Dynamic DAGs and Data Quality using DAGFactory

Ashir Alam
Gangfeng Huang

Intuit Credit Karma

About Intuit Credit Karma

- **Mission**: Champion financial progress for 140M+ members worldwide.

- **Vision**: A self-driving platform empowering every financial journey.

About Credit Ecosystem Team

We are the front door for all member financial data at Credit Karma. We ingest, model, and serve the core data that powers our products.

Data Ingestion: The first layer for all data provider & partner data (API & Batch).

Service Integration: Owning 20+ microservices.

Single Source of Truth: Building the unified model of our members' financial profiles.

About Us

Ashir Alam

- Senior Data Engineer at Intuit Credit Karma
- Graduated from USC in 2019
- Working with data migrations, Data Quality, microservices, Data platform, GCP Infra

Gangfeng Huang

- Business Intelligence Engineer at Intuit Credit Karma
- Go Illini!
- Working with scalable data pipelines, self-service platforms, SQL & dashboard optimization, data quality assurance

Agenda

01 **The Challenge:** DAG Sprawl & The Copy-Paste Problem

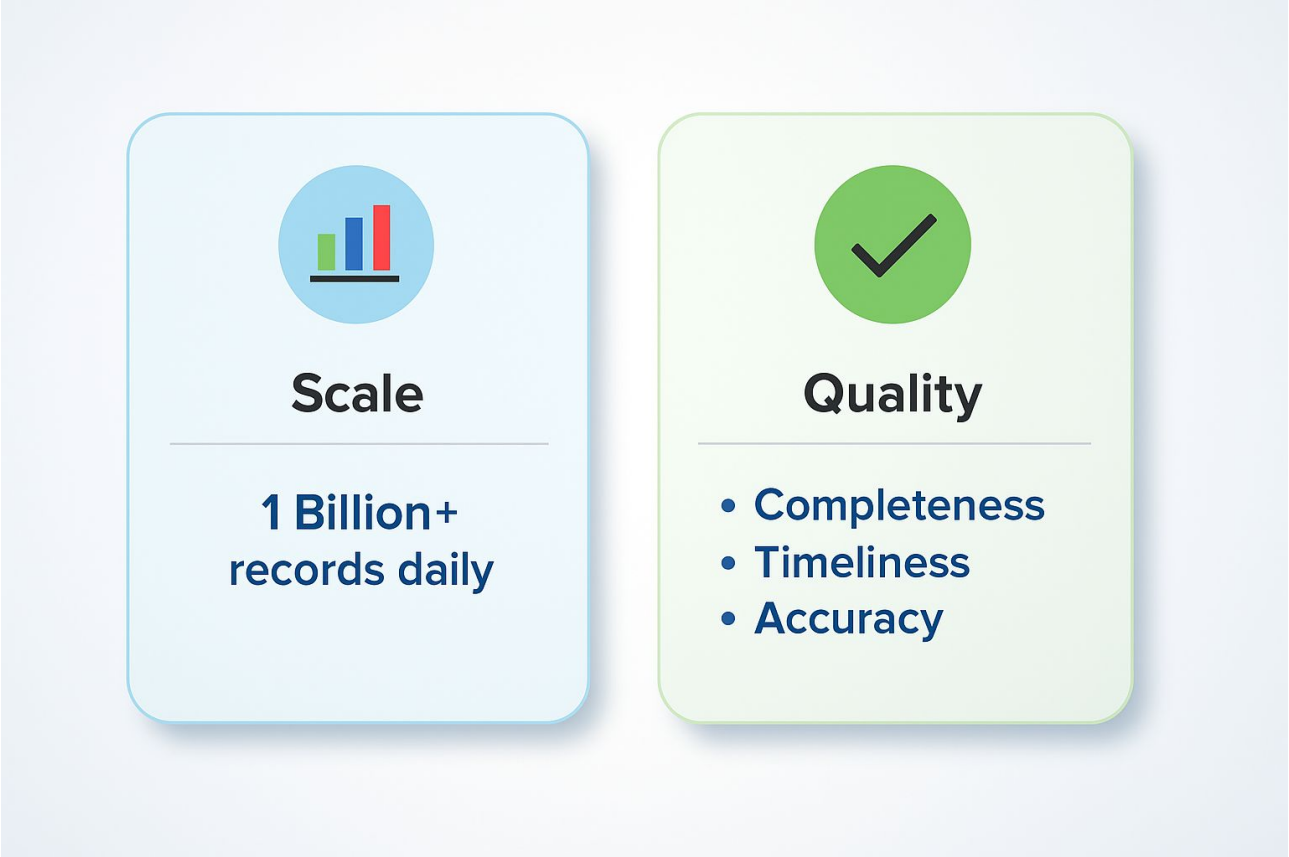
02 **The Solution:** A "No-Code" Approach to DAG Creation

03 **Demo:** A Walkthrough of the Experience

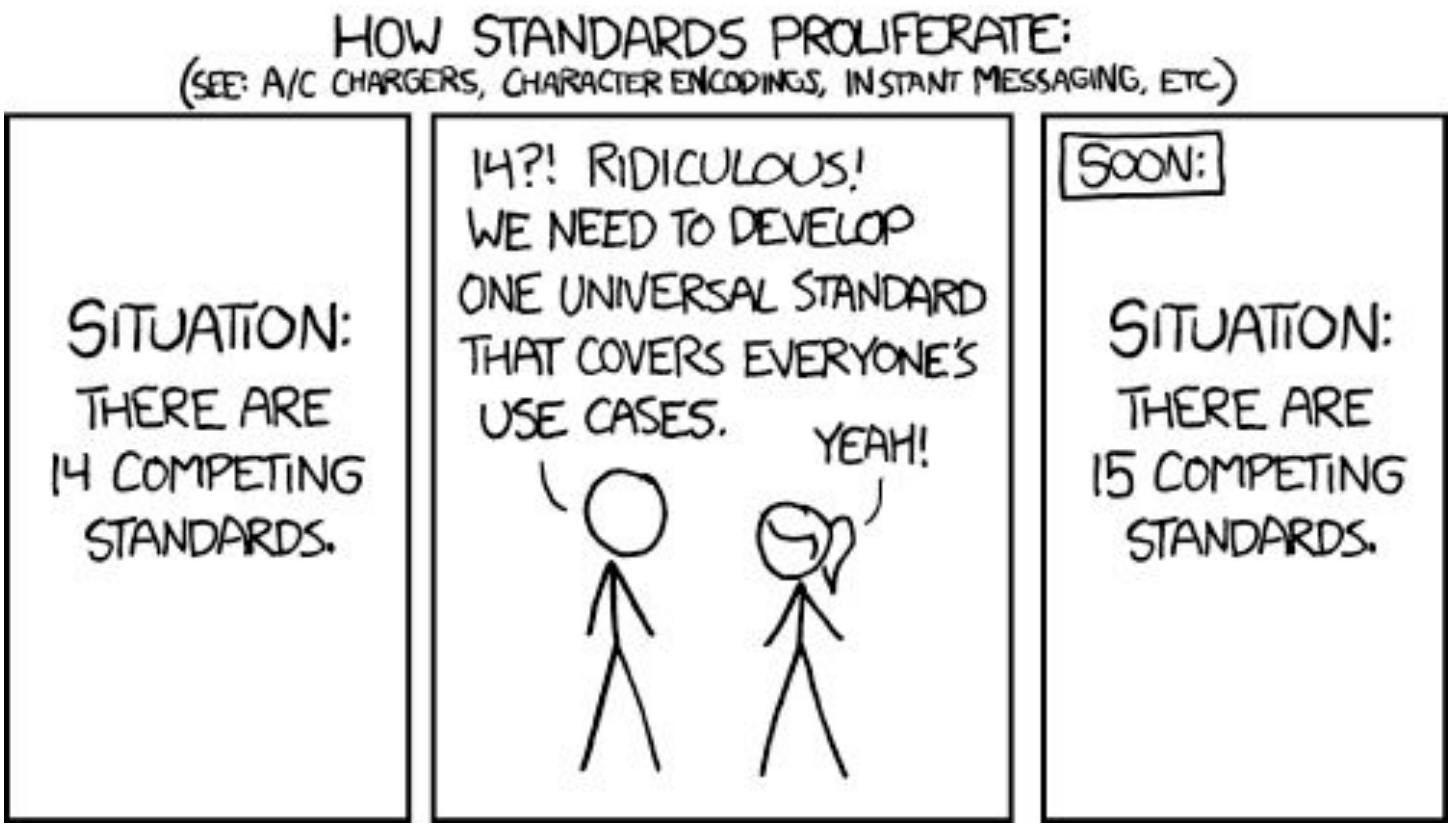
04 **Key Achievements & Impact**

05 **Q&A**

Data Pipelines Handled by Our Team



Data Quality Check Standards



The Problem: Scaling Data Quality Checks

DAG Sprawl

Dozens of nearly identical DAGs cluttering the system, making it impossible to manage efficiently.

The Copy-Paste Problem

Human errors introduced with every new check, creating inconsistencies and bugs across the codebase.

High Maintenance

A small change requires updating multiple files, consuming valuable development time.

The Bottleneck

Best engineers stuck writing repetitive boilerplate code instead of solving complex problems.

Our Previous Architecture & Limitations for DQ

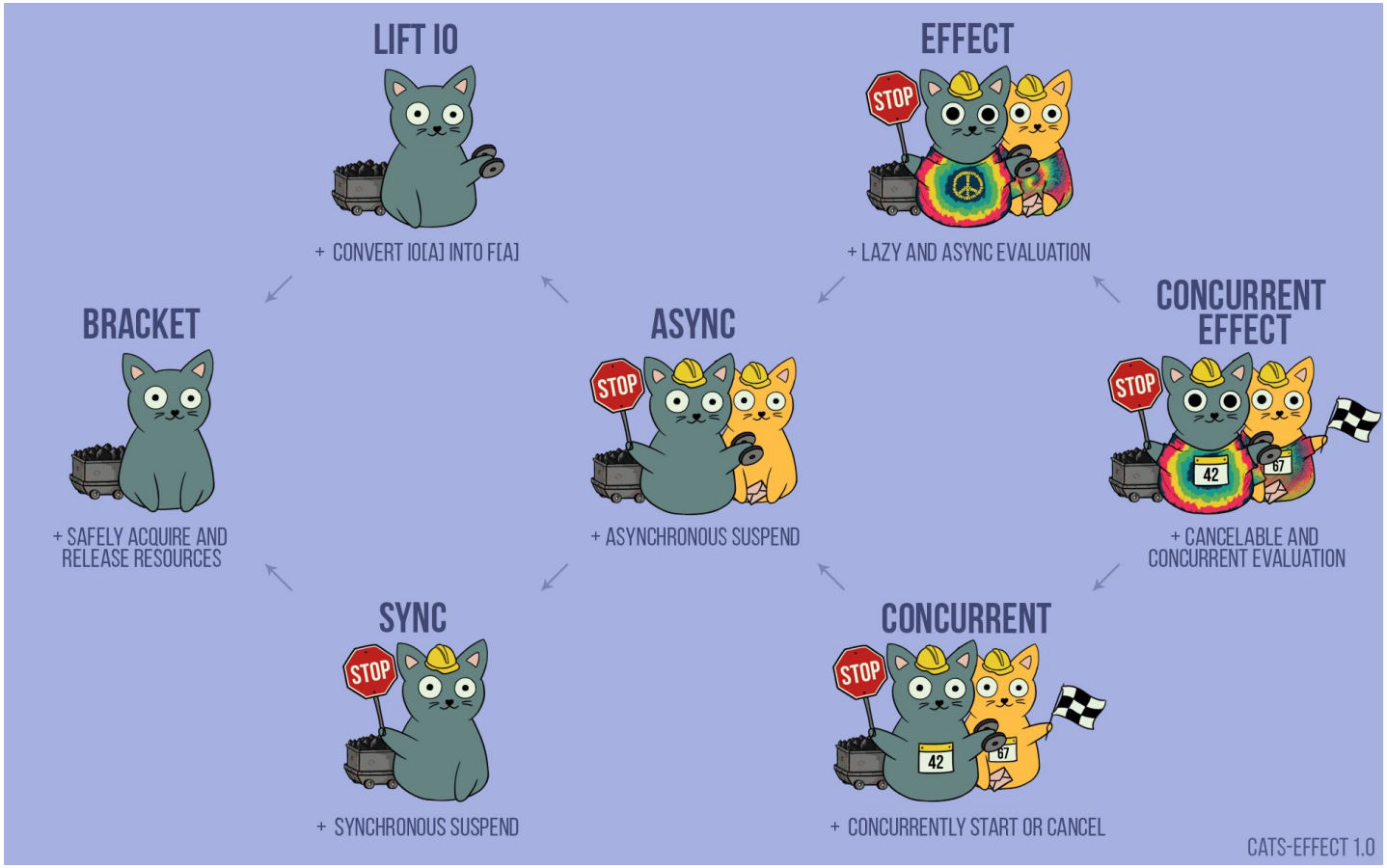
Tech Stack:

- Scala microservice (Finagle, Cats Effect, Akka Streaming)

Limitations:

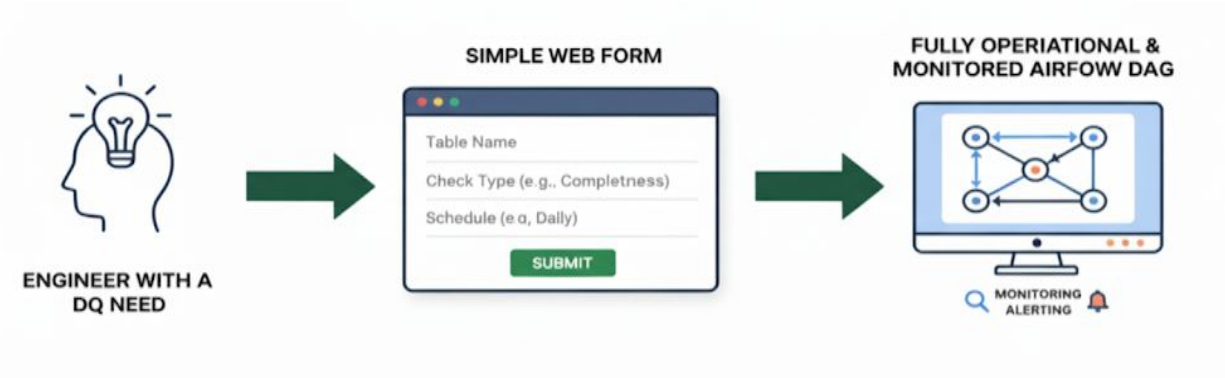
- Not Big Data-optimized → frequent timeouts
- Missing integrations (no PagerDuty, weak email alerts)
- High entry barrier → required Scala expertise
- Lack of Data Quality Alerts
- Lack of Data Quality metrics

Scala Cats Effects



The Solution: A UI-Driven, "No-Code" Approach

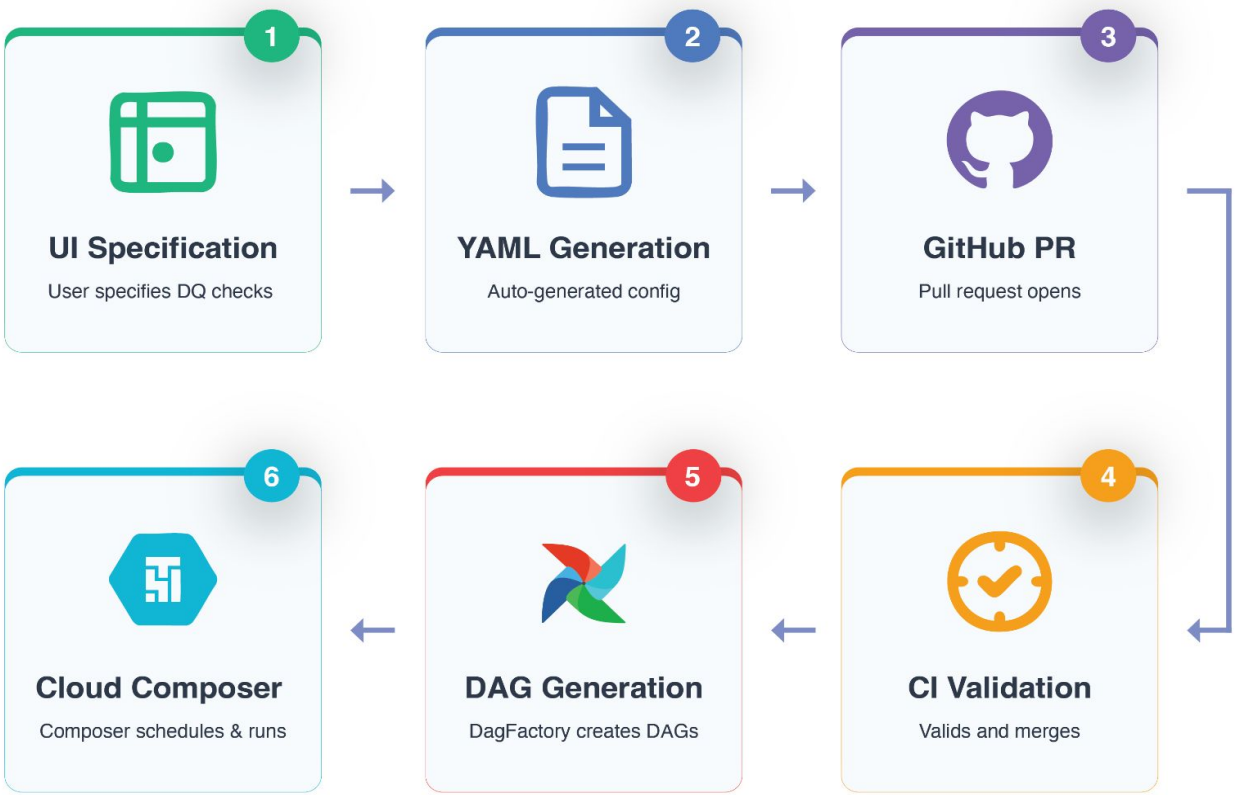
Our Mission: Author data quality pipelines without writing a single line of Python.



Core Principles:

- Empowerment: Enable any engineer to create DQ checks in minutes.
- Abstraction: Hide the complexity. Users declare what they need, not how to build it.
- Automated Governance: Bake standards, security, and alerting directly into the workflow.

Automated DQ Workflow Architecture



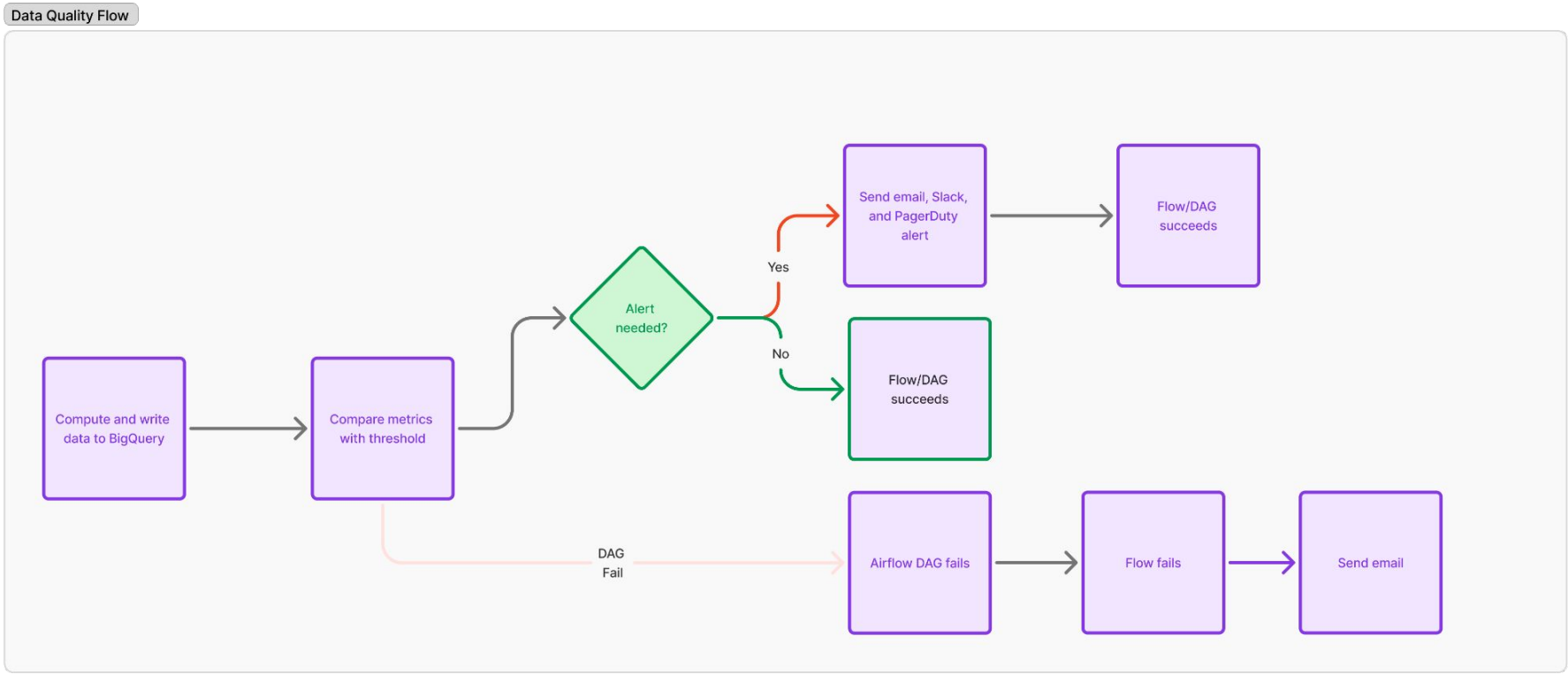
The Core: From YAML to a DAG

```JSON
bureau_daily_completeness_score_usa:
  default_args:
    owner: 'credit-eco'
    start_date: "2025-04-01"
    retries: 1
    retry_delay_sec: 10
    use_legacy_sql: False
    email_on_failure: True # receive email if failed
    email_on_retry: false
    email: # This is DAG failure email
      - "robert.huang@creditkarma.com"
      - "ashir.alam@creditkarma.com"
  schedule_interval: 0 20 15 * *
  catchup: False
  concurrency: 1
  max_active_runs: 1
  description: 'Alert Name: bureau_daily_completeness_score_usa'
  tags:
    - "completeness"

  tasks:
    calculate_metric_write_to_table:
      operator: airflow.contrib.operators.bigquery_operator.BigQueryOperator
      use_legacy_sql: False
      sql: "sql/dqm/completeness/bureau_daily_completeness_score_usa.sql"
      write_disposition: 'WRITE_APPEND'
      destination_dataset_table:
"prod-ce-data.ce_metrics.ce_data_quality_stats" # unified_data_quality_table
      create_disposition: 'CREATE_IF_NEEDED'
      allow_large_results: True
```

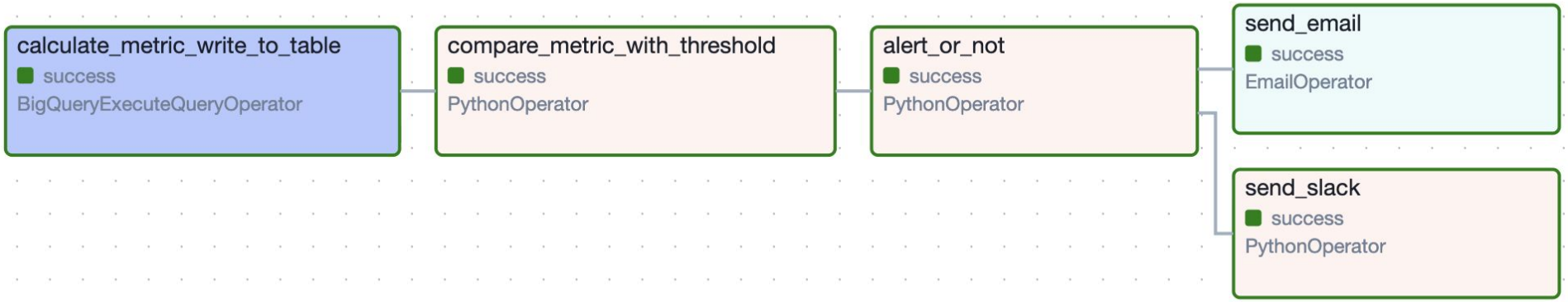
Python "Glue" Code

```
from pathlib import Path
from airflow import DAG
from airflow.configuration import conf as airflow_conf
from dagfactory import load_yaml_dags

config_dir = "/home/airflow/gcs/dags/configs"
load_yaml_dags(globals_dict=globals(), dags_folder=config_dir)
```

Data Quality DAG Flow



The End Result: A Production DAG



The End Result: Slack Alert

CE Data Alert Bot APP 12:02 PM

Alert_name: [redacted]accuracy_alert_usa_Inquiry_industry_code_description

Partition_date: 2025-10-01

Rule_name: Inquiry_industry_code_description

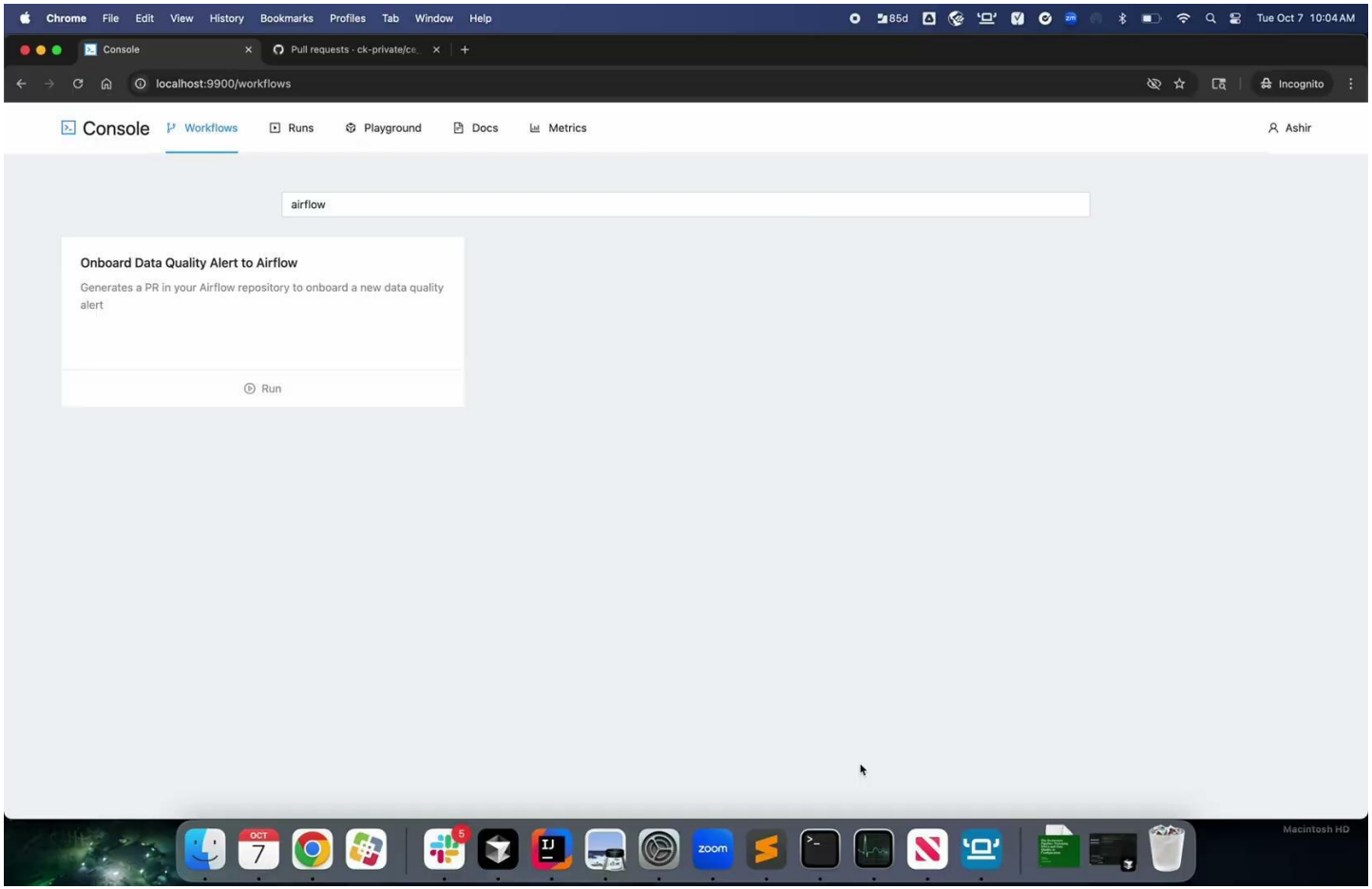
Metric: 1.0

Airflow: https://28d4f53b1d884a0885f674d4a7a3eaae-dot-us-central1.composer.googleusercontent.com/dags[redacted]alerts_accuracy_alert_usa_Inquiry_industry_code_description/grid

Message: @ce-offline-oncall [redacted] We have ingested Inquiry 1B alerts [redacted] in the last 24 hours with possibly anomalous industry code and description values

Runbook: https://credit-ecosystem.static.corp.creditkarma.com/external/services/[redacted]-alert-service/runbook

Demo: A User's Journey



Achievements & Impact

95%
Reduction in Time-to-Deploy
New DQ checks go live in minutes, not days

90%
Faster Alert Response
PagerDuty integration cuts response time dramatically

100%
Error Elimination
Zero copy-paste bugs and manual deployment errors

Self-Service
Developer Empowerment
Engineers build pipelines via UI—no specialist needed

Standardized
Unified Governance
DQ patterns and best practices across all teams

Legacy Scala microservice retired • Modern, scalable architecture

Challenges & Key Takeaways

Challenges We Faced	Key Lessons Learned
🎯 Balancing Simplicity & Power	💡 Abstract aggressively to make powerful tools accessible.
📈 Outgrowing Our Infrastructure	🚀 Plan to Scale Managed services accelerate development, but you must anticipate future growth.
📜 Rapid feature growth made versioning and documenting our YAML a major operational burden.	👑 Declarative is King YAML provides a version-controlled, auditable, and maintainable source of truth.
👥 Driving User Adoption Building trust to shift developers from code to our UI required training.	📚 Onboarding is Everything A great tool is useless without the documentation and training to build user confidence.

QUESTIONS?

Thanks!!

Ashir Alam

ashir_alam@intuit.com

https://www.linkedin.com/in/ashir-alam/

Gangfeng Huang

gangfeng_huang@intuit.com