

A Complete Version Control Workflow for Statistical Programming

jeetender.chauhan@merck.com, madhusudhan.ginnaram.reddy@merck.com,

sarad.nepal@merck.com, benjamin.wang@merck.com

Merck & Co., Inc., Rahway, NJ, USA

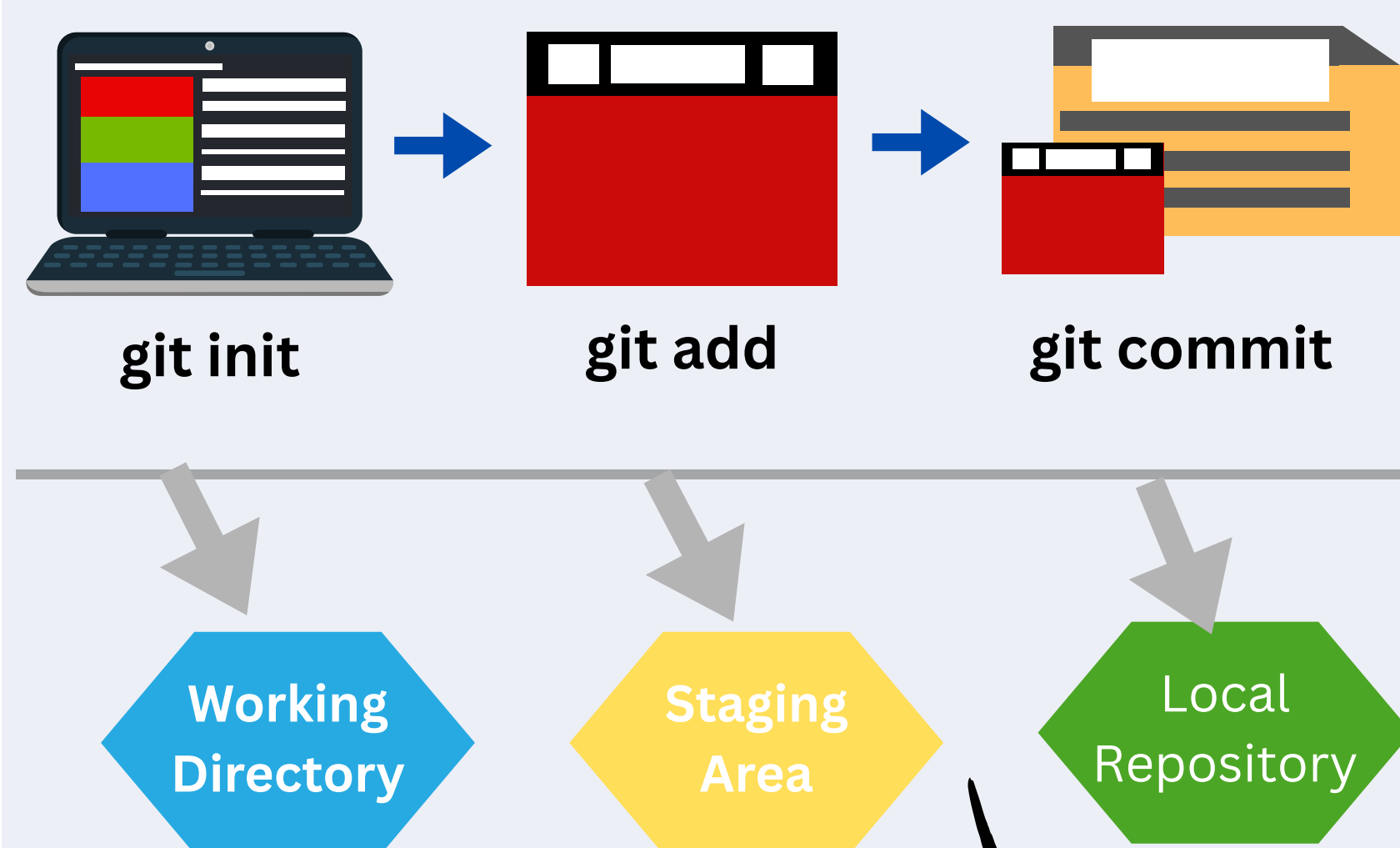
Why Version Control Matters

- Regulatory Requirements (FDA)
- Error-Prone Manual Versioning (final_v2 REALLY.R)
- Risk of Overwriting Code
- Need for Reproducibility

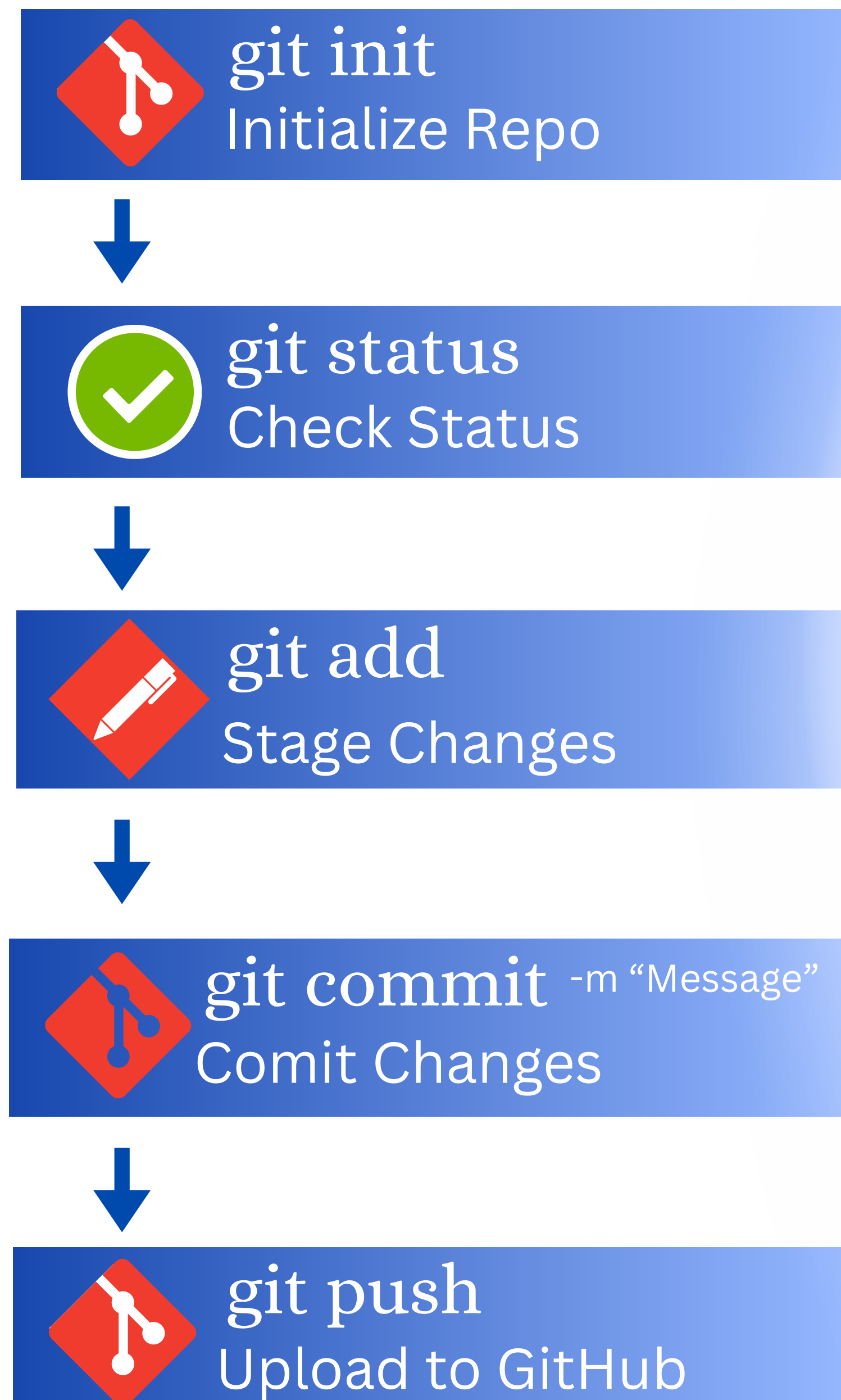
The Solution

- **Git** = Local Version Control
- **GitHub** = Cloud Collaboration

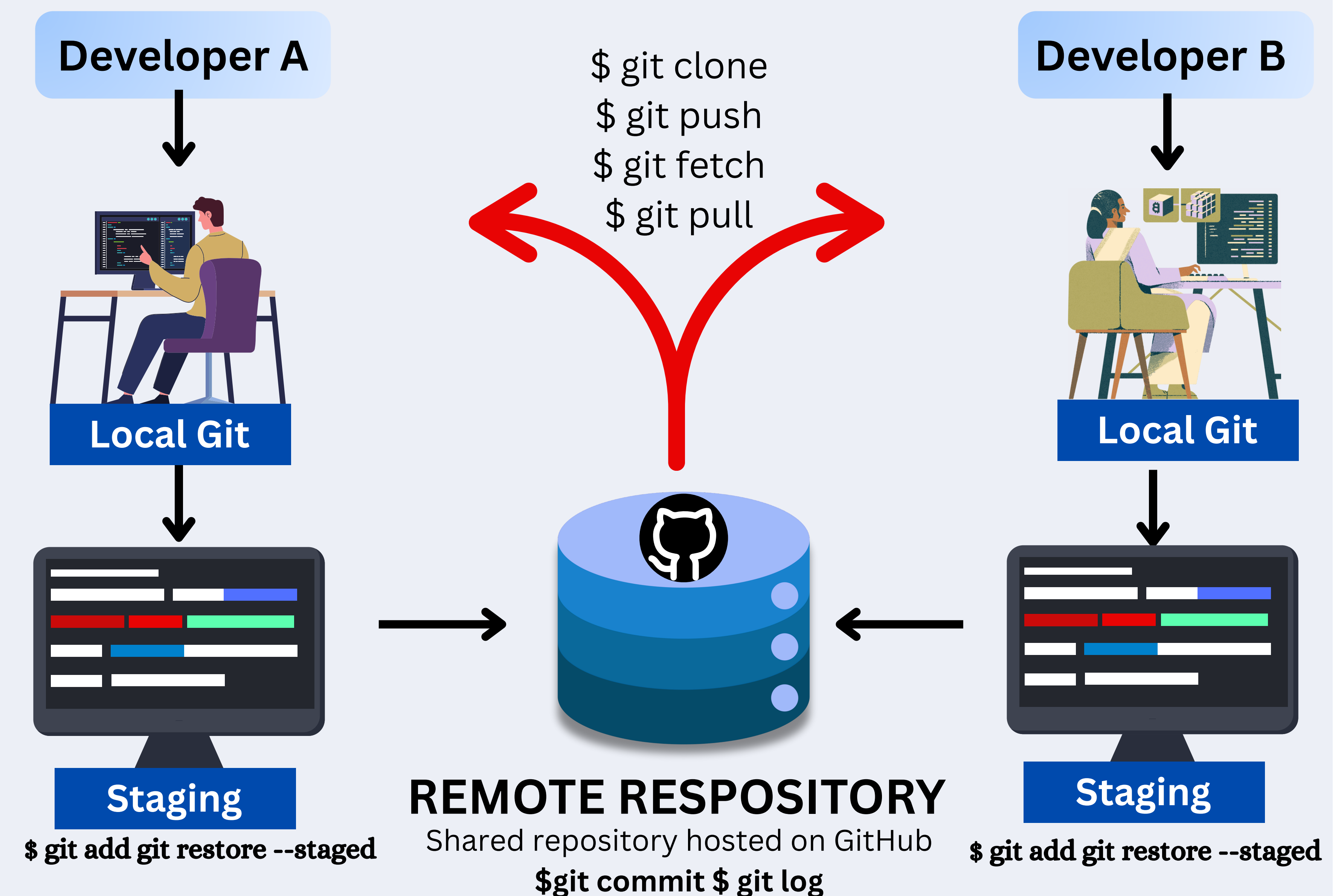
The 3 states of a File in Git



Basic Git Workflow



Conceptual Collaboration Model: Local Git + Remote Repository



Working Directory

- Create . Edit . Delete files
- Untracked: New files not yet added
- Tracked: Modified files after commit

Staging Area

- Select Changes for commit
- Only staged changes will be committed

Local Repository

- All files here are tracked and versioned
- Permanent History

Key Takeaways

- ✓ Full Audit Trail
- ✓ Distribution Collaboration
- ✓ FDA Complete

Conclusion

- Create GitHub Account
- Install Git
- Initialize a repository
- Push to Github
- Each developer works independently with a full local Git history, enabling offline development and experimentation.
- The remote repository acts as a controlled exchange point, not a shared working space.
- Changes are shared only through committed snapshots, preventing accidental overwrites.
- Git automatically detects conflicts during pull or merge, enforcing explicit resolution.
- This distributed model supports scalable, auditable team collaboration.