EUROPEAN MICROSOFT FABRIC Community Conference

STOCKHOLM 24-27 SEPTEMBER 2024

JOIN THE CONVERSATION

#FABCONEUROPE

Taking the Fabric Development Experience to the next level – with CI/CD and Testing

Arjen Kroezen & Marc Lelijveld

Microsoft & Macaw - Netherlands

Learning objectives

Git	Deployment	Testing
Understand how to integrate Git with your Fabric Workspaces	Understand the different options to orchestrate and structure your deployment	Improve development experience by developing data pipelines faster and more reliable

Marc Lelijveld

Technical Evangelist | Solution Architect
Macaw Netherlands

Microsoft Most Valuable Professional

sessionize MOST ACTIVE SPEAKER 2022

sessionize MOST ACTIVE SPEAKER 2023

MarcLelijveld

linkedin.com/in/MarcLelijveld

Data-Marc.com

DutchFabricUsergroup.com

Arjen Kroezen

Senior Software Engineer
Microsoft

ArjenDev

linkedin.com/in/ArjenKroezen

Setting the scene

CI / CD

Continuous Integration

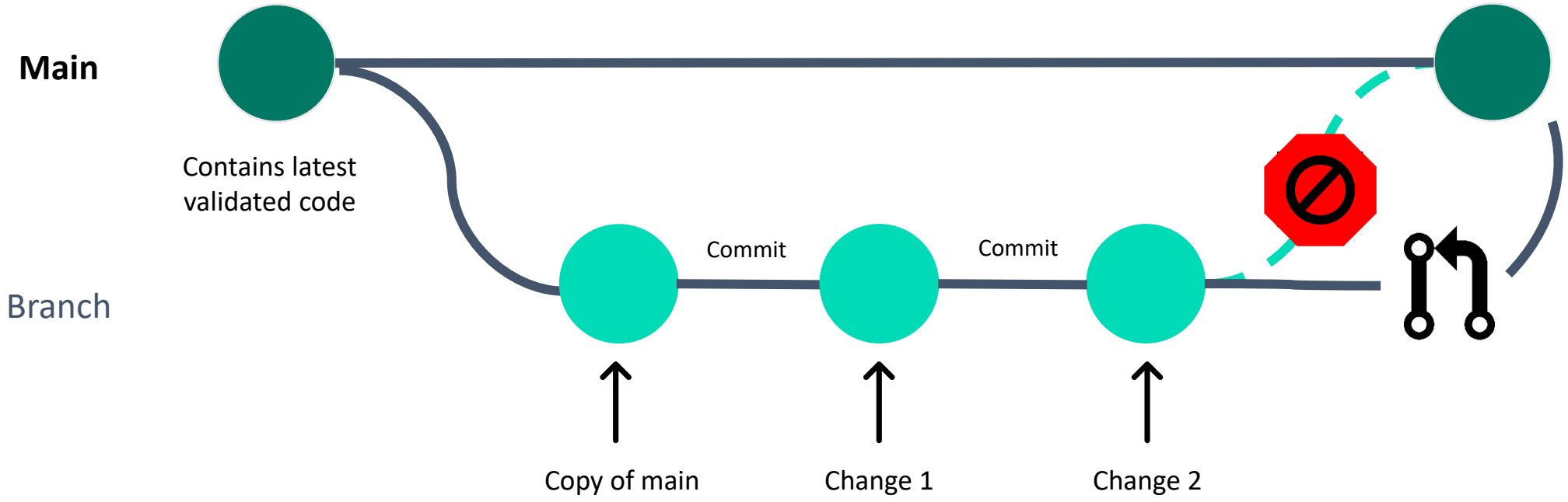
- Small but frequent changes
- Working with multiple people on various aspects of the solution
- Merging back into main branch

Continuous Deployment

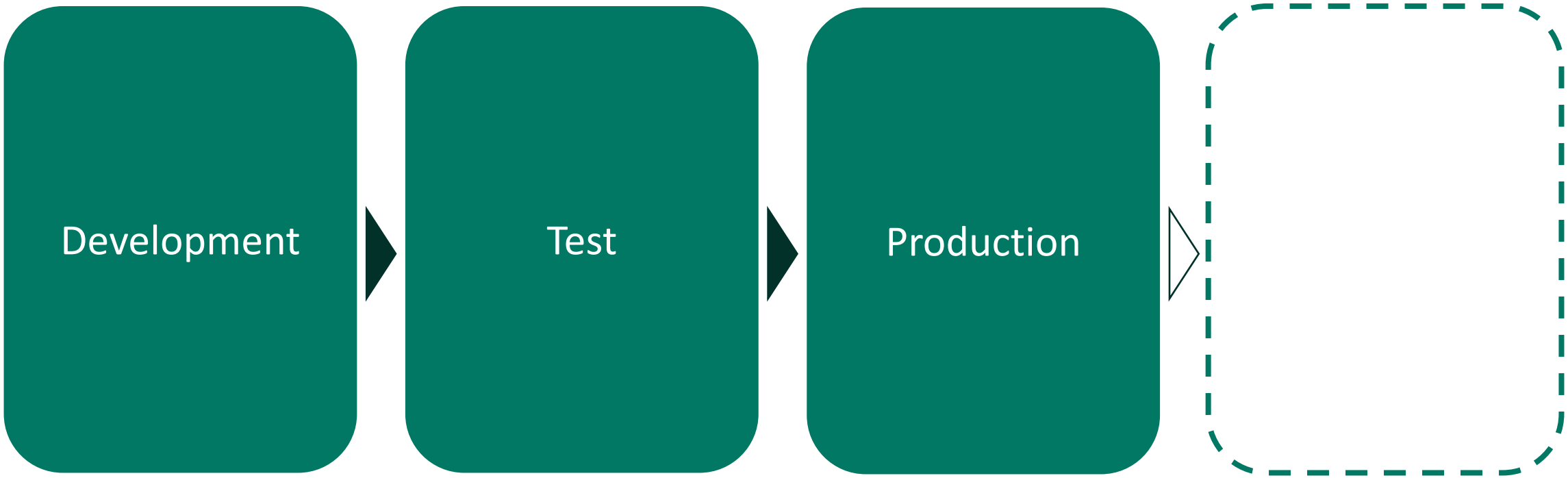
- Working in short cycles (often Agile)
- High frequency updates and releases
- Repeatable deployment process

This all, to improve the developer experience.

Git concept


Main
Contains latest validated code
Branch
Commit
Commit
Copy of main
Change 1
Change 2

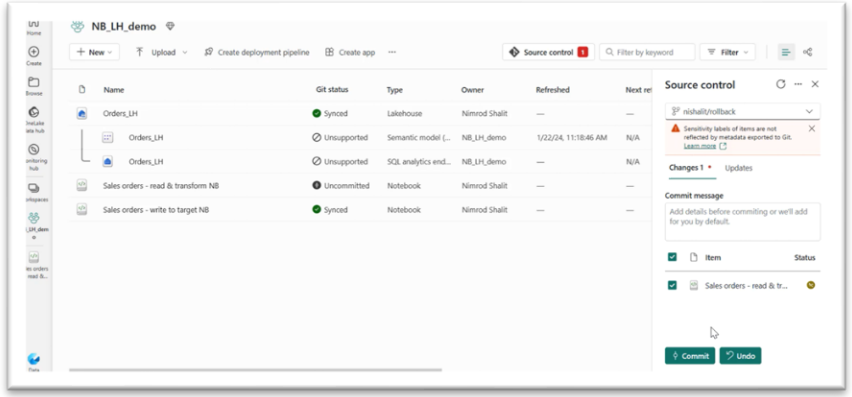
Staged approach



What do we have today?

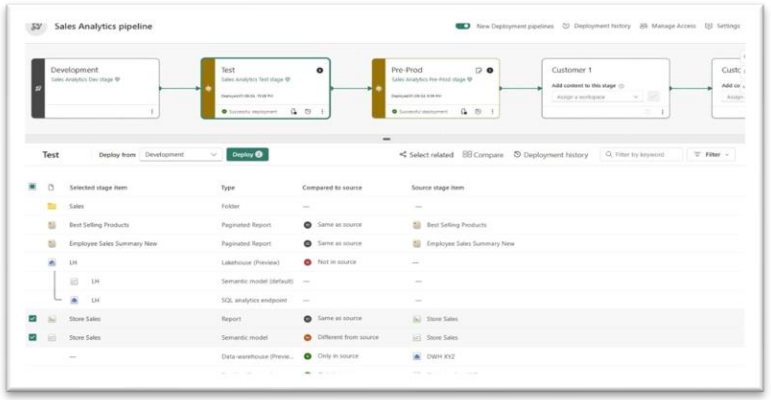
Fabric CI/CD platform

Built-in git integration



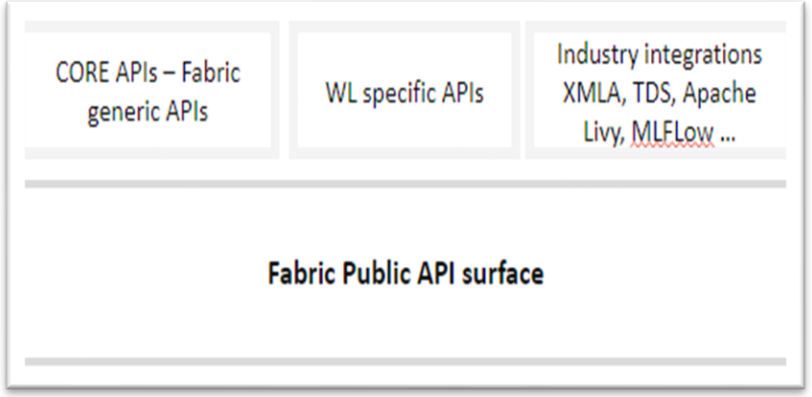
Public preview

Deployment pipelines



GA feature

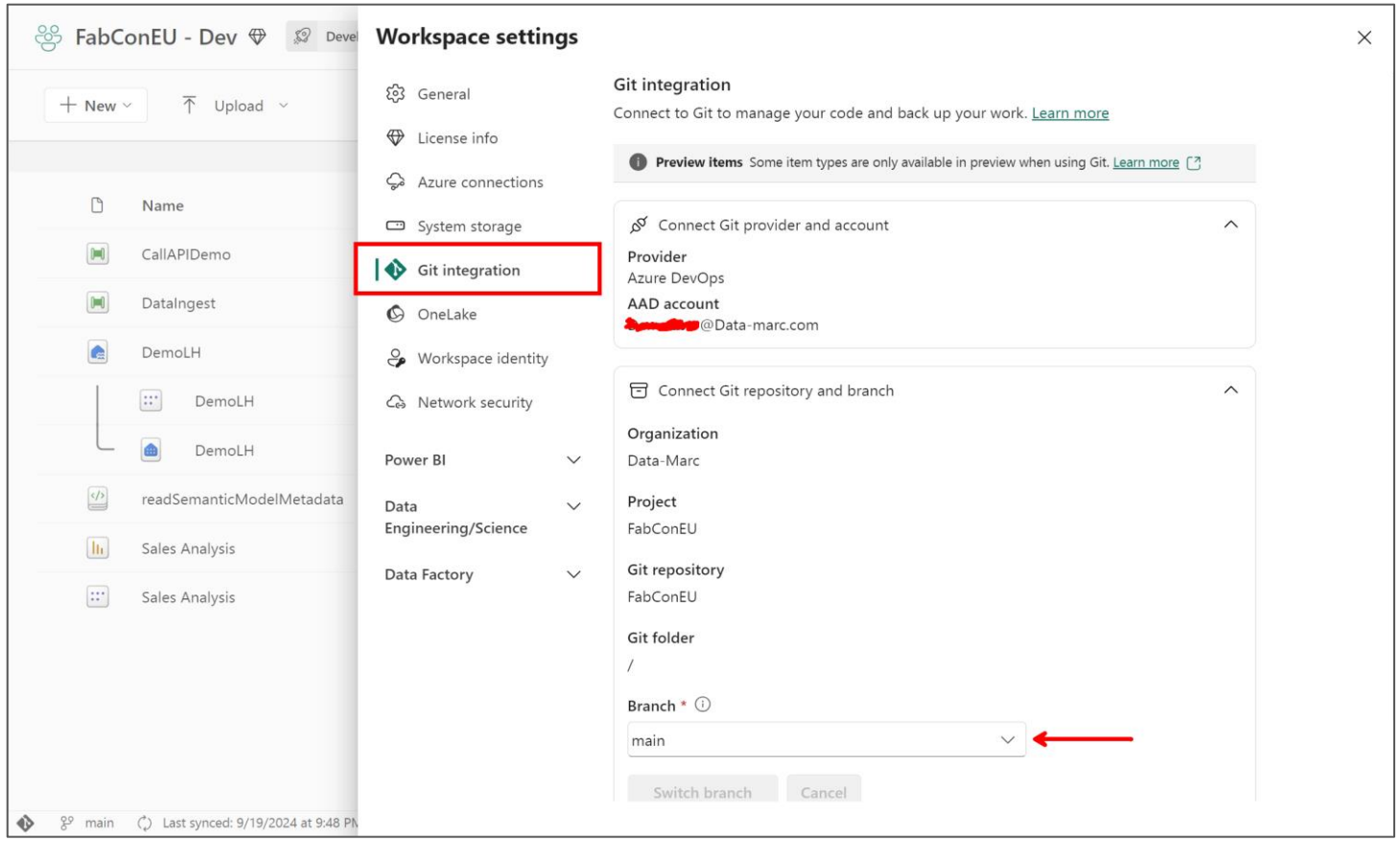
Fabric REST APIs



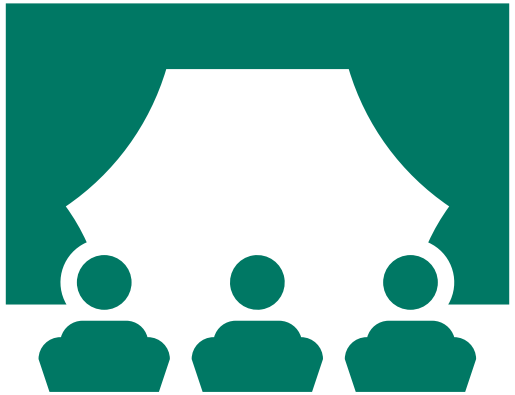
Public preview

Git integration

- Sync a Workspace to a Git branch
- Git providers
  - Azure DevOps
  - GitHub New
  - GitHub Enterprise New
- Fabric git APIs – REST APIs & PowerShell samples.
- Manage branches
  - Switch branch
  - Checkout new branch
  - Branch out to new workspace

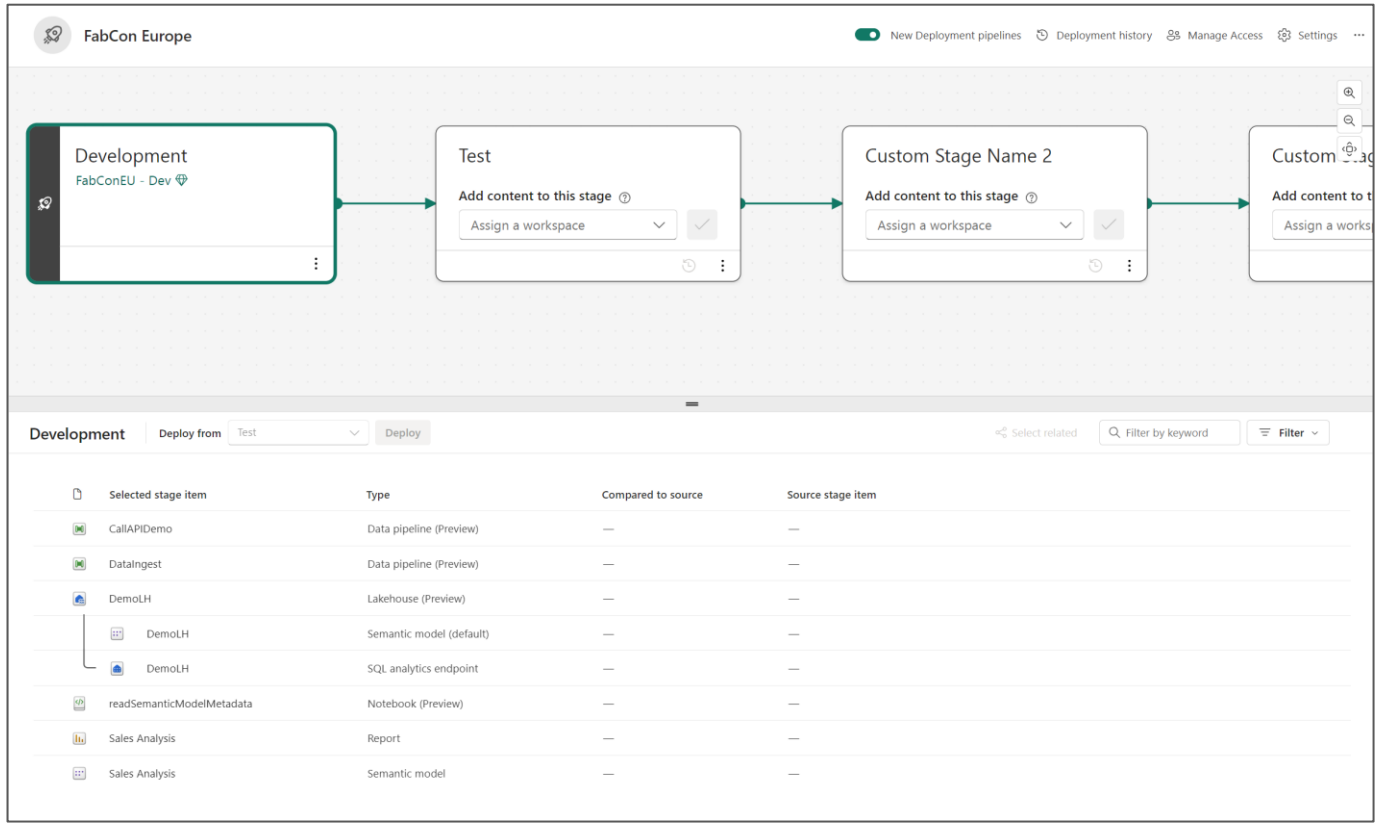


DEMO TIME!



Deployment pipelines

- Deploy items across Workspaces
- Apply rules on configuration
- Majority of Fabric items supported *(and more to come)*
- Compare changes on code-level *(only for semantic models)*
- Create a pipeline of 2-10 stages
  - Pipeline designer at creation
  - Ability to add custom stage names



Fabric User APIs

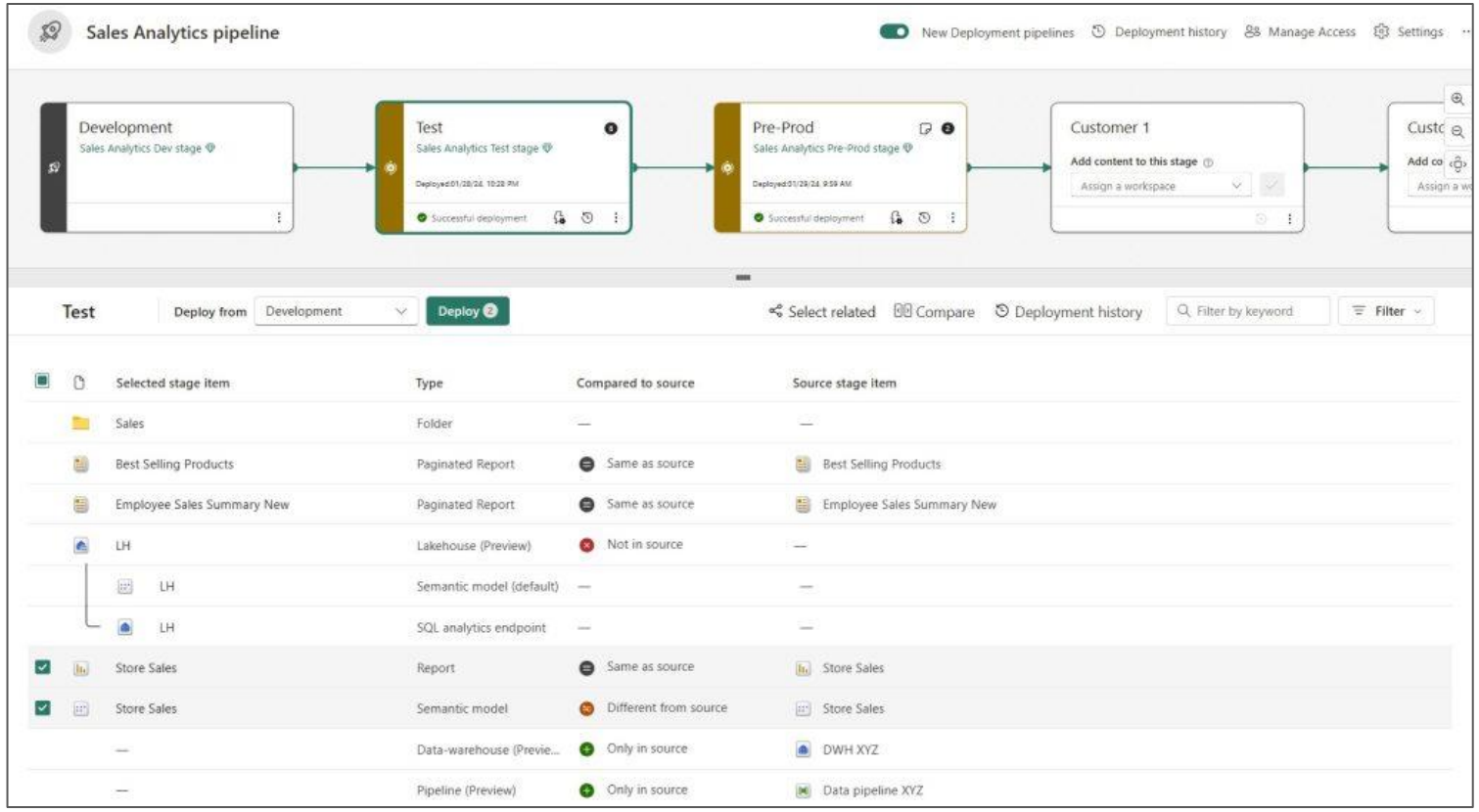
- Automated operations on behalf of Fabric users.
- Supporting CRUD operations
- Example usage scenarios:
  - Item management *(see table)*
  - Item definition
  - Workspace management
  - Workspace access management
  - Execute item jobs

Item type	Create (without definition)	Get	Update	Delete	List
Dashboard	✗	✗	✗	✗	✓
DataPipeline	✗	✓	✓	✓	✓
Datamart	✗	✗	✗	✗	✓
Eventhouse	✓	✓	✓	✓	✓
Eventstream	✓	✓	✓	✓	✓
KQLDatabase	✗	✓	✓	✓	✓
KQLQueryset	✓	✓	✓	✓	✓
Lakehouse	✓	✓	✓	✓	✓
MLExperiment	✓	✓	✓	✓	✓
MLModel	✓	✓	✓	✓	✓
MirroredWarehouse	✗	✗	✗	✗	✓
Notebook	✓	✓	✓	✓	✓
PaginatedReport	✗	✗	✗	✗	✓
Report	✗	✓	✗	✓	✓
SemanticModel	✗	✓	✗	✓	✓
SparkJobDefinition	✓	✓	✓	✓	✓
SQLEndpoint	✗	✗	✗	✗	✓
Warehouse	✓	✓	✓	✓	✓

New announcements

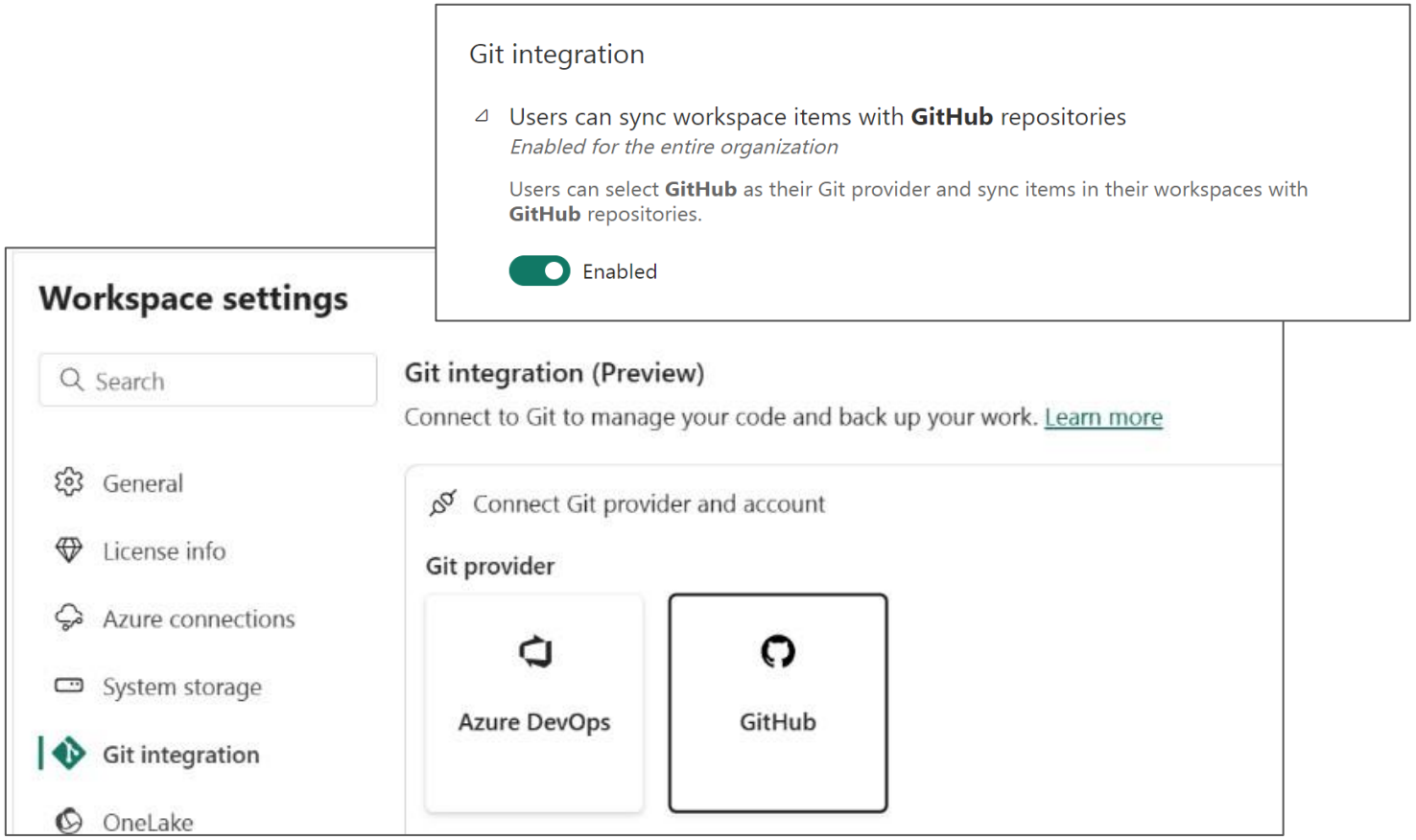
New UI for Deployment Pipelines

- Switch to enable/disable UI
- Easier navigate
- More focused (per stage)
- Smoother flow
- Folder structure
- Identify unsupported items



GitHub integration

- Second git provider next to Azure DevOps
  - GitHub + GitHub Enterprise
- Tenant admin explicitly must activate the feature
- Potential multi-geo restrictions not enforced
- API support coming soon



Generally available

Fabric Git integration

Supported items

- Data pipeline
- Lakehouse
- Warehouse
- Reflex New
- Report
- Paginated Report
- Semantic Model
- Real-Time Dashboard New
- Notebook
- Spark Job Definition
- Spark Environment
- Queryset New

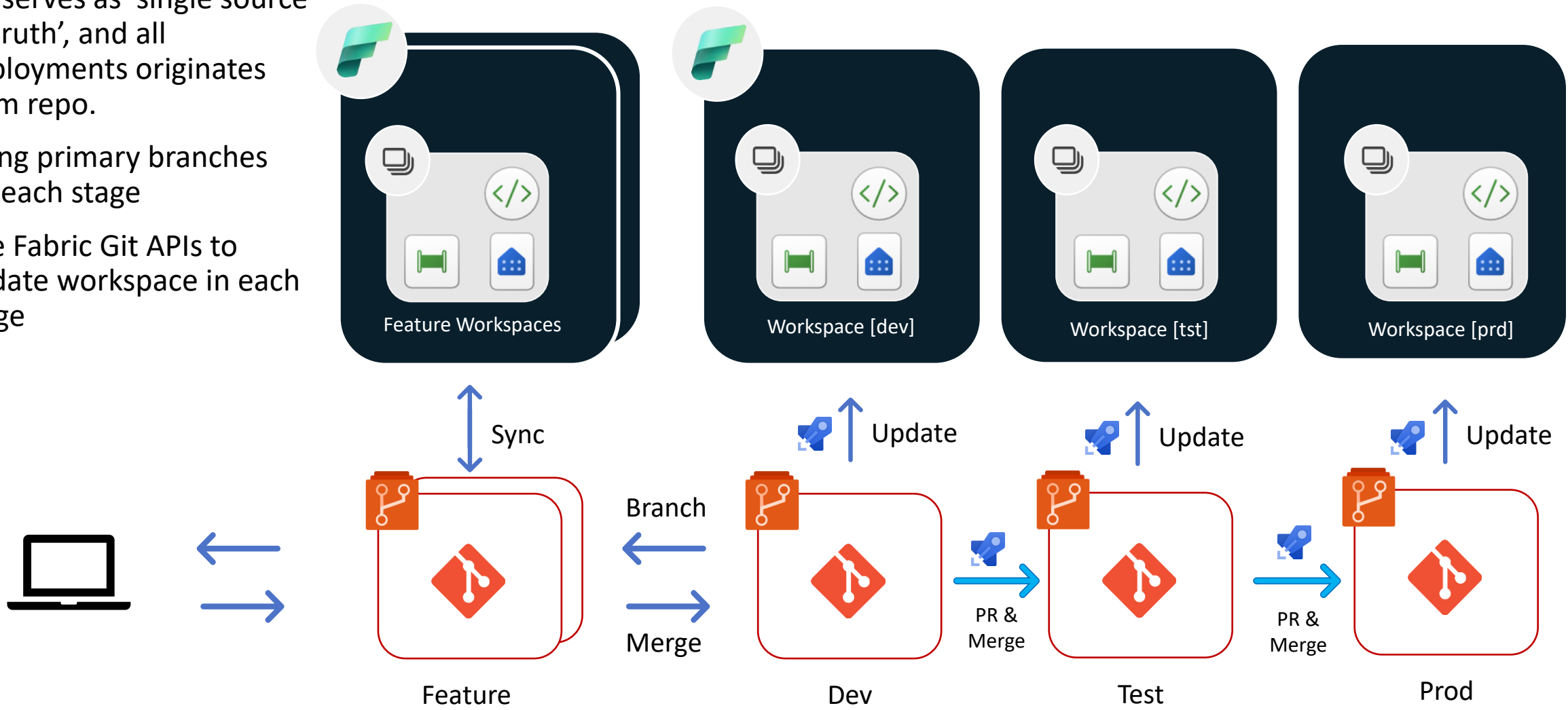
By end of the year

- Dataflow Gen2
- Org App
- Eventhouse
- Metrics Set
- Eventstream
- Mirrored Database
- GraphQL API

Deployment strategies

Scenario 1 – Git based deployments

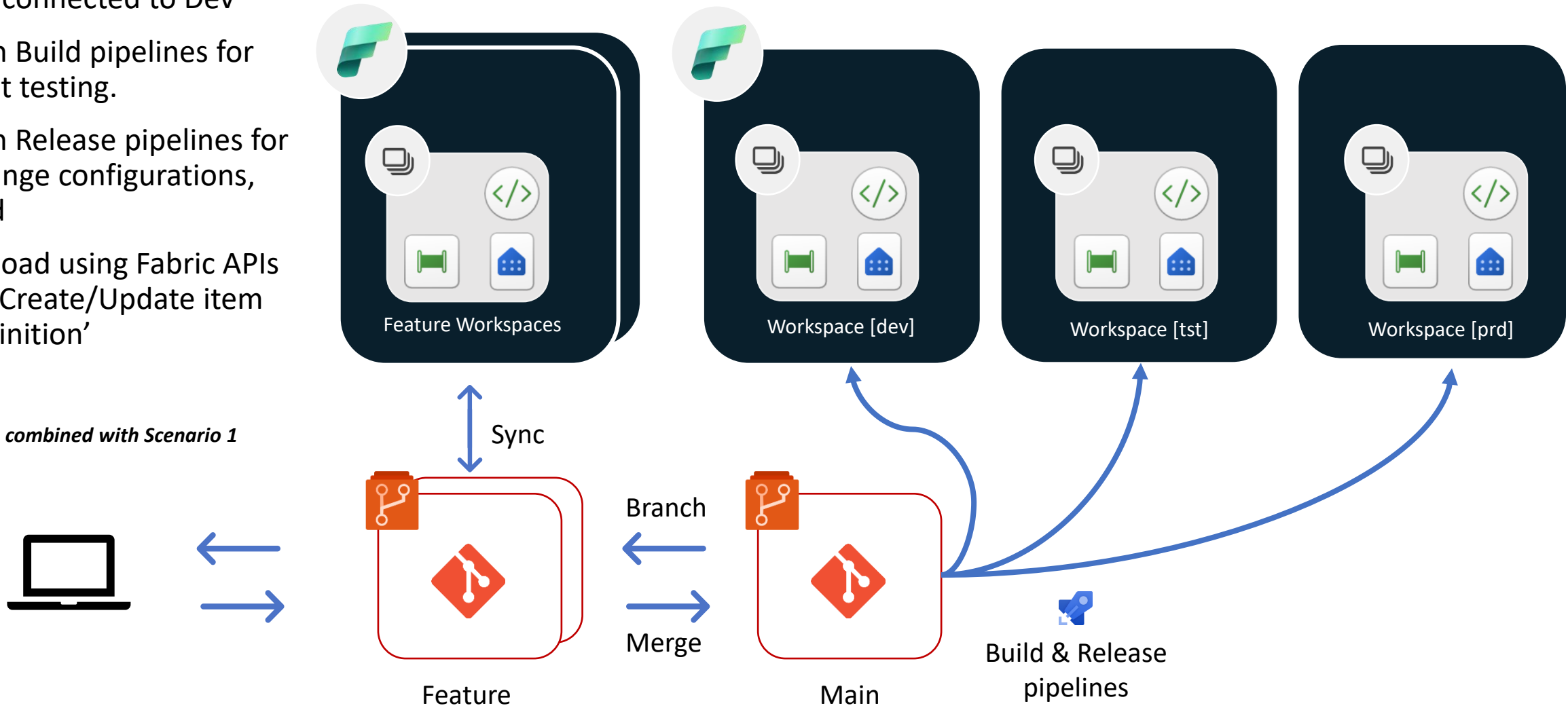
- Git serves as 'single source of truth', and all deployments originates from repo.
- Using primary branches for each stage
- Use Fabric Git APIs to update workspace in each stage



Scenario 2* – Git & Build environments

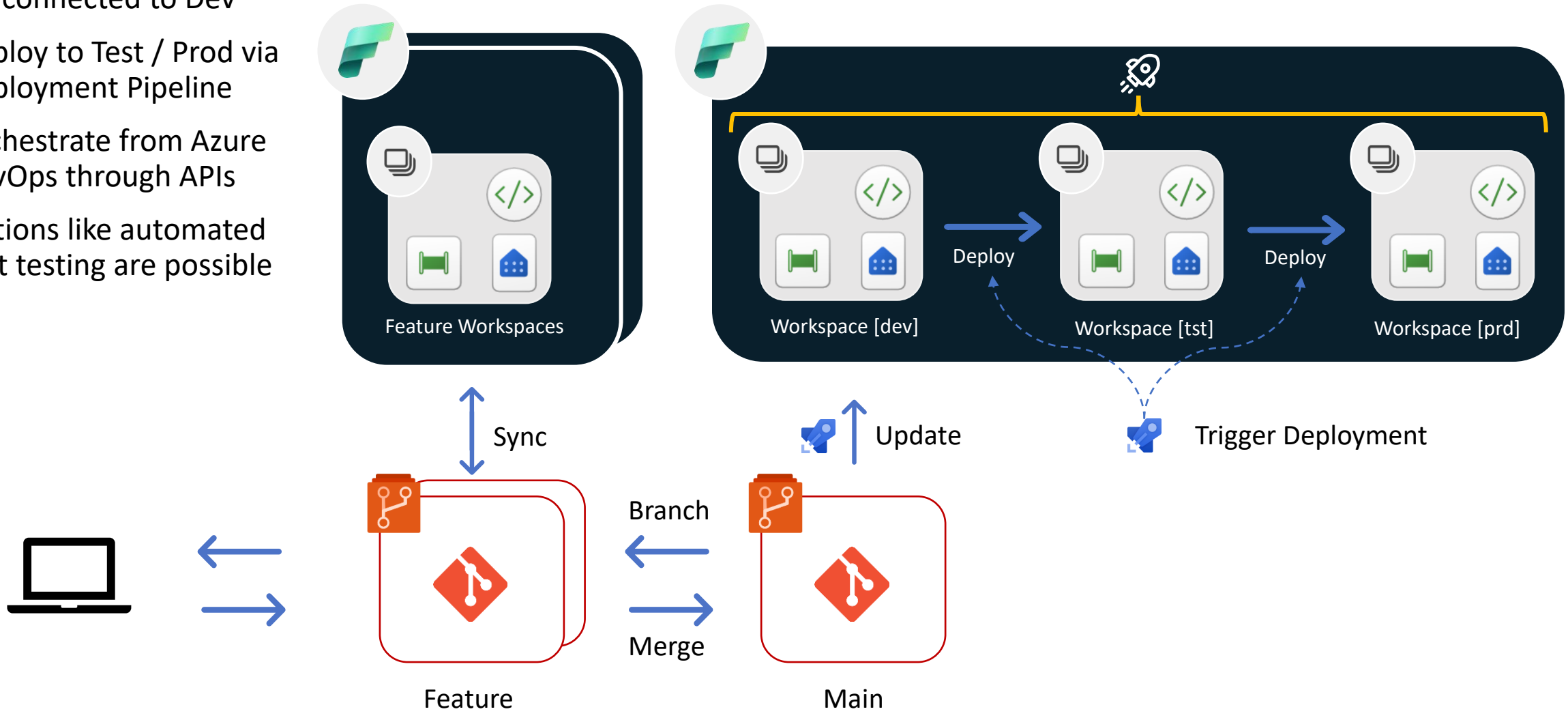
- Git connected to Dev
- Run Build pipelines for Unit testing.
- Run Release pipelines for change configurations, and
- Upload using Fabric APIs to 'Create/Update item definition'

**** Can be combined with Scenario 1***



Scenario 3 – Git & Deployment pipelines

- Git connected to Dev
- Deploy to Test / Prod via Deployment Pipeline
- Orchestrate from Azure DevOps through APIs
- Options like automated unit testing are possible

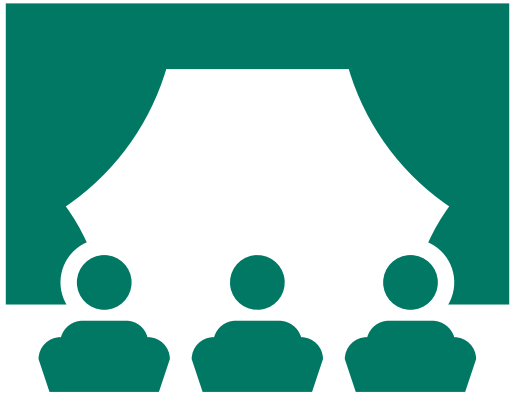


Opinionated view!

Entry point	Scenario	Code heaviness
Prefer an user interface?	Start with deployment pipelines interface, grow into **scenario 3** over time	Low
Comfortable working with git/code?	Your go-to scenario will be **scenario 1**.	Medium
Require a lot of customization?	Using the APIs in **scenario 2** allows you to customize everything to your wish	High

In greenfield scenario – opt for scenario 1!

DEMO TIME!



Taking it one step further…

Home Activities Run View

Export (Preview) Use a template (Preview) Validate Run

Web

Call API

General Settings

Connection * https://example.com arjenkroezen Refresh Edit

Relative URL @pipeline().parameters.BasePath

Method * GET

Current Development Experience using Data Pipelines in Fabric

Step 1 Build your pipeline in the UI

Step 2 Check if pipeline works

Step 3 Deploy pipeline through CI/CD

How would you check the pipeline works as expected?

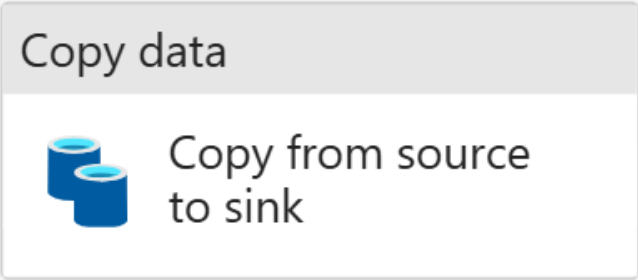
1. Validate and run pipeline manually
2. Validate programmatically with API’s

Typical example of testing Data Pipelines

Arrange — Prepare source storage

Act — Run pipeline

Assert — Validate sink storage



Different types of tests

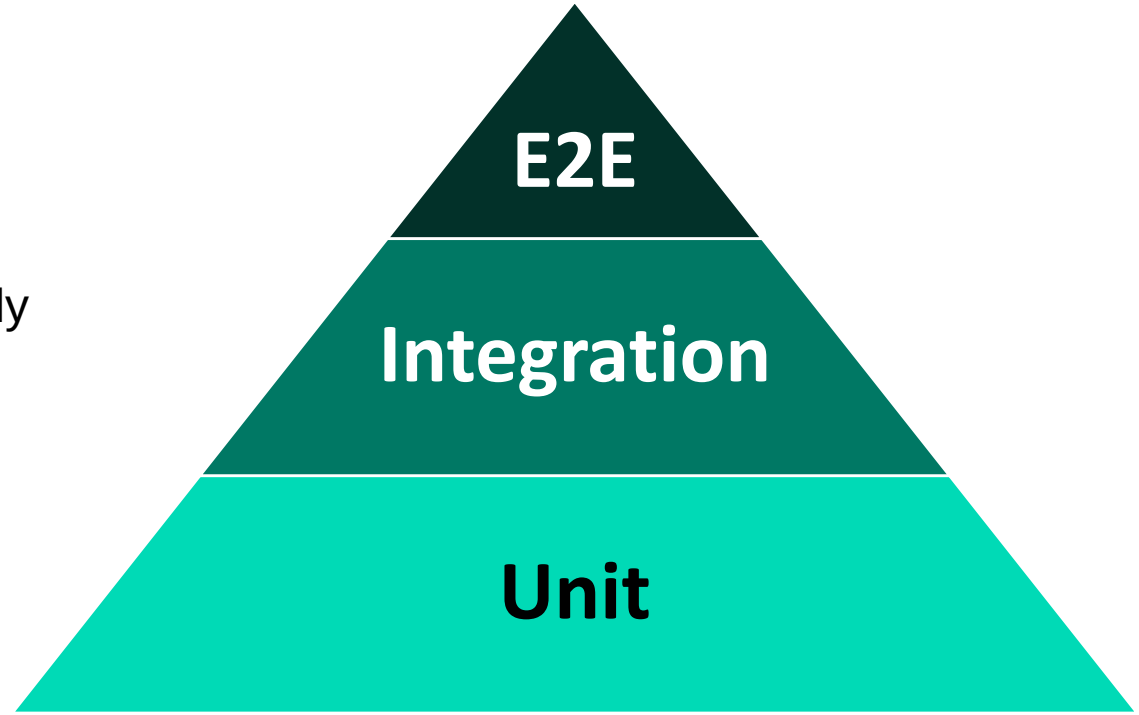
Previous example demonstrates a typical E2E test

Pros:

- Single test covers the entire pipeline
- Tests if external dependencies are integrated correctly

Cons:

- Slow to run and develop
- Requires manipulation of external dependencies
- Difficult to test all possible scenarios



Imagine a unit test for a data pipeline

Building pipelines is just like regular programming:

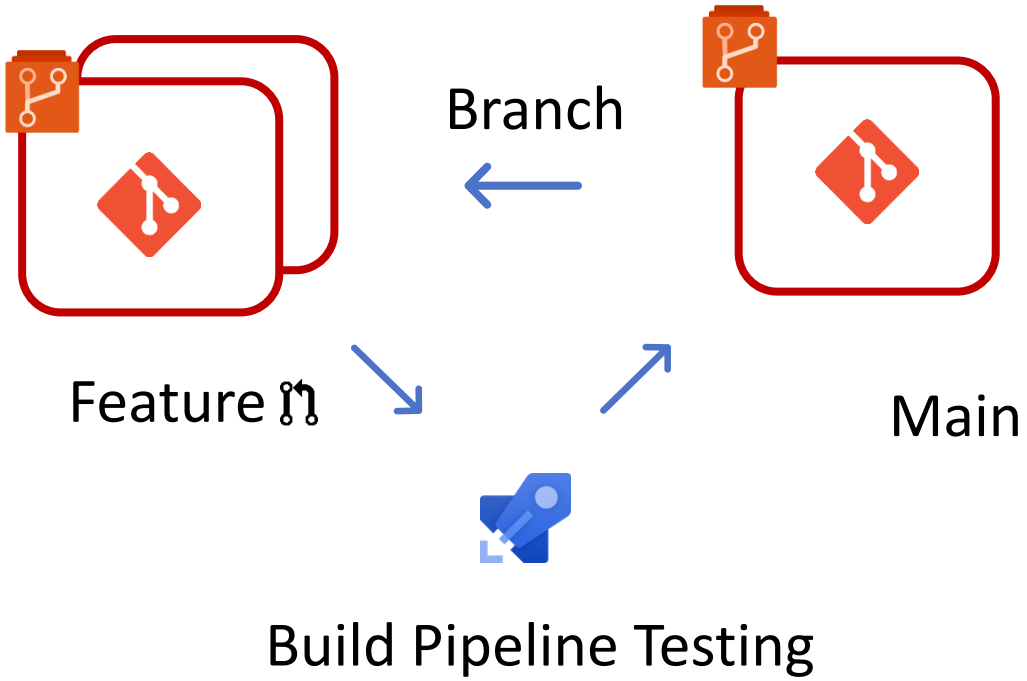
- Parameters
- Expressions (Domain-Specific Language)
- Activities
- Control Activities

A unit test would be able to:

- Evaluate and assert (complex) expressions
- Evaluate and assert execution flow of activities

Imagine unit tests integrated in our Development Experience

- Tests should be written and easily ran during development of the data pipeline
- Tests should be integrated in our CI/CD setup



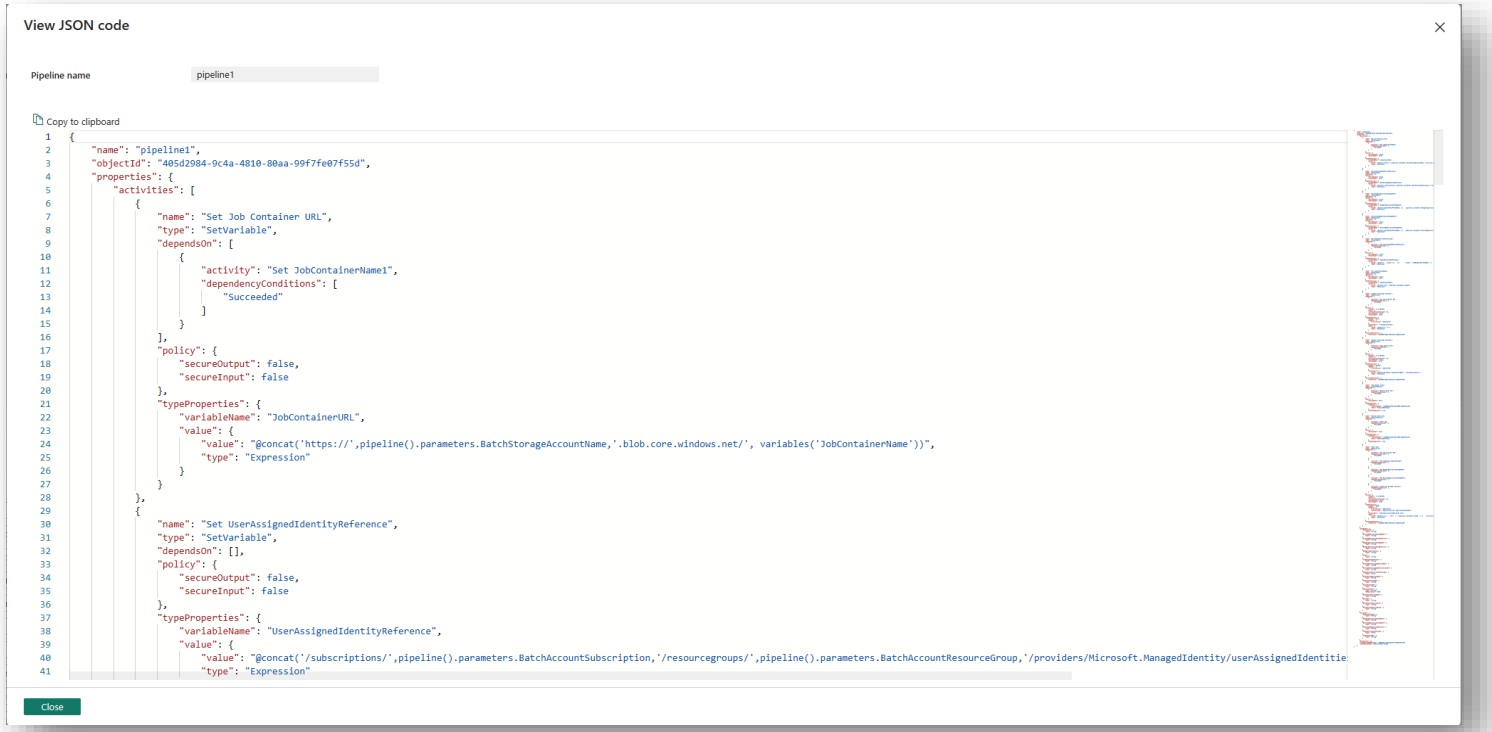
Introduction to Data Factory Testing Framework

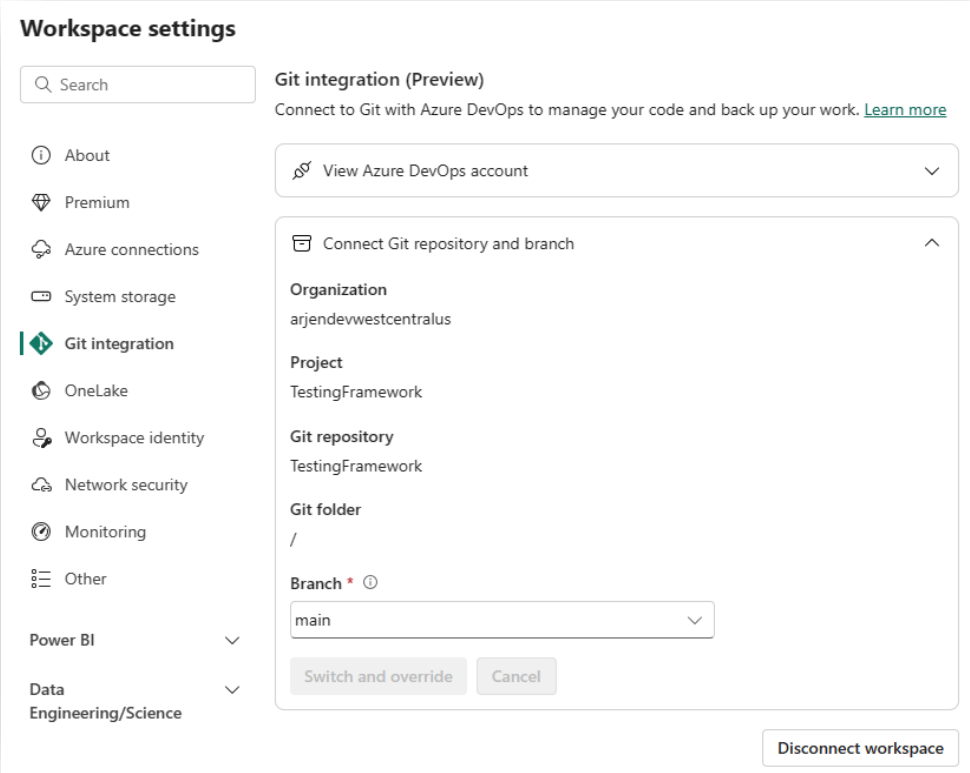
The framework understands the programming nature of Data Pipelines to allow you to write unit tests in Python

- Open source Python library available on PyPI (Python Package Index)
- Uses the pipeline definitions of Data Factory Data Pipelines
- Works on local development machine and in CI pipelines
- Supports Fabric, Azure Data Factory and Azure Synapse Analytics
- Disclaimer

Repository setup

- Sync pipeline JSON definition files with a git repository





Prerequisites

- Install the dotnet runtime 8.0 (not SDK) from here.
- Install Python 3.9+
- Setup Python environment
- Install the framework from PyPI: *data-factory-testing-framework*.

Initialize the framework

```
from data_factory_testing_framework import TestFramework

test_framework = TestFramework(
  framework_type=TestFrameworkType.Fabric,
  root_folder_path='/fabric’
)

pipeline = test_framework.get_pipeline_by_name("copy_pipeline")
activity = pipeline.get_activity_by_name("copy_data")
```

Activity testing – definition

Web

Start Job

```
{
    "name": "Start Job",
    "type": "WebActivity",
    "dependsOn": [],
    "typeProperties": {
        "relativeUrl": {
            "value": "@concat(pipeline().parameters.BasePath, '?api-version=', pipeline().parameters.ApiVersion)",
            "type": "Expression"
        },
        "method": "POST",
        "body": "{}"
    },
    "externalReferences": {
        "connection": "9a738523-63c3-40aa-b9f2-3c8d67cf3b86"
    }
}
```

Web

Start Job

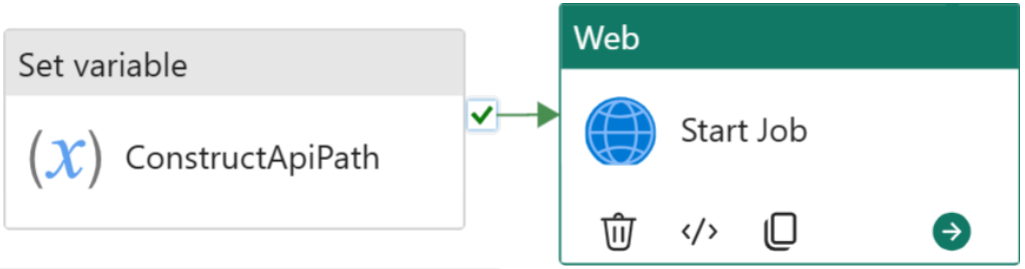
Activity testing

```
# Arrange
activity = pipeline.get_activity_by_name("Start Job")
state = PipelineRunState(
  parameters=[
    RunParameter(RunParameterType.Pipeline, "BasePath", "jobs"),
    RunParameter(RunParameterType.Pipeline, "ApiVersion", "2022-10-01.16.0"),
  ]
)

#Act
activity.evaluate(state)

# Assert
assert "jobs?api-version=2022-10-01.16.0" == activity.type_properties["relativeUrl"].result
```

Pipeline testing




```python
# Arrange
pipeline = test_framework.get_pipeline_by_name("batch_job")

# Act
activities = test_framework.evaluate_pipeline(pipeline, [
  RunParameter(RunParameterType.Pipeline, "BasePath", "jobs"),
  RunParameter(RunParameterType.Pipeline, "ApiVersion", "2022-10-01.16.0"),
])

# Assert
set_variable_activity = next(activities)
assert "ConstructApiPath" == set_variable_activity.name
assert "jobs?api-version=2022-10-01.16.0" == activity.type_properties["value"].result

post_job_activity = next(activities)
assert "StartJob" == post_job_activity.name
assert "jobs?api-version=2022-10-01.16.0" == post_job_activity.type_properties["relativeUrl"].result
assert "POST" == post_job_activity.type_properties["method"]

with pytest.raises(StopIteration):
  next(activities)
```

Summary

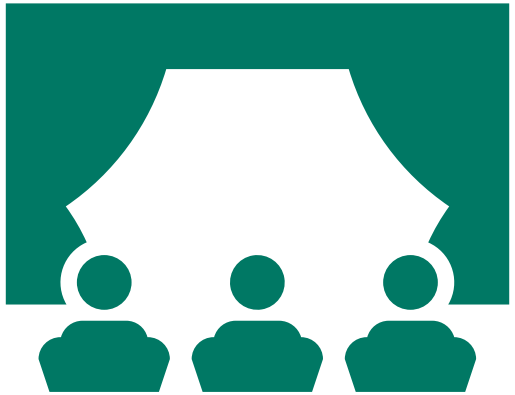
You **can** use the framework for:

- Expression language evaluation
- Test individual activities
- Test entire pipeline definition
- Quick iteration when developing pipelines
- Quality gate in continuous integration pipeline

You **cannot** use the framework for:

- Running the pipeline
- Build and run the tests in the Fabric UI

SECOND DEMO TIME!



Home - Microsoft Azure | CallAPIDemo - Power BI

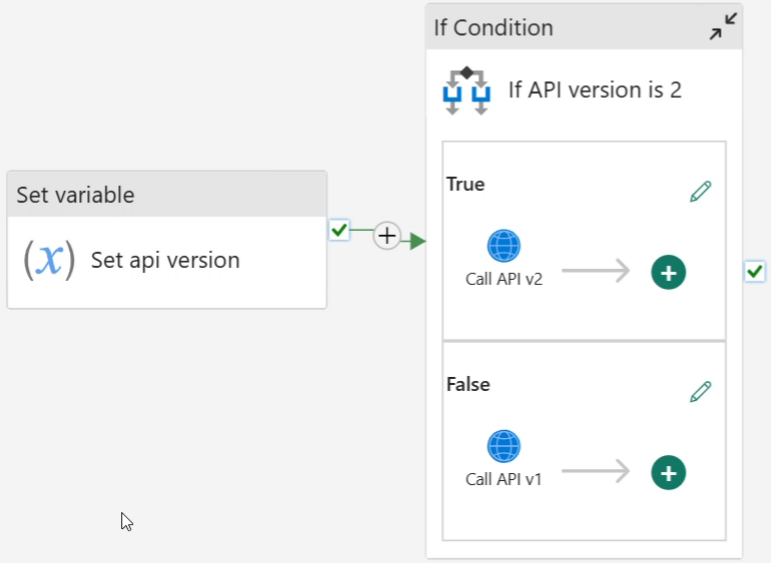
https://app.powerbi.com/groups/392943e9-4112-45aa-8f17-224f2310a474/pipelines/8291caf3-2ff5-4751-919f-a1c973fc10f9?experience=power-bi

Data-Marc.com | CallAPIDemo | Allowed to be shared external | Search

Home | Activities | Run | View

Edit JSON code | Auto align | Zoom to fit | Zoom 100%

Home
Create
Browse
OneLake data hub
Apps
Monitor
Workspaces
FabConEU - Dev
CallAPIDemo
...
Power BI



Parameters | Variables | Settings | Output

New | Delete

Name	Type	Default value
api_version_number	String	1

main

Wrap up

- Git
  - Azure DevOps, GitHub and GitHub Enterprise integration
  - Branching strategy
  - Continuous Integration and Continuous Delivery
- Deployment
  - 3 opinionated deployment strategies depending on your need
- Testing
  - Use framework to improve your development experience
  - Use as quality gate for better reliability of your data pipelines
  - Open for contributions and feedback, and helping you to onboard

Resources

- Fabric Deployment Pipelines
  https://learn.microsoft.com/en-us/fabric/cicd/deployment-pipelines/intro-to-deployment-pipelines
- Fabric Git Integration
  https://learn.microsoft.com/en-us/fabric/cicd/git-integration/intro-to-git-integration
- API samples for CI/CD
  https://learn.microsoft.com/en-us/fabric/cicd/git-integration/git-automation
- Microsoft Fabric Git REST APIs
  https://blog.fabric.microsoft.com/en-us/blog/automate-your-ci-cd-pipelines-with-microsoft-fabric-git-rest-apis
- Data Factory Testing Framework
  https://github.com/microsoft/data-factory-testing-framework

Questions?

Marc Lelijveld

Technical Evangelist | Solution Architect
Macaw Netherlands

Microsoft® Most Valuable Professional

MarcLelijveld

linkedin.com/in/MarcLelijveld

Data-Marc.com

DutchFabricUsergroup.com

Arjen Kroezen

Senior Software Engineer
Microsoft

ArjenDev

linkedin.com/in/ArjenKroezen

Please rate this session on the app

cvent

GET IT ON Google Play

Download on the App Store