



Turning Data Into Insights



DevOps for Clinical Data Science

A Leap into the Future of Clinical Analysis



Frikkie Oppel
Associate Director, Statistical
Programming

The Vision

DevOps-driven Clinical Data Science environment:

- ✓ Effective version control
- ✓ Automated validation
- ✓ Continuous traceability

The target state is a governed, automation-assisted operating environment that supports reproducible execution, controlled collaboration, and continuous inspection readiness.

Let's begin!

The Challenge

Manual Status Tracking

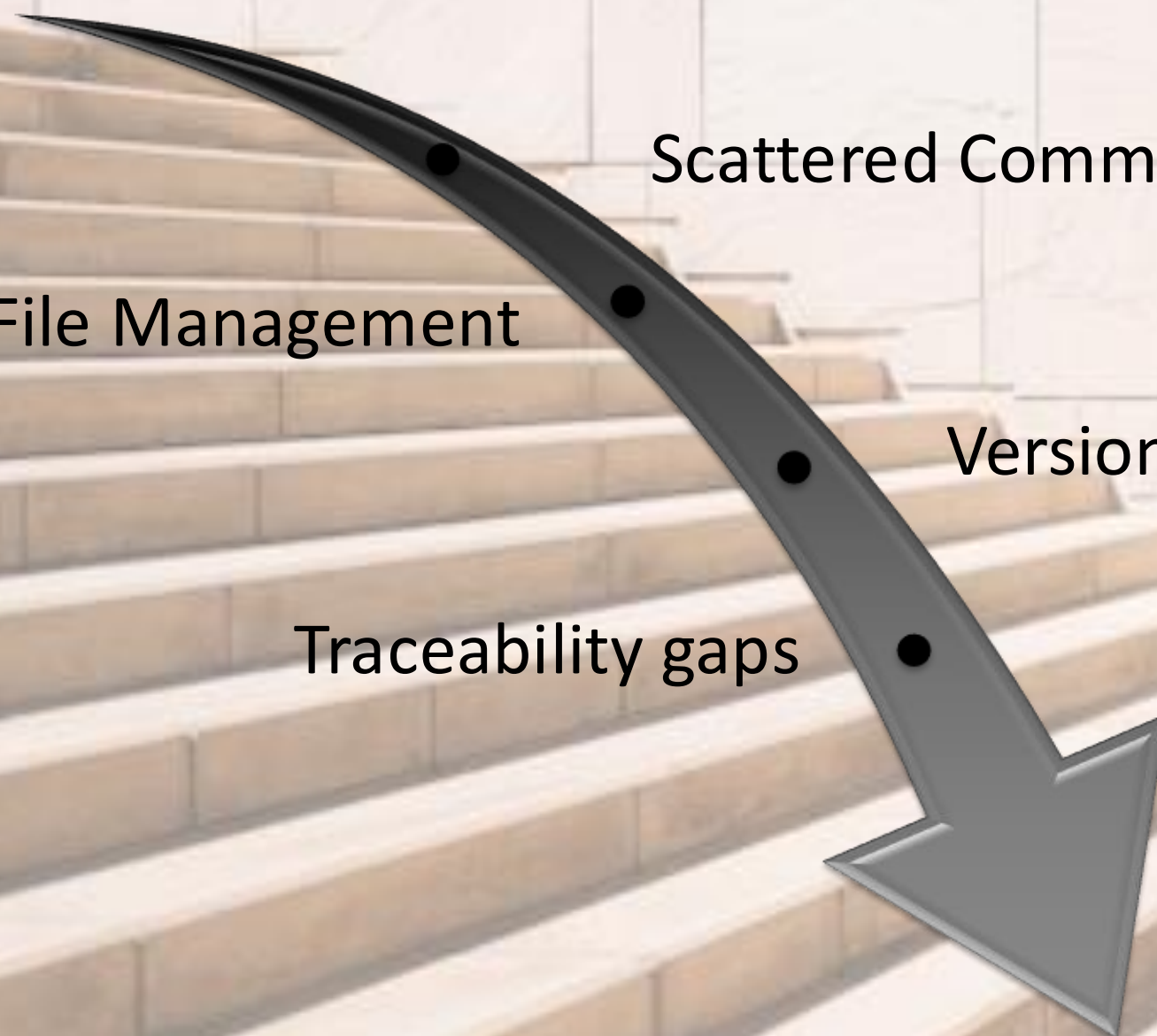
Scattered Communication

Manual File Management

Version Confusion

Traceability gaps

Inefficient Clinical Programming Practice



Evaluating a DevOps System

Objective: Achieve a security-and-quality-first transformation.

Operational Objectives:

1. Access Control and Security.
2. Computerized Version Control (VCS).
3. Repository and Folder Architecture.
4. DevOps Configuration.
5. CI/CD Pipelines for QC.
6. Performance Monitoring & VCS-based analytics.
7. Documentation, Training & Knowledge Base.
8. Business Outcomes Measurement.

Regulatory considerations:

- ✓ 21 CFR Part 11.
- ✓ ICH E6 Good Clinical Practice.
- ✓ ALCOA+ Data-integrity.

Access Control, Security, and the Computerized Version Control System

Project Outline:

Evaluate Systems: 3 Solutions identified and tested.

- ☐ Git
- ☐ Team Foundation Version Control
- ☐ Subversion (SVN)

Key Criteria: Collaboration Model; DevOps integration; Access Control Granularity; Handling of Large files/Binaries, as well as overall ease of adoption and ecosystem.

DevOps vs. Azure DevOps:

DevOps: A working approach emphasizing collaboration, automation, and reproducibility across analytical development.

Azure DevOps: A Microsoft platform with tools like Git repos, pipelines, and boards to support DevOps practices.

DevOps is the approach; Azure DevOps is the toolset that enables it.

Access Control and Security

Choosing a VCS: Git vs TFVC vs SVN

Criterion	Git*	TFVC	SVN
Collaboration model	Distributed; lightweight branching; strong PR workflows	Centralized; good for large monoliths	Centralized; simpler mental model
DevOps integration	Deep CI/CD, Boards, Policies	Native to Azure DevOps (legacy)	Limited; external CI or custom
Access control granularity	Repo/branch-level (use separate repos for blinding)	Fine at project level; not file-path in Git sense	Fine path-based permissions in one repo
Large files/binaries	Needs LFS/strategy; code-focused	Handles large repos well	Handles binaries better than Git
Adoption & ecosystem	Industry standard; rich tooling	Declining relevance vs Git	Stable but shrinking ecosystem

* Integrated in Azure DevOps.

Post-jo

Statistician (Marie)

Related Work

Link an Azure Repos [commit](#), [pull request](#) or [branch](#) to see the status of your development. You can also create a branch to get started.

31 ADSL
Updated 14m ago Done

Operationalizing Azure DevOps + Git: Maintaining the Blind

- ✓ **Separate Repositories**
- ✓ **Controlled Unblinding Handoff**
- ✓ **Post-Unblinding Changes Tracked**
- ✓ **Ongoing Blinded Work – Blinded Repo Stays Blind**



Compliance and Quality Controls

Ensured by System Design:

- ✓ **Branch Protections**
- ✓ **Git provides Immutable Audit Trails and Records**
- ✓ **Built-in Quality Culture**

What's Next?

- Expand automation - Infrastructure-as-code
- Data Driven Improvements through performance metrics
- Greater Scalability
- And much more

DevOps isn't about moving faster at the expense of quality

It's about building quality into the way we work. With Git and Azure DevOps, our clinical programming practice is more transparent, more collaborative, and inherently audit-ready. The destination is valuable, but the journey — the SOPs, the training, the culture — is what makes it sustainable.





Q&A

What would be your first pilot?

Thank you.