Thank you sponsors!

redgate

twintos

wortell

macaw

Your guide in the Era of AI.

Rubicon

Consulting & Technology

ilionx

Data Saturdays

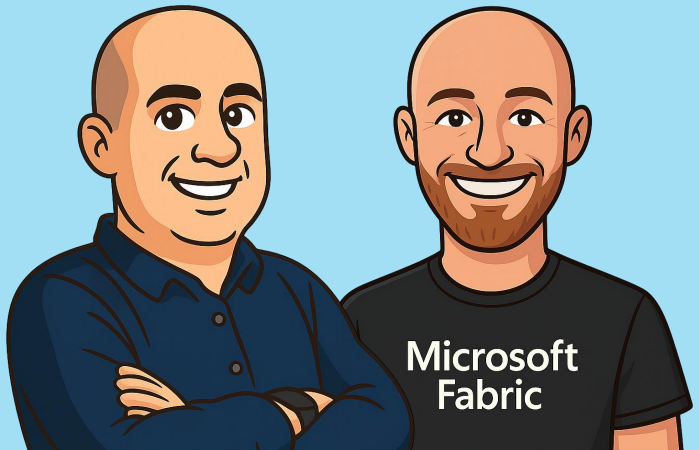
Rate Data Saturday Holland

Review to win

With custom Data Saturdays lego stickers!

FABRIC AUTOMATION ALL THE WAY

A DEEP DIVE INTO AUTOMATING YOUR FABRIC DATA PLATFORM



DATA SATURDAYS

WHO ARE WE?

Peer Grønnerup

Technical Architect & Community Advocate 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 Automation and CI/CD!

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

https://peerinsights.hashnode.dev

https://github.com/gronnerup

WHO ARE WE?

Erwin de Kreuk

Principal Consultant

Lead Data & AI at InSpark

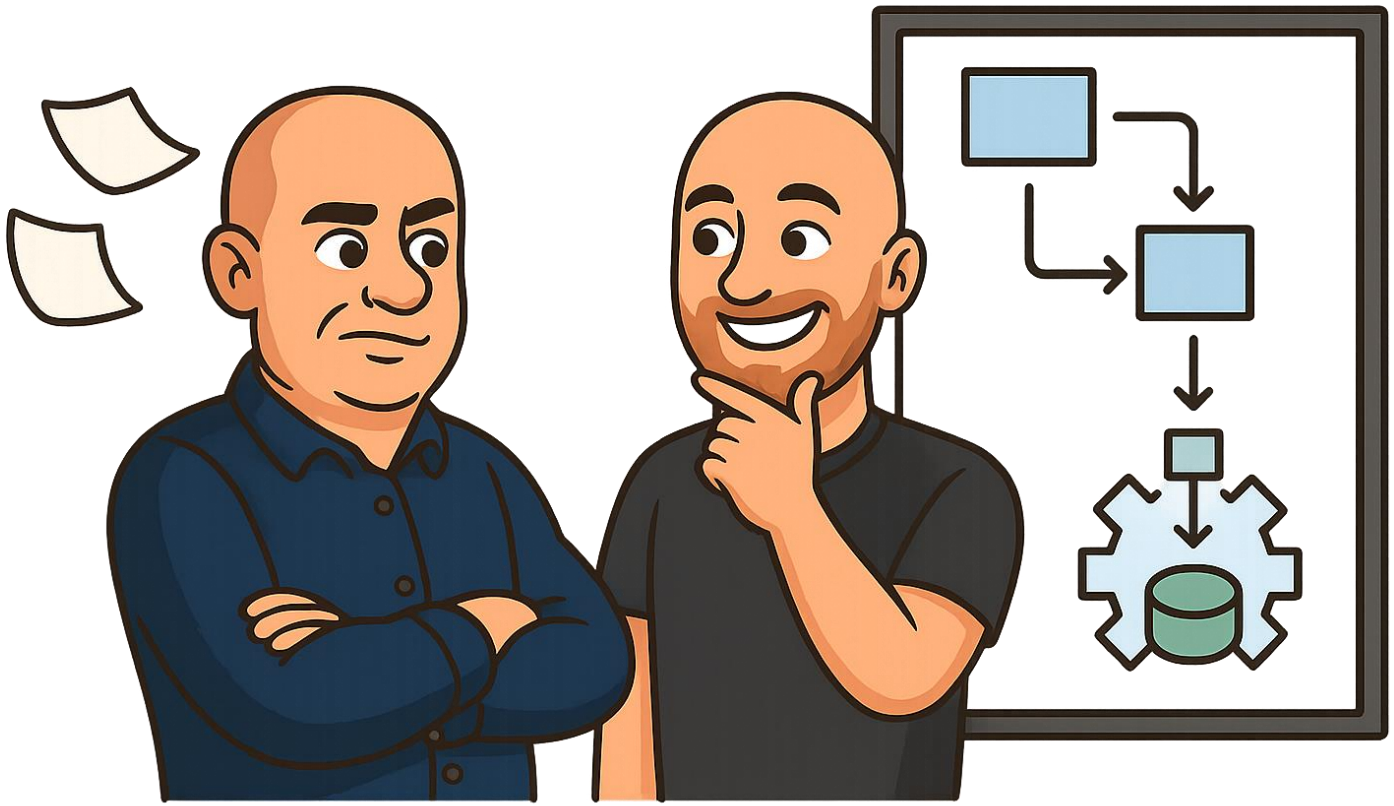
https://www.linkedin.com/in/erwindekreuk/

https://erwindekreuk.com

https://github.com/edkreuk

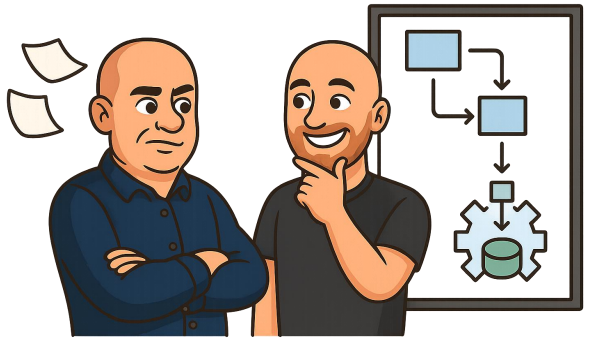
Https://Dutchfabricusergroup.com

SO, WHERE ARE WE GOING TODAY...?

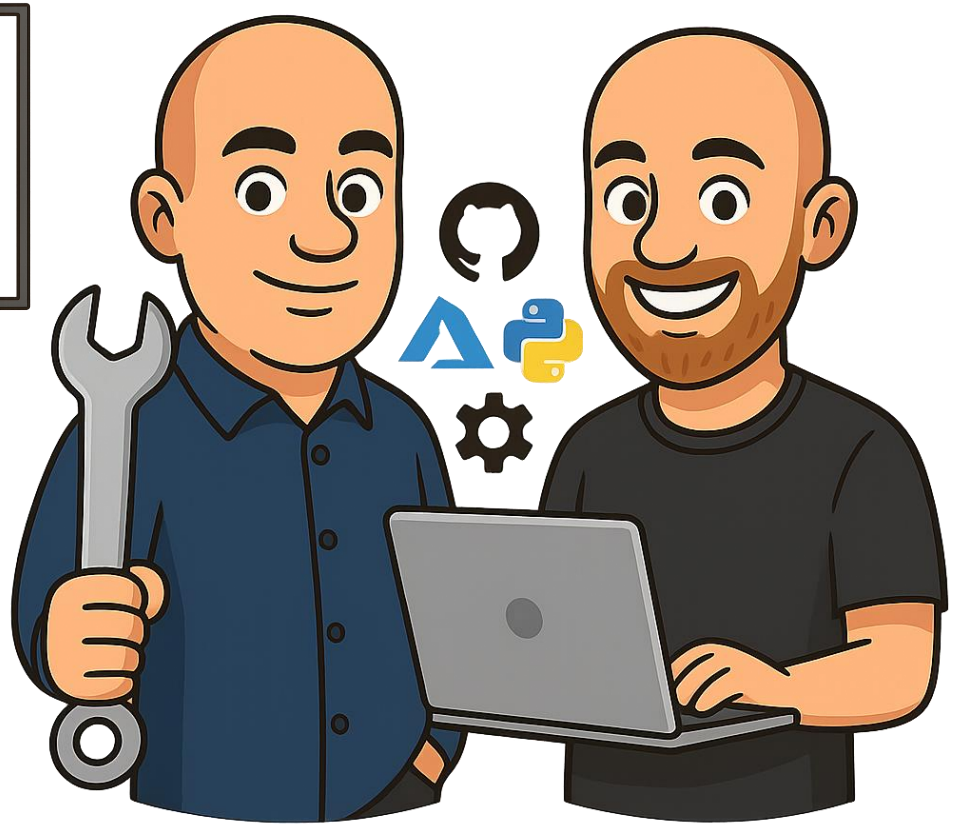


Why automation matters?

SO, WHERE ARE WE GOING TODAY...?

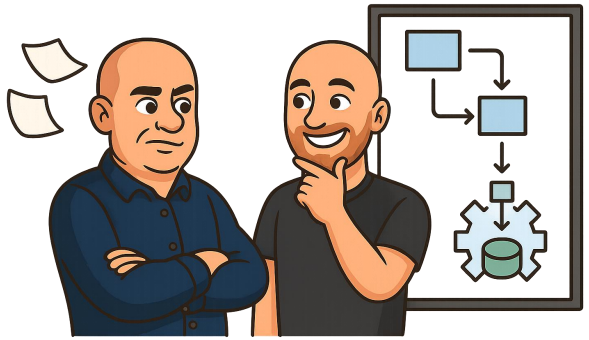


Why automation matters?



Tooling

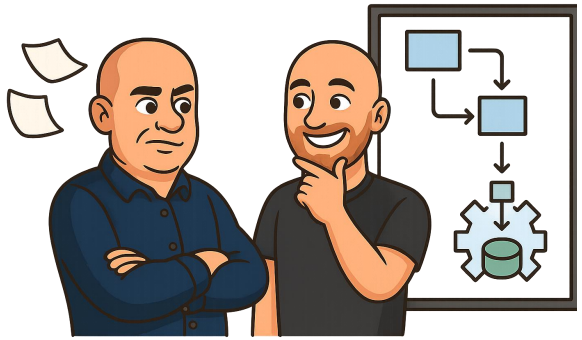
SO, WHERE ARE WE GOING TODAY...?



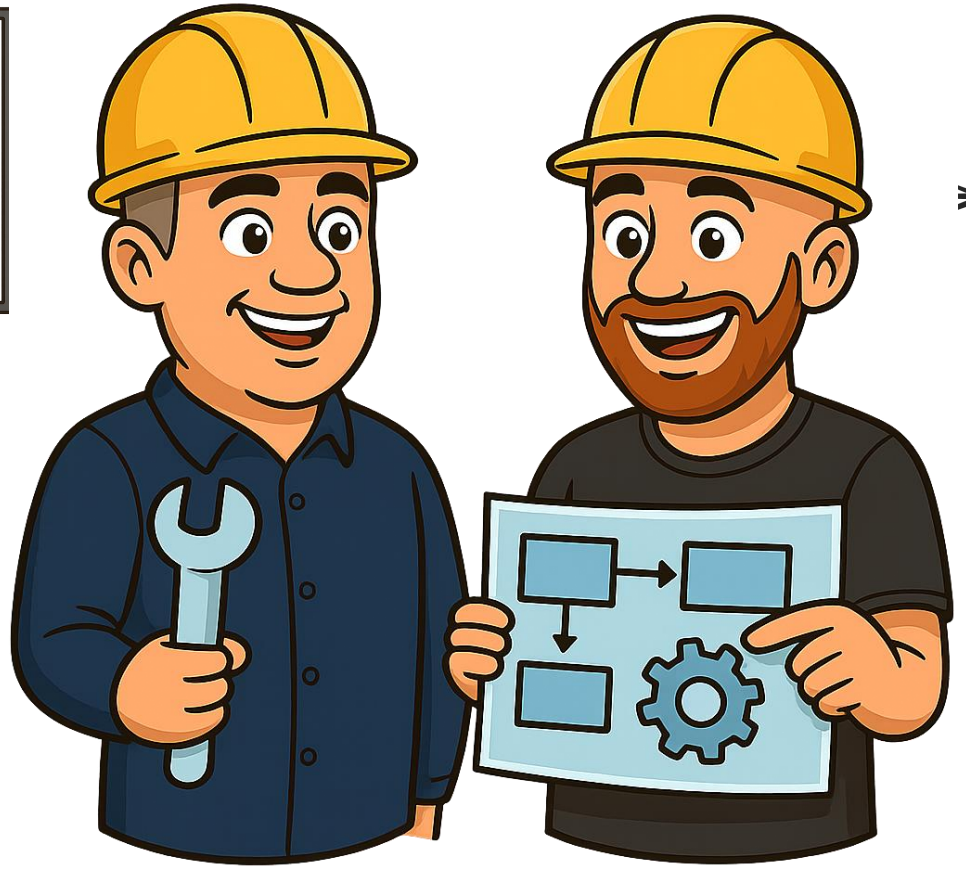
Why automation matters?

Getting started as a non hero

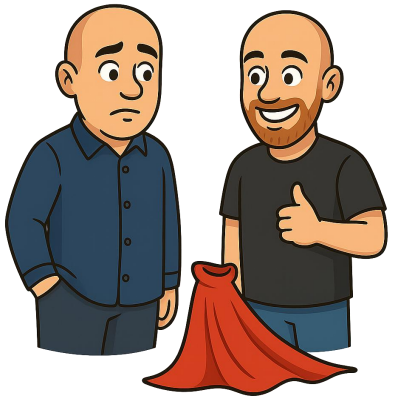
SO, WHERE ARE WE GOING TODAY...?



Why automation matters?

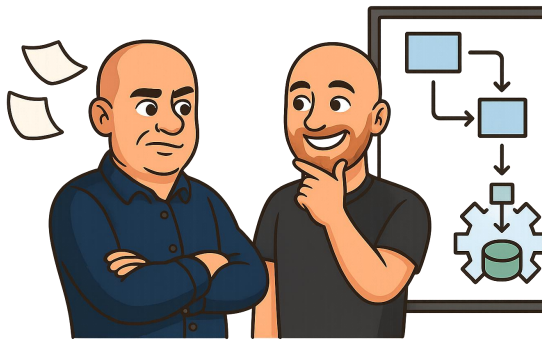


Architecting for automation



Getting started as a non hero

SO, WHERE ARE WE GOING TODAY...?



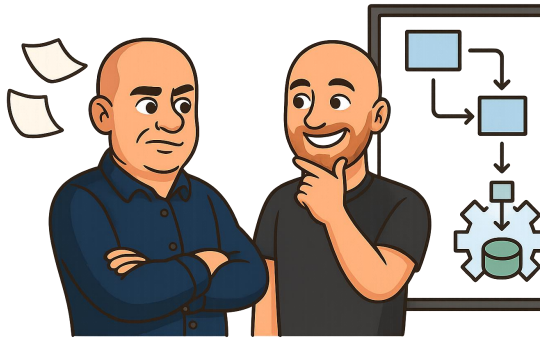
Why automation matters?

Fabric IaC

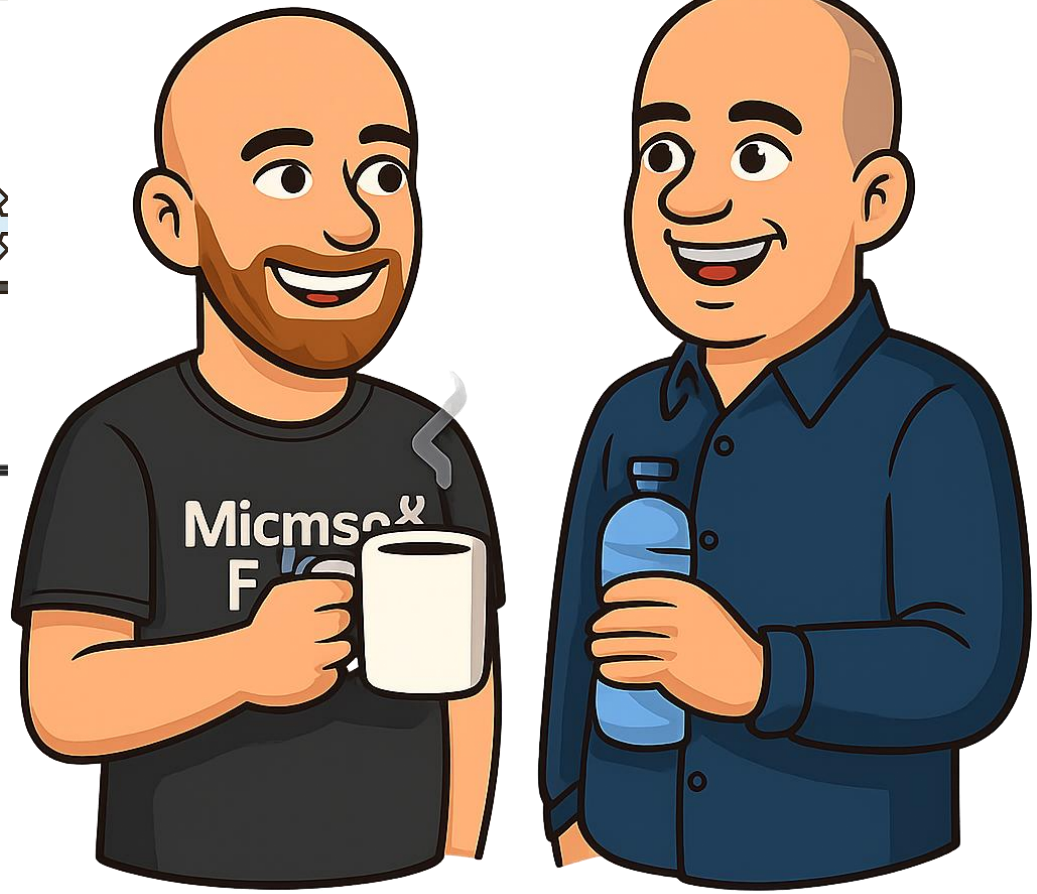
Getting started as a non hero

Architecting for automation

SO, WHERE ARE WE GOING TODAY...?



Why automation matters?

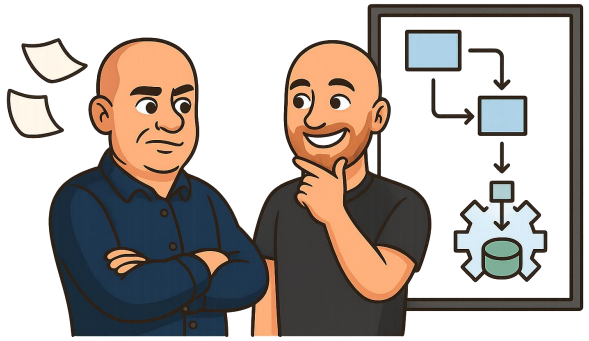


Getting started as a non hero

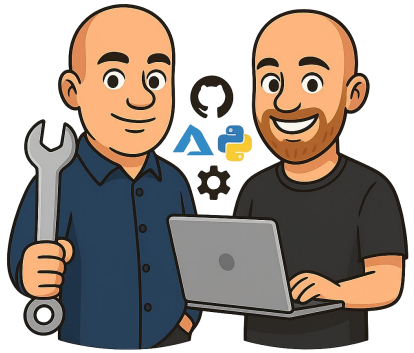
Architecting for aut

BREAK

SO, WHERE ARE WE GOING TODAY...?



Why automation matters?

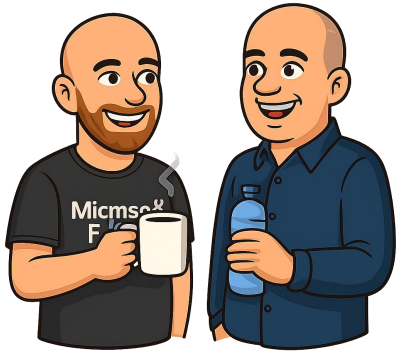


Tooling

Getting started as a non hero

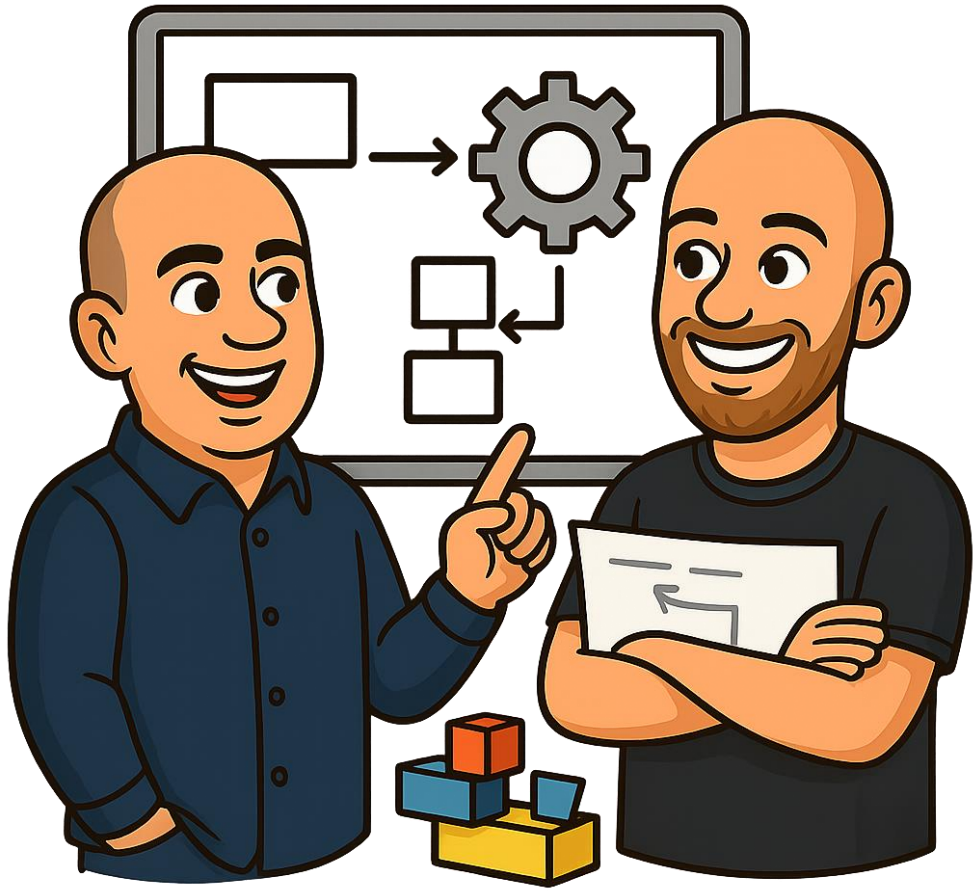
Architecting for automation

Fabric IaC



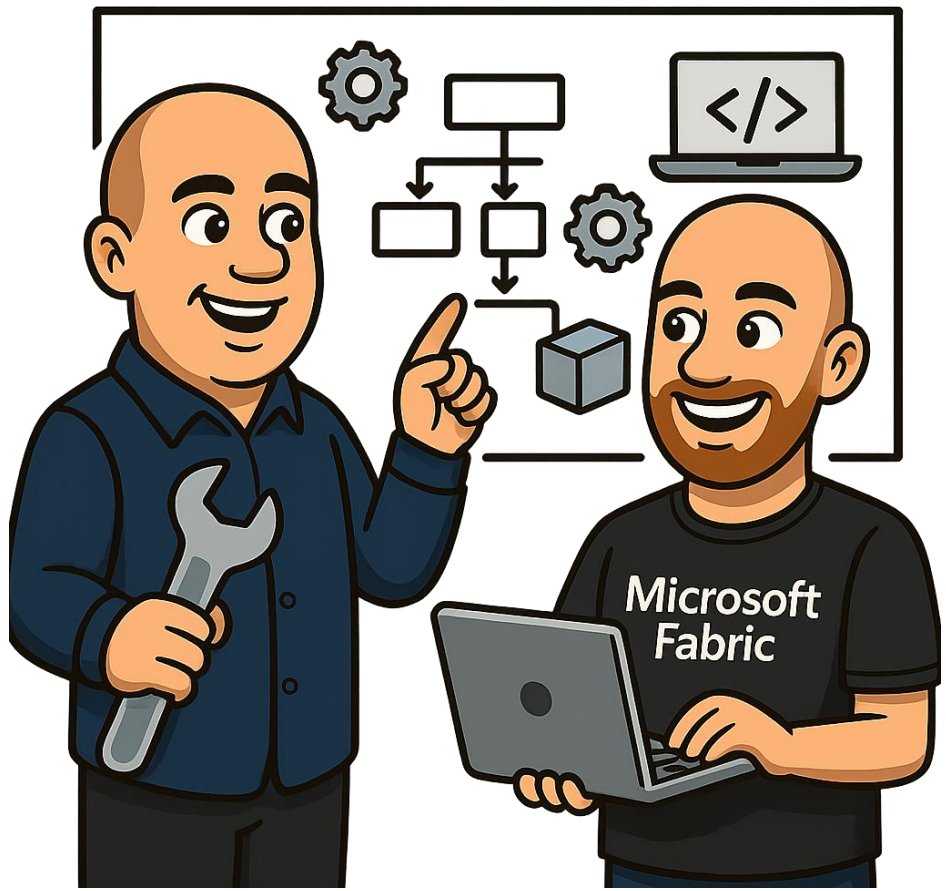
BREAK

...AND AFTER THE BREAK?



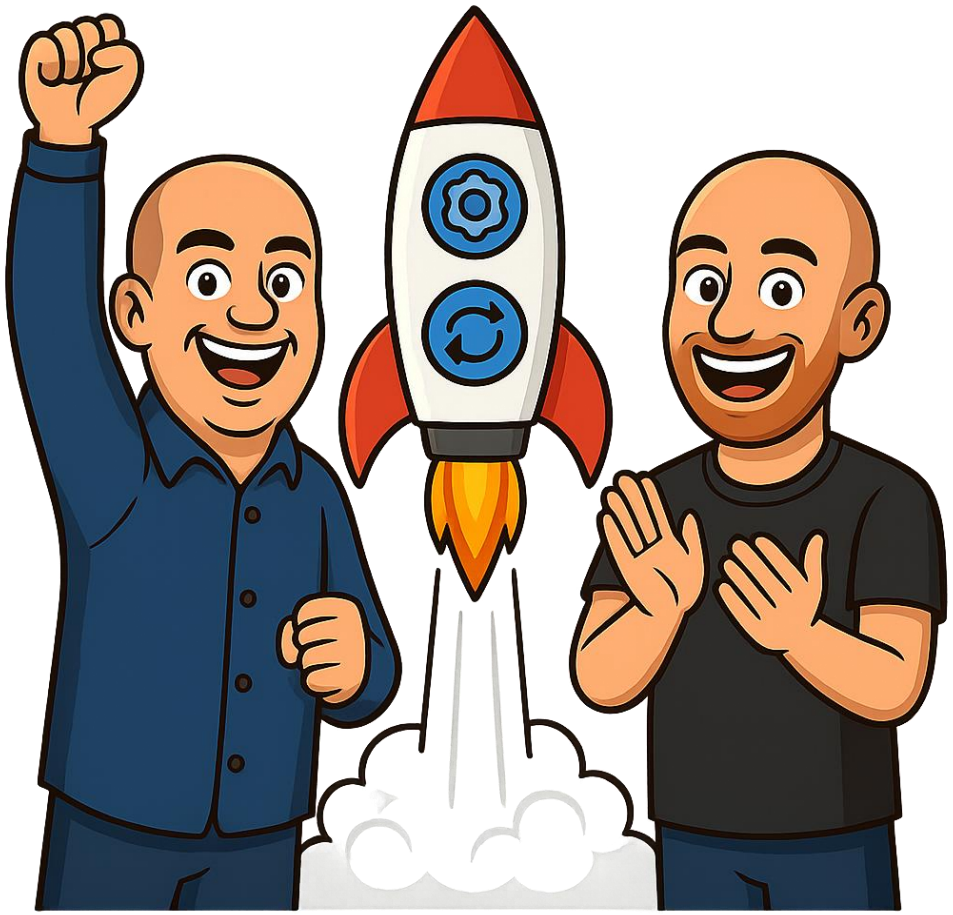
Ways-of-working

...AND AFTER THE BREAK?



Building our solution

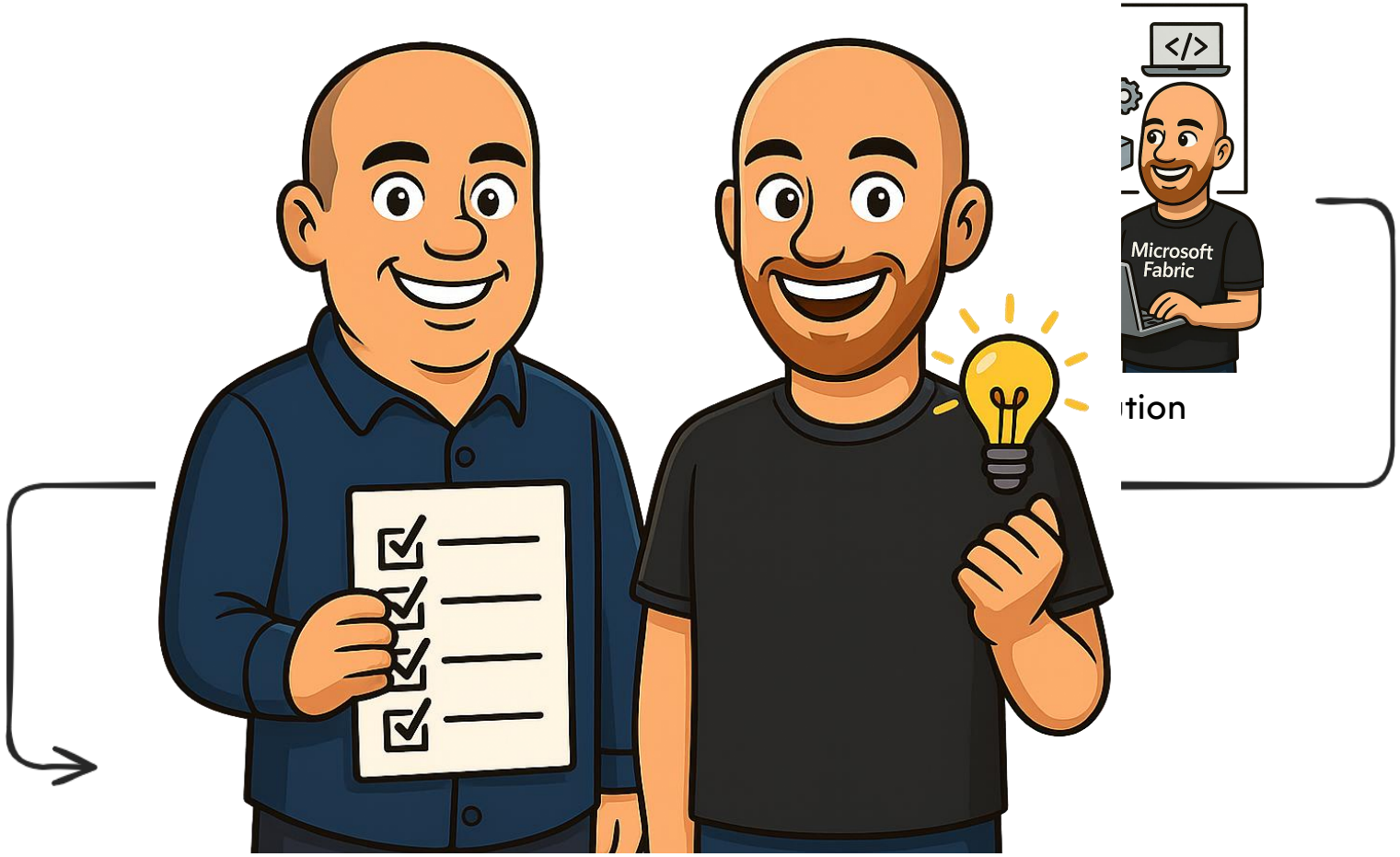
...AND AFTER THE BREAK?



Deployment

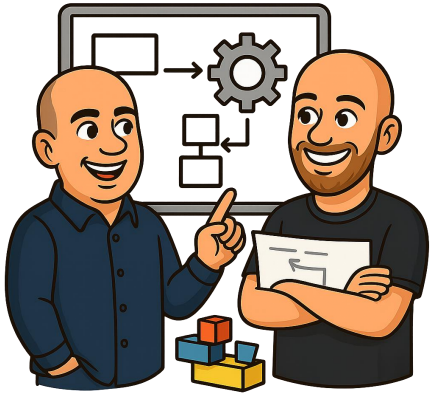
solution

...AND AFTER THE BREAK?

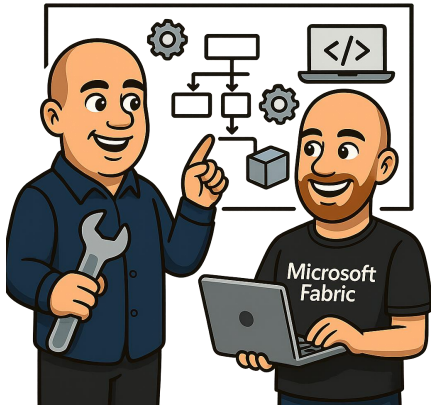


Best Practices, tips & tricks

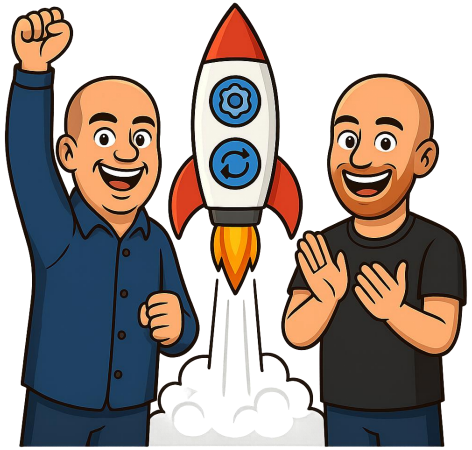
...AND AFTER THE BREAK?



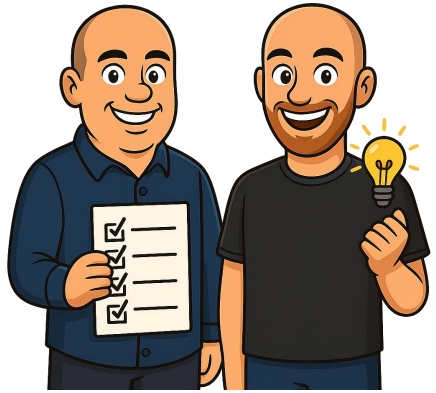
Ways-of-working



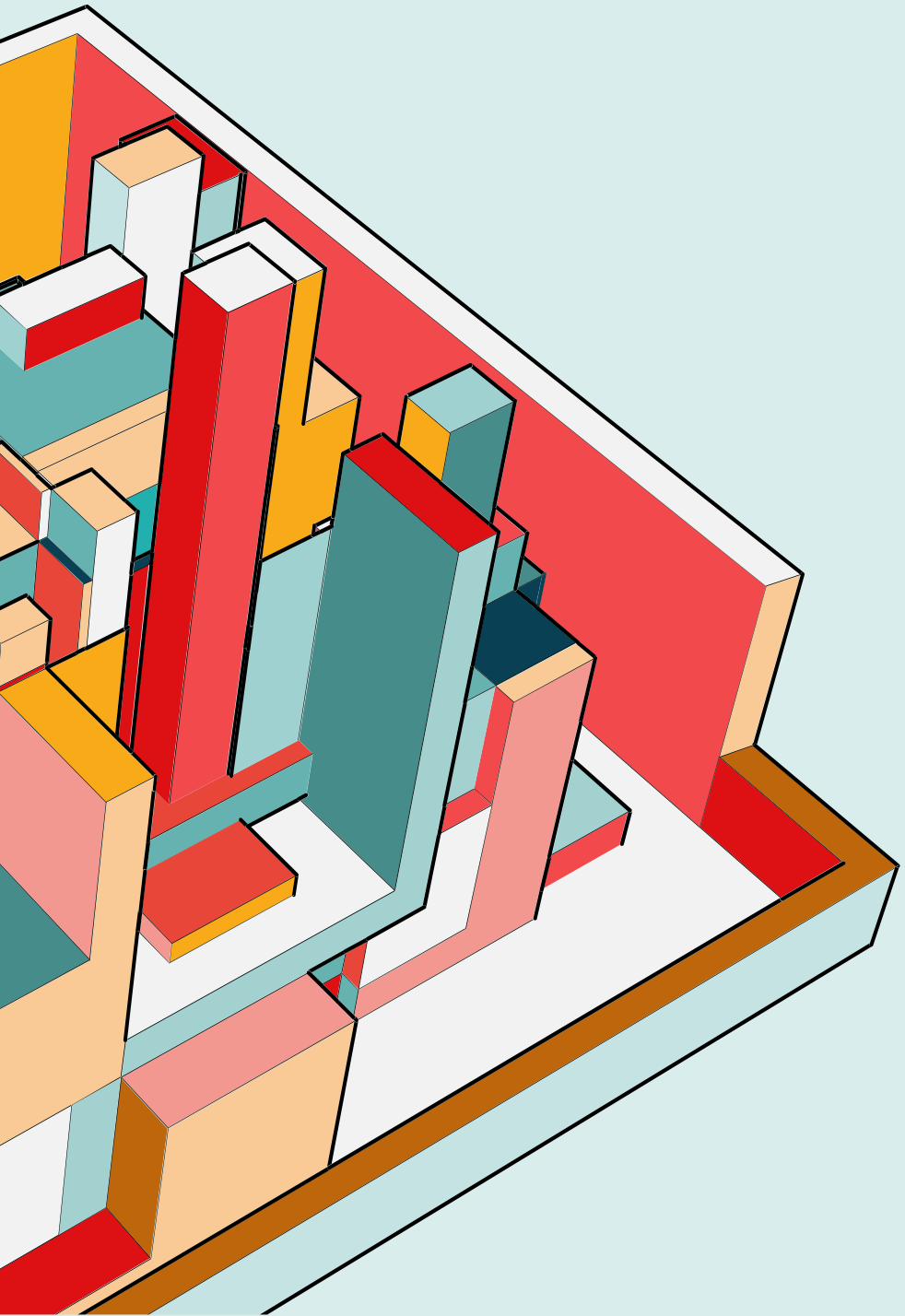
Building our solution



Deployment



Best Practices, tips & tricks

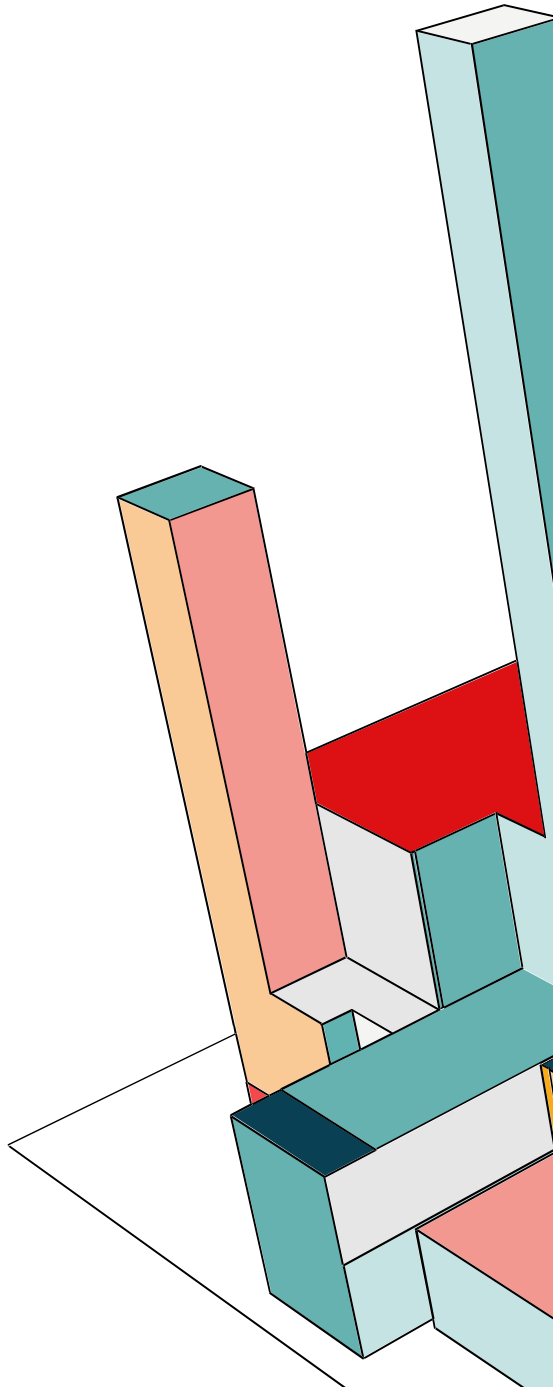


BEFORE WE START...

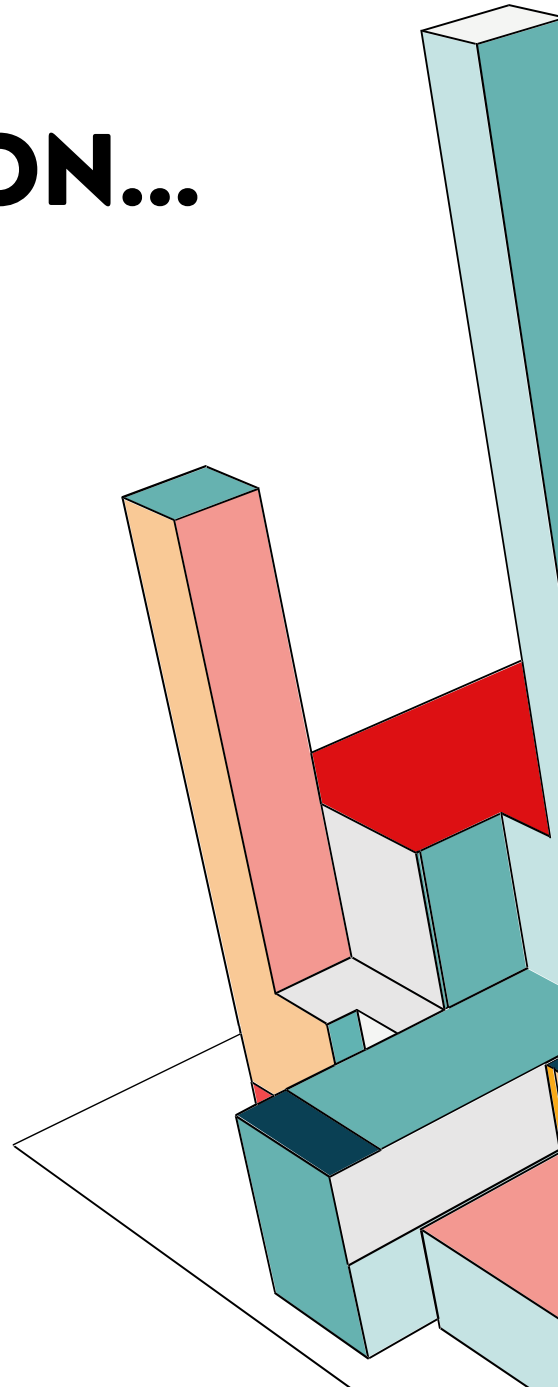
A quick poll

WHAT LEVEL ARE YOU?

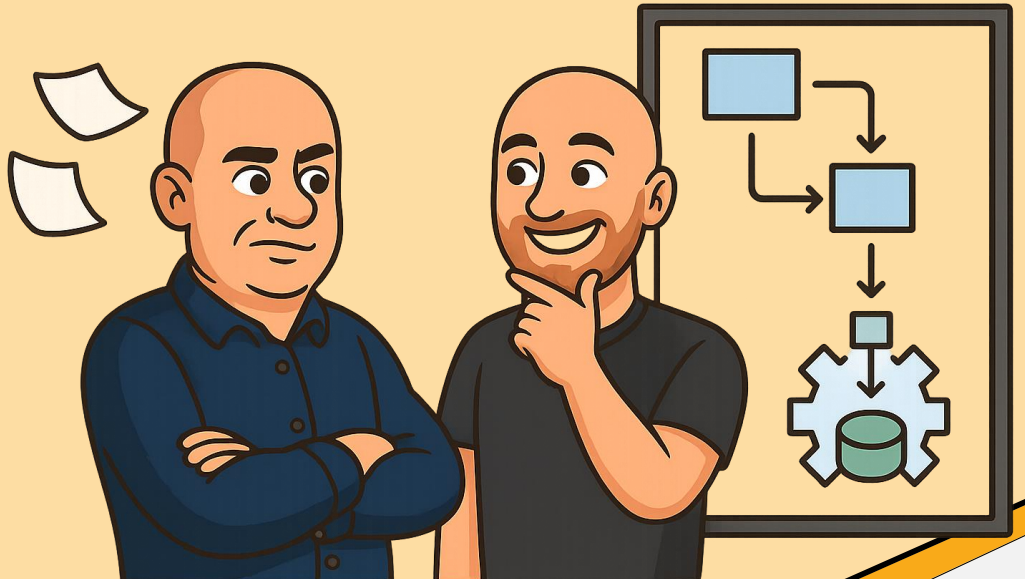
- Where are you on automation?
- And what tools are you using?
- Are you using Fabric git integration?



WHERE ARE WE NOW? POST-FABCON...



WHY AUTOMATION MATTERS!



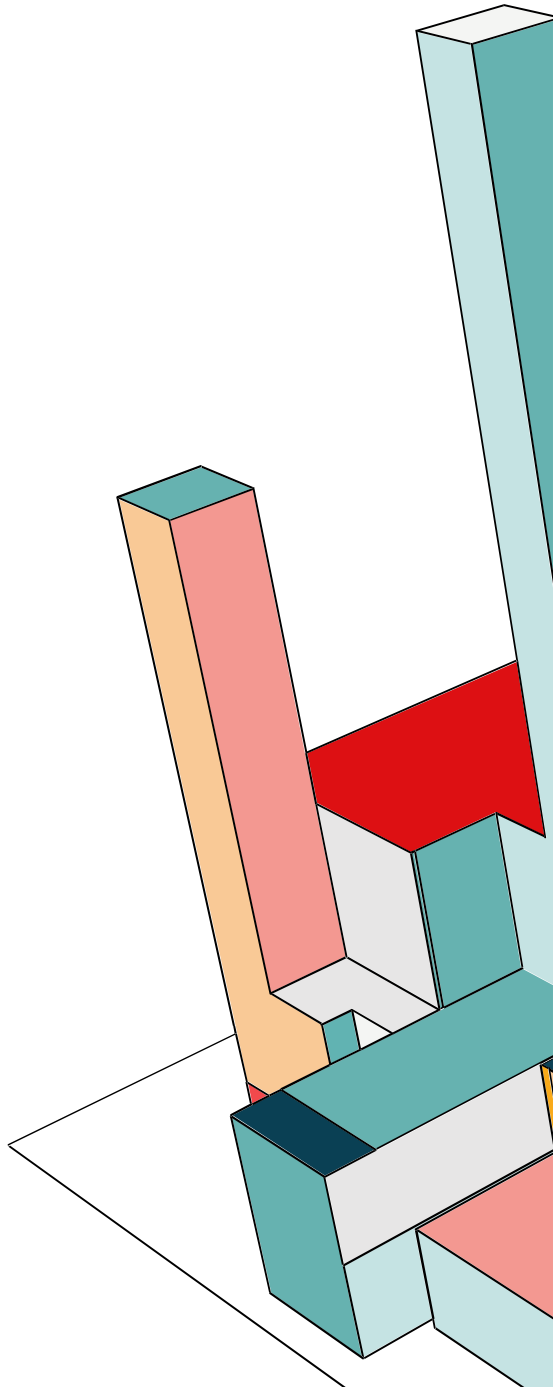
WHY AUTOMATION MATTERS!

Business drivers

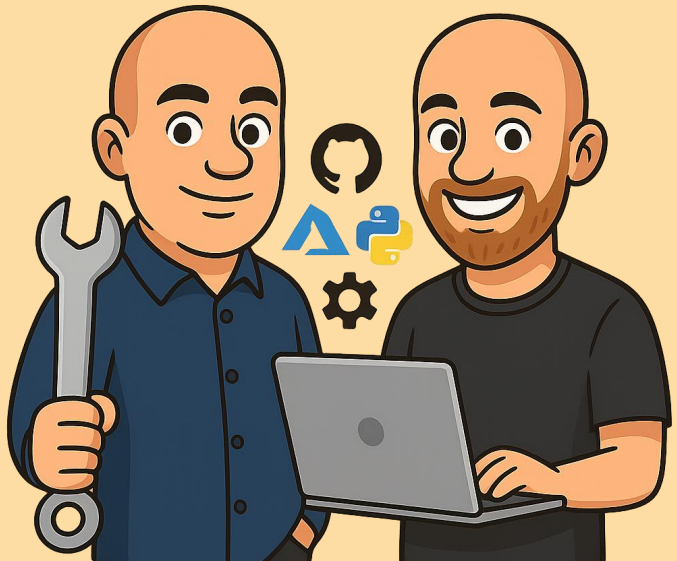
- Minimize risk
- Time to market
- Cost efficiency

Technical advantages

- Scalability
- Extensibility
- Flexibility



TOOLING LANDSCAPE FOR FABRIC AUTOMATION



FABRIC AUTOMATION & CI/CD LANDSCAPE

Git Integration	Variable library	fabric-cicd
Fabric REST APIs	Fabric CLI	Deployment pipelines
Semantic Links Lab	Terraform Provider	.NET SDK

FABRIC AUTOMATION & CI/CD LANDSCAPE

Git Integration

Variable library

fabric-cicd

Fabric REST APIs

Fabric CLI

Deployment pipelines

Semantic Links Lab

Terraform Provider

.NET SDK

FABRIC AUTOMATION & CI/CD LANDSCAPE

Git Integration	Variable library	fabric-cicd
Fabric REST APIs	Fabric CLI	Deployment pipelines
Semantic Links Lab	Terraform Provider	.NET SDK

FABRIC TOOLING LAYERS

High Level Tools
Enhance usability and automation

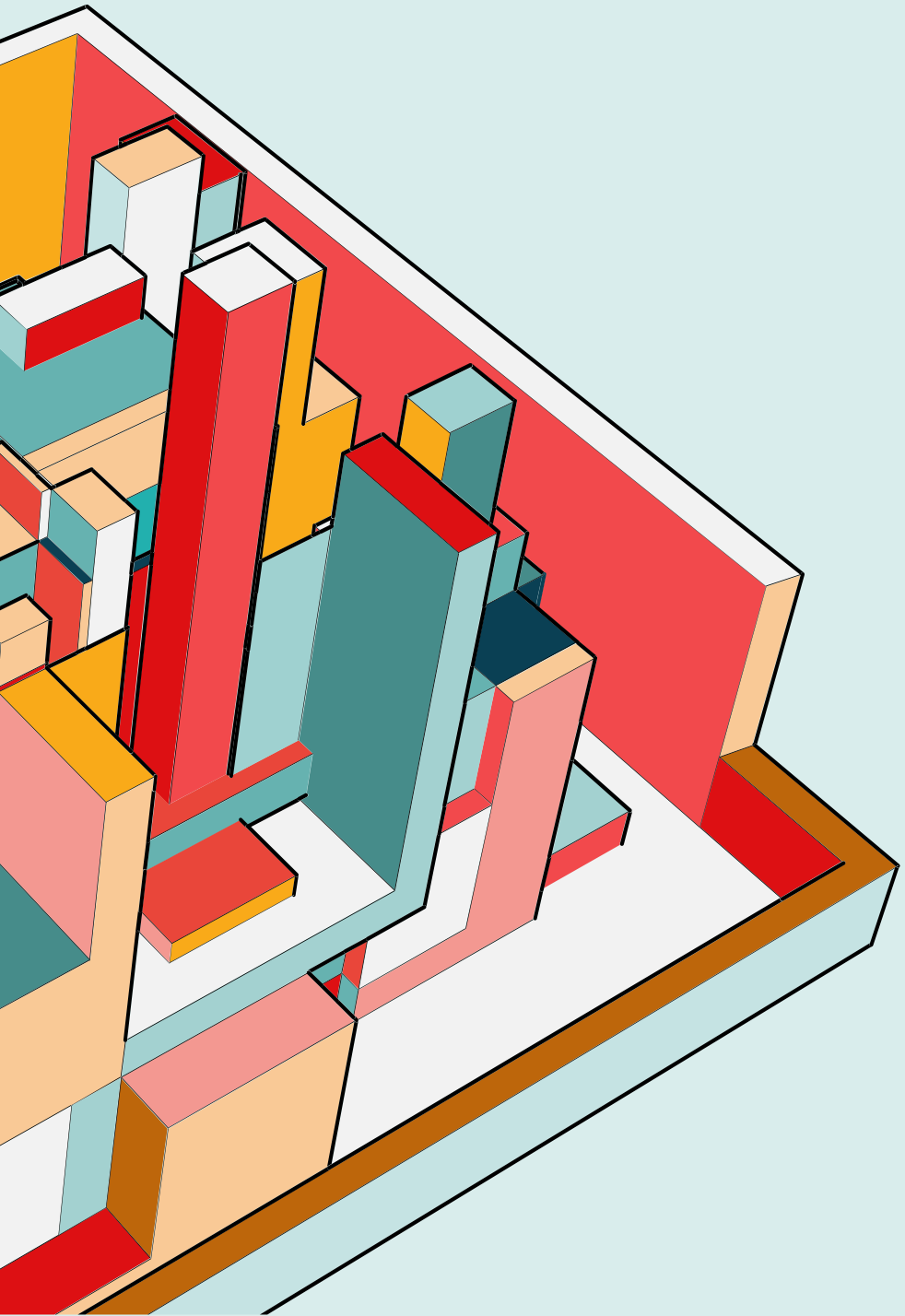
Terraform Provider
fabric-cicd library
Semantic Link Labs

Abstraction Layer
Provide direct interaction with the APIs

Fabric CLI
.NET SDK

Foundation Layer
Serve as the base for all interactions

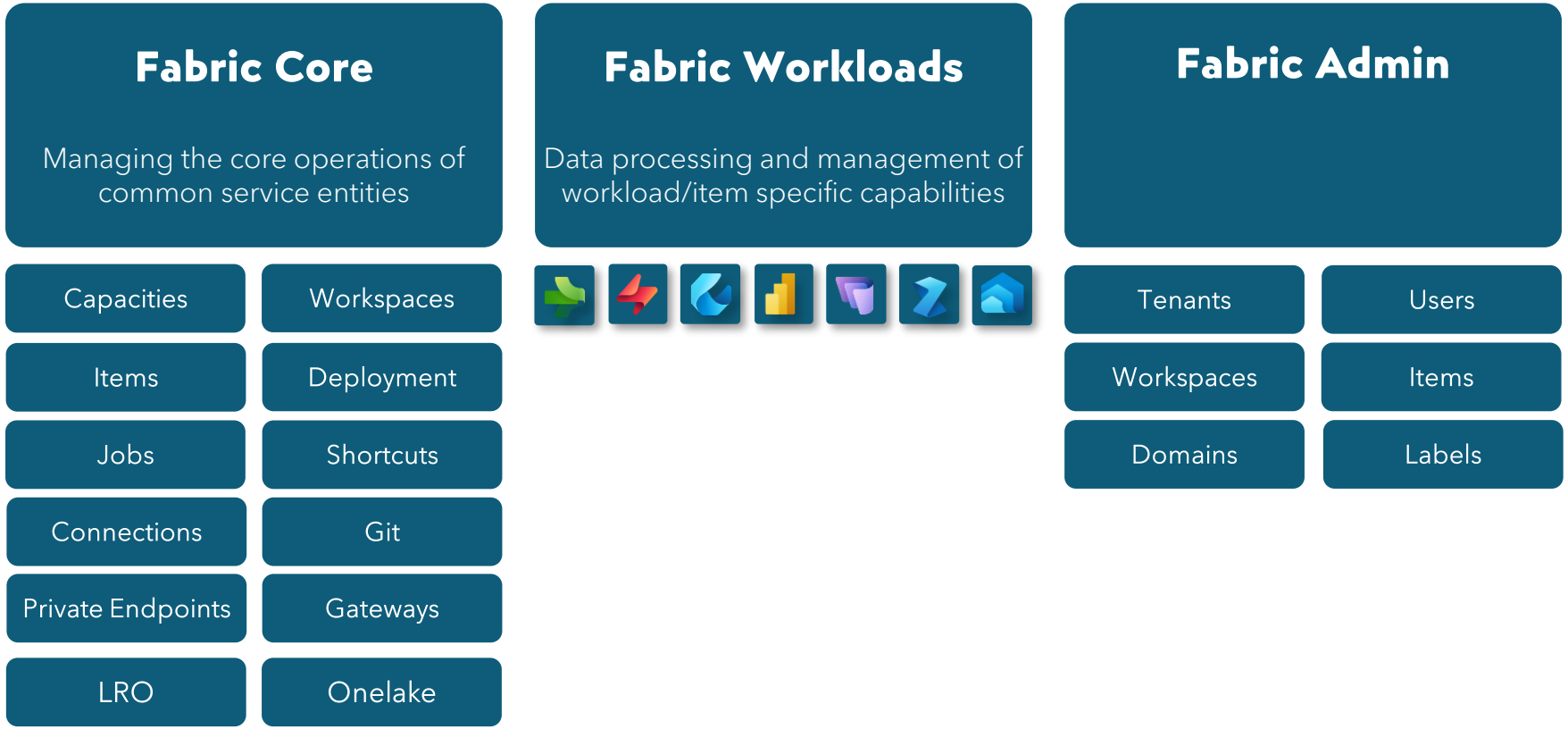
Fabric REST APIs



FABRIC REST APIS

The foundation for automation

AUTOMATING FABRIC USING REST APIS

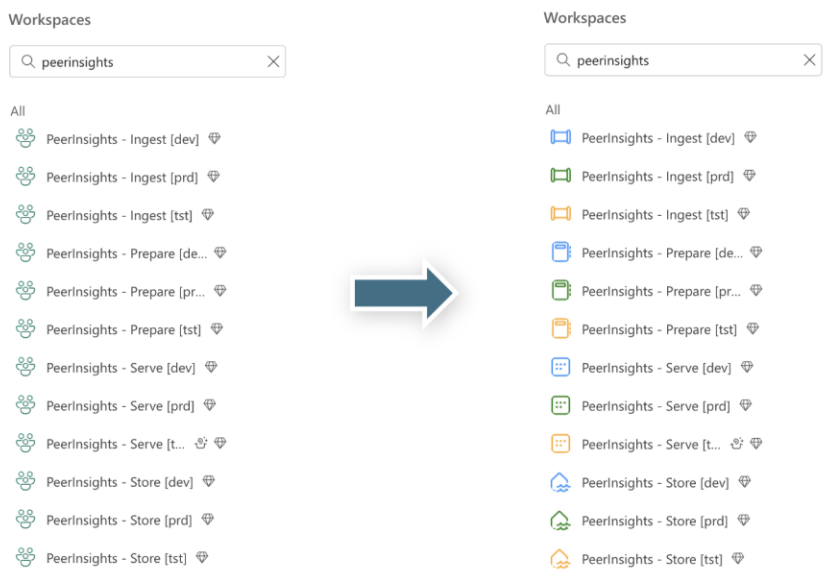


https://learn.microsoft.com/en-us/rest/api/fabric/articles/

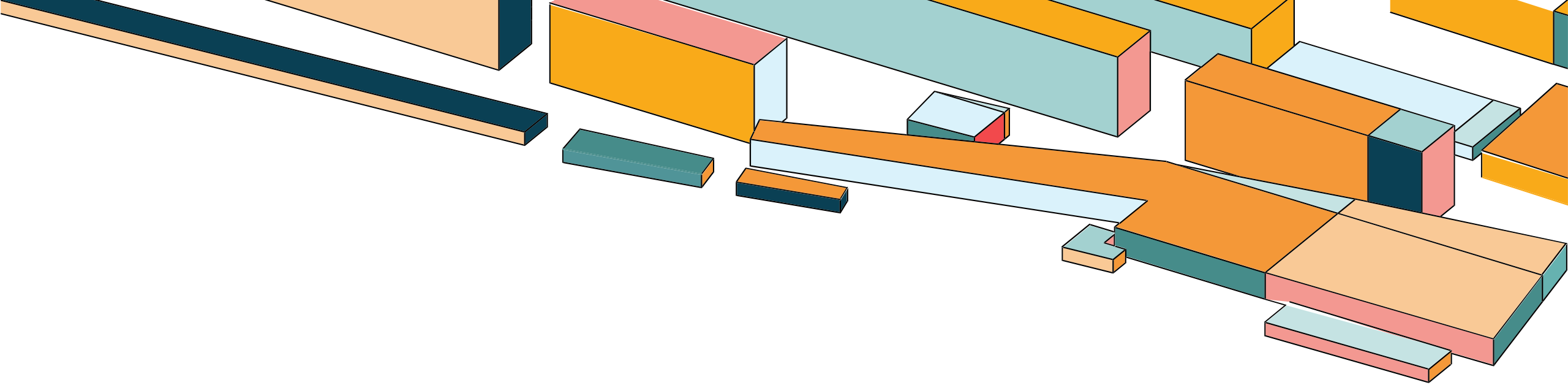
INTERNAL AND UNSUPPORTED APIS

Where there is a will – there is a way…

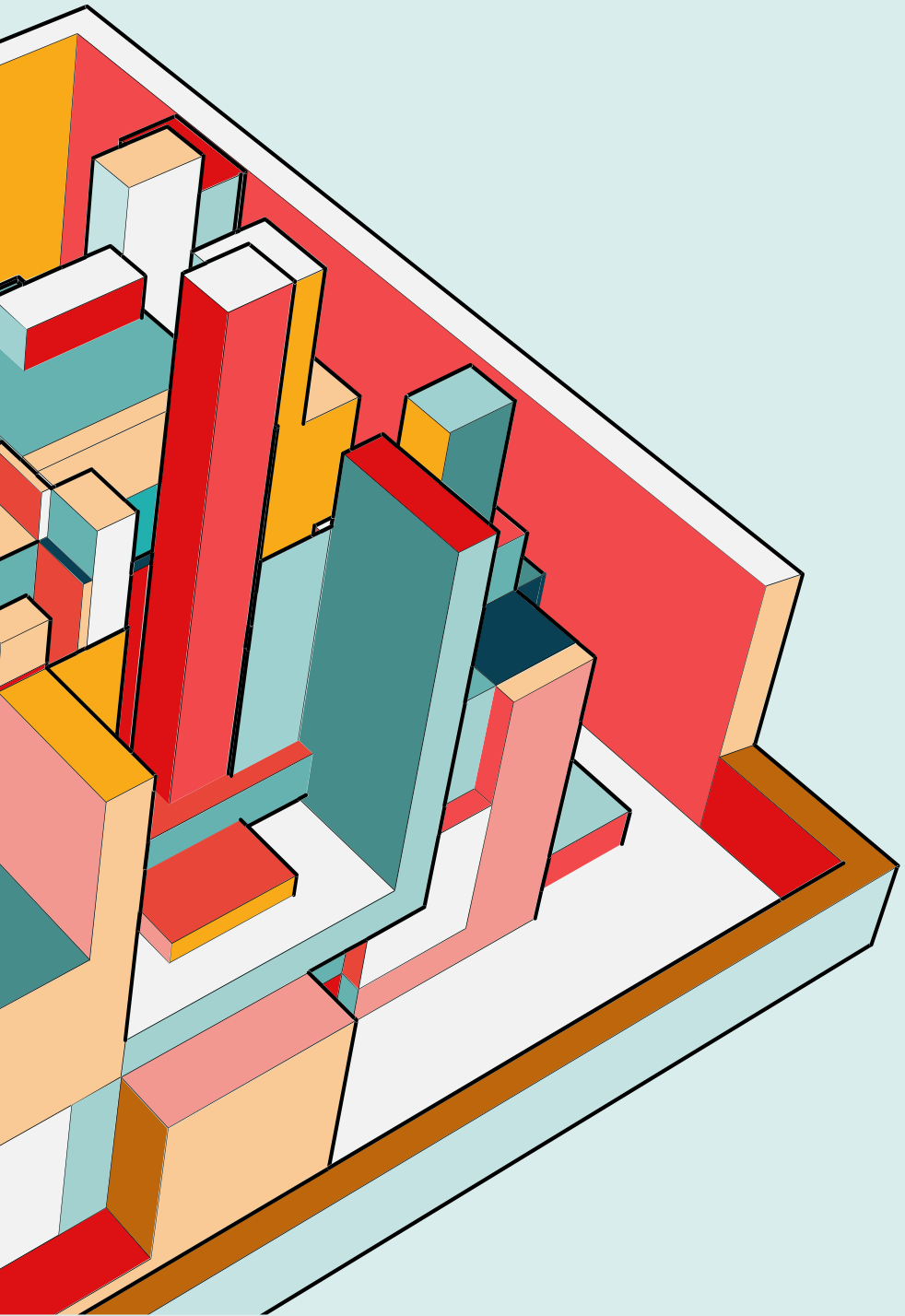
- What is possible in the Fabric/Power BI Service is most likely also possible to automate
- Get ready to do some reverse engineering
- Tooling:
  - Browser (developer mode)
  - Postman
  - VS Code, Cursor or other IDEs
  - Fabric Notebooks



- Example: https://peerinsights.hashnode.dev/automating-fabric-maintaining-workspace-icon-images



DEMO



FABRIC CLI

The Command Line Way

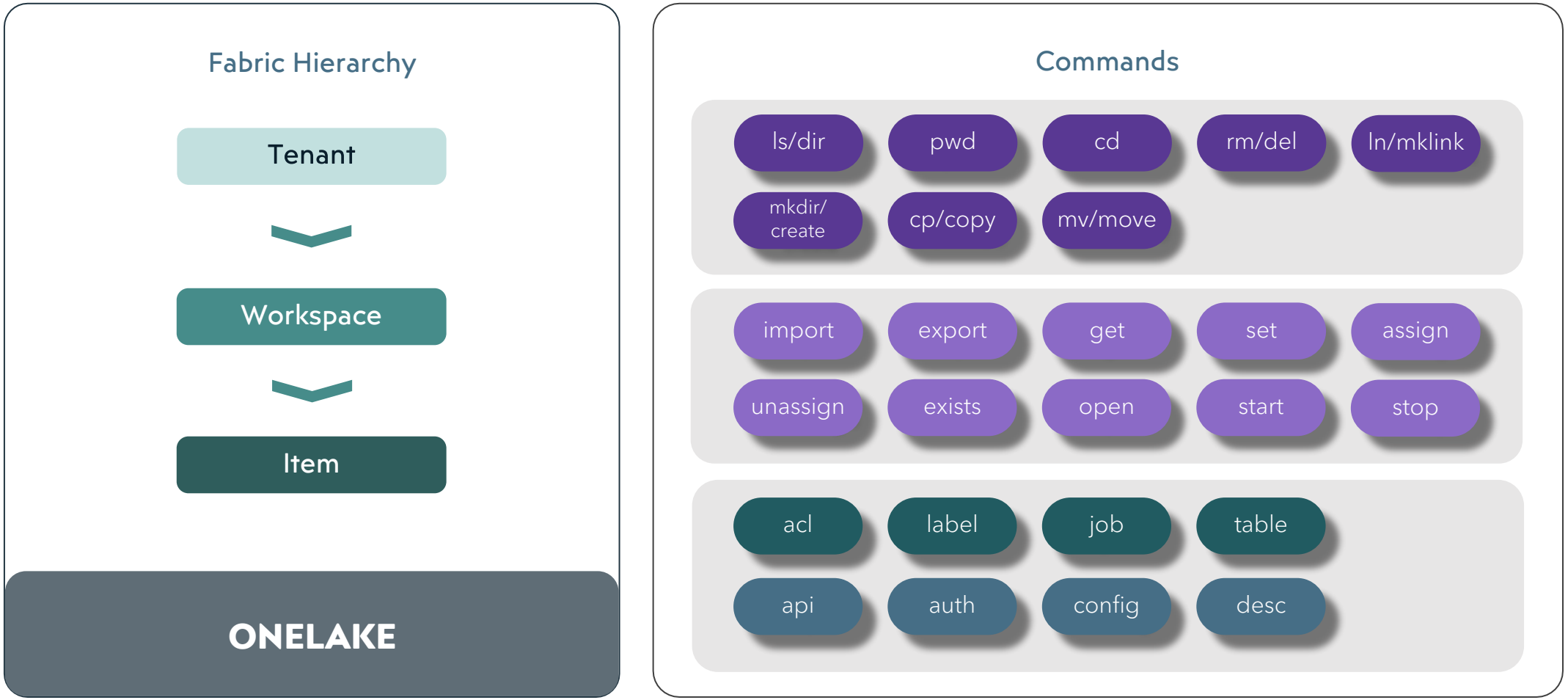
FABRIC CLI – THE COMMAND LINE WAY

GA & OSS

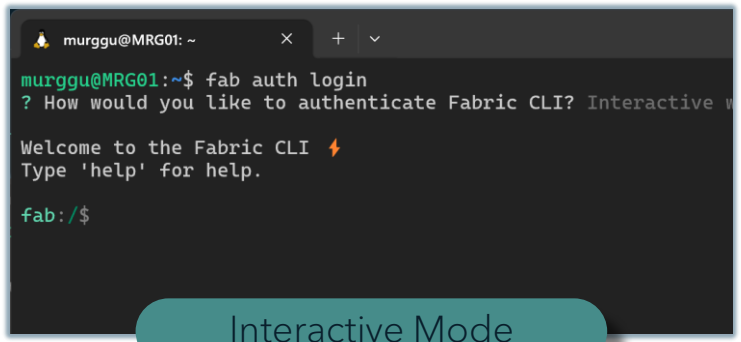
https://aka.ms/FabCLI

- **Built for Developers**
  - Operate and automate Fabric from your terminal, scripts, and pipelines.
  - No more bouncing to the UI.
- **Simple Mental Model**
  - Navigate Fabric like a file system.
  - Interactive and Non-Interactive modes
- **Based on Public APIs - without the complexity**
  - Wraps Fabric, Power BI, OneLake & ARM APIs.
- **Works with your favorite IDEs like VS Code etc.**
- **In GA and just open-sourced!**

FABRIC AS A FILE SYSTEM



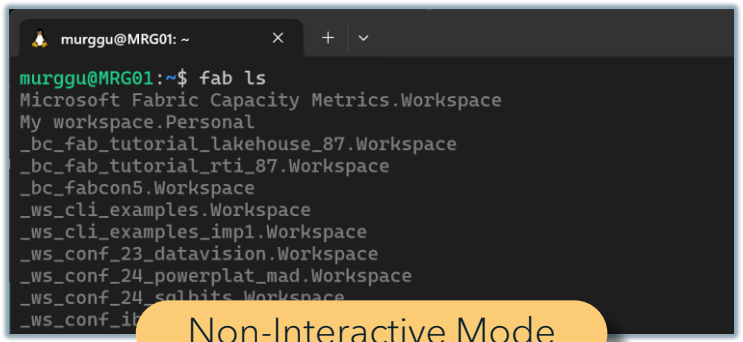
CLI MODES



Type queries interactively in a terminal, execute them, and see the results.

Type fab auth login and enter

- Run commands interactively within a session
- Results displayed in real-time



Pass queries directly as command-line arguments when invoking fab <command>.

Use fab <command> directly from terminal

- Useful for quick queries or automation
- Ideal for CI/CD with GitHub Actions, ADO pipelines, and scripts

WHY WE NEED A BETTER WAY

Enhanced local interaction

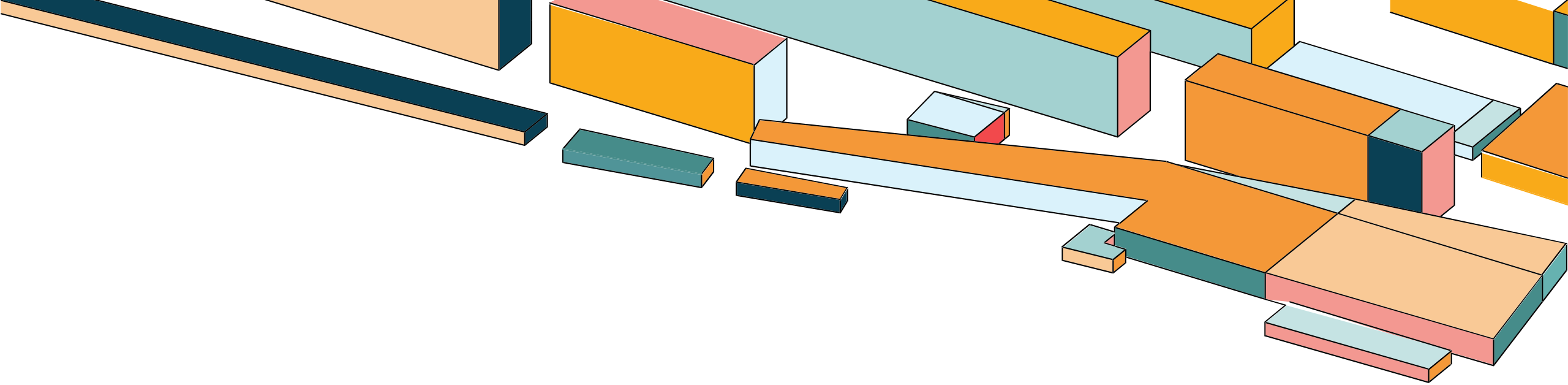
- No constant need to switch to the browser
- Streamline workflows and better focus
- Flexibility for quick, local tasks

Better developer experience

Automation & custom CI/CD

- Set up tailored pipelines
- Enhanced deployment automation
- Integrate with Azure DevOps and GitHub

Simplified automation and scripting



HOW TO GET STARTED WITH THE CLI

SETUP TO RUN CLI COMMANDS

Prerequisites

- Python version 3.10, 3.11, or 3.12
- Python must be is accessible from your terminal (PATH environment variable)

Installation

- pip install ms-fabric-cli

Use full configuration commands

- fab config set encryption_fallback_enabled true
- fab config set mode interactive

Authentication

Identity Type	Scenario
User	Interactive browser login
Service Principal	Secret Certificate Certificate + password Federated token
Managed Identity	System-assigned User-assigned

GETTING STARTED AS A NON HERO

WHY – AN EXAMPLE

Example: Creating the foundation for a medallion architecture with the Fabric CLI

- Create 3 workspaces
- Assign capacity
- Create 3 lakehouses
- Create shortcut
- Import Notebook

Using Fabric UI

Manual effort

No support for automation

~10 min.

~ 50 clicks + 100 keystrokes

Python & REST APIs

High initial effort

Supports automation

< 10 sec. / hours of initial work

+500 lines of code

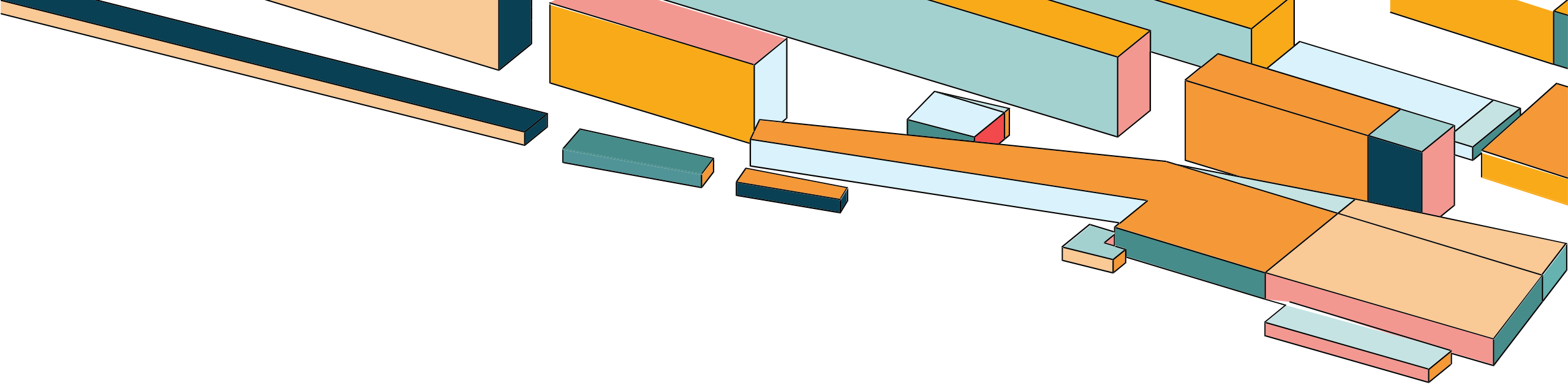
Using the Fabric CLI

Low initial effort

Supports automation

< 10 sec. / a few mins.

< 10 commands

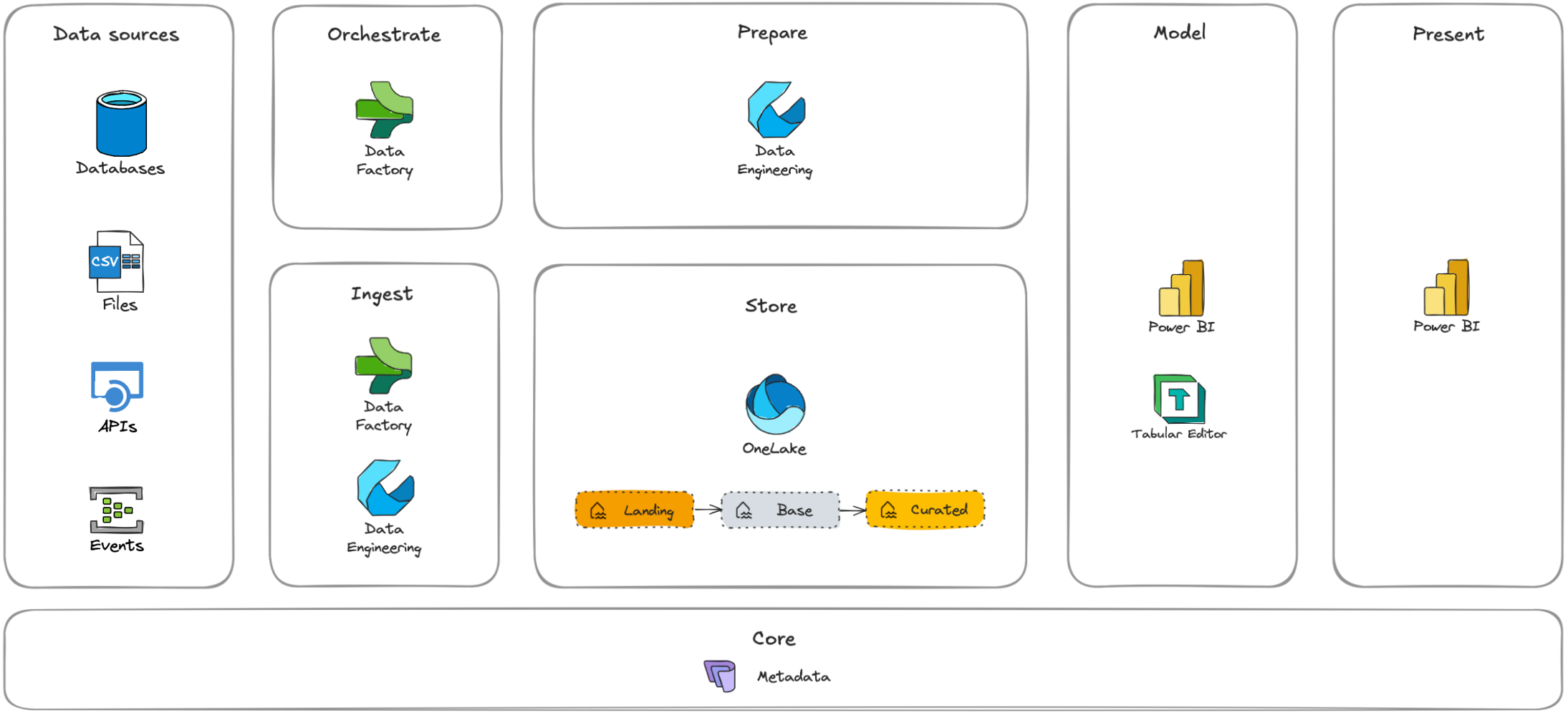


DEMO

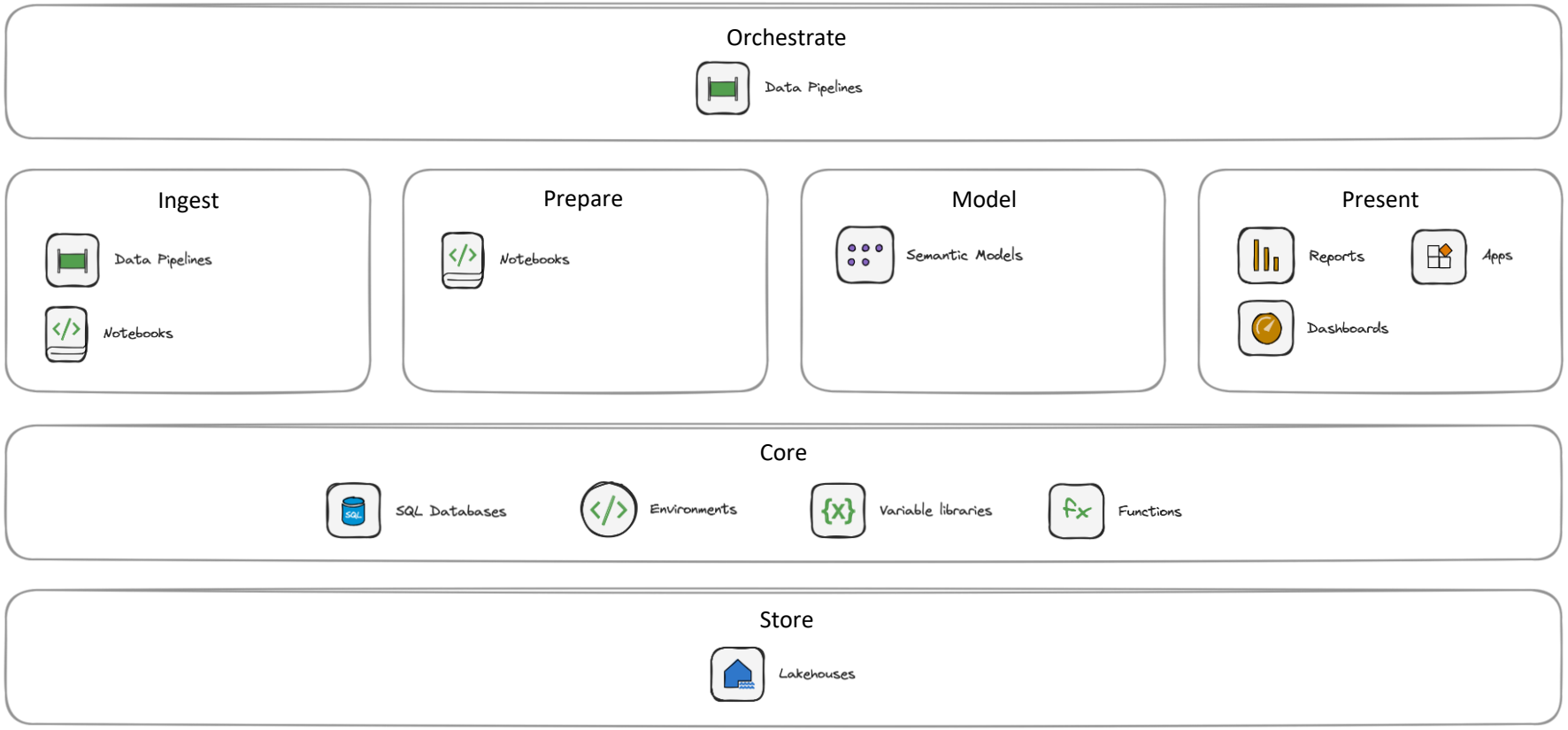
ARCHITECTING FOR AUTOMATION

HOW DO YOU ARCHITECT?

REFERENCE ARCHITECTURE AS A START



WORKSPACE STRUCTURE

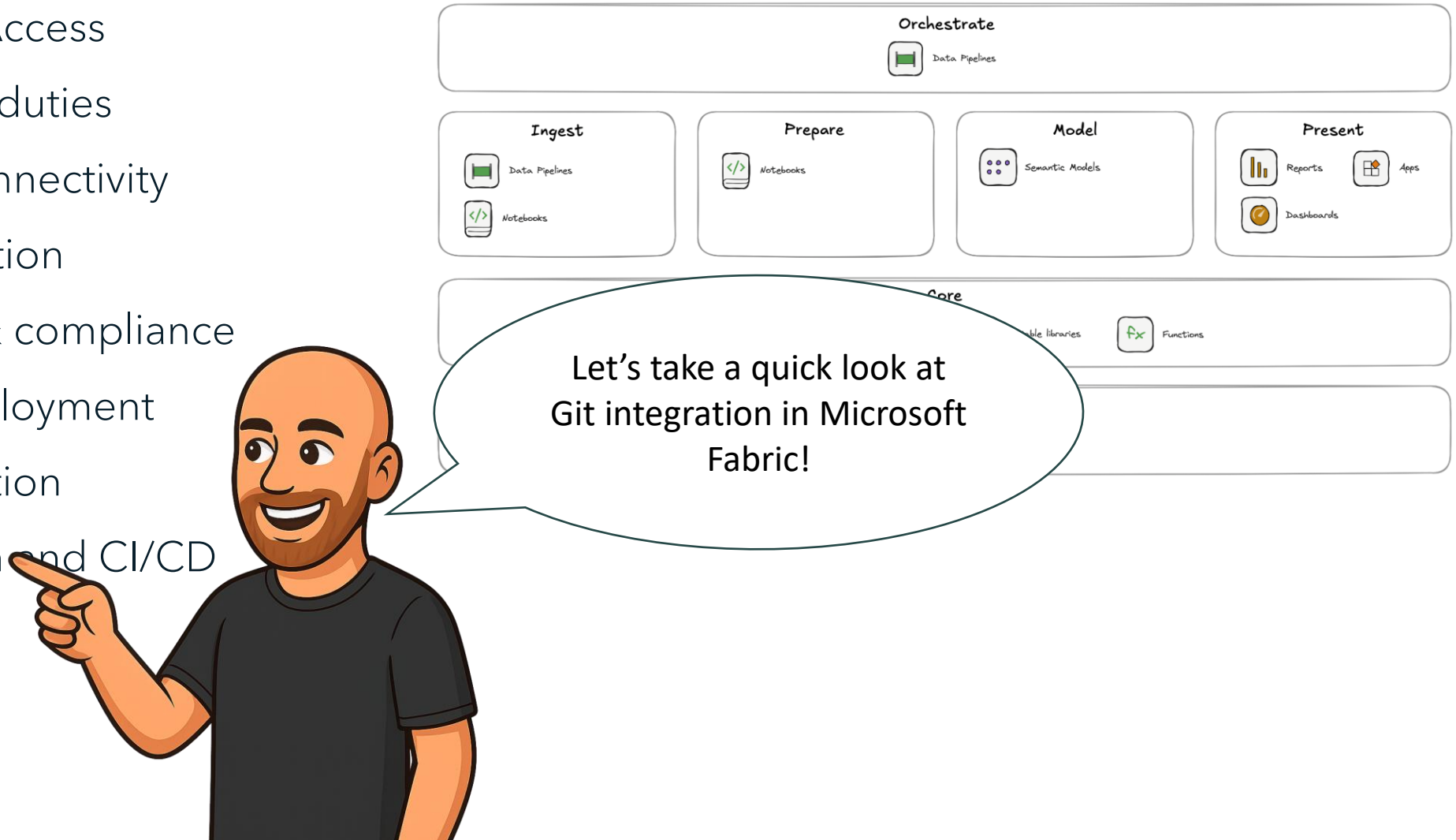


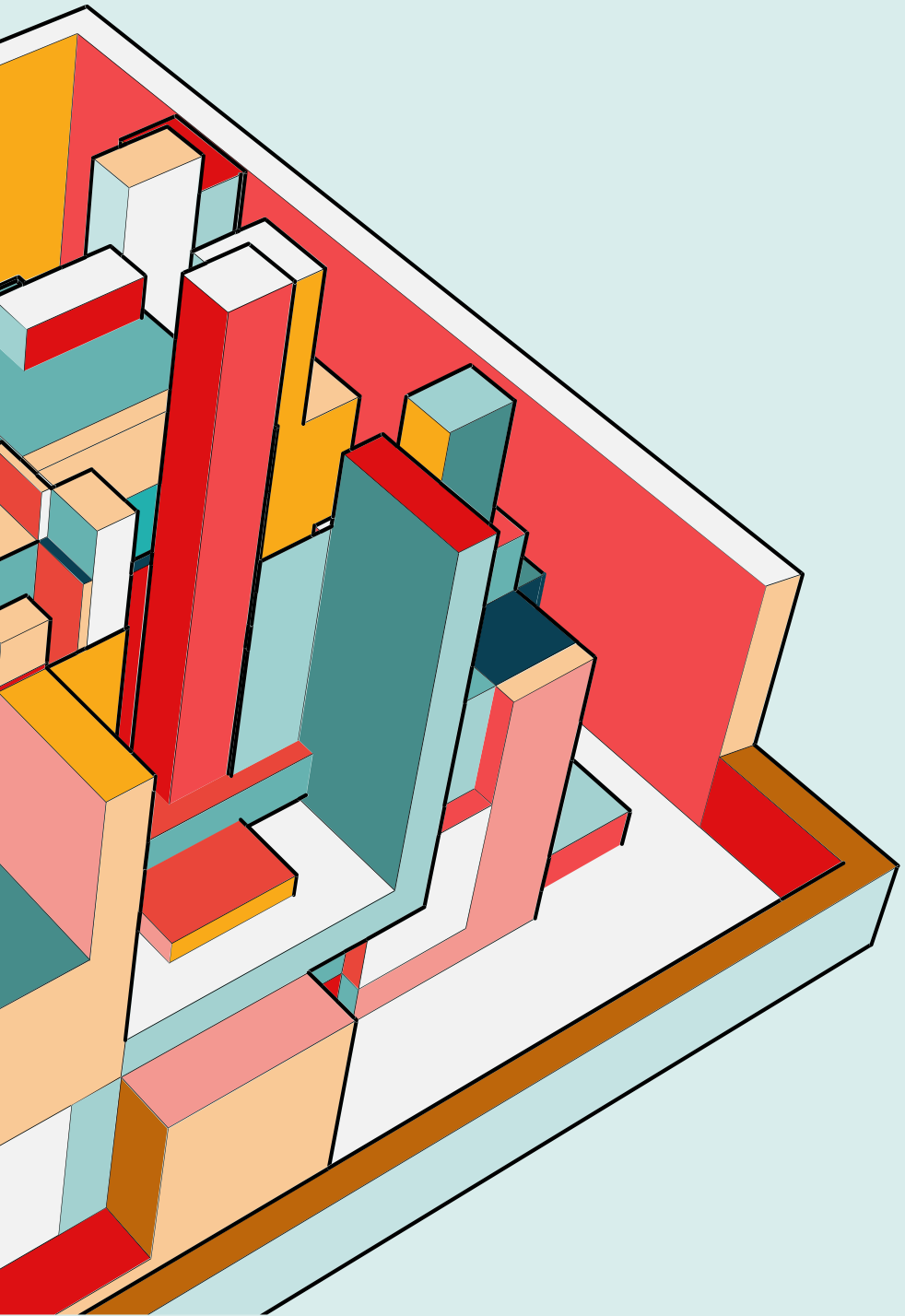
x3

[dev, tst, prd]

WORKSPACE STRUCTURE – WHY THIS?

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





GIT INTEGRATION

Source control in Fabric

WHY CI/CD & GIT MATTERS...

The classic foundations

Version control

Change reviews

Testing before releasing

Reproducibility

Collaboration

And the Fabric angle...

Declarative and cloud-native

Items map to Git folders

Enables controlled, repeatable deployments

Use of purpose-built tools

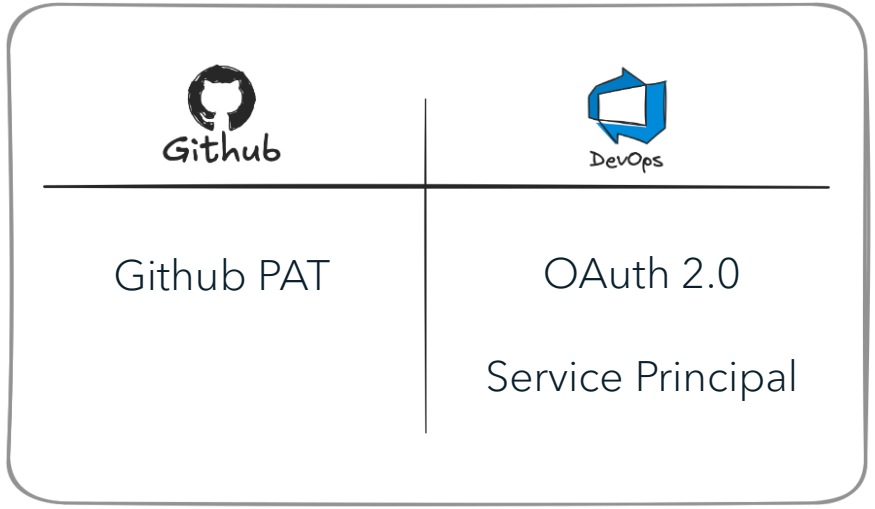
Support for using template repositories with boilerplate scripts, dynamic pipelines etc.

Collaboration at scale

GIT INTEGRATION IN FABRIC

- Integrates on workspace level
- Workspace -> branch
- Supports Github and Azure DevOps
- Sync is bi-directional (push/pull)
- Native support for **ALL** items
  (some in preview)
- API support (Fabric Core REST APIs)

Authentication options

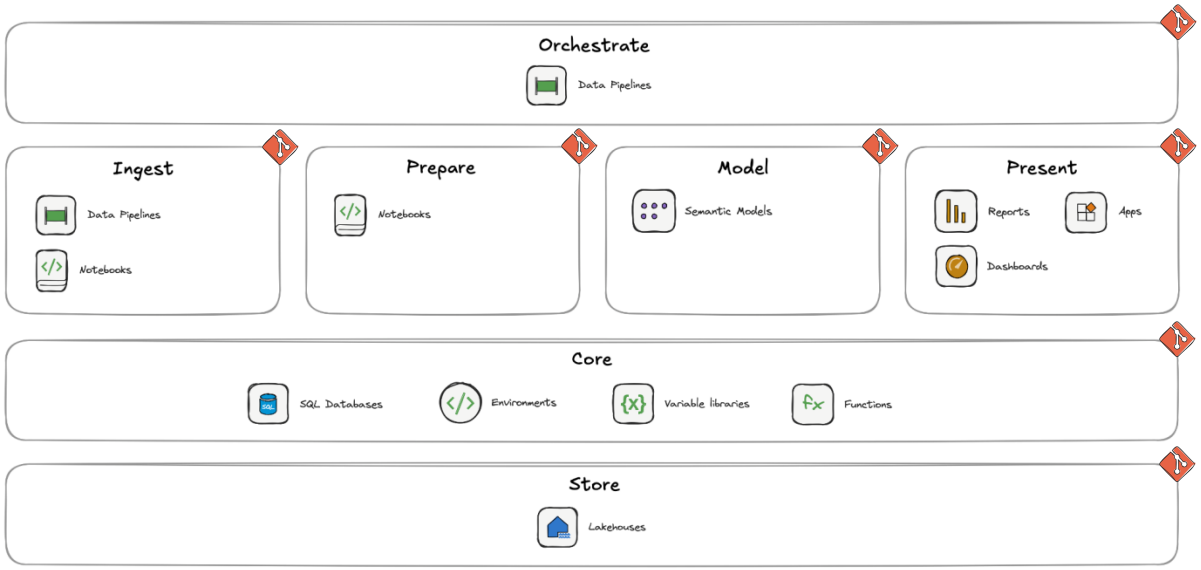


Read more: https://learn.microsoft.com/en-us/fabric/cicd/git-integration/intro-to-git-integration

A GO-TO FOR FABRIC GIT INTEGRATION

- Use Mono-repo
- Organized your workspaces by layer
- Using clear naming conventions
- Use the Git provider of your choice
- Branching strategy depends…

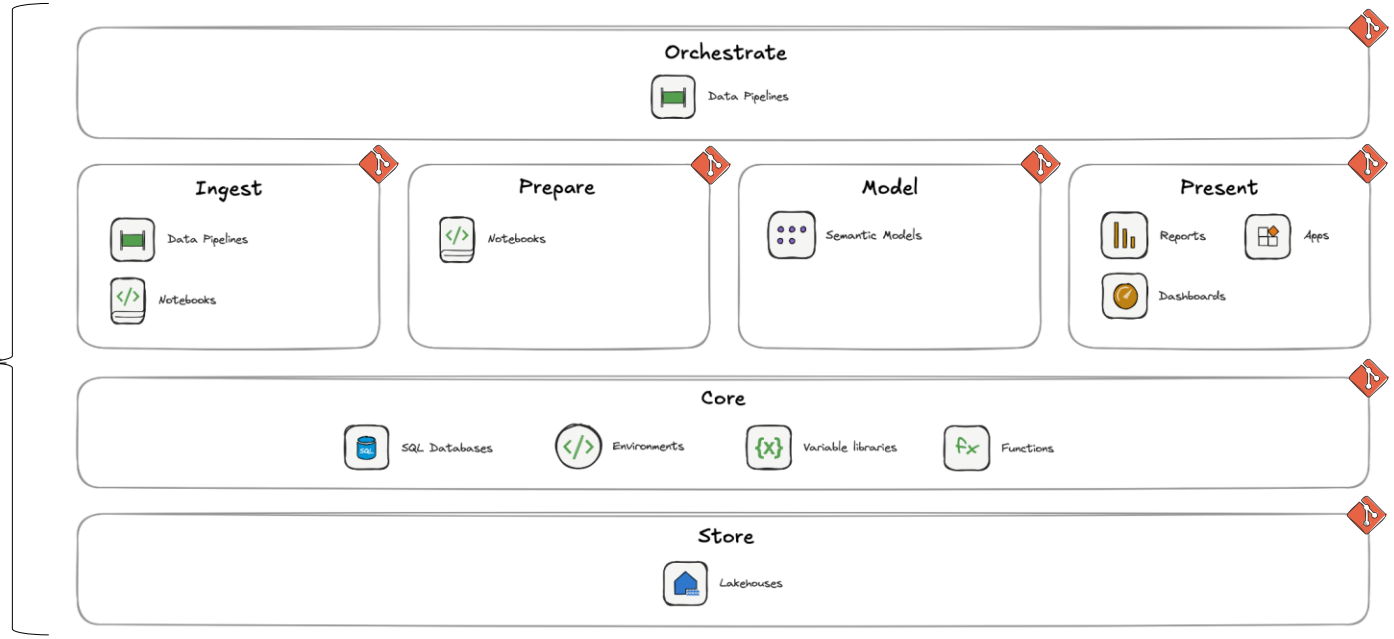
  there is no one-size fits all…
- Git integration all [dev] workspaces by default

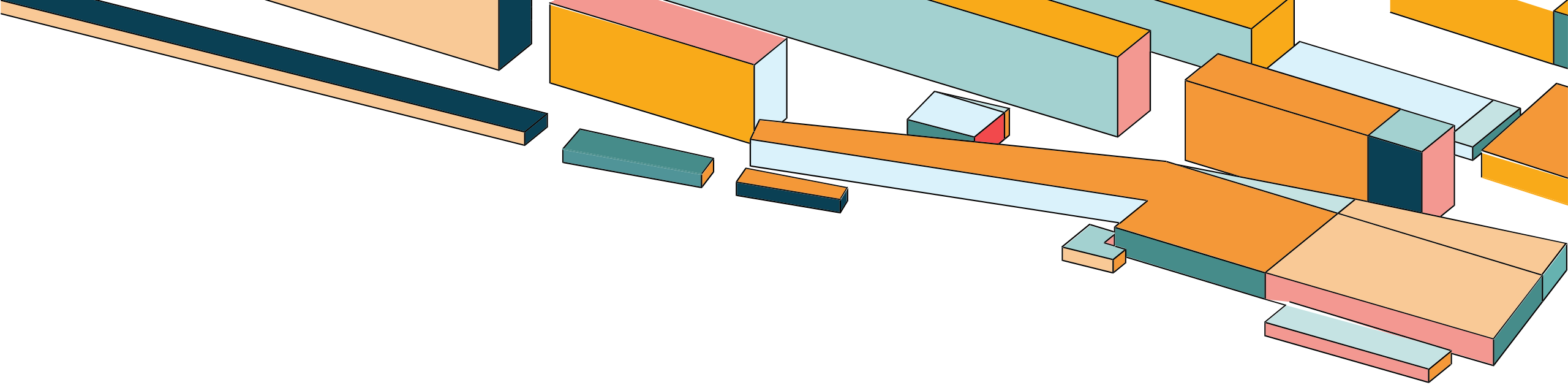


A GO-TO FOR FABRIC GIT INTEGRATION

Git Repo structure

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



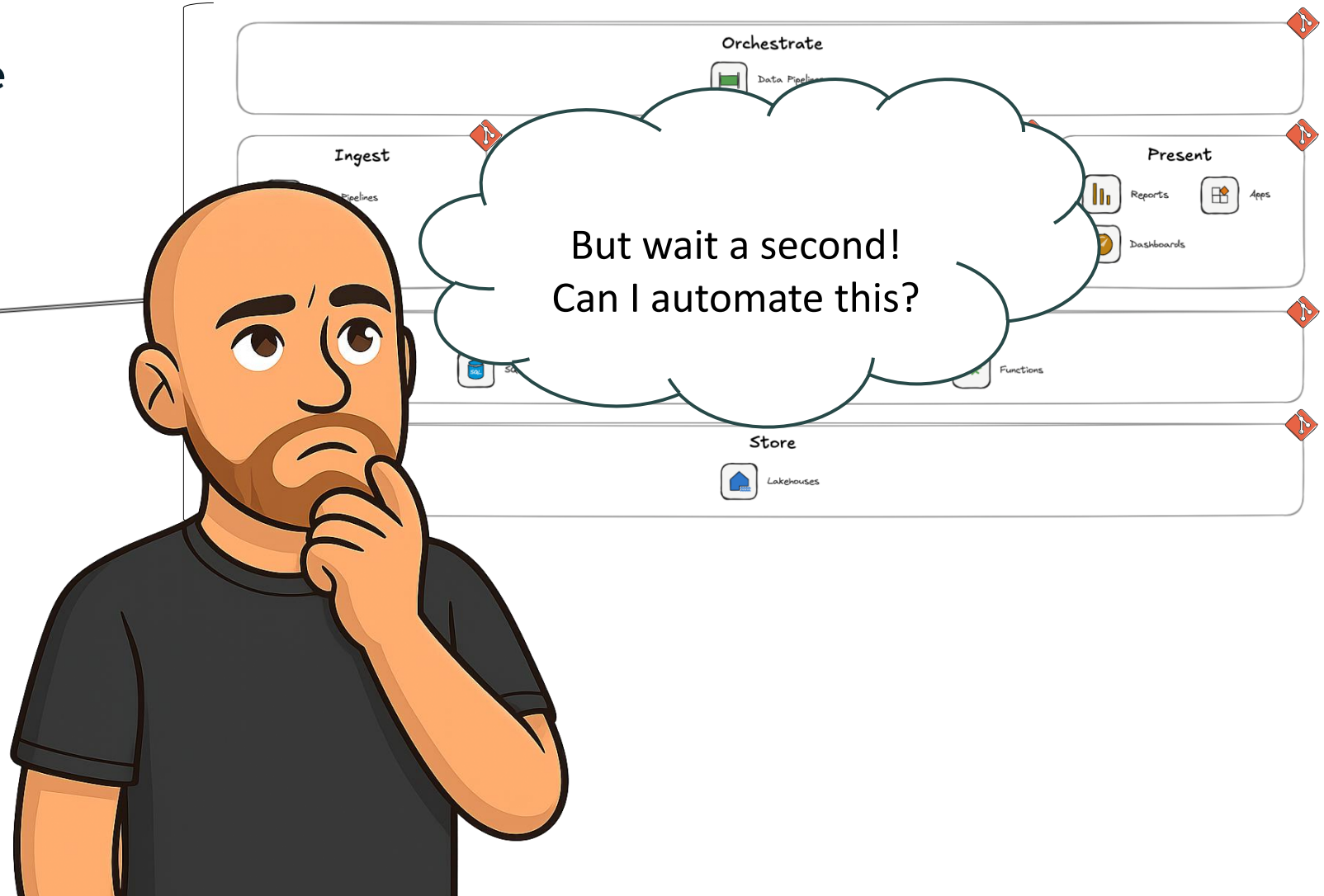


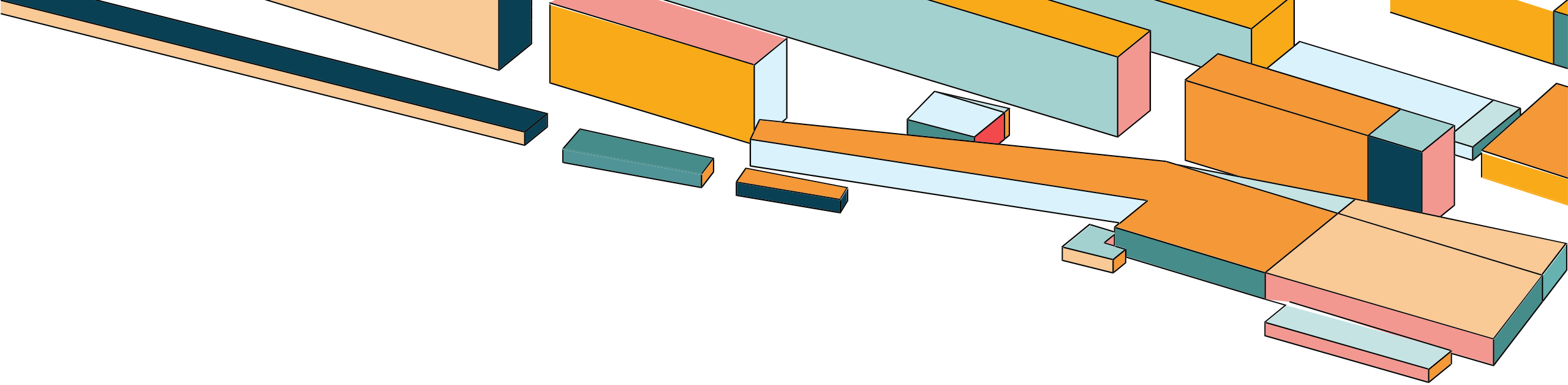
DEMO

A GO-TO FOR FABRIC GIT INTEGRATION

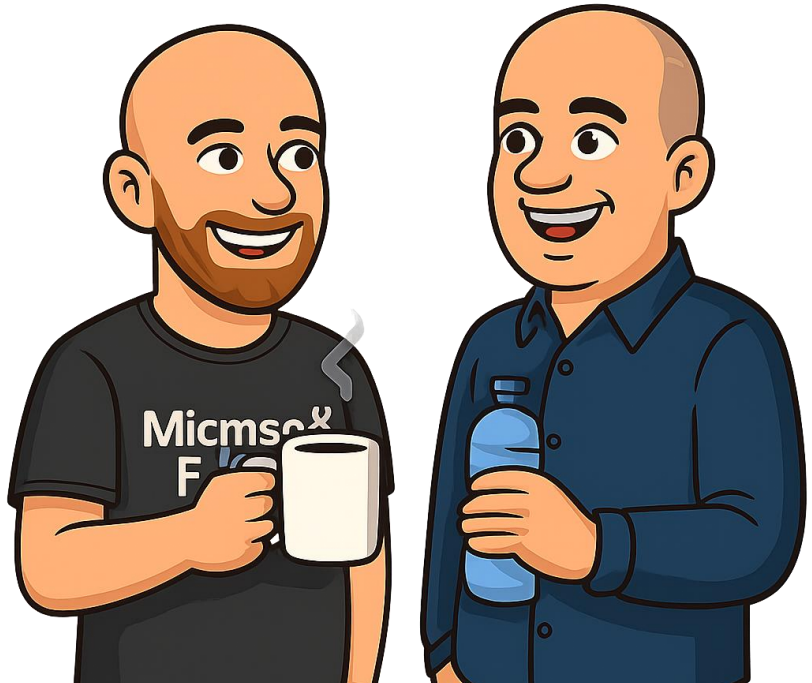
Git Repo structure

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

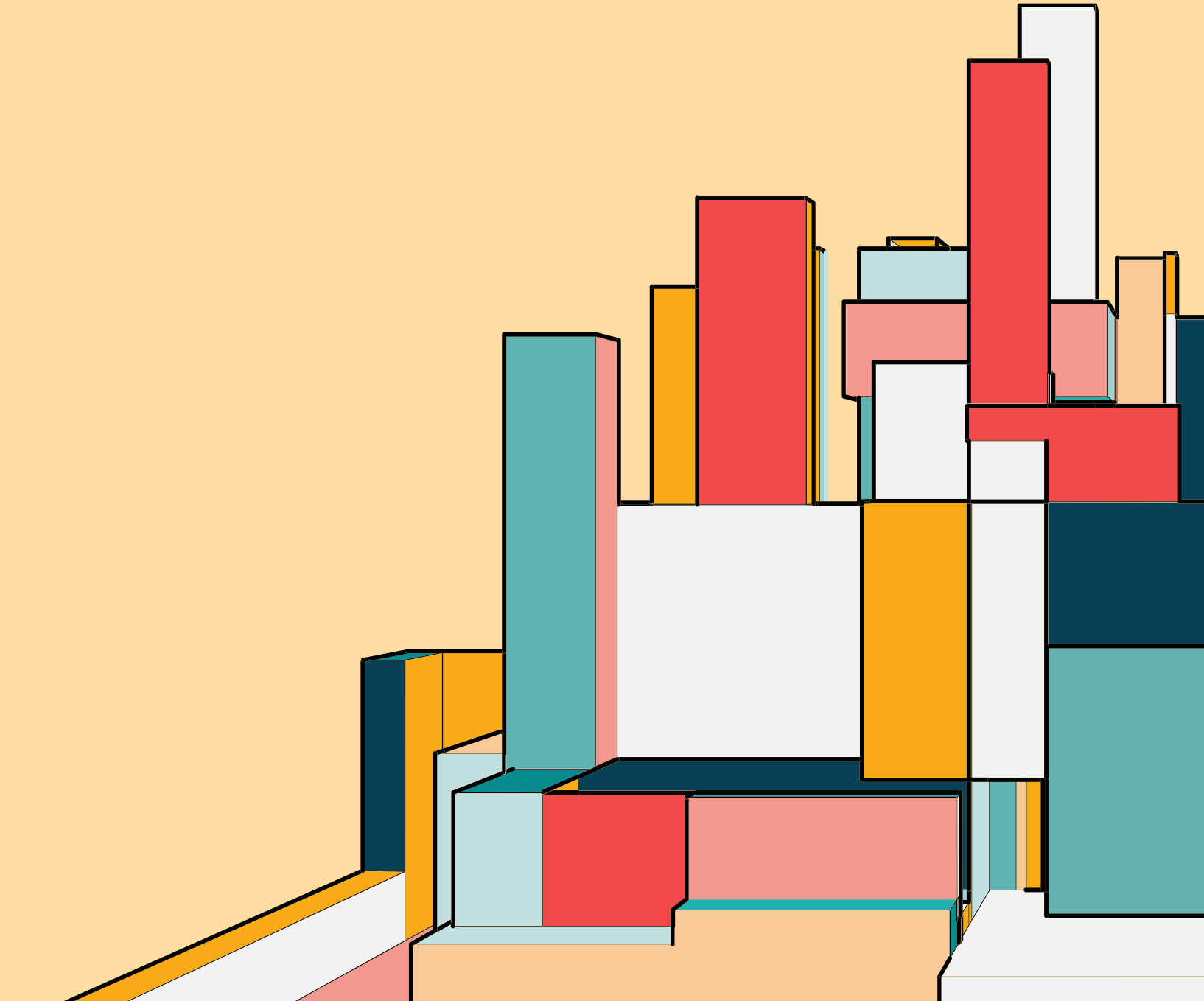




BREAK

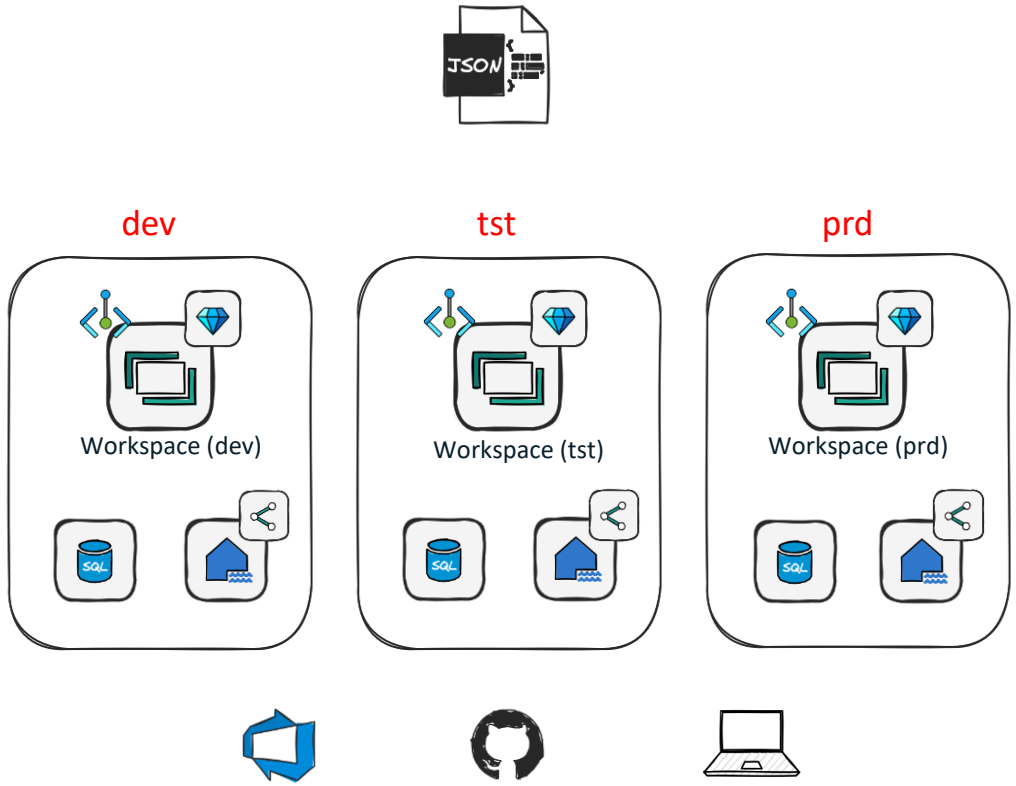


FABRIC IAC



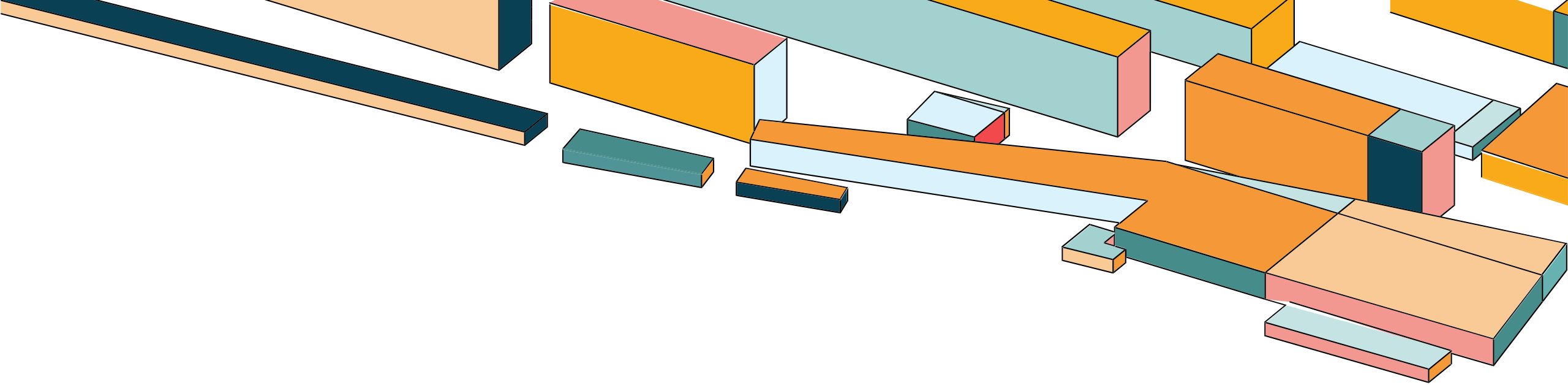
AUTOMATING FABRIC SOLUTION SETUP

- Recipe based solution (json)
- Utilizes Python and the Fabric CLI
- Can run from Azure DevOps and GitHub as well as on your locale machine
- Minimum requirements:
  - A Service Principal
  - A Fabric Capacity



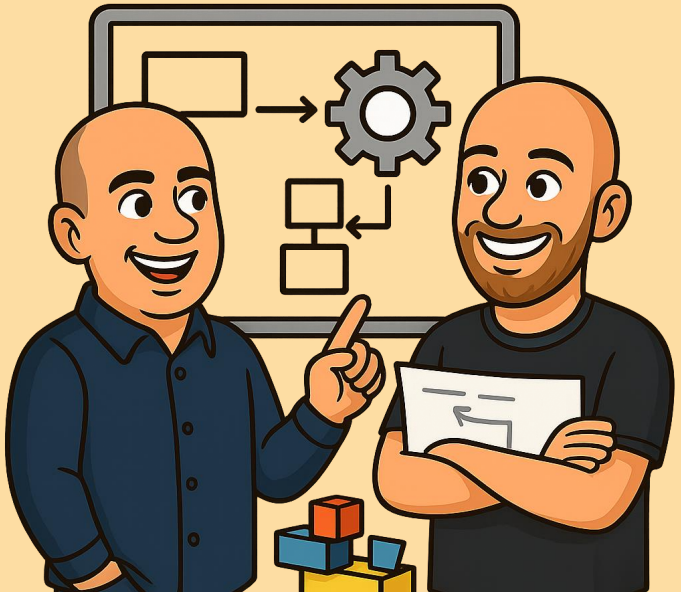
Github

Download source code: https://github.com/gronnerup/FabricAutomation



DEMO

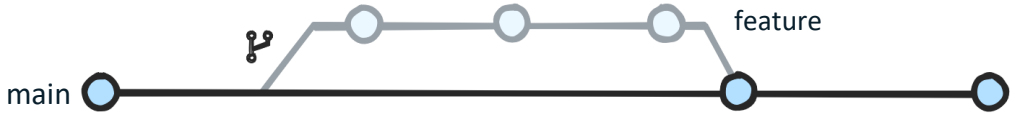
WAYS-OF-WORKING



BRANCH FOR REAL-WORLD COLLABORATION

Branching Isn't Just Git Hygiene - It's a Collaboration Strategy!

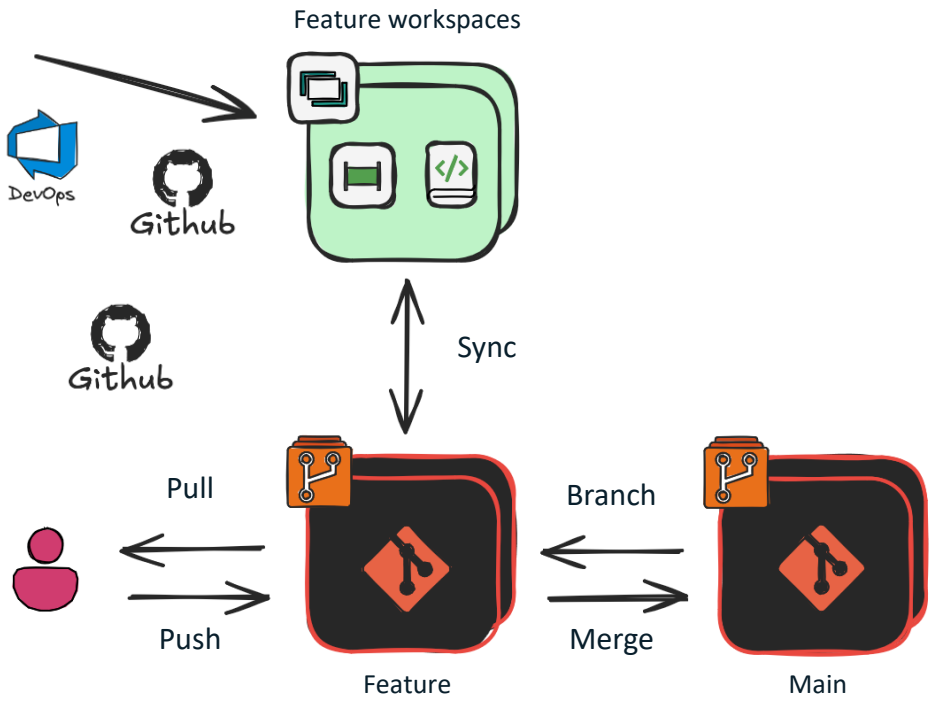
- Git is the source of truth, and branching is how we manage change
- Branches gives developers a seperate workspace for their code and…
  - Isolates development
  - Protects the mainline of our code
  - Enables parallel development
  - Help organize and structure releases
  - Is crucial for streamlined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



THE BRANCHING FLOW

Development flow - Supported by automation!

1. Create branch from main
2. Develop
3. Create PR
4. Merge feature to main
5. Delete feature branch

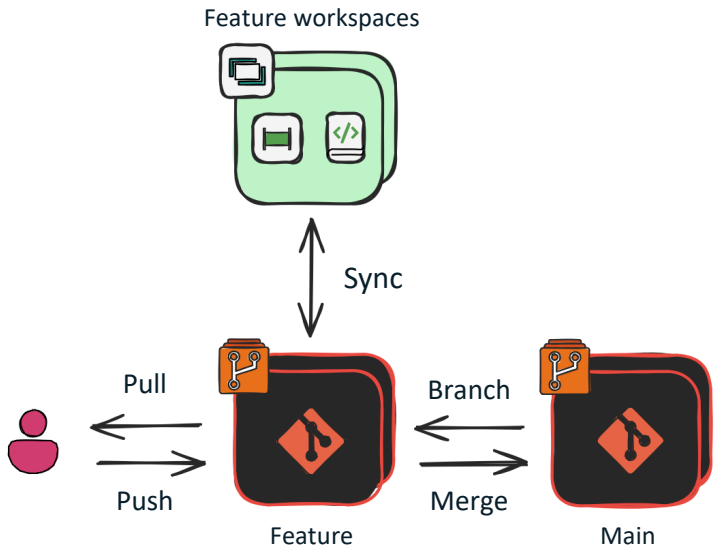


https://learn.microsoft.com/en-us/fabric/cicd/best-practices-cicd

THE BRANCHING FLOW – AUTOMATED

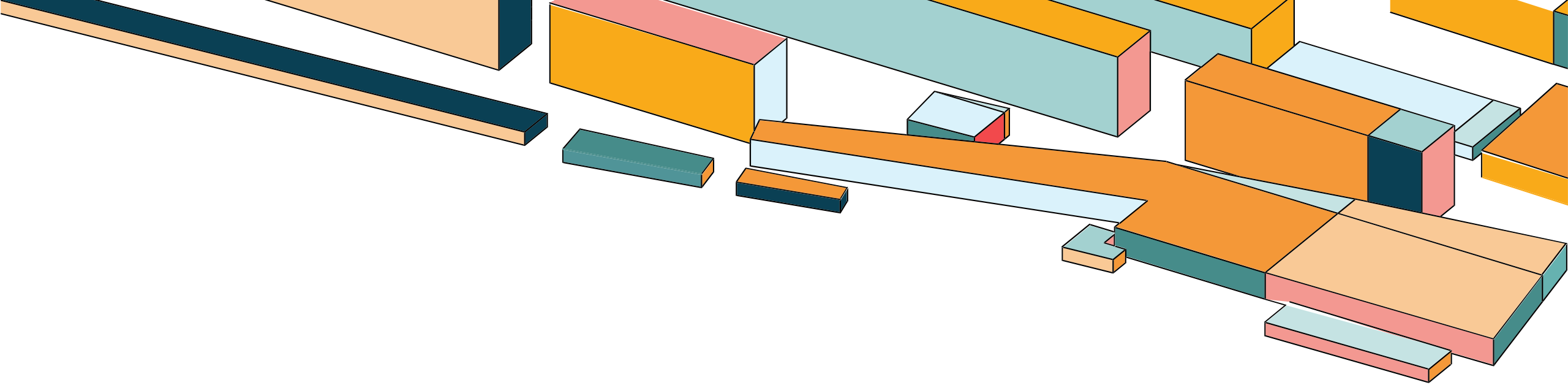
Why automate this?

ClickOps	Automated
Manual process	Fully automated - just create the branch
1 workspace – 1 branch	Support for multiple workspaces
No transfer/setup of: - ACL - Spark settings - Private Endpoints - WS Identity	Customize: - ACL - Spark settings - Private Endpoints - WS Identity - Capacity
Inherits source capacity	Supports automated sync and cleanup
Requires manual cleanup	



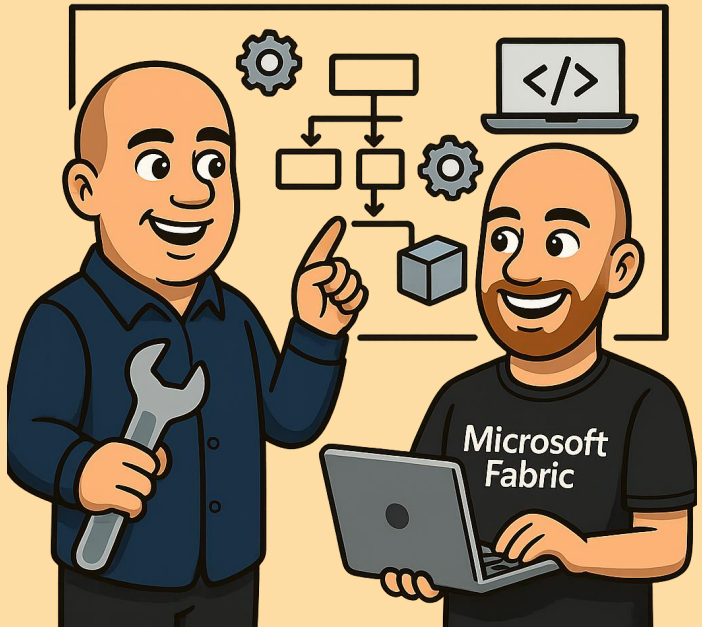
https://peerinsights.hashnode.dev/automating-feature-workspace-maintainance-in-microsoft-fabric

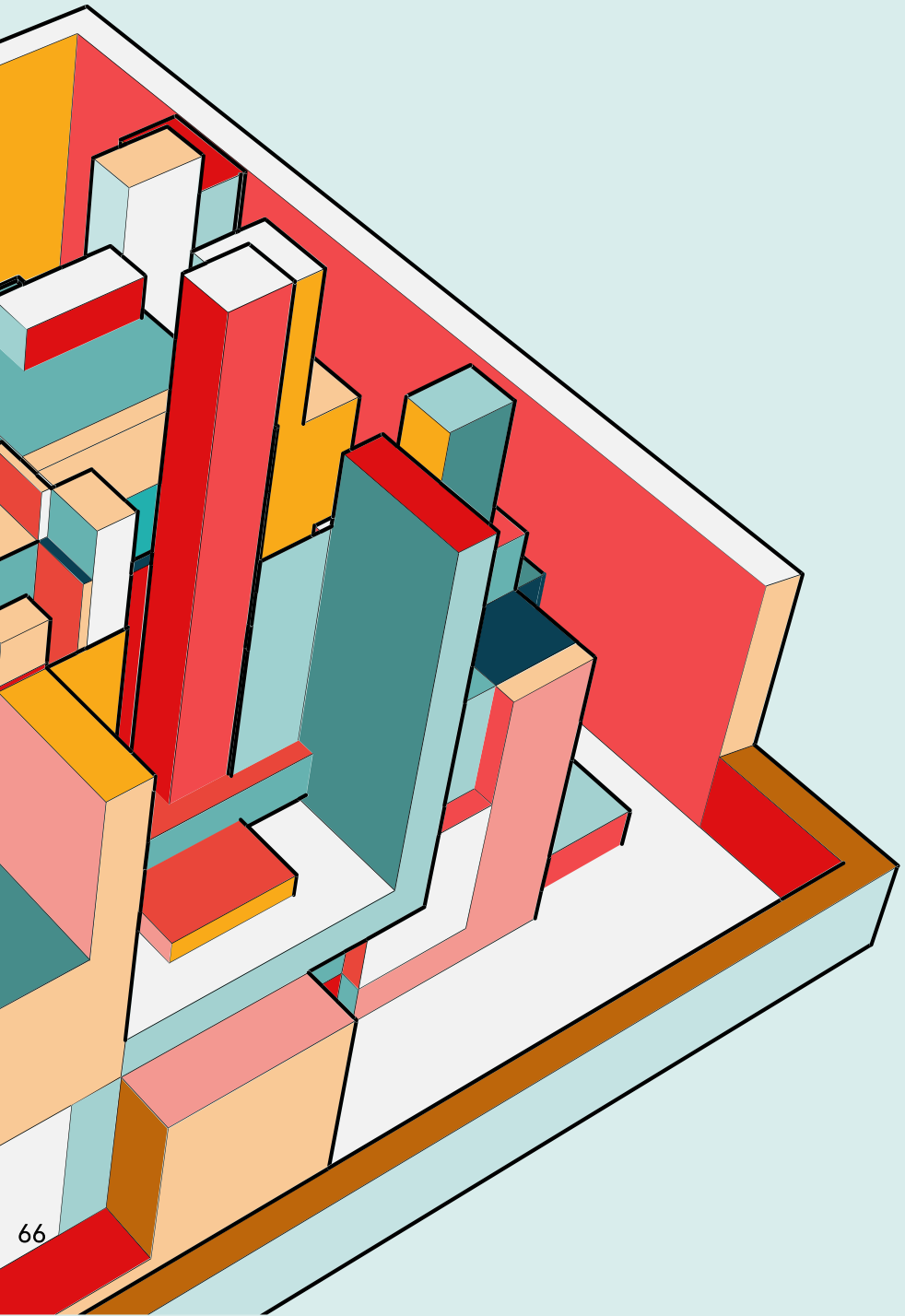
https://justb.dk/blog/2025/02/fabric-spark-notebooks-and-cu-consumption/



DEMO

BUILDING OUR SOLUTION





FABRIC METADATA DRIVEN FRAMEWORK

FABRIC METADATA DRIVEN FRAMEWORK

- Meets most of your needs
- Full control over design and features.
- Less development effort.
- Flexibility and extensibility.
- Lower upfront costs.

FABRIC METADATA DRIVEN FRAMEWORK

- Efficient data management is a cornerstone of modern organizations, and leveraging the right tools can make all the difference.
- The Fabric Metadata -Driven Framework is a cutting-edge solution designed to optimize data handling and utilization.
- This innovative framework harnesses the powerful capabilities of the Fabric SQL Database to build a robust, scalable, and flexible Metadata -driven architecture.

KEY FEATURES

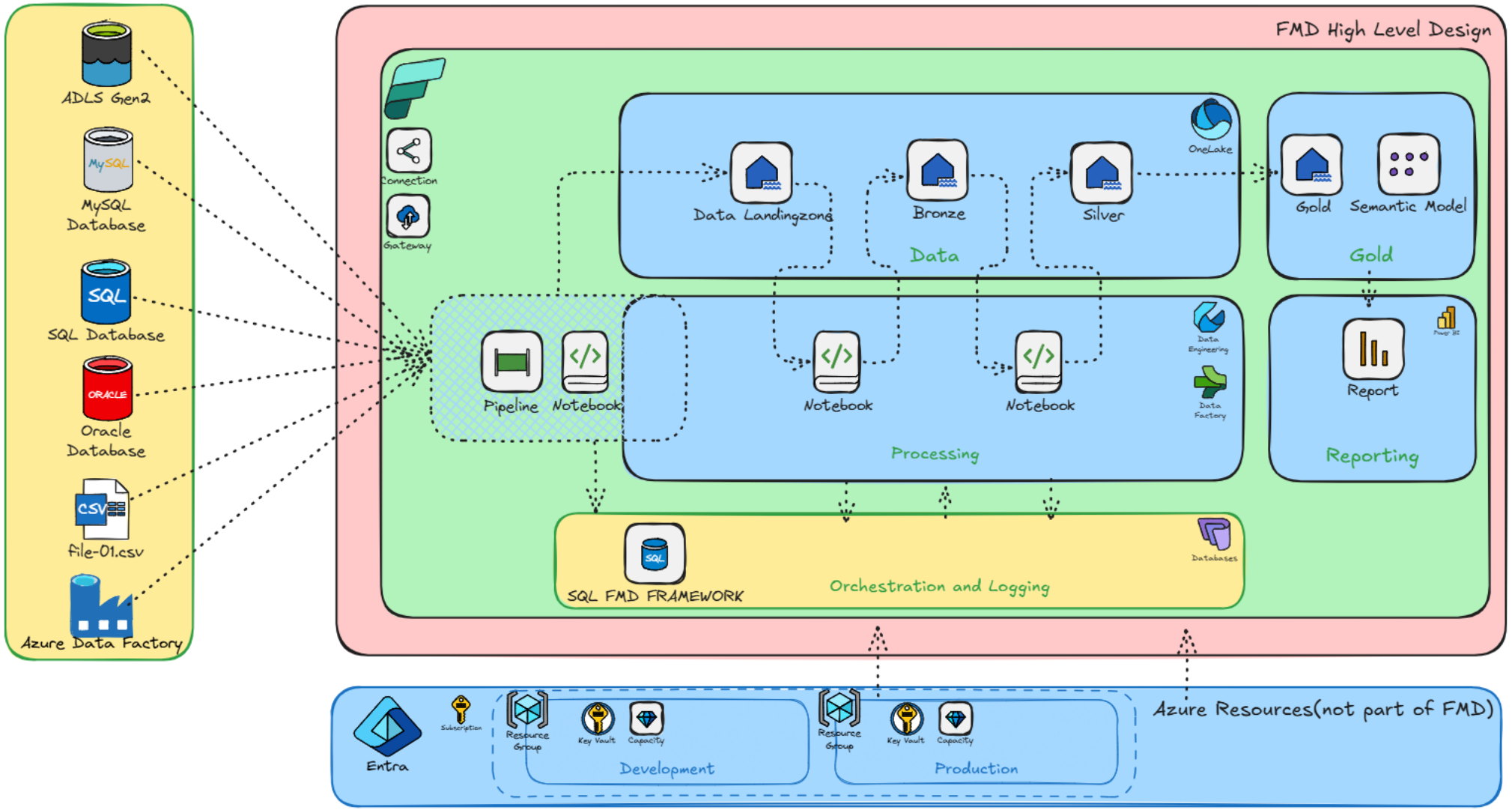
Scalability and Flexibility

Cost Efficiency

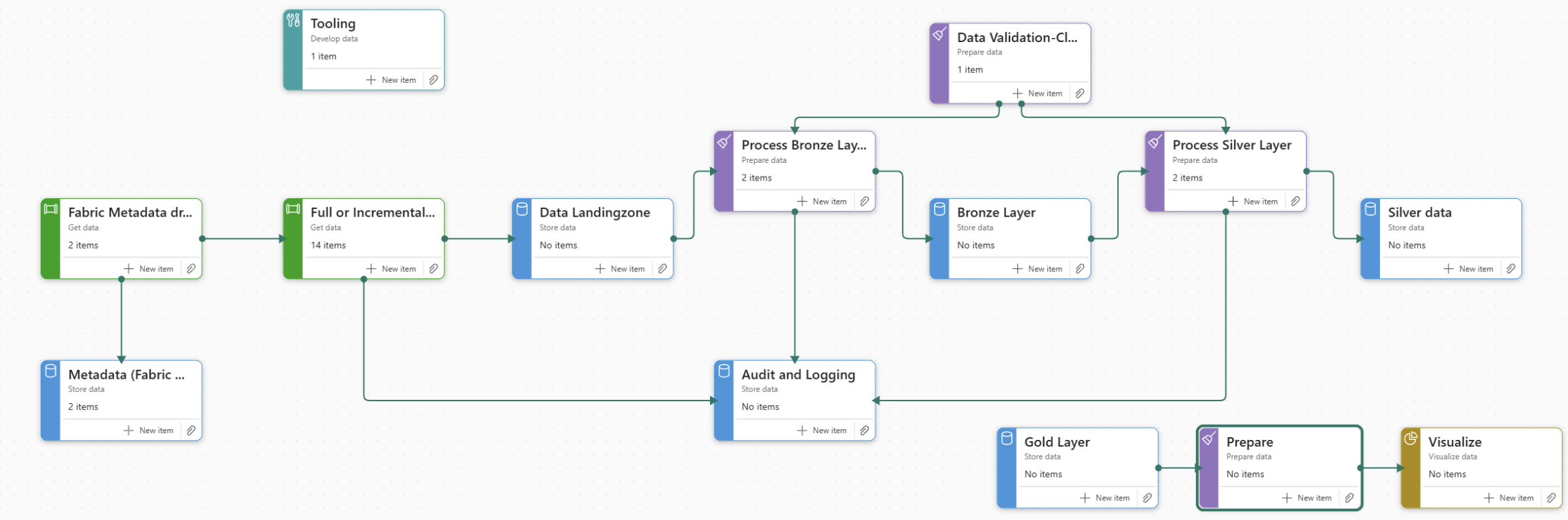
Streamlined Data Integration

Enhanced Data Governance

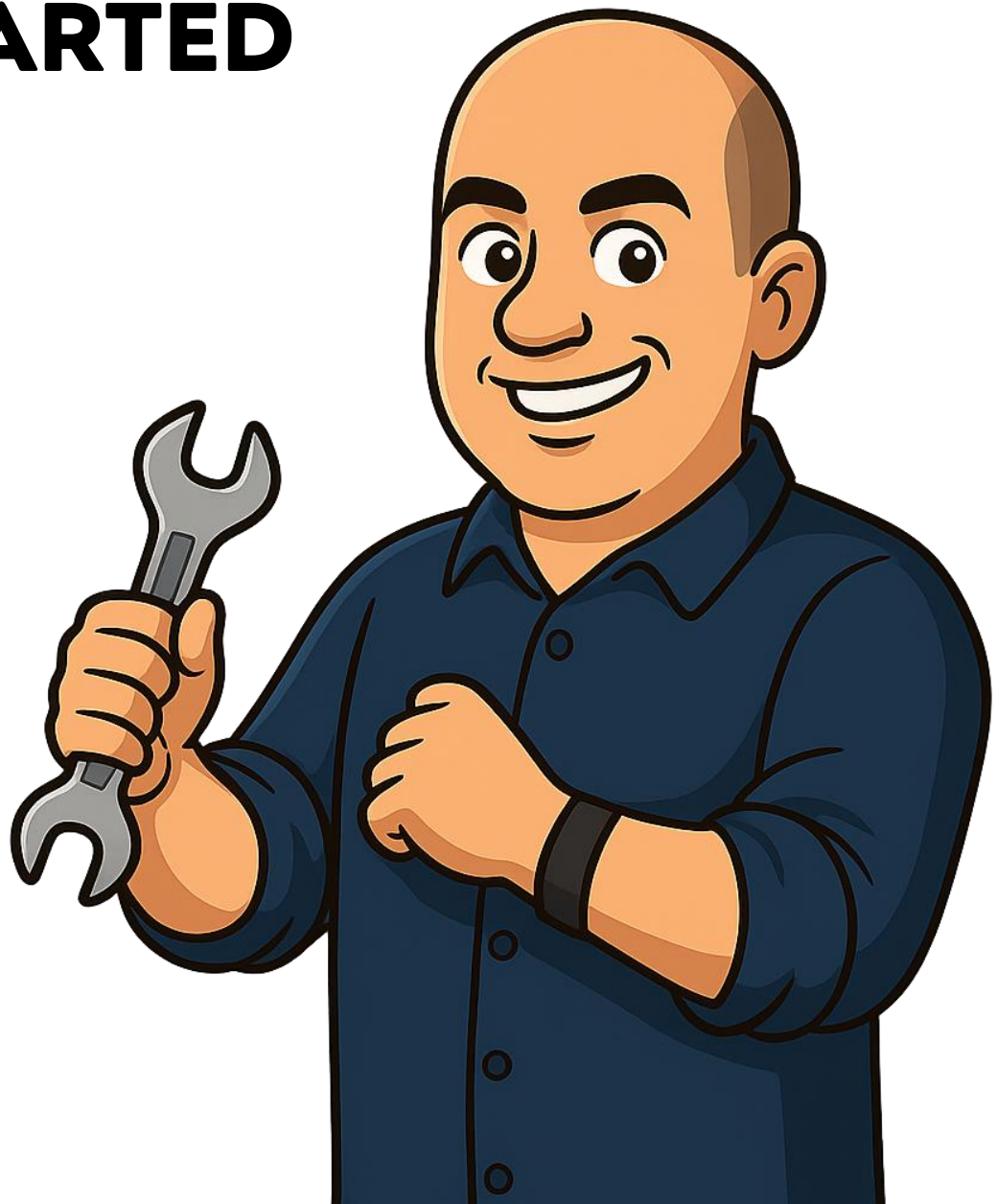
FABRIC METADATA DRIVEN FRAMEWORK

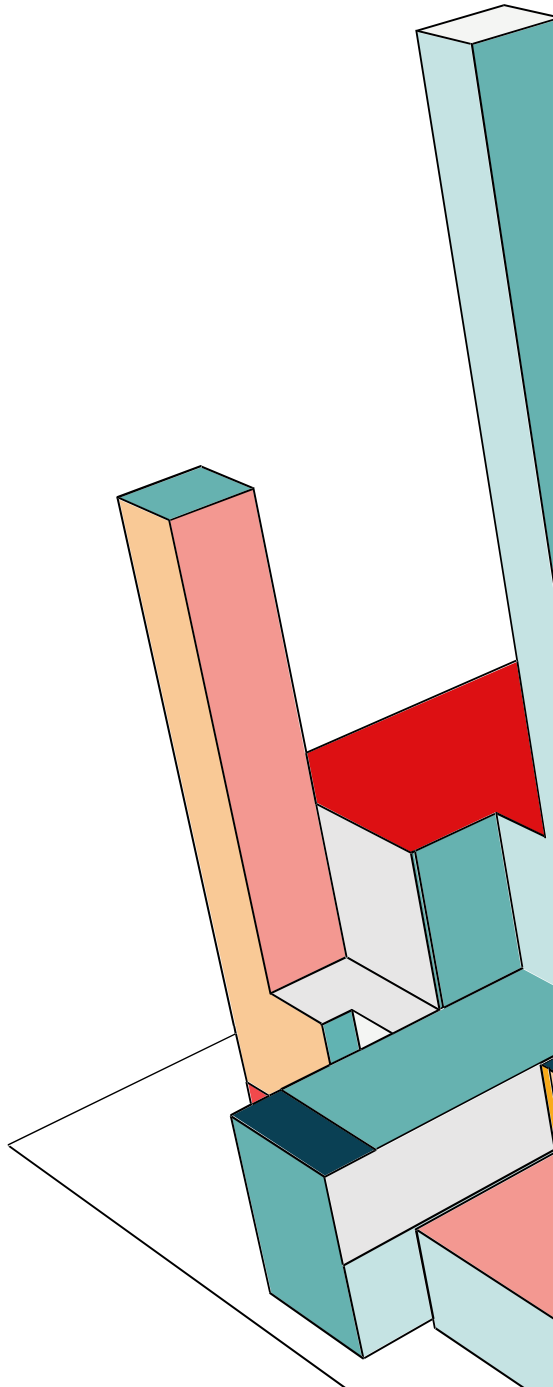


METADATA-DRIVEN DATA INGESTION FRAMEWORK COMPONENTS



LET'S GET STARTED

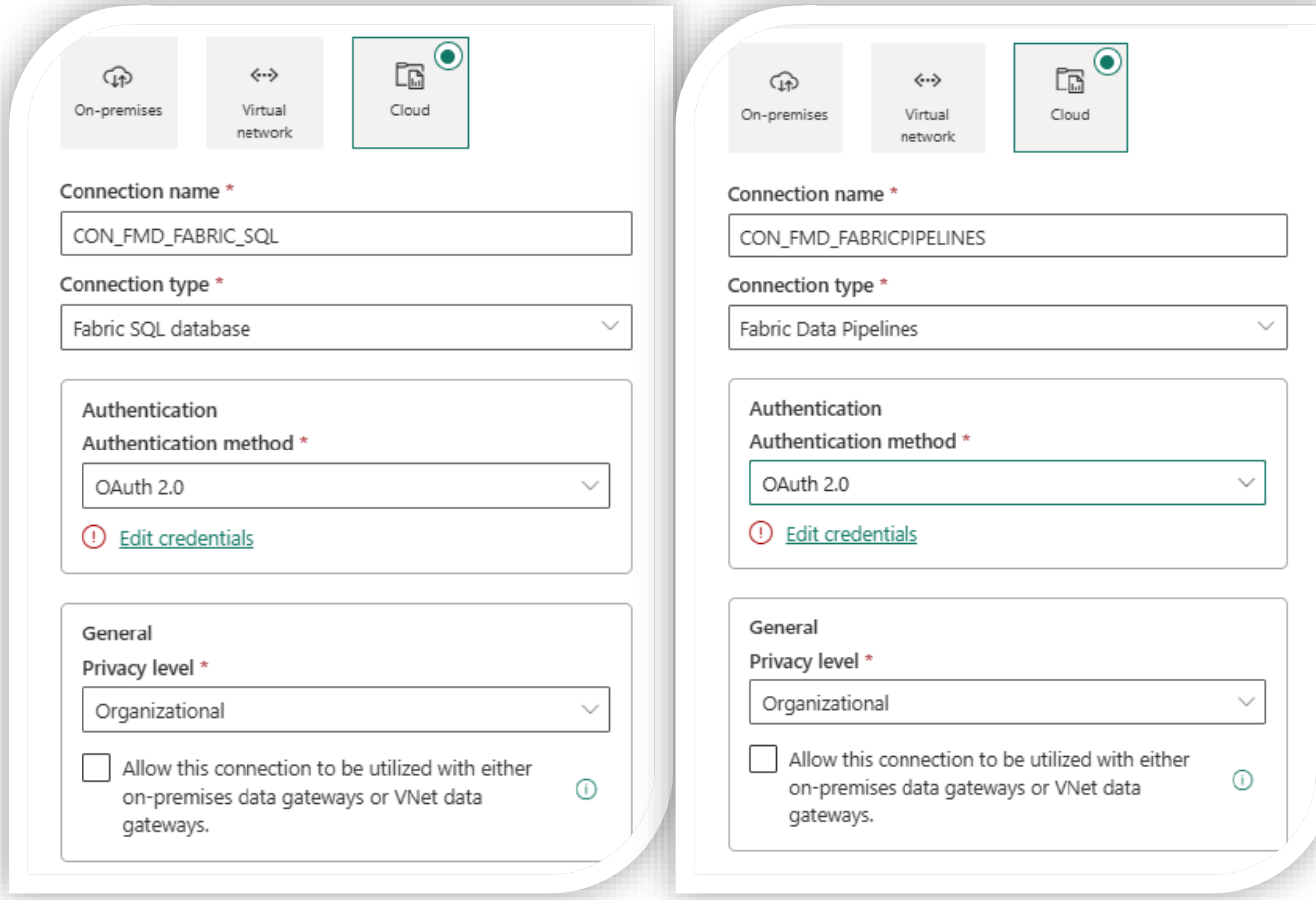




CONNECTIONS (REQUIRED)

- Required Connections
  - Fabric Data Pipelines
    - Authentication Oauth
      - Workspace Identity is Coming
      - Service Principal is also an option
  - Fabric SQL database
    - Authentication Oauth
      - Service Principal is also an option

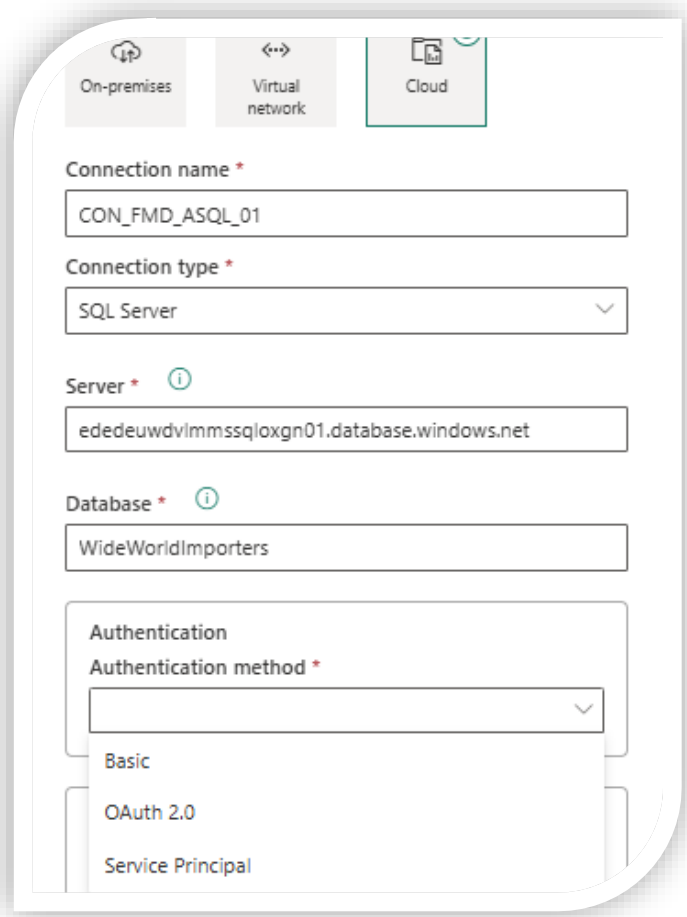
All other connections are based on your configuration



SUPPORTED SOURCES (CURRENTLY)

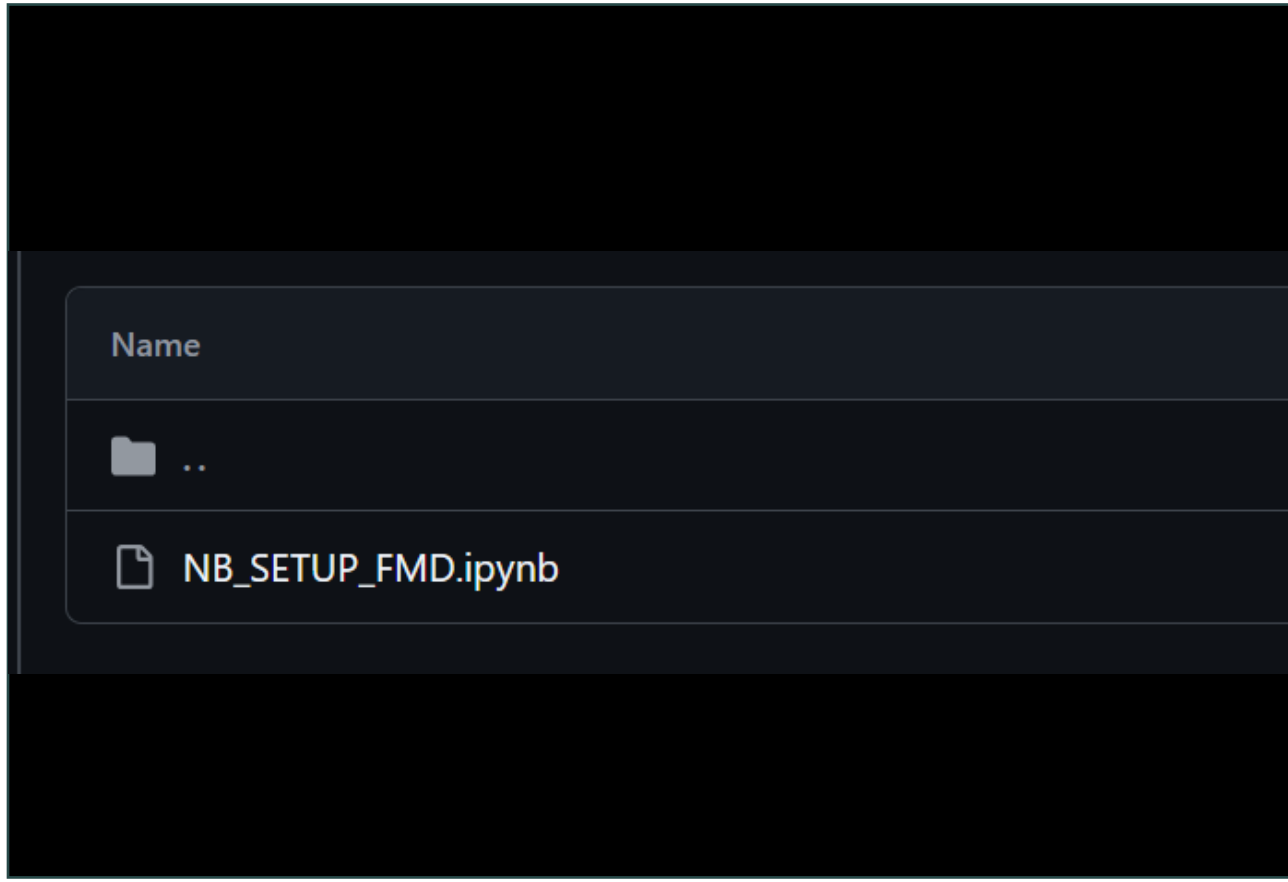
- SQL Server
- Data Lake Gen 2
- Onelake Tables
- Onelake Files
- FTP / SFTP
- ADF (Connect to ADF pipeline)

- All other connections can easily be added
    - Every connection needs a new Data Pipeline



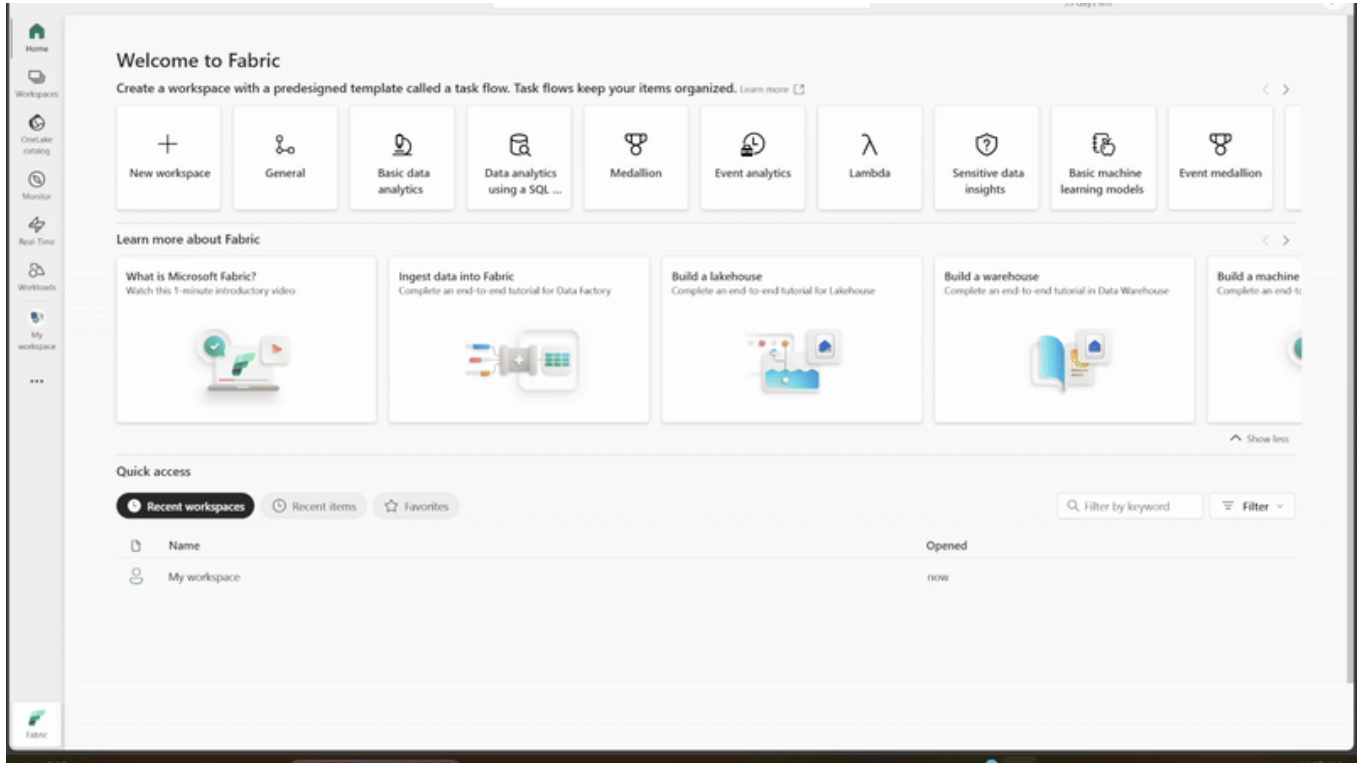
FABRIC METADATA DRIVEN FRAMEWORK

- Download Notebook from Github



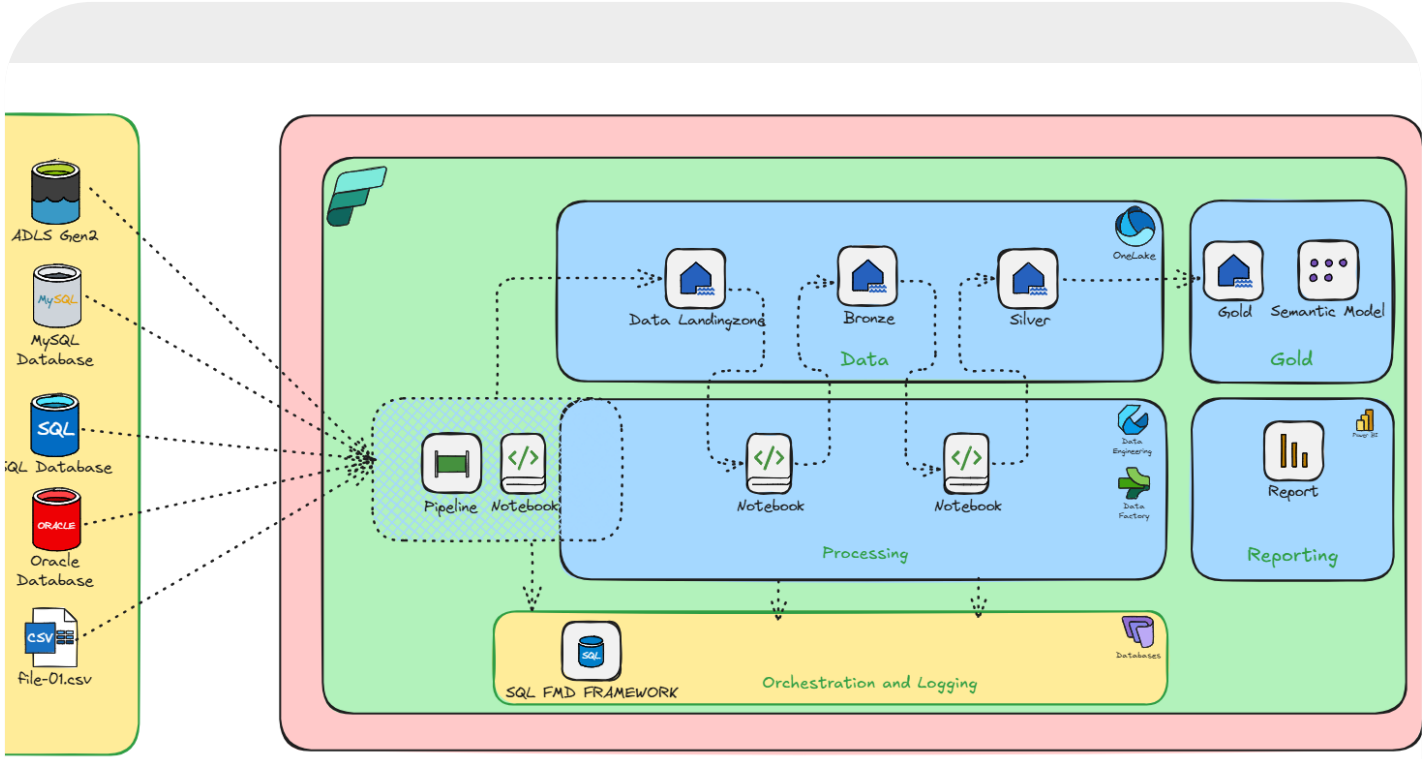
FABRIC METADATA DRIVEN FRAMEWORK

- Download Notebook from Github
- Create Workspace Name is up to you:
  - FMD_XXX_CONFIGURATION
- Assign a capacity
- Import Notebook to this Workspace



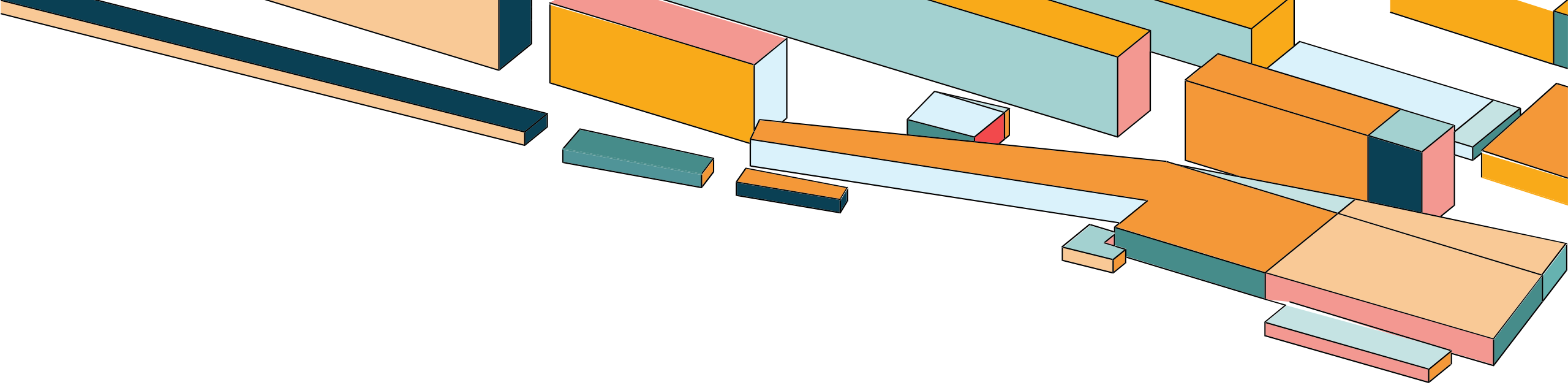
FABRIC METADATA DRIVEN FRAMEWORK

- Open Notebook
- NB_SETUP



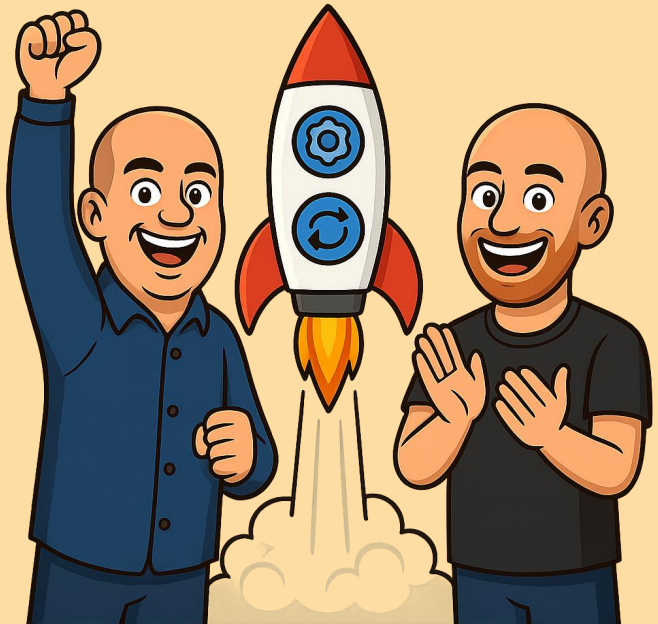

```
%pip install ms-fabric-cli --quiet
```

```
import subprocess
import os
from time import sleep, time
import json
from zipfile import ZipFile
```



DEMO

DEPLOYMENT



RELEASE PROCESS – THE MAIN OPTIONS…

Fabric Deployment pipelines 	Git based deployment 	Git based deployment using build environment 

• No code experience
• For simpler solutions
• No support for:
  • Pre-deployment opr.
  • Post-deployment opr.
  • Test & validation
	• Suitable when using Gitflow
	• Each environment connected to git branch
	• No need for building environments
	• Might require post-deployment operations
		• Supports building enviroments
		• Deployment through ADO Pipelines/Github actions
		• Supports manipulation of item definitions

CI/CD BUILDING BLOCKS

Tools & automation options

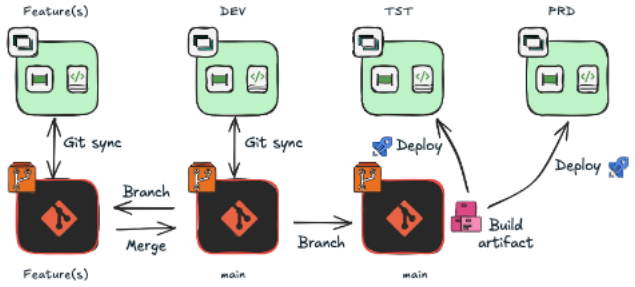
- Azure DevOps Pipelines
- Github actions
- Python scripts & wrappers
- Fabric CLI wrappers

fabric-cicd Python Library

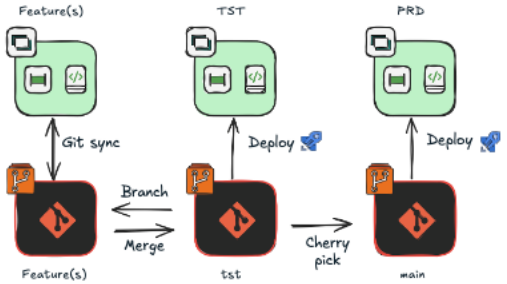
- Developer friendly
  - Code-first approach
  - No need to call Fabric APIs directly
- Environment supports
  - Parameterization
  - Reusable for locale, DevOps and more…
- Smart deployment
  - Deploy all supported items (with public APIs)
  - Auto-unpublish orphaned artifacts

INTRODUCING 3 RELEASE SCENARIOS

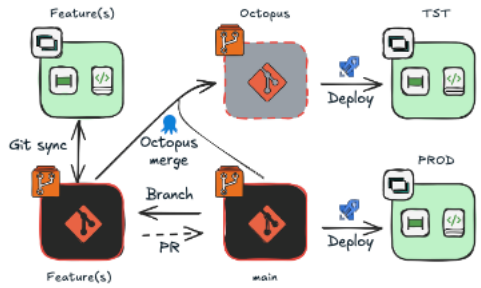
Scenario 1
The simple way



Scenario 2
Pick and choose

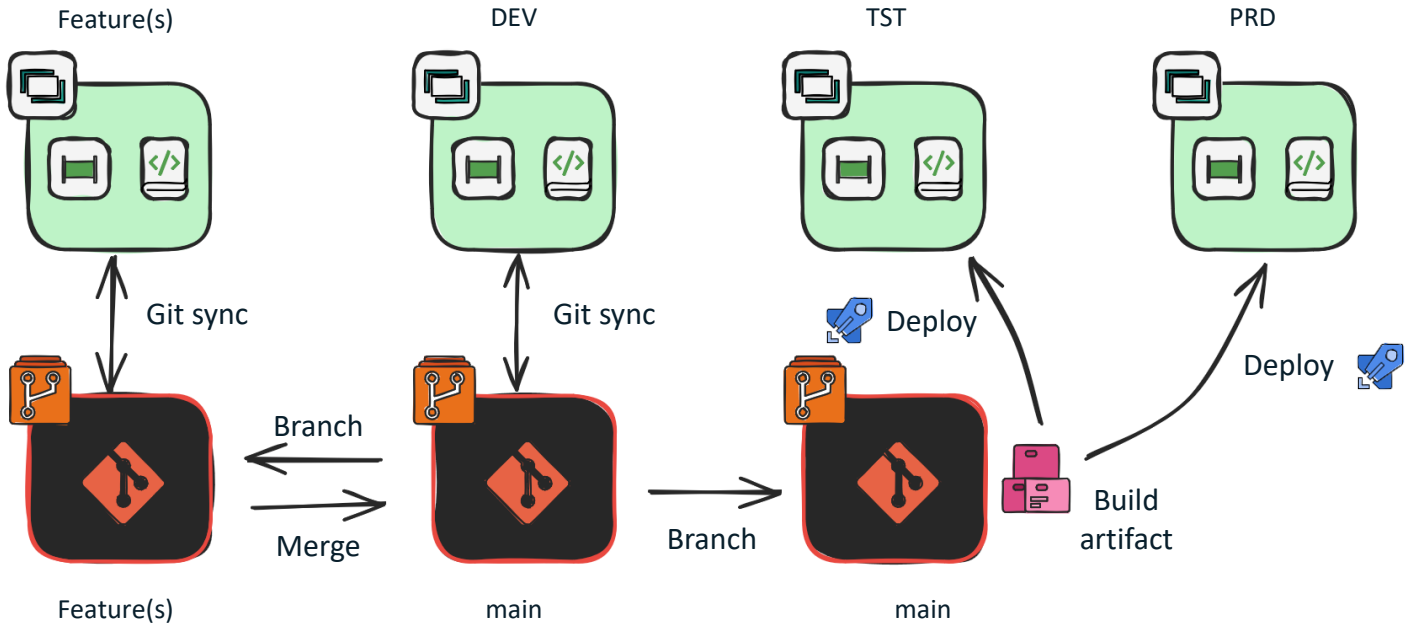


Scenario 3
The selective release option



RELEASE PROCESS – 3 SELECTED SCENARIOS

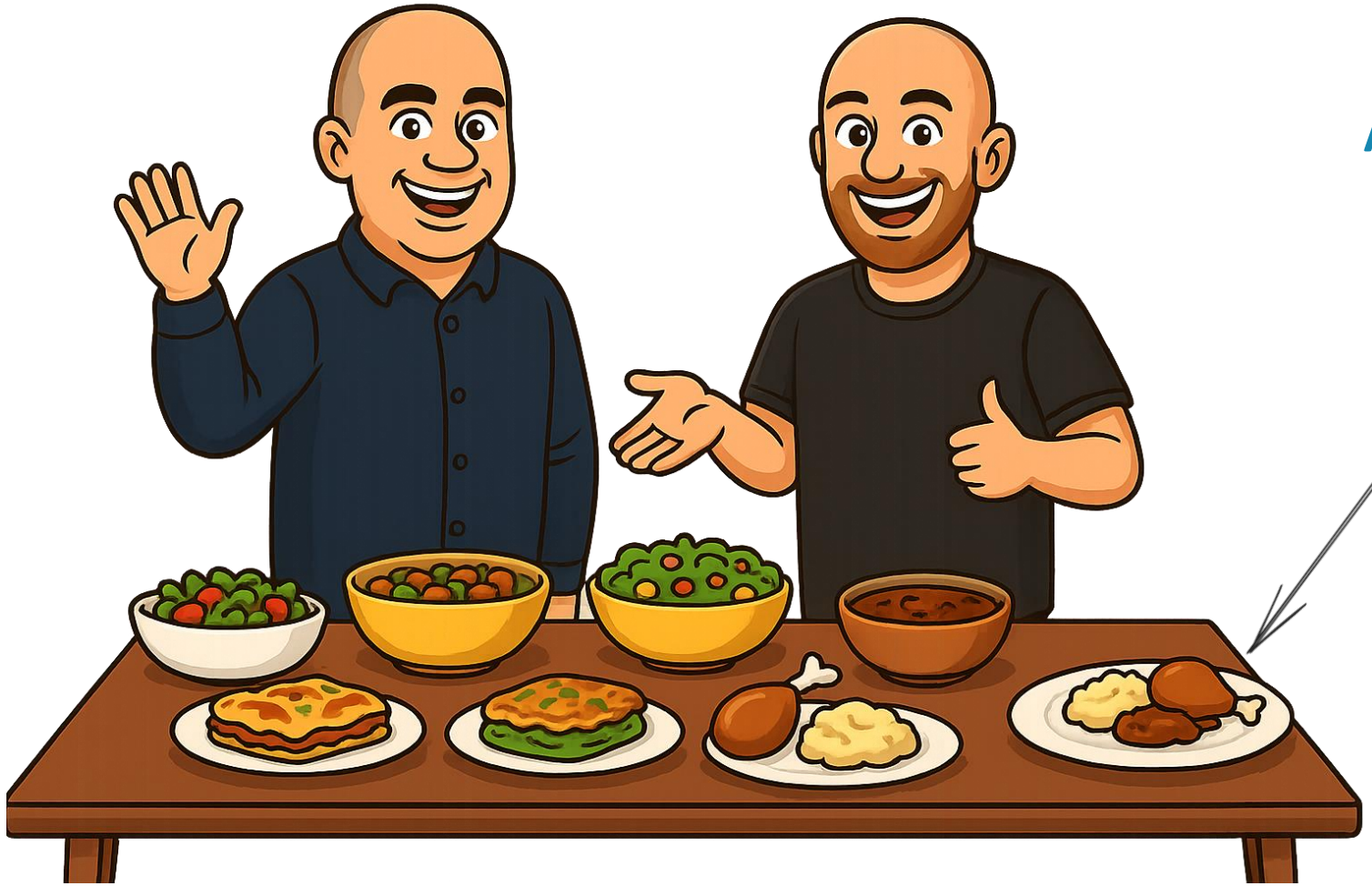
Scenario 1 – Main is always dev



Suitable for:

- Smaller and more simple solitions
- Small teams
- Low or no interdependency between features

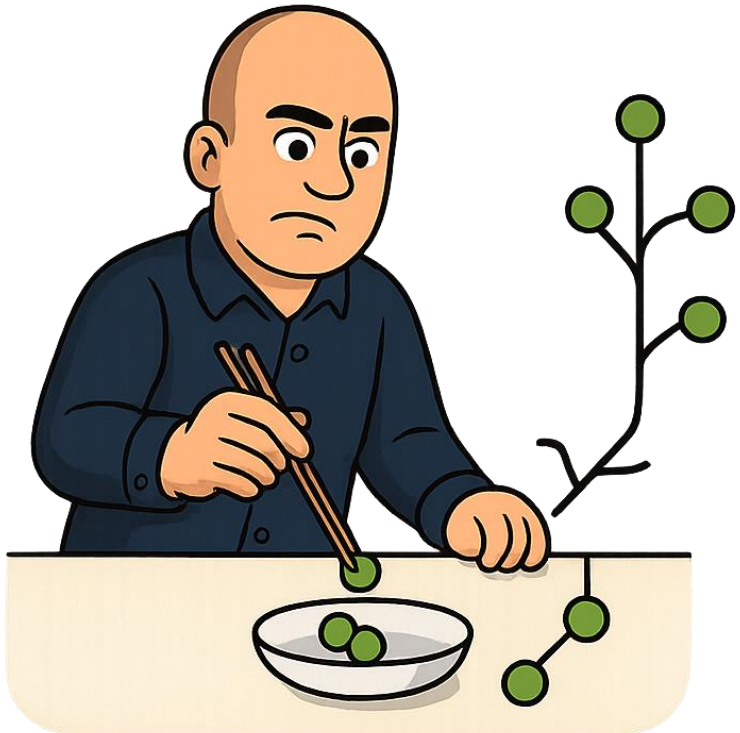
SO WHY INTRODUCE ALTERNATIVES?



APPROVED!

Let’s bring this to the table

RELEASE PROCESS – 3 SELECTED SCENARIOS

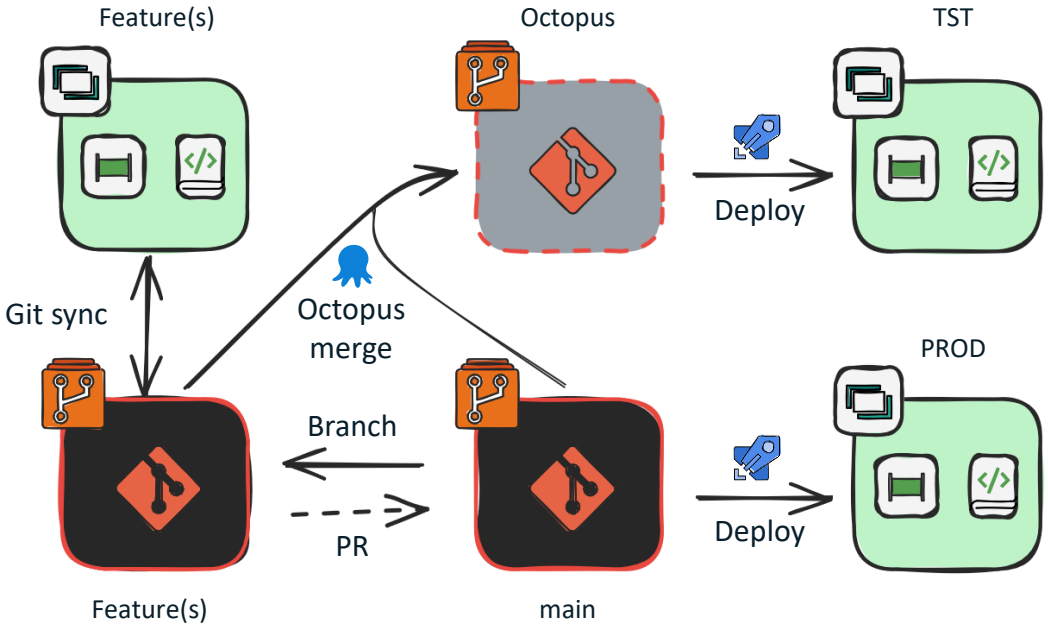


Cherry-pick into main

Octopus merge

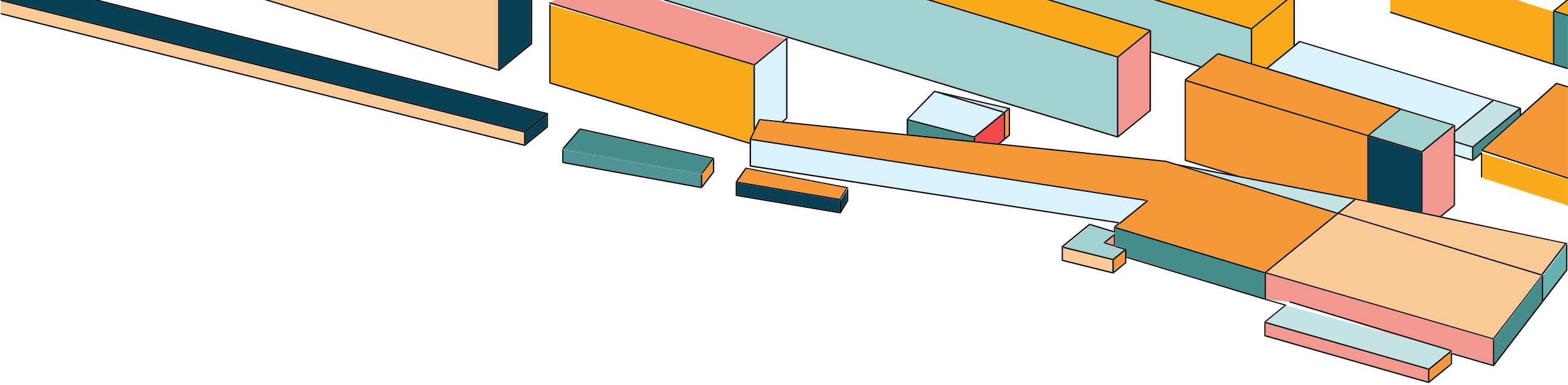
RELEASE PROCESS – 3 SELECTED SCENARIOS

Scenario 3 – Octopus merge



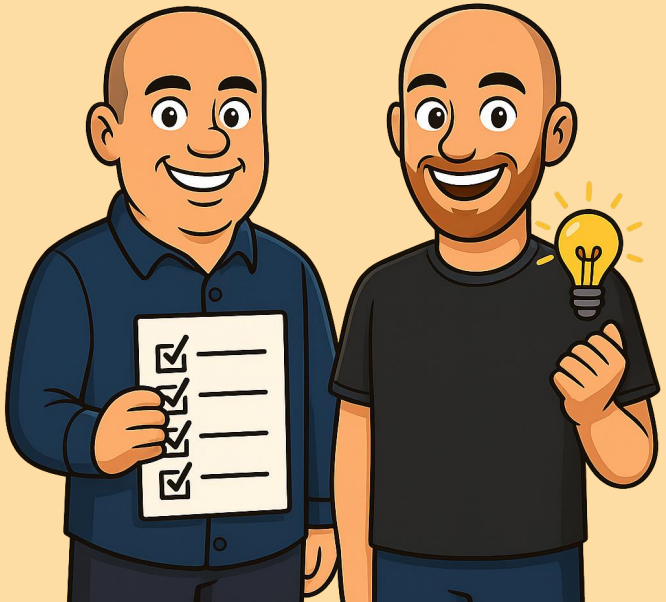
Suitable for:

- Large teams
- Many parallel features
- Interdependent changes
- High-risk changes
- Supports per-feature validation
- Supports incremental and selective testing and deployment



DEMO

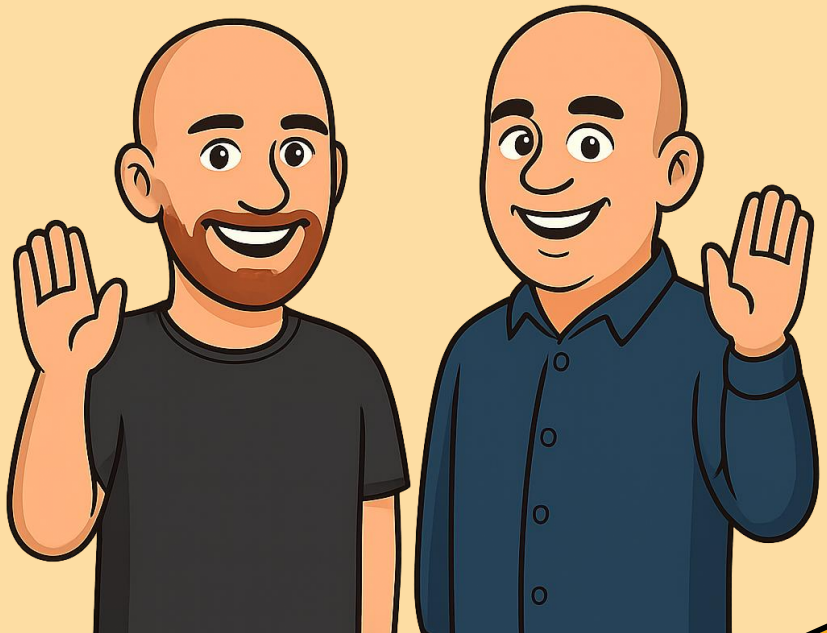
BEST PRACTICES, TIPS AND TRICKS



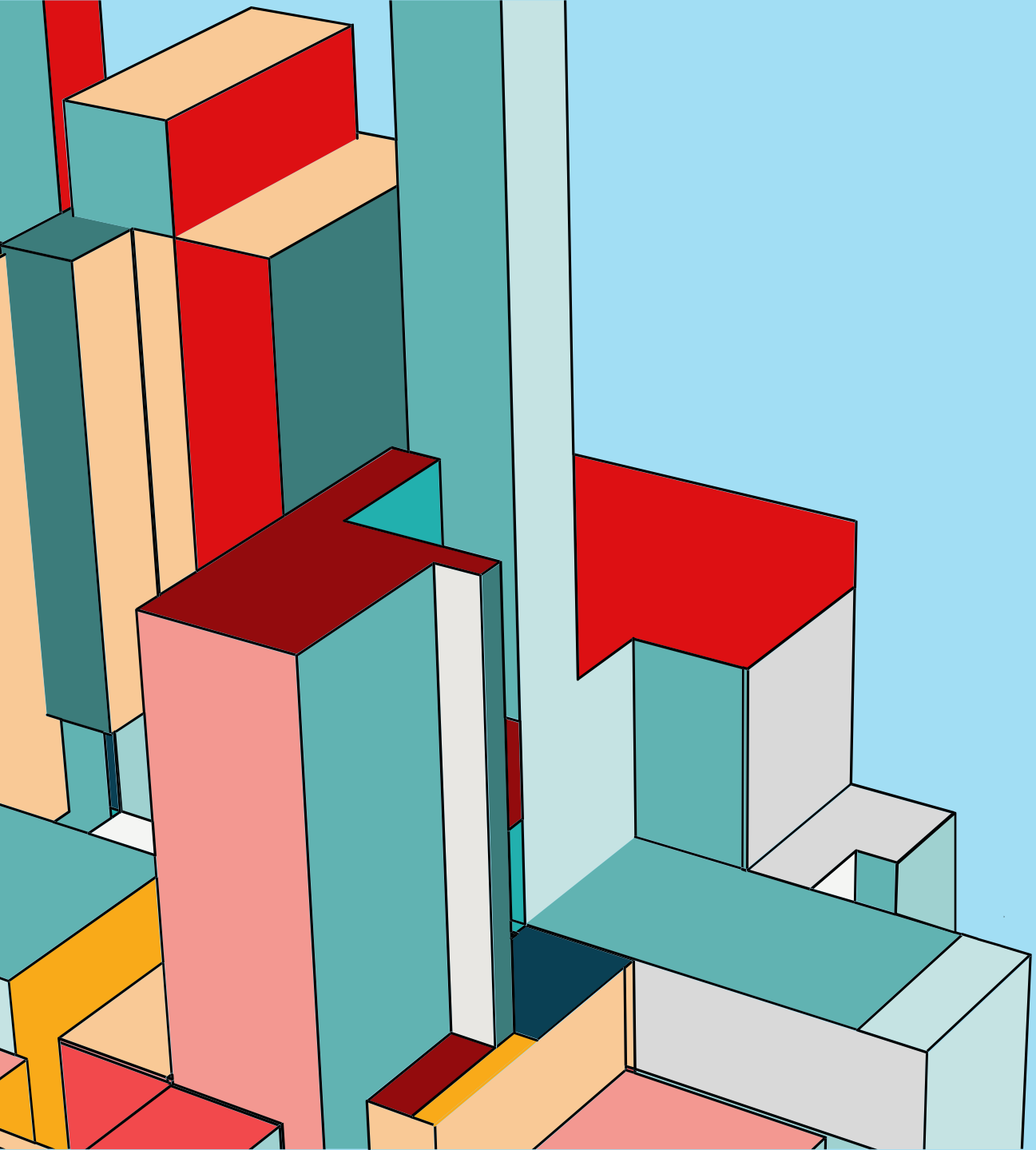
BEST PRACTICES, TIPS AND TRICKS

- Split your workloads and layers across multiple workspace
- Use mono-repo structure as a starting point
- 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
- Think BIG – start small and get experience
- Stay curious - Get inspired by what others build!

THANK YOU



Rate This session



END OF SLIDEDECK

FABRIC CI/CD PLATFORM

Continuous Integration

Small but frequent changes

Working with multiple people on various aspects of the solution

Merging back into main branch

Continuous Deployment

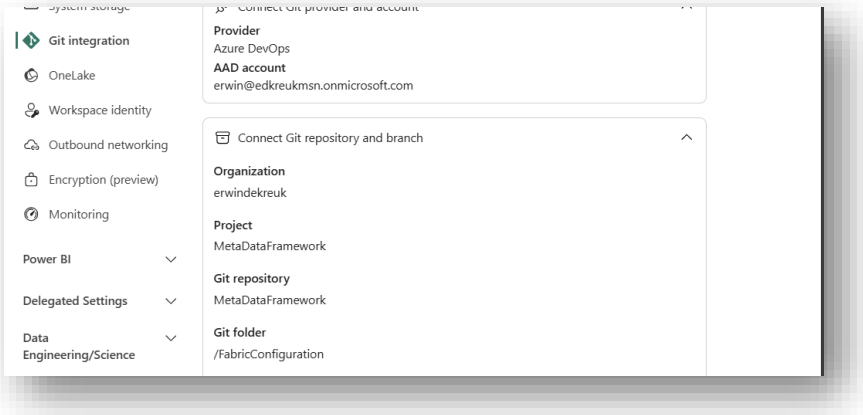
Working in short cycles (often Agile)

High frequency updates and releases

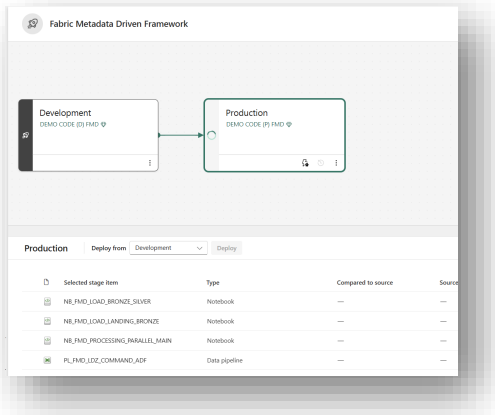
Repeatable deployment process

FABRIC CI/CD PLATFORM

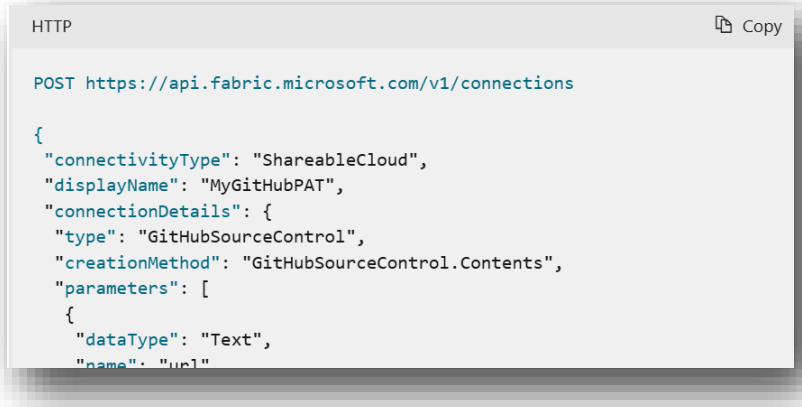
Built-in git integration



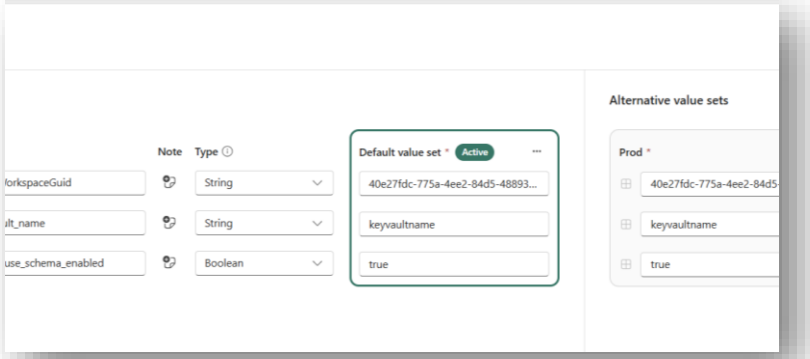
Deployment pipelines



Fabric REST APIs

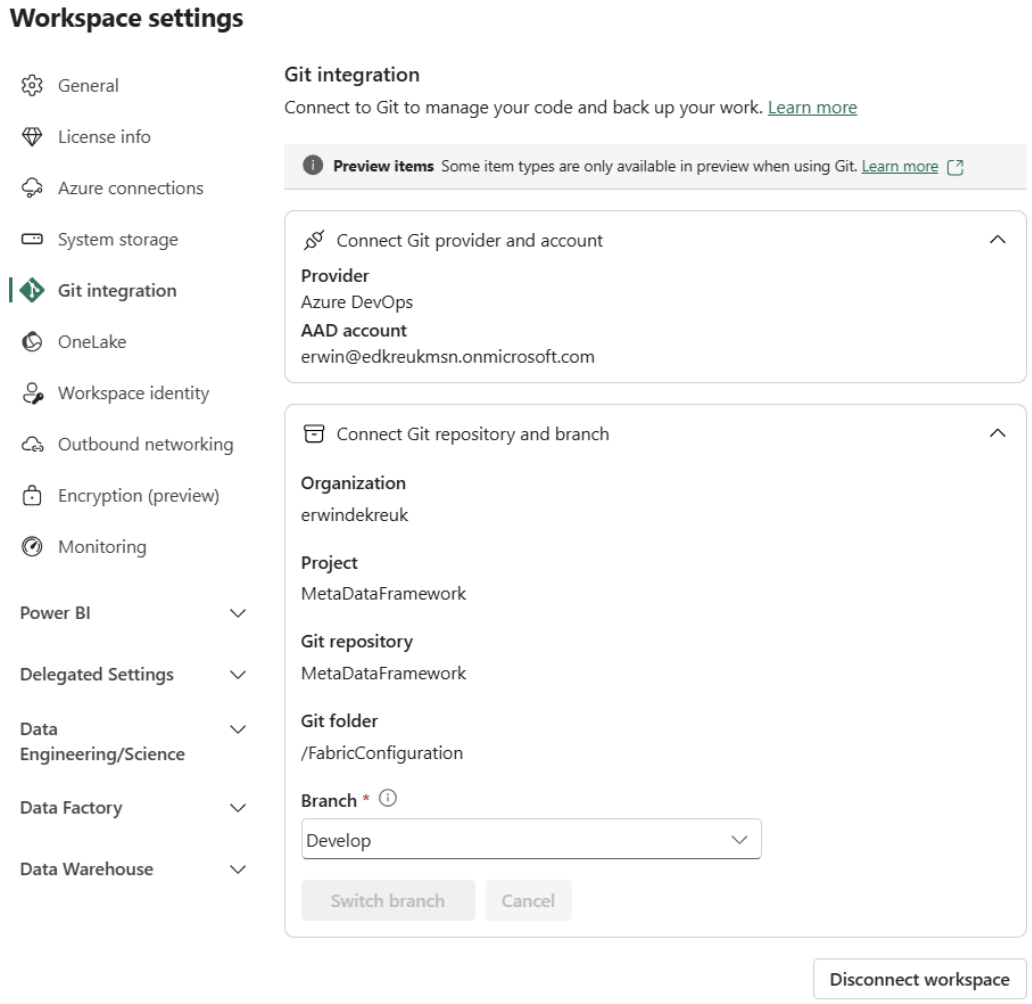


Variable Library



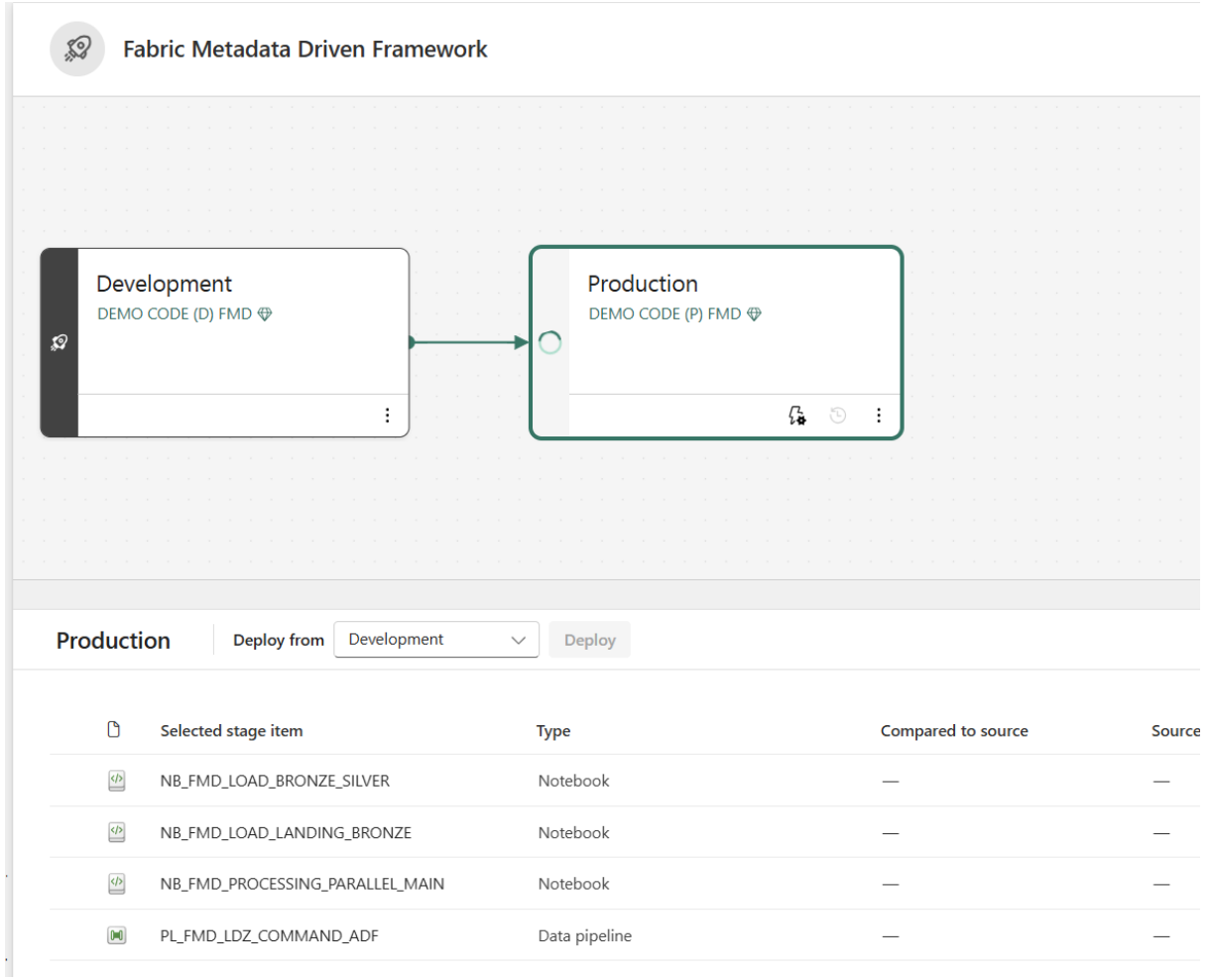
GIT INTEGRATION

- Sync a Workspace to a Git branch
- Git providers
  - Azure DevOps
  - GitHub
  - GitHub Enterprise
- Fabric git APIs – REST APIs & PowerShell samples.
- Fabric CLI
- Manage branches
  - Switch branch
  - Checkout new branch
  - Branch out to new workspace



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



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

Data Engineering items:
Environment
GraphQL (preview)
Lakehouse (preview)
Notebooks
Spark Job Definitions (preview)
User Data Functions (preview)

Data Factory items:
Copy Job (preview)
Dataflow gen2
Data pipeline
Mirrored database
Mount ADF (preview)
Variable library (preview)

Real-time Intelligence items:
Activator (preview)
Eventhouse
EventStream
KQL database
KQL Queryset
Real-time Dashboard

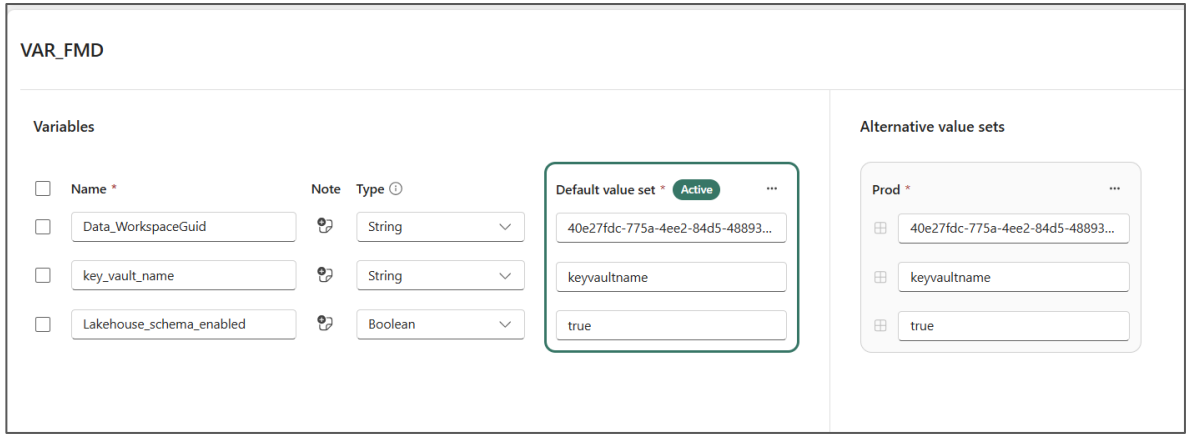
Data Warehouse items:
Warehouse (preview)
Mirrored Azure Databricks Catalog (preview)

Database items:
SQL database (preview)

Industry solutions:
Healthcare (preview)
HealthCare Cohort (preview)

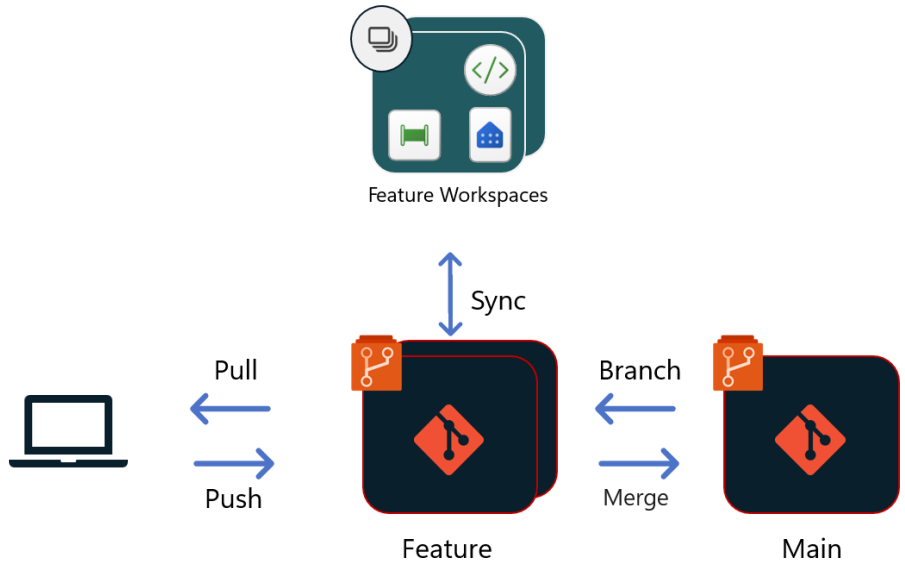
VARIABLE LIBRARY

- New item in Fabric
- Central location to manage configurations across items in a workspace.
- Value sets –
  - Set values for different stages of your CI/CD process.
  - Apply them in relevant WSs with a single API call/ click of a button.



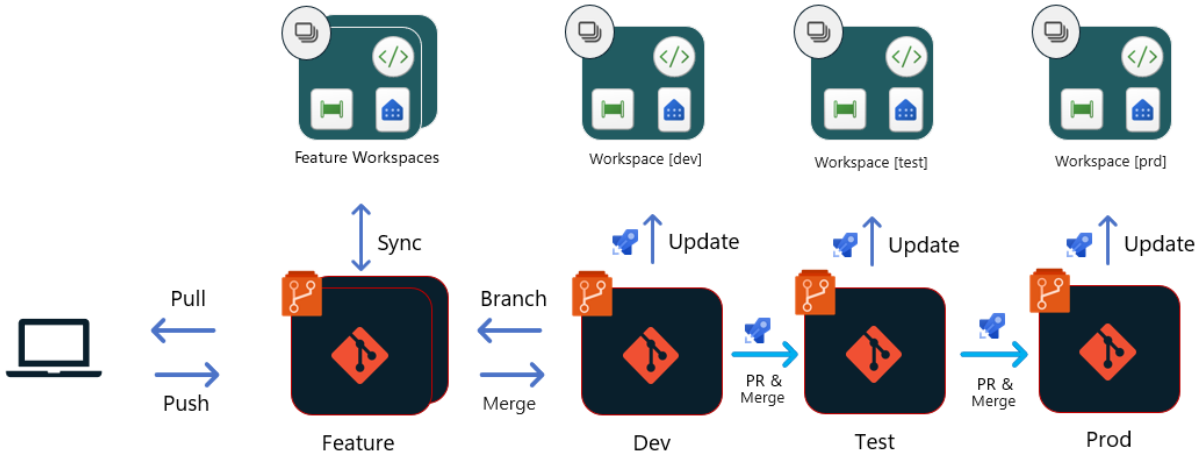
DEVELOPMENT PROCESS

- Development process is consistent across all deployment strategies.
- Deployment method is independent of how updates are released to production.
- Developers work in isolated environments when using source control.
- In Fabric, the isolated environment can be:
  - An IDE on your local machine (e.g., Power BI Desktop, VS Code).
  - A separate workspace within Fabric..



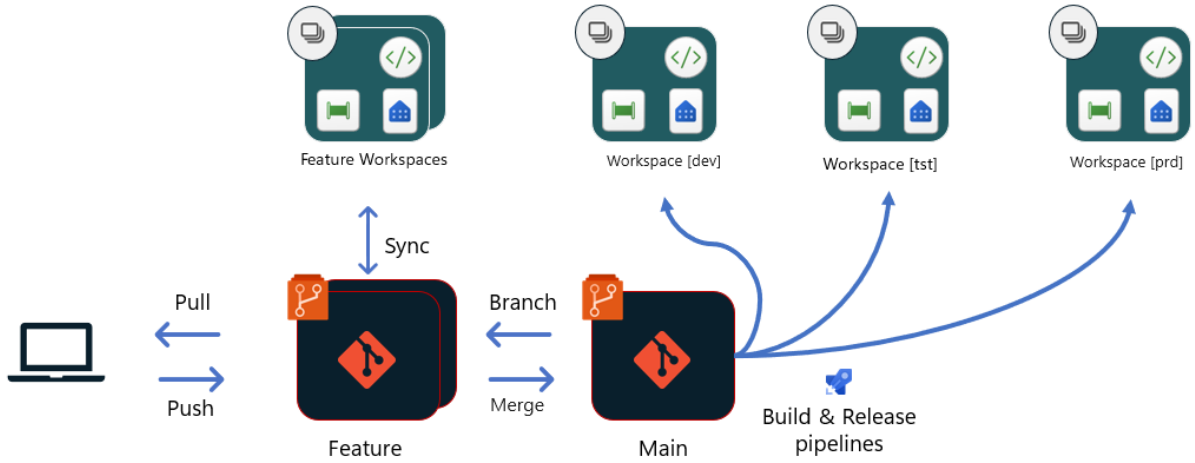
GIT- BASED DEPLOYMENTS

- Git repository as single source of truth for all deployments.
- Gitflow branching strategy used, supporting multiple primary branches.
- Direct upload from Git repo to workspace, bypassing build environments.
- Post-deployment customization possible via API calls or workspace actions.



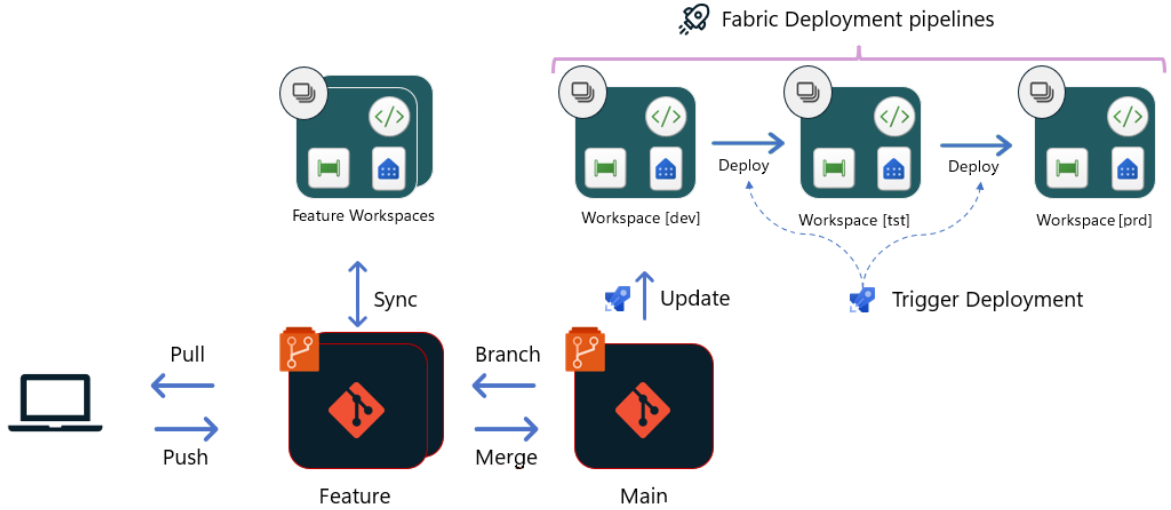
GIT- BASED DEPLOYMENTS USING BUILD ENVIRONMENTS

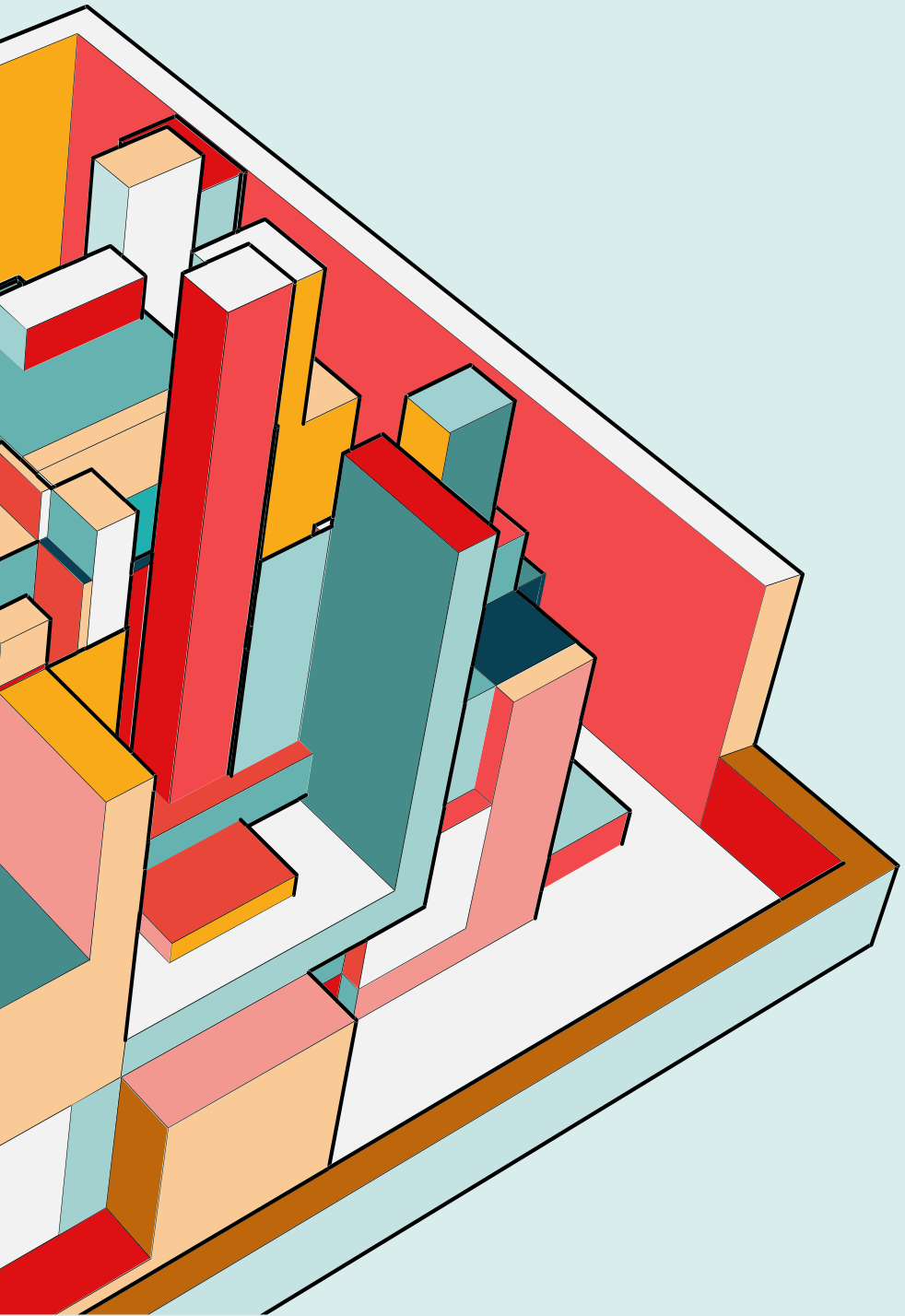
- Git as the single source of truth and origin of all deployments.
- Trunk-based development workflow followed by the team.
- Build environment required with custom scripts to modify workspace-specific attributes (e.g., connectionId, lakehouseId) before deployment.
- Custom release pipeline needed to:
- Retrieve item content from Git.
- Call Fabric Item APIs to create, update, or delete modified Fabric Items.



GIT- BASED DEPLOYMENTS USING BUILD ENVIRONMENTS

- Source control used for development only, not as the origin of deployments.
- Deployments occur directly between stages of the Fabric release pipeline.
- Deployment rules, autobinding, and available APIs are sufficient to manage stage-specific configurations.
- Fabric deployment pipeline features leveraged, including:
- Viewing changes in Fabric.
- Tracking deployment history.
- Linear deployment structure in Fabric pipelines.
- Additional permissions required to create and manage deployment pipelines.





OUR FIRST AUTOMATION CALL

A Live Demo

LET'S CREATE A WORKSPACE WITH 3 LAKEHOUSES

- Click Ops
- Fabric Cli

  Microsoft Fabric CLI | fabric-cli
- Fabric–cicd

  fabric-cicd
- Semantic Link Labs

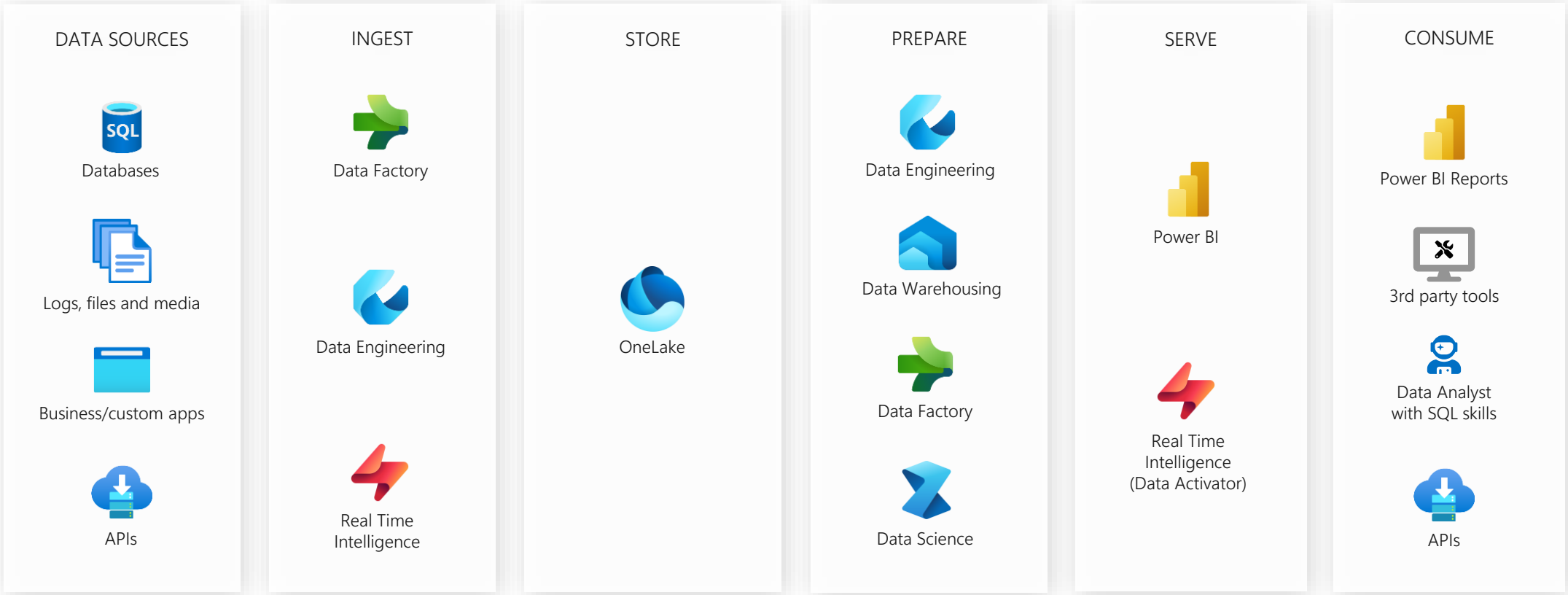
  sempy_labs – semantic-link-labs 0.12.1 documentation
- Rest API

  Item management - Microsoft Fabric REST APIs | Microsoft Learn

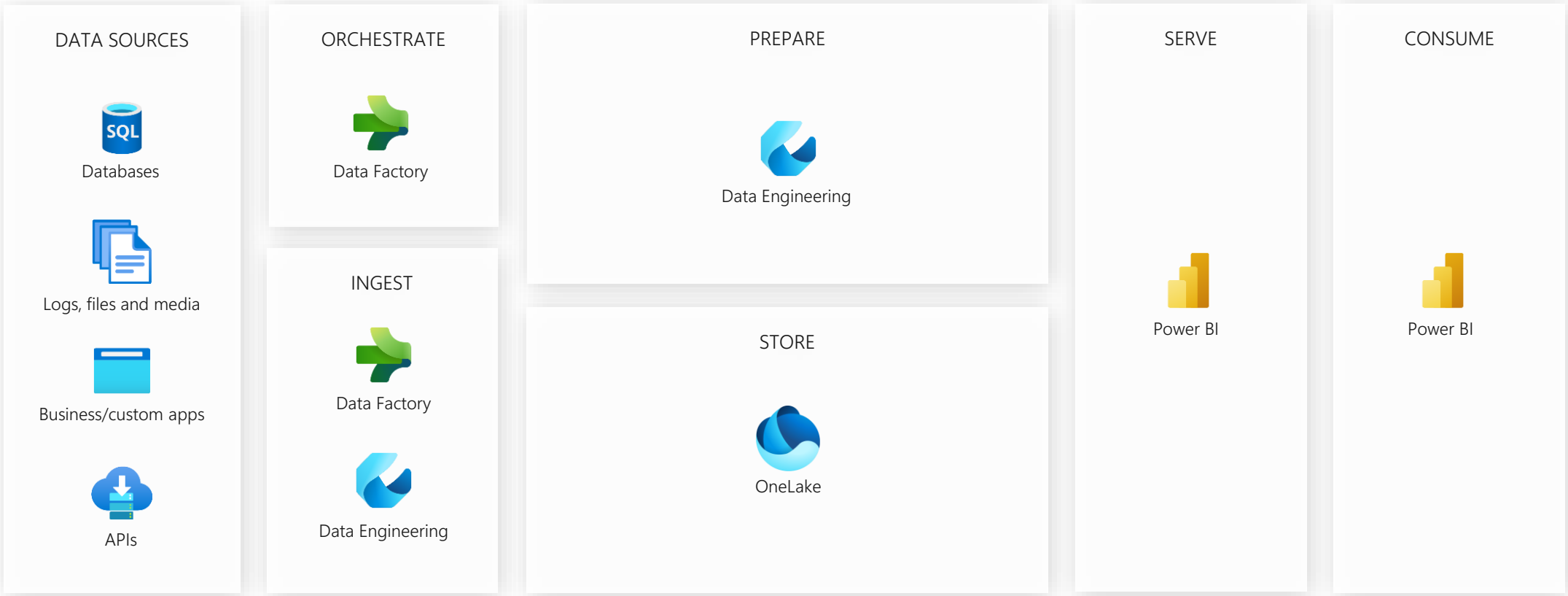
Microsoft Fabric
Community Conference

DESIGNING OUR FABRIC LAKEHOUSE PLATFORM

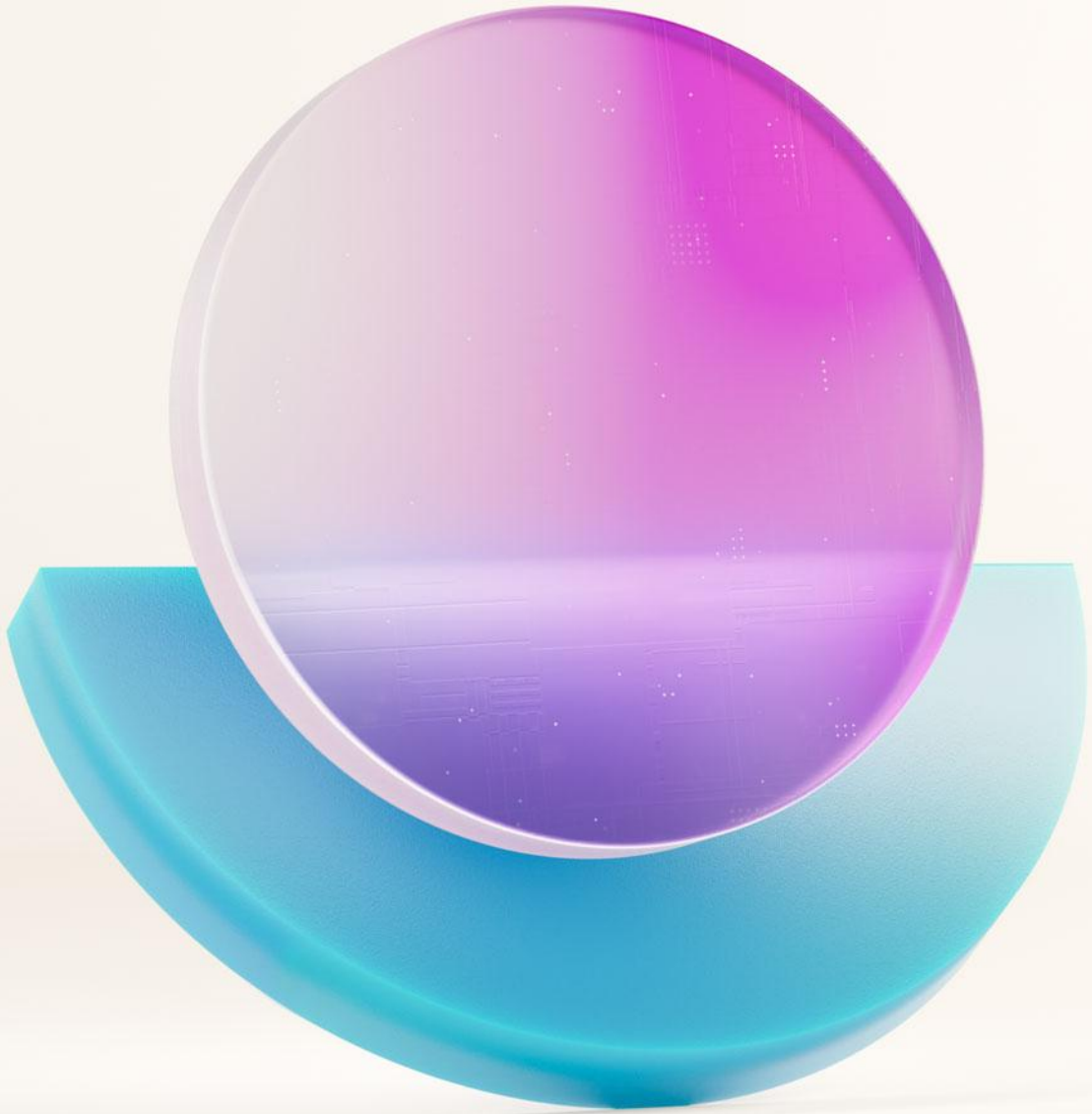
MULTIPLE COMPONENTS = MULTIPLE POSSIBILITIES



A LAKEHOUSE STARTING POINT ARCHITECTURE



Microsoft Fabric
Community Conference

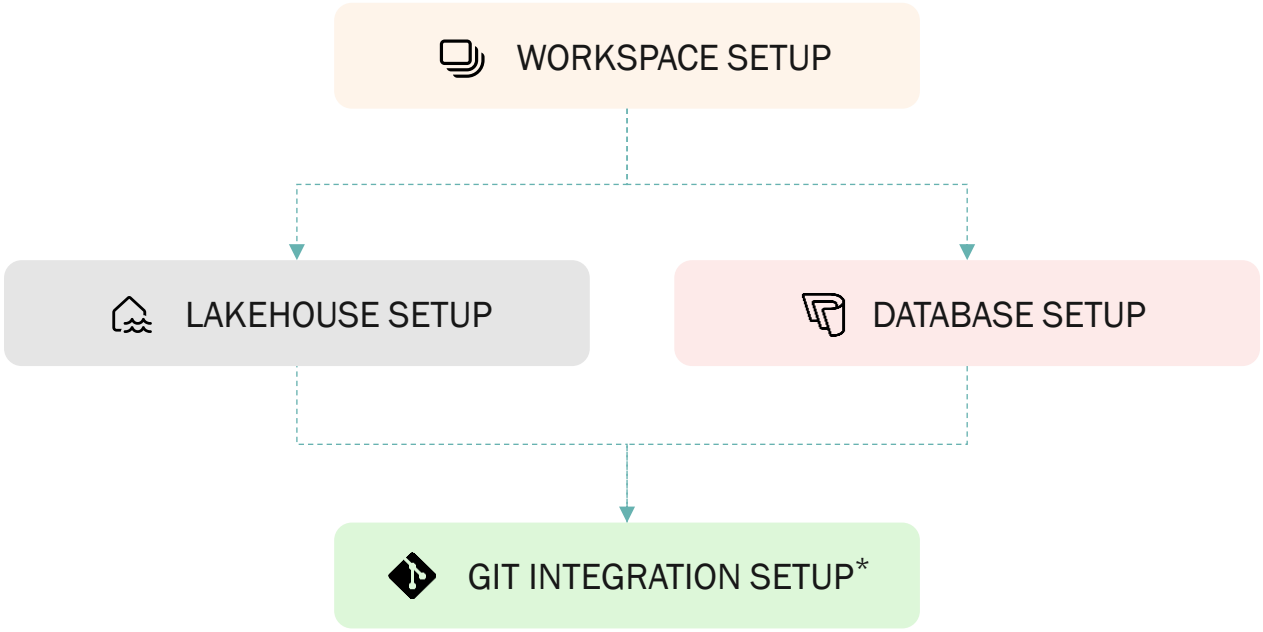


ACCELLERATE AND UNIFY FABRIC SOLUTION SETUP

AUTOMATING FABRIC SOLUTION SETUP - SHOWCASE

Prerequisites:

- A Fabric Capacity
- A Service Principal *
- Azure DevOps project *



Run options:

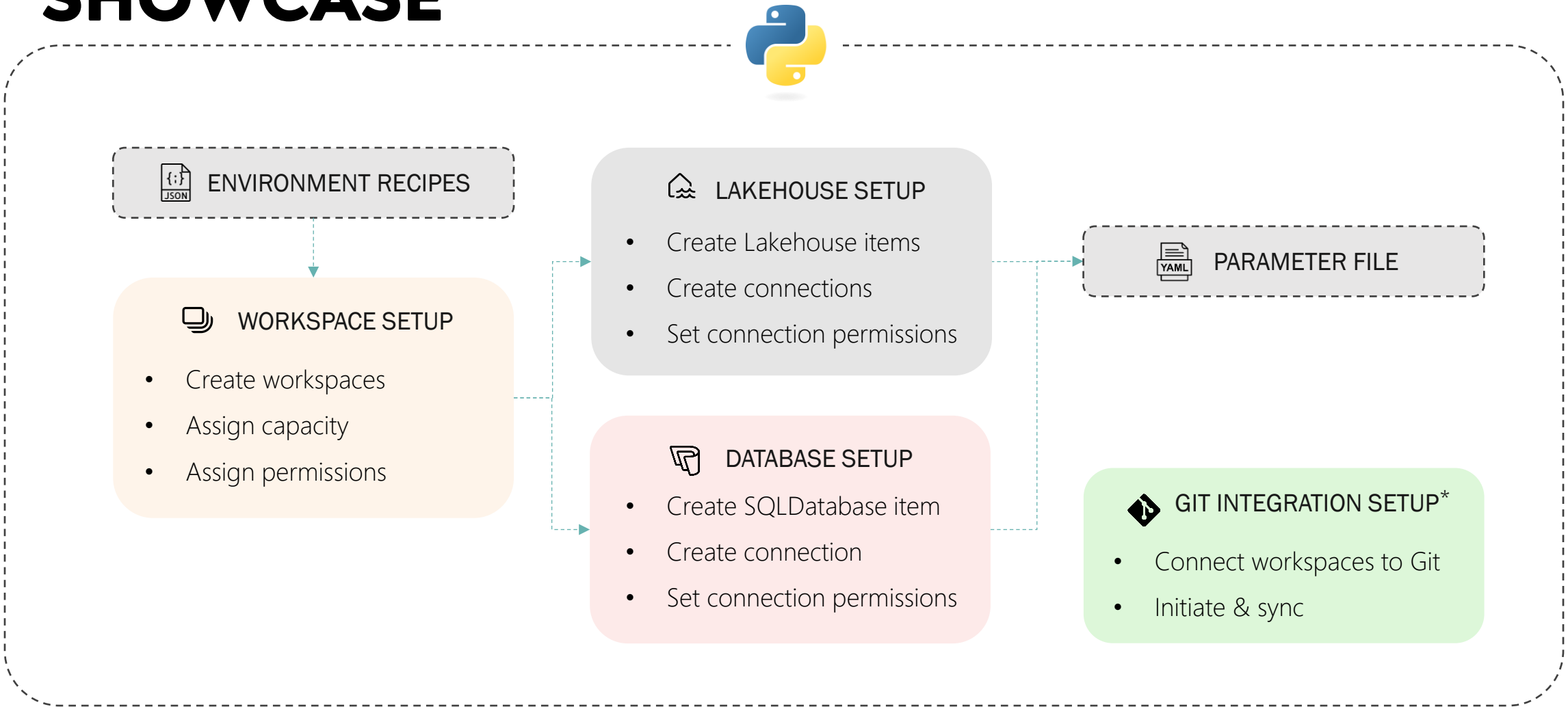
- Azure DevOps Pipeline *
- Local Python script

Recipe approach for declaring infrastructure

1. Download source code: https://github.com/gronnerup/Fabric

Blog post "Automating Fabric: Kickstart your Fabric Setup with Python and Fabric REST APIs": https://peerinsights.hashnode.dev

AUTOMATING FABRIC SOLUTION SETUP – SHOWCASE



USING A RECIPE-BASED APPROACH TO INFRASTRUCTURE-AS-CODE

BASE RECIPE

```
{
    "name": "*FabCon - {layer} [{environment}]",
    "generic": {
        "capacity_id": ”00000000-0000-0000-0000-000000000000",
        "permissions": { ... },
        "git_integration": { ... }
    },
    "layers" :
    {
        "Ingest": { },
        "Store": {
           "items": {
                "Lakehouse": [
                    {"item_name": "Landing", "connection_name": "FabCon-Landing [{environment}]"},
                    {"item_name": "Base", "connection_name": "FabCon-Base [{environment}]"},
                    {"item_name": "Curated", "connection_name": ”FabCon-Curated [{environment}]"}
                ],
                "SQLDatabase": [
                    {
                        "item_name": "Metadata",
                        "connection_name": ”FabCon-MetadataDB [{environment}]",
                        "sql_script": "../resources/Metadata.sql"....
```

RECIPE FILES

- infrastructure.dev.json
- infrastructure.json
- infrastructure.prd.json
- infrastructure.tst.json

QUICK OVERVIEW OF THE MAIN SETUP OPTIONS

REST APIs

Utilizing Fabric REST APIs

Pros
Flexible and scalable
Fully customizable
Integrates with DevOps
Improves governance
Better resource control

Cons
Higher inital setup effort
Possible API coverage gaps

ClickOps

Manually through the Fabric UI.

Pros
Quick and easy for tiny setups
Suitable for testing

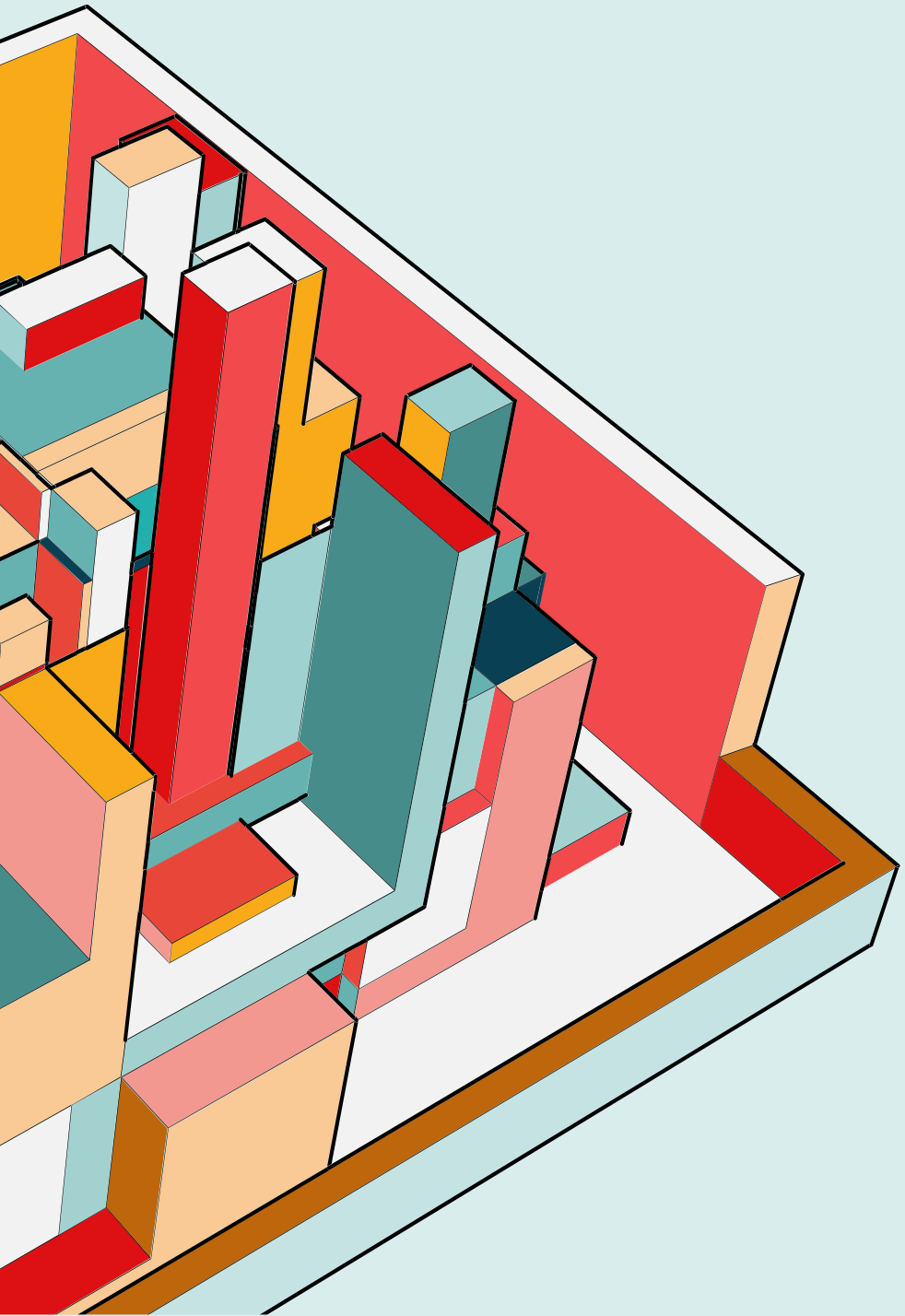
Cons
Not scalable
Error-prone
No version control
Not CI/CD friendly

Terraform

Leading IaC tool!

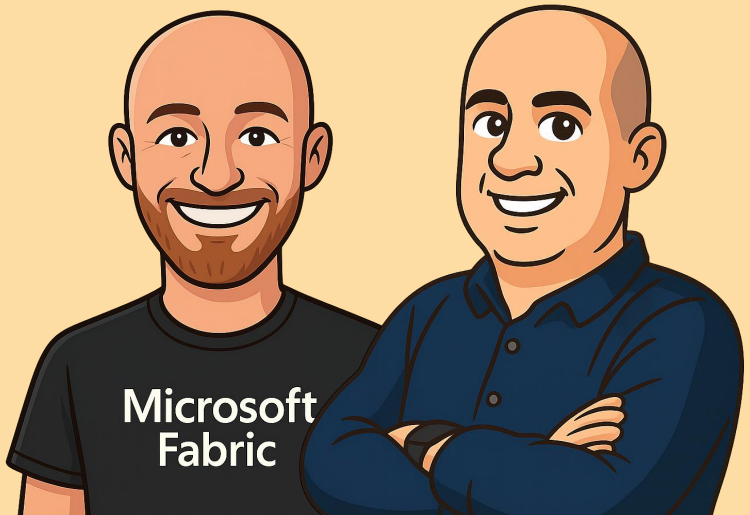
Pros
Industry standard
Readable and scalable
Ensures consistent deployment
Declarative & State Management
Multi-cloud / cross-service

Cons
Can be complex
Would still require a hybrid-setup



WHEN TO CHOOSE WHICH SCENARIO??

THANK YOU



Building a bridge from zero-to-hero

Microsoft Fabric