Applying DevOps practices to your Power BI deployments in Microsoft Fabric

Power BI

Microsoft Fabric

Kevin Chant

Data Céilí

Thank you to our sponsors. Please say hello to them and find out how they can help you.

micro warehouse

a sherweb company

Spanish Point technologies ltd

datalineo

Prodata SQL Centre of Excellence

REALTIME IT Recruitment Experts

Agenda

- Bio
- DevOps introduction
- Power BI Desktop projects
- Microsoft Fabric Git integration
- Suggested release/deployment options

Kevin Chant

- Lead Technology Advocate in the Netherlands
- Worked in IT since the days of Windows 95
- Experience in various sectors
- Various certifications, Data Platform MVP

- Twitter/Blue Sky: @kevchant
- LI: https://www.linkedin.com/in/kevin-chant/
- Blog: https://www.KevinRChant.com
- GitHub: https://github.com/kevchant

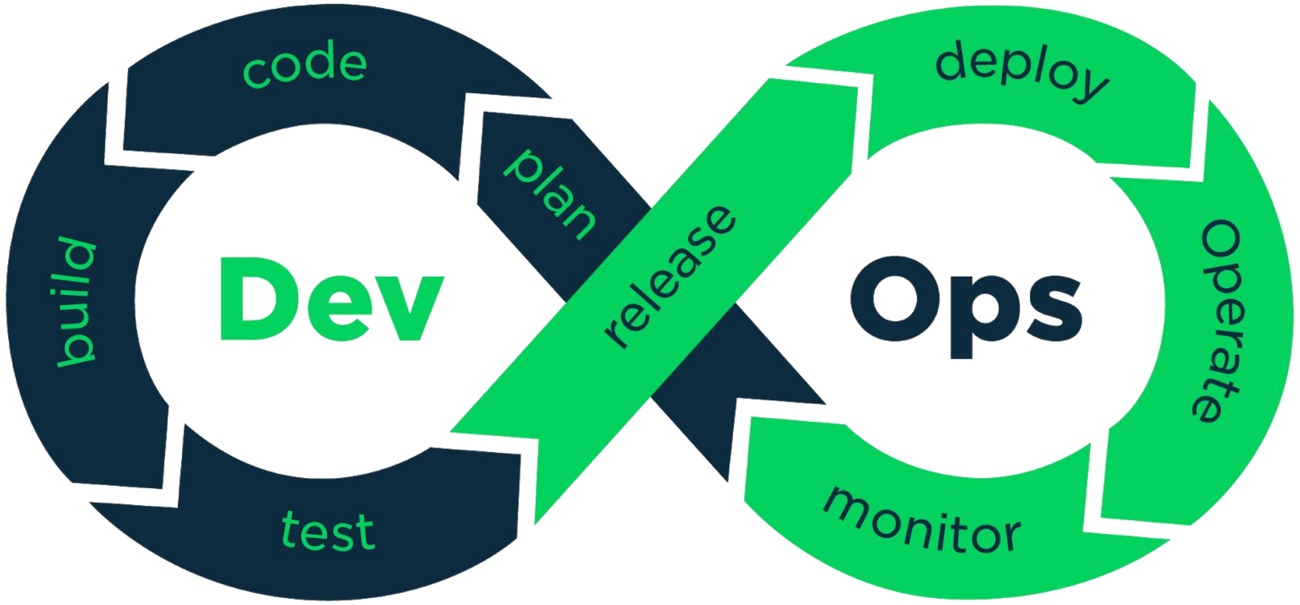
MVP Microsoft® Most Valuable Professional

Microsoft CERTIFIED DEVOPS ENGINEER EXPERT

Microsoft CERTIFIED Fabric Data Engineer

DevOps introduction

DevOps is practice that combines **Dev**elopment and **Op**erational processes together to improve the development lifecycle.



Typical Devops practices include

- Disciplined method to manage work items.
- Implementing version control of your code/metadata.
- Continuous Integration/Continuous Deployment.
- Implementing testing strategies.

Why apply DevOps to Power BI?

- Audit trail of changes.
- Faster updates, typically by CI/CD (Continuous Integration/Continuous Deployment).
- Reliable and more consistent deployments.
- Allows working with new industry innovations.

Version control

- Version control allows you to track and manage changes to files.
- Reduces conflicts and allows history of changes.
- Essential when looking to perform CI/CD with Fabric items.
- Various solutions for Power BI reports over the years. However, in past has been complex.
- Git version control is required for Fabric.

About Git

- Distributed version control system system.
- Git repository is a folder/directory where you store code/files you work with. It contains a hidden database in a .Git subfolder.
- Branches allow you to work with files in a new area in Git repository. It seems like a copy, but actually it is pointers.
- Git support for Power BI reports now easier thanks to one innovation.

Power BI Desktop projects

- Still in preview.
- Stores metadata in various files.
- Allows easy integration with version control.
- Supports two newer file formats:
  - Tabular Model Definition Language (TMDL)
  - Power BI Enhanced Report Format (PBIR)

TMDL

- New(ish) file format for the semantic model metadata.
- Aim is to replace ‘Model.bim’ file.
- Uses a YAML-like syntax for easier reading.
- Breaks down each object into a separate file.
- Integrates better with version control.

PBIR

- New(ish) format for report metadata.
- Uses a JSON syntax.
- Breaks down report items into a separate files.
- Integrates better with version control.
- Allows quick external editing in other tools.

Ways to create Power BI Projects

- Within Power BI Desktop
- When creating in a workspace with Git integration configured.
- Demo

Microsoft Fabric Git integration

- Version control for Fabric items.
- Allows supported items in a workspace to have metadata synchronized with a Git repository.
  - To be more precise a workspace synchronizes with a branch.
- Supports cloud-based versions of Azure DevOps and GitHub.
- Requires Fabric or Power BI Premium capacity.
- Items supported at various levels.

Example of supported items

- Data pipelines (preview)
- Dataflows gen2 (preview)
- Eventhouse and KQL database (preview)
- EventStream (preview)
- Lakehouse (preview)
- Mirrored database (preview)
- Notebooks
- Paginated reports (preview)
- Reflex (preview)
- Warehouses (preview)
- Reports (except reports connected to semantic models hosted in Azure Analysis Services, SQL Server Analysis Services, or reports exported by Power BI Desktop that depend on semantic models hosted in MyWorkspace) (preview)
- Semantic models (except push datasets, live connections to Analysis Services, model v1) (preview)
- Spark Job Definitions (preview)
- Spark environment (preview)
- SQL database (preview)

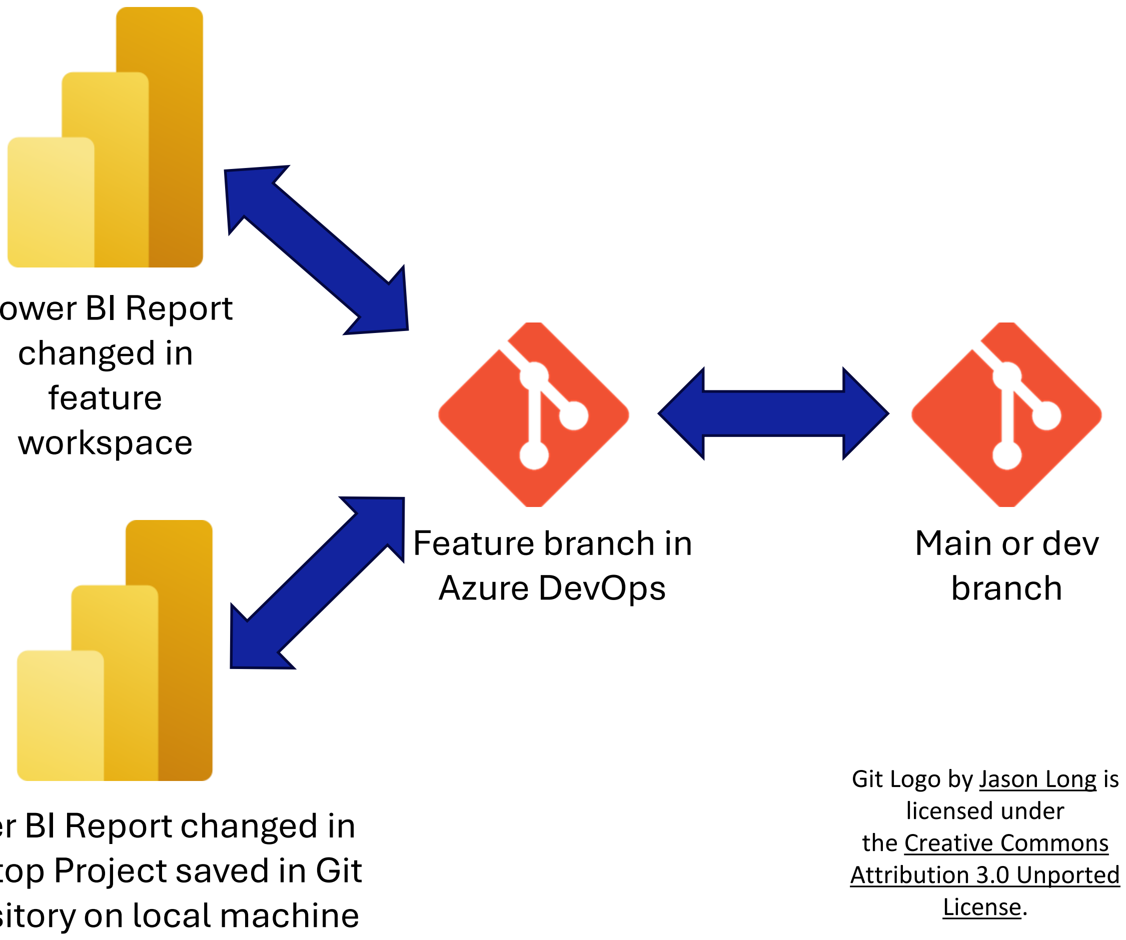
Git integration features

- Can change items in workspace and commit to repository.
- Can also update from repository.
- Can branch out to new workspace to create “feature” workspaces.

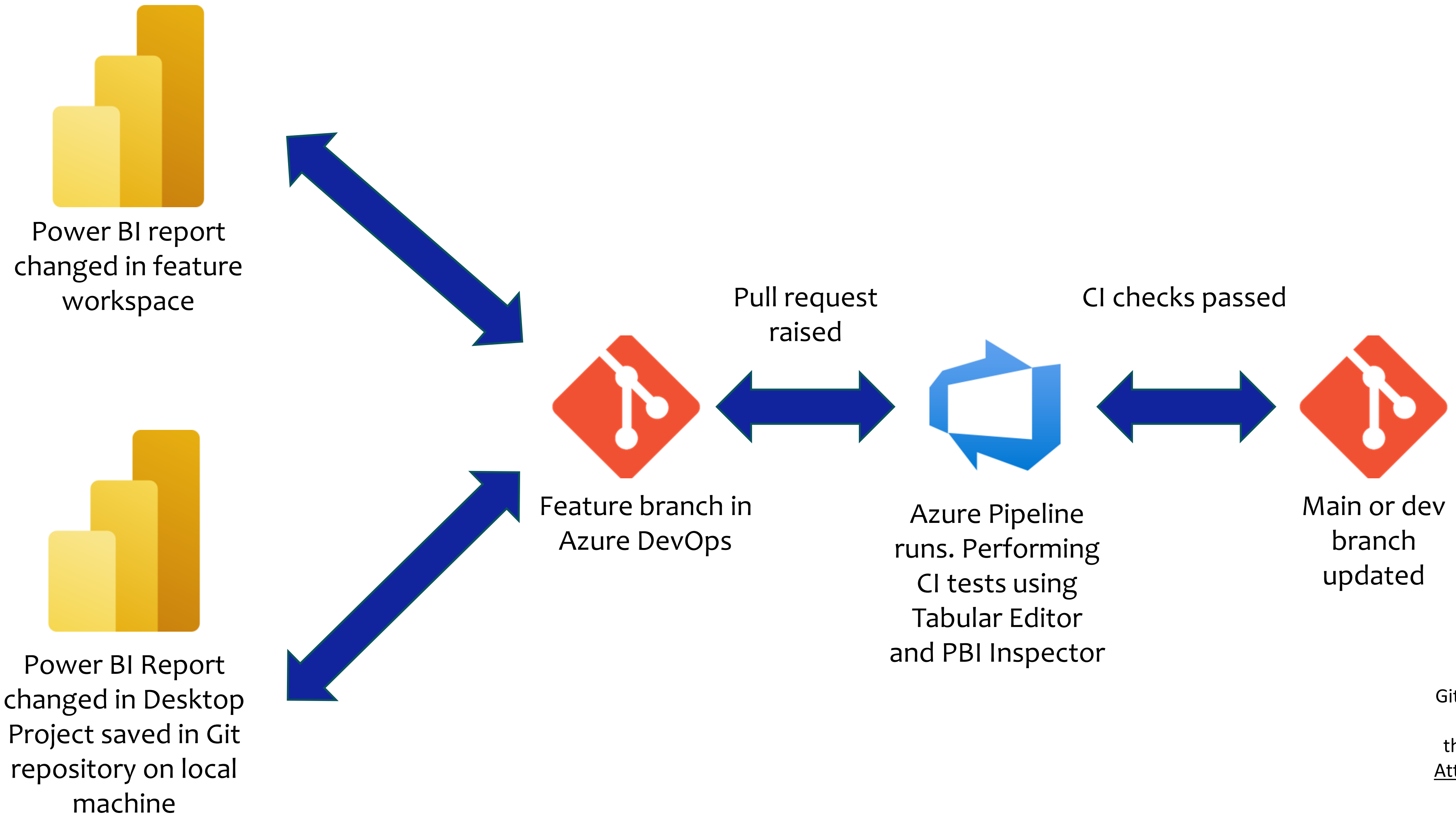
Feature workspaces

- Workspace that represents a specific feature.
- Can be achieved with “branch out to new workspace” functionality.
- However, this functionality requires permissions to create workspaces.
- Alternative is to create separate workspace for each developer.

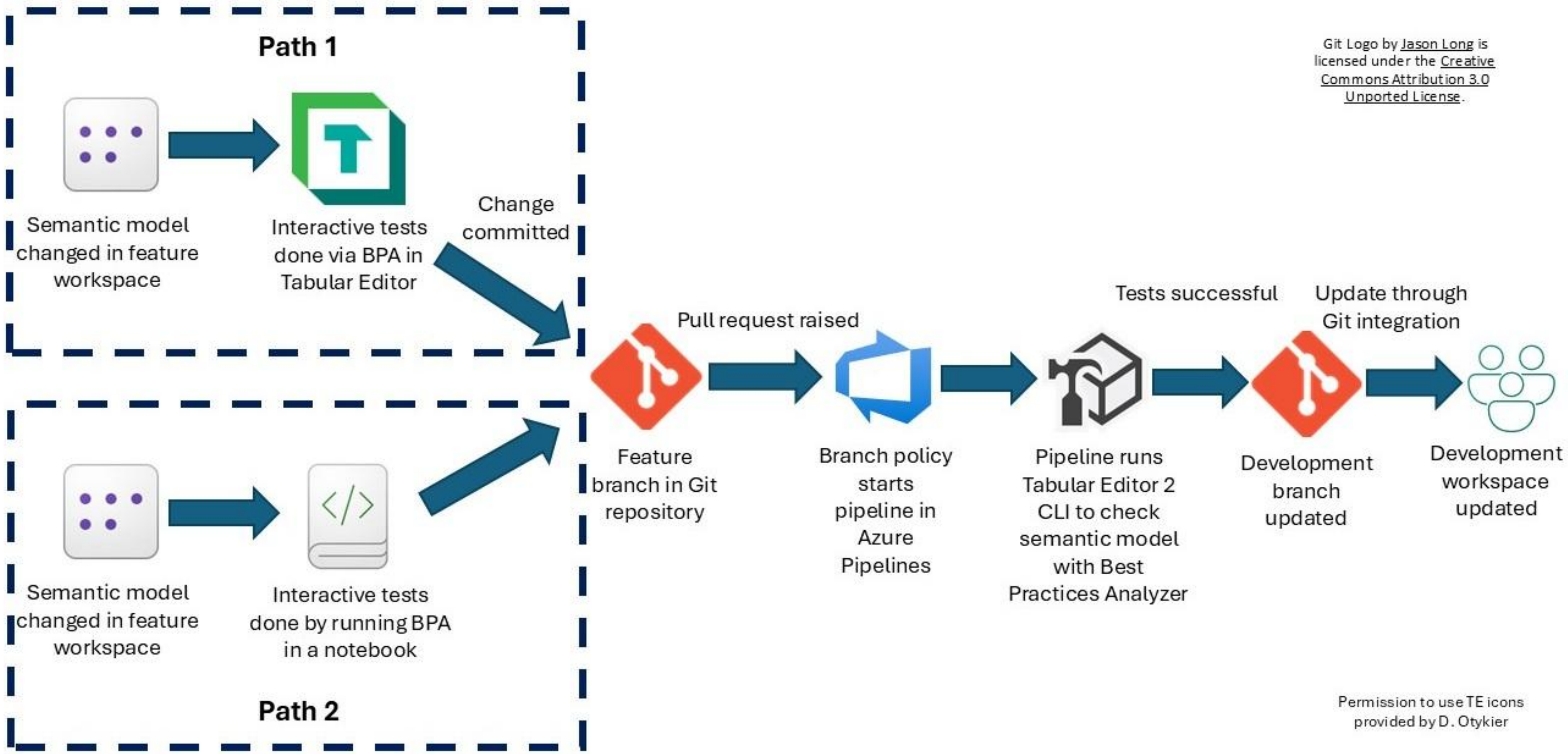
How feature workspaces aligns with recommended development process for Fabric


Power BI Report changed in feature workspace
Feature branch in Azure DevOps
Main or dev branch
Power BI Report changed in Desktop Project saved in Git repository on local machine

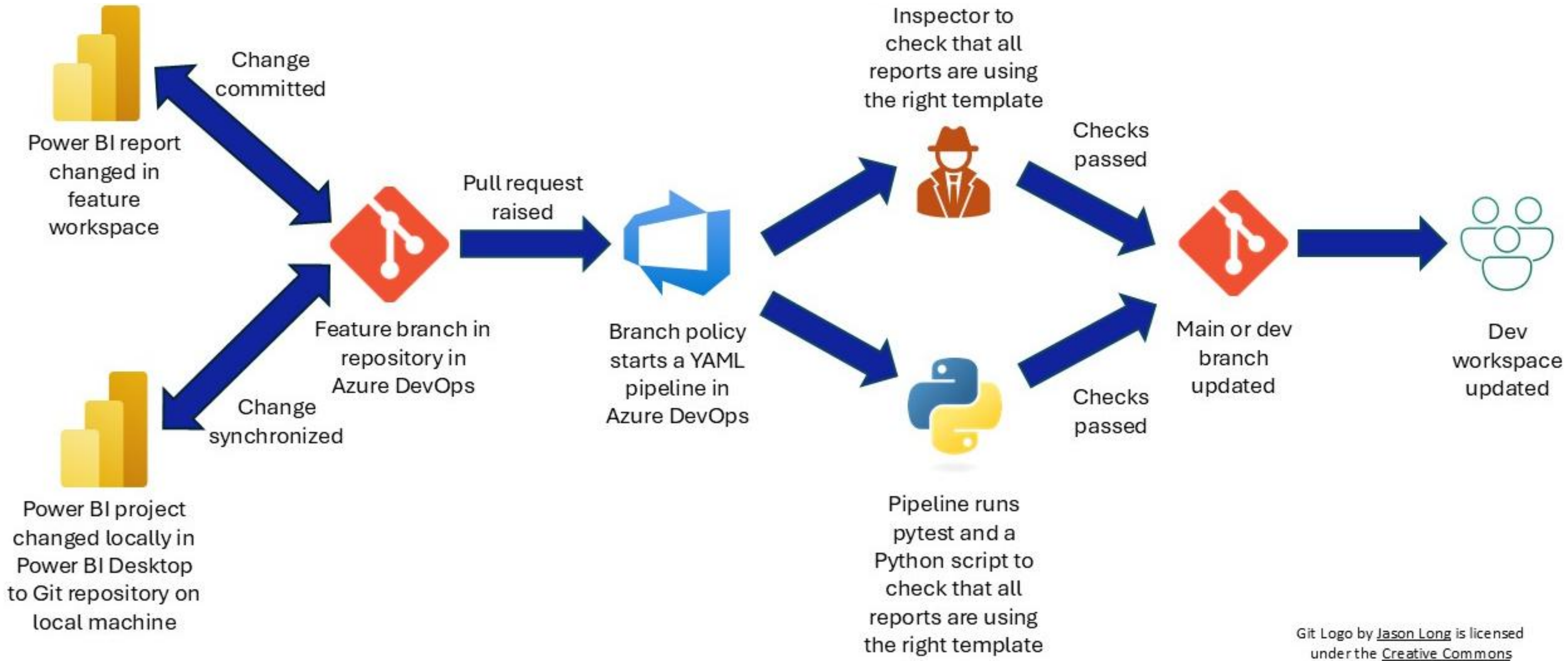
For true CI recommend unit tests


Power BI report changed in feature workspace
Power BI Report changed in Desktop Project saved in Git repository on local machine
Feature branch in Azure DevOps
Pull request raised
Azure Pipeline runs. Performing CI tests using Tabular Editor and PBI Inspector
CI checks passed
Main or dev branch updated

Happy paths to test semantic models


Path 1
Semantic model changed in feature workspace
Interactive tests done via BPA in Tabular Editor
Change committed
Path 2
Semantic model changed in feature workspace
Interactive tests done by running BPA in a notebook
Feature branch in Git repository
Pull request raised
Branch policy starts pipeline in Azure Pipelines
Pipeline runs Tabular Editor 2 CLI to check semantic model with Best Practices Analyzer
Tests successful
Development branch updated
Update through Git integration
Development workspace updated

Two ways to test report layout


Change committed
Power BI report changed in feature workspace
Pull request raised
Inspector to check that all reports are using the right template
Checks passed
Feature branch in repository in Azure DevOps
Branch policy starts a YAML pipeline in Azure DevOps
Main or dev branch updated
Dev workspace updated
Change synchronized
Checks passed
Power BI project changed locally in Power BI Desktop to Git repository on local machine
Pipeline runs pytest and a Python script to check that all reports are using the right template

Git integration demos

- Microsoft Fabric Git integration.
- Continuous integration tests.
- BPA bulk

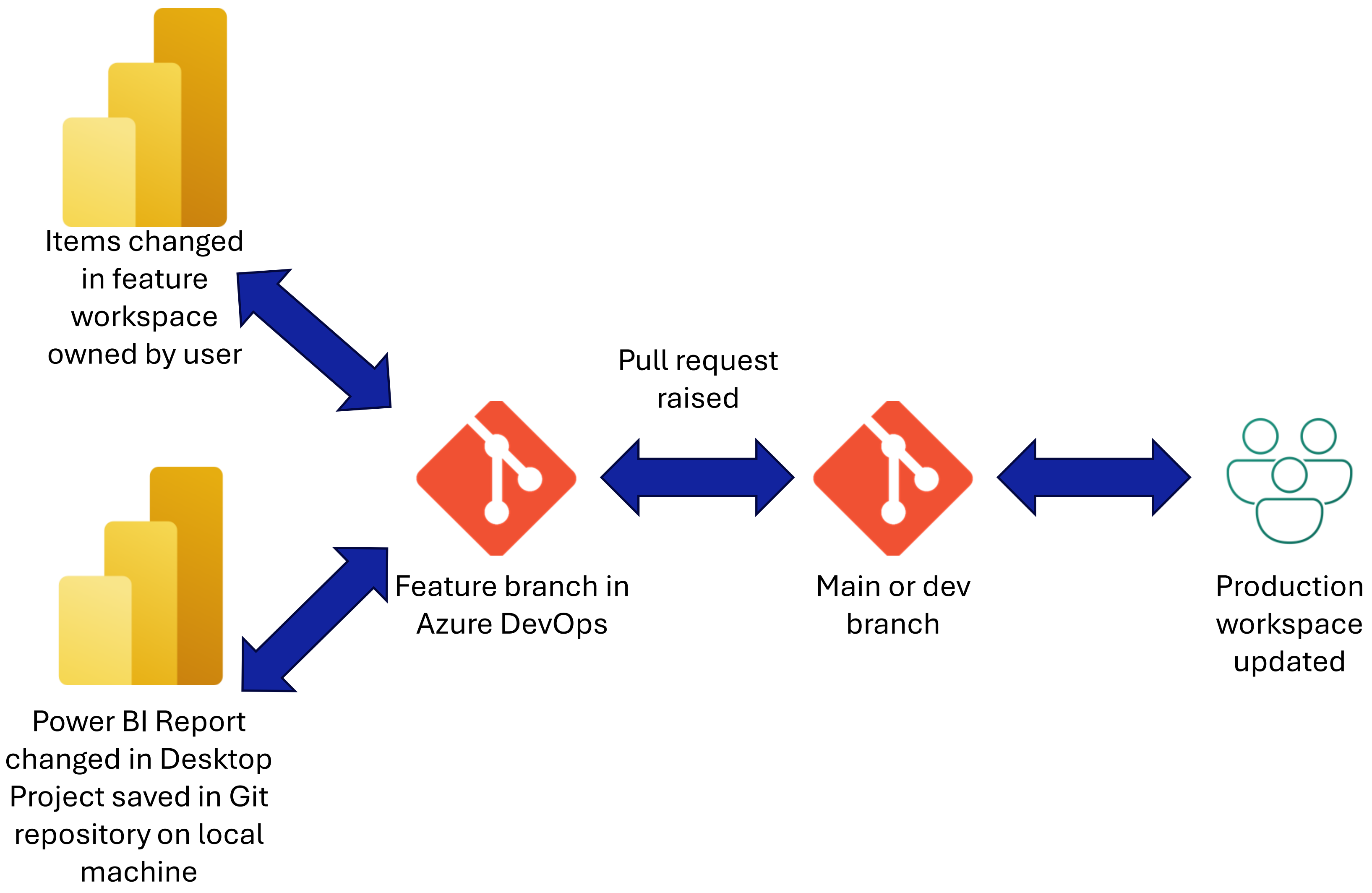
Recommended release options

- Microsoft released article last September.
- Includes previously shown recommended development process (CI) and release options (CD)
- I provide additional guidance and advice in context of Power BI.

Option 1 – Git-based deployments

- Have workspaces connected to different branches in the same Git repository.
- Update different workspaces via pull requests.
- Can be done with Fabric Git APIs supplemented by other APIs.
- Ideal for scenarios that require only minor source changes.
- Requires Git knowledge.

Option 1 diagram


Items changed in feature workspace owned by user
Pull request raised
Feature branch in Azure DevOps
Main or dev branch
Production workspace updated
Power BI Report changed in Desktop Project saved in Git repository on local machine

Option 2 – Git-based deployments using build pipeline

- Deploy to different workspaces through a suitable pipeline deployment service (Azure Pipelines, GitHub Actions). Typically with APIs.

- Recommendation is that workflow for each stage contains:
  - Build for unit tests
  - Release to perform update

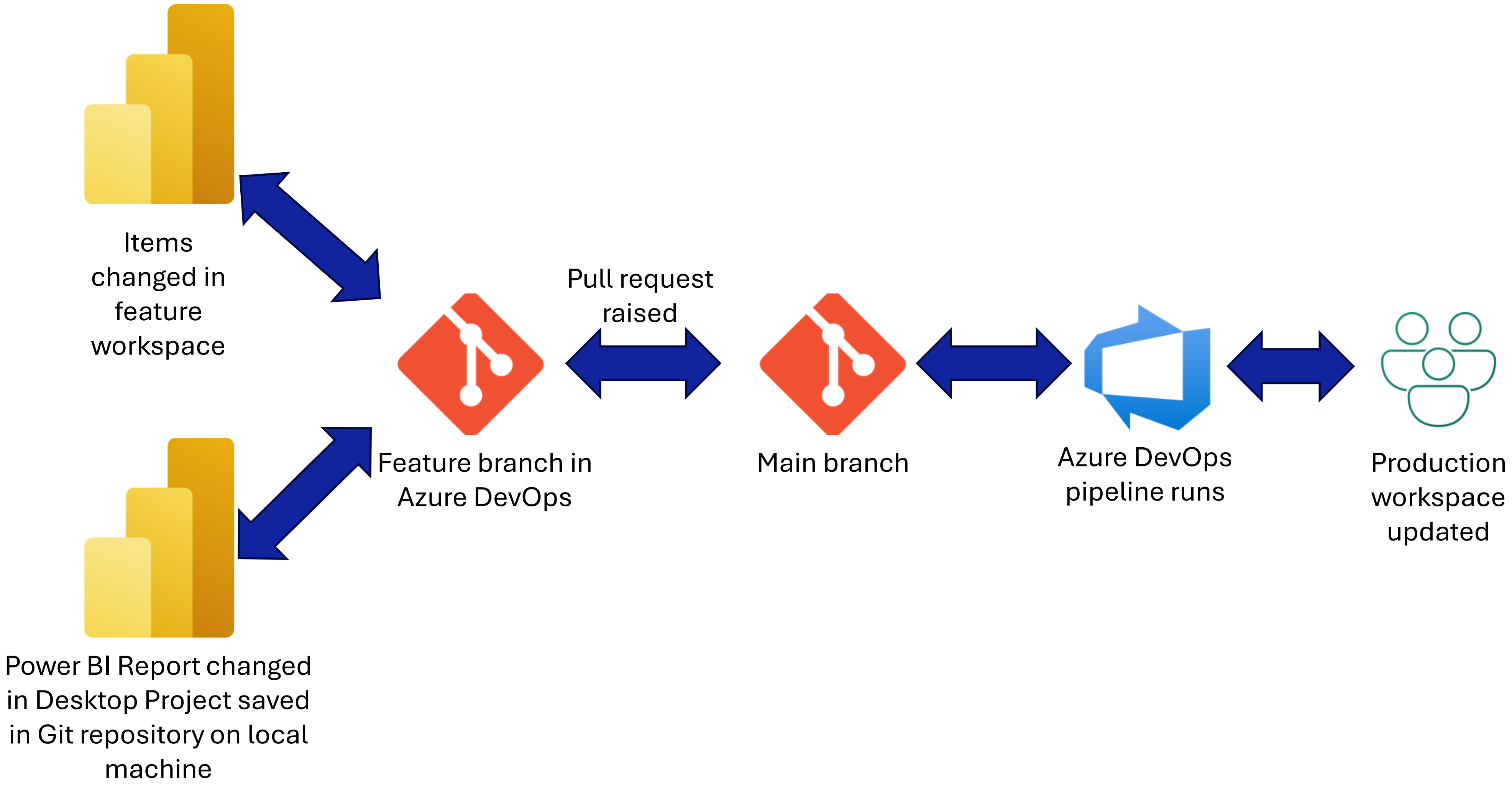
- Ideal for more complex scenarios and when you implement DataOps.

- Knowledge of ALM service required (Azure DevOps, GitHub).

Current Git-based deployment methods

- Direct REST API calls.
- PowerShell modules.
- Fabric-cicd Python library

Option 2 diagram


Items changed in feature workspace
Pull request raised
Feature branch in Azure DevOps
Main branch
Azure DevOps pipeline runs
Production workspace updated
Power BI Report changed in Desktop Project saved in Git repository on local machine

Advice for option 2 pipelines

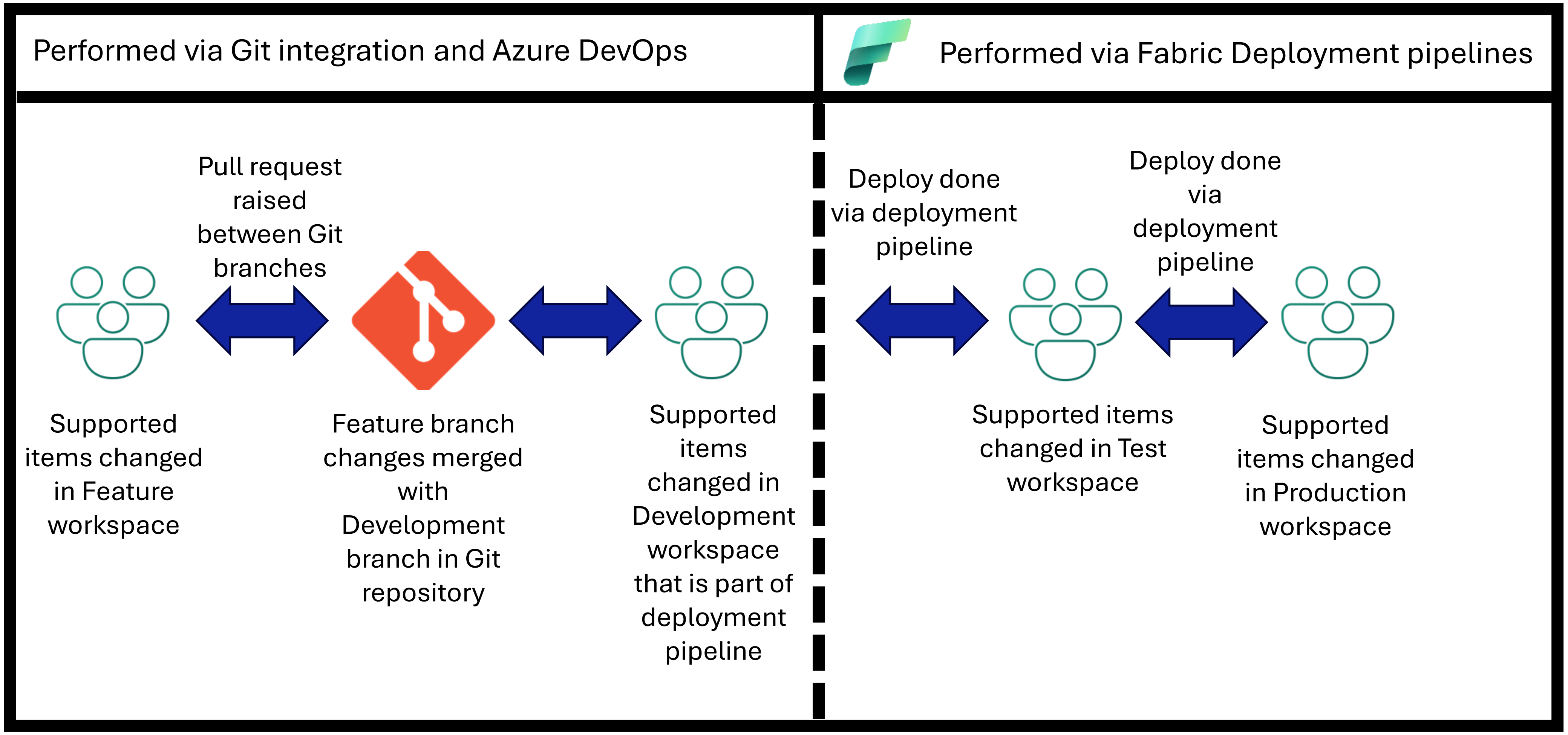
- Consider YAML pipelines for portability.
- Avoid hard-coding sensitive values. Either use the ALM offerings secrets store (variable groups, secrets) or Azure Key Vault.
- Strive to implement approvals process for production workloads.

Option 3 – Microsoft Fabric deployment pipelines

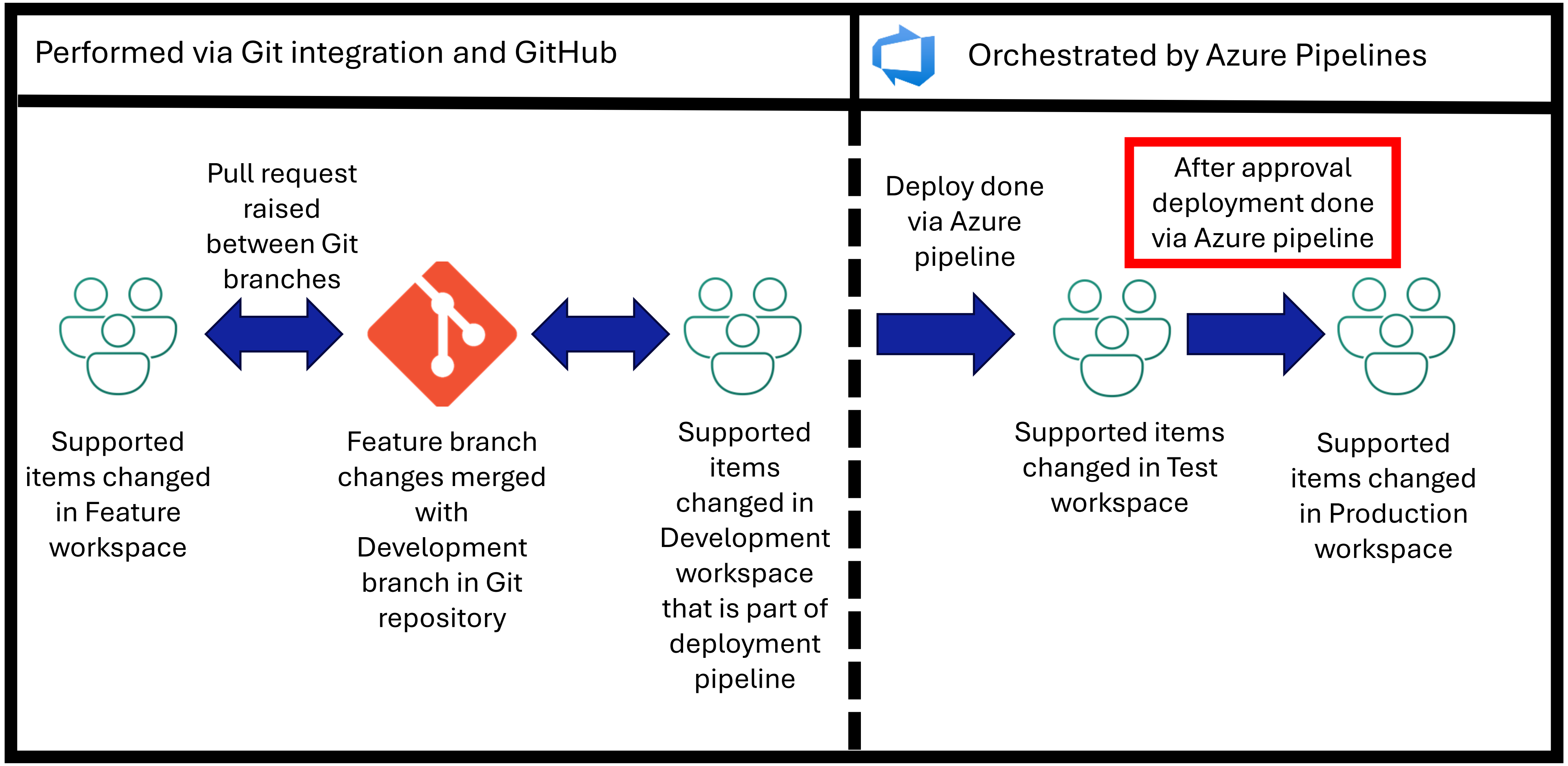
- Use development process to update a workspace that represents development environment/stage.

- From there, orchestrate with Microsoft Fabric deployment pipelines.

Option 3 with deployment pipelines


Performed via Git integration and Azure DevOps
Performed via Fabric Deployment pipelines
Pull request raised between Git branches
Supported items changed in Feature workspace
Feature branch changes merged with Development branch in Git repository
Supported items changed in Development workspace that is part of deployment pipeline
Deploy done via deployment pipeline
Deploy done via deployment pipeline
Supported items changed in Test workspace
Supported items changed in Production workspace

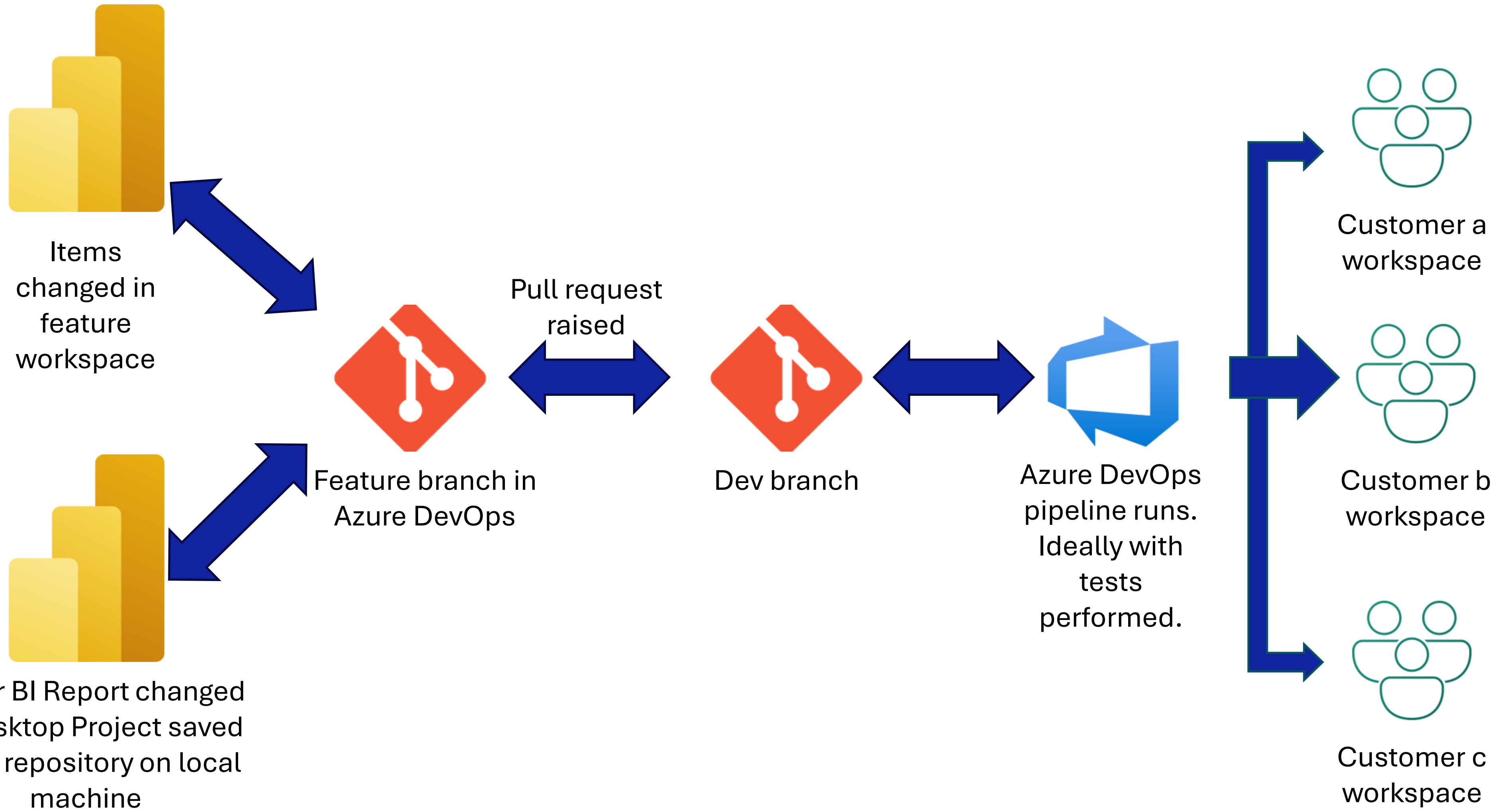
Deployment pipelines orchestrated by Azure Pipelines


Performed via Git integration and GitHub
Orchestrated by Azure Pipelines
Pull request raised between Git branches
Supported items changed in Feature workspace
Feature branch changes merged with Development branch in Git repository
Supported items changed in Development workspace that is part of deployment pipeline
Deploy done via Azure pipeline
Supported items changed in Test workspace
After approval deployment done via Azure pipeline
Supported items changed in Production workspace

Option 4 – For multiple clients/solutions/tenants

- Similar to option 2.
- All development and test done in same tenant.
- Then deployed to workspaces in other tenants via pipelines.
- Aimed mostly at ISV’s. However, useful for custom reports as well.

Option 4 diagram



Items changed in feature workspace
Power BI Report changed in Desktop Project saved in Git repository on local machine
Feature branch in Azure DevOps
Pull request raised
Dev branch
Azure DevOps pipeline runs. Ideally with tests performed.
Customer a workspace
Customer b workspace
Customer c workspace

Demos

- Option 2 - Fabric-cicd
- Option 3 - Deployment pipelines.

To summarize

- Applying DevOps practices can lead to fast and reliable deployments.
- Testing is essential, ideally at a very early stage.
- Recommend adopting Power BI Desktop projects.
- Experiment to see what release option suits your needs.

Questions

Thank you

Data Céilí

Kevin Chant

@kevchant

KevinRChant.com

kevchant