# A Blueprint for Agentic Data Workflows

## DataOps + FITT + Data Testing = 10x

Why AI alone is not enough—and how to build the scaffolding for autonomous data engineering.

DataKitchen

# Before We Start

Slides, recording, and transcription will be shared this week.

Put your questions in the chat window; we will answer at the end.

50-minute presentation followed by a Q&A session.

**Christopher Bergh**
CEO, Head Chef
*DataKitchen*

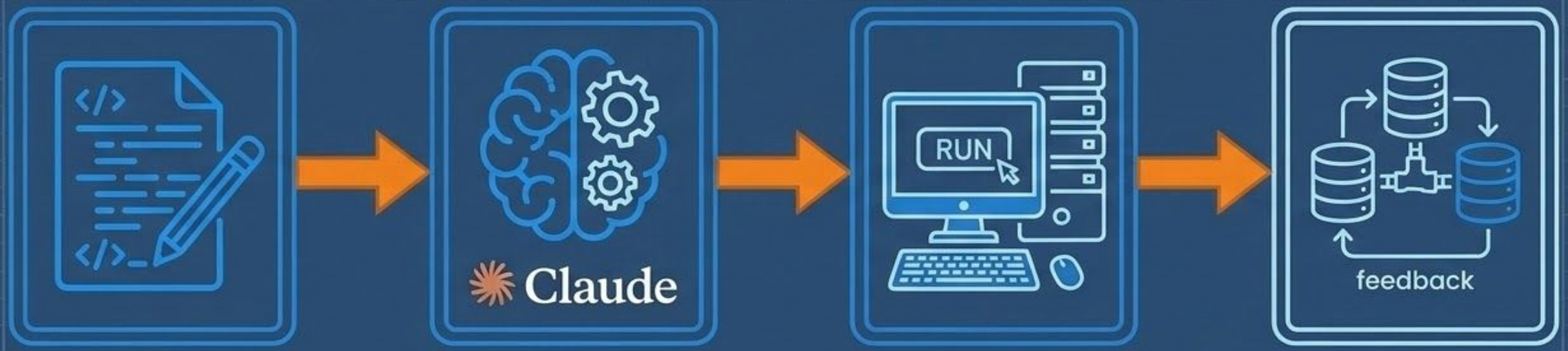
Copyright © 2026 DataKitchen, Inc. All Rights Reserved.

CLAUDE CODE · DATAOPS · FITT · DATA TESTING

# 10x Data Engineering Productivity With Claude

Practitioner patterns that actually work.

# How Claude Code Works with Data Engineering




### 1. Text-Based Instructions

User provides natural language prompts or existing code.

### 2. Claude Code Engine (AI)

AI interprets request & generates data engineering code.

### 3. Separate Local Engine

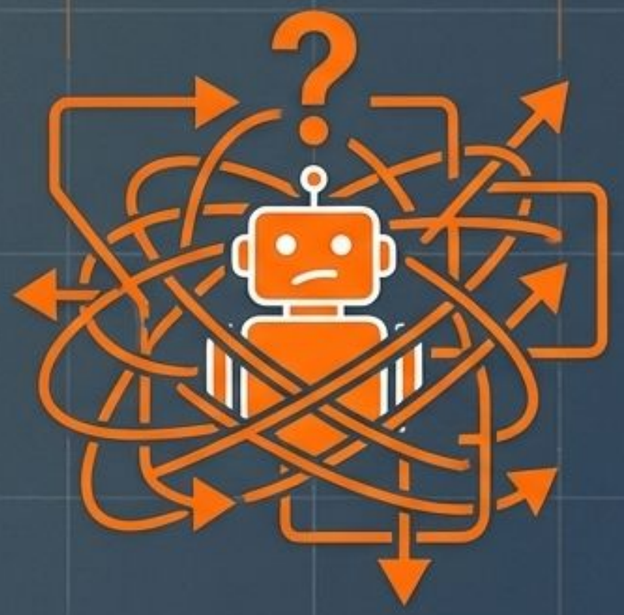
Code executes on a separate, controlled desktop/environment.

### 4. Data Pipeline & Iteration

Builds pipelines. Results fed back for refinement.

# AI is not a magic data tool. It is a coding and process engine.

## The Myth



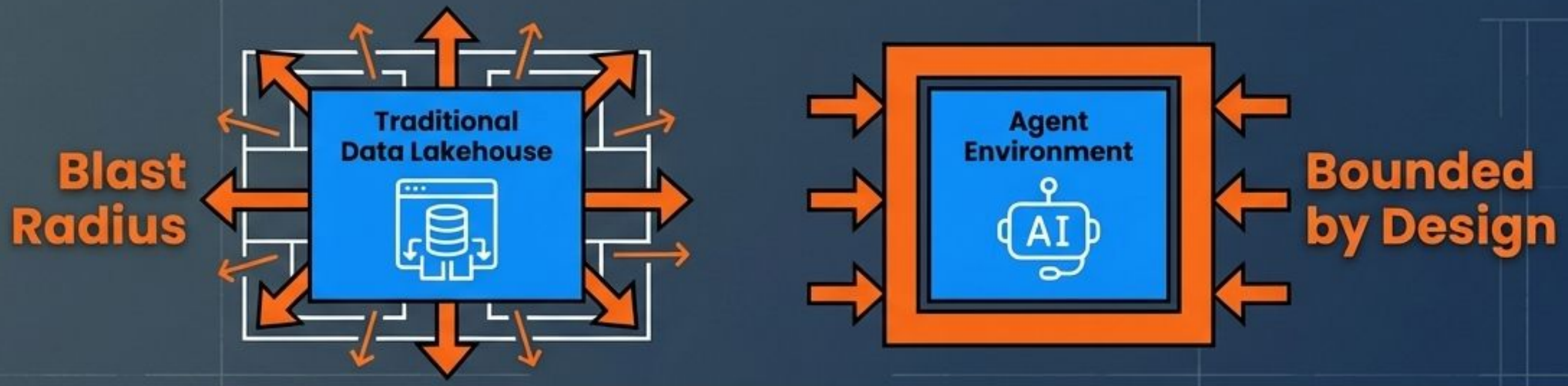
Dropping an AI agent into an unstructured codebase without isolated testing means flying blind.

## The Reality

The 10x productivity leap requires treating data engineering as a process problem, building a foundational architecture where agents can safely iterate

DataKitchen

# **The Trust Problem:** Why lakehouses break down under agent workloads




| | | Traditional Lakehouse | Agent-Ready Infrastructure |
|---|---|---|---|
| 1 | **Design Paradigm** | Human-operated, shared environments | API-based access, concurrent branching |
| 2 | **Safety Mechanism** | Human-readable audit trails, manual rollbacks | Ephemeral isolated environments, bounded by design |
| 3 | **Failure Mode** | Agents drop production tables or hallucinate data | Blast radius contained to isolated sandbox |

The path to trustworthy agentic workflows starts with solving the **infrastructure problem** first. You cannot bolt governance onto an environment not designed for agents.

DataKitchen

# Unlocking Claude Code requires treating data engineering as a process problem.

**DataOps + FITT + Data Testing = 10x**

**Layer 1: DataOps**
Provides Isolation.

**Layer 2: FITT**
Provides Structure.

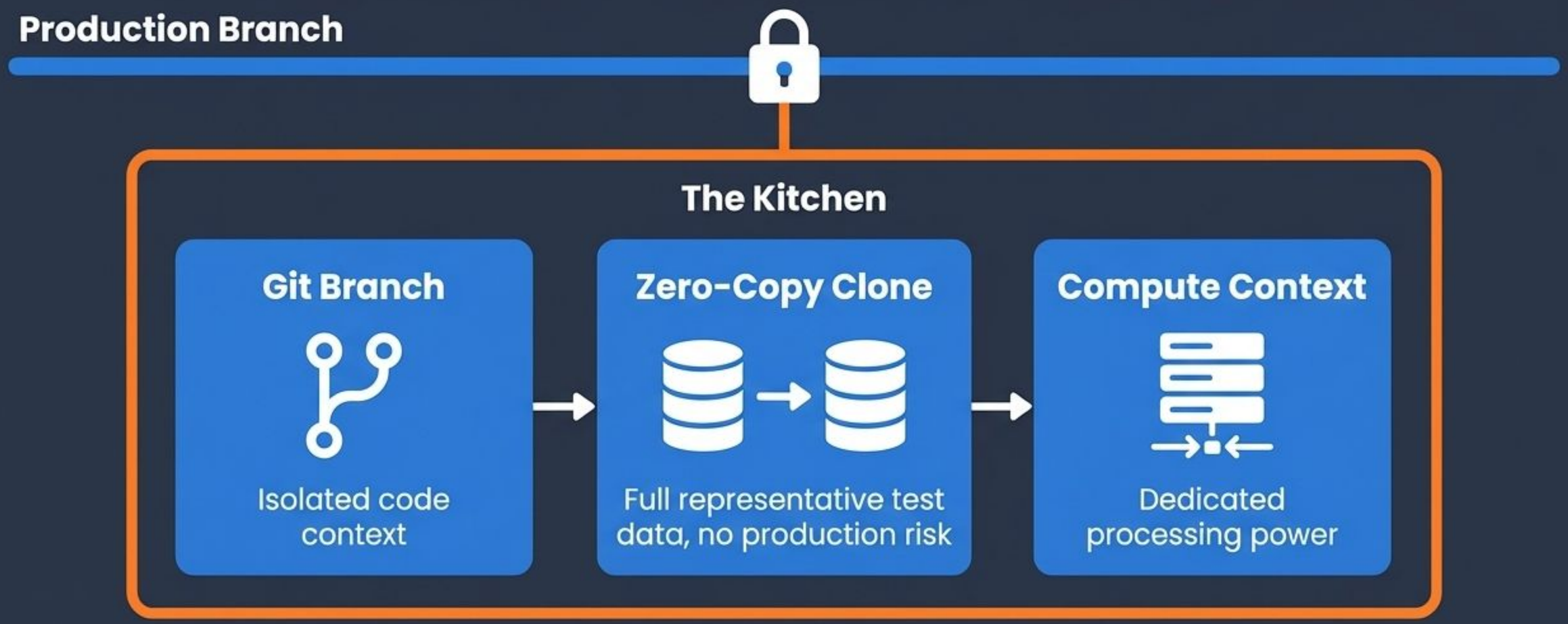
**Layer 3: Data Testing**
Provides Validation.

**Claude Code is the accelerant. This architectural foundation is what makes the acceleration safe.**

DataKitchen

# Layer 1: DataOps & The 'Kitchen' Isolation Model

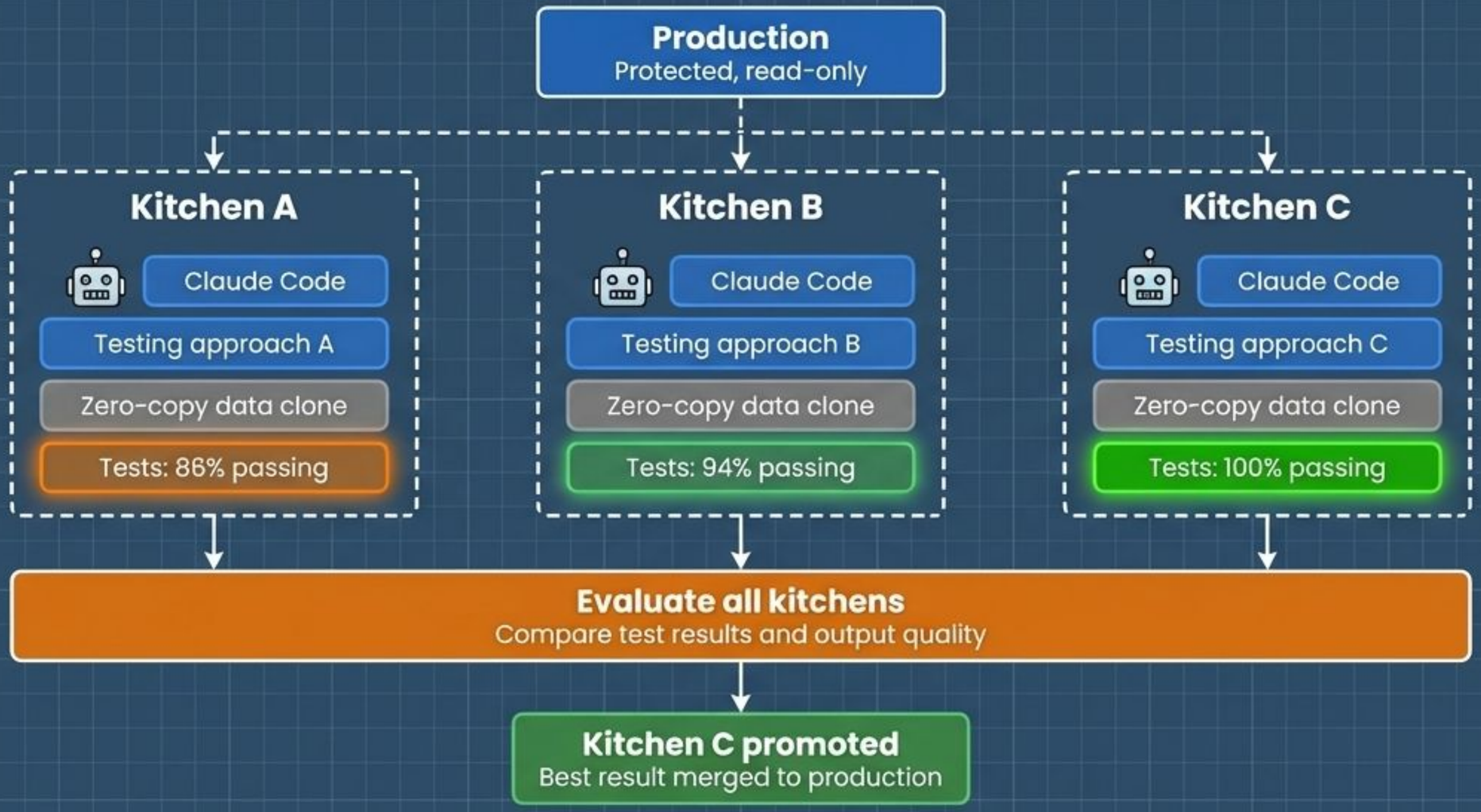
Concurrent AI workloads require Multi-Version Concurrency Control (MVCC) for pipelines.




Claude enters the Kitchen, runs trial-and-error loops unattended, and either fails safely or produces a clean dish. Production remains untouched until explicit promotion.

DataKitchen

# Parallel Agent Orchestrator



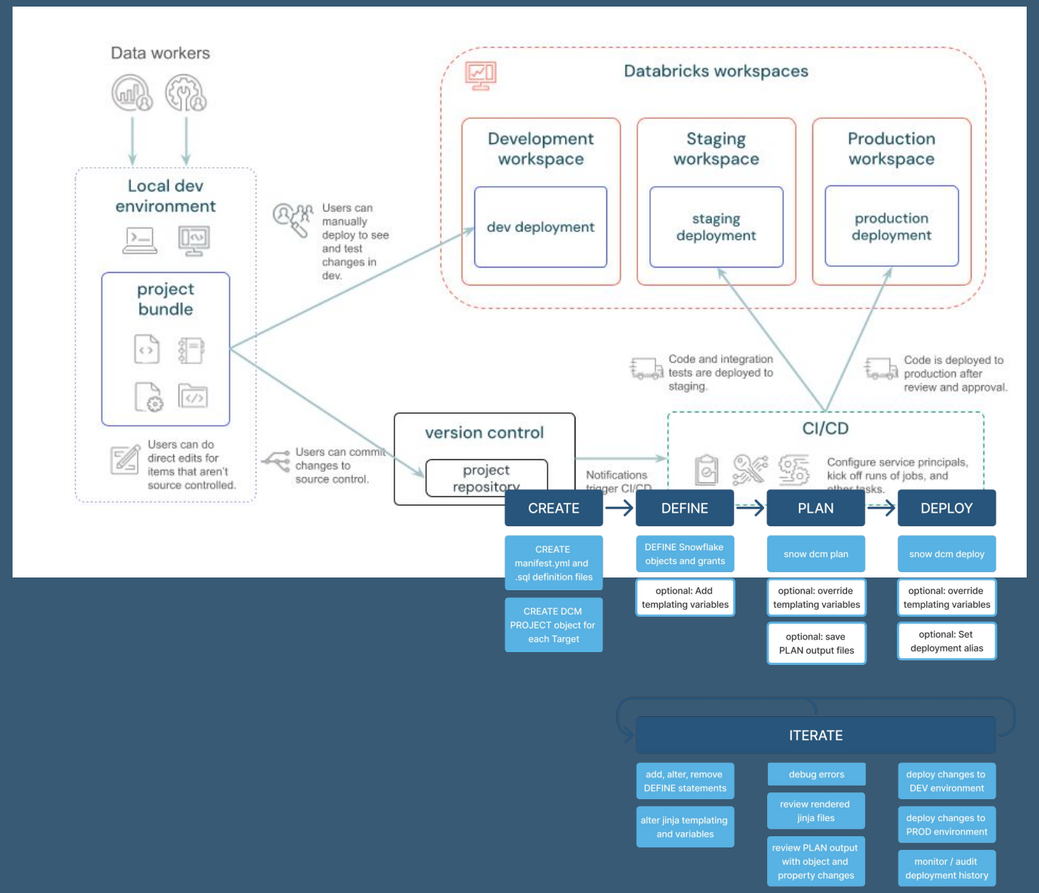

The mental model shift is significant: you move from writing pipeline code to directing a team of autonomous agents, each working in its own safe sandbox.

DataKitchen

# Other ‘Kitchen’ Types

DataKitchen Automation has had this idea for 12 years

- Databricks: Declarative Automation Bundles (formerly known as Databricks Asset Bundles)
- Snowflake released DCM Projects recently
- Abstraction of code, tests, data parameters, and environment.
- AI needs “Cut and paste” you work from one workplace to another




Copyright 2026 by DataKitchen, Inc. All Rights Reserved.

DataKitchen

# Layer 2: FITT Architecture

The architectural principles that make pipelines mathematically predictable and safe for AI trial-and-error.

## F

### Functional

Same input, same output, every time. No hidden state. Transforms are treated as pure mathematical equations.

## I

### Idempotent

Run it a thousand times, get the exact same result. Enables safe, continuous trial-and-error without stacking errors.

## T

### Tested

Feedback loops that enable autonomous quality control. Tests define the precise condition of 'done'.

## T

### Two-stage

Raw data to Final output. No intermediate medallion layers for agents to navigate or break.

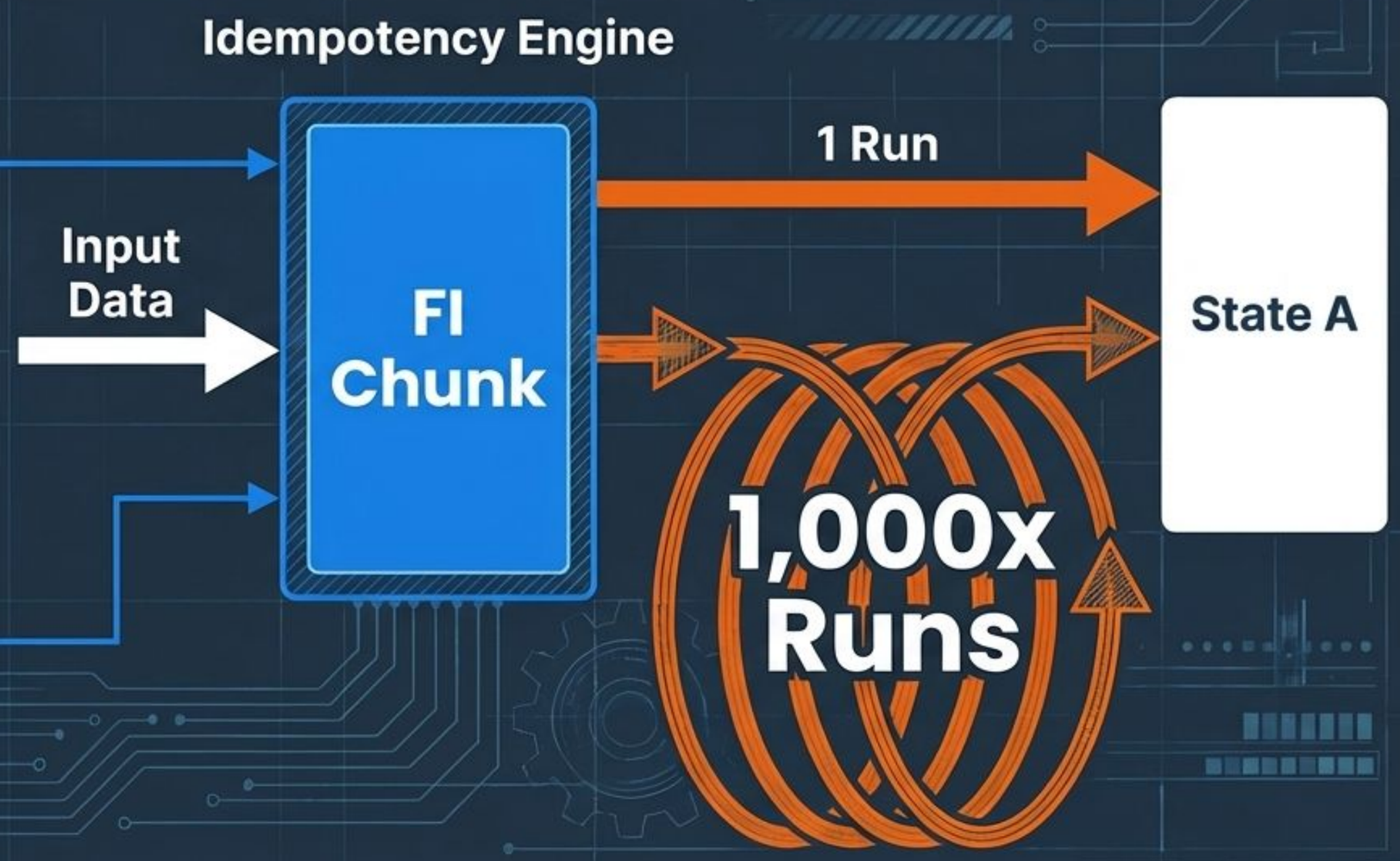
DataKitchen

# Functional & Idempotent: The prerequisites for AI iteration

**Functional:** Same input, same output, every time. No mystery dependencies or hidden state. Claude can treat the transform as a mathematical function.

**Idempotent:** Run it a thousand times, get the same result. If SQL isn't naturally idempotent, re-runs stack errors (e.g., duplicate records).

FI Chunks give Claude a clean slate for every experiment.

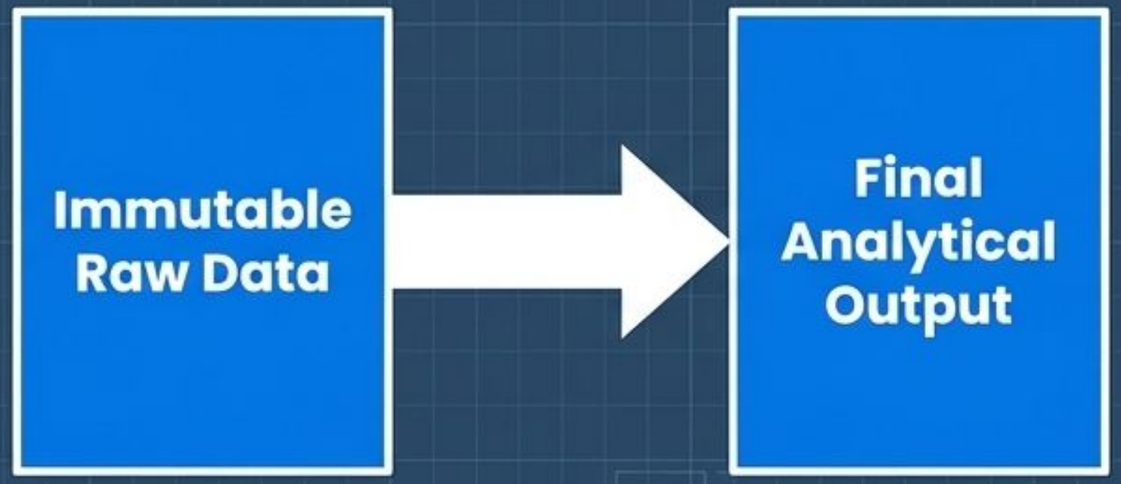



DataKitchen

## The Medallion Maze

Multi-layered architectures force AI to navigate fragile intermediate dependencies.

## The FITT Contract




A complete-rebuild approach in a single orchestrated pass. Claude receives a clear contract.

DataKitchen

# Layer 3: Data Testing (The Autonomous Feedback Loop)

AI can generate code, but it cannot judge correctness.

Claude writes a transform -> Runs the full test suite -> Reads the failure logs -> Adjusts approach -> Tries again.

**Takeaway**: Without data tests, a human must babysit every iteration. With tests, Claude self-directs for hours

DataKitchen

# The Scope of AI Test Coverage

Shift Left (Pipeline Stages)

Shift Down (Unit to E2E Integration)

## Automated Consistency (80%)

Freshness, schema validation, volume checks, and statistical drift detection.

### Domain-Specific Rules (20%)

Custom business logic (e.g., no shipments on restricted days)

## The 1:10:100 Rule

- Preventing a data quality problem at the source costs $1
- Fixing midstream costs $10
- Cleaning it post-customer costs $100

DataKitchen

# DataKitchen DataOps Data Quality TestGen:

## Open Source, Full Featured, Data Quality Tool.

DataKitchen

### IT DOES SIX TASKS:

1. Data Profiling
2. Dataset Screening And Hygiene Review
3. AI Generation of Data Quality Validation Tests, Custom Tests
4. Data Testing
5. Anomaly Monitoring
6. Data Quality Scoring

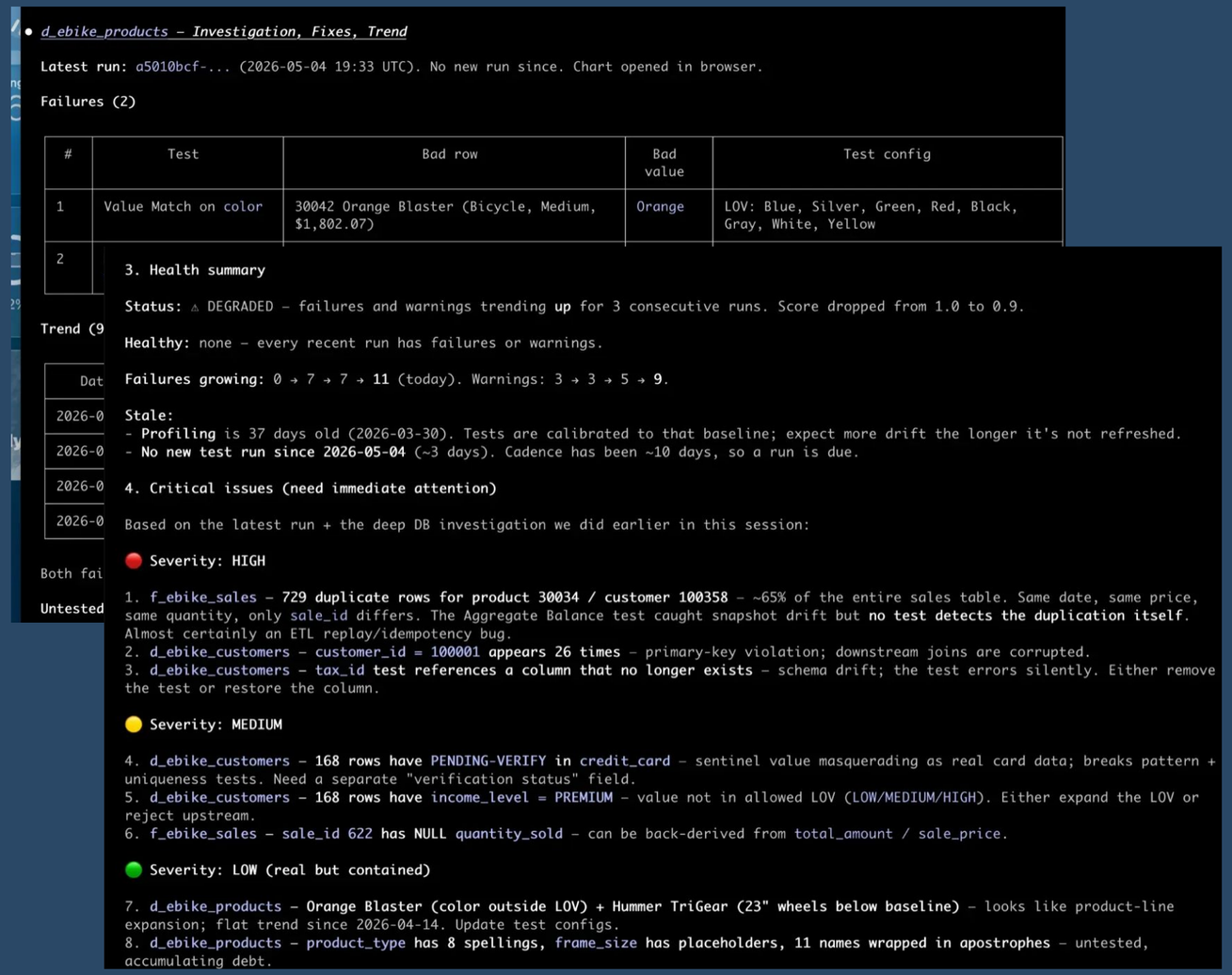
### KEY FEATURES

- In Database Execution
- Full Featured (UI, AI, Rules) One User
- Enterprise Version starts at $100 per user per month

Copyright 2026 by DataKitchen, Inc. All Rights Reserved.

# TestGen MCP interface

- TestGen 5.32.2 ships an MCP server. Open source, runs behind your firewall
- Allows You to view all test results and diagnose failures
- Have claude recommend tests
- Add new tests
- Blog: datakitchen.io/blog/while-you-slept-an-agent-fixed-14-data-quality-failures/



● *d_ebike_products – Investigation, Fixes, Trend*

**Latest run:** a5010bcf-... (2026-05-04 19:33 UTC). No new run since. Chart opened in browser.

**Failures (2)**

| # | Test | Bad row | Bad value | Test config |
|---|---|---|---|---|
| 1 | Value Match on color | 30042 Orange Blaster (Bicycle, Medium, $1,802.07) | Orange | LOV: Blue, Silver, Green, Red, Black, Gray, White, Yellow |
| 2 | | | | |

Trend (9

| Dat |
|---|
| 2026-0 |
| 2026-0 |
| 2026-0 |
| 2026-0 |

Both fai

Untested

**3. Health summary**

**Status:** ⚠ DEGRADED – failures and warnings trending **up** for 3 consecutive runs. Score dropped from 1.0 to 0.9.

**Healthy:** none – every recent run has failures or warnings.

**Failures growing:** 0 → 7 → 7 → **11** (today). Warnings: 3 → 3 → 5 → **9**.

**Stale:**
- **Profiling** is 37 days old (2026-03-30). Tests are calibrated to that baseline; expect more drift the longer it's not refreshed.
- **No new test run since 2026-05-04** (~3 days). Cadence has been ~10 days, so a run is due.

**4. Critical issues (need immediate attention)**

Based on the latest run + the deep DB investigation we did earlier in this session:

🔴 **Severity: HIGH**

1. f_ebike_sales – **729 duplicate rows for product 30034 / customer 100358** – ~65% of the entire sales table. Same date, same price, same quantity, only sale_id differs. The Aggregate Balance test caught snapshot drift but **no test detects the duplication itself**. Almost certainly an ETL replay/idempotency bug.
2. d_ebike_customers – customer_id = 100001 **appears 26 times** – primary-key violation; downstream joins are corrupted.
3. d_ebike_customers – tax_id **test references a column that no longer exists** – schema drift; the test errors silently. Either remove the test or restore the column.

🟡 **Severity: MEDIUM**

4. d_ebike_customers – **168 rows have** PENDING-VERIFY **in** credit_card – sentinel value masquerading as real card data; breaks pattern + uniqueness tests. Need a separate "verification status" field.
5. d_ebike_customers – **168 rows have** income_level = PREMIUM – value not in allowed LOV (LOW/MEDIUM/HIGH). Either expand the LOV or reject upstream.
6. f_ebike_sales – sale_id 622 **has NULL** quantity_sold – can be back-derived from total_amount / sale_price.

🟢 **Severity: LOW (real but contained)**

7. d_ebike_products – **Orange Blaster (color outside LOV) + Hummer TriGear (23" wheels below baseline)** – looks like product-line expansion; flat trend since 2026-04-14. Update test configs.
8. d_ebike_products – product_type **has 8 spellings,** frame_size **has placeholders, 11 names wrapped in apostrophes** – untested, accumulating debt.

Copyright 2026 by DataKitchen, Inc. All Rights Reserved.

### Step 1: Build Test Coverage

Start with error-prone transforms. Even 50% coverage changes what an agent can do autonomously.

1

### Step 2: Refactor to Idempotency

Restructure one messy pipeline stage into a pure FITT unit to surface hidden dependencies.

2

### Step 3: Stand Up Kitchens

Create isolated, branched environments. Agents cannot iterate safely without a staging sandbox.

3

### Step 4: Introduce Claude Code

Let the AI learn from the FITT units and execute safely within the established boundaries.

4

DataKitchen

**CONTEXT PERSISTENCE**
Aggressively use `CLAUDE.md` to persist memory, schemas, and business rules across sessions. Version control these files.

**CONTEXT FLUSHING**
Watch the context usage bar. Execute `/clear` at 75% to prevent degradation.

**SESSION MANAGEMENT**
Use `--continue` or `--resume` to manage scoped pipeline folders. Keep PRD documents updated dynamically.

## Prompting Discipline

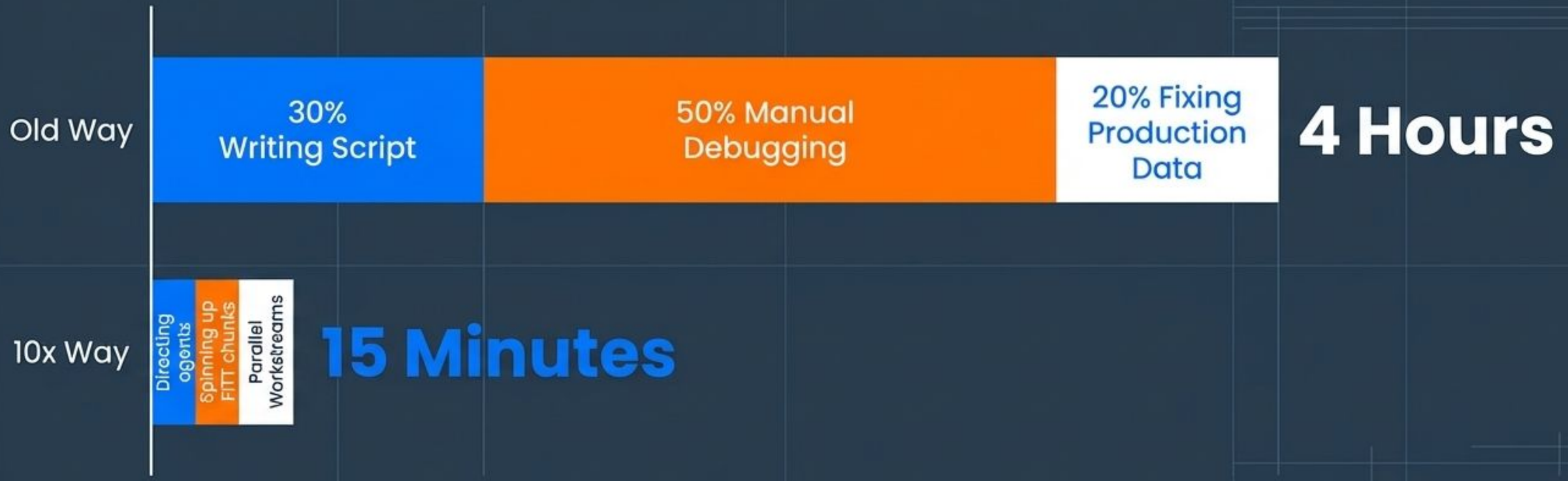
Ask for a plan before coding.

Demand minimal changes.

Force Claude to write a test for every single modification.

DataKitchen

# The Reality of 10x Data Engineering




10x does not mean AI writes 10x more lines of code. It means a 4-hour task collapses to 15 minutes. The focus shifts entirely to what needs to be built, not how to build it.

DataKitchen

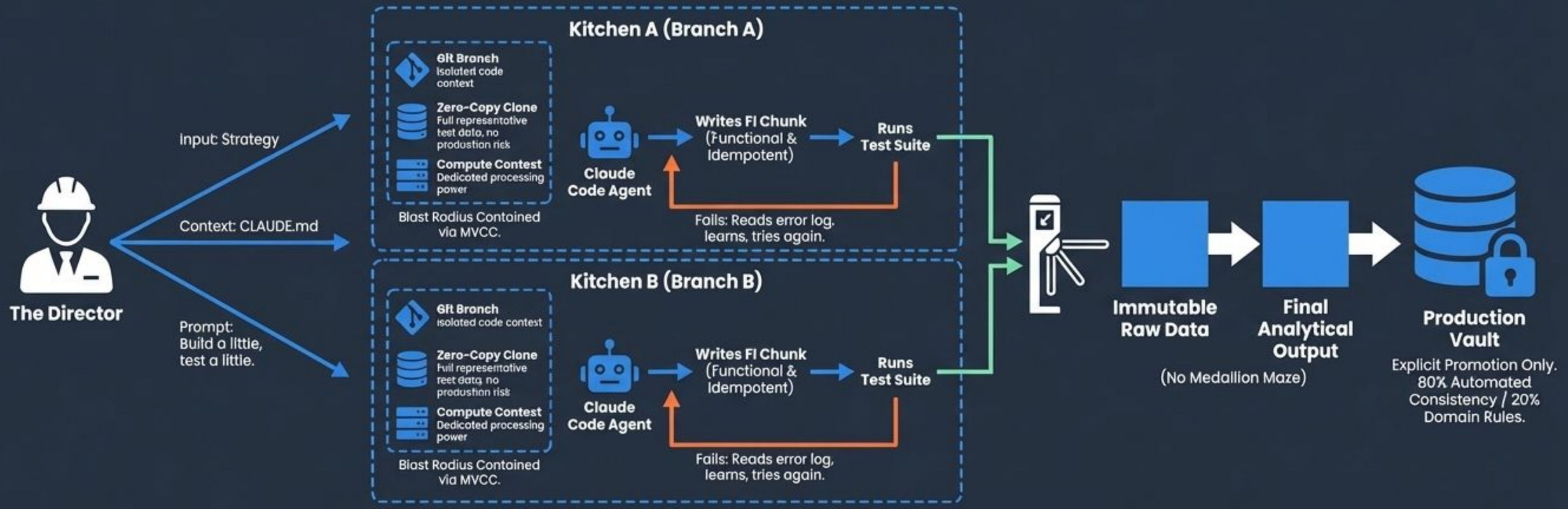
# The Blueprint for Autonomous Data Engineering

## DataOps + FITT + Data Testing = 10x Productivity

**Zone 1: The Orchestrator** (Left 15%)

**Zone 2 & 3: The Agentic Sandbox & Autonomous Loop** (Center 45%)

**Zone 4: The Filter & The Vault** (Right 40%)




**The 10x Paradigm Shift Matrix (Bottom 15%)**

| | | |
|---|---|---|
| **The Old Way vs. The 10x Way** | Fragile | **4 Hours** (Sequential branch -> Manual script -> Medallion maze -> Manual debugging) |
| | 10x Way - FITT & Agentic | **15 Minutes** (Parallel Kitchens -> Agentic loops -> 2-Stage FITT -> Automated gates) |

DataKitchen

# Opportunity to use AI to 10x the productivity of:

## The Integration Ecosystem

Data Engineers who are integrating the data

**DataOps + FITT + Data Testing = 10x**

## The Insight Ecosystem

Data Users directly using the data to give business insight

**DT + DX + CTX = 10x**

The architectural foundation is what makes the acceleration safe across the entire enterprise.

NotebookLM