Git Good!
Best Practices for CI/CD & Collaboration in POWER BI

Peer Grønnerup

A big thank you to our partners

plainwater de kracht van heldere data

webdashboard Powered by In Summa

sogeti Part of Capgemini

DashData power to your people

BEILO BI ZEELAND

infotopics apps for power bi

Motio

Kimura

FabriCode</>

SIGNON ICT TRAININGEN+

valcon

KASPAROV FINANCE & BI

ilionx

creates.

nine altitudes

Tabular Editor Better Data Models Faster

raedt-BI

OKVIZ

Blauwdruk BI

THE DATA COOKS

DataShareHub

easydash

DTX

GET RESPONSIVE

DataHues.

Quanto collective analytics

ANOTHER DIMENSION

Control Over Reports

DATAKINGDOM

ONE PORTAL

DataCentral

dexs

Barbanson Data Solutions

mountdata guide to impact

Fellowmind

DATATAKO

AXELIO

MINOVA

PBIG POWER BI GEBRUIKERSGROEP

GIT GOOD!

BEST PRACTICES FOR CI/CD & COLLABORATION IN POWER BI

WHO AM I?

Peer Grønnerup

Head of Engineering at Tabular Editor

+15 years working with BI, Data Platform design & automation

Part of the Fabric Private Preview (Project Trident)

… and all in on Fabric & Power BI Automation and CI/CD!

https://www.linkedin.com/in/peergroennerup/

https://peerinsights.emono.dev/

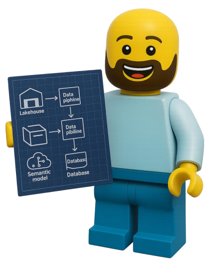
ACCESS THE FABRICOPS REPOSITORY

https://github.com/gronnerup/FabricOps

FabricOps

Disclaimer:
The solution demonstrated in this session is provided for **demonstration and educational purposes only**. It is **unsupported**, and there are no guarantees regarding functionality, stability, or future updates. You are free to use it **as-is** or modify it to fit your needs. Use at your own risk.

WHERE ARE WE GOING – GIT GOOD…

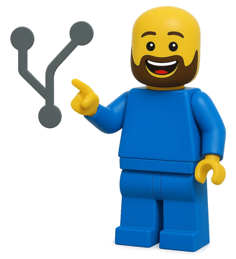


Architecting for scale

→

Workspaces, repos & environments

→



Supporting for real-world collaboration

→

Automating deployments

→

Tips, Tricks & Summing up

✓

- Architecture & platform setup
- Git integration & repo structure in Fabric/Power BI
- Ways-of-working & collaboration
- Touch branching strategies & release flows
- CI/CD pipelines
- fabric-cicd library and Fabric CLI
- Automation in general

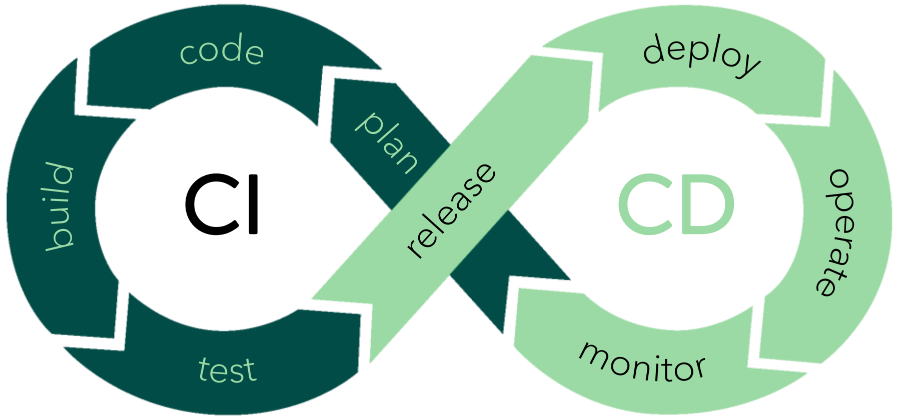
⊖

- Fabric solutions development & items
- How to implement dynamic code
- The Fabric Terraform Provider,
- The Fabric REST APIs or CLI in depth
- Fabric Deployment pipelines
- Security & governance
- Cover branching strategies in detail

BASICS OF CI/CD & VERSION CONTROL

CI/CD – The automation engine that powers DevOps workflows

- Git is the distributed version control for tracking changes over time
- Fabric/Power BI supports GitHub and Azure DevOps as Git providers
- 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

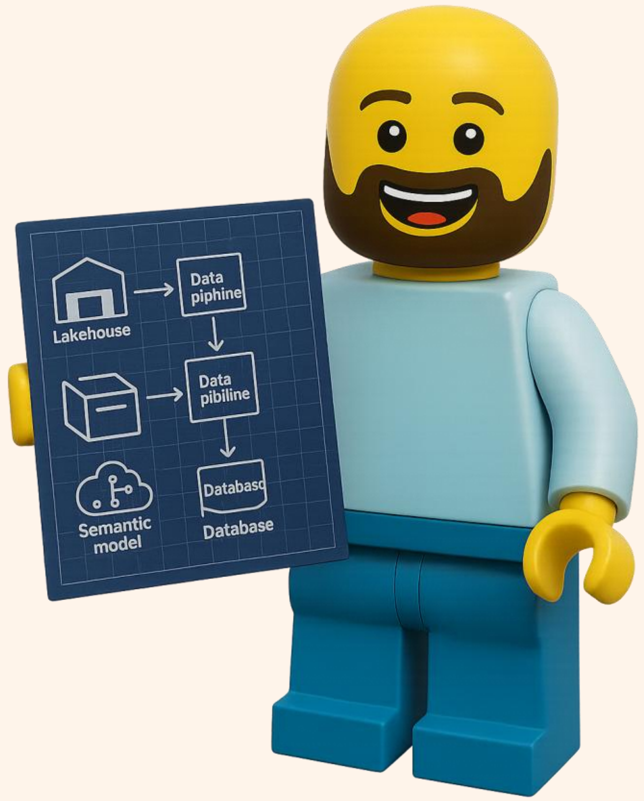
Testing & Consistency

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

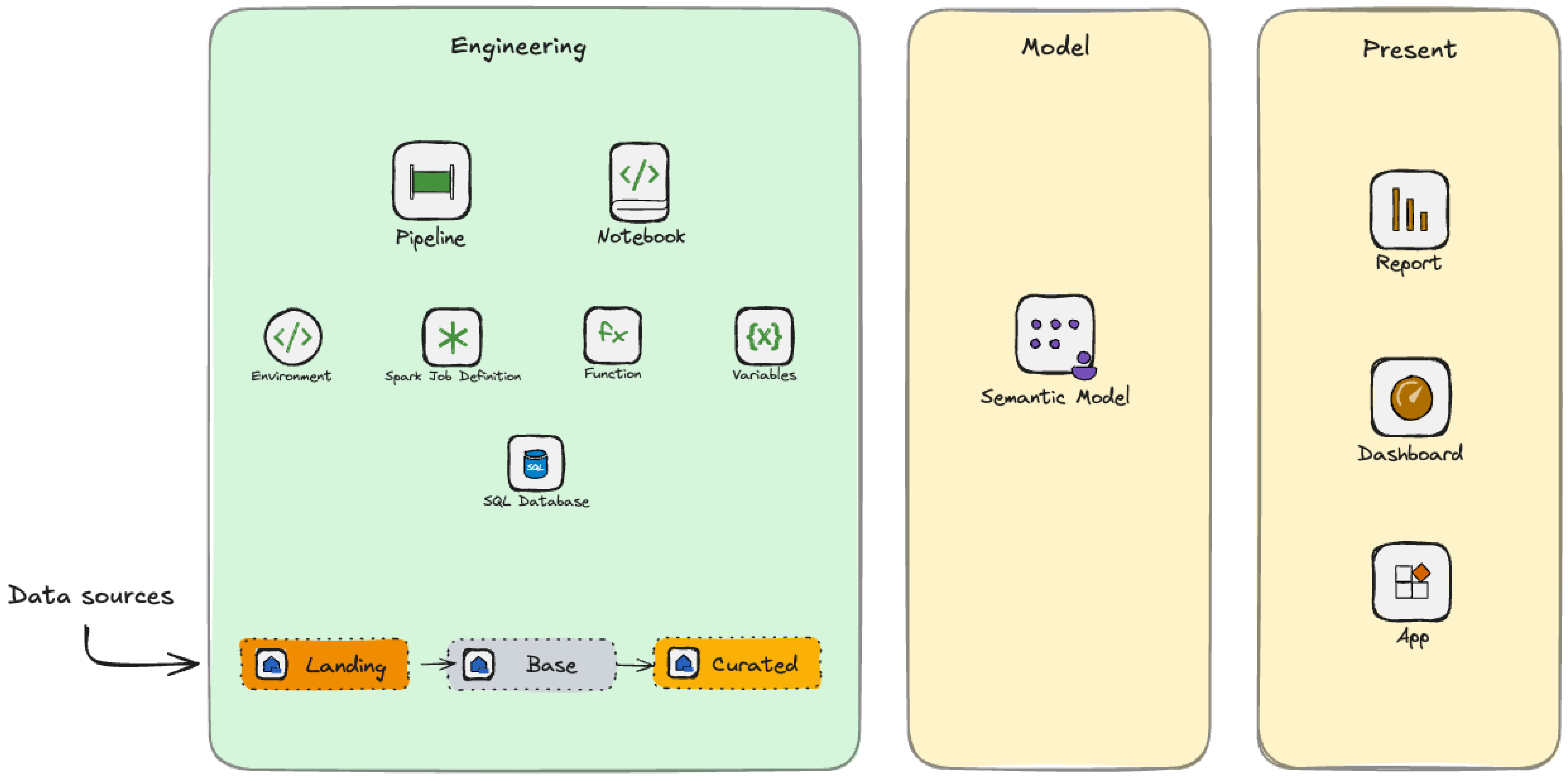
Automation & Tooling Integration

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

ARCHITECTING FOR SCALE



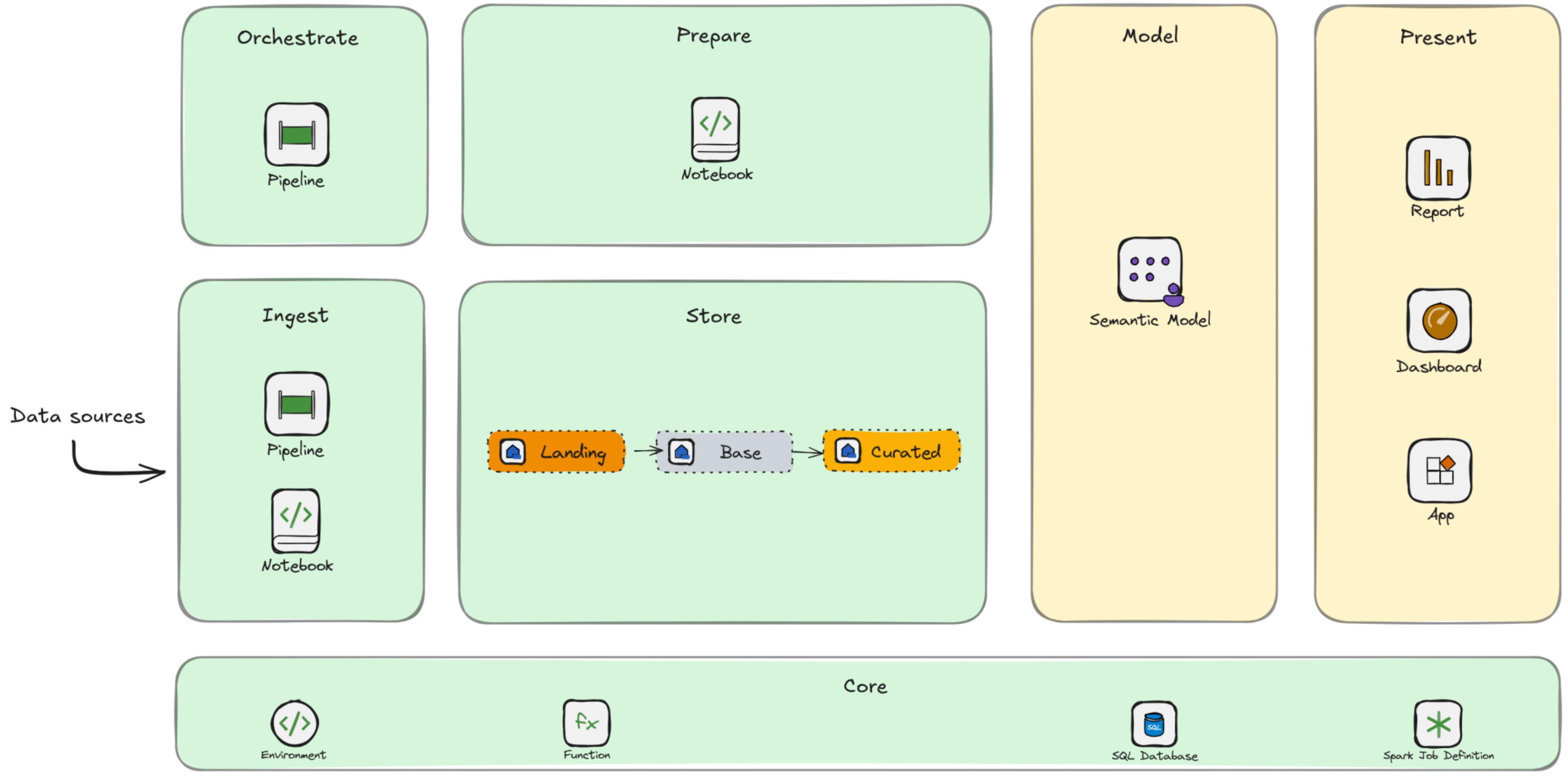
REFERENCE ARCHITECTURE - BASIC



Fabric workloads

Power BI workloads

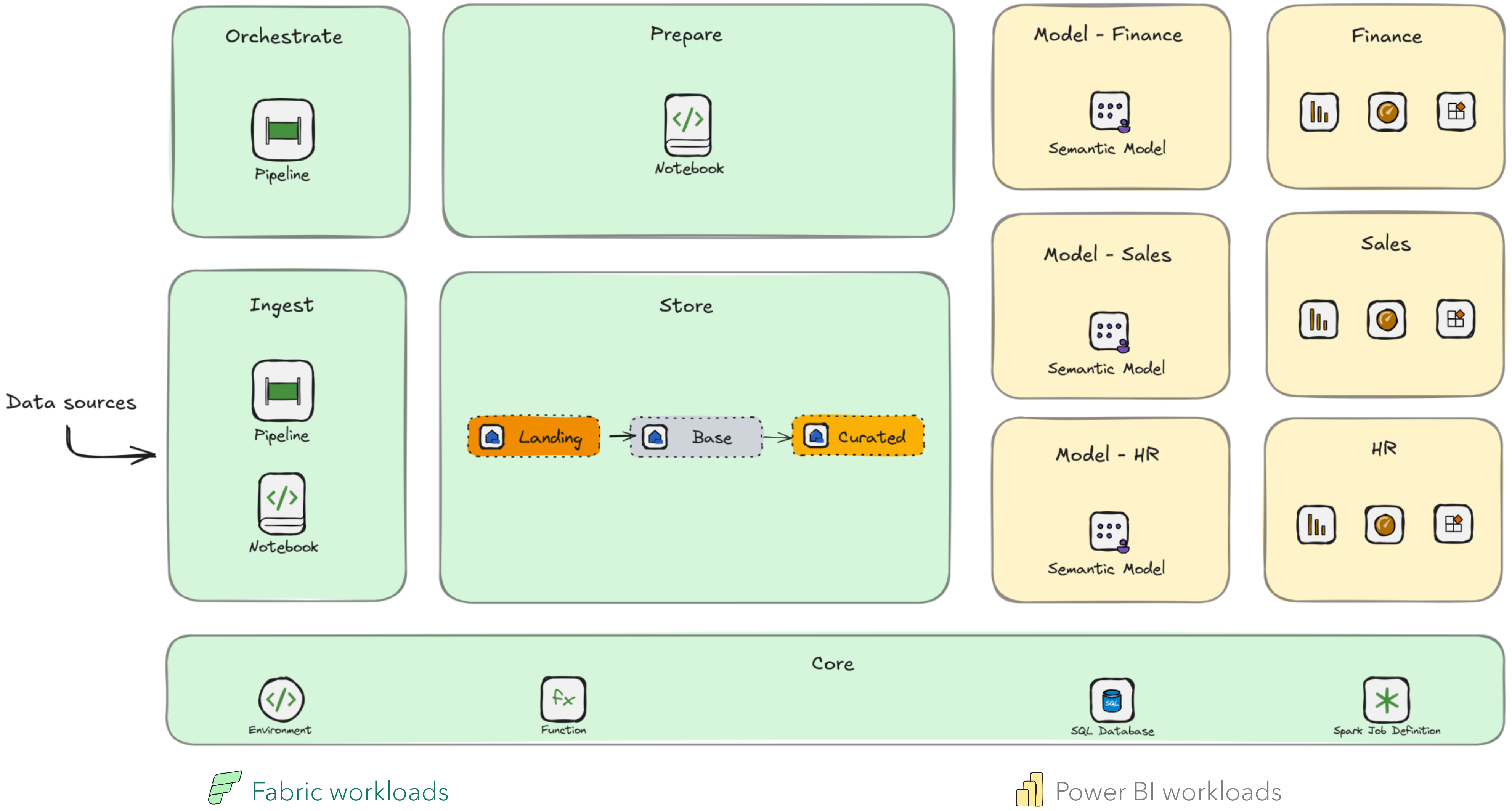
REFERENCE ARCHITECTURE - ENTERPRISE



Fabric workloads

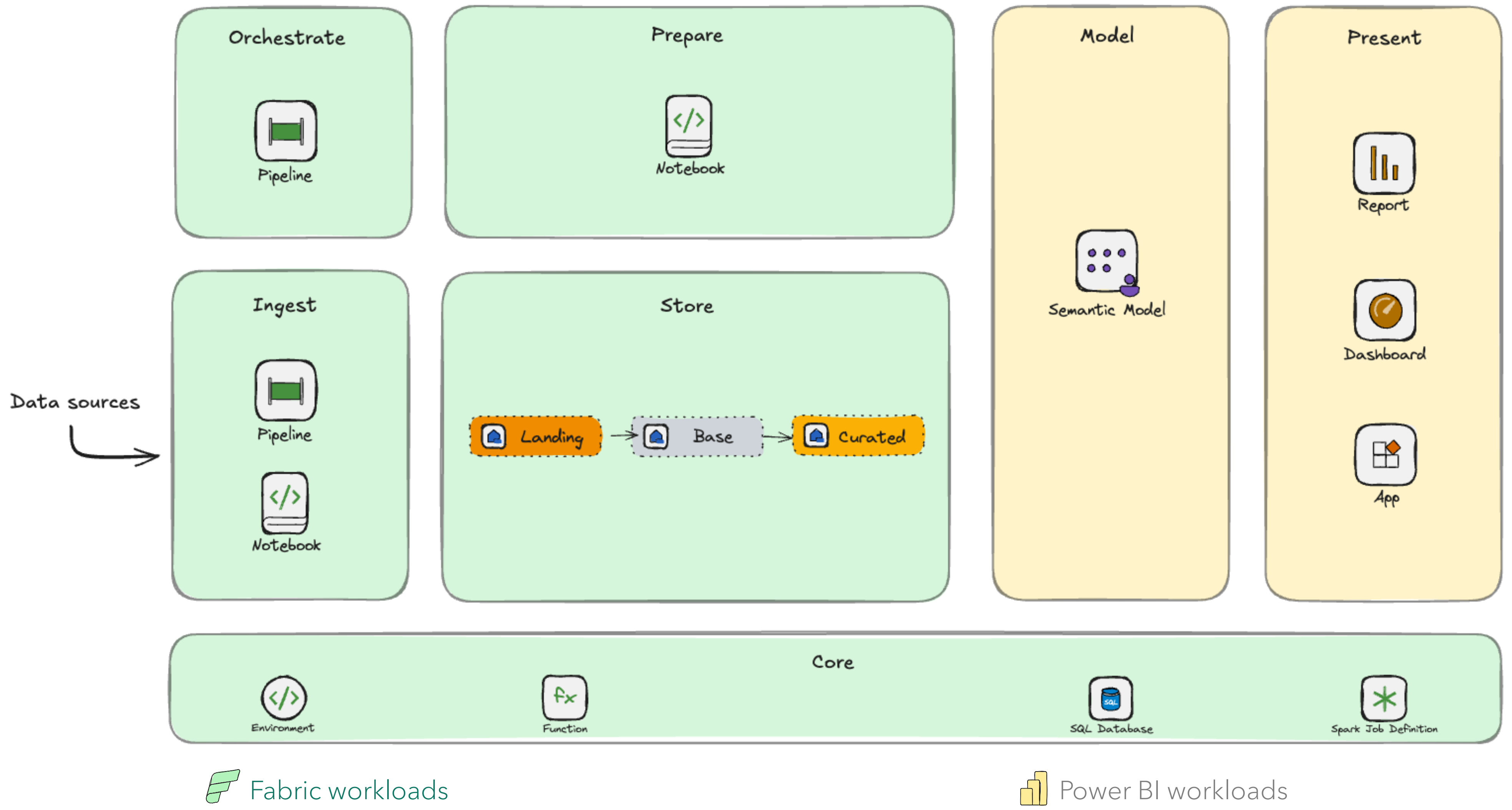
Power BI workloads

REFERENCE ARCHITECTURE - ENTERPRISE+

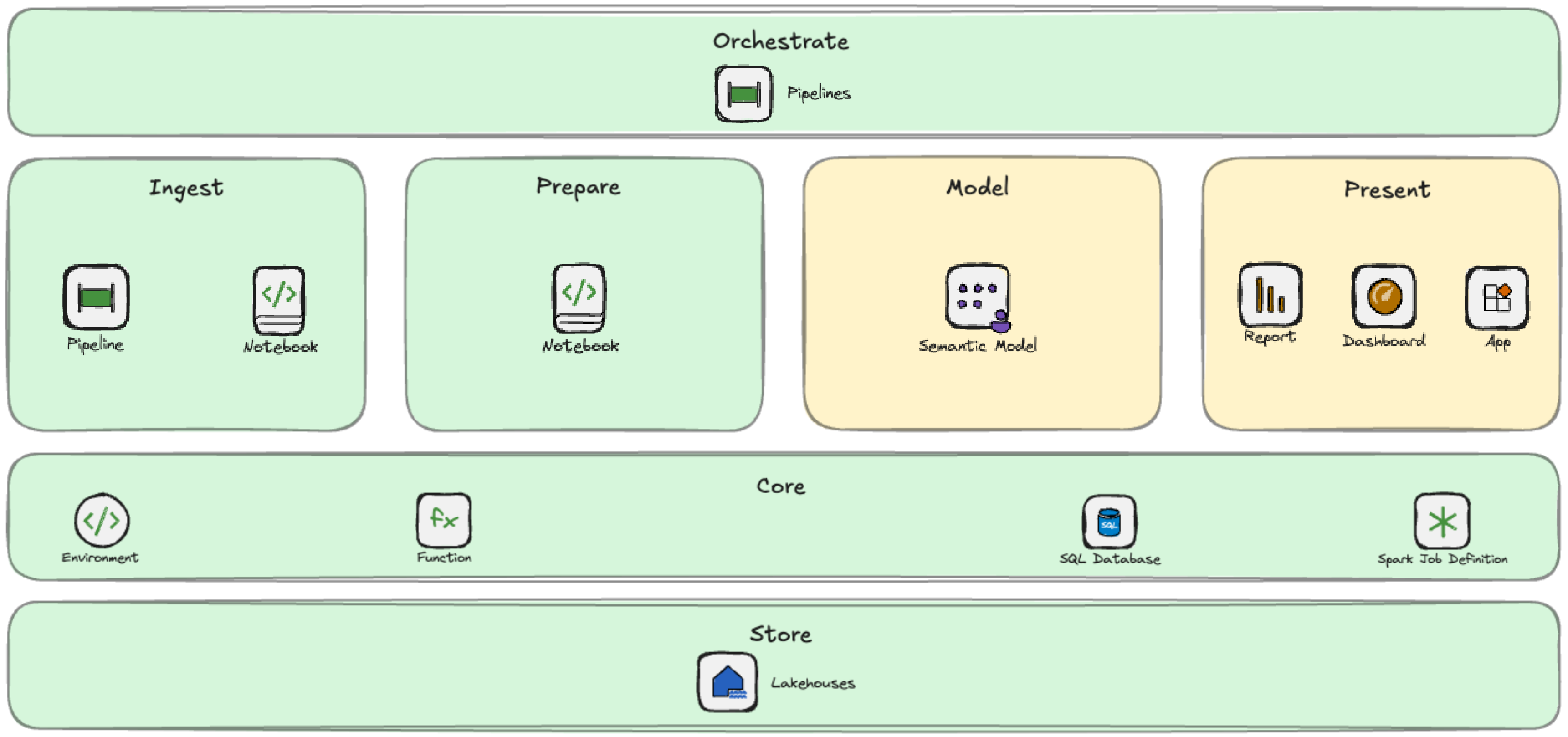


WORKSPACES, REPOS & ENVIRONMENTS

REFERENCE ARCHITECTURE - ENTERPRISE



FABRIC WORKSPACE STRUCTURE



x3

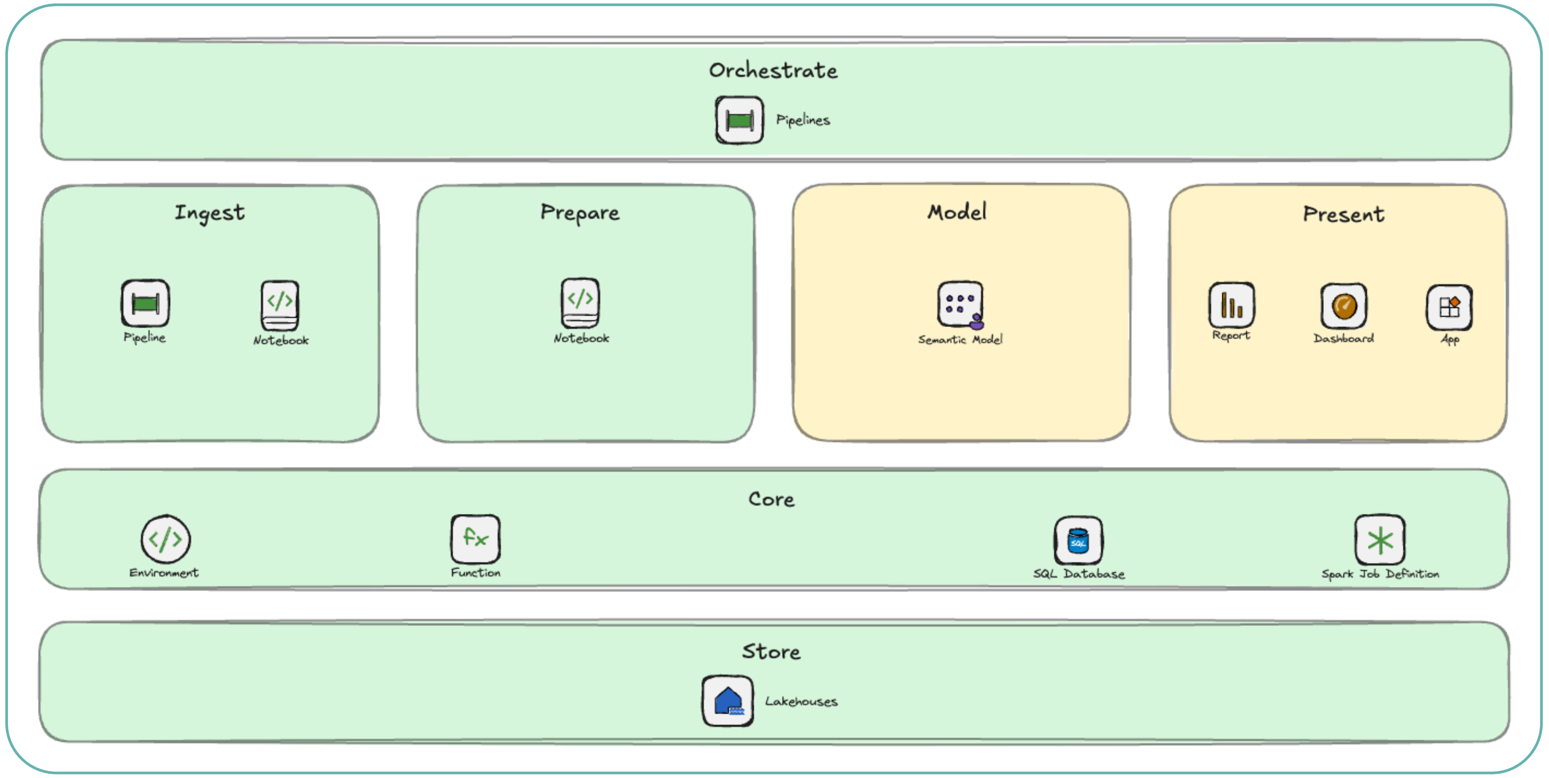
[dev, tst, prd]

Fabric workloads

Power BI workloads

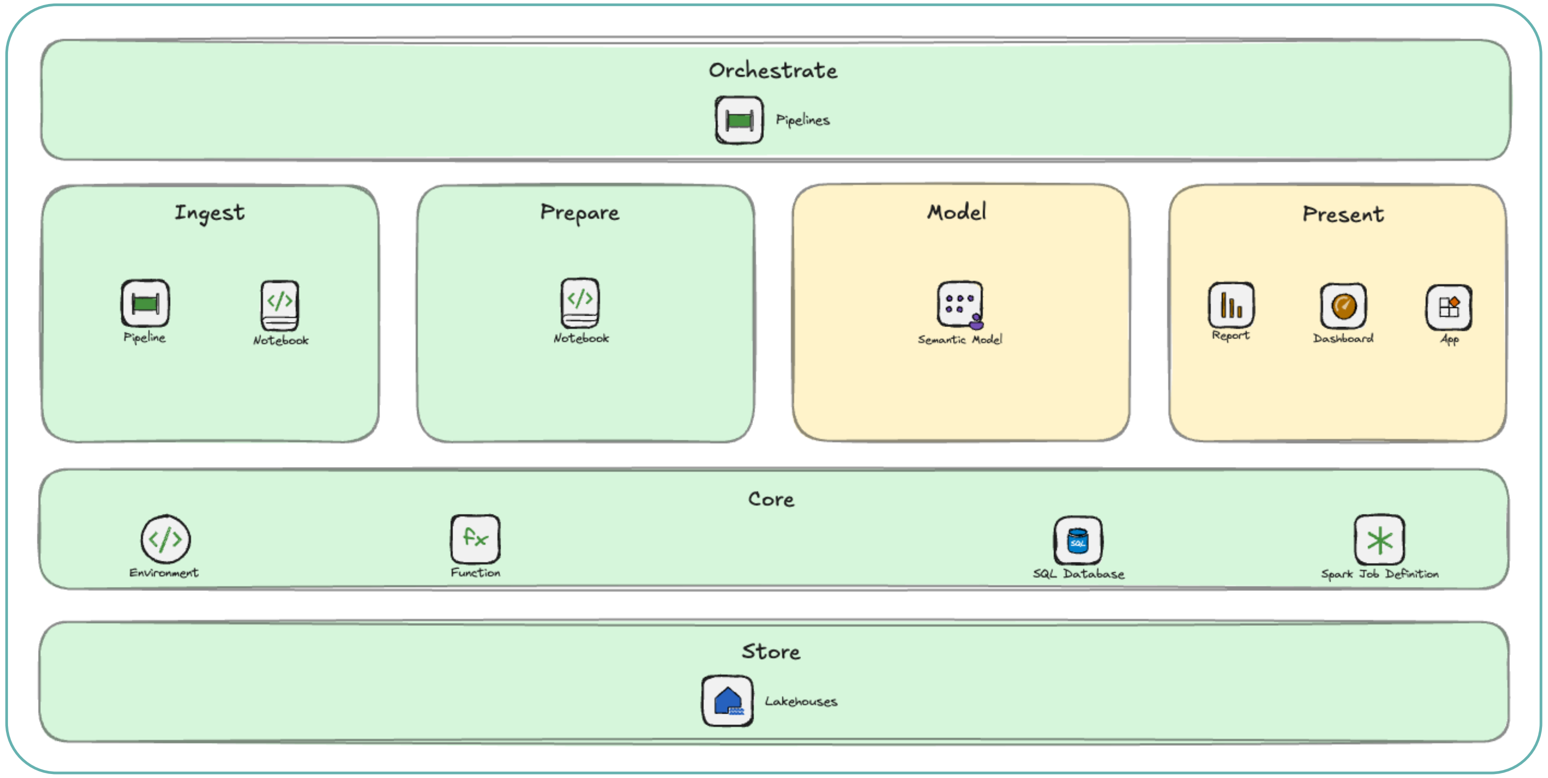
FABRIC WORKSPACE STRUCTURE

- Security and Access
- Separation of duties
- Network & connectivity
- Capacity isolation
- Governance & compliance
- Testing & Deployment
- Item organization
- Git integration and CI/CD



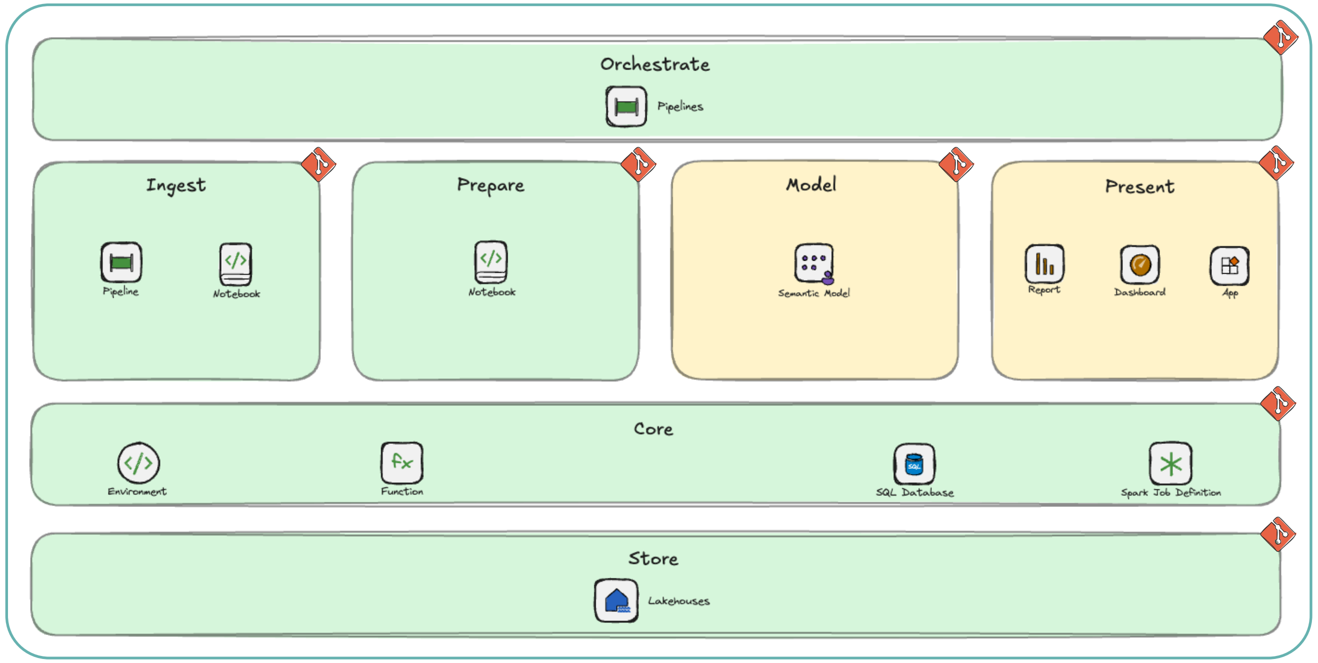
FABRIC WORKSPACE STRUCTURE

- Security and Access
- Separation of duties
- Network & connectivity
- Capacity isolation
- Governance & compliance
- Testing & Deployment
- Item organization
- **Git integration and CI/CD**



MY GO-TO FOR ENTERPRISE GIT SETUP

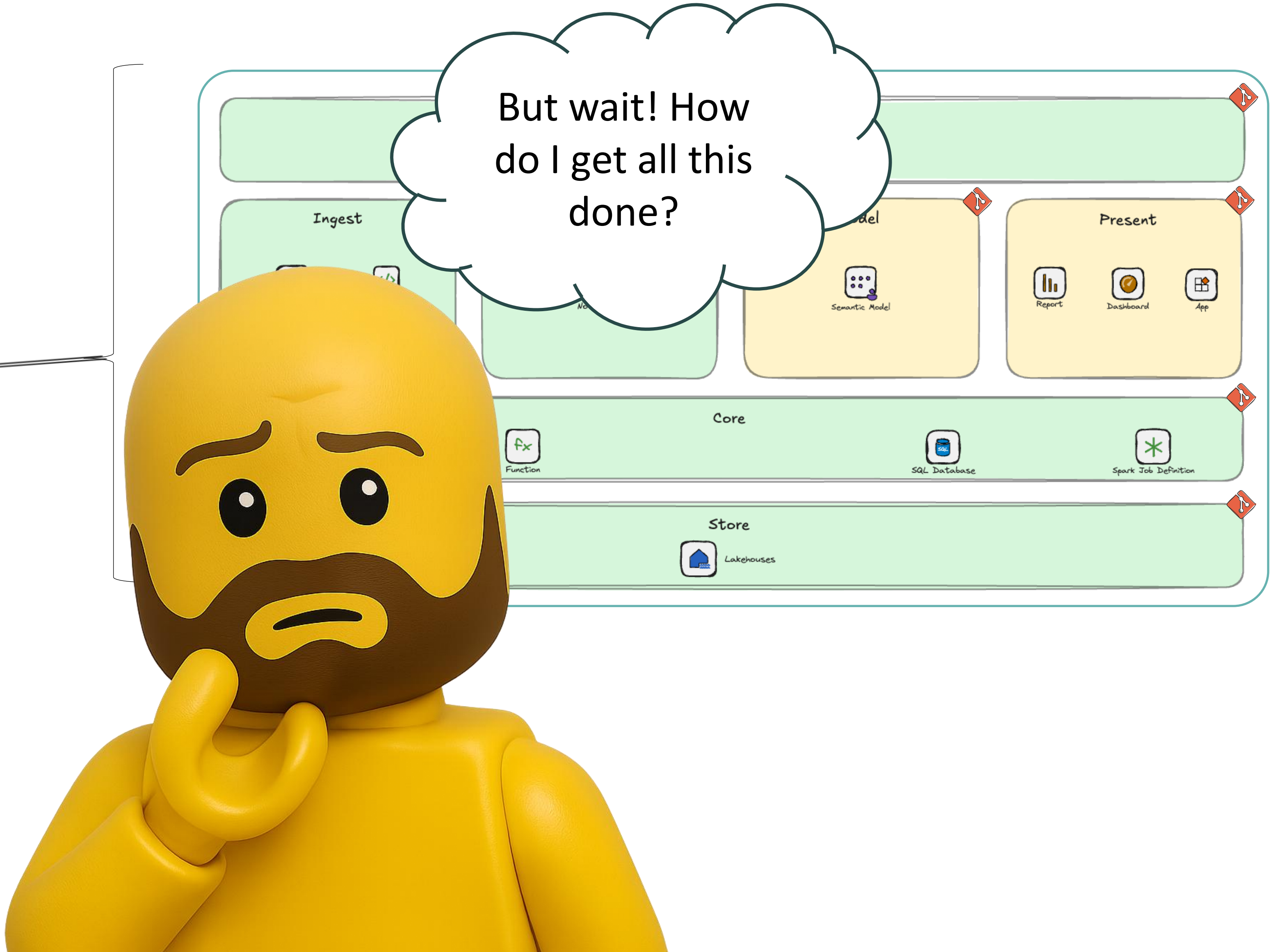
- Organize your workspaces by layer
- Use a clear naming conventions
- Use the Git provider of your choice
- Branching strategy depends…
- Setup Git integration on selected workspaces in development
- Use a mono repo approach



MY GO-TO FOR ENTERPRISE GIT SETUP

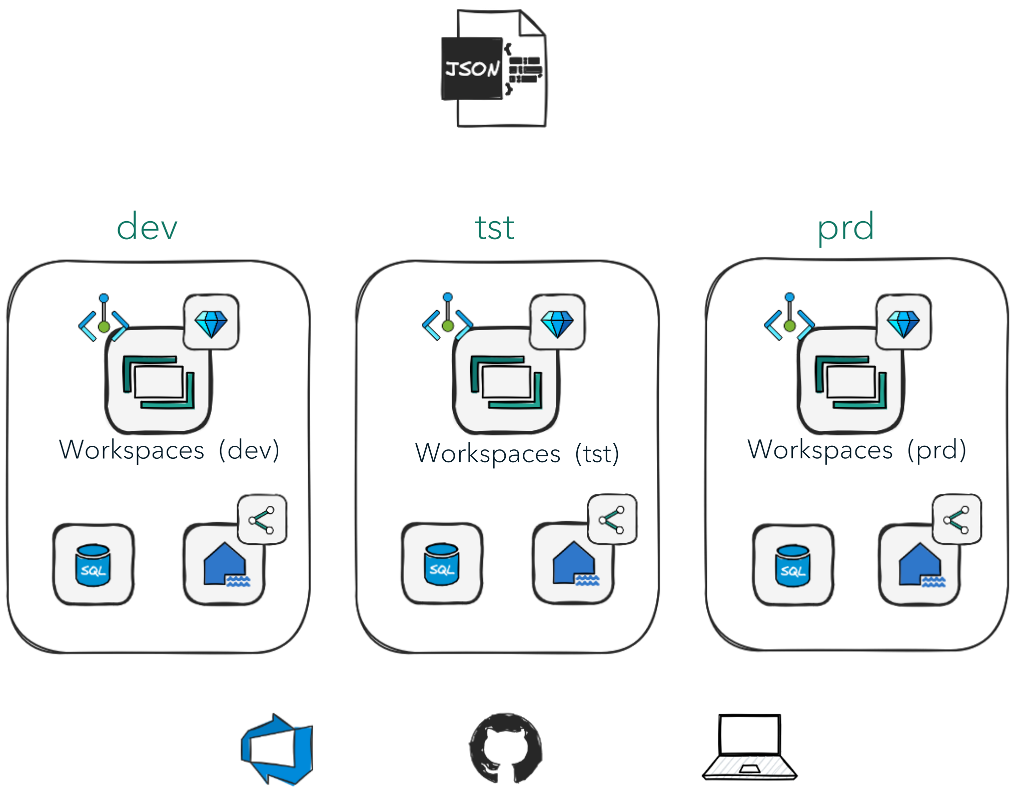
Git Repo structure

.azure-pipelines
.github
automation
documentation
solution
 /core
 /ingest
 /model
 /orchestrate
 /prepare
 /present
 /store



AUTOMATING SOLUTION SETUP

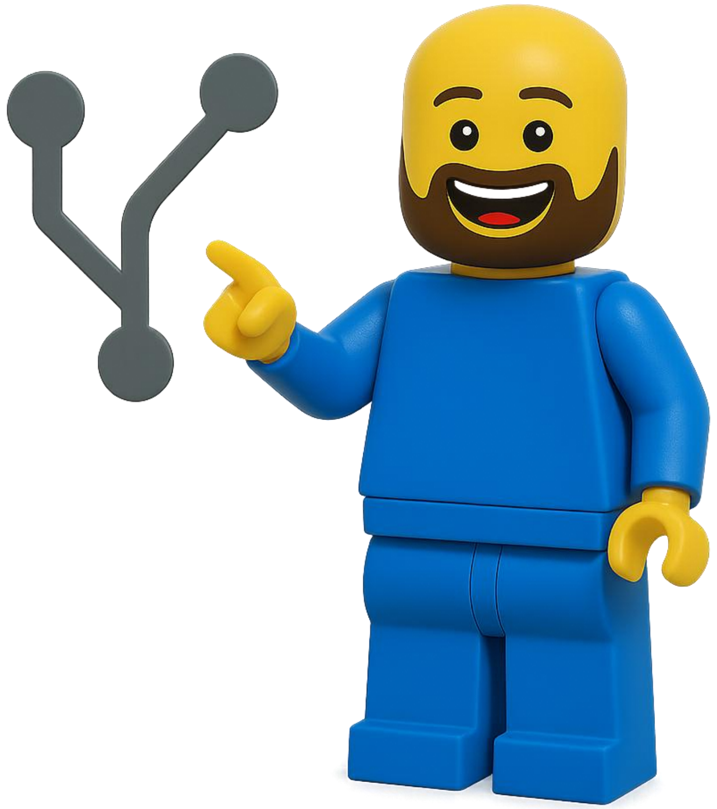
- Recipe based solution (json)
- Utilizes Python and the Fabric CLI
- Can run from Azure DevOps, GitHub or from any client machine running python
- Minimum requirements:
  - A Service Principal
  - A Fabric Capacity



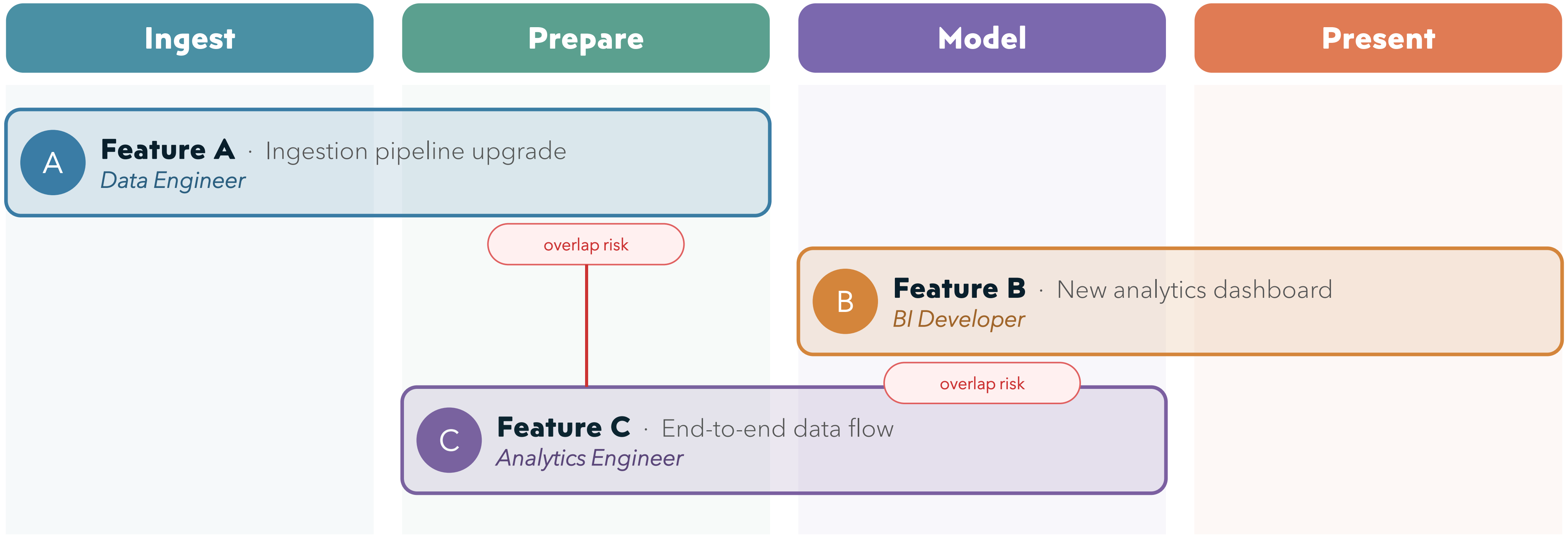
Github Download source code: https://github.com/gronnerup/FabricOps

SHOW TIME!

SUPPORTING REAL-WORLD COLLABORATION



HOW TO SUPPORT MULTIPLE DEVELOPERS?


Ingest
Prepare
Model
Present
A
Feature A · Ingestion pipeline upgrade
Data Engineer
overlap risk
B
Feature B · New analytics dashboard
BI Developer
overlap risk
C
Feature C · End-to-end data flow
Analytics Engineer

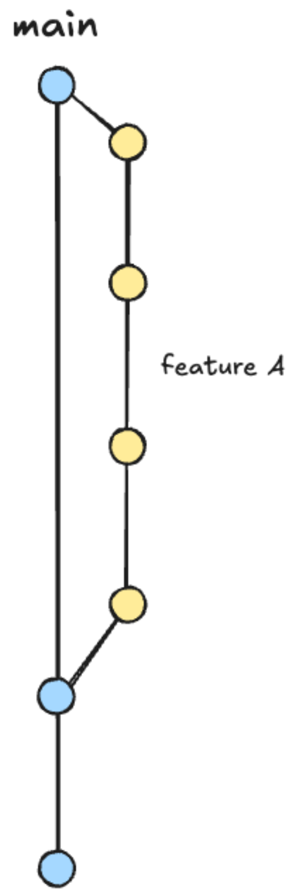
Real-world collaboration needs isolation **Branching is the answer**

Fabric Git Integration: Workspace linked to specific repository, branch and Git folder

BRANCHING – NOT JUST GIT HYGINE

It's a Collaboration Strategy!

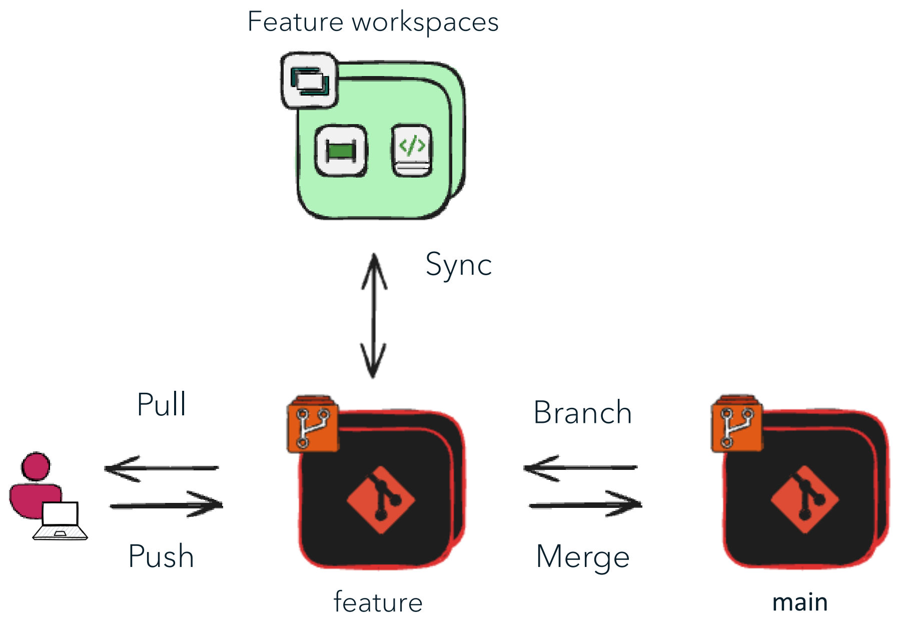
- Git is the source of truth, and branching is how we manage change
- Branches gives developers a seperate workspace for their code and…
    - Isolates development
    - Protects the mainline of our code
    - Enables parallel development
    - Help organize and structure releases
    - Is crucial for streamlined collaboration
    - Enables experiments



GENERAL DEVELOPMENT WORKFLOW

Development flow - Supported by automation!

1. Create new branch from main + isolated development workspaces
2. Implement feature/changes
3. Create PR and merge feature/changes to main
   - With possible validation steps
4. Delete feature branch + isolate development workspaces



https://learn.microsoft.com/en-us/fabric/cicd/best-practices-cicd

WHY AUTOMATE THIS FLOW?

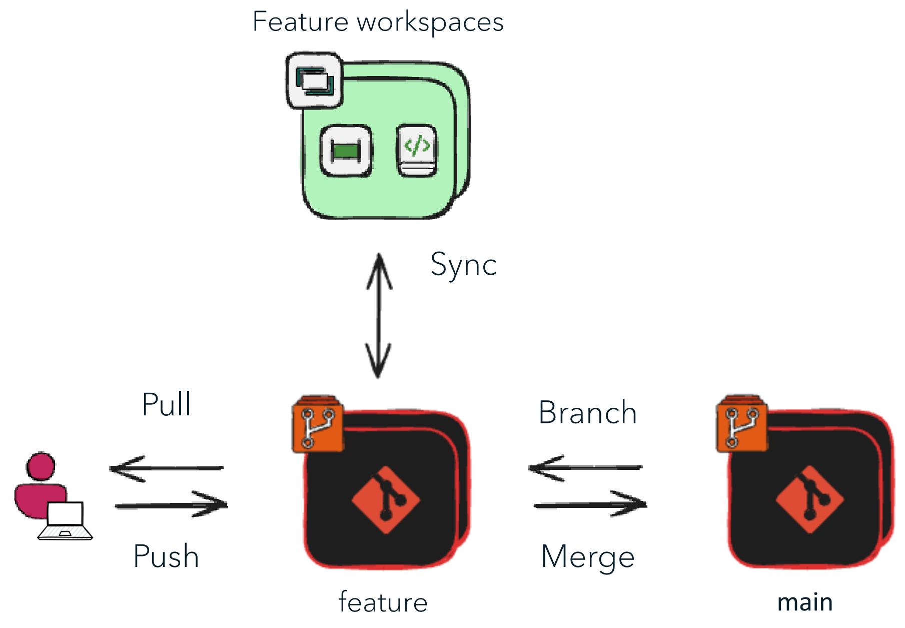
ClickOps

- ✗ Manual process
- ✗ 1 workspace – 1 branch
- ✗ No transfer/setup of ACL, Spark settings, Private Endpoints, WS Identity
- ✗ Inherits source capacity
- ✗ Requires manual cleanup

VS

Automated

- ✓ Fully automated – just create the branch
- ✓ Multiple workspaces supported per branch
- ✓ Customize ACL, Spark settings, Private Endpoints, WS Identity, Capacity
- ✓ Automated sync & cleanup



https://peerinsights.emono.dk/automating-feature-workspace-maintainance-in-microsoft-fabric

https://justb.dk/blog/2025/02/fabric-spark-notebooks-and-cu-consumption/

DEMO TIME!

CONSIDERATIONS ON MODELS & REPORTS

1. Often involves differenct developer tools and options
2. Live editing, online detached and locale development
3. Different formats (PBIX, PBIR, PBIP, TMSL, TMDL, Save to folder)
4. Developer testing environments – especially for semantic models

THE MAIN CHALLENGE

Fabric/Power BI Git Integration

SEMANTIC MODEL SERIALIZATION OPTIONS

	TMSL	TMDL	Save to folder* (TE)
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

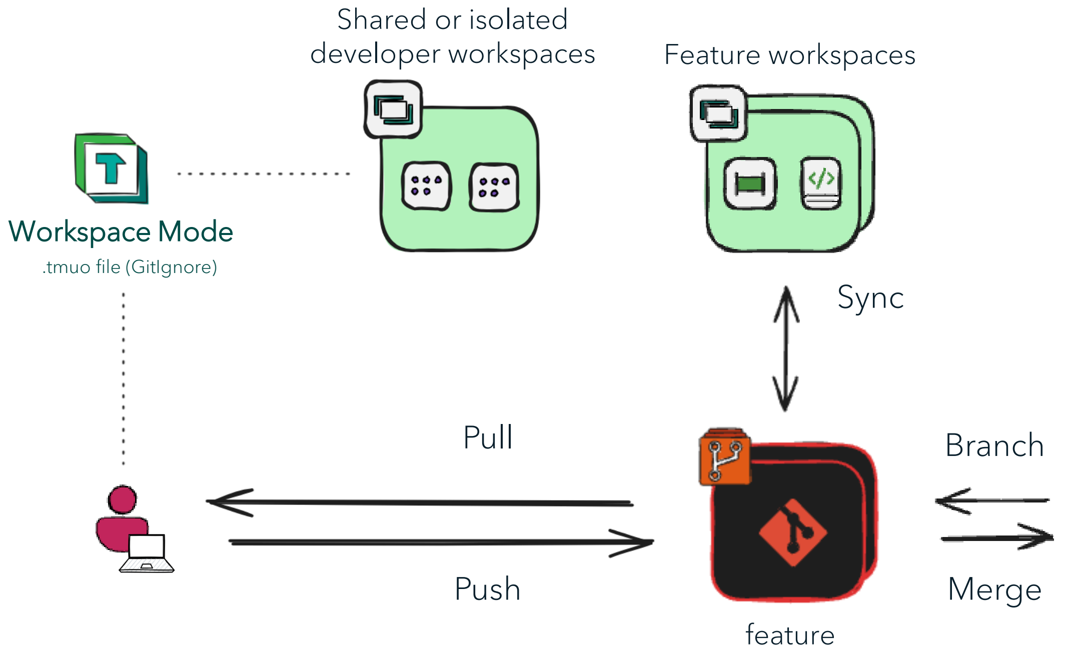
* https://docs.tabulareditor.com/features/save-to-folder.html

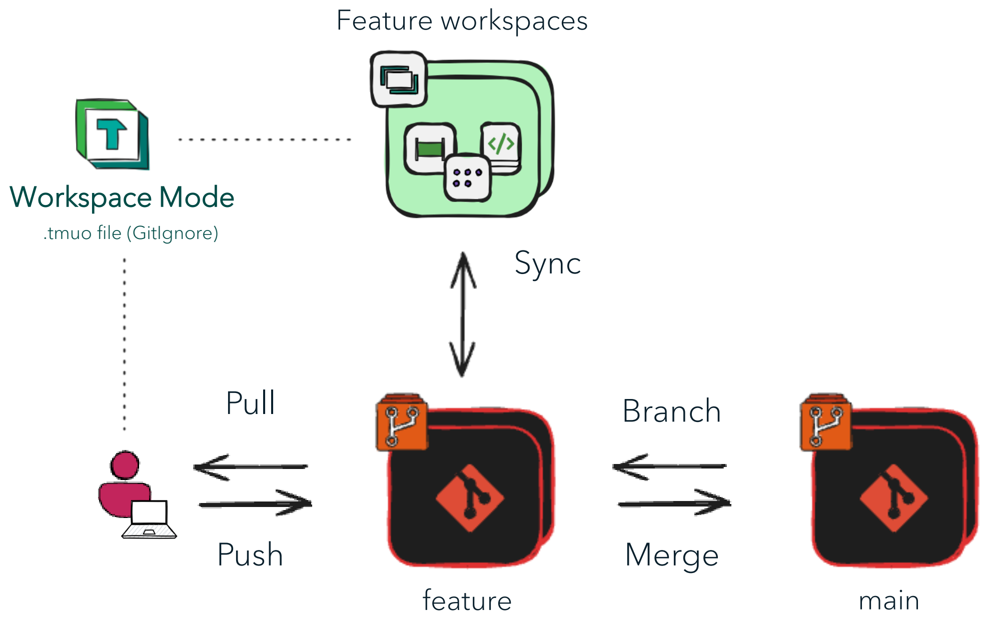
** Configurable

USING TABULAR EDITOR?

What works best? Well it all depends….

- Semantic model serialization mode
- Workspace layer
- Development practices
- Tools and features used





DEMO TIME!

AUTOMATING DEPLOYMENT

RELEASE PROCESS – THE MAIN 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 a dedicated git branch
- No need for building environments
- Might require post-deployment operations

Git based deployment using build environment

- Build environments
- Deployment through ADO Pipelines & Github actions
- Supports manipulation of item definitions

CI/CD BUILDING BLOCKS

Tools & automation options

- Azure DevOps Pipelines
- Github actions
- Python scripts & wrappers
- Fabric CLI wrappers
- Tabular Editor 2 CLI (free)

fabric-cicd Python Library

NOW OFFICIALLY SUPPORTED

Developer friendly

- Code-first approach
- No need to call Fabric APIs directly

Environment supports

- Parameterization
- Reusable for locale, DevOps and more…

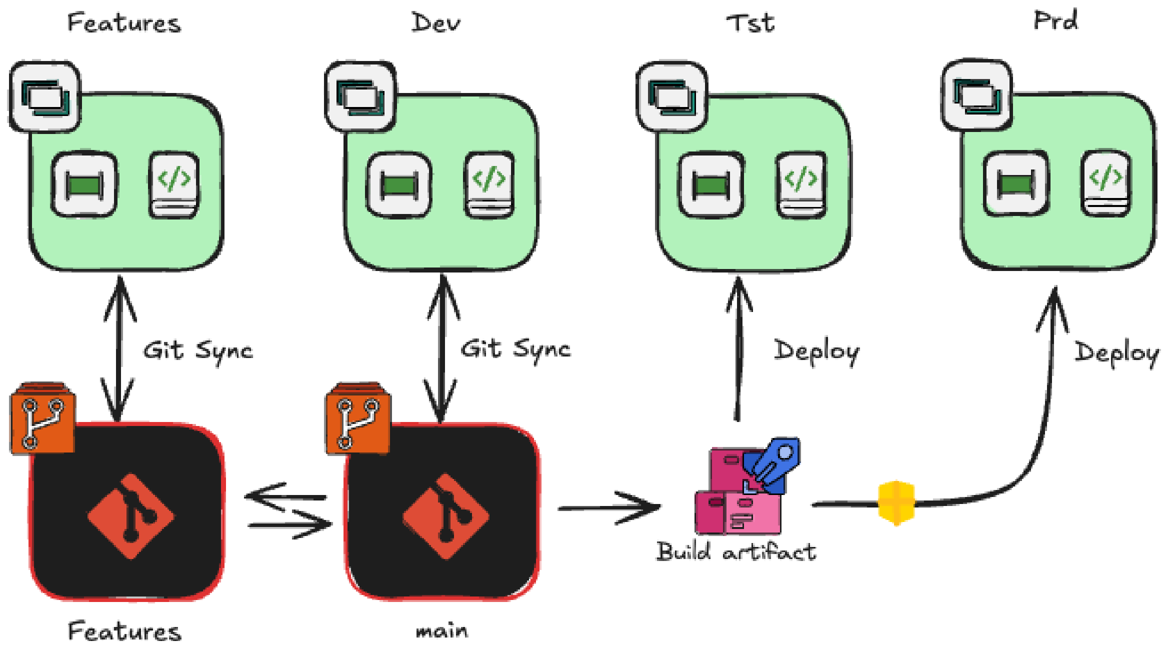
Smart deployment

- Deploy all items (with public APIs)
- Auto-unpublish orphaned artifacts

RELEASE PROCESS - 3 SELECTED OPTIONS

Option 1

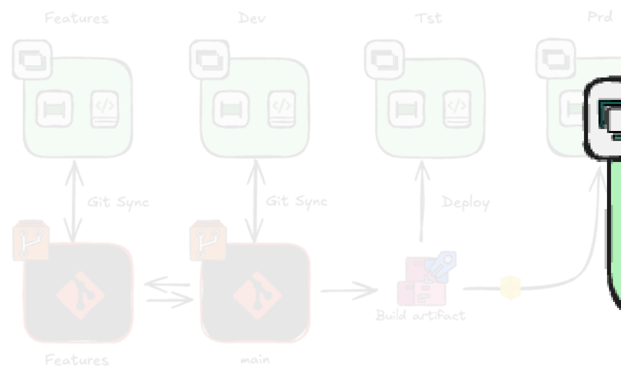
Dev = Main



RELEASE PROCESS – 3 SELECTED OPTIONS

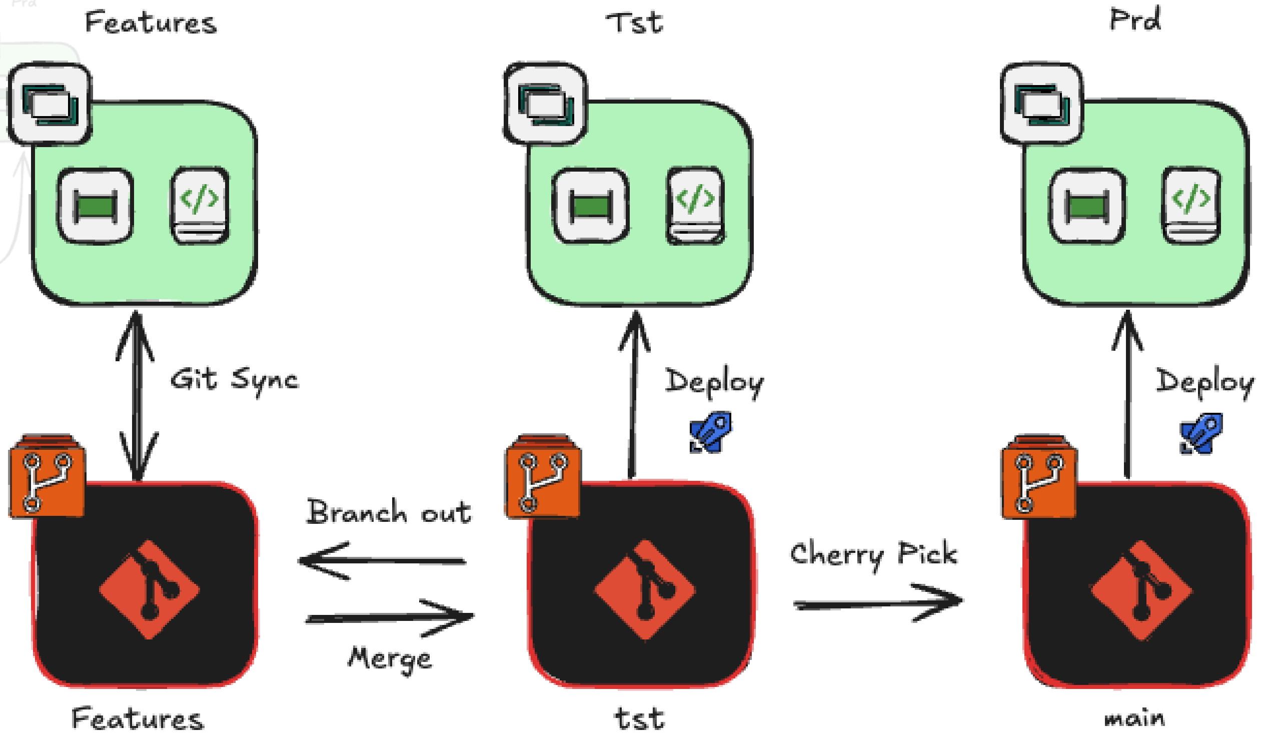
Option 1

Dev = Main



Option 2

PR to test – Cherry pick to main



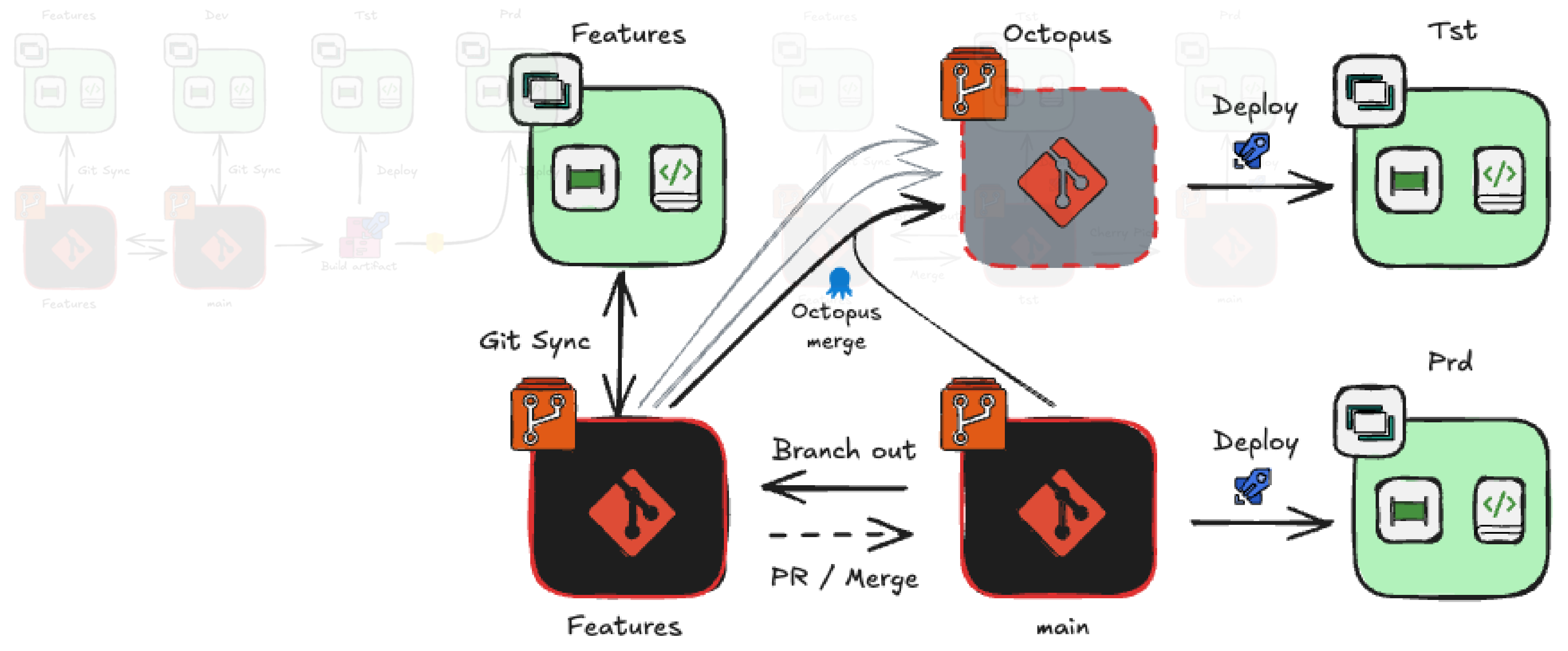
RELEASE PROCESS – 3 SELECTED OPTIONS

Option 1

Dev = Main

Option 3

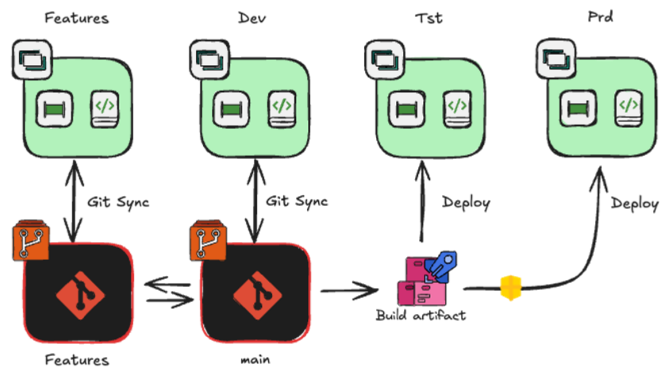
Octopus deploy



RELEASE PROCESS – 3 SELECTED OPTIONS

Option 1

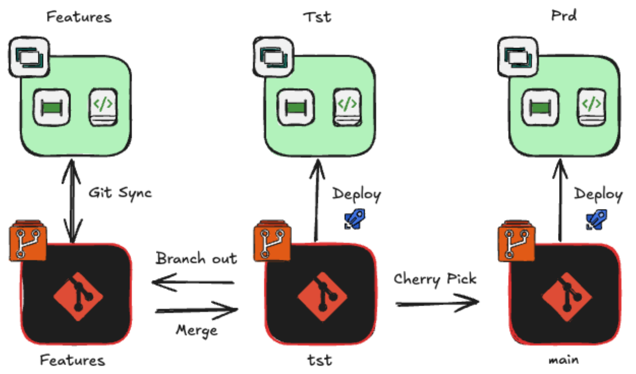
Dev = Main



- Smaller and more simple solitions
- Small teams
- No or very low interdependency between features

Option 2

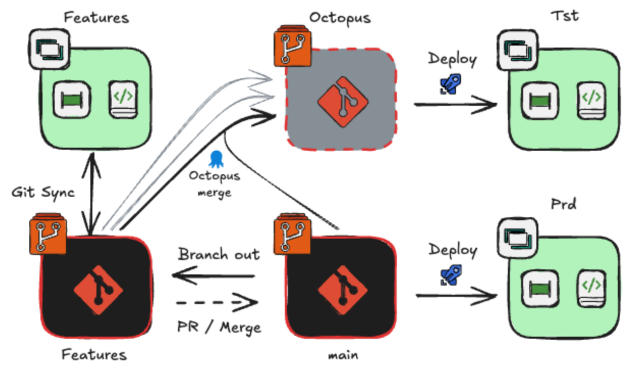
PR to test – Cherry pick to main



- Medium to large solutions
- High-risk changes
- Supports per-feature validation
- Supports incremental and selective testing and deployment

Option 3

Octopus deploy



- Large teams
- Many parallel features
- Interdependent changes
- High-risk changes
- Support validation and test

DEMO TIME!

TIPS, TRICKS & SUMMING UP

TIPS, TRICKS & SUMMING UP!

- Split your workloads and layers across multiple workspace
- Use mono-repo structure as a starting point
- Give thoughts to how to integrate workspaces with Git - Everything is not enterprise
- Leverage the Fabric CLI, REST APIs, and/or Terraform for automation
- Use the fabric-cicd Python library to deploy Fabric items
- Design everything with automation in mind
    - Invest in dynamic pipelines, notebooks and reusable patterns
    - Apply strict and consistent naming conventions
- Implement a metadata-driven framework for scalability robustness and speed
- Stay curious - Get inspired by what others build!

ACCESS THE FABRICOPS REPOSITORY

https://github.com/gronnerup/FabricOps

Disclaimer:
The solution demonstrated in this session is provided for **demonstration and educational purposes only**. It is **unsupported**, and there are no guarantees regarding functionality, stability, or future updates. You are free to use it **as-is** or modify it to fit your needs. Use at your own risk.

Loved it? Learned something? Tell us!
Share your feedback in just 1 minute

POWER BI GEBRUIKERSGROEP

TIME FOR QUESTIONS!

BEFORE CONNECTION IS TERMINATED BY PEER...