

# OPEN SOURCE DATAOPS AND MLOPS STRATEGIES





# HELLO!

I am **Lisa N. Cao**

- Data Analyst turned **Data Engineer**
- Background in GovTech, Biotech, startups, and consulting
- **Women TechMakers Ambassador** and Community Organizer
- Podcast called **SystemsNull** on Spotify

**[linktr.ee/lisancao](https://linktr.ee/lisancao)**

## — What I hope you'll leave with

- Teach you a bit of **what DataOps is**
- Give you some **relevant frameworks** for DataOps
- How to evaluate **where to concentrate your efforts**
- The **open source tooling** relevant to your DataOps needs



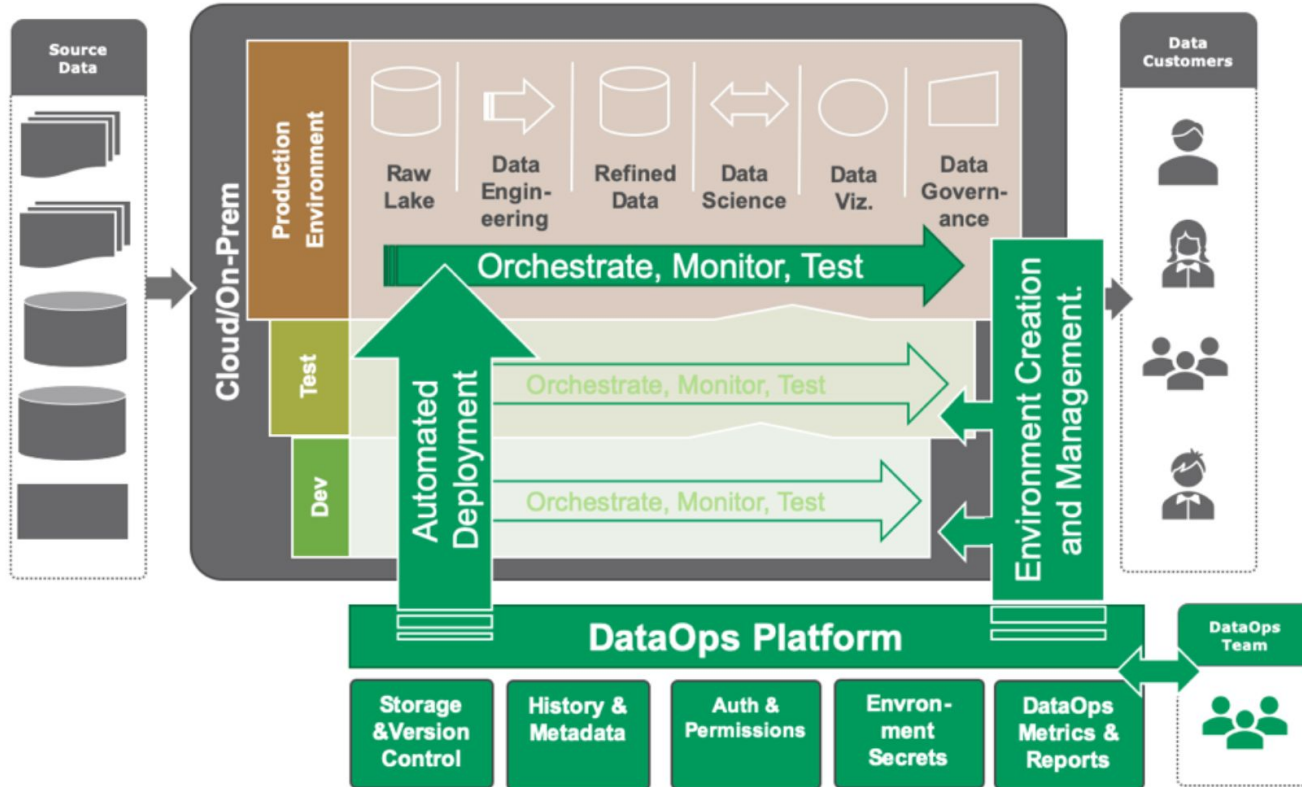
***“DataOps refers to the process of improving the reliability and performance of your data through automation, reducing data silos, and fostering quicker, more fault tolerant systems”***

*– Data Quality Fundamentals*

***“MLOps is a core function of Machine Learning engineering, focused on streamlining the process of taking machine learning models to production, and then maintaining and monitoring them”***

*– Databricks Blogpost*

# DataOps Functional Architecture



## — DataOps Components

- Orchestration of pipelines
- Governance, data contracts
- Data quality and monitoring (observability)
- CI/CD of data code and assets
- Documentation
- Deployment and containerization

## — Increased Need for DataOps

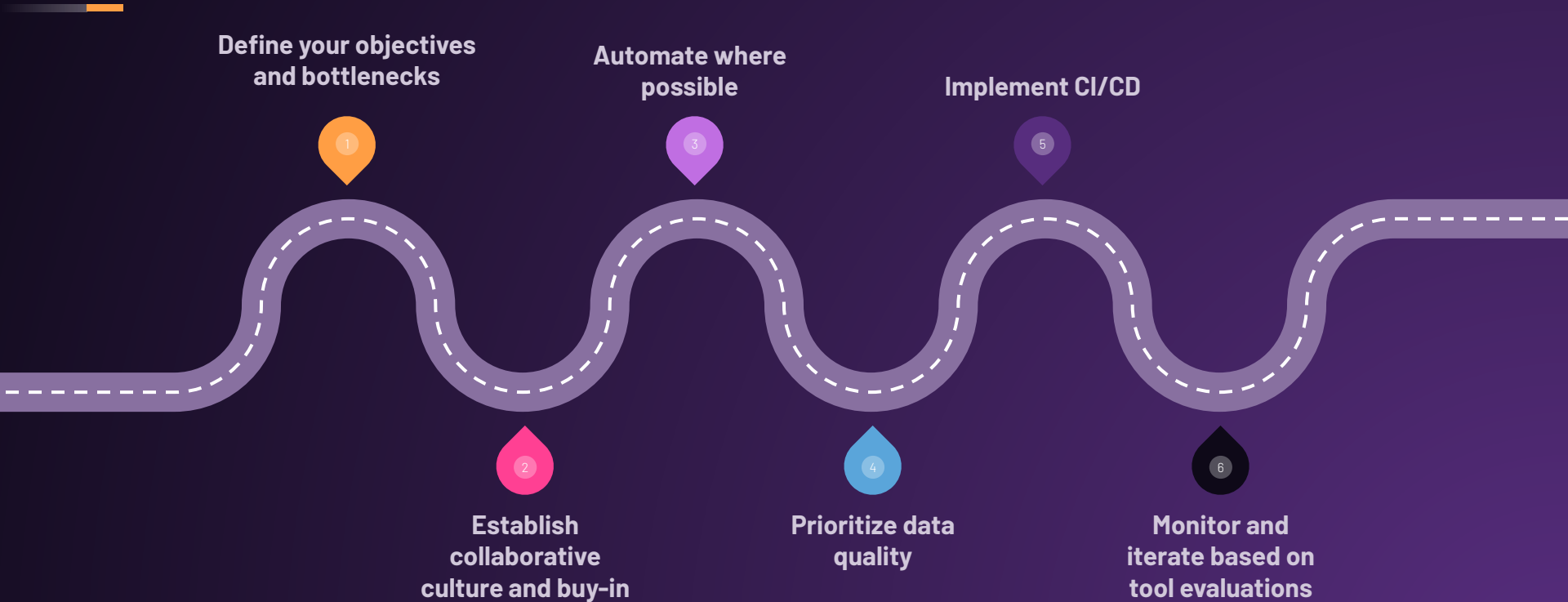
- Migration to Cloud
- More data sources
- Increasingly complex data pipelines
- More specialized data teams
- Decentralized data teams
- Data Mesh, Data Platforms, Data Streaming, and Data Lakehouses



## — Benefits of DataOps

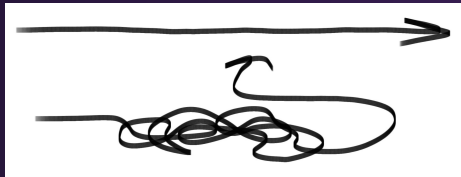
- Access to data lineage and **observability** metrics
- Practical data **governance** and legal compliance
- Increased **Security** and breach protection
- **Data Reliability Engineering**
- Automation and streamlining of Workflows
- **Data Pipelines/ML as Code**
- Versioning, iteration, configuration, dependency management

# Strategy Roadmap



## — Implementation Challenges

- Interoperability
- Standardization
- **Vendor Lock-In**
- **Managing complexity**
- Culture and adoption
- **Predicting future needs and requirements**
- Migration



DevOps

DataOps/MLOps

## — STRATEGY DOMAINS

### **Accuracy and data quality**

Data Quality Assurance  
Reporting & Metrics  
Data Contracts

Q

R

### **Reliability and latency**

Monitoring & Alerting  
Optimization  
Fault Tolerance  
Versioning, CI/CD

T

S

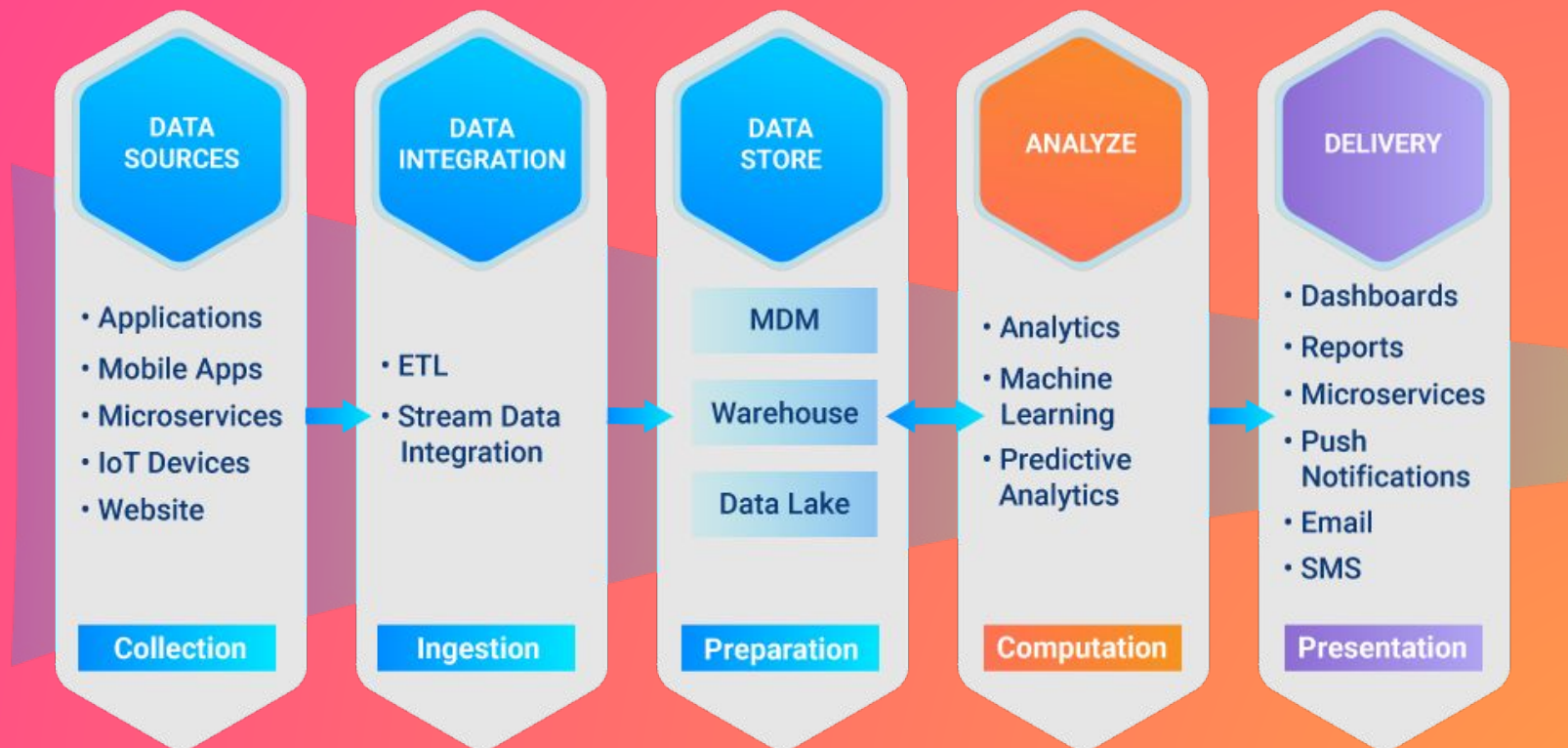
Data Lineage  
Metadata Collection  
Observability  
Documentation

### **Visibility and Transparency**

Encryption  
Credential Management  
Zero Trust Policies

### **Security**

# BIG DATA PIPELINE



[Image from Stonebranch](#)

## OPEN SOURCE PIPELINE TOOLING

### Data Collection and Ingestion

**Stream/Log:** Apache Kafka, Apache Flink, Apache Flume, Apache NiFi, LogStash

**ETL:** Airbyte, DLT, Meltano, Singer.io, Stitch Data, Apache Beam



### Data Transformation

OpenRefine, DBT, Apache Spark, Python/Pandas, FluentD, Apache Drill, Dataflow oriented tools



### Databases/Storage

Apache Hadoop, Apache Cassandra, CouchDB, MongoDB, ClickHouse, InfluxDB, Redis, Neo4j, Postgres, MariaDB, MySQL, CockroachDB, DuckDB

**See also:** Apache Hudi, Apache Iceberg, and Delta Lake



### Orchestration

Apache Airflow, Dagster, Prefect, Argo, Luigi, Kestra



### COST OPTIMIZATION

What are our ways of monitoring cloud costs? FinOps?



### REPORTING REQUIREMENTS

How often to we need reports and who needs them?



## — Data Quality and Health

- Can be impacted at **any stage** of the data life cycle
- Data quality issues are **unavoidable** at scale
- **Not all metrics are equal** and worth your time depending on your business model
- Paramount to gaining stakeholder and data consumer **trust**
- Write-Audit-Publish pattern



# 5 PILLARS OF DATA OBSERVABILITY

## **FRESHNESS**

Are your tables updating at the right time?

## **DISTRIBUTION**

Is the value within a normal range?

## **LINEAGE**

How are data assets connected across your data stack upstream and downstream?

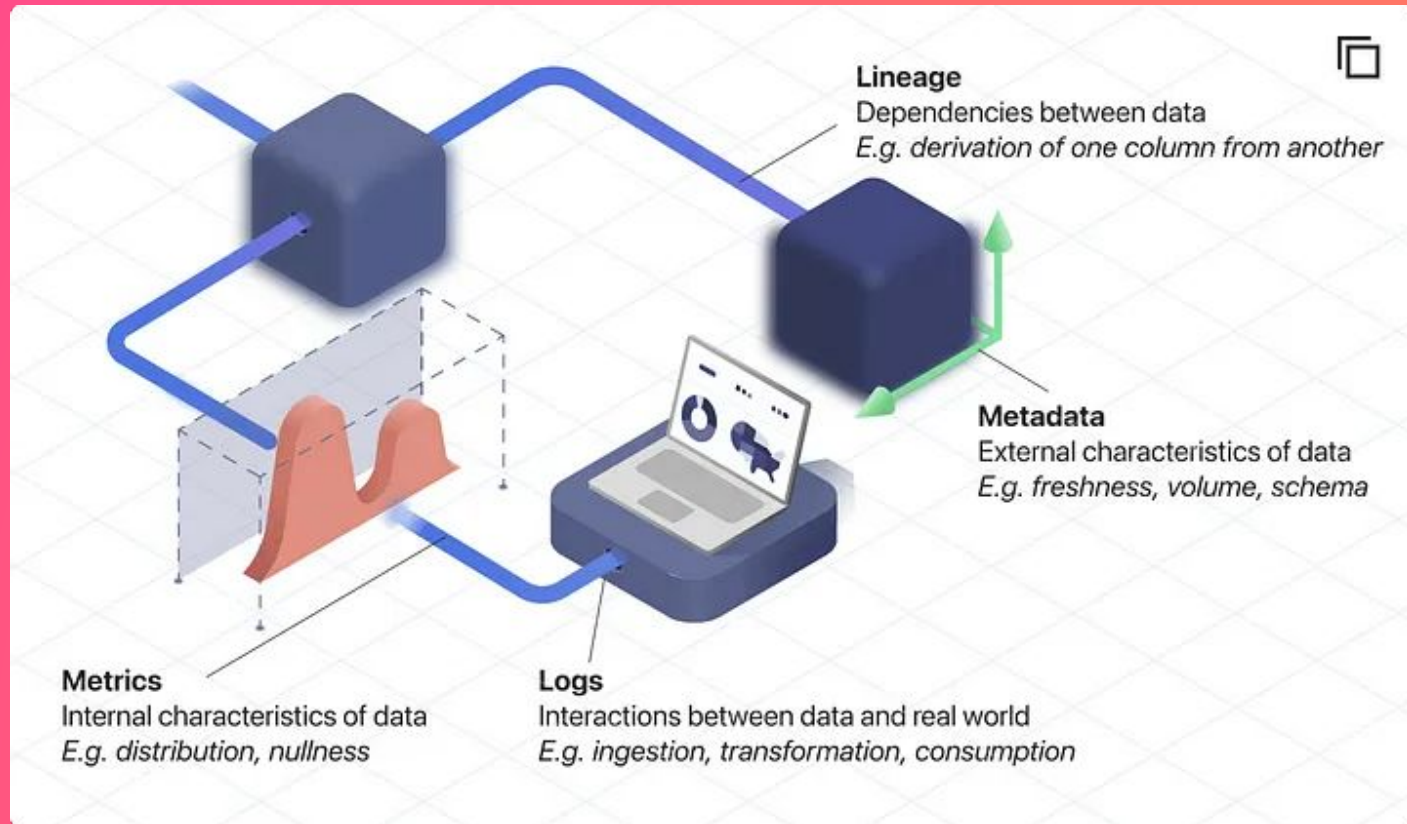
## **SCHEMA**

Has the organization of the data changed?

## **VOLUME**

Do you have too many or too few rows?

# Targets for Observability



## Software Observability



## Data Observability



### Nature of issues

Inevitable, compounds over time, disruptive

### Contracts

Explicit and implicit contracts with system users

### Issue identification

Historical data is used with rules and anomaly detection

### Ideal state

Self-healing, resilient systems that prevent issues at the source and alert on symptoms immediately with context

### Roles

Software engineers, DevOps engineers, Site Reliability engineers

Data engineers, analytics engineers, data / business intelligence analysts

### Pillars

Traces, logs, metrics

Metrics, metadata, lineage, logs

### Root cause analysis

Infrastructure dependencies + version-controlled code

Data lineage + version-controlled code

### Impact analysis

Machine-to-machine dependencies with clear path to identify user impact

Trace data lineage with frequently unclear user impact

### Remediation

Well-defined playbooks to identify, communicate, resolve, and learn

Ad hoc processes to remediate issues

### Technologies

Compute and storage at either low-level or application-level abstraction

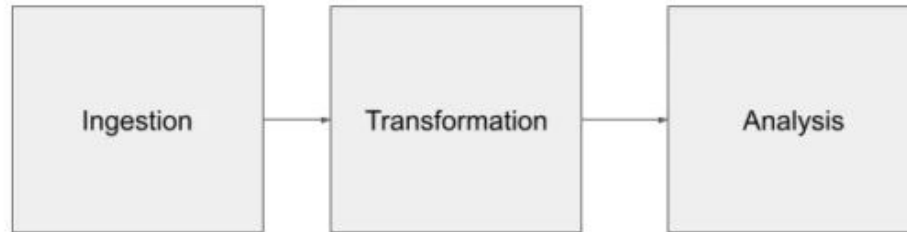
ELT, cloud data warehouse, transformation, BI tools

[Image from Kevin Hu](#)

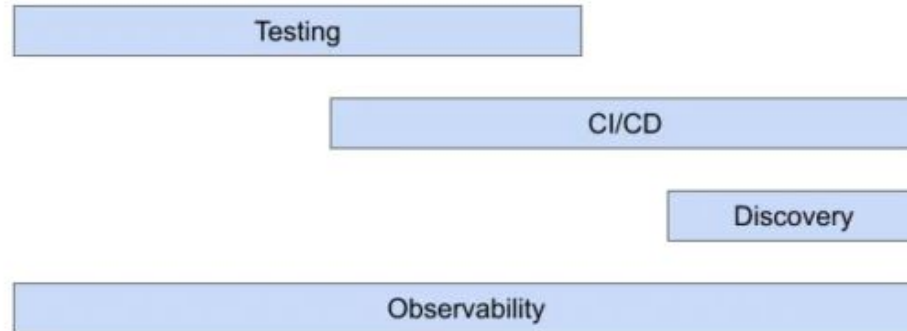
## — Open Source Data Quality Tools

- **Great Expectations** – Assertion Based
- **DataCleaner**
- **Apache Griffin** – Model-driven
- **Deequ** – AWS, build on Apache Spark
- **DBT** Unit Testing
- **Apache Airflow** – Circuit Breakers, SQL Check Operators

**Data processing stages**



**Data reliability stack**



[Image from MonteCarlo Data](#)

# DataOps

## – Infrastructure as Code

1. **Data Pipeline as Code**
2. **Version control** said code
3. **CI/CD** the repository
4. Automate **deployments**
5. **Declarative configuration** (including schemas) to match target and states
6. **Rollbacks and auditing as needed**
7. Collaboration and **branching**
8. **Observability and monitoring** of pipeline components
9. **Secrets and Credential Management**
10. **Documentation**



### Version Control

Git/GitOps  
Framework

DVC (Data Version  
Control) - git like

Debezium - CDC  
oriented



### Data Contracts

Define the  
structure, format,  
and semantics of  
data exchanged

Avro, Protocol  
Buffers, and JSON  
Schemas

"An Engineer's Guide  
to Data Contracts"  
by Chad Sanderson



### Monitoring and Alerting

Track health and  
performance of  
data pipelines

Prometheus

Grafana

OpenTelemetry

SigNoz



### Deployment and Scaling

Docker,  
containerization of  
pipeline  
components

Kubernetes, Helm,  
and Terraform stack

Coupling  
considerations



### Documentation

Create Data  
Catalogs

Data Pipeline  
Incident Runbooks

Data Governance,  
Ownership

Data Lineage  
Models, Graphs

Non-Technical  
Documentation

# Data Pipeline Management

Automated pipeline monitoring and reactive 24x7x365 response by our global Rackspace DataOps team



Data Pipeline Monitoring



Ticketing and Alerting



Incident Management



Troubleshooting



Operational Support



Platform Escalations



Runbook Resolution



Service Level Agreement

Transformation can be disruptive and exhausting

Data Pipeline Management lets you focus on outcomes and innovation...  
and maybe getting some sleep at night

[Image from Rackspace](#)

## Data Incident Management

- Incidence detection (alerting systems, anomaly detection)
- Response (Runbooks, Playbooks)
- On call processes

## Root Cause Analysis

1. Investigate data lineage
2. Look at code
3. Look at the data itself
4. Consider the operational environment
5. Leverage cross-functional teams and peers

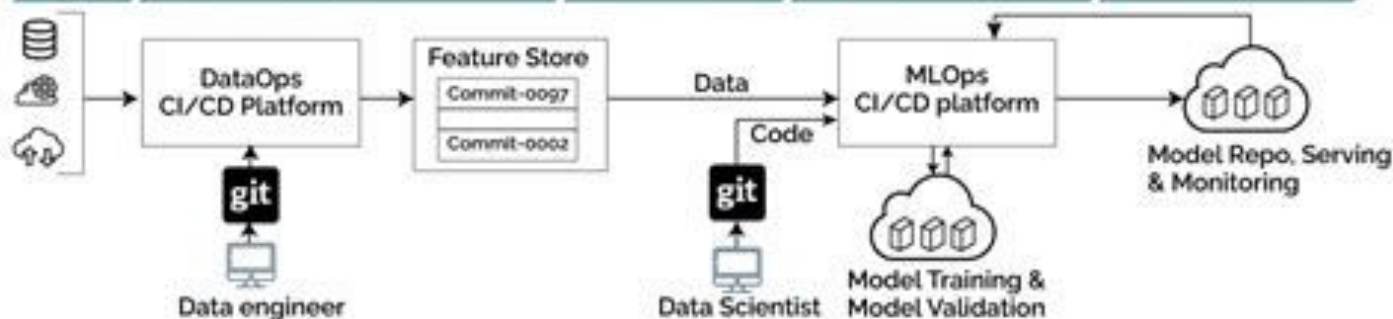
# MLOps= ML + DEV + OPS



**Experiment**  
Data Acquisition  
Business Understanding  
Initial Modeling

**Develop**  
Modeling + Testing  
Continuous Integration  
Continuous Deployment

**Operate**  
Continuous Delivery  
Data Feedback Loop  
System + Model Monitoring



## — MLOps Components

- Model and data asset **testing, validation, reporting**
- **Deployment of models** and AI-driven applications
- **Model management and serving**
- Developing and versioning ML code, **experiments, processes, and parameters**
- Continuous **drift detection** and monitoring
- Automation and streamlining of **ML workflows**



# MLOps Roadmap

**Version control ML Code and artifacts** (includes models, training scripts, preprocessing code, Jupyter Notebooks, performance metrics, metadata) **into Git repo**

1

**CI/CD that triggers model training**, including testing, validation, evaluation, and reporting

3

**Deploy and monitor at scale**, use declarative configuration where possible

5

**Model Versioning using Git tags** and link to versions of data assets with each iteration

2

**Package models** with traceable data and code dependencies through containerization

4

**Manage security, documentation, ownership**, while compounding with DataOps strategies previously mentioned!

6



### Model Development

Pandas, Numpy

SciKit-Learn

Tensorflow

Jupyter Notebooks



### Testing and Validation

Great Expectations

Pytest



### Model Management and Serving

MLflow - ML lifecycle Management

Kubeflow -  
Orchestration and  
deployment w/ K8s

Talos - Hyperparameter  
optimization

Kedro - Pipeline  
framework

Seldon - Models as  
microservices on K8s



### Drift Detection and Monitoring

Frouros, Evidently,  
deepchecks,  
nannyML - Python  
libraries for drift  
detection in  
machine learning



### Documentation

Sphinx

Graphiz

Custom reports

# Thank You!

Any questions? You can find me at **@lisancao** &  
**linktr.ee/lisancao**