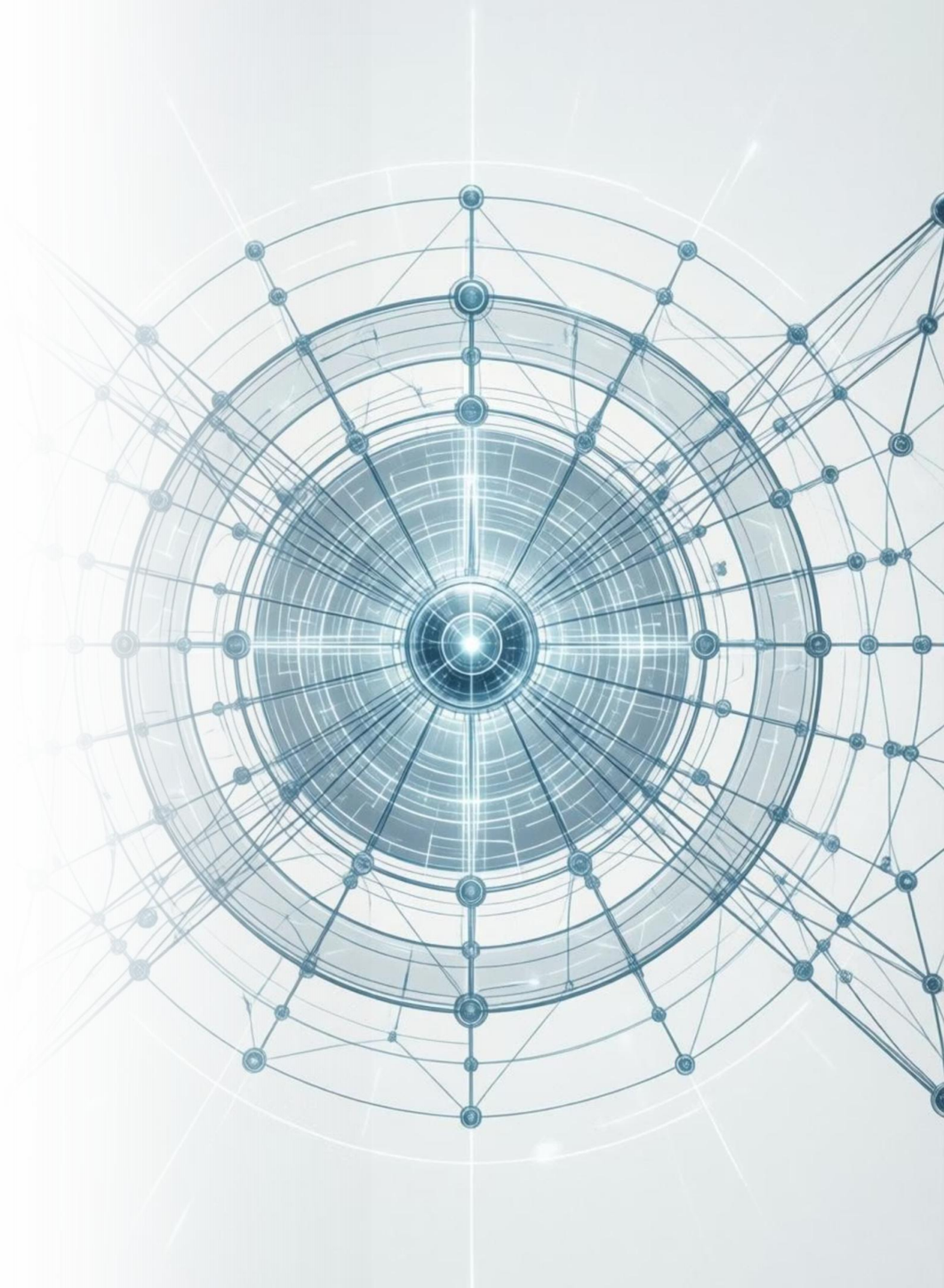
SECTION 02

Git Integration

Version Control for Fabric

> *"The safety net you didn't know you needed"*



"What Did I Change?"

1. **Friday, 4 PM**

   VP: "Sales dashboard is wrong. Western region numbers off. Fix before Monday's board meeting."

2. **Friday Evening**

   Alex makes changes. Deploys. Goes home.

3. **Monday, 9 AM**

   VP: "Now Eastern region is broken. What did you change?"

4. **The Problem**

   Alex: "Ummm... I'm not sure exactly. I made a few fixes Friday..."

Have you been Alex?

There's a better way.

Version Control Fundamentals

Think of Git like **Track Changes in Word**—but for code.

Every save = a commit

Snapshot of everything at that moment

You can go back

Undo any change to any commit

You can branch

Work without affecting main version

You can merge

Bring changes back together

Key insight: You never lose work. You always know what changed.

The Vocabulary You'll Need

Term	Meaning
Repository (Repo)	Where your files live (project folder + superpowers)
Commit	A snapshot: "Here's what everything looked like at this moment"
Branch	A parallel universe—work without affecting main version
Merge	Bring changes from one branch into another
Push	Send your commits to the remote repository
Pull	Get changes from the remote repository

Don't memorize—you'll learn by doing.

GitHub or Azure DevOps?

	GitHub	Azure DevOps
Free tier	Unlimited private repos	5 users free, then paid
Best for	Startups, most teams	Enterprises on Azure
We're using	✓ This workshop	-

Both work identically with Fabric.

Pick what your organization already uses.

Demo: The Friday Disaster Recovery

01

Connect workspace to Git

02

Make a change, commit it

03

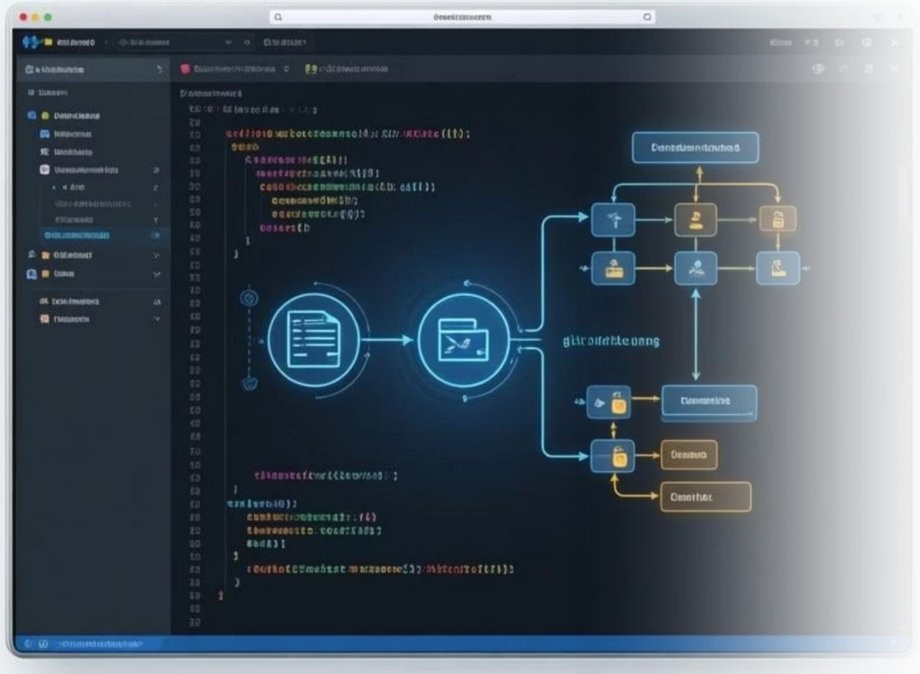
Break something intentionally

04

Recover using Git history

~30 minutes, live demo

Result: Alex's Friday disaster = Fixed in 30 seconds



DEMO

The Key Insight

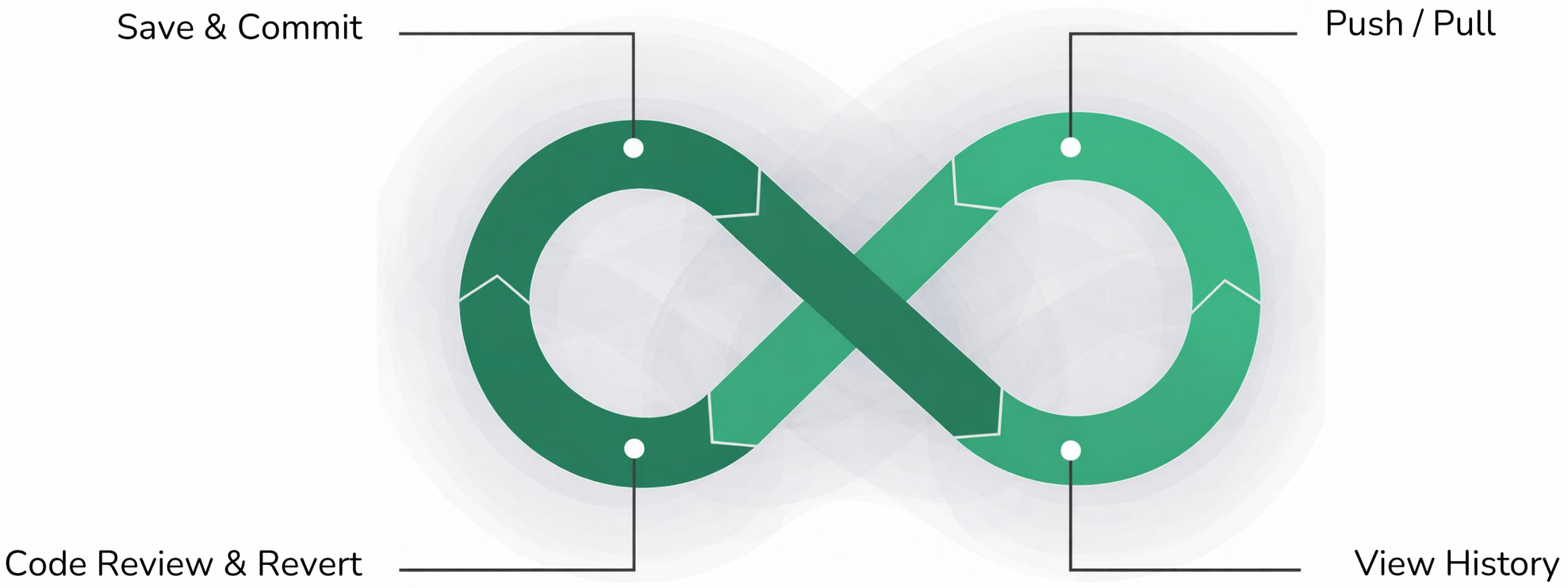
This isn't extra work. It's protection.

- Connected workspace to GitHub
  2 minutes
- Made changes, committed them
  Tracked history
- Broke something, rolled back
  30 seconds to fix

Every change is tracked. Nothing is lost.

> "Alex's Friday disaster? Fixed in 30 seconds. Show what changed. Roll it back."

Git Integration Flow



One workspace = One branch

Supported Fabric Items

✓ Syncs to Git

- Semantic models
- Reports (.pbir)
- Paginated reports
- Notebooks
- Spark job definitions
- Data pipelines
- Lakehouses
- Warehouses
- KQL databases

✗ Does NOT Sync

- Dashboards
- Dataflows Gen1
- Datamarts
- Real-time dashboards

Important: Lakehouse *definitions* sync. Lakehouse *data* doesn't.

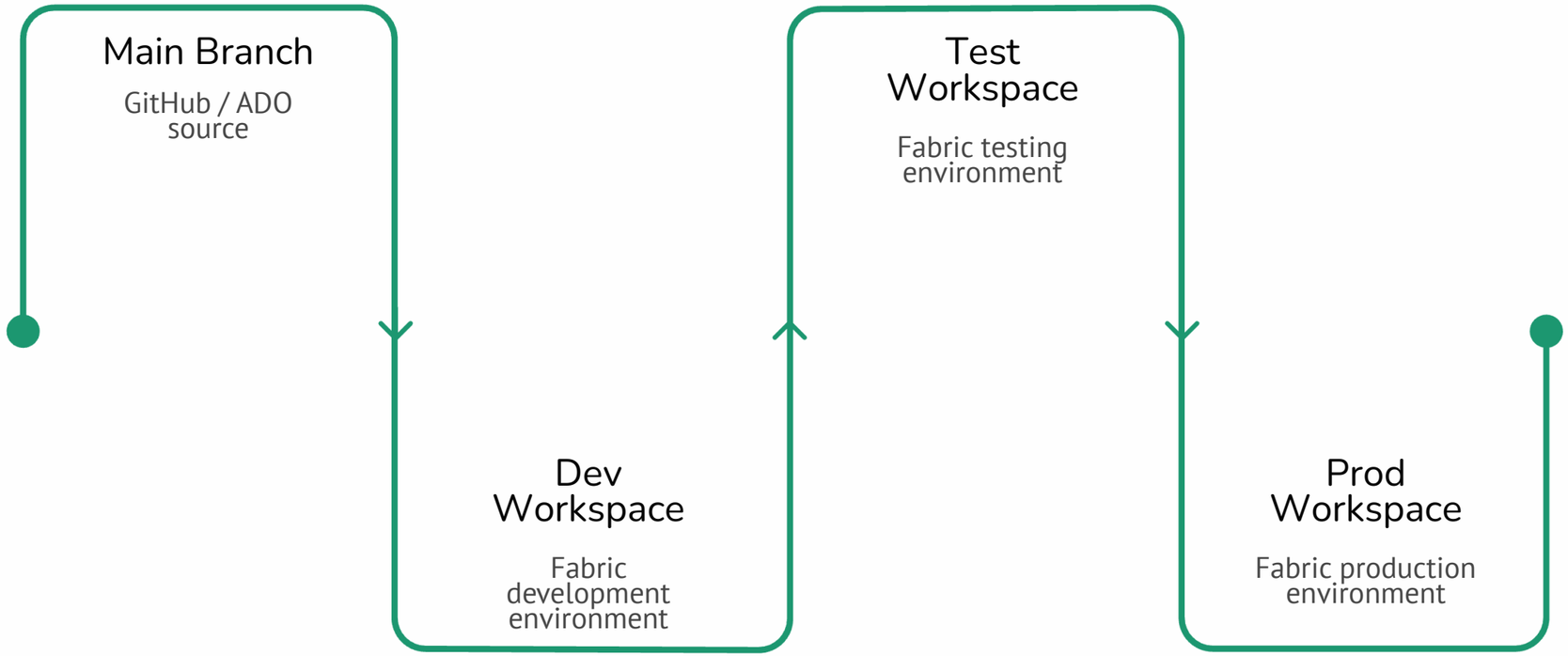
Git is for **code**, not **data**.

Two Tools, Different Jobs

Aspect	Git Integration	Deployment Pipelines
Purpose	Version control & history	Environment promotion
Workflow	Branch → PR → Merge	Dev → Test → Prod stages
Code review	✓ Yes (PRs)	✗ No
Cross-tenant	✓ Yes	✗ No (same tenant)
Rollback	Revert any commit	Redeploy previous stage
Setup	Requires GitHub/AzureDevOps	Built into Fabric

You can use BOTH.

The Complete Picture



Git = Source control

Pipelines = Promotion

Use Git for history & code review. Use Pipelines to promote between environments.

Watch Out For

Limitation	What It Means
One branch per workspace	Create separate workspaces for dev/test/prod
Merge conflicts	Fabric can't auto-resolve; last commit wins
All or nothing	Can't sync partial items; entire item syncs
Workspace = source of truth	First sync pushes TO Git, not from

Start simple. Add complexity as needed.

When Things Go Wrong

Saved but not committed

Undo in Fabric Source Control

Already committed

Revert in GitHub, then Update in Fabric

Need to experiment

Create a **branch** first

Commit often. Write good messages.

Your Action Items

01

Create a GitHub repo

Empty, private

02

Connect your dev workspace

03

Make one change, commit it

04

View the diff in GitHub

That's it. You're using Git.

Resources:

- MS Learn: "Git integration in Fabric"
- Slides available after session

End of Section 02