

# CS4262/5462 Machine Learning Systems



## ML Operations

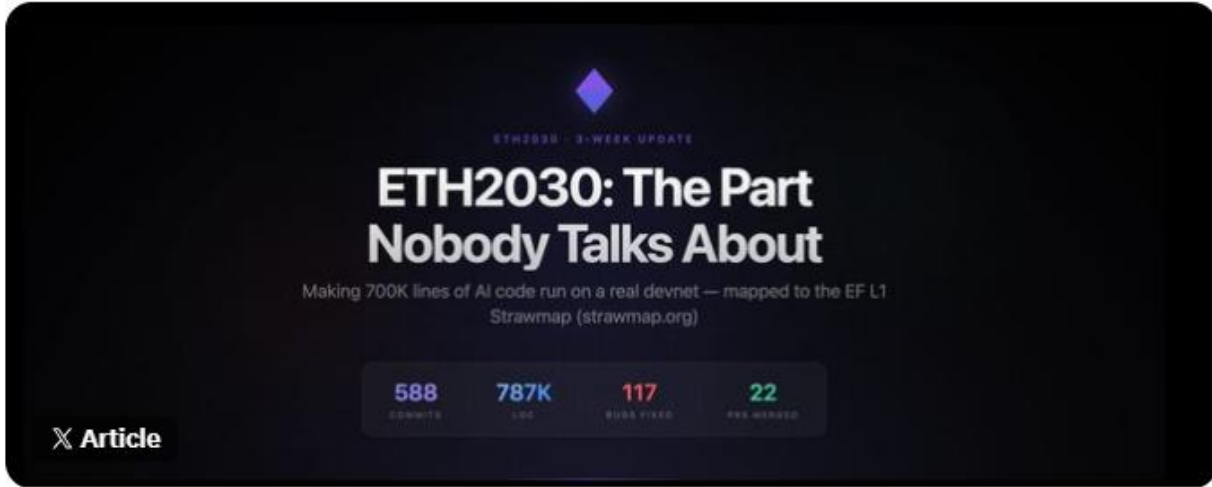
Yao LU

26 Mar 2026

National University of Singapore  
School of Computing


# AI Coding = ?

 YQ  @yq\_acc · Mar 13



**ETH2030: The Part Nobody Talks About**  
Making 700K lines of AI code run on a real devnet — mapped to the EF L1 Strawmap (strawmap.org)

588	787K	117	22
COMMIT	LOC	BUGS FIXED	PRs MERGED

 **Article**

**ETH2030 After 3 Weeks: The Part Nobody Talks About**

We published ETH2030.com on Feb 24 — a reference implementation of the entire Ethereum L1 Strawmap, from Glamsterdam through 2030, in 6 days with Claude. 702K lines of Go. 65/65 roadmap...

[https://x.com/yq\\_acc/status/2032377910733259141?s=20](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)

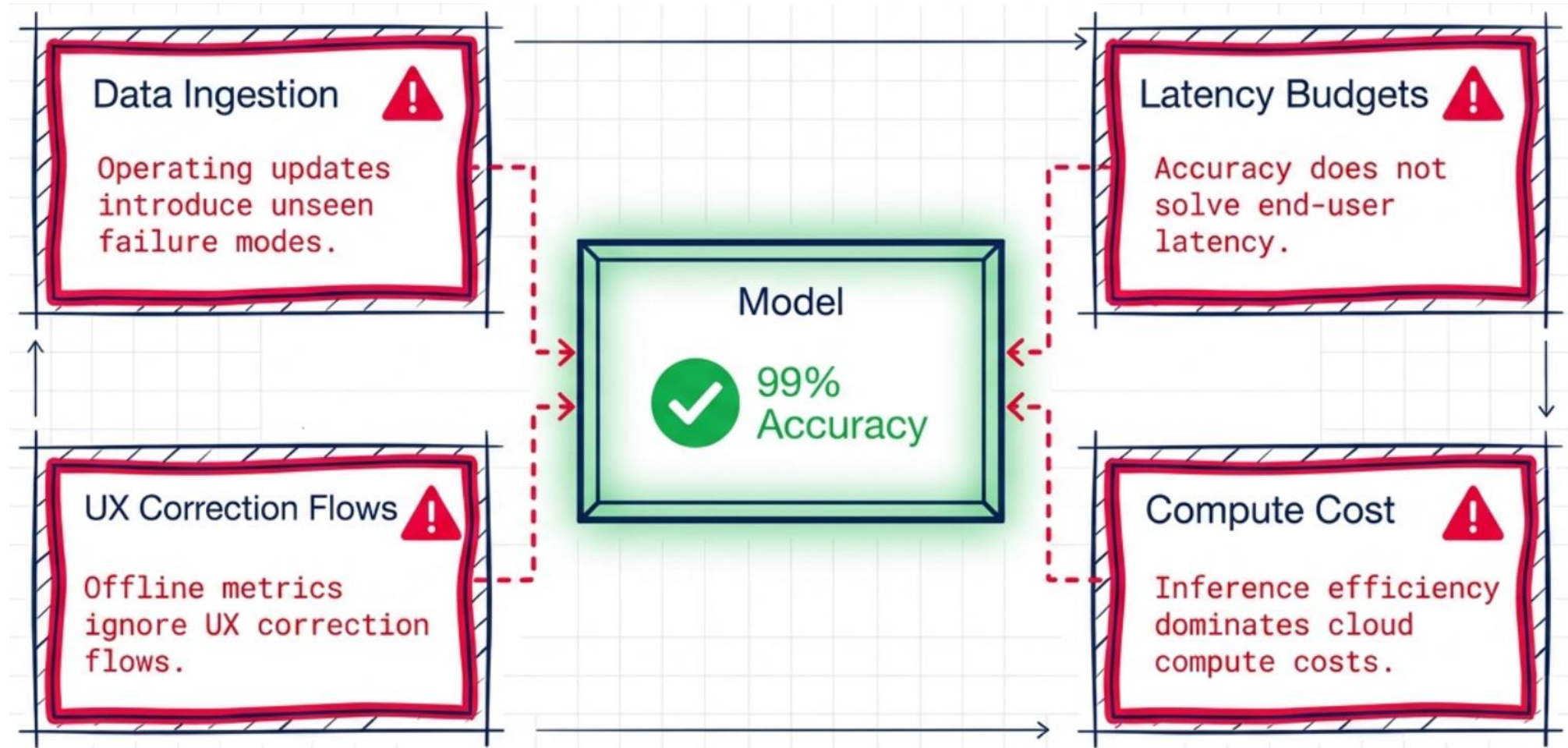
- Define the data in medical scenarios
- How data is generated
- How data is managed
- How models are trained
- How labeling & evaluation is done

# 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

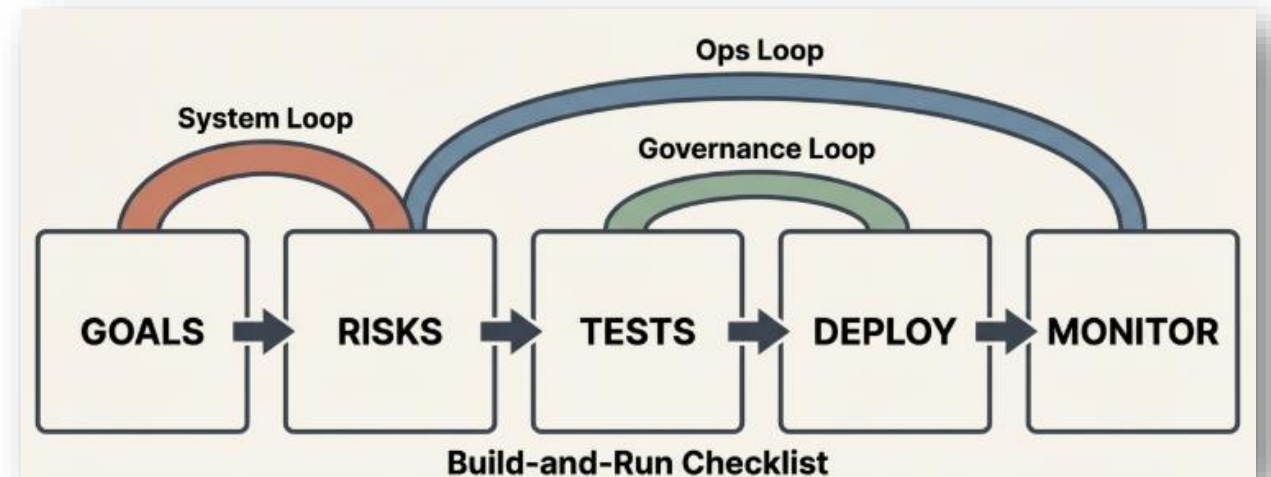
A transformer model might boost accuracy by 1%, but increase cost by 100%. Is it worth it?

# Great models $\neq$ reliable services



# Dev ops lifecycle

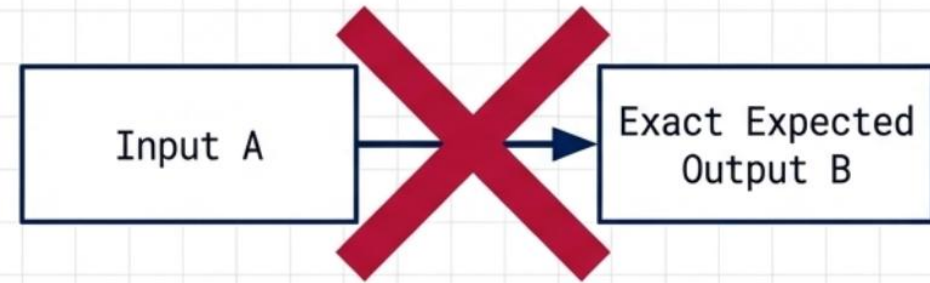
- **System loop:** define goals and outcomes before writing code.
- **Ops loop:** instrument the system to monitor, detect, and respond to drift
- **Governance loop:** enforce responsibility, lineage, and trust through tooling



# Behavioral testing with checklist

- We lack a perfect mechanism to assert exact outputs for unseen inputs.
- Instead, test behavioral invariants, not just aggregate metrics.

## The Oracle Problem

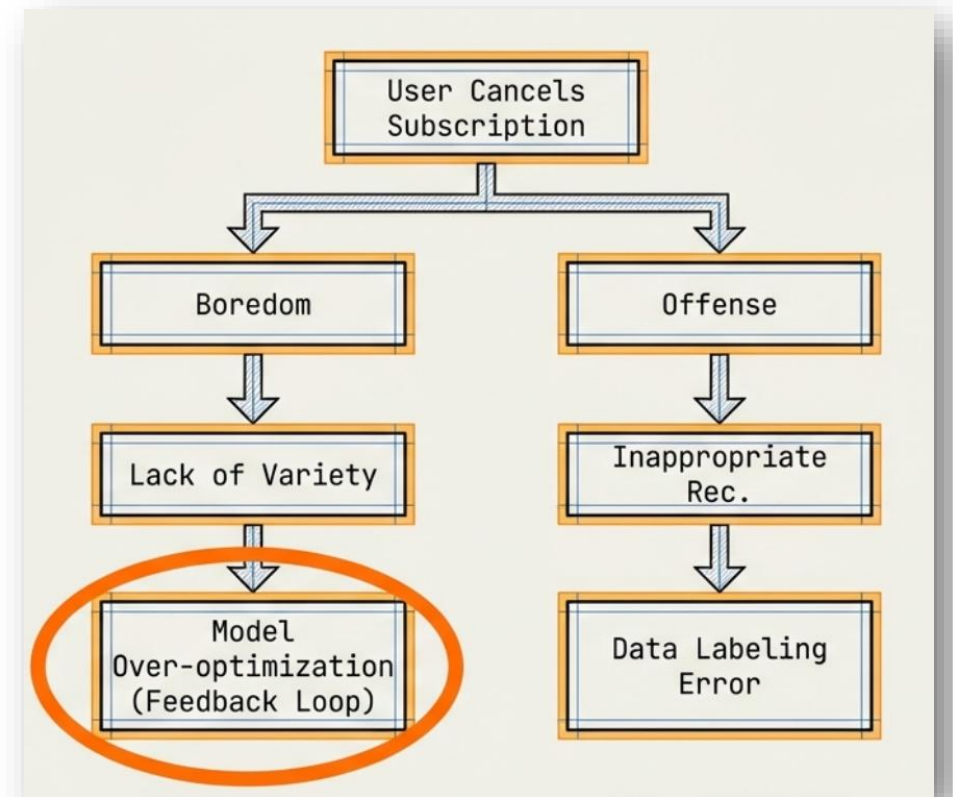


## The Solution: CheckList Capability Matrix

	Vocabulary	Robustness	Fairness
Invariance	✓ Checklist item...	✓ Adding typos to a movie search should not change the top result.	✓ Changing user gender profile should not alter action movie recommendations.
Directional Expectation	✓ Checklist item...	✓ Checklist item...	✓ Checklist item...
Minimum Functionality	✓ Checklist item...	✓ Checklist item...	✓ Checklist item...

## 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

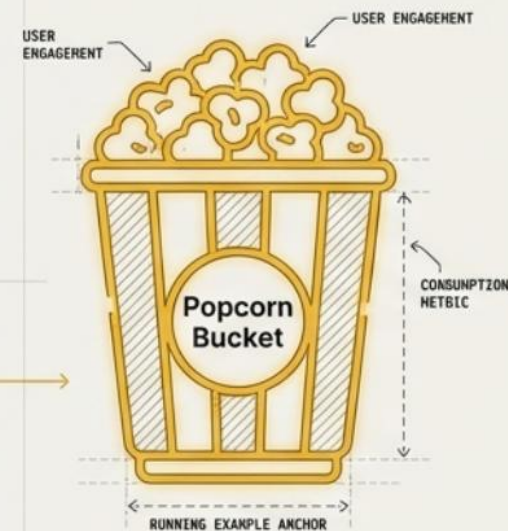
- Data tests
  - Feature monitoring, data quality checks, schema validation
- Model tests
  - Versioning, reproducibility, fairness analysis
- Infra tests
  - Rollback capability, load testing, deployment validation
- Monitoring tests
  - Drifts alerts, anomaly detection, latency tracking

# The movie recommendation engine

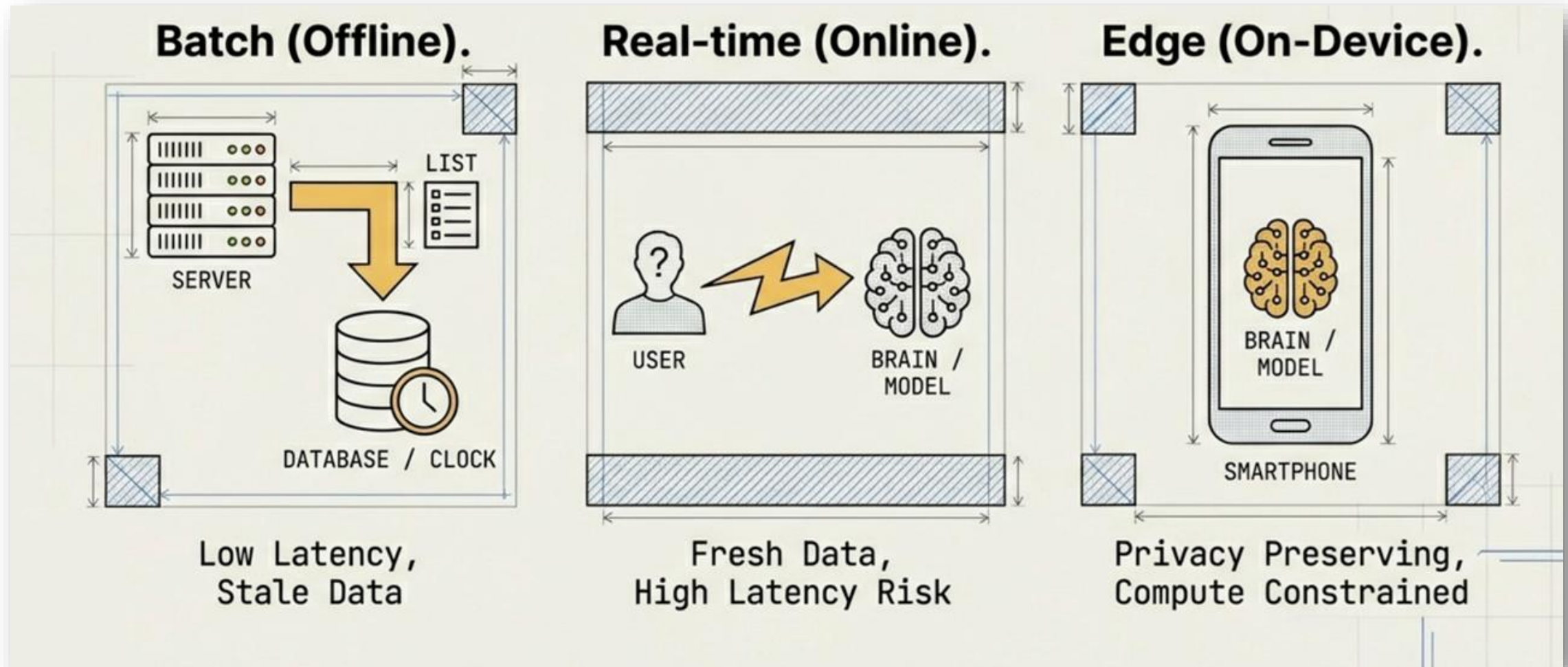
Goal: Predict the next watch.

Constraint: 50ms latency, 10M users.

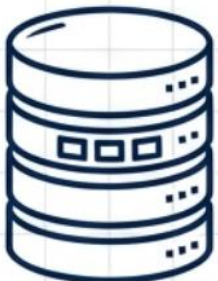
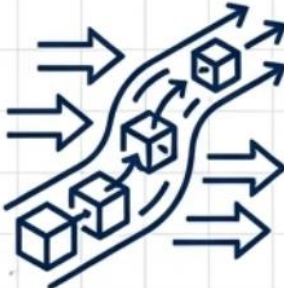
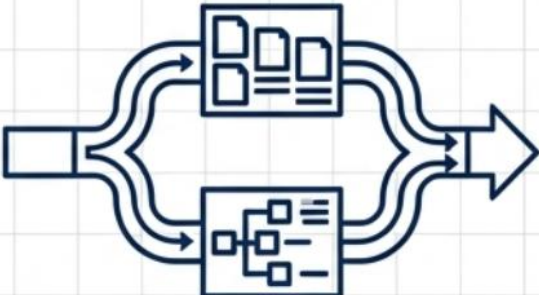
Reality: Tastes drift, data is messy.



# Where does the intelligence live?

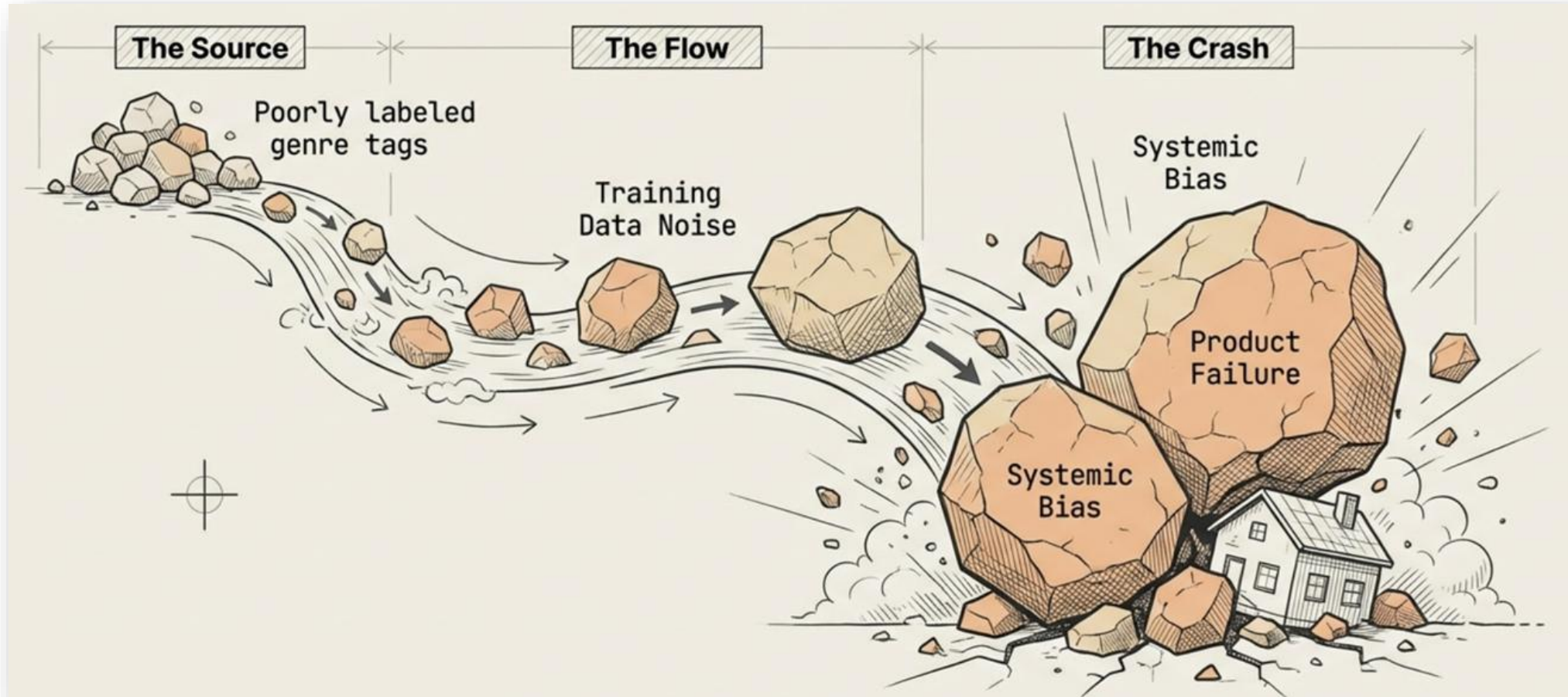


# Data infra trade-offs

Batch Architecture	Stream Processing (Kafka)	Lambda Architecture
		
<ul style="list-style-type: none"><li>● Pre-compute recommendations nightly.</li><li>● Low risk, high latency.</li><li>● Blind to intraday user trends.</li></ul>	<ul style="list-style-type: none"><li>● Update features in near-real-time based on live clicks.</li><li>● High infrastructure cost.</li><li>● Requires rigorous schema event retention.</li></ul>	<ul style="list-style-type: none"><li>● Blends offline batch and online stream.</li><li>● Highest complexity.</li><li>● Requires careful separation to avoid training-serving skew.</li></ul>

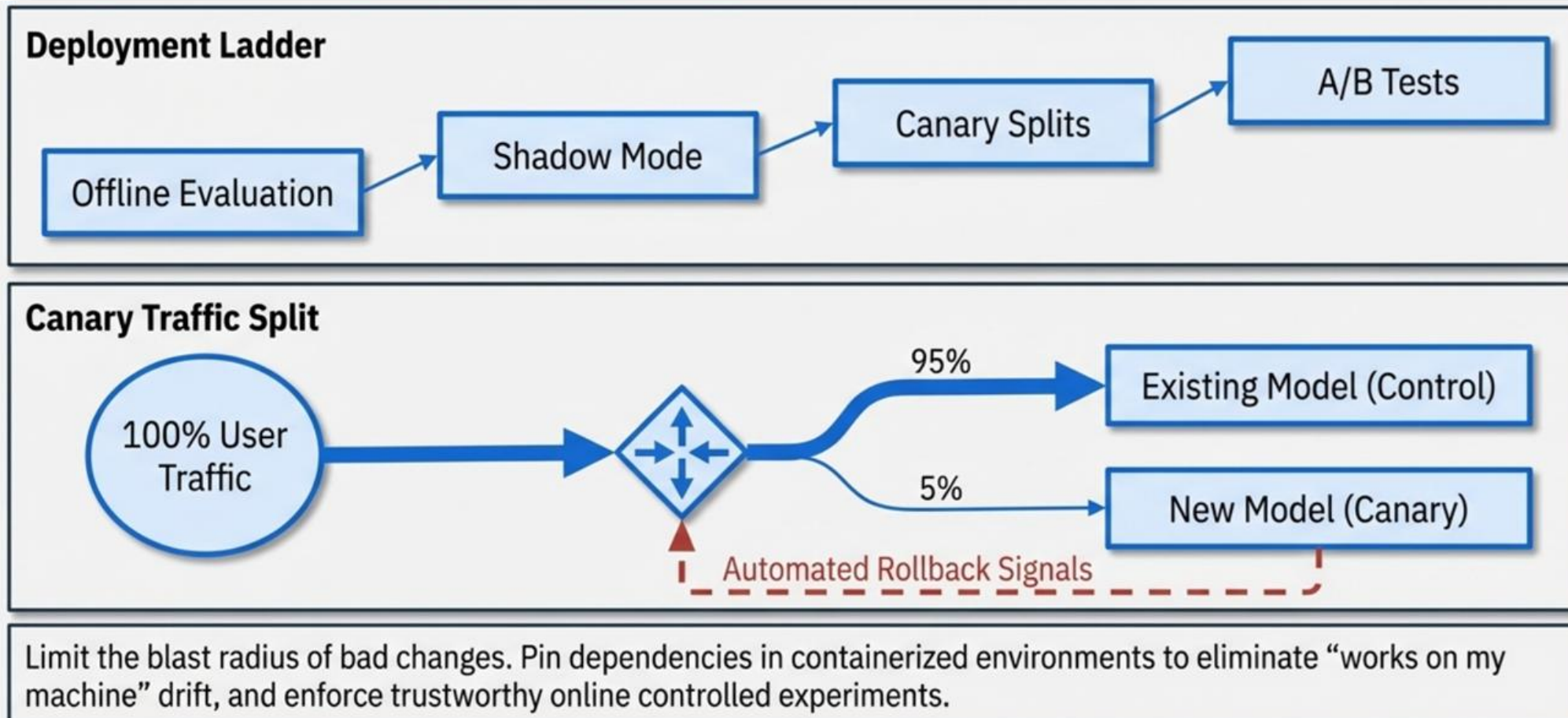
# Data cascades: garbage in, disaster out

- Upstream data issues compound into downstream product failures.



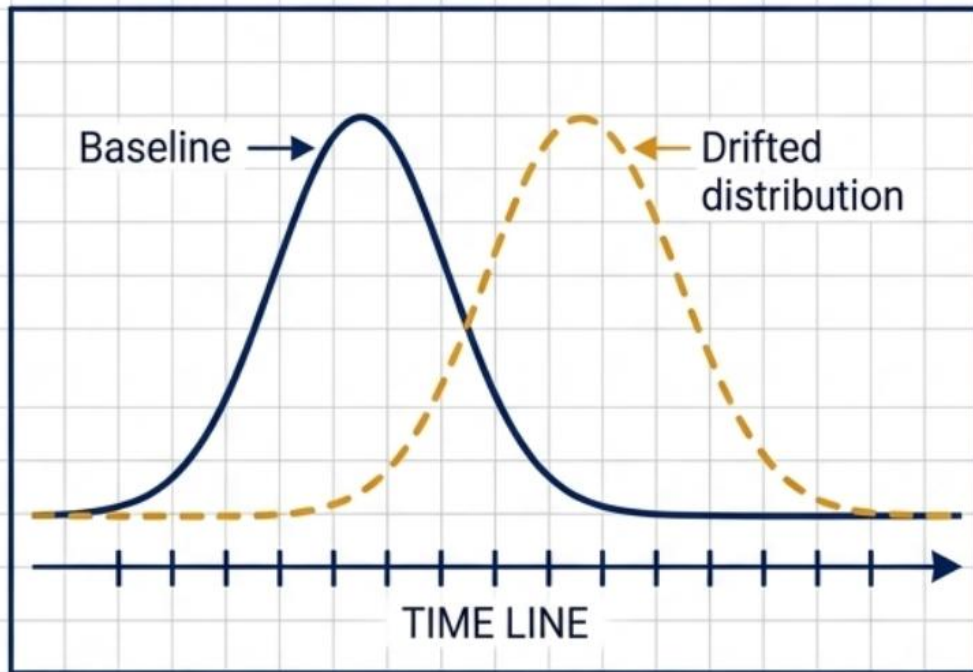
# Canary releases: limiting the blast radius

- Deploy code  $\neq$  release feature. Rollout slowly and rollback instantly.



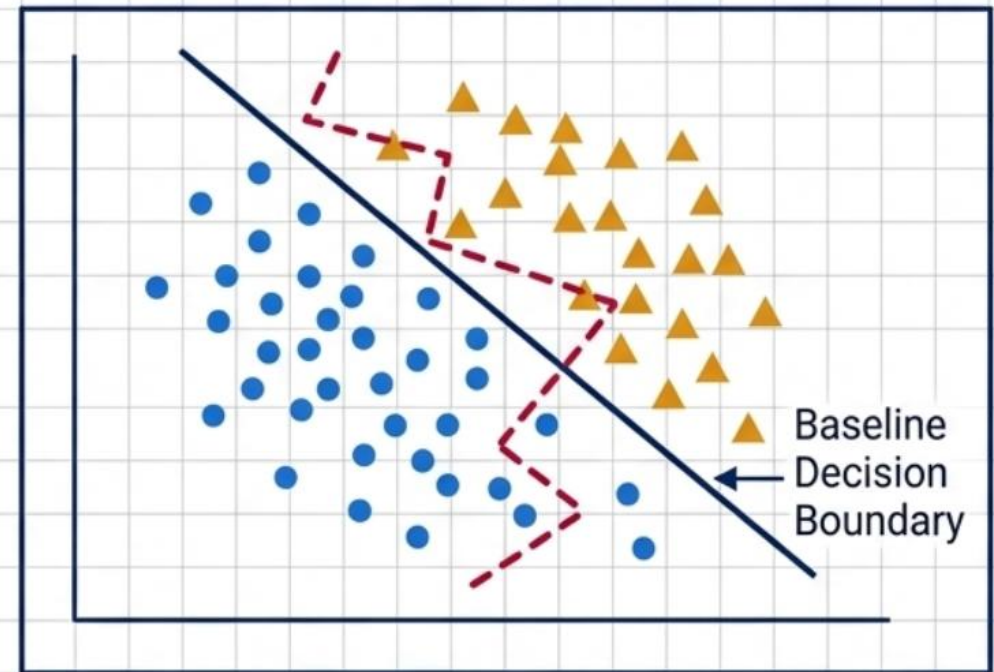
# Drift: the only constant

## Covariate Drift



The distribution of input features changes over time (e.g., users shift from desktop to mobile viewing).

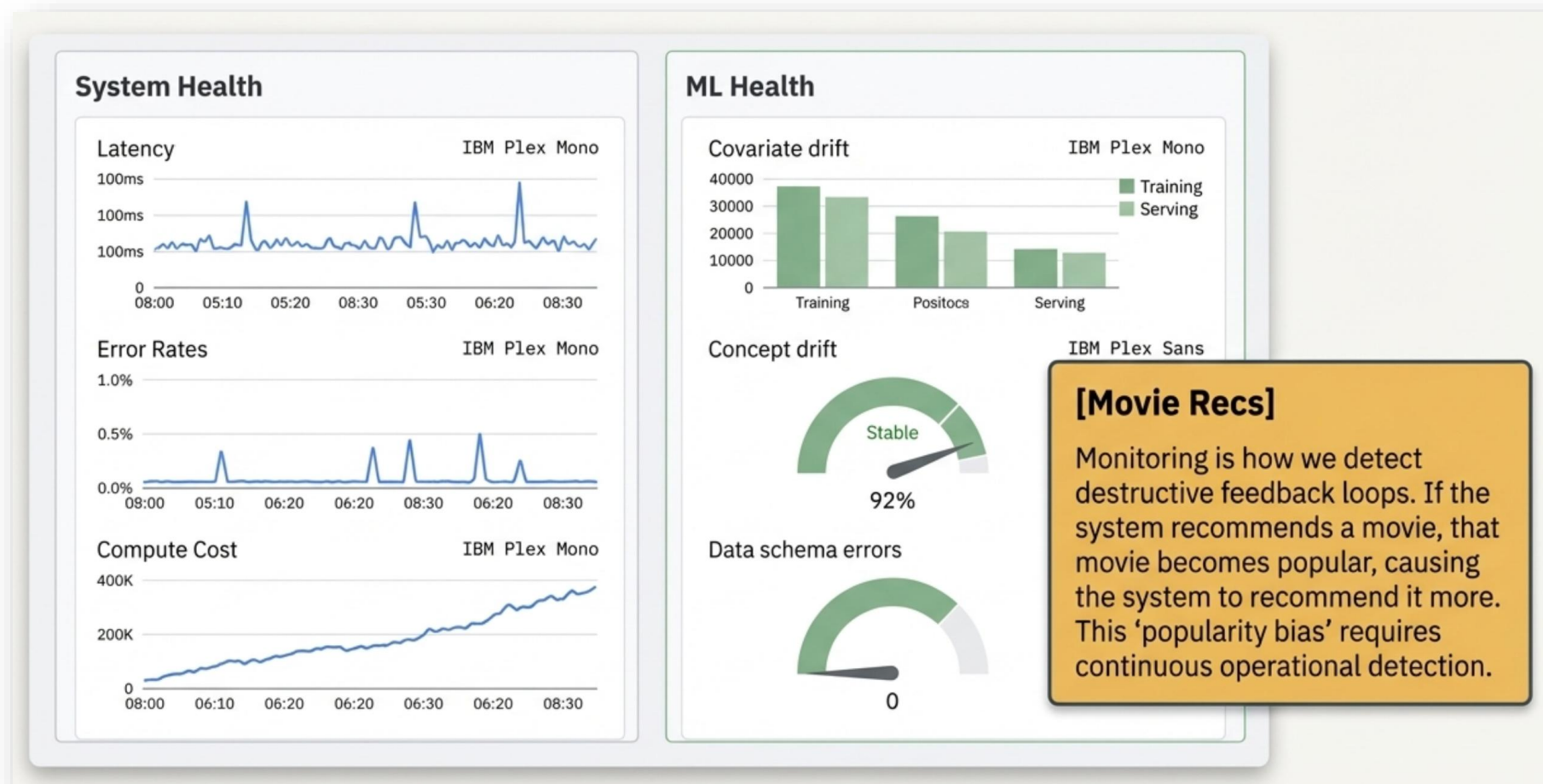
## Concept Drift



The underlying relationship between features and the target changes (e.g., a global pandemic changes entertainment preferences).

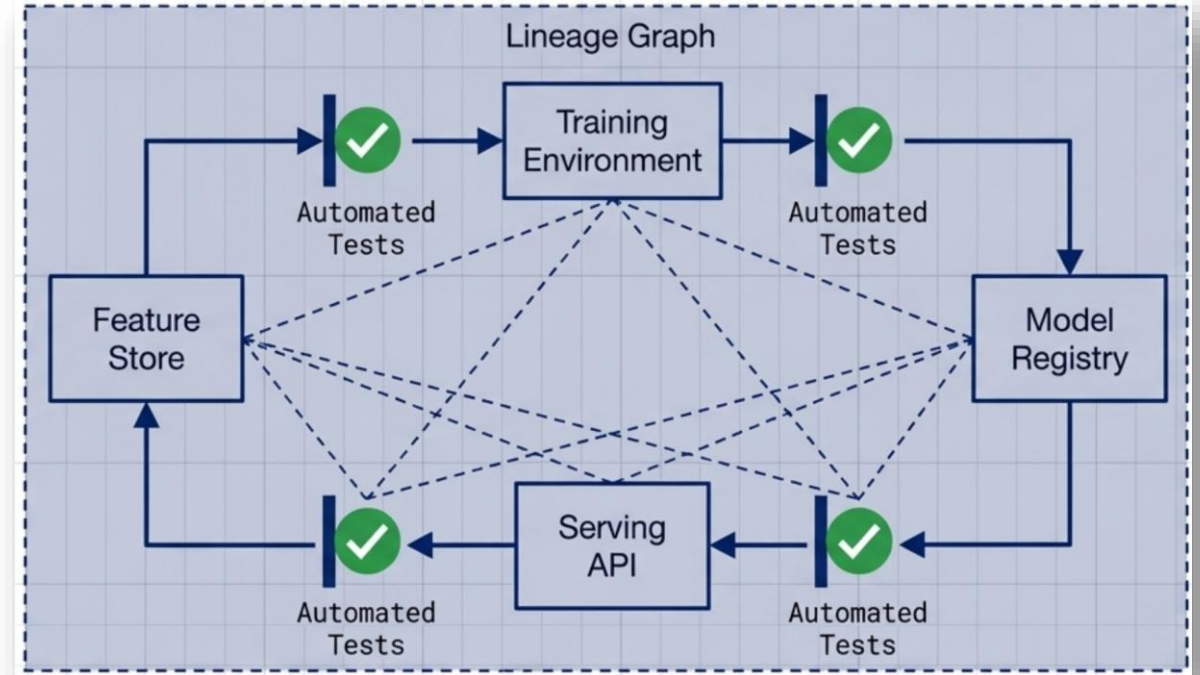
# Observability: telemetry doesn't come free

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



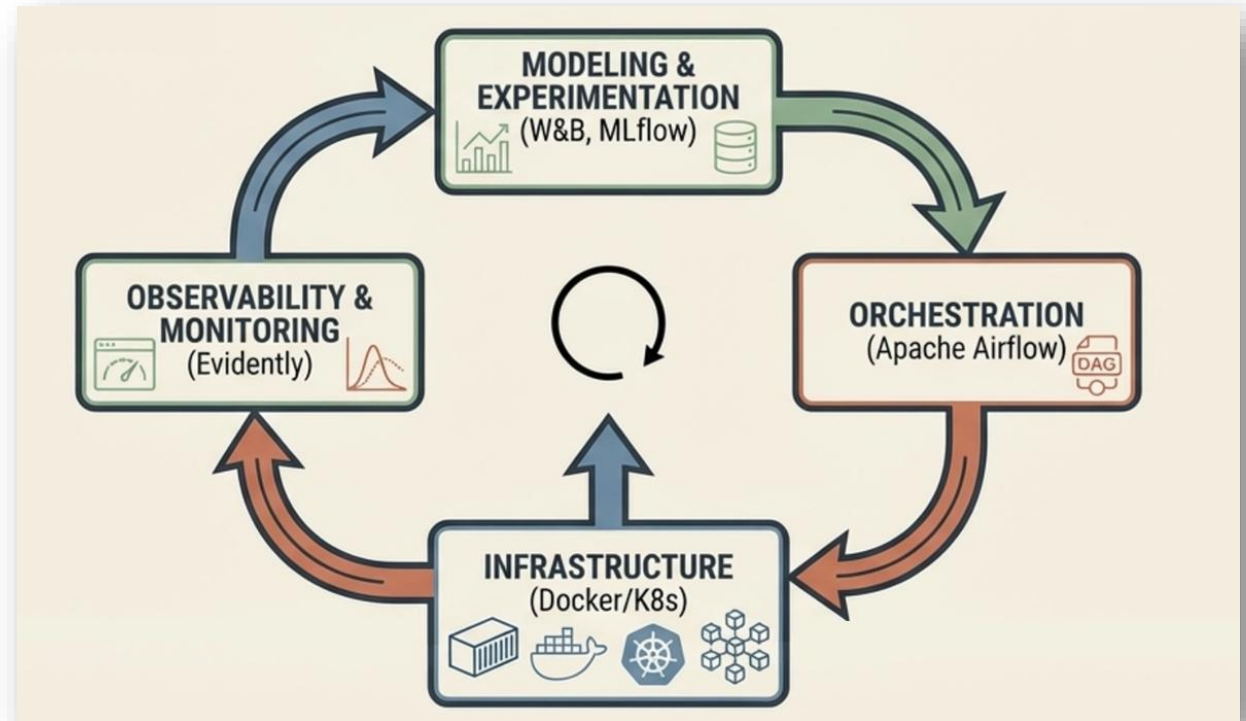
# ML ops lifecycle

- **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



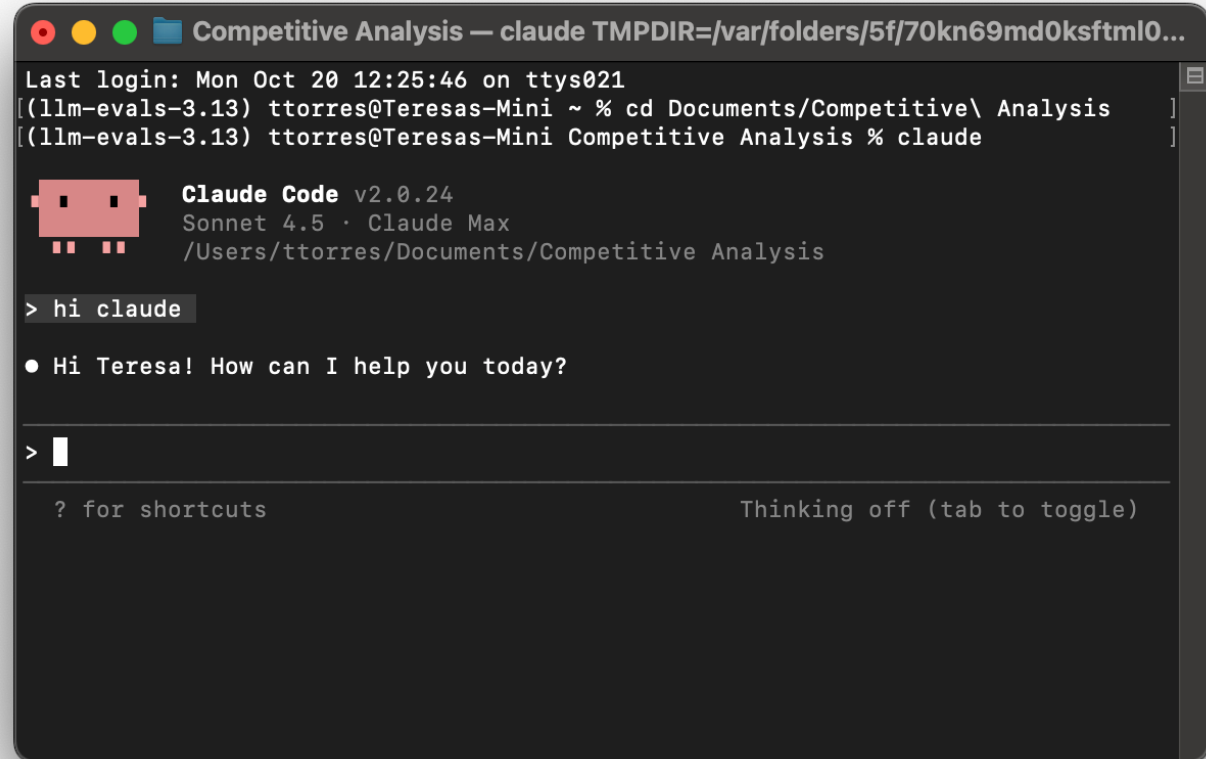
# Some sample tools

- **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)



The screenshot shows a terminal window titled "Competitive Analysis — claude TMPDIR=/var/folders/5f/70kn69md0ksftml0...". The terminal output includes the following text:

```
Last login: Mon Oct 20 12:25:46 on ttys021
[(llm-evals-3.13) ttorres@Teresas-Mini ~ % cd Documents/Competitive\ Analysis ]
[(llm-evals-3.13) ttorres@Teresas-Mini Competitive Analysis % claude ]
```

The interface displays the Claude Code logo (a red robot head) and the text: "Claude Code v2.0.24", "Sonnet 4.5 · Claude Max", and the current directory path: "/Users/ttorres/Documents/Competitive Analysis".

The user has entered the command: `> hi claude`

The response is: `• Hi Teresa! How can I help you today?`

The user has entered another command: `> |`

At the bottom of the terminal, there are two status indicators: "? for shortcuts" on the left and "Thinking off (tab to toggle)" on the right.

# Github (you often forget)

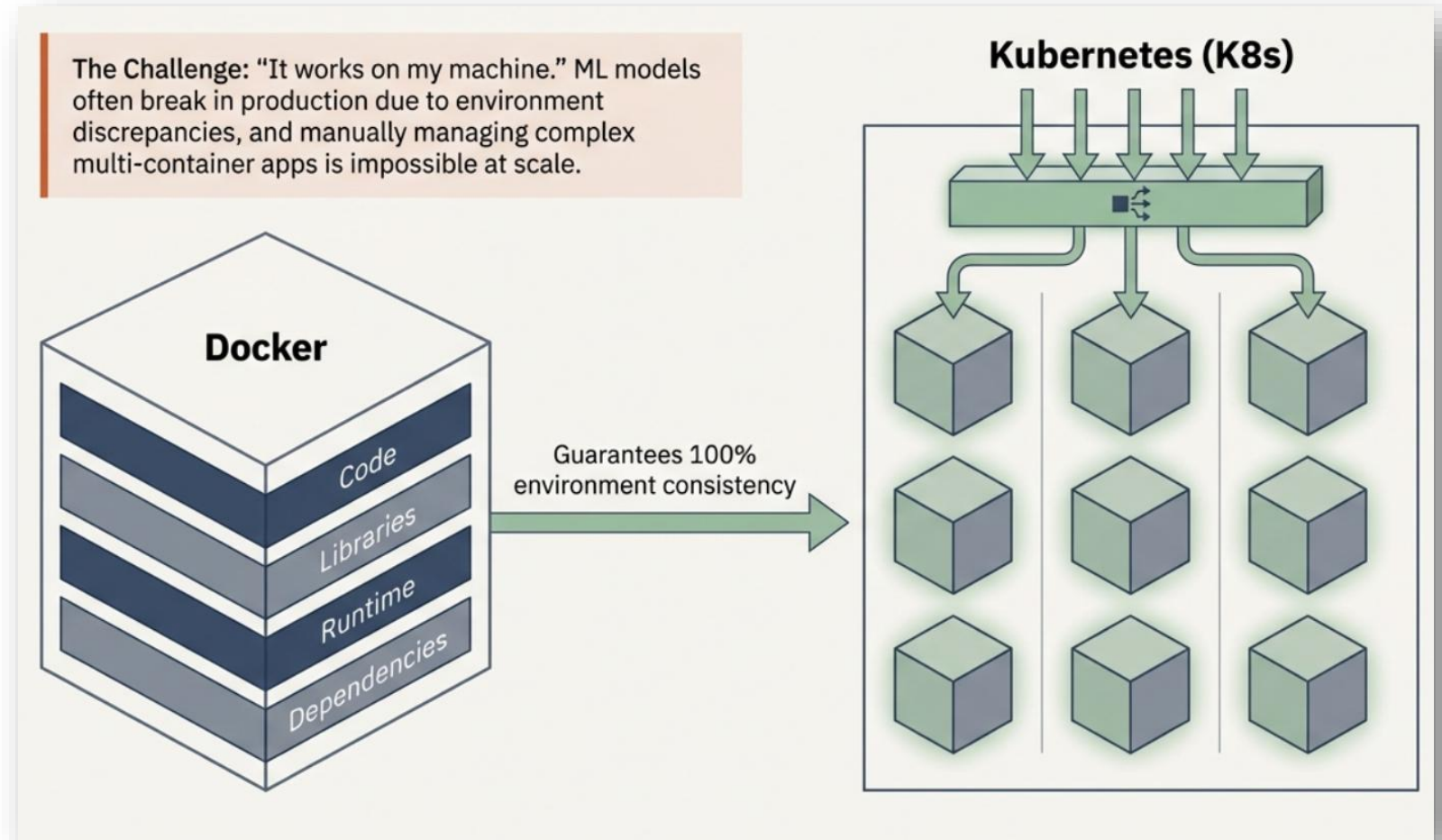
- Critical tool for AI coding to help you reach 100x+
  - **Versioning & teaming:** you should be familiar with this
  - **CI/CD:** build, test, deploy, release
  - **CT:** testing metrics beyond “just compiles”
  - **Reporting:** performance & feedbacks

The screenshot shows the GitHub Actions interface for the repository 'mlysys-io / PortfolioBench'. The 'Actions' tab is active, displaying a list of workflow runs. The left sidebar shows the 'All workflows' section with a 'New workflow' button and a list of workflows: 'Auto Test', 'Benchmark Report', and 'Deploy static content t...' (disabled). Below this, the 'Management' section includes 'Caches', 'Deployments', 'Attestations', 'Runners', 'Usage metrics', and 'Performance metrics'. The main area shows 'All workflows' with a search bar and a list of 199 workflow runs. The runs are filtered by 'Event', 'Status', 'Branch', and 'Actor'. The visible runs are:

Event	Status	Branch	Actor	Time
Benchmark Report	Success	main		Mar 16, 3:12 PM GMT+8 12m 1s
Merge pull request #62 from ...	Failure	main	tjluyao	Mar 14, 3:54 PM GMT+8 3m 42s
Merge pull request #62 from ...	Success	main	tjluyao	Mar 14, 3:54 PM GMT+8 2m 57s
Add comprehensive ...	Failure	claude/backtesting-test-cas...	tjluyao	Mar 14, 3:54 PM GMT+8 3m 37s

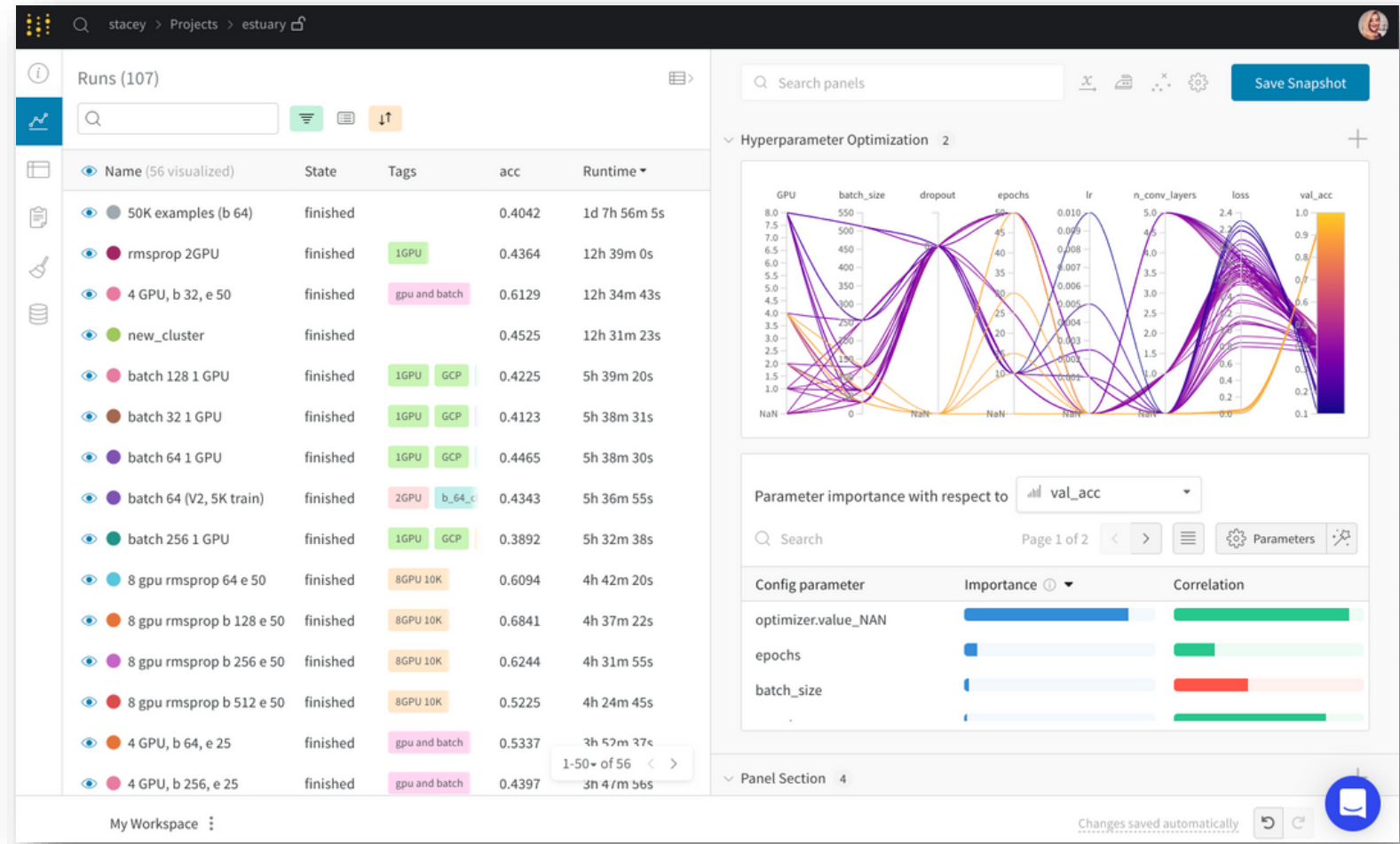
# Docker & Kubernetes

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



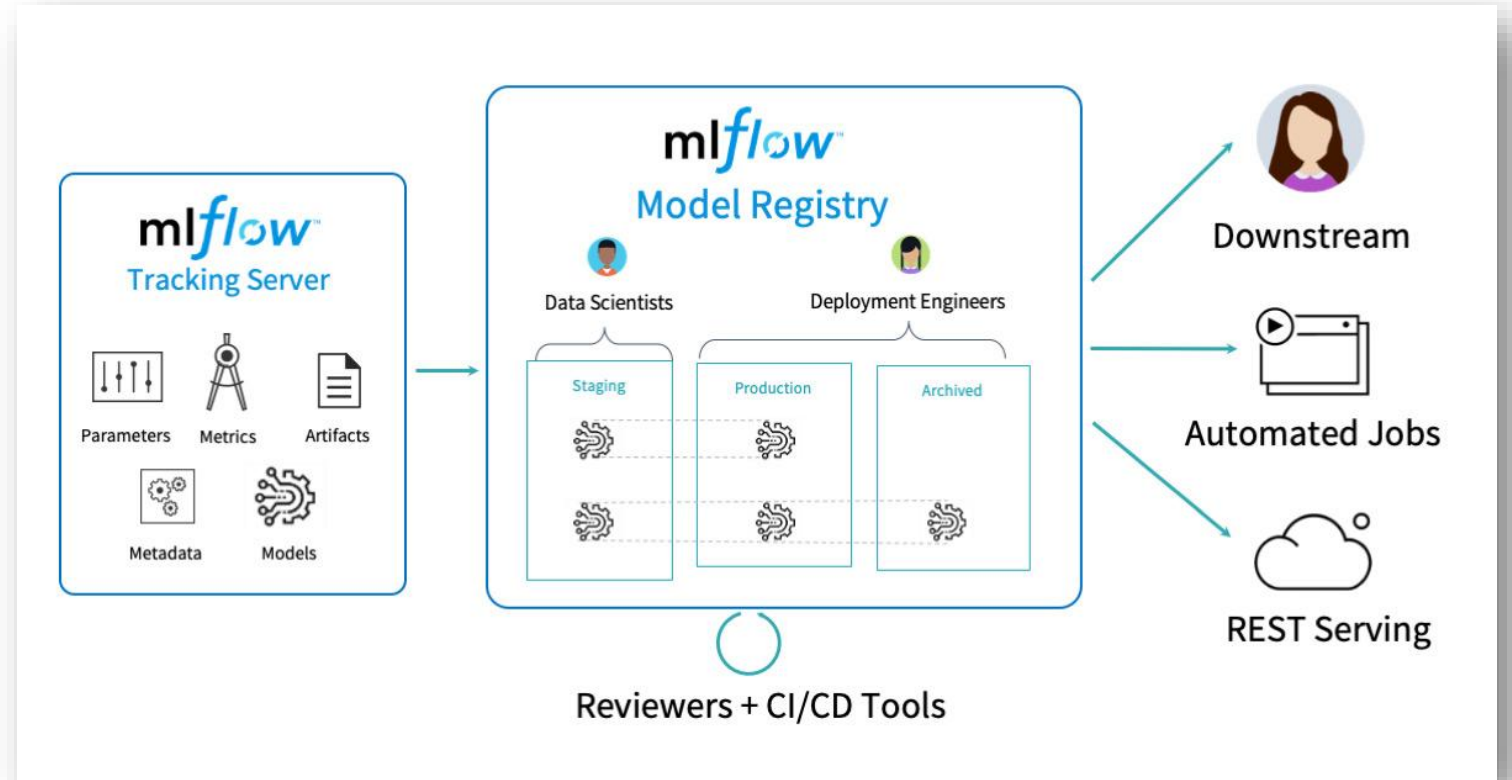
# Weights and bias

- 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

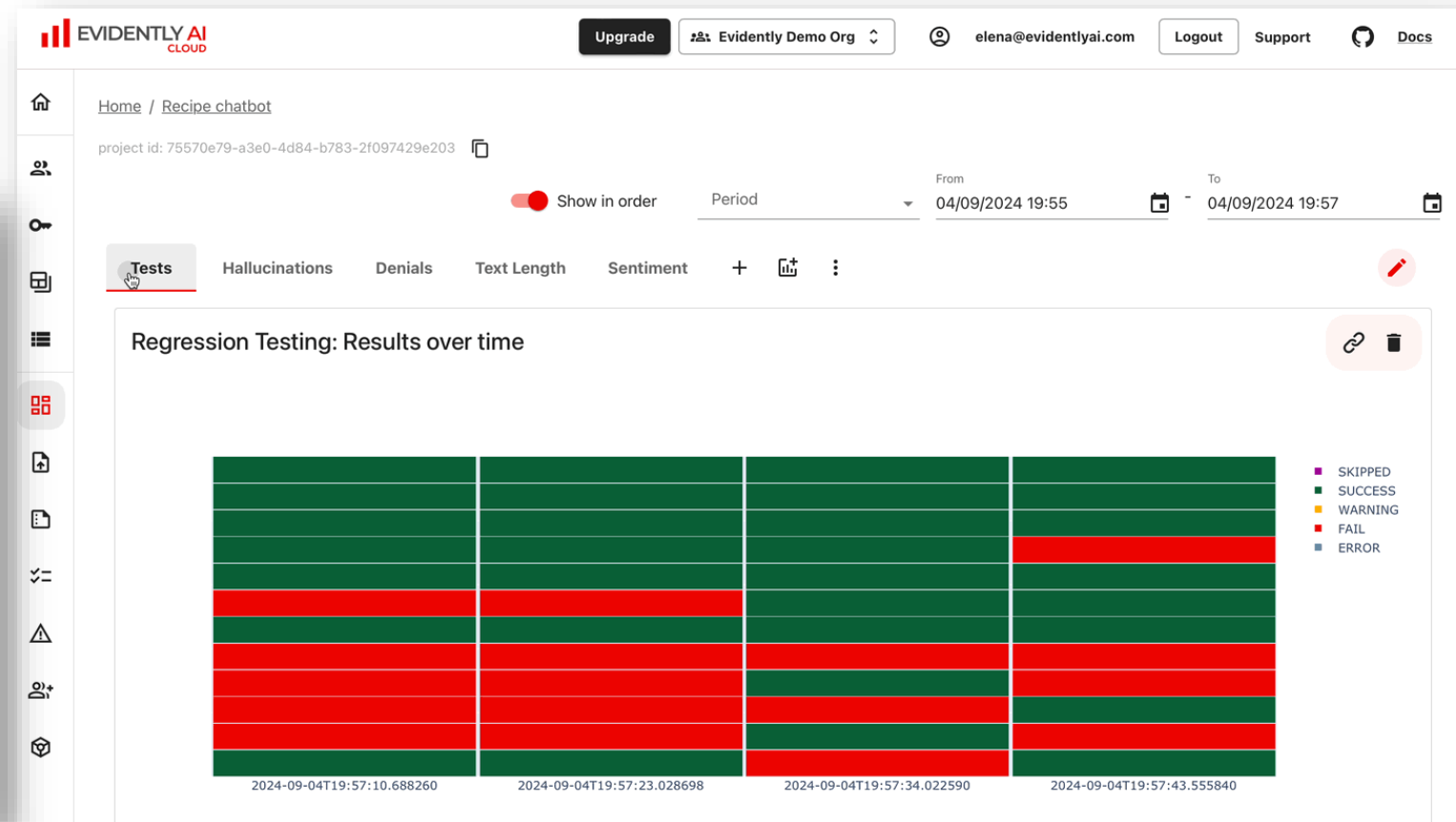
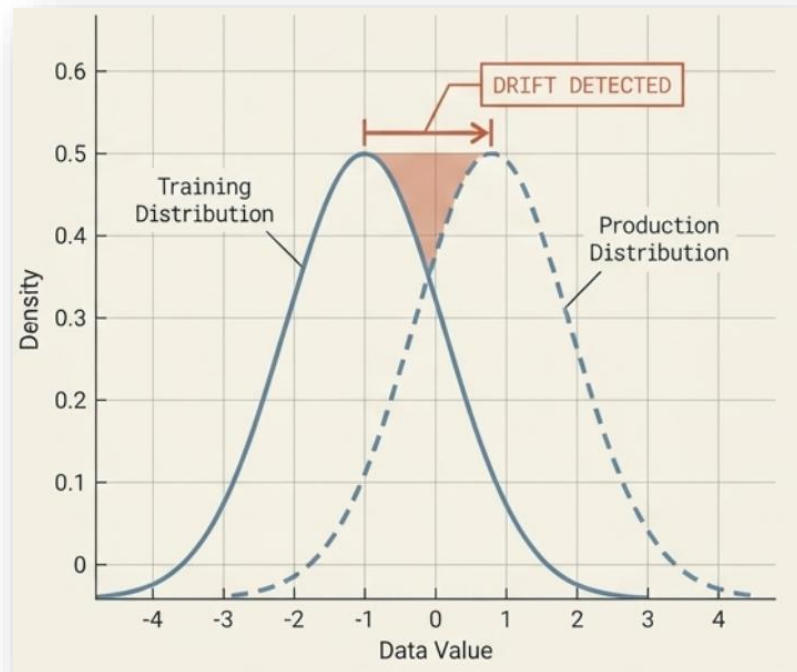
- An open-source platform to programmatically author, schedule and monitor workflows
- Replaces silent pipeline failures and manual cron jobs with reliable, code-based orchestration
- Manage complex data engineering pipelines, feature extraction tasks, and batch inference jobs via DAGs

The screenshot displays the Apache Airflow web interface for a DAG named 'toy\_chain\_linear\_vs\_chain\_complex'. The interface is divided into several sections:

- Left Sidebar:** Contains navigation links for Home, Dags, Assets, Browse, Admin, Docs, and User.
- DAG Overview:** Shows a bar chart of 'Last 14 Dag Runs' with a duration of 7.52 seconds. Below the chart is a grid of task status indicators (green for success, red for failure) for tasks like 'end\_chain\_linear\_1', 'start\_chain', 'chain\_linear\_10', etc.
- Task Details:** A table showing the latest run status (2025-04-21, 13:11:17), owner (airflow), and tags (chain(), dependency\_functions, core, etc.).
- Failed Tasks:** A section titled 'Recent Failed Task Logs' showing four failed runs of the 'empty\_1' task. Each failure is an 'ERROR' with the message 'Task failed with exception: source="task"' and 'Exception: Random failure'. The logs include file paths and line numbers.

# Evidently

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

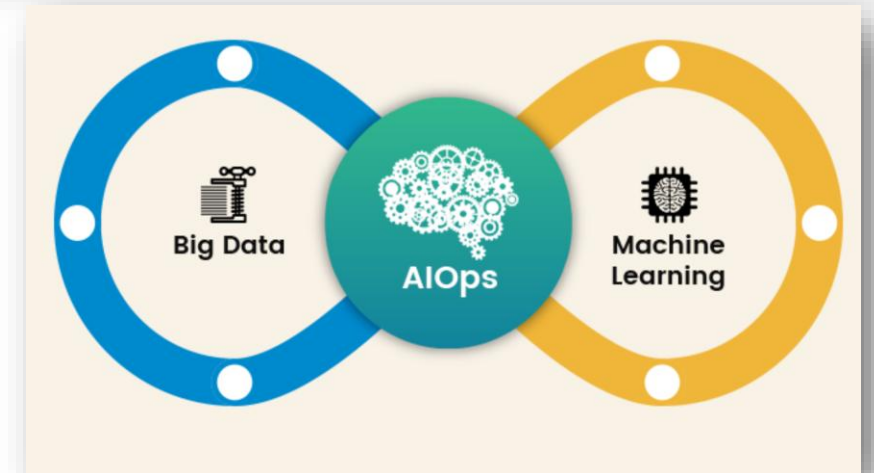
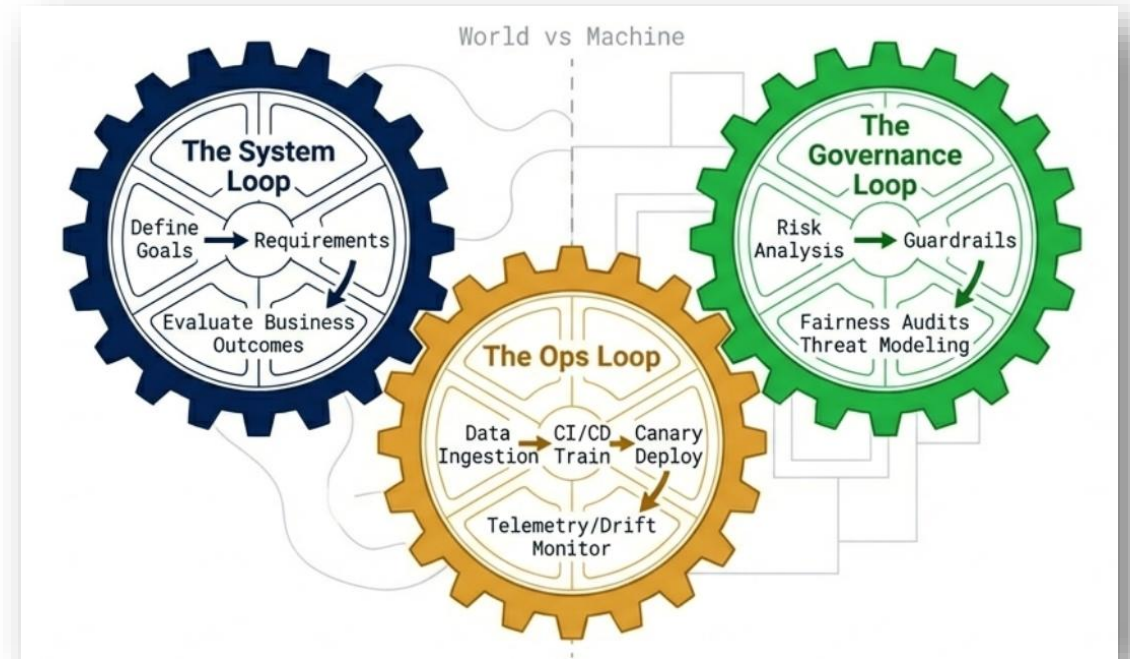


# The production readiness checklist

- 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>