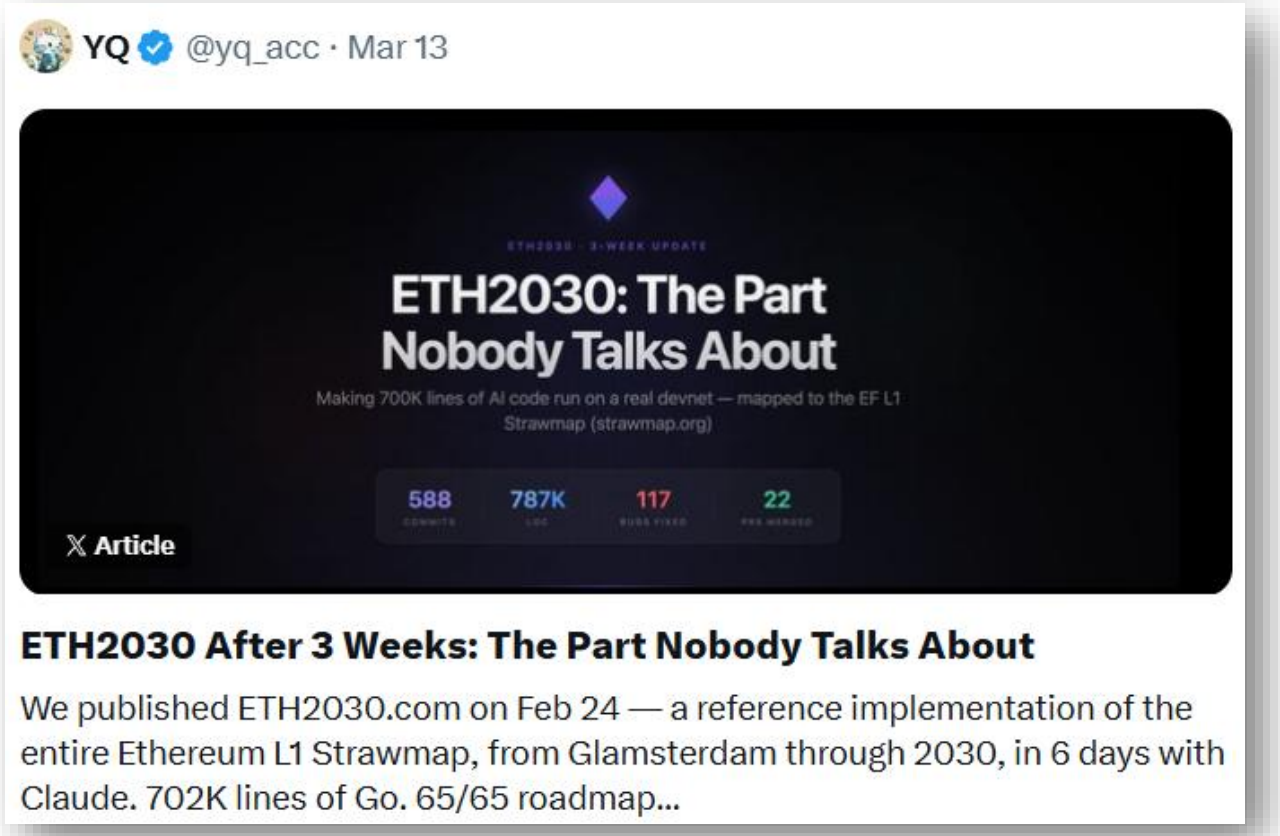
CS4262/5462 Machine Learning Systems
ML Operations

Yao LU

26 Mar 2026

National University of Singapore
School of Computing

AI Coding = ?



https://x.com/yq_acc/status/2032377910733259141?s=20

AI Coding = knowledge + ops

20%

What's in the spec

EIP text, Strawmap items, consensus specs

WHAT THE AI CAN FIND VS. WHAT YOU MUST PROVIDE

80%

Expert knowledge the AI doesn't have

Which fields Geth includes in the block hash (not in any spec)
How Lighthouse orders Engine API response fields
What gas values break Kurtosis devnet consensus
Which 18 EIPs interact in Glamsterdam repricing
How EIP-7706 composes with EIP-1559 and EIP-4844
The L1 Strawmap itself (published Feb 25, not in training data)
Lock contention patterns in concurrent block processing
Memory growth curves under real P2P transaction load

- **29%** — Cross-component (agent sees one module at a time)
- **19%** — State lifecycle (no model for 10K iterations)
- **15%** — Fork rules (EIPs applied in isolation, not composed)
- **13%** — Concurrency (sequential logic, misses races)
- **12%** — Protocol compat (Geth/Lighthouse undocumented behavior)
- **9%** — Gas pricing (18+ EIPs, wrong era's formula)
- **3%** — Tests that tested wrong behavior

- Knowledge & vision is the king

- Dev ops + data ops are the knight

- AI coding is the pawn

Agenda

- A real case from Prof. Dianbo Liu
  - Data operations

- ML Operations = dev operations + data operations
  - Concepts & product goals
  - Design choices: the data cascade & models
  - Deployment & operation
  - Monitoring & Analytics
  - Some tooling in production

Prof. Dianbo's sharing (data operations)

- Define the data in medical scenarios
- How data is generated
- How data is managed
- How models are trained
- How labeling & evaluation is done

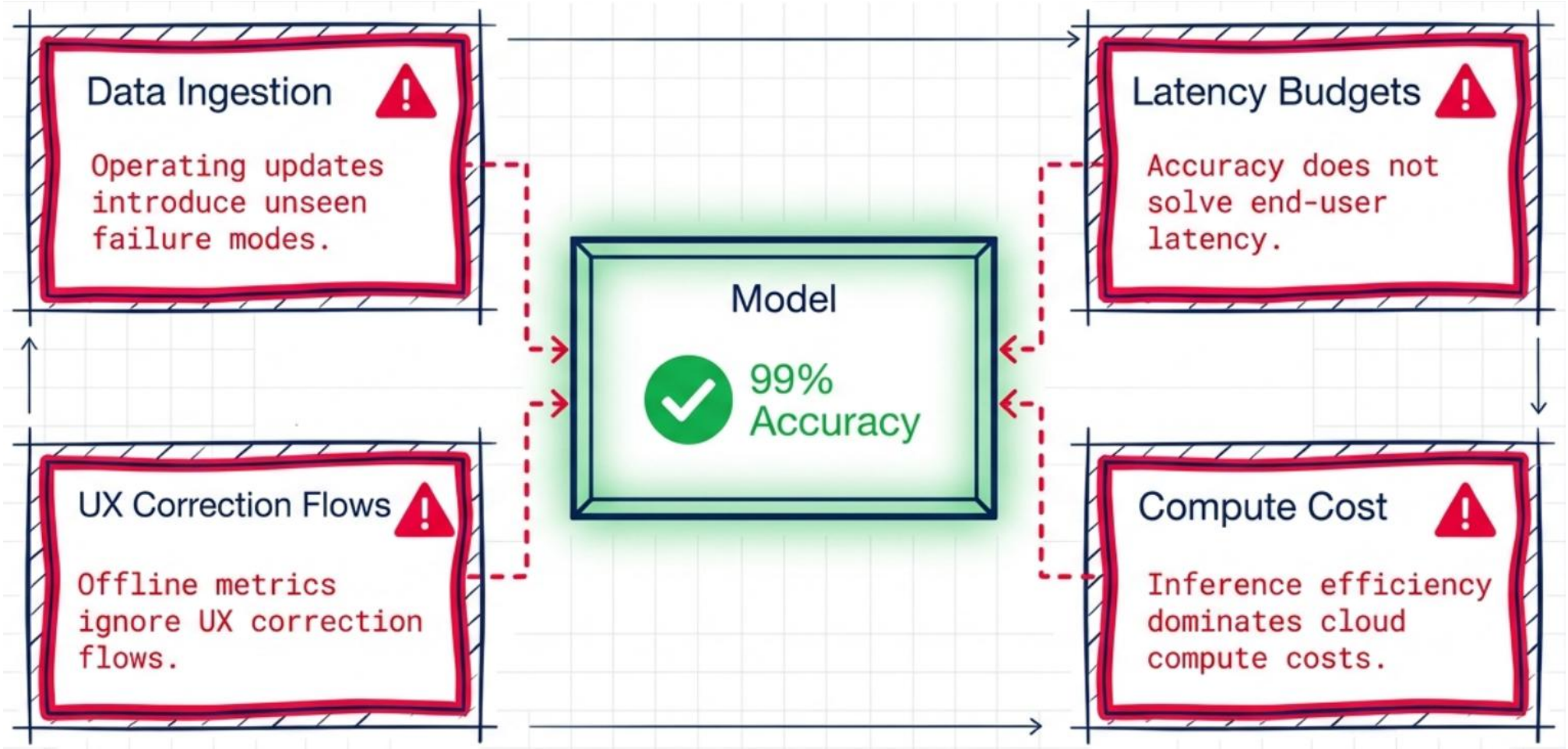
Agenda

- A real case from Prof. Dianbo Liu
  - Data operations

- ML Operations = dev operations + data operations
  - Concepts & product goals
  - Design choices: the data cascade & models
  - Deployment & operation
  - Monitoring & Analytics
  - Some tooling in production

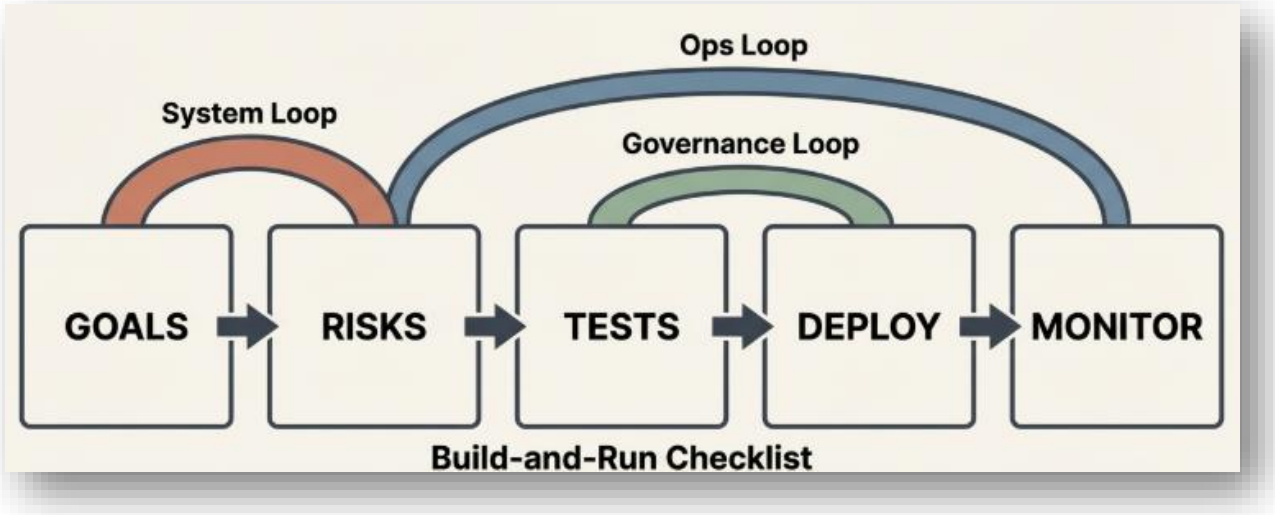
A transformer model might boost accuracy by 1%, but increase cost by 100%. Is it worth it?

Great models ≠ reliable services



Dev ops lifecycle

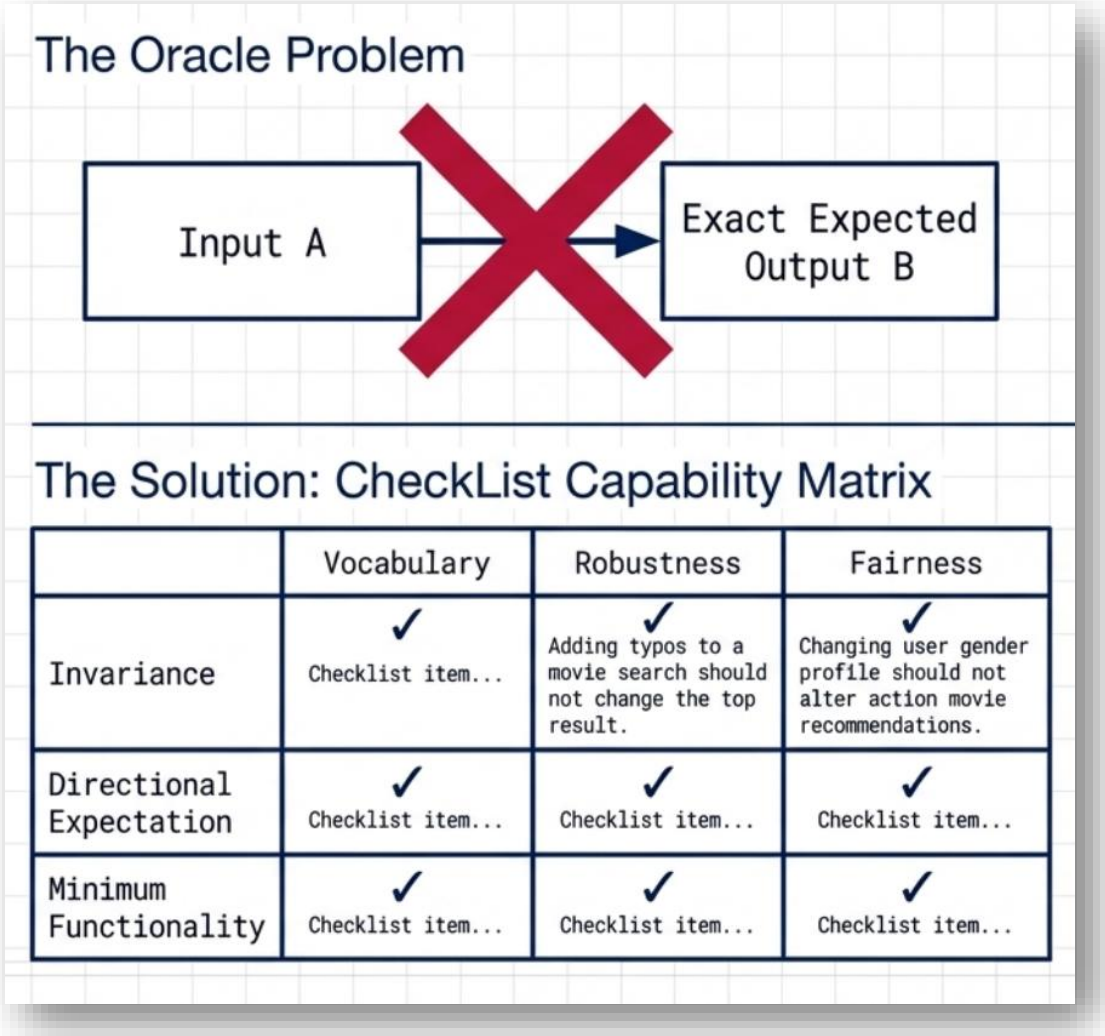
- **System loop:** define goals and outcomes before writing code.
- **Ops loop:** instrument the system to monitor, detect, and respond to drift
- **Governance loop:** enforce responsibility, lineage, and trust through tooling



Behavioral testing with checklist

- We lack a perfect mechanism to assert exact outputs for unseen inputs.

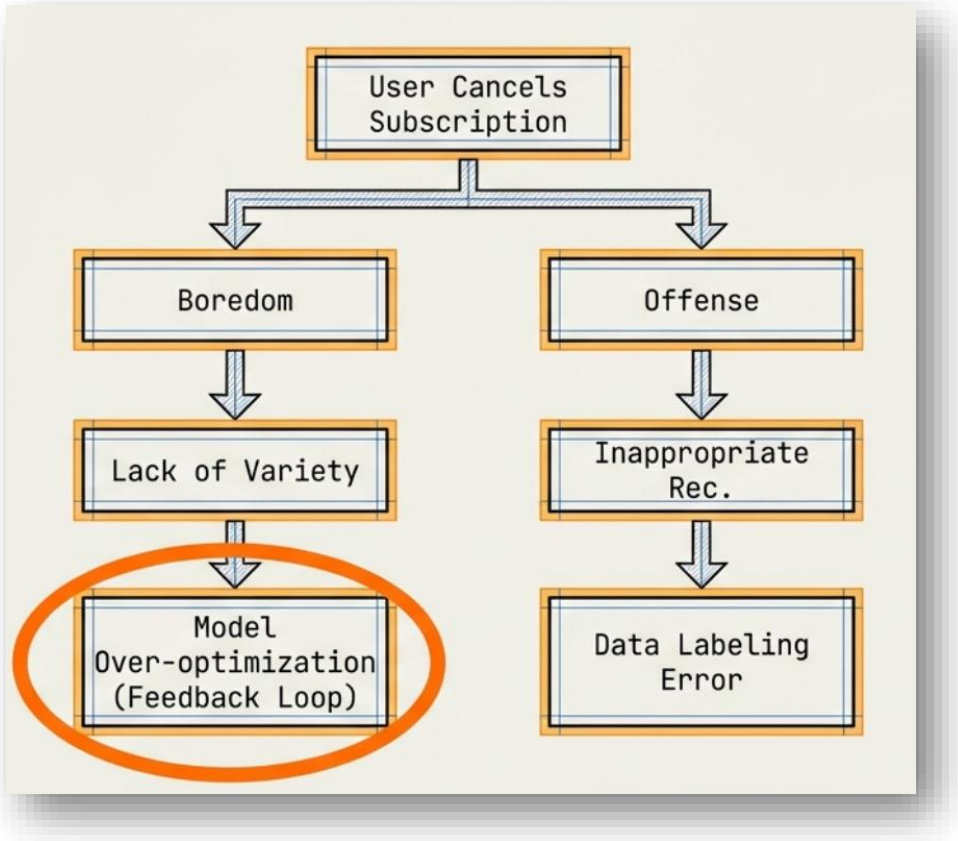
- Instead, test behavioral invariants, not just aggregate metrics.



Handling ambiguity & planning for mistakes

- Requirements in ML rely on assumptions. When the world changes, the machine fails silently.

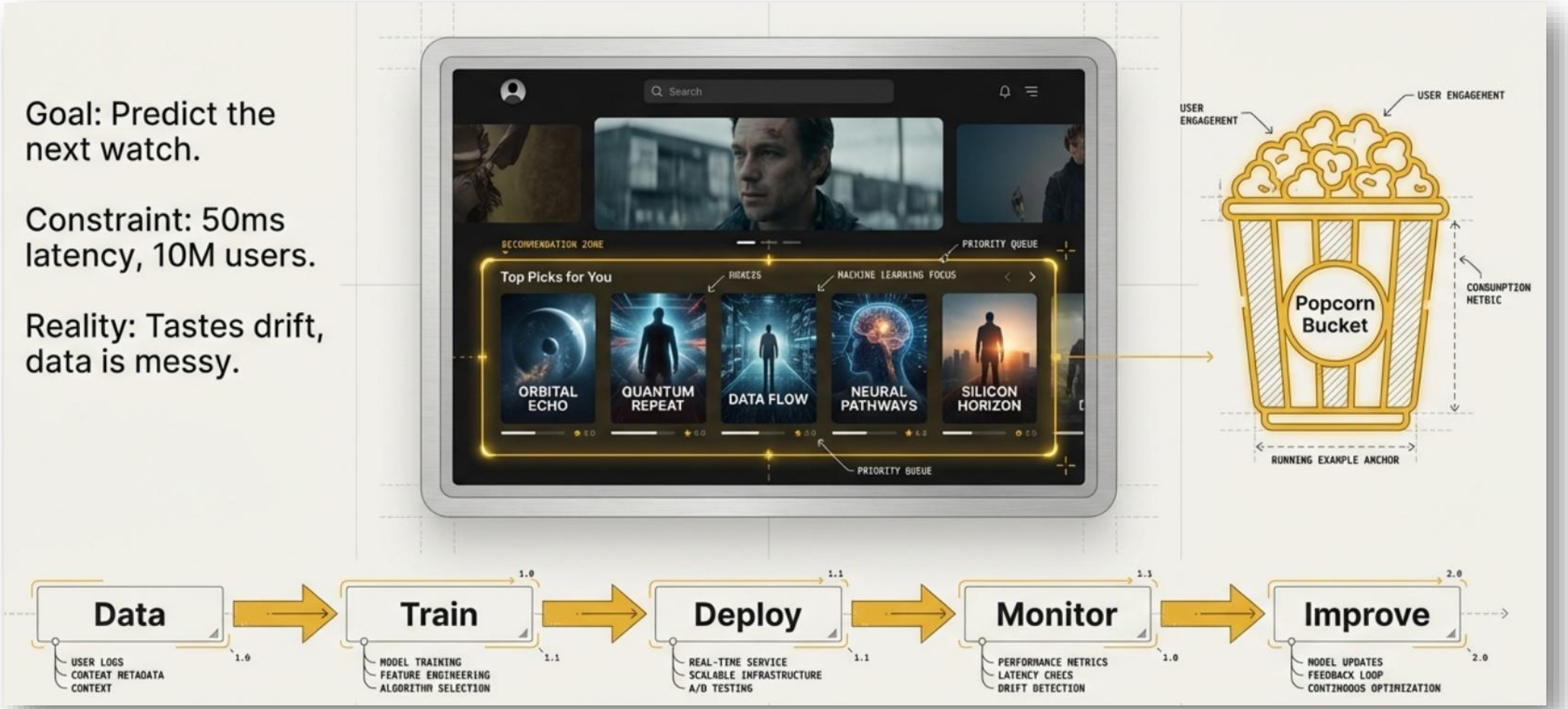
- Predict the failure before writing code



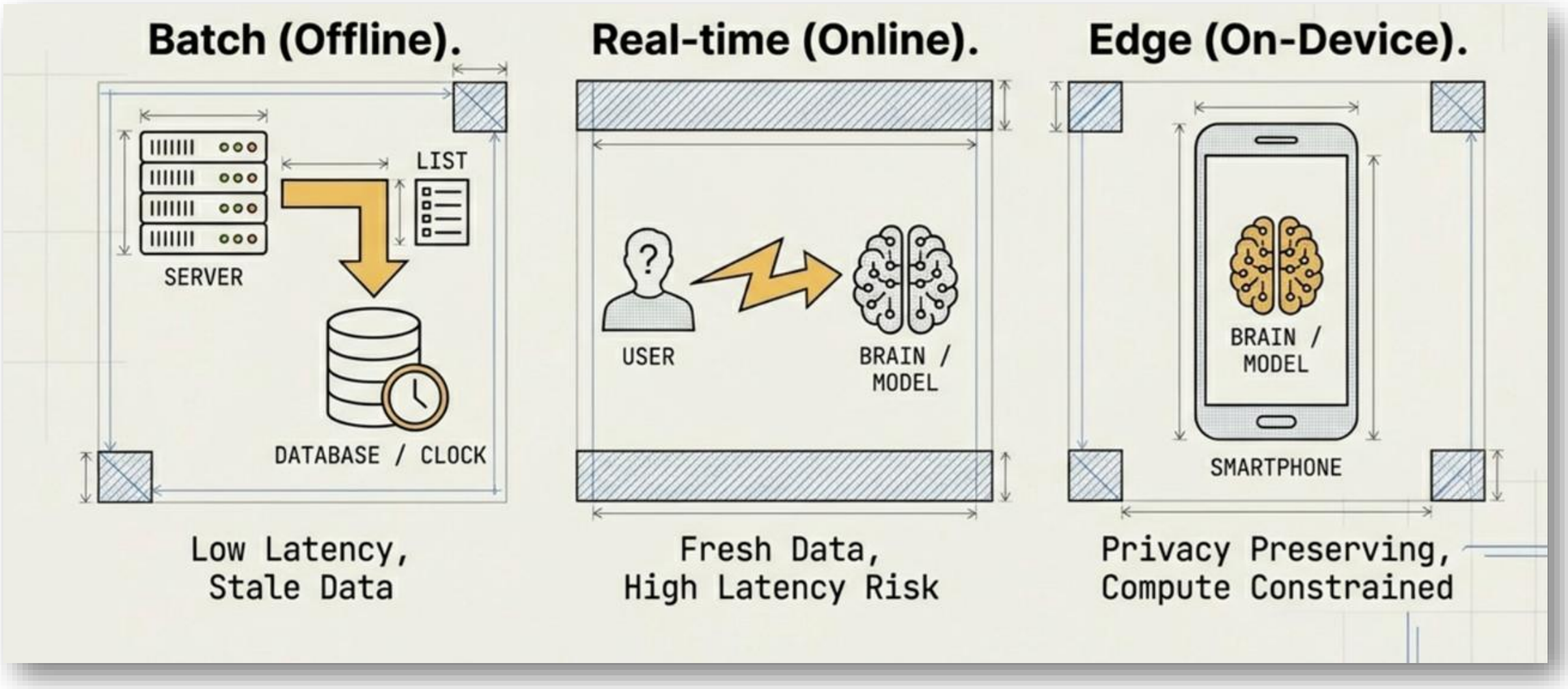
ML ops testing with checklist

- Data tests
  - Feature monitoring, data quality checks, schema validation
- Model tests
  - Versioning, reproducibility, fairness analysis
- Infra tests
  - Rollback capability, load testing, deployment validation
- Monitoring tests
  - Drifts alerts, anomaly detection, latency tracking

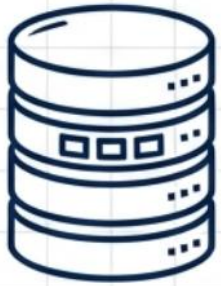
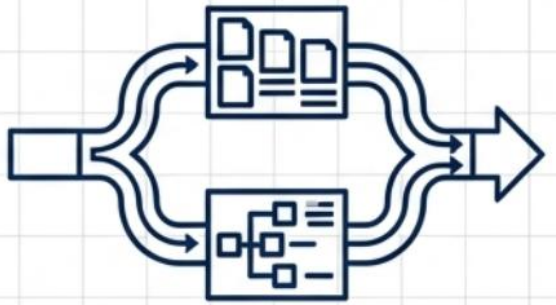
The movie recommendation engine



Where does the intelligence live?

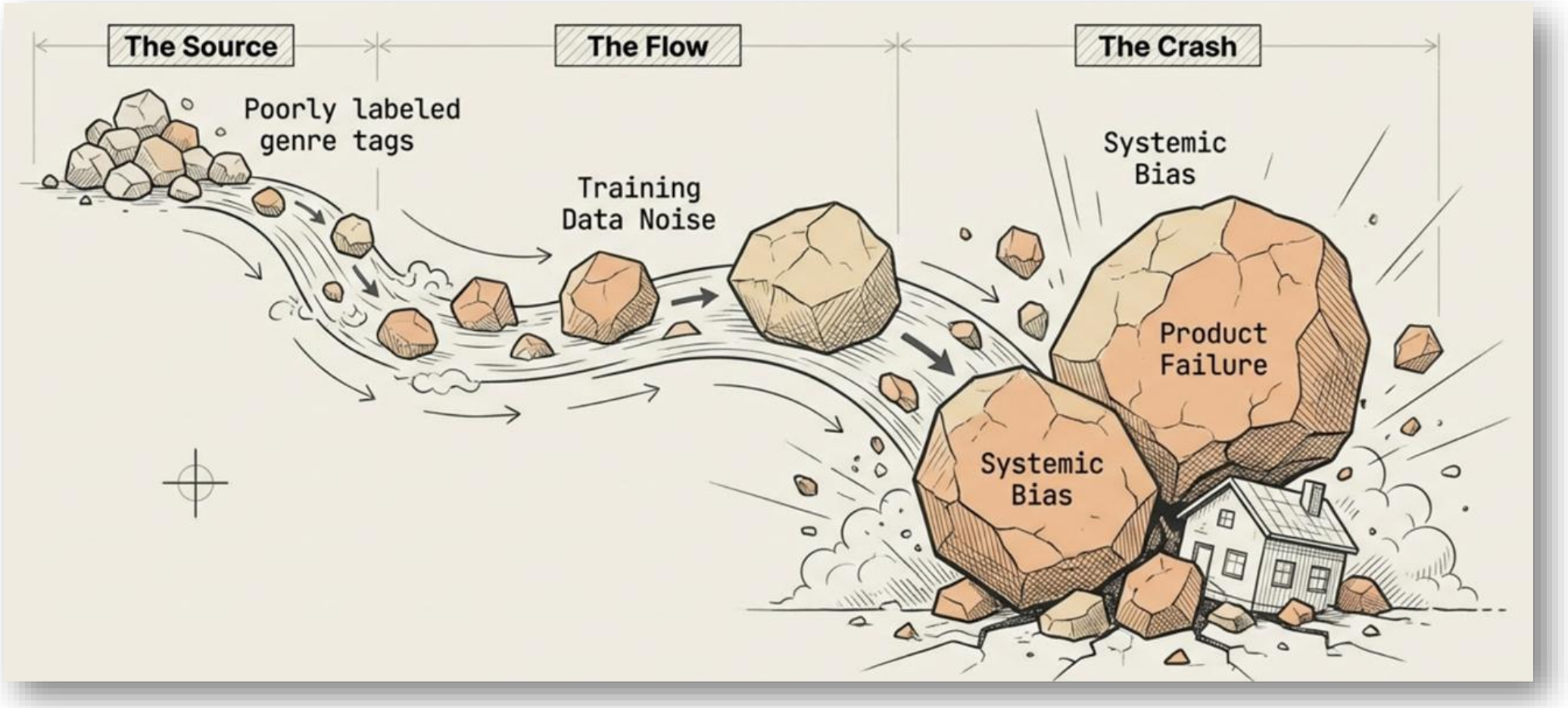


Data infra trade-offs

Batch Architecture	Stream Processing (Kafka)	Lambda Architecture
		
• Pre-compute recommendations nightly. • Low risk, high latency. • Blind to intraday user trends.	• Update features in near-real-time based on live clicks. • High infrastructure cost. • Requires rigorous schema event retention.	• Blends offline batch and online stream. • Highest complexity. • Requires careful separation to avoid training-serving skew.

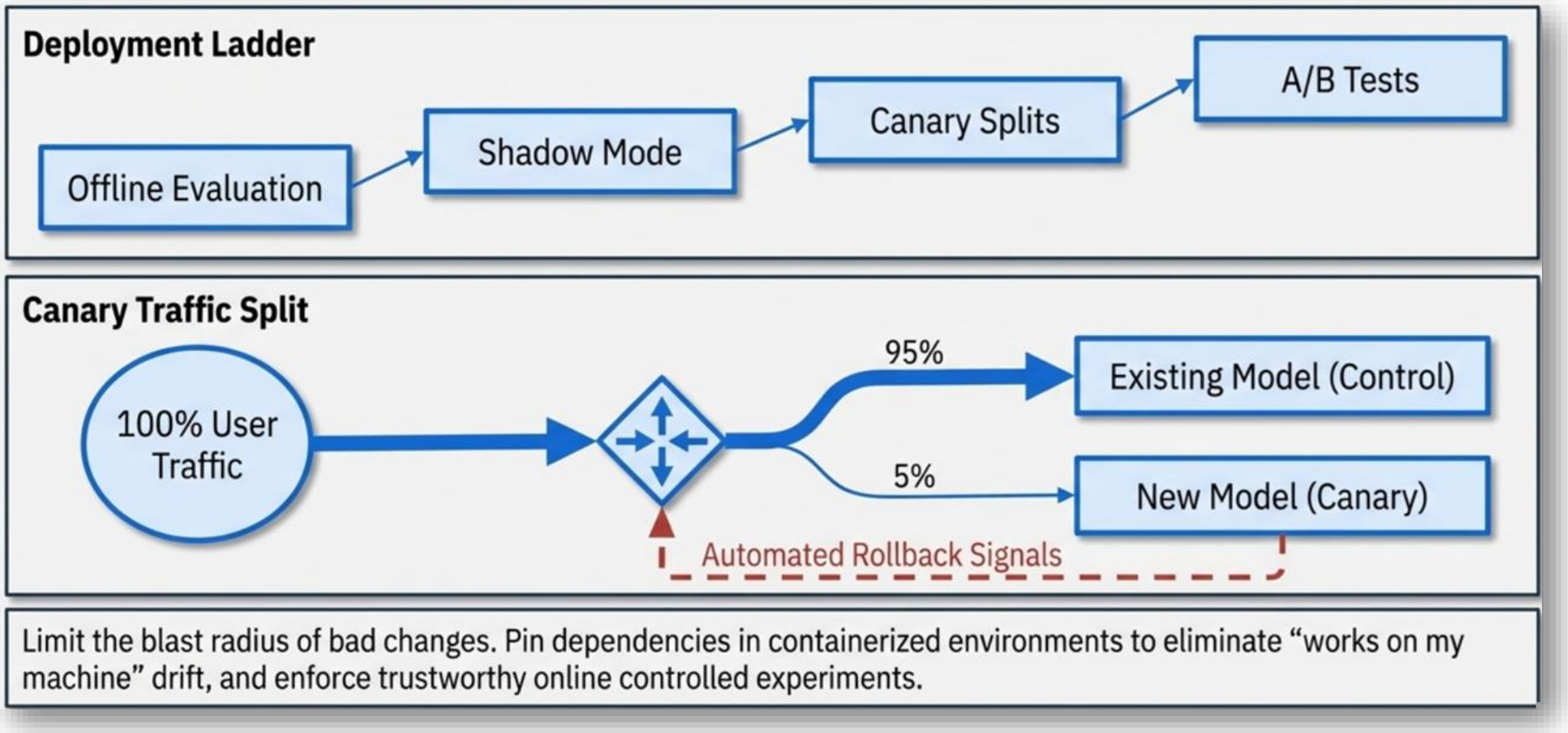
Data cascades: garbage in, disaster out

- Upstream data issues compound into downstream product failures.

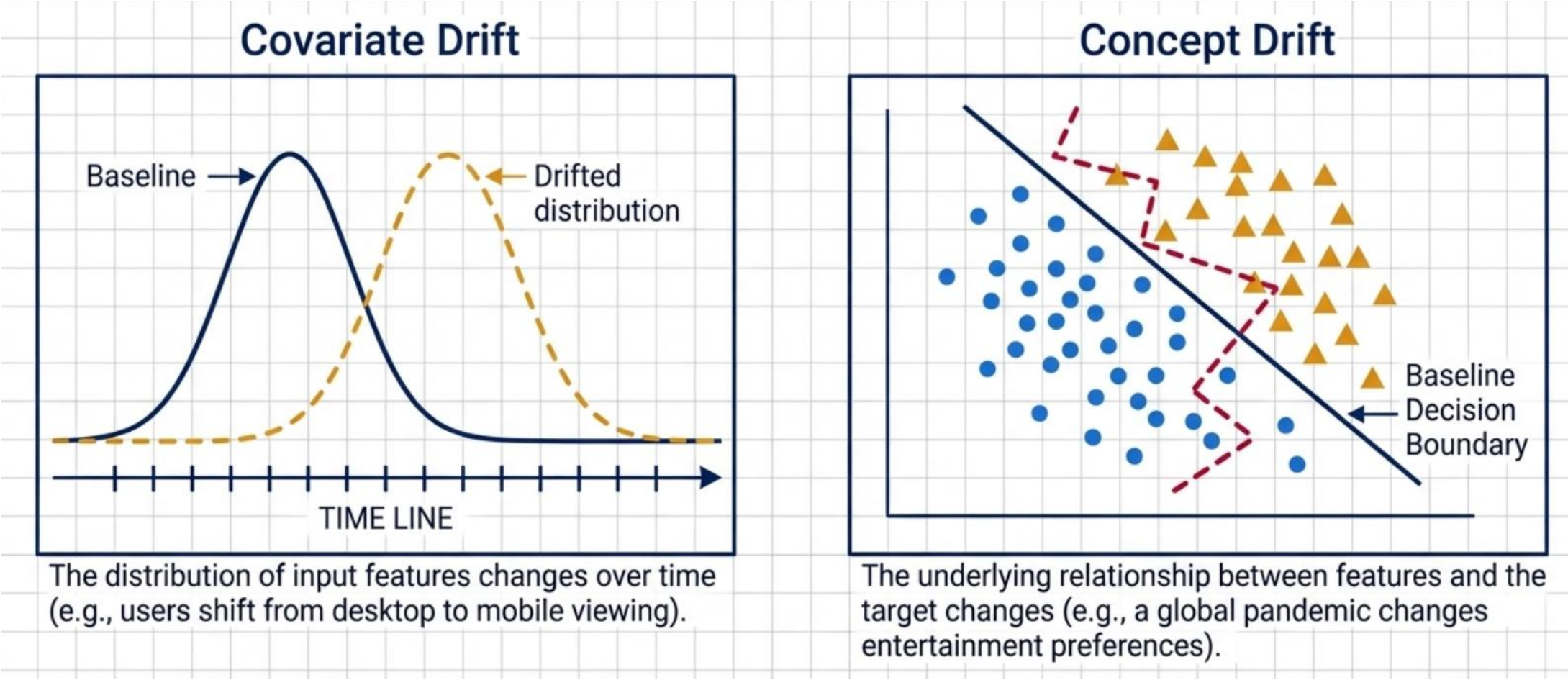


Canary releases: limiting the blast radius

- Deploy code ≠ release feature. Rollout slowly and rollback instantly.

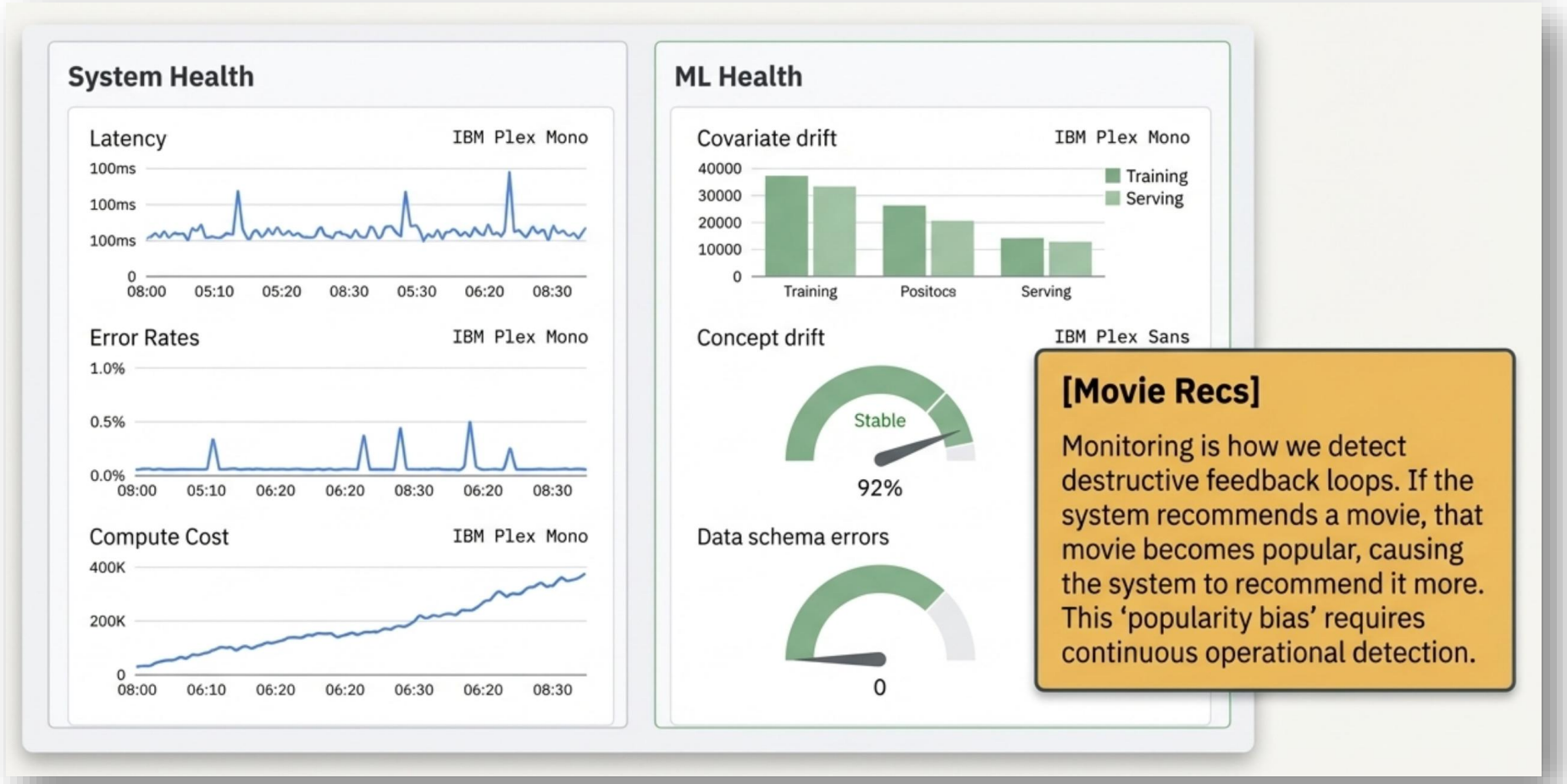


Drift: the only constant



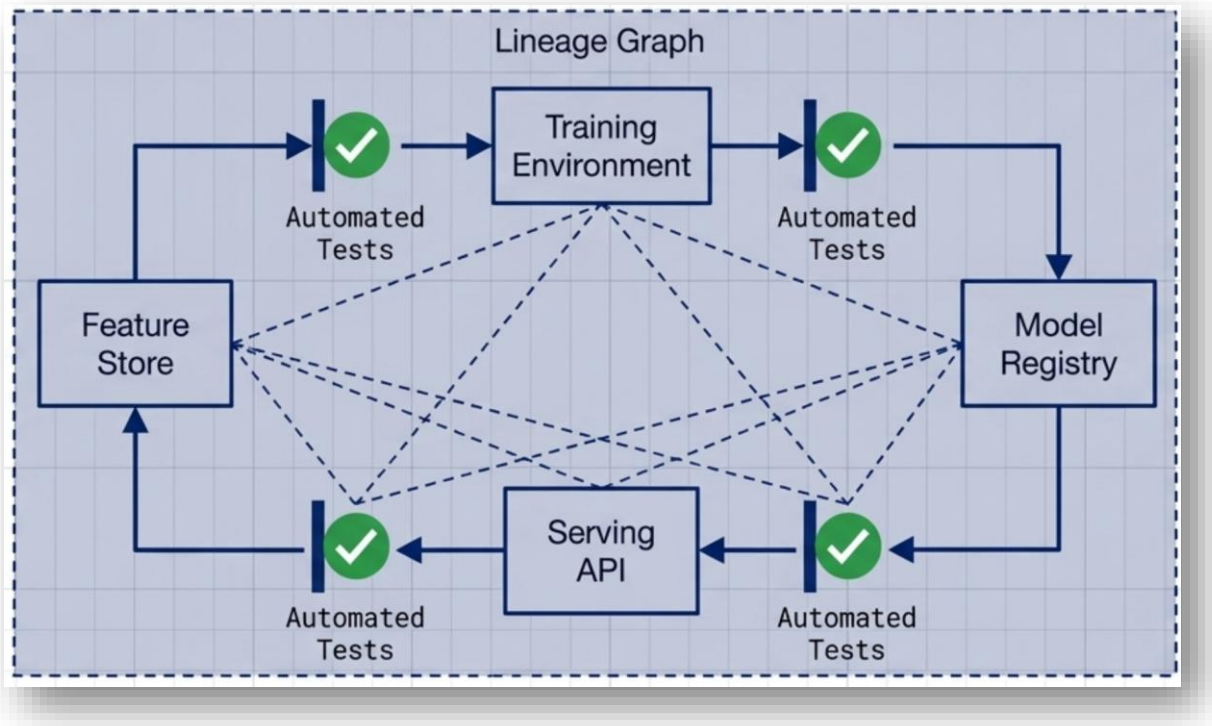
Observability: telemetry doesn't come free

- Watch for silent failures where the system returns OK but serves garbage.



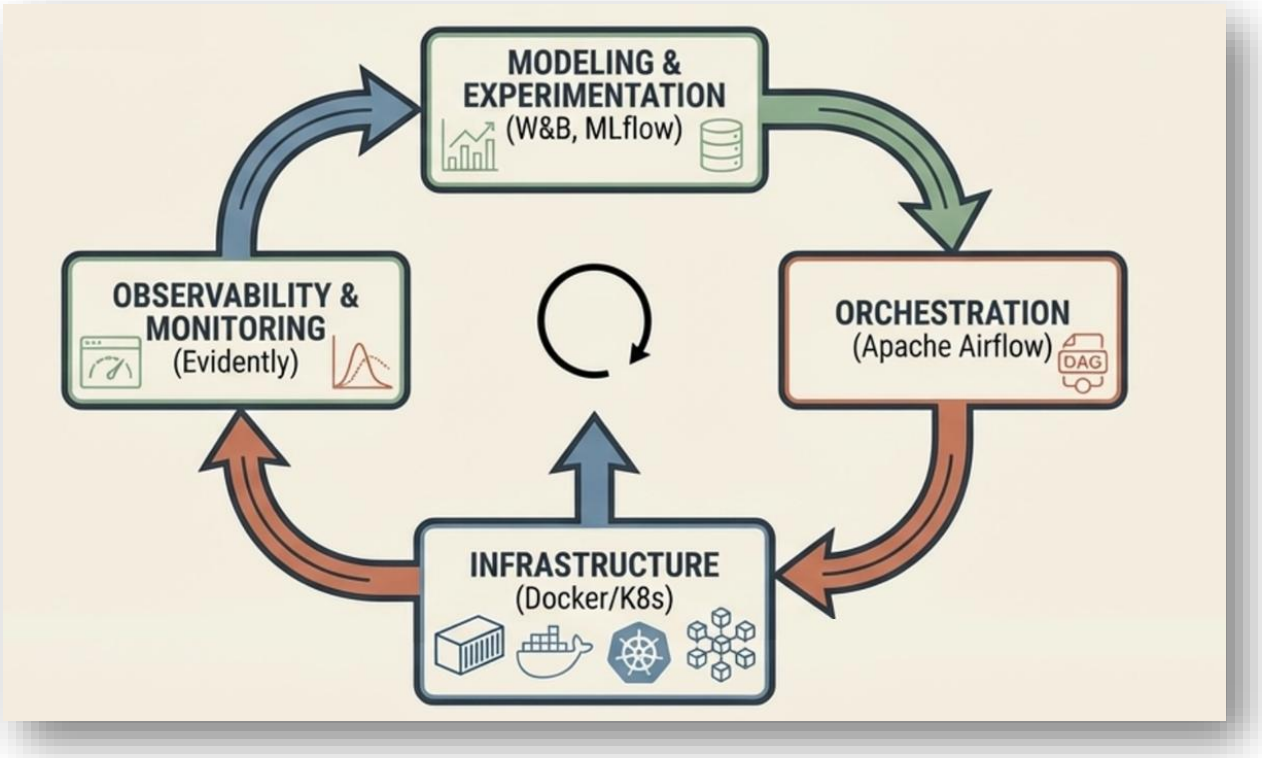
ML ops lifecycle

- **Versioning:** code, data, configs, models
- **Environment:** pin dependencies in containerized environments
- **Provenance and lineage** are engineering artifacts
- **Automating tests**: promote tested artifacts through automated gates



Some sample tools

- **Coding:** Claude Code & OpenAI Codex
- **Github:** dev ops (CI/CD/CT)
- **Infra:** docker & Kubernetes as the scalable foundation
- **Modeling:** Weights & Biases and Mlflow track, version, and manage the model
- **Orchestration:** Apache Airflow moves the data and triggers the jobs.
- **Observability:** Evidently watches the output, feeding insights back into the loop for continuous retraining

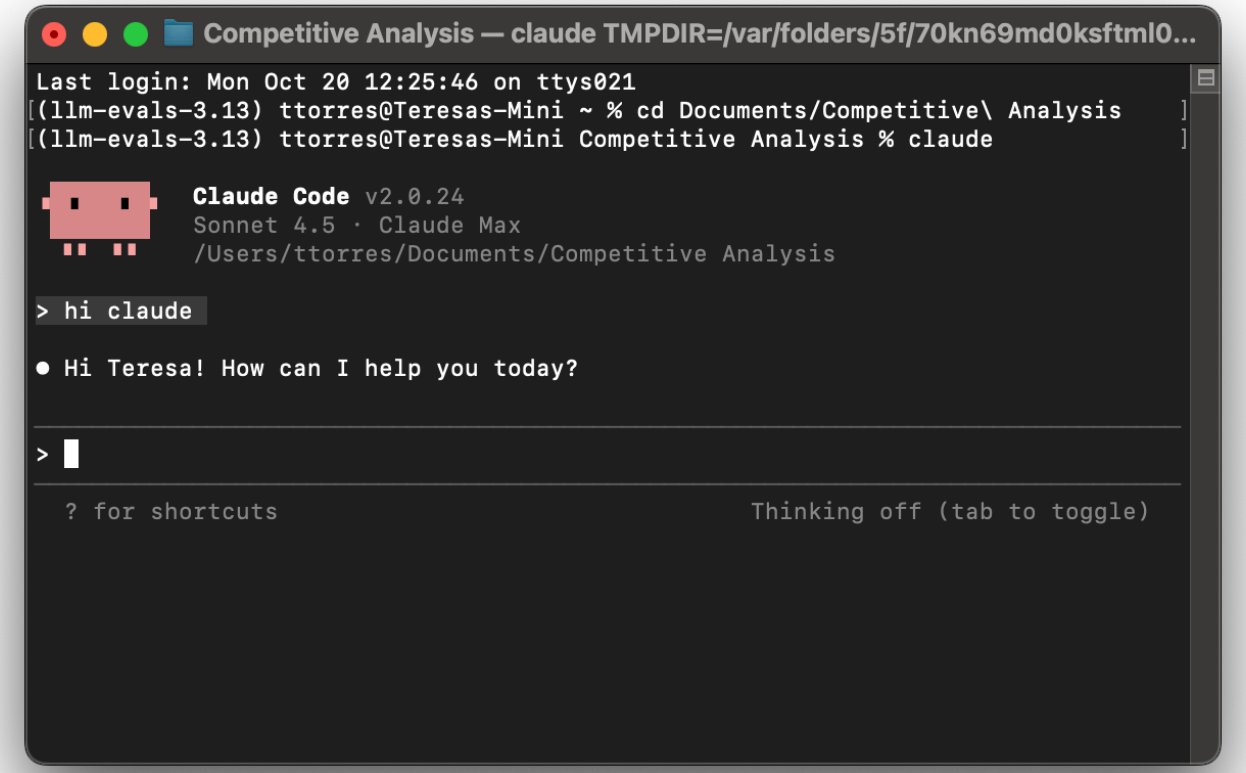


Claude Code & OpenAI Codex

- Strong coding agents

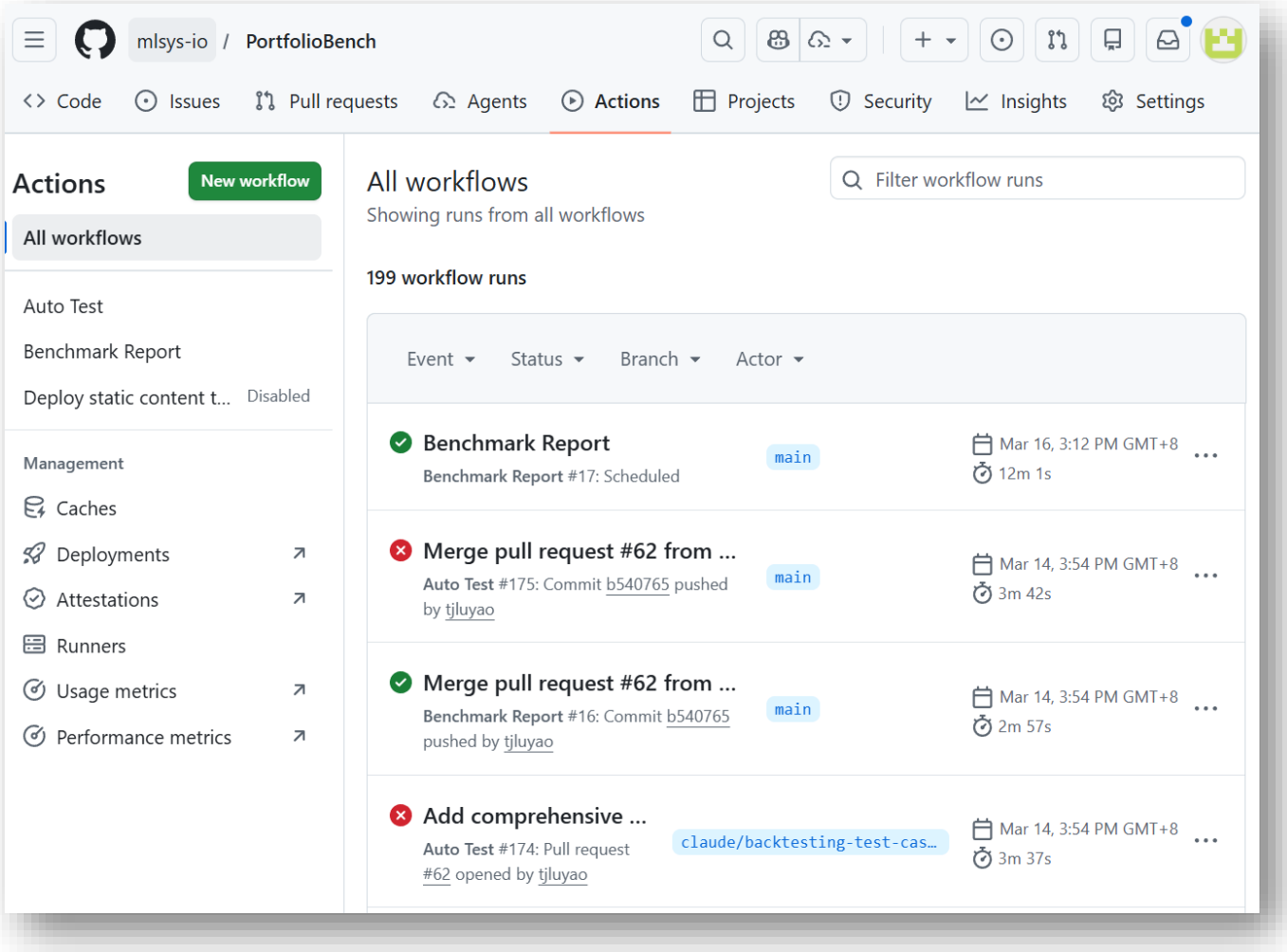
- But different people + AI coding = 10-1000x speedup. What makes a different includes
  - **Vision**
  - **Architecture**
  - **Ops, Ops, Ops!**

    (dev ops + data ops = ml ops)



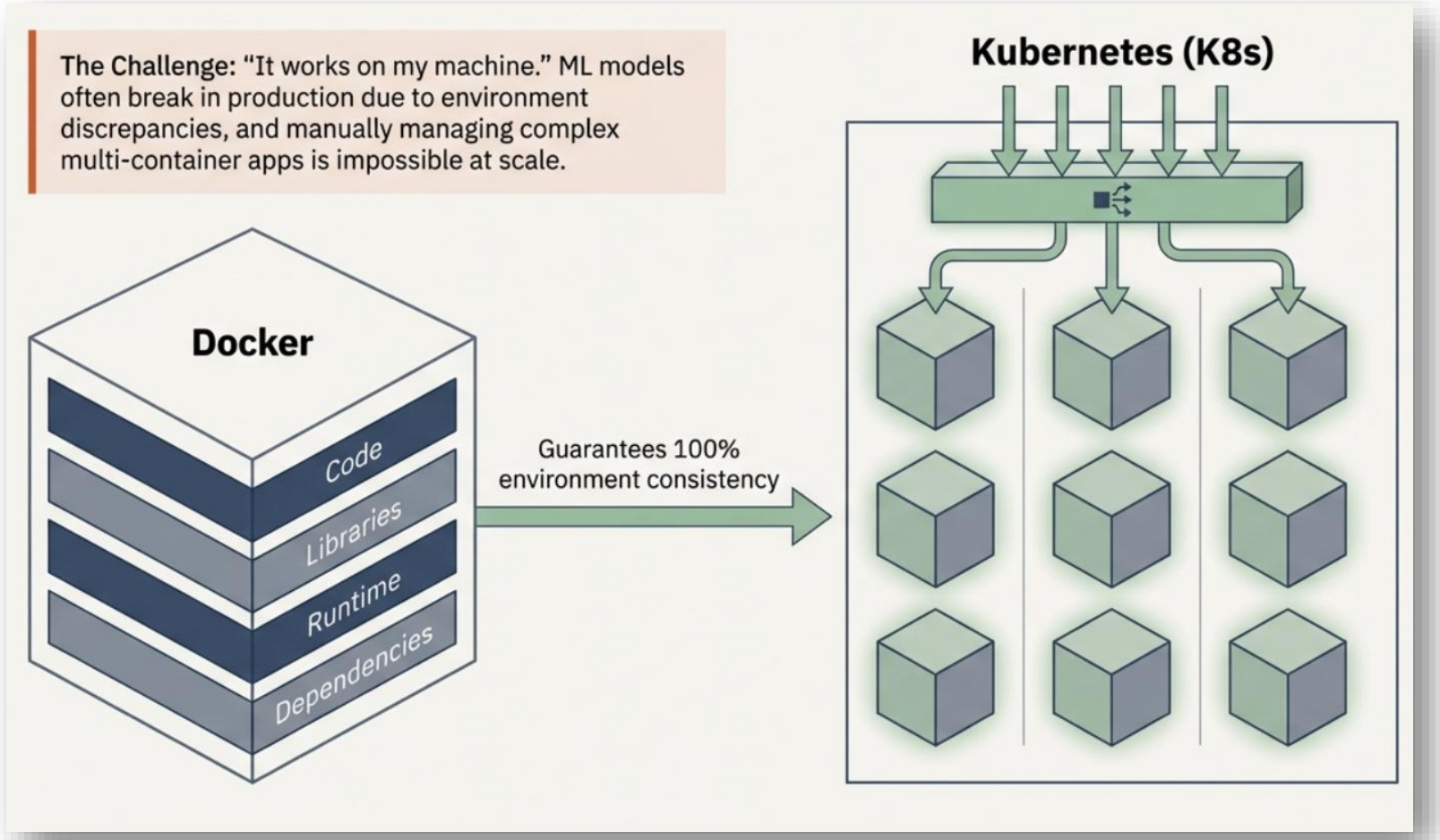
Github (you often forget)

- Critical tool for AI coding to help you reach 100x+
  - **Versioning & teaming:** you should be familiar with this
  - **CI/CD:** build, test, deploy, release
  - **CT:** testing metrics beyond “just compiles”
  - **Reporting:** performance & feedbacks



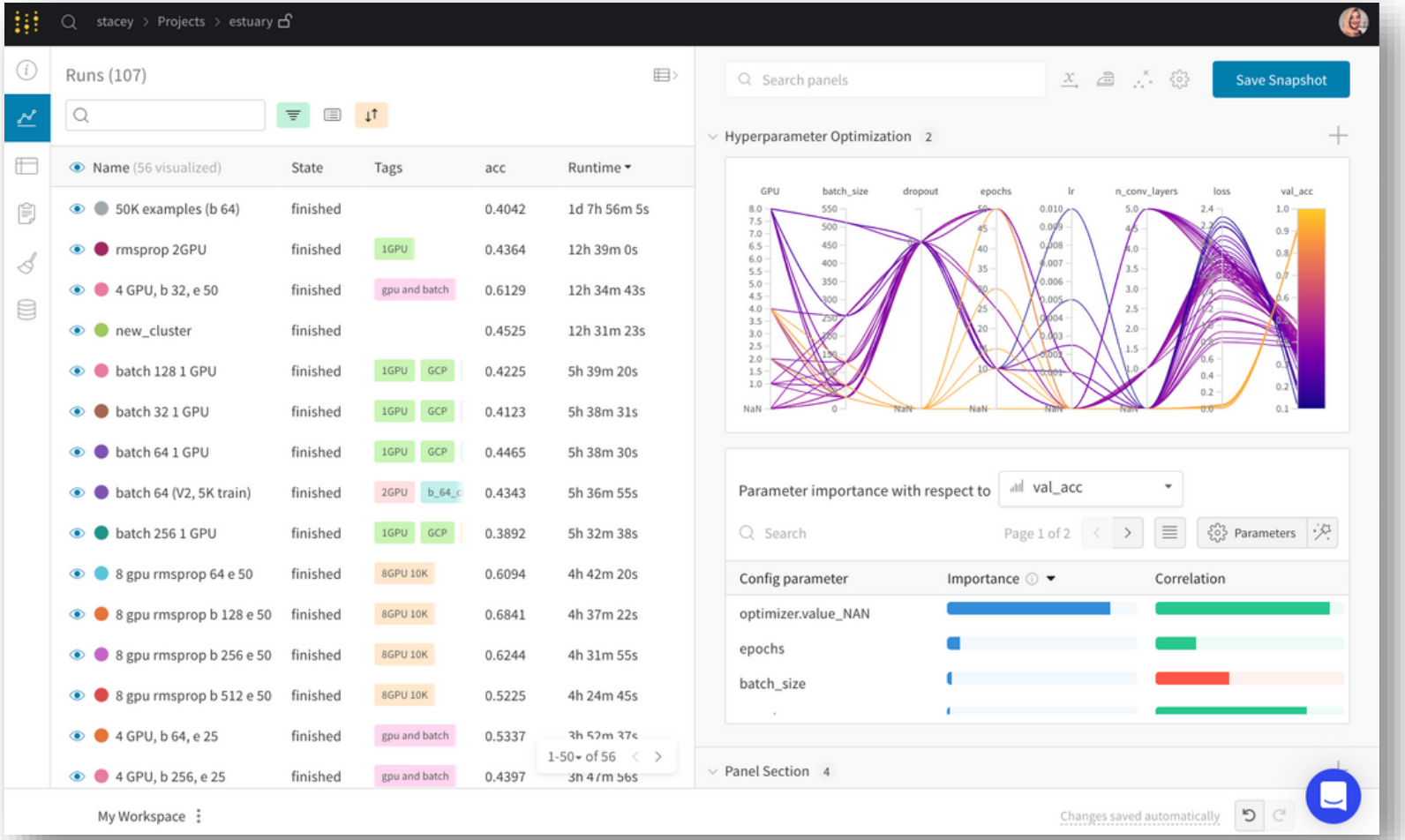
Docker & Kubernetes

- Automates deployment, failure handling, scales resources dynamically
- Handles heterogeneity
- Our short tutorial will be built upon dockers



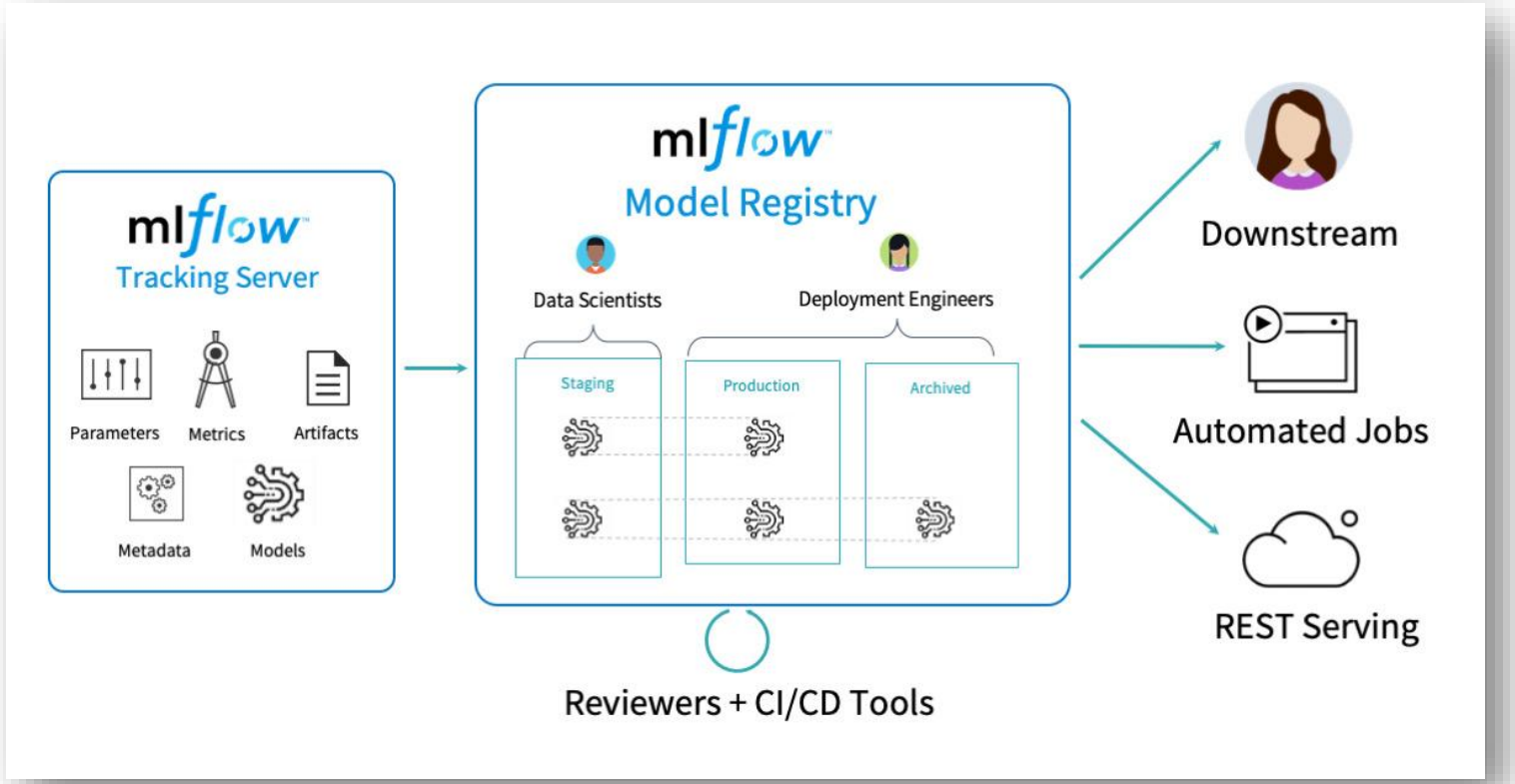
Weights and bias

- A developer tool for centralized experiment tracking away from scattered notes and local files
- Allow ML researchers / engineers to easily log metrics during training, optimize hyperparameters, visualize models and collaborate



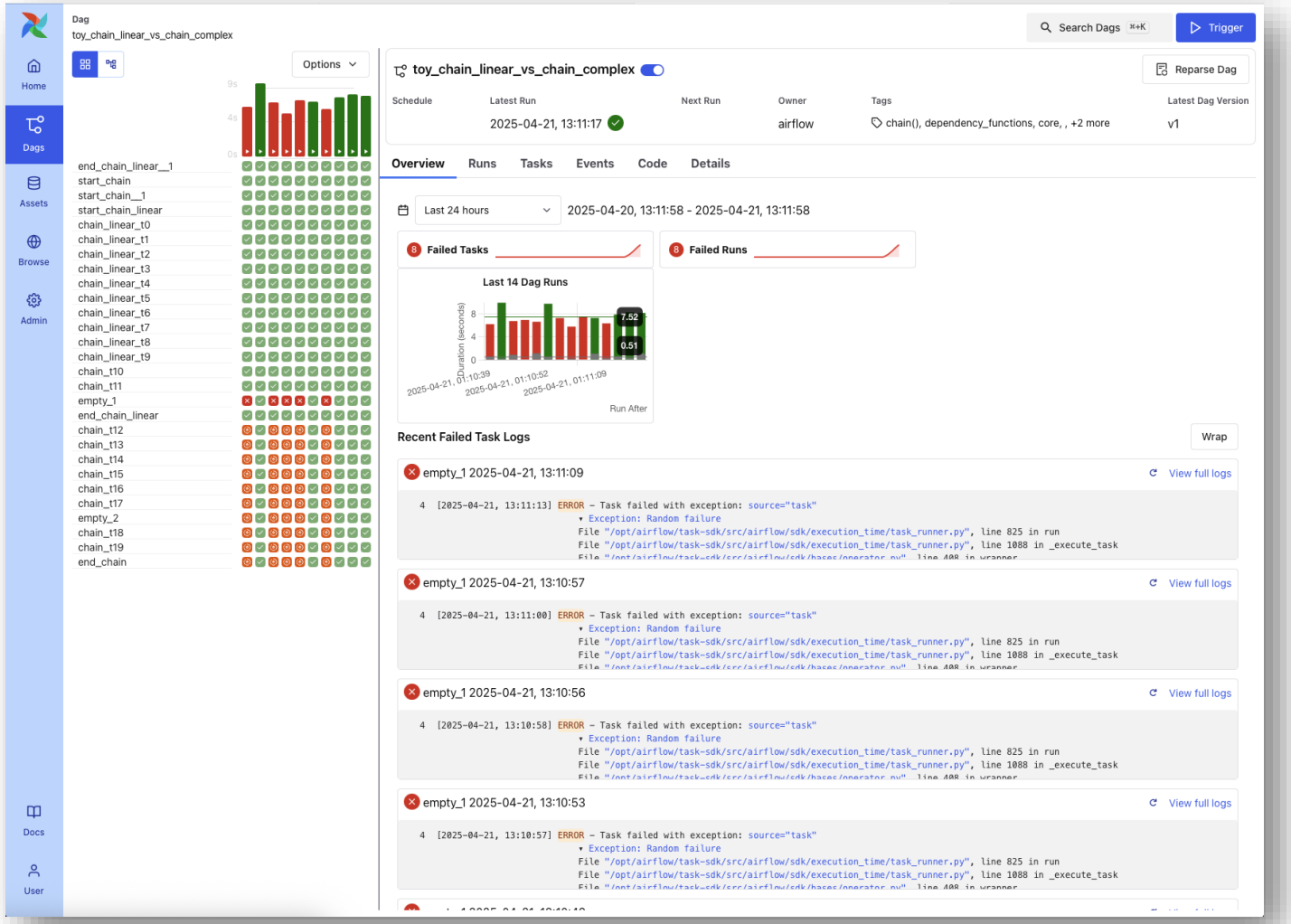
MLflow

- An integrated server for experiment tracking, model registries, versioning, and monitoring.
- Our short tutorial will be built upon MLflow



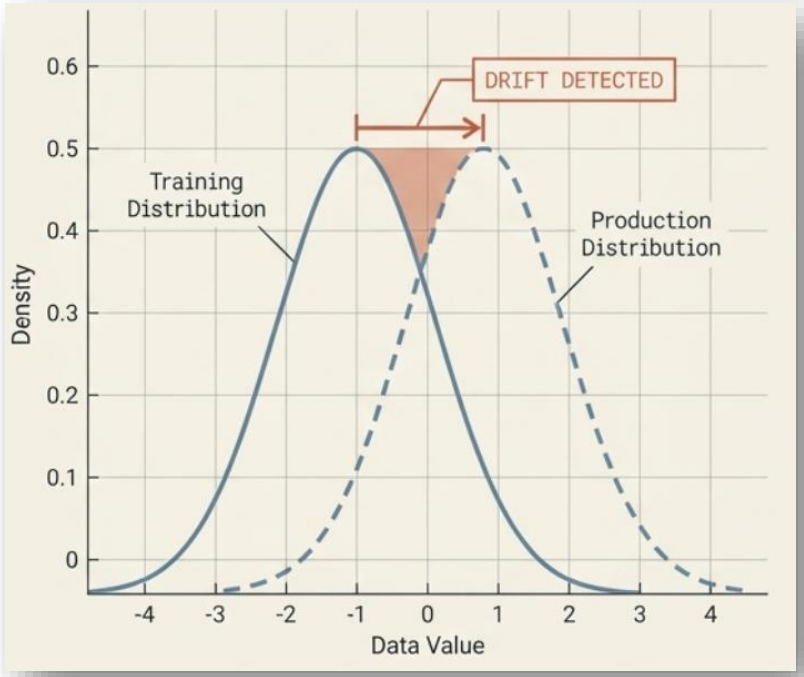
Apache Airflow

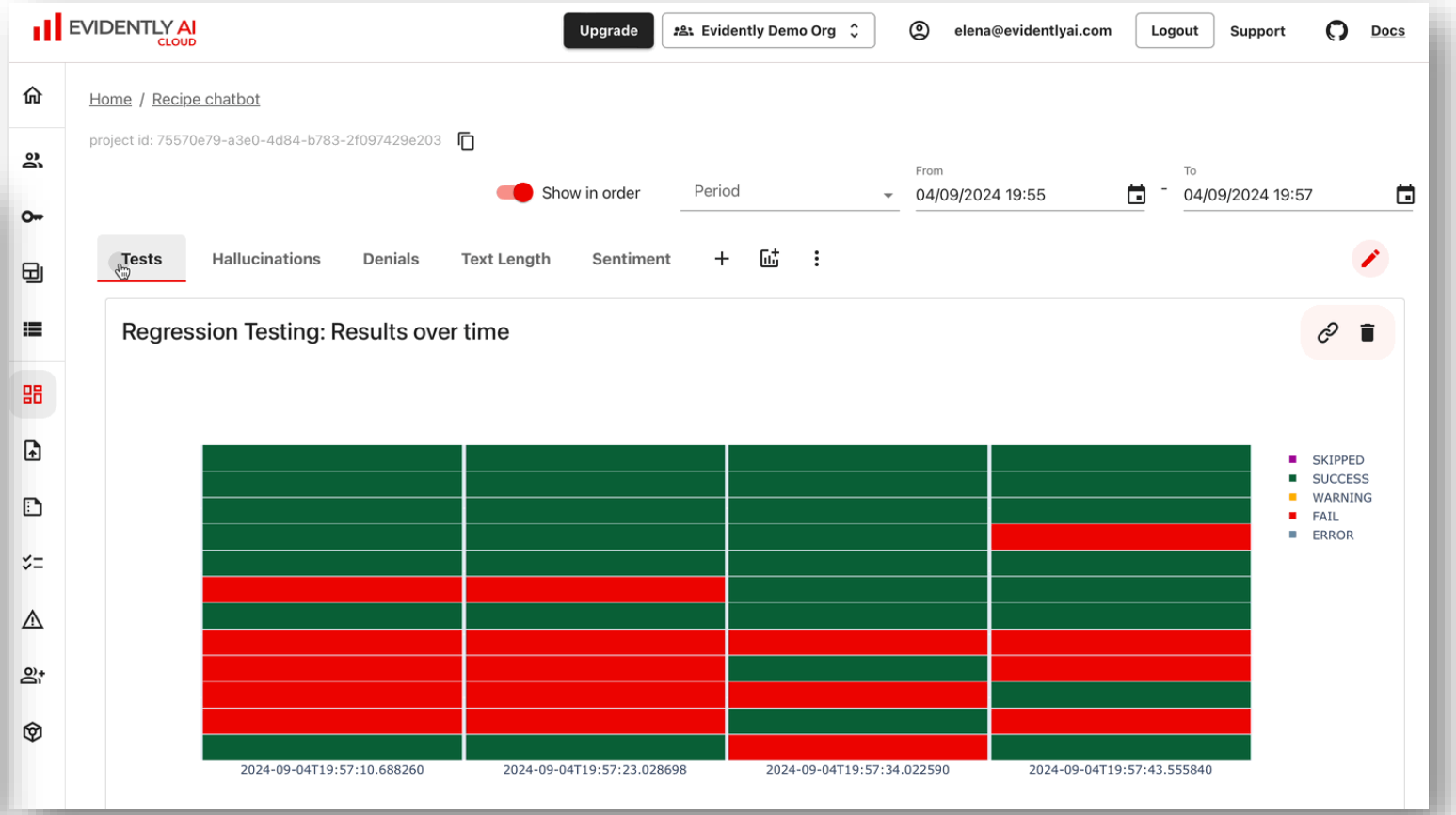
- An open-source platform to programmatically author, schedule and monitor workflows
- Replaces silent pipeline failures and manual cron jobs with reliable, code-based orchestration
- Mange complex data engineering pipelines, feature extraction tasks, and batch inference jobs via DAGs



Evidently

- Evaluate, test & monitor ML models from validation to production, through automated tests



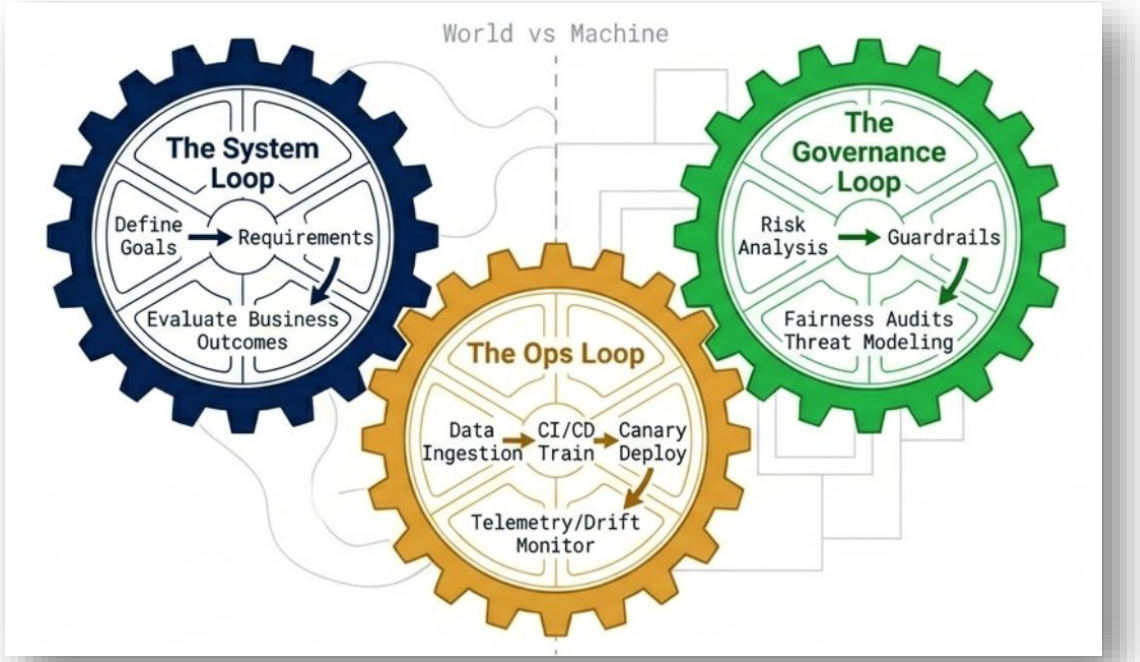


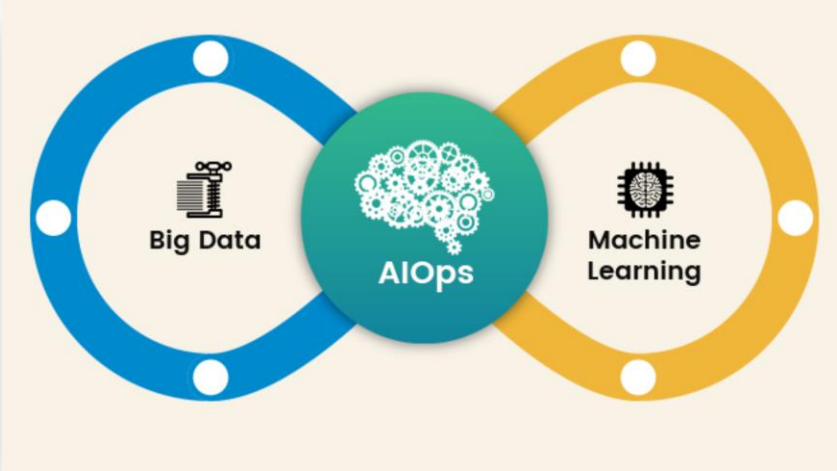
The production readiness checklist

- Goal: metric defined & proxy gap analyzed?
- Data: schema validated & lineage tracked?
- Model: behavioral tests passed?
- Infra: canary deployment ready?
- Ops: drift alerts & fallback mechanisms configured?
- Governance: fairness/bias audit completed?

Take home messages

- ML ops = dev ops + data ops
  - A critical tool in a successful product
  - AI coding's great helper
- ML Ops is a 3-body problem
  - ML in production is a continuous loop of adaptation.
- As you can imagine, AI Ops is taking over
  - Automate everything!!
  - Don't panic, before that happens, you are the king





Questions & comments?

https://PollEv.com/yaolu720