
fabric
FEBRUARY

# Thank you to our Fabric February Friends!

bouvet

twoday

KOKAI

sopra steria

Evidi

Crayon
CONSULTING

Profisee
Master Data Management

Tabular Editor

ANALYTICS
masterminds

VASS

Fraktal

Microsoft

KURANT

ARROW

EUROPEAN MICROSOFT FABRIC Community Conference
BARCELONA 25 SEPT - 01 OCT 2024

profitbase
PART OF HYPERGENE

Capgemini

# Share **your thoughts** and help our speakers!

**fabfeb.app/feedback**

fabric FEBRUARY

# Git Good: Best Practices for CI/CD and Collaboration in Microsoft Fabric

Peer Grønnerup

# WHO AM I?

## PEER GRØNNERUP

*Head of Engineering at Tabular Editor*

*+15 years working with BI, Data Platform design*

*Part of the Fabric Private Preview (Project Trident)*

***... and all in on Fabric Automation and CI/CD!***

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

https://peerinsights.emono.dk/

# Download FabricOps source code

https://github.com/gronnerup/FabricOps

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

# Where are we going – Git Good in Fabric

Architecting for scale


Workspaces, repos and environments

Branching for real-world collaboration

Automating deployments

Tips, Tricks & Summing up

---

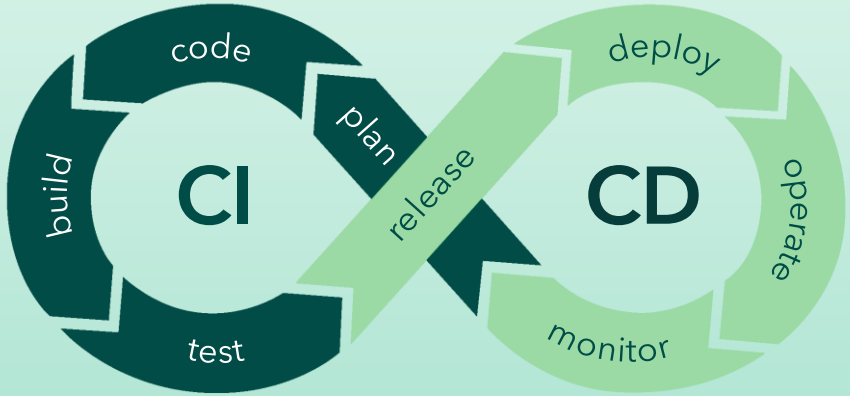
- Architecture & platform setup
- Fabric Git integration & repo structure
- Ways-of-working & collaboration
- Touch branching strategies & release flows
- CI/CD pipelines
- fabric-cicd library and Fabric CLI
- Automation in general

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

# The basics of CI/CD and Version Control - Why it matters

## The automation engine that powers DevOps workflows

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

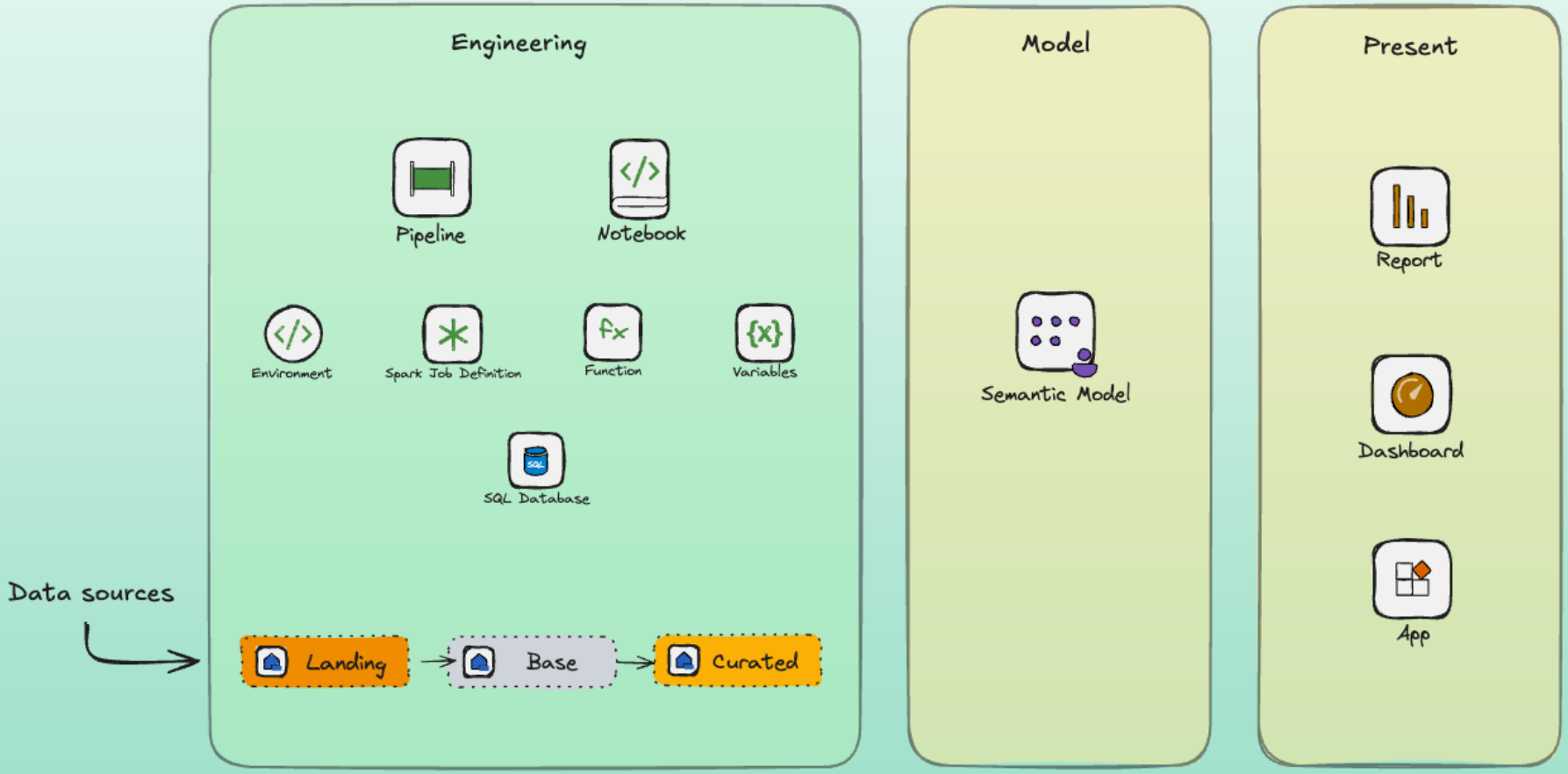



## The Fabric angle of things

- Fabric supports GitHub and Azure DevOps as git provider
- Fabric items in Git are declarative and cloud-native
- Items map directly to Git folders in your repository

# Architecting for scale

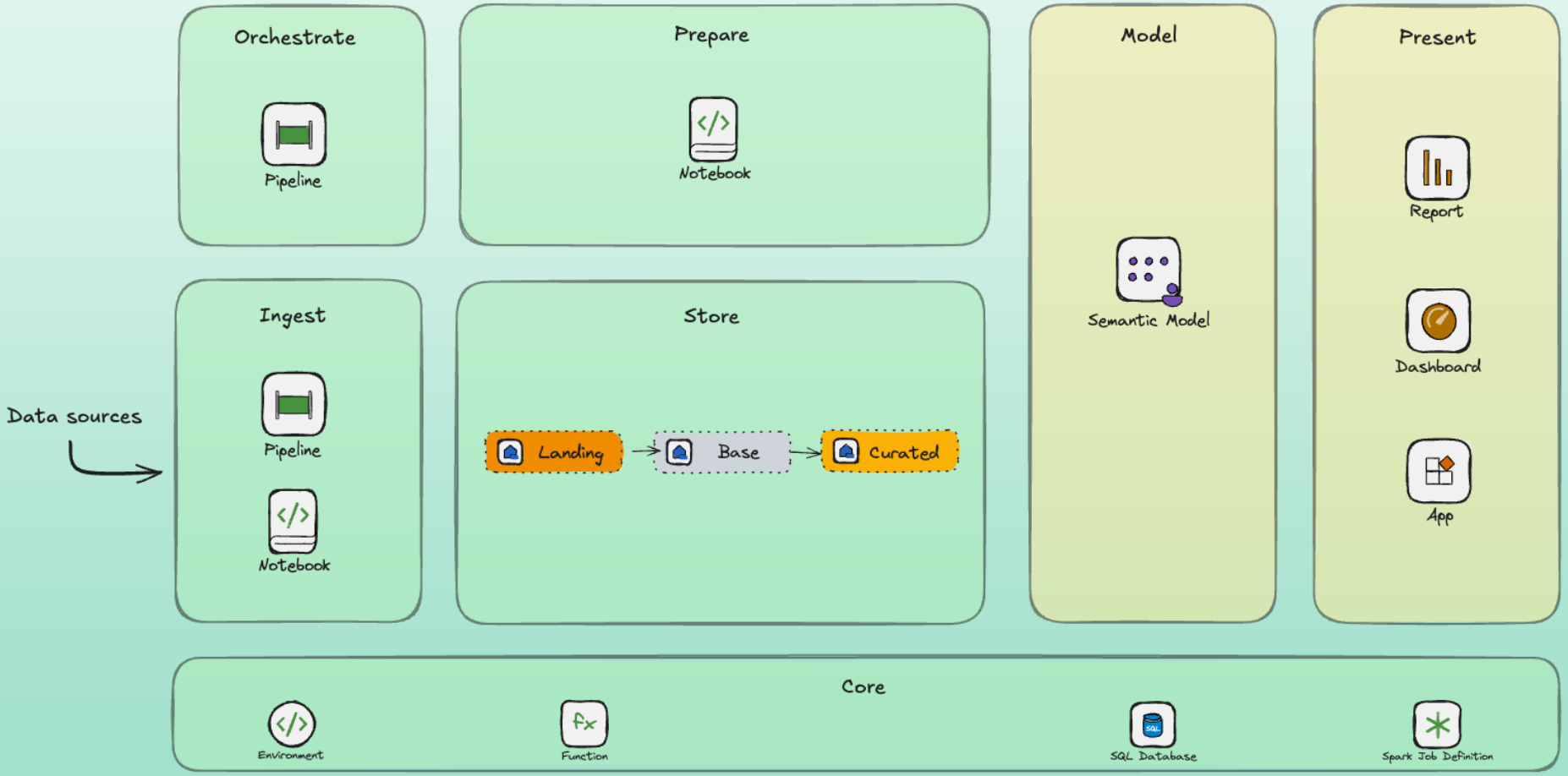
# Reference Architecture - Basic




Fabric workloads

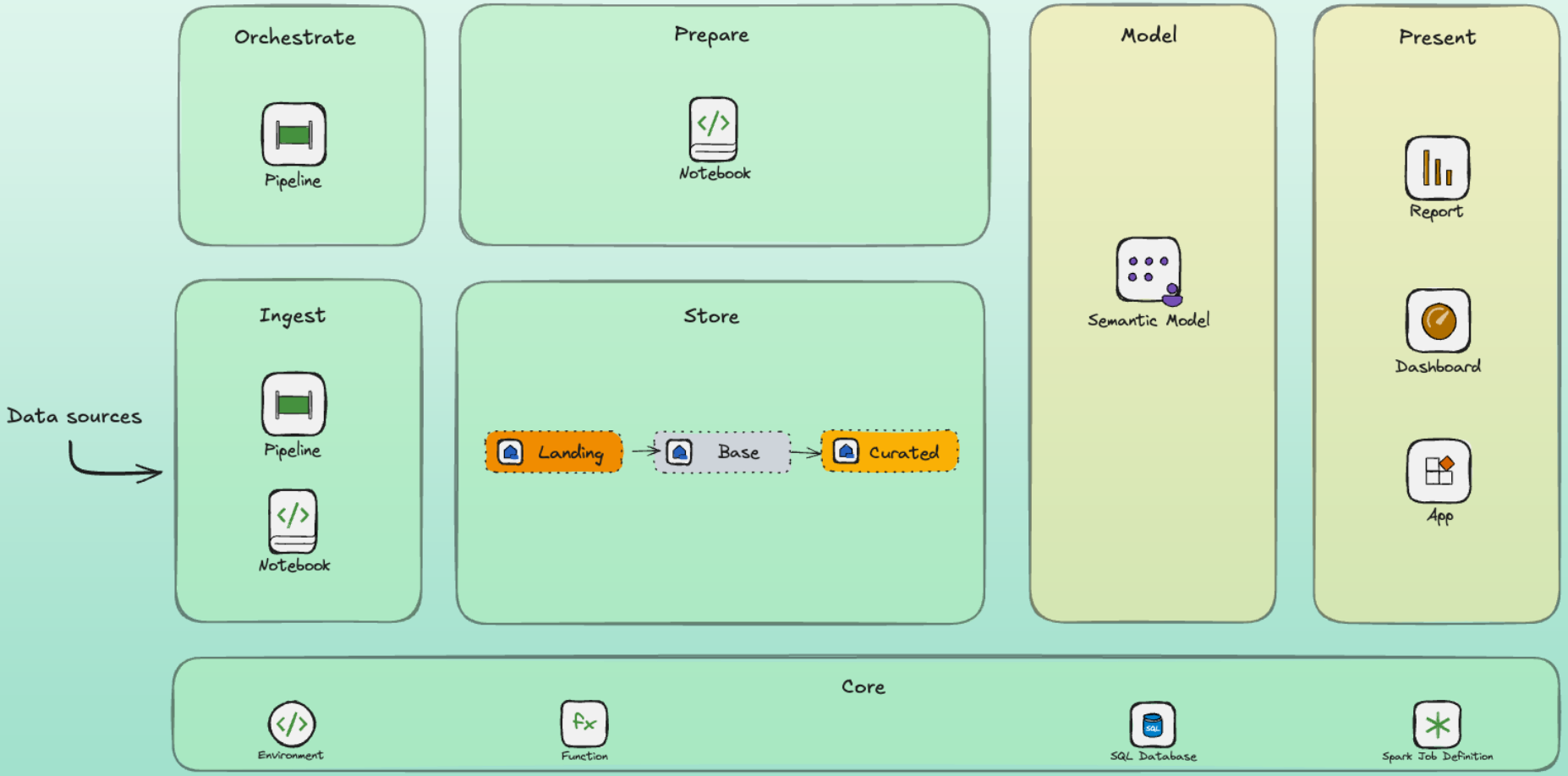
Power BI workloads

# Reference Architecture - Enterprise

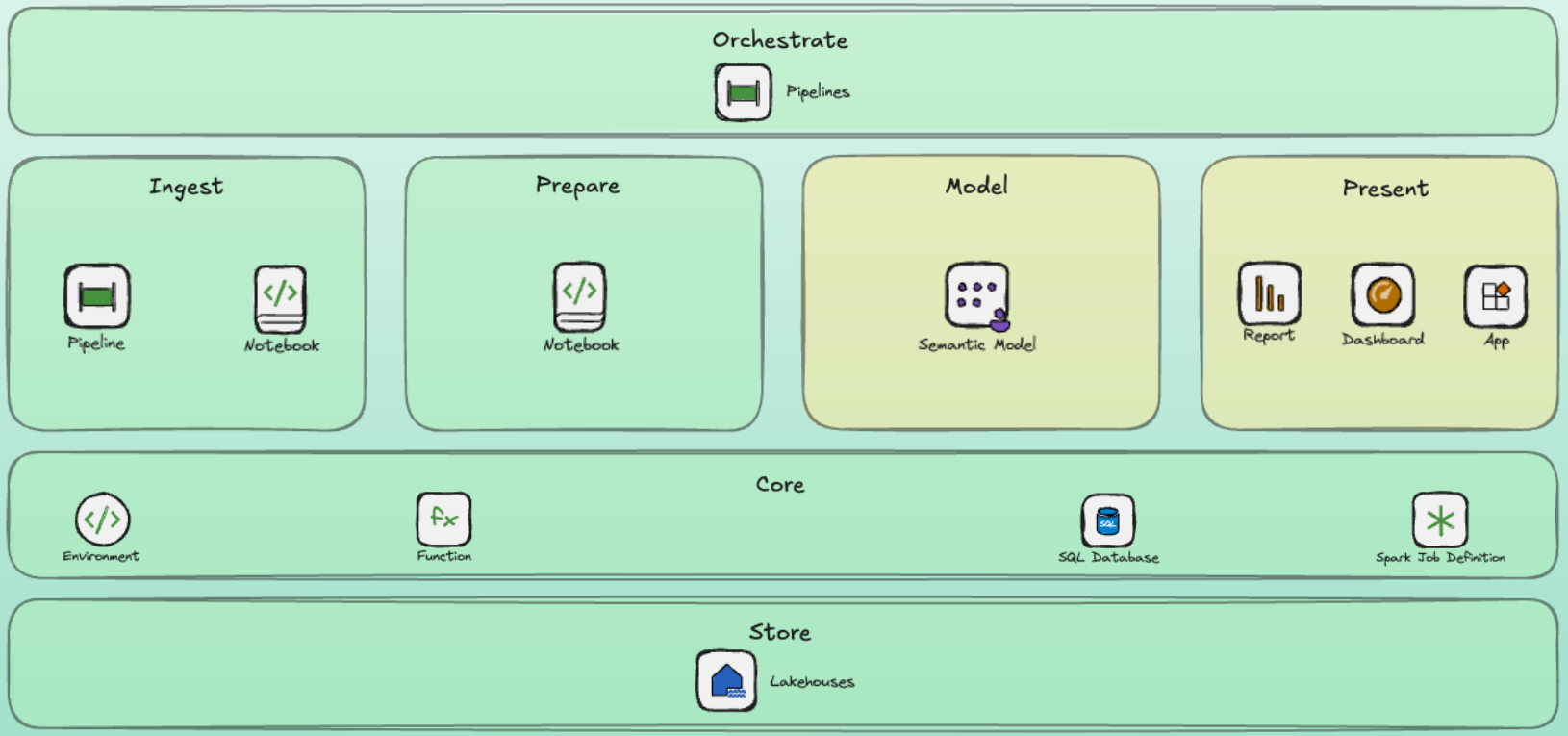


# Workspaces, repos and environments

# Reference Architecture - Enterprise



# Fabric Workspace structure




x3

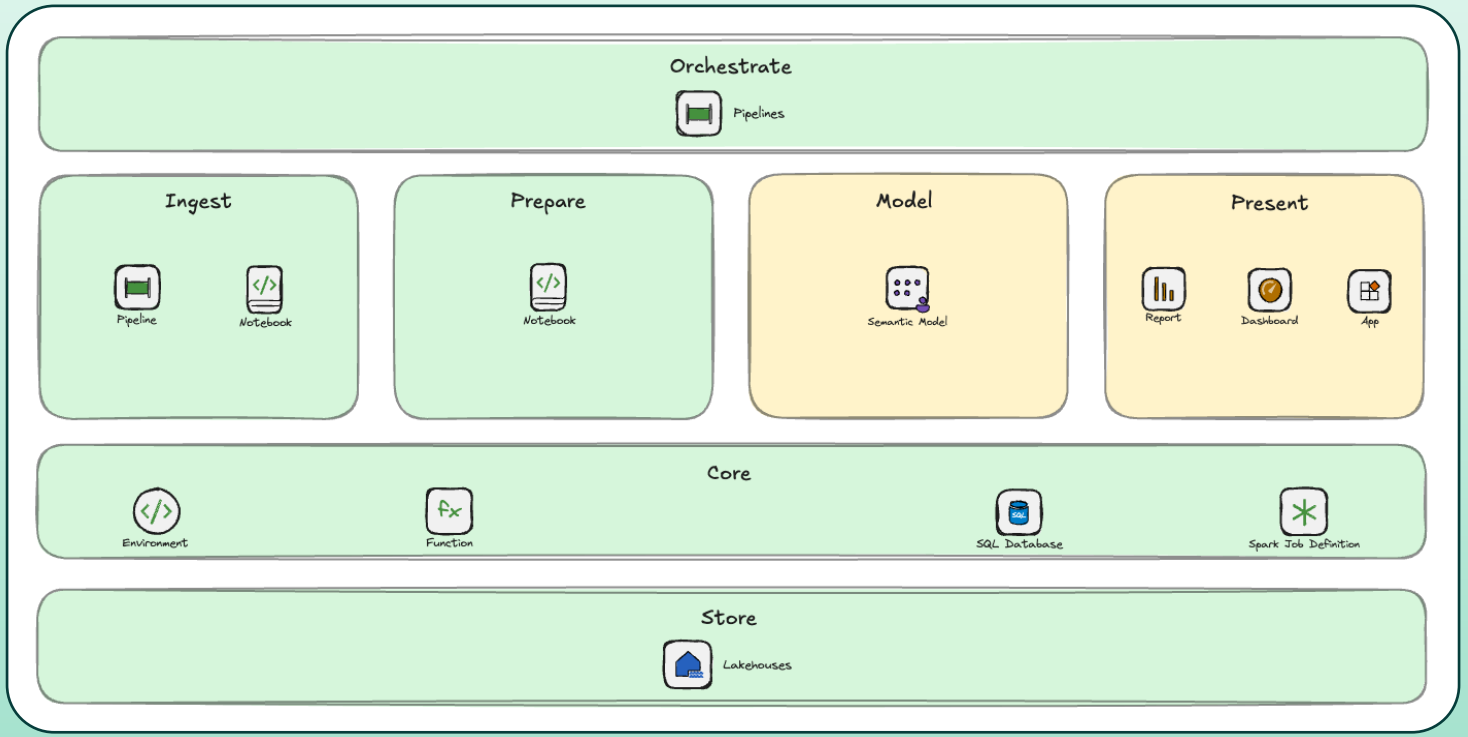
[dev, tst, prd]

Fabric workloads

Power BI workloads

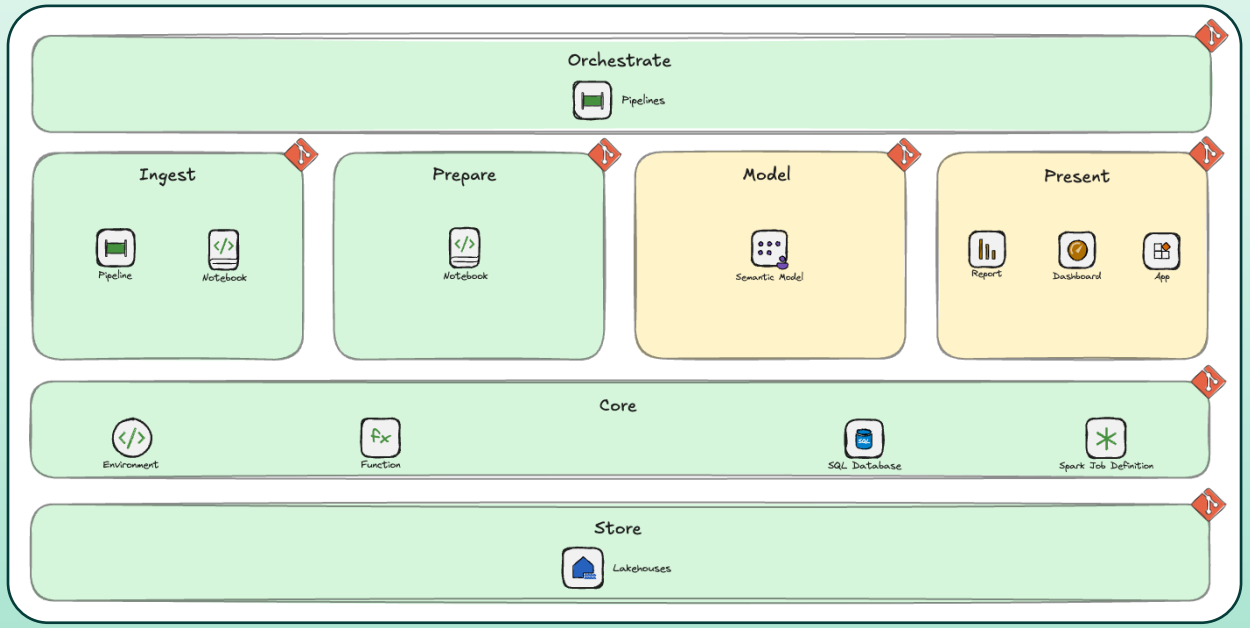
# Fabric Workspace structure

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



# My go-to for Microsoft Fabric Git Setup

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



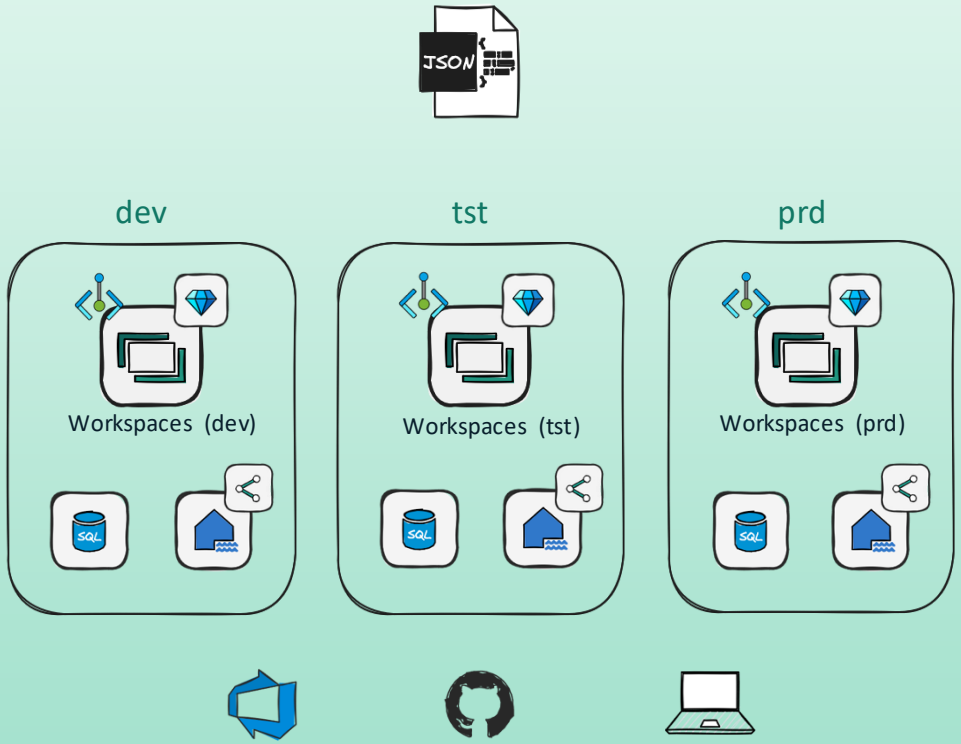
# My go-to for Microsoft Fabric Git Setup

Git Repo structure

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

# Automating Fabric solution setup

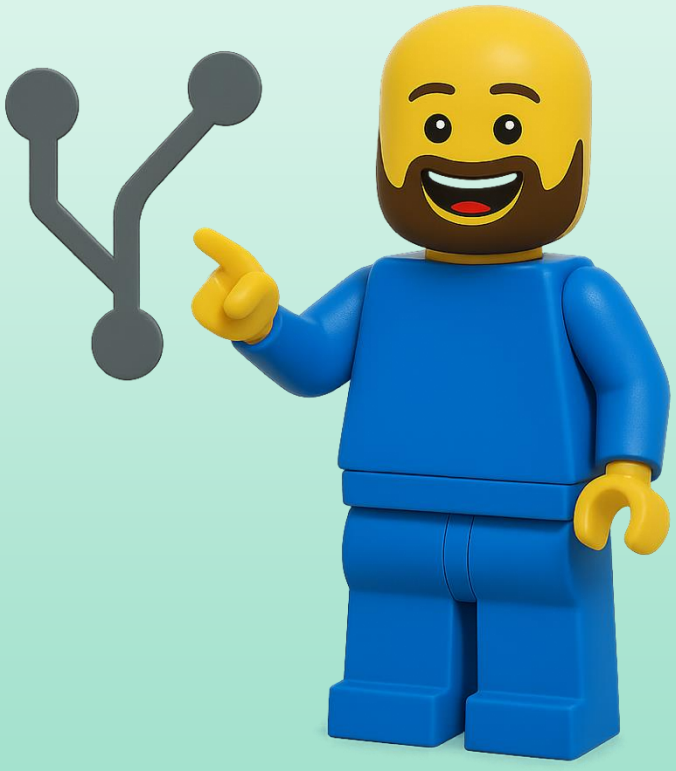
- Recipe based solution (json)
- Utilizes Python and the Fabric CLI
- Can run from Azure DevOps, GitHub or from any client machine running python
- Minimum requirements:
  - A Service Principal
  - A Fabric Capacity




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

# Show time!

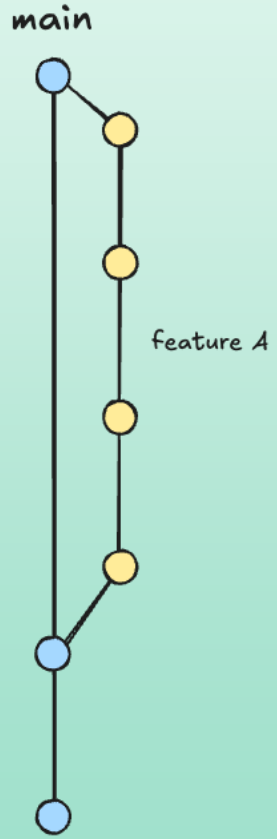
# Branching for real-world collaboration



# Branching for real-world collaboration

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

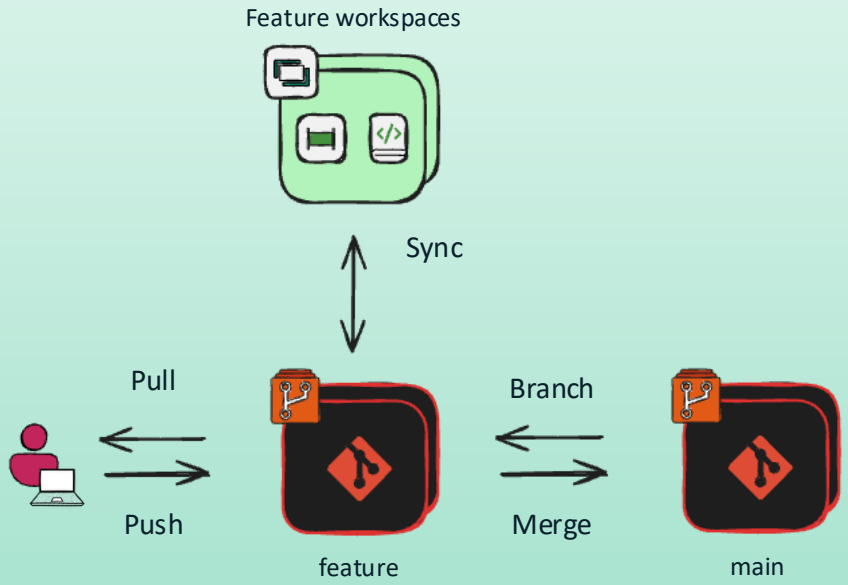
- Git is the source of truth, and branching is how we manage change
- Branches gives developers a seperate workspace for their code and…
  - Isolates development
  - Protects the mainline of our code
  - Enables parallel development
  - Help organize and structure releases
  - Is crucial for streamlined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



# Development process – Ways-of-working

**Development flow - Supported by automation!**

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




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

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

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

# Demo time!

# Automating deployment

# Release process – The main options…

## Fabric Deployment pipelines

- No code experience
- For simpler solutions
- No support for *:
  - Pre-deployment opr.
  - Post-deployment opr.
  - Test & validation

## Git based deployment

- Suitable when using Gitflow
- Each environment connected to git branch
- No need for building environments
- Might require post-deployment operations

## Git based deployment using build environment

- Supports building enviroments
- Deployment through ADO Pipelines/Github actions
- Supports manipulation of item definitions

# CI/CD Building Blocks for Microsoft Fabric

### Tools & automation options

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

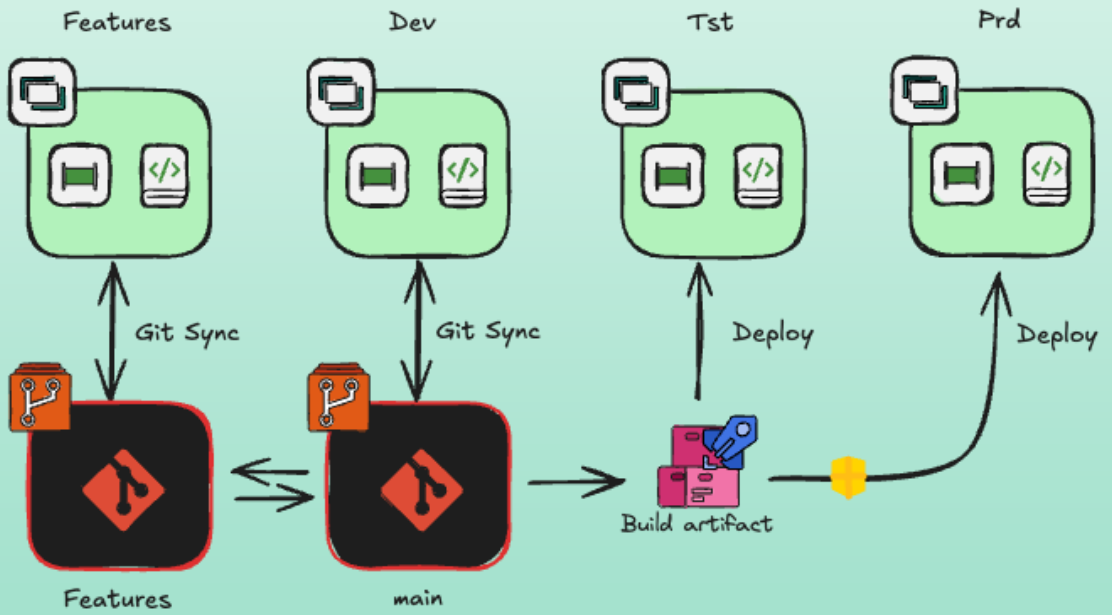
### fabric-cicd Python Library

- 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

# Release process - 3 selected options

## Option 1

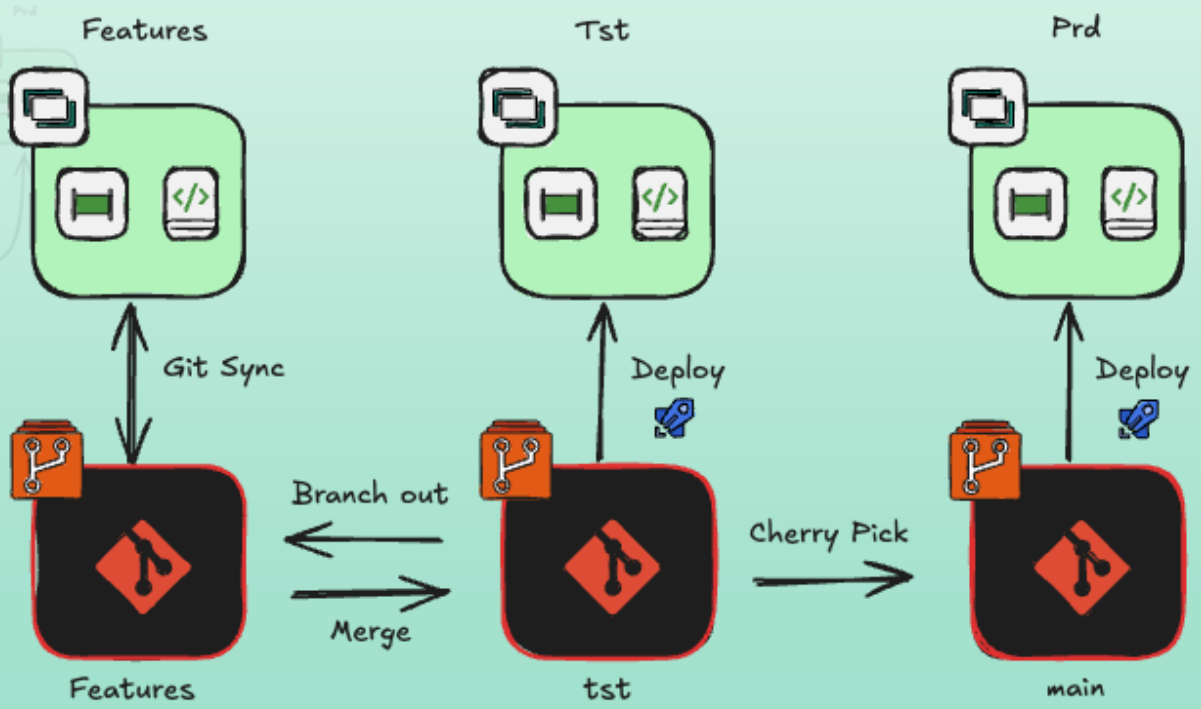
Dev = Main



# Release process - 3 selected options

## Option 2

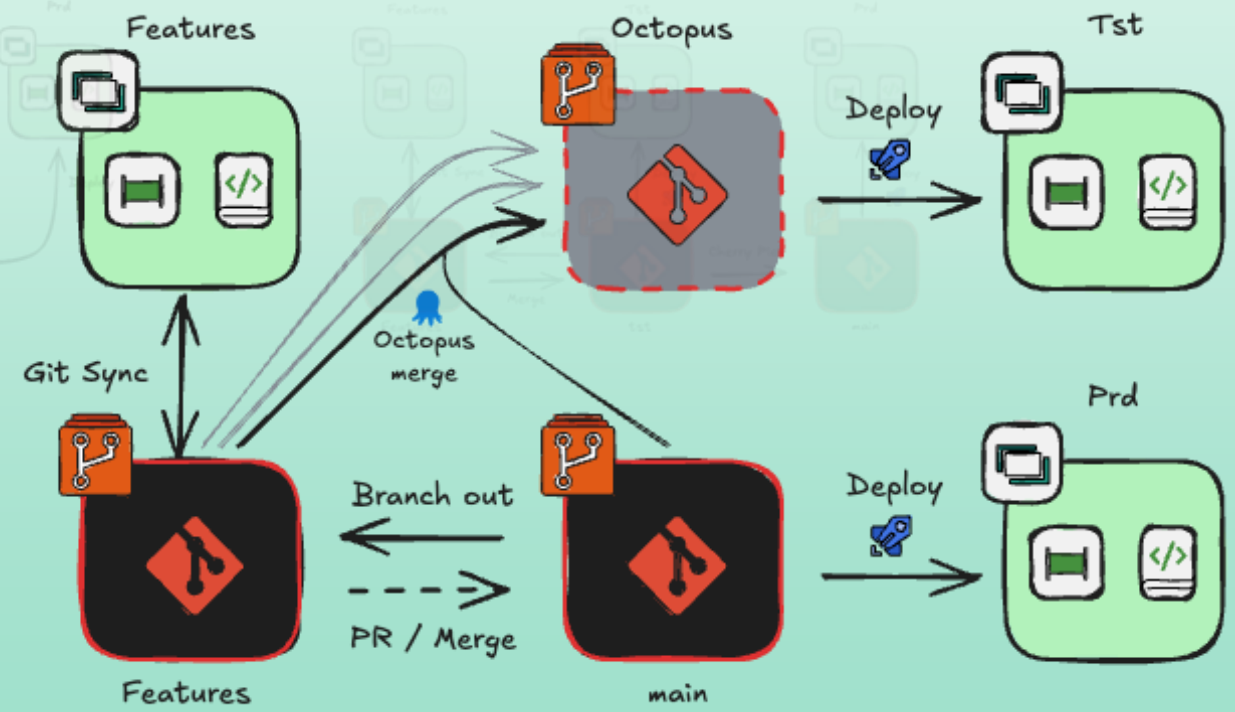
PR to test – Cherry pick to main



# Release process - 3 selected options

## Option 3

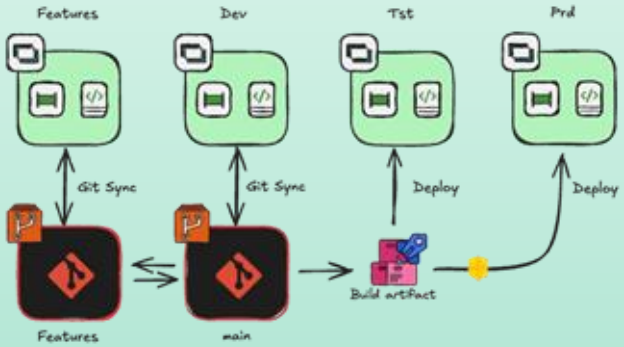
Octopus deploy



# Release process - 3 selected options

## Option 1

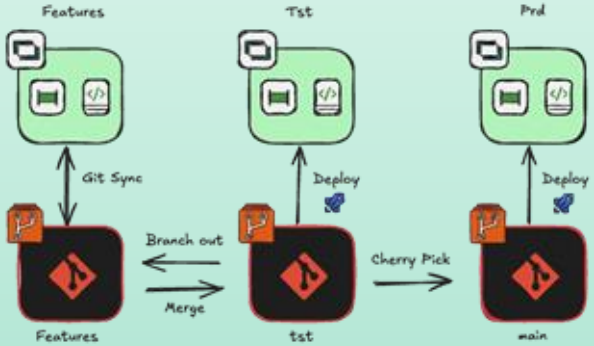
Dev = Main




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

## Option 2

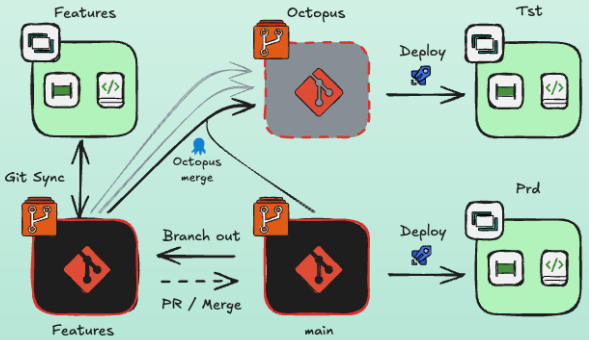
PR to test – Cherry pick to main




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

## Option 3

Octopus deploy




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

# Demo time!

# Tips, Tricks & Summing up

# Tips, Tricks & Summing Up!

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

# Download FabricOps source code

https://github.com/gronnerup/FabricOps

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

# TIME FOR QUESTIONS!

## BEFORE CONNECTION IS TERMINATED BY PEER...

# Share **your thoughts** and help our speakers!

**fabfeb.app/feedback**

# Thank you!