

FABRIC AUTOMATION AT SCALE

FROM CHAOS TO CONFIDENCE

Data Platform
NEXT STEP

Premium sponsors

VENZO_

INSPARI

valantic companies

itm8®

Eddytor™

dataon

Power your data

Standard sponsors

Backstage

Columbus

TheVIT

DATA - ANALYTICS - AI

twoday

WHO AM I?



Peer Grønnerup

Head of Engineering at Tabular Editor, Denmark

+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

https://github.com/gronnerup

WHERE ARE WE GOING...



From Chaos...



To Confidence...

WHERE ARE WE GOING...



From Chaos…



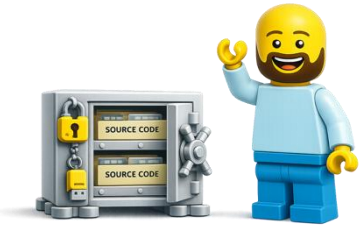
To Confidence…



Architecting for scale



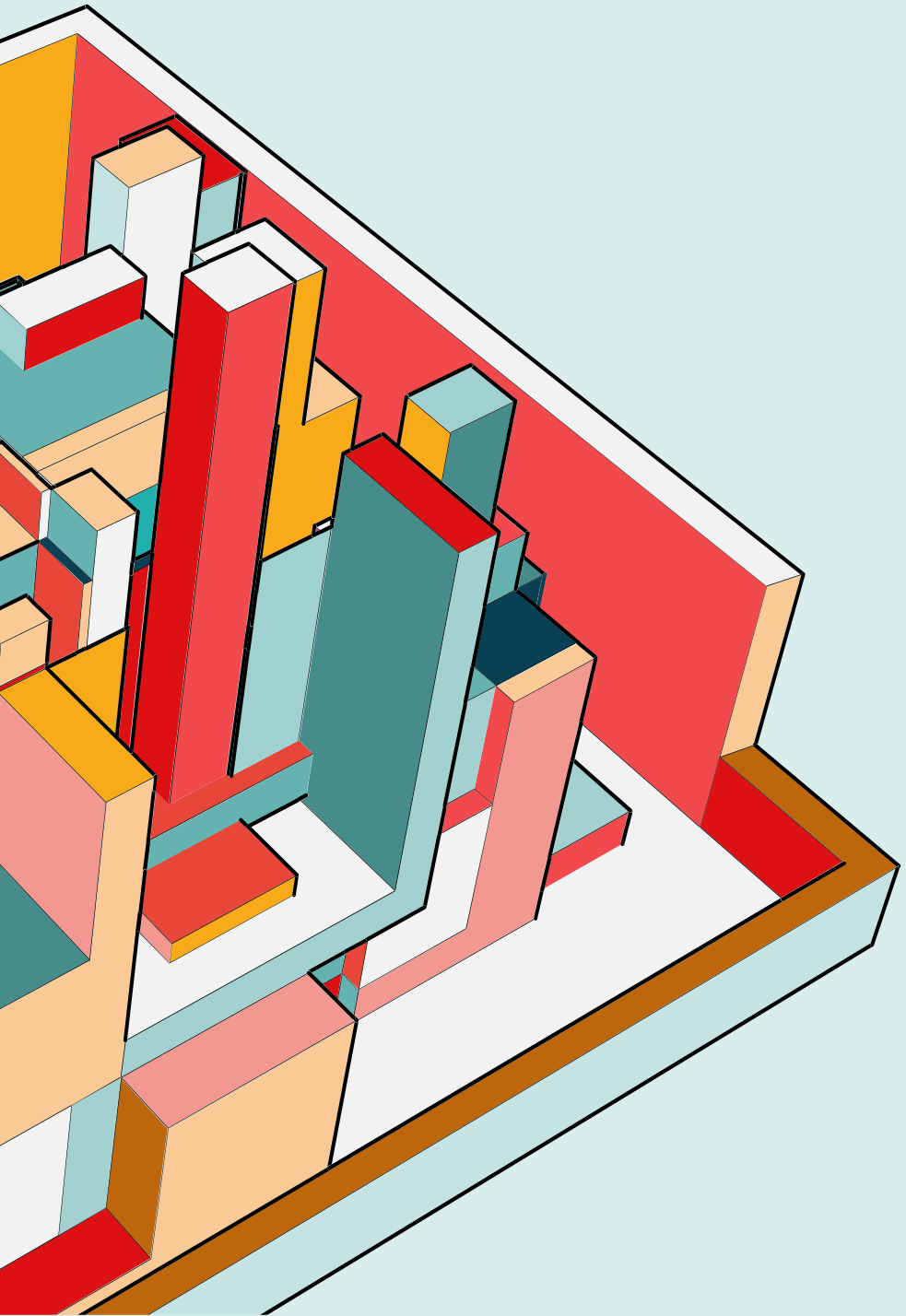
Workspaces, repos & environments



CI/CD & Source Control in Microsoft Fabric



Automating Way of Working & Deployment



FROM CHAOS TO CONFIDENCE

ENTERPRISE CI/CD AT SCALE

THE TYPICAL CHAOTIC FABRIC TENANT



THE ENTERPRISE WAY

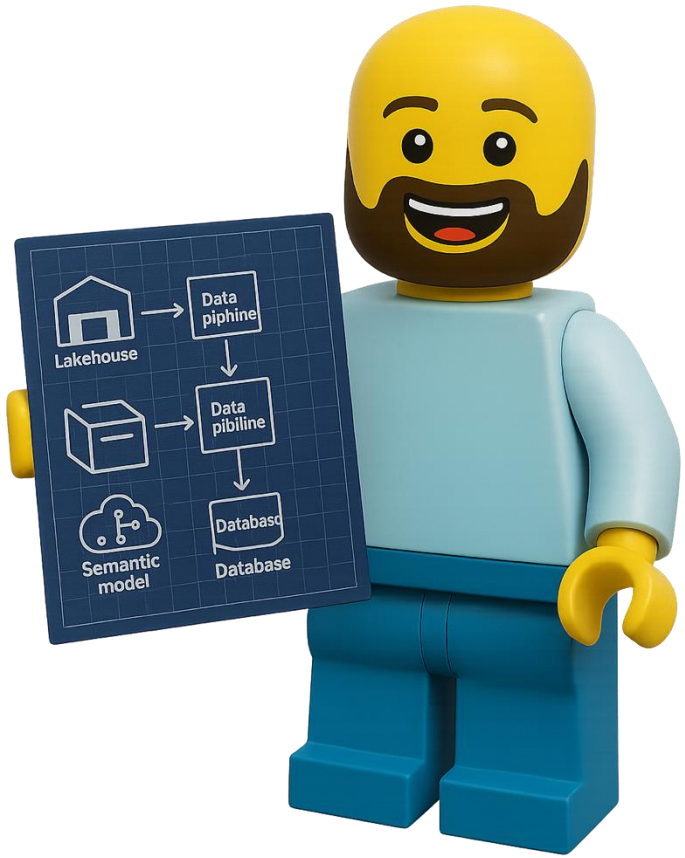
ARCHITECTED FOR SCALE
Medallion layers, clear domains, predictable patterns.

SOURCE CONTROL & REPO STRUCTURE
Git-backed structure that mirrors your architecture.

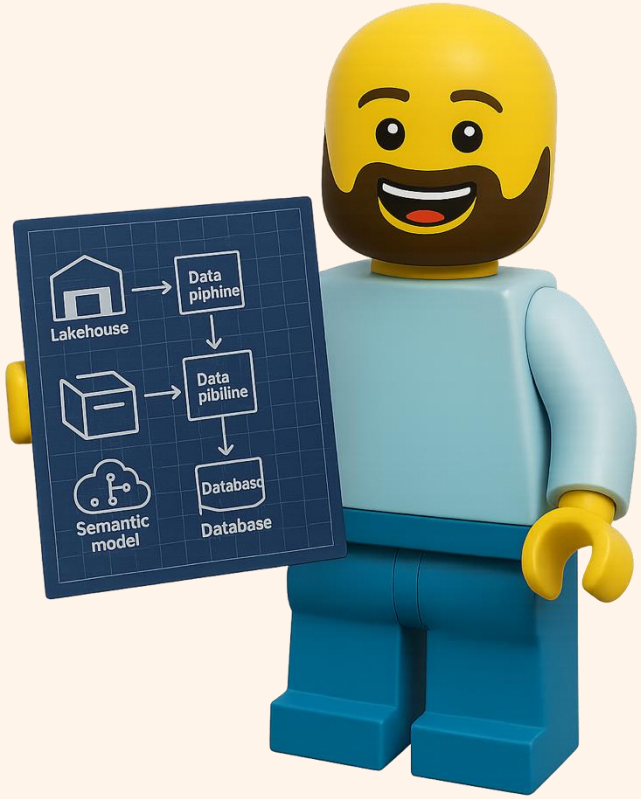
WORKSPACE ORGANIZATION
Consistent naming, scoped permissions, no sprawl.

WAYS OF WORKING · CI
Branching, PRs, peer review, automated feature workspaces.

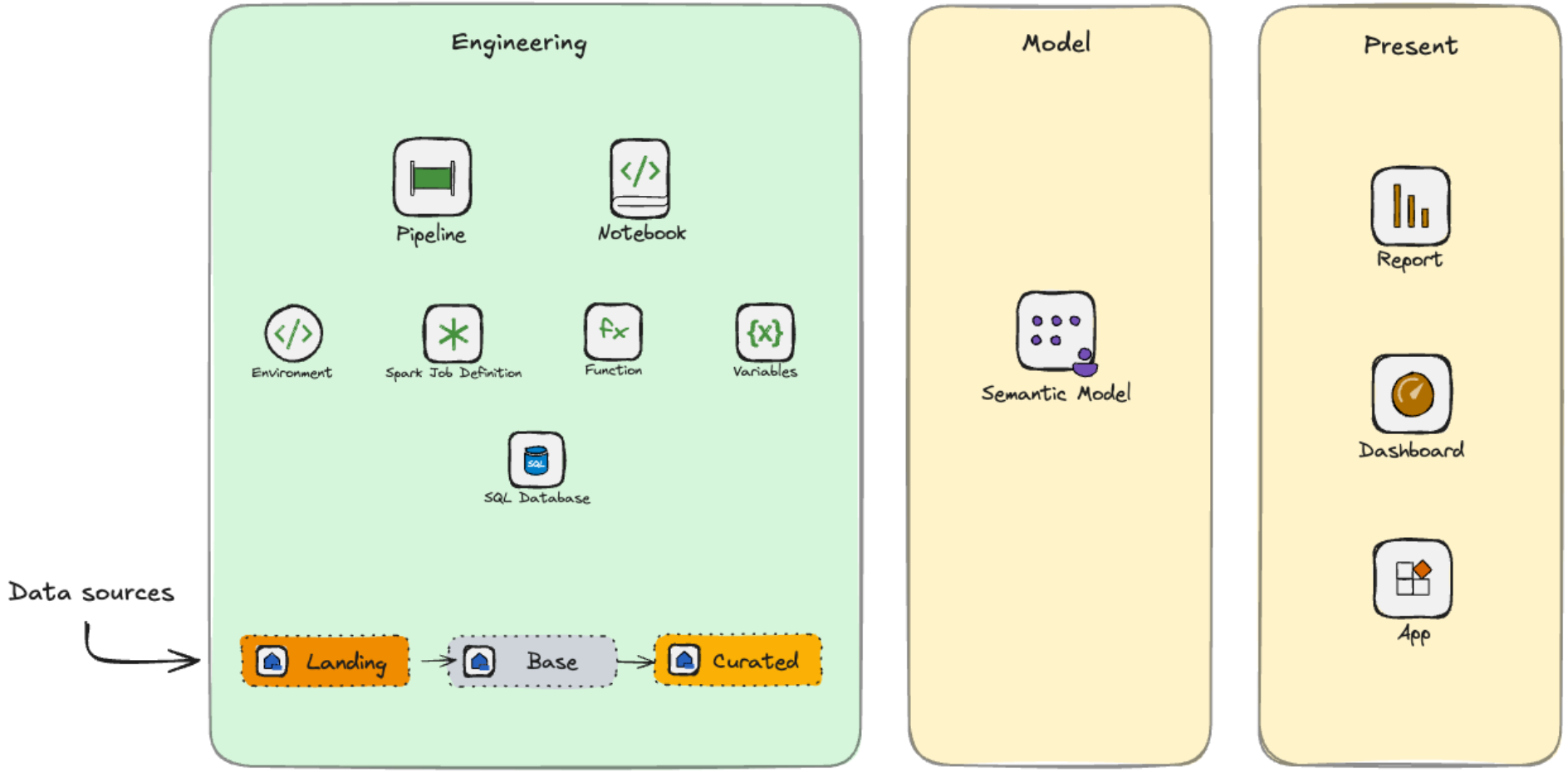
ENTERPRISE DEPLOYMENT · CD
Dev → Test → Prod. Via fabric-cicd. Safe and repeatable with no ClickOps.



ARCHITECTING FOR SCALE



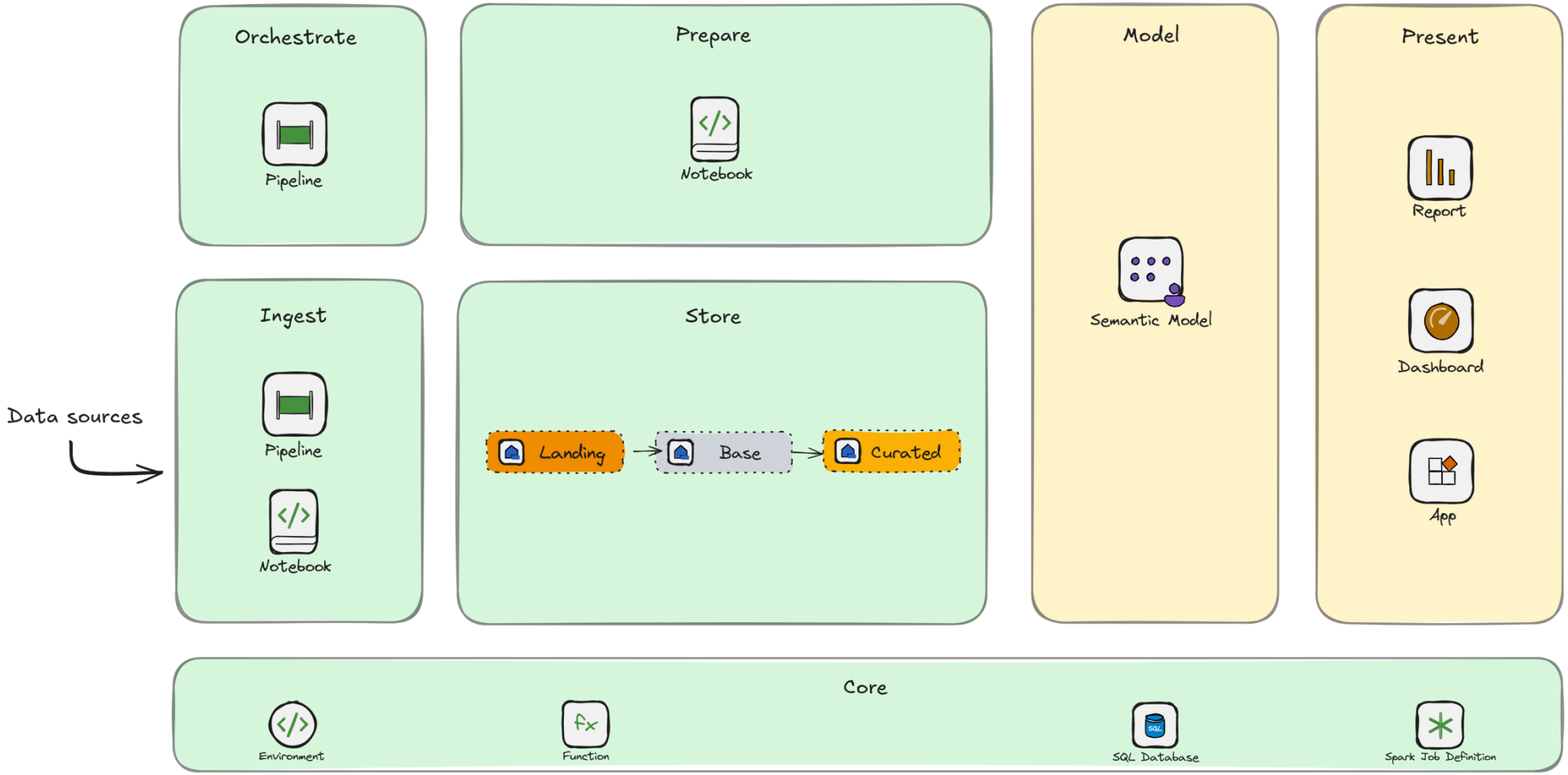
REFERENCE ARCHITECTURE - BASIC



Fabric workloads

Power BI workloads

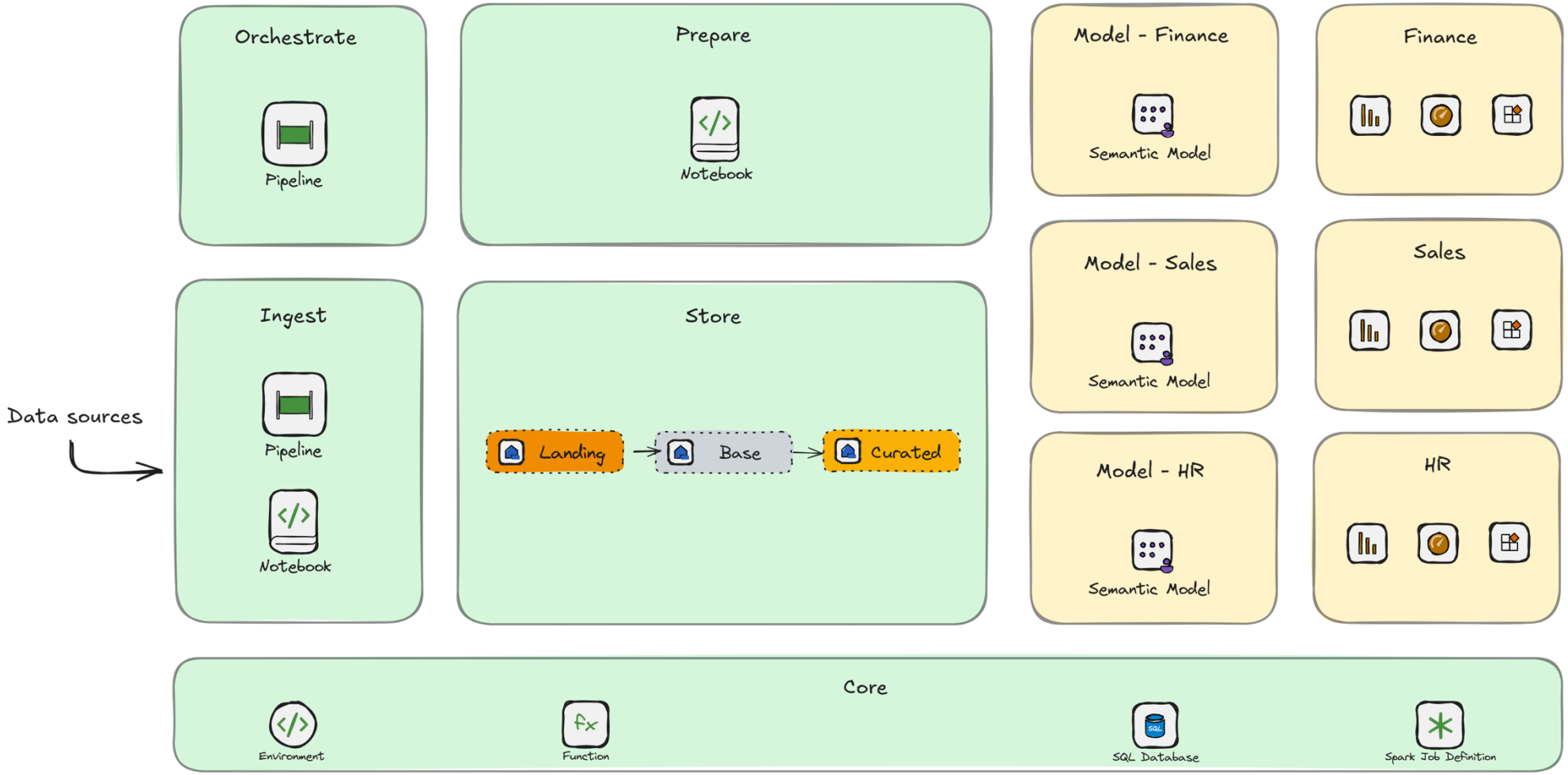
REFERENCE ARCHITECTURE - ENTERPRISE



Fabric workloads

Power BI workloads

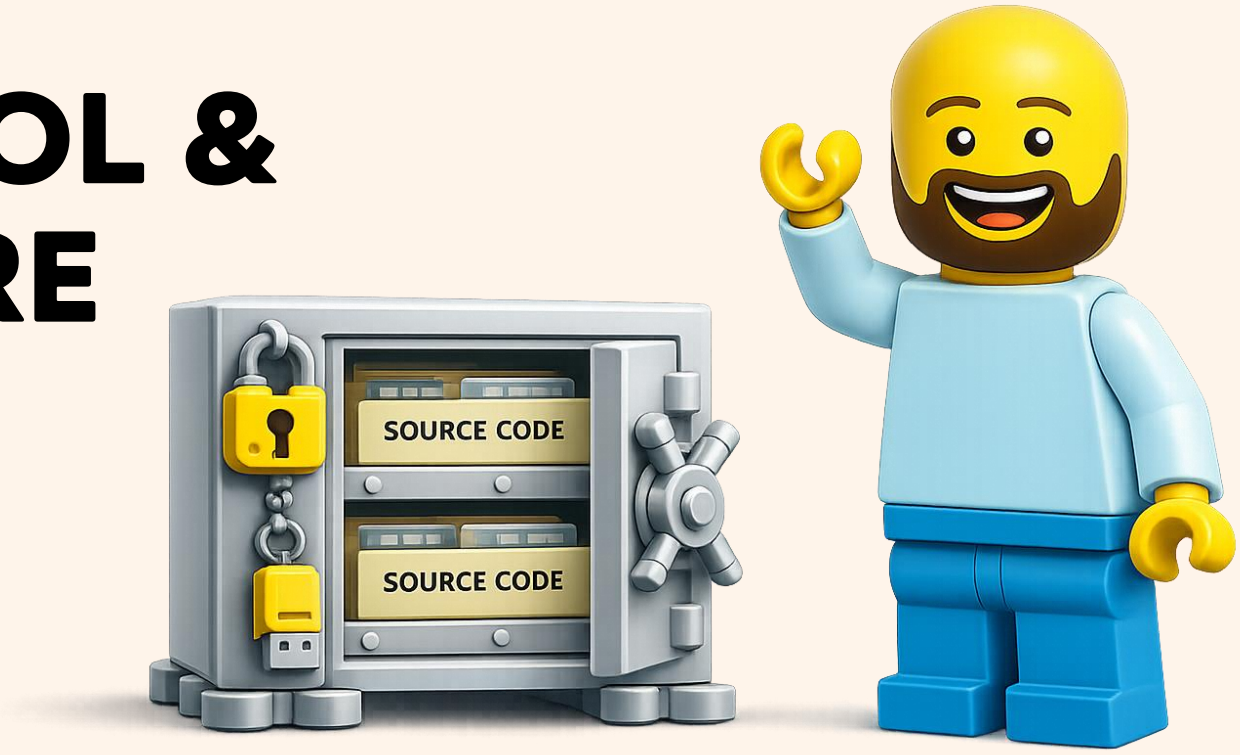
REFERENCE ARCHITECTURE - ENTERPRISE+



Fabric workloads

Power BI workloads

SOURCE CONTROL & REPO STRUCTURE



WHY SOURCE CONTROL IN FABRIC?

Git — The Single Source of Truth

- Full history & rollback. Every change to notebooks, pipelines, semantic models and reports is tracked
- Feature branches isolate work-in-progress safely – experiment without risk to production
- Collaboration without overlap. Multiple developers work in parallel via branches and pull requests
- Two-way sync with Fabric. Commit from the workspace, update the workspace from Git



Without source control, your platform has no memory, no safety net, and no way to collaborate at scale

FABRIC GIT INTEGRATION

How It Works

Git Providers: Azure DevOps | GitHub
Including cross-tenant ADO support and Service Principal for GitHub

Configured at Workspace Level
Connect workspace → repo → branch → folder

- Commit changes from workspace to Git
- Update workspace from Git (sync)
- Branch out to isolated feature workspaces
- Workspace folder structure preserved in repo
- PR-based workflow with code reviews

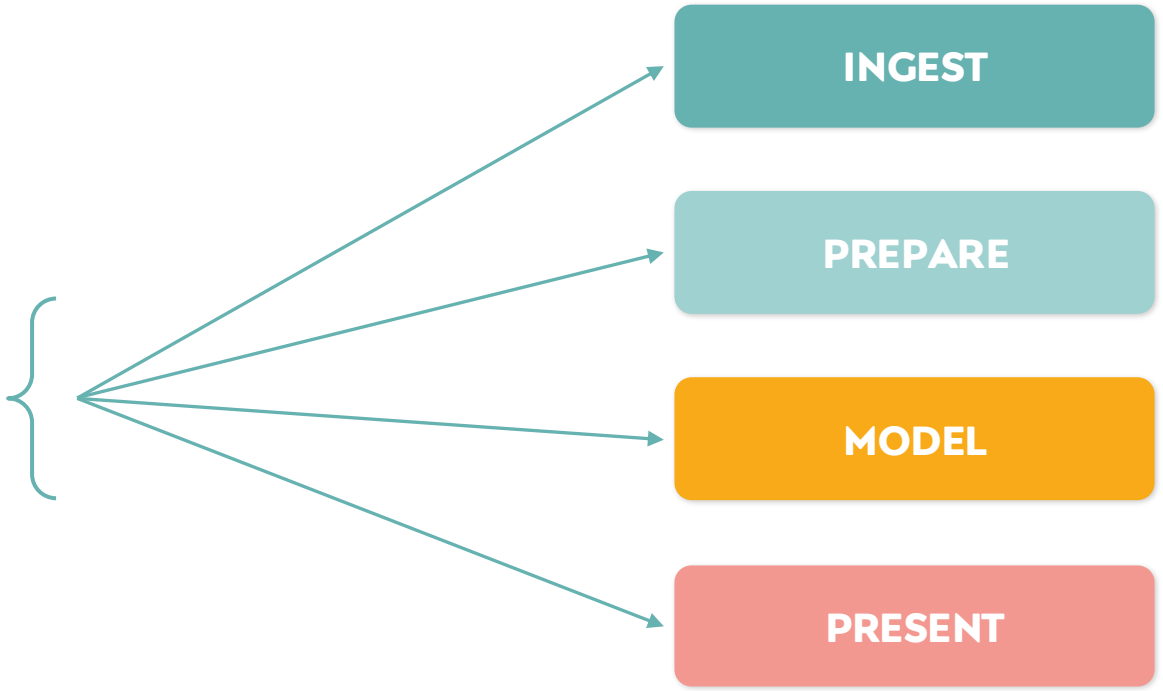
New Capabilities FabCon 2026

- Selective Branching – Branch out with only the items you need
- **Compare Code Changes** – Rich diff view – see exactly what changed before sync
- **Branched Workspaces** – Visual parent-child relationship between workspaces
- **Variable Library** – Manage configs across Dev/Test/Prod – no hardcoding
- **Fabric MCP + VS Code** – GitHub Copilot talks to Fabric natively (GA)
- **Extensibility Toolkit** – Full CI/CD for custom workloads (GA)

Read more: https://learn.microsoft.com/en-us/fabric/cicd/git-integration/intro-to-git-integration

A REPO THAT MIRRORS YOUR ARCHITECTURE

```
fabric-platform/
├── .azure-pipelines/
├── .github/
├── automation/
├── documentation/
└── solution/
    ├── ingest/
    ├── prepare/
    ├── model/
    └── present/
```

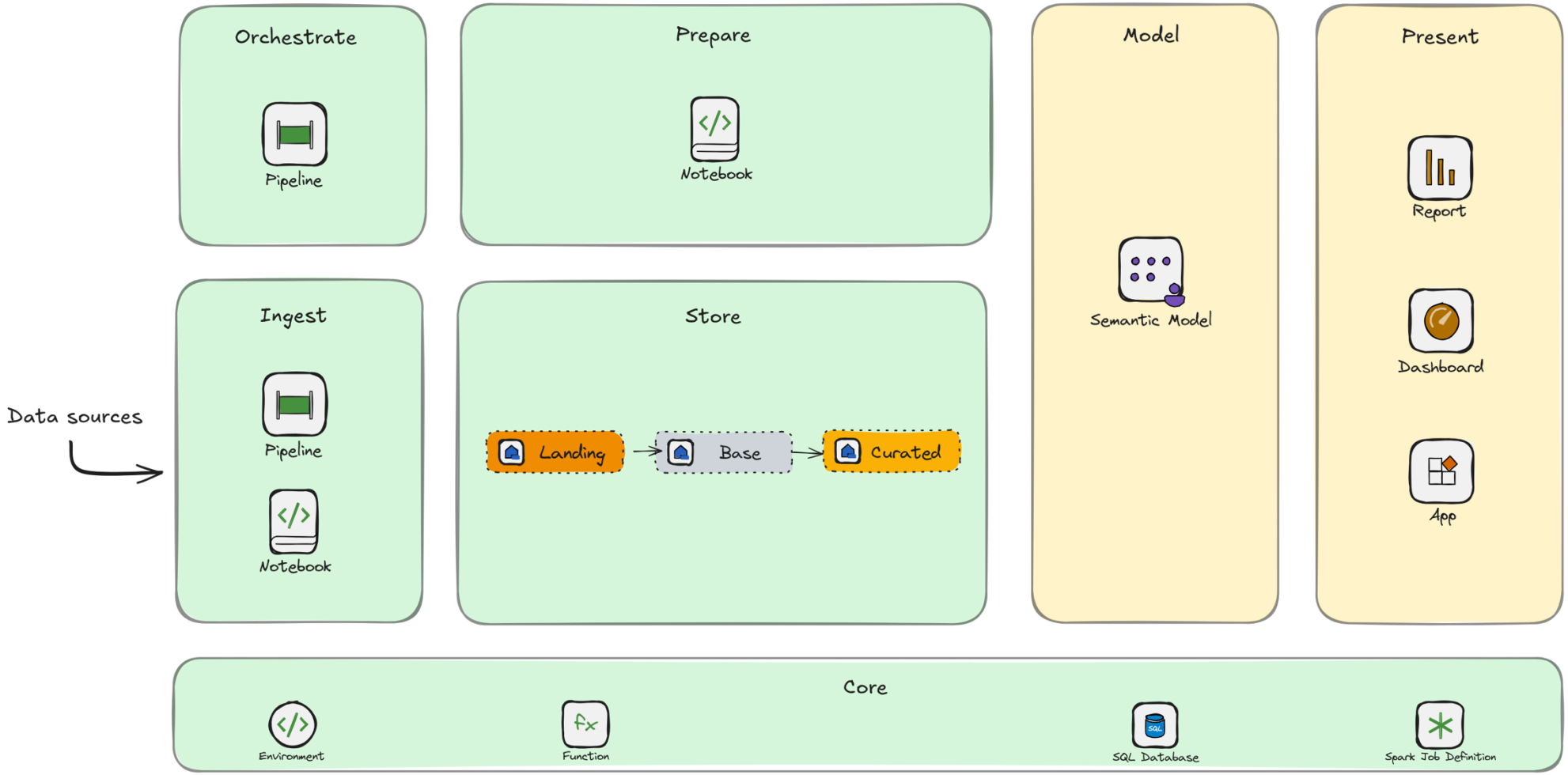


Your repo structure is your architecture - made versionable.

WORKSPACE ORGANIZATION



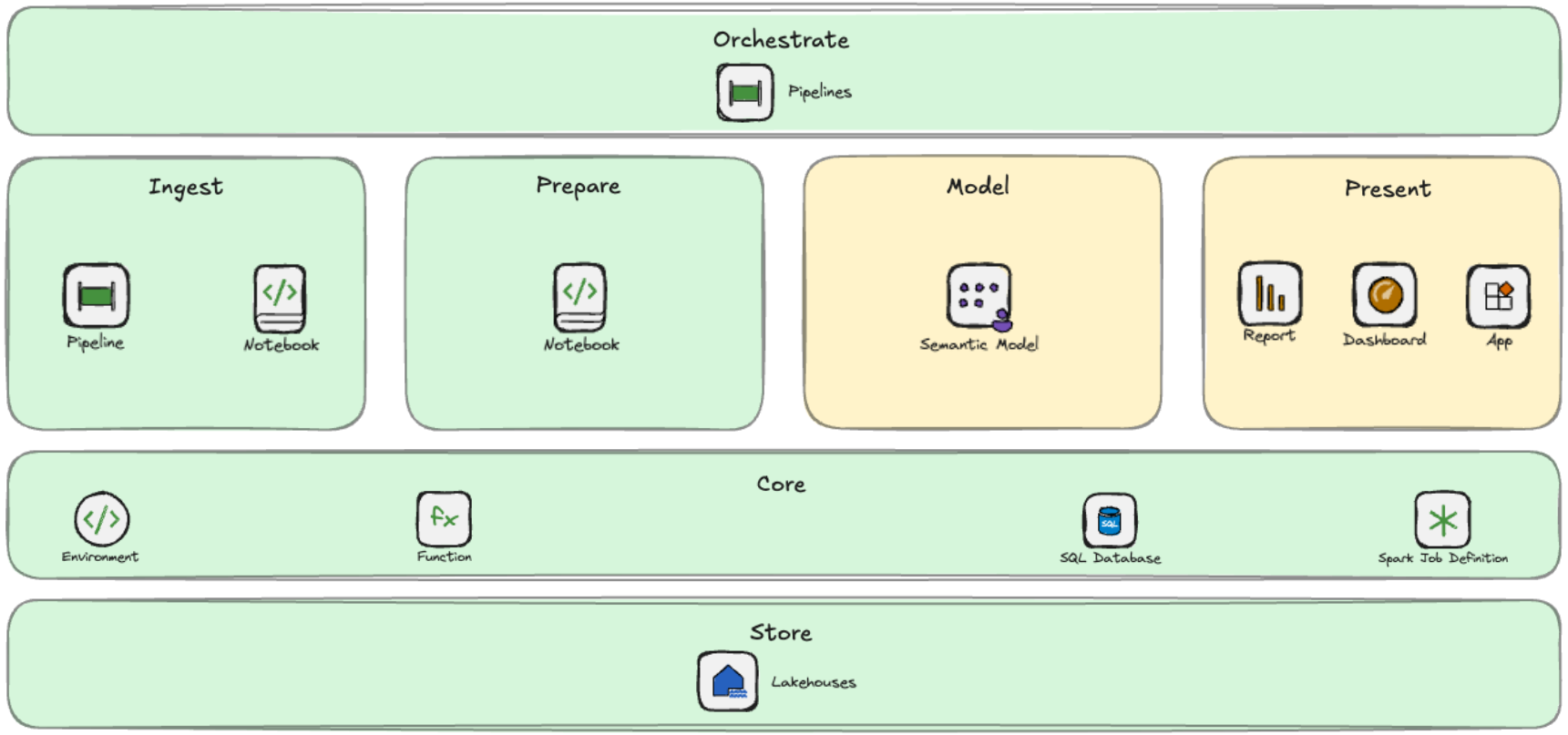
REFERENCE ARCHITECTURE - ENTERPRISE



Fabric workloads

Power BI workloads

FABRIC WORKSPACE STRUCTURE



x3

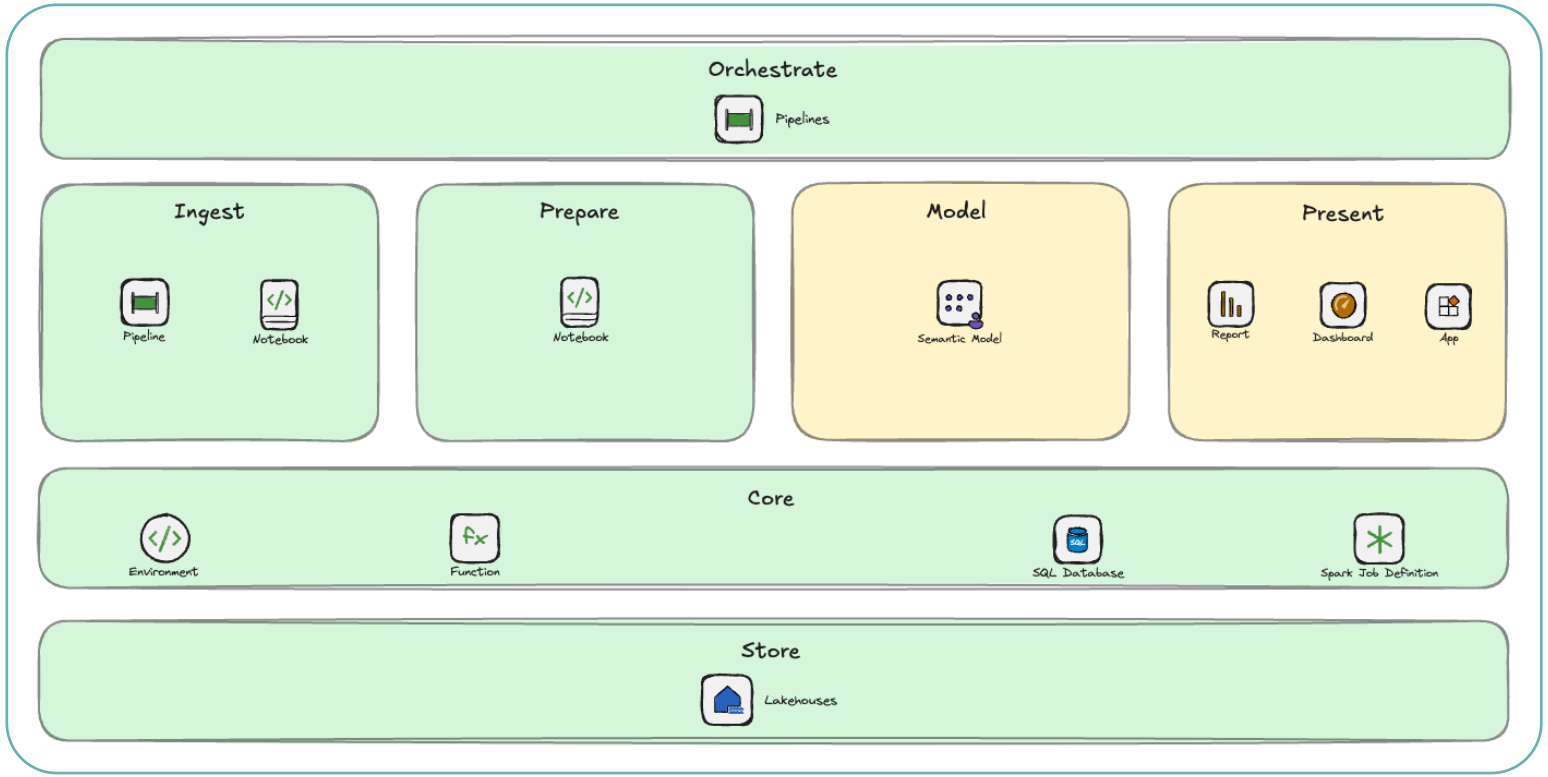
[dev, tst, prd]

Fabric workloads

Power BI workloads

WORKSPACE DESIGN CONSIDERATIONS

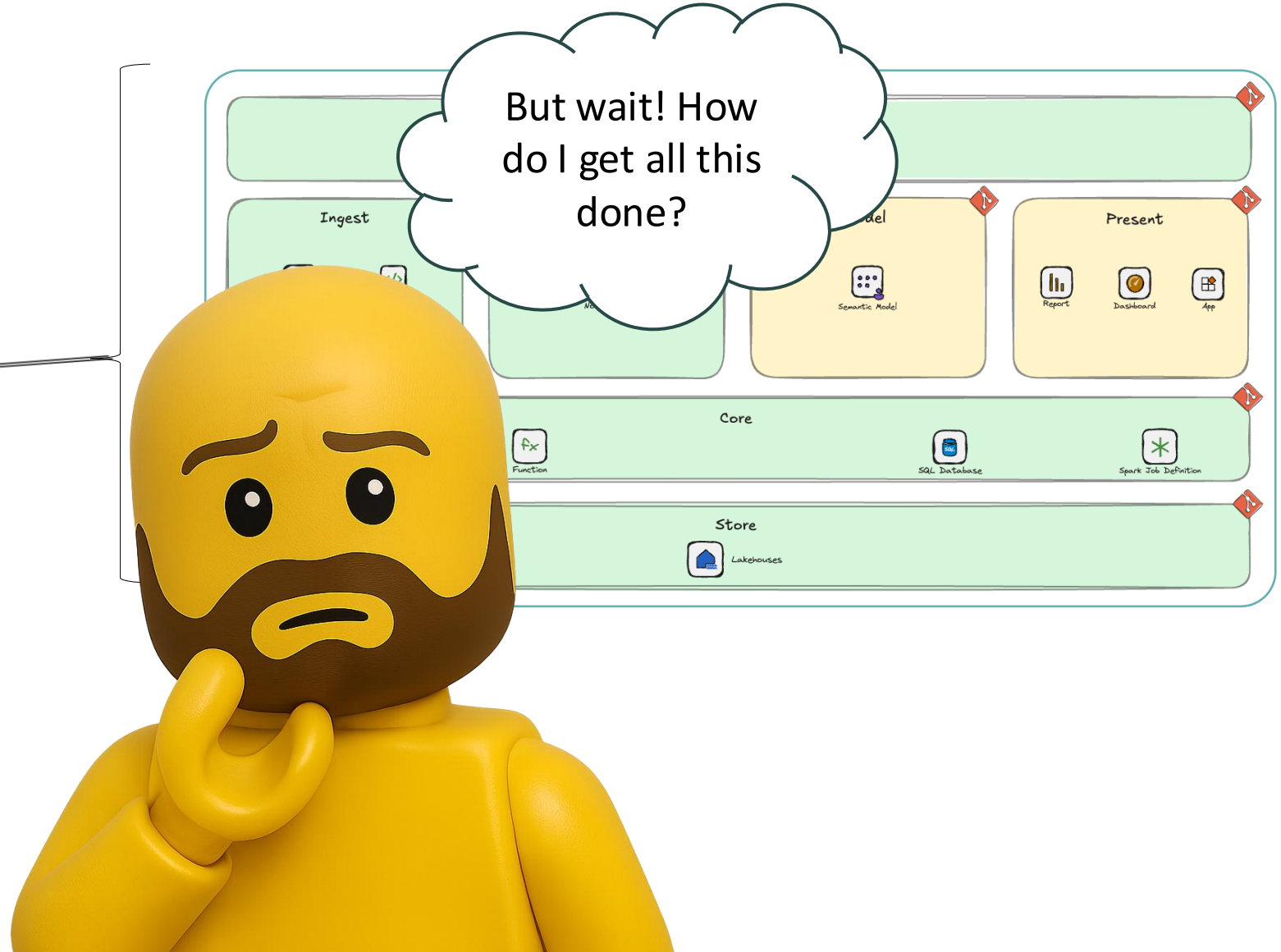
- Security and Access
- Separation of duties
- Network & connectivity
- Capacity isolation
- Governance & compliance
- Testing & Deployment
- Item organization
- Git integration and CI/CD



MY GO-TO FOR ENTERPRISE GIT SETUP

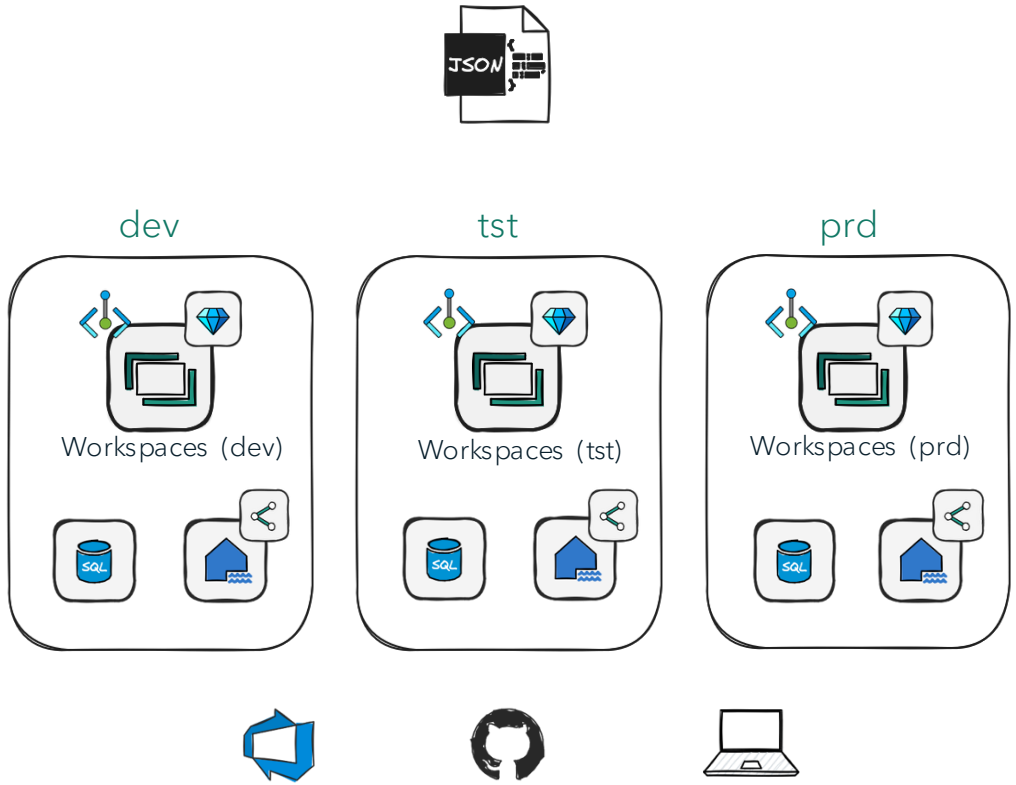
Git Repo structure

.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

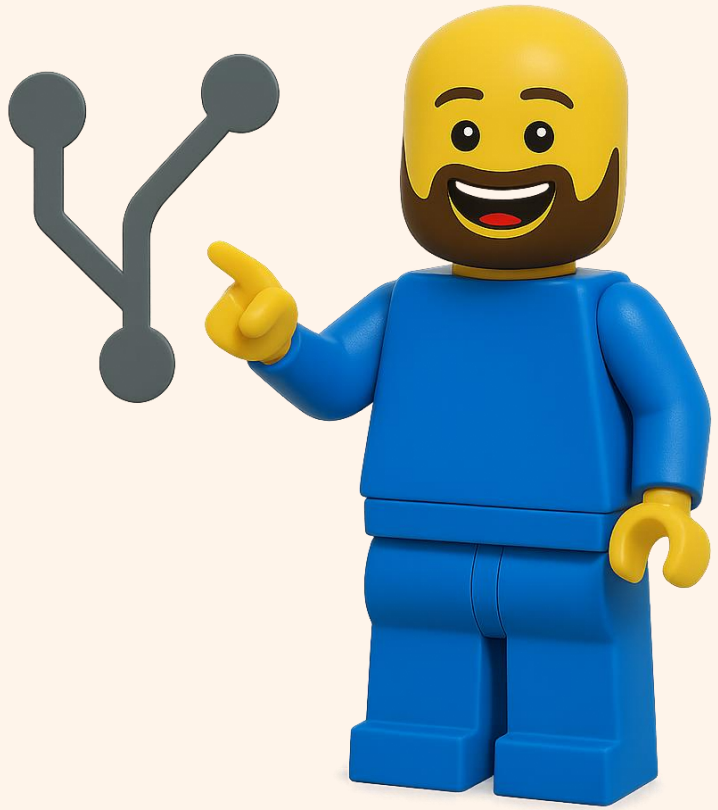


SHOW TIME!

WAYS OF WORKING

CONTINUOUS INTEGRATION

BRANCH. REVIEW. VALIDATE. REPEAT.



WHAT IS CI, AND WHY NOW?

CI = every change is built, validated and merged automatically and many times a day.

Branch

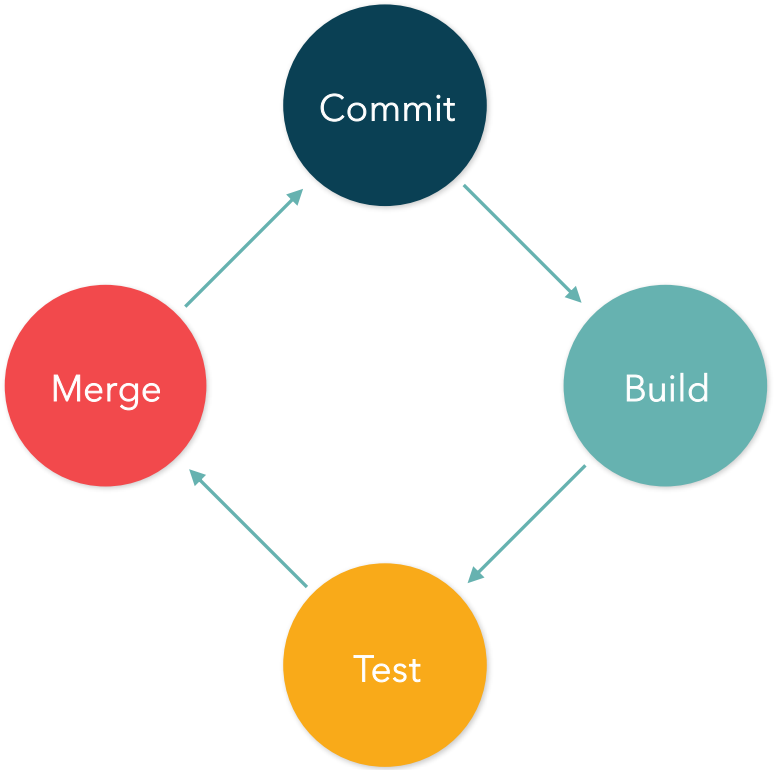
Work in isolation. No one's toes get stepped on.

Validate

Every Pull Request triggers checks before it merges.

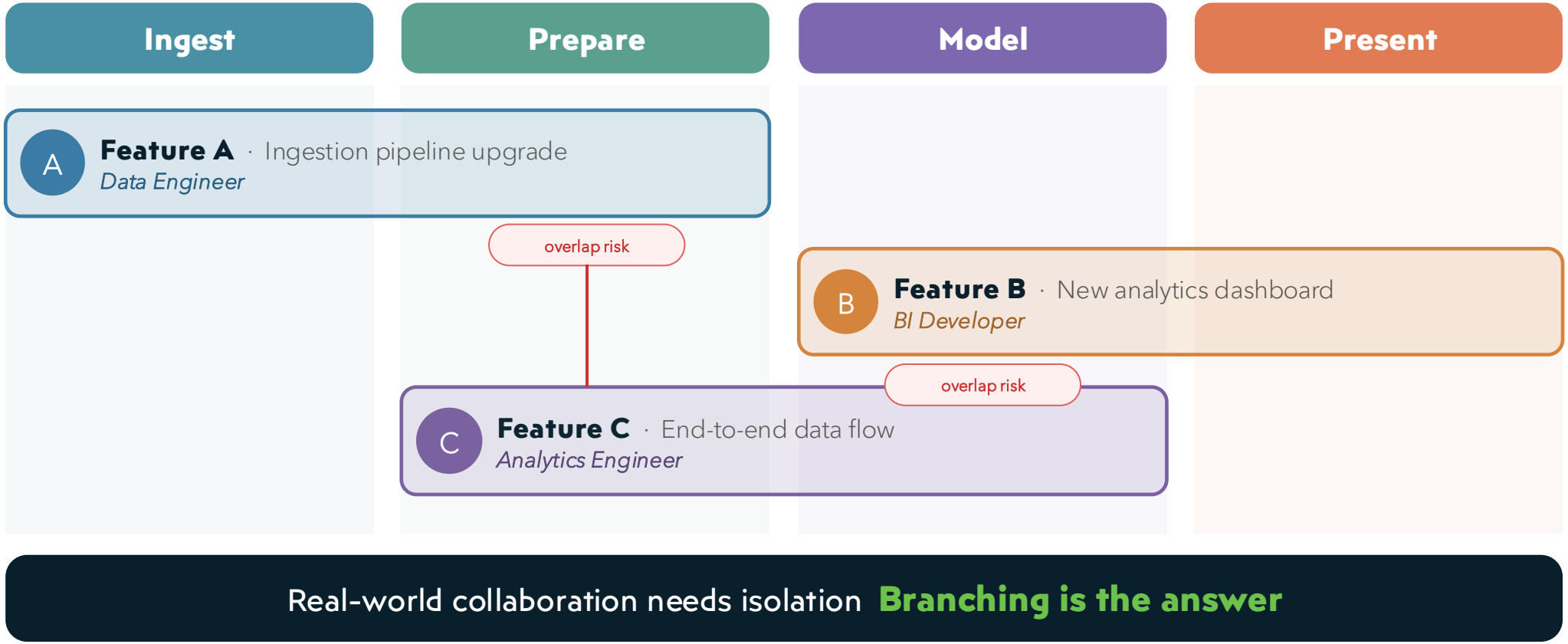
Integrate

Small, frequent merges to main. Low risk, high cadence.



Small changes, validated often, merged with confidence.

HOW TO SUPPORT MULTIPLE DEVELOPERS?

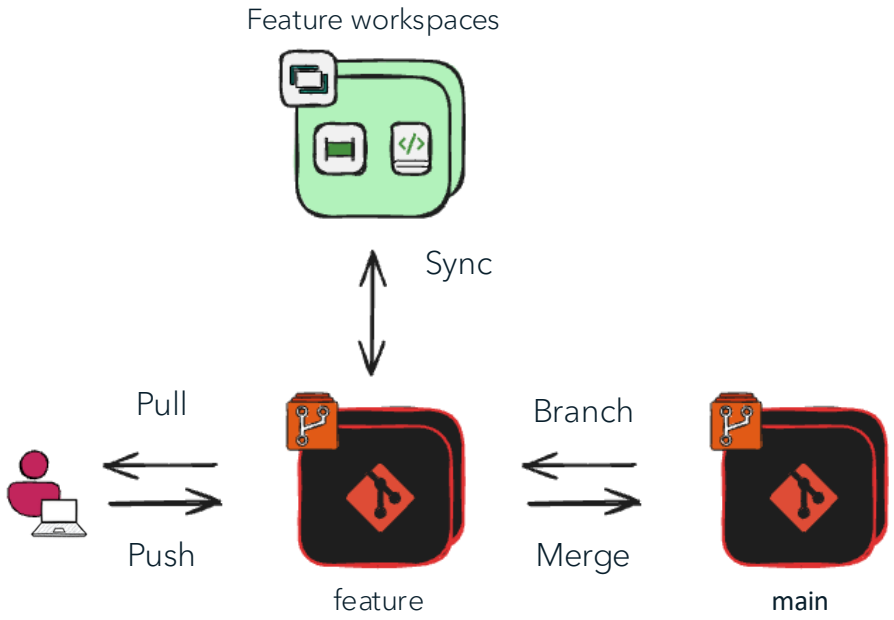


Fabric Git Integration: Workspace linked to specific repository, branch and Git folder

GENERAL DEVELOPMENT WORKFLOW

Development flow - Supported by automation!

1. Create new feature branch from main
2. Create isolated development workspaces
3. Implement changes
4. Commit changes back to feature branch
5. Create PR and merge changes to main
6. Delete feature branch + isolate development workspaces



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

WHY AUTOMATE THIS FLOW?

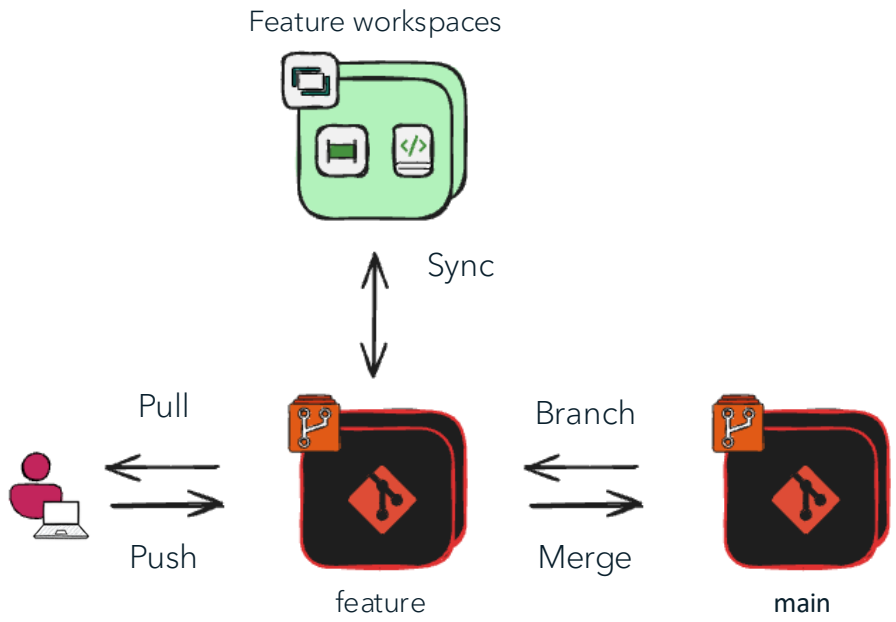
ClickOps

- ✗ 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!

ENTERPRISE DEPLOYMENT

CONTINUOUS DELIVERY

SAFE. REPEATABLE. NO CLICKOPS.



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
- Does 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 – THE TOOLBOX

ORCHESTRATION · Where pipelines run

Azure DevOps Pipelines

YAML pipelines, approval gates, environments, service connections.

GitHub Actions

Workflows in your repo. Reusable actions, OIDC to Azure / Fabric.

AUTOMATION CODE · What those pipelines invoke

Python + REST

Custom scripts that hit Fabric REST APIs directly. Maximum control, more code to own.

Fabric CLI

Microsoft's CLI for Fabric. Scriptable workspace, item, and capacity ops.

Tabular Editor 2 CLI

Free, OSS. Deploy, validate and script semantic models from any pipeline.

fabric-cicd

Python library purpose-built for Fabric workspace CI/CD

Now part of the Fabric CLI.

Officially Microsoft-supported

Pick your orchestrator + your code. fabric-cicd makes the Fabric part the easy part

FABRIC-CICD — THE FABRIC-NATIVE WAY

Code-first CI/CD for source-controlled Fabric workspaces — now part of the Fabric CLI.

Developer-friendly

- Code-first, Pythonic API
- No raw REST calls
- Runs anywhere Python runs

Environment-aware

- Parameterise per environment
- Reuse across dev/test/prod
- Locale & DevOps-friendly

Smart deployment

- Deploy all items in a workspace
- Auto-unpublish orphans
- Idempotent + repeatable

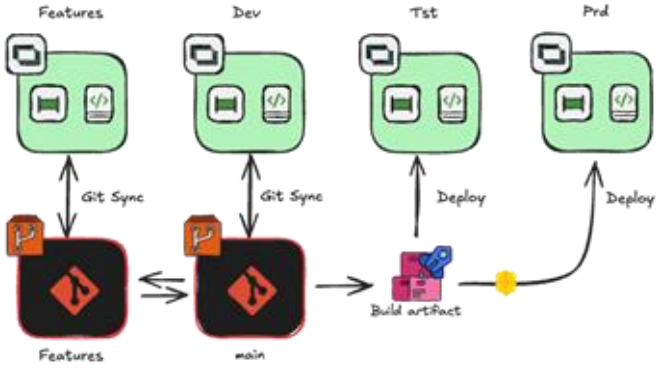
✓ **Now officially Microsoft-supported** ✓ **Integrated into the Fabric CLI**

→ source-controlled workspace ➜ deployment framework, in a few lines of Python

3 RELEASE PATTERNS

Option 1

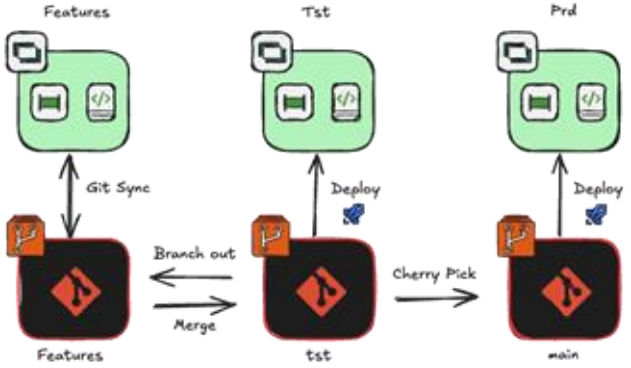
Dev = Main



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

Option 2

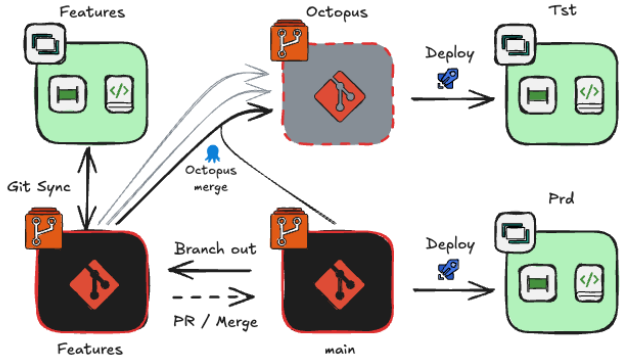
PR to test – Cherry pick to main



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

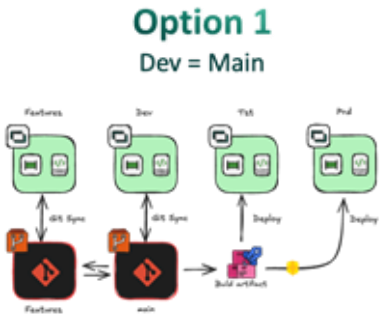
Option 3

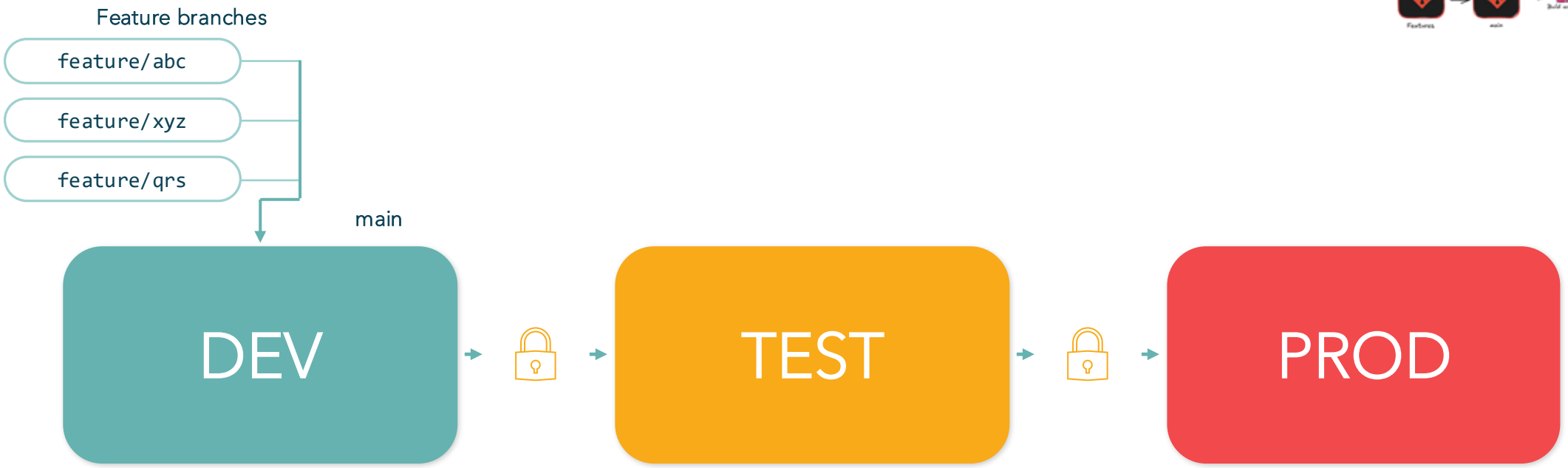
Octopus deploy



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

PROMOTION WITH GATES





Every step gated. No portal clicks.



DEMO TIME!

TIPS, TRICKS & SUMMING UP



TIPS, TRICKS & SUMMING UP!

- Start using Git integration in Fabric if you are not doing it already
- 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
- 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.

TIME FOR QUESTIONS!

BEFORE CONNECTION IS TERMINATED BY PEER...

