#FABCONSQLCON2026

FABCON
Microsoft Fabric
COMMUNITY CONFERENCE

SQLCON
Microsoft SQL
COMMUNITY CONFERENCE

ATLANTA MARCH 16 20, 2026

AI-Powered CI/CD for Microsoft Fabric: Building Enterprise-Grade Pipelines with GitHub Copilot

Eugene Paden
CTO – Ray Business Technologies

About Me + About You

Who Am I?

- Helped teams get their Fabric CI/CD working without losing their minds

About You (probably)

- You work with Fabric (or thinking about it)
- You've copy-pasted notebooks between workspaces and hated it
- You know Git exists (maybe even used it once or twice)
- Someone said "DevOps" and you're here to figure out what that means

You'll Get the Most Out of This If

- You've played around in Fabric workspaces (notebooks, lakehouses)
- You've heard of CI/CD (even if you've never set it up)
- You're comfortable clicking around Azure DevOps or GitHub

Don't Panic If You Haven't We'll explain the important stuff as we go. The demos will make it clear.

Not on Today's Menu (so you're not waiting for it):

- Fabric licensing decisions or capacity planning
- Deep-diving into Spark performance tuning
- Teaching you Python, Scala, or SQL from scratch
- Fixing your Git merge conflicts (sorry!)

What You'll Actually Walk Away With

The Good Stuff You Can Use Right Away

- Azure Pipelines CI/CD template
- Artifact validators you can drop into your project today
- Git branch strategy that won't make you cry at 2am
- Github Copilot Skills – Create PR, Review PR

- **BONUS** 1-hour free consultation we'll look at your setup and tell you what's going to break

This is stuff you can implement Monday morning without asking for budget approval

How This Actually Works

What We're Building Today

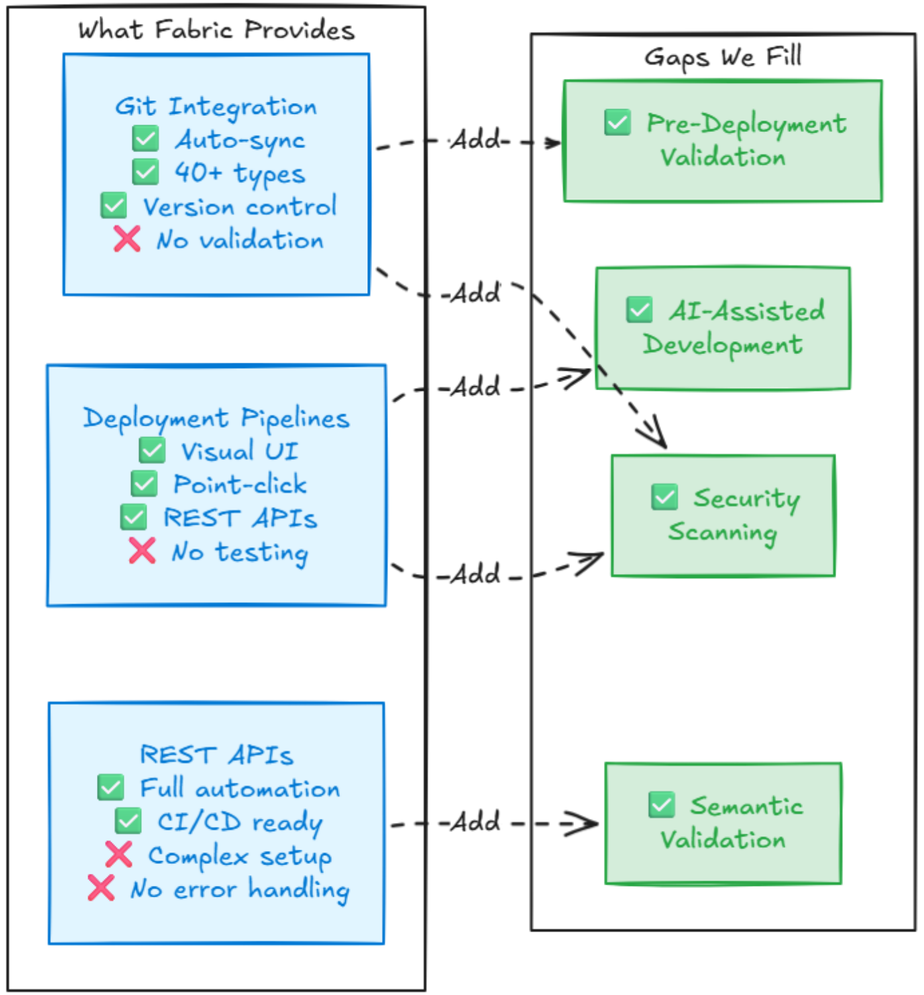
- Real-world patterns from actual Fabric projects (not theory)
- 5-dimension quality framework (sounds fancy, saves your bacon)
- Let GitHub Copilot write your CI/CD
- Deploy without downtime (yes, really)

Fabric Already Gives You Some Good Stuff

- Git Integration: 40+ item types, built-in auto-sync
- Deployment Pipelines: Visual UI, item pairing
- REST API Suite: Full automation capabilities

We Add Enterprise CI/CD Patterns

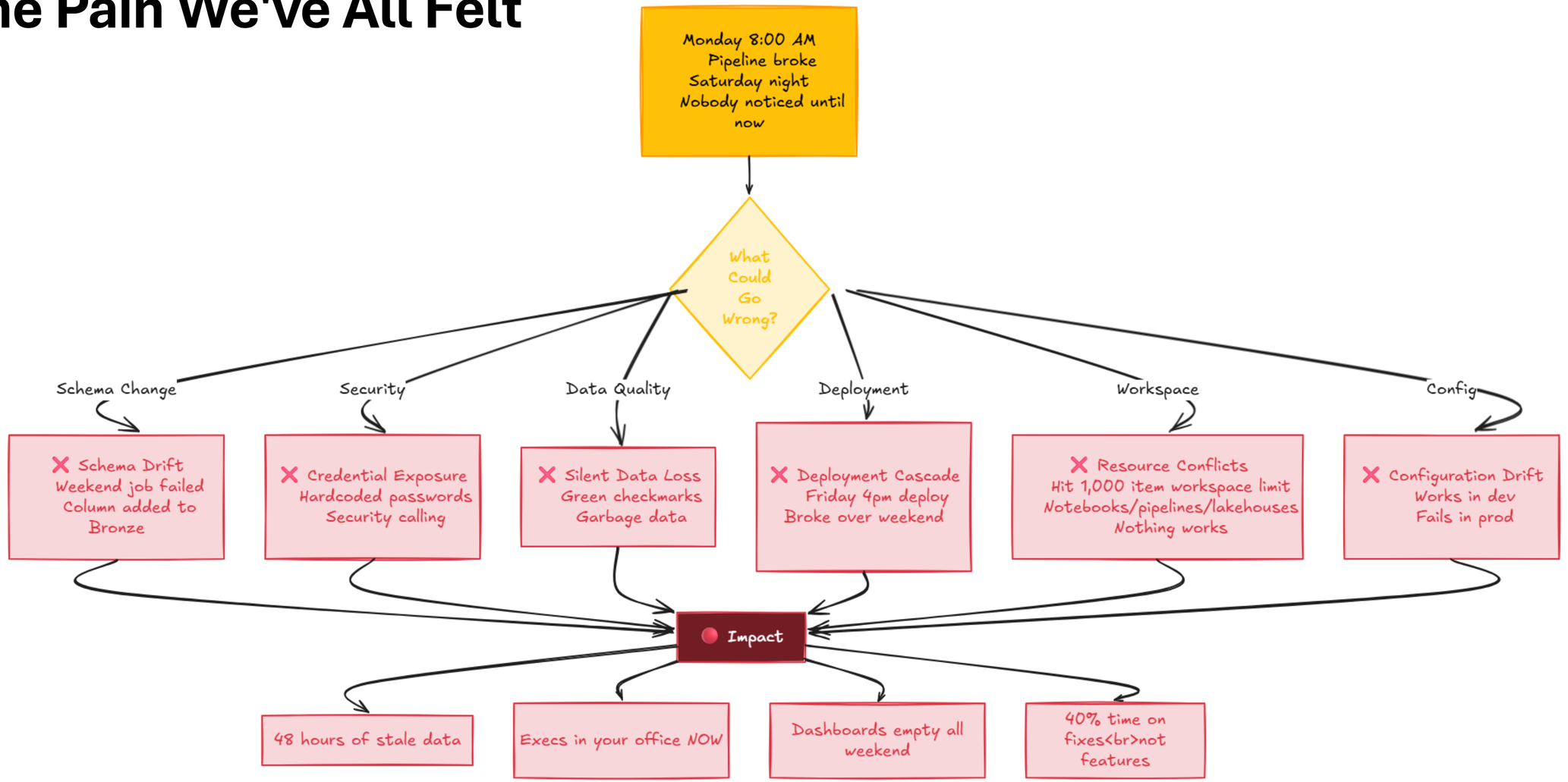
- Pre-deployment validation (schema, security, quality gates)
- AI-assisted development (GitHub Copilot integration)
- Zero-downtime deployments (Git branch orchestration)
- Rollback procedures



Partnership Approach: Build on Microsoft's platform with proven automation patterns

Session Focus: Practical patterns you can implement immediately

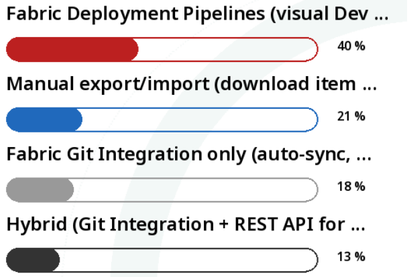
The Pain We've All Felt



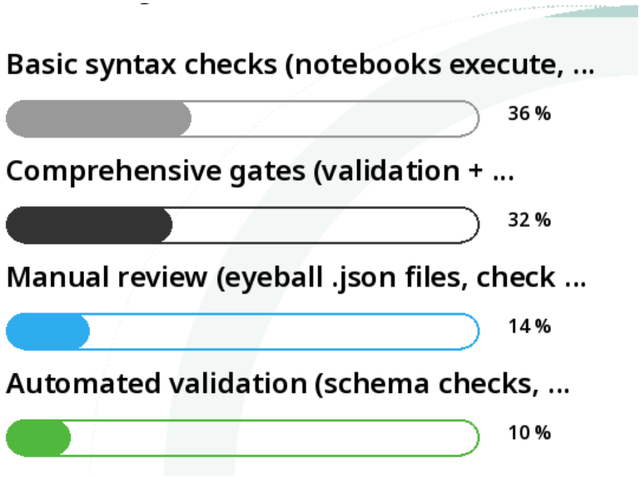
The Good News: "We've made all these mistakes so you don't have to. Seriously, we've broken production in creative ways you haven't even thought of yet. Let's chat after and I'll tell you war stories."

Survey Says....

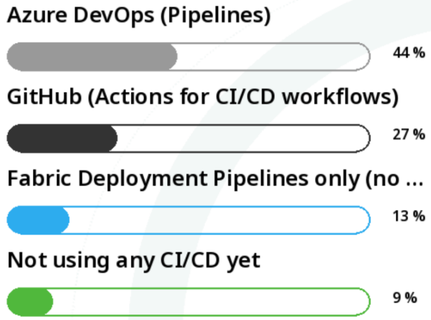
Poll question
Which Fabric deployment approach do you use TODAY?



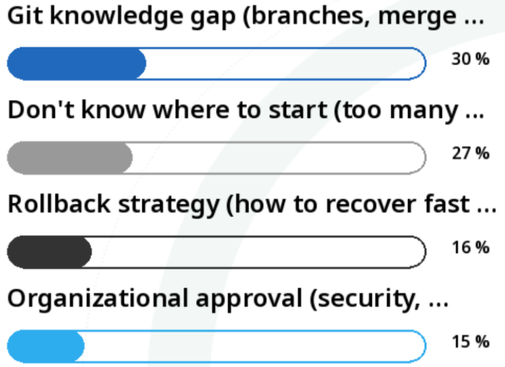
Poll question
How do you currently validate artifacts BEFORE production deployment?



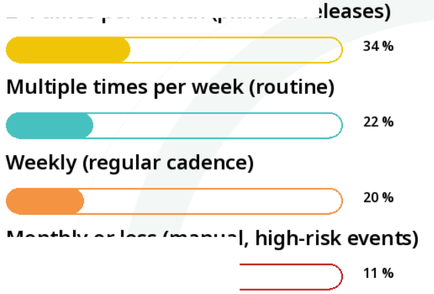
Poll question
Which CI/CD platform are you using (or planning to use)?



Poll question
What's your PRIMARY blocker for implementing Fabric CI/CD?



Poll question
How often do you deploy Fabric artifacts to production?



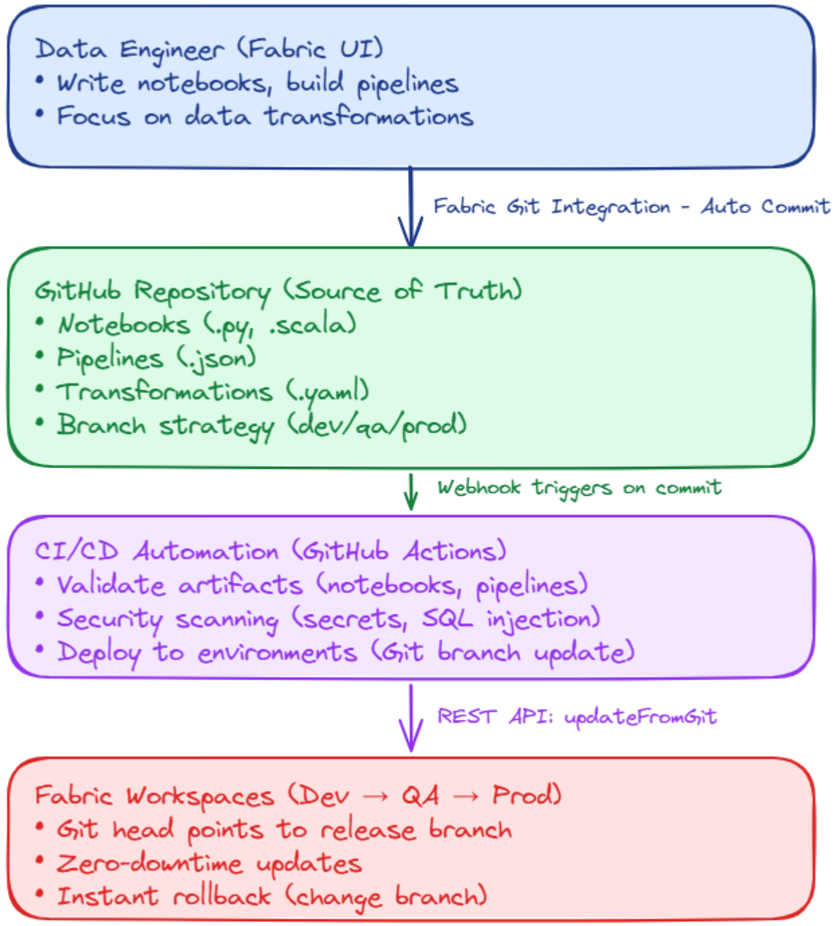
How We're Fixing This

AI-Assisted Development (eliminates the learning curve)

- Copilot generates CI/CD workflows from plain English
- No need to memorize Git commands or YAML syntax
- You focus on data, AI handles automation

Data-Centric CI/CD Patterns (adapted for data workflows)

- Schema validation against DBML (catch breaking changes early)
- Security scanning (credentials, SQL injection)
- Git branch strategy for zero-downtime deployments
- Artifact-specific validators (notebooks, pipelines, transformations)



"Data engineers stay productive with DevOps patterns without needing to become DevOps engineers."

Quick Context What's Fabric Data Transformer?

The Framework We're Using in These Demos

FDT in 60 Seconds

- **Scala library for spark –** reads YAML transformation configurations and executes them
- **YAML defines transformations –** FDT runtime does the heavy lifting
- **30 pre-built transformers** All the usual suspects Filter, Join, Merge, Aggregate, Validate, Union, Pivot, Unpivot…
- **Data Quality Built-In** Pre-write validation rules (34), post-write integrity checks (10), scheduled data audits (14)
- **Enterprise-grade** Built-in validation, performance tuning, security stuff, and we have license tiers (gotta pay the bills)

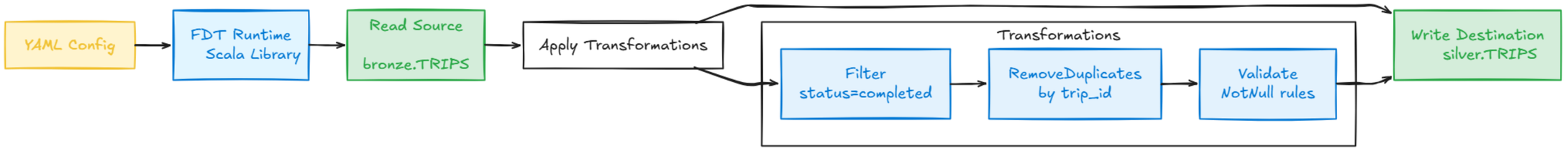
Why This Matters for CI/CD

- **YAML = Code** Version control, review, validate transformations like code
- **Schema Validation** CI pipeline validates YAML against DBML schemas
- **Security Scanning** CI detects SQL injection in filter expressions
- **Data Quality Gates** Pre-write validation rules catch issues before write, post-write checks confirm correctness
- **Consistent Patterns** All transformations follow same structure

For This Session We're focusing on the ****CI/CD automation****, not the framework itself (that's a different talk)

How It Works

```yaml
# YAML Configuration (What You Write)
source:
  tableName: "bronze.TRIPS"
  format: "delta"
  filter: "status = 'completed'"
transformations:
  type: RemoveDuplicates
    columns: ["trip_id"]
  type: Validate
    mode: "failOnError"
    rules:
      ruleType: NotNull
        name: "trip_id_not_null"
        description: "Trip ID and pickup time must exist"
        columnNames: ["trip_id", "pickup_datetime"]
        severity: "ERROR"
destination:
  tableName: "silver.TRIPS"
  format: "delta"
  writeMode: "overwrite"
```

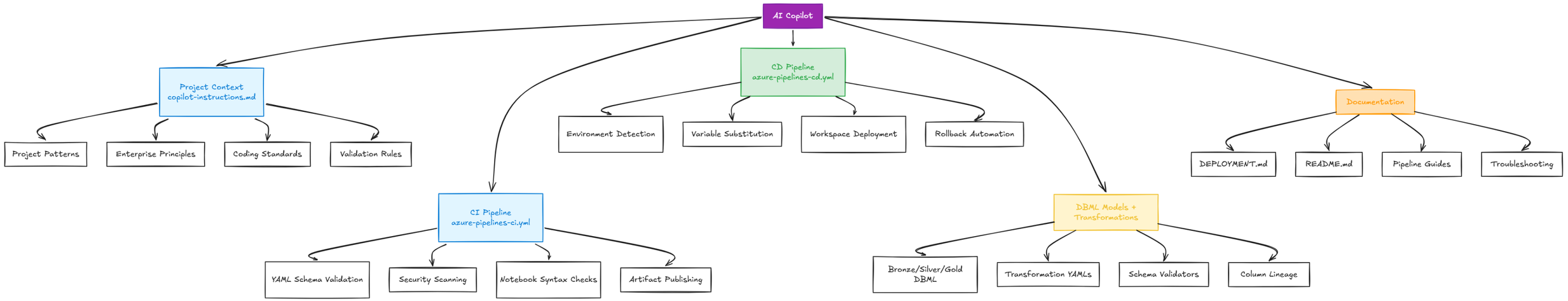


Let AI Build Your Pipeline

AI Writes All Your CI/CD Code (Seriously)

What You Still Do Manually (in Fabric Workspace UI the point-and-click stuff)

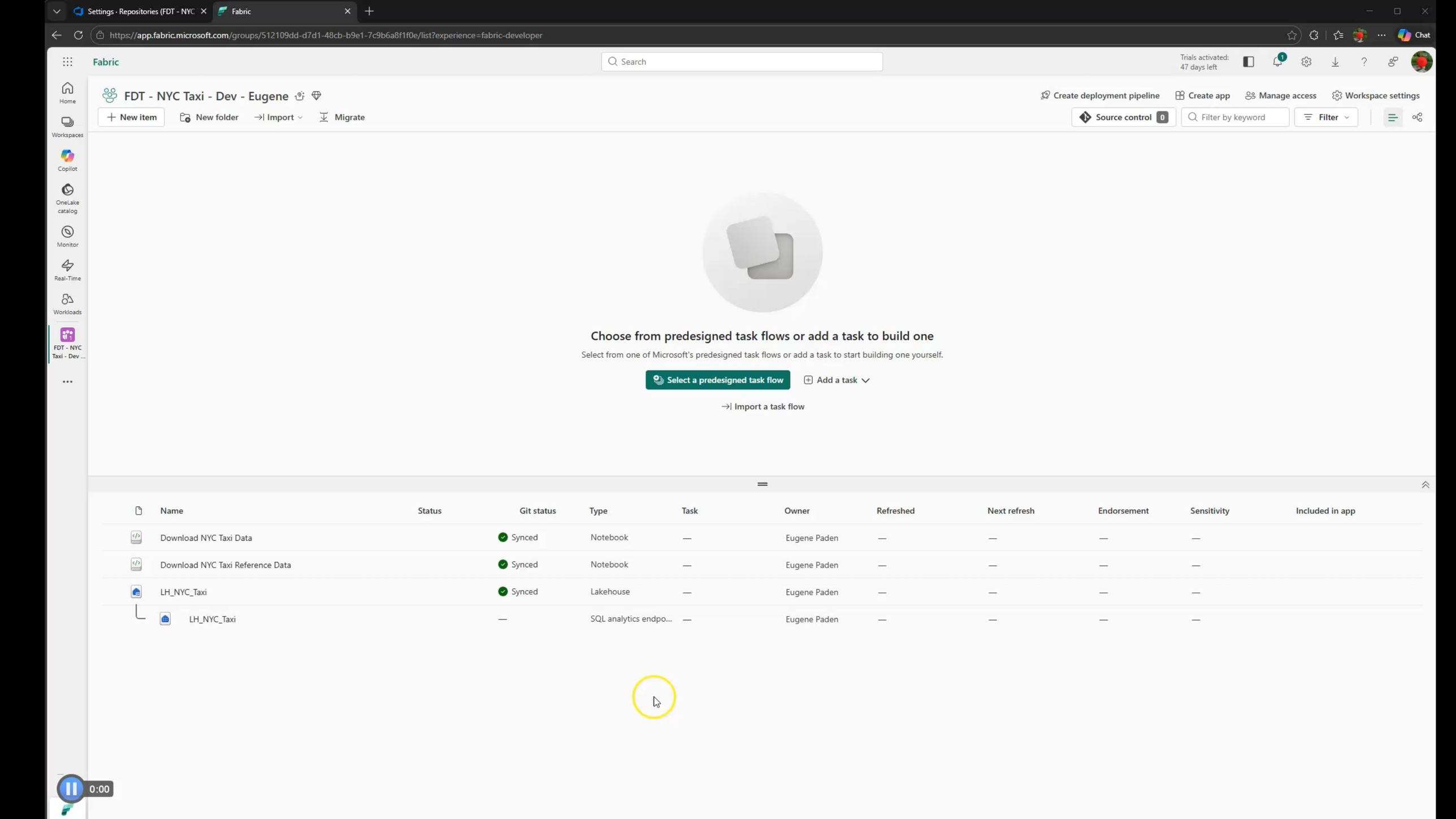
- Create lakehouses (LH_Control, LH_Bronze, LH_Silver, LH_Gold)
- Create warehouse (WH_Control)
- Create notebooks (your actual data pipelines)
- Create pipelines (orchestration stuff)
- Configure variable library
- Set up Git integration



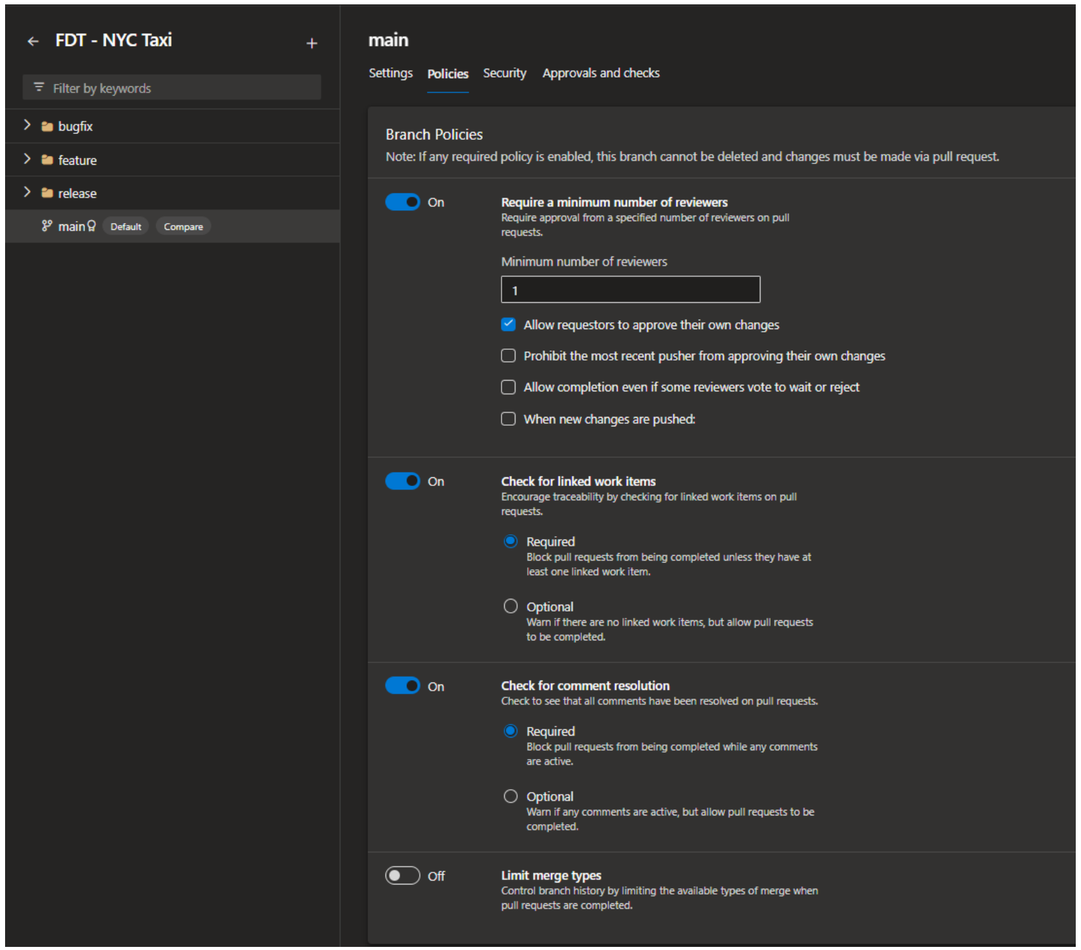
AI builds automation AROUND your manually-created Fabric stuff (it's not replacing your workspace, just making it safer)

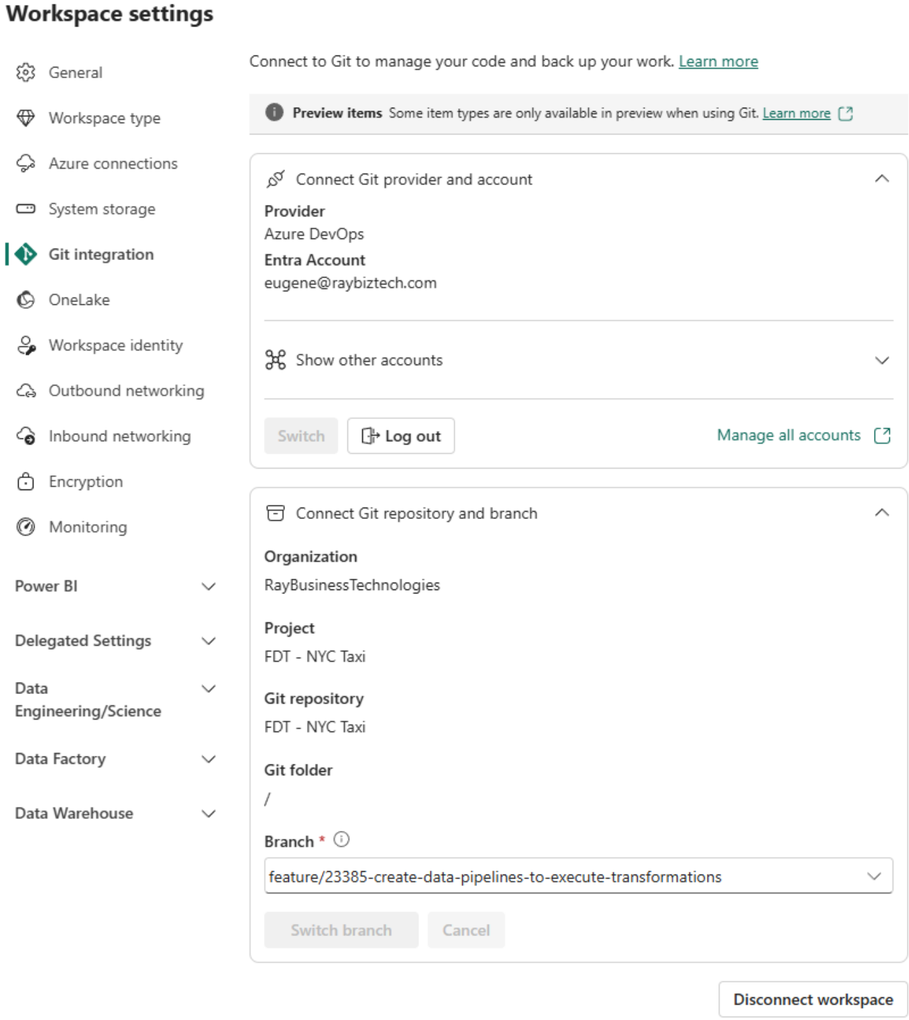
DEMO 1 – Setup CI and Validators

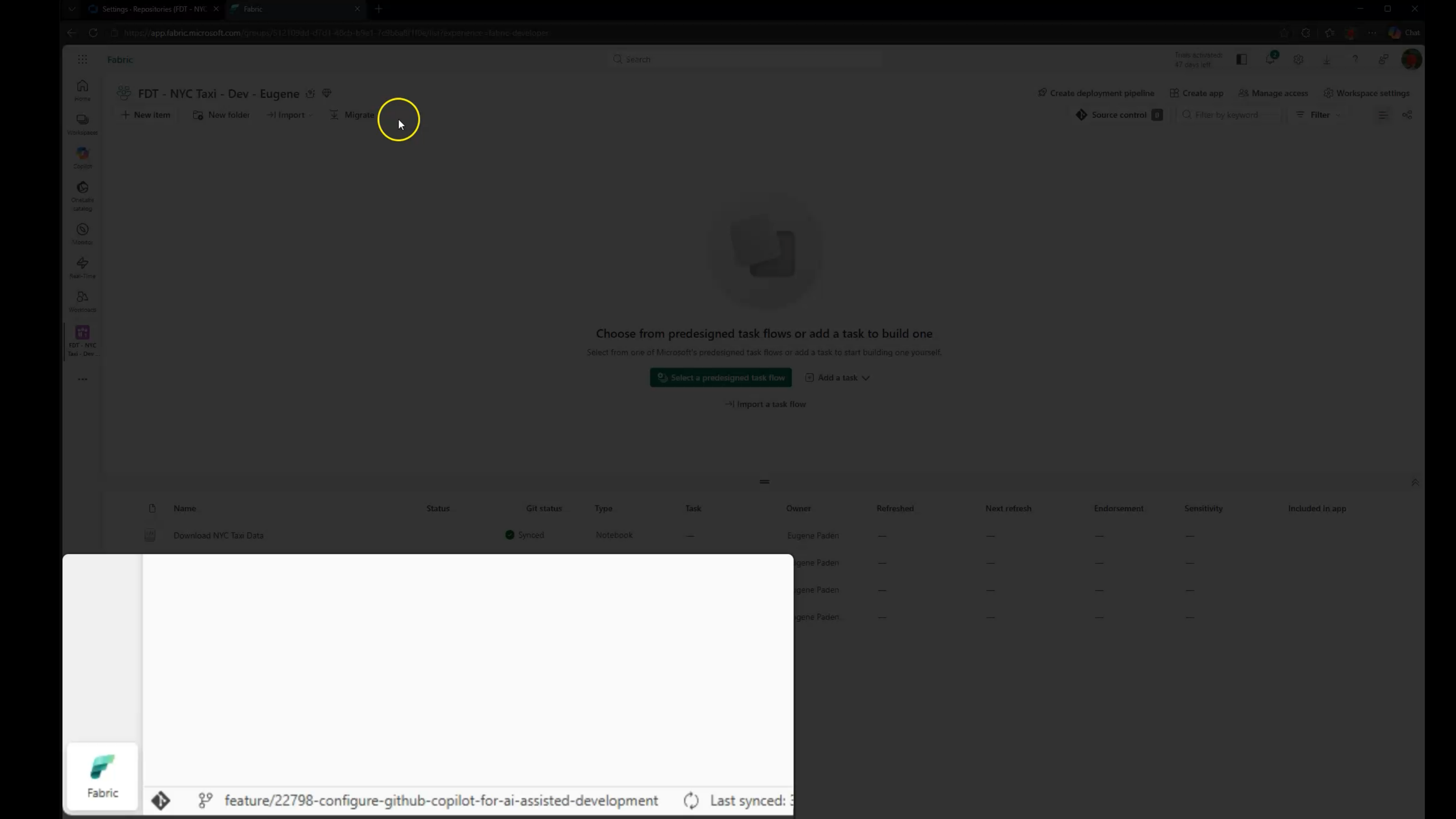
- Git Integration (8s)
- Setup Raw Data (34s)
- Copilot Instructions and Generate CI Pipeline (2:13)
- Create PR and PR Review Skills (1:29)
- Load into Bronze Layer, CI Run (1:27)
- Additional AI Artifact Generation and Validations
  - dbml
    - FDT Generate Models (2:48)
  - yaml
    - FDT Bronze to Silver Transformations (1:00)
    - FDT Silver to Gold Transformations (1:00)
- FDT Execution and AI Log Analysis – (1:40)


Settings · Repositories (FDT - NYC
Fabric
https://app.fabric.microsoft.com/groups/512109dd-d7d1-48cb-b9e1-7c9b6a8f1f0e/list?experience=fabric-developer
Chat
Fabric
Search
Trials activated:
47 days left
Home
Workspaces
Copilot
OneLake catalog
Monitor
Real-Time
Workloads
FDT - NYC Taxi - Dev ...
FDT - NYC Taxi - Dev - Eugene
Create deployment pipeline
Create app
Manage access
Workspace settings
New item
New folder
Import
Migrate
Source control 0
Filter by keyword
Filter
Choose from predesigned task flows or add a task to build one
Select from one of Microsoft's predesigned task flows or add a task to start building one yourself.
Select a predesigned task flow
Add a task
Import a task flow
Name
Status
Git status
Type
Task
Owner
Refreshed
Next refresh
Endorsement
Sensitivity
Included in app
Download NYC Taxi Data
Synced
Notebook
Eugene Paden
Download NYC Taxi Reference Data
Synced
Notebook
Eugene Paden
LH_NYC_Taxi
Synced
Lakehouse
Eugene Paden
LH_NYC_Taxi
SQL analytics endpo...
Eugene Paden
0:00

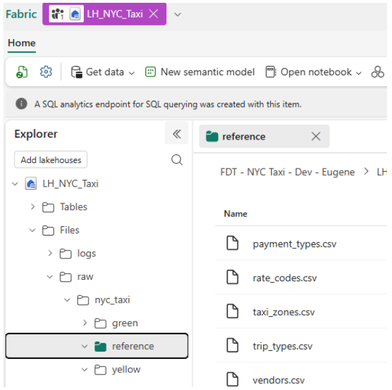
Branch Policies and Git Integration

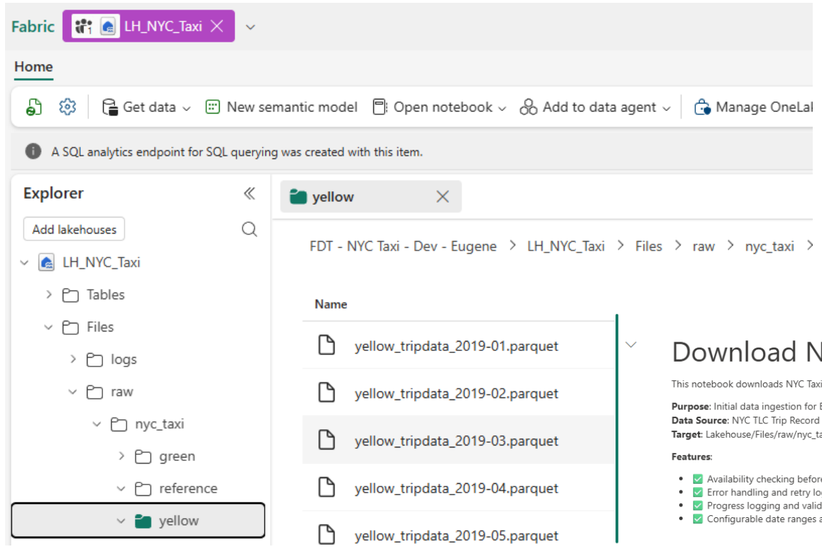


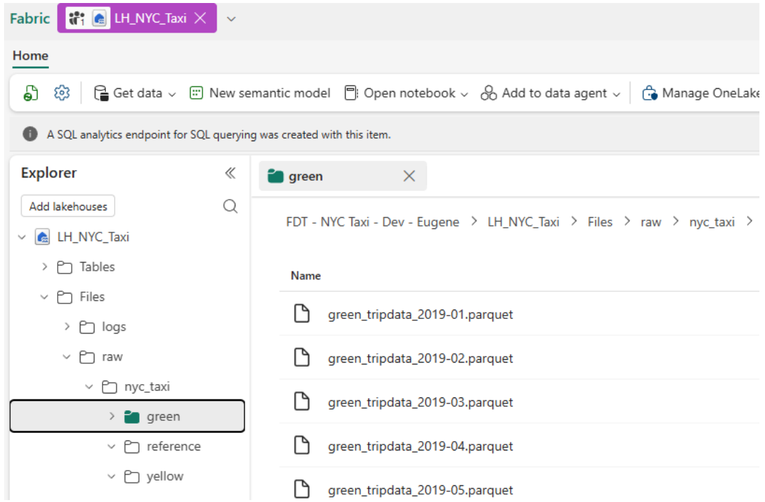



Settings - Repositories (FDT - NYC
Fabric
https://app.fabric.microsoft.com/groups/513109dd-d7d1-48cb-b9e1-7c9b6a8f1f0e/list?experience=fabric-developer
Chat
Fabric
Search
Trials activated
47 days left
FDT - NYC Taxi - Dev - Eugene
Create deployment pipeline
Create app
Manage access
Workspace settings
New item
New folder
Import
Migrate
Source control
Filter by keyword
Filter
Home
Workspaces
Copilot
OneLake catalog
Monitor
Real-Time
Workloads
FDT - NYC Taxi - Dev ...
Choose from predesigned task flows or add a task to build one
Select from one of Microsoft's predesigned task flows or add a task to start building one yourself.
Select a predesigned task flow
Add a task
Import a task flow
Name
Status
Git status
Type
Task
Owner
Refreshed
Next refresh
Endorsement
Sensitivity
Included in app
Download NYC Taxi Data
Synced
Notebook
Eugene Paden
Fabric
feature/22798-configure-github-copilot-for-ai-assisted-development
Last synced:

Load Raw Data







Download NYC Taxi Data to Lakehouse

This notebook downloads NYC Taxi parquet files from Azure Open Datasets and stores them in the Fabric lakehouse Files section.

Purpose: Initial data ingestion for Bronze layer
Data Source: NYC TLC Trip Record Data (via CloudFront CDN)
Target: Lakehouse/Files/raw/nyc_taxi/

Features:

- ✅ Availability checking before download
- ✅ Error handling and retry logic
- ✅ Progress logging and validation
- ✅ Configurable date ranges and taxi types

Setup: Lakehouse Configuration

This notebook uses parameters to configure the target lakehouse dynamically.

Default Lakehouse: `LH_NYC_Taxi`

The `%%configure` cell below uses parameterization to allow the lakehouse name to be overridden when called from a Fabric pipeline.

Apply Lakehouse Configuration

The `%%configure` cell below uses parameterization to attach the lakehouse dynamically.

Default: Uses `LH_NYC_Taxi` when run standalone
Pipeline: Accepts `defaultLakehouseName` parameter when called from a pipeline

```
%%configure -f
{
    "defaultLakehouse": {
        "name": {
            "parameterName": "defaultLakehouseName",
            "defaultValue": "LH_NYC_Taxi"
        }
    }
}
```

Download Configuration Parameters

Modify these parameters to control what data to download.

Taxi Types: Can be a single type (`"yellow"`) or multiple types (`["yellow", "green"]`)
Note: The lakehouse (`LH_NYC_Taxi` by default) is configured via the `%%configure` cell above.

Download NYC Taxi Reference Data

This notebook downloads NYC Taxi reference/lookup tables and stores them in the Fabric lakehouse.

Purpose: Download reference data for Bronze layer enrichment
Data Sources: NYC TLC official reference data
Target: Lakehouse/Files/raw/nyc_taxi/reference/

Reference Tables:

- ✅ Taxi Zones (LocationID → Borough, Zone, Service Area)
- ✅ Rate Codes (RateCodeID → Description)
- ✅ Payment Types (Payment Type ID → Description)
- ✅ Trip Types (Trip Type ID → Description)
- ✅ Vendor IDs (VendorID → Company Name)

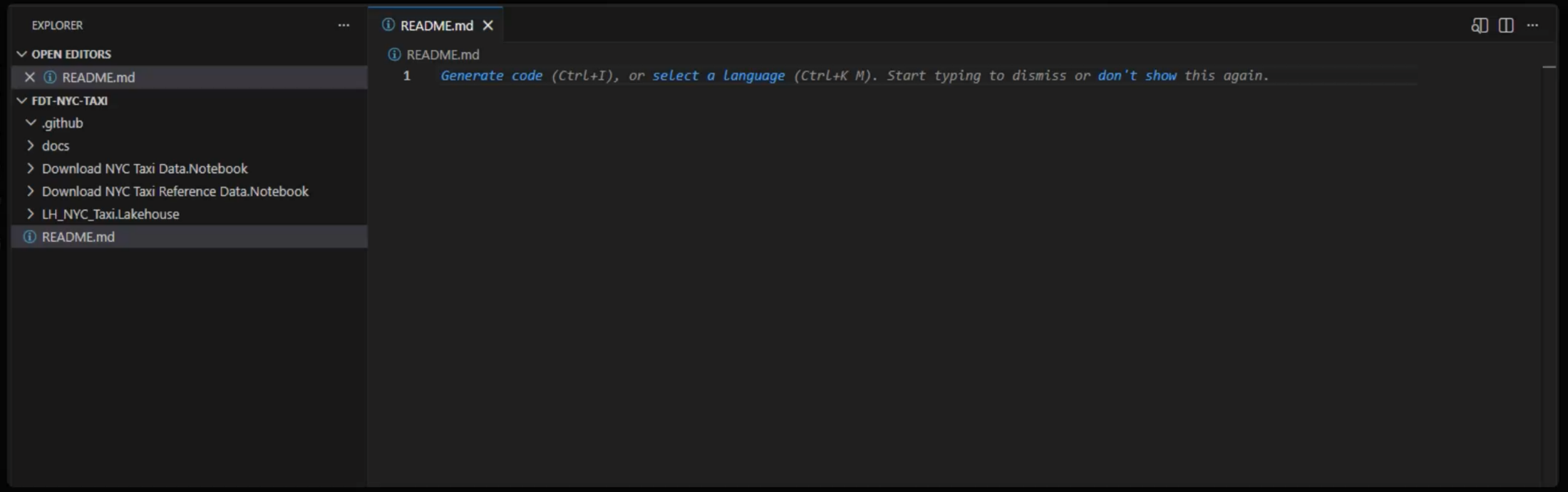
```
%%configure -f
{
    "defaultLakehouse": {
        "name": "LH_NYC_Taxi",
        "parameterName": "lakehouse_name"
    }
}
```

Imports

```
import requests
import time
import os
import io
import pandas as pd
from typing import Optional

print("✓ Libraries imported successfully")
```

Configuration

```
# Configuration
LAKEHOUSE_BASE_PATH = "/lakehouse/default/Files"
REFERENCE_FOLDER = f"{LAKEHOUSE_BASE_PATH}/raw/nyc_taxi/reference"

# Reference data URLs
TAXI_ZONES_URL = "https://d37ci6vzurychx.cloudfront.net/misc/taxi+_zone_lookup.csv"
```


EXPLORER
OPEN EDITORS
README.md
FDT-NYC-TAXI
.github
docs
Download NYC Taxi Data.Notebook
Download NYC Taxi Reference Data.Notebook
LH_NYC_Taxi.Lakehouse
README.md
README.md
README.md
1 Generate code (Ctrl+I), or select a language (Ctrl+K M). Start typing to dismiss or don't show this again.

Copilot Instructions and CI Pipeline

feature/22798-configure-github-copilot-for-ai-assisted-development / .github / copilot-instructions.md

copilot-instructions.md

Contents | Preview | History | Compare | Blame

You updated feature/23390-create-downloadable-ci-cd-package Yesterday

GitHub Copilot Instructions for Fabric NYC Taxi Project

Project Context

This is a Git-integrated Microsoft Fabric workspace demonstrating enterprise-ready data engineering practices. The project uses:

- **Source Control**: Azure DevOps Repositories
  - **Project**: FDT - NYC Taxi
  - **Repository**: FDT - NYC Taxi
- **CI/CD**: Azure DevOps Pipelines
- **Platform**: Microsoft Fabric (Lakehouse, Notebooks, Data Pipelines)
- **Development Approach**: AI-powered development with GitHub Copilot

Enterprise-Ready Principles

Every artifact and code change must follow these five principles in order:

1. Make It Work

- Implement functional, tested code that meets requirements
- Ensure data flows correctly through medallion architecture (Bronze → Silver → Gold)
- Validate all data transformations and business logic
- Test notebooks before committing

2. Make It Secure

- Never hardcode credentials, connection strings, or sensitive data
- Use Fabric environment variables and Key Vault references
- Implement row-level security (RLS) where applicable
- Sanitize user inputs and validate data sources
- Follow principle of least privilege for access controls
- Scan for security vulnerabilities in dependencies

3. Make It Scale

- Design for large datasets using Delta Lake optimization
- Use partition strategies for efficient data processing
- Implement incremental loading patterns (not full refreshes)
- Consider compute resource requirements and optimization
- Use appropriate Spark configurations for workload size
- Design for parallel processing where possible

feature/22798-configure-github-copilot-for-ai-assisted-development / azure-pipelines.yml

azure-pipelines.yml

Contents History Compare Blame

You updated feature/23390-create-downloadable-ci-cd-package Yesterday

```yaml
trigger:
  branches:
    include:
    - main

pr:
  branches:
    include:
    - main

pool:
  vmImage: 'ubuntu-latest'

variables:
  - name: pythonVersion
    value: '3.11'
  - name: validationFailed
    value: false

stages:
- stage: Validate
  displayName: 'Validate Fabric Artifacts'
  jobs:
  - job: ValidateArtifacts
    displayName: 'Validate All Artifacts'
    steps:

    - task: UsePythonVersion@0
      displayName: 'Use Python $(pythonVersion)'
      inputs:
        versionSpec: '$(pythonVersion)'
        addToPath: true

    - script: |
        python -m pip install --upgrade pip
        pip install pyyaml jsonschema bandit safety pylint
      displayName: 'Install validation dependencies'

    - script: |
        # Scan Python files for security issues
        echo "Scanning Python files with Bandit..."
        bandit -r . -f json -o $(Build.ArtifactStagingDirectory)/bandit-results.json --exit-zero || true

        # Show summary
        if [ -f "$(Build.ArtifactStagingDirectory)/bandit-results.json" ]; then
          echo "Bandit scan completed. Results saved to bandit-results.json"
        fi
      displayName: 'Security scan with Bandit'
      continueOnError: true

    - script: |
        # Check for known vulnerabilities in dependencies
        echo "Checking Python dependencies with Safety..."

        # Safety can fail if no requirements.txt exists, which is OK for Fabric projects
        if safety check --json --output json-file --file $(Build.ArtifactStagingDirectory)/safety-results.json 2>/dev/null; then
          echo "Safety scan completed successfully"
        elif safety check --json 2>&1 | grep -v "^+" > $(Build.ArtifactStagingDirectory)/safety-results.json; then
          echo "Safety scan completed with filtered output"
        else
          # If Safety fails (no requirements.txt or other error), create empty result
          echo '[]' > $(Build.ArtifactStagingDirectory)/safety-results.json
          echo "Safety scan skipped - no Python dependencies found"
        fi
      displayName: 'Dependency vulnerability scan with Safety'
      continueOnError: true
```

FDT - NYC Taxi

- .github
  - skills
  - copilot-instructions.md
- .vscode
- docs
- LH_NYC_Taxi.Lakehouse
- NB_100_BRONZE_Download_Taxi_Data.Noteb
- NB_101_BRONZE_Download_Reference_Data.I
- scripts
  - validators
    - validate-lakehouse.ps1
    - validate-notebook.ps1
    - validate-pipeline.ps1
  - generate-validation-report.ps1
  - post-pr-comment.ps1
  - README.md
  - validate-artifacts.ps1
- .gitignore
- .pylintrc
- azure-pipelines.yml
- README.md

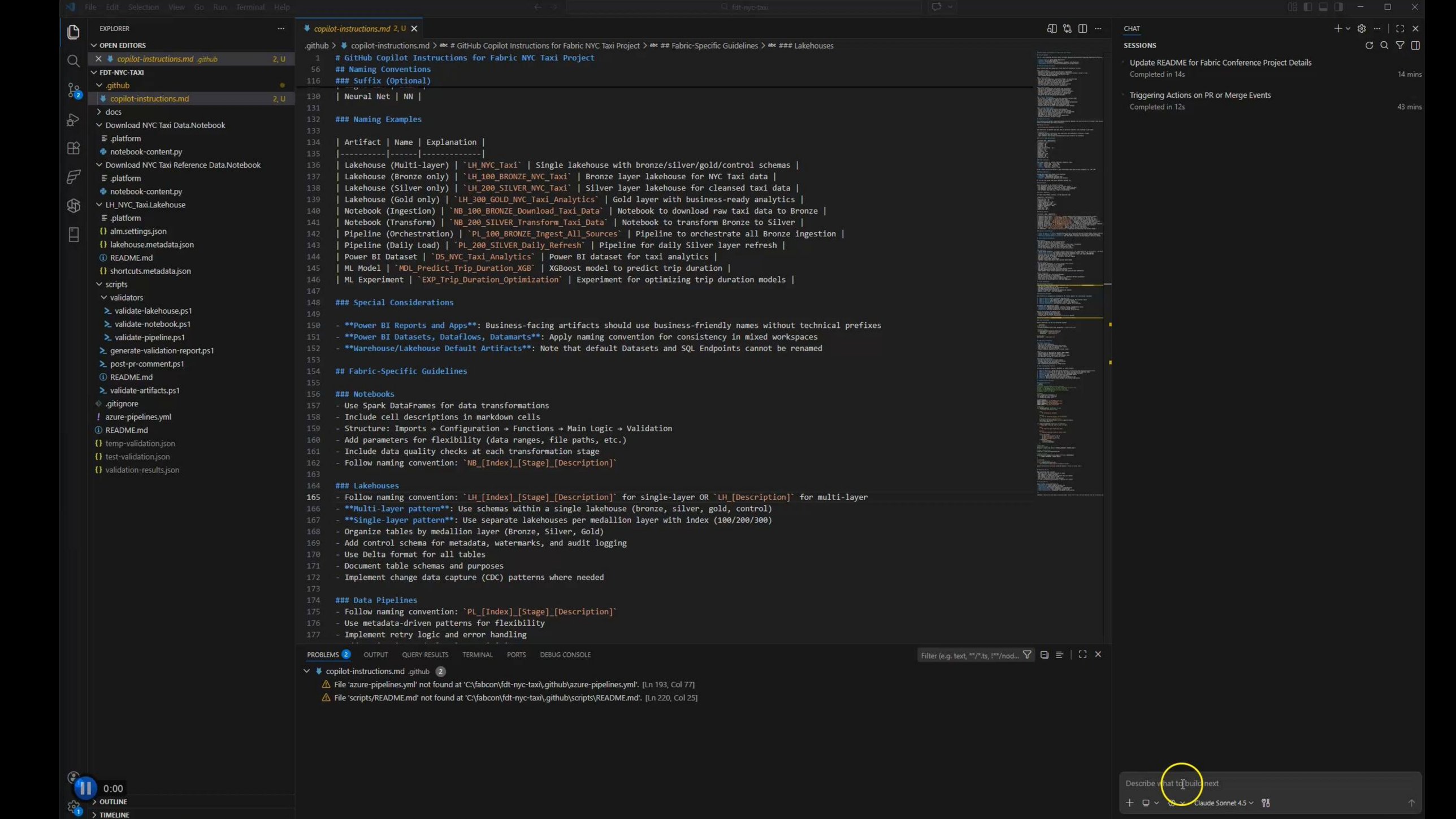
feature/22798-configure-github-copilot-for-ai-assisted-development / sc

validate-artifacts.ps1

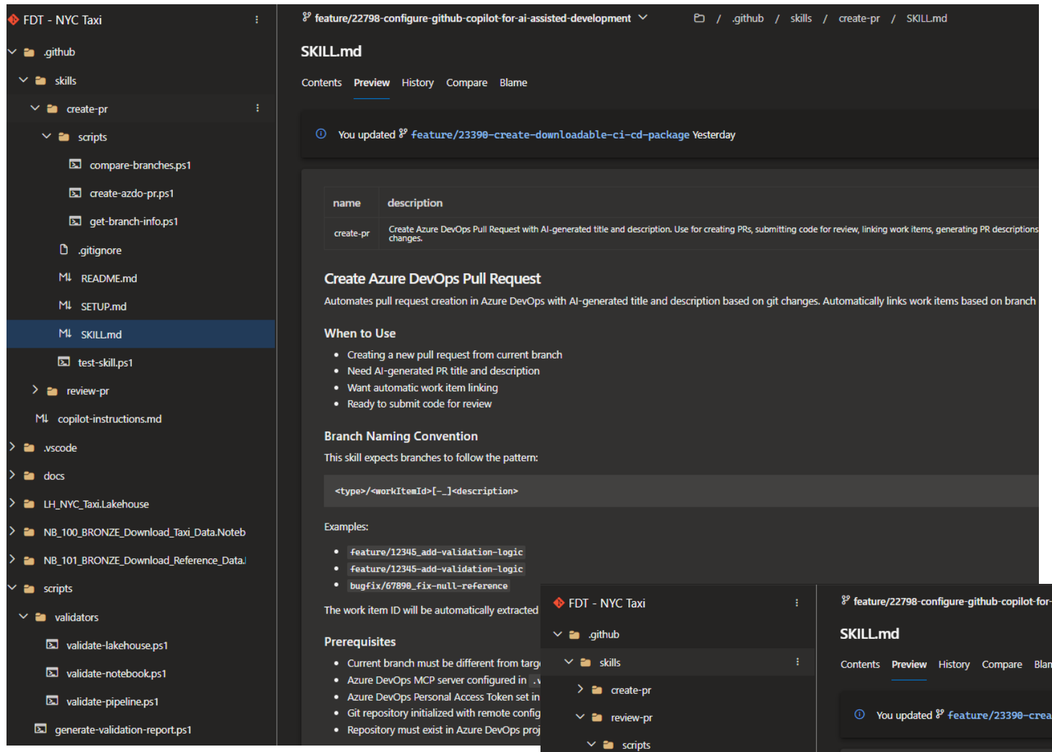
Contents History Compare Blame

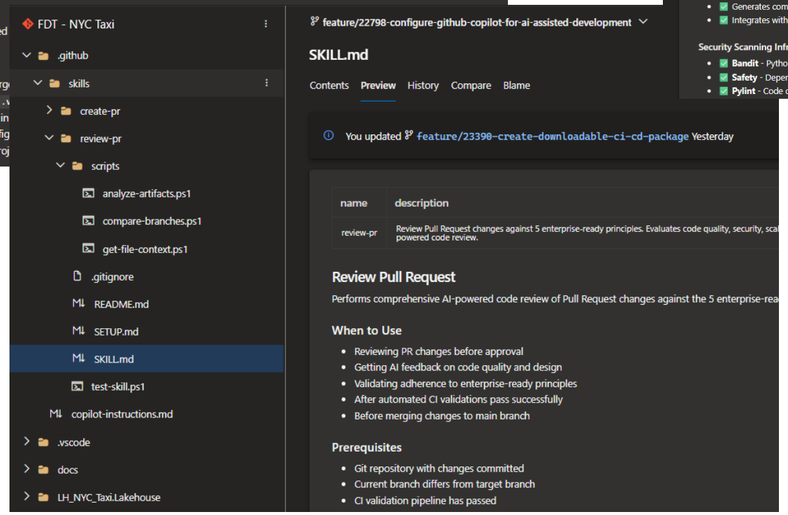
You updated feature/23390-create-downloadable-ci-cd-package Yesterday

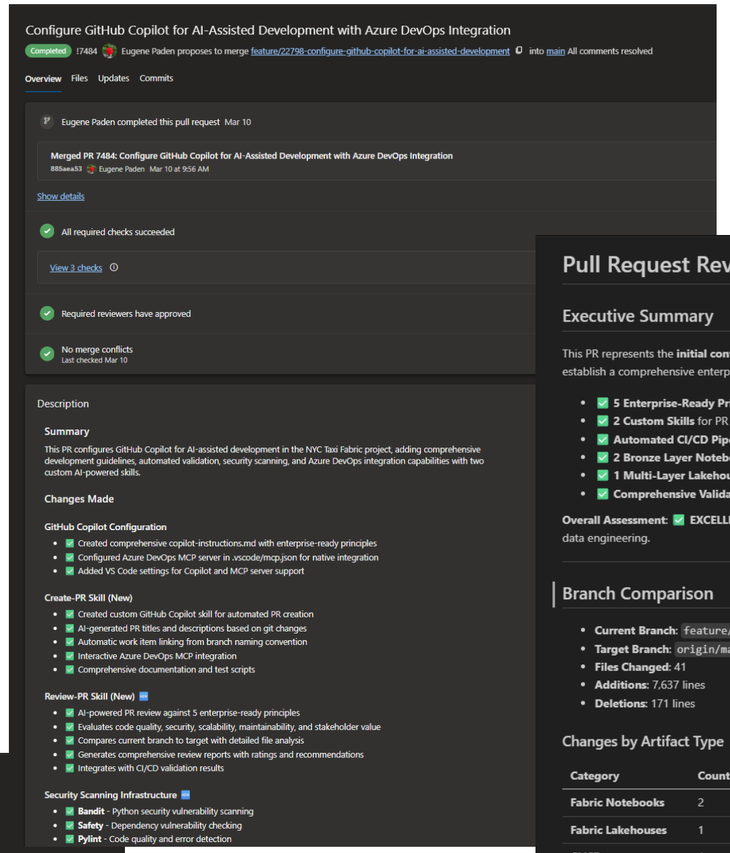
```powershell
# Validate Fabric Artifacts - Main Orchestrator
# This script orchestrates validation of different Fabric artifact types
# Calls separate validators for each artifact type for easier debugging

param(
    [Parameter(Mandatory=$true)]
    [string]$OutputPath
)

$ErrorActionPreference = "Continue"

# Get the script directory
$scriptDir = Split-Path -Parent $MyInvocation.MyCommand.Path
$validatorsDir = Join-Path $scriptDir "validators"

# Initialize results
$results = @{
    Timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss UTC"
    TotalArtifacts = 0
    PassedValidations = 0
    FailedValidations = 0
    WarningValidations = 0
    HasViolations = $false
    Artifacts = @()
}

Write-Host "====================================" -ForegroundColor Cyan
Write-Host "Fabric Artifact Validation" -ForegroundColor Cyan
Write-Host "====================================" -ForegroundColor Cyan
Write-Host "Working directory: $(Get-Location)" -ForegroundColor Gray
Write-Host "Validators directory: $validatorsDir" -ForegroundColor Gray
Write-Host ""

# Function to validate naming convention
function Test-NamingConvention {
    param([string]$Name, [string]$Type)

    $violations = @()

    # Expected pattern: ArtifactType_Index_Stage_Description_Suffix
    # At minimum: ArtifactType_Description

    $typeAbbreviations = @{
        "Notebook" = "NB"
        "Lakehouse" = "LH"
        "Pipeline" = "PL"
        "Dataflow" = "DFL"
        "Dataset" = "DS"
        "Warehouse" = "WH"
    }
```

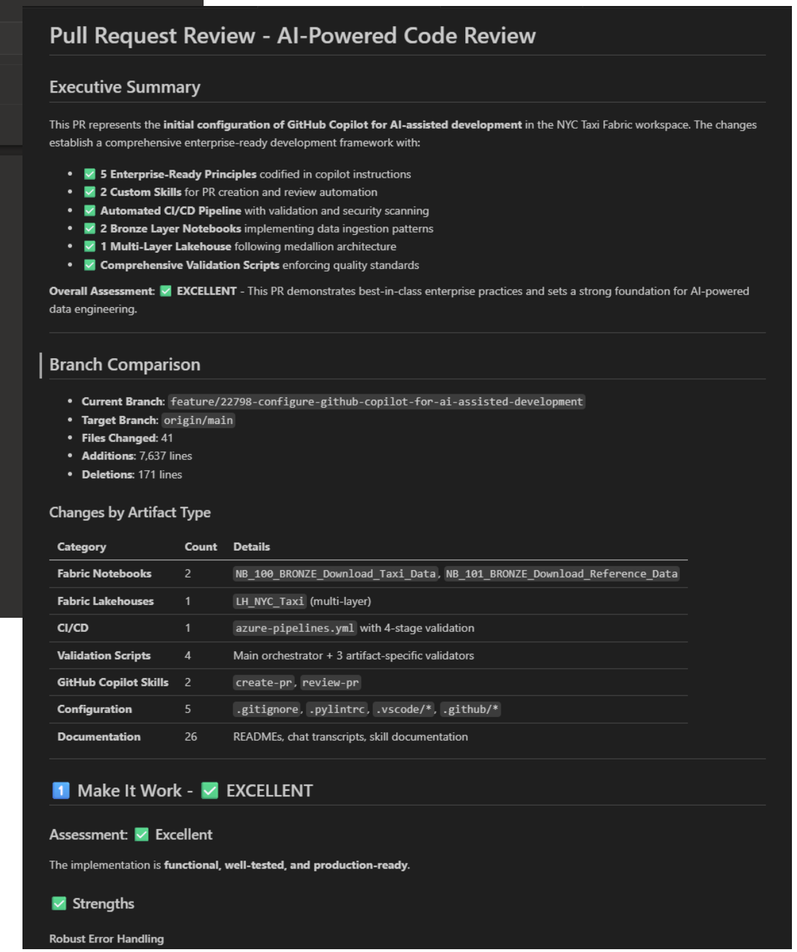

EXPLORER
OPEN EDITORS
copilot-instructions.md .github 2, U
FDT-NYC-TAXI
.github
copilot-instructions.md 2, U
docs
Download NYC Taxi Data.Notebook
.platform
notebook-content.py
Download NYC Taxi Reference Data.Notebook
.platform
notebook-content.py
LH_NYC_Taxi.Lakehouse
.platform
alm.settings.json
lakehouse.metadata.json
README.md
shortcuts.metadata.json
scripts
validators
validate-lakehouse.ps1
validate-notebook.ps1
validate-pipeline.ps1
generate-validation-report.ps1
post-pr-comment.ps1
README.md
validate-artifacts.ps1
.gitignore
azure-pipelines.yml
README.md
temp-validation.json
test-validation.json
validation-results.json
copilot-instructions.md 2, U
.github > copilot-instructions.md > # GitHub Copilot Instructions for Fabric NYC Taxi Project > ## Fabric-Specific Guidelines > ### Lakehouses
1 # GitHub Copilot Instructions for Fabric NYC Taxi Project
56 ## Naming Conventions
116 ### Suffix (Optional)
130 | Neural Net | NN |
132 ### Naming Examples
134 | Artifact | Name | Explanation |
135 |----------|------|-------------|
136 | Lakehouse (Multi-layer) | `LH_NYC_Taxi` | Single lakehouse with bronze/silver/gold/control schemas |
137 | Lakehouse (Bronze only) | `LH_100_BRONZE_NYC_Taxi` | Bronze layer lakehouse for NYC Taxi data |
138 | Lakehouse (Silver only) | `LH_200_SILVER_NYC_Taxi` | Silver layer lakehouse for cleansed taxi data |
139 | Lakehouse (Gold only) | `LH_300_GOLD_NYC_Taxi_Analytics` | Gold layer with business-ready analytics |
140 | Notebook (Ingestion) | `NB_100_BRONZE_Download_Taxi_Data` | Notebook to download raw taxi data to Bronze |
141 | Notebook (Transform) | `NB_200_SILVER_Transform_Taxi_Data` | Notebook to transform Bronze to Silver |
142 | Pipeline (Orchestration) | `PL_100_BRONZE_Ingest_All_Sources` | Pipeline to orchestrate all Bronze ingestion |
143 | Pipeline (Daily Load) | `PL_200_SILVER_Daily_Refresh` | Pipeline for daily Silver layer refresh |
144 | Power BI Dataset | `DS_NYC_Taxi_Analytics` | Power BI dataset for taxi analytics |
145 | ML Model | `MDL_Predict_Trip_Duration_XGB` | XGBoost model to predict trip duration |
146 | ML Experiment | `EXP_Trip_Duration_Optimization` | Experiment for optimizing trip duration models |
148 ### Special Considerations
150 - **Power BI Reports and Apps**: Business-facing artifacts should use business-friendly names without technical prefixes
151 - **Power BI Datasets, Dataflows, Datamarts**: Apply naming convention for consistency in mixed workspaces
152 - **Warehouse/Lakehouse Default Artifacts**: Note that default Datasets and SQL Endpoints cannot be renamed
154 ## Fabric-Specific Guidelines
156 ### Notebooks
157 - Use Spark DataFrames for data transformations
158 - Include cell descriptions in markdown cells
159 - Structure: Imports → Configuration → Functions → Main Logic → Validation
160 - Add parameters for flexibility (data ranges, file paths, etc.)
161 - Include data quality checks at each transformation stage
162 - Follow naming convention: `NB_[Index]_[Stage]_[Description]`
164 ### Lakehouses
165 - Follow naming convention: `LH_[Index]_[Stage]_[Description]` for single-layer OR `LH_[Description]` for multi-layer
166 - **Multi-layer pattern**: Use schemas within a single lakehouse (bronze, silver, gold, control)
167 - **Single-layer pattern**: Use separate lakehouses per medallion layer with index (100/200/300)
168 - Organize tables by medallion layer (Bronze, Silver, Gold)
169 - Add control schema for metadata, watermarks, and audit logging
170 - Use Delta format for all tables
171 - Document table schemas and purposes
172 - Implement change data capture (CDC) patterns where needed
174 ### Data Pipelines
175 - Follow naming convention: `PL_[Index]_[Stage]_[Description]`
176 - Use metadata-driven patterns for flexibility
177 - Implement retry logic and error handling
PROBLEMS 2
OUTPUT
QUERY RESULTS
TERMINAL
PORTS
DEBUG CONSOLE
Filter (e.g. text, **/*.ts, !**/nod...
copilot-instructions.md .github 2
File 'azure-pipelines.yml' not found at 'C:\fabcon\fdt-nyc-taxi\.github\azure-pipelines.yml'. [Ln 193, Col 77]
File 'scripts/README.md' not found at 'C:\fabcon\fdt-nyc-taxi\.github\scripts\README.md'. [Ln 220, Col 25]
CHAT
SESSIONS
Update README for Fabric Conference Project Details
Completed in 14s
14 mins
Triggering Actions on PR or Merge Events
Completed in 12s
43 mins
Describe what to build next
Claude Sonnet 4.5
0:00
OUTLINE
TIMELINE

Create PR and PR Review Skills









Branches - Repos | PL_1100_BRONZE_Load_Reference

https://app.fabric.microsoft.com/groups/512109dd-d7d1-48cb-b9e1-7c9b6a8f1f0e/pipelines/037a9c6d-2223-41e2-925c-7e7d92877a5d?experience=fabric-developer

Fabric | PL_1100_BRONZE_Load_Reference_Data | Search | Trials activated: 47 days left

Home | Activities | Run | View

Validate | Run | Schedule | Trigger | View run history | Copy data | Dataflow | Notebook | Lookup | Invoke Pipeline

Home | Workspaces | Copilot | OneLake catalog | Monitor | Real-Time | Workloads | FDT - NYC Taxi - Dev ...

Copy data

Copy data1

General | Source | Destination | Mapping | Settings

Connection * LH_NYC_Taxi (FDT - NYC Taxi - De... Refresh Open

Root folder Tables Files

File path type File path Wildcard file path List of files

File path raw/nyc_taxi/reference / payment_types.csv Browse Preview data

Recursively

File format * DelimitedText Settings

Advanced

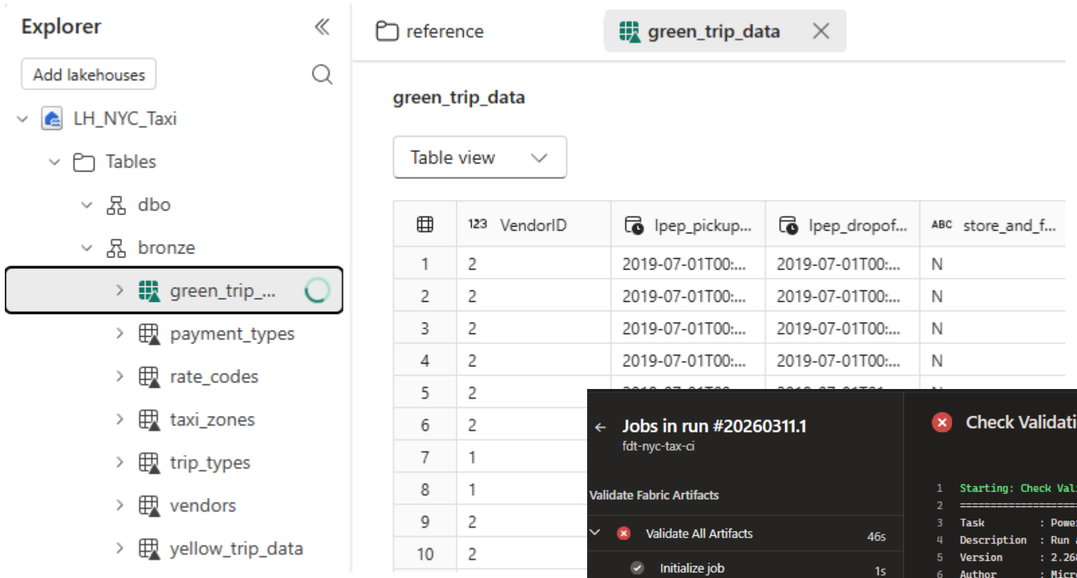
Filter by last modified Start time (UTC) End time (UTC)

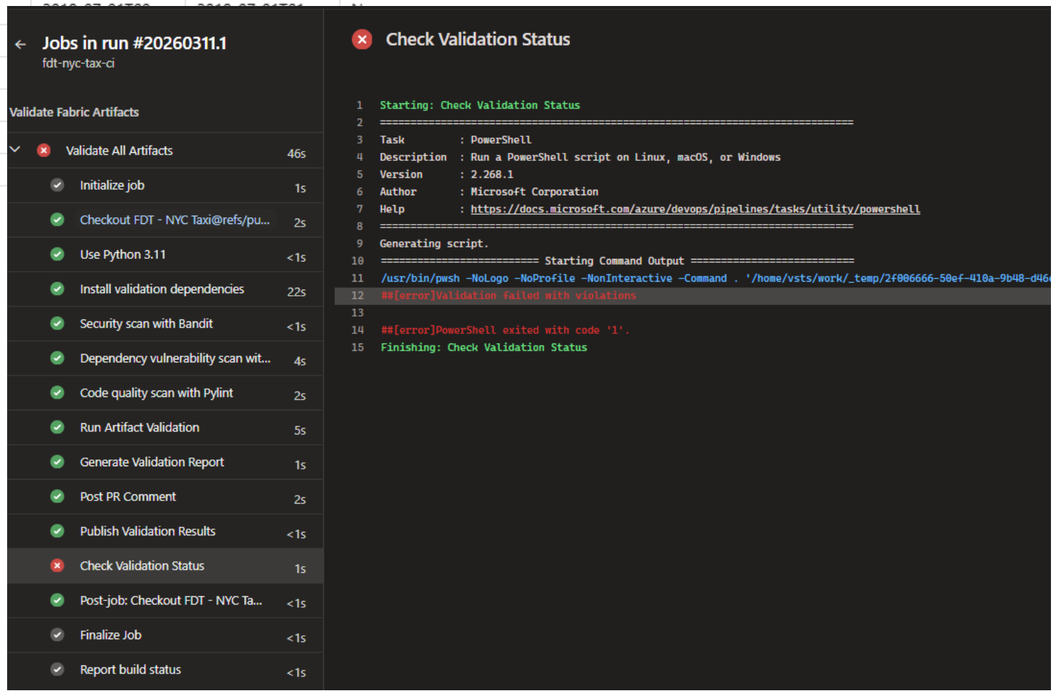
Enable partitions discovery

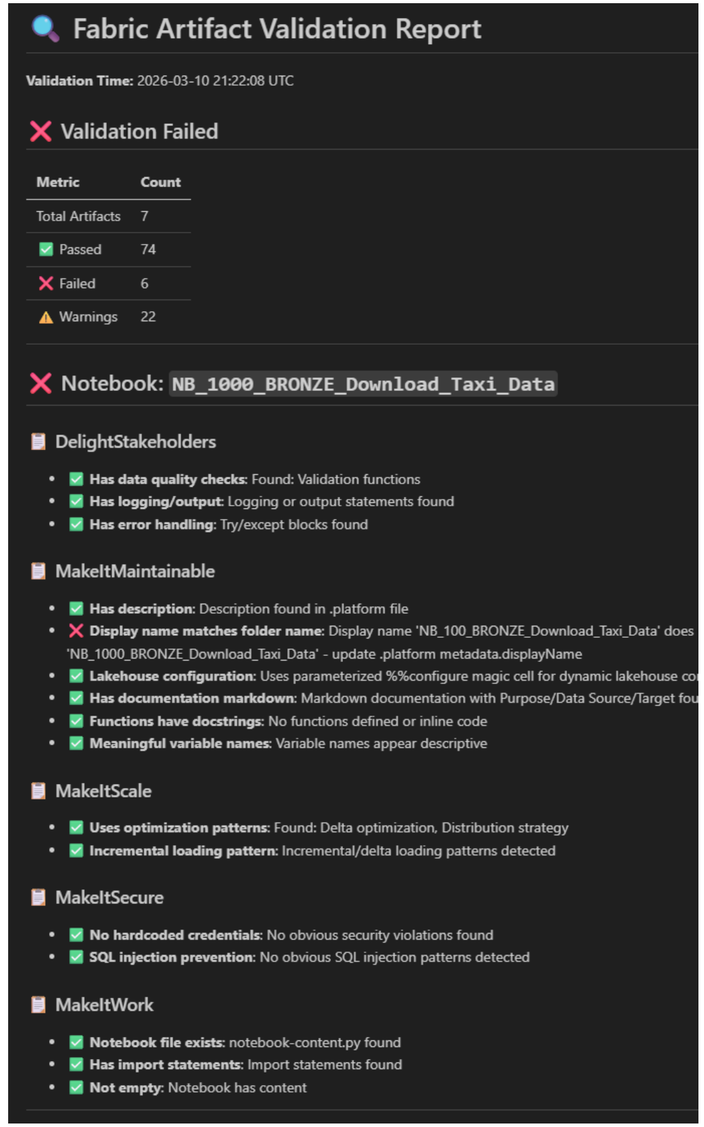
Max concurrent connections

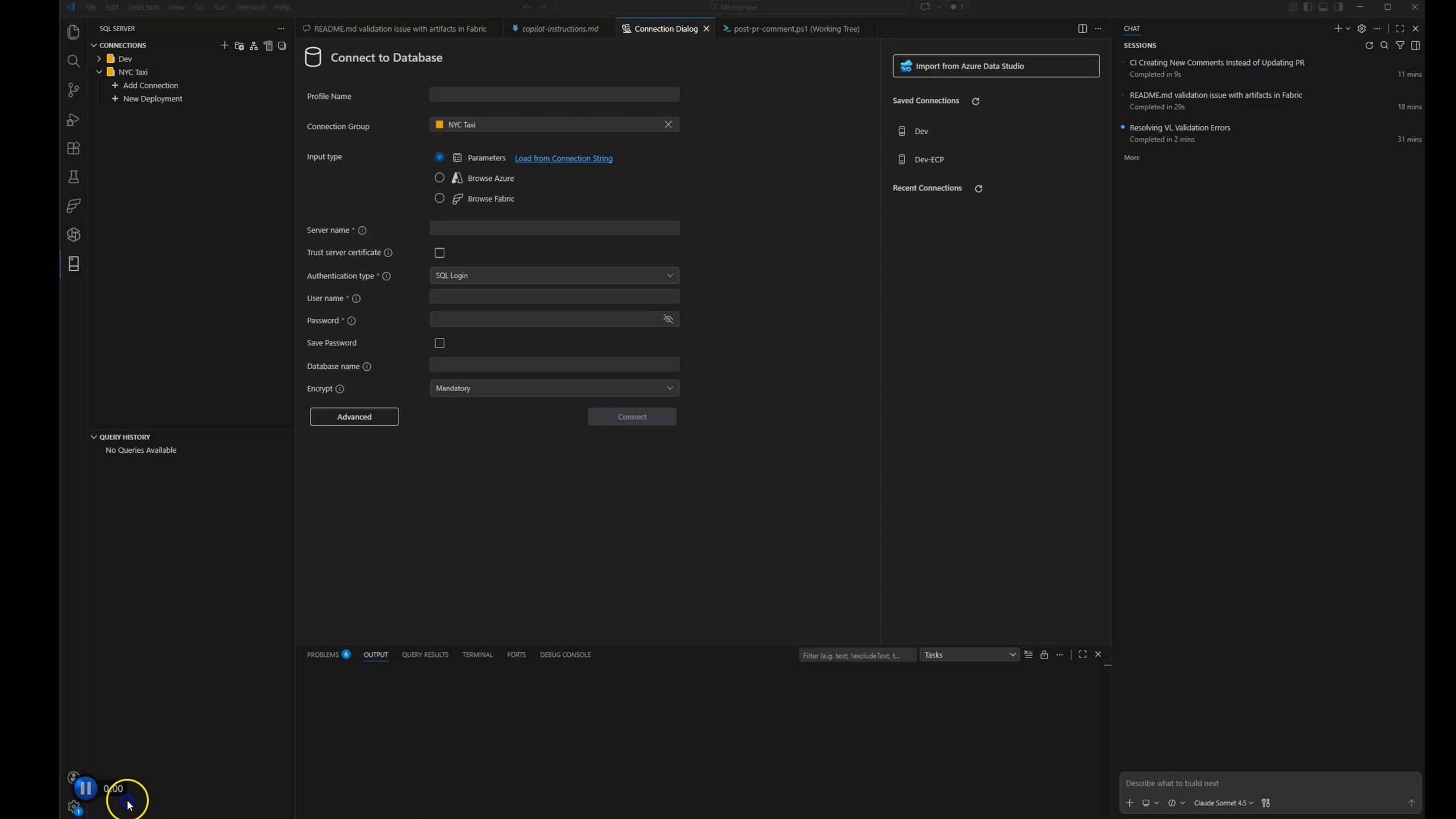
Additional columns New

Load Bronze Layer, CI Run





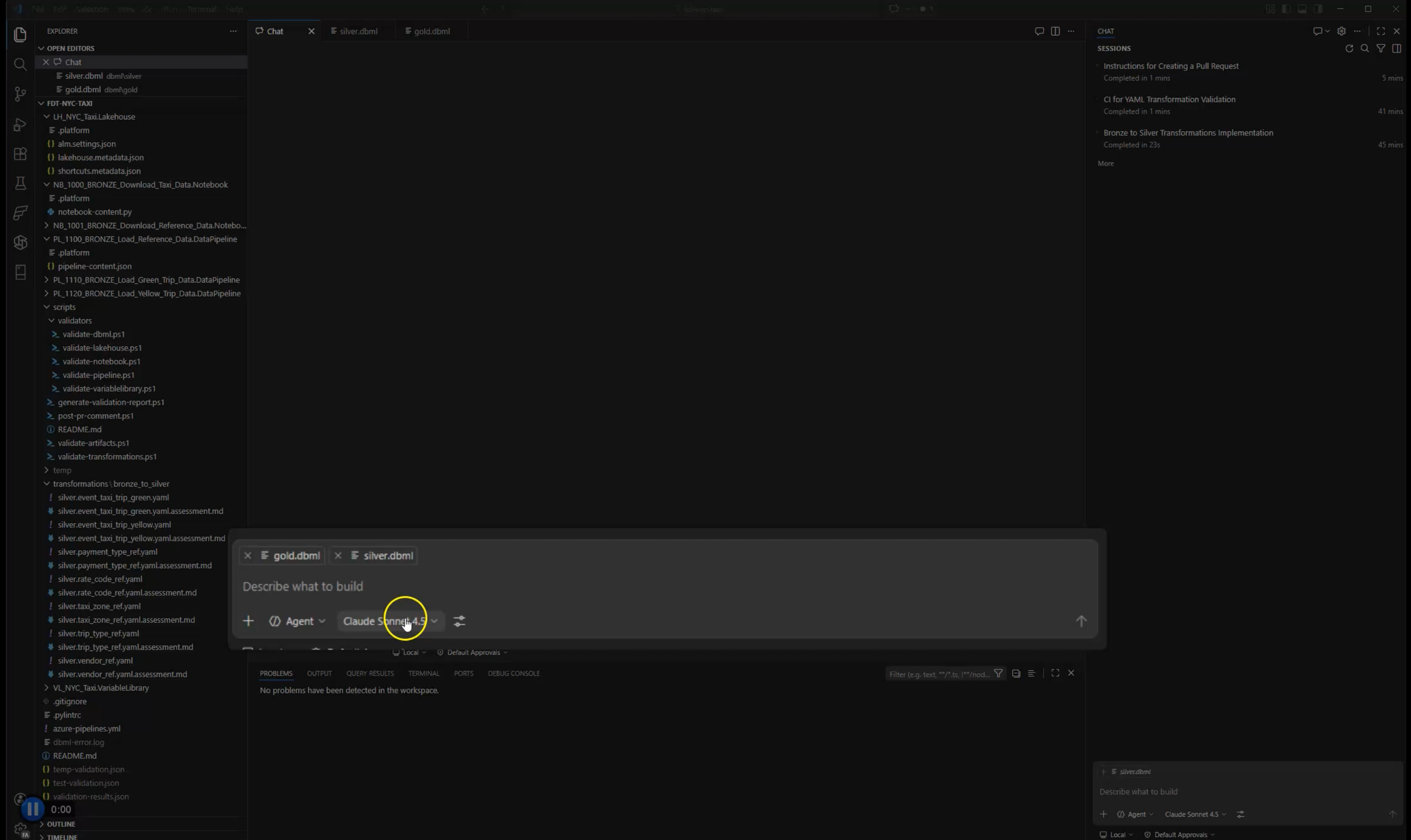



SQL SERVER
CONNECTIONS
Dev
NYC Taxi
Add Connection
New Deployment
QUERY HISTORY
No Queries Available
README.md validation issue with artifacts in Fabric
copilot-instructions.md
Connection Dialog
post-pr-comment.ps1 (Working Tree)
Connect to Database
Profile Name
Connection Group
NYC Taxi
Input type
Parameters
Load from Connection String
Browse Azure
Browse Fabric
Server name *
Trust server certificate
Authentication type *
SQL Login
User name *
Password *
Save Password
Database name
Encrypt
Mandatory
Advanced
Connect
Import from Azure Data Studio
Saved Connections
Dev
Dev-ECP
Recent Connections
CHAT
SESSIONS
CI Creating New Comments Instead of Updating PR
Completed in 9s
11 mins
README.md validation issue with artifacts in Fabric
Completed in 29s
18 mins
Resolving VL Validation Errors
Completed in 2 mins
31 mins
More
PROBLEMS 6
OUTPUT
QUERY RESULTS
TERMINAL
PORTS
DEBUG CONSOLE
Filter (e.g. text, !excludeText, t...
Tasks
Describe what to build next
Claude Sonnet 4.5
0:00

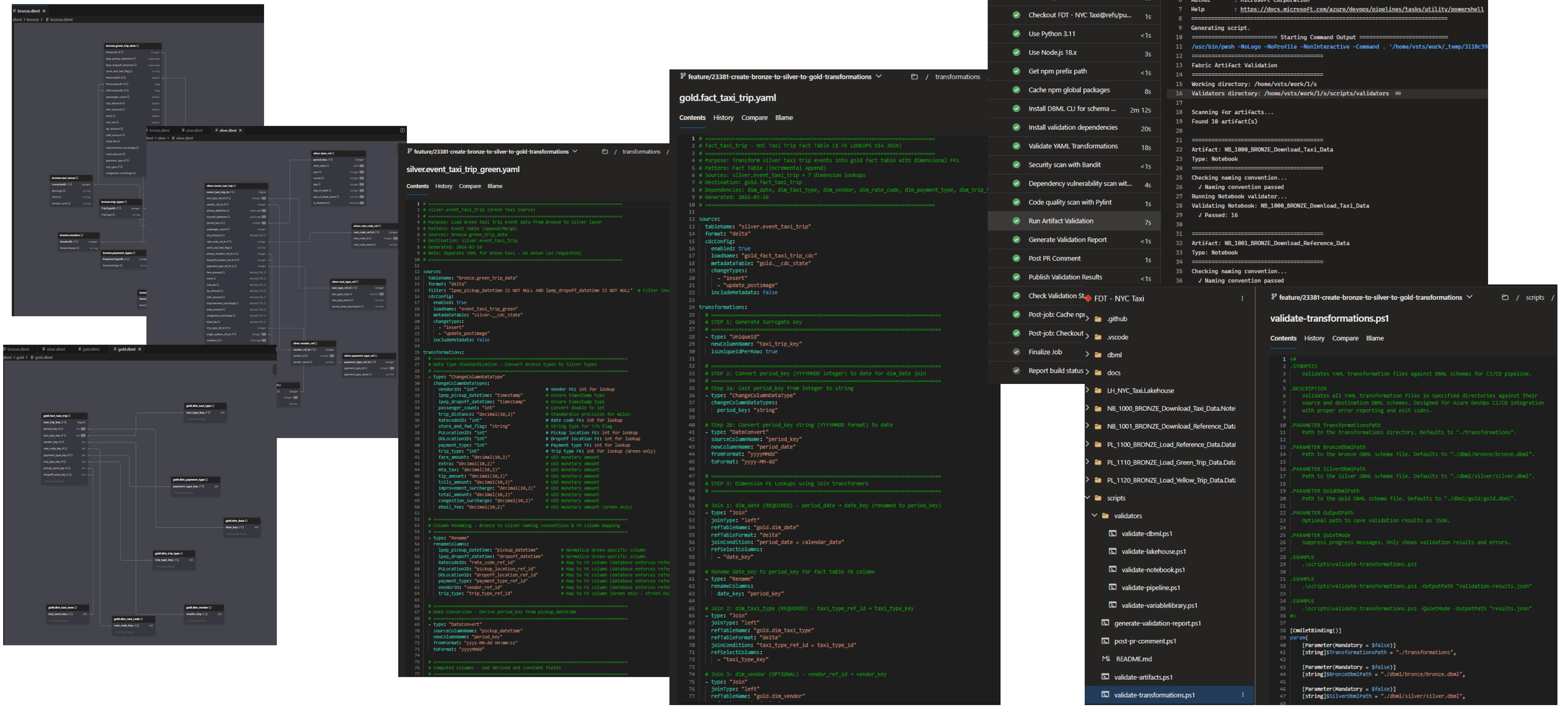
Chat | bronze.dbml

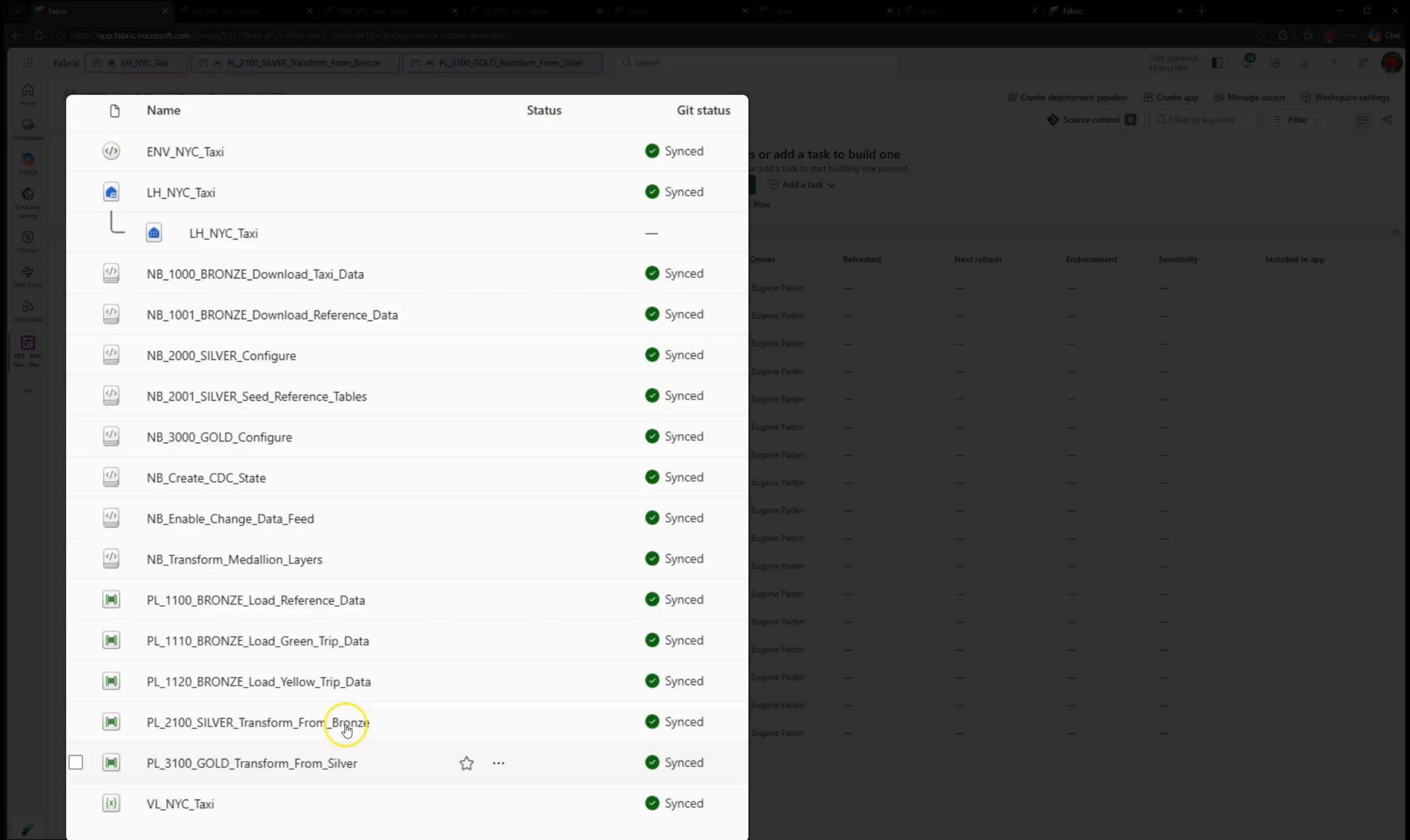
dbml > bronze > bronze.dbml

```
// Description: Raw taxi trip records and reference data as received from NYC TLC
// Last Updated: 2026-03-10

Project nyc_taxi_bronze {
  database_type: 'delta'
  Note: '''
    # NYC Taxi Bronze Layer

    This schema represents the Bronze (raw) layer of the NYC Taxi data platform.
    Data is ingested as-is from NYC TLC sources without transformation.

    **Data Sources:**
    - Yellow Taxi Trip Records: 2009-present
    - Green Taxi Trip Records: 2013-present
    - Reference Data: Taxi zones, rate codes, payment types

    **Schema:** bronze
    **Format:** Delta Lake
    **Update Pattern:** Monthly incremental loads
  '''
}

// ==========================================================================
// YELLOW TAXI TRIP RECORDS
// ==========================================================================

Table bronze.yellow_trip_data {
  VendorID integer [
    note: 'Source: NYC TLC Yellow Taxi Parquet files (NB_1000); Conformance: integer; Transformation: RAW; Validation: IN (1, 2); Nullable: true; Descri
  ]

  tpep_pickup_datetime timestamp [
    note: 'Source: NYC TLC Yellow Taxi Parquet files (NB_1000); Conformance: timestamp; Transformation: RAW; Validation: NOT NULL, >= 2009-01-01; Nullab
  ]

  tpep_dropoff_datetime timestamp [
    note: 'Source: NYC TLC Yellow Taxi Parquet files (NB_1000); Conformance: timestamp; Transformation: RAW; Validation: >= tpep_pickup_datetime; Nullab
  ]

  passenger_count double [
    note: 'Source: NYC TLC Yellow Taxi Parquet files (NB_1000); Conformance: double; Transformation: RAW; Validation: 0-9; Nullable: true; Description:
  ]

  trip_distance double [
    note: 'Source: NYC TLC Yellow Taxi Parquet files (NB_1000); Conformance: double; Transformation: RAW; Validation: >= 0; Nullable: true; Description:
  ]

  RatecodeID double [
    note: 'Source: NYC TLC Yellow Taxi Parquet files (NB_1000); Conformance: double; Transformation: RAW; Validation: 1-6; Nullable: true; Description:
  ]
```


EXPLORER
OPEN EDITORS
Chat
silver.dbml dbml\silver
gold.dbml dbml\gold
FDT-NYC-TAXI
LH_NYC_Taxi.Lakehouse
.platform
alm.settings.json
lakehouse.metadata.json
shortcuts.metadata.json
NB_1000_BRONZE_Download_Taxi_Data.Notebook
.platform
notebook-content.py
NB_1001_BRONZE_Download_Reference_Data.Notebo...
PL_1100_BRONZE_Load_Reference_Data.DataPipeline
.platform
pipeline-content.json
PL_1110_BRONZE_Load_Green_Trip_Data.DataPipeline
PL_1120_BRONZE_Load_Yellow_Trip_Data.DataPipeline
scripts
validators
validate-dbml.ps1
validate-lakehouse.ps1
validate-notebook.ps1
validate-pipeline.ps1
validate-variablelibrary.ps1
generate-validation-report.ps1
post-pr-comment.ps1
README.md
validate-artifacts.ps1
validate-transformations.ps1
temp
transformations \ bronze_to_silver
silver.event_taxi_trip_green.yaml
silver.event_taxi_trip_green.yaml.assessment.md
silver.event_taxi_trip_yellow.yaml
silver.event_taxi_trip_yellow.yaml.assessment.md
silver.payment_type_ref.yaml
silver.payment_type_ref.yaml.assessment.md
silver.rate_code_ref.yaml
silver.rate_code_ref.yaml.assessment.md
silver.taxi_zone_ref.yaml
silver.taxi_zone_ref.yaml.assessment.md
silver.trip_type_ref.yaml
silver.trip_type_ref.yaml.assessment.md
silver.vendor_ref.yaml
silver.vendor_ref.yaml.assessment.md
VL_NYC_Taxi.VariableLibrary
.gitignore
.pylintrc
azure-pipelines.yml
dbml-error.log
README.md
temp-validation.json
test-validation.json
validation-results.json
OUTLINE
TIMELINE
Chat
silver.dbml
gold.dbml
CHAT
SESSIONS
Instructions for Creating a Pull Request
Completed in 1 mins
5 mins
CI for YAML Transformation Validation
Completed in 1 mins
41 mins
Bronze to Silver Transformations Implementation
Completed in 23s
45 mins
More
gold.dbml
silver.dbml
Describe what to build
Agent
Claude Sonnet 4.5
Local
Default Approvals
PROBLEMS
OUTPUT
QUERY RESULTS
TERMINAL
PORTS
DEBUG CONSOLE
No problems have been detected in the workspace.
silver.dbml
Describe what to build
Agent
Claude Sonnet 4.5
Local
Default Approvals
0:00

DBML/YAML AI Artifacts and Validations




Fabric
LH_NYC_Taxi
PL_2100_SILVER_Transform_From_Bronze
PL_3100_GOLD_Transform_From_Silver
Search
Create deployment pipeline
Create app
Manage access
Workspace settings
Source control
Filter by keyword
Filter
Name
Status
Git status
ENV_NYC_Taxi
Synced
LH_NYC_Taxi
Synced
LH_NYC_Taxi
—
NB_1000_BRONZE_Download_Taxi_Data
Synced
NB_1001_BRONZE_Download_Reference_Data
Synced
NB_2000_SILVER_Configure
Synced
NB_2001_SILVER_Seed_Reference_Tables
Synced
NB_3000_GOLD_Configure
Synced
NB_Create_CDC_State
Synced
NB_Enable_Change_Data_Feed
Synced
NB_Transform_Medallion_Layers
Synced
PL_1100_BRONZE_Load_Reference_Data
Synced
PL_1110_BRONZE_Load_Green_Trip_Data
Synced
PL_1120_BRONZE_Load_Yellow_Trip_Data
Synced
PL_2100_SILVER_Transform_From_Bronze
Synced
PL_3100_GOLD_Transform_From_Silver
Synced
VL_NYC_Taxi
Synced
Owner
Refreshed
Next refresh
Endorsement
Sensitivity
Included in app

Transformation Execution and AI Log Analysis

FDT - NYC Taxi - Dev - Eugene > NB_Transform_Medallion_Layers > NB_Transform_Medallion_Layers_149efc8d-cea1-4f45

Refresh | Stop application | Monitor run series | Spark History Server

Jobs | Resources | Logs | Data | **Item snapshots**

Item snapshots

Snapshot: NB_Transform_Medallion_Layers

```
================================================================================

▶ Stage 1/2: Starting 6 transformation(s) in parallel...
[1/7] Starting (parallel): bronze.green_trip_data -> silver.event_taxi_trip
[2/7] Starting (parallel): bronze.payment_types -> silver.payment_type_ref
[3/7] Starting (parallel): bronze.rate_codes -> silver.rate_code_ref
[4/7] Starting (parallel): bronze.taxi_zones -> silver.taxi_zone_ref
[4/7] ✓ Completed (parallel) in 86335ms: bronze.taxi_zones -> silver.taxi_zone_ref
[5/7] Starting (parallel): bronze.trip_types -> silver.trip_type_ref
[3/7] ✓ Completed (parallel) in 87014ms: bronze.rate_codes -> silver.rate_code_ref
[6/7] Starting (parallel): bronze.vendors -> silver.vendor_ref
[2/7] ✓ Completed (parallel) in 98904ms: bronze.payment_types -> silver.payment_type_ref
[5/7] ✓ Completed (parallel) in 61461ms: bronze.trip_types -> silver.trip_type_ref
[6/7] ✓ Completed (parallel) in 60834ms: bronze.vendors -> silver.vendor_ref
[1/7] ✓ Completed (parallel) in 212411ms: bronze.green_trip_data -> silver.event_taxi_trip

▶ Stage 2/2: Starting 1 transformation(s) in parallel...
[7/7] Starting (parallel): bronze.yellow_trip_data -> silver.event_taxi_trip
[7/7] ✓ Completed (parallel) in 698634ms: bronze.yellow_trip_data -> silver.event_taxi_trip
✓ Parallel execution completed: 7 transformations
[SUCCESS] Transformation pipeline completed successfully
import scala.collection.mutable.ArrayBuffer
success: Boolean = true
outputBuffer: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer()
res85: Any = ()
```

FDT Pipeline Log Analysis Report

Log File: fdt-LH_NYC_Taxi-execute-2026-03-12-160506.log
Pipeline Type: Silver → Gold Transformations
Execution Date: March 12, 2026 at 16:05:10

Pre-Analysis Validation Checklist

```
✅ PRE-ANALYSIS VALIDATION:
├─ ✅ Reference guide: FDT-Pipeline-Log-Analysis-Guide.md
├─ ✅ Guide size: 2435 lines total
├─ ✅ Read progress: 2435/2435 = 100%
├─ ✅ Percentage >= 100%: YES
├─ ✅ Key methodologies extracted: Stage analysis, table metrics, integrity checks
├─ ✅ Extraction patterns documented: PowerShell patterns from guide
├─ ✅ Template structure understood: Yes
└─ ✅ Ready to proceed: YES
```

Pipeline Overview

Metric	Value
Total Transformations	8
Total Stages	2
Total Duration	587,625ms (9.8 minutes)
Total Rows Processed	315,642,876 rows
Status	✅ SUCCESS
Configuration File	`/lakehouse/default/Files/transformations/ecp/config.json`
YAML Directory	`/lakehouse/default/Files/transformations/ecp/silver_to_gold`
Execution Mode	Parallel (maxParallelJobs=4)

Runtime Environment:

- Scala: 2.12.18
- Spark: 3.5.5.5.4.20260211.1
- Java: 11.0.27 (Microsoft)
- Executor Memory: 56g
- Driver Memory: 56g
- Executor Cores: 8

Performance Analysis

Transformation Duration Breakdown

Fastest Transformations (<30s):

1. dim_vendor: 26.6s (2 rows)
2. dim_trip_type: 29.1s (2 rows)
3. dim_taxi_zone: 34.5s (265 rows)

Slowest Transformation:

1. **fact_taxi_trip: 439.4s** (315.6M rows) - Expected given volume
2. **dim_date: 139.8s** (2,589 rows) - Complex transformations (26 steps)

Throughput Analysis

Table	Rows	Duration	Throughput
fact_taxi_trip	315,640,219	439s	**~720K rows/sec**
dim_date	2,589	140s	~18 rows/sec
dim_payment_type	6	82s	<1 row/sec
dim_rate_code	6	82s	<1 row/sec
dim_taxi_type	2	94s	<1 row/sec

Note: Low throughput on dimension tables is expected due to:

- Small row counts (overhead dominates)
- Validation and integrity check execution time
- First-time table creation operations

⚠️ Issues and Warnings

None Detected ✅

- No ERROR log entries
- No WARN log entries
- No failed validation rules
- No failed integrity checks
- No orphaned foreign key references
- No CDC issues
- No concurrent append warnings

What The Validators Actually Check

Real Examples of Issues We Catch

Fabric Artifact Validation Report

Validation Time: 2026-03-10 21:22:08 UTC

❌ Validation Failed

Metric	Count
Total Artifacts	7
✅ Passed	74
❌ Failed	6
⚠️ Warnings	22

⚠️ Pylint (Code Quality)

Type	Count
❌ Errors	0
⚠️ Warnings	45
📝 Conventions	315

❌ VariableLibrary: `VL_NYC_Taxi`

DelightStakeholders

- ✅ **Variable types defined**: All variables have explicit types
- ✅ **Value set 'ECP' configured**: Value set 'ECP' has 1 override(s)

MakeItMaintainable

- ⚠️ **Variables are documented**: Add notes/descriptions to variables for clarity
- ✅ **Meaningful variable names**: Variable names are descriptive
- ✅ **Consistent naming convention**: Variables follow PascalCase convention
- ❌ **Has description**: Add description in .platform metadata.description field

❌ Pipeline: `PL_1100_BRONZE_Load_Reference_Data`

DelightStakeholders

- ⚠️ **Error handling configured**: Add error handling (on failure dependencies, notifications)
- ⚠️ **Retry policies configured**: Consider adding retry policies for transient failures
- ⚠️ **Monitoring and logging**: Consider adding logging/monitoring for observability

✅ Notebook: `NB_Transform_Medallion_Layers`

DelightStakeholders

- ✅ **Has data quality checks**: Found: Validation functions
- ✅ **Has logging/output**: Logging or output statements found
- ⚠️ **Has error handling**: Add try/except blocks for better error handling

MakeItMaintainable

- ✅ **Has description**: Description found in .platform file
- ✅ **Display name matches folder name**: Display name 'NB_Transform_Medallion_Layers' matches folder name
- ⚠️ **Lakehouse configuration**: Has %%configure cell but also has default_lakehouse in metadata - remove metadata defaults
- ✅ **Has documentation markdown**: Markdown documentation with Purpose/Data Source/Target found
- ✅ **Functions have docstrings**: No functions defined or inline code
- ✅ **Meaningful variable names**: Variable names appear descriptive

✅ YAML Transformation Validation

Transformation Files: 15 | ✅ Valid: 15 | ❌ Invalid: 0 ✅ All transformation files validated successfully against DBML schemas

✅ DBML: `gold`

DelightStakeholders

- ✅ **Has data lineage**: Data lineage information found in notes
- ✅ **Has validation rules**: Validation rules documented in notes
- ✅ **Has NOT NULL constraints**: NOT NULL constraints defined for data quality
- ✅ **Medallion layer alignment**: Tables use expected schema: gold

MakeItMaintainable

- ✅ **Has Project documentation**: Project Note documentation found
- ✅ **Tables have documentation**: All tables have Note documentation
- ✅ **Column notes include metadata**: 91.5% of columns have Source/Transformation/Conformance metadata
- ✅ **Consistent table naming**: All tables use schema prefix (e.g., bronze.table_name)

MakeItScale

- ✅ **Primary keys defined**: All tables have primary keys defined
- ✅ **Has index definitions**: Index definitions found
- ✅ **Has relationships defined**: Table relationships (Ref) defined

MakeItSecure

- ✅ **No hardcoded credentials**: No hardcoded credentials detected

MakeItWork

- ✅ **DBML file exists**: DBML file found
- ✅ **Not empty**: DBML file has content
- ✅ **Has Project declaration**: Project block found
- ✅ **Has Table definitions**: Table definitions found
- ✅ **DBML syntax valid**: DBML syntax validated successfully with @dbml/cli

⚠️ Semantic Model Best Practices

Models Analyzed: 1 | **Total Violations:** 55

⚠️ SM_NYC_Taxi

❌ Errors: 0 | ⚠️ Warnings: 55 | ℹ️ Info: 0

- ⚠️ **[Hygiene] Hide unused columns**: Column 'fact_taxi_trip'[pickup_datetime] violates rule "[Hygiene] Hide unused columns"n- ⚠️ **[Hygiene] Hide unused columns**: Column 'dim_date'[first_day_of_quarter] violates rule "[Hygiene] Hide unused columns"n- ⚠️ **[Hygiene] Hide unused columns**: Column 'dim_date'[last_day_of_quarter] violates rule "[Hygiene] Hide unused columns"n- ⚠️ **[Hygiene] Hide unused columns**: Column 'dim_date'[days_in_quarter] violates rule "[Hygiene] Hide unused columns"n- ⚠️ **[Hygiene] Hide unused columns**: Column 'dim_date'[first_day_of_year] violates rule "[Hygiene] Hide unused columns"n- ⚠️ **[Hygiene] Hide unused columns**: Column 'dim_date'[last_day_of_year] violates rule "[Hygiene] Hide unused columns"n- ⚠️ **[Hygiene] Hide unused columns**: Column 'dim_date'[is_leap_year] violates rule "[Hygiene] Hide unused columns"n- ⚠️ **[Hygiene] Hide unused columns**: Column 'dim_date'[days_in_year] violates rule "[Hygiene] Hide unused columns"n- ⚠️ **[Hygiene] Hide unused columns**: Column 'dim_date'[fiscal_year] violates rule "[Hygiene] Hide unused columns"n- ⚠️ **[Hygiene] Hide unused columns**: Column 'dim_date'[fiscal_quarter] violates rule "[Hygiene] Hide unused columns"n ... *and 45 more violations. See build artifacts for full report.*

Full BPA report available in build artifacts: `bpa-results.json`

DEMO 2 – Generate and Execute CD

- Create and configure CD Pipeline (1:14)
- Execute CD Pipeline (1:50)

File Edit Selection View Go Run Terminal Help

Untitled (Workspace)

5

SOURCE CONTROL

CHANGES

Message (Ctrl+Enter to commit on "feature/2...

Commit

Chat

CHAT

SESSIONS

Resolving BPA Warnings in Code
Completed in 5 mins 2 hrs

PowerShell script error in semantic model validation
Completed in 26s 2 hrs

Duplicate 'condition' definition in Azure Pipelines YAML
Completed in 1 mins 1 day

More

GRAPH

Auto

Merged PR 7514: Impleme... feature/22812-cr...

Merged PR 7501: Implement FDT-Powered Medallion...

Merged PR 7496: Add Complete Bronze-Silver-Gold ...

Merged PR 7495: Add DBML Schema Validation Fram...

Merged PR 7494: Add Bronze Layer Data Pipelines an...

Merged PR 7484: Configure GitHub Copilot for AI-As...

Merged PR 7415: Implemented #22778 - draft projec...

Added README.md Eugene Paden

Describe what to build

Agent Claude Sonnet 4.5

Local Default Approvals

PROBLEMS OUTPUT QUERY RESULTS TERMINAL PORTS DEBUG CONSOLE

Filter (e.g. text, **/*.ts, !**/nod...

No problems have been detected in the workspace.

Describe what to build

Agent Claude Sonnet 4.5

Local Default Approvals

CD Pipeline Definition

feature/22812-create-deployment-yaml-pipeline / azure-pipelines-cd.yml

azure-pipelines-cd.yml

Contents History Compare Blame

```yaml
# Continuous Deployment Pipeline for Microsoft Fabric
# Pipeline name: fdt-nyc-taxi-cd
# Implements Git-integrated deployment strategy with:
# - Build version tracking across all environments (date-based: YYYYMMDD.Rev)
# - Unique release branch per deployment (release/{env}/{version})
# - Variable library substitution with actual artifact IDs
# - Approval gates for controlled promotion

# Build number format: YYYYMMDD.r where r auto-increments per day
# Example: 20260314.1, 20260314.2, 20260314.3, ...
name: $(Date:yyyyMMdd)$(Rev:.r)

trigger:
  branches:
    include:
    - main

# Disable PR triggers - this is a deployment pipeline
pr: none

pool:
  vmImage: 'windows-latest'

# Note: buildVersion is calculated dynamically in the Build stage
# Format: YYYYMMDD.Rev or YYYYMMDD.Rev.PRNumber (e.g., 20260314.31 or 20260314.31.7514)

variables:
  # Optional: Shared service principal credentials (if not using per-environment creds)
  - group: Fabric-Shared-Auth
  # Azure service connection for Fabric API authentication
  # Create this in Azure DevOps: Project Settings > Service connections > New service connection > Azure Resource Manager
  # If not set, will use service principal from Fabric-Shared-Auth variable group
  - name: azureSubscription
    value: ''  # Set to your Azure service connection name (e.g., 'Fabric-Deployment')

stages:
```

feature/22812-create-deployment-yaml-pipeline / templates / deploy-stage.yml

deploy-stage.yml

Contents History Compare Blame

```yaml
# Deployment Stage Template for Microsoft Fabric Workspaces
# This template handles deployment to a single environment (CI, DEV, QA, or PROD)

parameters:
  - name: environmentName
    type: string
  - name: environmentDisplayName
    type: string
  - name: azureDevOpsEnvironment
    type: string
  - name: variableGroup
    type: string
  - name: deployBranchPrefix
    type: string
  - name: azureSubscription
    type: string
    default: ''
  - name: dependsOn
    type: object
    default: ['Build']
  - name: cleanValueSets
    type: boolean
    default: true

stages:
- stage: Deploy${{ parameters.environmentName }}
  displayName: 'Deploy to ${{ parameters.environmentDisplayName }}'
  dependsOn: ${{ parameters.dependsOn }}
  condition: succeeded()
  variables:
    - group: ${{ parameters.variableGroup }}
    - name: buildVersion
      value: $[ stageDependencies.Build.CreateBuildVersion.outputs['VersionInfo.buildVersion'] ]
    - name: environmentName
      value: '${{ parameters.environmentName }}'
    - name: deployBranchPrefix
      value: '${{ parameters.deployBranchPrefix }}'
```

Implement Fabric CD Pipeline with Git Integration and Environment Publishing

Active !7515 Eugene Paden proposes to merge feature/22812-create-deployment-yaml-pipeline into main

Approve | Set auto-complete

Overview | Files | Updates | Commits

1 of 3 required checks running now
1 optional check not yet run

fdt-nyc-tax-ci Build in progress

View 4 checks

Eugene Paden must approve

No merge conflicts
Last checked Just now

Description

Summary

Implements a comprehensive CI/CD pipeline for Microsoft Fabric workspace deployments with Git integration, automated artifact management, and environment publishing capabilities.

Changes Made

🚀 CD Pipeline Infrastructure

- **azure-pipelines-cd.yml**: Multi-stage deployment pipeline (CI → DEV → QA → PROD)
- **templates/deploy-stage.yml**: Reusable deployment stage template with environment-specific configuration
- **Documentation**: Complete setup guide (docs/cd-pipeline-setup.md) and deployment checklist (docs/cd-pipeline-checklist.md)

🔧 Core Deployment Scripts

- **deploy-to-fabric.ps1**: Full Git integration automation
  - Service principal authentication
  - Workspace Git connection management (connect/disconnect/initialize)
  - Branch switching with proper reconnection logic
  - UpdateFromGit API with PreferRemote policy
  - Environment artifact publishing with LRO polling
- **substitute-variable-libraries.ps1**: Intelligent artifact ID substitution
  - Matches variable IDs to workspace artifacts by name (e.g., LakehouseId → LH_NYC_Taxi)
  - Type-specific matching (Lakehouse vs SQLEndpoint)
  - Sets DeployedVersionId to build version
  - Removes valueSets folders and clears settings.json
  - Commits changes to Git and syncs workspace via UpdateFromGit API
- **sync-workspace-from-git.ps1**: Workspace synchronization helper
  - Uses actual commit hashes from Git Status API

Reviewers

Add

Required

Eugene Paden
No review yet

Optional

No optional reviewers

Tags

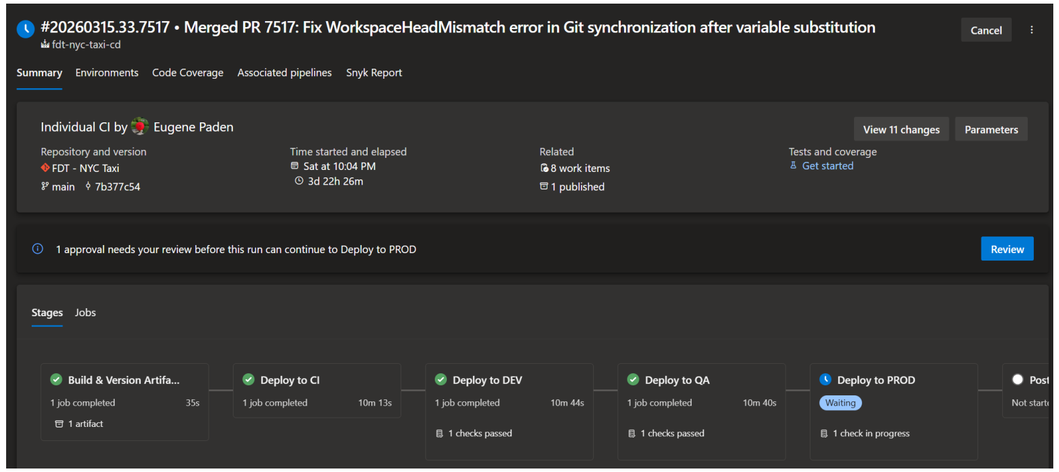
No tags

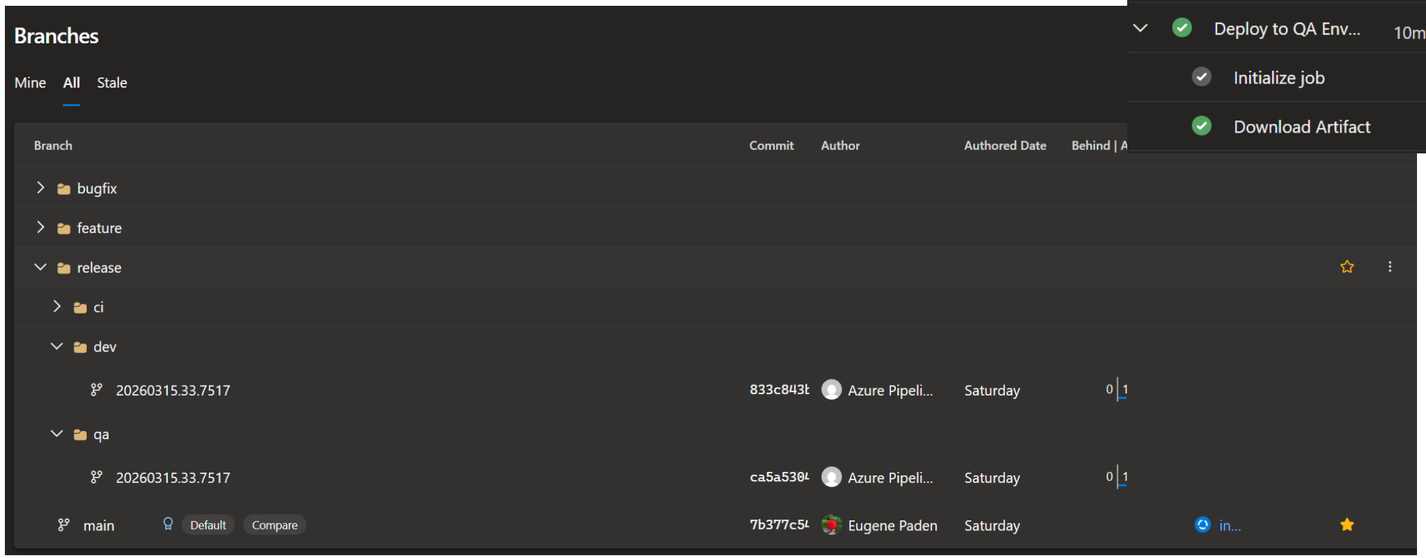
Work items

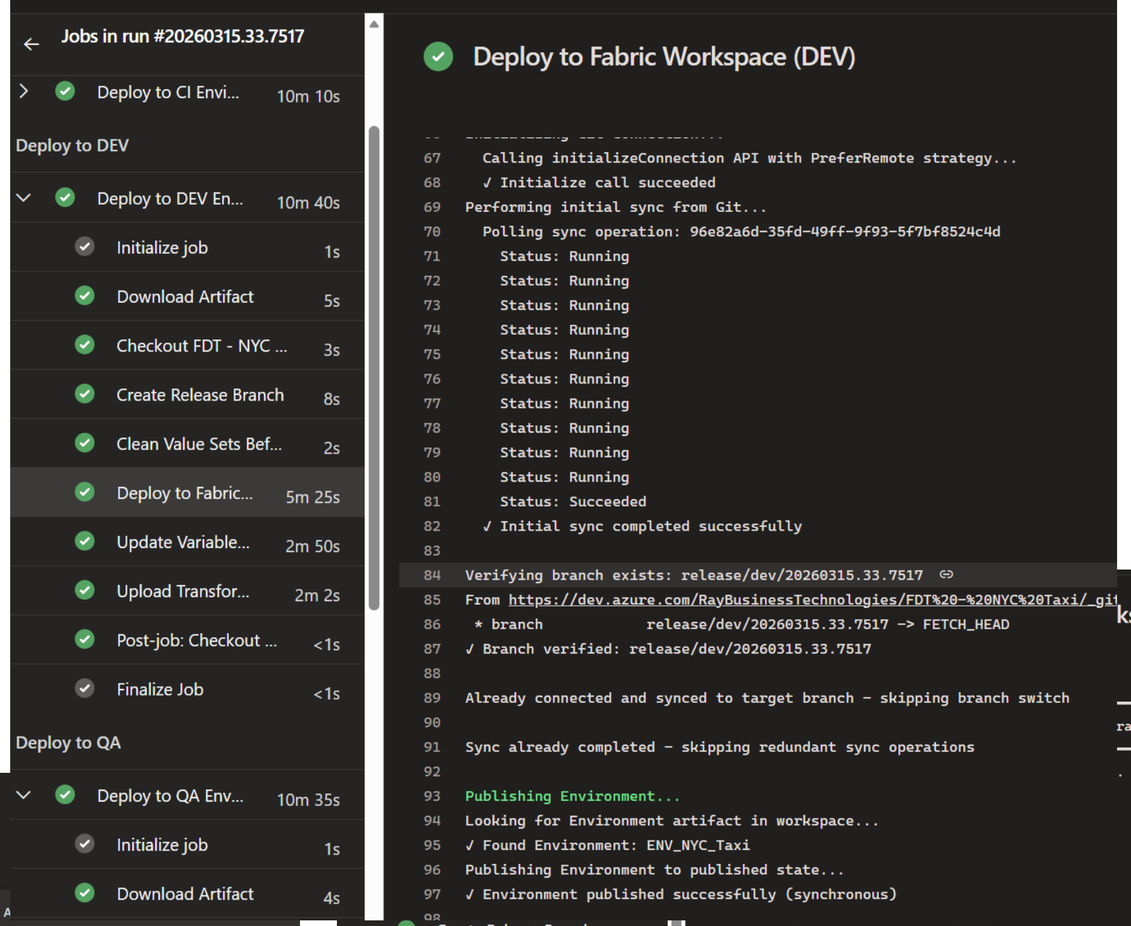
Task 22812: Create deployment YAML
Updated Just now, To Do

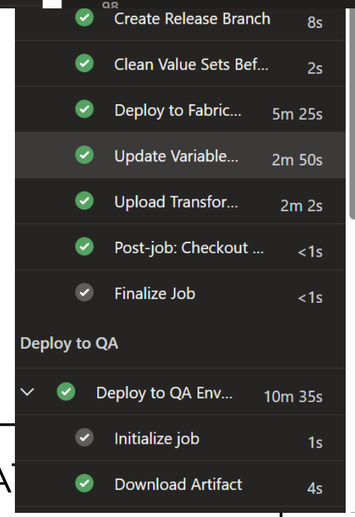
0:00

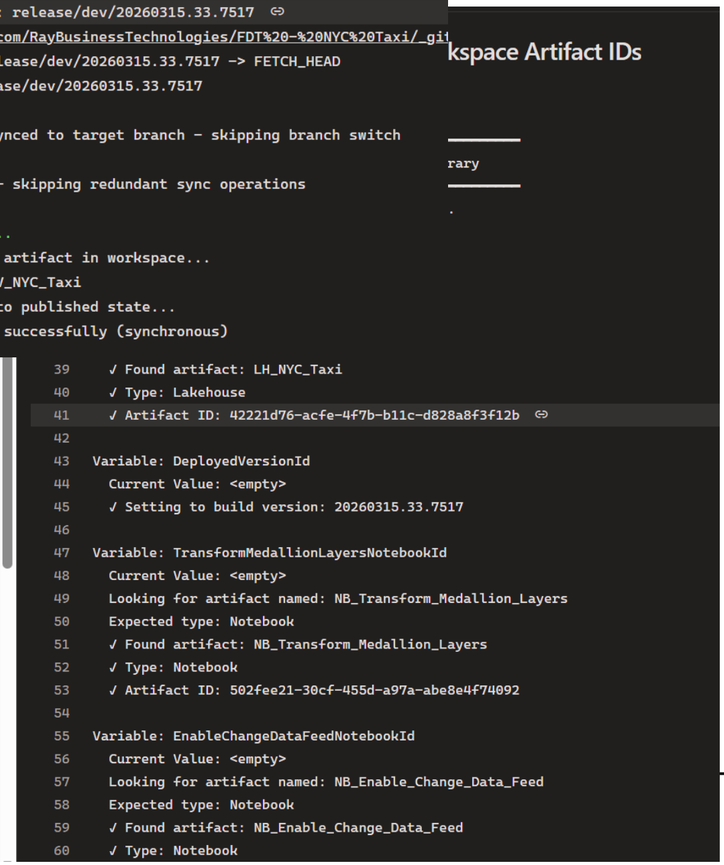
CD Pipeline Execution





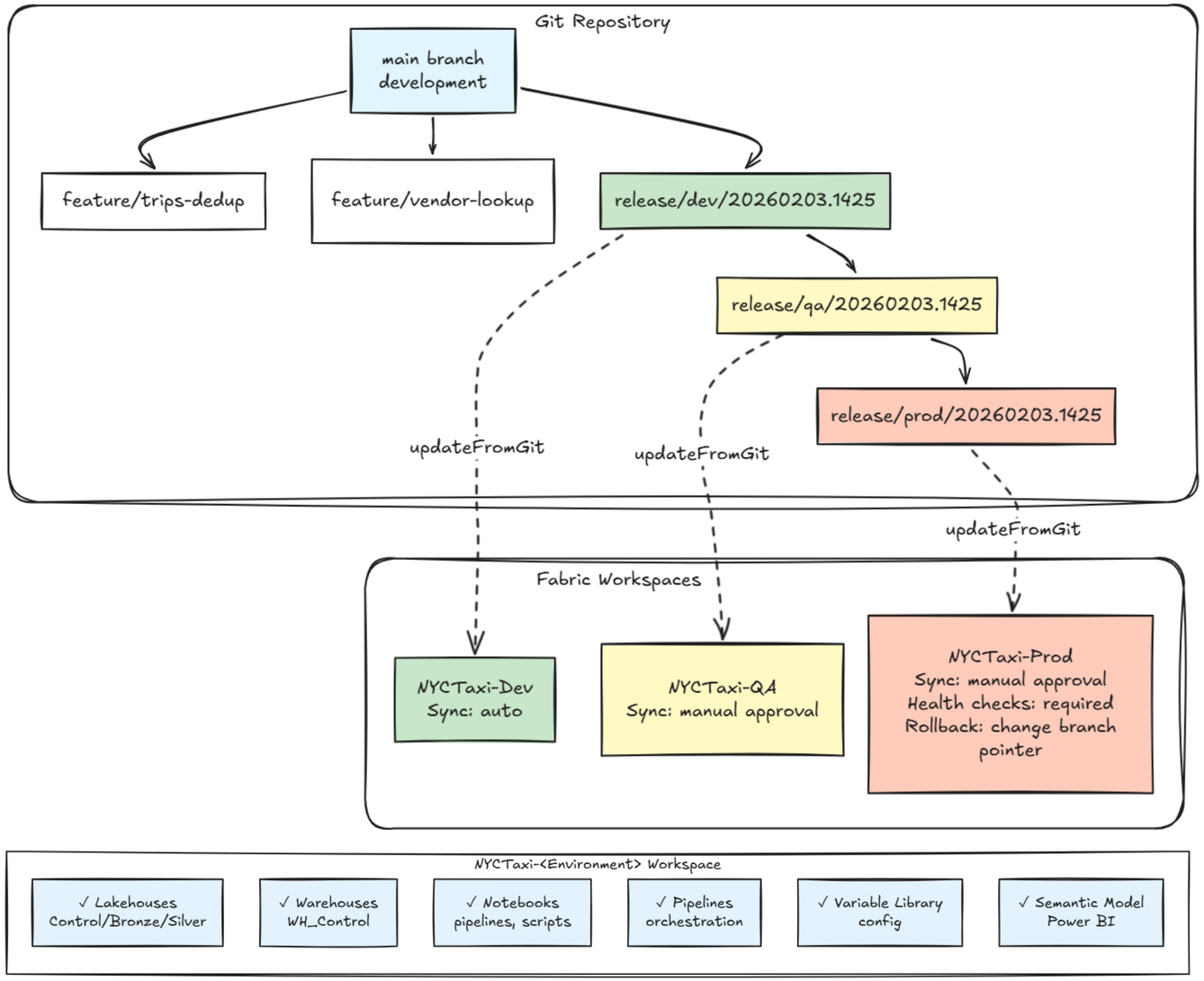






The Git Branch Magic How This Actually Works

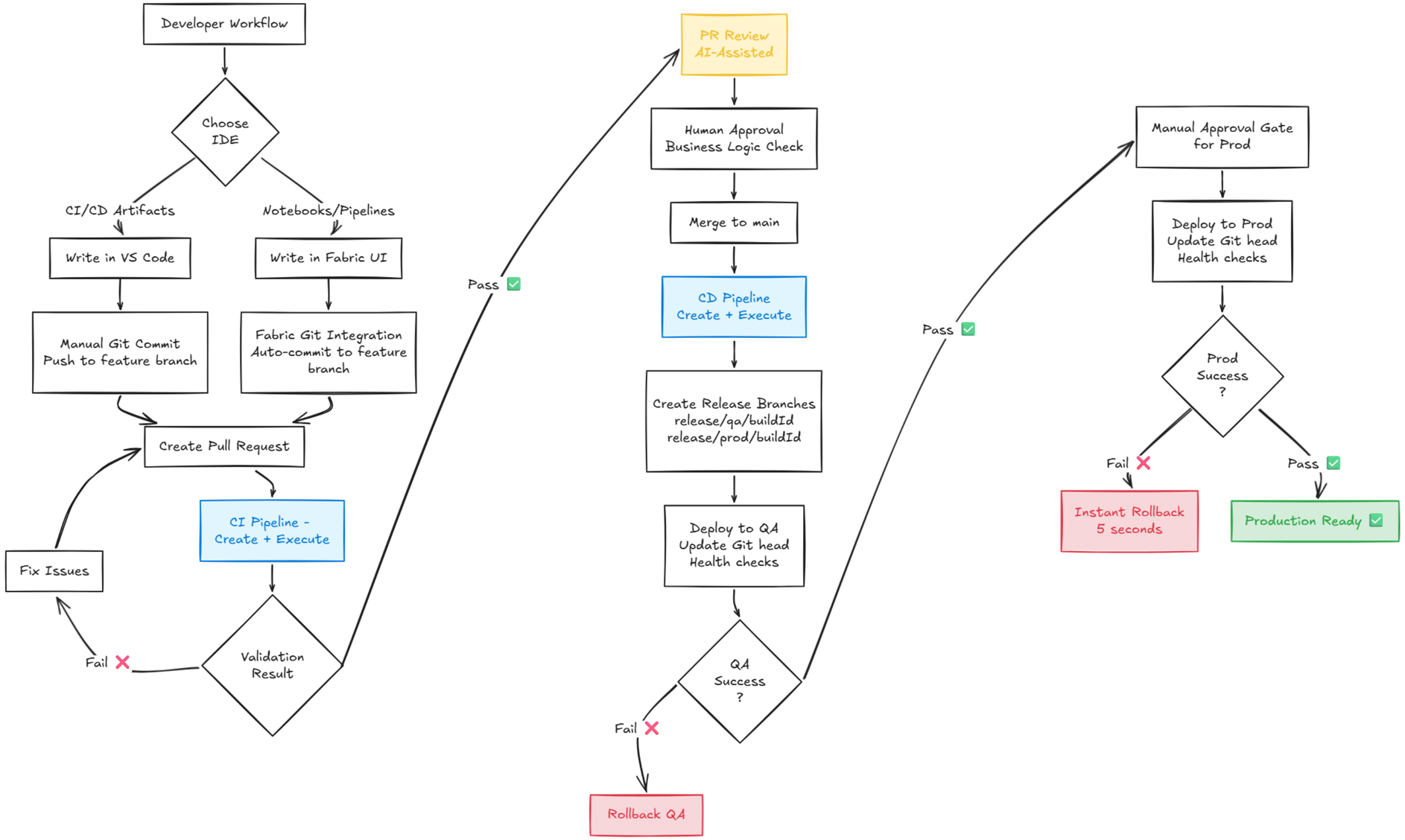
Behind the Scenes: Fabric + Git Integration



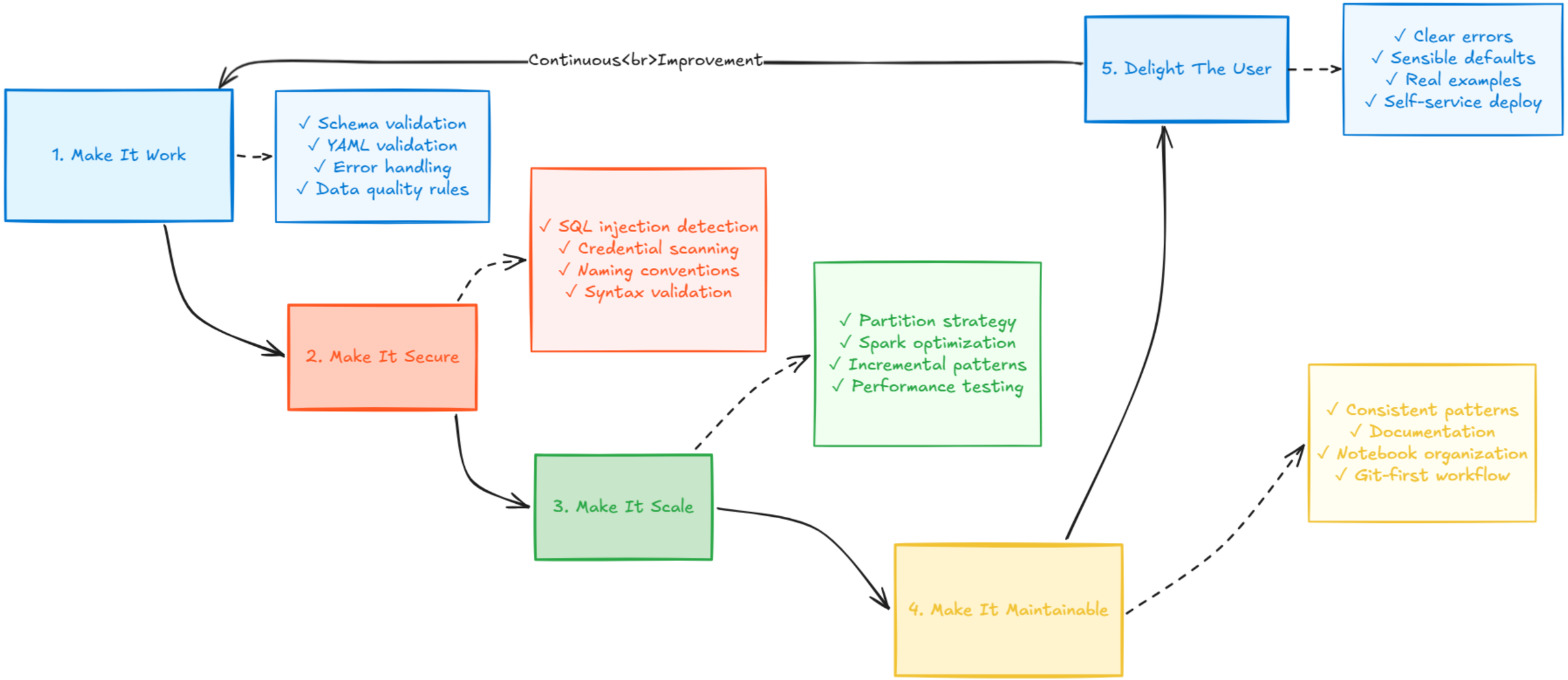
Why This Matters (Architecture Trade-offs)

- **Complete isolation** No cross-environment contamination
- **Instant rollback** Change Git branch pointer
- **Zero downtime** Atomic updates (all resources or none)
- **Audit trail** Git history tracks all changes
- **Workspace duplication** 3x storage (Dev, QA, Prod)
- **Branch proliferation** Old releases archived monthly
- **Capacity planning** Each workspace needs separate capacity (Dev/QA can share)

Putting It All Together - The Complete Pipeline



The 5-Dimension Quality Framework (Your New Checklist)



The Stuff You Need to Remember

If You Forget Everything Else, Remember These

- **Validate in CI, not production -** catch errors before they break stuff
- **Let Copilot write your CI/CD -** 5x faster, better quality
- **Deploy with Git branches -** zero downtime, instant rollback
- **Security from day one -** no creds in code, validate everything
- **Start small, build iteratively -** don't try to do everything at once
- **Use YAML, not code -** for transformations (easier to maintain, version, and review)

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!