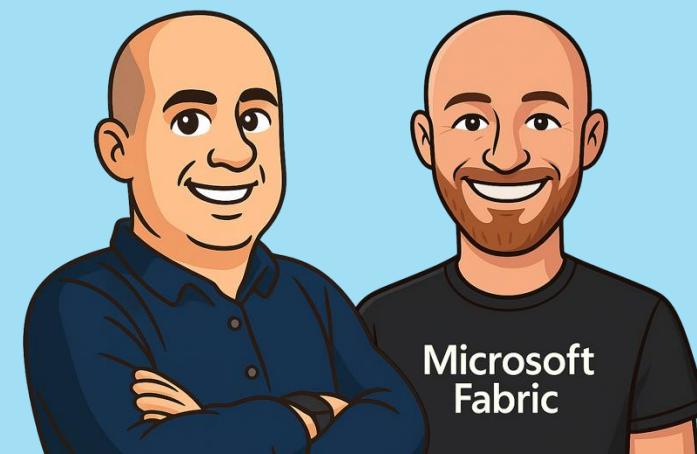




# FABRIC AUTOMATION ALL THE WAY

A DEEP DIVE INTO AUTOMATING  
YOUR FABRIC DATA PLATFORM



# Thank you sponsors!



**redgate**

**twintos** **wortell** **.infoSupport**  
Solid Innovator

 **INSPARK**

**macaw**

Your guide in the Era of AI.

**Rubicon**  
Consulting & Technology

**illionx**



**BRANDER**  
  
**company**

# Rate Data Saturday Holland

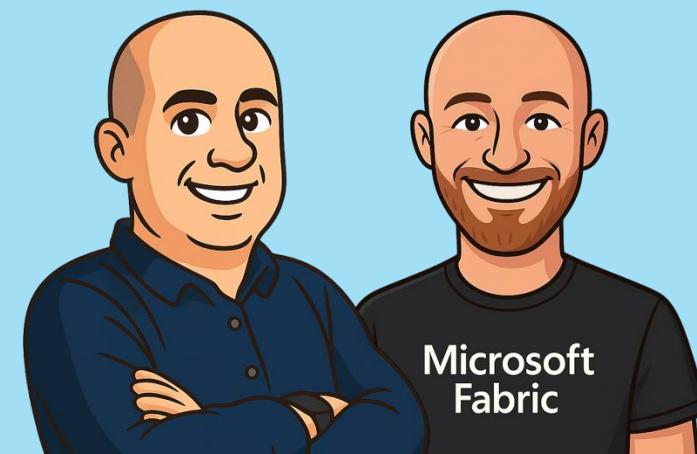


**Review to win  
With custom Data Saturdays lego stickers!**

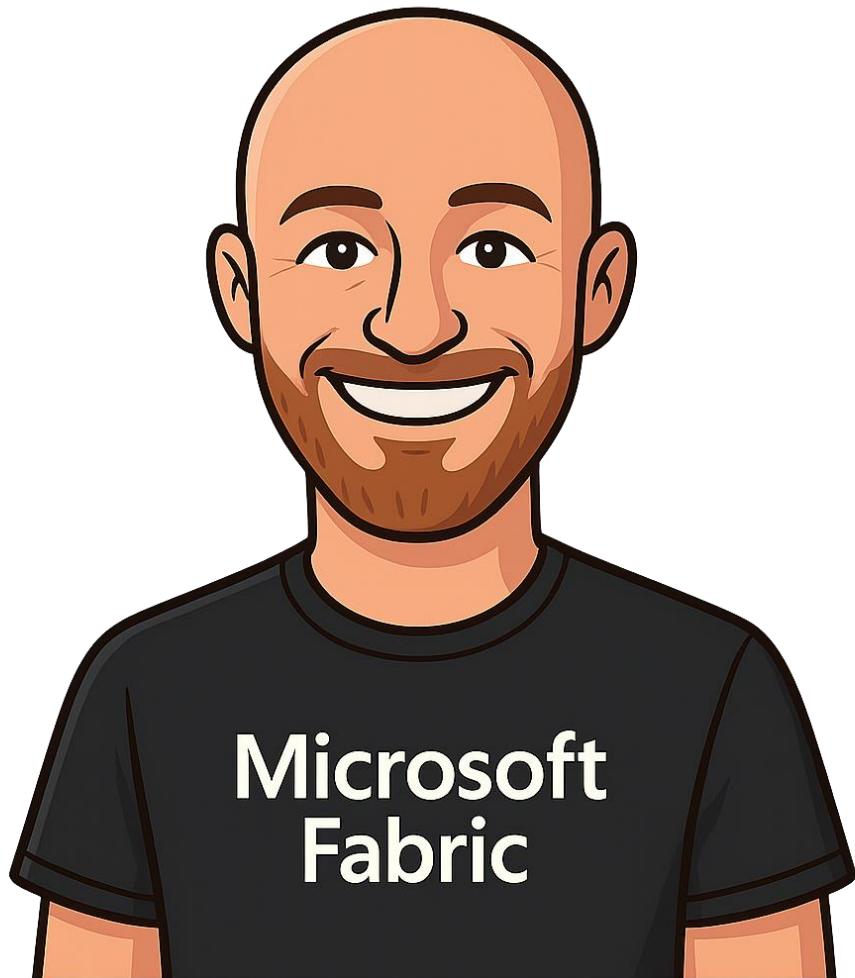


# FABRIC AUTOMATION ALL THE WAY

A DEEP DIVE INTO AUTOMATING  
YOUR FABRIC DATA PLATFORM



# 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/>

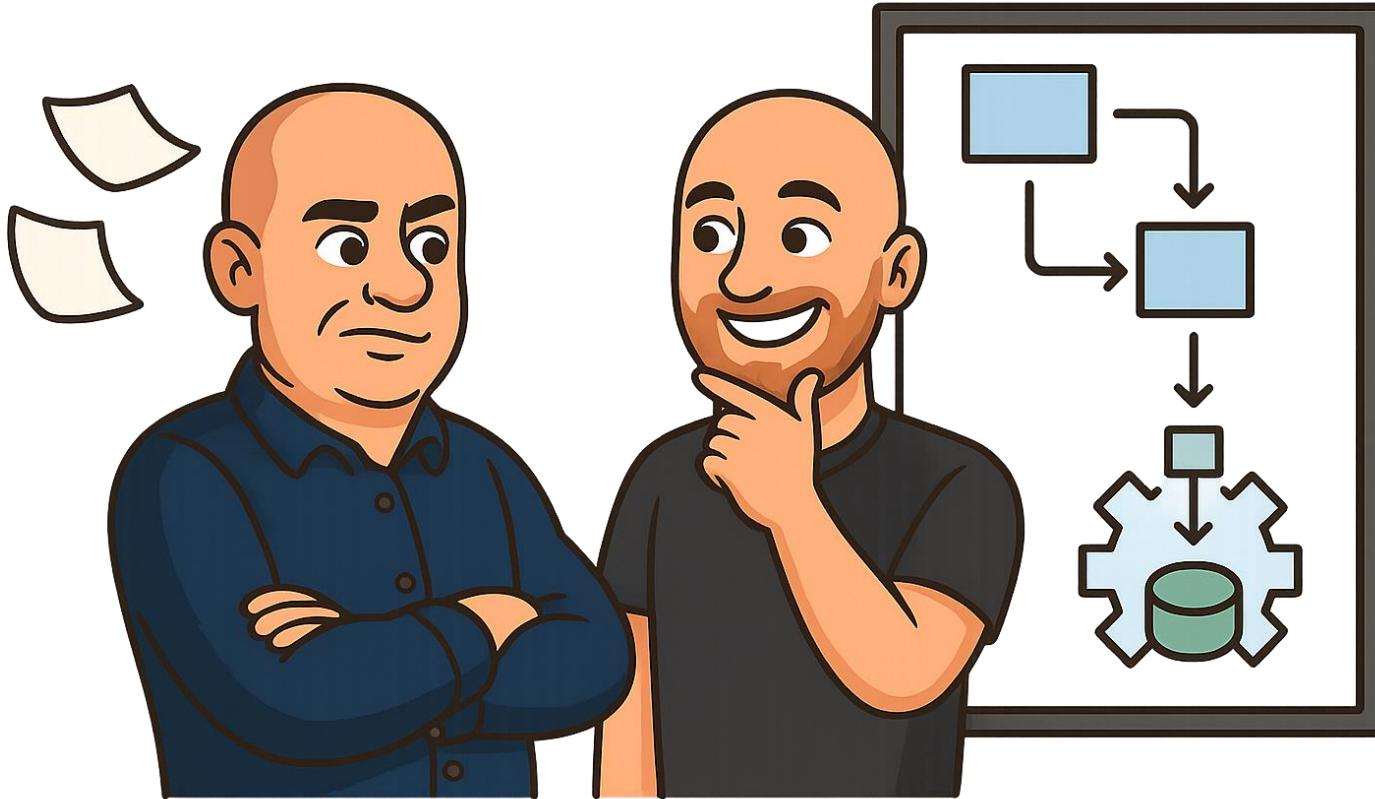
<https://erwindekreuk.com>

<https://github.com/edkreuk>

<Https://Dutchfabricusergroup.com>

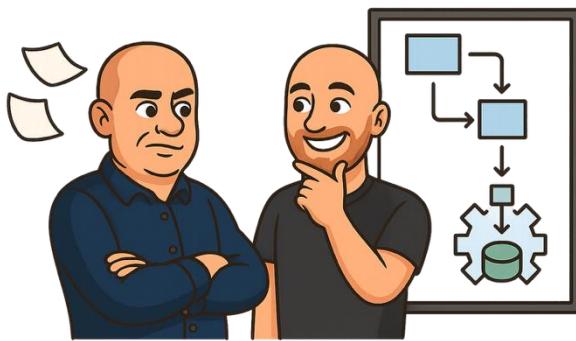


# SO, WHERE ARE WE GOING TODAY...?



Why automation matters?

# SO, WHERE ARE WE GOING TODAY...?



Why automation matters?



Tooling

# **SO, WHERE ARE WE GOING TODAY...?**

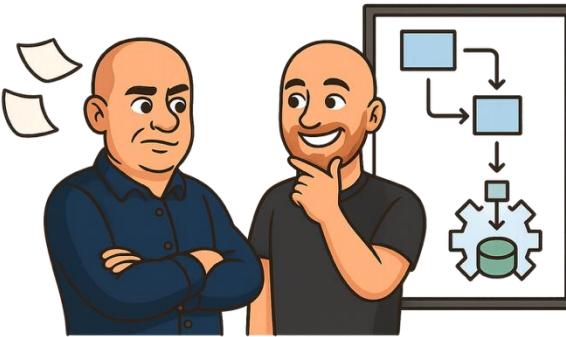


Why automation matters?

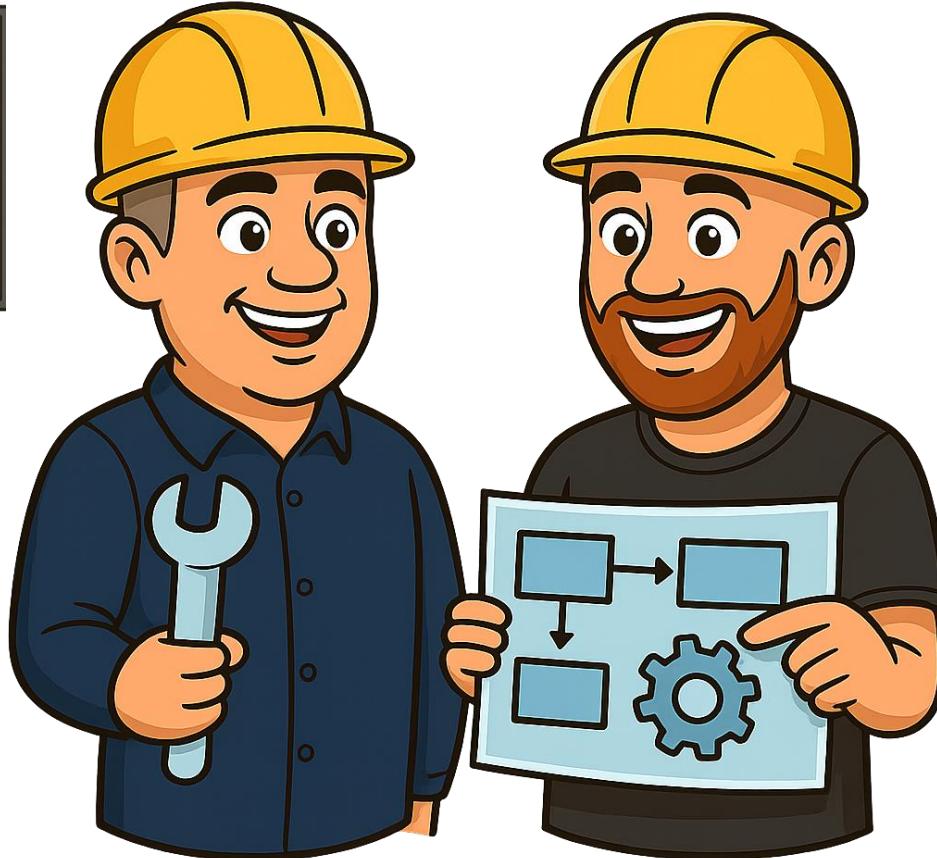


Getting started as a non hero

# SO, WHERE ARE WE GOING TODAY...?



Why automation matters?



Getting started as a non hero

Architecting for automation

# SO, WHERE ARE WE GOING TODAY...?



Why automation matters?



Getting started as a non hero



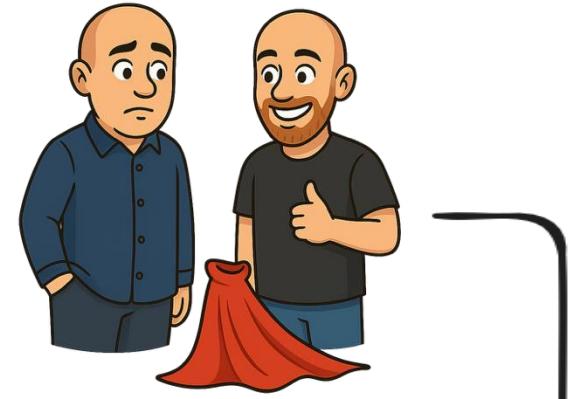
Architecting for automation

Fabric IaC

# SO, WHERE ARE WE GOING TODAY...?



Why automation matters?



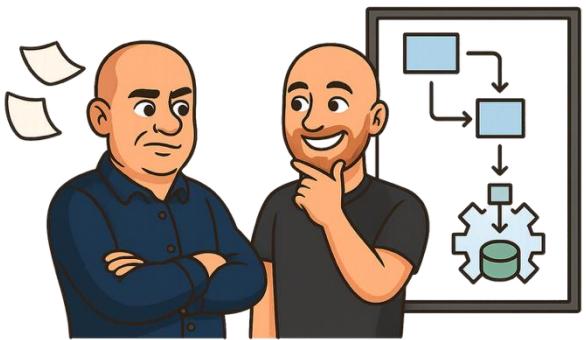
Getting started as a non hero



Architecting for aut

**BREAK**

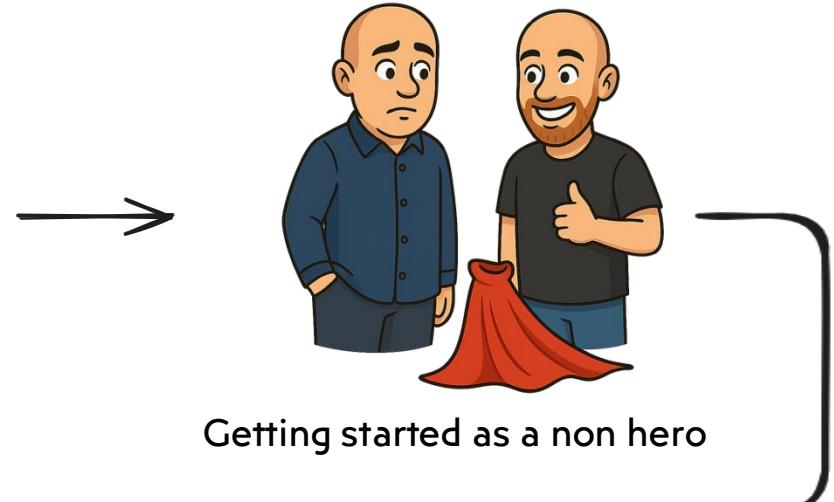
# SO, WHERE ARE WE GOING TODAY...?



Why automation matters?



Tooling



Getting started as a non hero



Architecting for automation

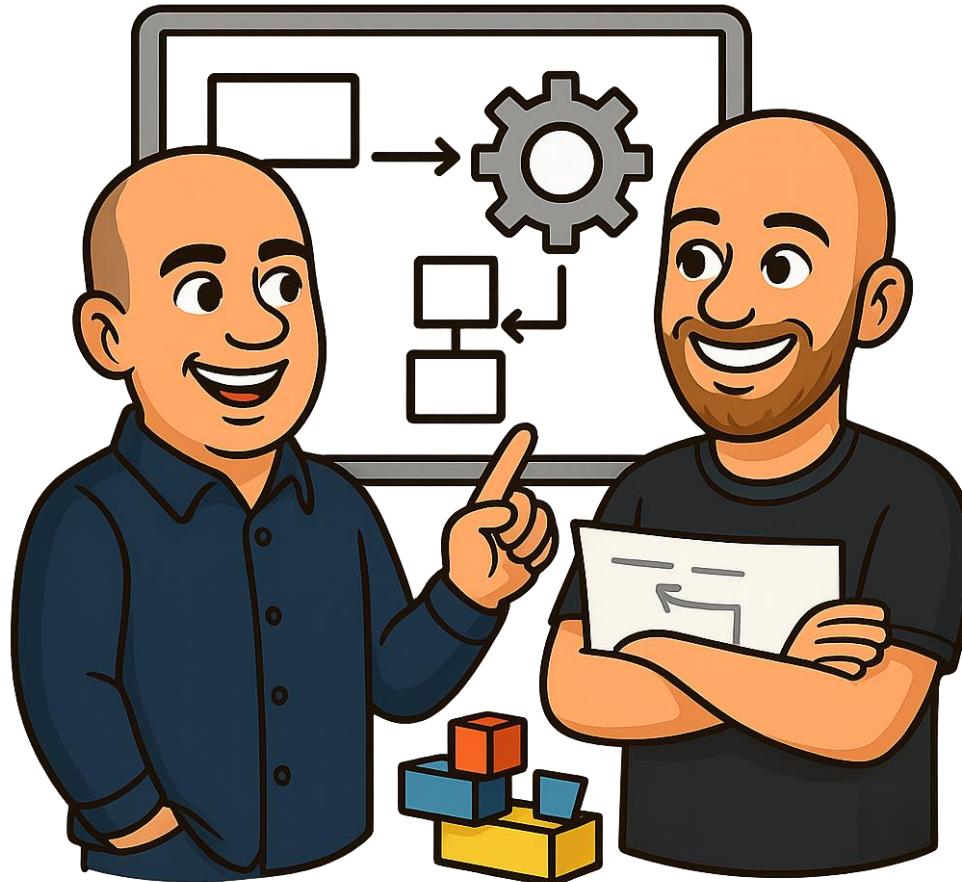


Fabric IaC



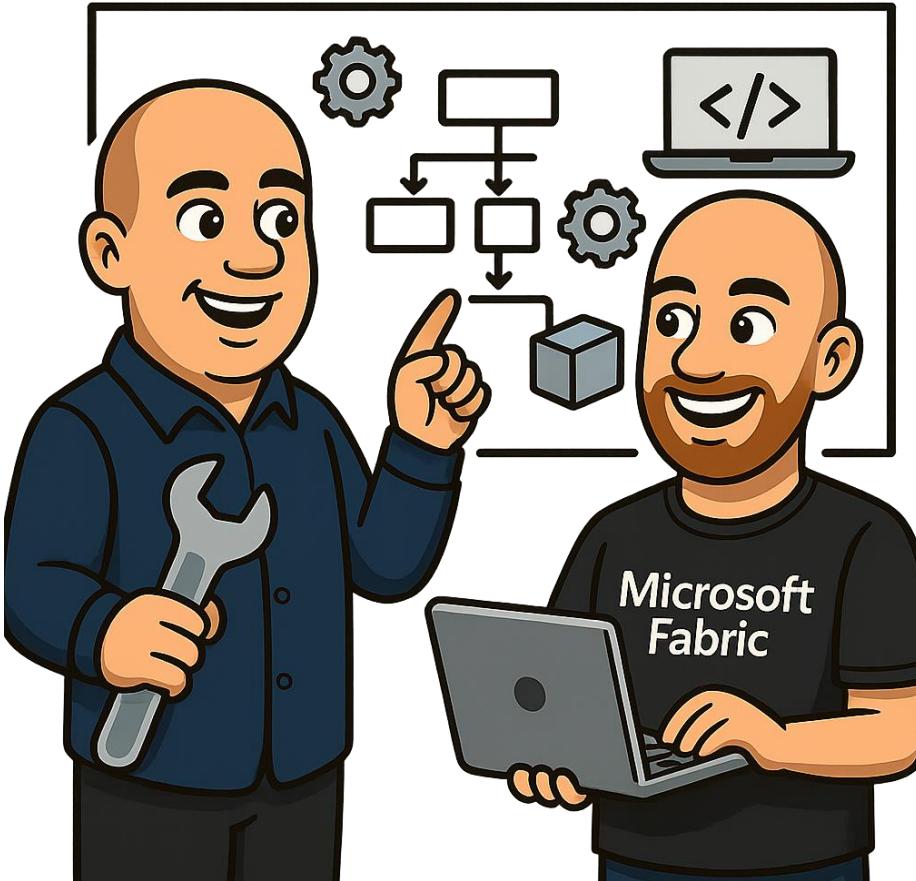
BREAK

# ...AND AFTER THE BREAK?



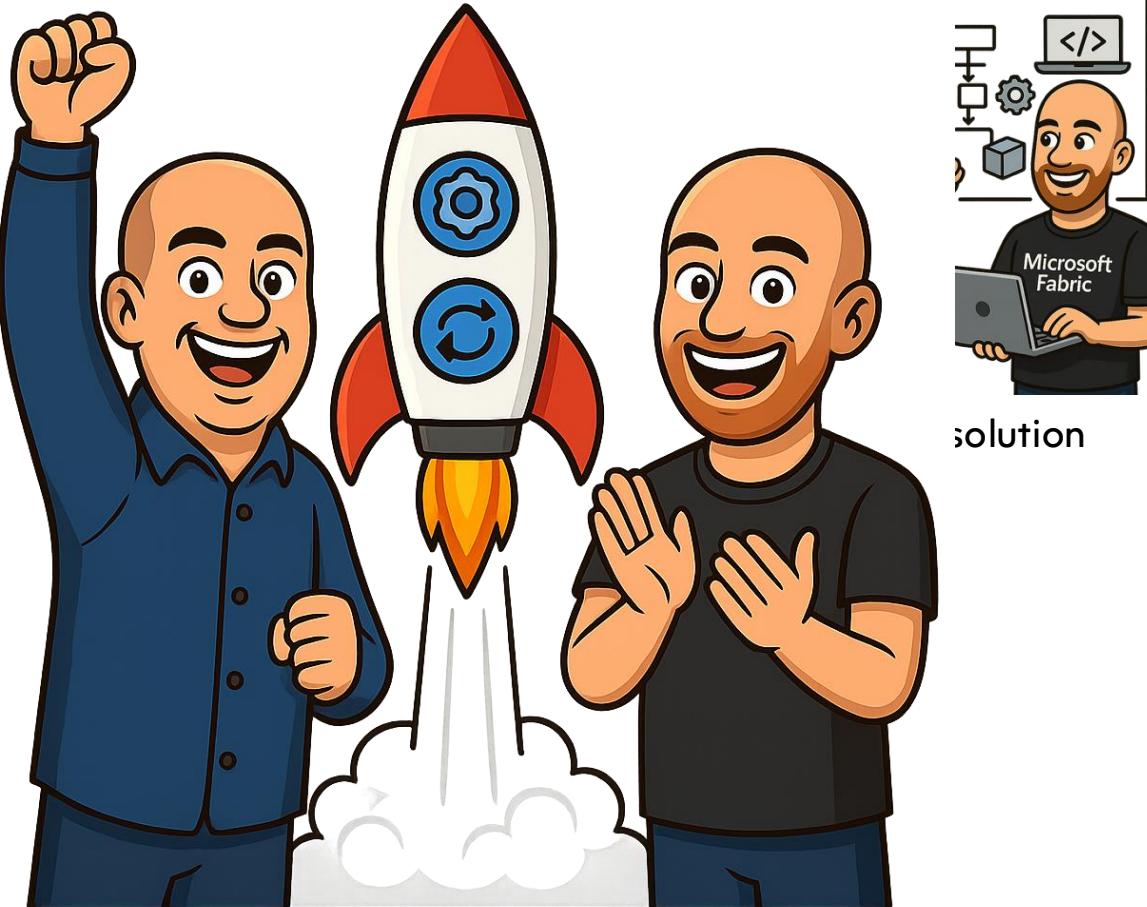
Ways-of-working

# ...AND AFTER THE BREAK?



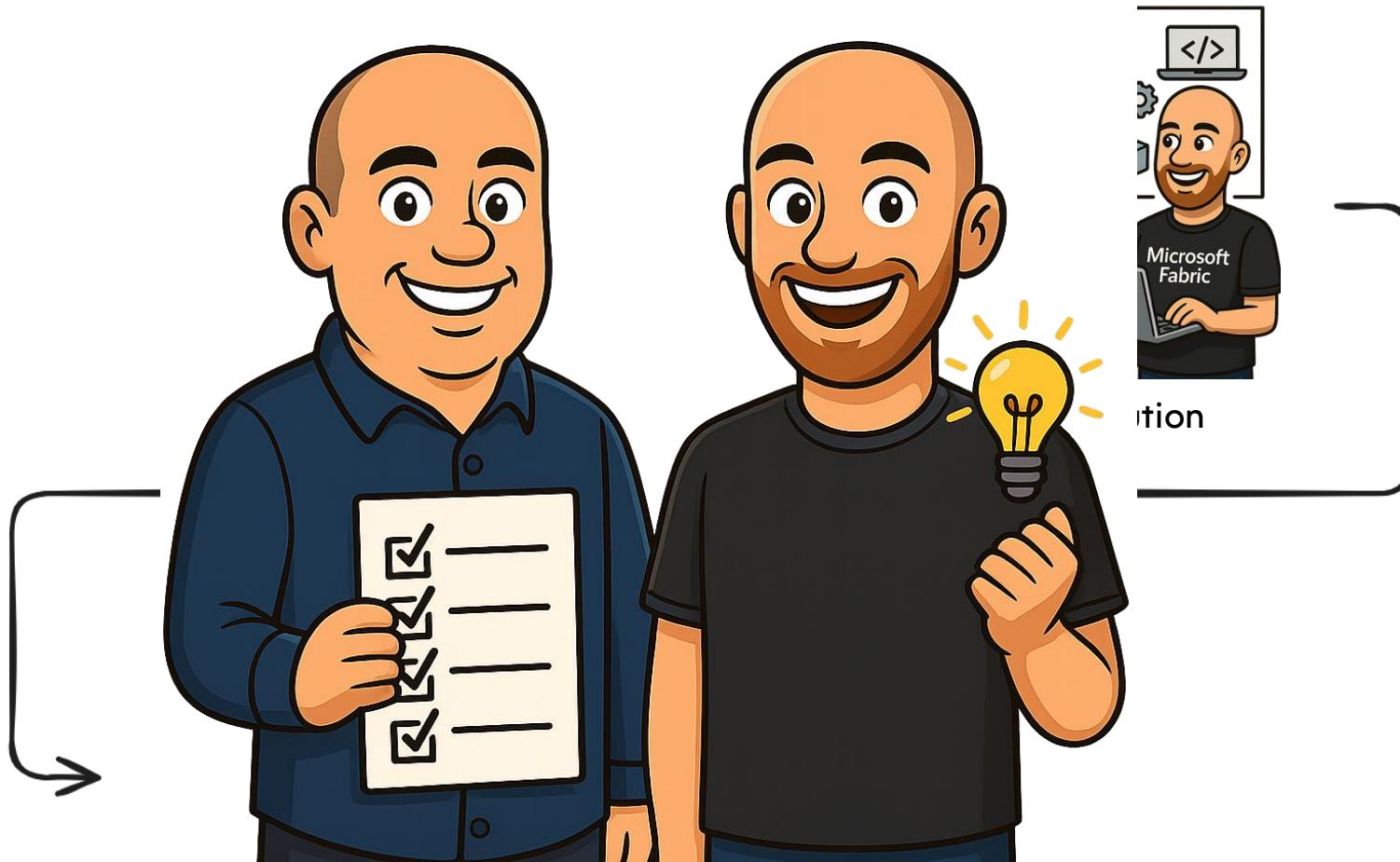
Building our solution

# ...AND AFTER THE BREAK?



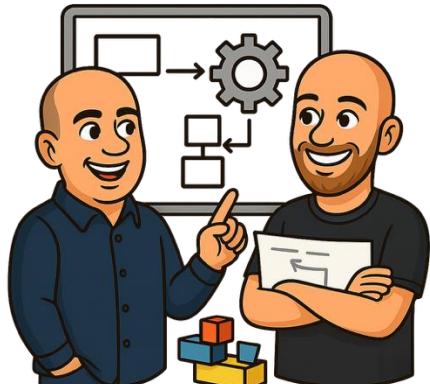
Deployment

# ...AND AFTER THE BREAK?

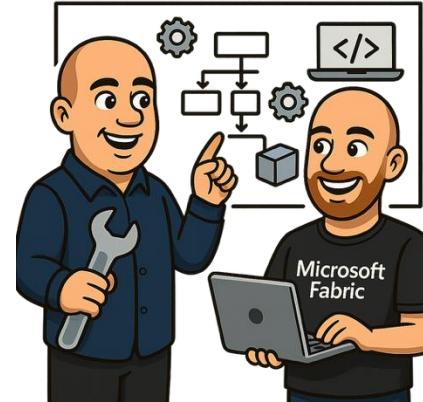


Best Practices, tips & tricks

# ...AND AFTER THE BREAK?



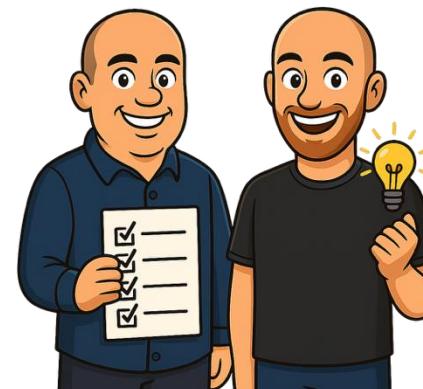
Ways-of-working



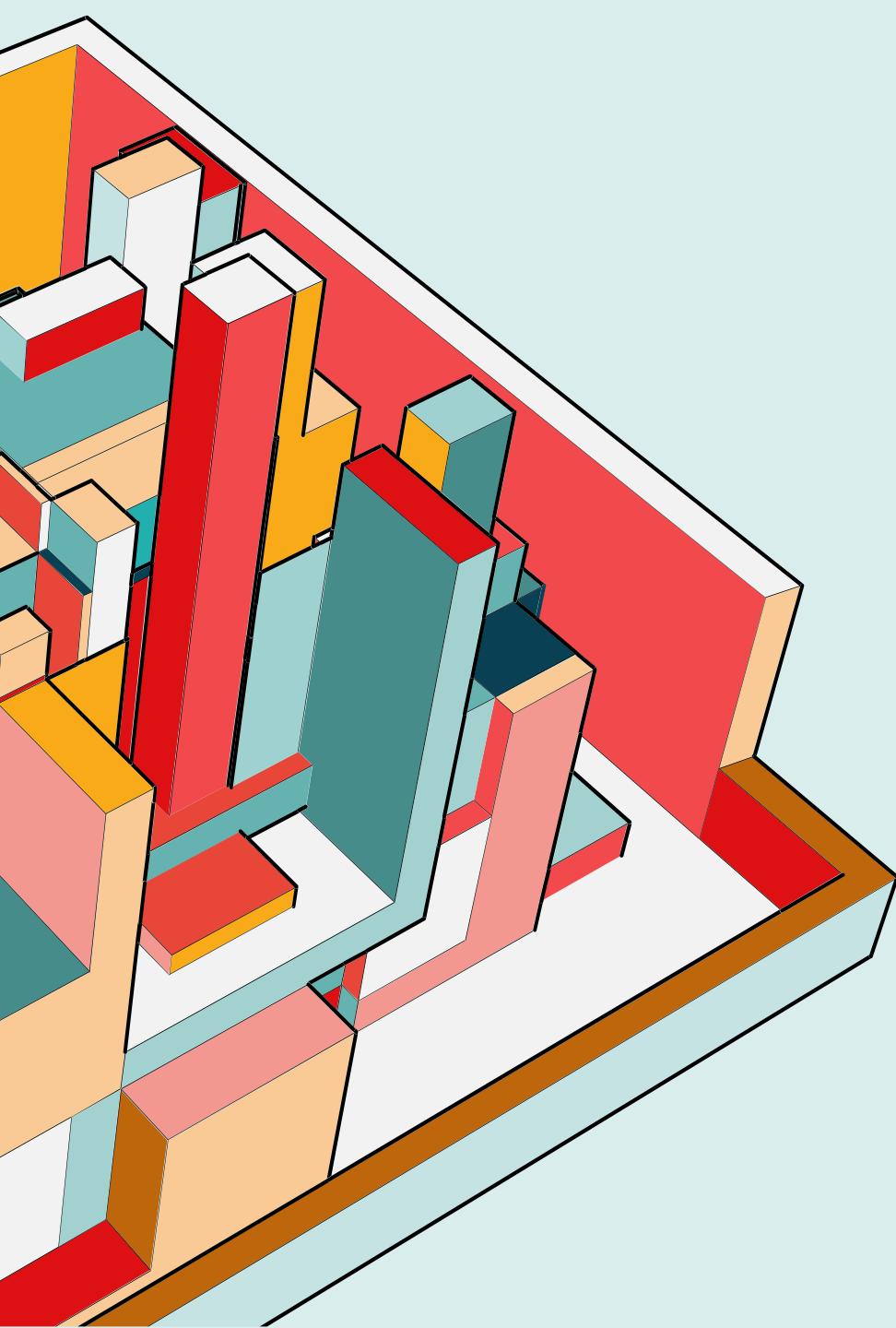
Building our solution



Deployment



Best Practices, tips & tricks

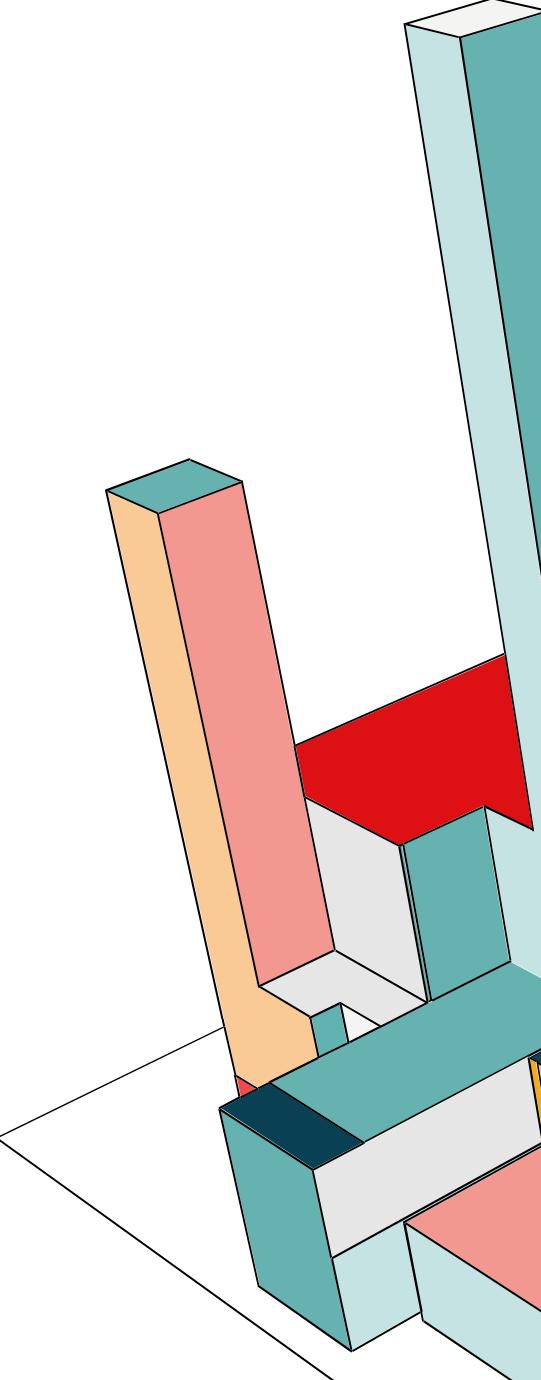


# BEFORE WE START...

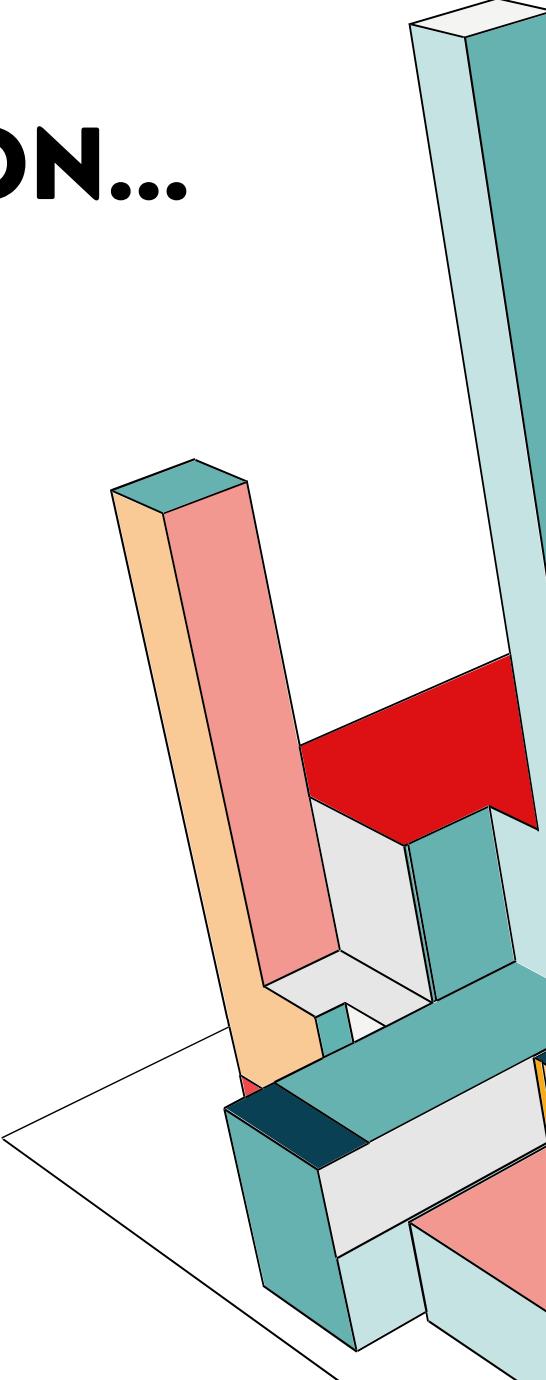
A quick poll

# WHAT LEVEL ARE YOU?

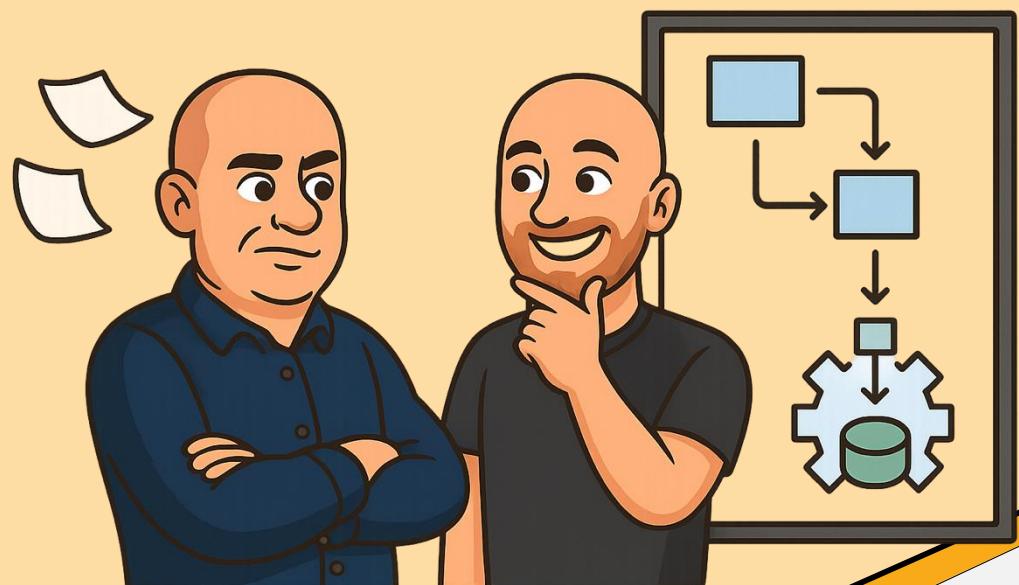
- Where are you on automation?
- And what tools are you using?
- Are you using Fabric git integration?



# WHERE ARE WE NOW? POST-FABCON...



# WHY AUTOMATION MATTERS!



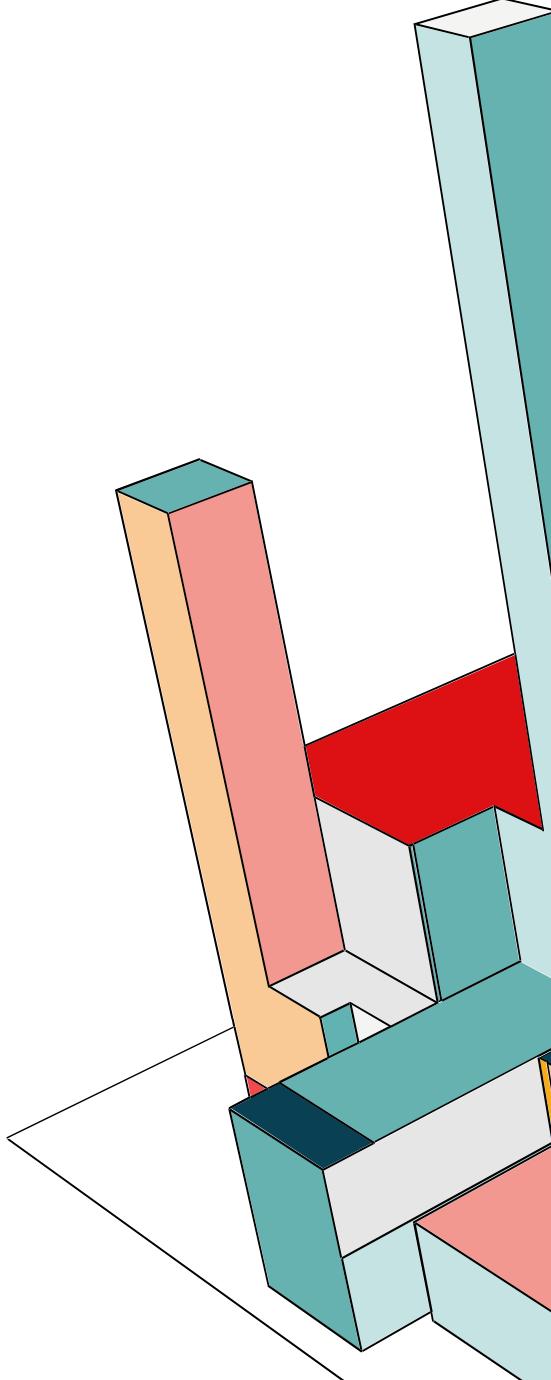
# WHY AUTOMATION MATTERS!

## 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

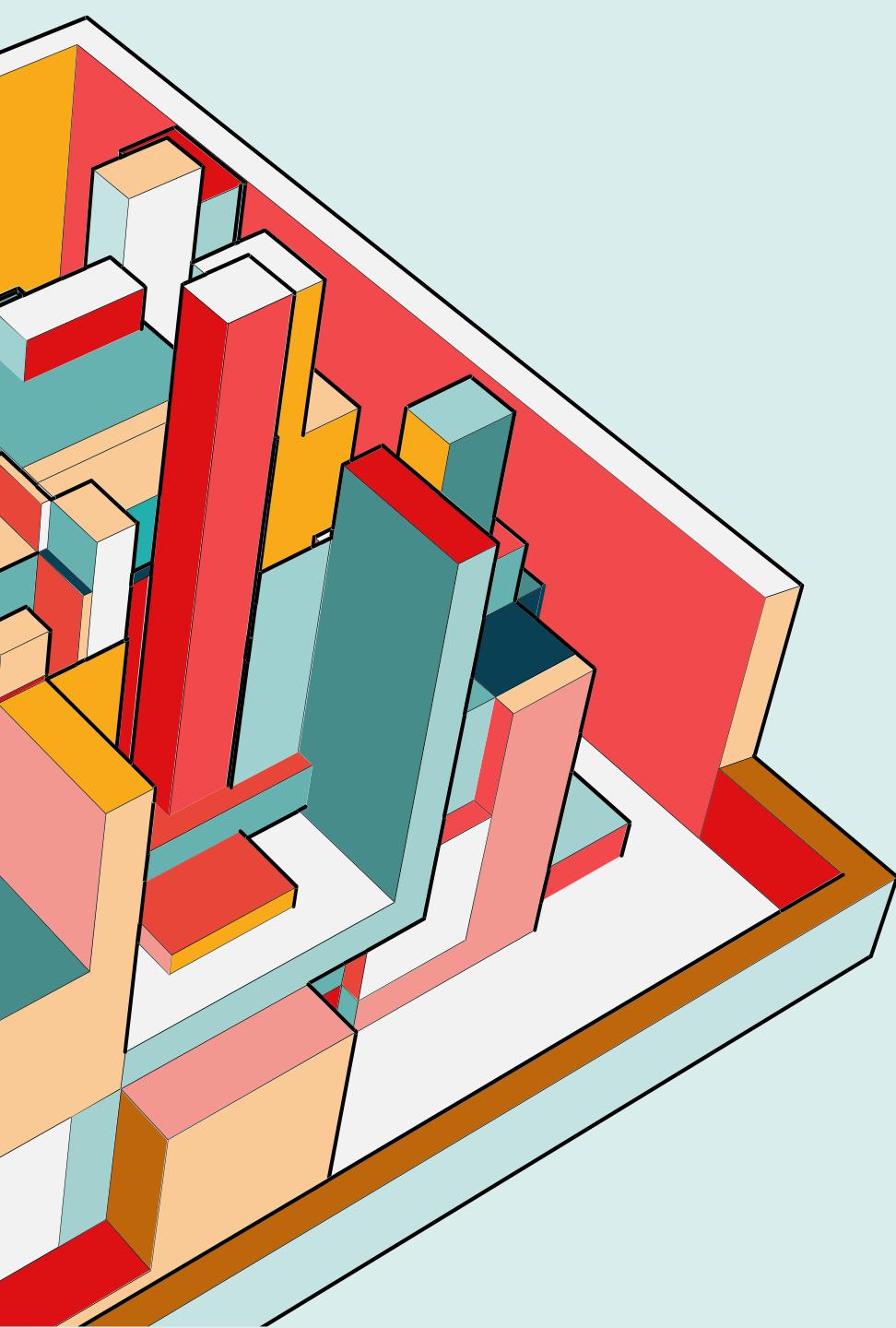


## Foundation Layer

Serve as the base for all interactions

Fabric REST APIs

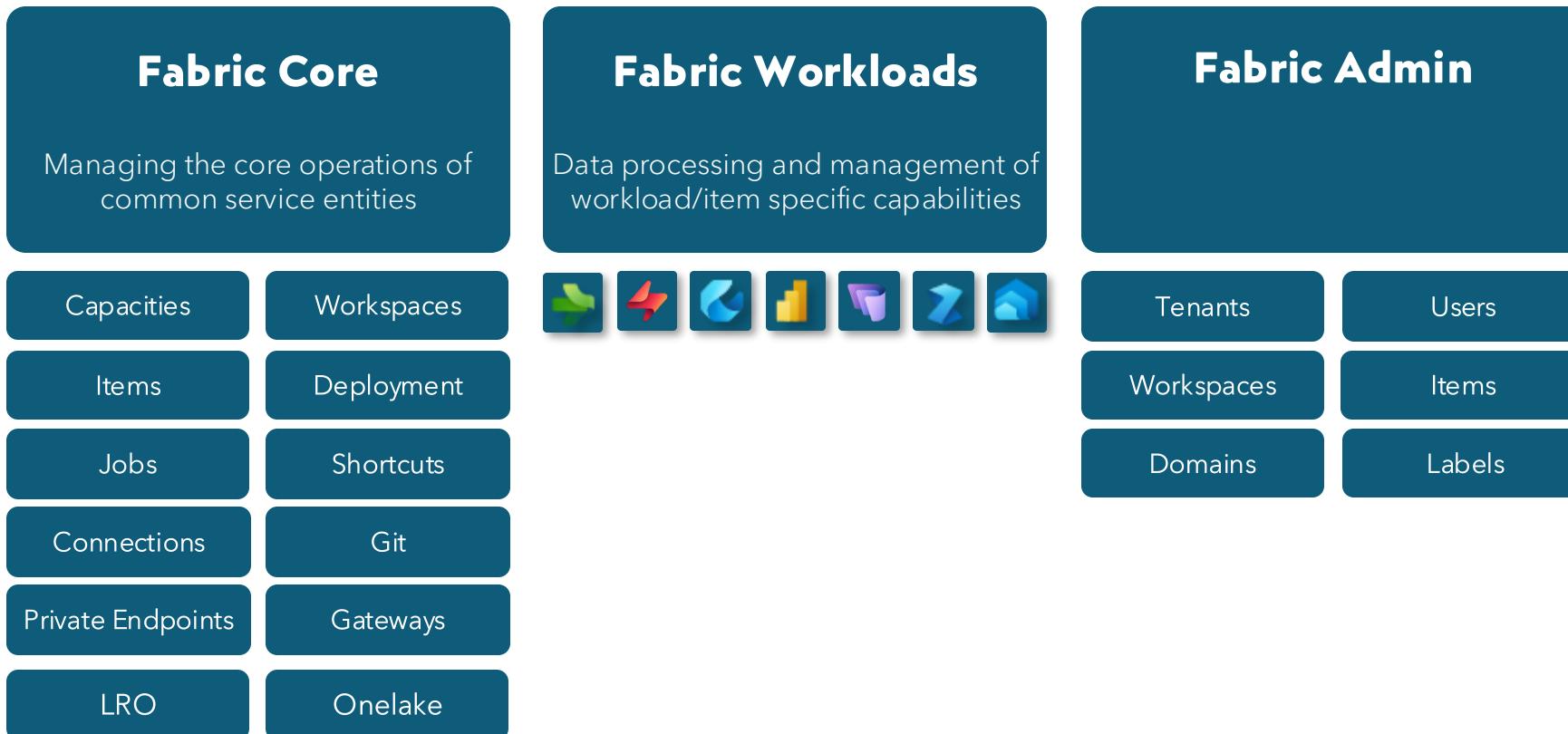




# FABRIC REST APIs

The foundation for automation

# AUTOMATING FABRIC USING REST APIs

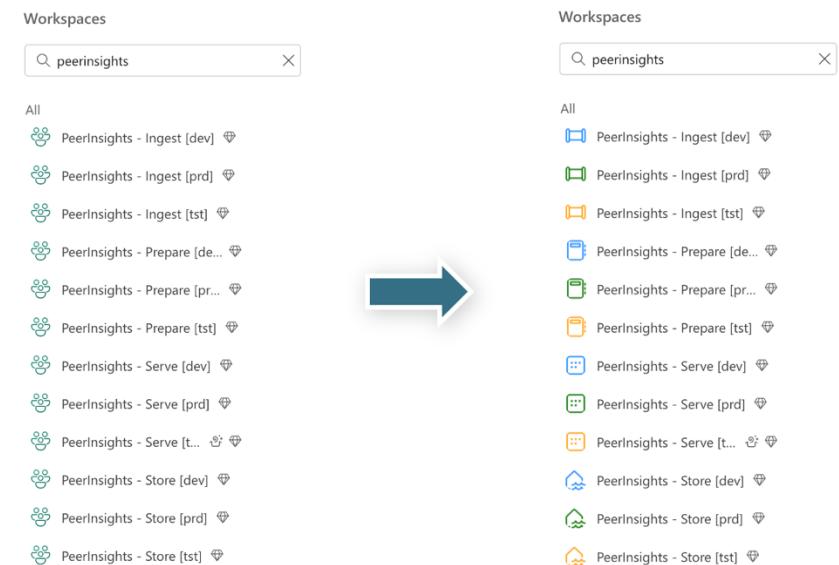


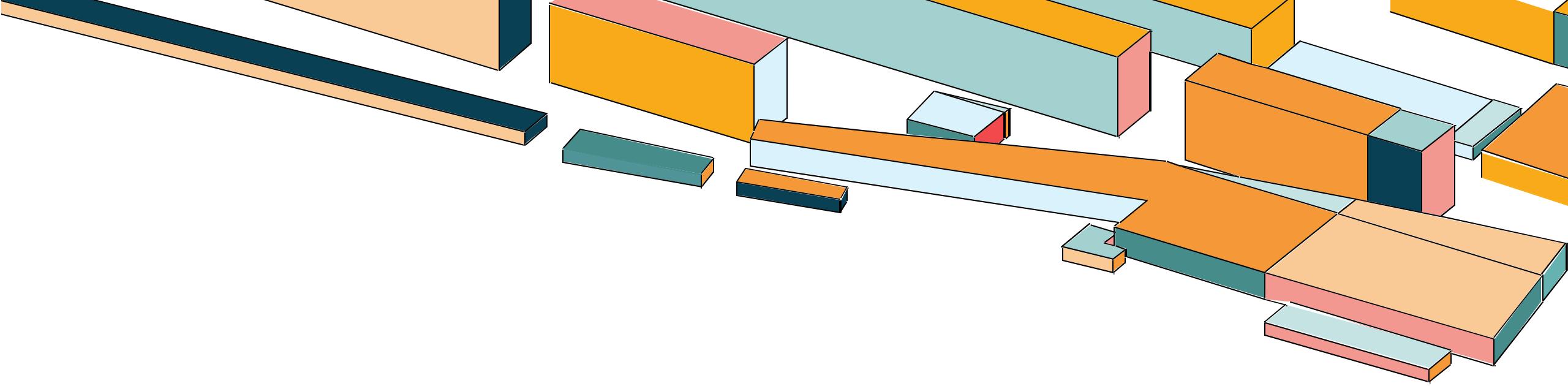
→ <https://learn.microsoft.com/en-us/rest/api/fabric/articles/>

# INTERNAL AND UNSUPPORTED APIs

Where there is a will – there is a way...

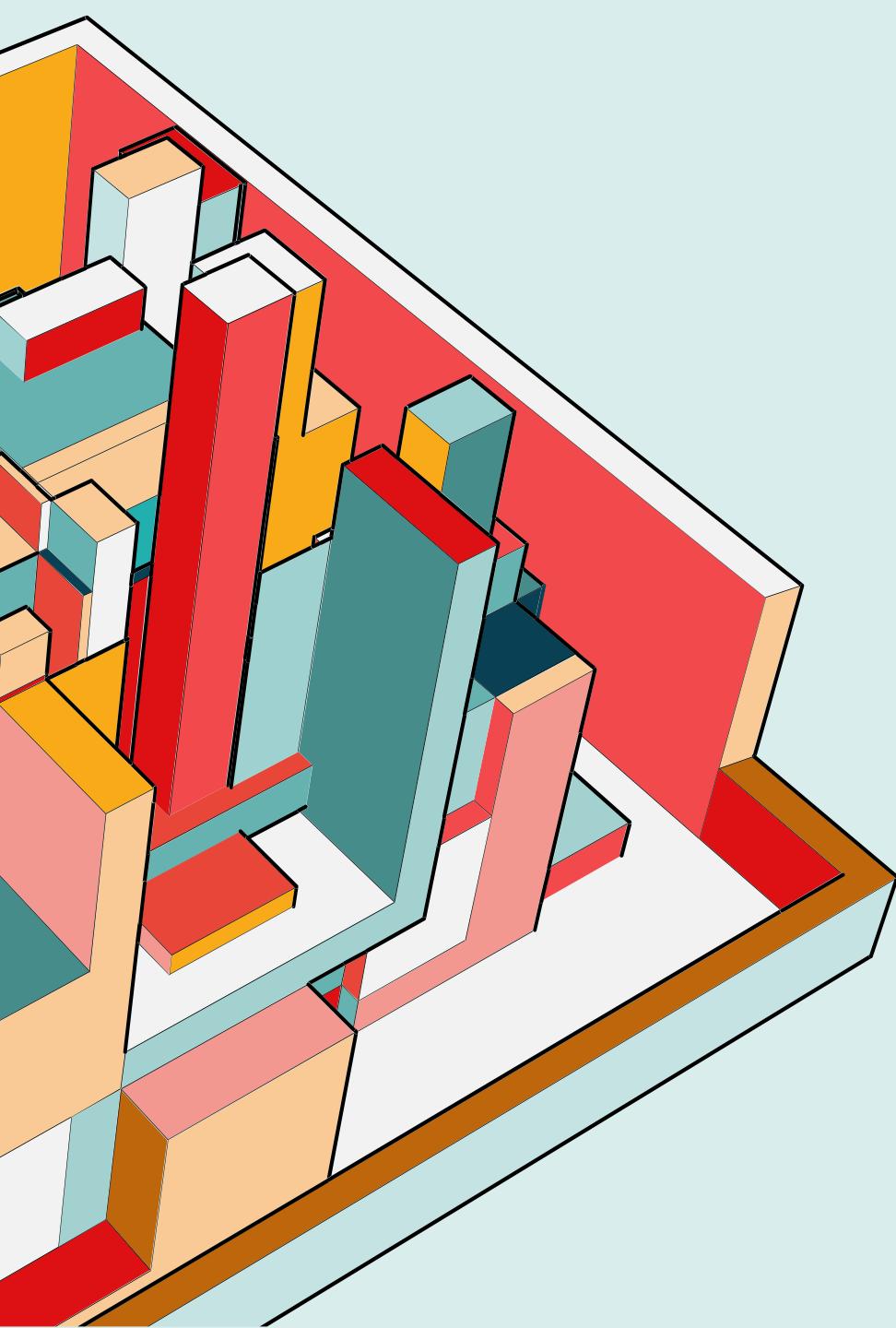
- What is possible in the Fabric/Power BI Service is most likely also possible to automate
- Get ready to do some reverse engineering
- Tooling:
  - Browser (developer mode)
  - Postman
  - VS Code, Cursor or other IDEs
  - Fabric Notebooks
- Example:  
<https://peerinsights.hashnode.dev/automating-fabric-maintaining-workspace-icon-images>





**DEMO**

The background of the slide features a collection of 3D rectangular blocks of various sizes and colors, including orange, teal, light blue, and pink. These blocks are suspended in the air, creating a sense of depth and motion. The arrangement is somewhat scattered, with some blocks appearing to overlap or be positioned higher than others. The colors of the blocks provide a vibrant contrast against the plain white background.

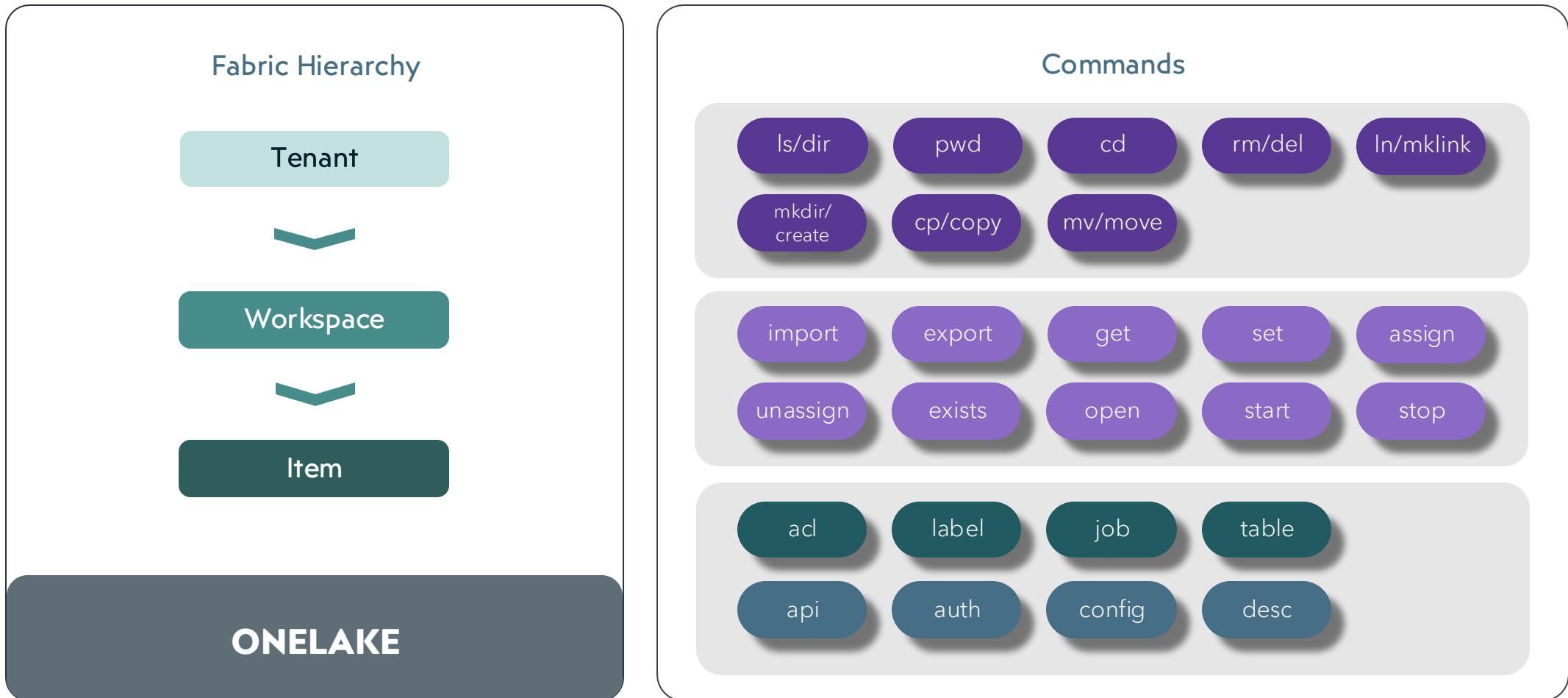


# FABRIC CLI

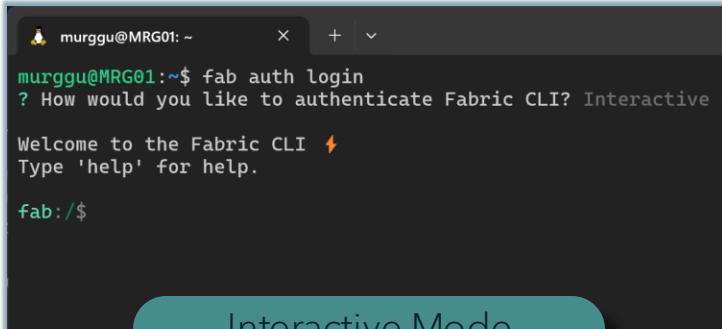
The Command Line Way



# FABRIC AS A FILE SYSTEM



# CLI MODES



```
murggu@MRG01:~$ fab auth login
? How would you like to authenticate Fabric CLI? Interactive mode
Welcome to the Fabric CLI ✨
Type 'help' for help.

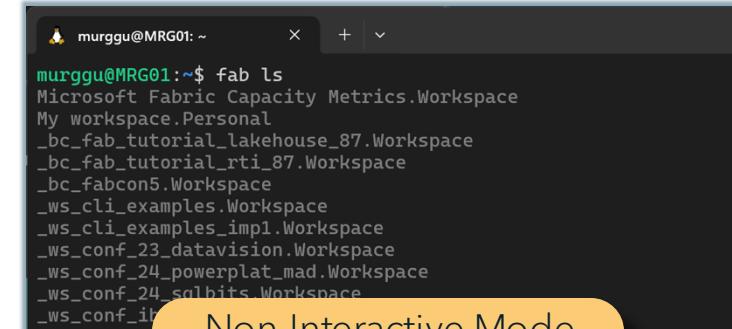
fab:/$
```

Interactive Mode

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



```
murggu@MRG01:~$ fab ls
Microsoft Fabric Capacity Metrics.Workspace
My workspace.Personal
_bc_fab_tutorial_lakehouse_87.Workspace
_bc_fab_tutorial_rti_87.Workspace
_bc_fabcon5.Workspace
_ws_cli_examples.Workspace
_ws_cli_examples_impl1.Workspace
_ws_conf_23_datavision.Workspace
_ws_conf_24_powerplat_mad.Workspace
_ws_conf_24_salbits.Workspace
_ws_conf_ir...
```

Non-Interactive Mode

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



Automation & custom CI/CD

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

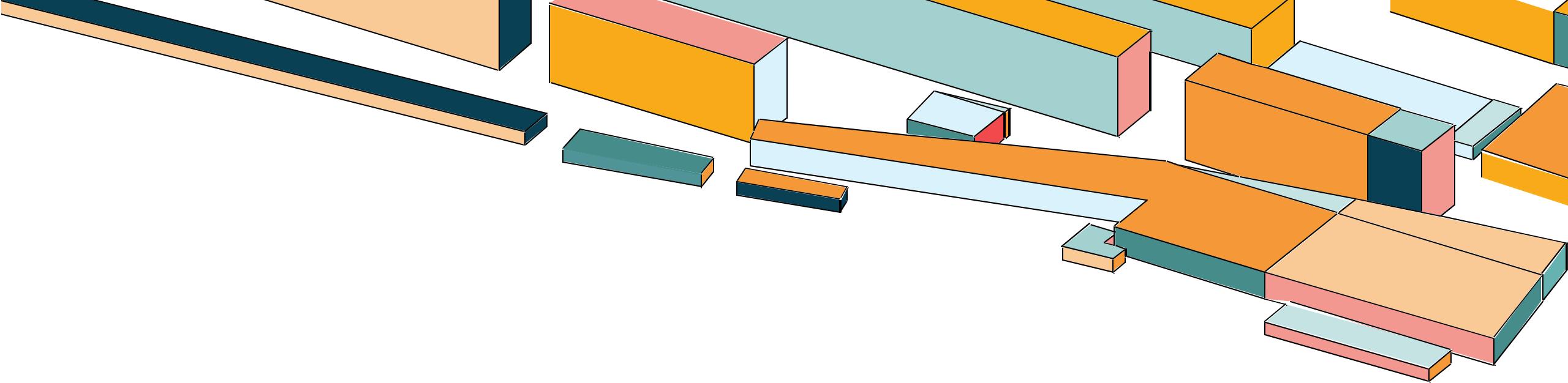


Better developer experience

- 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 accessible from your terminal (PATH environment variable)

## Installation

- pip install ms-fabric-cli

## Use full configuration commands

- fab config set encryptionFallbackEnabled 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



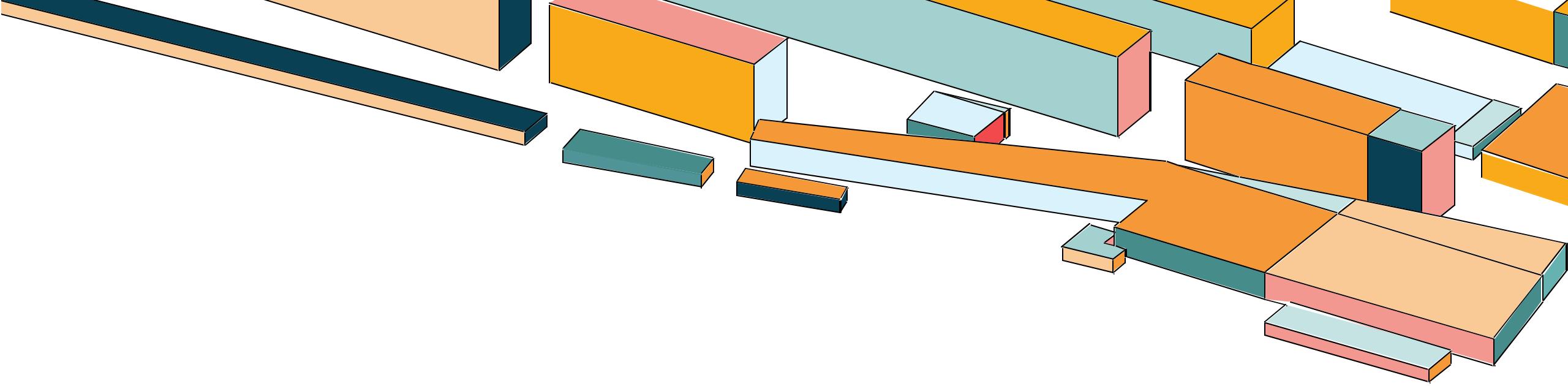
### Using the Fabric CLI

Low initial effort

Supports automation

< 10 sec. / a few mins.

< 10 commands



**DEMO**

The background of the slide features a collection of 3D rectangular blocks of various sizes and colors, including orange, teal, light blue, and white, suspended in the air against a plain white background. The blocks are arranged in a loose, overlapping cluster, with some appearing to float higher than others. The lighting creates soft shadows and highlights on the edges of the blocks, giving them a sense of depth and volume.

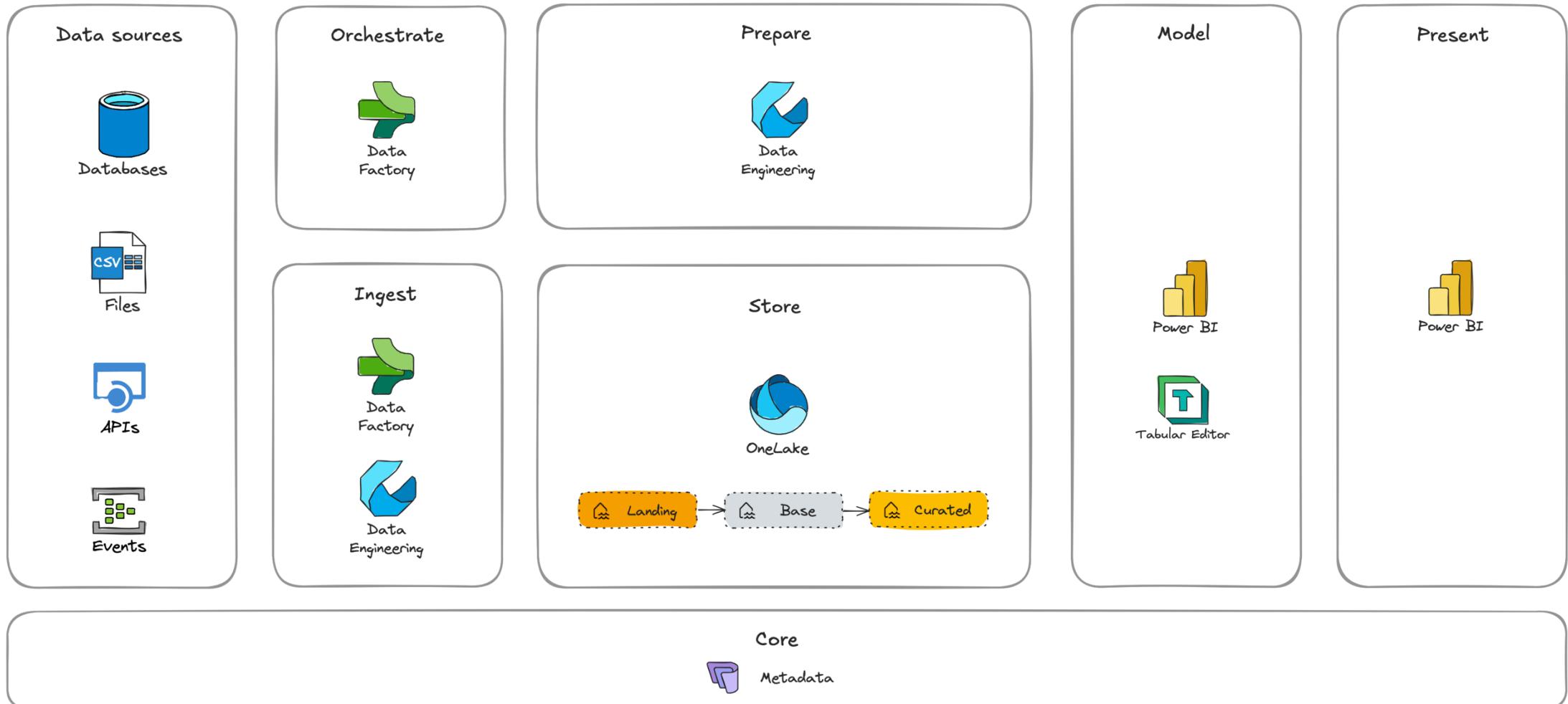
# ARCHITECTING FOR AUTOMATION



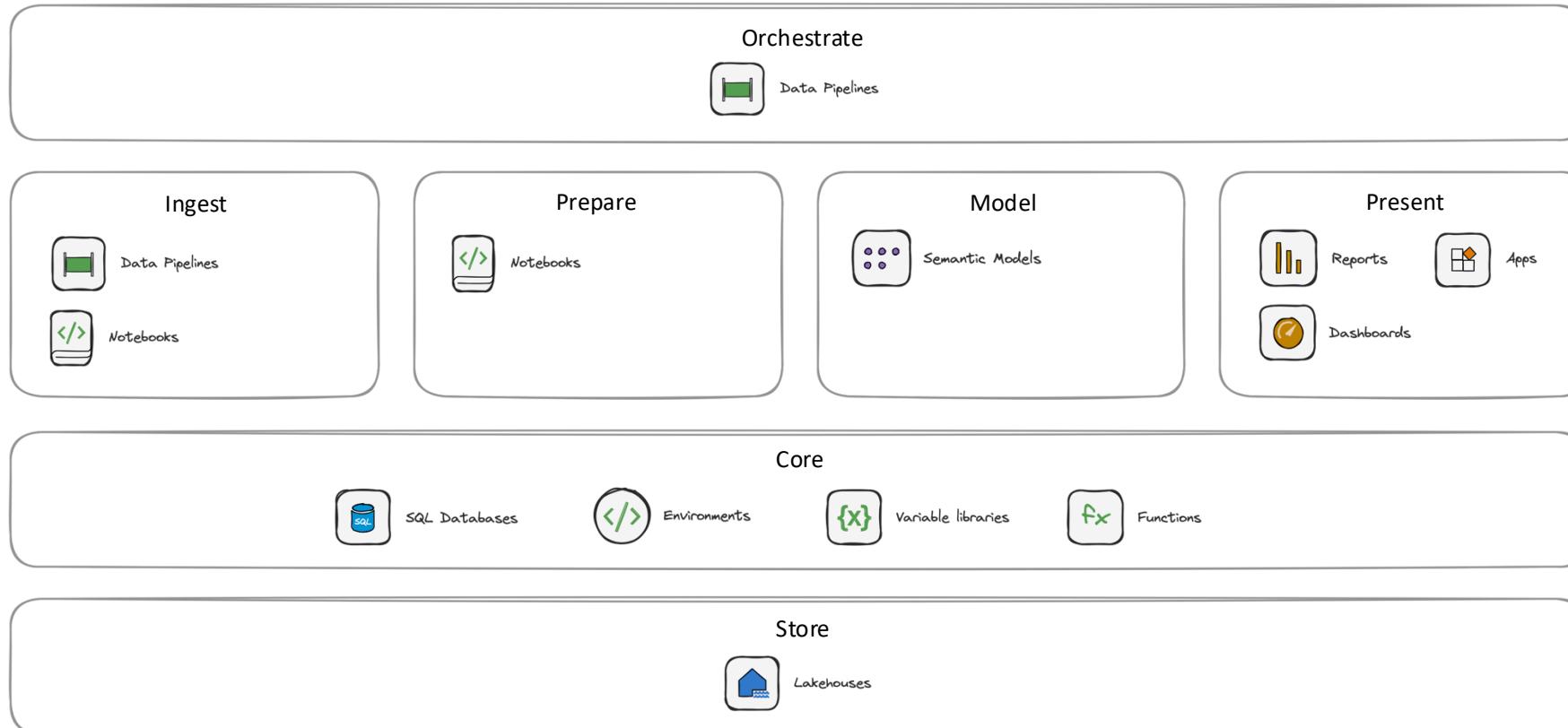
# HOW DO YOU ARCHITECT?



# REFERENCE ARCHITECTURE AS A START



# WORKSPACE STRUCTURE

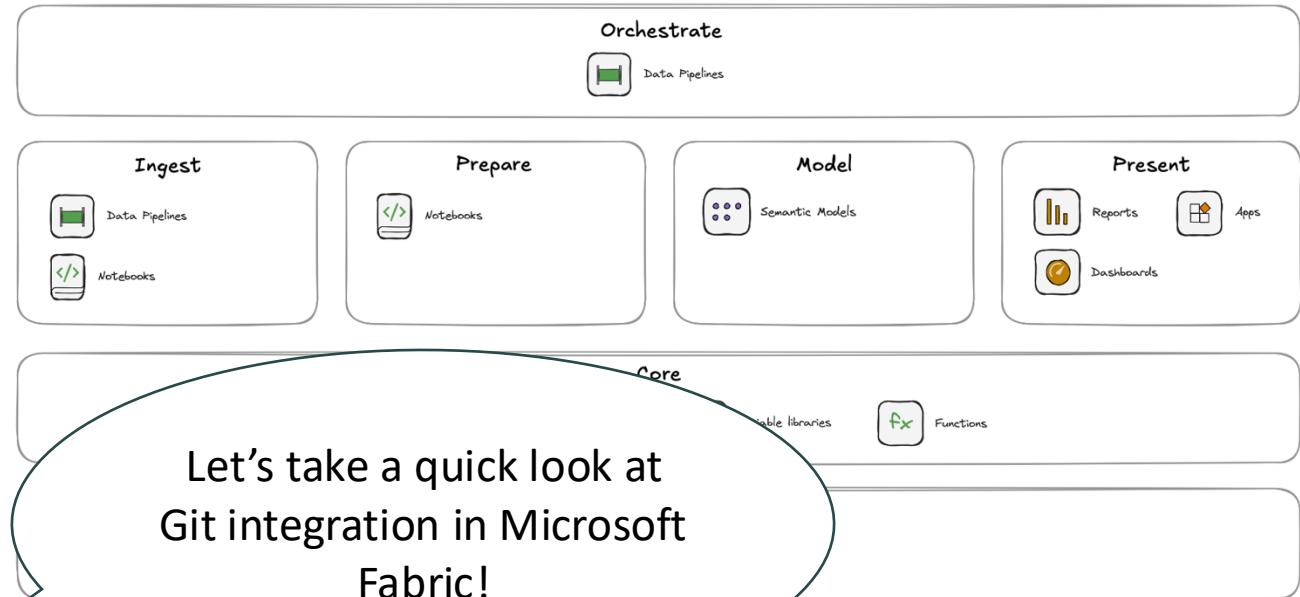


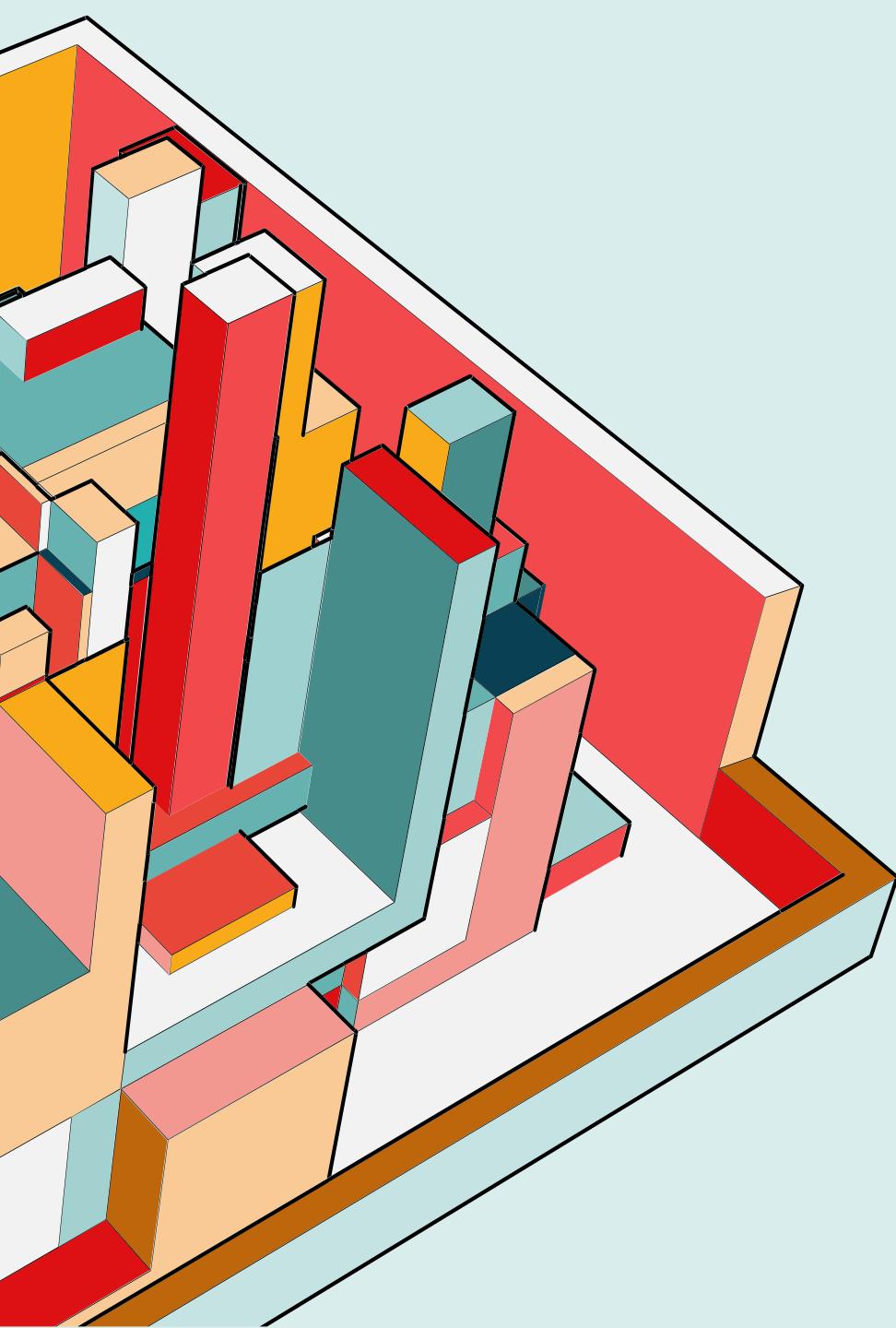
x3

[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

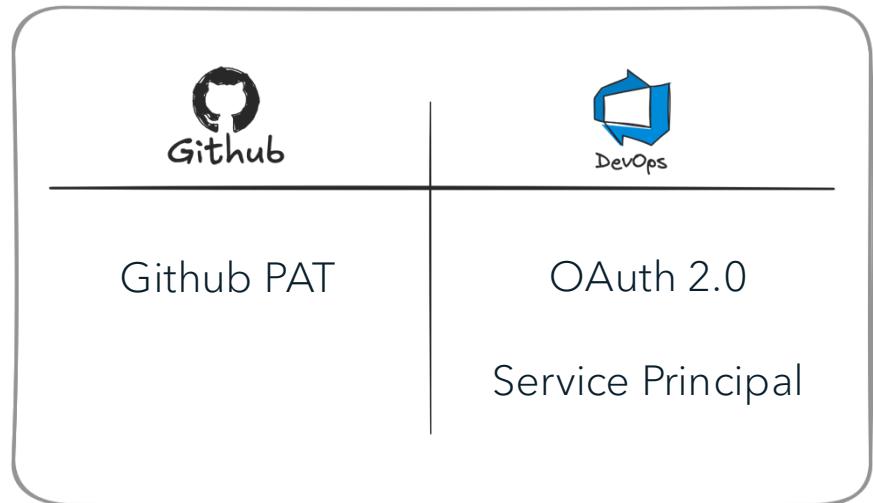
Support for using template repositories with boilerplate scripts, dynamic pipelines etc.

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)

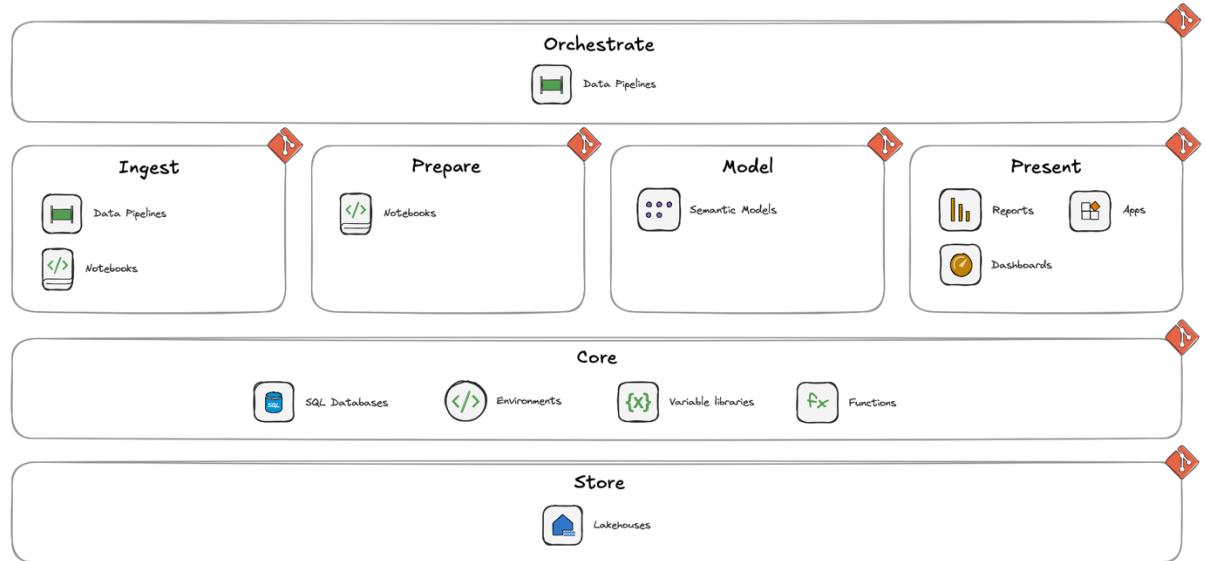
## Authentication options



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

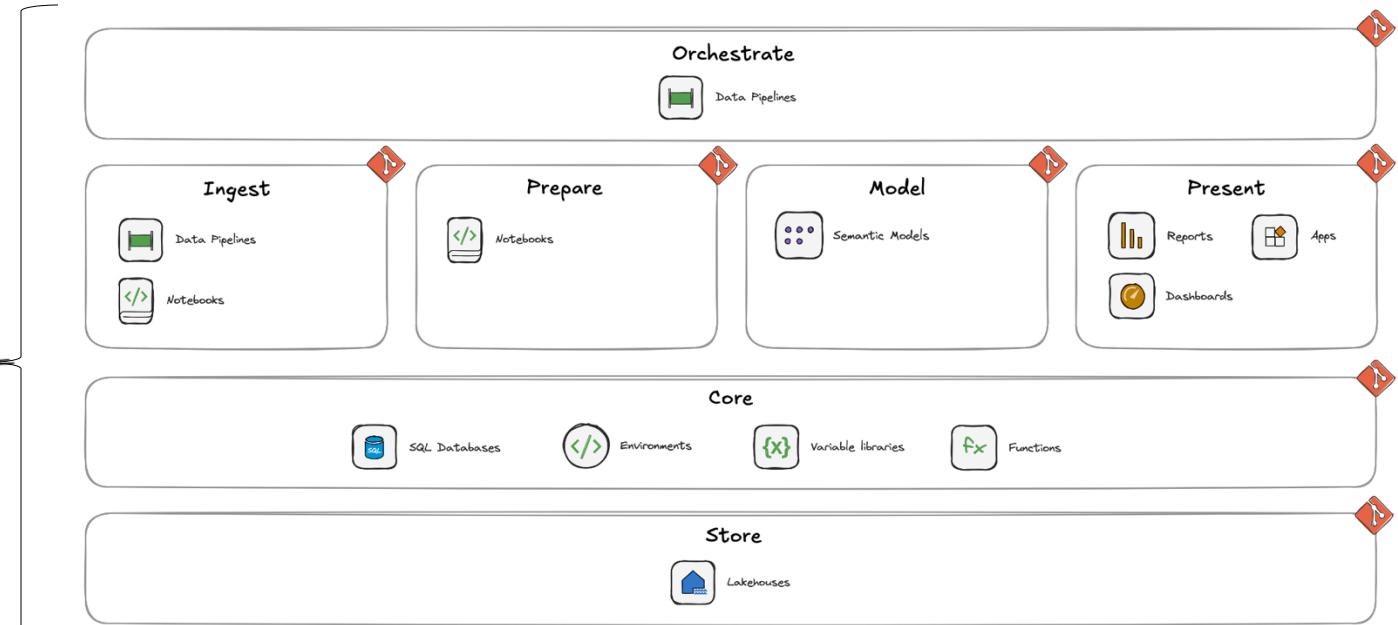


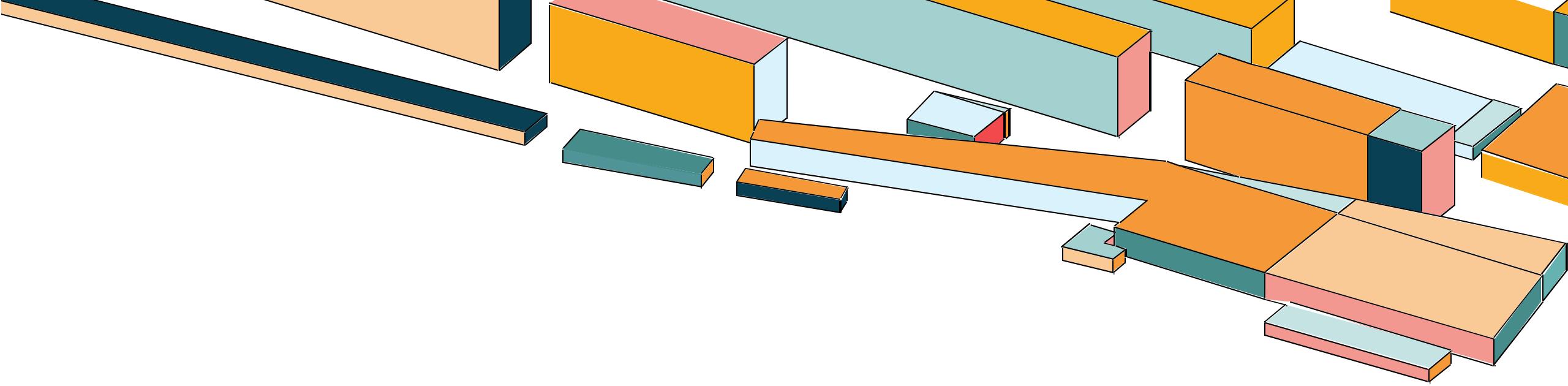
# A GO-TO FOR FABRIC GIT INTEGRATION



## Git Repo structure

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





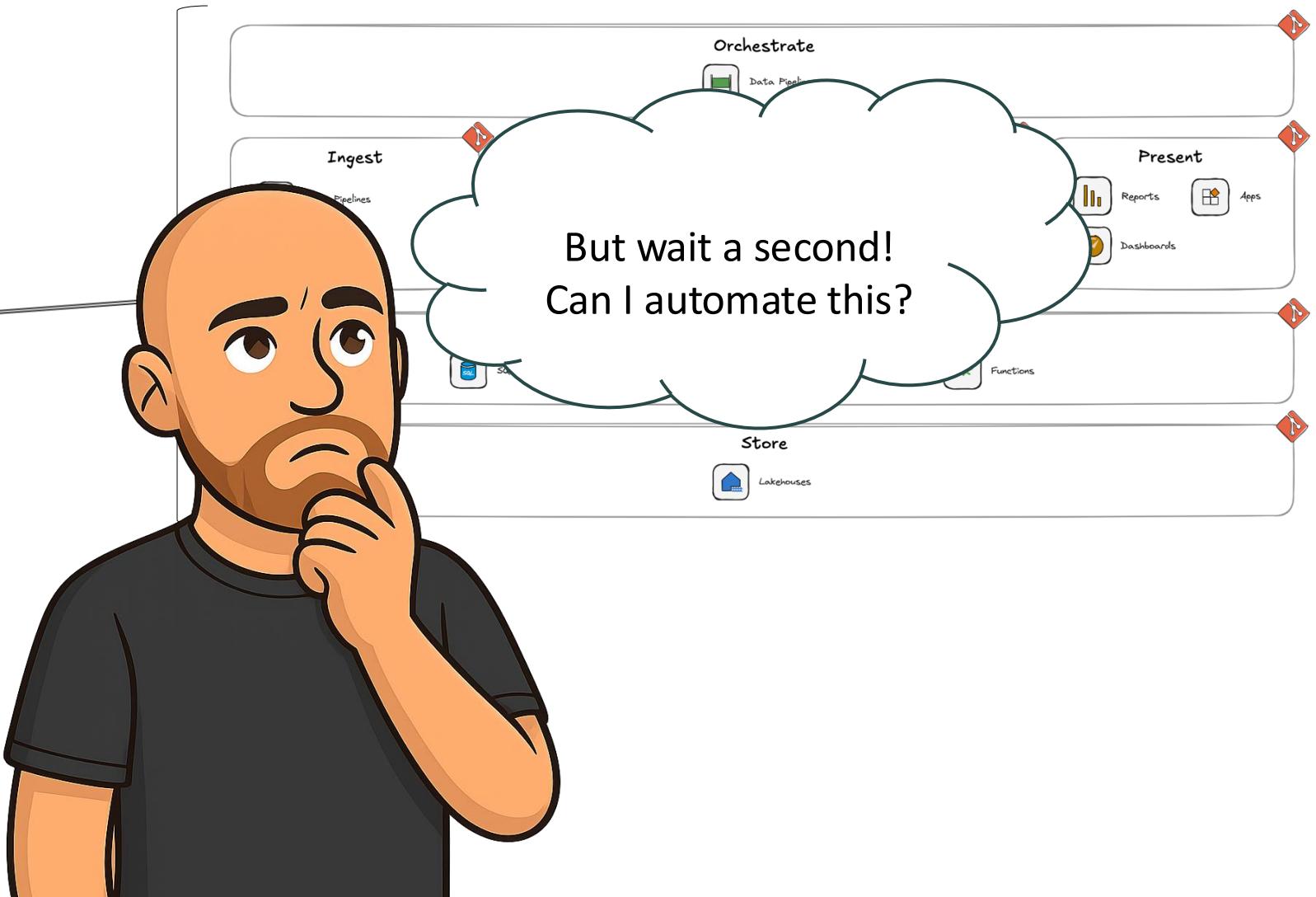
**DEMO**

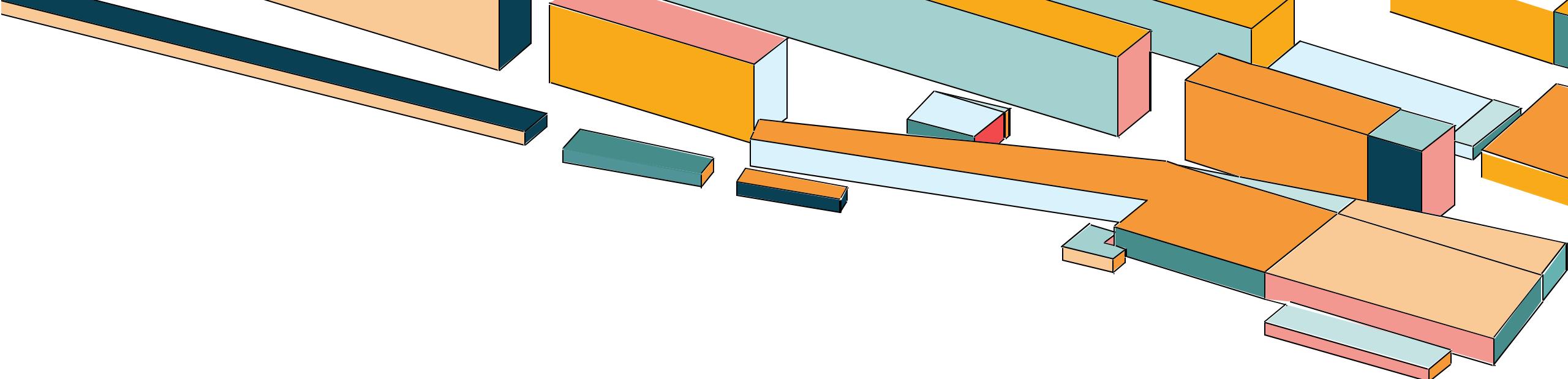
# A GO-TO FOR FABRIC GIT INTEGRATION



## Git Repo structure

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





**BREAK**

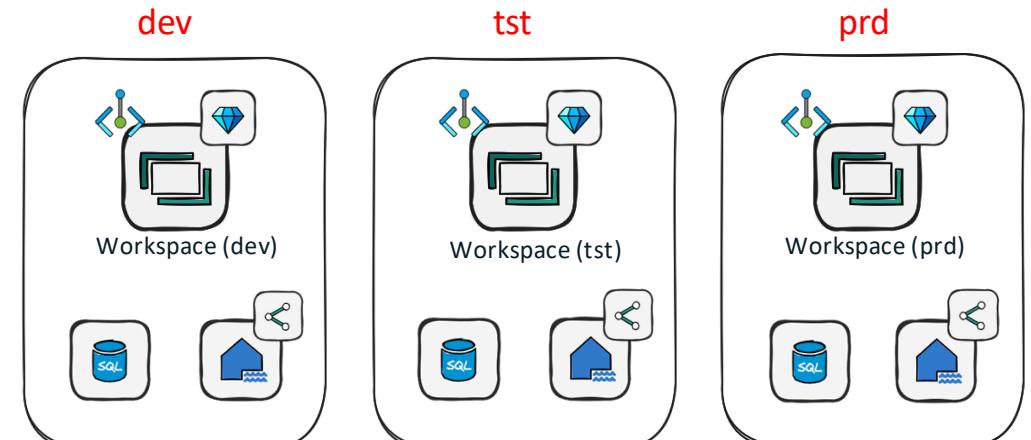


# FABRIC IAC

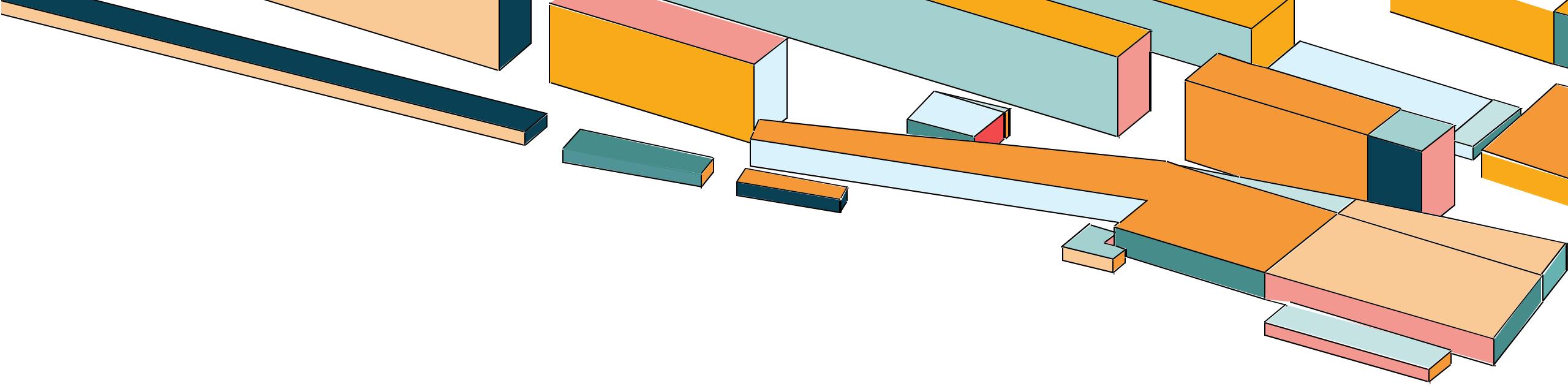


# AUTOMATING FABRIC SOLUTION SETUP

- 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



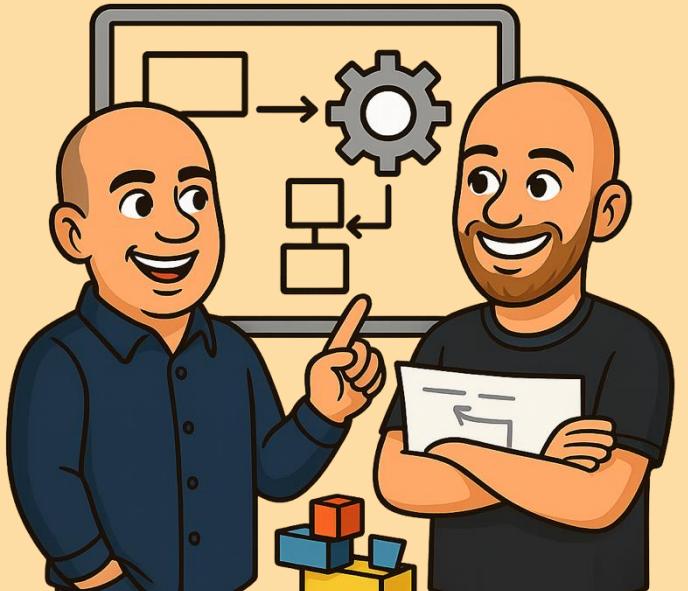
Download source code: <https://github.com/gronnerup/FabricAutomation>



**DEMO**

The background of the slide features a collection of 3D rectangular blocks of various sizes and colors, including orange, teal, light blue, and white, suspended in the air against a plain white background. The blocks are arranged in a loose, overlapping cluster, with some appearing to float higher than others. The lighting creates soft shadows and highlights on the edges of the blocks, giving them a sense of depth and volume.

# WAYS-OF-WORKING

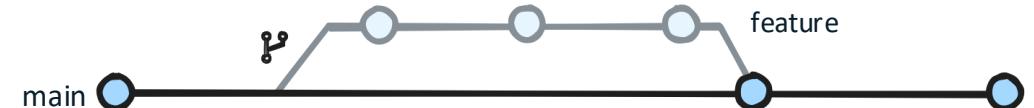


# BRANCH FOR REAL-WORLD COLLABORATION



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

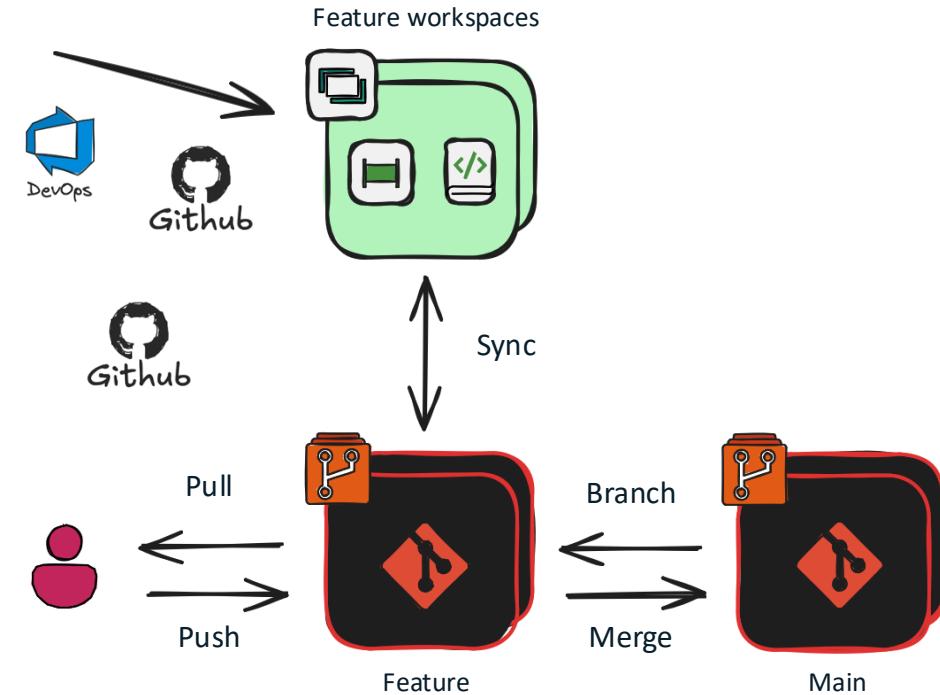
- Git is the source of truth, and branching is how we manage change
- Branches give developers a separate workspace for their code and...
  - Isolates development
  - Protects the mainline of our code
  - Enables parallel development
  - Helps organize and structure releases
  - Is crucial for streamlined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



# THE BRANCHING FLOW

Development flow - Supported by automation!

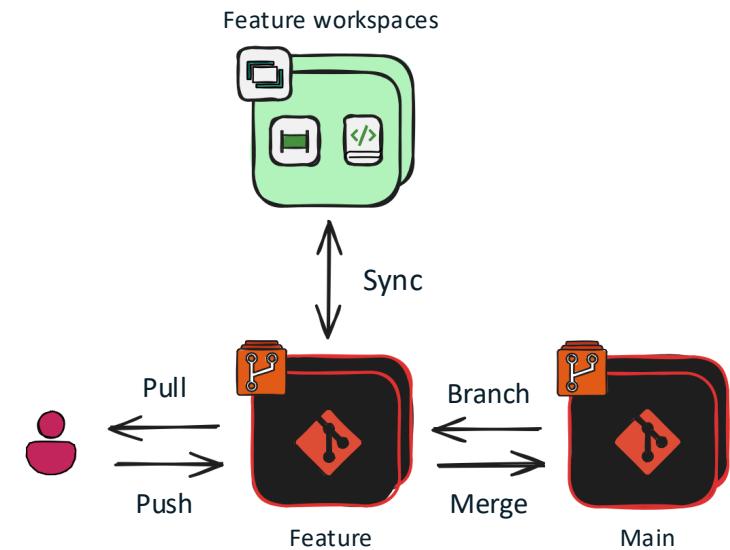
1. Create branch from main
2. Develop
3. Create PR
4. Merge feature to main
5. Delete feature branch



# THE BRANCHING FLOW – AUTOMATED

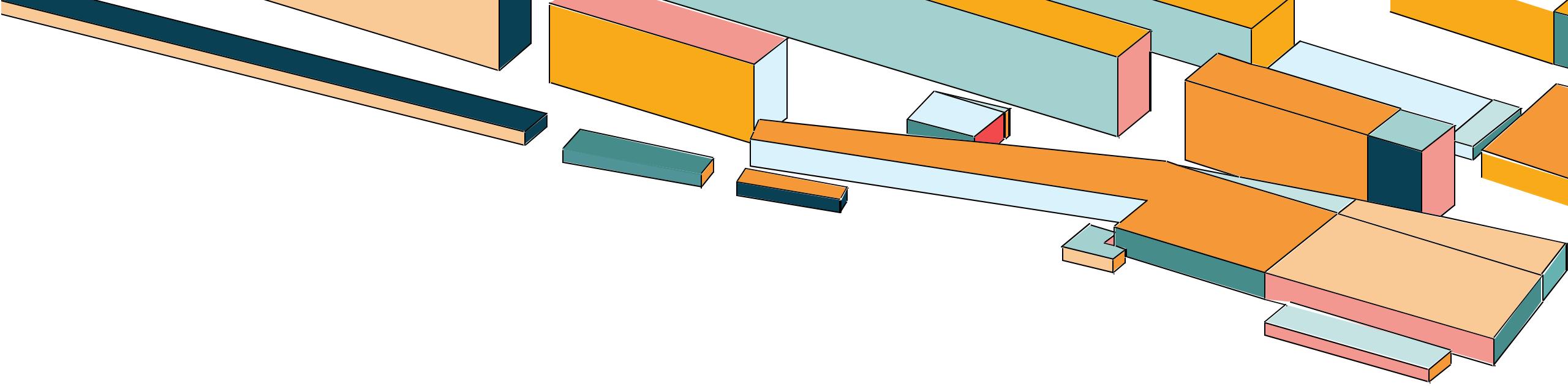
Why automate this?

ClickOps	Automated
Manual process	Fully automated - just create the branch
1 workspace – 1 branch	Support for multiple workspaces
No transfer/setup of:	Customize:
- ACL - Spark settings - Private Endpoints - WS Identity	- 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>

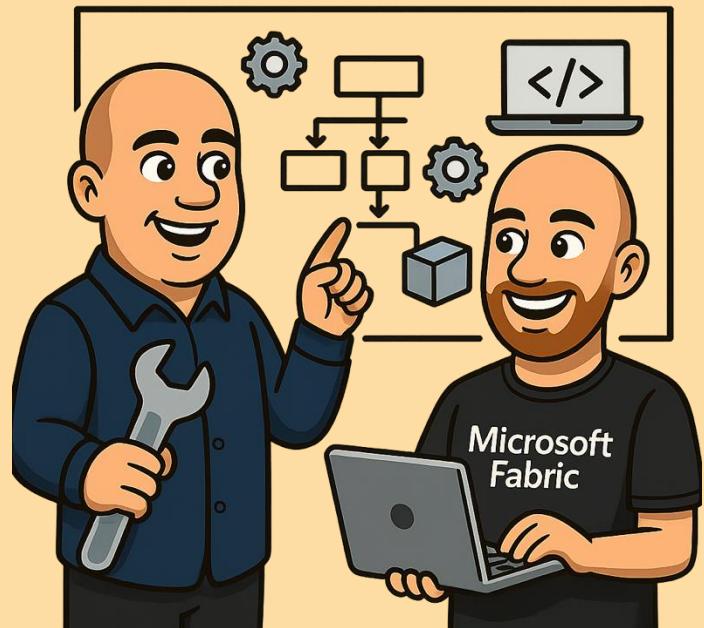
<https://justb.dk/blog/2025/02/fabric-spark-notebooks-and-cu-consumption/>

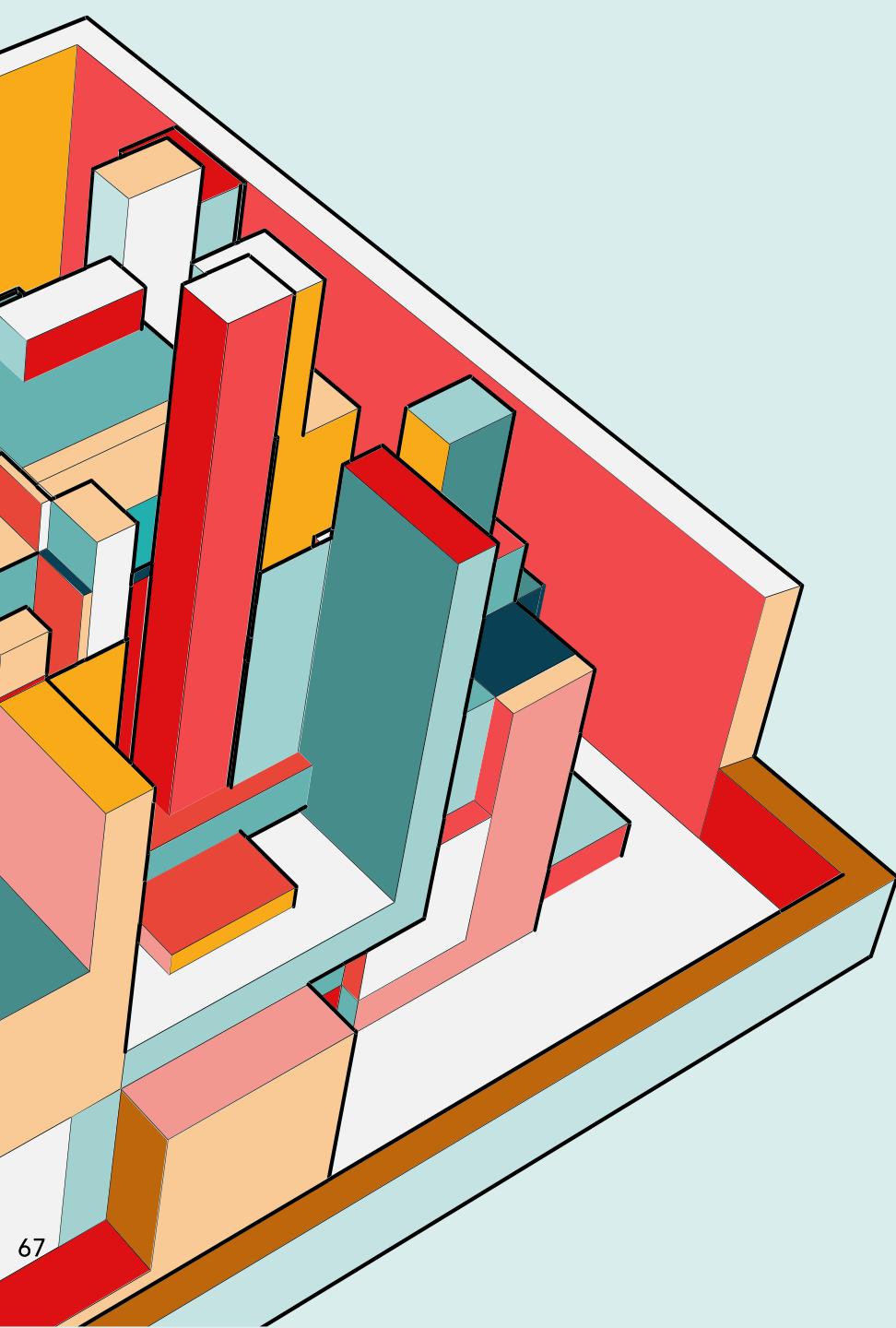


**DEMO**

The background of the slide features a collection of 3D rectangular blocks of various sizes and colors, including orange, teal, light blue, and white, suspended in the air. These blocks are arranged in a loose, overlapping cluster, creating a sense of depth and motion. The overall aesthetic is clean and modern, with a focus on geometric shapes and color.

# BUILDING OUR SOLUTION



A collection of colorful, 3D geometric shapes (cubes, rectangular prisms) in various sizes and colors (red, blue, yellow, orange, pink) are arranged within a perspective frame. The frame has a red background and white borders. The shapes are scattered throughout the frame, some standing upright, some tilted, creating a sense of depth and volume.

# 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

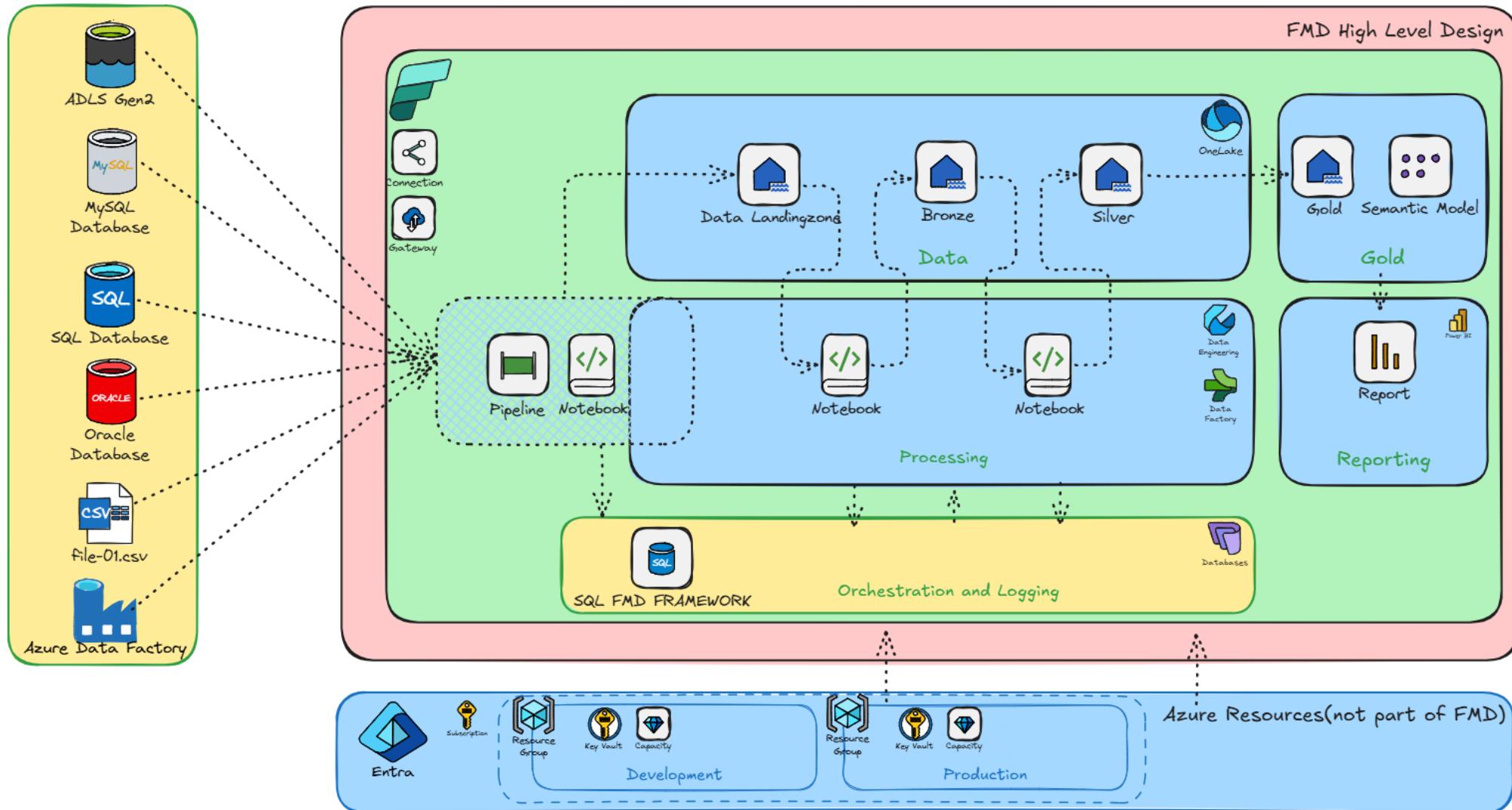
Scalability and  
Flexibility

Cost Efficiency

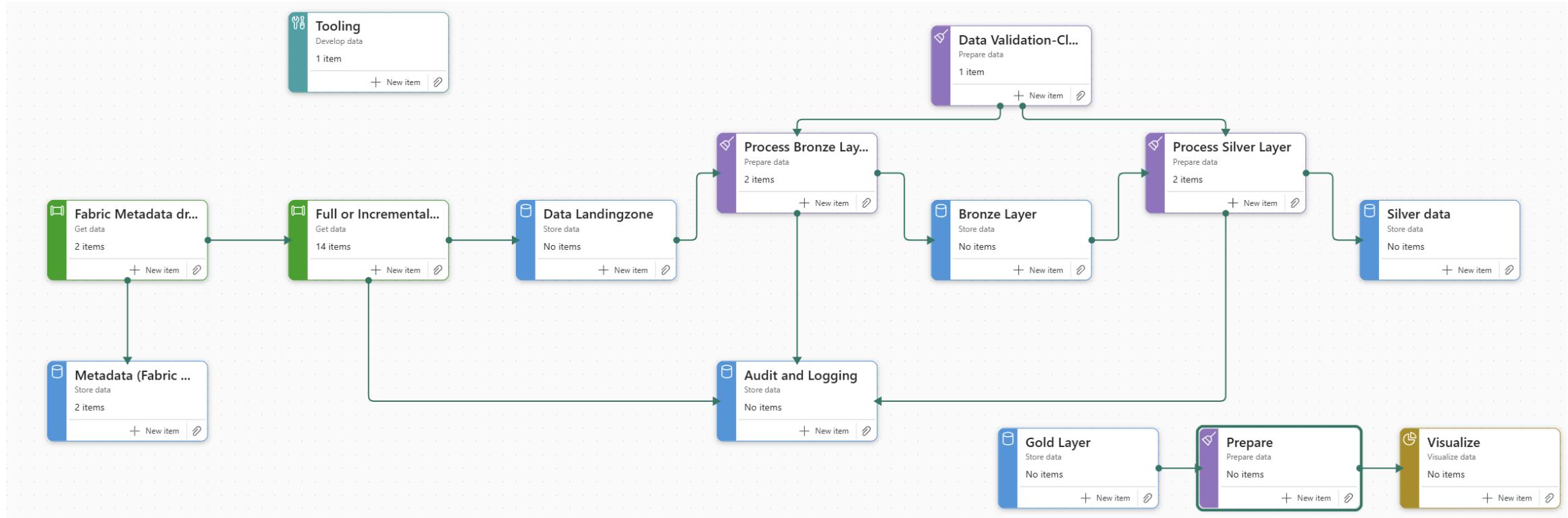
Streamlined Data  
Integration

Enhanced Data  
Governance

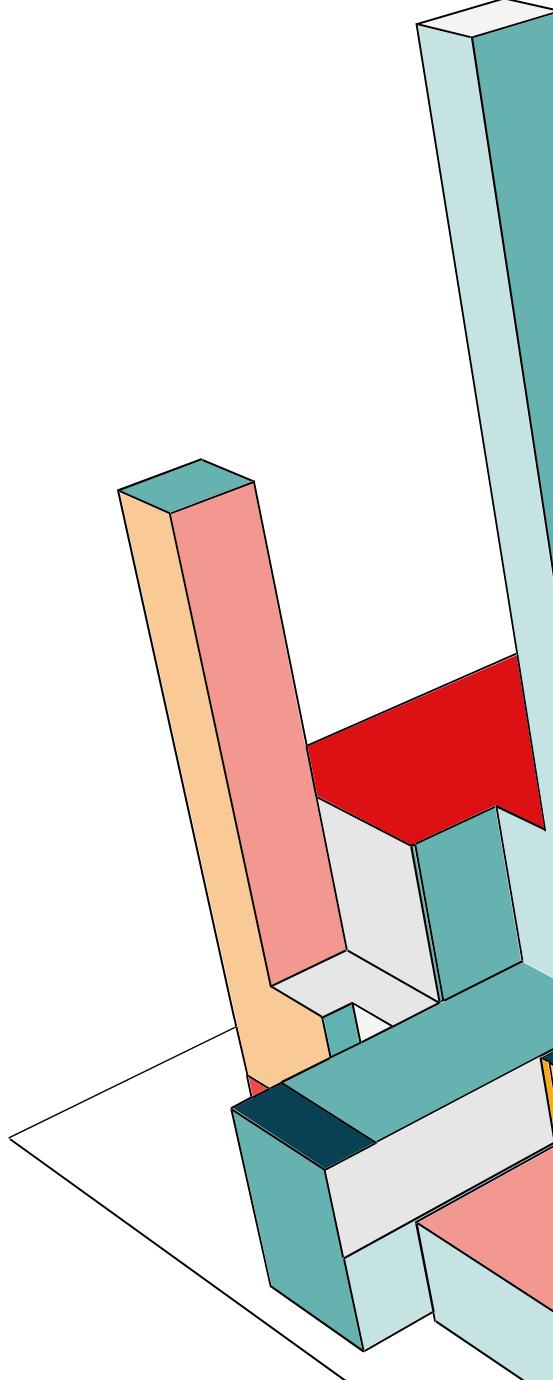
# FABRIC METADATA DRIVEN FRAMEWORK



# METADATA-DRIVEN DATA INGESTION FRAMEWORK COMPONENTS



# LET'S GET STARTED



# CONNECTIONS (REQUIRED)

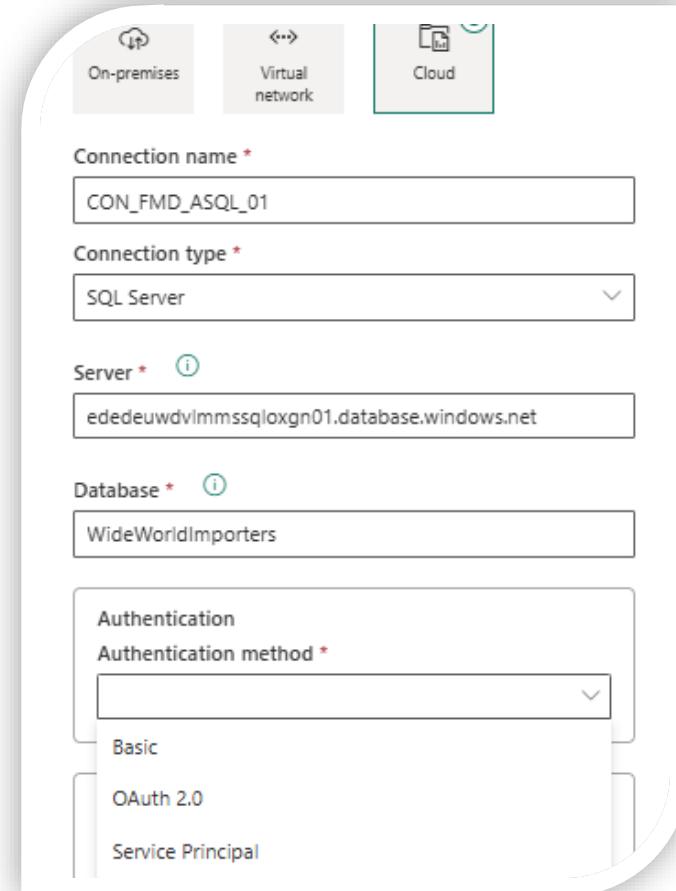
- Required Connections
  - Fabric Data Pipelines
    - Authentication Oauth
      - Workspace Identity is Coming
      - Service Principal is also an option
    - Fabric SQL database
      - Authentication Oauth
        - Service Principal is also an option

All other connections are based on your configuration

The image displays two side-by-side screenshots of a cloud-based connection configuration interface. Both screenshots show the 'Cloud' connection type selected. The left screenshot is for a 'Fabric SQL database' connection, showing 'OAuth 2.0' as the authentication method and 'Organizational' as the privacy level. The right screenshot is for 'Fabric Data Pipelines', also showing 'OAuth 2.0' as the authentication method and 'Organizational' as the privacy level. Both screenshots include a checkbox for utilizing on-premises or VNet data gateways.

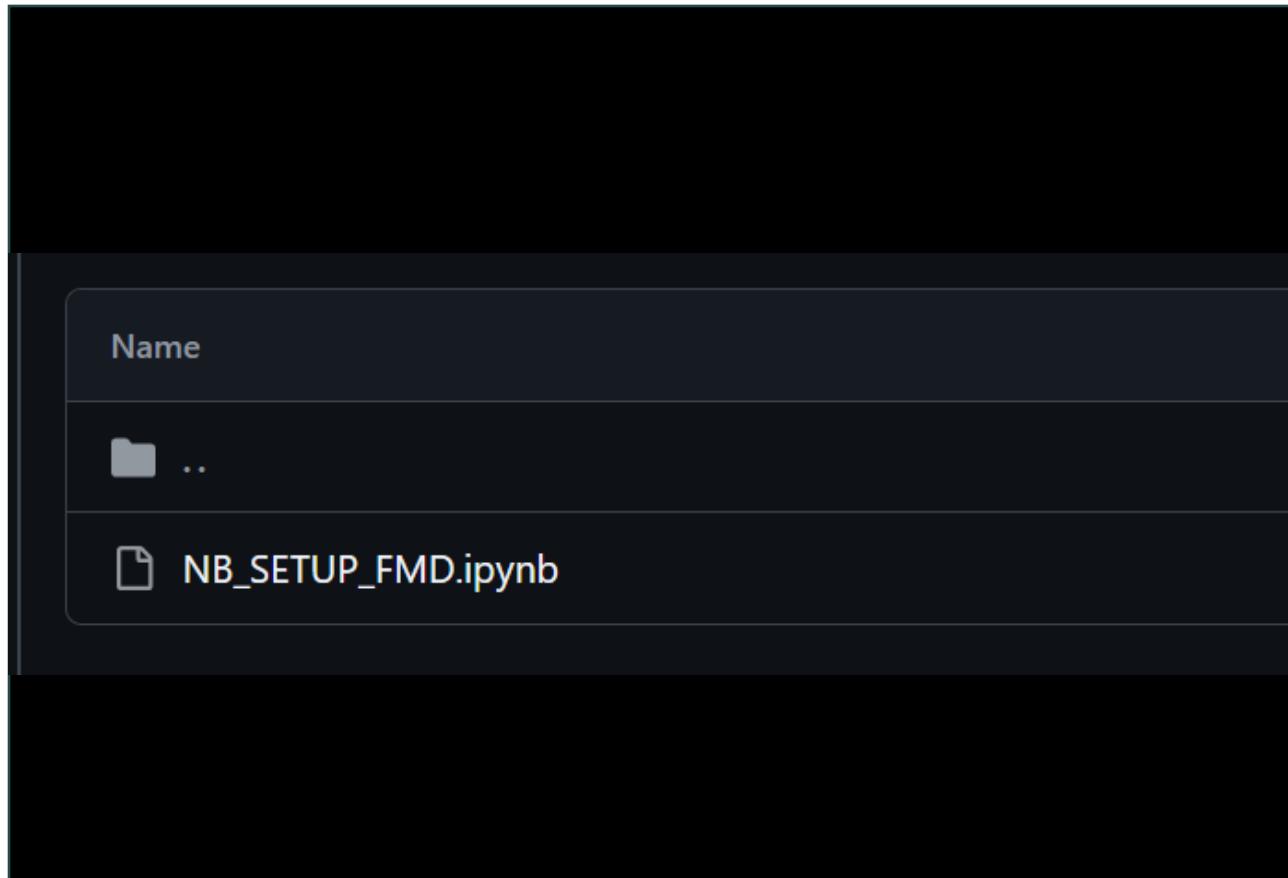
# SUPPORTED SOURCES (CURRENTLY)

- 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



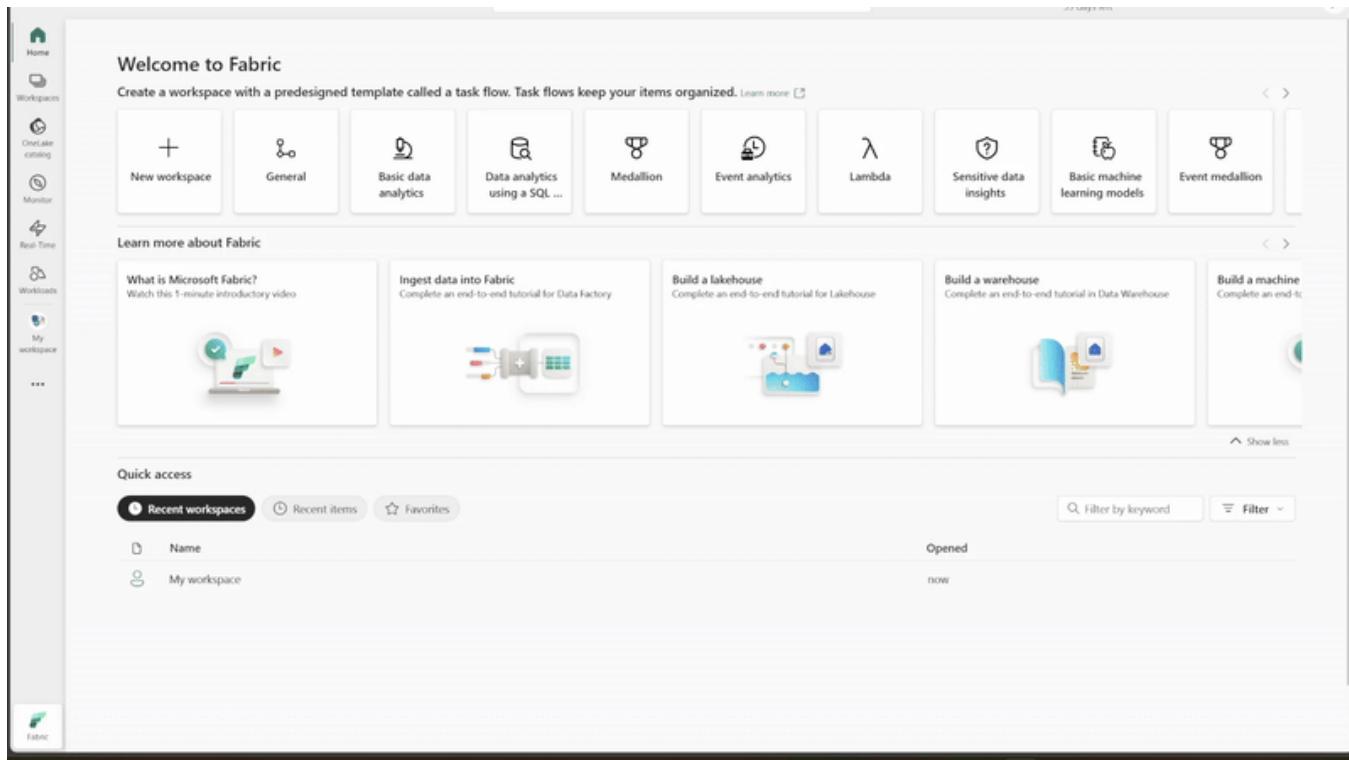
# FABRIC METADATA DRIVEN FRAMEWORK

- Download Notebook from Github



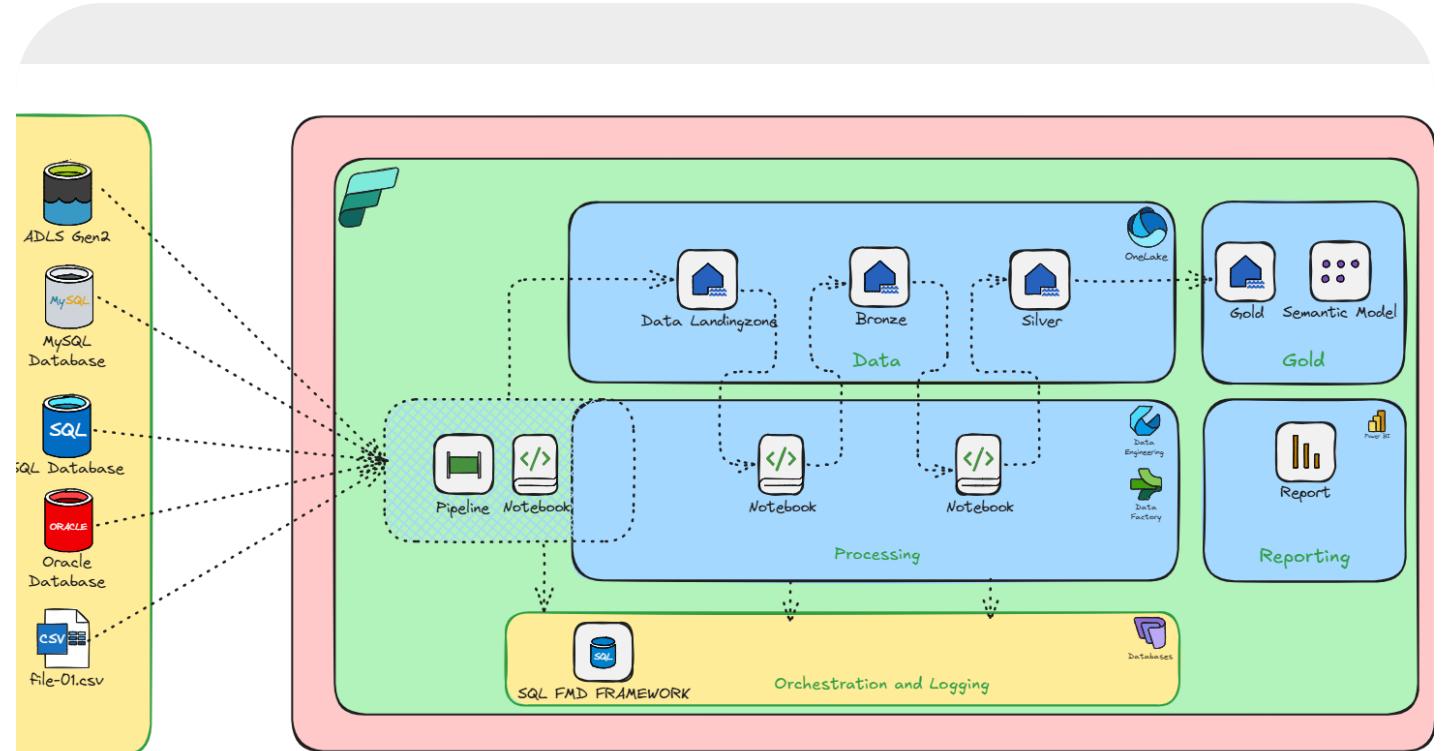
# FABRIC METADATA DRIVEN FRAMEWORK

- Download Notebook from Github
- Create Workspace Name is up to you:
  - FMD\_XXX\_CONFIGURATION
- Assign a capacity
- Import Notebook to this Workspace



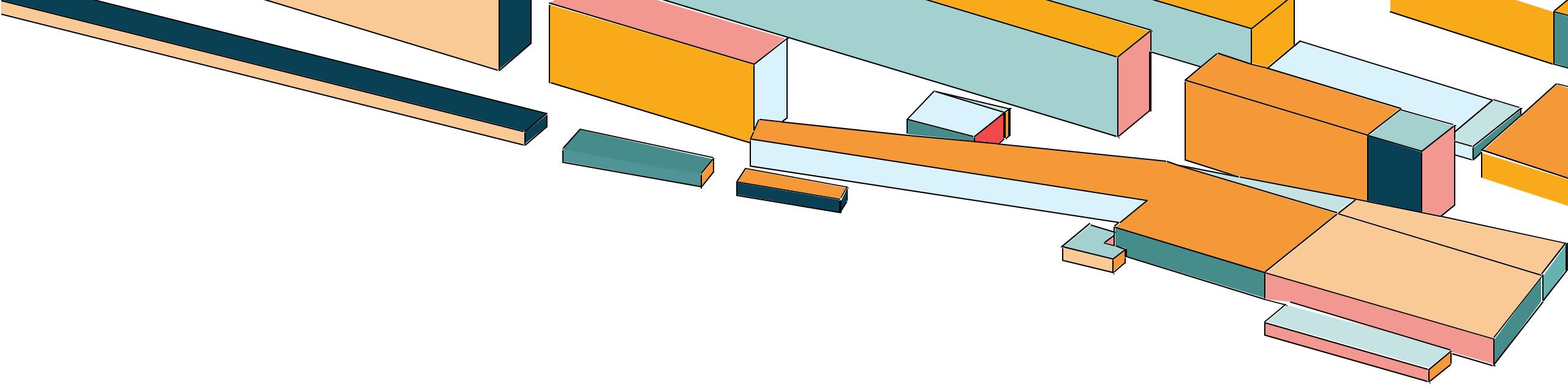
# FABRIC METADATA DRIVEN FRAMEWORK

- Open Notebook
- NB\_SETUP



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

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



**DEMO**

The background of the slide features a collection of 3D rectangular blocks of various sizes and colors, including orange, teal, light blue, and pink. These blocks are arranged in a non-linear, overlapping fashion, creating a sense of depth and movement. Some blocks appear to be floating in the air, while others are positioned closer to the viewer. The overall effect is a minimalist yet visually interesting abstract design.

# DEPLOYMENT



# RELEASE PROCESS – THE MAIN OPTIONS...

Fabric Deployment pipelines	Git based deployment	Git based deployment using build environment
<ul style="list-style-type: none"><li>• No code experience</li><li>• For simpler solutions</li><li>• No support for:<ul style="list-style-type: none"><li>• Pre-deployment opr.</li><li>• Post-deployment opr.</li><li>• Test &amp; validation</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Suitable when using Gitflow</li><li>• Each environment connected to git branch</li><li>• No need for building environments</li><li>• Might require post-deployment operations</li></ul>	<ul style="list-style-type: none"><li>• Supports building environments</li><li>• Deployment through ADO Pipelines/Github actions</li><li>• Supports manipulation of item definitions</li></ul>

# CI/CD BUILDING BLOCKS

## Tools & automation options

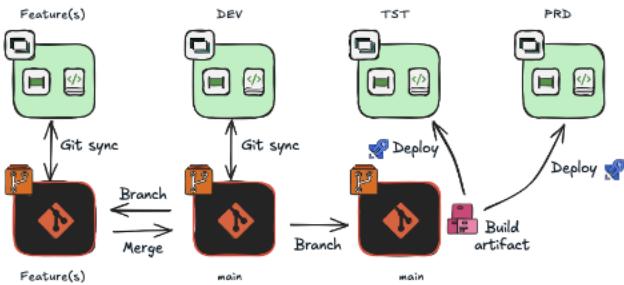
-  Azure DevOps Pipelines
-  Github actions
-  Python scripts & wrappers
-  Fabric CLI wrappers

## fabric-cicd Python Library

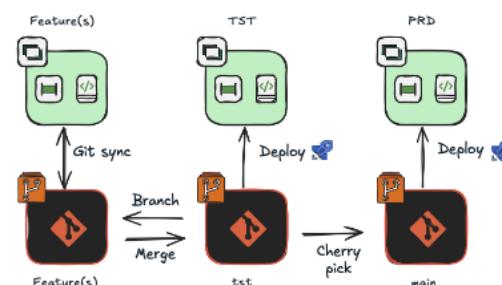
-  Developer friendly
  - Code-first approach
  - No need to call Fabric APIs directly
-  Environment supports
  - Parameterization
  - Reusable for locale, DevOps and more...
-  Smart deployment
  - Deploy all supported items (with public APIs)
  - Auto-unpublish orphaned artifacts

# INTRODUCING 3 RELEASE SCENARIOS

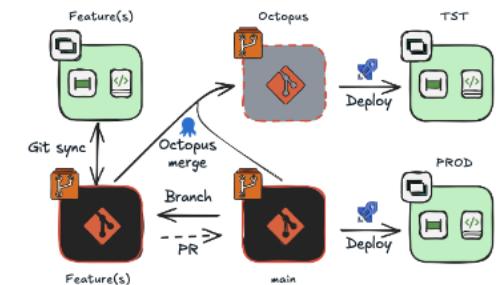
## Scenario 1 The simple way



## Scenario 2 Pick and choose

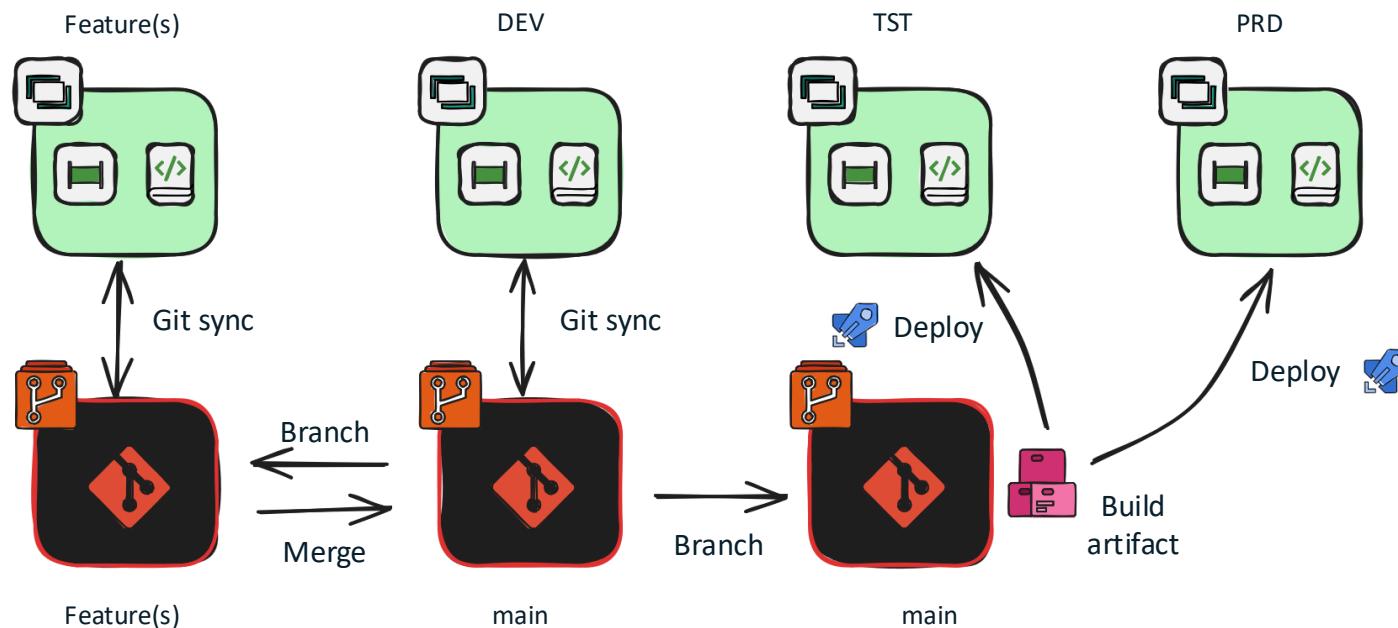


## Scenario 3 The selective release option



# RELEASE PROCESS – 3 SELECTED SCENARIOS

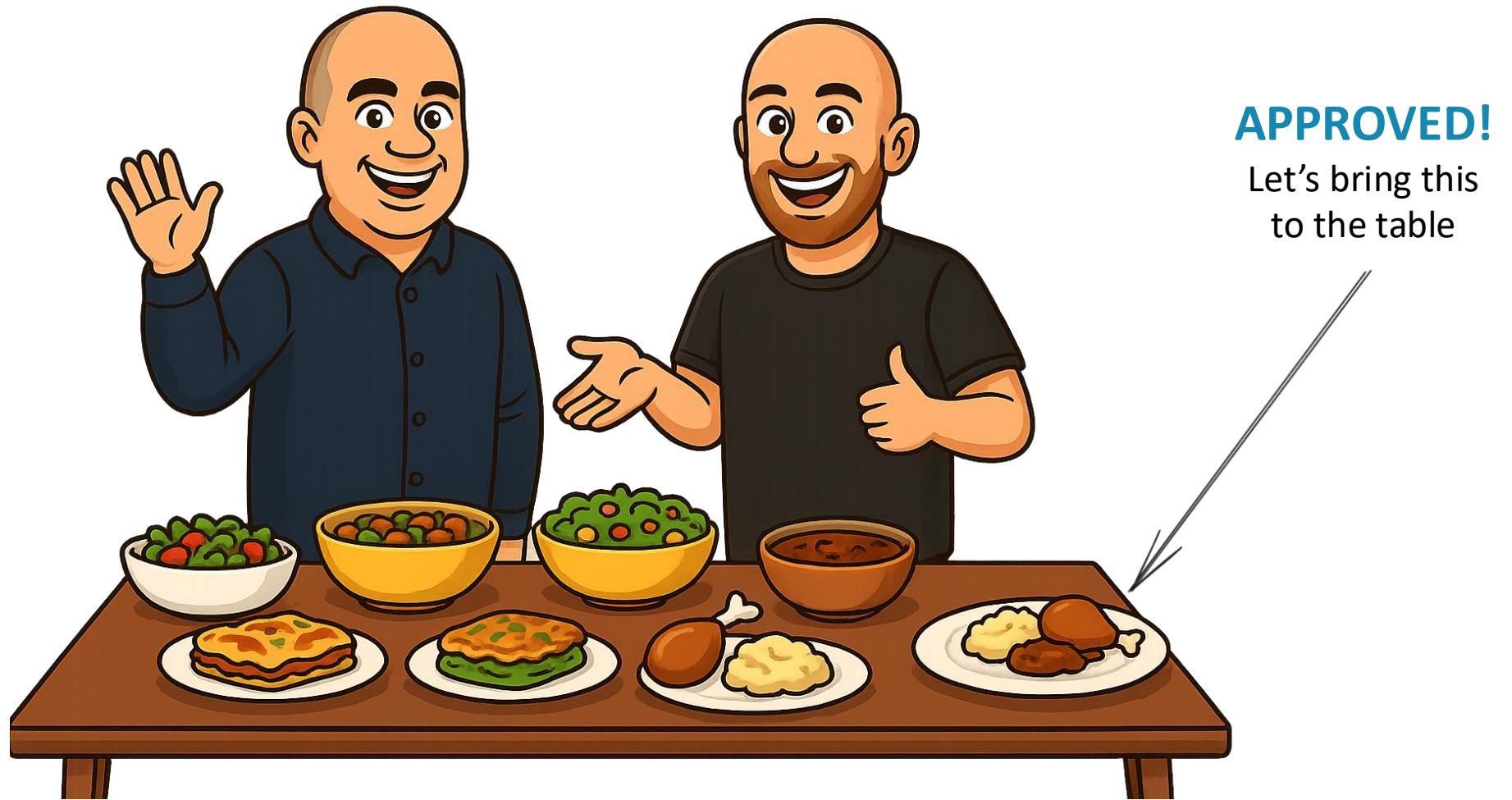
## Scenario 1 – Main is always dev



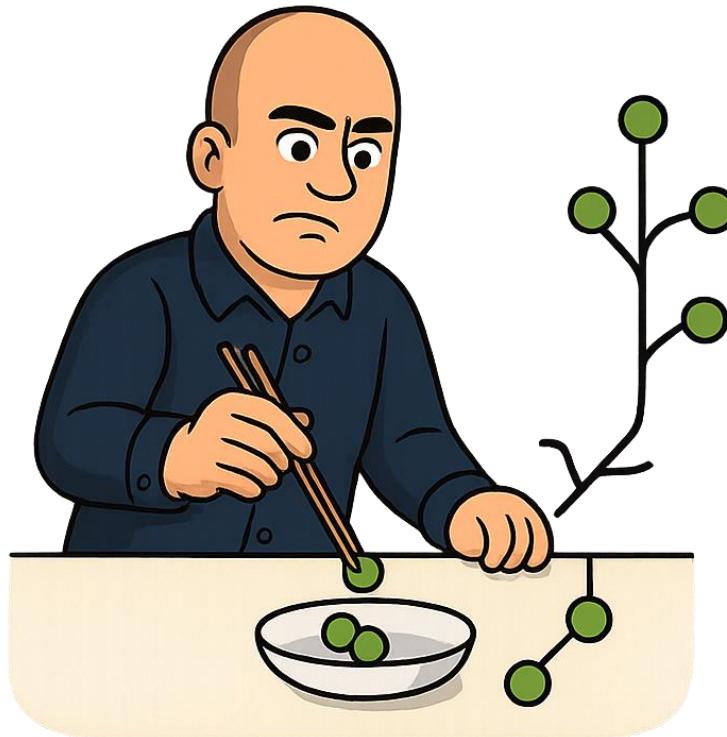
### Suitable for:

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

# SO WHY INTRODUCE ALTERNATIVES?



# RELEASE PROCESS – 3 SELECTED SCENARIOS



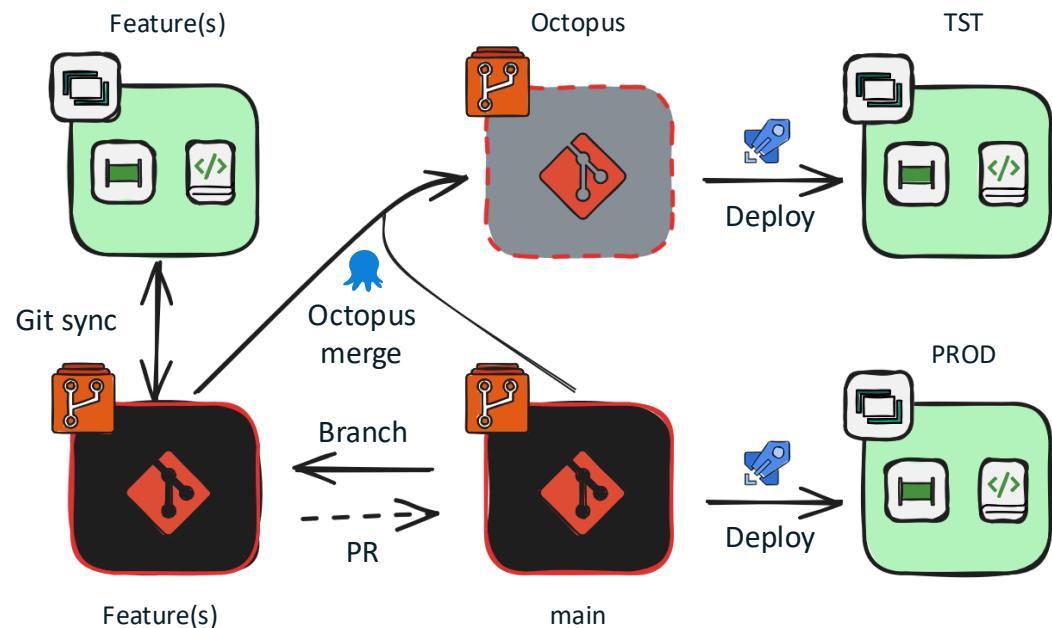
**Cherry-pick  
into main**



**Octopus merge**

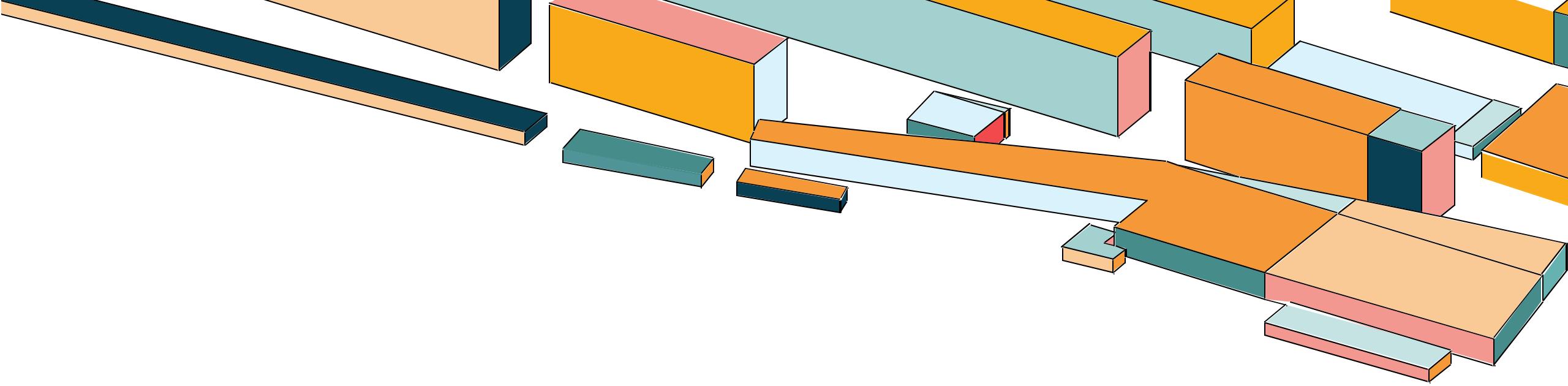
# RELEASE PROCESS – 3 SELECTED SCENARIOS

## 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**

The background of the slide features a collection of 3D rectangular blocks of various sizes and colors, including orange, teal, light blue, and white, arranged in a non-linear, overlapping fashion. These blocks appear to be suspended in a white space, creating a sense of depth and volume. The colors are vibrant and provide a modern, minimalist aesthetic.

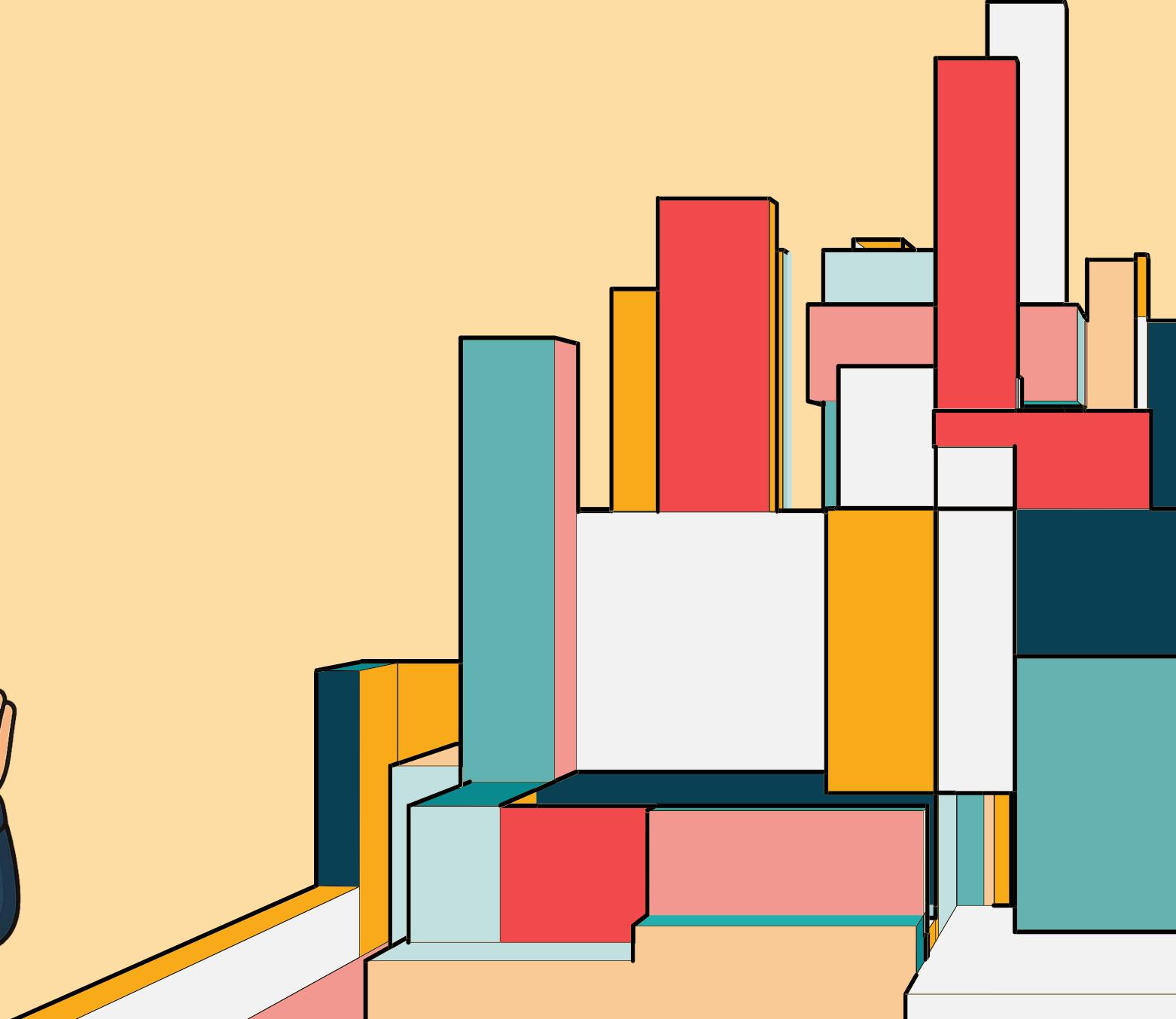
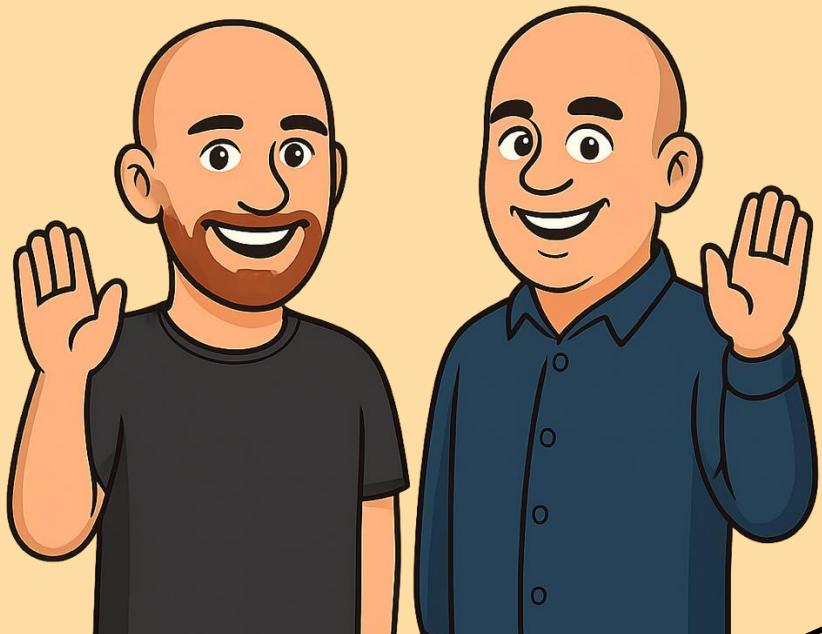
# BEST PRACTICES, TIPS AND TRICKS



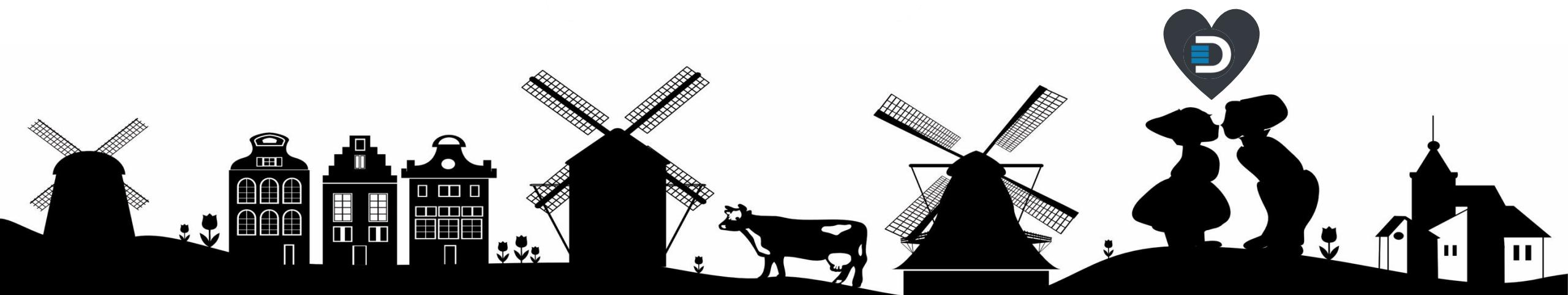
# BEST PRACTICES, TIPS AND TRICKS

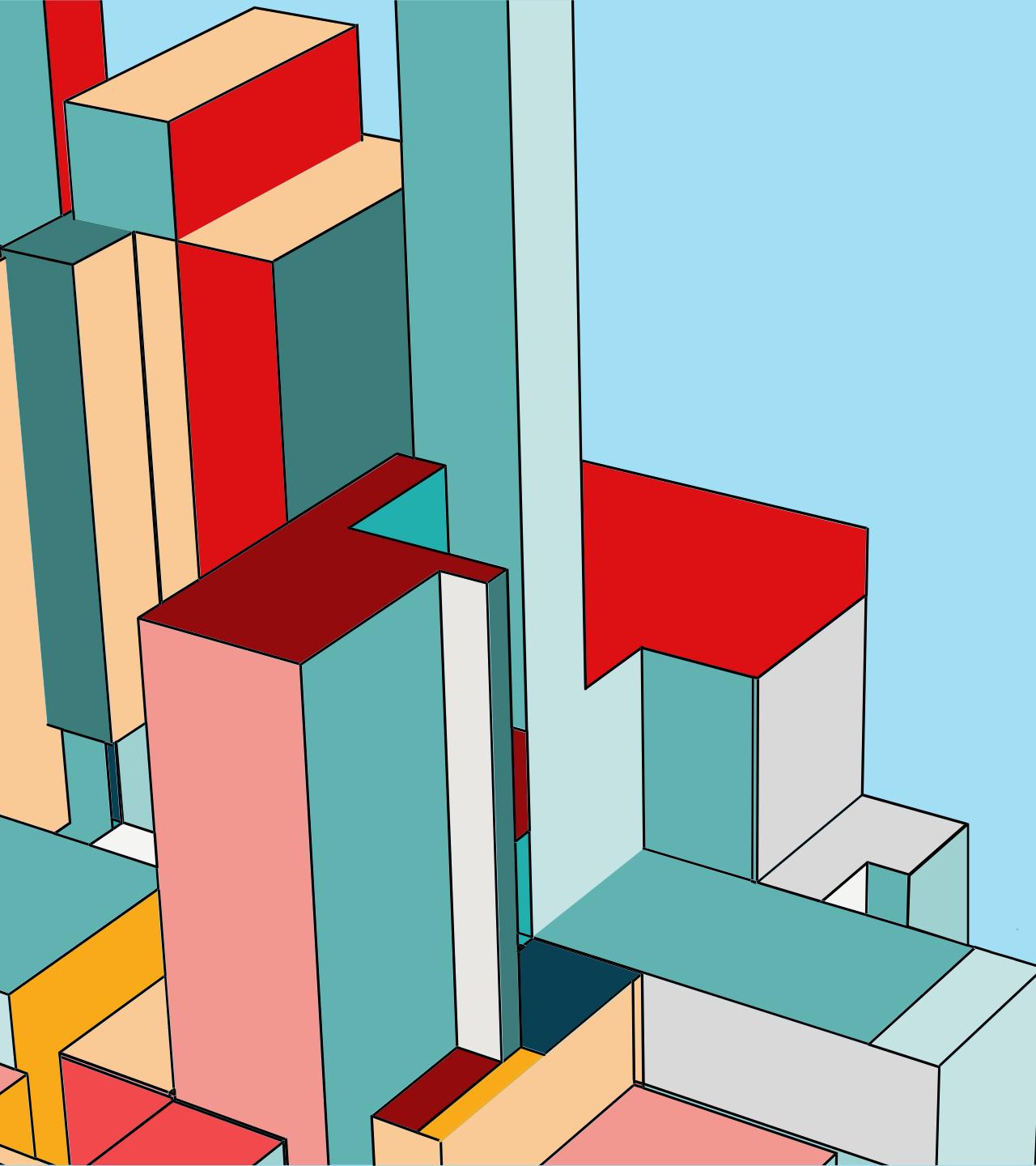
- Split your workloads and layers across multiple workspace
- Use mono-repo structure as a starting point
- Leverage the Fabric CLI, REST APIs, and/or Terraform for automation
- Use the fabric-cicd Python library to deploy Fabric items
- Design everything with automation in mind
  - Invest in dynamic 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

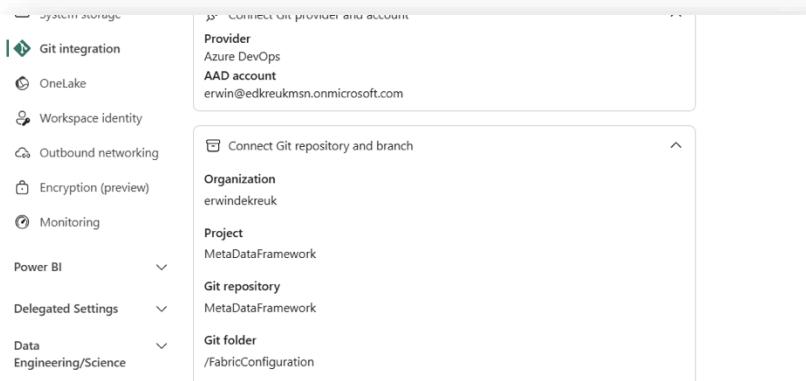
## Continuous Deployment

Working in short cycles (often Agile)

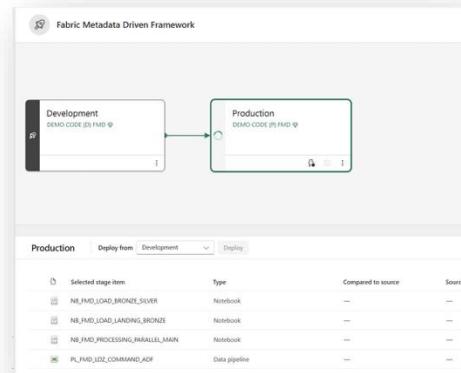
High frequency updates and releases  
Repeatable deployment process

# FABRIC CI/CD PLATFORM

Built-in git integration



Deployment pipelines



Fabric REST APIs

The screenshot shows a POST request to the Fabric REST API endpoint: `POST https://api.fabric.microsoft.com/v1/connections`. The request body is a JSON object:

```
{  
  "connectivityType": "ShareableCloud",  
  "displayName": "MyGitHubPAT",  
  "connectionDetails": {  
    "type": "GitHubSourceControl",  
    "creationMethod": "GitHubSourceControl.Contents",  
    "parameters": [  
      {  
        "dataType": "Text",  
        "name": "url"  
      }  
    ]  
  }  
}
```

Variable Library

The screenshot shows the Fabric CI/CD Platform's Variable Library interface. It displays a table with columns: Note, Type, and Value. The variables listed are: "forkspaceGuid" (String type, value 40e27fdc-775a-4ee2-84d5-48893...), "ilt\_name" (String type, value keyvaultname), and "use\_schema\_enabled" (Boolean type, value true). To the right, there are sections for "Default value set" (Active) and "Alternative value sets". The "Default value set" contains "Prod" with values: 40e27fdc-775a-4ee2-84d5, keyvaultname, and true. The "Alternative value sets" section is currently empty.

# 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

## Workspace settings

⚙️ General

💎 License info

☁️ Azure connections

📁 System storage

### Git integration

🌐 OneLake

👤 Workspace identity

🔗 Outbound networking

🔒 Encryption (preview)

⌚ Monitoring

Power BI

Delegate Settings

Data

Engineering/Science

Data Factory

Data Warehouse

## Git integration

Connect to Git to manage your code and back up your work. [Learn more](#)

ⓘ Preview items Some item types are only available in preview when using Git. [Learn more](#) ↗

🔗 Connect Git provider and account

### Provider

Azure DevOps

### AAD account

erwin@edkreukmsn.onmicrosoft.com

🔗 Connect Git repository and branch

### Organization

erwindekreuk

### Project

MetaDataFramework

### Git repository

MetaDataFramework

### Git folder

/FabricConfiguration

### Branch \*

Develop

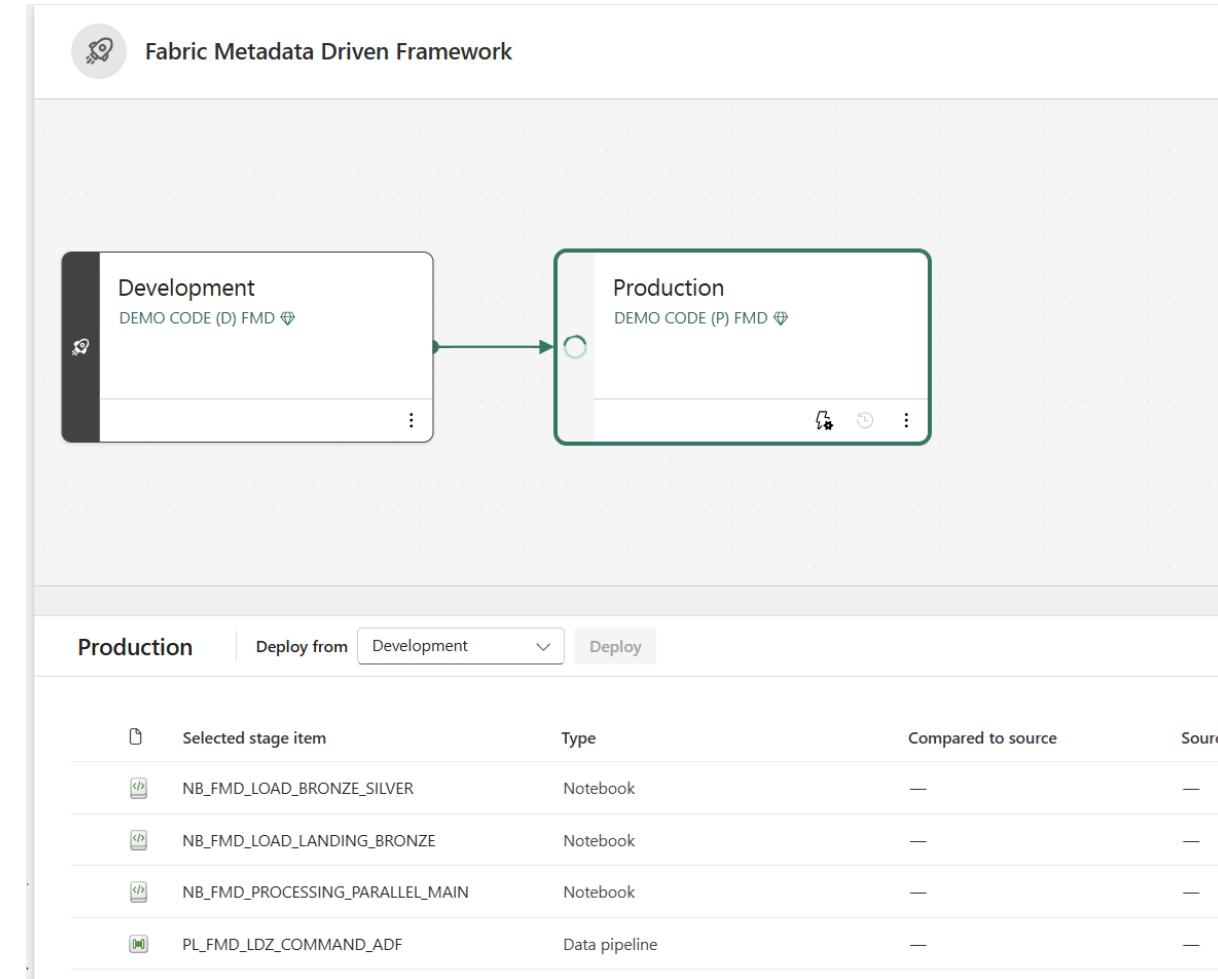
Switch branch

Cancel

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

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

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

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

# VARIABLE LIBRARY

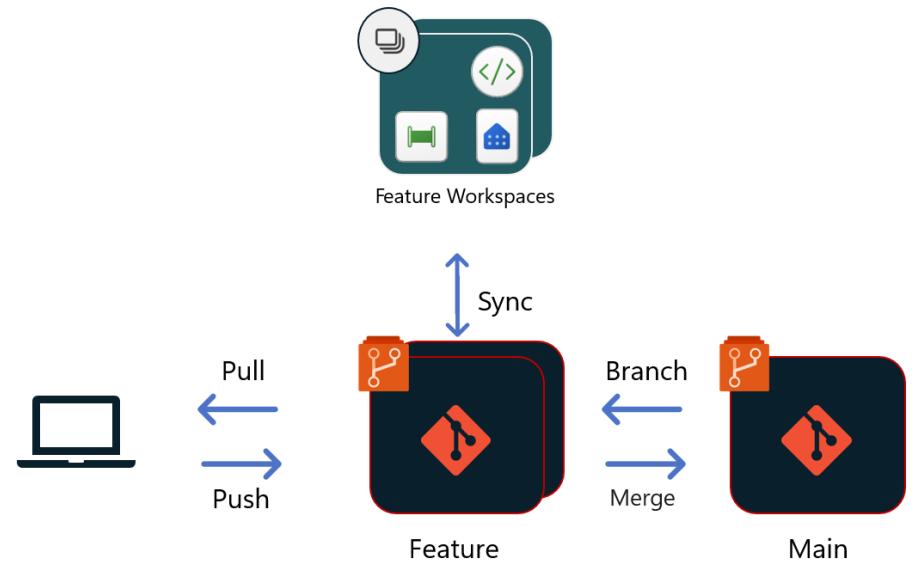
- New item in Fabric
- Central location to manage configurations across items in a workspace.
- Value sets -
  - Set values for different stages of your CI/CD process.
  - Apply them in relevant WSSs with a single API call/ click of a button.

VAR\_FMD

Variables		
<input type="checkbox"/> Name *	Note	Type ⓘ
<input type="text"/> Data_WorkspaceGuid	<input type="button"/>	String
<input type="text"/> key_vault_name	<input type="button"/>	String
<input type="text"/> Lakehouse_schema_enabled	<input type="button"/>	Boolean
Default value set * Active		
<input type="text"/> 40e27fdc-775a-4ee2-84d5-48893...		
<input type="text"/> keyvaultname		
<input type="text"/> true		
Alternative value sets		
Prod *		
<input type="text"/> 40e27fdc-775a-4ee2-84d5-48893...		
<input type="text"/> keyvaultname		
<input type="text"/> true		

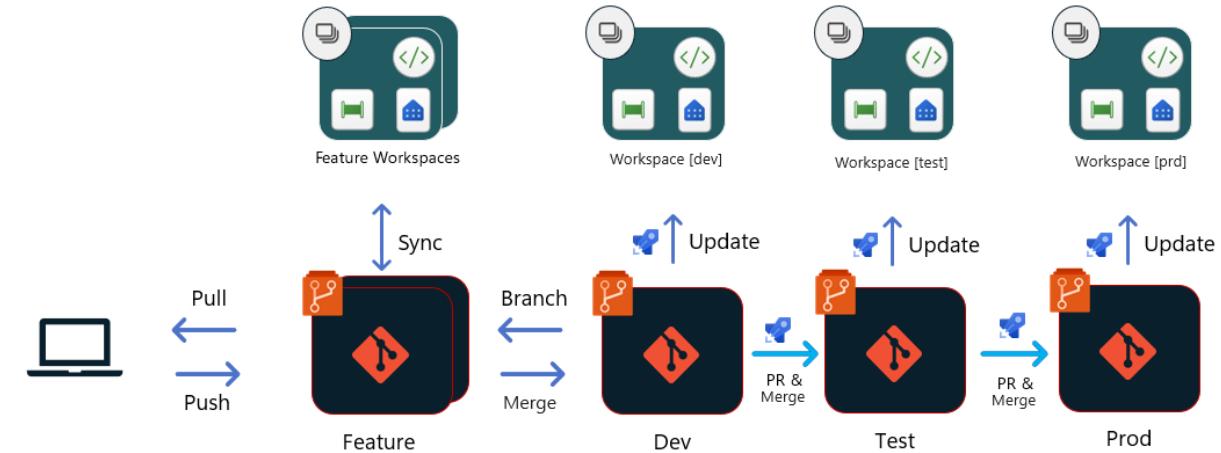
# DEVELOPMENT PROCESS

- 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..



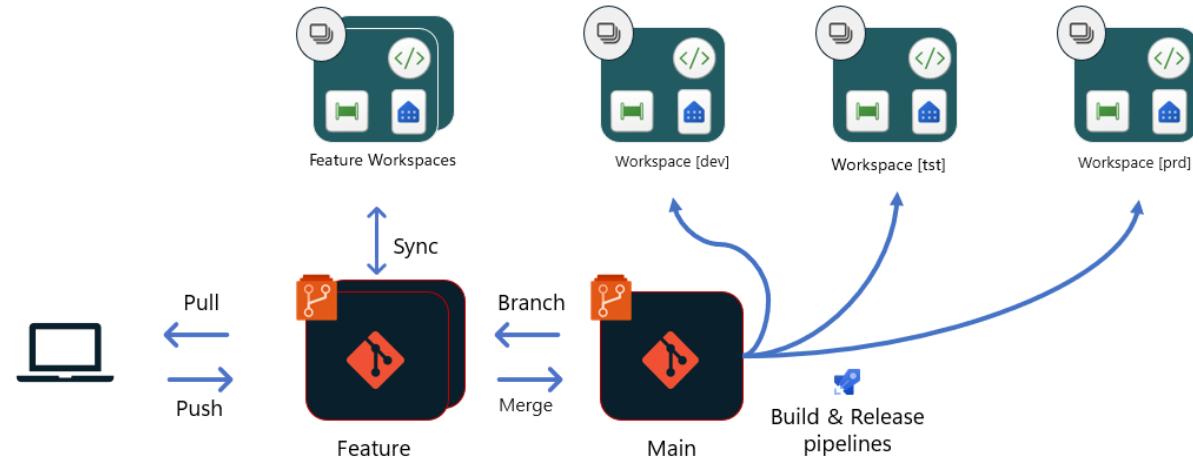
# GIT- BASED DEPLOYMENTS

- 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.



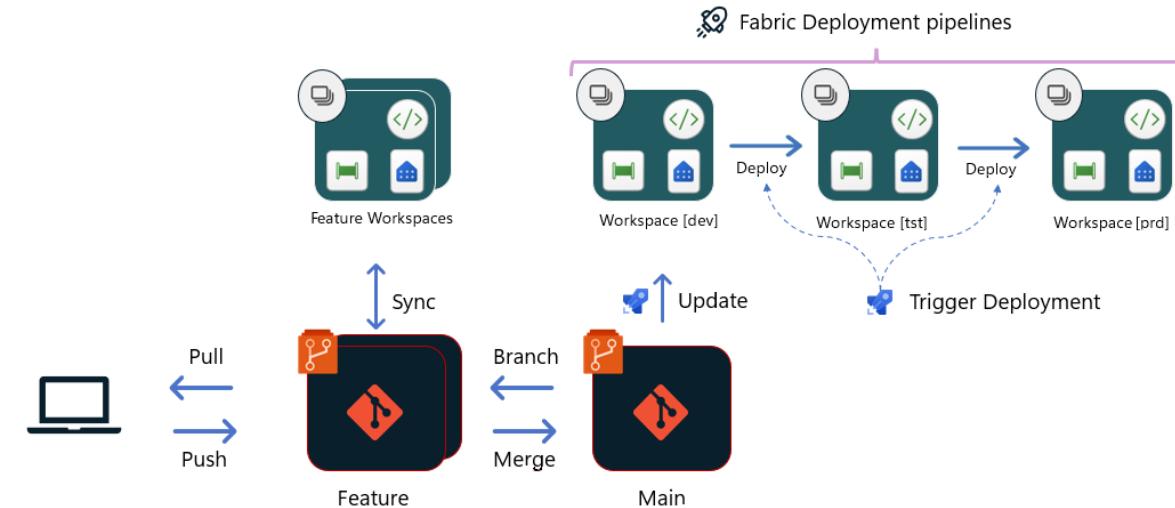
# GIT- BASED DEPLOYMENTS USING BUILD ENVIRONMENTS

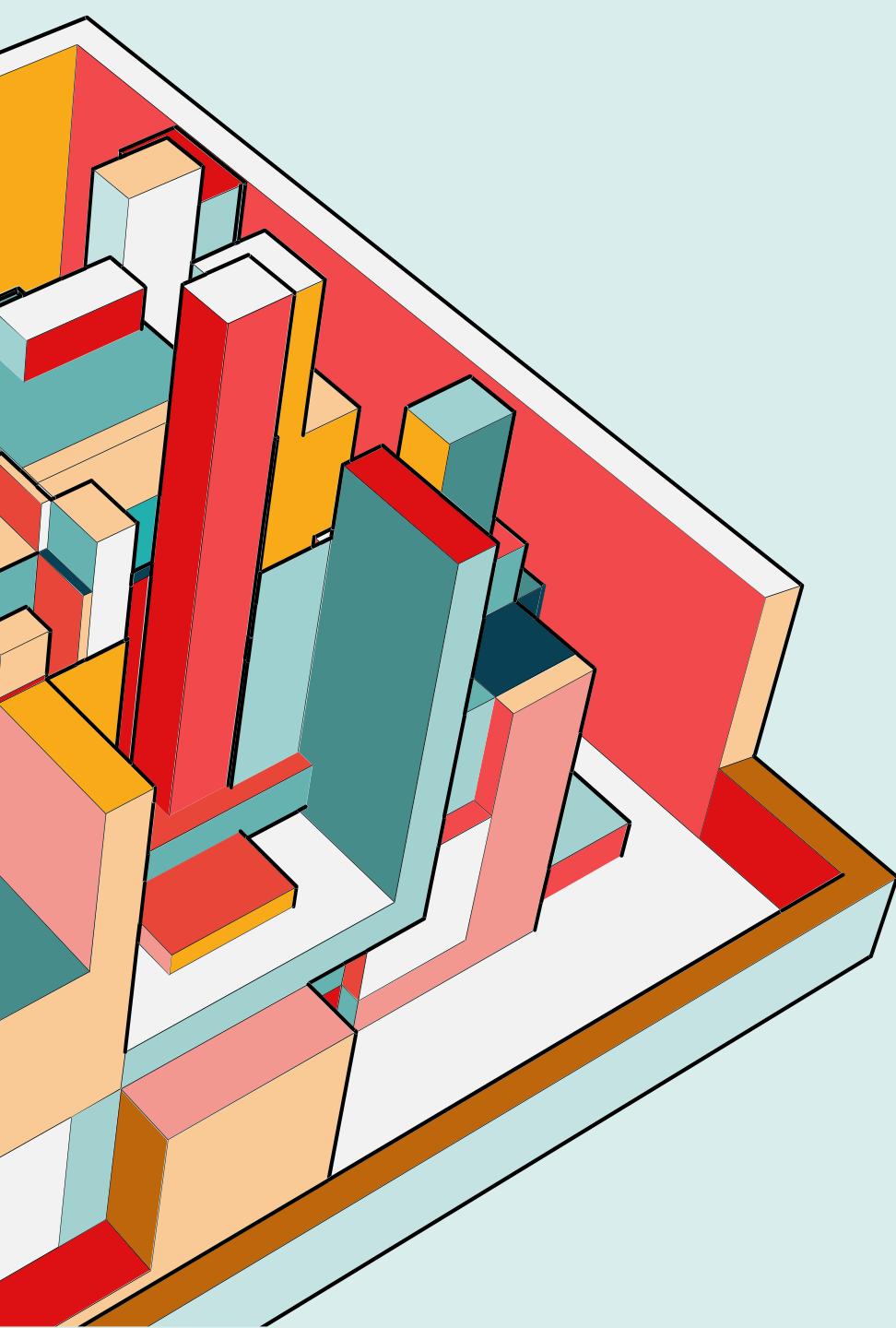
- Git as the single source of truth and origin of all deployments.
- Trunk-based development workflow followed by the team.
- Build environment required with custom scripts to modify workspace-specific attributes (e.g., connectionId, lakehouseld) before deployment.
- Custom release pipeline needed to:
  - 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](#)

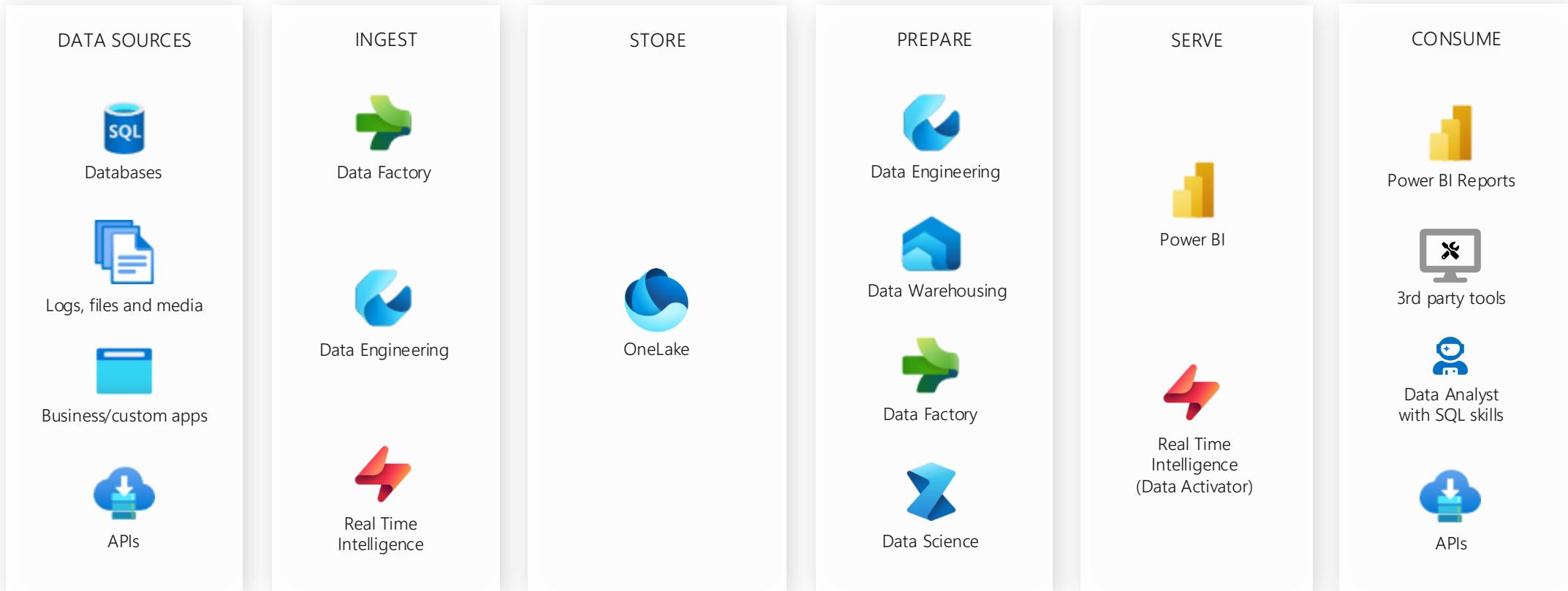
- Rest API

[Item management - Microsoft Fabric REST APIs | Microsoft Learn](#)

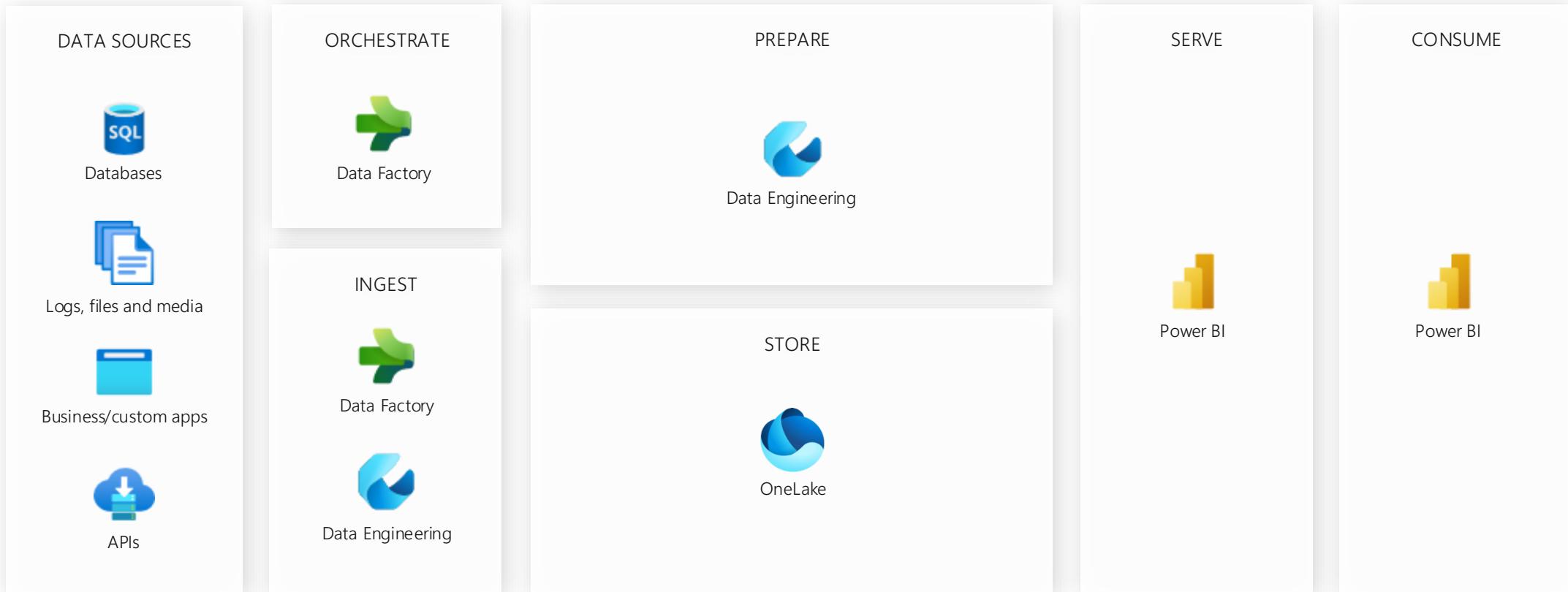


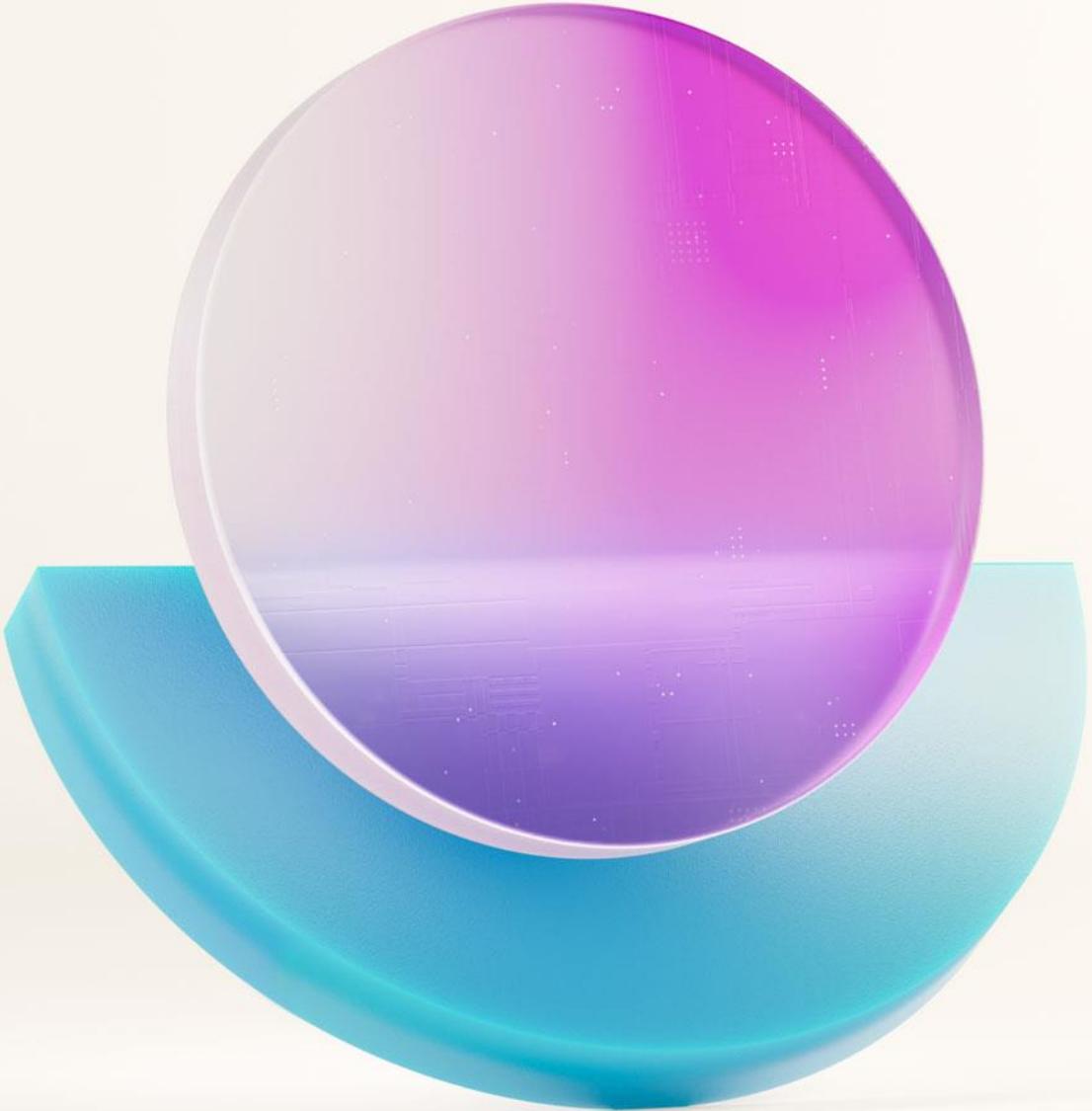
# **DESIGNING OUR FABRIC LAKEHOUSE PLATFORM**

# MULTIPLE COMPONENTS = MULTIPLE POSSIBILITIES



# A LAKEHOUSE STARTING POINT ARCHITECTURE



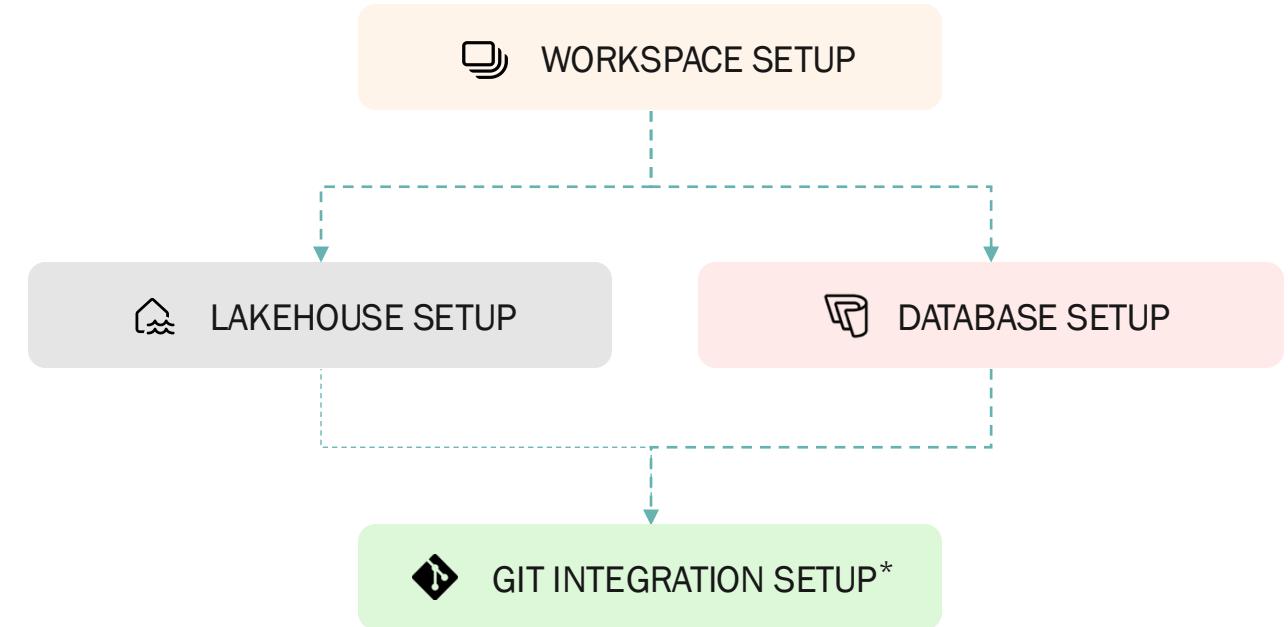


**ACCELERATE AND  
UNIFY FABRIC  
SOLUTION SETUP**

# 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

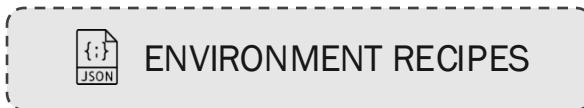
infrastructure

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>

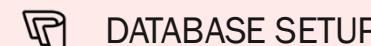
# AUTOMATING FABRIC SOLUTION SETUP - SHOWCASE



- Create workspaces
- Assign capacity
- Assign permissions



- Create Lakehouse items
- Create connections
- Set connection permissions



- Create SQLDatabase item
- Create connection
- Set connection permissions



- Connect workspaces to Git
- Initiate & sync

# 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": {  

```



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 initial 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

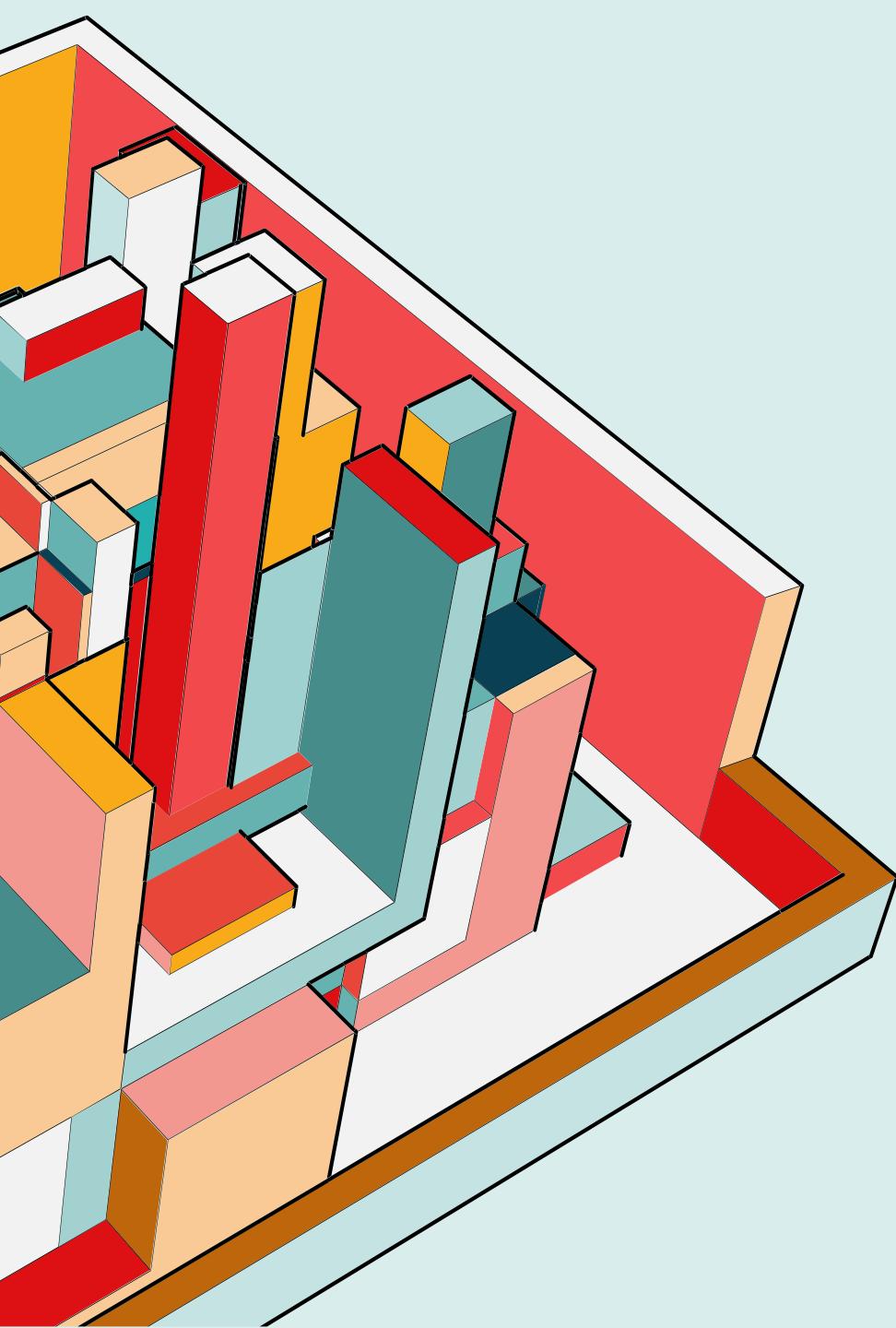
Leading IaC tool!

### 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

