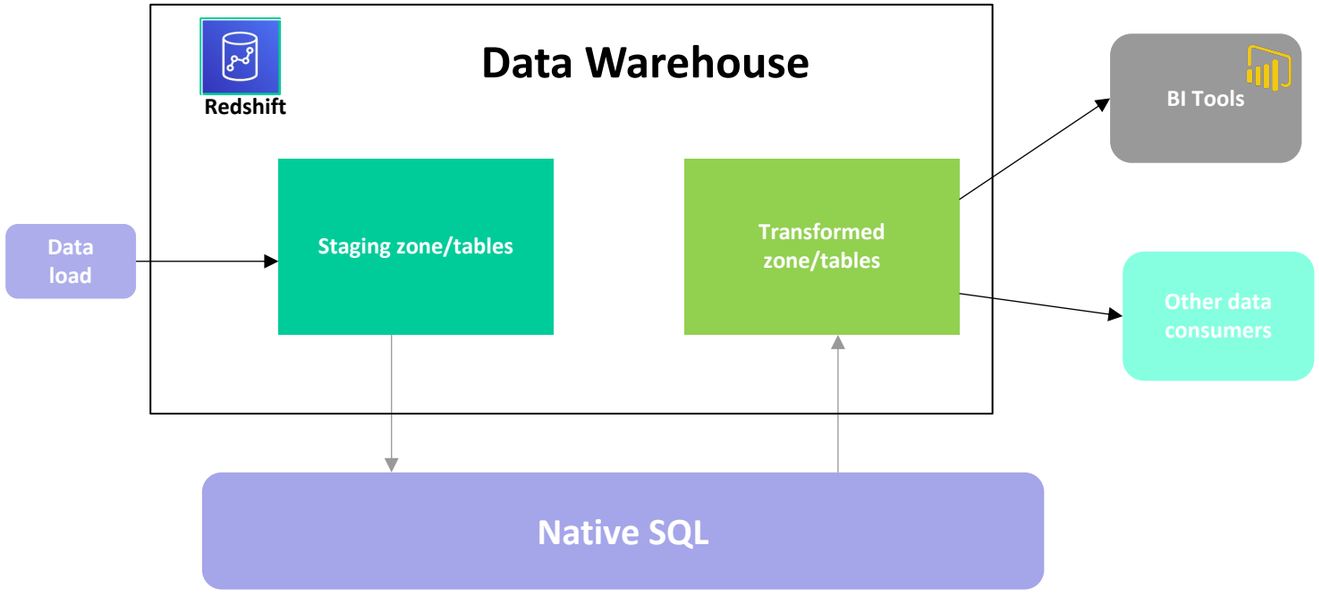


- **Use SQL**
- **Documentation**
- **Model Lineage**
- **Testing**
- **Reusability**
- **Integration with many DB's**

Advantages of using Dbt	Overhead of Not using Dbt
• Easy to use for non engineers (shared data knowledge between engineering and non engineering teams). • Version controlled way of writing transformations using just SQL. • Extremely flexible data model (recreate data easily, backfills are easy). • If most of your transformations are at a data warehouse level, this tool makes it extremely easy to do. All computational work is pushed to the DW, so requires minimal resources • Built in testing for data quality. • Online, searchable data catalog and lineage. • Production run using dbt cloud or through Airflow trigger. • No need to write DDL or DML code (CREATE TABLE AS, CREATE VIEW AS, INSERT Table, DELETE, etc.) as it is abstracted. • Integration of python code to enrich SQL code. • Git-type code integration. • Test which code may break a logic step. Helps productionize code for ML into prod.	• Complex Redshift Native SQL code and transformations and UDFs. • Overhead of writing custom data quality and test code. • Need to write entire code flow with DDL, DML as nothing is inferred automatically. • No searchable catalog or lineage information with Redshift native SQL. • Medium complexity of integration with Airflow for production runs. • Must have external version control integration with tools like Git. • Difficult to introduce break points for debugging.



- Idempotent operation, run as many times as possible to get the same output.
- Data engineers create models, connect them, and then leave the work of creating the actual dataset into the database to dbt.
- Models are materialised using views, tables and other methods depending on the underlying database.
- Support incremental loads, documentation, validate the quality of datasets



- Industry standard Postgres Layer for SQL, but developers need good hold on the postgres SQL/functions
- Write DDL, DML and SQL for transformation and data loading to different tables.
- Manually write logic for schema and data quality validation.
- Write and Use UDF for customer operations like flattening etc.
- Difficult to maintain SQL code as it grows during the project life cycle.
- No lineage graph between tables.
- Manual Documentation

Important Links:

- https://github.com/netxillon/Hadoop/tree/master/Courses_Offered
- https://www.getdbt.com/
- https://datacoves.com/post/an-overview-of-testing-options-in-dbt-data-build-tool

DEMO

Any Questions ?

Thanks !

trainings@netxillon.com