# Abstract

## DevOps for the humble data engineer

Data engineers store data in warehouses, lakehouses, or databases. But what about the SQL scripts, notebooks, or KQL queries? Often, they're stored on local hard drives or SharePoint 😖.
However, a better option exists: Git 😉

Git is a distributed version control system that allows easy versioning and collaboration for scripts and notebooks. This session will teach data engineers how Git can support their work in developing data solutions.

Once the scripts and notebooks are fully versioned with Git, how do you get them from the Git repo to the Fabric workspace? Of course you could copy it manually, but wouldn't it be nicer if it just deployed automagically whenever a change occurs? The DevOps world has a solution: Pipelines. Learn how a pipeline can take your notebook and deploy it to dev, staging, and prod environments, adapting connection string and other parameters to match each environment automatically. Pipelines are also a fully integrated part of Microsoft Fabric, that we will use for a practical demonstration.

Git and DevOps pipelines have transformed software engineering and can do the same for data engineering, easing daily tasks. Learn more about Git and Pipelines to improve your workflow.

# DevOps for the humble Data Engineer

# Who are we?

## Marisol Steinau

- Autodidact Data Enthusiast
- > 8 years experience using Microsoft Data Platform
- Data Solution Architect @Be-terna
- Frequent guest at Legoland Resort Germany
- linkedin.com/in/marisol-steinau-bb1253253
- marisol.steinau@steinautech.com

## Sebastian Steinau

- Techie (Azure, .NET, DevOps, IaC)
- Stepping into the data world with Fabric
- Lead Consultant @ Devoteam M-Cloud
- TV-Series geek and cook
- linkedin.com/in/sebastian-steinau-312888200
- sebastian.steinau@steinautech.com

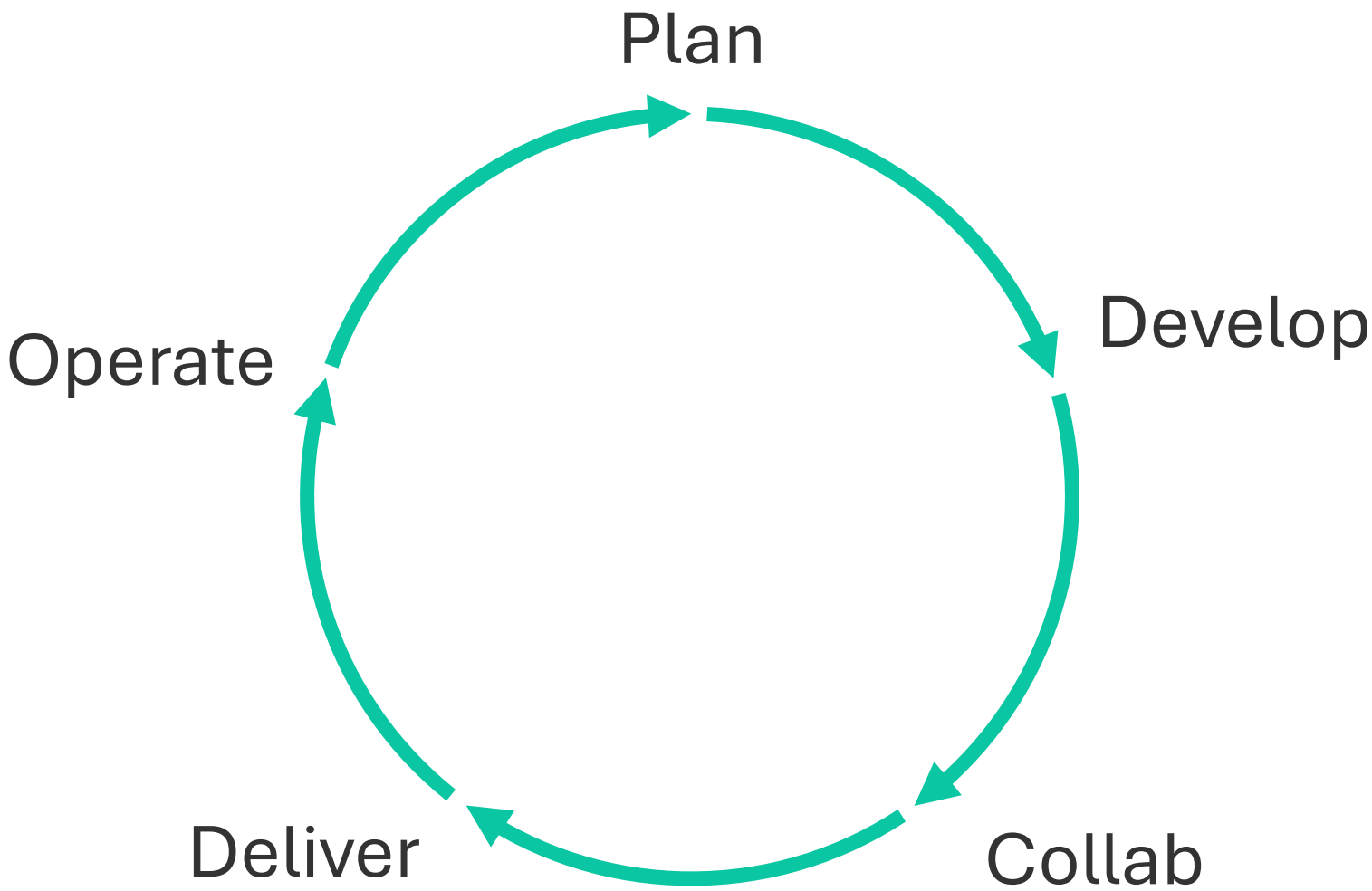
Joint blog: refugeinaudacity.eu

(Work in progress)

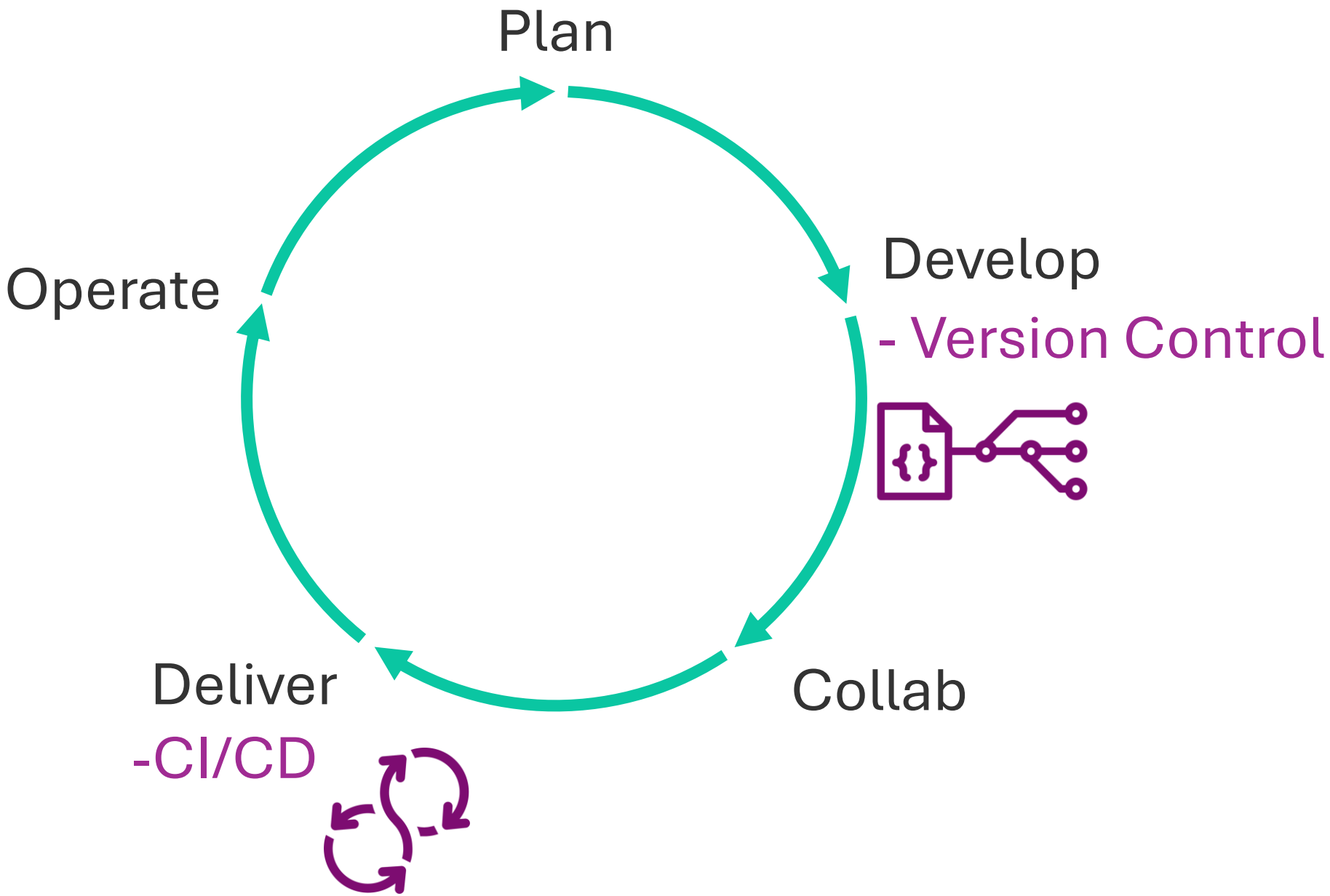
# Agenda

- Introduction
- Version control with Git
- Git in Fabric
- CICD in Fabric
- Improving Fabric with DevOps Pipelines
- Conclusions

# DevOps



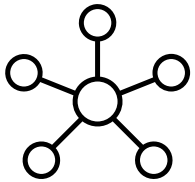
# DevOps



# Advantages of Version Control

## Versionable

**If something doesn’t work, you can go back to a working version**



## Unambiguous

**A central place acts as a single point of truth**

## Traceability

**What changes were made, who made them and why allows to track progress and resolve issues**

## Integrity

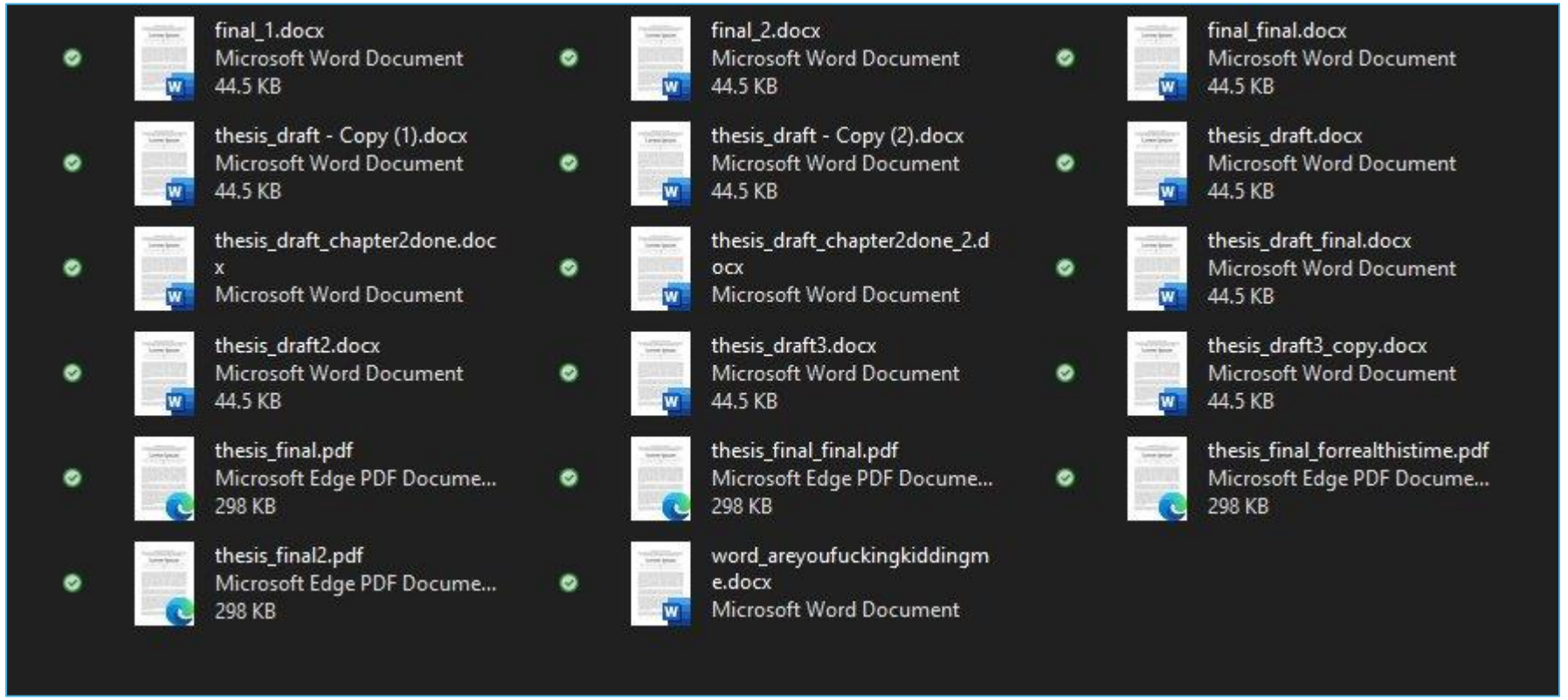
**The code or files stored in the system remain unaltered and uncorrupted**

# Version Control

Do I need version control If I am a team of one?

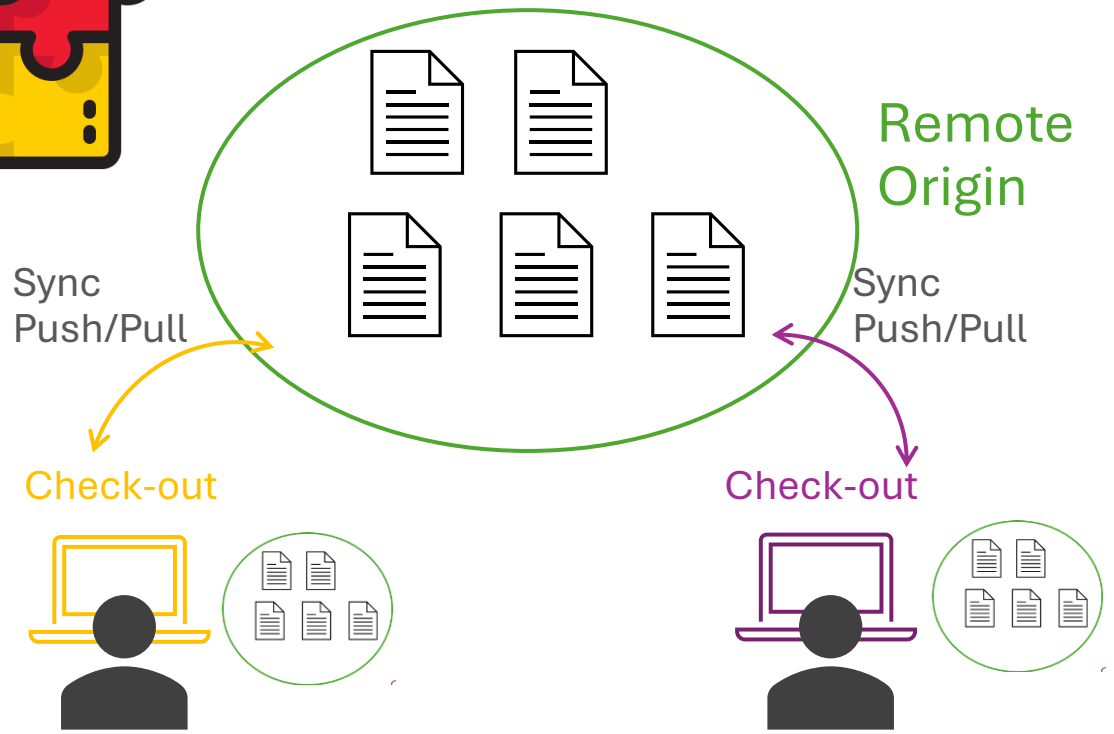
# Version Control

## Do I need version control If I am a team of one?

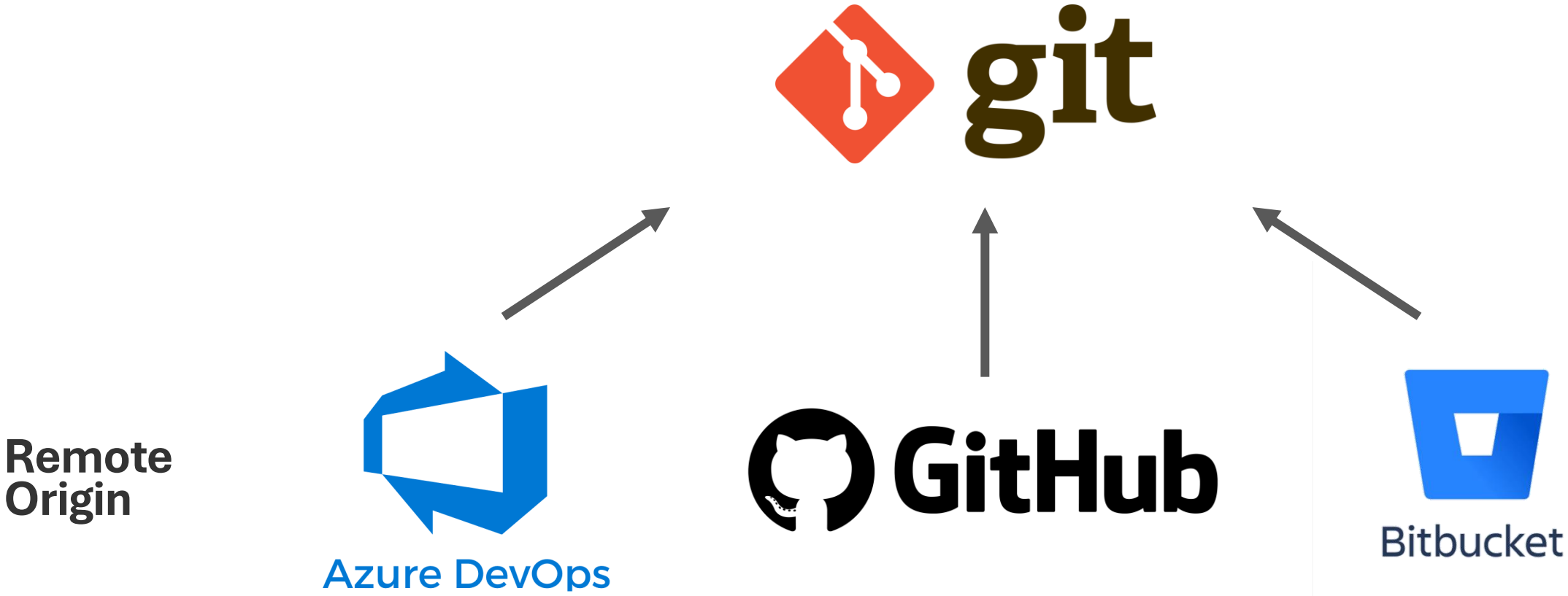


# Version Control System

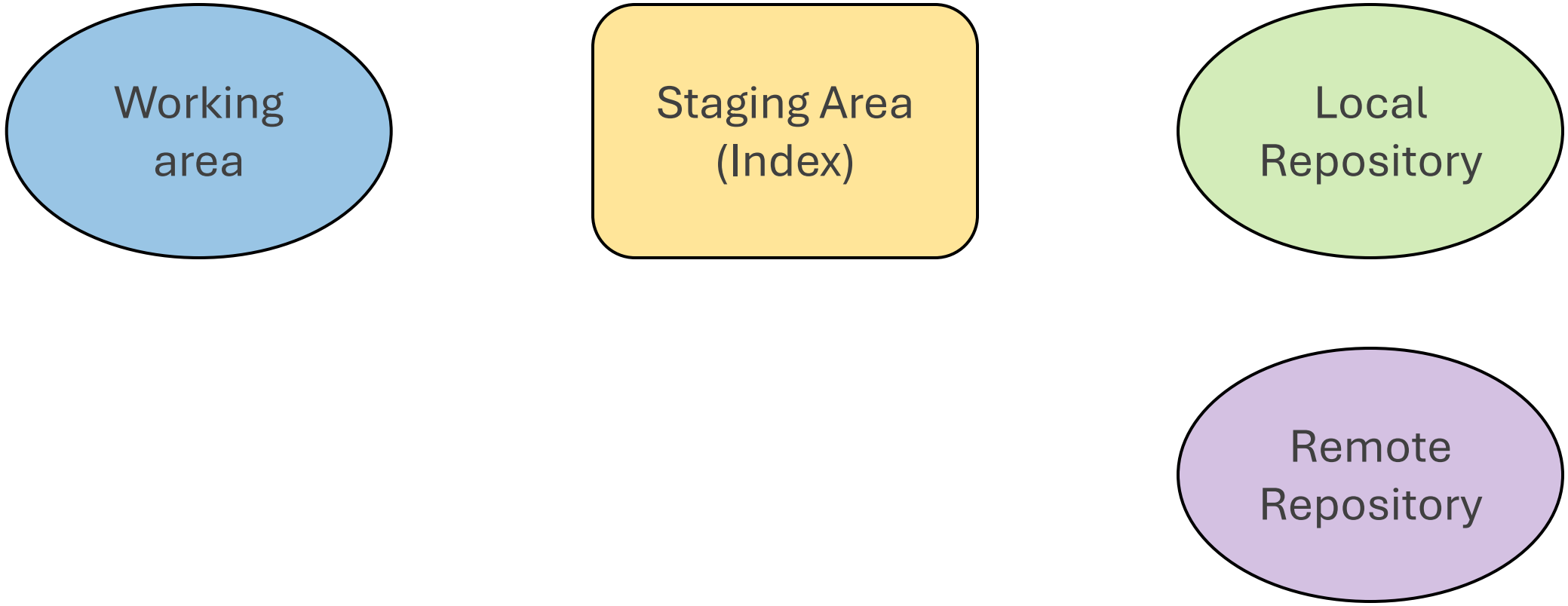
**Integrate often!**



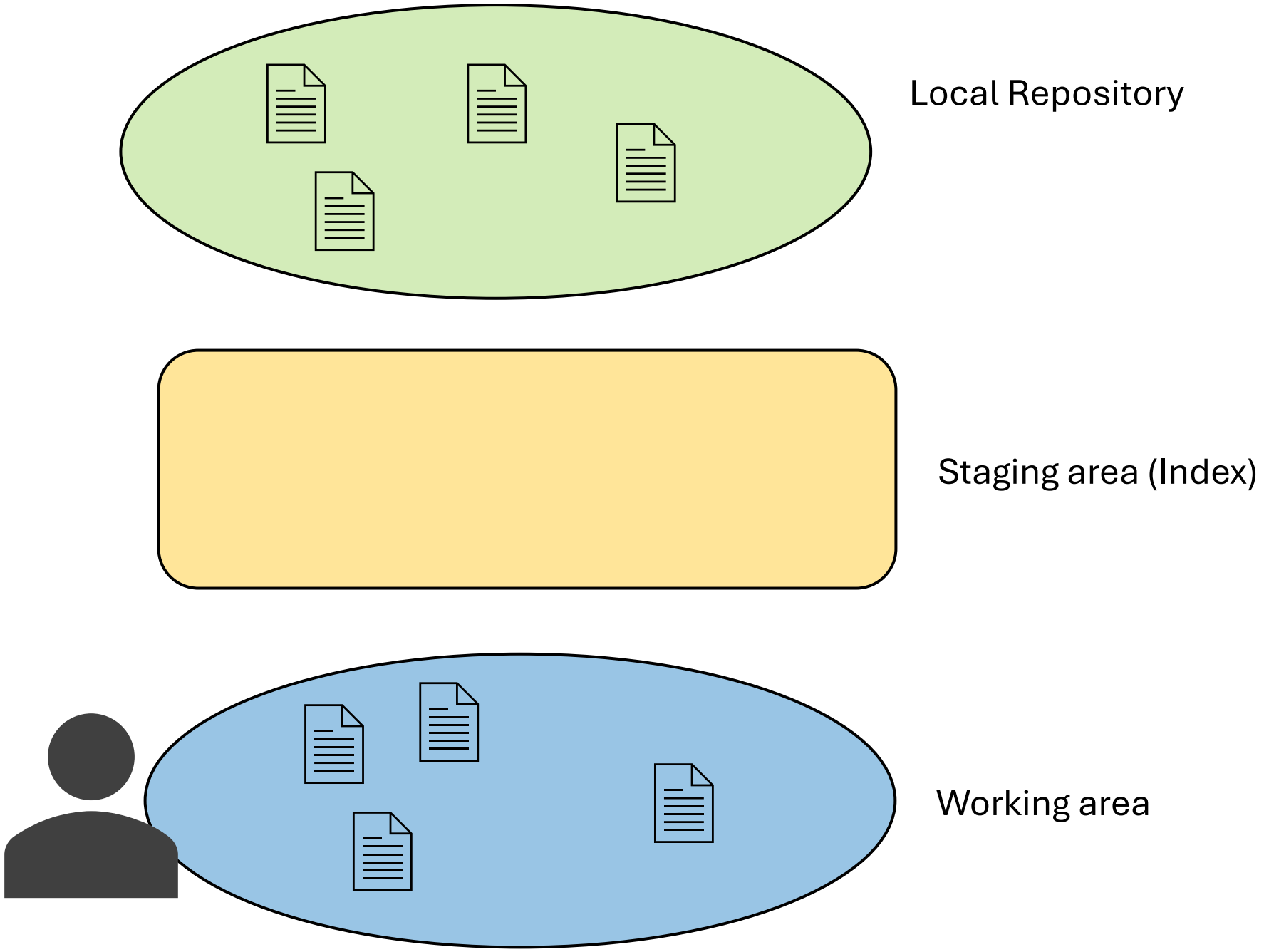
git
Versioning
Enables collaboration
Source of Truth
Consistency
Immutable history


Remote
Origin
git
Azure DevOps
GitHub
Bitbucket

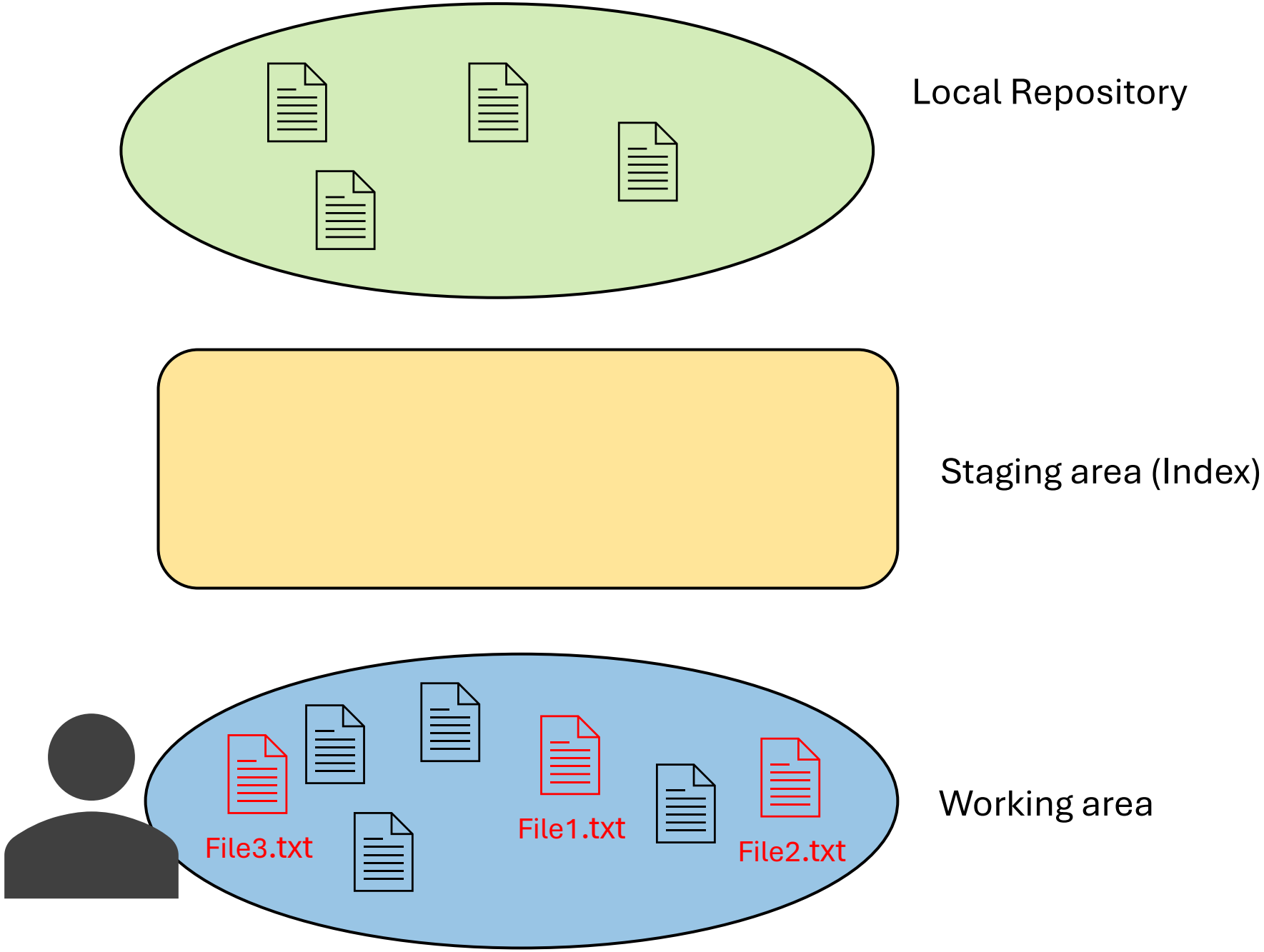
# Git


Working area
Staging Area (Index)
Local Repository
Remote Repository

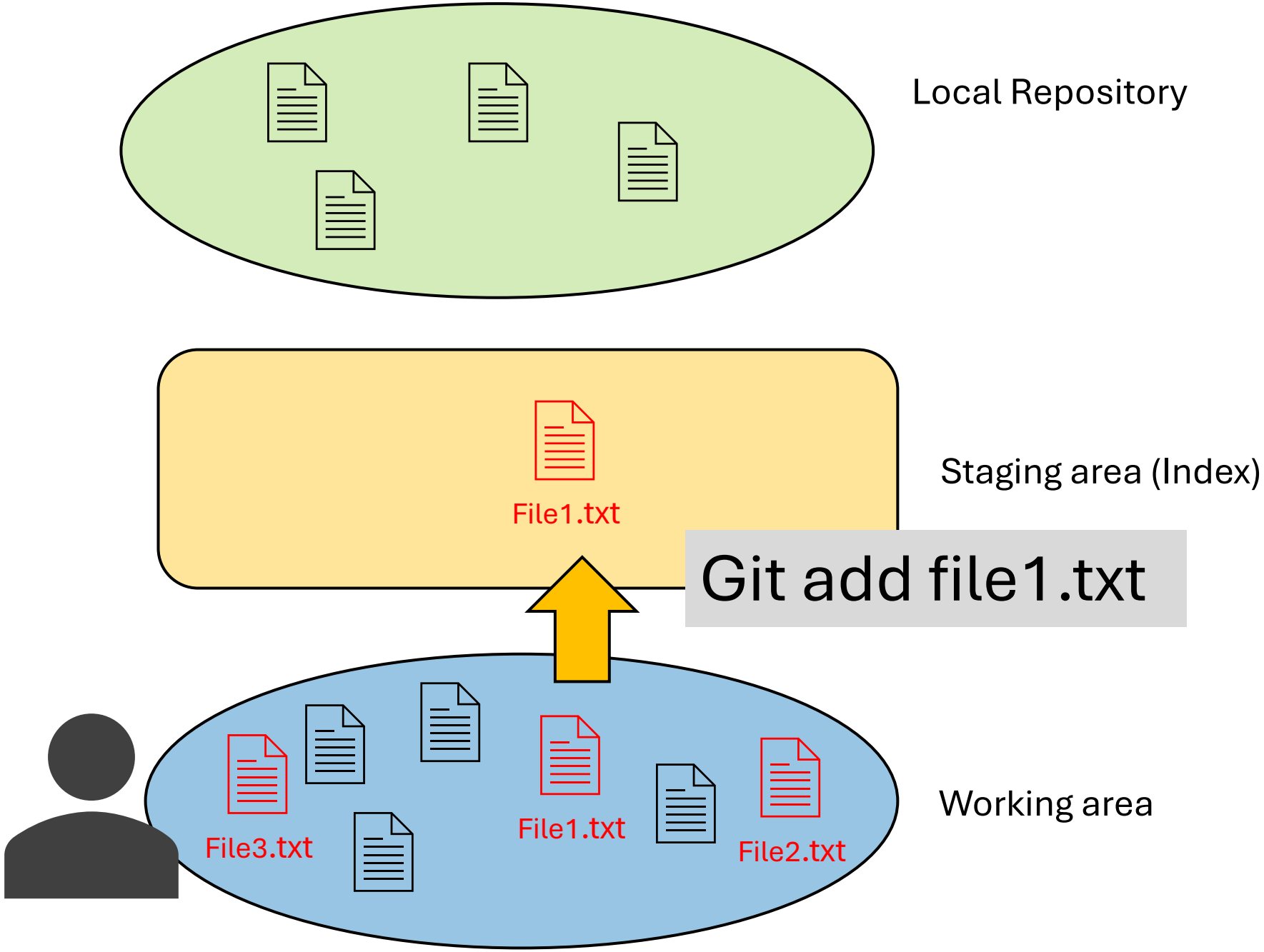
# Git



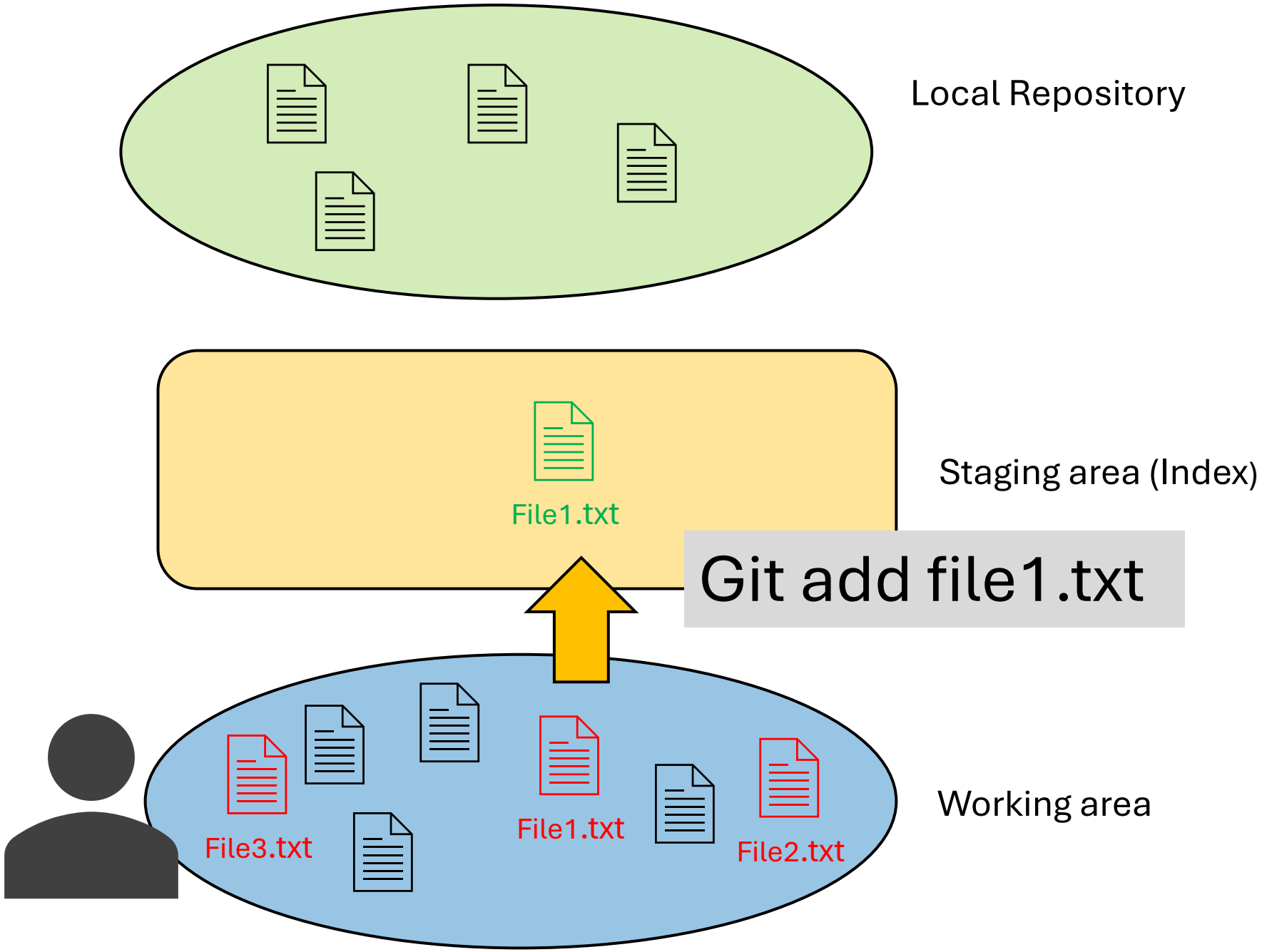
# Git


Local Repository
Staging area (Index)
Working area
File3.txt
File1.txt
File2.txt

# Git

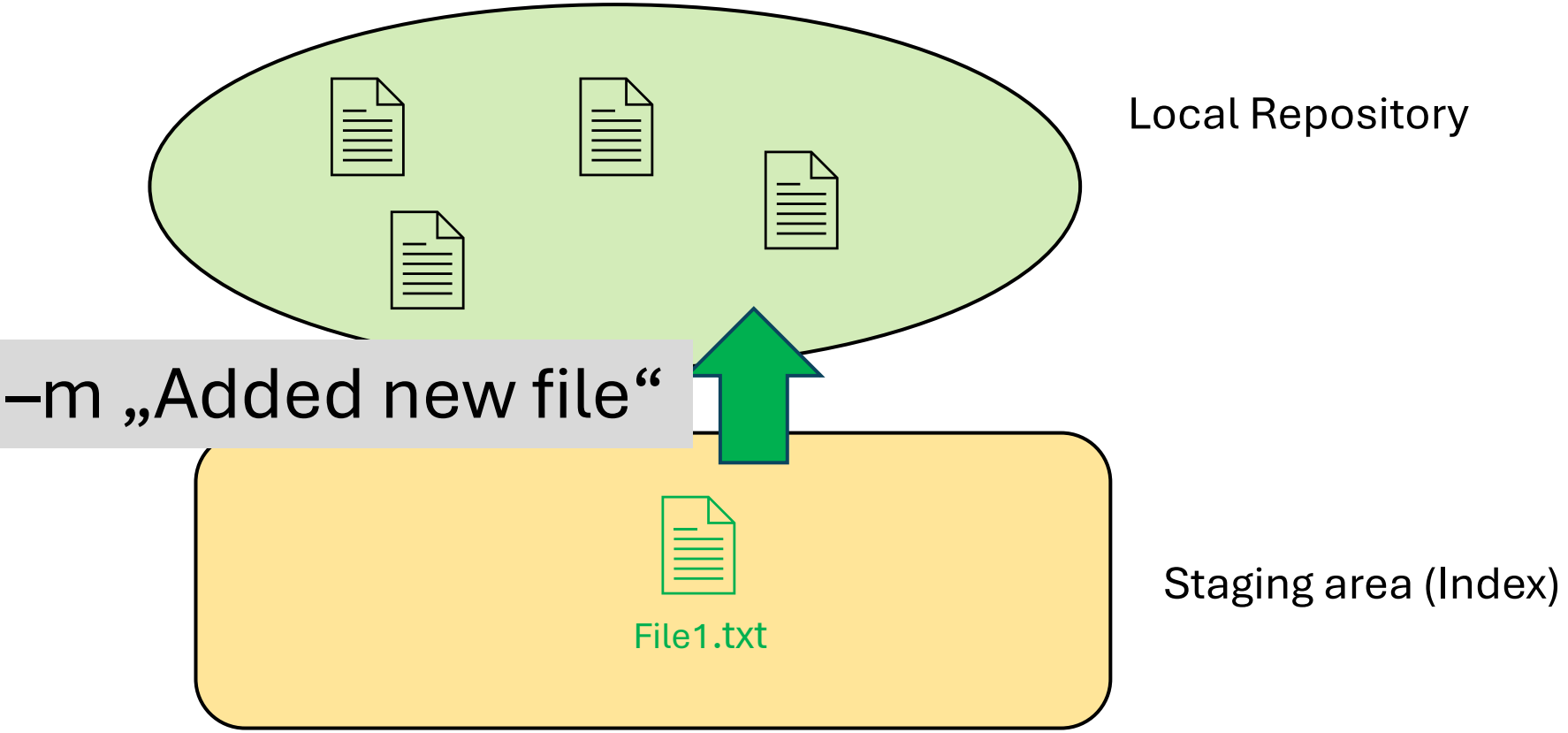

Local Repository
File1.txt
Staging area (Index)
Git add file1.txt
File3.txt
File1.txt
File2.txt
Working area

# Git

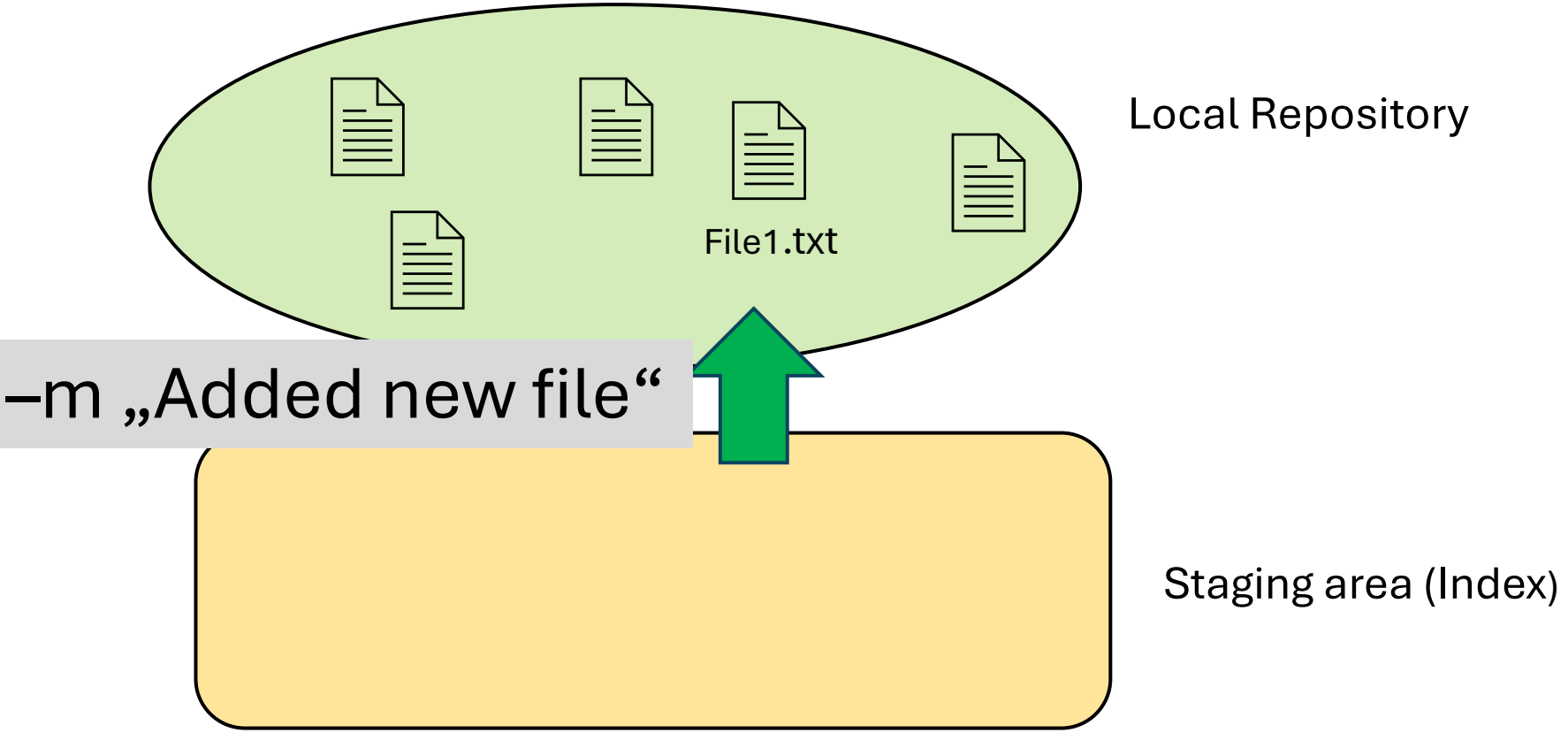

Local Repository
File1.txt
Staging area (Index)
Git add file1.txt
File3.txt
File1.txt
File2.txt
Working area

# Git

Git commit –m „Added new file“



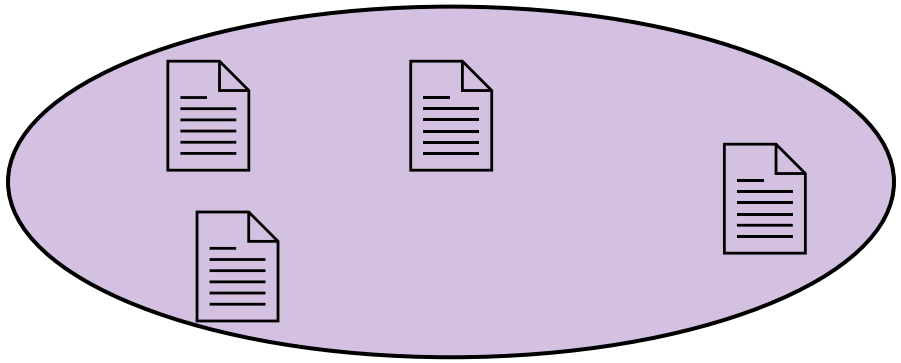
# Git

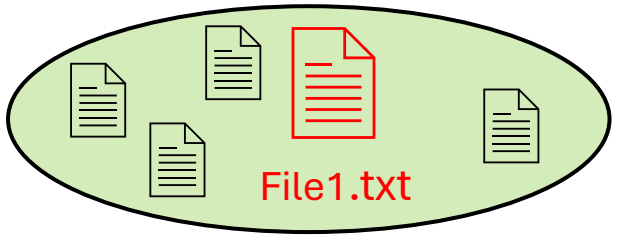



Git commit –m „Added new file“

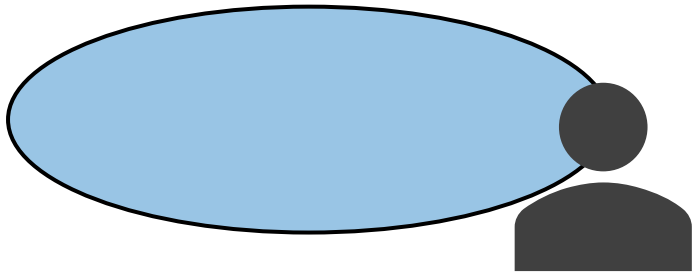
# Git

Remote Repository



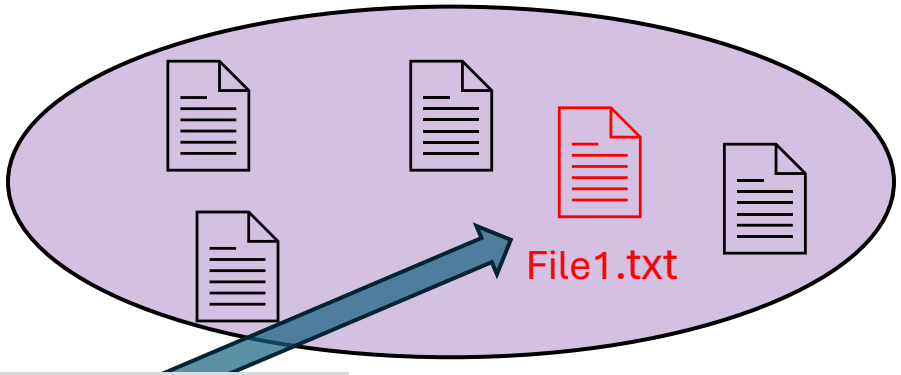



Local Repository

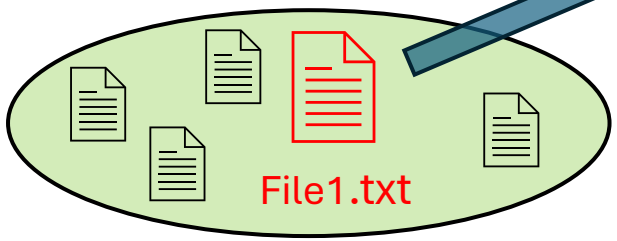


# Git

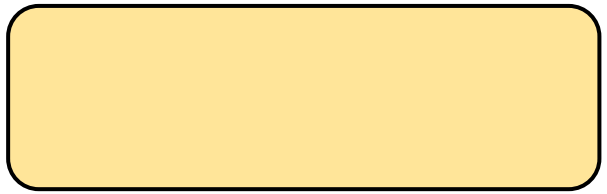
Remote Repository

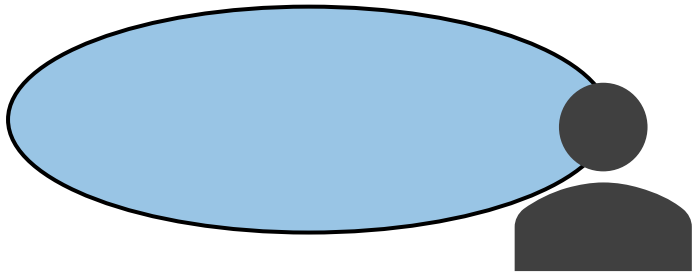



Git push origin main

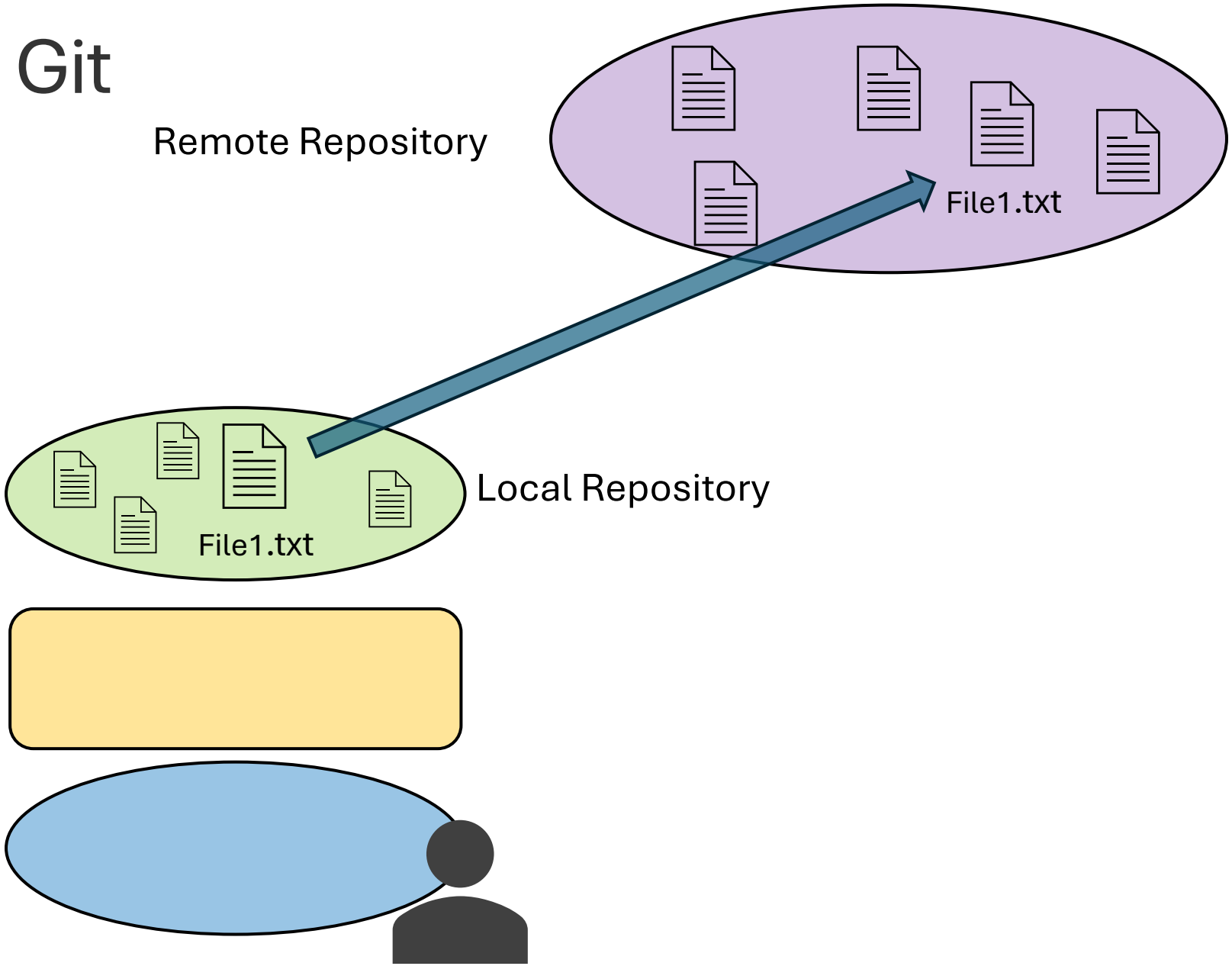



Local Repository



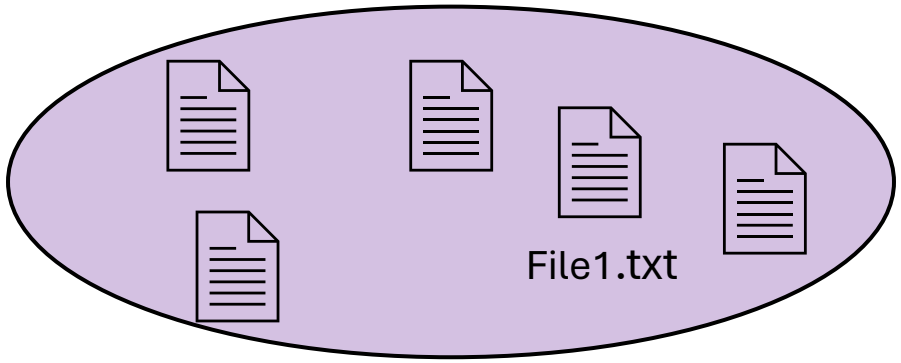


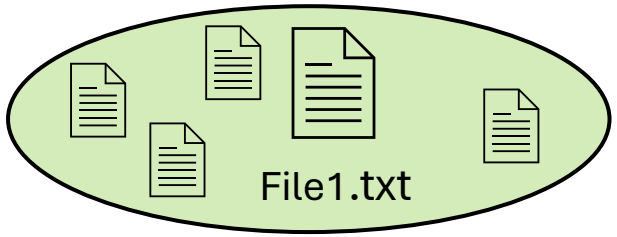
# Git



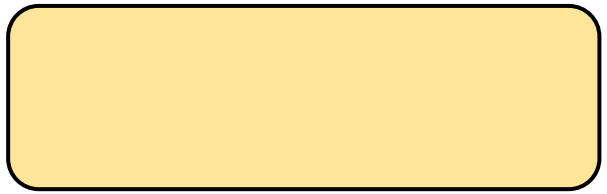
# Git

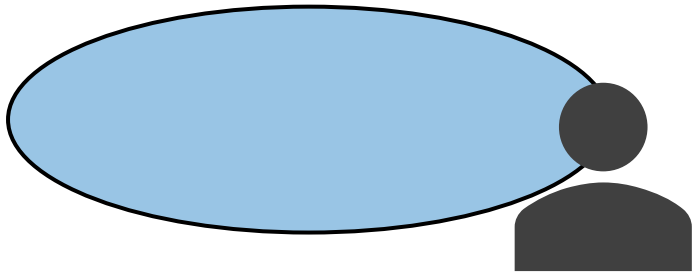
Remote Repository



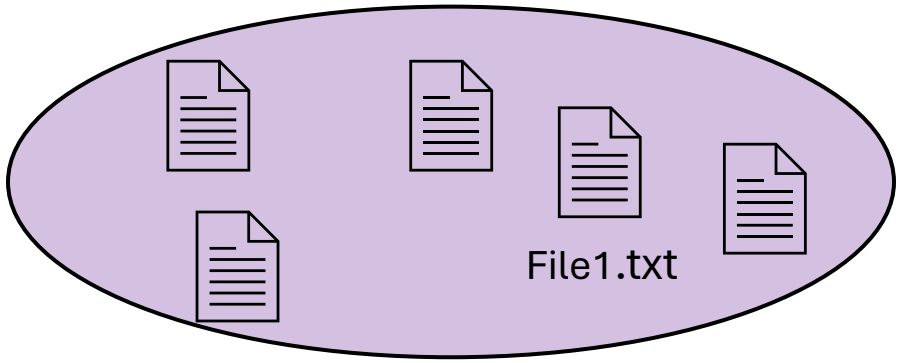

Local Repository

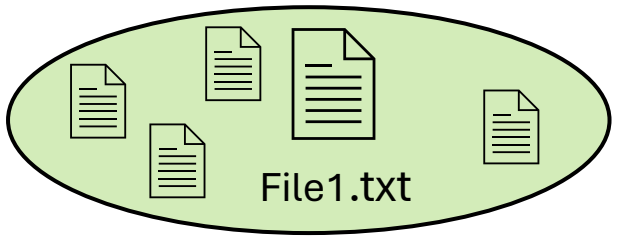




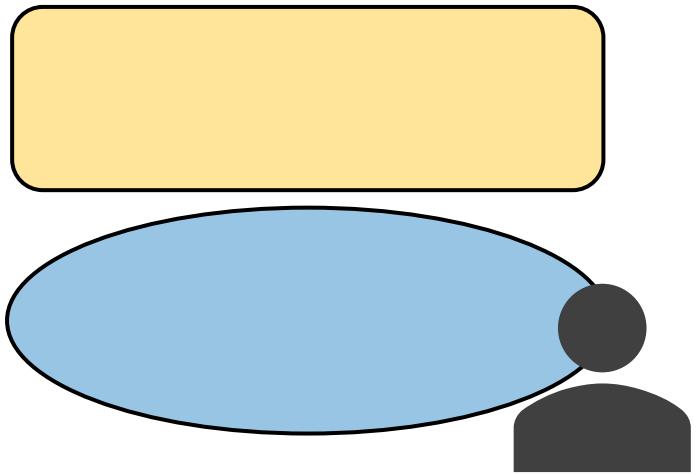
# Git

Remote Repository

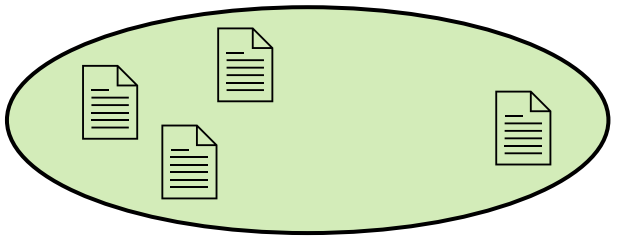


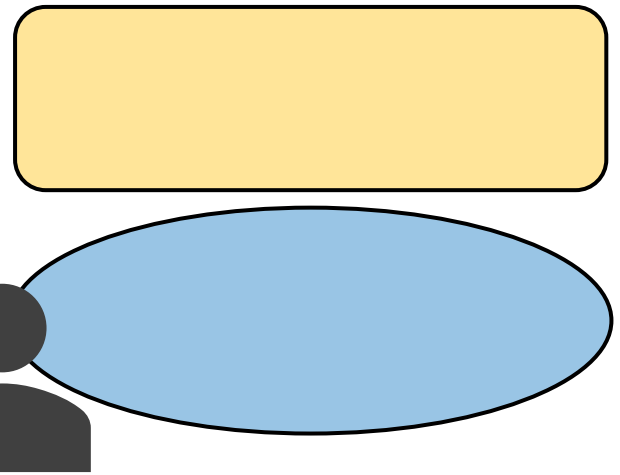




Local Repository

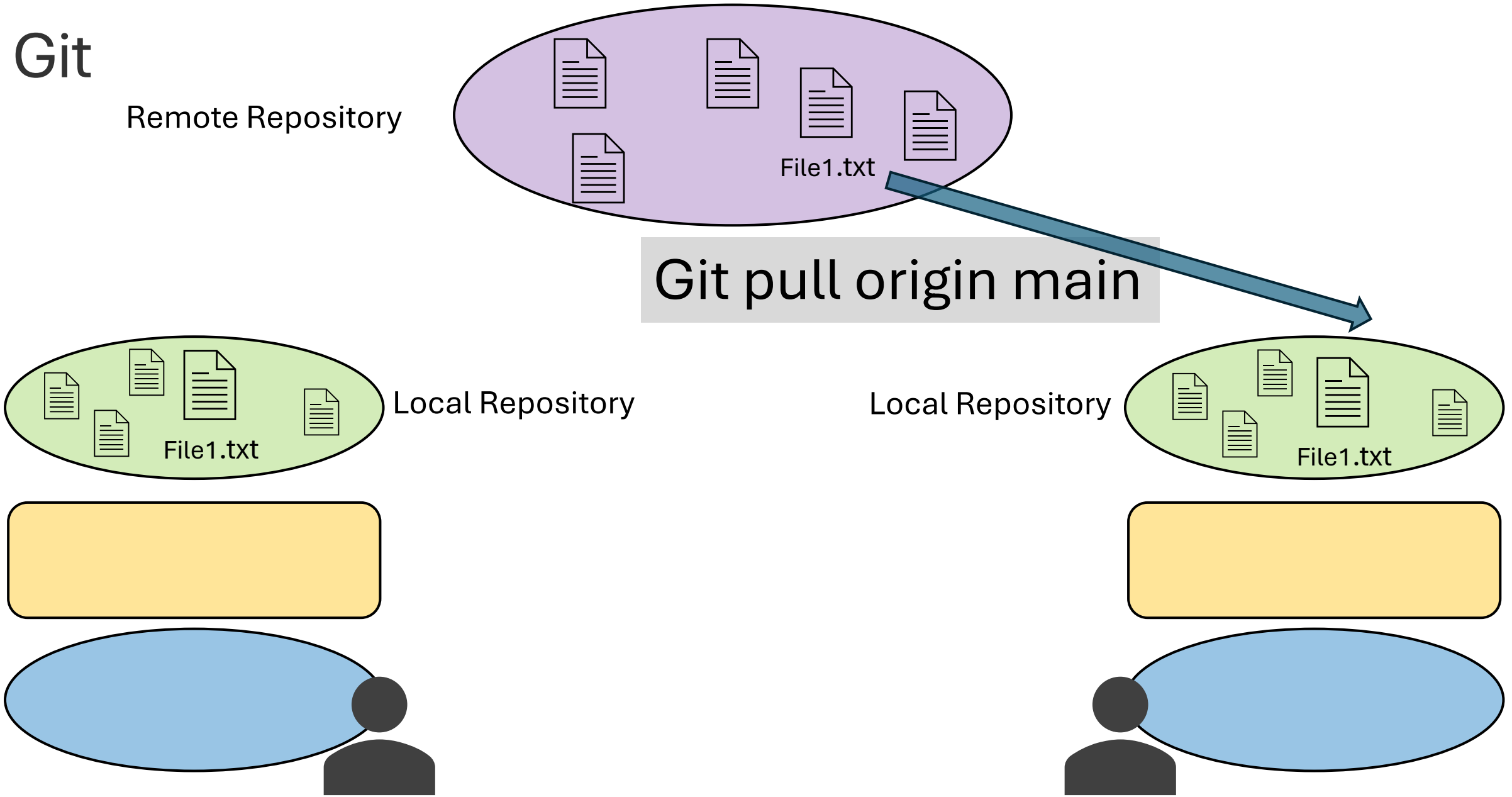


Local Repository



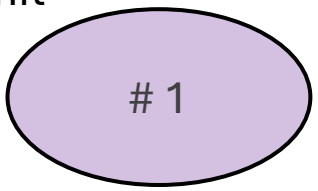


# Git

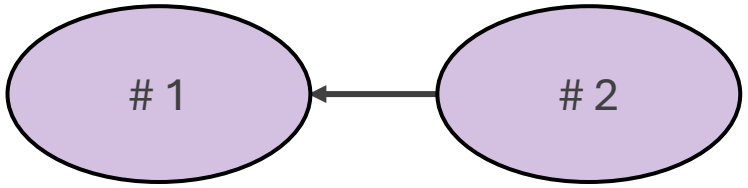

Remote Repository
File1.txt
Git pull origin main
Local Repository
File1.txt
Local Repository
File1.txt

# Branches

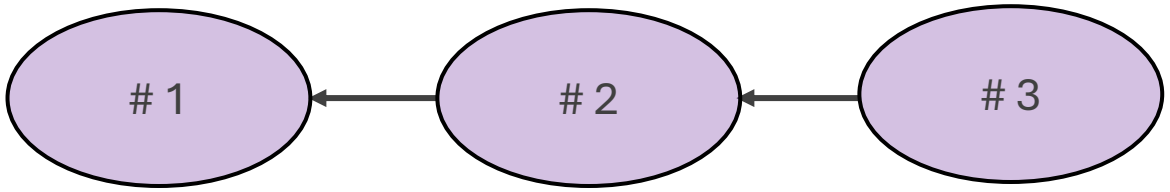
Commit



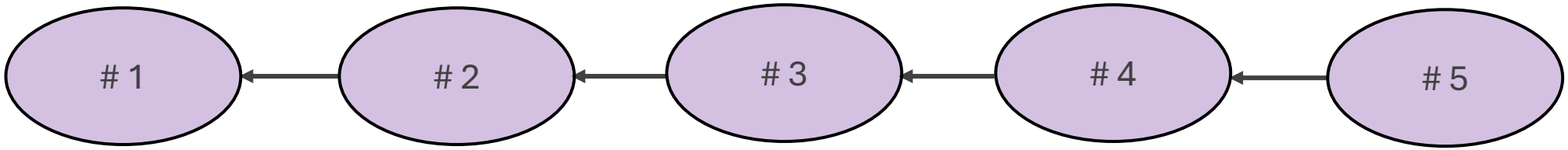
# Branches



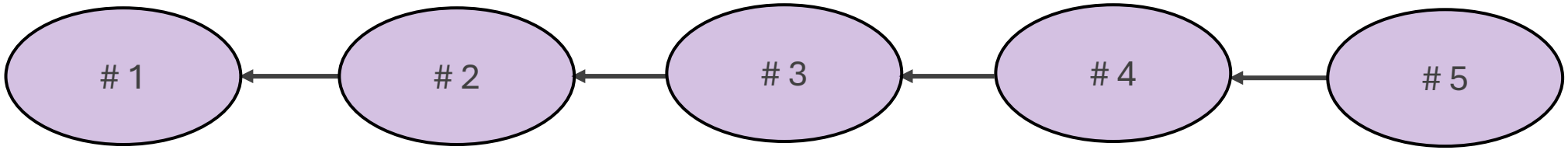
# Branches



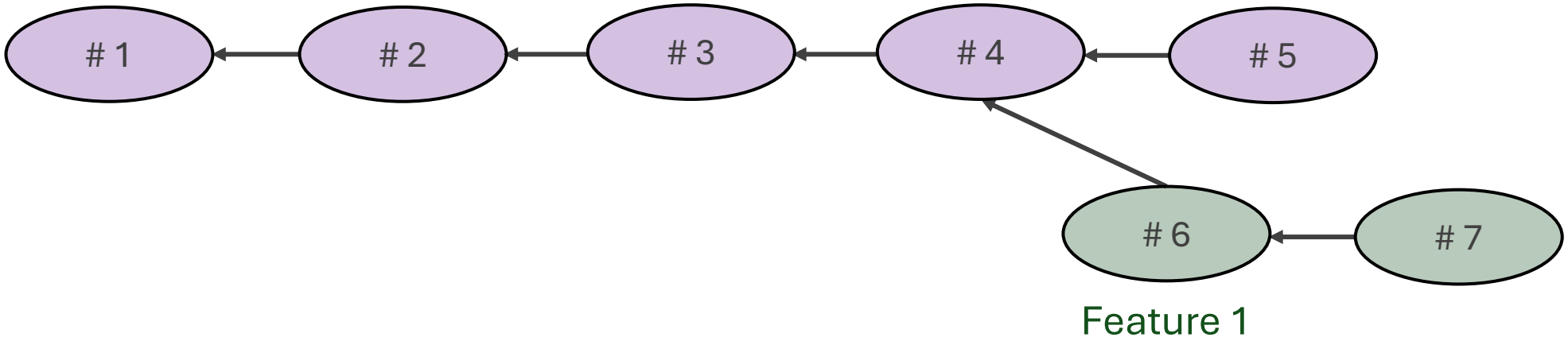
# Branches



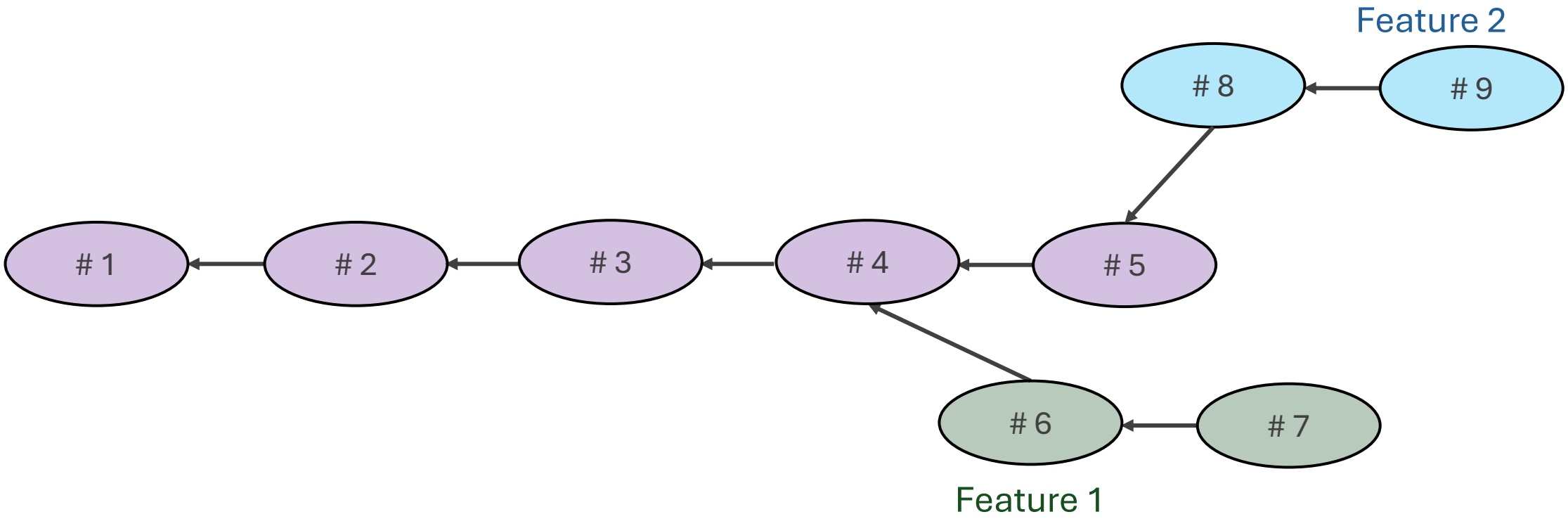
# Branches



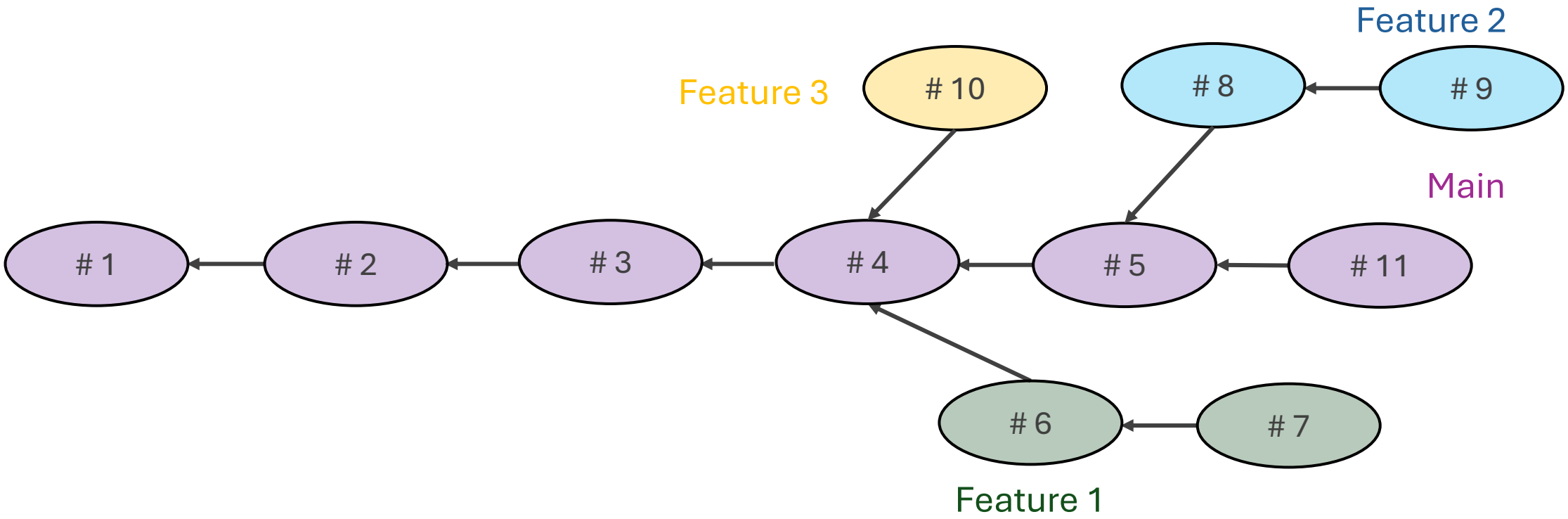
# Branches



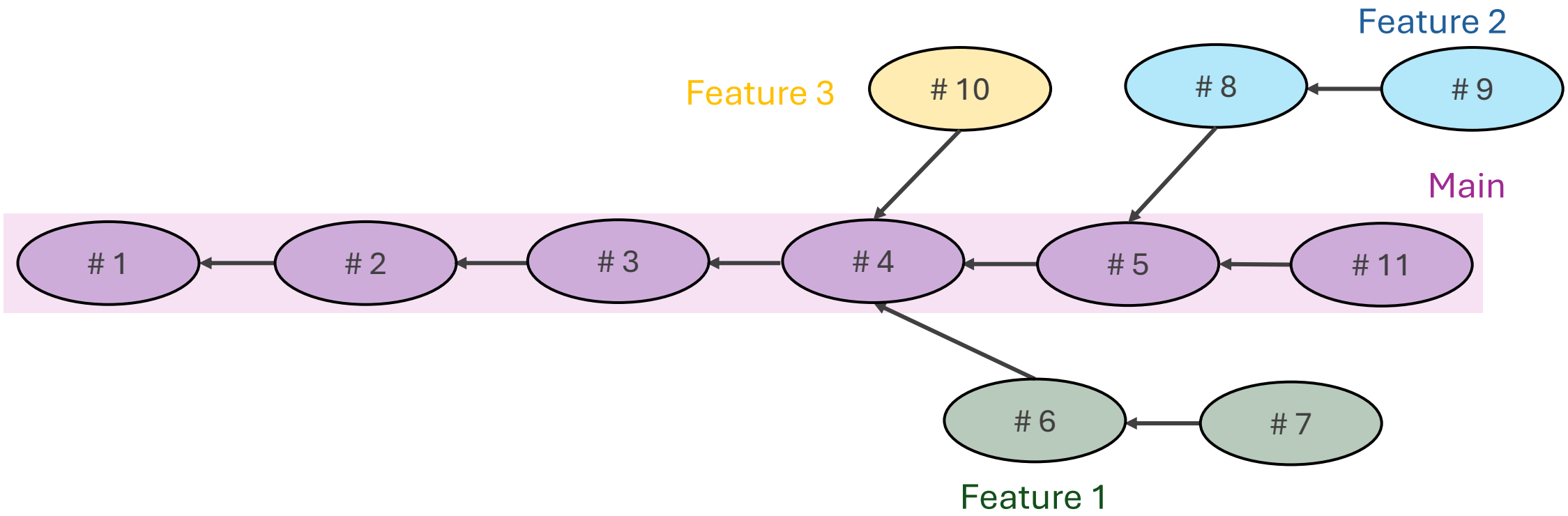
# Branches



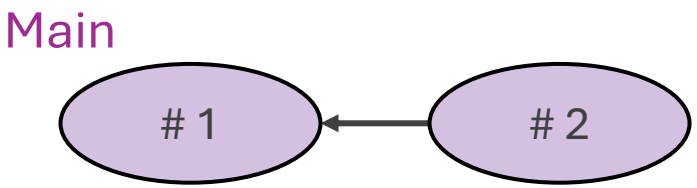
# Branches



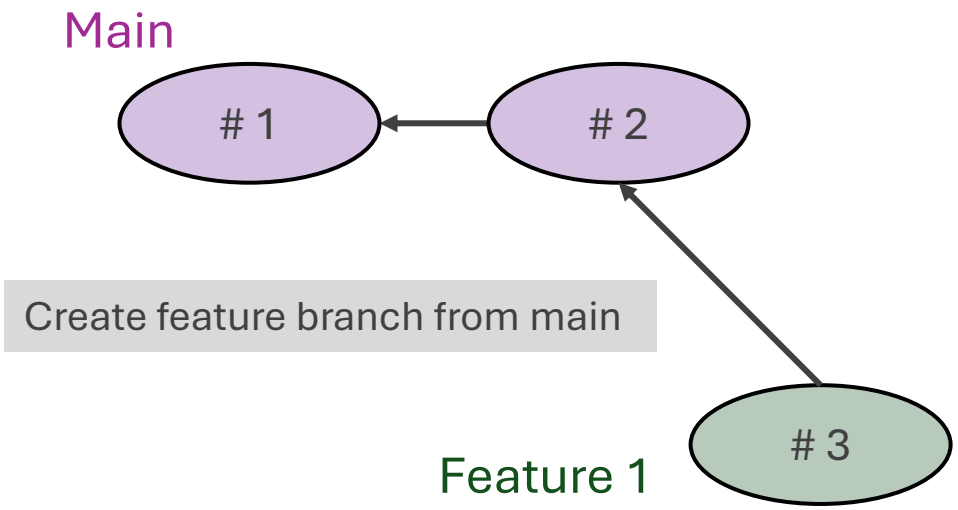
# Branches



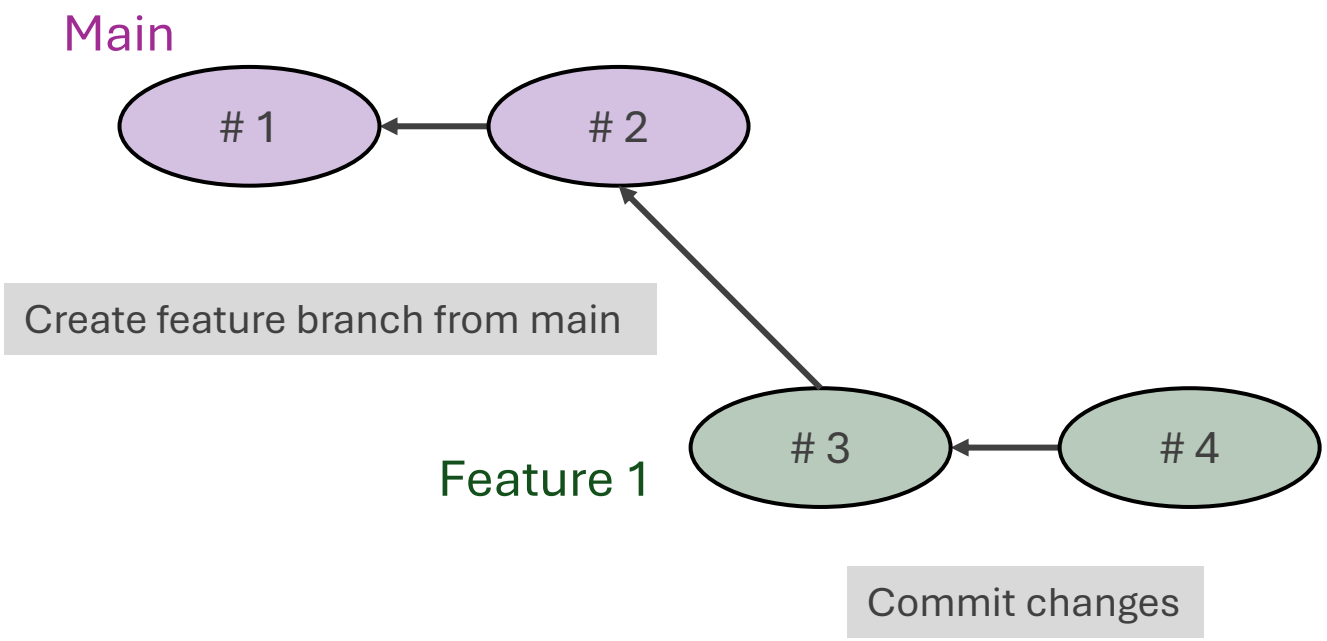
# Branches



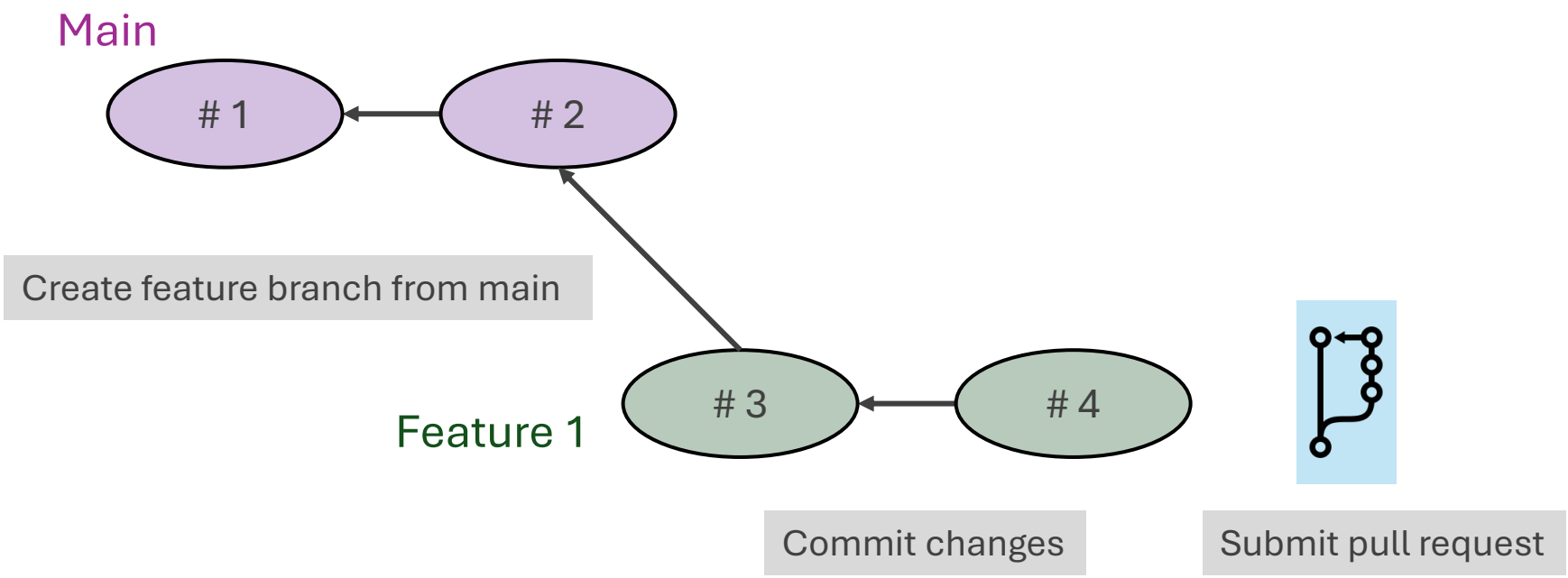
# Branches



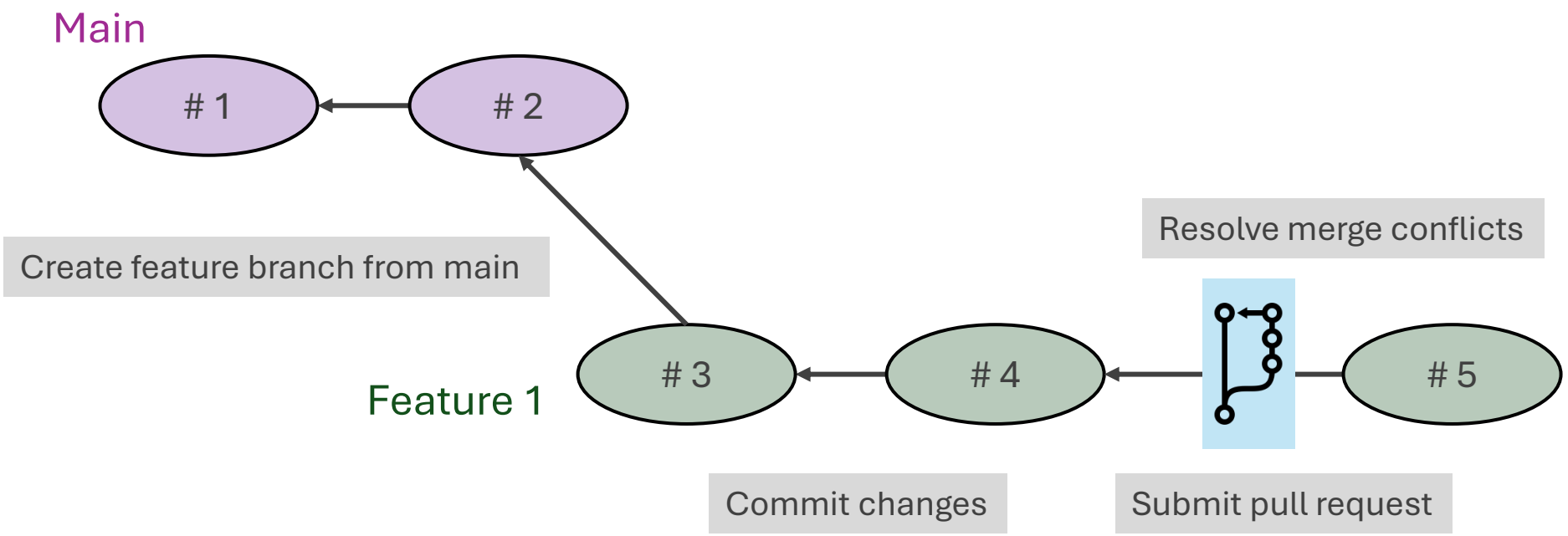
# Branches


Main
# 1
# 2
Create feature branch from main
Feature 1
# 3
# 4
Commit changes

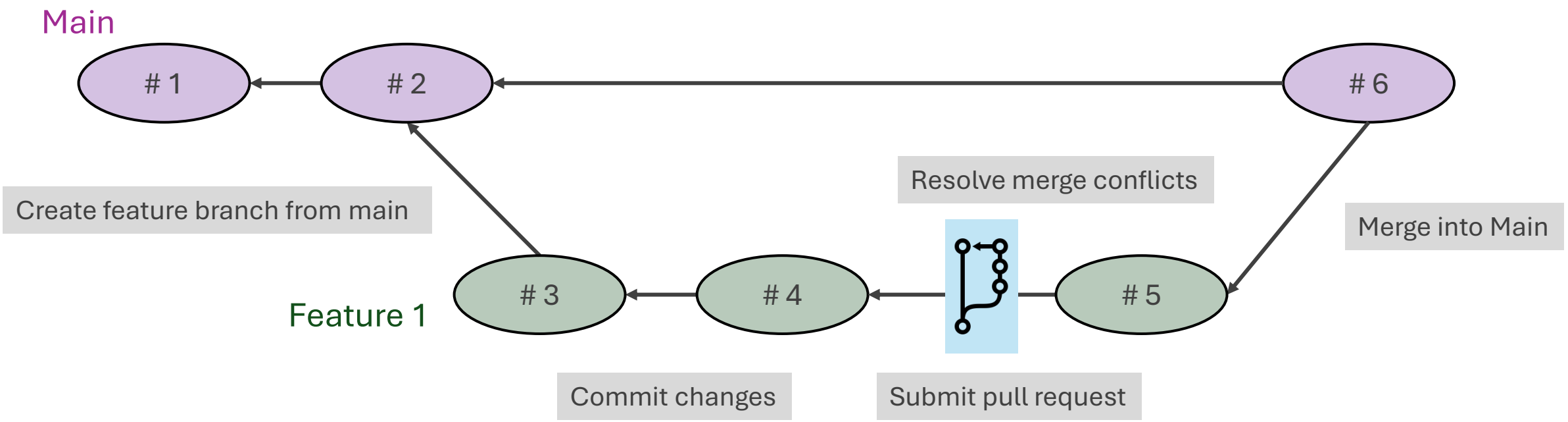
# Branches


Main
# 1
# 2
Create feature branch from main
Feature 1
# 3
# 4
Commit changes
Submit pull request

# Branches


Main
# 1
# 2
Create feature branch from main
Resolve merge conflicts
Feature 1
# 3
# 4
# 5
Commit changes
Submit pull request

# Branches


Main
# 1
# 2
# 6
Create feature branch from main
Resolve merge conflicts
Merge into Main
Feature 1
# 3
# 4
# 5
Commit changes
Submit pull request

# Git in Action

# Git in Microsoft Fabric

## Limitations

- Actions related to the workspace connection (connect, disconnect or switch branches) can only be performed by an Admin.
- **Not all items are supported in Git.**
- No direct way to go back to a previous commit in the Fabric UI. This has to be done in local repository with *Git revert* or *Git reset*, pushed to remote origin and then update the workspace with that new commit.
- If there were updates made to the Git branch, commits are disabled until workspace is updated. You can only sync in one direction at a time - committing and updating simultaneously is not possible.
- The commit size is limited to 125 MB.
- **Merge conflicts are only partially solvable in the Fabric UI.** You can either accept incoming changes from the remote origin or retain the current version for each item in the workspace.
- After approving a pull request (PR) and automatically deleting the branch post-merge, the workspace to which the feature branch was connected is not automatically cleaned up. This can lead to an accumulation of messy workspaces in Fabric over time.

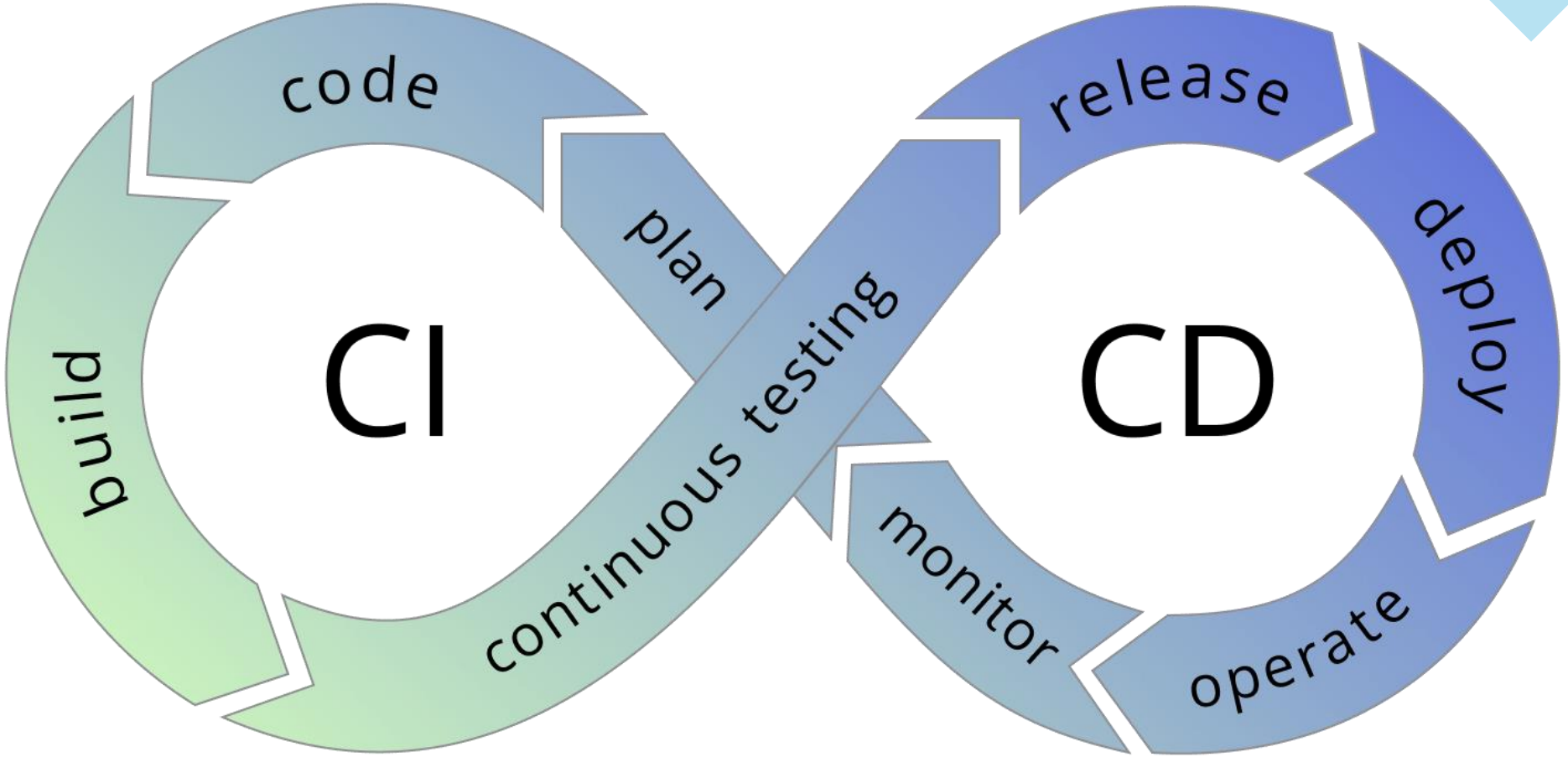
Overview of Fabric Git integration - Microsoft Fabric | Microsoft Learn

# Git in Microsoft Fabric

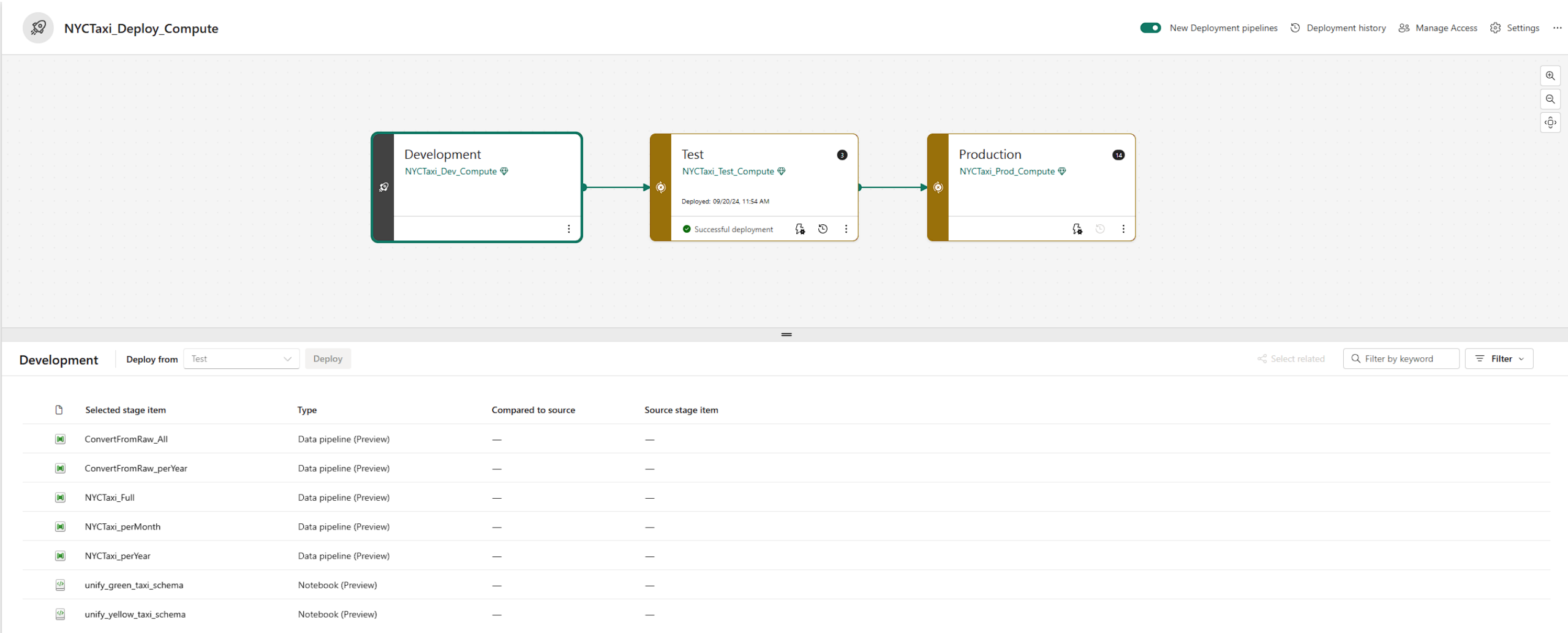
## Required Fabric permissions for popular actions

| Operation | Workspace role |
|---|---|
| Connect workspace to Git repo | Admin |
| Sync workspace with Git repo | Admin |
| Disconnect workspace from Git repo | Admin |
| Switch branch in the workspace (or any change in connection setting) | Admin |
| View Git connection details | Admin, Member, Contributor |
| See workspace 'Git status' | Admin, Member, Contributor |
| Update from Git | All of the following:<br><br>Contributor in the workspace (WRITE permission on all items)<br><br>Owner of the item (if the tenant switch blocks updates for nonowners)<br><br>BUILD on external dependencies (where applicable) |
| Commit workspace changes to Git | All of the following:<br><br>Contributor in the workspace (WRITE permission on all items)<br><br>Owner of the item (if the tenant switch blocks updates for nonowners)<br><br>BUILD on external dependencies (where applicable) |
| Create new Git branch from within Fabric | Admin |
| Branch out to a new workspace | Admin, Member, Contributor |

https://learn.microsoft.com/en-us/fabric/cicd/Git-integration/Git-integration-process


code
plan
build
continuous testing
CI
release
deploy
monitor
operate
CD

# Deployment Pipelines



# CI/CD in Microsoft Fabric

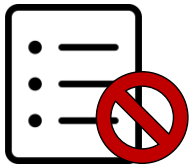
## Drawbacks

### No Git synchronization enforcement

**Workspaces are not automatically synced with Git before or after deployment**

### Triggered Manually

**Prone to human errors**



### Limited support for deployment rules

**Certain configurations are not parametrizable during deployment, forcing to manually adjust some items post-deployment**

# CI/CD in Microsoft Fabric

## Limitations

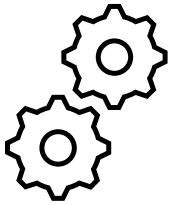
| Fabric Item | Git Integration | Deployment Pipelines |
|---|---|---|
| Lakehouses | x (partially) | x |
| Warehouses | x (bugged) | x (bugged) |
| Data Pipelines | x | x |
| Notebooks | x | x |
| Paginated Reports | x | x |
| Spark Job Definitions | x | |
| Spark Environments | x | x |
| Reports | x | x |
| Semantic Models | x | x |
| Dataflows Gen2 | | |
| Datamarts | | |
| Dashboards | | |
| Eventhouses | | Announced |
| Eventstreams | | |
| KQL Database | | |
| KQL Queryset | | |
| ML Model | | |
| ML Experiment | | |

# CI/CD in Microsoft Fabric

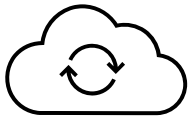
## Desired features

Version control for all workspaces (Dev, Test, Prod)

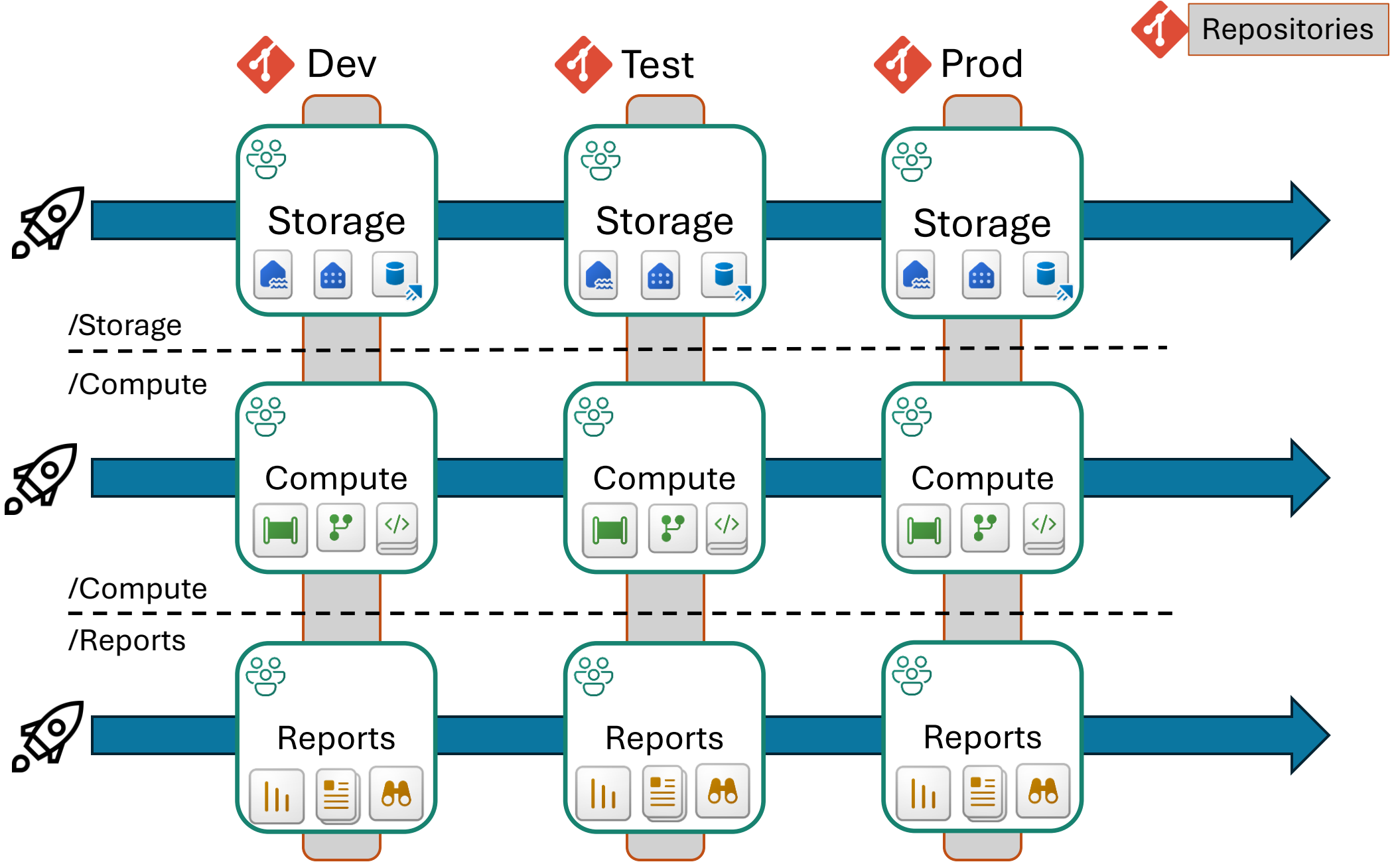
Adapting References (Dev Lakehouse -> Test Lakehouse)

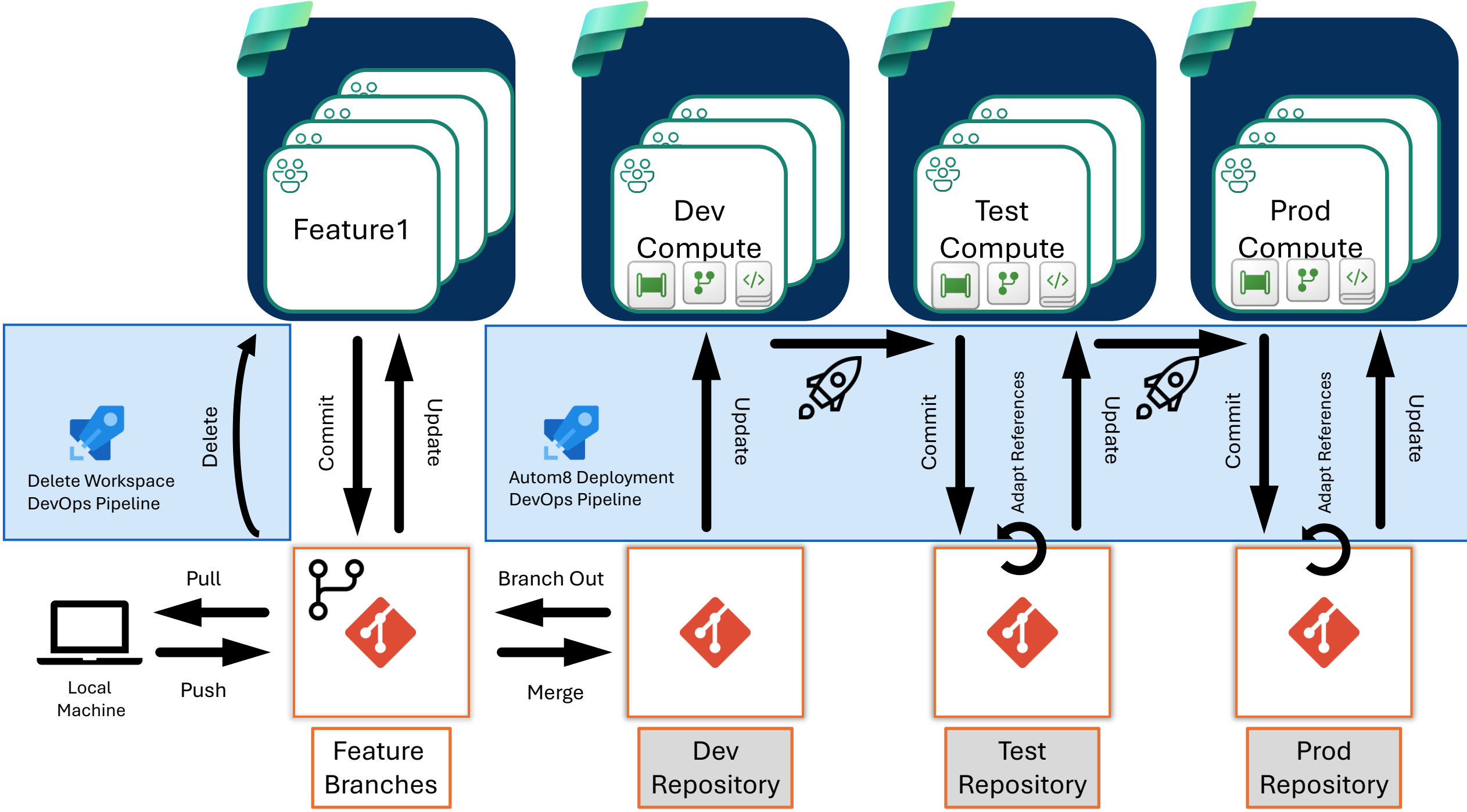


A pull request should trigger a deployment pipeline automatically



Workspaces should be synced with Git automatically


Repositories
Dev
Test
Prod
Storage
Storage
Storage
/Storage
/Compute
Compute
Compute
Compute
/Compute
/Reports
Reports
Reports
Reports
Deployment Pipelines



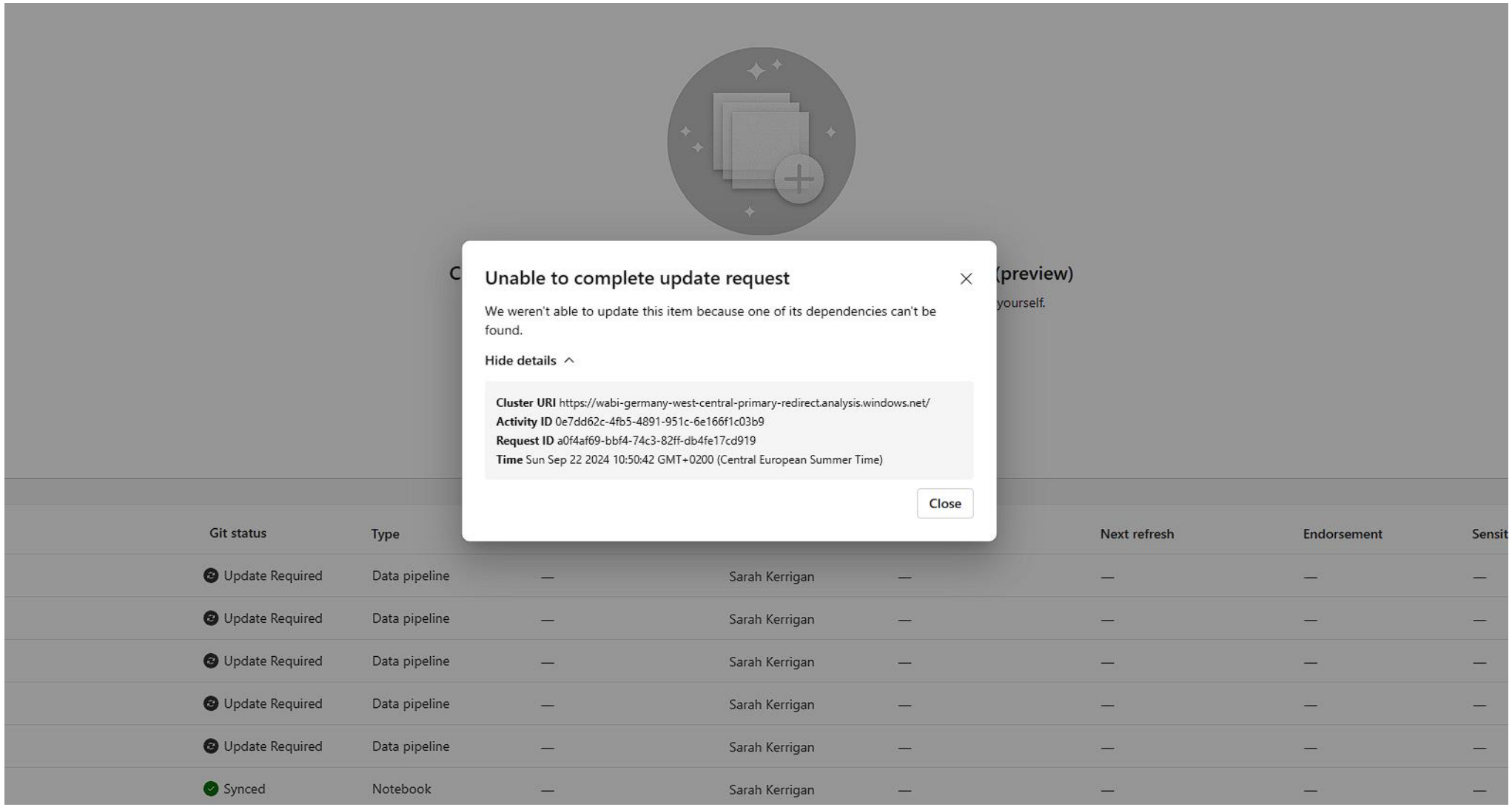

Compare to: https://data-marc.com/2024/07/09/fabric-ci-cd-with-Git-deployment-and-release-strategies/

# CI/CD in Action

# Fixing Git and CI/CD

| Fabric Item | Git Integration | Deployment Pipelines |
|---|---|---|
| Lakehouses | x (partially) | x |
| Warehouses | x | x |
| Data Pipelines | x | x |
| Notebooks | x | x |
| Paginated Reports | x | x |
| Spark Job Definitions | x | x |
| Spark Environments | x | x |
| Reports | x | x |
| Semantic Models | x | x |
| Dataflows Gen2 | x | x |
| Datamarts | x | x |
| Dashboards | x | x |
| Eventhouses | x | x |
| Eventstreams | x | x |
| KQL Database | x | x |
| KQL Queryset | x | x |
| ML Model | x | x |
| ML Experiment | x | x |

# Fixing CI/CD in Microsoft Fabric



# Fixing CI/CD in Microsoft Fabric

fabric_trigger_deployment_pipeline.yml

Edit

Contents History Compare Blame

```
69
70  - task: AzureKeyVault@2
71    inputs:
72      azureSubscription: 'Visual Studio Enterprise-Abonnement MPN(3bc27d58-dfd4-4148-be
73      KeyVaultName: 'kv-stone-boot-vis-001'
74      SecretsFilter: 'kerriganpw'
75      RunAsPreJob: false
76
77  - task: PowerShell@2
78    displayName: Login
79    inputs:
80      targetType: 'inline'
81      script: |
82        az login -u sarah.kerrigan@steinaudev.onmicrosoft.com -p $(kerriganpw)
83      pwsh: true
84
```

Non-interactive user login

One needs to **compromise the security** of **the entire Azure tenant** for this to work. Do not try at home.

Service Principals must work for all Endpoints

Thank you for your attention.
Any questions?

Feedback is kindly appreciated <3