

WEBINAR | DECEMBER 10, 15:00-15:45 (CET)

CI/CD and version control for Microsoft Fabric & Power BI with Tabular Editor

With host Peer Grønnerup



Before we start

- Our team will be available to answer questions throughout the webinar. You can ask them by clicking "Questions" in the lower right corner of the screen.
- You can use the chat (also in the lower right corner) if you experience technical problems.
- We will send a recording of the webinar immediately after it ends.



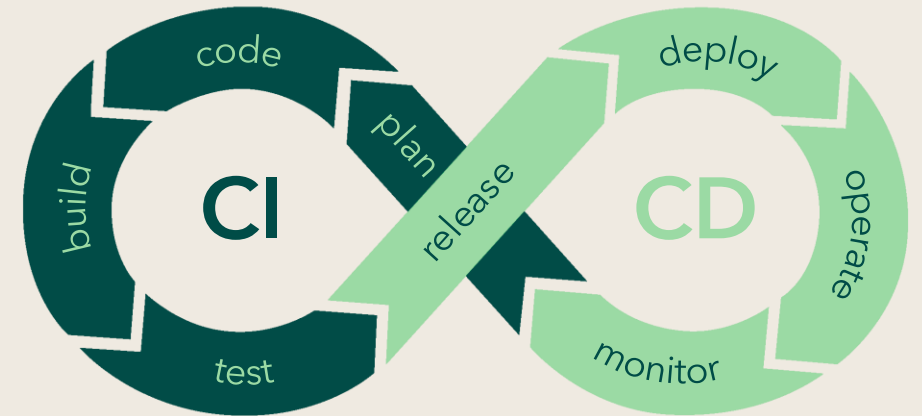
What we cover in today's webinar

1. CI/CD and version control basics and why does it matter?
2. Enterprise Data Platform Architecture in Fabric
3. Ways-of-working
4. Deployment
5. Wrap-up and Q&A

The basics of CI/CD and Version Control

CI/CD - The automation engine that powers DevOps workflows

- Git is the distributed version control for tracking changes over time
- Fabric supports GitHub and Azure DevOps as git provider
- Git uses commits, branches, and pull requests to manage work
- CI validates changes automatically (build, test, checks)
- CD packages and deploys changes consistently across environments
- CI/CD pipelines automate validation and deployment



Why CI/CD and version control?

The Foundation for Collaboration, Quality, and Controlled Releases

Collaboration & Parallel development

- ✓ Version control & tracking
- ✓ Work in parallel
- ✓ Isolate changes safely
- ✓ Limit risk of conflicts
- ✓ Clear ownership

Reviews & Governance

- ✓ Pull requests
- ✓ Structured code review
- ✓ Traceability
- ✓ Governance built-in

Testing & Consistency

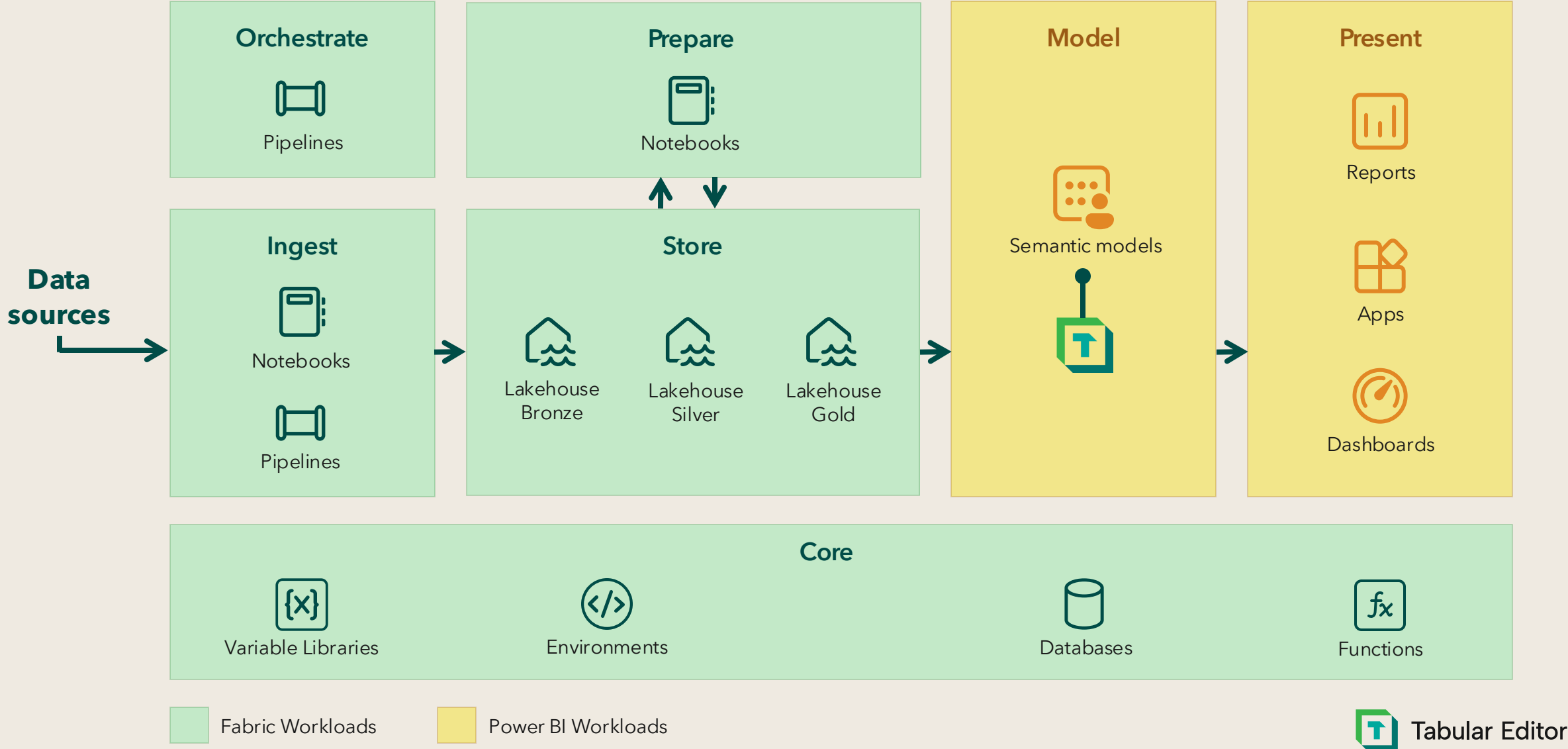
- ✓ Automated provision
- ✓ Automated test & validation
- ✓ Naming conventions
- ✓ Workspace standards
- ✓ Consistent security
- ✓ Supports experimenting

Automation & Tooling Integration

- ✓ Automated deployment
- ✓ Use of purpose built tools
- ✓ Reduce manual operations
- ✓ Achieve deterministic & repeatable releases

Enterprise Data Platform Architecture in Microsoft Fabric

Enterprise Data Platform - Reference architecture



Three Principles for Scale

Separation of Duties

- Layered workspaces
- Item organization
- Security and access
- Governance & compliance
- Capacity isolation
- Network & connectivity

Source-Controlled Development

- Clear mapping to repo folders
- Clear mapping to branches
- Purpose specific development
- Supports all items
- Caters for collaboration
- Supports specific WoW

Automated Deployments

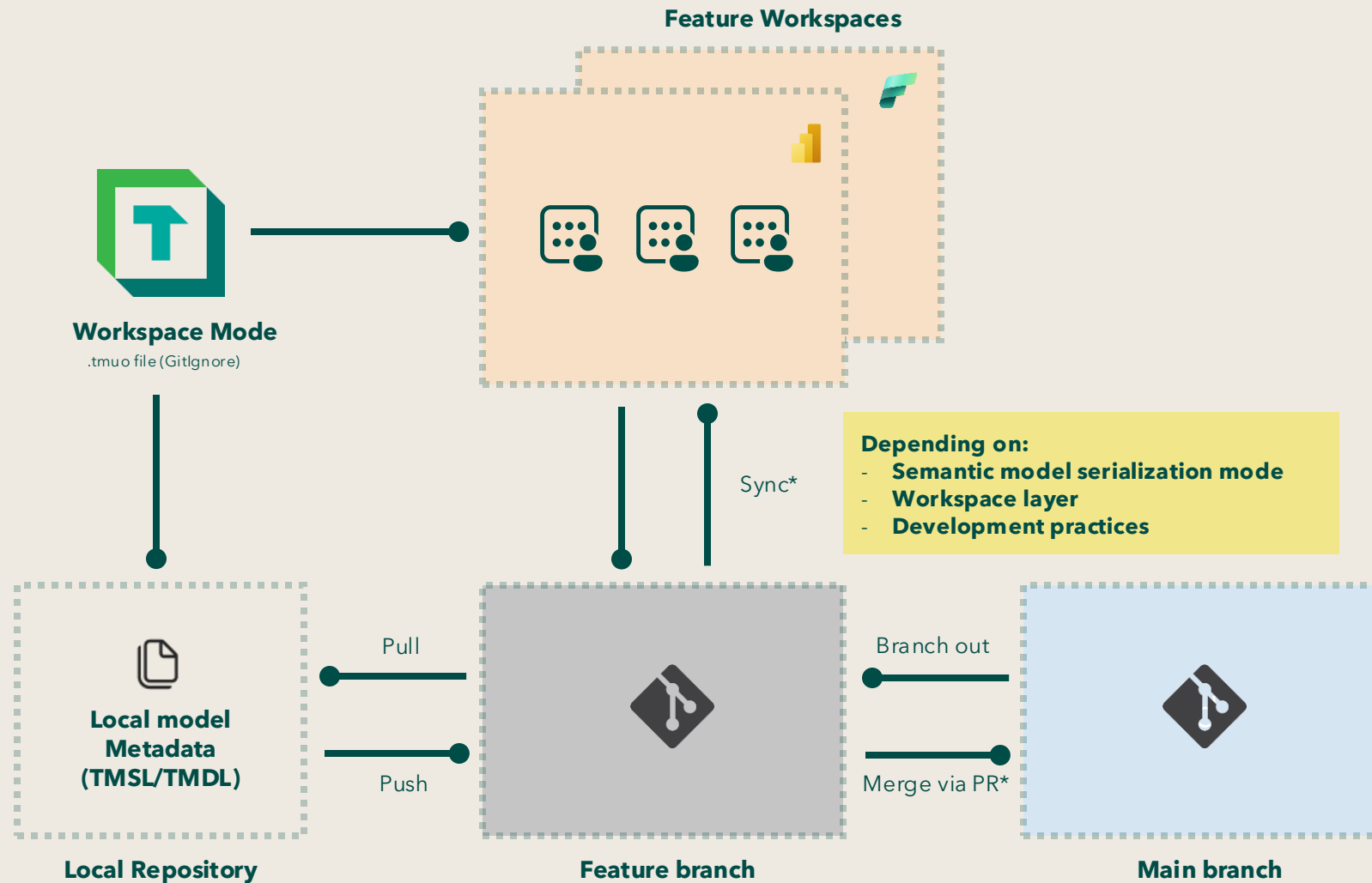
- CI/CD handles validation and promotion
- No manual pre- or post-deployment tasks required
- Re-usable and scalable
- Enforces quality

Demo

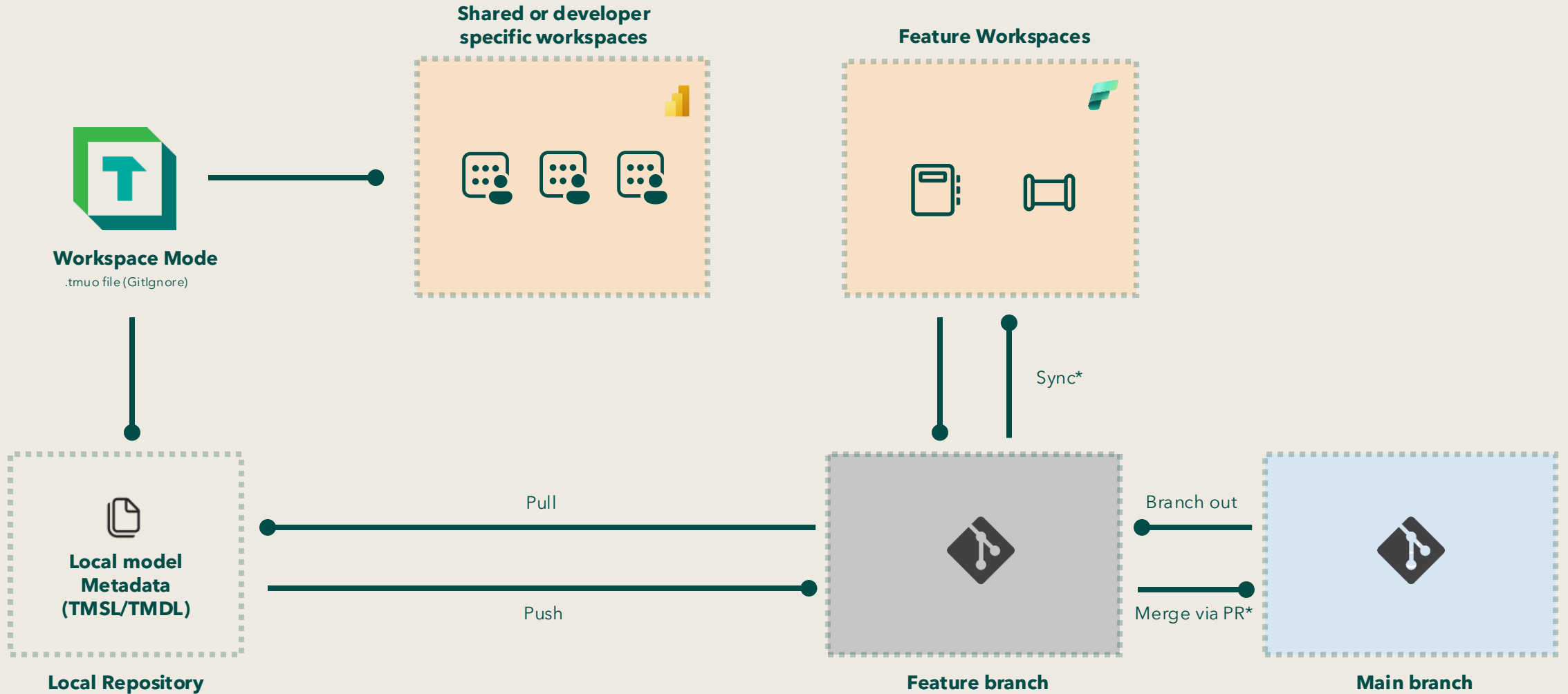
Platform architecture, git
integration and repo structure

Ways-of-Working in Microsoft Fabric

Ways-of-Working - Source Control with GIT and TE3



Ways-of-Working - Source Control with GIT and TE3



Semantic model serialization options

	TMSL	TMDL	Save to folder
Format	JSON	YAML like format	Object-based JSON
Granularity	Model	Component	Object*
Reusability / Modularity	Limited	Strong	Strong
Compatibility	All	SSAS 2016+, AAS, PBI/Fabric	TE only
Git maturity	Low	Medium	High

* Configurable

Demo

Feature development, review etc.

Deploying your solution to Fabric

One goal - multiple options

Fabric Deployment pipelines

No code experience

For simpler solutions

No support for:

- Pre-deployment opr.
- Post-deployment opr.
- Test & validation

Git-based deployment

Suitable when using Gitflow

Each environment connected to git branch

No need for building environments

Might require post-deployment operations

Git with build environment

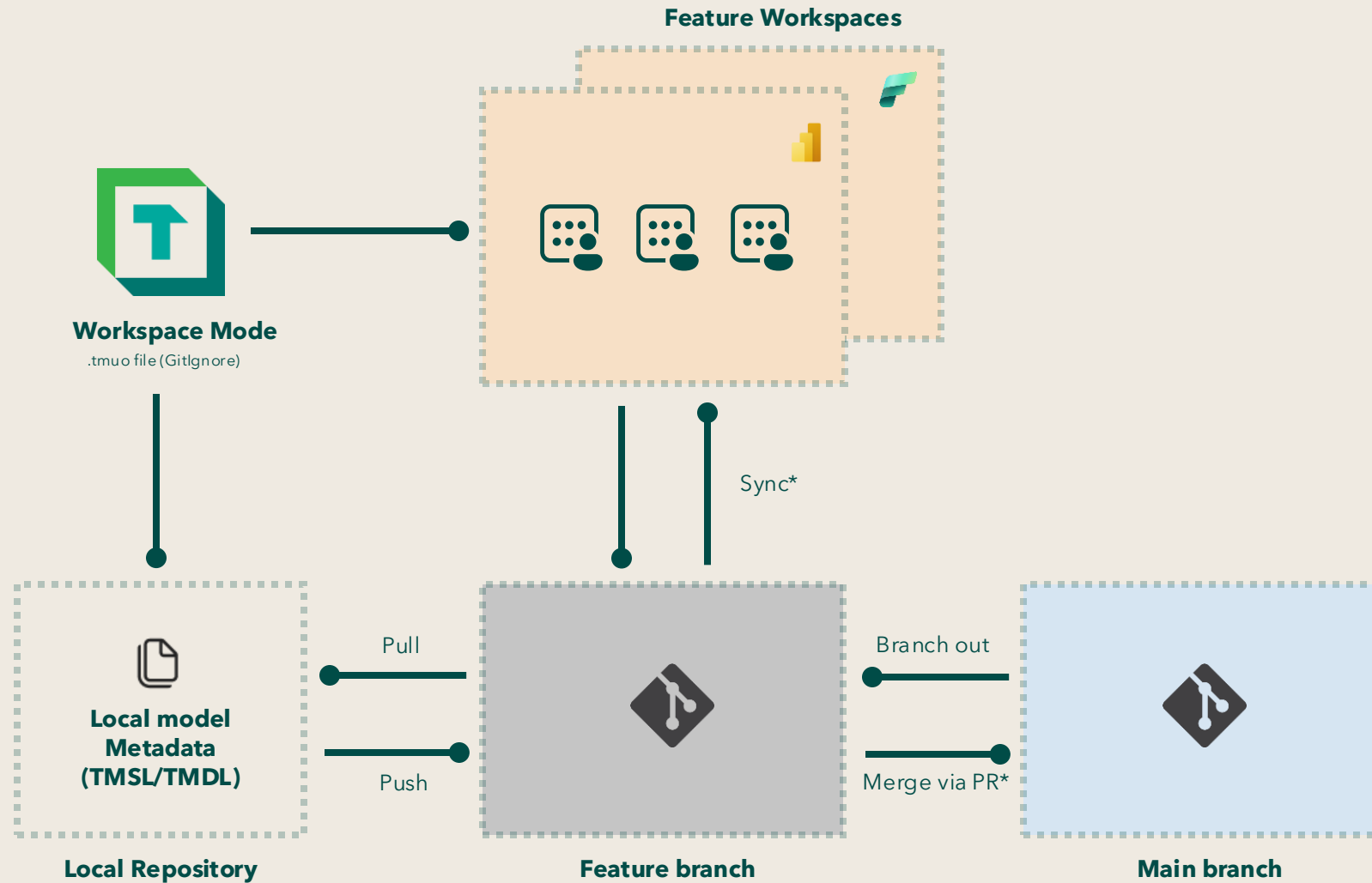
Supports building environments

Deployment through ADO Pipelines/Github actions

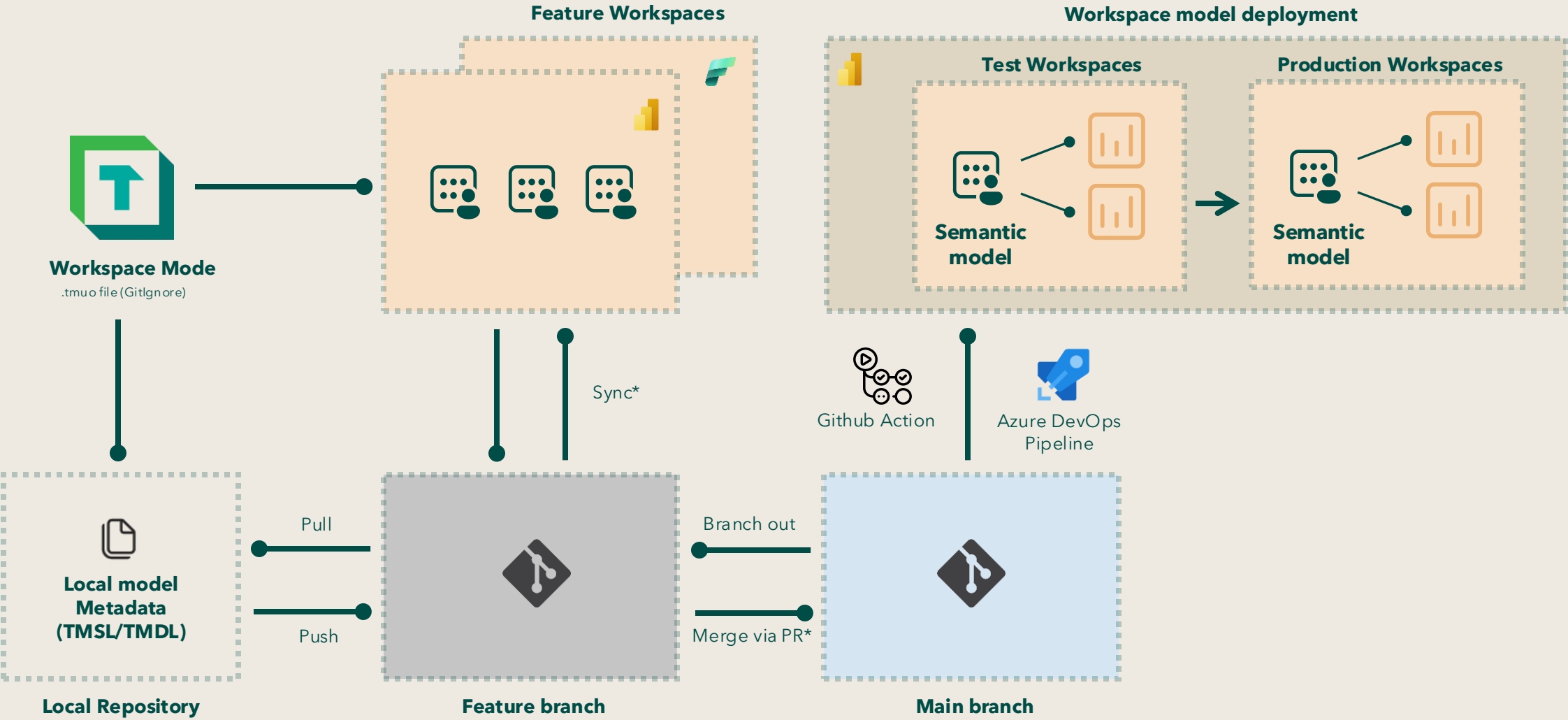
Supports manipulation of item definitions



Deployment



Deployment



CI/CD Building Blocks for Fabric



Tools and automation options

- Azure DevOps Pipelines
- Github actions
- TE CLI
- Fabric CLI
- Python
- Powershell

fabric-cicd Python Library

- Open source
- A code first approach
- Parameterization
- Semantic model binding
- Supports all items
- Auto unpublish

Demo

Enterprise Fabric deployment



Wrap up and time for questions

Take away recommendations

- Split your workloads and layers into multiple workspaces
- Use a mono-repo structure as a starting point
- Apply strict and consistent naming conventions
- Design with automation in mind
- Leverage the TE CLI, Fabric CLI, fabric-cicd python library, REST APIs and Terraform for automation
- Stay curious - Get inspired by what others build!
- Choose a branching strategy that suits your team
- Do **not** enable Git Integration in the Fabric workspace(s) hosting your workspace databases
- You save-to-folder to minimize risk of conflicts
- Implement test and validation as part of your CI/CD flow
- Implement automation supporting ways-of-working

Want to learn more?



Tabular Editor Insights

A comprehensive list of blog posts on CI/CD and git integration



<https://tabulareditor.com/blog/tag/ci-cd>

Other relevant references

- **Fabric CLI**
- **fabric-cicd**
- **Microsoft Learn:**
[Best practices for lifecycle management in Fabric - Microsoft Fabric](#)

[Azure DevOps build pipeline integration with Power BI Desktop projects - Power BI](#)



Thank you
for your time!