#FABCONSQLCON2026

FABCON
Microsoft Fabric
COMMUNITY CONFERENCE

SQLCON
Microsoft SQL
COMMUNITY CONFERENCE

ATLANTA MARCH 16 - 20, 2026

Motio

CI/CD and DevOps for Power BI: Beyond the Happy Path

Amir Sarir

Senior BI Solutions Engineer

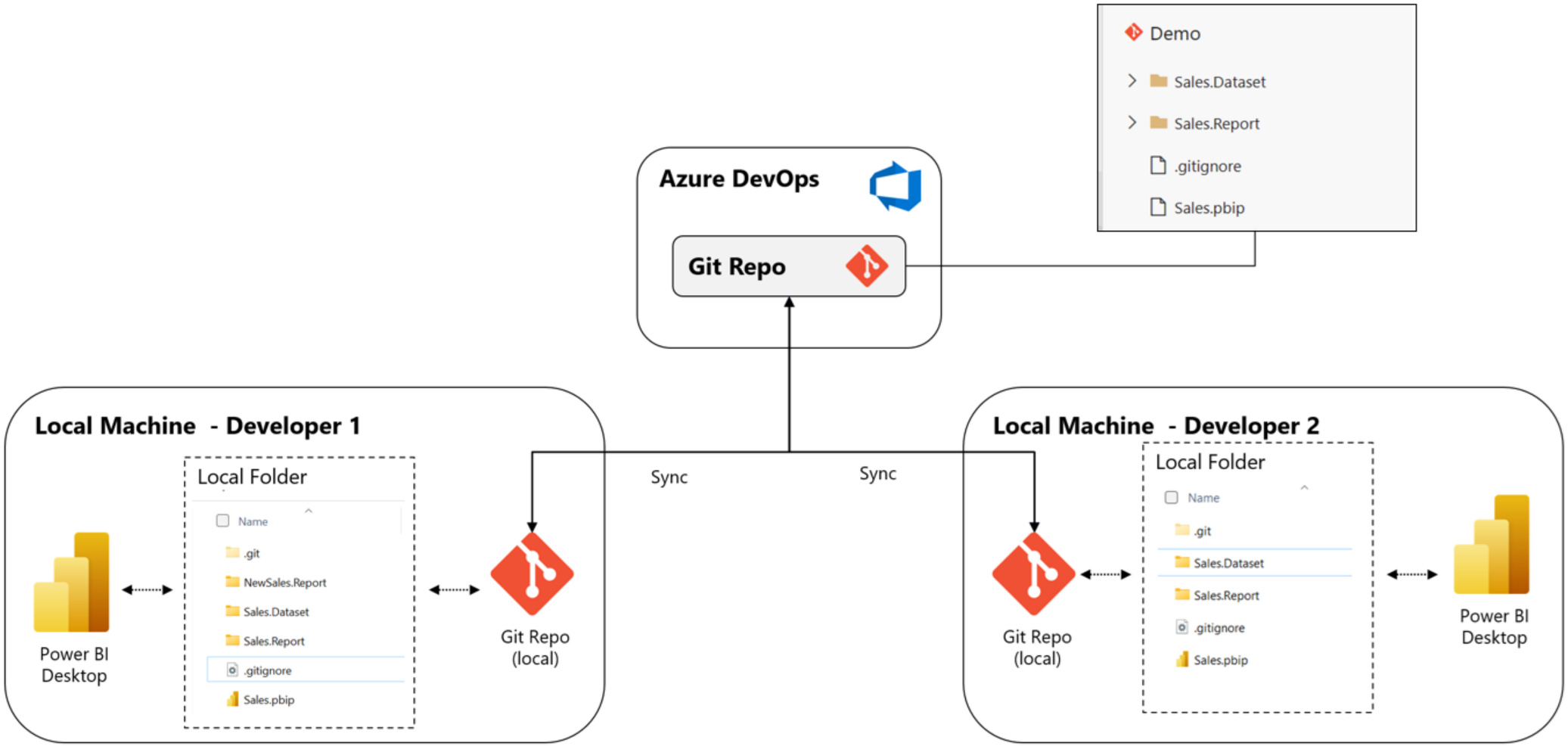
Announcement…

We will have a drawing for a **$50 Amazon gift card** by answering a single question survey about the session.

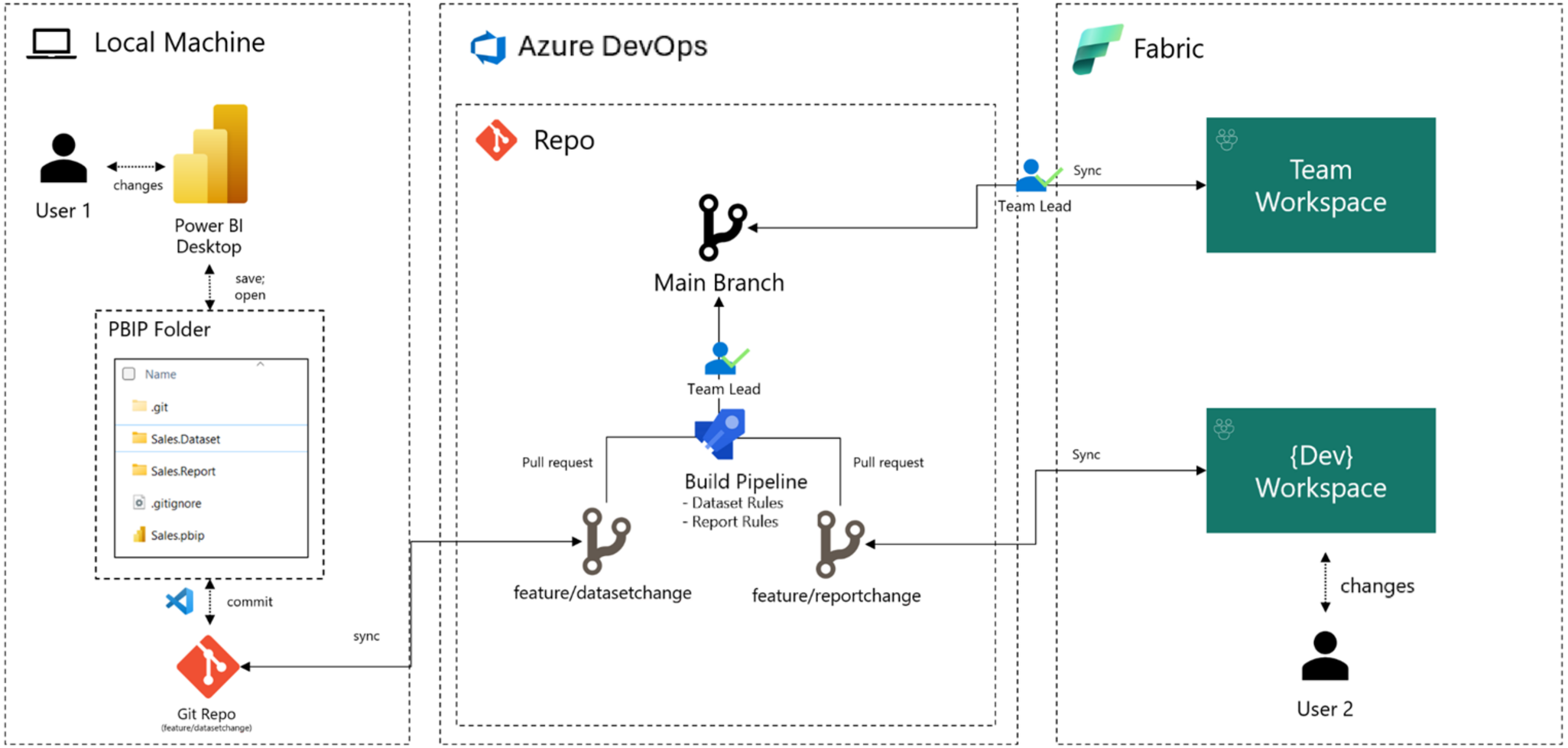
Agenda

- The “Official” Happy Path
- What is the Single Source of Truth?
- DevOps in Power BI vs. Software Development
- Beyond the Happy Path of Git
- Rollbacks: the Myth and the Legend
- Beyond the Happy Path of Deployment
- Patterns that Actually Work

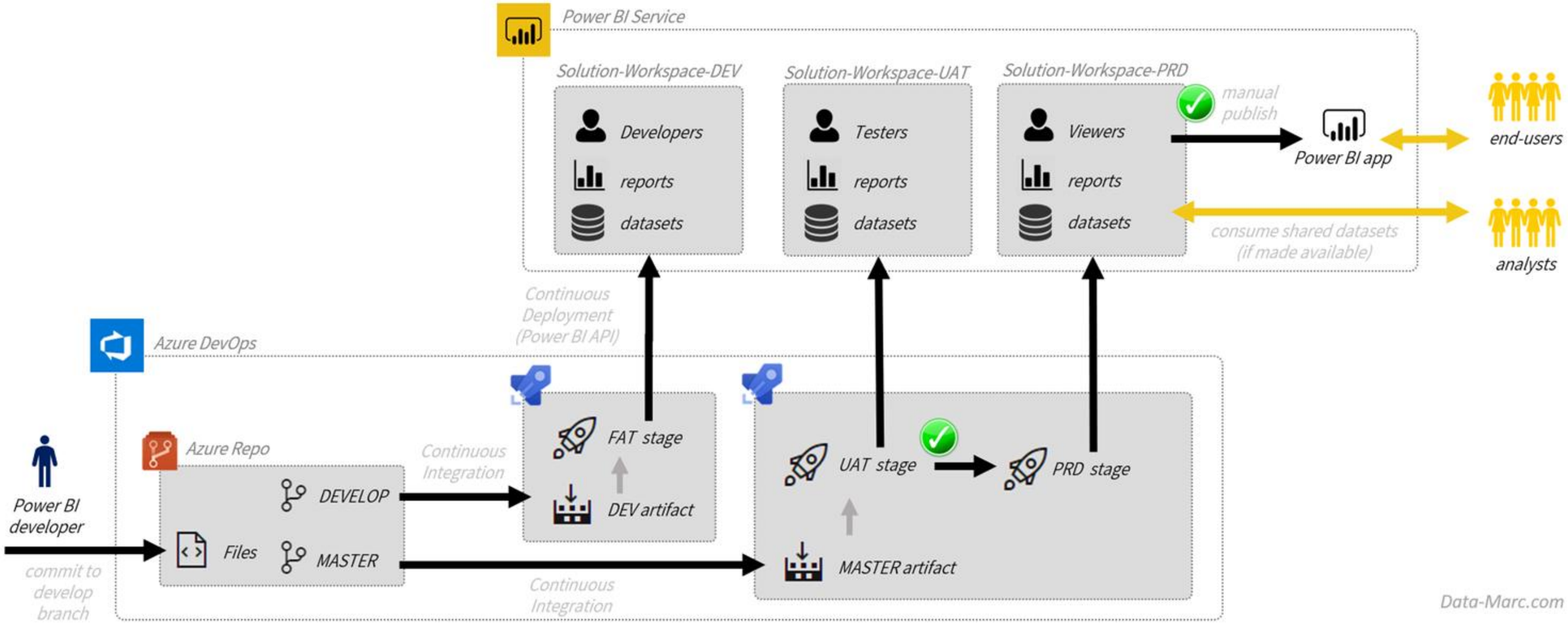
So far…


Demo
Sales.Dataset
Sales.Report
.gitignore
Sales.pbip
Azure DevOps
Git Repo
Local Machine - Developer 1
Local Folder
Name
.git
NewSales.Report
Sales.Dataset
Sales.Report
.gitignore
Sales.pbip
Power BI Desktop
Git Repo (local)
Sync
Sync
Local Machine - Developer 2
Local Folder
Name
.git
Sales.Dataset
Sales.Report
.gitignore
Sales.pbip
Git Repo (local)
Power BI Desktop

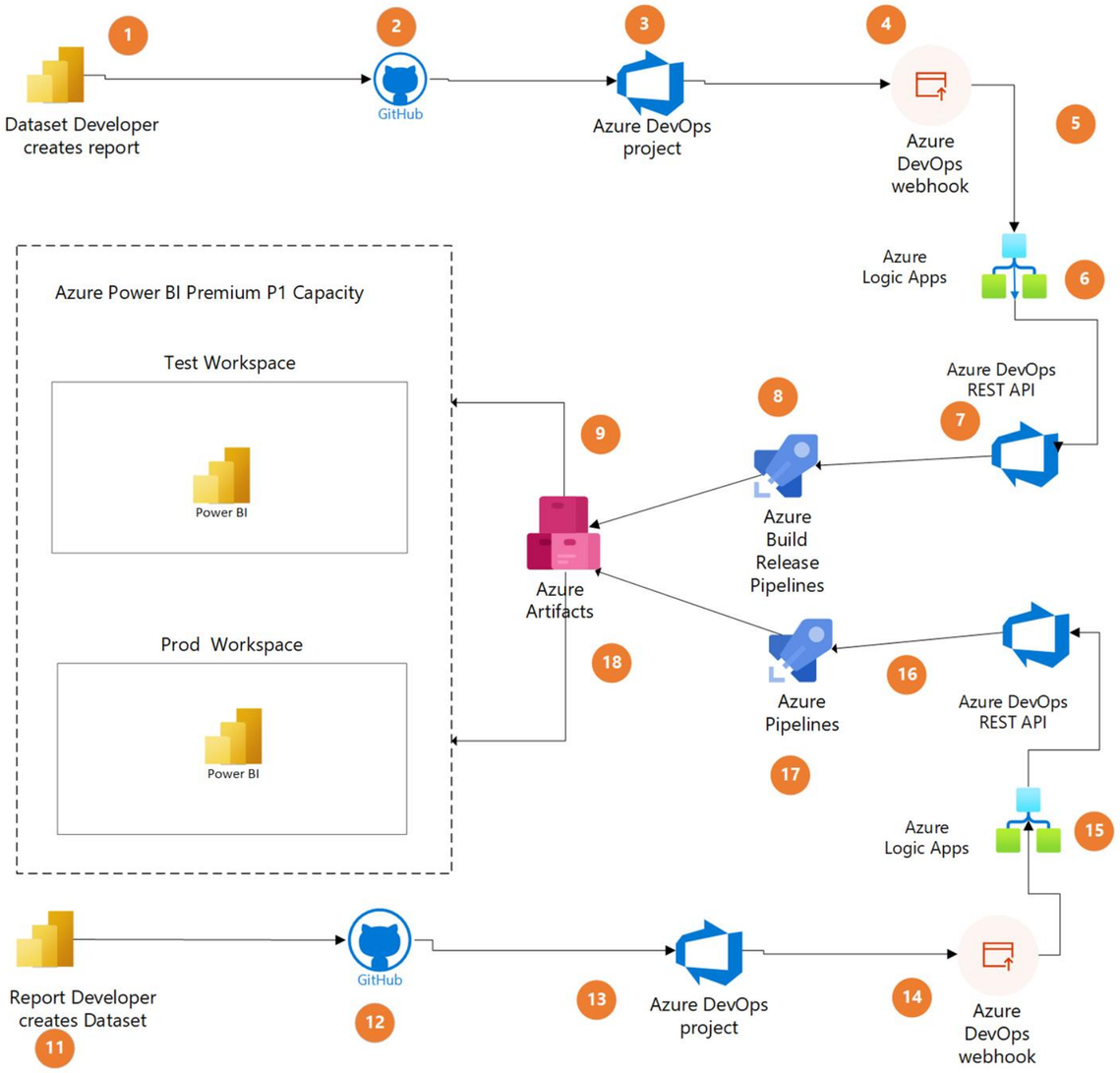
So far…


Local Machine
User 1
changes
Power BI Desktop
save; open
PBIP Folder
Name
.git
Sales.Dataset
Sales.Report
.gitignore
Sales.pbip
commit
sync
Git Repo
(feature/datasetchange)
Azure DevOps
Repo
Main Branch
Team Lead
Pull request
Build Pipeline
- Dataset Rules
- Report Rules
Pull request
feature/datasetchange
feature/reportchange
Fabric
Team Lead
Sync
Team Workspace
Sync
{Dev} Workspace
changes
User 2

So far…


Power BI Service
Solution-Workspace-DEV
Solution-Workspace-UAT
Solution-Workspace-PRD
Developers
reports
datasets
Testers
reports
datasets
Viewers
reports
datasets
manual publish
Power BI app
end-users
consume shared datasets (if made available)
analysts
Continuous Deployment (Power BI API)
Azure DevOps
Azure Repo
Power BI developer
commit to develop branch
DEVELOP
Files
MASTER
Continuous Integration
FAT stage
DEV artifact
UAT stage
PRD stage
MASTER artifact
Continuous Integration
Data-Marc.com

So far…


1
Dataset Developer creates report
2
GitHub
3
Azure DevOps project
4
Azure DevOps webhook
5
Azure Logic Apps
6
Azure DevOps REST API
7
8
Azure Build Release Pipelines
9
Azure Artifacts
Azure Power BI Premium P1 Capacity
Test Workspace
Power BI
Prod Workspace
Power BI
18
16
Azure Pipelines
17
Azure DevOps REST API
Azure Logic Apps
15
Report Developer creates Dataset
11
GitHub
12
13
Azure DevOps project
14
Azure DevOps webhook

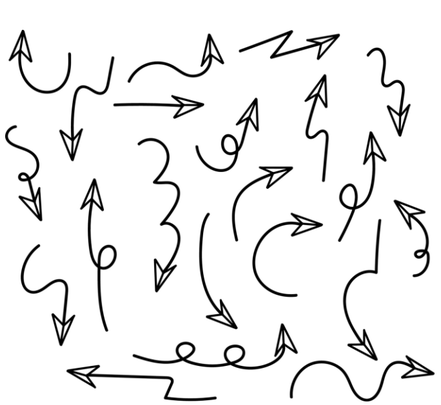
So far…

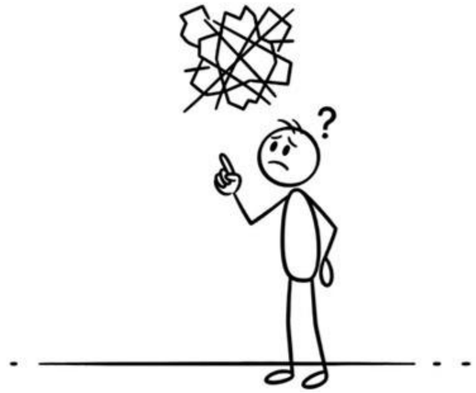
Gotta love AI…

Why This Talk Exists

Key Organizational, Governance, and Technical Barriers to Establishing DevOps Practices in Power BI Environments

- Difficult to implement
- Requires changes to existing practice
- Most teams don’t trust it
- Trust is the real failure metric





The Official Happy Path

AZURE DEVOPS

- **Version control host** for Power BI artifacts (PBIP, JSON, scripts)
- **Runs pipelines** to automate validation, promotion, and deployment
- **Work item tracking** tied directly to commits and releases
- **Enforces process** (branch policies, approvals, gates releases)

GIT

- Stores **Power BI project files** (PBIX, JSON definitions, metadata, scripts)
- Enables **branching, merging, and pull requests** so teams can collaborate safely
- Acts as the source of truth for CI triggers; pushes/PRs can kick off pipelines.

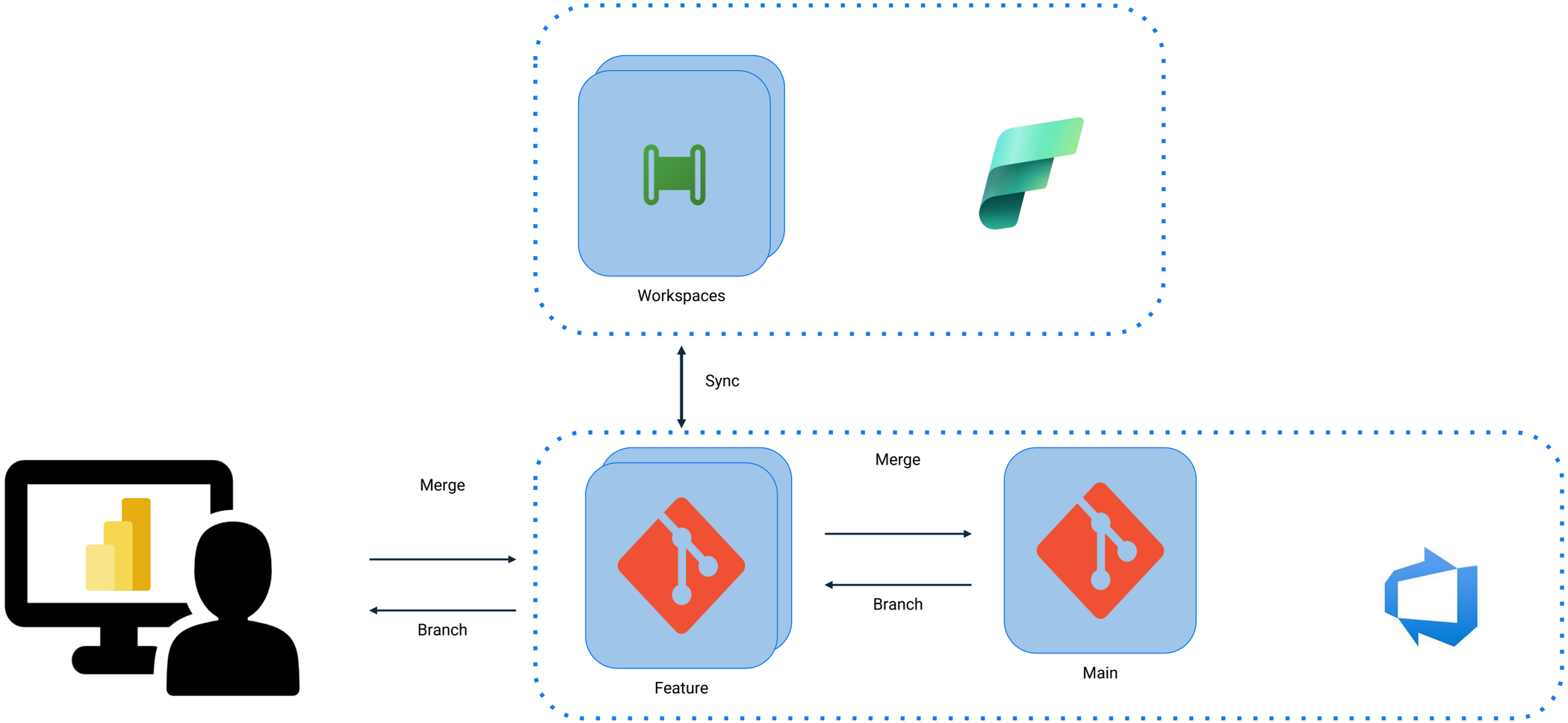
PIPELINES

- **Validate** Power BI artifacts (e.g., model syntax checks).
- Package **deployment artifacts**.
- Deploy reports/datasets via REST APIs or Power BI Deployment tasks.
- Manage Dev → Test → Prod promotions.
- Fabric: Provide UI-based staging and promote content between workspaces.

MICROSOFT FABRIC

- Provides **workspace** staging pipelines tightly integrated with deployment flows.
- Supports **deployment APIs** and automation hooks that Azure DevOps pipelines can call.
- Centralizes **governance, security, and environment promotion** for analytics assets.

The Official Happy Path

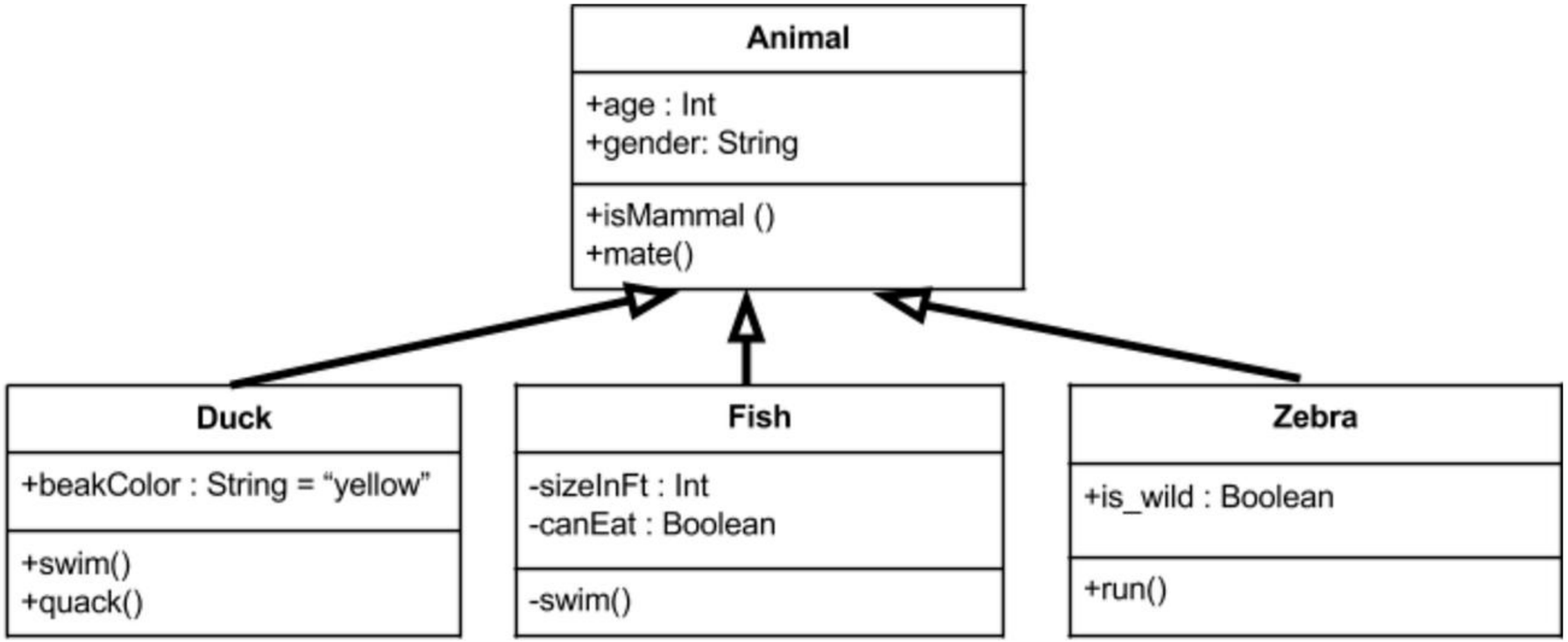


“Single Source of Truth”

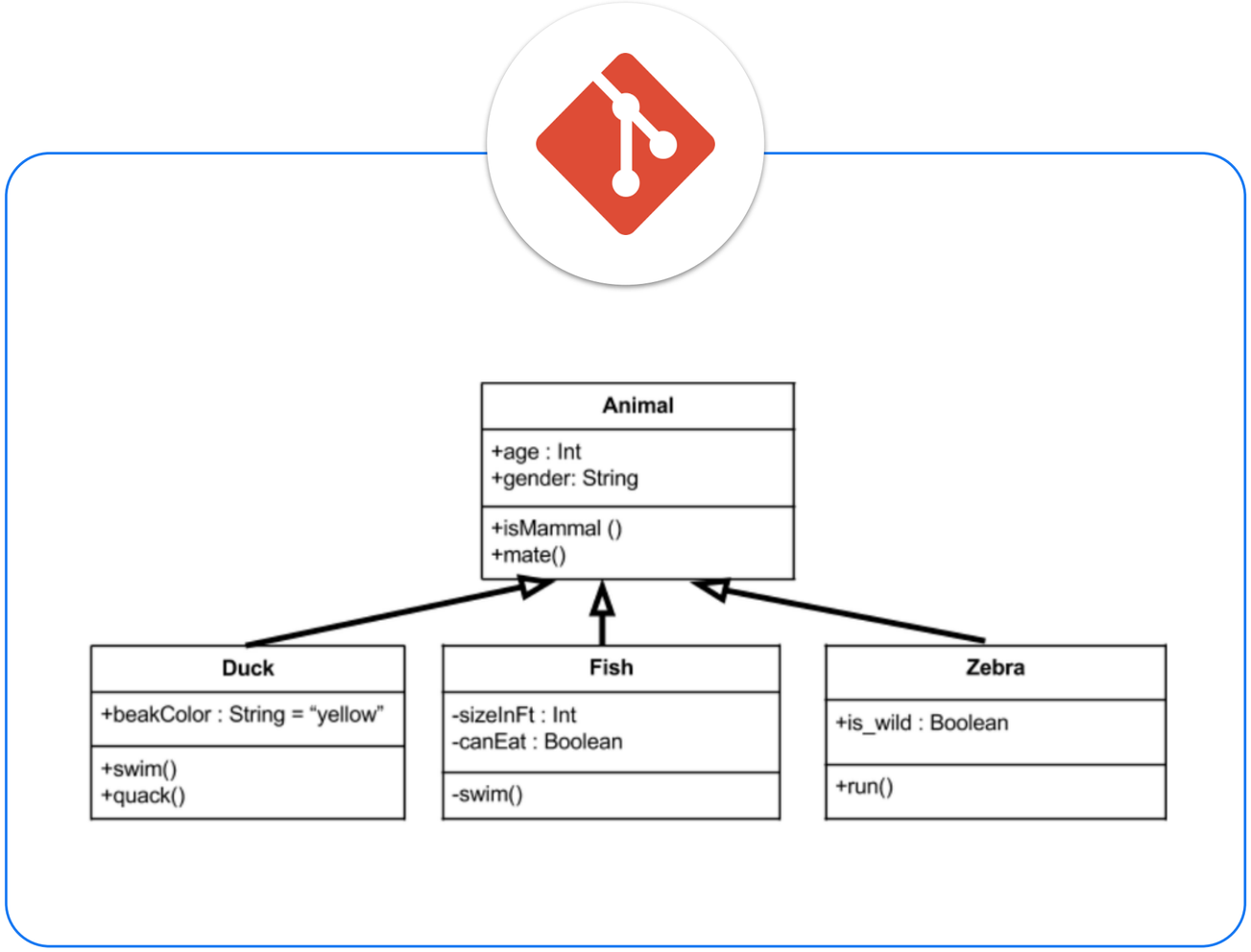
The practice of structuring information models and associated data schemas such that every data element is mastered (or edited) in only one place.

- Wikipedia

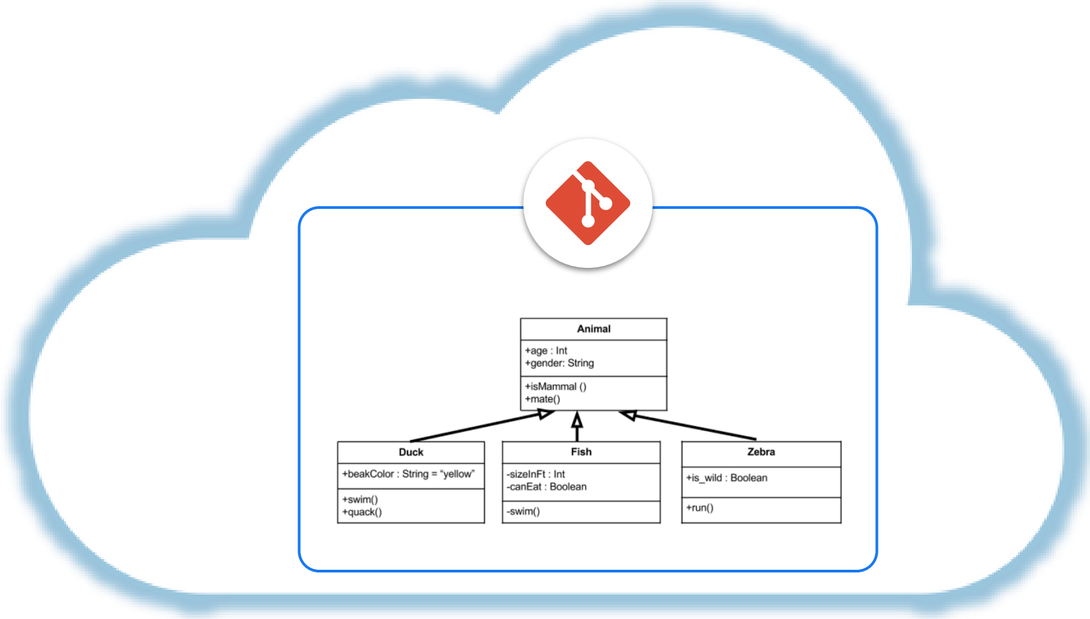
Single Source of Truth: Software



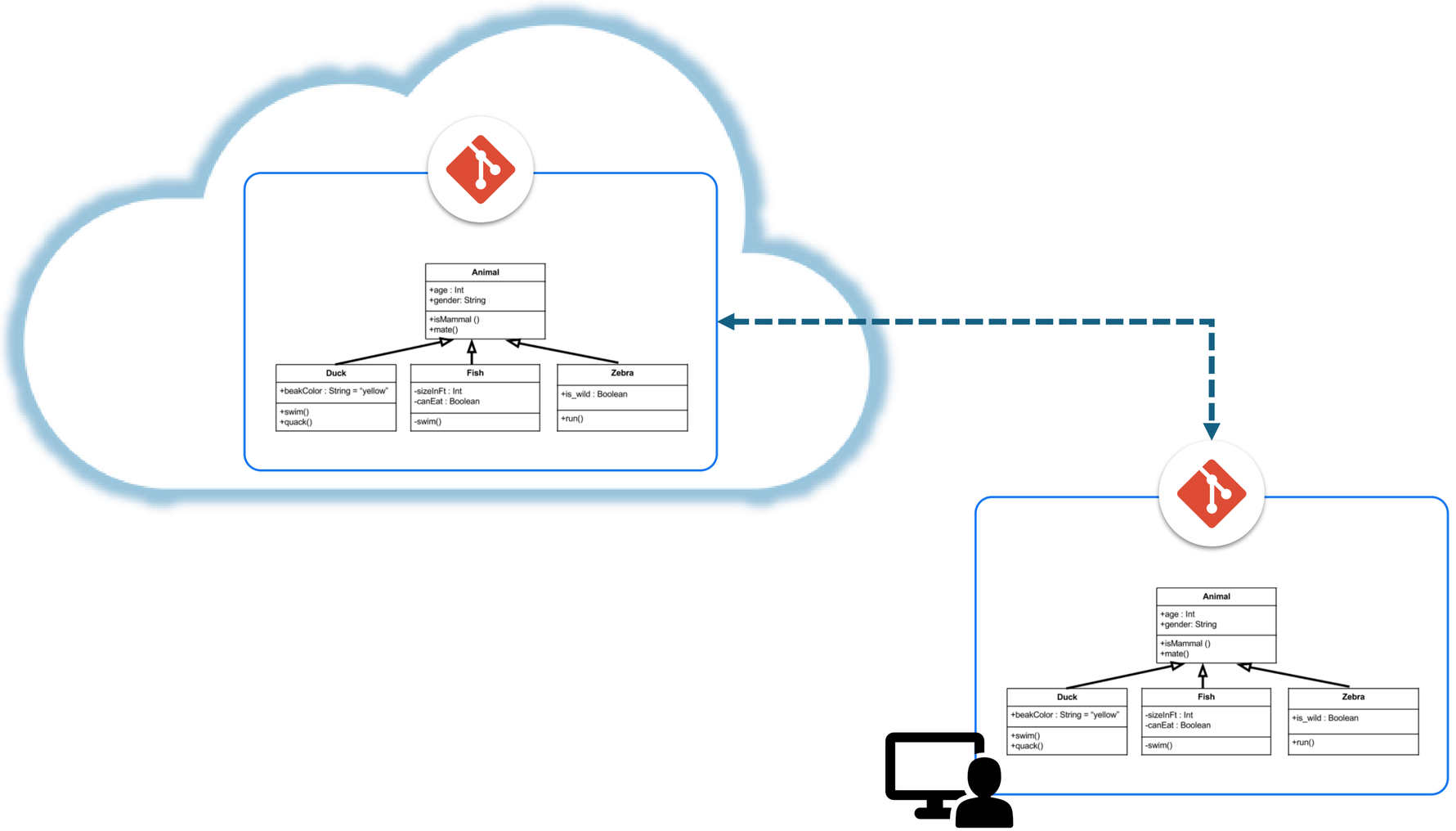
Single Source of Truth: Software



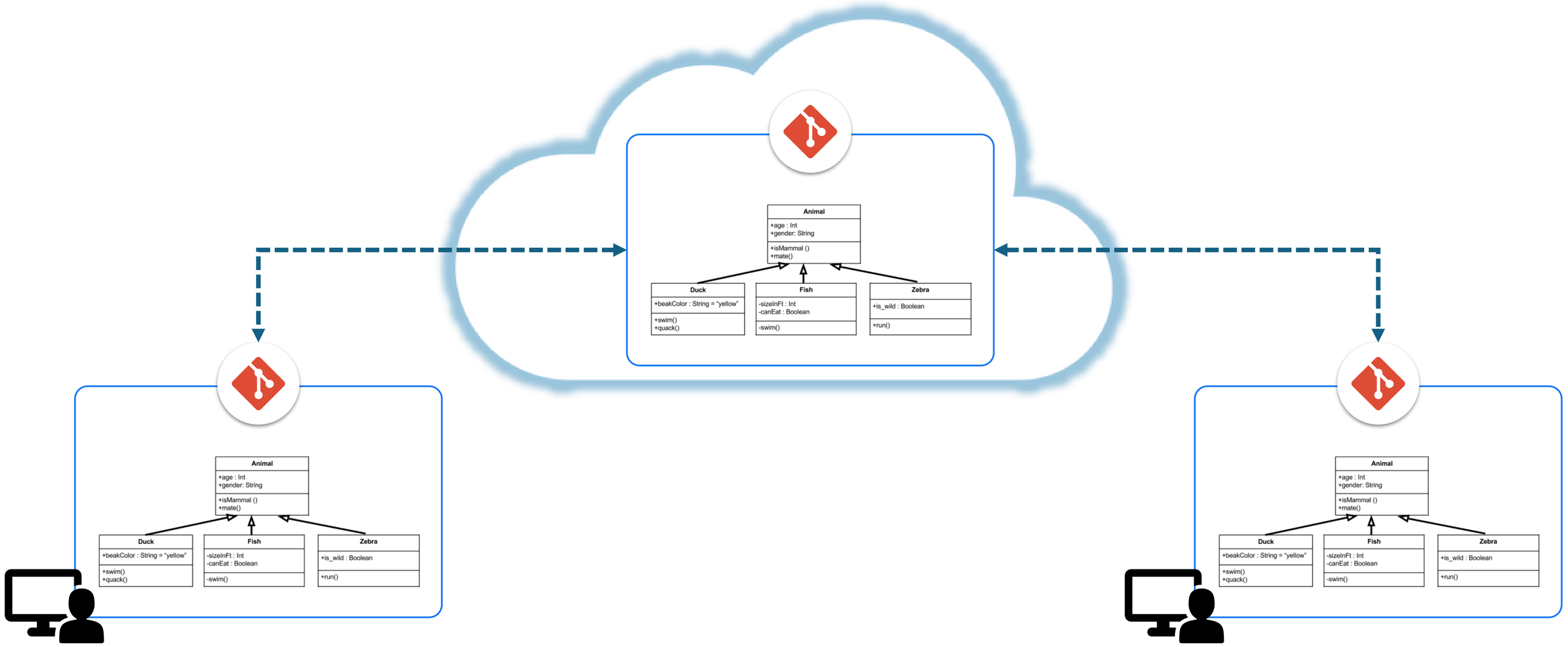
Single Source of Truth: Software



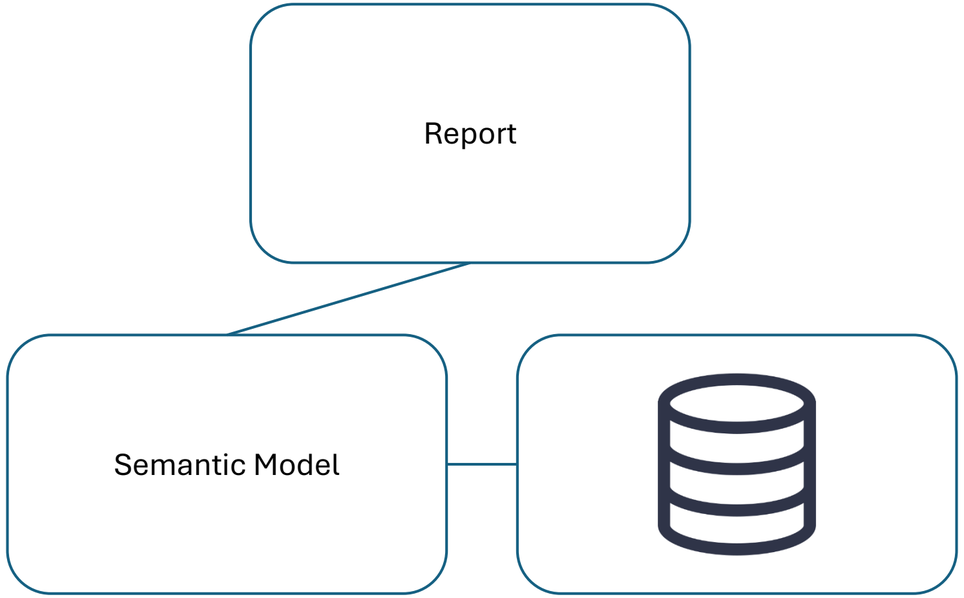
Single Source of Truth: Software



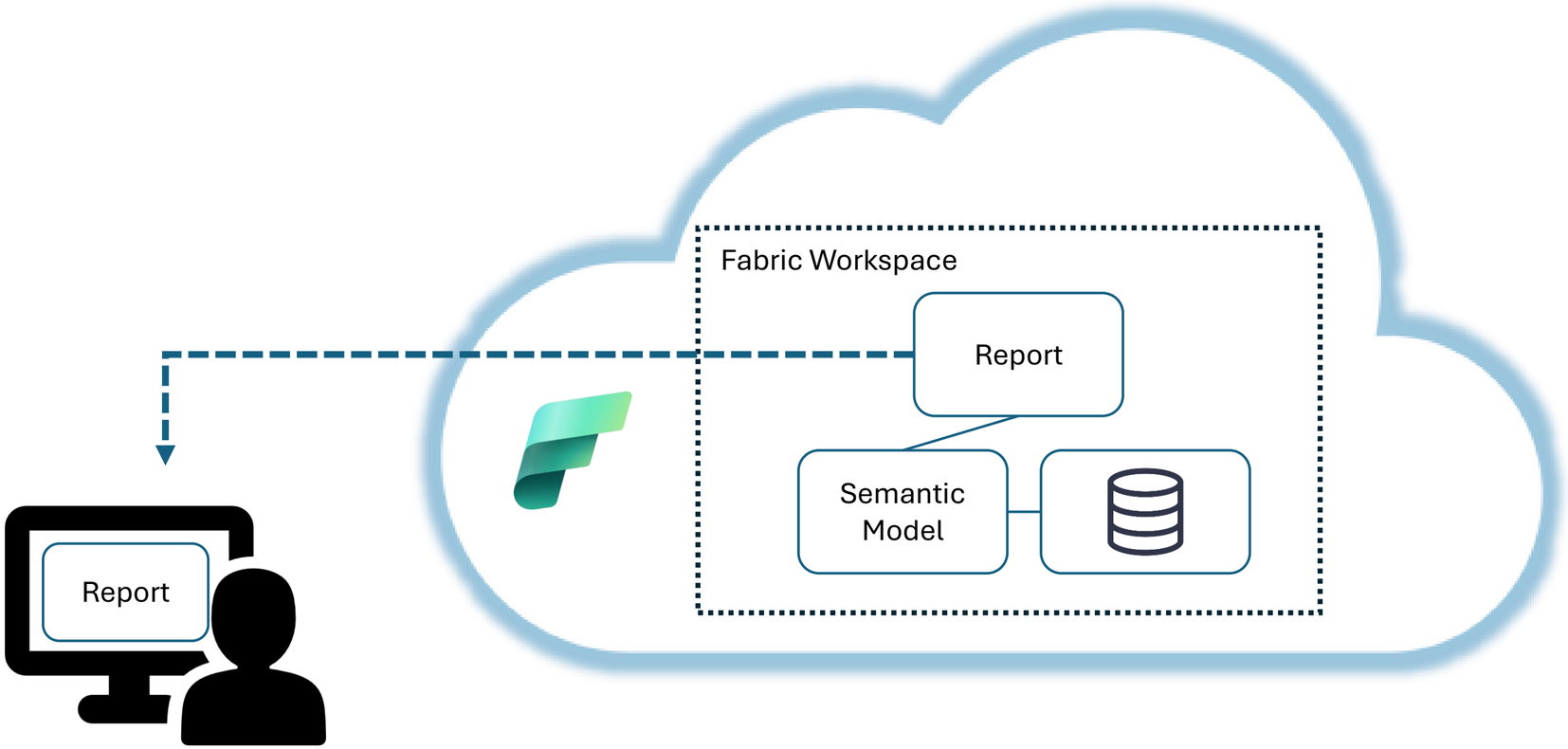
Single Source of Truth: Software



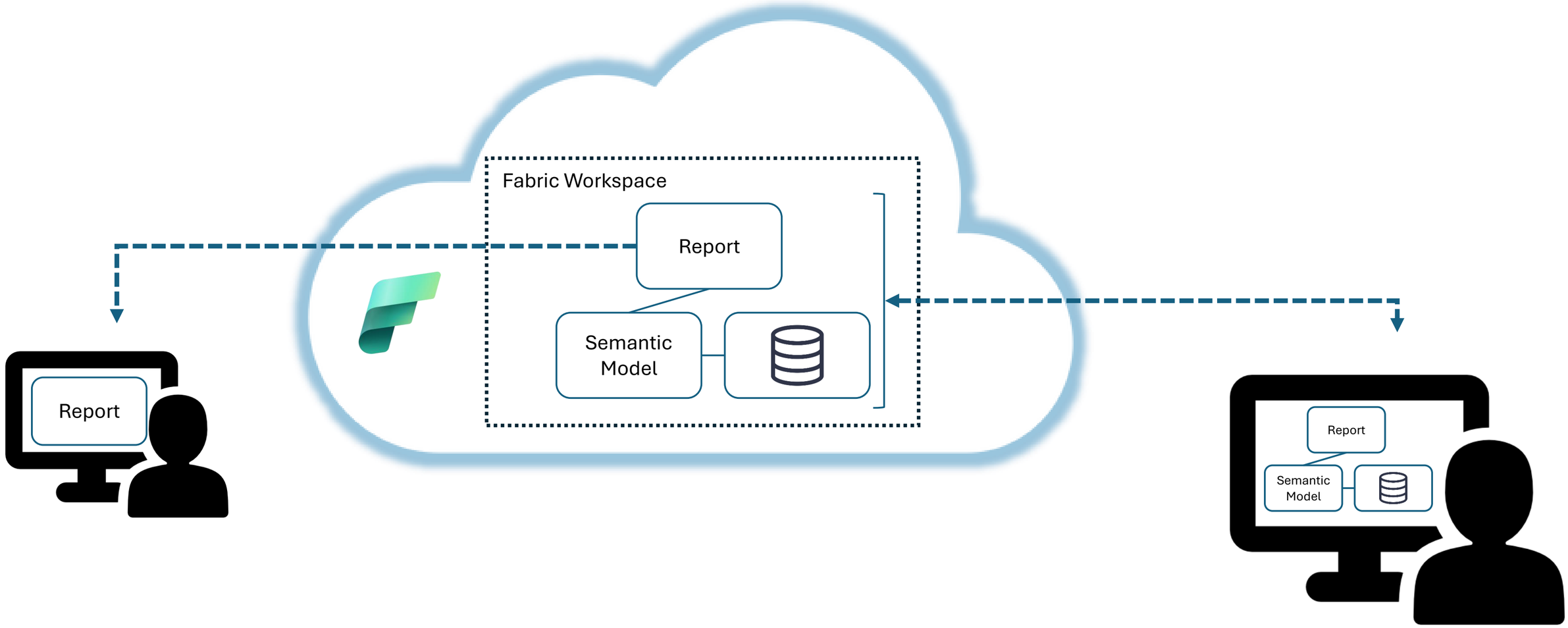
Single Source of Truth: Power BI



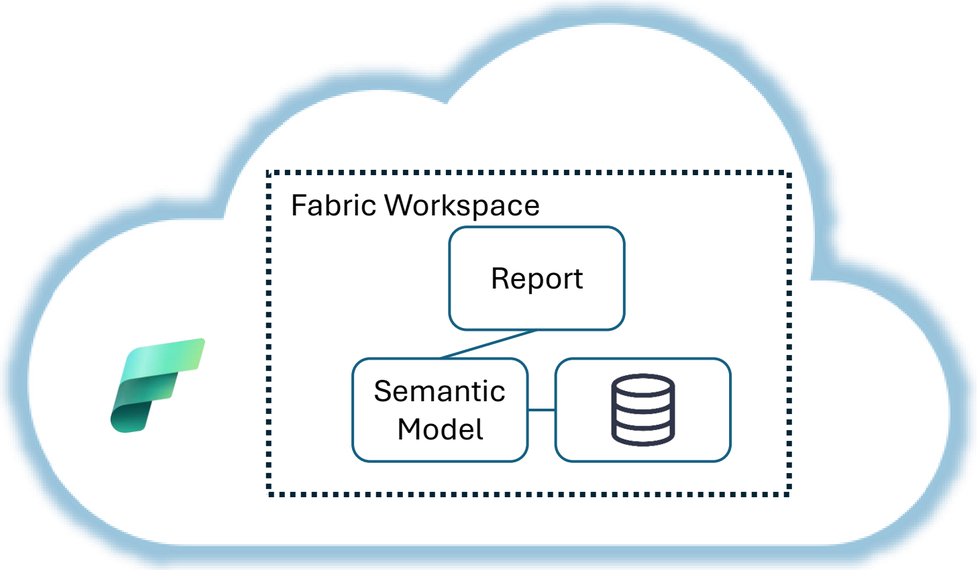
Single Source of Truth: Power BI



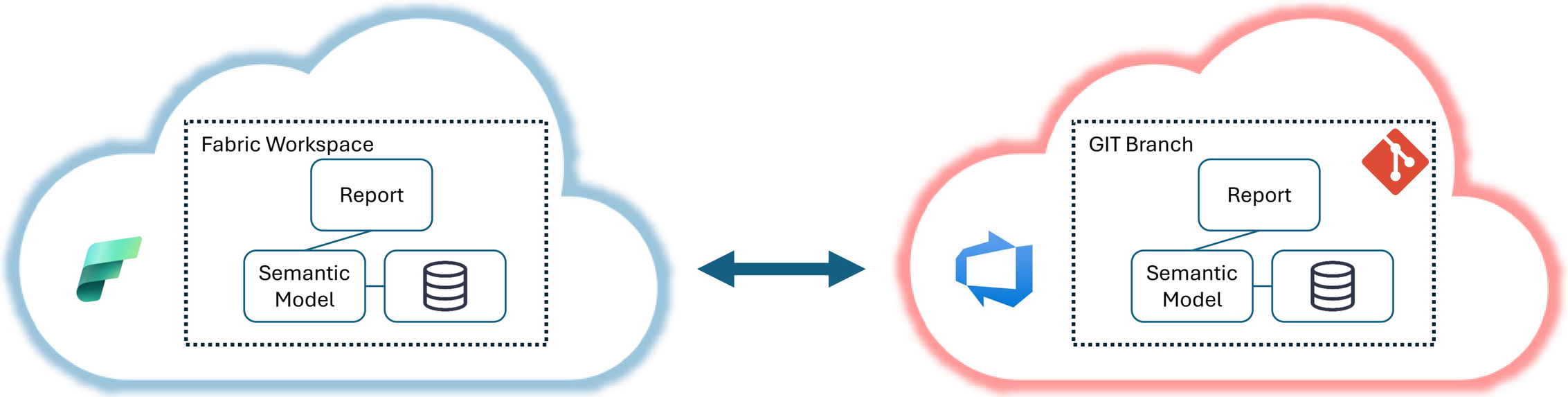
Single Source of Truth: Power BI



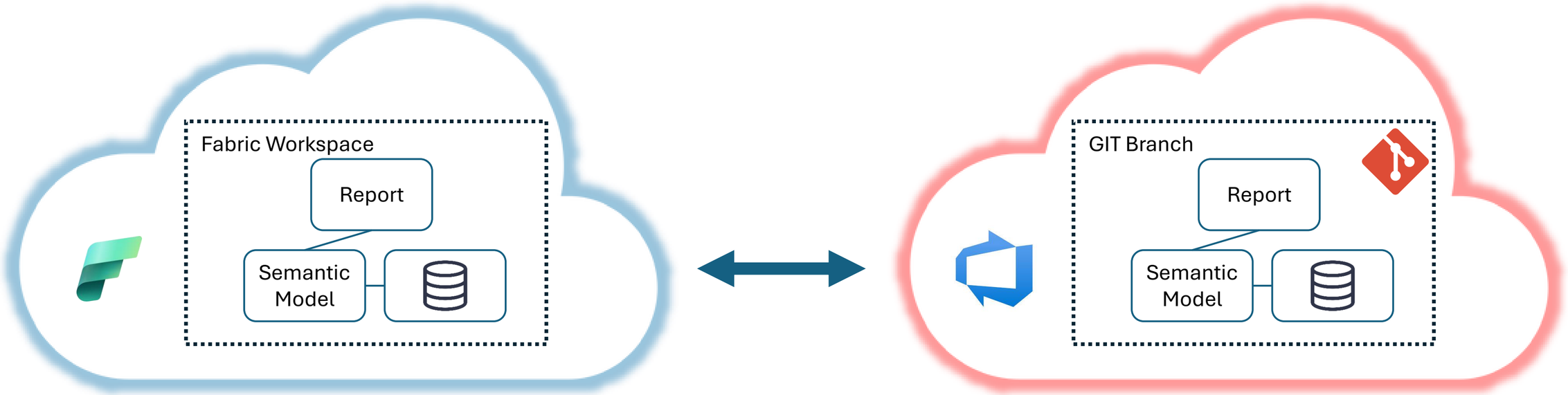
Single Source of Truth: Power BI



Single Source of Truth: Power BI

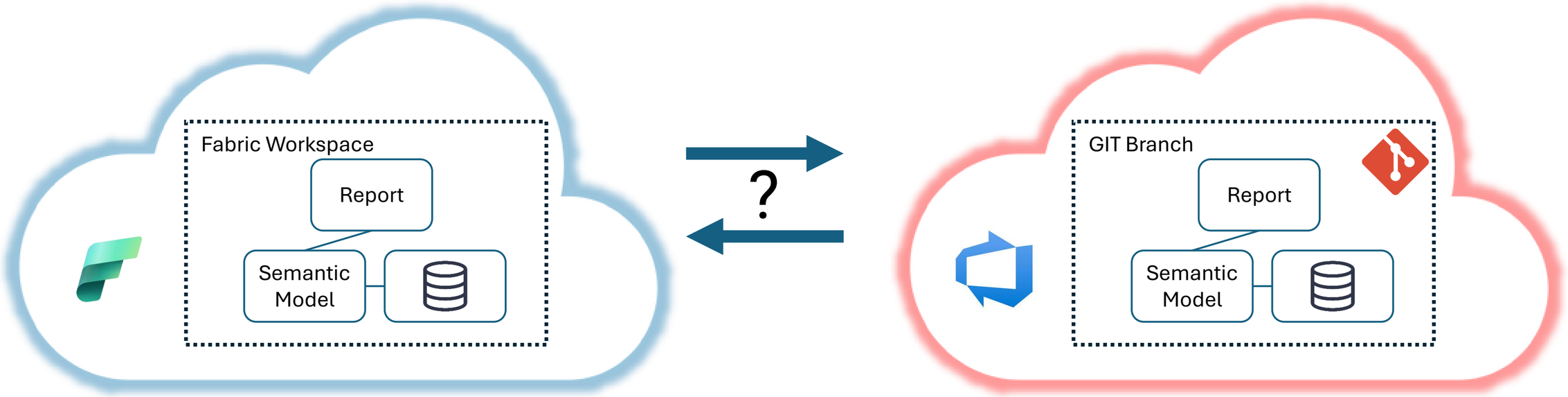


Single Source of Truth: Power BI



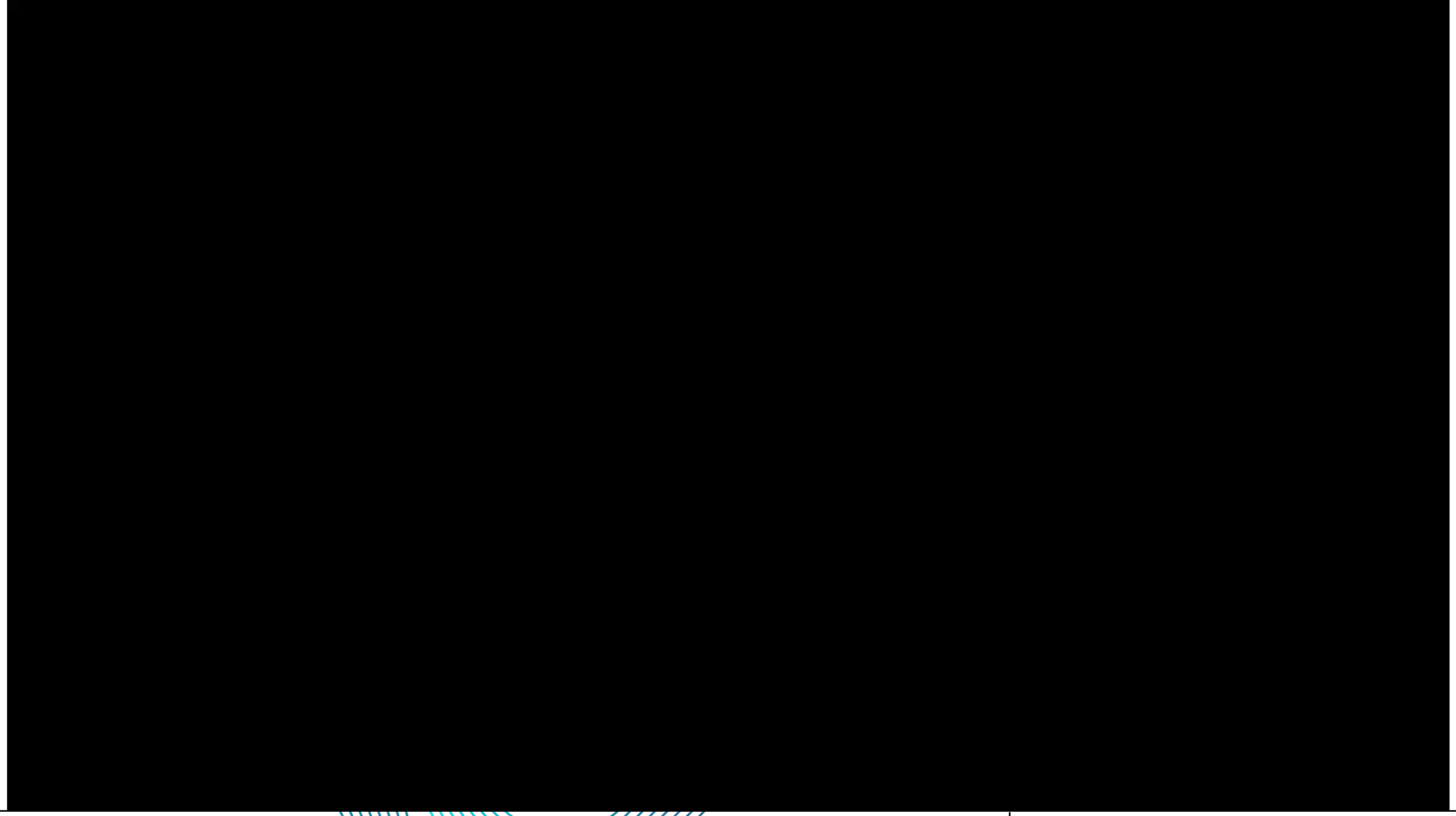
"*Single* Source of Truth?"

Single Source of Truth: Power BI



Single Source of Truth: Power BI

Demo



The Two Competing Schools of Thought

Workspace-First

Desktop → Publish → Workspace → Commit to Git

- Manual steps
- Risk of uncommitted changes
- Hotfixes in prod never making it back
- Drift between environments
- Git becomes a “backup,” not authority

Git-First

Desktop (.pbip) → Git → Workspace Sync

- Pull request governance
- Branch-based environments
- CI validation
- Rollback via commit history
- Automated deployment pipelines

Code-First Systems vs Power BI

Code-First

- Text Files
- Deterministic Builds
- Repo = Truth

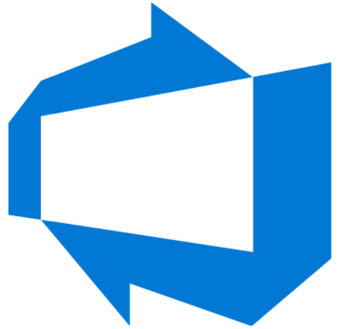
Power BI

- Visual + Metadata
- Service State
- Split Ownership of Truth

Tool #1: Git (What Teams Do)

“Just put it in Git”

- Current popular providers: GitHub and Azure DevOps
- Store PBIP (or legacy PBIX) artifacts
- Feature branches per report or model change
- Pull requests for review
- Inline diffs
- Pipelines triggered on PRs or merges (Azure DevOps)



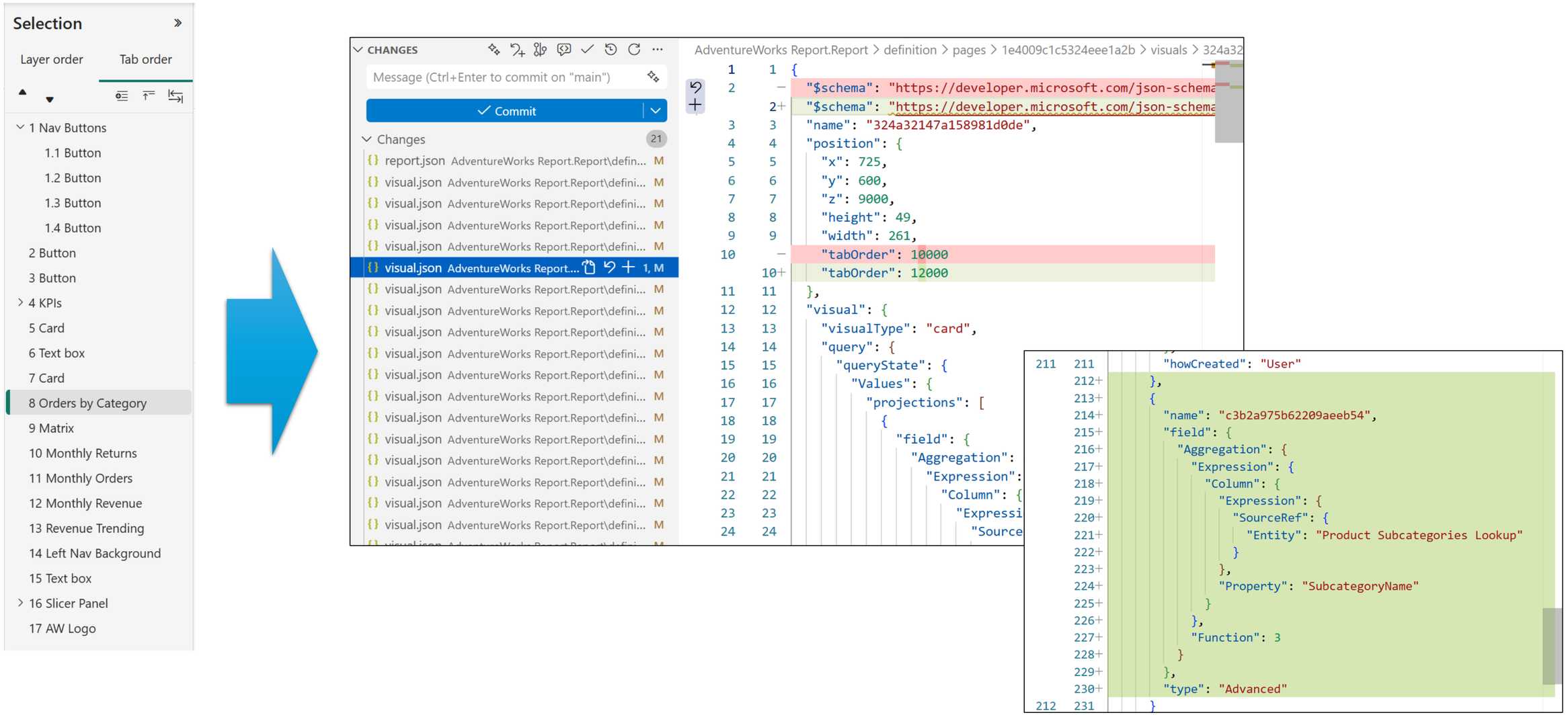
Azure DevOps

GitHub

Git Failure Modes (Beyond the Happy Path)

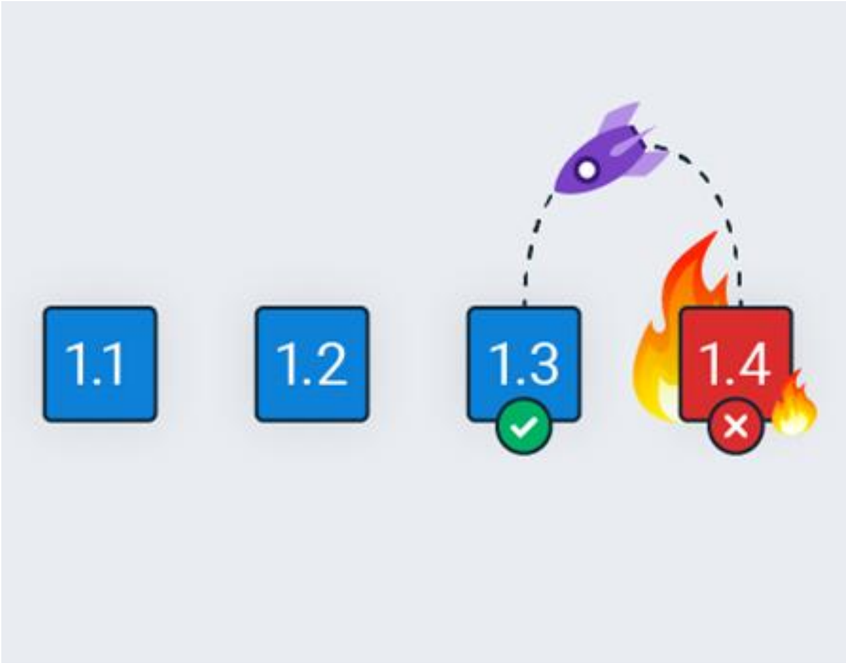
- PBIP JSON is **structurally mergeable**, but **semantically fragile**
- **Conflicts often “resolve cleanly” and still break models**
- UI edits can create massive, low-signal commits
- Binary PBIX diffs
- Repo ≠ source of truth
  a. PBIP partial truth
  b. Workspaces drift
  c. Hotfixes happen in Prod
  d. Deployment pipelines bypass Git
  e. Git becomes *a* source of truth, not *the* source
- False rollback confidence

Git Failure Modes (Beyond the Happy Path)



Why Rollback Rarely Works

- Pipelines don’t snapshot state
  - Dataset refresh state
  - Model cache state
  - Query plan behavior
- Data has already changed
  - Source data evolves
  - When you rollback the model logic may be old, the data is not
- Downstream dependencies are already mutated
  - Rolling back the semantic model does not roll back reports, etc.
- Manual fixes
- Credentials and security aren’t versioned

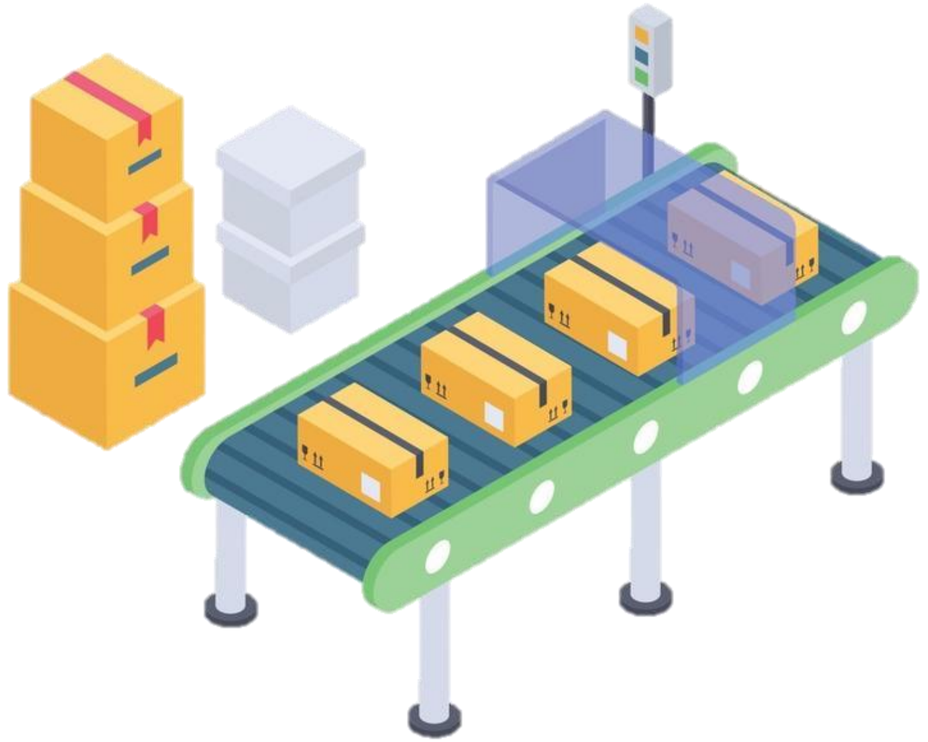


Tool #2: Deployment Pipelines (What They Promise)

- Moving artifacts between workspaces
- Preserving dataset ↔ report bindings
- Applying environment-specific rules
- Reduce manual republishing
- Preserve dataset/report bindings
- Simplify environment promotion
- Lower operational friction

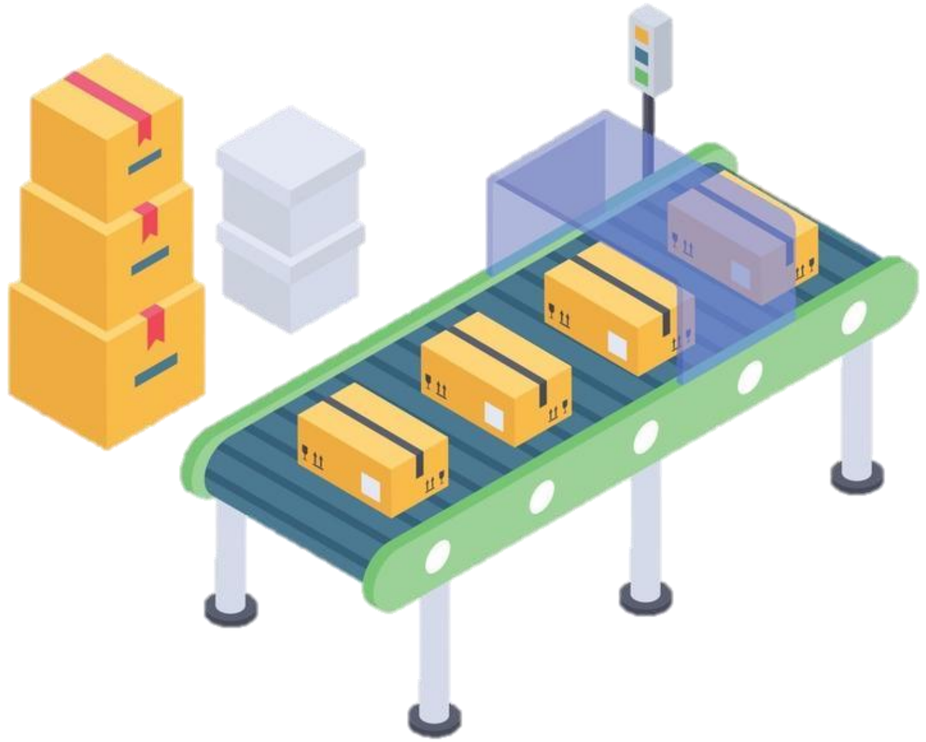
They are **excellent conveyor belts**.

They are **not safety systems**.



Deployment Pipeline Failure Modes

- Promotion without validation **(Broken models propagate *efficiently*)**
  a. Semantic correctness
  b. Measures
  c. Relationship evaluation
  d. Does it succeed on refresh?
  e. Queries execute successfully?
- Environment drift
  a. “Dev → Test → Prod stays in sync via promotion.”
  b. Hotfix in Prod at 2am
  c. Credential fix in Test only
  d. Parameter tweak directly in workspace
  e. Emergency report republish



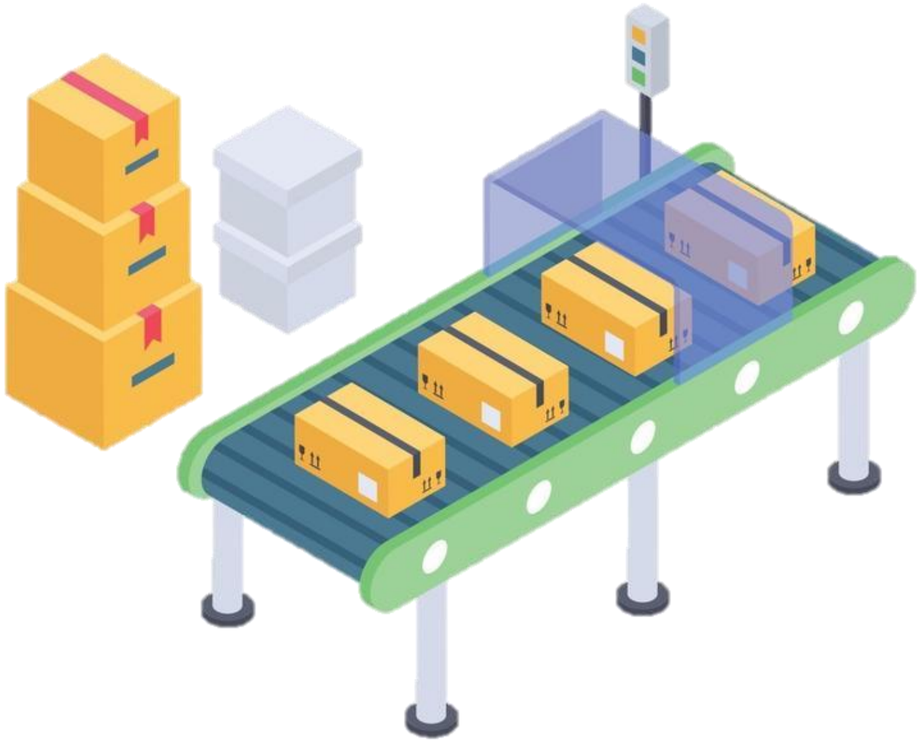
Deployment Pipeline Failure Modes

- No true rollback
  a. Deployment pipelines:
     i. Don’t version state
     ii. Don’t snapshot dependencies
     iii. Don’t restore data
     iv. Don’t undo side effects

By the time you “re-deploy”:

- Data has refreshed
- Incremental partitions changed
- Schema drift already occurred
- Downstream reports adapted (or broke)

Service-only state leaks



“Deployment Pipelines move artifacts, not correctness.”

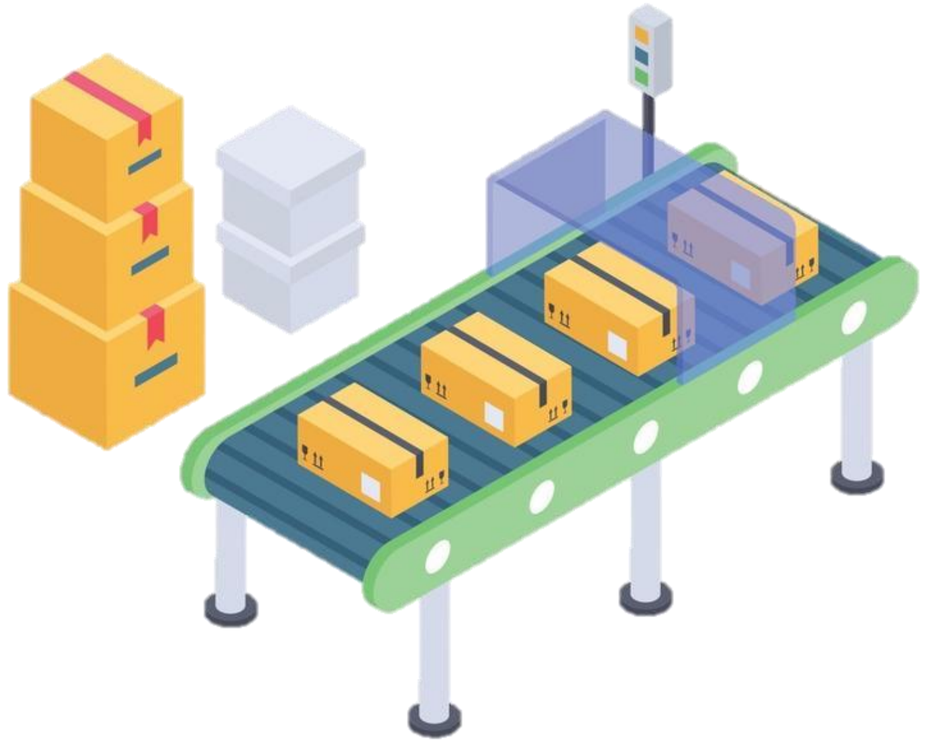
Deployment Pipeline: How mature teams compensate

They add **guardrails outside the pipeline**:

- Semantic validation *before* promotion
- Automated refresh tests
- Model diffing
- Workspace drift detection
- Git as *reference*, not authority

The pipeline becomes:

- The last mile, not the quality gate.

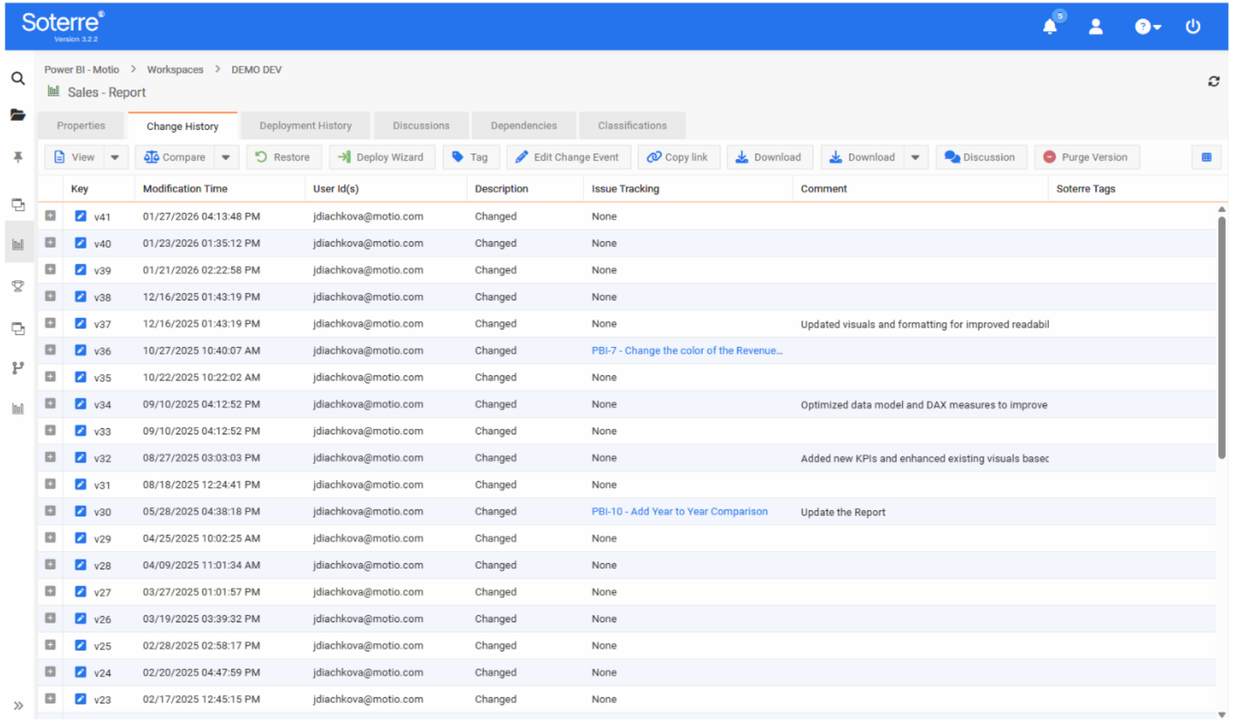


What Actually Works: Reframing CI/CD

Automated Guardrails

Zero-Touch Version Control

Single way of Deployment



Pattern 1: Automated Guardrails

- Auto schema drift detection
  - Medallion approach: silver layer health checks
  - Automated checks in Azure DevOps pipelines

Bringing DataOps to Power BI

Pattern 1: Automated Guardrails

- Data Quality Validation using Notebooks:
  - Utilize Power BI REST API to access measures dynamically
  - Connect to a SQL Server database for real-time validation
  - Create a logging table to document differences automatically
- Creates clear audit trail for every data quality check

Mastering Data Quality Validation in Microsoft Fabric for Power BI Reports
By Kristyna Ferris

Pattern 1: Automated Guardrails

Validate your relationships

- Referential integrity violations
  - Missing values on the “from” side that are present on the “to” side.
  - How we get “blanks”.
- Unexpected query results
  - Produce results that differ from the expected baseline
- Slow query evaluation times
  - High cardinality
  - Also use the Performance Analyzer

Use Tabular Editor scripts for automated validation

Validate semantic model relationships – Tabular Editor
By Kurt Buhler

Pattern 1: Automated Guardrails

Contextual Data Quality Checks using AI

- Create a template prompt
- Use Python notebook to execute
- Analyze the output

Unstructured To Structured : Using Fabric AI Functions For Contextual Data Quality Check By Sandeep Pawar

Pattern 2: Zero-Touch Version Control

Aim for:

- Automated Commits:
  - Changes are automatically saved and versioned without manual “commit” actions.
- No Manual Edits:
  - Achieve “immutable” releases without “hotfixes” or manual edits in prod.

Result:

- Easy Rollbacks:
  - Users can revert to previous versions, undoing mistakes confidently.
- Simplified DevOps:
  - It reduces the confusion around the state of the environment.
- Focus on Development:
  - It lets developers focus on building rather than administration.

Soterre for Power BI

Pattern 3: One Deployment for All

Pick one method for deployment and stick to it.

- When you use multiple methods, your "source of truth" becomes fragmented
- Managing a DevOps lifecycle requires a specific skill set
- Security is much harder to enforce when there are multiple "back doors" into your production environment

A Practical Evaluation Framework

Five questions:

- What is our source of truth?
- How do we detect drift?
- How do we block bad releases?
- How do we explain history?
- How are we preventing “quick fixes”?

Final Word

The goal isn’t faster deployments.
The goal is trusted analytics.

Short Survey…

What is **ONE** thing you took away from this session?

One person will be randomly selected to win a $50 Amazon gift card!

Sound off.
The mic is all yours.
Influence the product roadmap.

Join the Fabric User Panel

Share your feedback directly with our Fabric product group and researchers.

https://aka.ms/JoinFabricUserPanel

Join the SQL User Panel

Influence our SQL roadmap and ensure it meets your real-life needs

https://aka.ms/JoinSQLUserPanel

How was the session?

Complete Session Surveys in Whova for your chance to WIN PRIZES!