



# 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

Understand how to integrate Git with your Fabric Workspaces

## Deployment

Understand the different options to orchestrate and structure your deployment

## Testing

Improve development experience by developing data pipelines faster and more reliable



# Marc Lelijveld

Technical Evangelist | Solution Architect  
Macaw Netherlands



MarcLelijveld



[linkedin.com/in/MarcLelijveld](https://www.linkedin.com/in/MarcLelijveld)



[Data-Marc.com](https://Data-Marc.com)



[DutchFabricUsergroup.com](https://DutchFabricUsergroup.com)



# Arjen Kroezen

Senior Software Engineer  
Microsoft



ArjenDev



[linkedin.com/in/ArjenKroezen](https://www.linkedin.com/in/ArjenKroezen)



# Setting the scene



# CI / CD

## Continuous Integration

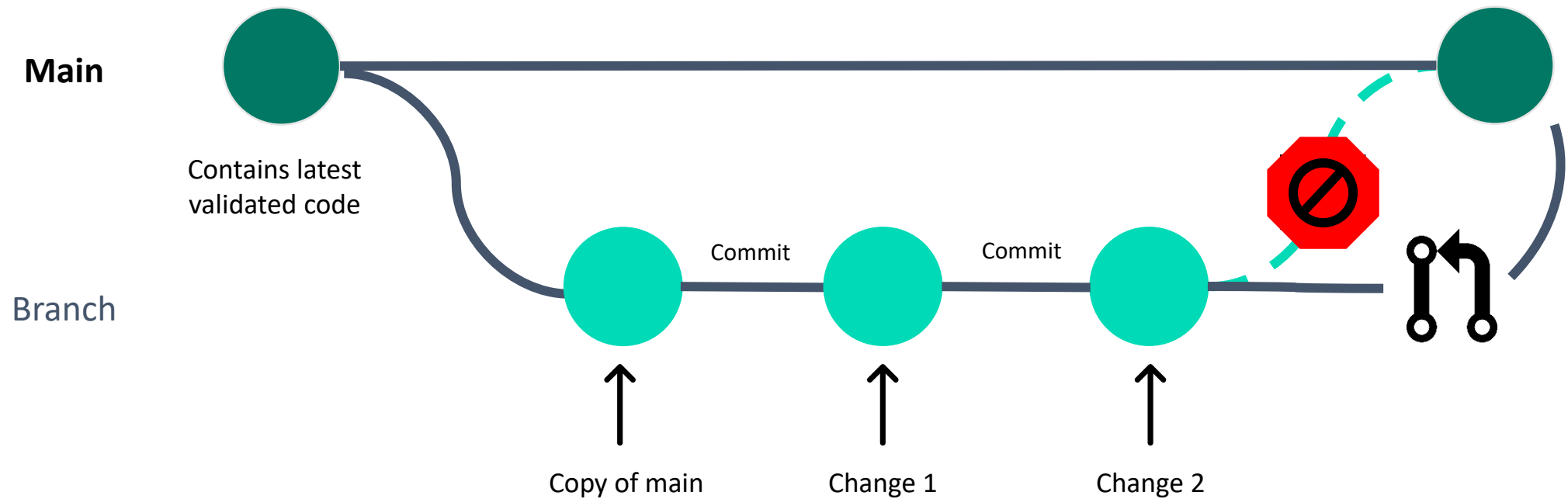
- Small but frequent changes
- Working with multiple people on various aspects of the solution
- Merging back into main branch

## Continuous Deployment

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

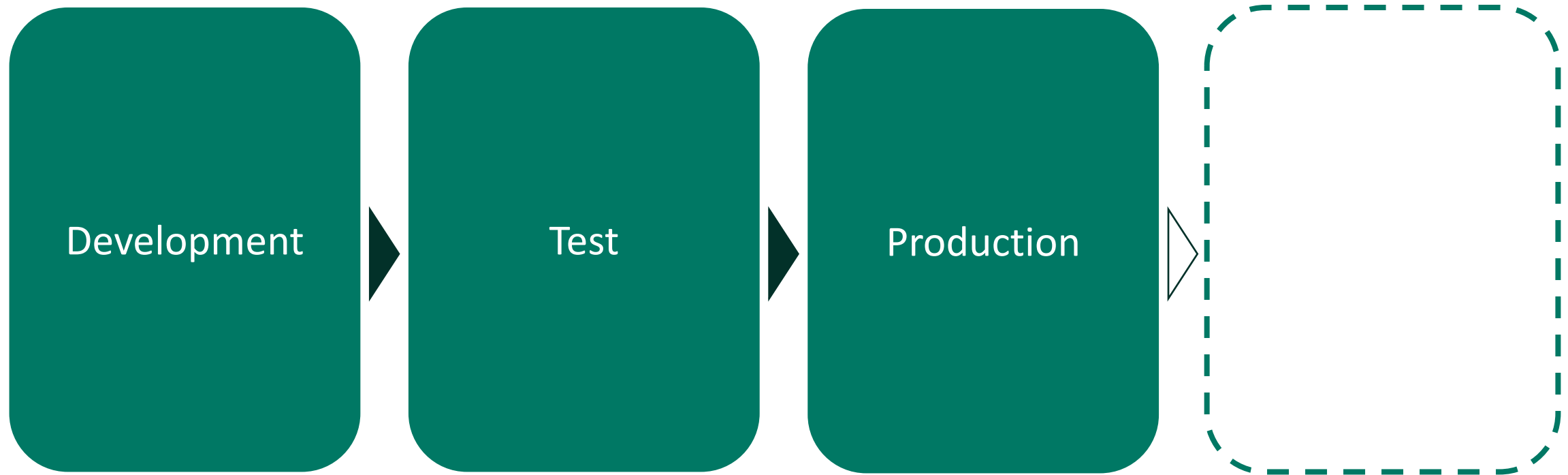
This all, to improve the developer experience.

# Git concept

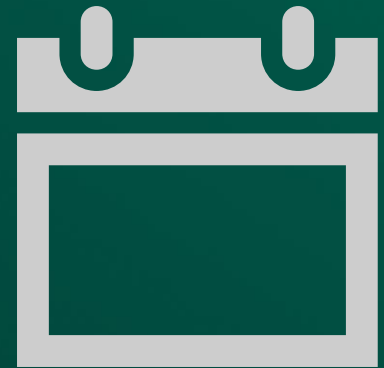




# Staged approach

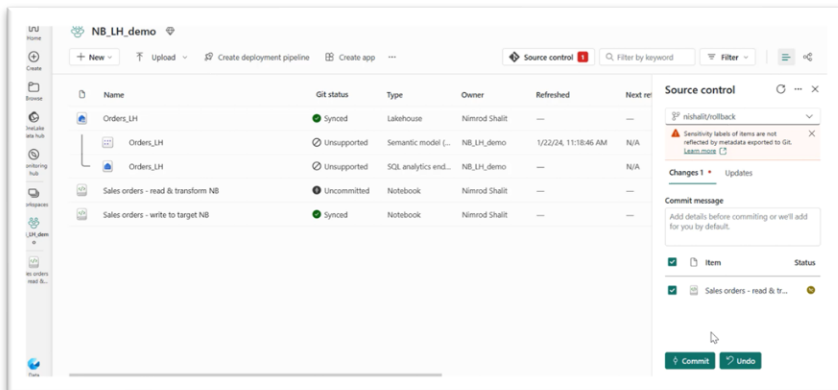


# What do we have today?



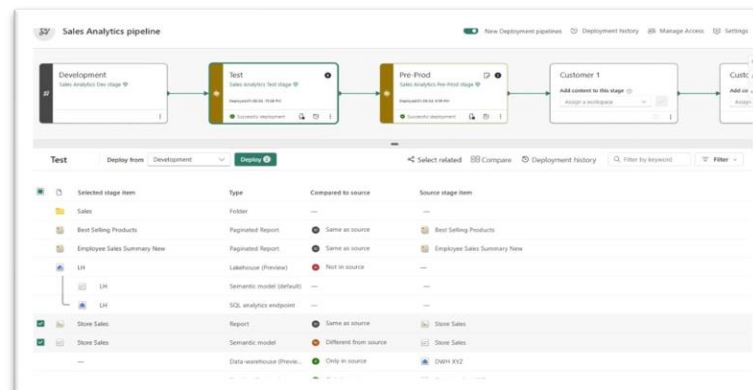
# Fabric CI/CD platform

## Built-in git integration



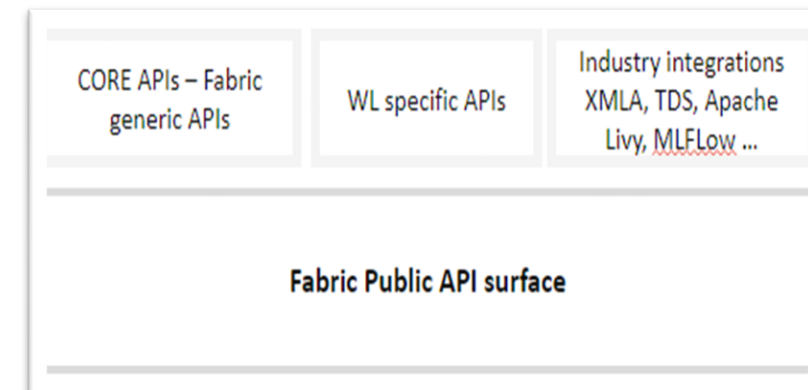
Public preview

## Deployment pipelines



GA feature

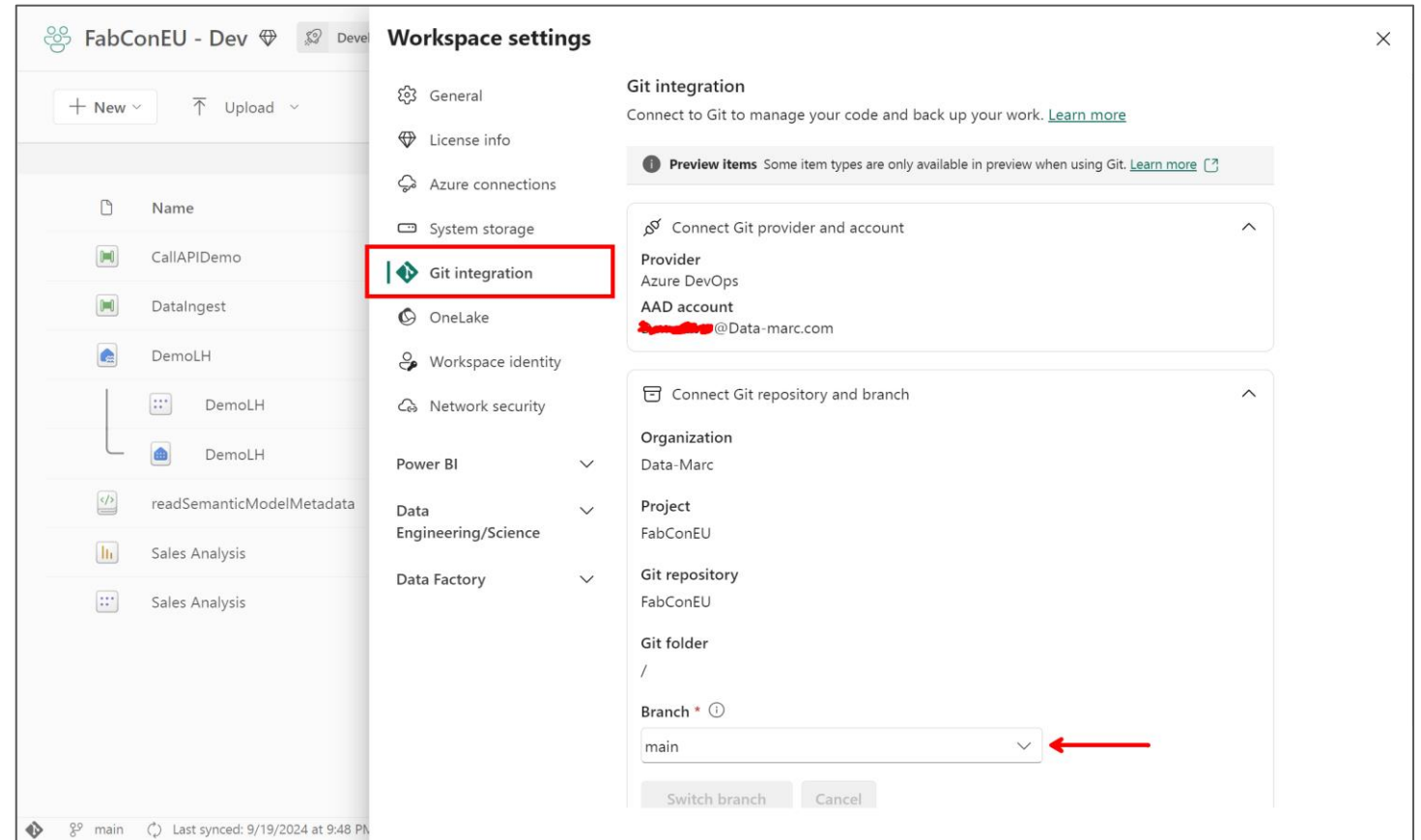
## Fabric REST APIs



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

The screenshot displays the Microsoft Fabric Deployment Pipelines interface. The top section shows a visual pipeline with stages: Development (FabConEU - Dev), Test, Custom Stage Name 2, and Custom Stage Name 3. Each stage has an 'Add content to this stage' button and an 'Assign a workspace' dropdown. The bottom section is a table titled 'Development' showing a list of items to be deployed.

| Selected stage item       | Type                     | Compared to source | Source stage item |
|---------------------------|--------------------------|--------------------|-------------------|
| CallAPIDemo               | Data pipeline (Preview)  | —                  | —                 |
| DataIngest                | Data pipeline (Preview)  | —                  | —                 |
| DemoLH                    | Lakehouse (Preview)      | —                  | —                 |
| DemoLH                    | Semantic model (default) | —                  | —                 |
| DemoLH                    | SQL analytics endpoint   | —                  | —                 |
| readSemanticModelMetadata | Notebook (Preview)       | —                  | —                 |
| Sales Analysis            | Report                   | —                  | —                 |
| Sales Analysis            | Semantic model           | —                  | —                 |

# 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

The screenshot displays the 'Sales Analytics pipeline' interface. The top navigation bar includes 'New Deployment pipelines', 'Deployment history', 'Manage Access', and 'Settings'. The pipeline stages are 'Development' (Sales Analytics Dev stage), 'Test' (Sales Analytics Test stage), 'Pre-Prod' (Sales Analytics Pre-Prod stage), and 'Customer 1'. The 'Test' stage is selected, showing a 'Deploy from' dropdown set to 'Development' and a 'Deploy' button. Below the stages, a table compares items between the source and target environments.

| Selected stage item        | Type                     | Compared to source    | Source stage item          |
|----------------------------|--------------------------|-----------------------|----------------------------|
| Sales                      | Folder                   | —                     | —                          |
| Best Selling Products      | Paginated Report         | Same as source        | Best Selling Products      |
| Employee Sales Summary New | Paginated Report         | Same as source        | Employee Sales Summary New |
| LH                         | Lakehouse (Preview)      | Not in source         | —                          |
| LH                         | Semantic model (default) | —                     | —                          |
| LH                         | SQL analytics endpoint   | —                     | —                          |
| Store Sales                | Report                   | Same as source        | Store Sales                |
| Store Sales                | Semantic model           | Different from source | Store Sales                |
| —                          | Data-warehouse (Preview) | Only in source        | DWH XYZ                    |
| —                          | Pipeline (Preview)       | Only in source        | Data pipeline XYZ          |

# 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

The screenshot displays the 'Workspace settings' page in Azure DevOps. On the left is a sidebar with navigation links: 'General', 'License info', 'Azure connections', 'System storage', 'Git integration' (highlighted with a green bar), and 'OneLake'. The main content area is titled 'Git integration (Preview)' and includes a search bar at the top. Below the title, it says 'Connect to Git to manage your code and back up your work. [Learn more](#)'. A section titled 'Connect Git provider and account' contains a 'Git provider' subsection with two selectable options: 'Azure DevOps' and 'GitHub'. The 'GitHub' option is currently selected, indicated by a black border. To the right of the main content area, a callout box titled 'Git integration' provides additional details: it states that users can sync workspace items with GitHub repositories, that the feature is enabled for the entire organization, and that users can select GitHub as their Git provider. At the bottom of this callout box is a green toggle switch labeled 'Enabled'.

Git integration

- △ Users can sync workspace items with **GitHub** repositories  
*Enabled for the entire organization*

Users can select **GitHub** as their Git provider and sync items in their workspaces with **GitHub** repositories.

☒ Enabled

**Workspace settings**

Search

- General
- License info
- Azure connections
- System storage
- Git integration**
- OneLake

**Git integration (Preview)**

Connect to Git to manage your code and back up your work. [Learn more](#)

Connect Git provider and account

**Git provider**

Azure DevOps

GitHub



Generally available

# Fabric Git integration



Supported  
items

By end of  
the year



Data pipeline



Lakehouse



Warehouse



Reflex

New



Report



Paginated Report



Semantic Model



Real-Time Dashboard

New



Notebook



Spark Job Definition



Spark Environment



Queryset

New



Dataflow Gen2



Org App



Eventhouse



Metrics Set



Eventstream



Mirrored Database



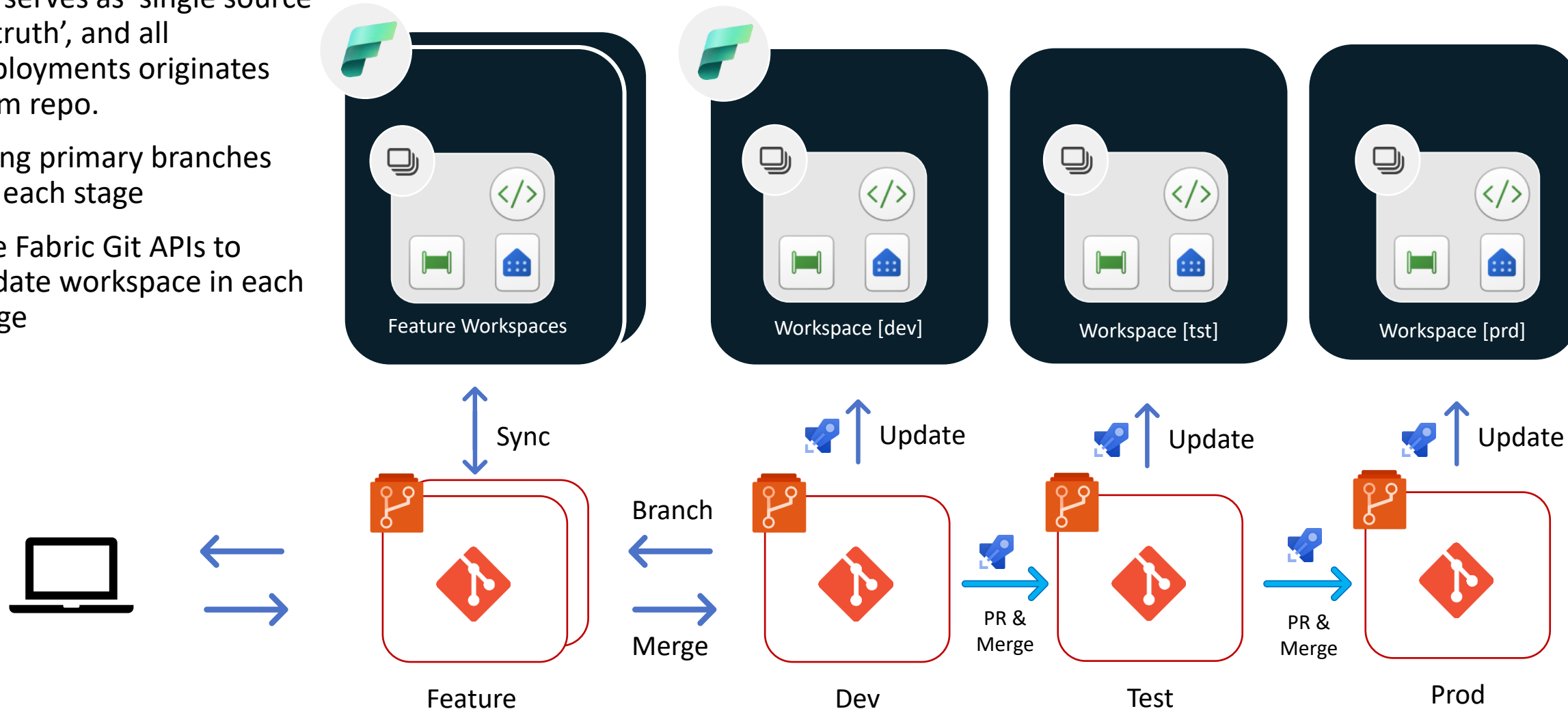
GraphQL API

# Deployment strategies



# Scenario 1 – Git based deployments

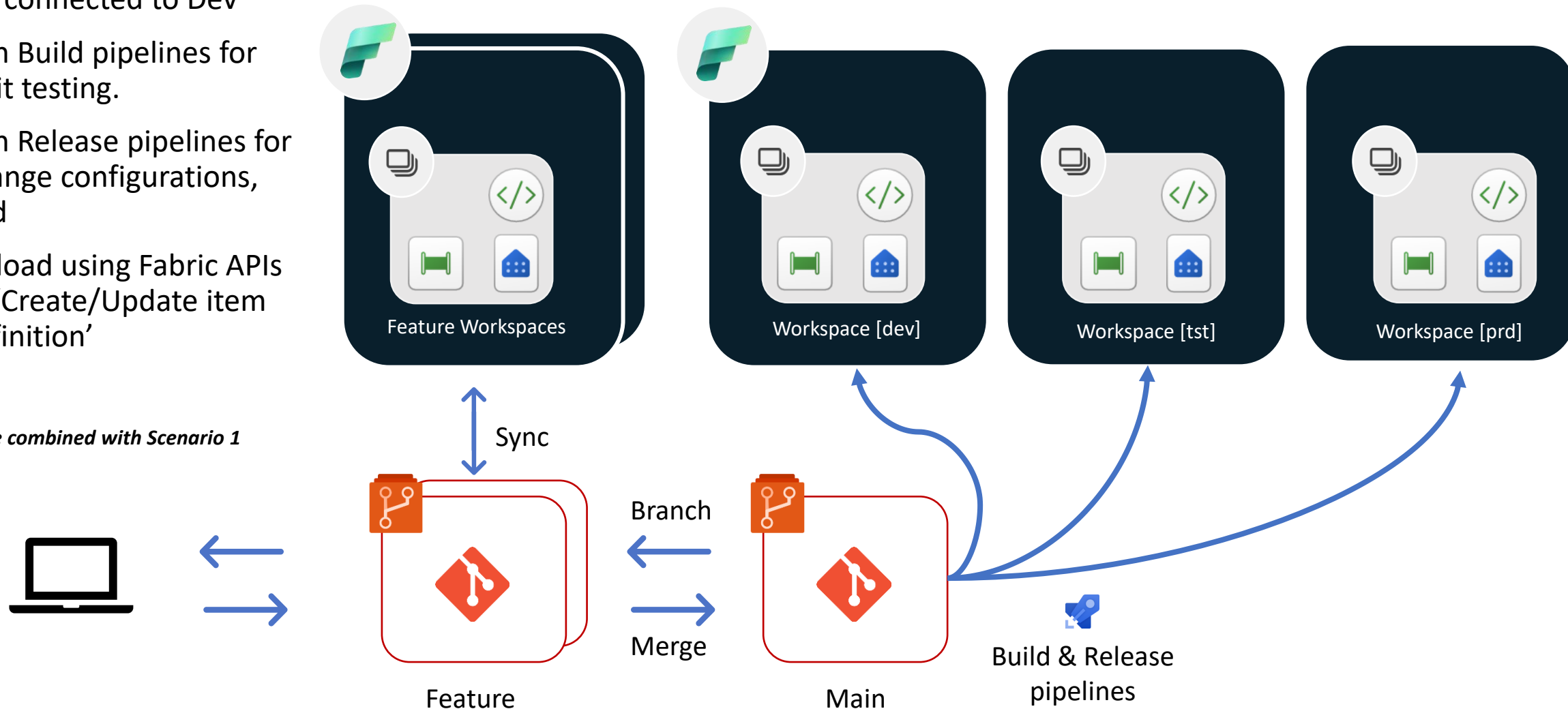
- Git serves as 'single source of truth', and all deployments originates from repo.
- Using primary branches for each stage
- Use Fabric Git APIs to update workspace in each stage



# Scenario 2\* – Git & Build environments

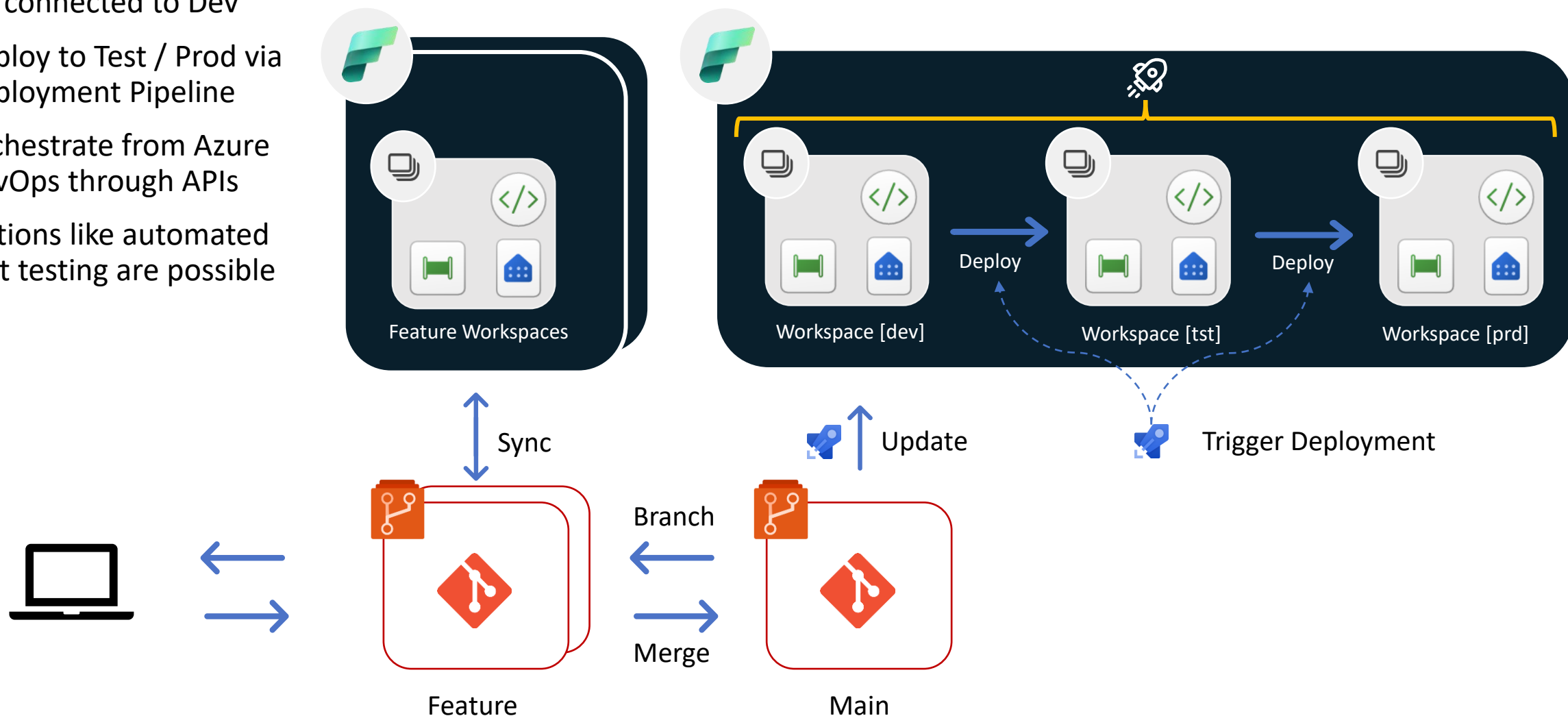
- Git connected to Dev
- Run Build pipelines for Unit testing.
- Run Release pipelines for change configurations, and
- Upload using Fabric APIs to 'Create/Update item definition'

\* 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 <b>scenario 3</b> over time    | Low            |
| Comfortable working with git/code? | Your go-to scenario will be <b>scenario 1</b> .                                     | Medium         |
| Require a lot of customization?    | Using the APIs in <b>scenario 2</b> allows you to customize everything to your wish | High           |

*In greenfield scenario – opt for scenario 1!*

# DEMO TIME!



# Taking it one step further...





Export (Preview)



Use a template (Preview)



Validate



Run

## Web



Call API



General

Settings

Connection \* ⓘ



https://example.com arjenkroezen



Refresh



Edit

Relative URL

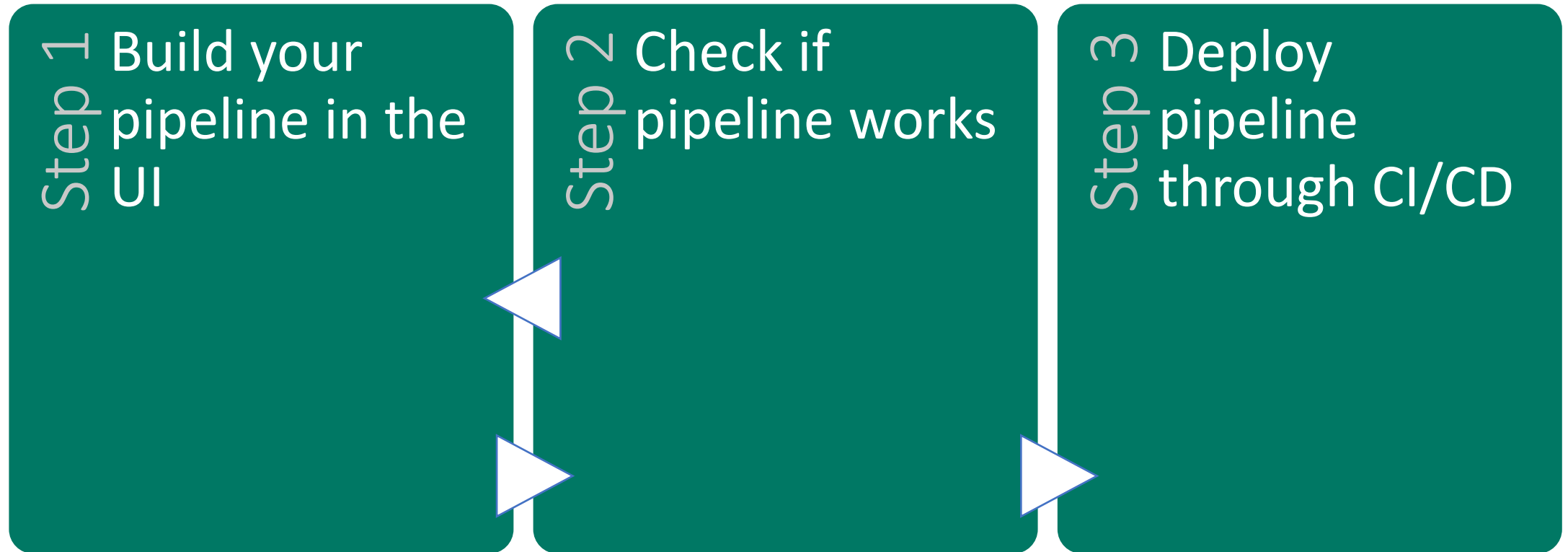
@pipeline().parameters.BasePath

Method \* ⓘ

GET



# Current Development Experience using Data Pipelines in Fabric

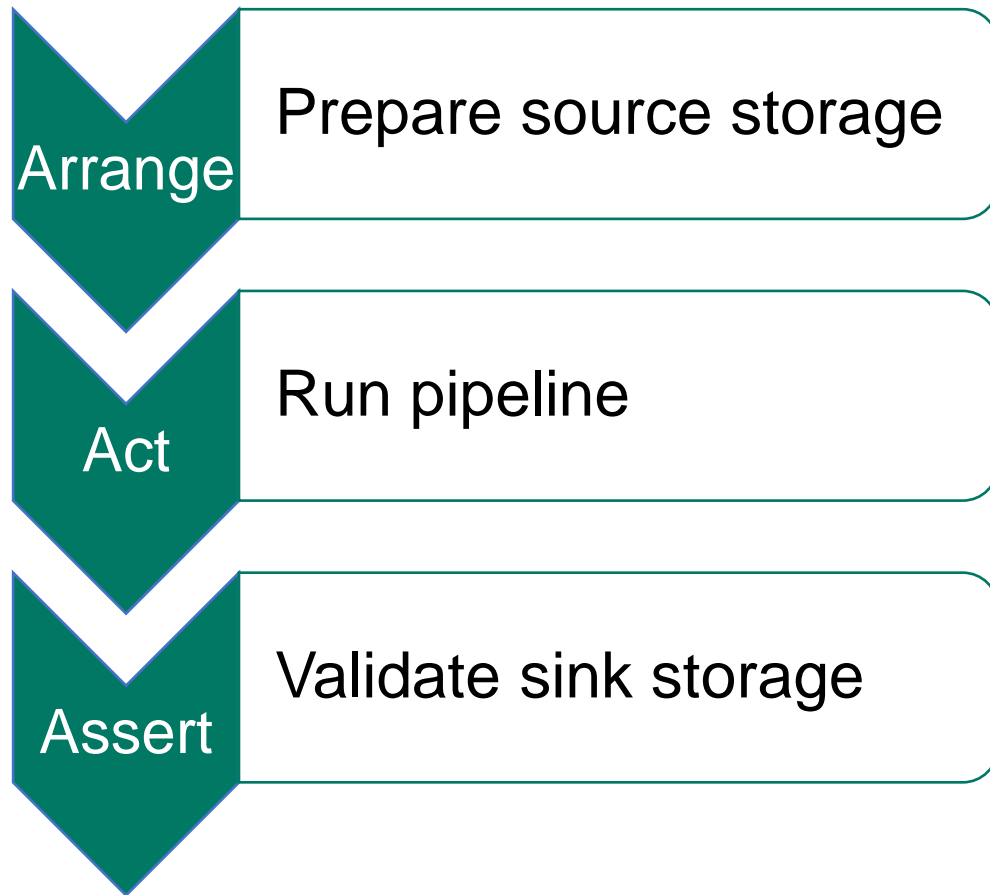




# How would you check the pipeline works as expected?

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

# Typical example of testing Data Pipelines



Copy data



Copy from source  
to sink

# Different types of tests

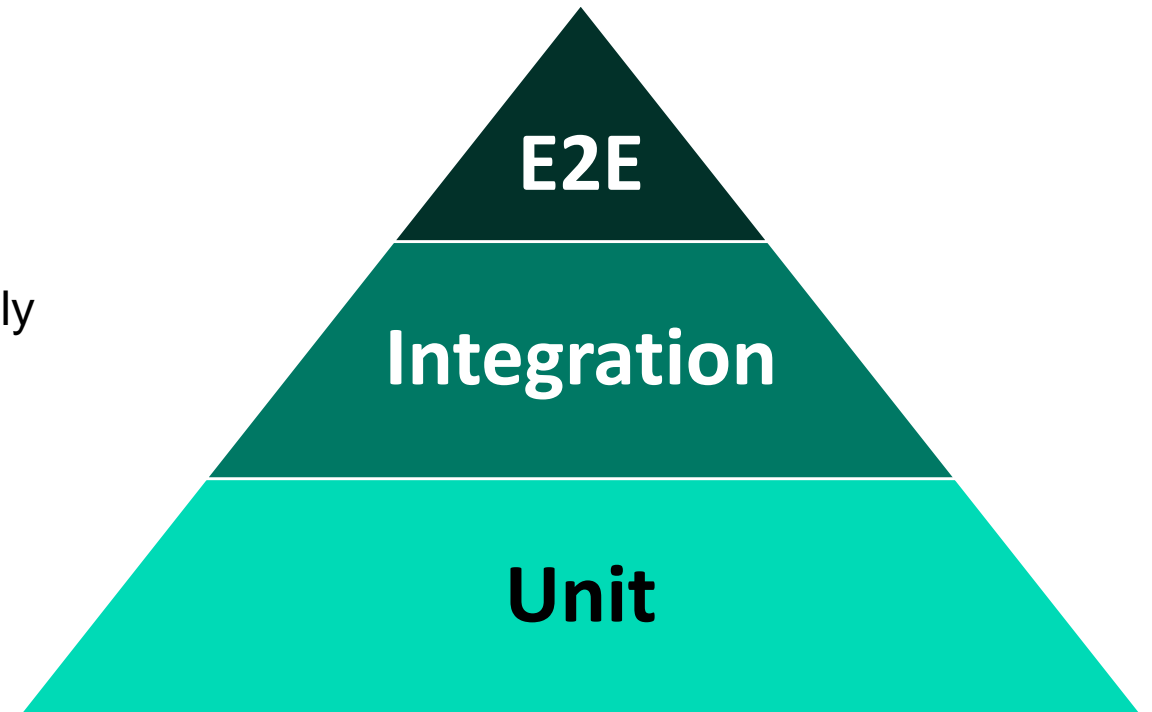
Previous example demonstrates a typical E2E test

## 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:

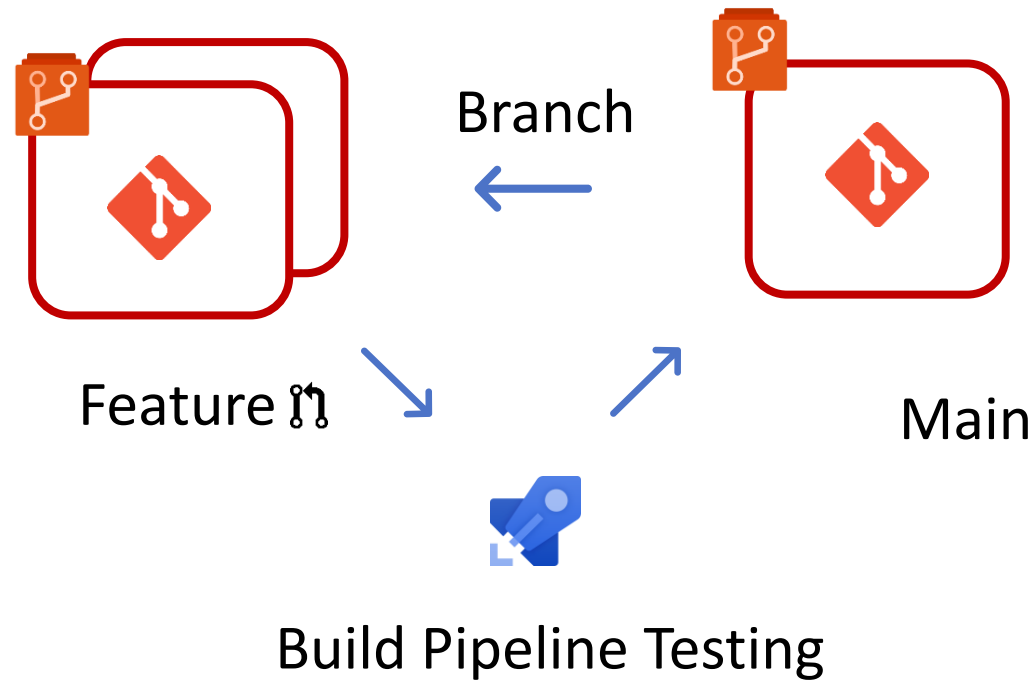
- Parameters
- Expressions (Domain-Specific Language)
- Activities
- Control Activities

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

View JSON code

Pipeline name: pipeline1

Copy to clipboard

```
1 {
2   "name": "pipeline1",
3   "objectId": "405d2904-9c4a-4810-80aa-99f7fe07f55d",
4   "properties": {
5     "activities": [
6       {
7         "name": "Set Job Container URL",
8         "type": "SetVariable",
9         "dependsOn": [
10          {
11            "activity": "Set JobContainerName1",
12            "dependencyConditions": [
13              "Succeeded"
14            ]
15          }
16        ],
17        "policy": {
18          "secureOutput": false,
19          "secureInput": false
20        },
21        "typeProperties": {
22          "variableName": "JobContainerURL",
23          "value": {
24            "value": "@concat('https://', pipeline().parameters.BatchStorageAccountName, '.blob.core.windows.net/', variables('JobContainerName'))",
25            "type": "Expression"
26          }
27        }
28      },
29      {
30        "name": "Set UserAssignedIdentityReference",
31        "type": "SetVariable",
32        "dependsOn": [],
33        "policy": {
34          "secureOutput": false,
35          "secureInput": false
36        },
37        "typeProperties": {
38          "variableName": "UserAssignedIdentityReference",
39          "value": {
40            "value": "@concat('/subscriptions/', pipeline().parameters.BatchAccountSubscription, '/resourcegroups/', pipeline().parameters.BatchAccountResourceGroup, '/providers/Microsoft.ManagedIdentity/userAssignedIdentities/', variables('UserAssignedIdentityName'))",
41            "type": "Expression"
42          }
43        }
44      }
45    ]
46  }
47 }
```

Close

Workspace settings

Search

Git integration (Preview)  
Connect to Git with Azure DevOps to manage your code and back up your work. [Learn more](#)

View Azure DevOps account

Connect Git repository and branch

Organization  
arjendewestcentralus

Project  
TestingFramework

Git repository  
TestingFramework

Git folder  
/

Branch \* ⓘ  
main

Switch and override Cancel

Disconnect workspace



# 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

```
{
  "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"
  }
}
```



# 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



# 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)
```





Set variable

(x) ConstructApiPath



Web

 Start Job

# 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!







# 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



MarcLelijveld



[linkedin.com/in/MarcLelijveld](https://linkedin.com/in/MarcLelijveld)



[Data-Marc.com](https://Data-Marc.com)



[DutchFabricUsergroup.com](https://DutchFabricUsergroup.com)

# Arjen Kroezen

Senior Software Engineer  
Microsoft



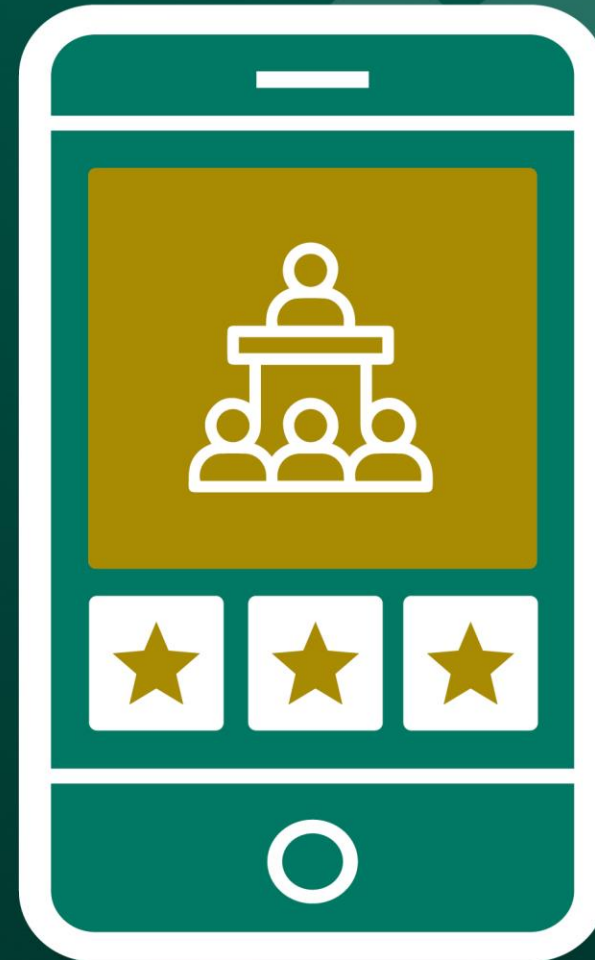
ArjenDev



[linkedin.com/in/ArjenKroezen](https://linkedin.com/in/ArjenKroezen)



Please rate  
this session  
on the app



cvent

